

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3505

**MOBILNA APLIKACIJA ZA PRAĆENJE
TROŠKOVA KUĆANSTVA**

Josip Basioli

Zagreb, lipanj 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3505

**MOBILNA APLIKACIJA ZA PRAĆENJE
TROŠKOVA KUĆANSTVA**

Josip Basioli

Zagreb, lipanj 2014.

Zagreb, 10. ožujka 2014.

ZAVRŠNI ZADATAK br. 3505

Pristupnik: **Josip Basioli**
Studij: Računarstvo
Modul: Programsko inženjerstvo i informacijski sustavi

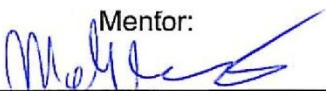
Zadatak: **Mobilna aplikacija za praćenja troškova kućanstva**

Opis zadatka:

Napraviti mobilnu aplikaciju koja će omogućiti praćenje troškova kućanstva. Oblikovati odgovarajući model relacijske baze podataka. Aplikacija mora omogućiti definiciju periodičkih prihoda (npr. plaće) i rashoda (npr. režijski troškovi, krediti), te unos troškova (npr. kupovina). Svaki član kućanstva može nezavisno unositi troškove, koji se potom sinkroniziraju s ostalim članovima kućanstva. Troškove organizirati u kategorije. Statistički obraditi i vizualizirati podatke.

Uz otisnuti primjerak rada priložiti optički disk s izgrađenom programskom potporom u izvornom i izvršnom obliku, te tekstom rada.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

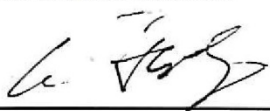
Menzor:


Doc.dr.sc. Igor Mekterović

Djelovodja:


Doc.dr.sc. Ivica Botički

Predsjednik odbora za
završni rad modula:



Prof.dr.sc. Krešimir Fertalj

Posebno se zahvaljujem mentoru Doc. dr. sc. Igoru Mekteroviću na strpljenju, poticaju, konstruktivnim sugestijama i interesu za temu

Sadržaj

1. Uvod.....	1
2. Povijest mobilnih aplikacija.....	2
3. Pregled korištenih tehnologija za razvoj aplikacije	3
3.1. Java	3
3.2. JSON.....	4
3.3. SQLite	6
3.4. PHP	6
4. Aplikacija „Household“	8
4.1. Opis aplikacije.....	8
4.2. Arhitektura i model podataka.....	9
4.3. Sinkronizacija.....	12
5. Korisničke upute za aplikaciju „Household“	15
5.1. Prvo pokretanje	15
5.2. Početni zaslon	17
5.2.1. Novi trošak	18
5.2.2. Pregled svih kućanstva	19
5.2.3. Graf troškova	20
6. Zaključak	22
7. Literatura	23
8. Sažetak.....	24
9. Ključne riječi.....	26

1. Uvod

Android aplikacija („Android app“) je naziv za mobilnu aplikaciju razvijenu za korištenje na uređajima pokretanim Google-ovom Android platformom. Te aplikacije dostupne su putem online dućana kao što su Google Play (bivši Android Market), Amazon Appstore-u i drugim stranicama fokusiranim na distribuciju Android aplikacija.

Android aplikacije se uglavnom razvijaju pomoću programskog jezika Java (postoje razvojna okruženja koja dopuštaju programiranje u nekom drugom jeziku, tipa C#, zatim u pozadini prevode kod u Javu). Do nedavno nije postojalo razvojno okruženje specijalizirano za razvoj na Android platformi, te se najčešće koristio Eclipse IDE za razvoj Android aplikacija. Od prošle godine Google je lansirao Android Studio kao razvojno okruženje za android programere koje u sebi ima ugrađene sve potrebne alate za razvoj Android aplikacija.

Aplikacija „Household“ je mobilna aplikacija koja korisnicima istog kućanstva omogućava pohranjivanje raznih troškova te izvještaje na mjesečnoj ili godišnjoj bazi uz grafove koji prikazuju udio troškova u tom razdoblju s obzirom na vrstu troška. Aplikacija se oslanja na web server koji drži podatke sinkroniziranim među više korisnika istog kućanstva. Svako kućanstvo je zaštićeno imenom i lozinkom tako da novi korisnici mogu dobiti pristup samo onom kućanstvu za koje znaju i ime i lozinku. Aplikacija je oblikovana po principu debelog klijenta da bi se server što više rasteretio tojest da može posluživati više klijenata u isto vrijeme.

2. Povijest mobilnih aplikacija

Kod prvih generacija mobilnih telefona konkurencija je bila čvrsta i proizvođači su pažljivo čuvali tajne koje se kriju iza njihovih uređaja. Zbog tog razloga aplikacije su razvijane samo unutar tvrtke. Programeri koji nisu bili dio tog kruga nisu imali prilike raditi aplikacije za mobitele.

U to vrijeme proizvođači su počeli prodavati uređaje s već instaliranim igrama (aplikacijama za razbibrigu). Prvi uspjeh je doživio proizvođač Nokia, koji je svoje uređaje počeo izdavati s već instaliranom igrom „Zmija“. Drugi su ga pratili, dodavajući „Pong“, „Tetris“ i „Križić-kružić“.

S vremenom, kupci su tražili sve više i više opcija i igara na mobitelima, ali proizvođači nisu imali motivacije ni resursa da naprave svaku aplikaciju koju su kupci poželjeli. Trebali su neki način da omoguće servise za zabavu i informacije bez dozvoljavanja direktnog pristupa mobitelu.

