UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Simon

RAZVOJ PRAVOPISNE VADNICE ZA MOBILNO PLATFORMO ANDROID

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Danijel Skočaj

Ljubljana, 2012

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00236/2012
Datum: 05.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDRAŽ SIMON**

Naslov: **RAZVOJ PRAVOPISNE VADNICE ZA MOBILNO PLATFORMO ANDROID**

DEVELOPMENT OF A SPELLING BOOK FOR ANDROID MOBILE PLATFORM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Z razvojem informacijsko-komunikacijske tehnologije se odpirajo nove možnosti za popestritev in izboljšanje pedagoškega procesa. Otroci so zelo dojemljivi za tehnične novosti in uvajanje le-teh lahko potencialno poveča njihovo zanimanje za učenje oz. poveča učinkovitost učenja. Ena takšnih tehnologij, ki so doživele razmah v zadnjem času, je mobilna platforma Android, oz. pametni telefoni in tablični računalniki, ki podpirajo ta operacijski sistem. V diplomskem delu izdelajte aplikacijo za izvajanje pravopisnih vaj na mobilni platformi Android. Implementirajte različne tipe nalog in pri tem izkoristite specifične lastnosti platforme, ki lahko naredijo učenje bolj zanimivo oz. učinkovito. Aplikacija naj deluje kot odjemalec, ki do nalog, rešitev ter podatkov o pravilno oz. nepravilno rešenih nalogah za vsakega uporabnika dostopa preko interneta. Na strežniški strani omogočite enostaven vnos novih nalog ter učinkovito administracijo sistema.

Mentor:

doc. dr. Danijel Skočaj

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Andraž Simon,

z vpisno številko 63040474,

sem avtor diplomskega dela z naslovom:

Razvoj pravopisne vadnice za mobilno platformo Android

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 1. 9. 2012 Podpis avtorja: _____

ZAHVALA

Zahvaljujem se doc. dr. Danijelu Skočaju za strokovno pomoč in nasvete pri izdelavi diplomske naloge, prav tako se zahvaljujem tudi učiteljici Marti Pavlin.
Posebno se zahvaljujem svojim staršem, ki so mi omogočili študij in me vsa leta podpirali.

KAZALO

SEZNAM KRATIC IN OKRAJŠAV

SUPB – sistem za upravljanje s podatkovnimi bazami
SQL – Structured Query Language
MD5 – Message-Digest Algorithm
SSL – Secure Sockets Layer
PHP – PHP Hypertext Preprocessor
HTML – Hyper Text Markup Language
CSS – Cascading Style Sheets
FTP – File Transfer Protocol
OS – operacijski sistem
IDE – Integrated development environment
SDK – Software development kit
USB – Universal Serial bus
XML – Extensible Markup Language
IP – Internet Protocol
JSON – JavaScript Object Notation

POVZETEK

Število mobilnih naprav in njihovih uporabnikov se iz dneva v dan veča. Igrice in ostale multimedijske vsebine na pametnih telefonih in tabličnih računalnikih so vzbudile veliko pozornosti pri otrocih. Učence osnovne šole bi lahko učne vsebine na takih napravah spodbudile, da bi svoje znanje pridobivali tudi na nov, zanimiv in igriv način. V diplomski nalogi je predstavljena aplikacija za mobilne naprave z nameščenim operacijskim sistemom Android, ki uporabnikom omogoča reševanje pravopisnih nalog. Na voljo je več načinov reševanja; najzanimivejši vključuje uporabo pospeškomera, kjer nagibanje naprave povzroča premikanje vsebine na zaslonu. Poleg mobilne vadnice je na voljo tudi spletna stran, ki uporabnikom omogoča reševanje nalog, učiteljem pa izdelovanje novih. Tako je poskrbljeno tudi za tiste, ki nimajo primerne mobilne naprave. Učitelji lahko na spletni strani z uporabo v naprej dogovorjenih oblik za shranjevanje besedila izdelajo naloge, ki bodo pritegnile več pozornosti.

Ključne besede:
vadnica, pravopis, pospeškomer, mobilne naprave, operacijski sistem Android.

ABSTRACT

The number of mobile devices and their users is increasing on day to day basis. Games and other multimedia contents on smart phones and tablets raised children's attention on a large scale. Primary schools could trigger the pupils' interest by learning also through new, interesting, and playful methods. In the thesis, an application for mobile devices with Android operating system which enables exercises in spelling is presented. There are several possibilities of problem-solving; the most interesting includes the use of accelerometer where tilting of the device makes content to move on the screen. Besides spelling book, a website is also available for users to exercise and for teachers to make new exercises. Therefore, it is also taken care of those who do not have an appropriate mobile device. Using text saving formats set in advance, teachers can create exercises which will trigger more attention of children.

Key words:
spelling book, spelling, accelerometer, mobile devices, Android operating system.

1 UVOD

Kar nekaj let je že, odkar so osnovne šole dobile prve računalniške učilnice. Spomnimo se, kako smo se učili angleškega jezika s pomočjo glasbe, animacije in angleških podnapisov besedil raznih pesmic. Na sredini učilnice je bilo v krogu nameščenih nekaj računalniških ekranov; v vsakega pa nas je gledalo več učencev skupaj.
Vpletenost tehnologije v naš učni sistem se je z leti stopnjevala in novi učni sistemi so kmalu omogočali individualno interakcijo z uporabnikom. Starejši učni sistemi so bili shranjeni na zgoščenkah. Potrebno jih je bilo namestiti na vsak računalnik in učenci so lahko vaje reševali individualno ali v skupini. Učiteljica je hodila od računalnika do računalnika in preverjala napredek učencev.
Z naprednejšimi internetnimi povezavami so se razvili tudi sistemi, ki delujejo na principu odjemalec – strežnik. Logika sistema je shranjena na strežniku, uporabniki pa do vsebin dostopajo s spletnim brskalnikom, ki je odjemalec. Eden takšnih sistemov je tudi e-učilnica, ki uporabnikom med drugim omogoča reševanje nalog; sistem zna preveriti odgovore in jih zapisati v obliki ocene. Za dostopanje do takega sistema potrebujemo le internetno povezavo oziroma povezavo preko lokalnega omrežja.
Danes lahko razmišljamo, kako bi omogočili naprednejše tehnologije in prenosljivost vsebin, da uporabniku ni potrebno sedeti na enem mestu.

Dobili smo zamisel, da bi tak sistem realizirali na pametnih telefonih, ki jih bo v naslednji generaciji imel že vsak osnovnošolec. Razvili smo aplikacijo Slo-vadnica za mobilno platformo Android, ki omogoča reševanje pravopisnih vaj. Otrokom so všeč tehnološke novosti, se nanje hitro privadijo in se vsebino prej naučijo. Računalnik je pritegnil pozornost najmlajših tudi z računalniškimi igricami. Brez računalnika, priznajmo, danes ne gre več.
Mobilne naprave so danes že pravi mali računalniki, ki jih nosimo s seboj, kamor koli gremo. Tako lahko vsaj nekaj opravil opravimo kje drugje, ne pa nujno doma za računalnikom. Osnovnošolec bi s pametnim telefonom in nameščeno aplikacijo Slo-vadnica lahko namesto igric reševal pravopisne vaje na avtobusu na poti v šolo. Prav tako bi lahko na hodniku pred učilnico z nekaj vajami ponovil snov tik pred pisnim preizkusom.

Vsekakor je potrebno poskrbeti tudi za tiste, ki pametnih telefonov nimajo. V ta namen bi bile vaje dostopne tudi na spletni strani, kamor bi lahko dostopali vsi uporabniki. Na voljo smo imeli vaje iz slovenskega pravopisa v besedilni obliki za peti razred osnovne šole, ki naj bi jih ponudili učencem na spletni strani in mobilnih napravah z operacijskim sistemom Android. Shranili smo jih v podatkovno bazo na strežniku, do katere dostopamo preko interneta. Eden večjih problemov s stroškovnega vidika pametnih telefonov je prenos podatkov v mobilnem omrežju. Cilj mobilne aplikacije je torej tudi, da podatke shranjuje lokalno na napravi in da ni potrebno trošiti podatkovnega prenosa.

Pomembno je, da uporabnikom predstavimo drugačen način reševanja nalog, kot so ga bili vajeni do sedaj. Učence pogosto manj zanimajo učne vsebine, ki se jim ne zdijo privlačne. Če jim ponudimo izbiro učne vsebine na računalniku ali pa v delovnem zvezku, v katerega morajo odgovore pisati s pisalom, bodo v večini izbrali prvo možnost. Čas je, da pokažemo še tretjo možnost in jim ponudimo aplikacijo za pametne telefone.

Sodobne mobilne naprave zaradi novih tehnologij že prehitevajo običajne namizne računalnike. Učenci bodo na naprednejših napravah spremljali učne vsebine z večjim zanimanjem, hkrati jim bodo vsebine še bližje, tako rekoč v žepu. Podobno kot velja nevarnost, da se lahko računalnik izkoristi predvsem za računalniške igrice, tako bi lahko bili pametni telefoni privlačni učencem le zaradi igric in ostalih vsebin, ki so namenjene zabavi. Da bi jim približali tudi resnejše teme, jim lahko prikažemo učenje tudi tako, da se združi učenje in zabavo; potrebno je le ponuditi reševanje nalog na več zanimivih načinov, saj bi se enega samega hitro naveličali.

Tako smo razvili sistem, kjer lahko učenci z aplikacijo Slo-vadnica ali preko spletne strani dostopajo do vsebin in rešujejo pravopisne vaje. Ob tem se beležijo odgovori in sistem prikazuje pravilnost rešenih nalog.
Naloge se lahko rešuje na tri različne načine. Najbolj zanimiv je tisti, kjer moramo s pomočjo nagibanja mobilne naprave premikati besedilo po zaslonu nad tisto polje, kamor spada. Takega načina reševanja nalog na spletni strani ni mogoče izvesti, zato so tam na voljo izbirni gumbi.

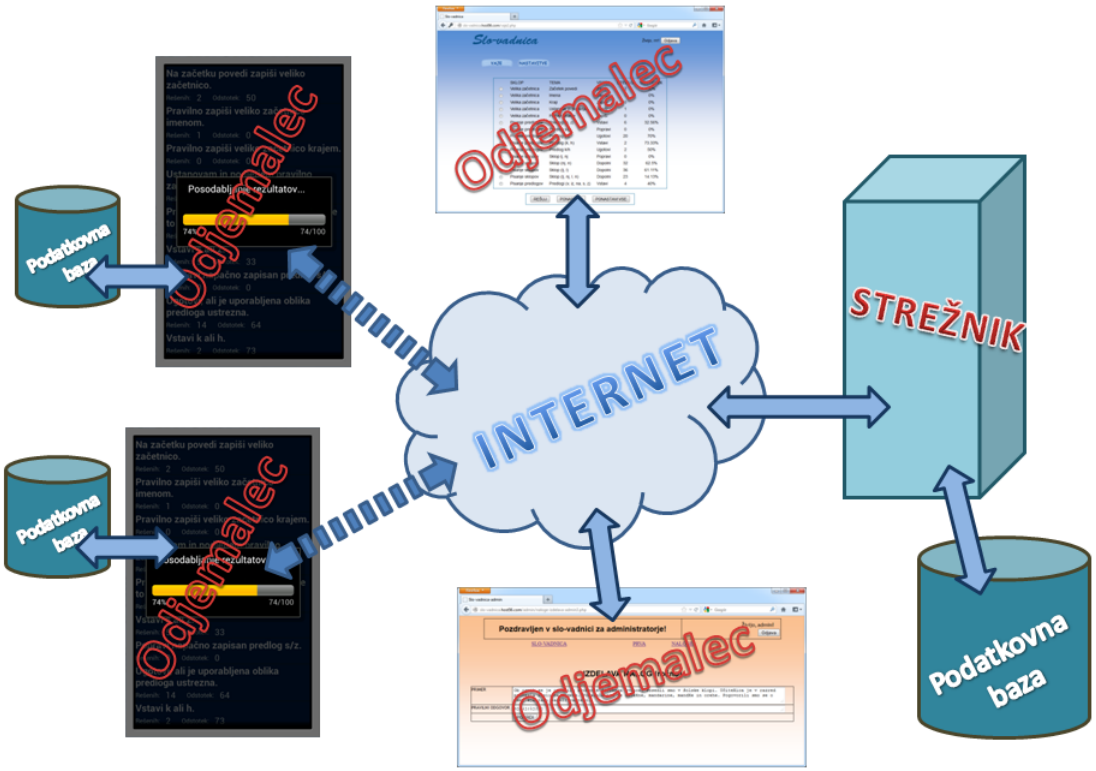
Aplikacija Slo-vadnica ima enostavno strukturo in potek aktivnosti: prijava, izbor naloge in reševanje. Poleg tega se lahko na uporabnikovo zahtevo vsebine posodobijo. V aktivnosti, ki prikazuje vrste nalog, je to mogoče narediti preko menijske funkcije Posodobi.
Ciljno občinstvo so poleg osnovnošolcev tudi učitelji, ki lahko prek spletne strani vnašajo lastne naloge za reševanje. Zapišejo jih v taki obliki, da jih sistem lahko samodejno obdela in zmore prikazovati na različne načine.

