САДРЖАЈ

1.УВОД

Android је прва open source бесплатна и потпуно прилагодљива платформа за мобилне уређаје. Дизајниран је за уређаје са touchscreen-ом као што су паметни телефони и таблет рачунари, а све више се користити и код новије генерације netbook рачунара који имају touchscreen, паметних сатова, Android TV-а, мултимедије у аутомобилима. Androidu се у почетку није прогнозирао велики успех због чињенице да је систем заснован на отвореном коду, али тврдње су се испоставиле погрешним будући да је Android тренутно најраспрострањенија платформа код паметних телефона.

Мобилна комуникација је употреба технологије која омогућава комуникацију без употребе било какве физичке везе. Мобилни телефон је типичан пример мобилне комуникације. Мобилна комуникација омогућава пренос гласовних и мултимедијских података путем рачунара или мобилног уређаја без повезивања с било којом физичком или фиксном везом. Мобилна комуникација постала је неопходна за све и развија се из дана у дан.

У пословном окружењу стање мобилних комуникација има два главна значаја, прво је везано за мобилне операторе, а друго за пословно окружење. У протеклом периоду мобилни оператори су били посвећени изградњом инфраструктуре и повећањем броја претплатника. Повећање мобилних претплатника долази до засићења, а технологија, услуга и мрежа омогућавају промену нових решења.

За потребе овог мастер рада, формирана је база података у софтверу XAMPP. XAMPP је потпуно бесплатан open source серверски пакет за једноставну инталацију Apache сервера на рачунаре са оперативним системом Windows, Linux или OSX (зато је прво слово у називу X). Xampp је намењем за употребу у локалној мрежи не као web сервер и служи пре свега програмерима да креирају сервер ради тестирања својих скрипти, односно MySQL база података у off-line режиму.

За проверу система за синхронизацију мобилних апликација на SQL серверу, направљен је софтвер коришћењем *Android Studio 3*.

Android Studio представља интегрсано развојно окружење за развој Андроид апликација. Састоји се од низа алата који служе за развој мобилног програмског производа, укључујући алате за дизајн корисничког окружења, писање програмског кода, тестирање и отклањање грешака, комуникацију са мобилним уређајем, припрему програмског производа, припрему апликације за објављивљње и слично.

Развој апликација за Андроид је вођен захтевом за одвајањем корисничког окружења од програмске логике. Тај захтев доводи до тога да се у фази развоја програмског производа користе различити језици за опис споменутих апликативних слојева: Јава за опис програмске логике и XML за опис и дизајн окружења.

Идеја за формирање интерфејса је да се направи апликација која ће да се инсталира на неки од мобилних уређаја (телефон, таблет), а основна му је сврха да се конектује на жељени сервер базе података. Циљ је да сам интерфејс буде веома једноставан и прилагођен за рад на мобилним уређајима.

2. БЕЖИЧНА КОМУНИКАЦИЈА

Бежична комуникација подразумева, као што само име каже, комуникацију без физичког медијума ("жице") која би повезала два уређаја за комуникацију. Веза се успоставља путем радио фреквенција (RF). Да би се успоставила комуникација између два корисника, телекомуникацијски систем мора да спроведе низ корака и да проже кроз низ жичаних. Ипак, овај процес, па чак и данас, није у потпуности безжичан. Будућност бежичне технологије је проширење бежичног повезивања с мобилним системима за покривање удаљених и слабо насељених подручја. Један од циљева безжичане технологије је и пружање хитних услуга у случају квара копнених бежичних услуга због природних догађаја, као што су поплаве, земљотреси и слично. Овакав систем би у значајно мери смањио зависност од жичаних компоненти, али је мала вероватноћа да се "жица" може у потпуности елиминисати. Са становишта корисника је битна покривеност бежичне комуникације, како би могли да користе предности потпуне мобилности у разговору са другим лицима без обзира и на могуће повећање трошкова претплате[1].

2.1. МОБИЛНА КОМУНИКАЦИЈА

Мобилна комуникација је употреба технологије која омогућава комуникацију без употребе било какве физичке везе. Мобилни телефон је типичан пример мобилне комуникације. Он представља уређај који се користи за двосмерну радио телекомуникацију преко ћелијске мреже базних станица - ћелијска локација.

Мобилна обрада података (mobile computing) је посебан случај дистрибуираних систем , која укључује обраду и пренос података преко дистрибуираних и мобилних чворова. Ова технологија омогућава, као и у телекомуникацционим технологијама, пренос гласа, али и пренос и обраду података од тачке до тачке или од особе до особе[1].

Мобилна комуникација омогућава пренос гласовних и мултимедијских података путем рачунара или мобилног уређаја без повезивања с било којом физичком или фиксном везом. Мобилна комуникација постала је неопходна за све и развија се из дана у дан.

У XXI веку, услуге мобилне комуникације имају све већу примену било у сврху основних животних потреба (породица, посао) или само из навике. Мобилни телефон постао је не само статусни симбол већ и неопходно средство за рад.

Мобилне комуникације су, тренутно, најбрже растућа технологија у свету. Последица овог тренда је та да је забележен огроман пад потошње и фиксним мрежама, а у исто време се бележи, готово, свакоднвни раст потрошње у мобилним мрежама, као и огроман раст интернет промета

У пословном окружењу стање мобилних комуникација има два главна значаја, прво је везано за мобилне операторе, а друго за пословно окружење. У протеклом периоду мобилни оператори су били посвећени изградњом инфраструктуре и повећањем броја претплатника. Повећање мобилних претплатника долази до засићења, а технологија, услуга и мрежа омогућавају промену нових решења. Мобилни оператери почињу све више да се фокусирају на пословне компаније и на примену мобилних услуга у пословном окружењу.

Предности мобилне комуникације:

- Флексибилност - Бежична комуникација омогућава људима да комуницирају једни с другима без обзира на локацију.
- Ефикасност трошкова – Трошак је смањен, у бежичној комуникацији нема потребе за физичком инфраструктуром.
- Брзина - Мрежна повезаност или приступачност знатно су побољшани у тачности и брзини.
- Приступачност - Уз помоћ бежичне технологије лак је приступ удаљеним подручјима.
- Стална повезаност - Стална повезаност осигурава да људи могу реаговати на ванредне ситуације релативно брзо.

Пословне компаније се све више оријентишу на примену мобилних технологија у циљу побољшања и унапређења пословних процеса уз знатно краће време повратка уложених улагања. Пред компанијама је могућност увођења новог квалитеа у комуникацијама применом инфраструктуре која интегрише говор и податке (разне врсте датотека са подацима. Наравно, оријентација на увођење и примену мобилне технологије, прати и захтеве за што флексибилнијим радним процесима. Из ових разлога, данас се, на тржишту мобилних уређаја могу наћи искључиво "паметни" уређаји, који интегришу и говорну комуникацију, интернет комуникацију, као и апликације за обраду података. Мобилне комуникације засноване су на различитим безбедносним стандардима и протоколу преноса који стоји иза њега[1].

2.2. ГЛАВНЕ КАРАКТЕРИСТИКЕ И ОБЕЛЕЖЈА WIRELESS APPLICATION PROTOKOLA (WAP)

Wireless Application Protocol (WAP) је технички стандард за приступ информацијама преко мобилне бежичне мреже који се користи код нових бежичних уређаја за комуникацију са сервером инсталираним у мобилној мрежи. Овај протокол омогућава кориснику да не само да види жељене податке већ и да оствари интеракцију са њима кроз корисничке програме.. Бежичне комуникације се преносе путем таласа различите фреквенције и представља технички стандард за приступ информацијама преко мобилне бежичне мреже[2].

Ова технологија повезује интернет и мобилне комуникације. WAP не представља само комуникацијски језик већ је и платформа за развој и међусобно повезивање. Омогућава приступ и интеракцију с информацијама и услугама на Интернету коришћењем мобилног апарата или других бежичних уређаја који подржавају WAP протокол. Користи се као средство за преглед интернет странице користећи ограничен пренос капацитета и мале екране преносних бежичних уређаја.

WAP је отворени индустријски стандард, развијен од стране WAP Forum-a, првенствено намењен за интегрисање мобилне телефоније и Интернет-а, а у циљу, да корисницима мобилних телефона и других бежичних терминала, какви су пејџери и PAD-ovi (Personal Dugital Assustants), обезбеди приступ телефонским и информационим сервисима, укључујући Интернет и Web. WAP је пројектован да ради са свим бежичним мрежним техикама какве су GSM, CDMA и TDMA, а такође је базиран и на Интернет стандардима какви су IP, XML, HTML, и HTTP. Поред тога WAP поседује и могућност сигурног (криптованог) преноса информације. WAP Forum је основан 1997 године од стране компанија Ericsson, Motorola, Nokиа, и PhoneCom. Данас форум чине више стотина компанија. Јуна 2000 године WAP Forum је прогласио важећи стандард WAP v1.2

WAP стандард у суштини дефинише скуп стандарда који на један обједињени начин дефинишу како бежични ручни терминали комуницирају преко бежичне мреже, и на који се начин доставља садржај информације и пружају сервиси који се извршавају од стране мобилних терминала (handsets). Уз помоћ WAP-а, стандардни мобилни ручни терминали могу да успоставе везу са WAP-овом инфраструктуром која је намењена за бежични пренос података, и да од те инфраструктуре захтевају разне садржаје информације и сервиса, а након тога да прибављени садржај и сервисе представе кориснику. WAP стандард нема ефекат само на имплементацију мобилних ручних терминала него и на инфраструктуру сервиса који егзистира у структури мрежног оператера (network operator), провајдера сервиса (servuce provider), и провајдера садржаја (content provider).

Мобилни провајдери услуга пре увођења WAP-а пружали су могућност ограниченог преноса података, али је је била неопходна и подршка за Интернет и web апликације (што данас представља стандард), као што су:

- E-maiл путем мобилног телефона;
- Праћење цена деоница;
- Спортски резултати;
- Вести и преузимање музике[2].

Ову технологију, која се непрекидно развија, састоји се од следећих компоненти:

- Језика вишеструке намене као што је HDML (Hand Device Markup Language) и WML (wireless Markup Language);
- Физичких јединица или бежичних уређаја познатих као WAP уређаји који су направљени да би се спојили са net-ом;
- Gateways – везни чвор преко којег се обавља пренос података из бежичне мреже у Интернет мрежу и обрнуто;

Алати за програмско инжењерство намењени за тестирање и развој WAP апликација.

3.АНДРОИД ОПЕРАТИВНИ СИСТЕМ

Компанија Andorid је основана 2003. године у Palo Altu у Калифорнији. Њен основни циљ је било тржиште оперативних система за дигиталне камере, али се преусмеравала на тржиште оперативних система за "паметне" мобилне телефоне. У августу 2005. године Google је откупио компанију Android.

Android је прва open source бесплатна и потпуно прилагодљива платформа за мобилне уређаје. Дизајниран је за уређаје са touchscreen-ом као што су паметни телефони и таблет рачунари, а све више се користити и код новије генерације netbook рачунара који имају touchscreen, паметних сатова, Android TV-а, мултимедије у аутомобилима. Androidu се у почетку није прогнозирао велики успех због чињенице да је систем заснован на отвореном коду, али тврдње су се испоставиле погрешним будући да је Android тренутно најраспрострањенија платформа код паметних телефона.

Овај оперативни систем је доступан на уређајима направљеним од стране разних произвођача. Његова улога је да уређају говори шта да уради, баш као што Windows оперативни систем покреће лаптопове и десктоп рачунаре. Он се заснива се на Linux језгру, а имплементира виртуалну машину Java (Dalvik), прегледач веба на бази *WebKut*-а, базу *SQL* и комплетан приступ хардверу уређаја.

Када се инсталира Androud апликација на Androud уређај тада се заправо инсталира апликација која је написана посебно за Androud оперативни систем. Androud апликација може се инсталирати и на Windows рачунарима иако тада рачунар треба имати посебан софтвер који се назива androud emulator. Он ствара virtualni Androud уређај на рачунару[3][4].

Android за проширивање функционалности уређаја има доста програмера за писање апликативних програма. Постоји више од 450.000 апликација доступних за Android, што га чини другим најпопуларнијим мобилним развојним окружењем. Програмери пишу контролисани код у Java језику, контролишући уређај преко Googl развојне Java библиотеке.

Android OS се састоји од 12 милиона линија кода, укључујући 3 милиона линија XML кода, 2,8 милиона линија C кода, 2,1 милиона линија Java кода и 1,75 милиона линија C++ кода. Android представља софтверски стек који је намењен за мобилне телефоне који у себи укључује оперативни систем, посреднички софтвер као и апликације за нормално функционисање.

3.1.АНДРОИД ПЛАТФОРМА

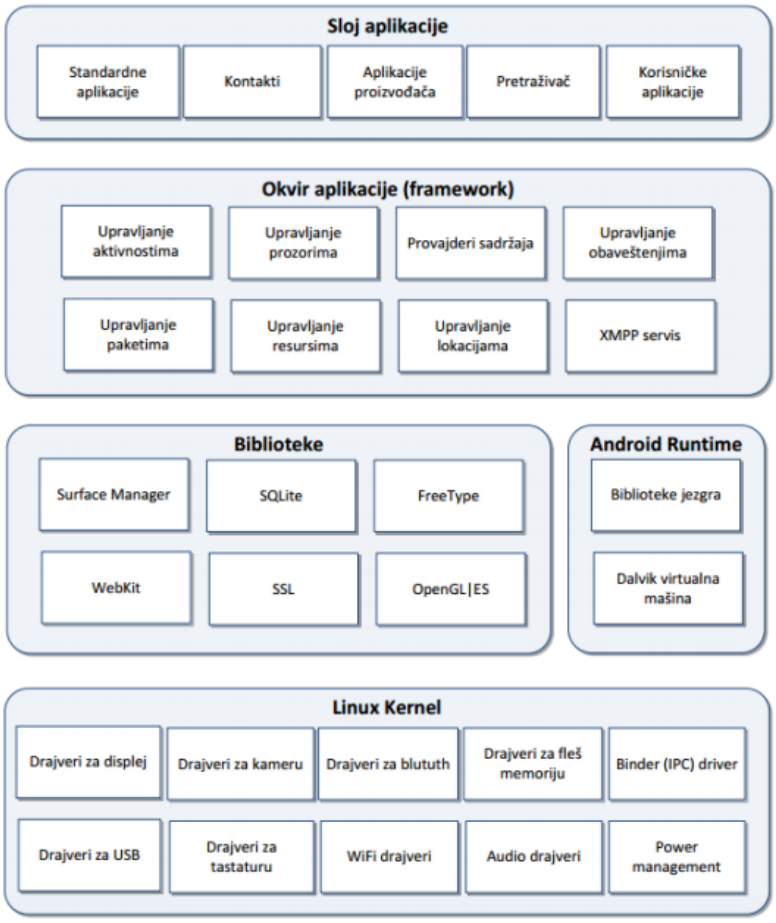
Android SDK обезбеђује алатке и API који су неопходни за даљи развој апликација на Android платформи користећи JAVA програмски језик.