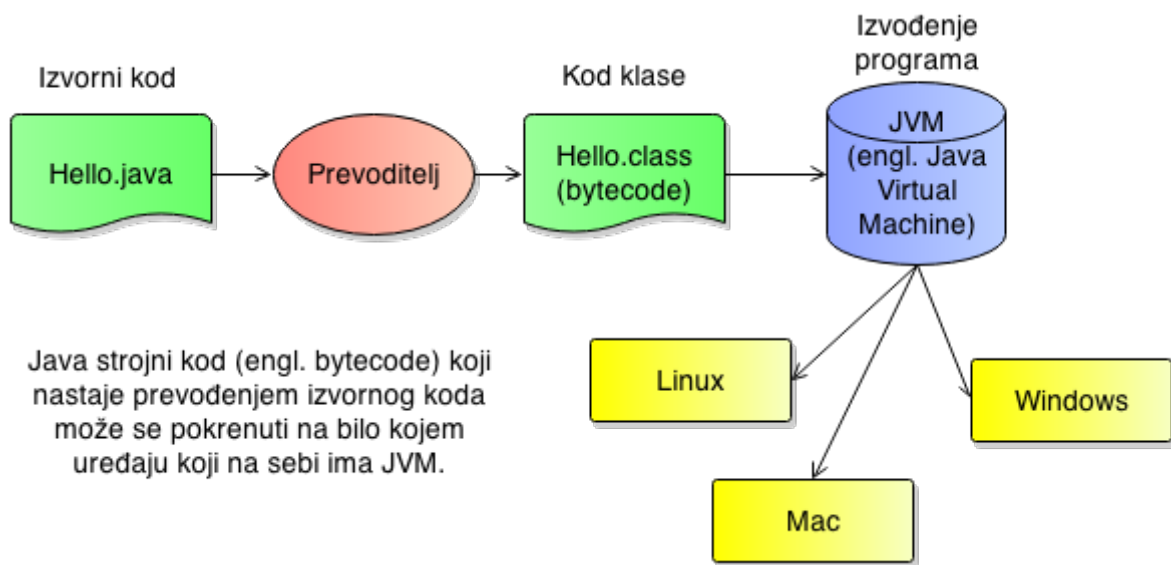
Proizvođači su shvatili da će morati odati neke tajne javnosti da bi razvojni programeri neovisni od njihove tvrtke mogli razvijati aplikacije za uređaj. Time će korisnici imati eksponencijalno veću količinu dostupnih aplikacija te će biti zadovoljniji s uređajem. Zbog takve ideje s vremenom su se počeli razvijati operacijski sustavi za mobilne telefone kao Symbian, Windows Phone, iOS, Android.

3. Pregled korištenih tehnologija za razvoj aplikacije

Razvoj android aplikacije bio bi teško ostvariv bez podrške razvojnog okruženja i popratnih alata. Do nedavno Eclipse IDE je bio programerski prvi izbor, ali s obzirom na to da je Android Studio napravljen isključivo za android aplikacije, odabrao sam tu razvojnu okolinu. S obzirom na to da aplikacija mora komunicirati sa serverom, odlučio sam koristiti Apache server, SQLite bazu podataka, te u PHP-u obrađivati JSON podatke i upravljati bazom podataka.

3.1. Java

Java je objektno orijentirani programski jezik temeljen na klasama, koji može paralelno obavljati više dijelova programa gdje čovjeku nije vidljivo da se ti dijelovi zapravo izvode slijedno (konkurentnost). Ta funkcionalnost je omogućena uz pomoć više procesa i niti (threadova), ali i pomoću ključne riječi koja omogućava da se određeni dio programa može izvoditi samo od strane jednog procesa u isto vrijeme. Kod napisan u Javi na jednoj platformi ne treba biti ponovno preveden na nekoj drugoj platformi. Razlog tome je što se Java programi prevode u *bytecode* koji se može pokrenuti na bilo kojem Java virtual machine-u (JVM), neovisno o arhitekturi računala kao što pokazuje slika 3-1. Od 2014. godine Java je jedan od najpopularnijih programskih jezika u uporabi.



Slika 3-1 Prevođenje Java programa

Google je odlučio koristiti Javu kao ključni stup u stvaranju Android operacijskog sustava. Iako je Android OS većinom napisan u C-u, Android SDK (Software development kit) koristi Javu kao osnovu za Android aplikacije.

3.2. JSON

JSON (engl. *JavaScript Object Notation*) je format koji koristi tekst čitljiv ljudima za prijenos podatkovnih objekata koje sačinjavaju parovi atribut-vrijednost. Primarno se koristi za prijenos podataka između servera i web aplikacije, kao alternativa XML-u.

JSON je podatkovni format neovisan o programskom jeziku, te je kod za parsiranje i generiranje JSON podataka dostupan u većini programskih jezika (kao i u Javi i PHP-u).

Osnovni tipovi JSON podataka:

Number – JSON ne prepoznaje razliku između integer-a i float-a, te može koristiti eksponencijalnu notaciju (E).

String – slijed nula ili više znakova. Stringovi se označavaju dvostrukim navodnicima

Boolean – true ili false

Array – uređena lista od nula ili više vrijednosti, od kojih svaka vrijednost može biti bilo kojeg tipa. Array koristi notaciju uglatih zagrada, a elementi su razdvojeni zarezom.

Object – neuređen asocijativni niz (parovi ime/vrijednost). Objekti se označavaju vitičastim zagradama te koriste zarez za razdvajanje svakog para, dok u svakom paru dvotočka razdvaja ključ ili ime od vrijednosti. Svi ključevi moraju biti stringovi te različiti unutar jednog objekta

Null – prazna vrijednost

U nastavku je primjer JSON-a.

```
{ "menu": {  
  "header": "SVG Viewer",  
  "items": [  
    { "id": "Open" },  
    { "id": "OpenNew", "label": "Open New" },  
    null,  
    { "id": "ZoomIn", "label": "Zoom In" },  
    { "id": "ZoomOut", "label": "Zoom Out" },  
    { "id": "OriginalView", "label": "Original View" },  
    null,  
    { "id": "Quality" },  
    { "id": "Pause" },  
    { "id": "Mute" },  
    null,  
    { "id": "Find", "label": "Find..." },  
    { "id": "FindAgain", "label": "Find Again" },  
    { "id": "Copy" },  
    { "id": "CopyAgain", "label": "Copy Again" },  
    { "id": "CopySVG", "label": "Copy SVG" },  
    { "id": "ViewSVG", "label": "View SVG" },  
    { "id": "ViewSource", "label": "View Source" },  
    { "id": "SaveAs", "label": "Save As" },  
    null,  
    { "id": "Help" },  
    { "id": "About", "label": "About Adobe CVG Viewer..." }  
  ]  
}}
```

3.3. SQLite

SQLite je SUBP (sustav za upravljanje bazama podataka) sadržan u C programskoj biblioteci. SQLite je u skladu s ACID-om (Atomarnost, Konzistentnost, Izolacija, Izdržljivost) te implementira većinu SQL standarda. SQLite dinamički određuje tipove, tj ne dodjeljuje određeni tip stupcu već se svakoj vrijednosti pridružuje svoj tip što je neobično za baze podataka. Još jedna iznimka su „slabi“ tipovi podataka što znači da se primjerice u integer stupac može unijeti string podatak što može utjecati na integritet domene.