V drugem poglavju diplomskega dela so na grobo opisane zahteve sistema.
V tretjem poglavju so naštete nekatere uporabljene tehnologije, ki so razdeljene na strežniški in odjemalni del.
V četrtem poglavju je podrobneje opisano delovanje sistema, v zadnjem poglavju pa so predstavljene nekatere slabosti in nekaj možnih izboljšav.

2 ZAHTEVE SISTEMA

2.1 Zgradba sistema

Želeli smo ustvariti sistem, ki bi ga z arhitekturnega vidika razdelili na strežniški in odjemalni del. Na strežniku bi imeli vsebine shranjene v podatkovni bazi, datoteke spletnih strani in skripte PHP. Odjemalni del smo želeli realizirati z aplikacijo za mobilne naprave in s spletno stranjo, kjer bi učenci lahko reševali pravopisne vaje. Učitelji pa bi nove naloge lahko vnašali preko spletne strani. Zaradi varčevanja s podatkovnim prenosom bi aplikacija imela svojo, lokalno podatkovno bazo, ki bi se posodobila na željo uporabnika. Pogojni dostop aplikacije do strežniške podatkovne baze je s prekinjenimi puščicami prikazan na Sliki 1, kjer je prikazana shema sistema. Ostali odjemalci morajo komunicirati s podatkovno bazo na strežniku neposredno prek interneta.



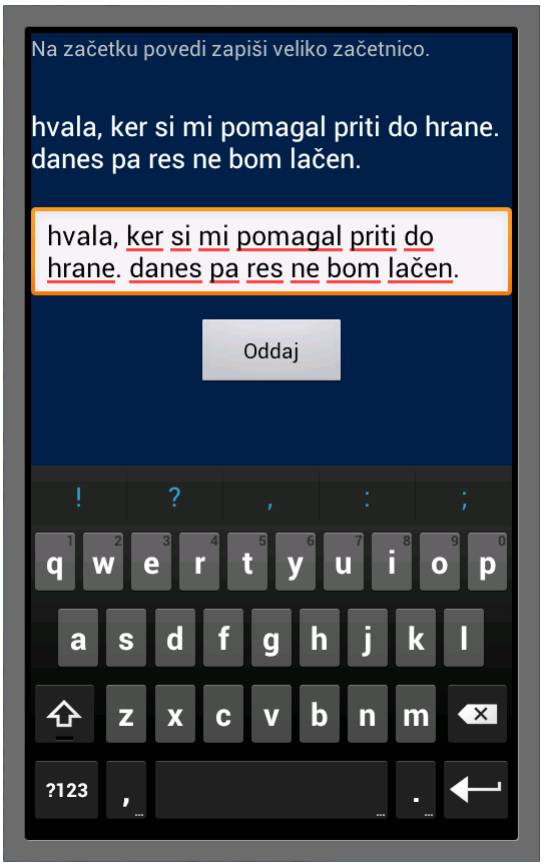
Slika 1: Prikaz sheme sistema.

2.2 Ciljna aplikacija

Glavni cilj je bil razviti aplikacijo za mobilne naprave, v katero bi se učenci prijavili z uporabniškim imenom in geslom. Po prijavi bi se prikazal seznam nalog, ki bi jih učenec lahko reševal na več načinov. Za vsak način bi aplikacija zagnala primerno aktivnost.

Prvi način reševanja nalog bi od učenca zahteval, da bi zapisal odgovor v vnosno polje s celo povedjo. Odgovor bi se zato ocenjeval kot celota.
Drugi način bi zahteval od učenca, da bi z nagibanjem telefona usmerjal besedilo po zaslonu in s tem podal odgovor, ali je primer pravilen ali ne.
Tretji način bi prikazoval besedilo, v katerem bi moral učenec na določenih mestih menjati vsebino z dotikom. Taka mesta bi bila obarvana zato, da bi jih učenec lažje prepoznal.

Aktivnost, ki bi bila namenjena prikazovanju naloge z vnosnim poljem, je prikazana na Sliki 2. Ta vrsta naloge bi bila manj zanimiva za reševanje, saj je pisanje z virtualno tipkovnico težje kot z računalniško. Vseeno bi bila to najenostavnejša oblika za prikaz neke splošne naloge, ki je ne bi bilo možno predstaviti na ostala dva načina.

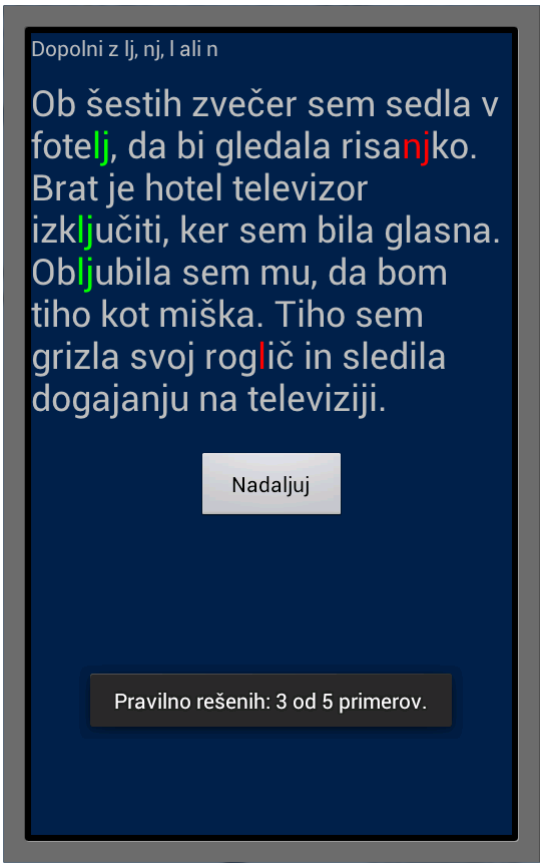


Slika 2: Prikaz aktivnosti za prikaz nalog z vnosnim poljem.

Aktivnost, ki bi učencu omogočala reševanje nalog s pomočjo pospeškomera je prikazana na Sliki 3. Ta način reševanja nalog bi bil bolj zanimiv. Pospeškomer je tehnologija, ki jo imajo sodobne mobilne naprave, navadni računalniki pa ne. S tem bi dosegli več zanimanja s strani učencev. Z nagibanjem mobilne naprave bi učenec premikal tudi besedilo, ki je prikazano na zaslonu, in bi ga usmeril na zeleno ali rdeče polje.

Slika 3: Primer aktivnosti za prikaz nalog, kjer se uporablja pospeškomer.

Aktivnost za prikazovanje nalog z vstavljanjem bi temeljila na prikazu obogatenega besedila, v katerem bi učenec na določenih mestih menjal vsebino. Po tem ko bi nalogo oddal, bi se vsebine pobarvale zeleno ali rdeče in učenec bi lahko preveril svoje odločitve. Tako vrsto naloge prikazuje Slika 4.

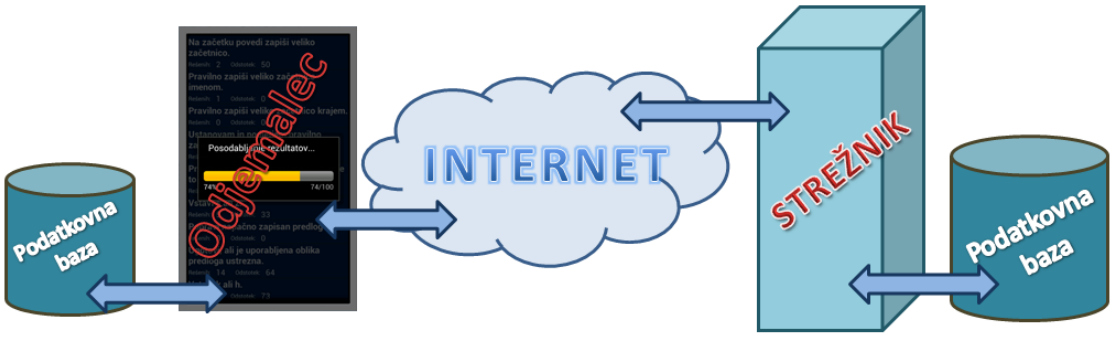


Slika 4: Primer aktivnosti za prikaz naloge s polji na dotik.

Da bi omogočili reševanje nalog tudi tistim, ki te aplikacije ne bi imeli, pa bi bile vsebine dostopne tudi na spletni strani. Učenci bi se lahko prijavili z istim uporabniškim imenom in geslom tudi na spletno stran, kjer bi lahko reševali naloge. To pomeni, da bi morali rezultate reševanj za posameznega učenca sešteti skupaj. Tiste z mobilne naprave in tiste s spletne strani. Zato bi bile potrebne posodobitve vsebin.

Kljub želji, da aplikacija ne bi komunicirala s strežnikom prek interneta neprestano, bi bilo potrebno poskrbeti za vsebine, ki bi jih prikazovala. Aplikacija bi uporabljala lokalno podatkovno bazo, ki bi se posodobila ob prvi prijavi učenca, v nadaljevanju pa samo na njegovo zahtevo. Na ta način bi dosegli nadzor nad tem, kdaj bi se aplikacija posodabljala. Tako bi se lahko vse posodobitve izvedle takrat, ko bi bila mobilna naprava v nekem domačem omrežju oziroma tam, kjer bi lahko brezplačno prenašali podatke.

Ob posodabljanju vsebin bi aplikacija postala odjemalec. Na Sliki 5 je prikazana shema principa odjemalec – strežnik pri posodabljanju aplikacije.

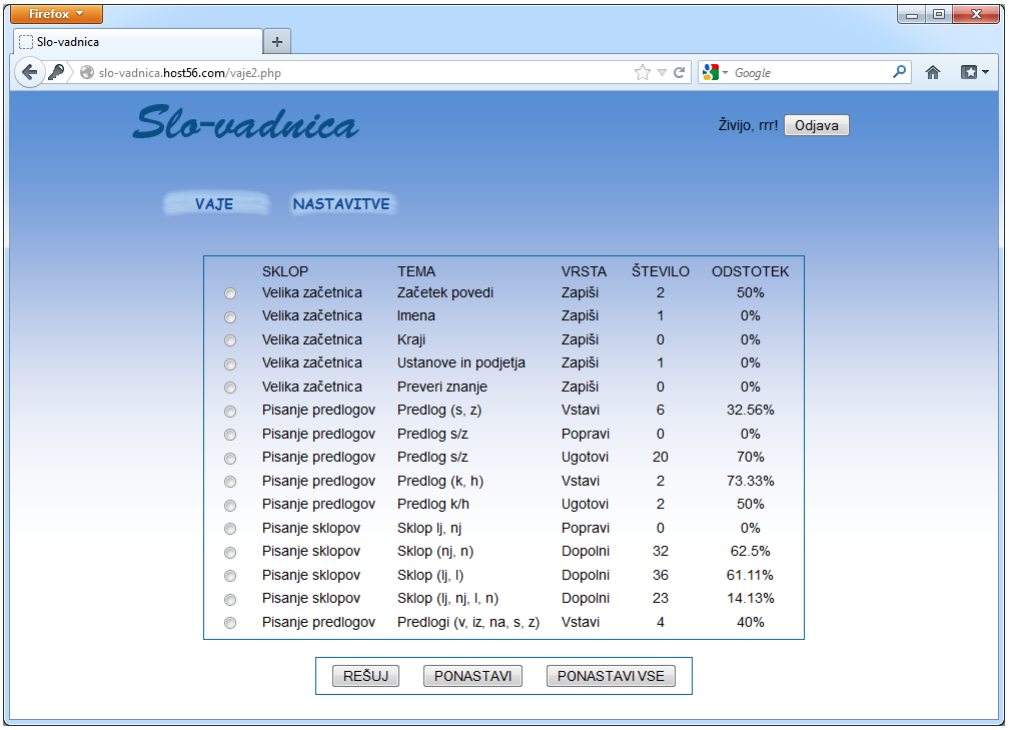


Slika 5: Shema principa odjemalec – strežnik pri posodabljanju Slo-vadnice.