Карактеристике Android програмског окружења:

- отвореност - програмеру омогућава потпуну слободу у развоју нових и већ постојећих апликација, а произвођачу уређаја коришћење и прилагођавање платформе без плаћања ауторских права;
- све апликације су равноправне - не постоји разлика између основних језгарних апликација уређаја и додатних апликација. Свим апликацијама омогућен је равноправни приступ ресурсима покретног уређаја што омогућава прилагођавања уређаја специфичним потребама појединачног корисника;

- аутоматско управљање животним циклусом апликације - омогућава надзор покретања и извршавања апликација на системском нивоу оптимизованим коришћењем меморије и снаге уређаја. Крајњи корисник више не брине о гашењу одређених апликација пре покретања других;
- уклањање граница класичних апликација - могућност развоја нових и иновативних апликација заснованих на међусобној колаборацији технологија;
- брз и једноставан развој апликација - омогућен је богатом базом корисних програмских библиотека (eng. libraries) и алата за израду апликација;
- висококвалитетни графички приказ и звук - подржана 2D векторска и 3D OpenGL (engl. Open Graphics Librari) графика, те уграђени кодеци свих често коришћених аудио и видео формата;
- компатибилност са већином садашњег и будућег хардвера - укључује преносивост Androidovih апликација на APM, к86 и остале архитектуре, те прилагодљивост система улазним и излазним компонентама.

Архитектура Android система базира се на Linux-у 2,6 језгру (eng. kernel) користи се као слој апстракције хардвера (HAL, eng. Hardvare Abstraction Laier). Разлог за коришћење језгра оперативног система Linux је доказана погонска подршка (eng. driver model), могућност управљања меморијом и процесима, сигурносни модел, мрежни систем, константни развој и унапређивање система. Слика 1. приказује архитектуру Android система.



Слика 1. Архитектура андроид платформе [3]

3.2.ИЗВОРНЕ ПРОГРАМСКЕ БИБЛИОТЕКЕ

Изворне програмске библиотеке (eng. native libraries) писане су у програмским језицима C и C++ и чине следећи слој у архитектури система. Неке од значајнијих су:

- UI - програм за управљање графичким интерфејсом (eng. Surface Manager) - библиотека која је одговорна за правилно исцртавање различитих апликационих компоненти у времену и простору;
- OpenGL ES (eng. OpenGL for Embedded Sistems) - библиотеке које се користе за хардверску 3D акцелерацију или за 3D растеризацију;
- SGL (eng. Scalable Graphics Librari) - представља 2D библиотеке на којима је заснована већина апликација. Потребно је споменути да 2D и 3D елементи се могу комбиновано приказивати у једном корисничком интерфејсу;
- Media Framework - група кодека за снимање и репродукцију аудио формата, видео формата и непомичних слика. Која је омогућена од стране PacketVidea;
- FreeType - библиотека која служи за векторску растеризацију облика писма (eng. font);
- SSL (eng. Secure Socket Laier) - омогућава резервну комуникацију преко Интернета;
- SQLite - програмска библиотека која имплементира базу података (eng. database engine);
- WebKit - Језгро претраживача који подржава JavaScript и остале стандарде на малом уређају;
- System C library - имплементација стандардне C-ове системске библиотеке (libc) изведене из оперативног система BSD.

3.3. РАДНО ОКРУЖЕЊЕ - ANDROID RUNTIME

Следећи слој у архитектури Androida је радно окружење (eng. Android runtime) којег чине виртуелна машина Dalvик (DVM, eng. Dalvik Virtual Machine) и језгрене библиотеке (eng. core librari).

DVM је регистарски базирана виртуелна машина, док је класична Javina виртуелна машина (JBM, eng. Java Virtual Machine) базиран на стеку. Базне библиотеке писане су у програмском језику Java и представљају све есенцијалне класе као што су класе за манипулацију колекцијама, класе за комуникацију са околином и слично. Битна новост је и то што се Androidom језгрене библиотеке разликују од библиотека у Java Standard Edition (J2SE) и Java 2 Micro Edition (J2ME).

Уместо употребе стандардног Java 2 Micro Edition (J2ME) као механизма за покретање Javinih апликација на мобилним уређајима, Google је развио сопствену виртуелну машину за Android. DVM је највероватније развијен како би се заобишла проблематика с дозволама око коришћења Sunovog J2ME. Сваки мобилни уређај који у себи садржи J2ME мора код Suna лиценцирати било какву промену изворног кода J2ME-a.

Виртуелна машина Dalvik добила је име по истоименом исландском граду у којем су живели преци главног девелопера DVM-a Dan Bornsteun. Основна разлика између Sun Java виртуелних машина и DVM-a је у томе што су прво споменути виртуални уређаји базирани на стеку, док је DVM регистар базиран на виртуелној машини.

Међукод (eng. bytecode) Dalvik виртуелне машине трансформише се помоћу алата dk (који је саставни део Андроид SDK-a) из Javinih класних датотека (eng. Java class file) преведених Javinim преводиоцем у нову класу *.dex (eng. Dalvik Ekecutable) формата.

Међукод који извршава DVM није Javin међукод, већ управо споменути .dex облик. Трансформација формата омогућава бољу прилагођеност за рад на мањим процесорима бољим искоришћавањем расположиве меморије и процесорске снаге. Резултат свега је могућност вишеструког инстанцирања саме виртуелне машине што значи да се свака Android апликација покреће као засебни процес, с властитом инстанцом виртуелне машине Dalvik.

3.4. АПЛИКАЦИОНИ ОКВИР

Слој апликативних оквира (eng. Application Framevork) написан је у програмском језику Java и могуће је проширити скуп програмских компоненти којег користе све апликације уређаја. Framework подржава многобројне open source језике као што су openssl, sqlite и libc. Такође подржава и језик Android језгра. Са гледишта сигурности, framework се базира на UNIX file system овлашћењима која осигуравају да апликације поседују само оне могућности које им је власник телефона дао при инсталацији апликације.

Неки од важнијих елемената су:

- управљање активностима (eng. Activiti Manager) - управљање животним циклусом апликације;
- управљање програмским пакетима (eng. Package Manager) - садржи информацију о апликацијама инсталираним на систему;
- управљање прозорима (eng. Window Manager) - управљање апликационим прозорима;
- управљање позивима (eng. Telephone Manager) - садржи API-је који се користе при изради апликација за управљање позивима;
- пружаоци садржаја (eng. Content Providers) - омогућавају заједничко коришћење података од стране више апликација;
- управљање ресурсима (eng. Resource Manager) - служи за складиштење делова апликације који нису код (нпр. слике);
- систем графичког приказа (eng. Viev Sistem) - садржи базу готових графичких приказа и алата (eng. vidget);
- управљање локацијски заснованим услугама (eng. Location Manager) и
- управљање нотификацијама (eng. Notification Manager) - управљање нотификацијама и догађајима.

Апликативни слој је последњи слој у архитектури система Android и чине га корисничке апликације уређаја. Укључује неке од стандардних системских апликација као што су web претраживач, листа контаката, телефон, итд.

3.5. ЖИВОТНИ ЦИКЛУС АКТИВНОСТИ

Активност (Activity) представља компоненту апликације која се може поистоветити с једним прозором апликације у којем је корисник у могућности да изврши одређену радњу. Активност служи да би кориснику мобилног телефона омогућила позивање одређеног броја, сликање фотографије коришћењем уграђене камере, слање e-maiл поруке, преглед мапе и др. Апликација може да садржи једну или више дефинисаних активности, где је једна од активности увек дефинисана као примарна активност. Прелаз између активности се одвија тако што актуелна активност позива нову активност. Иако више активности чини један компактан кориснички интерфејс, треба имати на уму да су оне међусобно независне. Свака активност се имплементира тако што засебна класа наслеђује класу Activity, па је сама одговорна за чување свог стања у животном циклусу апликације.

Главни део сваке Android апликације је класа активности. Програмер већи део времена проводи на дефинисању и имплементацији активности за сваки екран у апликацији.

Свакој Android апликацији мора бити додељена активност унутар Android manifest fajla. Друге класе активности могу бити означене у манифест фајлу да се покрену при одређеним околностима. Овим секундарним улазним тачкама се управља помоћу Android manifest fajla са одређеним филтерима.

Апликације могу бити прекинуте због разних догађаја вишег приоритета, као што је телефонски позив. У једном тренутку може постојати само једна активна апликација. Андроид апликације су одговорне за управљање њиховим стањем, као и њиховом меморијом, ресурсима и подацима. Андроид оперативни систем може искључити активност која је на паузи, стопирана или уништена када је меморија слаба. Свака активност која није у првом плану мора бити искључена. Другим речима, Android апликација мора одржавати стање и бити спремна да буде прекинута или чак искључена у сваком тренутку.

Животним циклусом активности једне Android апликације управља се имплементацијом одговарајућих метода, слика 2. Свака активност има три основна стања:

- Resumed (Running) - Активност је покренута.
- Paused - Друга активност је покренута и она је у првом плану, али је текућа активност и даље покренута и видљива, али не заузима цео екран. Активност чије је стање paused је потпуно активна, међутим ова активност може да буде диспозована у случају ниске слободне меморије.
- Stopped - Активност чије је стање stopped је и даље активна, ради у позадини. Објекат класе Activvity и даље се налази у меморији, чува сва стања, али се више не налази у оквиру window manager-a. Ову активност више не види корисник и у случају потребе за меморијом систем може да је заустави.

Метода која се користи за почетак животног циклуса је onCreate(), а завршава се заједно са методом onDestroy(). У оквиру методе onCreate(), програмер би требало да дефинише изглед и глобално стање активности, као што је распоред елемената корисничког интерфејса. Имплементацијом методе onDestroy() требало би ослободити коришћене ресурсе.

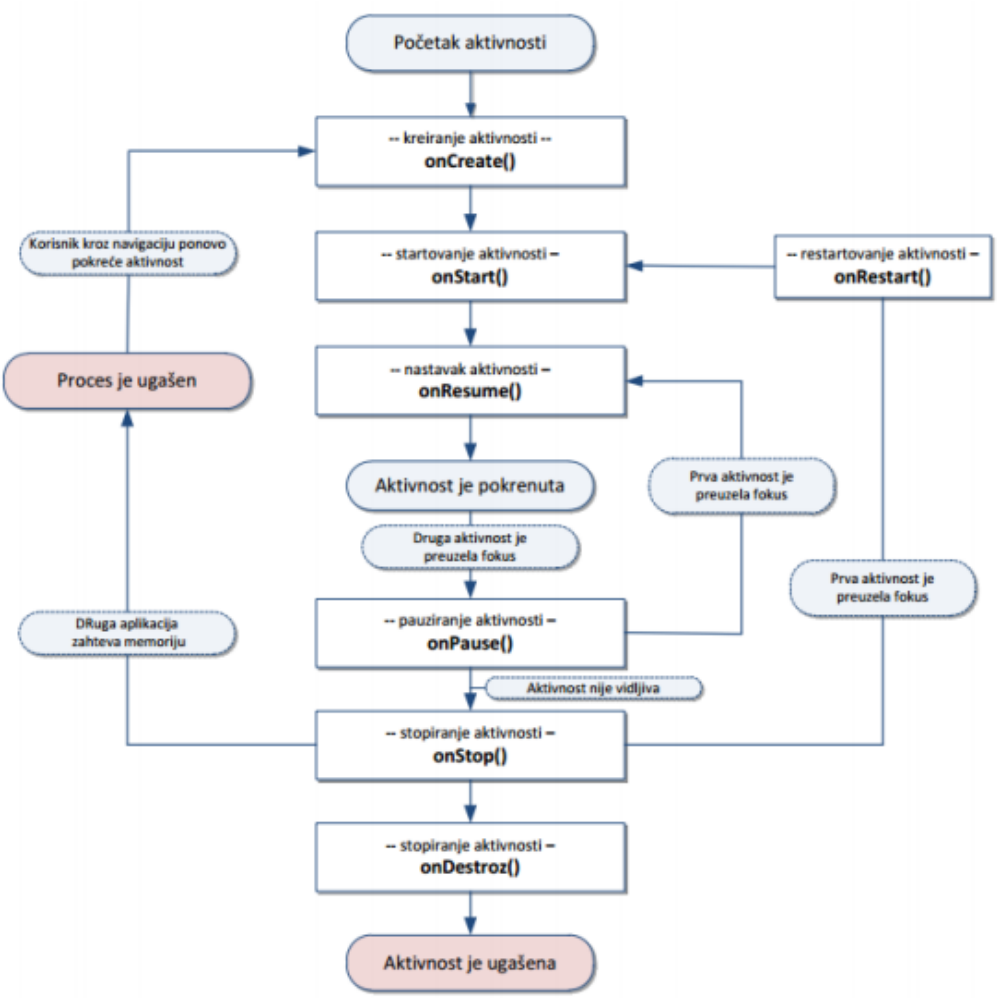
Активност је видљива (visible) између позива метода onStart() и onStop(). Док је активност видљива, корисник је види на екрану и може да врши интеракцију са њом (нпр. да уноси податке). Метода onStop() се позива када се стартује нова активност, а текућа активност више није видљива.

Активност је фокусирана (у првом плану) између позива метода onResume() и onPause(). Тада се та активност налази испред свих осталих активности. Активност може често да прелази из првог плана у позадину и обрнуто. Метода onPause() се позива када уређај пређе у стање мировања (sleep).

3.6. РЕСУРСИ У АНДРОИД АПЛИКАЦИЈИ

Све android апликације састоје се од две ствари:

- функционалност – представља код како се апликација понаша на неки захтев. Овде спадају сви алгоритми који покрећу апликацију
- ресурси или подаци – обухватају текстове, слике, аудио и видео фајлове, датотеке, иконе и друге податке које користи апликација. Ресурси обухватају текст у облику стринга, слике и иконе, аудио датотеке, видео записе и друге податке које користи апликација.



Слика 2. Животни циклус активности [1]

Ресурси се деле на два типа: ресурсе апликације и ресурсе система. Ресурсе апликације дефинише developer унутар фајлова Android пројекта и тачно су одређени за неку апликацију. Ресурси система су стандардни ресурси који су дефинисани Android платформа и доступни су свим апликацијама кроз Android SDK.

Ресурси апликације се праве и чувају унутар Android пројекта у директоријуму. Користећи већ дефинисану али флексибилну структуру директоријума, ресурси су организовани, дефинисани и упаковани са пакетом апликације. Ресурси апликације се не деле са остатком Анроид система. Сви ресурси апликације се чувају у структури директоријума и састављени су у пројекат у тренутку прављења апликације. Ресурси апликације се могу користити програмерски. На њих се могу односити и ресурси других ресурса апликација.

Сваки пут кад се сачува фајл са ресурсима у Eclipsu, R.java klasa fajl се поново саставља како би се убациле нове измене. Ако није коришћен прави директоријум или правила о називу фајла, појављује се грешка у компајлеру у прозору Eclipse Problems. Resursima апликације се може приступити програмерски користећи генеришу класу која се зове R.java. Како би се додала референца на ресурс, мора се доћи до објекта ресурса апликације користећи класу getResources() и тако направити одговарајући позив на команду, који је заснован на типу ресурса који жели да се дода.

