Nenativne mobilne aplikacije

Slaven Sarić, Visoka poslovna škola Minerva, Dugopolje

Sažetak – Količina ponude raznih naprednih mobilnih uređaja koji su se pojavili posljednjih godina donijela je dodatne probleme razvijateljima aplikacija. Dok je prije manje više bilo dovoljno znati programirati u C++, Javi ili C# za Windows OS i imali ste pokriveno gotovo cjelokupno tržište, danas za pokrivenost cijelog tržišta trebate poznavati barem tri programska jezika čiji se tzv. nativni programi izvršavaju na 3 različita operacijska sustava: Objective-C za iOS, Java za Android i C# za Windows mobile. To naravno nije lako, da ne kažemo da je nemoguće, i zato ćemo u ovom radu pokazati kako razviti (jednu te istu) aplikaciju za sve tri navedene generacije mobilnih uređaja i pritom ćemo koristiti alate koji su podržani u svakom od njih, a to su JavaScript, HTML i CSS uz dodatak PHP-a na serverskoj strani.

I. Uvod

Kao ogledni primjer izradit ćemo aplikaciju koja će se moći koristiti u nekom prodajnom mjestu koje obavlja jednostavniji oblik maloprodaje (kafić, zlatarnica...) i imat će sljedeće funkcionalnosti:

- Unos i izmjena podataka o artiklima
- Izrada i tiskanje računa
- Pregled prodaje

Cjelokupno rješenje će se izvršavati u prozoru web pretraživača mobilnog uređaja i uglavnom će se oslanjati na klijentske alate (JavaScript, HTML, CSS i AJAX) dok će se PHP koristiti isključivo za pretraživanje i izmjene u bazi podataka koja se nalazi na web serveru. Iako HTML5 preko objekta *localstorage*[1] omogućuje skladištenje podataka na klijentskoj strani, sve to je puno lakše s pravom bazom podataka kao što je MySQL koju ćemo mi koristiti. Iz toga naravno proizlazi da je za rad aplikacije neophodno imati mobilni pristup Internetu što danas nije nikakav problem niti iziskuje posebne troškove, a što se tiče brzine prijenosa ni tu nema posebno velikih zahtjeva jer će se količina prenesenih podataka mjeriti više u bajtovima nego u kilobajtima ili megabajtima.

Slika 1

II. Programsko rješenje

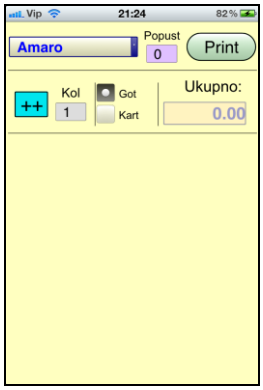
A. *Glavna forma aplikacije*

S obzirom da je riječ o web aplikaciji, glavna forma aplikacije dobit će se pokretanjem HTML dokumenta spremljenog u mobilnom uređaju (sl.1). Na zaslonu vidimo četiri HTML *input* elementa tipa *button* čiji atributi *onclick* kao vrijednost imaju poziv JavaScript funkcije:

```
function f(url) {window.open(url, "_self");}
<input type="button" value="Računi"
       onclick="f('racuni.php')" />
```

Argument funkcije je URL adresa web stranice koja će biti pozvana, a u slučaju tipke *Računi* to je PHP skripta na web serveru koja će izvršiti sljedeće zadatke:

1. Generirat će HTML kôd koji će osigurati željeni prikaz u prozoru web pretraživača. Također, generirat će i obradu događaja (*events*) nad HTML objektima koji omogućuju interakciju između HTML objekata i korisnika (kao npr. odabir artikla, upis količine, davanje popusta, način plaćanja i sl.)
2. Otvorit će bazu podataka na web serveru, generirati SQL upit i popuniti HTML *select/option* objekt s artiklima.