2.3 Spletna stran

Spletna stran bi bila namenjena tistim, ki nimajo mobilnih naprav z nameščenim operacijskim sistemom Android in si ne bi mogli namestiti aplikacije.
Po prijavi bi se učencu na spletni strani prikazale vse vrste nalog, kakor je prikazano na Sliki 6. Prikazalo bi se tudi število rešenih nalog in uspešnost pri reševanju (izražena v odstotkih). Na voljo bi tudi tu bili trije načini reševanja nalog, saj bi se učenec enega samega verjetno hitreje naveličal.
Prvi način bi nalogo prikazal v širokem besedilnem polju, kamor bi učenec moral napisati odgovor s celo povedjo. Odgovor bi se tudi tu, tako kot v aplikaciji za mobilne naprave, ocenjeval v celoti.

Drugi način bi nalogo prikazal tako, da bi učenec z gumbom označil, ali je odgovor pravilen ali ne. Ta način reševanja nalog bi se na spletni strani reševal hitreje kot v aplikaciji, kjer bi nagibali mobilno napravo.
Tretji način pa bi nalogo prikazal tako, da bi učenec moral v majhna besedilna polja, ki bi bila vrinjena med navadno besedilo, vpisovati le delce besed.
Ko bi učenec zaključil z reševanjem, bi se prikazali rezultati oddane naloge. Svoje rezultate bi lahko tudi ponastavil.

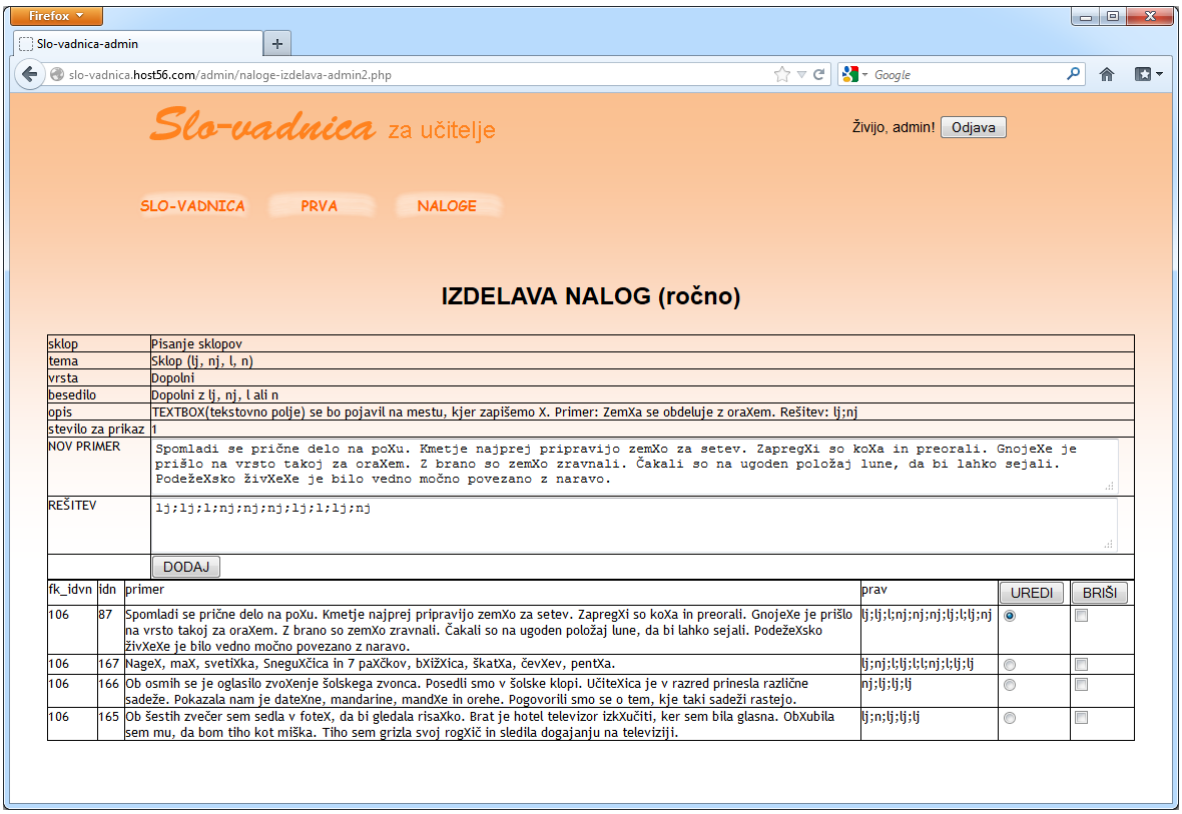


Slika 6: Prikaz vseh vrst nalog na spletni strani za učence.

2.4 Urejanje vsebin

Del sistema, namenjenega učiteljem, bi bil realiziran s spletno stranjo, na katero bi se učitelj prijavil. Po prijavi, bi imel na spletni strani celovit pregled nad vsebino, ki bi jo lahko dodajal in spreminjal po potrebi/želji.

Učenci bi naloge lahko reševali na tri načine, zato bi moral učitelj ob izdelovanju novih nalog upoštevati dogovorjene oblike za vnašanje besedila. S tem bi omogočil samodejno oblikovanje nalog, ki bi jih imel učenec na voljo za reševanje. Učitelj bi po vnosu besedila spremembe potrdil. Obrazec, ki bi omogočal učiteljem vnašanje novih ali popravljanje starih nalog je prikazan na Sliki 7.



Slika 7: Izdelava nalog na spletni strani za učitelje.

3 TEHNOLOGIJE IN ORODJA

3.1 Splošno

Za razvoj celotnega sistema smo uporabili veliko tehnologij in si pomagali z več orodji. Opisane so nekatere strežniške tehnologije in nekaj tistih, ki so potrebne za delovanje aplikacije Slo-vadnica.

Spletna stran je napisana v skriptnem jeziku PHP, ki je integriran v HTML. Za oblikovanje smo uporabili CSS. Za boljšo uporabnikovo izkušnjo pa še JavaScript, ki preverja uporabnikove akcije in vsebino spletnih obrazcev, preden se jih pošlje na strežnik.

Za urejanje datotek smo uporabili orodje NetBeans IDE. Omogočilo nam je urejanje datotek CSS s predogledom njihovega videza. Z orodjem NetBeans IDE smo prek protokola FTP dostopali do strežnika in si shranili povezavo. Izbrali smo ponudnika brezplačnega spletnega gostovanja in uporabljali orodje phpMyAdmin za dostopanje do podatkovne baze MySQL.

Opisan je operacijski sistem Android in osnovna arhitektura aplikacije. S podatkovnim modelom je na mobilni napravi prikazana lokalna podatkovna baza. Omenjen je paket orodij Android software development kit, ki ga skupaj z Eclipse IDE uporabljamo za razvijanje aplikacije Android.

3.2 Spletno gostovanje

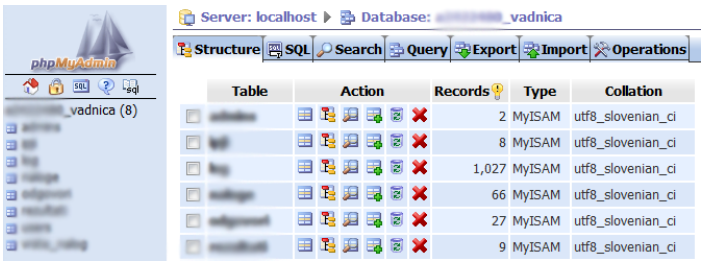
Za izvedbo sistema smo potrebovali prostor, kamor smo vsebine shranili, in sistem, ki omogoča prikaz in dostop do njih. Odločili smo se za uporabo brezplačnega spletnega gostovanja pri enem od mnogih ponudnikov takih storitev.
Brezplačne storitve imajo nekaj slabosti, vendar zadostujejo za trenutne potrebe našega sistema. Na voljo imamo dovolj prostora za shranjevanje datotek in obilo prenosa podatkov. Ta se bo s časom in z večanjem uporabe našega sistema povečeval, a nikoli ni prepozno zaprositi ponudnika spletnega gostovanja za boljši paket storitev, ki bo omogočal dovolj resursov.
Večja slabost brezplačnega spletnega gostovanja se kaže v tako imenovanih »downtime« obdobjih [1], ko strežnik, na katerem imamo postavljeno spletno stran, ni na voljo. Naš sistem je trenutno izpostavljen takim obdobjem in prepuščeni smo na milost in nemilost programski opremi, napravam in vzdrževalcem pri izbranem ponudniku omenjenih storitev.
V prihodnosti bi sistem lahko preselili na kakšen primernejši strežnik na šoli, ki bi sistem uporabljala.

3.3 Strežniške tehnologije

3.3.1 Shranjevanje podatkov in MD5

Spletni gostitelj ponuja uporabo sistema za upravljanje s podatkovnimi bazami (SUPB) MySQL. Je široko uporabljena, zmogljiva, odprtokodna implementacija relacijske podatkovne baze [2]. Deluje na principu odjemalec – strežnik, pri čemer odjemalec pošilja zahteve na strežnik s pomočjo SQL stavkov. Jezik SQL je standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami in je standard že od leta 1986 [3].

Za izdelavo podatkovne baze smo uporabili orodje phpMyAdmin, ki je odprtokodno, brezplačno in je namenjeno upravljanju SUPB MySQL preko brskalnika [4]. Izdelava podatkovne baze, tabel in vrstic je s pomočjo tega orodja zelo enostavna. Dodajamo lahko tudi pravice uporabnikom, namesto obdelovanja s pomočjo klikanja pa lahko vpisujemo tudi SQL stavke. Pogled v orodje phpMyAdmin je prikazan na Sliki 8.



Slika 8: Pogled v orodje phpMyAdmin.

Dostop do podatkovne baze, brisanje in dodajanje novih vrstic je izvedeno s pomočjo SQL stavkov, ki jih na spletni strani dinamično oblikujemo in izvajamo. Tako se ob registraciji učenca v podatkovno bazo vpišejo njegovi podatki, geslo pa se najprej obdela s pomočjo algoritma MD5. Ta algoritem enosmerno preoblikuje poljubno dolgo vsebino in jo zapiše kot 128 bitno vrednost [5]. Predstavimo jo lahko z 32-mestnim šestnajstiškim številom.

Enosmernost algoritma v tem primeru pomeni, da se ne da ugotoviti prvotne vsebine. V Tabeli 1 je razvidno, da je pri algoritmu MD5 vsak izhodni bit odvisen od vsakega vhodnega bita. Z drugimi besedami – sprememba le ene črke v originalnem nizu vrne povsem drugačen rezultat.

Niz	MD5
Žan pomiva posodo.	543453a43b71d2ac0cf7771f0f12ab96
Jan pomiva posodo.	96118075d18e76c95e8c95ff1af23238

Tabela 1: Prikaz spremembe izhodnega niza algoritma MD5.

MD5 je bil razvit leta 1991 in je nadomestil starejši MD4. Od takrat se je pokazalo, da ni primeren za certifikate SSL ali digitalne podpise, saj ni preveč težko najti dveh različnih vsebin, kjer bi algoritem za njiju podal enak rezultat [6].

Slabost tega algoritma je tudi to, da danes obstaja že veliko strežnikov s t. i. mavričnimi tabelami, ki vsebujejo kombinacije originalne vsebine in rezultata za to vsebino. Tako lahko na način »brute force« poiščemo, če kombinacija že obstaja v tabelah [5]. Vseh kombinacij, ki jih lahko predstavimo z 32-mestnim šestnajstiškim številom, je $3{,}4 * 10^{38}$.