Developeri могу да приступе ресурсима Android система (као додатак ресурсима апликације). Могу се делити многи ресурси система кроз више апликација због обичног изгледа и осећаја.На пример, класа ресурса Android система (androud.R.string) садржи стрингове за речи као што су OK, Cancel, Yes, No, Cut, Copy и Paste.

Системски ресурси се чувају у андроид пакету. Постоје класе за сваки од већих типова ресурса. На пример, класа android.R.string садржи string системског ресурса. Како би се добио string системског ресурса за OK прво се мора користити статичкиа команда класе Resursa getSystem() како би се довео глобални системски објекат Resource. Онда се позива команда getString() са одговарајућим string именом ресурса.

4.ОБРАДА ПОДАТАКА И МОБИЛНОСТ

Информација од стране корисника с мобилним уређајима као што су мобилни, PDA (Personal Digital Assistant), постала су свакодневна активност. Навигацијски систем у возилима су сада стандардна опрема као што су и музички систем. Ови инструменти су врло корисни и разумљиви, јер они могу преузети жељене податке из базе података с било којег места путем бежичног канала. Али имају озбиљна ограничења: проток информација у тим системима је само од сервера до корисника. Ова ограничења не допуштају корисницима упите и манипулисање базе података које се могу налазити било где у свету. У терминологији базе података тих система нису способни за управљање с трансакцијским активностима.

Мобилна обрада података, укључује мобилност хардвера, података и софтвера у рачунарским апликацијама. Може бити посматрана као специјализована класа дистрибуираних система који се састојао од модела клијент-сервер.

Стручњаци који истражују базе података, практичари и комерцијалне организације имају заједничку визију изградње система за управљање информацијама на мобилној платформи која је у стању да пружити пуно управљање трансакцијама и функционалност базе података с било којег места и било када.

Постоји велики број ситуација у којима је примена мобилних база података оптимално и често једино решење. Неке од њих су:

- мобилним корисницима мора бити омогућен рад у условима када бежична веза није присутна усред слабе или сасвим одсутне везе;
- апликације морају омогућити значајну интеракцију;
- апликације морају бити у стању да приступе локалним уређајима као што су штампачи, бар код скенери или GPS јединицама (за мапирање или системи за аутоматско одређивање положаја возила);
- пропусност мора бити гарантована;
- корисници не захтевају приступ свим изворним подацима, већ само скоро модификованим подацима;

Ако апликација задовољава било који од ових захтева велике су шансе да ће најбоље решење бити пројектовање мобилне базе података са могућношћу синхронизације.

Коришћење мобилних радијских система брзо се проширила, а то је требало контролисати употребу радијских фреквенција. Године 1934. Конгрес Сједињених Америчких Држава створио је Федералну комуникацијску комисију (Federal Communicatиons Commissиon-a (FCC)), којој је уз регулацију система за земљишну телефонску линију, такође успело коришћење тих фреквенција. 1935. фреквенцијску модулацију је изумио и развио је професор на Свеучилишту Columbиa Maj. Edwin H. Armstrong, који се користи за побољшање мобилне радио комуникације. Године 1940. нове фреквенције између 30 и 40 MHz постају доступне од стране FCC, која пружа потребне ресурсе и појединцима да раде своје мобилне јединице. У истој години Connecticut State државна полиција у Hartfordu и већина полицијских система је даље земље претворила у FM технологију. То је обележило рађење мобилне телефоније.

Мобилно окружење пружа апликацијске базе података с корисничким аспектима бежичне технологије, који је познат као мобилна база података. Овај напредак у технологији је створио ново

доба корисника базе података. Корисници приступају бази података кроз мрежу, корисник с бежичном везом спојен на информацијску мрежу не захтева одржавање сталног присуства у мрежи[6].

4.1.ТИПОВИ МОБИЛНОСТИ

Елементи мреже у мобилном окружењу су врло динамични и могу бити врло променљиви. База података представља информације о покретним објектима и њиховом положају поред информације о стационарним објектима. Један од важних параметара је положај мобилног уређаја при проналажењу тих уређаја који могу држати потребне податке при избору података, посебно за локацију зависну о информацијским услугама.

У том случају трошкови за претраживање, да би се пронашао мобилни уређај, додају се на цену сваке комуникације која их укључује. Мобилни оквир се састоји од жичних и бежичних компоненти људских корисника. Мобилна наука дефинише две врсте мобилности:

- терминална мобилност (terminal mobility) и
- лична мобилност (personal mobility).

У терминалној мобилности, веза је успостављена између две тачке, а не између две особе, повезујући један другог. Овај тип везе омогућује коришћење комуникацијских уређаја који се деле између било кога. То омогућују мобилни уређаји (лаптоп, мобилни, PDA, итд.) за приступ жељених услуга с било којег места, док је у покрету или мирује, без обзира на то ко носи уређај. Мобилни се може користити од стране свог власника или од стране неког. У терминалној мобилности, одговорност бежичне мреже је идентификација комуникацијских уређаја.

У личној мобилности подржана је мобилност особе. Корисник не мора носити никакву комуникацијску опрему са собом; већ може користити било који комуникацијски уређај за успостављање контакта с друге стране. Овај објект захтева идентификацијски план провере особа које желе комуницирати.

5. СИСТЕМ ЗА УПРАВЉАЊЕ МОБИЛНИМ АПЛИКАЦИЈАМА ПОДАТАКА

Сада сви главни добављачи система за управљање базом нуде мобилне DBMS. Реч је о развоју који је деломично одговоран за раст драматичног који се продаје за велике DBMS добављаче. Већина произвођача промовише своје мобилне DBMS-ове као способне за комуникацију с низом великих релацијских DBMS-ова и пружање услуга базе података које захтевају ограничене рачунарске ресурсе како би одговарали онима који имају мобилне уређаје.

Додатна функционалност потребна мобилним DBMS-овима укључује способност да:

- комуницирају с централизираним сервером базе података кроз начине, као што су бежични или жични приступ интернету;
- могу понављати податке о централизираном серверу базе података и мобилних уређаја;
- могу синхронизовати податке о централизираном серверу базе података и мобилних уређаја;
- прикупљају податке из различитих извора, као што су на интернету;
- управљају подацима на мобилном уређају;
- анализирају податке на мобилном уређају и
- створе прилагођене мобилне апликације.

Мобилни DBMS-ови требају задовољити следеће услове:

- Мали меморијски отисак;
- Flash-optimized систем;
- Синхронизацију података;
- Сигурност и
- Ниска потрошња енергије[6].

Све више и више мобилних апликација захтева податке за рад, а база података је већ неко време представља стандард за складиштење и управљање подацима. У типичном сценарију, мобилна апликација користи базу података у Cloud-у, а повезује на даљину са коисницима како би они приступили својим подацима. То подразумева активну и врло брзу мрежну везу.

Уградиве базе података су лагане, самосталне датотеке без серверских компоненти, без потребе за администрацијом, као и са ограниченим изворима. Мобилне апликације могу бити статички или динамички повезане с њима.

SQLite је open-source C језика, која пружа релациони систем за управљање базом података (RDBMS). Може бити сачуван и на диску и екстерној меморији. Она подржава динамичко писање, трансакције, потпуно текстуално претраживање и приступ.

SQLite чува сваку базу података као једну датотеку. То значи да се можете креирати SQLite база података у једном реду, а затим се користити у другом реду с различитом архитектуром с једноставним копирањем одговарајућих датотека. SQLite је такође укључен и у iPhone и Android оперативном систему.

SQLite база података може се користити на web страницама без обзира имале оне малу или средњу количину прегледа, што значи да одлично функционише са страницама које имају 100.000 прегледа по дану. Међутим доказано је да има могућност радити чак и с милион прегледа по дану. SQLite је идеалан избор за уграђивање у сервисе који би требали радити аутоматски и без подршке људи[7].

За израду базе података потребно је инсталирати SQLite програм. Како би инсталирали програм путем SQLite странице за преузимање, треба преузети потребну датотеку на свом рачунару. Након тога важно је створити мапу C: SQLite у коју се распакују две зип датотеке у којима се налазе sqlite.def, sqlite.dll и sqlite.exe датотеке. За најлакше креирање база користи се програм под називом SQLite Browser. SQLite Browser је висококвалитетни и визуални open-source алат за израду, дизајн и уређивање базе података компатибилно с SQLite-ом. За инсталацију овог програма потребно је преузети потребну датотеку на свом рачунару и покренути га. Након једноставних корака који нас воде кроз инсталацију програм је спреман за израду базе.

База података користе се за чување, манипулацију и прикупљање података. База је организована и уређена целина међусобно повезаних података. База података омогућава приступ подацима више корисника и складиштење и преузимање података сачуваних у бази података путем сервера. Настанак базе података везана је за систем за аутоматску обраду података пописа становништва у Америци. Patent је пријавио 1884 године Herman Holerith. Његов систем заснивао се на бушеним картицама које су се ручно уносиле у уређај за очитавање, док је систем аутоматски пребројавао податке. Системи управљања базом података можемо поделити у 4 основна модела:

- Хијерархијски модел; - подаци су смештени у серију слогова (записа). Да би се успоставила веза између слогова, хијерархијски модел успоставља релацију родитељ – наследник.
- Мрежни модел - омогућава да се вишеструки скупови података користе заједно путем показивача (или поинтера). Неке колоне садрже показиваче на друге табеле уместо самих података. На тај начин, табеле су повезане показивачима и могу се посматрати као мрежна структура. Док у хијерархијском моделу сваки слог има један „родитељски“ слог и неограничено „наследника“, мрежни модел омогућава сваком запису да има вишеструке родитеље и наследнике, креирајући мрежасту структуру
- Релациони модел - у срцу овогмодела налази се концепт табеле (која се назива и релација) у којој су смештени сви подаци. Свака табела је начињена од слогова (редова у табели), а сваки слог има своја поља (атрибуте).
- Објектни модел - заснива се концепту објеката, који представљају скуп података и операција које се на њима могу извршавати

MySQL је релацијска база података тј. релацијски систем управљања базама података. MySQL је отвореног кода што значи да исти дозвољава кориштење свакоме и прилагођавање својим потребама. Дистрибуиран је под GPL лиценцом (GNU General Public License-бесплатно за кућну употребу). Сам приступ подацима и претраживање базе података остварује се помоћу SQL упита (енг. Structured Query Language- структурни језик за претраживање) који је уједно најчешћи стандардизован језик за приступање базама података.

Дистрибуирана мрежа све више постаје стандард за развој предузећа и установа. Са порастом мобилности радне снаге и децентрализацијом корпорацијских огранака, предузеће мора пажљиво радити на осигурању потребних информација свим расположивим корисницима у свакој организационој јединици предузећа и установе.

Оно што компликује овај задатак је чињеница да су ти корисници веома често на удаљеним местима где могу имати само ограничене пропусне капацитете за повезивање на мрежу.

Посебан случај су мобилни корисници, као што су продавци на терену који доста времена уопште нису пријављени на мрежу а којима је ипак потребно обезбедити приступ различитим апликацијама како би обезбедили очување перформанси.

Дистрибуираној обради овакви услови представљају изазов у једном континуираном пословном процесу где је потребна висока расположивост података. Када водећа предузећа и установе данас захтевају врхунску репликацију и синхронизацију података ради дистрибуиране обраде података, она се окрећу ефикасном решењу у виду DDBMS (Distributed Data Base Management System) који је систем за управљање дистрибуираним базма података. Под дистрибуираном обрадом упита се подразумева дистрибуирана оптимизација као и дистрибуирано извршавање упита[8].

Предности које доноси дистрибуирани концепт базе података:

- Локална аутономија података, управљања и контроле. Окружење у којем се DBP примењује је и само дистрибуирано. Дистрибуирање база података као и систем за управљање њима, омогућава појединим групама да локално контролишу властите податке, уз могућност приступа подацима на другим локацијама када је то потребно.
- Већи капацитет и поступни раст. Чест разлог за инсталирање дистрибуираног система је немогућност једног рачунара да прими и обрађује све потребне податке. У случају да потребе надмаше постојећи капацитет, додавање чвора дистрибуираном систему је од великог значаја и једноставније него заменити систем већим.
- Поузданост и расположивост. Дистрибуирани системи могу наставити своје функционирање и када неки од чворова изгубе функционалност.
- Ефикасност и флексибилност. Подаци су физички близу ономе ко их ствара и користи, па је знатно смањена потреба за удаљеном комуникацијом[8].

Систем дистрибуираних база података је систем од две или више база података које би апликацијама требале као једна јединствена база података. Многе организације дистрибуирају копије или делове базе података на различите мрежне сервере. Дистрибуиране базе се могу налазити на различитим серверима Web, интранет, ехтранет или неке друге компанијске мреже. Дистрибуиране базе могу бити копије операцијских или аналитичких, хипермедијалних или неких других типова база података.

Дистрибуција база података умањује перформансе и сигурност базе података. Гаранција да ће сви подаци у организацијским дистрибуираним базама података бити конзистентно ажурирани је један од главних задатака дистрибуираних DBMS-а[9].

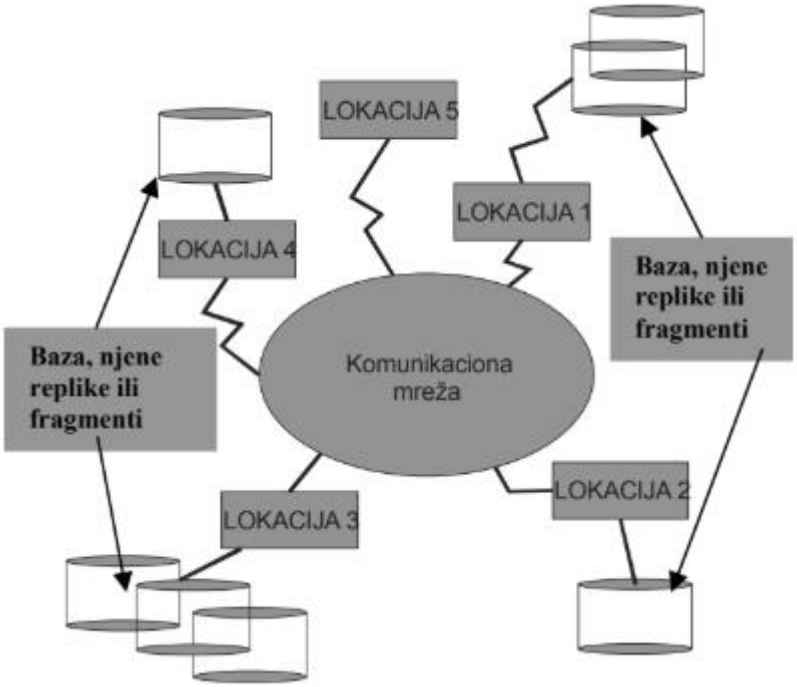
Дистрибуирана база података је база података која се не налази цела на једном рачунару, већ је подељена на више локација које су повезане комуникацијском мрежом. Свака локација, која се зове и чвор комуникацијске мреже има свој властити систем за управљање базама података, са властитом контролом, односно управљачем трансакција и оправка од пада, и осталим важним функцијама, а има и свој процесор и улазно/излазне уређаје. Апликације истовремено приступају и мењају податке на више различитих база података у мрежи, где мрежа може бити LAN или WAN.