SQLite je popularan izbor kao integrirana baza podataka za lokalnu pohranu u aplikacijama kao što su web preglednici.

SQLite samo donekle podržava okidače (*triggers*) i poglede (*views*). Pogledi se mogu stvarati, ali su samo za čitanje (read-only) tj, ne može se pisati ili brisati iz njih. Također ne podržava neke ALTER TABLE naredbe kao što su brisanje ili izmjena stupaca.

3.4. PHP

PHP (Hypertext preprocessor) je skriptni jezik koji se izvršava na poslužiteljskom kraju. Dizajniran je za razvoj web-a, ali se također koristi kao programski jezik opće svrhe. PHP kod se može miješati s HTML kodom koristeći posebne znakove. Zbog tog razloga web stranice mogu imati dodatne funkcionalnosti koje HTML sam ne može izvesti kao što je interakcija s bazom podataka ili izvođenje raznih proračuna (npr. koji je današnji dan, matematičke jednadžbe). U nastavku, na slici 3-2 je primjer HTML-a koji sadrži u sebi PHP, a predstavlja login formu.

```

1 <!-- primjer HTML-a isprepletenog s PHP-om koji ostvaruje formu za unos korisničkih podataka -->
2 <?php
3     if(isset($_POST['BtnSubmit']))
4     { //dinamički ispis php-om na stranici o uspjehu slanja forme
5         echo "<h3>Form Submitted Successfully!! </h3>";
6         echo "Submitted Data Array As Bellow: <br/>";
7         print_r($_POST); //ispis samih podataka s forme
8     }
9     ?>
10
11 <h2>HTML Login Form Example in PHP</h2>
12 <!-- HTML forma -->
13 <form name="LoginForm" method="POST" action="#">
14     Username :
15     <!-- vrijednost dobivena pomoću PHP-a -->
16     <input name="Username" type="text" value="<?php echo $_POST['Username']; ?>"><br/><br/>
17     Password :
18     <input name="Password" type="password" value="<?php echo $_POST['Password']; ?>"><br/><br/>
19     <input name="BtnSubmit" type="submit" value="Login">
20 </form>

```

Slika 3-2 PHP primjer

4. Aplikacija „Household“

4.1. Opis aplikacije

Namjena aplikacije „Household“ praćenje troškova u kućanstvu. Aplikacija omogućava unos proizvoljnog broja kućanstva te svako kućanstvo štiti sa zadanom lozinkom. Definiraju se dvije lozinke, jedna za pristup podacima o kućanstvu, druga za administratora kućanstva odnosno mijenjanje postavki kućanstva. Lozinku za pristup koristi član zajedničkog kućanstva koji nije unio kućanstvo tako da može povući podatke s poslužitelja.

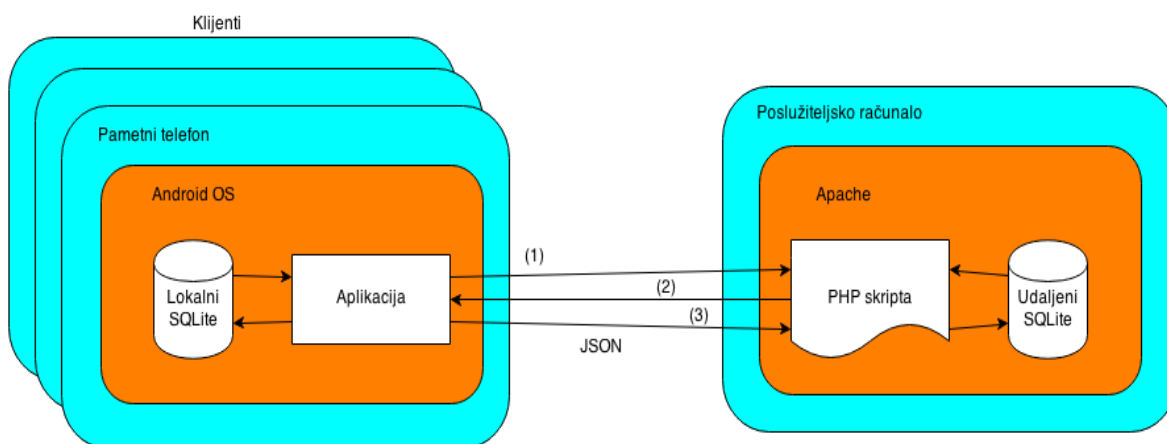
U postavkama kućanstva korisnik može odrediti koje vrste troškova želi unositi u aplikaciju, a koje troškove nema ili ga ne zanimaju. To su opcije koje se prvi put zadaju pri unosu kućanstva.

Korisnik može dodavati troškove kućanstva i to samo one koje je odabrao u postavkama. Također može zatražiti prikaz grafova koji prikazuju udio pojedine vrste troška u odabranom vremenskom razdoblju za neku skupinu troškova.

S obzirom na to da aplikacija predviđa više korisnika istog kućanstva, postoji problem sinkronizacije podataka. Zbog tog razloga aplikacija se oslanja na server, na kojem se nalazi baza podataka pomoću koje aplikacija održava lokalne podatke i podatke na serveru ažurnima.

4.2. Arhitektura i model podataka

Aplikacija se sastoji od klijenskog dijela, tj. same mobilne aplikacije te SQLite baze podataka na serveru i pripadnih php skripti. Arhitektura je prikazana na slici 4-1.