Preizkusili smo orodje MD5 Decrypter [7], ki je eden izmed iskalnikov MD5 kombinacij in ugotovili, da enostavna gesla niso varno shranjena s kombinacijo MD5, saj so že vsebovana v mavričnih tabelah. V Tabeli 2 so vrstice z najdenimi nizi obarvane temneje. Kot vidimo, je najpomembnejše, da si izbiramo zahtevnejša gesla. Kljub slabostim je ta algoritem dovolj dober za naše potrebe, saj so glavno ciljno občinstvo učenci osnovne šole.

Niz	MD5	NAJDEN NIZ
andraz	7e9c6d0286ec25dc9428a4282d120624	andraz
andraz123	10ef2090c3fa29b32a63913043a997e9	andraz123
Andraz	665a7951b3f4d13eaeb3879738e6c728	Andraz
Andraz123	7733e29f4dabe8f811645d6cc39a659d	[Not Found]
AndraZ	c06dbc8c9bd4adf395c2dbd72d63e220	[Not Found]
sandraz	d89b71a32d19f24f7340e67492ea9150	sandraz
sandraz123	441b2b5355e6b7708e63092efeff54ff	sandraz123
s1andraz3	d77f57caa782db39a92bccb02ba6e36e	[Not Found]
s123andraz	448eb0097f71419536a723136040f9de	[Not Found]
123andraz	682563e9a03527a90dc0382ef30c173c	[Not Found]
123sandraz	6ec0d511d7c55e16554e7eceb81a80a5	[Not Found]
S123Andraz	b35a62254b357a7c183b45d274eb272d	[Not Found]
Andraz123S	fb278c394d8da53ab3264f8c40cd0404	[Not Found]
A1ndraz2S3	9c334b6f0ed1ca0bc942035422470bc5	[Not Found]
AndraZ123	984a460c3a9c0413b9596282b353c87d	[Not Found]

Tabela 2: Prikaz najdenih kombinacij MD5 v eni izmed mavričnih tabel.

3.3.2 HTML, CSS in PHP

HTML predstavlja osnovo spletnega dokumenta, ki ga lahko tudi oblikujemo, vendar je primarni namen izdelati strukturo in semantiko vsebine [8]. Največkrat se izognemo oblikovanju v HTML, saj moramo spremembo, ki bo vidna na vseh spletnih straneh, aplicirati na veliko mestih. Za oblikovanje zato uporabimo CSS, da lahko sprememba v eni datoteki

vpliva na obliko na vseh spletnih straneh. Vse spletne strani sistema uporabljajo tudi PHP za dinamičen prikaz vsebin.
PHP je eden izmed prvih razvitih skriptnih jezikov, ki ga lahko umestimo v dokument HTML [9]. Strežnik z interpretiranjem takih dokumentov generira spletno stran, ki jo pošlje odjemalcu.

3.3.3 Protokol FTP

Za dostopanje do datotek na strežniku smo uporabili protokol FTP. Deluje na principu odjemalec – strežnik in je namenjen pošiljanju in prejemanju datotek [10]. Med strežnikom in odjemalcem sta vzpostavljeni dve povezavi, kontrolna in podatkovna povezava. Slednja je lahko realizirana na aktiven način, če je vzpostavljena od strežnika k odjemalcu – ali na pasiven, če je obratno. Slika 9 prikazuje smeri povezav za oba načina.
Odjemalec vzpostavi kontrolno povezavo od naključnih vrat do strežniških vrat 21. Pri aktivnem načinu nato strežnik vzpostavi podatkovno povezavo od vrat 20 do posredovanih odjemalčevih naključnih vrat. Pri pasivnem načinu pa odjemalec vzpostavi podatkovno povezavo od naključnih vrat do posredovanih strežniških naključnih vrat.
Aktivni način povezave je primernejši z vidika varnosti strežnika, vendar potrebuje nekaj več nastavitev na strani odjemalca [11].

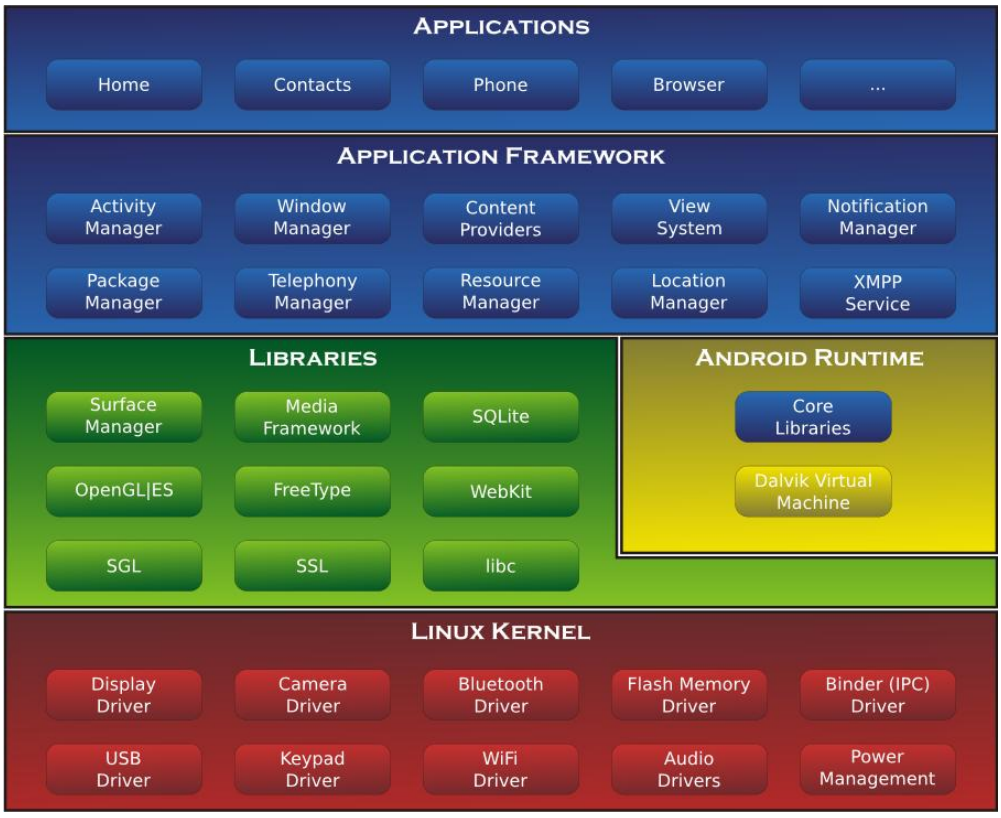
Slika 9: Aktivni in pasivni način povezave FTP, povzeto po [11].

3.4 Tehnologije v ozadju Slo-vadnice

3.4.1 Operacijski sistem Android

Linux je brezplačen, odprtokodni operacijski sistem. Na njegovem jedru temelji tudi operacijski sistem Android, ki ga že od leta 2005 naprej razvija podjetje Google in je primarno namenjen mobilnim napravam [12]. Prvi pametni telefon z OS Android je v prodajo prišel leta 2008 in s tem nakazal konkurenco Appleovemu iPhoneu [13]. Do sredine leta 2012 je razvitih deset različic OS Android in aktiviranih 400 milijonov naprav [14].

Glavne komponente OS Android so vidne na Sliki 10. Na najnižjem nivoju imamo jedro Linux, ki skrbi za upravljanje s procesi in pomnilnikom. Knjižnice, ki so napisane v programskem jeziku C in C++ uporabljajo različne komponente OS Android. Tu se nahaja tudi knjižnica SQLite, ki jo v naši aplikaciji uporabimo za izdelavo lokalne podatkovne baze. Zaradi varnosti se vsaka aplikacija Android izvaja v svojem navideznem stroju Dalvik. Aplikacijsko ogrodje vsebuje komponente, ki jih uporabljajo vse aplikacije, tudi tiste, ki jih pišemo sami. Napisano je v programskem jeziku Java. Na najvišjem nivoju imamo aplikacije, ki so najbližji vmesnik med človekom in strojem.



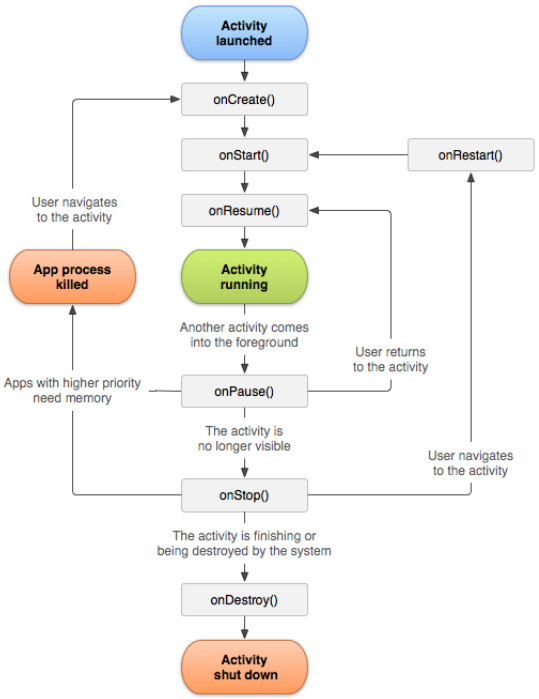
Slika 10: Glavne komponente operacijskega sistema Android [12].

3.4.2 Arhitektura aplikacije

Osrednji produkt sistema je aplikacija za Android. Tok aplikacije nas vodi skozi aktivnost prijave, preko aktivnosti za izpis vseh vrst nalog in rezultatov do aktivnosti, kjer nalogo rešujemo. *Aktivnost* (Activity) je ena od štirih komponent aplikacije in ima svoj življenjski cikel [15] prikazan z diagramom na Sliki 11. Njen namen je komunikacija z uporabnikom preko uporabniškega vmesnika. Ostale komponente so še *storitev* (Service), *ponudnik vsebin* (ContentProvider) in *sprejemnik oddajanja* (BroadcastReceiver) [16]. *Storitev* se izvaja v ozadju in je namenjena izvajanju časovno zahtevnejših operacij, kot je predvajanje glasbe v ozadju. S pomočjo *ponudnika vsebin* lahko določene vsebine vidijo in obdelujejo tudi druge aplikacije. Če želimo ugotoviti, kaj se dogaja s sistemom, na primer ali je baterija prazna, lahko to naredimo s pomočjo *sprejemnika obvestil* v sistemu.

S pomočjo sklada upravljamo z aktivnostmi tako, da pred zagonom nove aktivnosti starejšo odložimo na sklad. Ko uporabnik zaključi z neko aktivnostjo, mu z vrha sklada postrežemo z zadnjo aktivnostjo, ki je bila v ospredju pred tem. Pred zagonom nove aktivnosti se pri trenutni izvede metoda *onPause*. Če želimo shraniti spremembe, ki jih je naredil uporabnik v trenutni aktivnosti, moramo za to poskrbeti v tej metodi, saj ni nujno, da se bo uporabnik v to aktivnost še kdaj vrnil.

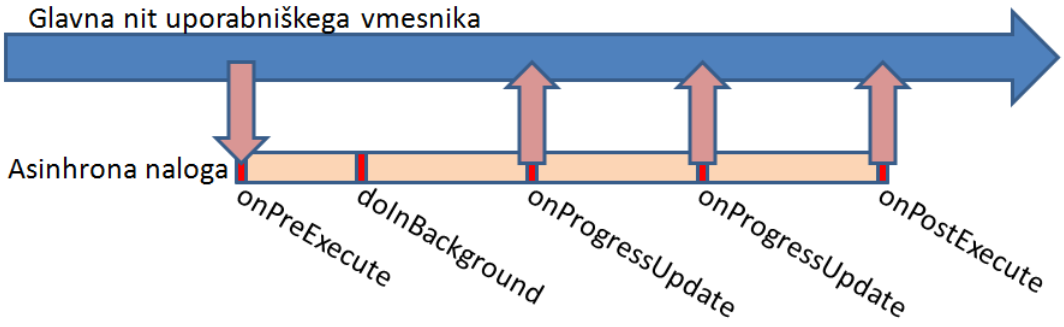
Obstaja več razlogov, zakaj bi lahko prišlo do izgube uporabnikovih sprememb. Če ima uporabnik v ospredju aktivnost, ki potrebuje veliko pomnilnika in je ta zaseden s strani druge aktivnosti, se druga zaključi in prepusti vire aktivnosti v ospredju. Včasih pa uporabnik preko upravitelja opravil ročno počisti pomnilnik in s tem prisilno zapre aktivnost, ki čaka v ozadju.