Основни значај DBP:

- Скуп логички повезаних података који се деле;
- Подаци су раздвојени на више фрагмената;
- Фрагменти се могу копирати;
- Фрагменти припадају локацијама;

- Локације су повезане комуникацијском мрежом;
- Подаци на свакој локацији су под надзором DBMS-а;
- DBMS на свакој локацији може управљати локалним апликацијама аутономно;
- Сваки DBMS учествује у најмање једној глобалној апликацији.

Дистрибуциона база података је колекција великог броја, логички интегрисаних база, дистрибуираних преко одређене хетерогене рачунарске мреже. Систем за управљање дистрибуционим базама података је софтвер који управља механизмом приступа дистрибуираним базама података и чини га транспарентним за све кориснике. То је више или мање спрегнут мултипроцесорски рачунарски систем са синхронизованом временском поделом задатака, Слика 3.



Слика 3. Окружење DDBMS

Развој система управљања базама података је помогао да подаци у потпуности постигну своју независност, без обзира на којој локацији у мрежи се налазе. Систем обезбеђује централизовани и контролисани приступ у администрирању података, њихову транспарентност као и да апликација бива имуна на физичку и логичку организацију датотека. Дистрибуирани системи база података имају следеће предности:

- Већу поузданост која се огледа у прављењу копија – Копирање свих компонената базе. Грешке се при томе своде на минимум, јер прекинута комуникациона елеменат обраде не обара цео систем, подаци ће е преузети са неког другог места. Дистрибуирана обрада гарантује доследност базе података и конкурентност.
- Боље перфомансе због могућности одређивања најкраћег пута и најбољих карактеристика мреже у датом тренутку, због таквог распореда података на дистрибуираним серверима база Да се најчешће коришћени фрагменти базе могу налазити у близини клијената који их потражују. На основу овога се смањују кашњења одзива на упите базе са неког удаљеног места.

Од великог је значаја, у зависности од пропусности мреже и времена кашњења, правилно одабрати релације и њихове фрагменте који ће бити преношени са једне локације на другу у циљу обраде упита (глобална оптимизација). У зависности од тога да ли је у оквиру дистрибиуране базе окружење свих DBMS система хомогено или нехомогено, може се лакше или мало теже вршити проширивање система са новим нодовима који одмах могу располагати свим транспарентним копијама дистрибуираних података. У систему DDBMS је на тај начин обезбеђена спољашња и унутрашња паралелност упита.

Једноставни системски развој омогућују појединачни DBMS системи који се налазе на свакој локацији и могу да раде независно при извршавању упита, јер сваки нод локалне базе података мора бити равноправан са осталим нодовима у бази и може се користити независно од остатка дистрибуираног система. Ово се назива локалном аутономијом сваког нода.

Развој система прати глобални каталог, као системска база података или њен део који садржи информације о разним објектима у систему као што су базне табеле, погледи, индекси, базе података, планови извршавања упита, овлашћења за приступ објектима, итд. У случају дистрибуираног система каталог садржи информације о начину и локацијама на које су подаци раздељени и поновљени[9].

6. РАЗВОЈ И ИМПЛЕМЕНТАЦИЈА СИСТЕМА ЗА СИНХРОНИЗАЦИЈУ МОБИЛНИХ АПЛИКАЦИЈА НА SQL СЕРВЕРУ

6.1. АЛАТИ КОРИШЋЕНИ ЗА АПЛИКАЦИЈУ

За развој апликације поред потребног познавања андроид програмирања, потребно је познавати прграмски језик SQL, HTML, PHP.

SQL (Structured Query Language) је стандардни релациони упитни језик (ANSI и ISO стандард) над релационим базама података. Данас се за SQL може рећи да је комплексан, процедурално/декларативан језик. SQL ради са табелама. Табела се креира једном извршном наредбом. Одмах по креирању табела је расположива за коришћење. Сви подаци меморисани су у табелама и резултат, било које операције, се логички приказује у облику табеле.

PHP је open–sorce server–side програмски језик за динамичко генерисање HTML кода. Другим речима, PHP је програмски језик помоћу којег се може креирати HTML страница на серверу пре него што се она пошаље клијенту попуњена динамичким садржајем. Овим начином генерисања садржаја клијент не може видети код (скрипту) који је генерисао садржај који гледа, већ има приступ чистом HTML коду.

6.1.1. ЈЕЗИК РЕЛАЦИОНИХ БАЗА ПОДАТАКА - SQL

База података је скуп међусобно повезаних података, смештених у спољној меморији рачунара. Подаци су истовремено доступни разним корисницима и апликацијским програмима. Убацивање, промена, брисање и читање података обавља се посредством заједничког софтвера. Корисници и апликације притом не морају познавати детаље физичког приказа података, већ се референцирају на логичку структуру базе.

Комуникација корисника односно апликацијског програма и DBMS-а одвија се помоћу посебних језика. Ти језици традиционално се деле на следеће категорије:

- Језик за опис података (Data Description Language - DDL). Служи пројектанту базе или администратору у сврху записивања шеме или погледа. Дакле, тим језиком дефинишу се подаци и везе међу подацима, и то на логичком нивоу. Постоји посебна варијанта језика за шему, а посебна за погледе. Наредбе DDL обично подсећају на наредбе за дефинисање сложених типова података у језицима попут COBOL, PL/I, C, Pascal.
- Језик за манипулисање подацима (Data Manipulation Language - DML). Служи програмеру за успостављање везе између апликацијског програма и базе. Наредбе DML омогућују "маневрисање" по бази података, као и једноставне операције као што су упис, промена, брисање или читање записа. У неким софтверским пакетима, DML је заправо библиотека потпрограма: "наредба" у DML своди се на позив потпрограма. У другим пакетима заиста се ради о посебном језику: програмер тада пише програм у којем су измешане наредбе двају

језика, па такав програм треба преводити с два преводиоца (DML-precompiler, обични compiler).
- Језик за постављање упита (Query Language - QL). Служи непосредном кориснику за интерактивно претраживање базе. То је језик који подсећа на говорни (енглески) језик. Наредбе су непроцедуралне, такве да само специфицирају резултат који желимо добити, а не и поступак за добијање резултата.

Оваква подела на три језика данас је већ прилично застарела. Наиме, код релационих база постоји тенденција да се сва три језика обједине у један свеобухватни. Пример таквог интегрисаног језика за релацијске базе је SQL: он служи за дефинисање података, манипулисање и претраживање. Интегрисани језик се може користити интерактивно (преко онлине интерпретера) или се он може појављивати уклопљен у апликацијске програме.

Овде треба напоменути, да горе споменуте врсте језика нису програмски језици. Дакле ти језици су нужни да би се повезали с базом података, али они нису довољни за развој апликација које ће нешто радити с подацима из базе.

Традиционални начин развоја апликација које раде с базом је коришћење класичних програмских језика (COBOL, PL/I, C, Pascal . . .) с угњежденим DML – наредбама. У 80-тим годинама 20. века били су доста популарни и језици 4. генерације (4-th Generation Languages - 4GL): реч је о језицима који су били намењени искључиво за рад с базама, па су зато, у том контексту, били продуктивнији од класичних програмских језика опште намене. Проблем с језицима 4. генерације је био у њиховој нестандардности: сваки од њих је у правилу био део неког одређеног софтверског пакета за базе података, па се није могао користити изван тог пакета (базе).

У данашње време, апликације се најчешће развијају у стандардним објектно оријентисаним програмским језицима (Java, C++, VisualFox Pro, Visual Basic, PHP, JavaScript, ...). За интеракције с базом користе се унапред припремљене класе објеката. Оваква техника је довољно продуктивна због коришћења готових класа, а коначни програм се лако дотерује, уклапа у веће системе или преноси с једне базе на другу.

Пошто је тема овог специјалистичког рада развој PHP апликације за рад са базама података, у наставку овог поглавља се дају основни принципи SQL – а, као интегришућег елемента за PHP апликацију која је развијена за потребе овог рада.

SQL (Structured Query Language) је стандардни релациони упитни језик (ANSI и ISO стандард) над релационим базама података. SQL скраћеница, у ствари значи "Structured Query Language" што би се могло превести као структурирани језик за упите. Његов творац је Чемберлин (Chamberlin), а настао је у IBM–овој истраживачкој Лабораторији (IBM Research Laboratory) у San Hoseu, Калифорнија 1974. године, дакле на истом месту где је Код 1970. дефинисао основне концепте релационог модела података, кроз 12 правила.

Правила гласе :
- Релациони DBMS (Database Management System) – систем за управљање базама података мора бити у могућности да управља у потпуности базама података кроз своје релацијске могућности.
- Правила о информацијама - све информације у релационој бази података (укључујући имена табела и колона) представљају се као вредности у табели.
- Осигуравање приступа - свакој се вредности у релационој бази података може приступити употребом комбинације имена табеле, вредности примарног кључа и имена колоне.

- Системска подршка недефинисаним вредностима – DBMS осигурава системску подршку у раду са недефинисаним величинама (непознати или непримењиви типови података), који се разликују од дефинисаних вредности и независни су.
- Активан, увек доступан Релациони каталог - опис базе података и њеног садржаја је представљен на логичком нивоу у виду табела и може се претраживати помоћу језика базе података.
- Разумљив подјезик података - бар један подржани језик мора имати добро дефинисану синтаксу и бити разумљив. Мора се подржати дефиниција података, управљање подацима, правила интегритета, ауторизација и трансакције.
- Правило за ажурирање погледа - сви погледи који се теоретски могу ажурирати, ажурирају се кроз систем.
- Уношење, ажурирање и уклањање података на нивоу скупова – DBMS за добијање података на нивоу скупова и за уношење, исправке и уклањање података.
- Физичка независност података - мењање физичког записа структуре или методе приступа не утиче на апликације или програме.
- Логичка независност података - колико је год могуће, промена структуре табела не утиче на апликације или програме.
- Независност интегритета -језик базе података мора осигурати начин за дефинисање правила интегритета. Она морају бити сачувана у каталогу, који је увек доступан и не може се игнорисати.
- Независност од дистрибуције - прва или поновна дистрибуција података не утиче на захтеве апликације.
- Заштита података - не сме постојати могућност заобилажења правила интегритета дефинисаних језиком базе података употребом језика који раде на ниском нивоу.

Језик се у почетку звао SEQUEL (Structured English Query Language) и представљао је програмски интерфејс (API) за System P, прототипски систем за управљање базом података (SU8P) који се развијао као део истраживачког пројекта под истим називом. Крајем седамдесетих и почетком осамдесетих година прошлог века јављају се и прве комерцијалне верзије релационих система, са SQL–ом као упитним језиком. Међу њима су најзначајнији Oracle и два IBM–ова производа: SQL/DS и DB2.

SQL је у сталном развоју. На почетку је био прилично једноставан, близак кориснику и у великој мери декларативан (непроцедуралан). Непроцедуралност SQL–а се огледала у томе да се њиме дефинисало ШТА се жели, а не КАКО се добија: који подаци се желе, које табеле се референцирају и који услови треба да буду испуњени, без спецификације процедуре за добијање жељених података.

Данас се за SQL може рећи да је комплексан, процедурално/декларативан језик. SQL ради са табелама. Табела се креира једном извршном наредбом. Одмах по креирању табела је расположива за коришћење. Сви подаци меморисани су у табелама и резултат, било које операције, се логички приказује у облику табеле. Да би се повећала функционалност језика, у SQL: 1999 стандарду, уведена је процедурална надградња SQL–а, коју углавном чине управљачке структуре сличне упављачким структурама класичних програмских језика.

SQL наредбе сврставане су у три категорије (закључно са SQL-92 стандардом):
- наредбе за дефинисање података (Data Definition Statements – DDS),
- наредбе за руковање подацима (Data Manipulation Statements – DMS),
- наредбе управљачких функција (Data Control Statements – DCS).

Наредбе за дефинисање података омогућују дефинисање објеката базе података. Примери наредби ове категорије су:

- CREATE TABLE (креирање табеле базе података)
- CREATE VIEW (креирање виртуелне табеле - "погледа")
- CREATE INDEX (креирање индекса над комбинацијом колона табеле)
- ALTER TABLE (измена дефиниције табеле)
- DROP TABLE (избацивање табеле из базе података)

Наредбе за руковање подацима омогућују ажурирање и приказ података из табела базе података:

- SELECT (приказ садржаја релационе базе података)
- UPDATE (измена вредности колона табеле)
- DELETE (избацивање редова табеле)
- INSERT (додавање редова постојећој табели)

Наредбе за управљачке функције омогућују опоравак, конкурентност, сигурност и интегритет релационе базе података:

- GRANT (додела права коришћења сопствене табеле другим корисницима)
- REVOKE (одузимање права коришћења сопствене табеле од других корисника)
- COMMIT (пренос дејстава трансакције на базу података)
- ROLLBACK (поништавање дејстава трансакције)

6.1.1.1. SQL ТИПОВИ ПОДАТАКА

SQL стандард подржава нумеричке, текстуалне, бинарне, датумске, интервалне и логичке типове података. Ово су основни типови, док системи за рад са базама података имају изведене типове, како би пројектантима олакшали креирање табела базе података.

Типови података су:

- Целобројни:
  - Bit; податак који је 1 или 0
  - int (integer): који износи од -2^{31} (-2,147,483,648) до $2^{31}-1$ (2,147,483,647) сачуван у 4 byte-а
  - smallint: цели број сачуван у 2 byte-а; 2^{15} (-32,768) do $2^{15} - 1$ (32,767)
  - tinyint: податак од 0 – 255
- Децимални:
  - decimal или numeric $-(10)^{38} - 1$ do $10^{38} - 1$. Пример: decimal(15, 3); Прва цифра означава укупан број цифара, а друга број децималних места иза децималне тачке.
- Новац:
  - money: тип податка је исти као и децимал. Разлика је у исписивању. -2^{63} до $2^{63} - 1$.
  - smallmoney: 214,748.3648 do +214,748.3647, код новчаног типа податка подаци се чувају са четири децимална места.
- Прилагодљиви зарез:
  - Float: Floating $-1.79E^{+308}$ до $1.79E^{+308}$.
  - Real: $-3.40E^{+38}$ до $3.40E^{+38}$.

- Датуми:
  - Datetime: 1.Јануар, 1753, до 31.Децембра, 9999 уз тачност од 3.33 милисекунде.
  - Smalldatetime: 1.Јануар, 1900, до 6. Јуна, 2079 уз тачност од 1 минута.

- Низови знакова:
  - char (character): знаковни низ, на пример, char (9) у табели базе података податак ће заузимати 9 знакова без обзира на уношену дужину што значи да може доћи до крађења или допуњавања. Максимално 8000 знакова.
  - nchar (national char): Чувају се знакови који спадају у Unicode. Максималне дужине 4000 знакова.
  - Text: чува текстуалне податке . Може садржати 2,147,483,647 знакова.
  - Varchar: промењива дужина (у табелама базе података се чува тренутна дужина податка), не Уницоде знакова. Максималне дужине 8000 знакова.
  - nvarchar (national char varying): промењива дужина Unicode знакова. Може садржати 4000 знакова.
- Бинарни
  - Binary: бинарни податак максималне дужине 8000 бајтова.
  - Varbinary: бинарни податак промењиве дужине. Максималне дужине 8000 бајтова.
  - Image: бинарни податак промењиве дужине, максималне дужине 2,147,483,647 бајтова.