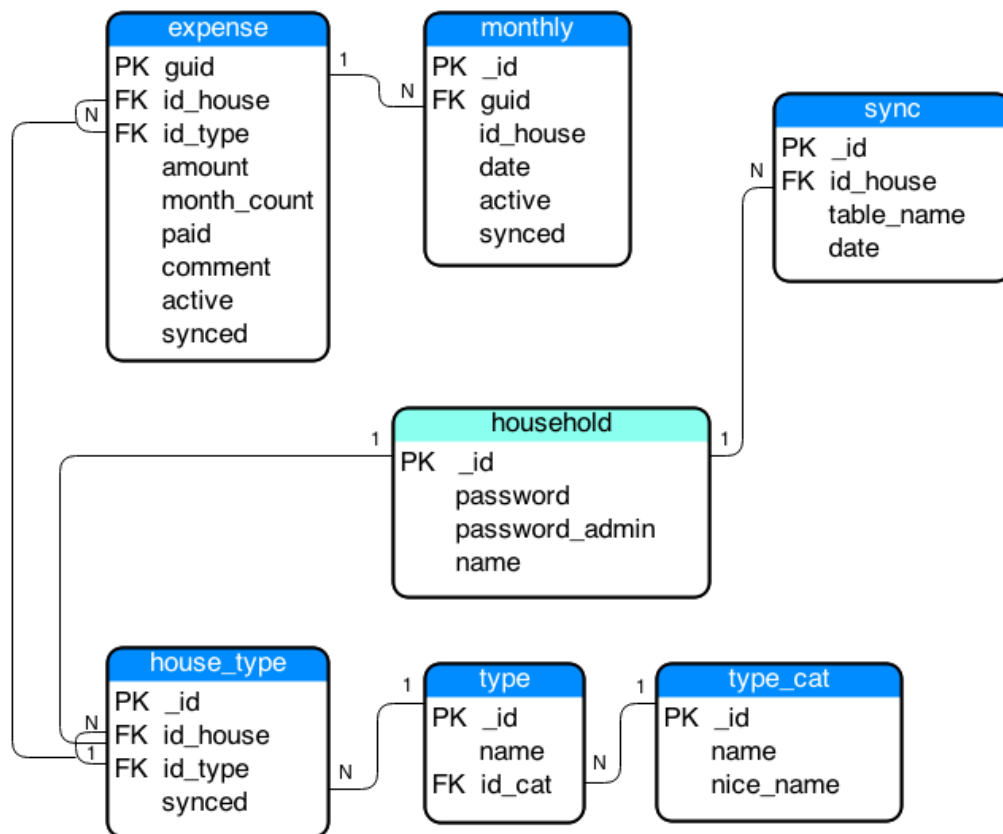
Slika 4-1 Arhitektura sustava

Poslužitelj služi za komunikaciju klijentskog dijela s bazom podataka, točnije za sinkronizaciju između klijentske i poslužiteljske baze podataka. Kad aplikacija želi obaviti sinkronizaciju, ona pošalje serveru podatke o posljednjem vremenu sinkronizacije (1) koji tada vrši upit nad bazom podataka i s obzirom na rezultat upita vraća aplikaciji odgovarajuću poruku (2). Ako nema novih promjena, u poruci šalje status koji opisuje da nije potrebna sinkronizacija, a ako ima promjene tada server u poruci šalje redove iz tablica koji su noviji od vremena koje je poslala aplikacija. Konačno aplikacija šalje serveru podatke koji su postojali samo lokalno na aplikaciji (3). Za komunikaciju s bazom, na serveru se koristi SQLite3 klasa programskog jezika php. Pomoću te klase pojednostavljuje se spajanje na bazu i izvršavanje SQL upita, a također se znatno povećava sigurnost od napada SQL injekcijom.

Klijent je pisan u programskom jeziku Java, a baza podataka je također SQLite. U ovom slučaju upiti nad bazom podataka se vrše pomoću klasa u Javi koje također omogućuju pojednostavljenu komunikaciju s bazom. Klijent je dizajniran da obavlja što više posla, tako da server bude što rasterećeniji. Cilj

takvog dizajna jest omogućiti serveru konkurentni rad sa što više klijenata. Osim od baze podataka, klijentska aplikacija se sastoji od nekoliko aktivnosti (engl. „Activity“) koje služe za obavljanje funkcionalnosti programa, a većina je opisana kasnije, u poglavlju s korisničkim uputama.

U nastavku je prikazan model podataka na slici 4-2 i pojašnjenje modela.



Slika 4-2 Relacijski model podataka

Household:

Household je osnovna tablica u modelu podataka u kojoj se pohranjuju podaci o svakom kućanstvu. *_id* je primarni ključ dok je *name* jedinstveni atribut koji osigurava da ne postoje dva ili više kućanstva istog imena. Na *name* je postavljeno svojstvo da velika i mala slova ne čine razliku čime se osigurava da može postajati samo jedno kućanstvo istog imena bez obzira na velika i mala slova (npr. Horvat, horvat i HORVAT). *password* i *password_admin* su atributi u koje se spremaju lozinke.

House_type:

Tablica koja sadrži podatke o opcijama kućanstva tj. troškovima koje korisnik želi da se prate ili ne prate. Primarni ključ je *_id* dok je strani ključ *id_house* koji je povezan s tablicom *household* te može na jedno kućanstvo imati više zapisa o opcijama. Drugi strani ključ je *id_type* koji je povezan s tablicom *type*, a označava koja vrsta troška je zapisana. Na kraju imamo atribut *synced* u koji se pohranjuje vrijeme zadnje promjene retka te služi u svrhu sinkronizacije. Atribut *synced* se nalazi i u još nekim tablicama, ali ima istu ulogu te neće biti ponovno objašnjavan.

Type:

Rječnik koji povezuje identifikator vrste troška (*_id*) s njegovim imenom (*name*) i identifikatorom kategorije u koju spada (*id_cat*). *id_cat* je strani ključ povezan s tablicom *type_cat*.

Type_cat:

Rječnik koji povezuje identifikator kategorije troška (*_id*) s njegovim nazivom (*name*) i opisom koji se koristi za korisnika (*nice_name*). *name* je kratki naziv koji se koristi samo u programske svrhe.

Expense:

Tablica koja sadrži podatke o svakom trošku unesenom u aplikaciju. Primarni ključ je *guid* (engl. „Global unique identifier“), a to je identifikator koji se dobiva pomoću nasumičnog algoritma te garantira da će svaki *guid* biti jedinstven. Razlog uvođenju *guid*-a je više korisnika. Kad bi se koristio običan *id* koji za svaki novi redak raste za jedan došlo bi do sljedećeg problema. Ako više korisnika unosi troškove za isto kućanstvo lokalno, onaj korisnik koji zadnji sinkronizira svoje podatke dobio bi s poslužitelja retke s istim identifikatorom kao što ima u lokalnoj bazi podataka, te bi tu nastala greška zbog pravila primarnog ključa postavljenog nad bazom podataka. Kad se koristi *guid* takav slučaj nije moguć, pobliže vjerojatnost da se dogodi je astronomski mala jer svaki korisnik dobiva jedinstveni *guid*. *id_house*, *id_type* je strani ključ povezan s tablicom *house_type* koji ne dopušta da se unose podatci o trošku kućanstva koje nije zapisano u *house_type*. Kućanstvo može imati više troškova za istu vrstu troška. *amount* je iznos u lipama

(SQLite dopušta samo unos cijelih brojeva, a ne decimalnih te zato nije moguće imati iznos u kunama), *month_count* je broj mjeseci na koliko je raspoređen dotični trošak (primjerice u nekim slučajevima se voda naplaćuje dvomjesečno), *paid* je datum kad je trošak podmiren, *comment* je korisnički komentar na trošak, a *active* označava da li je zapis važeći ili ne (koristi se zbog sinkronizacije, te se nalazi u još nekim tablicama i neće biti ponovno objašnjavan). U slučaju kad je trošak raspoređen na jedan ili više mjeseci dodaju se zapisi u tablicu *monthly*.