Slika 11: Življenjski cikel aktivnosti v aplikaciji Android [15].

3.4.3 Razred AsyncTask

Razred *AsyncTask* uporabimo za izvajanje določenih operacij v ozadju in objavljanju rezultatov v ospredje. Glavna nit uporabniškega vmesnika se izvaja v ospredju. To je z uporabniškega vidika najpomembnejša nit, saj uporabnik preko nje komunicira z aplikacijo. Za izvajanje dolgotrajnih operacij, kot je komunikacija s strežnikom na spletu, ne zaposlimo glavne niti, ki zaradi takega izvajanja ne bi upoštevala novih zahtev uporabnika in bi uporabnik dobil občutek, da je z aplikacijo nekaj narobe. V ta namen uporabimo razred *AsyncTask*. Ko zaženemo asinhrono nalogo, se ta izvede v štirih korakih preko metod: *onPreExecute*, *doInBackground*, *onProgressUpdate* in *onPostExecute* [17].
Metoda *onPreExecute* se izvede takoj, ko opravilo zaženemo, in je namenjena nastavitvam osnovnih parametrov opravila. Metoda *doInBackground* je namenjena izvajanju opravila v ozadju, izvede se takoj za metodo *onPreExecute*. Tu lahko uporabimo metodo *publishProgress*, ki pošlje stanje metodi *onProgressUpdate*. S pomočjo metode *onProgressUpdate* je možno glavni niti sporočati stanje zagnanega opravila. Nazadnje se izvede še metoda *onPostExecute*, ki glavni niti posreduje končne rezultate opravila. Na Sliki 12 je prikazan potek dogodkov v asinhroni nalogi.

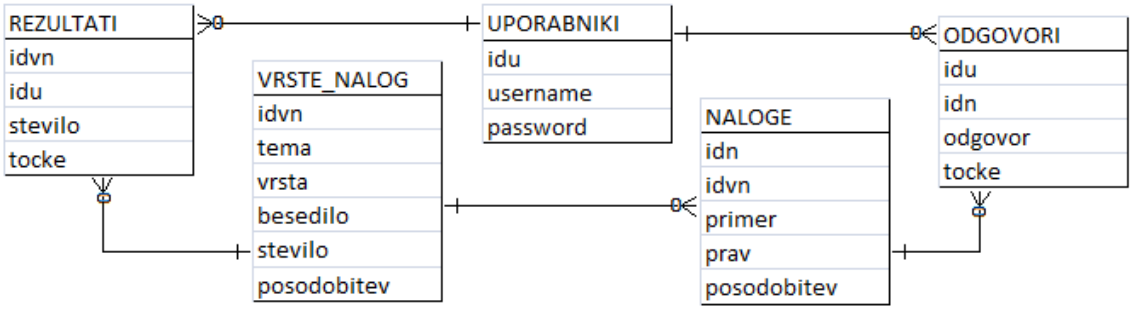


Slika 12: Potek dogodkov v asinhroni nalogi.

3.4.4 Lokalna podatkovna baza SQLite

Zaradi hitrejše odzivnosti in varčevanja s pretokom podatkov smo se odločili, da bo aplikacija Slo-vadnica uporabljala lokalno podatkovno bazo. Uporabili smo SUPB SQLite in razred *DatabaseHandler*, ki razširja razred *SQLiteOpenHelper*. To je razred, ki skrbi za izdelavo, odpiranje in posodabljanje podatkovne baze.
Vsebuje pet tabel: *Uporabniki*, *Vrste_nalog*, *Naloge*, *Odgovori* in *Rezultati*. V tabeli *Uporabniki* so shranjena uporabniška imena in gesla uporabnikov, ki so se uspešno prijavili v aplikacijo na isti mobilni napravi. Vrste nalog in naloge so določene z unikatno številko; s pomočjo atributa posodobitev pa lahko ugotovimo, kdaj je bila naloga shranjena v tabelo. Odgovori so določeni z unikatno številko uporabnika in naloge. V to tabelo se odgovor vpiše

vsakič, ko uporabnik rešuje naloge in jih odda. Ob posodabljanju vsebin pa se iz nje brišejo vsi njegovi odgovori. Na tabelo *Rezultati* lahko vplivajo le funkcije, ki so povezane v proces posodobitve, da lahko obdržimo konsistentnost podatkov. Na Sliki 13 je prikazan podatkovni model aplikacije.

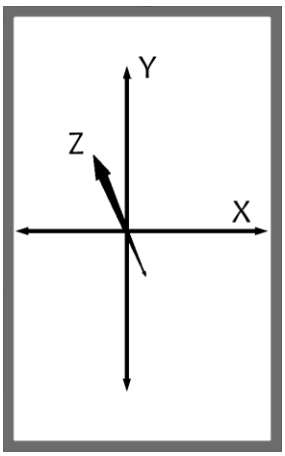


Slika 13: Podatkovni model aplikacije.

V aplikaciji izdelamo objekte, ki so v pomoč pri obdelovanju podatkov. To so objekti, ki ustrezajo vsebinam tabel iz podatkovne baze. V funkcijah razreda *DatabaseHandler* izvajamo poizvedbe SQL. S kazalcem se pomikamo po vsebini odgovora in shranjujemo vrednosti v primerne objekte. Nekatere funkcije razreda *DatabaseHandler* vrnejo seznam takih objektov.

3.4.5 Pospeškomer

Na voljo imamo naloge, ki se jih rešuje s pomočjo nagibanja mobilne naprave. Pospeškomer zazna premikanje mobilne naprave na treh koordinatnih oseh. Razred *SensorEventListener* se uporablja za pridobivanje obvestil pospeškomera s pomočjo metode *onSensorChanged*, ki se izvede, ko se vrednosti senzorja spremenijo [18]. Tako lahko ob vsaki spremembi nagiba naprave izračunamo pozicijo, ki jo uporabimo pri določenih nalogah. Senzor zazna pospeške na osi X, Y in Z, kot je prikazano na Sliki 14.

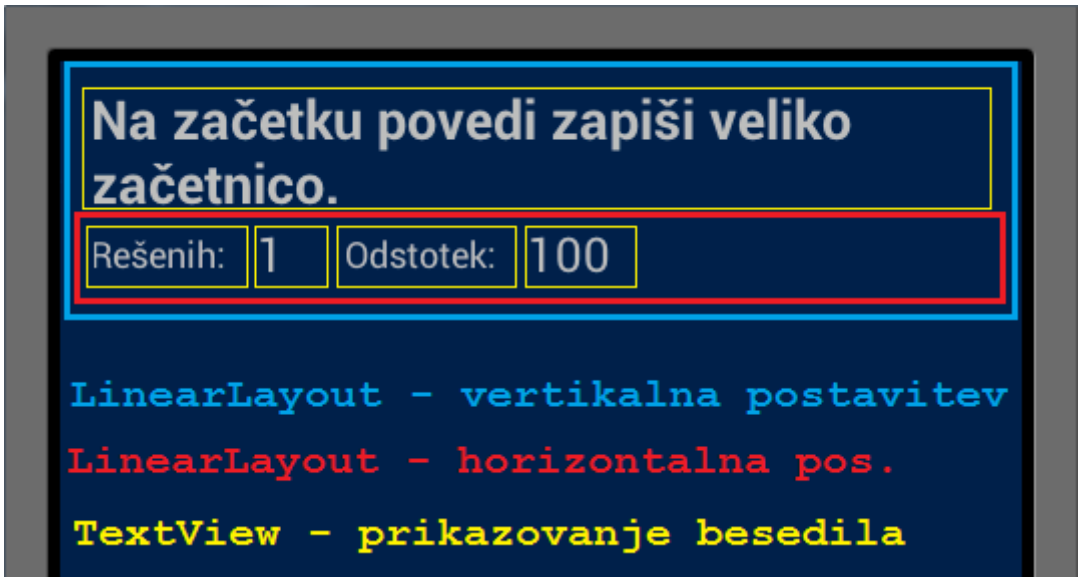


Slika 14: Osi merilnika pospeškov.

Zanimata nas samo osi X in Y. Napravo položimo na ravno površino, z zaslonom obrnjenim navzgor. Ko napravo nagnemo, se na osi X ali Y poveča pospešek zaradi privlačne sile Zemlje. Če je zaslon naprave pravokotno na ravno površino in naprave ne premikamo, bo maksimalen pospešek na pravokotni osi glede na ravno površino približno 9,81 ms^2.

3.4.6 Komponente uporabniškega vmesnika

Za primer podajmo uporabniški vmesnik aktivnosti, kjer prikažemo seznam vrst nalog. Vsak element seznama vsebuje pet komponent *TextView*, ki so namenjene prikazovanju besedila. Razvrščene so v dve linearni razporeditvi *LinearLayout*. Element seznama je najprej vertikalno razdeljen na dva dela. Prvi del je namenjen komponenti *TextView*, ki prikazuje besedilo naloge, drugi del pa vsebuje novo linearno razporeditev, tokrat horizontalno. V njej si zaporedno sledijo štiri komponente *TextView*, dve od njih polnimo dinamično. Elementi seznama so objekti razreda *ElementSeznama*, ki jih posebej sestavimo v funkciji *prikaziSeznam*. Za prikaz takih objektov potrebujemo vmesnik *ElementSeznamaAdapter*, ki je prilagojen za prikazovanje drugačne vsebine seznama, kot je le besedilo. Na Sliki 15 so naštete komponente označene z različno barvo in lahko opazimo, kakšno postavitev zavzamejo.



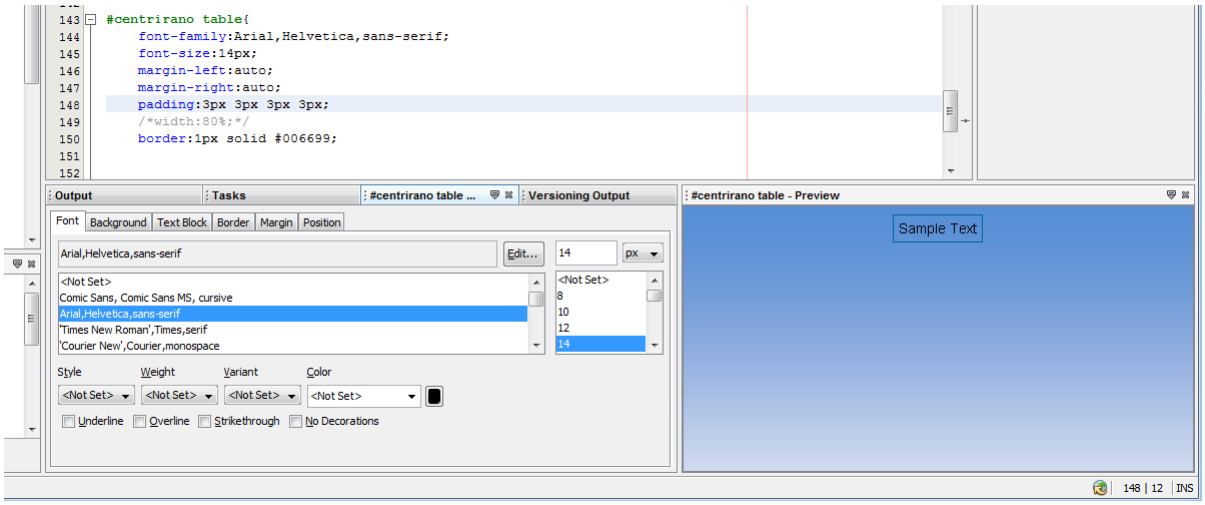
Slika 15: Postavitev komponent uporabniškega vmesnika.

Komponente so zapisane v datoteki XML, kjer je opisana njihova postavitev in lastnosti. Dokumenti XML imajo posebno obliko, razumljivo človeku in računalniku, ki temelji na preprostosti, splošnosti in uporabnosti na internetu [19].