6.1.1.2. SQL УПИТИ

Основа SQL–а је упитни блок облика:

```
SELECT <lista atributa>
FROM <lista relacija>
WHERE <kvalifikacioni izraz >;
```

Листом атрибута задаје се операција ПРОЈЕКЦИЈЕ. Квалификационим изразом задају се услови СЕЛЕКЦИЈЕ и СПАЈАЊА, односно искази слични исказима у релационом рачуну. Клаузуле SELECT і FROM су обавезне, док клаузула WHERE није.

Упити типа пројекције, приказују неке или све колоне посматране табеле и све њене врсте без икакве селекције, односно филтера.

Синтакса овог типа упита је:

```
SELECT <lista atributa>
FROM <lista relacija>;
```

Када се траже сви атрибути неке релације, уместо навођења сваког атрибута појединачно, могуће је користити знак "*" са истим дејством:

```
SELECT *
FROM <lista relacija> ;
```

Резултат ових упита могу у неким случајевима имати као резултат дупле редове. Да би се то избегло користи се клаузула DISTINCT:

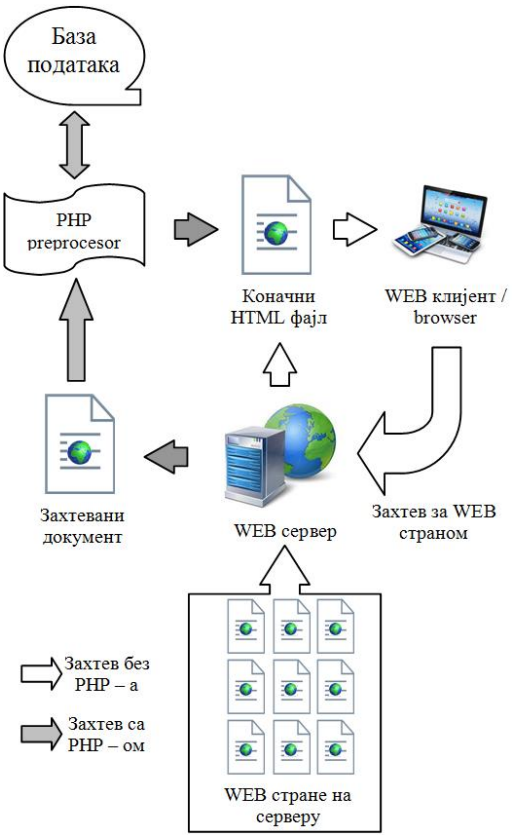
```
SELECT DISTINCT <lista atributa>
FROM <lista relacija>;
```

У релационом моделу могу се дефинисати две врсте нула (NULL) вредности. Прва је нула вредност са значењем – још непозната вредност, а друга је нула вредност са значењем неприменљиво својство, која означава да неки атрибут није применљиво својство за све n-torke дате релације.

SQL:1999 подржава само једну нула вредност. Пројектант и корисник треба да претпоставе да је у питању нула вредност са смислом још непозната вредност. Дали неки атрибут релације има нула вредност – тестира се преко клаузуле IS NULL или IS NOT NULL.

6.1.2. ПРОГРАМСКИ ЈЕЗИК PHP

PHP је open–sorce server–side програмски језик за динамичко генерисање HTML кода. Другим речима, PHP је програмски језик помоћу којег се може креирати HTML страница на серверу пре него што се она пошаље клијенту попуњена динамичким садржајем. Овим начином генерисања садржаја клијент не може видети код (скрипту) који је генерисао садржај који гледа, већ има приступ чистом HTML коду, слика 4.



Слика 4. Извршавање динамичке HTML скрипте од стране PHP програма

PHP је данас једна од најнапреднијих и најкоришћенијих server–side скриптних технологија у употреби. Он је својом синтаксом сличан многим другим сличним језицима, чак и има истозначне (исте по синтакси и функционалности) функције као и неки други језици као што је C. То значи да се једна радња можете извести коришћењем више различитих функција.

PHP је програмски језик на серверској страни и врло моћан алат за креирање динамичких и интерактивних Web страна. PHP програмски код се извршава на серверу и резултат изврашавања је HTML фајл који се шаље Web претраживачу. PHP фајлови имају екстензију .php.

Након пријема захтева са PHP документом сервер извршава PHP код и на основу њега генерише HTML код и шаље га клијенту. То значи да страница која се приказује у претраживачу клијента не постоји у том облику нигде на серверу одакле ју је клијент примио. Ово може створити потешкоће при позиционирању – рангирању креираних страница. Другим речима, PHP је скрипт језик помоћу којег се креира HTML страница на серверу пре него што се она, попуњена динамичким садржајем, пошаље клијенту. Овим начином генерисања садржаја клијент не може видети код (скрипту) који је генерисао садржај који гледа, већ има приступ чистом HTML коду. Као што је већ напоменуто: PHP је server–side скриптни програмски језик за динамичко генерисање HTML koda, где server–side скрипте значи да се оне извршавају на серверу (послужитељу) када сервер прими захтев за PHP документом.

Програм који се напише у PHP-u не захтева превођење (компајлирање), него се интерпретира при сваком извршавању.

PHP интерпретатор може радити:

- по PHP CGI принципу, односно тако што ће интерпретатор постојати као екстерна апликација, која се позива да изврши дату скрипту сваки пут кад буде захтевана од неког корисника,
- а може бити инсталиран и као модул веб–сервиса.

Друга варијанта је данас у највећој употреби јер пружа знатно већу брзину извршавања – интерпретатор је на тај начин увек учитан у меморију па се не мора позивати спољашњи програм.

Уобичајен сценарио по ком се извршавају PHP скрипте је следећи: клијент (корисник Интернета који користи неки интернет прегледач) захтева PHP страницу са сервера, сервер прослеђује захтев сервису за веб (програм веб-сервер на серверу). Веб-сервер препознаје да се тражи PHP датотека не шаље његов садржај клијенту, него га извршава као програм помоћу PHP модула излазни текст програма (стандардни излаз) се шаље клијенту. Као резултат захтева, клијент препознаје врсту резултата (HTML код, слиика, PDF садржај, архива итд.) и резултат се приказује клијенту на одговарајући начин.

За разлику од већине програмских језика који поседују почетну функцију (main у C-u, први блок BEGIN у Paskalu, класа са main методом у Javi ...), PHP датотека, налик на већину скриптних језика, једноставно садржи скуп наредби, које се извршавају једна за другом, од прве до последње, где се посљедња уједно сматра и крајем PHP програма.

6.1.2.1. СИНТАКСА PHP-A

Као што је то већ поменуто, програме писане у програмском језику PHP није потребно преводити у извршни облик (енглески Compile) већ се они изводе приликом покретања у интерпретеру. Програми се пишу као део HTML странице. Сам програм се налази унутар HTML ознаке који почиње са <?php, а завршава се са ?>.

Све унутар ове ознаке се сматра PHP програмом и ако је на серверу инсталиран и исправно подешен PHP интерпретер, а датотека завршава sa .php, тада ће се тај део програма аутоматски извршити.

Постоје (најмање) четири стила за исписивање PHP ознака:

- XML стил уз помоћ већ поменутих ознака <?php и ?> и такав стил се препоручује, а обавезан је када је PHP код усађен у HTML код. У специјалном случају дозвољава се да посљедња ознака ?> у задњој PHP скрипти не буде наведена.
- скраћени стил између ознака <? и ?>. Може да се користи у колико се у конфигурацискоj датотеци пхп.ини активирана опција short_open_tag, али се његова употреба не препоручује зато јер у неким окружењима није подржан. Међутим, од верзије 5.4, скраћени стил у облику <?= ... ?> је омогућен без обзира на подешавања унутар конфигурацијске датотеке, а служи као замена за <?php ... ?>.
- скрипт стил између ознака <script language="php"> и </script>, је дужа верзија XML стила и обично се препоручује када се истовремено користе и други скриптни језици, посебно ако HTML едитор прави проблеме.
- ASP стил између ознака <% и %> представља још једну скраћену верзију. Најчешће се користи у ASP и ASP.NET окружењу, али је неопходно претходно активирати опцију short_open_tag у конфигурационој датотеци php.ini.

PHP као и сваки други програмски језик, користи властити, ограничени скуп речи које имају посебна значења. Комбинивањем тих речи по унапред задатим правилима пишу се нови програми. У ствари овде се ради о такозваним резервисаним, односно кључним речима (енглески: keywords), табела Т–1.

Изрази (expressions) су један од основних појмова сваког вишег програмског језика, па и PHP-а. Израз рачуна и даје резултат или узрокује неку активност (која се понекад назива бочни ефекат) и они представљају комбинацију оператора и операнда. Једноставније речено, израз је све оно што има неку вредност. Вредност неког израза одређује се на основу синтаксе израза, односно правила првенства сваког од оператора у изразу.

Табела 1. Кључне – резервисане речи у PHP–у

PHP Кључне речи				
hallt_compiler()	abstract	and	array()	as
break	callable (as of PHP 5.4)	case	catch	class
clone	const	continue	declare	default
die()	do	echo	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
final	finally (as of PHP 5.5)	for	foreach	function
global	goto (as of PHP 5.3)	if	implements	include
include_once	instanceof	insteadof (as of PHP 5.4)	interface	isset()
list()	namespace (as of PHP 5.3)	new	or	print
private	protected	public	require	require_once
return	static	switch	throw	trait (as of PHP 5.4)
try	unset()	use	var	while
xor	yield (as of PHP 5.5)			

Наредбе (statements) или искази одређују неку акцију. Свака PHP скрипта је састављена од низа наредби. По правилу наредбе се извршавају оним редом којим су наведени у програму, али овај редослед може да се промени неком од наредби за пренос тока програма, при чему пренос може ићи на било који други део програма или ван њега.

Основни облик наредбе је израз иза кога следи знак тачка са зарезом (;), као завршни знак наредбе. Сложена наредба или блок наредба је механизам који групише скуп наредби у једну семантичку целину. Блокови могу да буду уклопљени и сваки може да садржи своје сопствене декларације, иницијализације и извршне наредбе.

Оператори су симболи који омогућавају извршавање операција над вредностима и променљивама. Могу имати један, два или више аргумената. Параметри, односно аргументи се још називају и улазни подаци, а сам резултат функције њеним излазним податком.

Управљачке структуре представљају наредбе којима се одређује ток извршавања под одређеним условима и могу да се групишу у једну засебну структуру. Оваква структура се поставља између витичастих заграда: { и }. Управљачке структуре, саме за себе, представљају једну наредбу у облику блока. Користе се различити облици управљачких (контролних) структура (control structure), које се међусобно разликују по начину испитивања услова.

6.1.2.2.PHP ПРОМЕНЉИВЕ

Променљиве су величине које током извршавања PHP програма могу мењати своју вредност. У PHP– у ознаке за променљиве морају почети са знаком долар $. Ово је чисто начин говорења PHP преводиоцу да се ради о променљивој, а не о тексту. Уколико се изостави апликација ће јавити грешку (у најбољем случају), а прећи ће преко ње (у најгорем случају) и уместо садржаја променљиве ће исписати само њено име.

Иза знака долар назив мора почети словом, а послије тога могу се писати бројеви, слова или неки други знакови али не и размак. PHP разликује велика и мала слова у називу промјенљиве, имена променљивих су case-sensitive. Ово правило илустровано је следећим примерим: $tekst, $TEKST, $Tekst, $tekST, овде су у ствари наведене 4 различите променљиве. Исто тако, у именима променљивих не смеју се користити размаци, као ни било који други специјални карактери осим [и] који се користе у низовима и код неких метода рада са стринговима.

Приликом употребе променљивих у програмском језику PHP није потребно декларисати променљиву прије њеног коришћења, као што је то случај у другим програмским језицима. Због тога је писање програма нешто једноставнији, али и лакше долази до већих проблема јер у случају да се погрешно напише име променљиве или се погрешно упишу велика и мала слова PHP ће наставити извршавати програм, а као вредност погрешно написане променљиве узимат ће се њена подразумевана вредност. Такве грешке је тешко уочити, али их PHP може исписивати подешавањем вредности поставке E_WARNING у конфигурацијској датотеци.

6.2. MYSQL БАЗА ПОДАТАКА

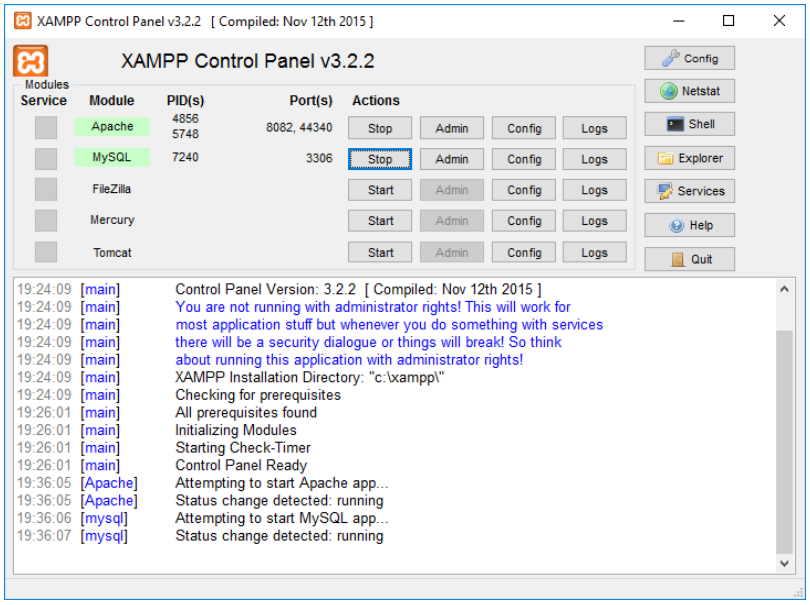
За потребе овог мастер рада, формирана је база података у софтверу XAMPP. XAMPP, слика 5., је потпуно бесплатан open source серверски пакет за једноставну инталацију Apache сервера на рачунаре са оперативним системом Windows, Linux или OSX (зато је прво слово у називу X). Xampp је намењем за употребу у локалној мрежи не као web сервер и служи пре свега програмерима да креирају сервер ради тестирања својих скрипти, односно MySQL база података у off-line режиму.

Xampp припада фамилији WAMP софтверких пакета, где је WAMP скраћеница од Windows, Apache, MySQL, а P се може односити на PHP, Python или Perl.