Monthly:

Tablica koja sadrži podatke o mjesecima ili godinama na koje je trošak raspoređen. Primarni ključ je *_id*. *guid* je strani ključ povezan s tablicom *expense* koji dozvoljava da se unose samo oni zapisi čiji atribut *guid* već postoji u tablici *expense*. Moguće je imati više zapisa u tablici *monthly* za pojedini trošak u tablici *expense*. *id_house* označava na koje kućanstvo se zapis odnosi. *date* je atribut u kojem se nalazi mjesec ili godina na koji se trošak odnosi.

Sync:

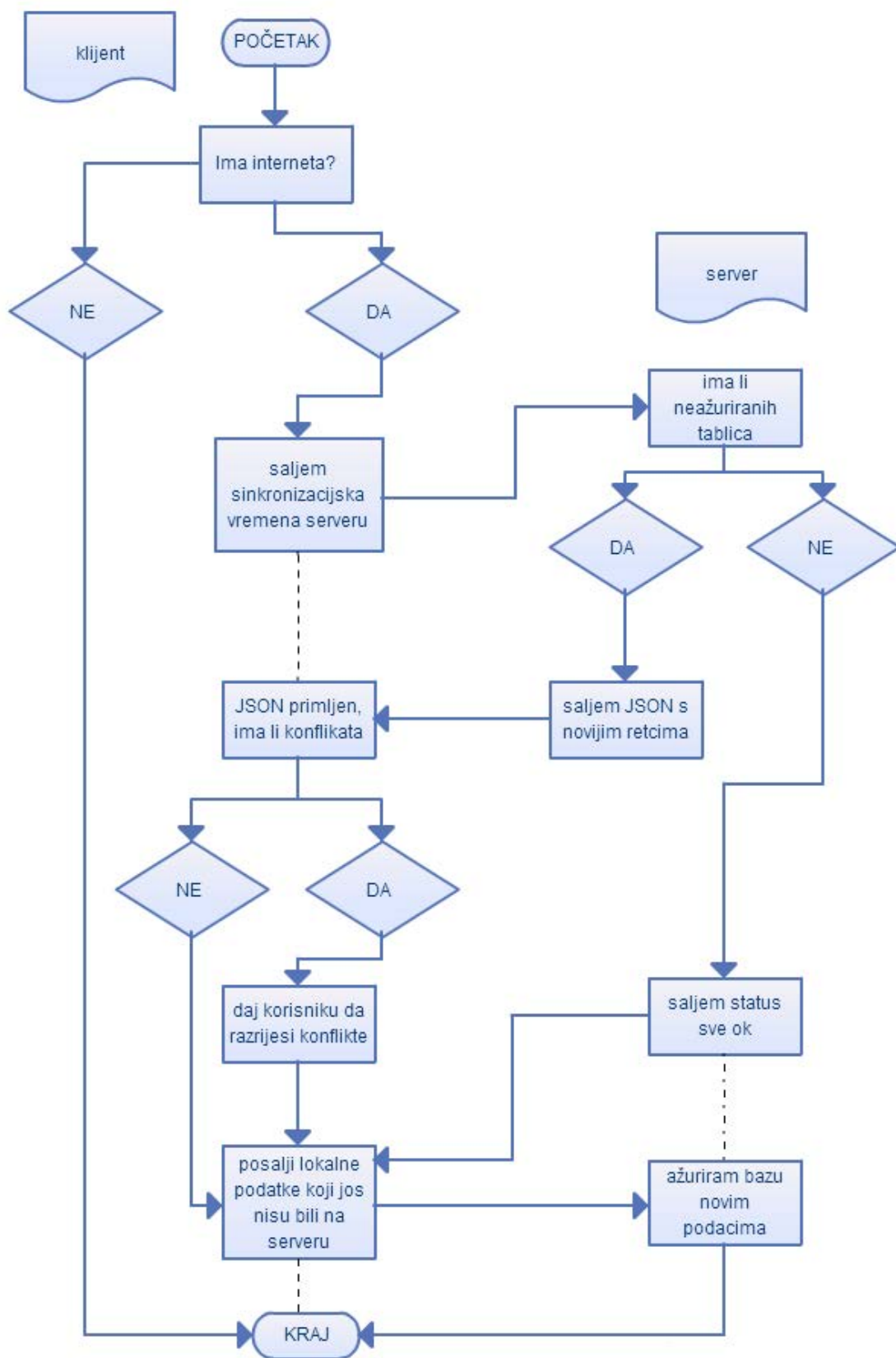
Tablica koja služi za sinkronizaciju. Primarni ključ je *_id*, a alternativni ključ je (*id_house*, *table_name*). Pohranjuje se vrijeme zadnje promjene za određenu tablicu nekog kućanstva. *date* je vrijeme zadnje promjene u sekundama. Iz ove tablice aplikacija šalje poslužitelju vrijeme zadnje promjene tablice.

Sam vizualni dizajn aplikacije bazira se na ugrađenim Android temama koje se koriste u većini Android aplikacija. Tim načinom se iskorištavaju u potpunosti grafički elementi koji su instalirani sa samim Android operacijskim sustavom, te se smanjuje efektivna veličina aplikacije.

4.3. Sinkronizacija

U ovom dijelu ćemo objasniti na koji način se vrši sinkronizacija i kako se rješavaju konflikti nastali kad više korisnika unosi podatke o jednoj vrsti troška za isto razdoblje. Nakon dvije stranice ovaj proces se može vidjeti u obliku dijagrama toka na slici 4-3.