3.5 Razvojna okolja in orodja

3.5.1 Razvojno okolje NetBeans IDE

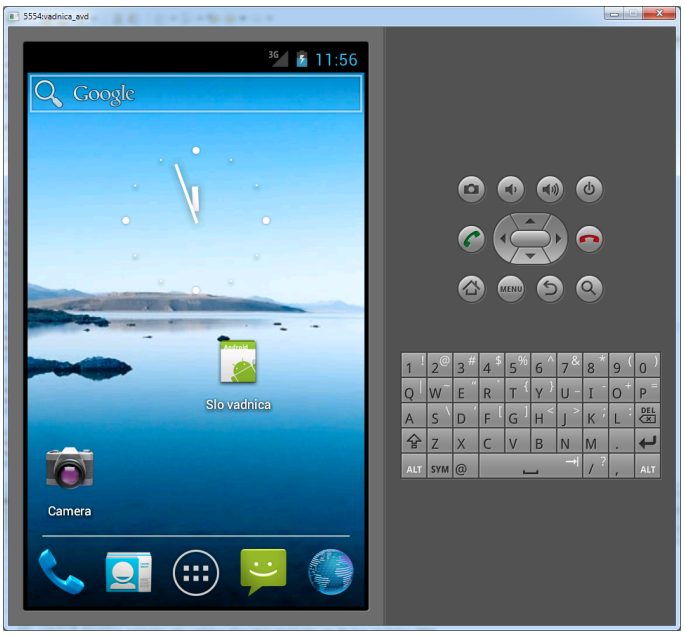
Na začetku smo želeli čim hitreje vzpostaviti spletno stran, ki bi ponujala vsebine iz podatkovne baze. Izbrali smo si orodje NetBeans IDE, ki omogoča, da projekt povežemo s strežnikom na spletu in preko protokola FTP sproti pošiljamo datoteke na pasivni način. V lastnostih projekta nastavimo zagonsko konfiguracijo, ki vključuje podrobnosti o strežniku. NetBeans IDE omogoča tudi predogled datotek CSS, kar vidimo na Sliki 16. Ob spreminjanju parametrov lahko rezultate sprememb opazujemo v okencu za predogled zato, da ni potrebno odpirati spletne strani.



Slika 16: Pogled v razvojno okolje NetBeans IDE, urejanje datotek CSS.

3.5.2 Android SDK in Eclipse IDE

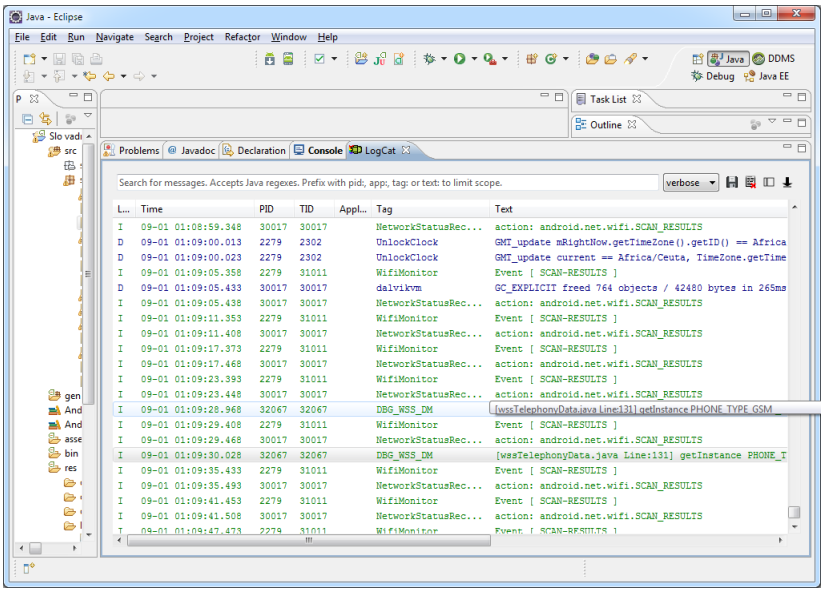
Android software development kit (SDK) vsebuje celovit paket orodij za razvoj, testiranje in razhroščevanje aplikacij Android [20]. Vključuje tudi simulator mobilne naprave, ki posnema videz in obnašanje mobilne naprave, prikazan je na Sliki 17. Uradno podprto integrirano razvojno okolje za izdelovanje aplikacij Android je Eclipse IDE, ki uporablja Android Development Tools (ADT) vtičnik [21], vendar lahko razvijalci pišejo programsko kodo tudi v drugih urejevalnikih besedil in preko komandne vrstice uporabljajo orodja za izgradnjo in razhroščevanje aplikacij Android.



Slika 17: Simulator pametnega telefona.

Žal s simulatorjem ni mogoče posnemati vseh funkcij, ki jih imajo mobilne naprave, in je tudi manj odziven. Tako smo morali za testiranje pospeškomera uporabiti pametni telefon. Uporabili smo pametni telefon z nameščenim OS Android 2.2 Froyo.

V nastavitvah pametnega telefona smo omogočili razhroščevanje USB in ga povezali z računalnikom preko USB kabla. Eclipse IDE je napravo prepoznal, aplikacijo smo lahko poganjali kar na pametnem telefonu in ne več samo na simulatorju. Hkrati smo spremljali stanje pametnega telefona in izpisovali želena sporočila za nadzor delovanja. Razširjeno polje za prikazovanje takih sporočil je vidno na Sliki 18.



Slika 18: Eclipse IDE z razširjenim poljem za prikaz sporočil.

S pomočjo Eclipse IDE lahko pri ustvarjanju grafičnega vmesnika za aplikacijo uporabimo način povleci in spusti (ang. drag and drop), kjer komponente povlečemo iz nabora vseh komponent ob strani in jih spustimo na želeno mesto. V ozadju se samodejno generira dokument XML. Del vsebine datoteke XML je viden na Sliki 19.

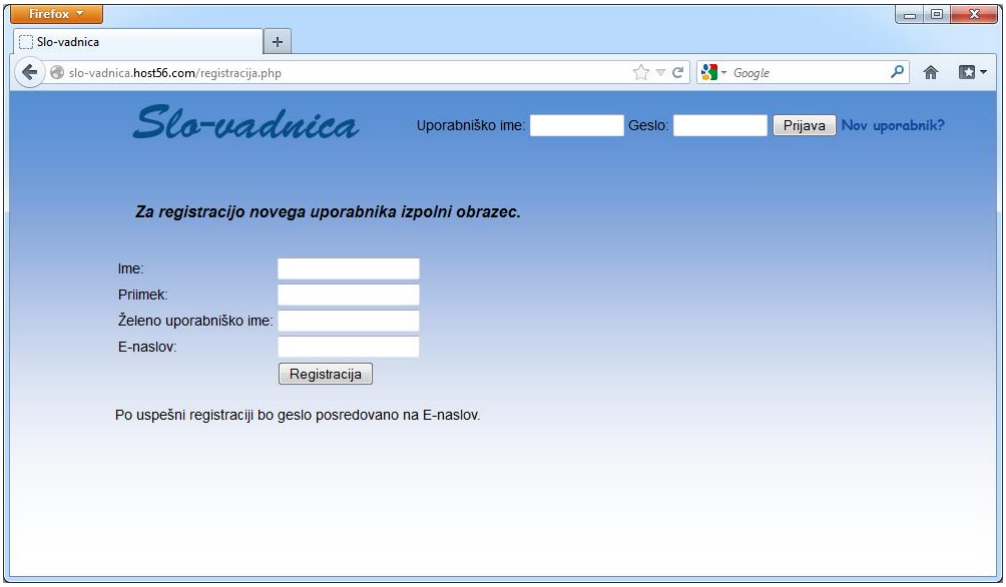
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/slovadnica_ozadje">