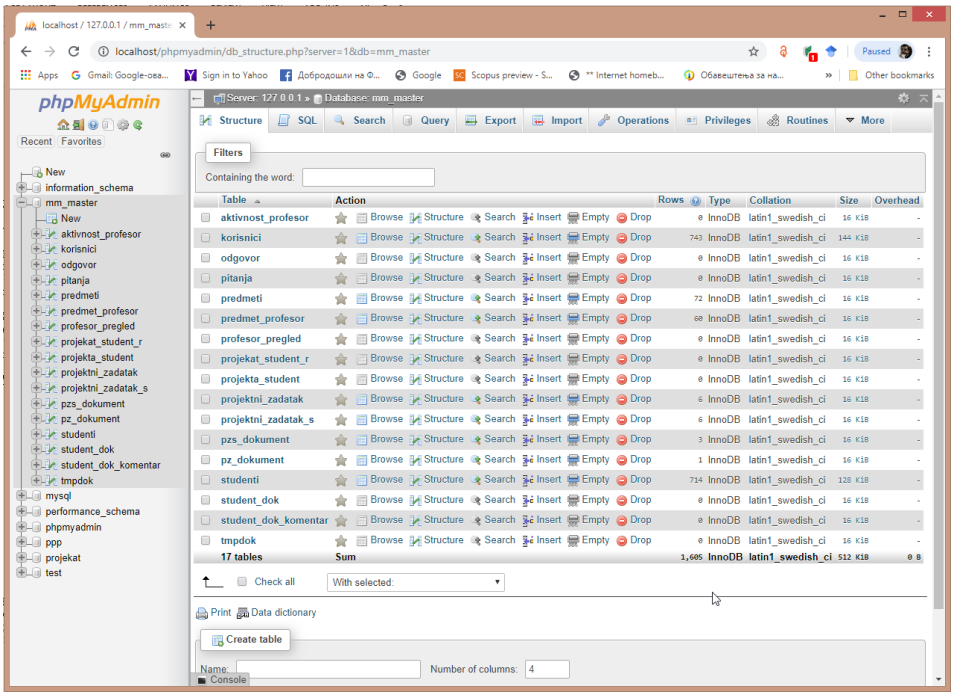
Након инсталације и покретања програма, потребно је покренути Apache сервер, односно MySQL сервер. На тај начин, омогућава се рад се сервером базе података. Кликом на командно дугме Admin, слика 5., у реду који означава MySQL сервер базе података, отвара се радно окружење MySQL базе података, у овом случају на локалном серверу, слика 6.

Употребом алата, које нуди phpMyAdmin, формирана је база података под називом mm_master. Ова база података је намењена да чува матичне податке о професорима и студентима, као и да омогући дефинисање пројектних задатака, семинарских и домаћих радова за сваког студента. Ова база података је комплетно трансформисана на сервер базе података *byethost.com*.

На слици 7., је приказан део релационе повезане базе података са табелама које су коришћене у овом мастер раду, а у табели 2., је приказана структура сваке од приказаних табела.



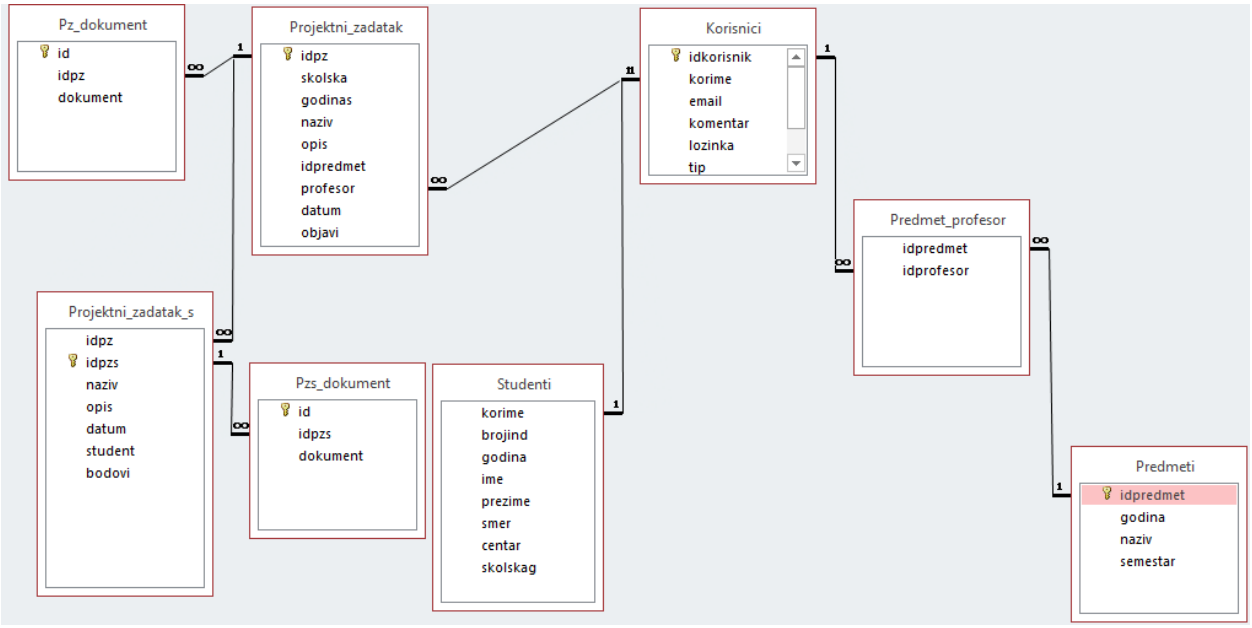
Слика 5. Контролна табла Xampp – а



Слика 6. Контролна табла Xampp – а

Табеле су релационо повезане преко примарних и страних кључева. Везе између табела, као што се види са слике 7., су махом 1 : N, осим у случају везе табела *korisnici* и *studenti*, где је веза 1:1.

Табела *korisnici* служи за чување података о свим корисницима који се пријављују на сајт – студенти и професори. Приликом пријављивања, у зависности од корисничког имена, софтвер проверава поље *tip*, ако је у пољу тип S (што означава студента), покреће се апликација за информисање студента о обавезама за предмете и професоре који су те обавезе унели.



Слика 7. Релационо повезане табеле базе података mm_master

Табела *studenti* , поред основних података о студенту, садржи и податке о тренутној години студента, смеру који је студент уписао, као и податак о актуелној школској години. Атрибут *korime*, табела 2., је страни кључ који референцира ка табели *korisnici*, у односу 1:1.

Сценарио за ажурирање базе података је следећи:

1. професору (мора да буде пријављен у табели *korisnici*), се додељују предмети које предаје у текућој школској години (*Profesori (1) – Predmet_profesor – Predmeti (N)*, у табели *predmet_profesor* су страни кључеви (2) који референцира ка табели *Profesori* и (3) који референцира ка табели *Predmeti*, табела 2.) .

2. Професор формира пројектни задатак (ажурира се табела *Projektni_zadatak* у којој су два страна кључа (4) који референцира ка табели *Predmeti*) и који референцира ка табели *profesori* (5), табела 2. Сваки пројектни задатак може, а не мора да има пратећа документа (0,..,N) и они се ажурирају у табели *Pz_dokument*, у којој страни кључ (8) референцира ка табели *Projektni_zadatak.*

3. Уколико сваки студент има посебне обавезе у оквиру пројекта, професор може да дефинише те обавезе, што се ажурира у табели *Projektni_zadatak_s*, која има два страна кључа (6) који референцира ка табели *Projektni_zadatak* и (7) који референцира ка табели *Studenti*. Наравно, и у овом случају је могуће доделити одговарајућа документа (0,..,N) и они се ажурирају у табели *Pzs_dokument*, у којој страни кључ (9) референцира ка табели *Projektni_zadatak_s.*

Табела 2. Структура табела базе података mm_master

Табела	Атрибут	Тип податка	Дужина	Индекс	Врста индекса
Korisnici	idkorisnik	integer	5 (Auto_Incemrnet)	ДА	Примарни кључ
	korime	varchar	15	ДА	Без понављања
	email	varchar	50	ДА	Без понављања
	komentar	text			
	lozinka	varchar	15	ДА	Без понављања
	tip	varchar	1		
	ime	varchar	15		
	prezime	varchar	30		
Studenti	korime	varchar	15	ДА	Страни кључ[1)]
	brojind	varchar	8	ДА	Без понављања
	godina	varchar	3		
	ime	varchar	15		
	prezime	varchar	30		
	smer	varchar	150		
	centar	varchar	2		
	skolskag	varchar	7		
Predmeti	idpredmet	int	5	ДА	Примарни кључ
	godina	int	1		
	naziv	varchar	50		
	semestar	int	1		
Predmet_profesor	idpredmet	int	5	ДА	Страни кључ[2)]
	idprofesor	int	5	ДА	Страни кључ[3)]
Projektni_zadatak	idpz	int	5	ДА	Примарни кључ
	skolska	varchar	7	ДА	Без понављања

	godinas	int	5		
	ime	varchar	15		
	naziv	varchar	70		
	opis	text			
	idpredmet	int	5	ДА	Страни кључ[4)]
	skolskag	varchar	7		
	profesor	varchar	15		Страни кључ[5)]
	datum	date			
Projektni_zadatak_s	idpz	int	5	ДА	Страни кључ[6)]
	idpzs	int	5	ДА	Примарни кључ
	naziv	varchar	30		
	semestar	int	1		
	opis	text			
	datum	date			
	student	varchar	15		Страни кључ[7)]
	bodovi	int	3		
Pz_dokument	id	int	5	ДА	Примарни кључ
	idpz	int	5	ДА	Страни кључ[8)]
	dokument	varchar	150		
Pzs_dokument	id	int	5	ДА	Примарни кључ
	idpzs	int	5	ДА	Страни кључ[9)]
	dokument	varchar	150		

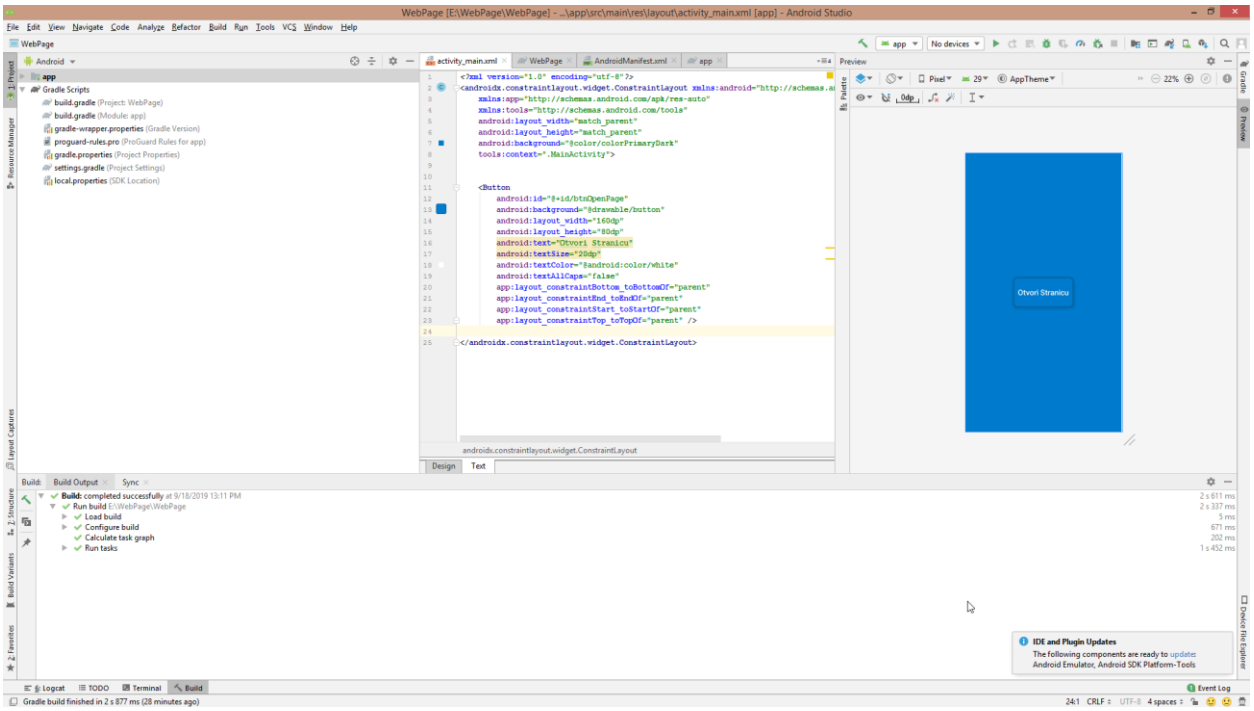
6.3. РЕАЛИЗАЦИЈА СОФТВЕРА

6.3.1. ANDROID ИНТЕРФЕЈС ЗА РАЗВОЈ И ИМПЛЕМЕНТАЦИЈА СИСТЕМА ЗА СИНХРОНИЗАЦИЈУ МОБИЛНИХ АПЛИКАЦИЈА НА SQL СЕРВЕРУ

За проверу система за синхронизацију мобилних апликација на SQL серверу, направљен је софтвер коришћењем *Android Studio 3*.

Android Studio представља интегрсано развојно окружење за развој Андроид апликација [10]. Састоји се од низа алата који служе за развој мобилног програмског производа, укључујући алате за дизајн корисничког окружења, писање програмског кода, тестирање и отклањање грешака, комуникацију са мобилним уређајем, припрему програмског производа, припрему апликације за објављивљње и слично.

Основни изглед прозора за рад у *Android Studio* дат је на слици 8. Прозор је подељен у четири основна дела. У врху се налази трака с алатима, с леве стране је панел структуре пројекта, десно се налази радна површина и такозвани *preview* прозор, а доле су у основном погледу панели за надгледање статуса уређаја и надгледање порука о догађајима.



Слика 8. Основно окружење Android Studio развојног окружења

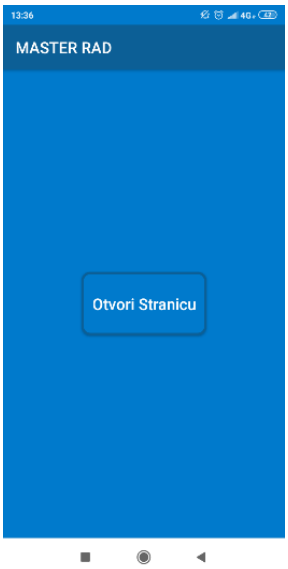
Развој апликација за Андроид је вођен захтевом за одвајањем корисничког окружења од програмске логике. Тај захтев доводи до тога да се у фази развоја програмског производа користе различити језици за опис споменутих апликативних слојева: Јава за опис програмске логике и XML за опис и дизајн окружења.

Користећи панел *Project* и поглед *Android*, слика 9., који служе за приказ структуре пројектних докумената, посматраних као елементе *Android* апликације, можеме се уочити *Java* класе смештене у *java* чвор пројектног стабла, а сви описи ресурса су смештени у *res* чвор пројектног стабла.

Осим наведених у *manifest* чвору је смештен основни апликацијски код - *Manifest*, који дефинише све захтеве апликације. У потпуно одвојеној целини под називом *Gradle Scripts* су смештене све конфигурацијске скрипте потребне за *Gradle* – аутоматски процес који у реалном времену анализира и по потреби компајлира програмски код.

Анализом садржаја *java* чвора, уочава се да су све програмске класе смештене у основни апликацијски пакет, као и два помоћна пакета који служе за тестирање програмског производа.