Kad korisnik koristi aplikaciju, ona će se pokušati povezati s poslužiteljem i obaviti sinkronizaciju. Prvo ide provjera da li je uređaj spojen na internet. Za sinkronizaciju nije važno da li je uređaj spojen na Wifi ili na mobilni internet iz razloga jer se radi o izrazito niskom prometu podataka. Ako je spojen, tada aplikacija šalje poslužitelju podatke o zadnjem ažuriranju lokalne baze za trenutno kućanstvo. Poslužitelj to vrijeme uspoređuje sa svojim podacima o sinkronizaciji, tj. ispituje da li postoji tablica koja je ažurirana nakon vremena poslanog s klijenta. U slučaju da takva tablica ne postoji šalje se poruka s oznakom statusa da nije potrebna sinkronizacija klijentu i tu se prekida komunikacija. U slučaju da jedna ili više tablica jesu ažurirane nakon vremena koje je poslano s klijenta tada poslužitelj šalje klijentu poruku s retcima tablice koji su ažurirani nakon danog vremena. Ako nastanu konflikti (jedan primljeni redak je podatak o vrsti troška i razdoblju kojeg korisnik već ima pohranjenog lokalno ali još nije bio ažuriran na serveru) tada se korisnika pita želi li zadržati svoj lokalni podatak ili podatak primljen sa poslužitelja. U slučaju da korisnik želi zadržati lokalni podatak tada se konfliktni redak označava kao nevažeći te nakon što se pohrane svi ostali retci poslužitelju se šalje poruka da taj redak označi nevažećim. U slučaju da korisnik želi zadržati redak primljen sa servera tada se lokalni redak briše, a pohranjuje se redak primljen od poslužitelja. Kad se razriješe svi konflikti, klijent šalje na poslužitelj sve retke koji su pohranjeni samo lokalno, ako takvih ima. Server prima poruku koja se sastoji od tih redaka te pošto tu više ne može biti konflikata, on ažurira svoju bazu podataka te vraća poruku o uspješnoj sinkronizaciji i završava komunikaciju.



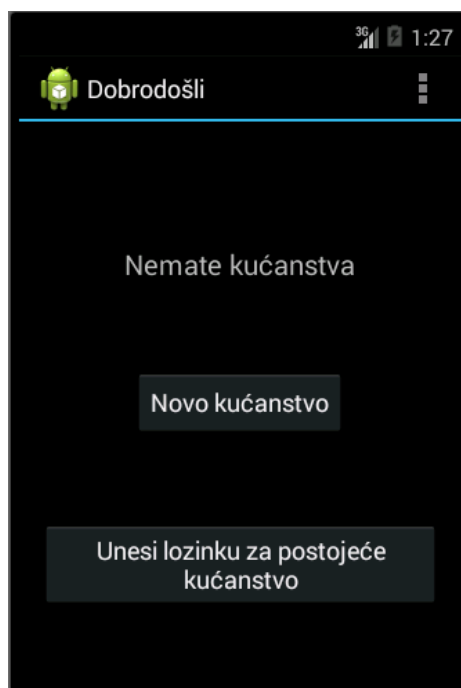
Slika 4-3 Dijagram toka sinkronizacije

5. Korisničke upute za aplikaciju „Household“

Da bi mogli koristiti aplikaciju „Household“ potrebno je imati pametni telefon ili tablet na kojem je instaliran Android operacijski sustav. Aplikacija se instalira kroz „Google play“ uslugu.

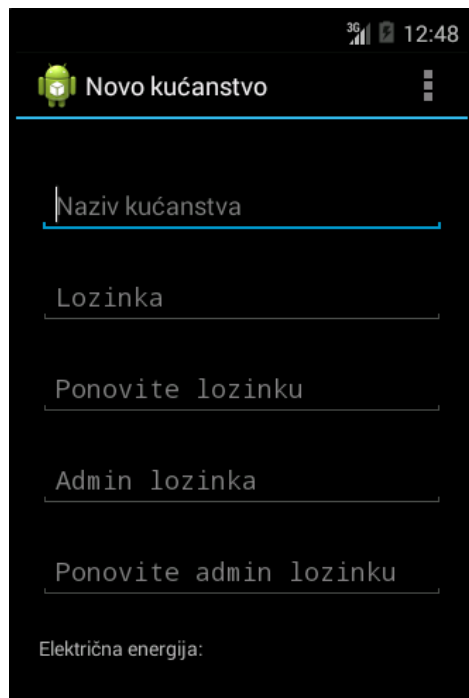
5.1. Prvo pokretanje

Nakon što je aplikacija instalirate te se prvi put pokrene pojavit će se zaslon kao što je prikazano na slici 5-1. S obzirom na to da se prvi put pokreće, nema nikakvih kućanstva unesenih u bazu te aplikacija nudi opciju unosa novog kućanstva ili ažuriranja baze s postojećim kućanstvom. Ako je već drugi član kućanstva unio kućanstvo u sustav tada se može odabrati druga opciju, ali za pristup je potrebno poznavati ime i lozinku kućanstva.



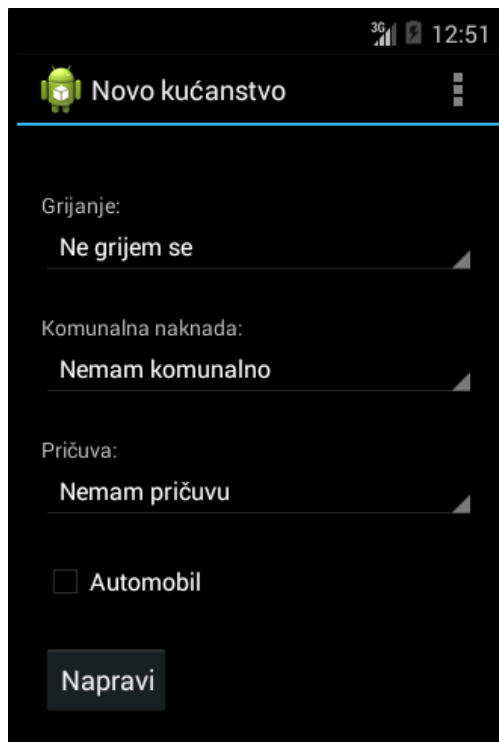
Slika 5-1 Početni zaslon

Ako je odabrano „Novo kućanstvo“ tada se otvara novi zaslon kao što je prikazano na slici 5-2. Za početak potrebno je upisati ime kućanstva koje mora biti jedinstveno među svim kućanstvima na serveru, tako da svaki put kad se upiše znak u polje za ime aplikacija provjerava da li je to ime dostupno. Nakon toga potrebno je upisati lozinku i administratorsku lozinku dva puta zbog zaobilaženja greški u tipkanju. Lozinke moraju imati barem 6 znakova. U nastavku je potrebno ispuniti detalje o kućanstvu.