    <TextView android:id="@+id/txtTitle1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="18dp"
        android:textStyle="bold"
        android:background="@color/slovadnica_ozadje"/>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <TextView android:id="@+id/txt1"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
```

Slika 19: Del vsebine datoteke XML.

4 DELOVANJE SISTEMA

4.1 Registracija in prijava v sistem

Učenec, ki želi uporabljati sistem, se mora registrirati na spletni strani. Tam izpolni obrazec, ki zahteva vpis e-naslova, na katerega sistem pošlje e-sporočilo z naključno generiranim geslom.
Ko se učenec prijavi, ima možnost zamenjati geslo v nastavitvah. Da se delno izognemo zlorabam in preobremenitvi, sistem dopušča le določeno število registracij za posamezen naslov IP in le eno registracijo za posamezen e-naslov. Ob registraciji sistem tudi preveri, če želeno uporabniško ime že obstaja, in učenca o tem obvesti. Na Sliki 20 je prikazan obrazec za registracijo.

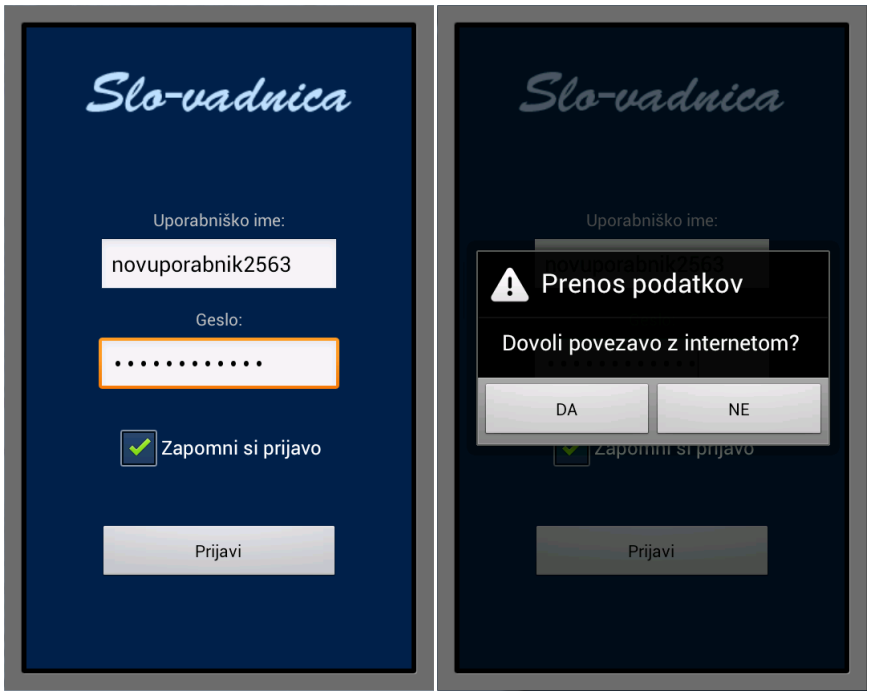


Slika 20: Obrazec za registracijo uporabnika na spletni strani.

4.2 Prijava v aplikacijo

Ko zaženemo aplikacijo Slo-vadnica, se prikaže uporabniški vmesnik prijavne aktivnosti. V vnosni polji je potrebno vpisati uporabniško ime in geslo. Na voljo imamo potrditveni gumb, ki ga lahko označimo, če želimo, da si aplikacija zapomni na zadnje prijavljenega uporabnika, nato lahko kliknemo na gumb Prijavi.
Ob kliku na gumb Prijavi aplikacija preveri, če uporabniško ime in geslo obstajata v lokalni podatkovni bazi, sicer nas vpraša za dovoljenje za poizvedbo preko interneta. Če to odobrimo, se v asinhroni nalogi *(AsyncTask)* izvede funkcija za preverjanje uporabniških imen v podatkovni bazi na internetu. Če obstaja ta kombinacija uporabniškega imena in gesla tudi tam, jo shranimo v lokalno podatkovno bazo. Na Sliki 21 je na levi prikazan prijavni obrazec,

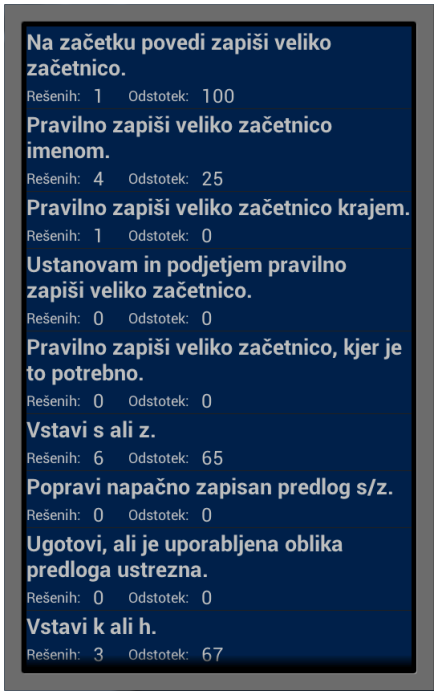
s katerim se prijavimo v aplikacijo Slo-vadnica, in na desni opozorilno okno, ki nas vpraša, ali aplikaciji dovolimo povezavo z internetom.



Slika 21: Uporabniški vmesnik prijavne aktivnosti (levo) in opozorilno okno (desno).

4.3 Prikaz nalog

Po uspešni prijavi se prikaže seznam vrst nalog in rezultatov, kot je vidno na Sliki 22.



Slika 22: Seznam vseh vrst nalog in rezultatov.

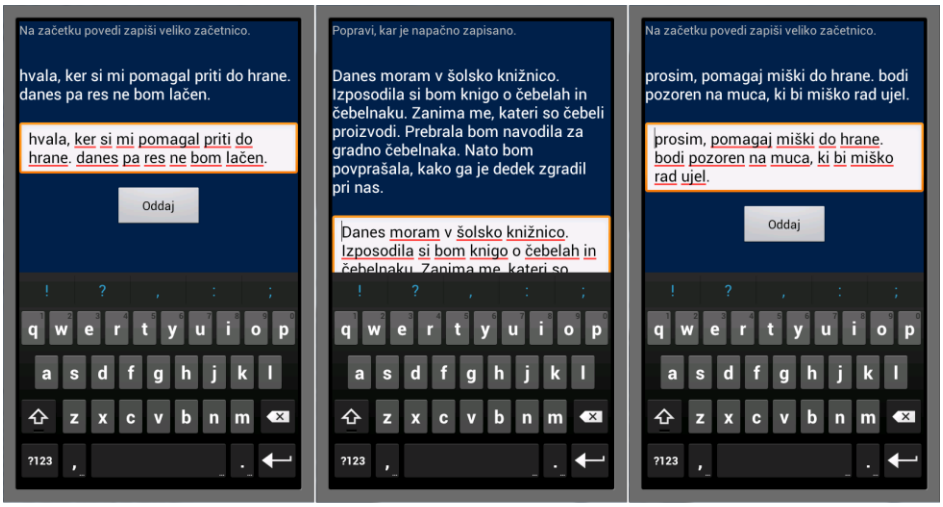
Na njem so vse vrste nalog in rezultati učenca pri reševanju. V tej aktivnosti lahko preko menijskega gumba vsebine posodobimo, kar je podrobneje opisano v nadaljevanju. Seznam je vertikalno premičen, z dotikom na posamezno nalogo pa zaženemo nalogi primerno aktivnost.

Na voljo so trije načini reševanja nalog. Učenec jih prepozna po ključnih besedah v navodilih. Za naloge tipa zapiši, popravi in morebitne nove tipe nalog se zažene aktivnost, ki prikaže besedilno polje. Za naloge tipa ugotovi se zažene aktivnost, ki z uporabo pospeškomera izrisuje vsebino na zaslonu. Naloge tipa vstavi, dopolni ali izberi pa zaženejo aktivnost, kjer mora uporabnik na določenih mestih vsebino menjati z dotikom.

4.4 Reševanje nalog

4.4.1 Naloge z vnosnim poljem

Na Sliki 23 so prikazani trije primeri reševanja nalog z vnosnim poljem. Uporabniku se prikažeta naloga in vnosno polje, kamor lahko vpiše odgovor. Preverja se cel odgovor, zato je lahko le-ta pravilen ali nepravilen. To mora učitelj ob izdelovanju novih nalog upoštevati in temu primerno oblikovati nalogo. Ta način reševanja je primernejši za krajše naloge.



Slika 23: Primeri reševanja nalog z vnosnim poljem.

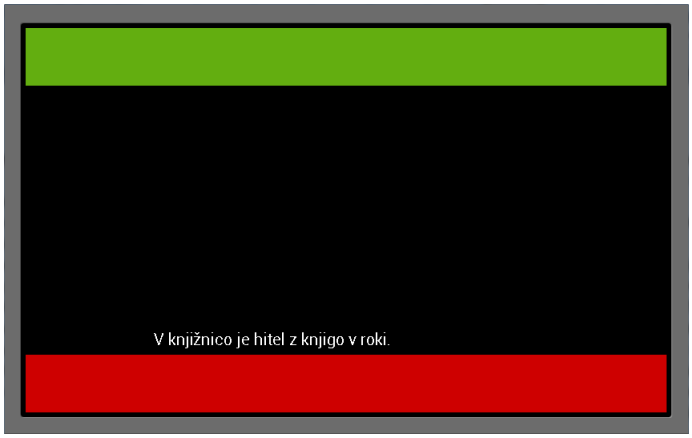
Naloge so izbrane naključno iz množice vseh nalog določene vrste. Ko zaključimo z reševanjem, se točke za pravilne odgovore prištejejo k skupnemu rezultatu. Aktivnost v aplikaciji Slo-vadnica se zaključi in spet se prikaže seznam vrst nalog.

Bistveno lažje je tako vrsto naloge reševati na spletni strani, saj virtualna tipkovnica uporabnikom na mobilni napravi včasih povzroča težave. Na Sliki 24 je prikazan primer reševanja nalog z vnosnim poljem na spletni strani.

Slika 24: Prikaz nalog z vnosnim poljem na spletni strani.

4.4.2 Naloge z uporabo pospeškomera

Z izbiro naloge tipa ugotovi v aplikaciji Slo-vadnica zaženemo novo aktivnost in začnemo brati podatke o pospeških, ki delujejo na napravo. Pri tem pomagajo funkcije razreda *SensorEventListener*. Na zaslon začnemo prikazovati komponente s pomočjo razreda *SimulacijaView*. Vsakič ko zaznamo spremembo pospeška, si vrednost zapomnimo in jo po obdelavi prištejemo h koordinati, ki opisuje pozicijo besedilne komponente na zaslonu. Vrednost obdelamo zato, da dosežemo manjšo občutljivost na nagib. Najprej jo razpolovimo, nato pa preverimo, ali je dovolj velika. Zelo majhno vrednost nastavimo na nič in zato ne vpliva na pozicijo točke na zaslonu. Pozicija besedilne komponente je točka, kamor izrisujemo besedilo in tako vemo, kje se nahaja. Slika 25 prikazuje način reševanja take vrste naloge v aplikaciji Slo-vadnica.



Slika 25: Primer reševanja naloge s pomočjo pospeškomera.

Aktivnost izbira naključne naloge brez ponavljanja in si njihove identifikacijske številke zapisuje v tabelo. Naloge se prikazujejo zaporedno, cilj naloge pa temelji na ugotovitvi, ali je primer napisan pravilno ali nepravilno. Če uporabnik ugotovi, da je zapisan pravilno, ga z nagibanjem naprave usmeri nad zeleno polje, sicer pa nad rdeče. Po nekaj trenutkih se odločitev uporabnika zapiše med odgovore in prikaže se sporočilo o pravilnosti odgovora. Da smo ugotovili, koliko časa je odgovor nad določenim poljem, smo uporabili primerjanje sistemskih časov, ki smo si jih ob določenih trenutkih zapomnili s pomočjo razreda *System* in funkcije *nanoTime*.

S pomočjo menijskega gumba na napravi prikažemo meni, kjer imamo na voljo ročno nastavitev orientacije, možnost kalibracije merilnika pospeškov in izhod iz aktivnosti. Kalibracija je namenjena nastavitvi vrednosti pospeškov na X in Y osi na nič v stanju, ko je naprava položena na ravno podlago. V tem trenutku si zapomnimo vrednosti, ki jih upoštevamo kasneje pri obdelovanju.

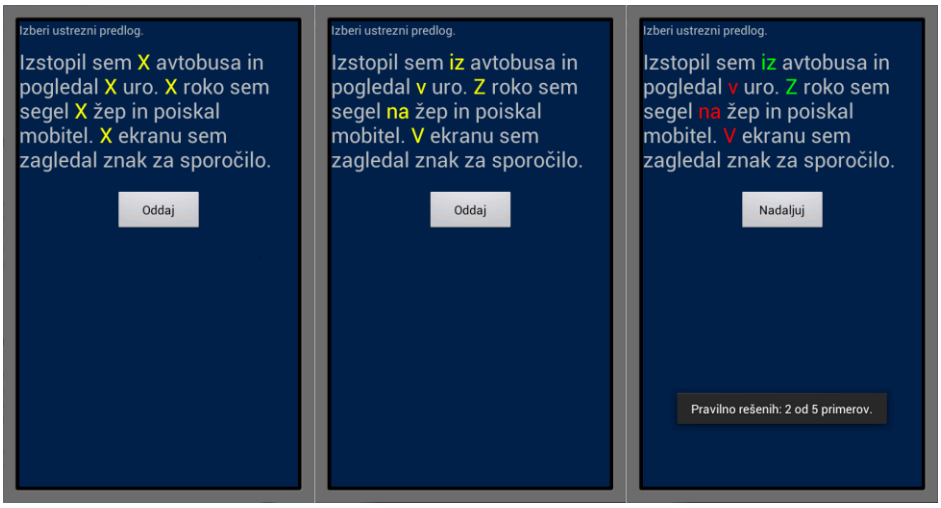
Ko spreminjamo orientacijo zaslona, je potrebno shranjevati vrednosti spremenljivk v aktivnosti. S pomočjo metode *onSaveInstanceState* shranimo identifikacijsko številko naloge, ki je prikazana, in vse ostale, ki še čakajo na prikaz. Če učenec rešuje nalogo več časa in se med tem ni dotaknil zaslona ali pritisnil kake tipke, se zaslon lahko ugasne. Uporabniku to ne bi bilo všeč, zato ta aktivnost obdrži zaslon prižgan s pomočjo razreda *WakeLock*.

Take vrste nalog rešujemo na spletni strani nekoliko drugače. Na Sliki 26 je prikazan primer. Tu se prikaže več nalog hkrati, potrebno je označiti, ali je primer ustrezen ali ne. Naloge lahko sicer rešujemo hitreje, a ne na tako zanimiv način kot v aplikaciji Slo-vadnica.

Slika 26: Primer reševanja naloge tipa ugotovi na spletni strani.

4.4.3 Naloge z vstavljanjem

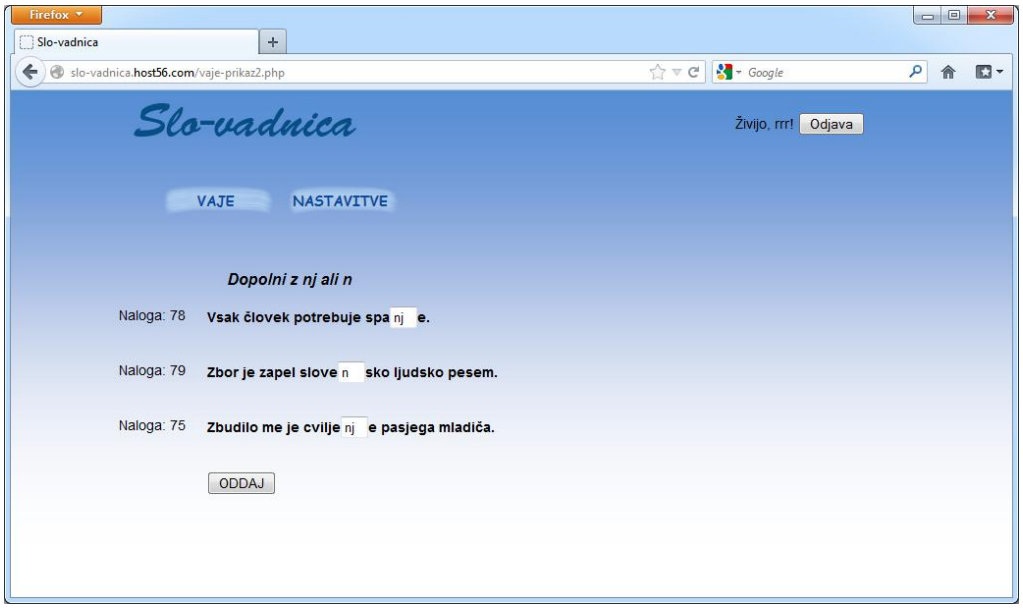
Tretji način reševanja nalog temelji na menjavi vsebine na določenih mestih v nalogi. Ta mesta določi učitelj, ko vpisuje novo nalogo. Tam, kjer zapiše X, se med besedilom pojavi polje, ki je občutljivo na dotik. Ta vrsta naloge zažene svojo aktivnost, v kateri prikažemo besedilo, ki ima interaktivne lastnosti. Mesta, kjer se pojavi X, so na začetku obarvana rumeno. Vsebino na tem mestu je potrebno z dotikom menjati, dokler ni pravilna. Ko učenec zaključi s spreminjanjem vsebin, pritisne na gumb Oddaj. Takrat se odgovor zabeleži v podatkovno bazo in se primerja s pravilnim.
Odgovori teh vrst nalog so zapisani v eno besedilno polje brez presledkov in ločeni s podpičjem. Tako strukturo lahko hitro obdelamo v tabelo in v zanki preverjamo vsebino na vseh mestih v tabeli. Ko ugotovimo, na katerih mestih se je uporabnik zmotil, lahko ta mesta obarvamo rdeče, ostala obarvamo zeleno. Prikaže se tudi izpis, koliko primerov je uporabnik rešil pravilno. Ko si ogleda rezultat, lahko pritisne na gumb Nadaljuj. Če imamo več nalog, se prikaže nova, sicer se aktivnost zaključi. Na Sliki 27 je prikazan primer reševanja take vrste naloge.



Slika 27: Primer reševanja naloge s polji na dotik.

Aktivnost izbere naključne naloge take vrste brez ponavljanja. Če je učitelj določil, da jih je potrebno rešiti več hkrati, se identifikacijske številke nalog shranijo v tabelo in jih prikazujemo zaporedno. Ko učitelj na spletni strani ustvari novo vrsto naloge, določi tudi število nalog, ki se bodo zaporedno prikazovale. Ko dodaja novo nalogo te vrste, mora v prvo besedilno polje vpisati besedilo naloge in postaviti X tja, kamor želi, da učenec nekaj vpiše oziroma popravi. Vsebine, ki naj bi bile na mestih X, pa ločene s podpičji vpiše v drugo besedilno polje.