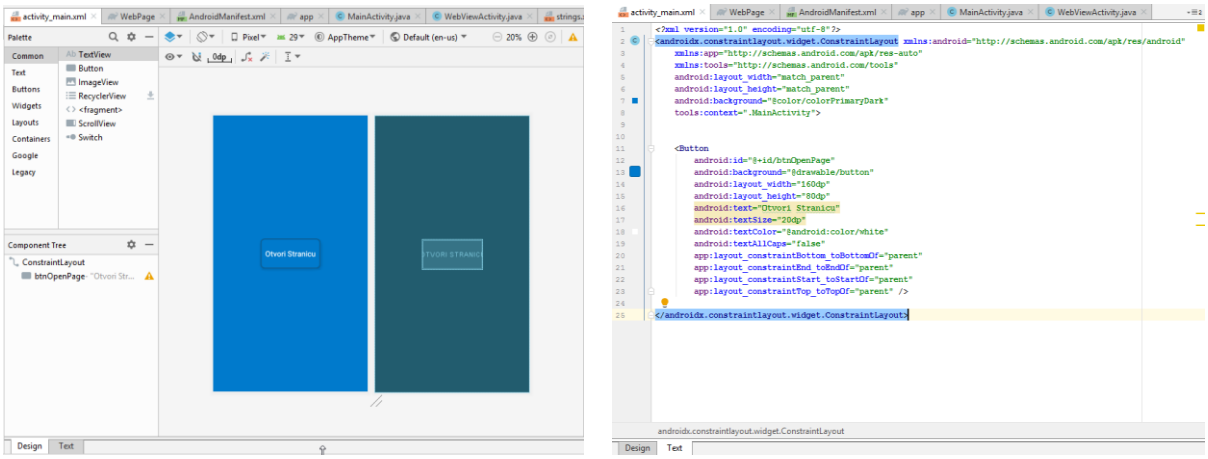
Идеја за формирање интерфејса је да се направи апликација која ће да се инсталира на неки од мобилних уређаја (телефон, таблет), а основна му је сврха да се конектује на жељени сервер базе података. Циљ је да сам интерфејс буде веома једноставан и прилагођен за рад на мобилним уређајима, слици 9. Као што се види са слике 9., интерфејс је веома једноставан и састоји се само од једног командног дугмета (*Otvori Stranicu*).



Слика 9. Апликација за повезивање на сервер базе података

Када се формира нови пројекат, *Android Studio*, аутоматски формира кодове везане за активност (*Activity*) и манифест (*Manifest*). Активност представља низ наредби који се извршавају у апликативном контексту, и у оквиру овог дела андроид пројекта, могуће је приказати, у посебном прозору, изглед корисничког окружења андроид апликације, слика 10. Aktivnost, у ствари представља корисничку функционалност над програмском апликацијом, и у оквиру ње је могуће контролисати (додавање, измене, брисање) све елементе које ће да садржи корисничко окружење.

Са слике 10., може се уочити да постоје две активности: *acivity_main.xml* и *MainActivity.java*. Прва наведена процедура у ствари дефинише кориснички интерфејс, А-01, док друга је програмски код којим се извршавају команде дефинисане у оквиру корисничког интерфејса, А-02. Из програмског кода А – 02., *MainActivity* наслеђује класу *AppCompatActivity* (подкласа класе *Activity* из *Android* развојног окружења) која омогућује компатибилност са свим верзијама оперативног система које су означене као циљне верзије при креирању пројекта



Слика 10. Activity процедура

.

A–01. *activity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimaryDark"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/btnOpenPage"
        android:background="@drawable/button"
        android:layout_width="160dp"
        android:layout_height="80dp"
        android:text="Otvori Stranicu"
        android:textSize="20dp"
        android:textColor="@android:color/white"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />
 </androidx.constraintlayout.widget.ConstraintLayout>
```

A–02. *MainActivity.java*

```
package com.template.webpage;
import android.content.Intent;xmlns:android="http://schemas.android.com/apk/res/android"
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       Button btnOpenPage = findViewById(R.id.btnOpenPage);
       btnOpenPage.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               startActivity(new
               Intent(MainActivity.this,WebViewActivity.class));
           }
       });
    }
 }
```

Унутар класе постоји имплементација само једне методе, која је у овом случају надјачање методе *onCreate* из надкласе. Уважавајући конвенцију писања назива метода у Андроиду, свака метода која одговара на неки догађај има префикс *on*. Тако да метода *onCreate* одговара на догађај креирања активности, па се извршава одмах након што је активност креирана.

Програмски код А–02., у начелу говори да када се креира активност, аутоматски јој се додели дизајн графичког интерфејса из *R.layout.activity_main*, линија 13.

Програмски код дат у А–02, одмах при креирању активности прихвата дентификатор ресурса који дефинише изглед корисничког окружења, па га прослеђује методи *setContentView* која га приказује кориснику, линија 13.

Кључну улогу у повезивању ресурса и програмског кода има аутоматски генерисана класа *R*, која садржи јединствене идентификаторе свих ресурса дефинисаних у склопу пројекта. Ресурси у тој класи су статички и хијерархијски су организовани складно са мапама у које су физички смештени. Сваки пута, када корисник промени ресурсе дефинисане у *XML*–у, ако је резултирајући *XML* валидан, *Android Studio* ће аутоматски освежити класу *R* која се одмах потом може користити и у програмском коду.

У линији 14, у коду А–02., дефинисано је командно дугме, слика 9., која покреће апликацију којом се корисник, власник мобилног уређаја, повезује на сервер базе података. Догађај који се покреће кликом на командно дугме *Otvori stranu*, је дефинисан у линији 16. И овде важи иста конвенција као за догађај *Create*, тако да се формира догађај *onCreate*, који се извршава када корисник кликне на командно дугме. Процедура која се извршава дата је у програмском коду А–03, а базира се на уграђеној класи *WebViewActivity*.

А–03. *WebViewActivity.java*

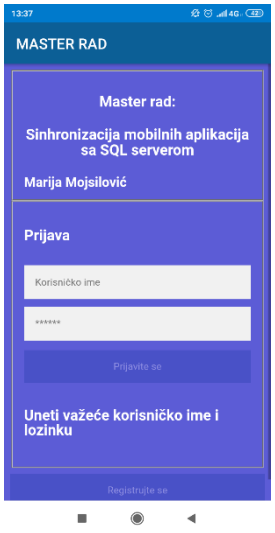
```
 1  package com.template.webpage;
 2  import android.annotation.SuppressLint;
 3  import android.os.Bundle;
 4  import android.webkit.WebChromeClient;
 5  import android.webkit.WebView;;
 6  import android.webkit.WebViewClient;
 7  import androidx.appcompat.app.AppCompatActivity;
 8  public class WebViewActivity extends AppCompatActivity {
11     @SuppressLint("SetJavaScriptEnabled")
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14        super.onCreate(savedInstanceState);
15        setContentView(R.layout.activity_web_view);
16        WebView myWebView = findViewById(R.id.myWebView);
17        myWebView.setWebViewClient(new WebViewClient());
20        myWebView.setWebChromeClient(new WebChromeClient());
21        myWebView.getSettings().setJavaScriptEnabled(true);
22        myWebView.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
23        myWebView.getSettings().setAllowFileAccessFromFileURLs(true);
24        myWebView.getSettings().setAllowUniversalAccessFromFileURLs(true);
25        myWebView.getSettings().setLoadWithOverviewMode(true);
26        myWebView.getSettings().setUseWideViewPort(true);
27        myWebView.getSettings().setBuiltInZoomControls(true);
28        myWebView.getSettings().setDisplayZoomControls(false);
29        myWebView.getSettings().setSupportZoom(true);
30        myWebView.setScrollBarStyle(WebView.SCROLLBARS_OUTSIDE_OVERLAY);
31        myWebView.setScrollbarFadingEnabled(false);
32        myWebView.loadUrl("http://mastermm.byethost31.com");
33        }
34  }
```

На основу класе *WebViewActivity*, уграђени *web* прегледач (*web browser*) преузима садржај са странице где се налази сервер базе података, линија 32, код А–03. У поменутом програмском коду формира се променљива *myWebView*, а на основу уграђене класе *WebView*, која се преузима из формиране класе корисничког окружења *R*, што је претходно и објашњено. У линијама од 17 – 31 се подешавају параметри за рад у локалном *web* прегледачу.

Овиме је практично завршен софтвер који је намењен за инсталацију на мобилним уређајима. Следећи корак је формирање апликације која треба да буде прилагођена било ком мобилном уређају, а којом ће се корисник улоговати, односно повезати на сервер базе података у циљу преузимања или ажурирања података. Ова апликација је формирана коришћењем *HTML* и *PHP* алата. *HTML* алат се користи за формирање прилагодљивог корисничког интерфејса и контролу над објектима, док се *PHP* алат користио за конекцију са базом података и програмирање акција везаних за ажурирање и преглед података са сервера.

6.3.2. АПЛИКАЦИЈА ЗА СИНХРОНИЗАЦИЈУ ПОДАТАКА НА SQL СЕРВЕРУ

Почетни екран серверске апликације дат је на слици 11. Овај екран добија се кликом на командно дугме *Otvori stranicu*, слика 10., на мобилном уређају. Овде је важно напоменути да се корисничко окружење прилагођава величини екрана мобилног уређаја, 4 линија, код А–04.



Слика 11. Почетни екран апликације - форма за пријављивање на сервер базе података

Слика 12. Форма за пријављивање корисника на сервер базе података

А–04. Део HTML код за дефинисање корисничког окружења апликације

```
1.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cz">
2.      <head>
3.          <meta charset="UTF-8">
4.          <meta name="viewport" content="width=device-width">
5.          <link rel="stylesheet" type="text/css" href="definicije.css" title="default">
6.          <title>Master rad</title>
7.      </head>
8.      <style>
            ...
9.      </style>
10.     <body>
11.         <form role = "form" action = "<?php echo htmlspecialchars($_SERVER['PHP_SELF'])?>" method = "post">
12.             <table class="tabela">
```

```
13.                     <col width="100%"/>
14.                     <tr>
15.                         <fieldset>
                            ...
17.                         </fieldset>
18.                     </tr>
19.                     <tr>
20.                         <fieldset>
                            ...
21.                         </fieldset>
22.                         <div>
23.                             <button class = "dugme" type = "submit" name = "prijava">Registrujte se</button>
24.                         </div>
25.                     </tr>
26.                 </table>
27.             </form>
28.         </body>
29.     </html>
```

Сама форма је у ствари представља табелу која је дефинисана као наследник класе *table*, код А–05., а уграђена је у коду А–04. (између линија 8 <style> и 9 </style>) , дефинисањем објекта *tabela* (линије од 8 до 36, код А–05). У оквиру дефинисања стила (8 <style> и 9 </style>) дефинисан је основни изглед апликације *body* (линије од 2 до 7, код А–05).

А–05. Део HTML код за објекта *tabela* и основног изгледа апликације *body*

```
1.      <style>
2.          body{
3.              background-color: #5c5cd6;
4.              font-family: sans-serif;
5.              font-size: 1.25rem;
6.              line-height: 100%;
7.          }
8.          tabela{
9.              border-collapse: colla;
10.             margin: 25px 0;
11.             font-size: 0.9em;
12.             min-width: 400px;
13.             border-radius: 5px 5px 0 0;
14.             overflow: hidden;
15.             box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
16.         }
17.         .tabela thead tr {
18.             background-color: #009879;
19.             color: #ffffff;
20.             text-align: left;
21.             font-weight: bold;
22.         }
23.         .tabela th, .tabela td {
24.             padding: 12px 15px;
25.             height:105px;
26.         }
27.             .tabela tbody tr:nth-of-type(even) {
28.             background-color: #f3f3f3;
29.         }
30.         .tabela tbody tr:last-of-type {
31.             border-bottom: 2px solid #009879;
32.         }
33.         .tabela tbody tr.active-row {
34.             font-weight: bold;
35.             color: #009879;
36.         }
```

```
37.  </style>
```

Кориснику, апликација нуди две могућности. Да унесе корисничко име и лозинку и кликом на дугме *Prijavite se*, да се повеже на сервер базе података, или ако корисник није пријављен може се пријавити кликом на дугме *Registrujte se* (линија 23, А–04), слика 11. Програмски код којим се покреће форму за регистрацију корисника, слика 12, је дат у А–06.

А–06. PHP код за покретање форме за регистрацију корисника

```
1.  <?php
2.      if($prijava=='DA')
3.      {
4.          echo "<script>location.href='indexPrijava.php'</script>";
5.      }
6.      else{
7.          $prijava=='NE';
8.      }
9.      echo "<br/>";
10. ?>
```

У циљу повезивања корисника на сервер базе података формирана је PHP процедура, чији код је дат у А–07. Овом процедуром се проверава да ли је успешно обављена конекција на сервер, линија 7., односно да ли је извршено успешно повезивање на базу података *mm_master*, линија 9.

А–07. PHP код за повезивање на сервер базе података.

```
1.  <?php
2.      $bp="mm_master";
3.      $korisnik=$_POST["korisnik"];
4.      $lozinka=$_POST["lozinka"];
5.      $host="localhost";
6.      $veza=mysqli_connect("localhost","root","");
7.      if($veza)
8.      {
9.          if(mysqli_select_db($veza,"mm_master"))
10.         {
11.             $upit=mysqli_query($veza,"select idkorisnik, email,lozinka,korime,tip, ime, prezime from korisnici
                where korime='".$korime."'");
12.             $niz=mysqli_fetch_array($upit);
13.             if(($niz['korime']==$korisnik) and( $niz['lozinka']==$lozinka))
14.             {
15.                 echo "Uspesna prijava!!!"
16.             }
17.             else
18.             {
19.                 echo "Neuspesna prijava!!!"
20.             }
21.             }
22.             else
23.             {
24.                 echo"Baza podataka 'projekat' ne postoji!";
25.             }
26.         }
27.     else
28.     {
29.         echo "NEUSPELO POVEZIVANJE!";
30.     }
31. ?>
```

Након попуњавања форме за регистрацију, слика 12, корисник кликом на дугме *Sačuvati podatke*, повезује се на сервер базе података и попуњавају се унети подаци, линија 23., у табелу *korisnici*, код А–08. У случају да не жели да се региструје кликом на дугме *Odustati*, враћа се на почетну форму.

А–08. PHP код за додавање - регистрацију новог корисника.