Slika 5-2 Unos kućanstva

Slika 5-3 prikazuje dio opcija koje se mogu odabrati za kućanstvo. Kad su odabrane sve željene opcije dodirom na tipku „Napravi“ kućanstvo će se dodati u aplikaciju kao i na poslužitelj. Zatim se otvara početni zaslon.

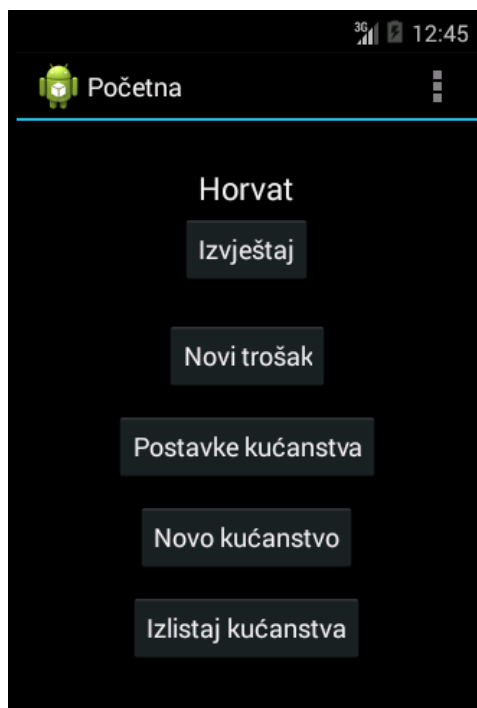


Slika 5-3 Unos kućanstva 2

5.2. Početni zaslon

Na slici 5-4 je prikazano kako izgleda početni zaslon aplikacije kad je kućanstvo postavljeno. Iz ovog zaslona moguće su sljedeće opcije:

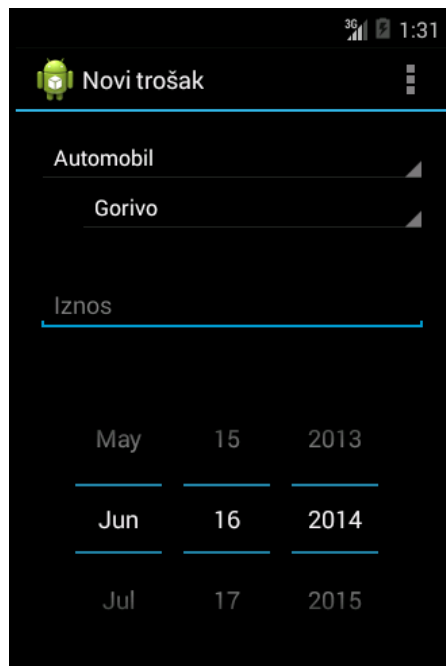
- Dodavanje novog troška trenutnom kućanstvu,
- Mijenjanje administratorskih postavki trenutnog kućanstva
- Pregled i odabir svih kućanstva unutar aplikacije
- Dodavanje novog kućanstva u aplikaciju
- Prikaz grafa troškova kućanstva



Slika 5-4 Početni zaslon

5.2.1. Novi trošak

Za dodavanje novog troška potrebno je dodirnuti tipku „Novi trošak“. Otvara se novi zaslon kao što je prikazano na slici 5-5. U ovom slučaju može se birati vrsta troška. Ponuđeni su samo oni troškovi koji su odabrani na postavkama kućanstva i standardni troškovi. Ovisno o odabranom trošku može se pojaviti lista kućica za označavanje vremenskog razdoblja (mjeseca ili godina) na koje se taj trošak odnosi, dok se za jednokratne troškove označava samo datum kad su se dogodili. U polje iznos upisuje se iznos u kunama gdje se lipe odvajaju točkom. Opcionalno se može napisati nešto u polje komentar za lakše praćenje na što se koji trošak odnosi.

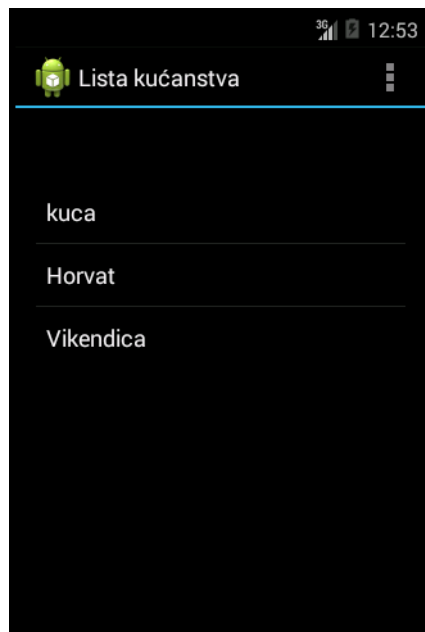


Slika 5-5 Novi trošak

Dodirom tipke dodaj, trošak će se dodati neovisno o tome da li je uređaj spojen na internet, te se vraća na početni zaslon.

5.2.2. Pregled svih kućanstva

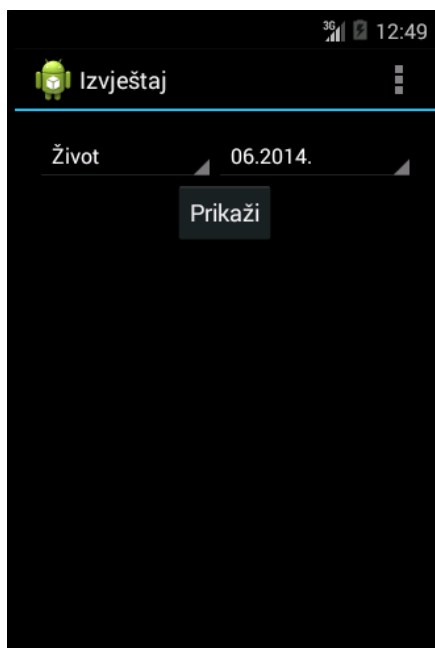
Dodirom tipke „Izlistaj kućanstva“ otvara se popis svih kućanstava u aplikaciji kao što je prikazano na slici 5-6. Dodirom bilo kojeg zapisa odabire se to kućanstvo kao glavno i vraća se na početni zaslon.