Tudi to vrsto nalog rešujemo na spletni strani nekoliko drugače. Tam, kjer bi bilo v aplikaciji Slo-vadnica polje občutljivo na dotik, se na spletni strani pojavi vnosno polje, v katerega

mora učenec vpisati vsebino s tipkovnico. Primer takega načina reševanja na spletni strani je prikazan na Sliki 28.



Slika 28: Primer reševanja nalog z vstavljanjem na spletni strani.

4.5 Beleženje odgovorov in rezultatov

Pri vsaki nalogi, ki jo učenec reši, se v podatkovno bazo zapiše nova vrstica v tabelo odgovorov, ki vsebuje unikatno številko učenca in naloge, odgovor in število točk za ta odgovor. Točke so shranjene v obliki decimalnega števila. S pomočjo povpraševanj SQL dobimo podatke za posameznega učenca: koliko nalog določene vrste je rešil in kako uspešen je bil; rezultat je izražen v odstotkih.

Ko aplikacijo posodobimo, se združijo odgovori zabeleženi na aplikaciji Android in tisti zabeleženi na spletni strani. Dobimo skupno število rešenih nalog in točk za učenca za vsako vrsto naloge; te podatke shranimo v tabelo rezultatov. Takrat pobrišemo tudi lokalne odgovore učenca na aplikaciji Android.

4.6 Posodabljanje vsebin

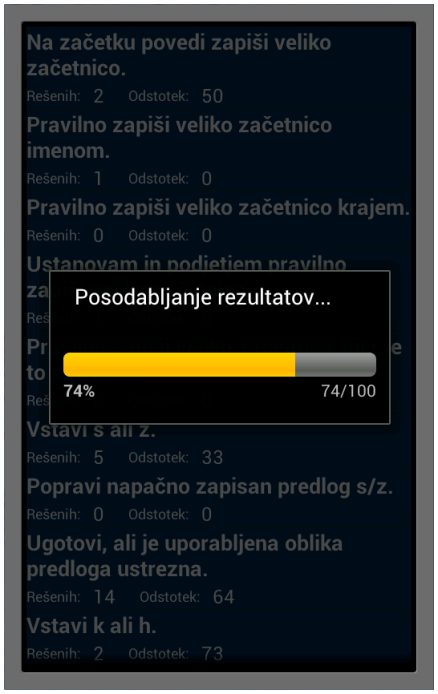
Z idejo, da bi aplikacija uporabljala lokalno podatkovno bazo, je nastala težava, kako poskrbeti za sveže vsebine. V aktivnosti, ki prikazuje seznam nalog in rezultatov, se ob pritisku na menijski gumb mobilne naprave prikaže kontekstni meni.

Če izberemo Posodobitev, se prikaže opozorilno okno, ki nas vpraša, če dovolimo aplikaciji dostop do interneta. Po odobritvi se prične postopek posodobitve v treh korakih: preverjanje stanja omrežne povezave, posodobitev nalog ter posodobitev rezultatov učenca. Preden aplikacija poskuša dostopati do strežniške podatkovne baze, je potrebno preveriti, ali je

omrežna povezava sploh na voljo. Včasih se zgodi, da mobilna naprava ni v dosegu nobenega omrežja, ali pa je uporabnik onemogočil brezžično povezavo in podatkovni prenos preko mobilnega omrežja. Za preverjanje stanja omrežne povezave uporabimo razred *ConnectivityManager*.
Ko je povezava preverjena, lahko s posodobitvijo nadaljujemo.

Najprej dobimo podatek, katera je zadnja posodobitev v lokalni podatkovni bazi. V asinhroni nalogi nato poženemo funkcijo, ki pošlje podatke o posodobitvi na strežnik. Na strežniku je v ta namen napisana skripta PHP, ki čaka na poizvedbe s strani aplikacije Android. Skripta s pomočjo poizvedb SQL komunicira s podatkovno bazo na strežniku. Ko dobi odgovor, ga preoblikuje in vrne v obliki besedila, aplikacija Android pa to besedilo obdela in izdela objekte tipa *VrstaNaloge* in *Naloga*, ki jih vpiše v lokalno podatkovno bazo. V odgovoru s strežnika so zapisane tudi vse identifikacijske številke nalog in vrste nalog, da lahko iz lokalne podatkovne baze pobrišemo tiste primere, ki jih na strežniški bazi ni.

V tretjem koraku posodobimo še odgovore učenca, ki jih v obliki JSON v asinhroni nalogi pošljemo skripti PHP. Ta skripta je zadolžena za preverjanje rezultatov tega učenca na spletni strani, seštevanje obojih rezultatov, preoblikovanje skupnih rezultatov ponovno v obliko JSON in pošiljanje odgovora odjemalcu. V podatkovni bazi aplikacije popravimo vsebino tabele Rezultati. Med posodabljanjem se prikaže vrstica napredka, kot je vidno na Sliki 29.



Slika 29: Prikaz vrstice napredka med posodabljanjem.

5 SKLEPNE UGOTOVITVE

Cilj diplomske naloge je dosežen z izdelano aplikacijo za izvajanje pravopisnih vaj na mobilni platformi Android. Vaje prikazujemo na več načinov, tisti z uporabo merilnika pospeškov je za platformo specifičen. Naloge so shranjene v podatkovni bazi na strežniku, ob posodobitvi vsebin preko interneta pa jih shranimo tudi na lokalno podatkovno bazo, kar omogoča, da vaje izvajamo brez uporabe interneta. Za izdelavo in popravljanje nalog je preko spletne strani omogočen dostop učiteljem. Ta se nahaja na istem strežniku kot podatkovna baza. Tam se nahaja tudi spletna stran za učence; namenjena je registraciji novih učencev in reševanju vaj preko spletnega brskalnika.

Med izdelovanjem aplikacije smo prišli do novih idej in spoznanj, kje bi bilo potrebno aplikacijo še nadgraditi ali popraviti. Nekaj slabosti je povezanih s posodobitvami vsebin preko interneta, te se namreč prenašajo nezaščiteno. Poleg tega bi morali ločiti posodobitev nalog od posodobitve rezultatov učenca. Verjetno se bo večkrat zgodilo, da bo učenec želel posodobiti svoje rezultate, kot pa da bo učitelj dodal ali popravil kakšno nalogo. Zato bi morali učencu omogočiti izbiro vsebin, ki se bodo posodobile. Pri urejanju nalog na strežniku bi morali beležiti brisane naloge, da pri posodobitvi aplikacije ni potrebno prenašati celega seznama identifikacijskih številk, ampak le tiste, ki so bile brisane.

Učenec bi moral imeti dober pregled nad vrstami nalog, ki so sedaj nanizane ena za drugo in jih lahko loči le po navodilu naloge. S tem izgubi precej časa, če želi vaditi le točno določeno vrsto naloge. Vsaj nekoliko bi morali biti ločeni tudi načini reševanja nalog. Označeni bi lahko bili z ikono pred besedilom naloge. Zanimivejše načine reševanja nalog bi tako prej opazili.

Nekatera spoznanja smo upoštevali že med razvojem. Ugotovili pa smo, da se vseh idej ne da realizirati, saj se z vsako realizacijo ideje porodi vsaj še ena nova. Ena takih idej je preskok aktivnosti prijave. Delno smo jo rešili z uvedbo potrditvenega gumba, ki si zadnje prijavljenega učenca zapomni.

Uvedli bi lahko reševanje domačih nalog in se na tem področju približali razvitim sistemom, kot je e-učilnica. Učitelj bi na spletni strani izdelal šablono domače naloge, ki bi se za vsakega učenca napolnila z naključnimi nalogami. Rezultate reševanja takih nalog bi lahko učitelj spremljal ali celo ocenjeval s pomočjo spletne strani.

Ena zanimivejših idej za izboljšavo je vključevanje multimedijskih vsebin. Za pomoč pri reševanju bi vsaka vrsta naloge imela nekaj teorije in natančna navodila. To bi lahko dosegli z animacijami ali z video posnetki, ki bi jih na željo uporabnika predvajali pred začetkom

reševanja naloge. Slabost tega bi bilo povečanje prostorske zahtevnosti aplikacije, saj bi te vsebine morale biti shranjene na mobilni napravi. Multimedijske vsebine pritegnejo pozornost in povečajo zmožnosti predstavitve vsebin. Dodali bi lahko nove načine reševanja nalog, kjer bi si učenci na primer najprej ogledali sliko, nato pa bi naloga zahtevala odgovore povezane z sliko.

Namen diplomske naloge seveda ni bil razviti celotnega sistema do potankosti, ampak zgolj predstaviti nov, zanimivejši način učenja. Razvoj aplikacije za mobilno platformo Android je bila dobra odločitev, saj se platforma uspešno razvija in ima po svetu vedno več uporabnikov in razvijalcev. Izkoriščanje mobilnih naprav v izobraževalne namene bodo podprli učitelji, učenci in njihovi starši.

VIRI

[1] (2012) Downtime Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Downtime
[2] (2012) MySQL Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/MySQL
[3] (2012) SQL Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/SQL
[4] (2012) phpMyAdmin. Dostopno na:
www.phpmyadmin.net/
[5] (2012) MD5 Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/MD5
[6] (2012) Collision resistance Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Collision_resistant
[7] (2012) MD5 Decrypter. Dostopno na:
http://md5decrypter.co.uk/
[8] (2012) HTML Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/HTML
[9] (2012) PHP. Dostopno na:
http://www.php.net/
[10] (2012) File Transfer Protocol Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/File_Transfer_Protocol
[11] (2012) The difference between Active and Passive FTP modes. Dostopno na:
http://www.serverintellect.com/support/ftp/ftp-active-passive-diff.aspx
[12] (2012) Android (operating system) Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Android_%28operating_system%29
[13] (2012) HTC Dream Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/HTC_Dream
[14] (2012) Android. Dostopno na:
http://www.android.com/
[15] (2012) Activity. Dostopno na:
http://developer.android.com/reference/android/app/Activity.html
[16] (2012) Application Fundamentals. Dostopno na:
http://developer.android.com/guide/components/fundamentals.html
[17] (2012) AsyncTask. Dostopno na:
http://developer.android.com/reference/android/os/AsyncTask.html
[18] (2012) SensorEventListener. Dostopno na:
http://developer.android.com/reference/android/hardware/SensorEventListener.html
[19] (2012) XML Wikipedia. Dostopno na:

http://en.wikipedia.org/wiki/XML
[20] (2012) Exploring the SDK. Dostopno na:
http://developer.android.com/sdk/exploring.html
[21] (2012) Android software development Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Android_software_development

KAZALO SLIK