```
1.      <?php
2.          include("PoveziSe.php");
3.          if(isset($_POST['registracija']))
4.          {
5.              $veza=PoveziSe();
6.              $_porukaBre=-1;
7.              $_korime=$_POST['korime'];
8.              $ime=$_POST['ime'];
9.              $prezime=$_POST['prezime'];
10.             $lozinka=$_POST['lozinka'];
11.             $tip=$_POST['tip'];
12.             $email=$_POST['e_mail'];
13.             $plozinka=$_POST['plozinka'];
14.             $upit1="SELECT korime FROM korisnici WHERE korime='$_korime';";
15.             $rez=mysqli_query($veza,$upit1);
16.             if(mysqli_num_rows($rez)==0)
17.             {
18.                 if(!empty($_korime) && !empty($ime) && !empty($prezime) && !empty($lozinka) && !empty($tip) &&
                    empty($email) && !empty($plozinka))
19.                 {
20.                     $faliunos=0;
21.                     if($lozinka==$plozinka)
22.                     {
23.                         $upit="INSERT INTO korisnici (korime,email,lozinka,tip,ime,prezime) VALUES
                            ('$_korime','$email','$lozinka','$tip','$ime', '$prezime');";
24.                         if(mysqli_query($veza,$upit))
25.                         {
26.                         $faliunos=3;/*USPESNO DODAT KORISNIK*/
27.                         }
28.                         else
29.                         {
30.                         $faliunos=4;/*KONEKCIJA SA SERVEROM BAZE PODATAKA NIJE USPELA*/
31.                         }
32.                     }
33.                     else
34.                     {
35.                         $faliunos=2;
36.                     }
37.                 }
38.                 else
39.                 {
40.                       $faliunos=1;
41.                 }
42.             }
43.             else
44.             {
45.                   $faliunos=5;
46.             }
47.             mysqli_close($veza);
48.         }
49.     ?>
```

У линији 14, кода А–08., проверава се да ли у табели *korisnici* постоји унето корисничко име, у случају да постоји, генерише се порука да је постоји то корисничко име и да је потребно унети ново. Уколико су сви тражени подаци унети (провера у линији 18.) упитом датим у линији 23., додају се

подаци у табели *korisnici*. Након ове акције, кликом на *Povratak na log in formu*, корисник се враћа на почетни екран за пријављивање на сервер.

Да би се корисник успешно пријавио мора се формирати упит којим се проверава да ли унето корисничко име постоји у табели *korisnici*, код А–09. У ту сврху формиран је упит којим се проверава да ли постоје редови - ред, линија 11., са у табели *korisnici*, са примарним кључем *korime*, чија је вредност унето корисничко име. Уколико корисник не постоји програм генерише поруку о погрешно унетом корисничком имену или лозиинци.

Када се корисник успешно пријави, добија се форма за избор пројеката са детаљима, слика 13. На овој слици се уочава објекат *IZABRATI ZADATAK*, који у ствари представља листу са свим пројектним и домаћим задацима, односно обавештењима који су додељени пријављеном кориснику. Листа *IZABRATI ZADATAK*, се попуњава помоћу функције *Popuni_Zadatke*, А–10.

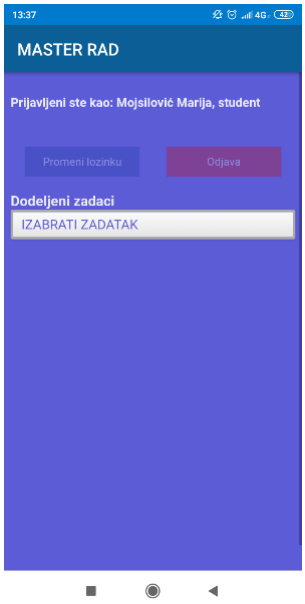
А–09. Код за пријаву на сервер базе података.

```
1.  <?php
2.      include("PoveziSe.php");
3.      $veza=PoveziSe();
4.      $msg = '';
5.      $idi='';
6.      $prijava='NE';
7.      if (isset($_POST['login']) && !empty($_POST['korime']) && !empty($_POST['lozinka']))
8.      {
9.          $korime=$_POST['korime'];
10.         $lozinka=$_POST['lozinka'];
11.         $upit=mysqli_query($veza,"select idkorisnik, email, lozinka,korime,tip, ime, prezime from korisnici where
            korime='".$korime."'");
12.         $niz=mysqli_fetch_array($upit);
13.         if(($niz['korime']==$korime) and( $niz['lozinka']==$lozinka))
14.         {
15.             $_SESSION['valid'] = true;
16.             $_SESSION['timeout'] = time();
17.             $kor_ime=$niz['korime'];
18.             $_SESSION["kor_ime"] = $kor_ime;
19.             $_SESSION["id_kor"] = $niz['idkorisnik'];
20.             $_SESSION["tipPS"] = $niz['tip'];
21.             $tip_PS=$niz['tip'];
22.             $_SESSION["imeprez"]=$niz['prezime']." ". $niz['ime'];
23.             $msg = 'Ispravno uneto korisni&ccaron;ko ime i lozinka' . '<br>' . ' korisnik: '  . $niz['korime'];
24.             $idi='DA';
25.             $prijava='NE';
26.         }
27.         else
28.         {
29.             $msg = 'Pogre&scaron;no uneto korisni&ccaron;ko ime i/ili lozinka';
30.             $idi='NE';
31.         }
32.         mysqli_free_result($upit);
33.         mysqli_close($veza);
34.     }
35.     if (isset($_POST['prijava']))
36.     {
37.         $prijava='DA';
38.     }
39.     else
40.     {
41.         $prijava='NE';
42.     }
43. ?>
```

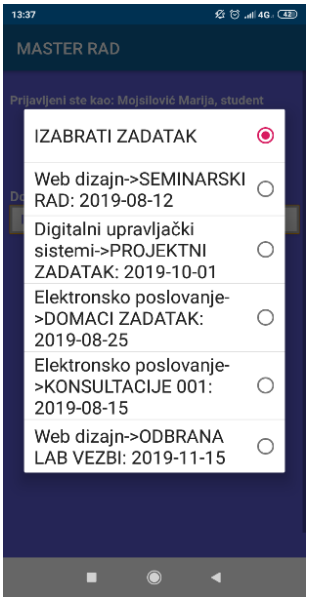
А–10. Код за попуњавање листе обавеза за пријављеног корисника.

```
1.  function Popuni_Zadatke($izabranZ,$korime)
2.  {
3.      $veza=PoveziSe();
4.      $stavka='';
5.      $upit=mysqli_query($veza,"select a.idpzs as 'idbs', a.idpz, b.naziv as 'nazivs', a.naziv as 'nazivp', a.datum as 'datums',
        c.naziv as 'predmetn' from projektni_zadatak_s a INNER JOIN projektni_zadatak b ON a.idpz=b.idpz INNER JOIN
        predmeti c ON b.idpredmet=c.idpredmet where a.student ='$korime'");
6.      while($red = mysqli_fetch_array($upit))
7.      {
8.          $_vred=$red['predmetn'].'->'.$red['nazivs'].': '.$red['datums'];
9.          $_kljuc=$red['idbs'];
10.         if($izabranZ==$_kljuc)
11.         {
12.             $stavka .= '<option value="' . $_kljuc . '" selected       >' . $_vred . '</option>';
13.         }
14.         else
15.         {
16.             $stavka .= '<option value="' . $_kljuc . '">' . $_vred . '</option>';
17.         }
18.     }
19.     mysqli_free_result($upit);
20.     mysqli_close($veza);
21.     return $stavka;
22. }
```

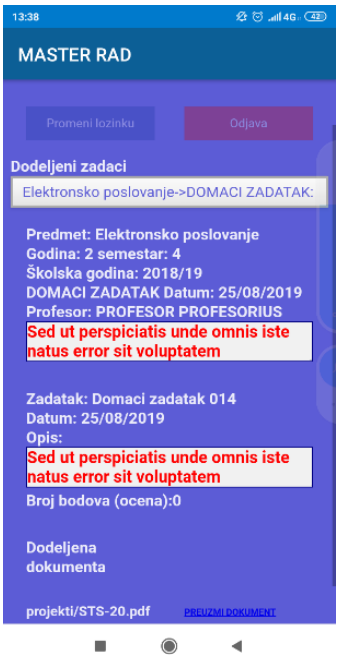
Упитом, дефинисаним у линији 5., се прикупљају подаци из табеле *projektni_zadatak*, а на основу унетог корисничког имена. У упиту су повезане 3 табеле *projektni_zadatak_s*, *projektni_zadatak* и *predmeti*, на основу успостављених релационих веза у бази података, слика 7.



Слика 13. Форма за пријављивање избор пристиглих обавеза



Слика 14. Листа пристиглих обавеза



Слика 15. Детаљи изабране обавезе

Кликом на изабрати задатак, добија се листа са унетим обавезама за пријављеног корисника, слика 14. Кликом на жељено обавештење, добија се форма са свим потребним детаљима, слика 15. Код за прикупљање података из одговарајућих табела базе података, приказан је у А–11.

А–11. Код за попуњавање листе обавеза за пријављеног корисника.

```
1.  $prikazi=0;
2.  $veza=PoveziSe();
3.  $upit1="SELECT idpz, naziv, opis, datum, bodovi FROM projektni_zadatak_s WHERE idpzs=$izabranZ;";
4.  $rez1=mysqli_query($veza,$upit1);
5.  if(mysqli_num_rows($rez1)!=0)
6.  {
7.      while($red1 = mysqli_fetch_array($rez1))
8.      {
9.          $n_zadatka=$red1['naziv'];
10.         $o_zadatka=$red1['opis'];
11.         $d_zadatka=$red1['datum'];
12.         $_DatumZ=substr($red1['datum'],8,2).'/'.substr($red1['datum'],5,2).'/'.substr($red1['datum'],0,4);
13.         $b_zadatka=$red1['bodovi'];
14.         $_idpz=$red1['idpz'];
15.         $upit=mysqli_query($veza,"select a.skolska as 'skolska', a.godinas as 'godinas', a.naziv as 'pnaziv', a.opis as
            'popis', a.idpredmet as 'predmet', a.profesor as 'profesor', a.datum as 'pdatum', b.naziv as 'npredmet', b.godina
            as 'gpredmet', b.semestar as 'spredmet', c.ime as 'ime', c.prezime as 'prezime' from projektni_zadatak a INNER
            JOIN predmeti b ON a.idpredmet=b.idpredmet INNER JOIN korisnici c ON a.profesor=c.korime where a.idpz
            =$_idpz;");
16.         while($red = mysqli_fetch_array($upit))
17.         {
18.             $prikazi=$prikazi+1;
19.             $_skolska=$red['skolska'];
20.             $_godinas=$red['godinas'];
21.             $_pnaziv=$red['pnaziv'];
22.             $_popis=$red['popis'];
23.             $_pdatum=$red['pdatum'];
24.             $_DatumP=substr($red['pdatum'],8,2).'/'.substr($red['pdatum'],5,2).'/'.substr($red['pdatum'],0,4);
25.             $_npredmet=$red['npredmet'];
```

```
26.                 $_gpredmet=$red['gpredmet'];
27.                 $_spredmet=$red['spredmet'];
28.                 $_ime=$red['ime'];
29.                 $_prezime=$red['prezime'];
30.             }
31.             mysqli_free_result($upit);
32.         }
33.         mysqli_free_result($rez1);
34.         mysqli_close($veza);
```

У коду А–11., примећује се да су формирана 2 упита, линија 3 и 15. Упитом у линији 3., се проверава да ли имају подаци у табели *projektni_zadatak_s*, која у ствари чува детаље обавеза за пријављеног корисника. У овом упиту се проверава да ли се страни кључ *idpzs* подудара са изабраним пројектом чија је вредност примарног кључа *$izabranZ*. Уколико има унетих података, број редова у табели који задовољавају услов из упита, линија 3., је већи од 0, и извршавају се линије од до 31. Упитом у линији 15, а на основу релационо повезаних табела, слика 7., у бази података, прикупљају се жељени подаци, смештају се у променљиве (линије од 19 до 29) да би се потом приказали у одговарајућим текстуалним објектима у форми 15.

На овај начин је извршена синхронизација мобилних података са SQL сервером.

7. ЗАКЉУЧАК

У пословном окружењу стање мобилних комуникација има два главна значаја, прво је везано за мобилне операторе, а друго за пословно окружење. У протеклом периоду мобилни оператори су били посвећени изградњом инфраструктуре и повећањем броја претплатника. Повећање мобилних претплатника долази до засићења, а технологија, услуга и мрежа омогућавају промену нових решења. Мобилни оператери почињу све више да се фокусирају на пословне компаније и на примену мобилних услуга у пословном окружењу.

Развој система управљања базама података је помогао да подаци у потпуности постигну своју независност, без обзира на којој локацији у мрежи се налазе. Систем обезбеђује централизовани и контролисани приступ у администрирању података, њихову транспарентност као и да апликација бива имуна на физичку и логичку организацију датотека.

База података је скуп међусобно повезаних података, смештених у спољној меморији рачунара. Подаци су истовремено доступни разним корисницима и апликацијским програмима. Убацивање, промена, брисање и читање података обавља се посредством заједничког софтвера. Корисници и апликације притом не морају познавати детаље физичког приказа података, већ се референцирају на логичку структуру базе.

SQL (Structured Query Language) је стандардни релациони упитни језик (ANSI и ISO стандард) над релационим базама података. Данас се за SQL може рећи да је комплексан, процедурално/декларативан језик. SQL ради са табелама. Табела се креира једном извршном наредбом. Одмах по креирању табела је расположива за коришћење. Сви подаци меморисани су у табелама и резултат, било које операције, се логички приказује у облику табеле.

PHP је open–sorce server–side програмски језик за динамичко генерисање HTML кода. Другим речима, PHP је програмски језик помоћу којег се може креирати HTML страница на серверу пре него што се она пошаље клијенту попуњена динамичким садржајем. Овим начином генерисања садржаја клијент не може видети код (скрипту) који је генерисао садржај који гледа, већ има приступ чистом HTML коду.

Кључну улогу у повезивању ресурса и програмског кода има аутоматски генерисана класа *R*, која садржи јединствене идентификаторе свих ресурса дефинисаних у склопу пројекта. Ресурси у тој класи су статички и хијерархијски су организовани складно са мапама у које су физички смештени. Сваки пута, када корисник промени ресурсе дефинисане у *XML*–у, ако је резултирајући *XML* валидан, *Android Studio* ће аутоматски освежити класу *R* која се одмах потом може користити и у програмском коду.

8. ЛИТЕРАТУРА

[1] S. K. Madria, M. Mohania, S. S. Bhowmick and B. Bhargava. 2002. Mobile data and transaction management. Information Sciences. 141(3): 279-309.

[1] http://www.ericsson.com/hr/etk/dogadjanja/mipro_2008/1230.pdf

[2] Shao Kumar / Database System: Concept, Design and Applications University of Missouri

[3] D. Alex, R. Holly, J. Hildenbrand and A. Martonik, "Android History". Dostupno na: *https://www.androidcentral.com/android-history*.

[4] http://www.verizonwireless.com/support/android-os-faqs/

[5] http://www.dummies.com/how-to/content/looking-at-the-androidoperating-system0.html

[6] Vijay Kumar / Mobile Database Systems, Computer Science an Informatics University of Missouri – Kansas City, Wiley Interscience 2006

[7] http://www.developereconomics.com/five-popular-databases-for-mobile/

[8] Goran Panić, DISTRIBUIRANE BAZE PODATAKA, PRIRODNO – MATEMATIČKI FAKULTET, UNIVERZITETA U NOVOM SADU, 2009. god.

[9] Silberschatz−Korth−Sudarshan, Database System Concepts, Fourth Edition, The McGraw−Hill, Co 2001.

[10] Google Inc. “Android Studio - The Official IDE for Android“, Android Developers, 2016. [Online]. Dostupno na: https://developer.android.com/studio/index.html.

[11] R. Nixon, Learning PHP, MySQL&JavaScript, With jQuery, CSS & HTML5, Fourth edition, O Reilly Media, Inc, Sebastopol, CA, USA 2015.