Slika 5-6 Lista kućanstva

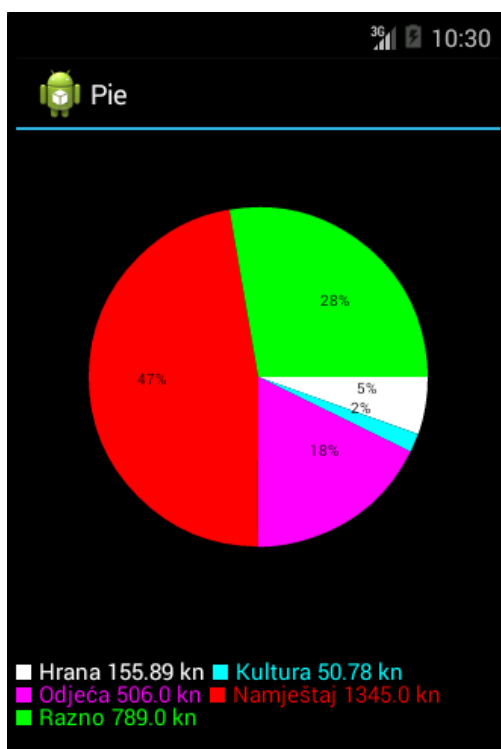
5.2.3. Graf troškova

Dodirom tipke „Izvještaj“ otvara se zaslon za odabir vrste troška i razdoblja za graf kao na slici 5-7. U prvom padajućem izborniku odabire se skupina troškova koja se želi prikazati, a u drugom mjesec za koji će biti napravljen izvještaj u obliku grafa. Ponuđeni su samo oni mjeseci na koje se odnosio neki od troškova iz odabrane skupine.



Slika 5-7 Odabir izvještaja

Dodirom tipke „Prikaži“ pojavljuje se graf kao na slici 5-8. Na grafovima se lako može vidjeti koji tip troška uzima najviše a koji najmanje kućnog budžeta.



Slika 5-8: Prikaz grafa

6. Zaključak

Korisnici sve više koriste mobilne aplikacije ponajviše iz razloga jer ih mogu koristiti kad i gdje se sjete. Zbog tog je bitno da aplikacija ima što intuitivnije korisničko sučelje i da se što lakše dolazi do željenih radnji. Također od velikog je značaja omogućiti korisniku da promjenom uređaja lagano prebaci podatke iz aplikacije sa starog na novi. Mobilne aplikacije su osjetljive na neefikasan kod jer uređaji na kojima se izvode imaju ograničenu količinu resursa. Zbog toga je bitno aplikaciju napraviti što jednostavnijom.

Aplikacija „Household“ je mobilna aplikacija napravljena za Android operacijski sustav čija namjena je praćenje troškova kućanstva. Klijentski dio aplikacije je kodiran u Javi korištenjem razvojnog okruženja Android Studio. Poslužiteljska strana je Apache server sa SQLite bazom podataka programiran u PHP-u. Aplikacija podržava više korisnika istog kućanstva koje sinkronizira preko web poslužitelja. Korisnik može imati više kućanstva, te aplikacija omogućava prikaz grafa skupine troškova za određeni mjesec.

7. Literatura

- [1] *Java (Programming language)*
[http://en.wikipedia.org/wiki/Java_\(programming_language\)](http://en.wikipedia.org/wiki/Java_(programming_language)) – Preuzeto
12.6.2014.
- [2] *JSON* - <http://en.wikipedia.org/wiki/JSON> - Preuzeto 12.6.2014.
- [3] *Php documentation* - <http://www.php.net/docs.php> - Preuzeto 9.6.2014.
- [4] Professor John F. Clark *History of mobile applications* -
<http://www.uky.edu/~jclark/mas490apps/History%20of%20Mobile%20Apps.pdf>
- Preuzeto 3.6.2014
- [5] *Php example* - <http://www.php-example.com/2011/05/html-form-example-in-php.html> - Preuzeto 26.6.2014.

MOBILNA APLIKACIJA ZA PRAĆENJE TROŠKOVA KUĆANSTVA

8. Sažetak

Android aplikacije su aplikacije namijenjene za Android operacijski sustav koji se nalazi u pametnim mobitelima, tabletima, pametnim televizorima. Pri izradi takvih aplikacija cilj je postići željenu funkcionalnost uz upotrebu ograničenih resursa koje pružaju takvi uređaji (slabi procesor, nedovoljno memorije). U današnje vrijeme aplikacije postaju sve sofisticiranije te korisnik traži određenu dozu jednostavnosti kod korištenja istih, kao što je automatsko izvršavanje nekih radnji primjerice osvježavanje lokalnih podataka s onima na serveru, intuitivnost korisničkog sučelja, jednostavan prijelaz s jednog uređaja na drugi i sl.

Aplikacija „Household“ je Android aplikacija za praćenje troškova kućanstva. Ona korisniku omogućava određivanje tipova troškova za kućanstvo, unos novih troškova, prikaz grafove mjesečne ili godišnje potrošnje iz kojih korisnik vidi na kojem području je previše potrošio u nekom razdoblju. Također aplikacija podržava više korisnika istog kućanstva čije se aplikacije sinkroniziraju preko servera.

Summary

Android apps are applications made for Android OS which is installed on smartphones, tablets, smart TVs. While developing those application it is crucial to achieve desired functionality using limited resources these devices offer (weak processor, low memory). Nowadays apps become more and more sophisticated and users require a certain amount of simplicity while using them, such as automatic execution of some actions like refreshing local files with those located on server, user interface intuitivity, simple transition from one device to another etc.

„Household“ app is an Android app for household expenses monitoring. Users are able to specify expenses types for household, enter new expenses, display of monthly or yearly expenses graphs from which user can see what area costed

them most in a specific period. App supports multi-user households where users can keep their apps synchronized over server.

9. Ključne riječi

- Android
- Aplikacija
- Java
- PHP
- JSON
- Kućanstvo
- Trošak

Keywords

- Android
- Application
- Java
- PHP
- JSON
- Household
- Expense