Slika 2: Izrada računa

Slika 3: Odabir artikla

Slika 4: Odabrani artikli

Prvo će biti urađeno integracijom HTML kôda unutar PHP datoteke[2] i to je većinom niz HTML *input* elemenata s različitim vrijednostima atributa *type* (sl.2,3,4) raspoređenih po zaslonu pretraživača, dok će se drugo izvršiti sa standardnim PHP funkcijama za pristup bazi podataka[3]:

```
$sql = "select * from Artikli order by Naziv";
$res = @mysql_query($sql);
$nRows = @mysql_num_rows($res);
$nFields = @mysql_num_fields($res);

printf("<select class=\"o2\" id=\"arts\" size=\"10\"
                onblur=\"DodajArtikal()\" >");
for($i=0;$i<$nRows;$i++) {
  $row = @mysql_fetch_array($res);
  printf("<option class=\"o2\" %s value=\"%s;%s;%s\">%s</option>",
   !$i ? "selected=\"selected\"" : "", $row[0], $row[3], $row[2], $row[1]);
}
printf("</select>");
```

Veliku pomoć pri formatiranju HTML elemenata i njihovih sadržaja pruža nam stara dobra *printf* funkcija koja u prozor pretraživača ispisuje formatirani string.

Tako gore prikazana funkcija *printf* prilikom definiranja *select/option* objekta s podacima (artiklima) iz baze podataka definira i atribut *value* elementa *option* s podacima dobivenim iz baze podataka, a to su šifra artikla, porezna grupa i cijena (*$row[0], $row[3] i $row[2]*) pojedinog artikla.

Nakon odabira artikla u *select/option* objektu podatke o odabranom artiklu spremamo u JavaScript niz na način da sadržaj pojedinog člana niza bude string u kojem će biti zapisani pojedini potrebni podaci o odabranom artiklu, a to su redom: šifra artikla, naziv artikla, količina, porezna grupa, cijena, eventualni popust i način plaćanja (ukupno 7 vrijednosti). Šifru artikla, poreznu grupu i

cijenu ćemo pročitati iz vrijednosti atributa *value* elementa *option*, dok ćemo ostale podatke dobiti jednostavnim čitanjem sadržaja HTML elemenata.

```
pos = document.getElementById('Artikli').selectedIndex;
str = document.getElementById('Artikli').options[pos].value;
arr[ct++] = str.substring(0, str.indexOf(";"));
arr[ct++] = document.getElementById('Artikli').options[w].text;
arr[ct++] = Kolicina.value;
arr[ct++] = Cijena.value;
arr[ct++] = str.substring(str.indexOf(";")+1, str.lastIndexOf(";"));
arr[ct++] = Popust.value;
arr[ct++] = (NacinPlacanja.checked == true ? 0 : 1);
```

Vrijednost atributa *value* elementa *option* zapisana je u CSV formatu: *Šifra;PoreznaStopa;Cijena* pa pojedine vrijednosti čitamo koristeći JavaScript funkcije *indexOf(";")* i *lastIndexOf(";")* koje vraćaju poziciju (indeks) prvog, odnosno zadnjeg pojavljivanja znaka u nizu.

Budući da o pojedinom artiklu zapisujemo 7 podataka, broj artikala na računu će biti ukupan broj članova niza podijeljen sa 7, a ako korisnik ponovo odabere već odabrani artikal ili mu samo pritiskom na tipku ++ poveća količinu za 1 nećemo povećati broj članova niza nego ćemo među već postojećim članovima niza pronaći onaj sa šifrom odabranog artikla i samo mu povećati količinu.

```
var BrojPolja = 7;
var flag = false;
var pos = 0;
for(i=0; i<arr.length; i += BrojPolja)
   if(parseInt(arr[i]) == parseInt(Sifra))
   {
      flag = true;
      pos = i;
      break;
   }
   if(flag)   // Artikal je već na računu, samo povećaj količinu
     arr[pos+2] = parseFloat(arr[pos+2]) + parseFloat(Kolicina.value);
   else       // Artikal nije na računu, dodaj ga u niz
      ...
```

Tipke za unos količine i popusta te odabir načina plaćanja su standardni HTML elementi jednako kao i tipka *Print* koja je ovako definirana:

```
<input type="button" class="i2c" value="Print" onclick="PrintRn()" />
```

Vidimo da je vrijednost atributa *onclick* poziv funkcije *PrintRn* koja će uraditi dvoje:

- spremiti račun u bazu podataka pozivom PHP skripte
- tiskati račun

Prije ovih dviju operacija radimo i neke standardne provjere kao što su: je li uopće odabran ijedan artikal te provjera želi li se stvarno otiskati račun. Ovo drugo pitamo jedino ako je tako korisnik definirao u postavkama u već spomenutom objektu *localstorage*.

```
if(ct != 0)  // Ako ima barem jedan artikal
{
   if(localStorage.printati == "da")
```

```
    {
       if(confirm("TISKATI RAČUN ?") == true)
         Print();
       else
         return;
    }
    else
       Print();
}
else
  alert("Nije odabran niti jedan artikal.");
```

Spremanje računa u bazu podataka radimo sa PHP skriptom koju pozivamo JavaScript funkcijom *Window.Open* [4], a podatke o artiklima koje treba zapisati u bazu podataka kao što su šifra, količina, itd... šaljemo s GET metodom. POST metoda načina slanja nam nije neophodna jer količina poslanih podataka u sklopu URL adrese je mala, a ni tajnovitost poslanih podataka nam nije od posebne važnosti.

```
window.open("spremi.php?podaci=" + podaci, "_self");
```

```
Cafe bar ABC
Ante Starcevica 27 Trilj
OIB: 1234567890

Naziv                 Kol Cijena  Iznos
-----------------------------------------
Kava                 2.00   5.00  10.00
Coca Cola            1.00  10.00  10.00
Fanta                1.00  10.00  10.00
-----------------------------------------
UKUPNO:                          30.00 kn

Racun br. 3365
27.05.2012.
Hvala na posjetu.
```

Slika 5: Izgled računa

Slika 6: Odabir printera

PHP skripta *spremi.php* će podatke o artiklima pročitati kao sadržaj $_GET varijable, ekstrahirat će pojedina polja iz CSV zapisanog formata te će slogove zapisati u bazu podataka.

Račun tiskamo tako da najprije, opet uz pomoć PHP funkcije *printf*, složimo račun u željenom i propisanom formatu (sl.5) koristeći neki od neproporcionalnih fontova kao što je Courier New zbog lakše poravnanja te nakon toga pozovemo JavaScript metodu *window.print()*.

Račun se tiska koristeći standardni poziv za printanjem od strane operacijskog sustava (sl.6), a tiskanje je i inače dio aplikacije koji će vam stvoriti najviše problema kod održavanja kompatibilnosti aplikacije (je li printer ima rezač iznad ili ispod glave, je li Epson ili Star kompatibilan, je li termalni ili ima ribon, a ako ima ribon je li može tiskati u dvije boje ili ne, je li printer bežični ili žični, ako je žični na kojem je portu spojen itd...). Međutim, isti ti problemi postojat će i kod nativne aplikacije jer ako korisnik i prijeđe na mobilnu aplikaciju on s punim pravom može reći da ne želi kupiti novi printer ako mu stari koji se spaja preko serijskog porta još uvijek dobro radi.

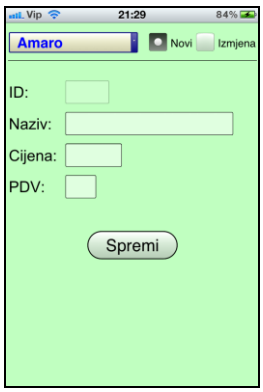
Tipka *Print* ima i definiran atribut *class*, njegova je vrijednost u ovom slučaju *i2c* koja je pak definirana u posebnoj CSS datoteci na koju se vežemo u HTML dokumentu preko elementa *link* i atributa *href*. `<link rel="stylesheet" type="text/css" href="safari.css" />`

```
input.i2c  /* Print */
{
  width: 270px;
  height: 110px;
  font-size: 65px;
  background-color: lightgreen;
  border: solid 4px;
}
```

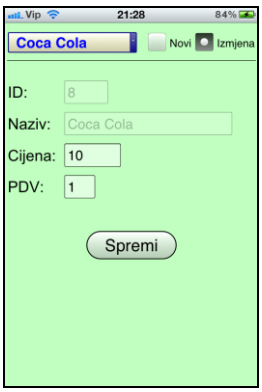
Na ovaj način, s odvajanjem strukture od izgleda uz pomoć CSS-a, osiguravamo da cjelokupni kôd (HTML, JavaScript i PHP) bude isti za sve vrste mobilnih uređaja i njihovih operacijskih sustava i pretraživača, a svu razliku definiramo u CSS datotekama. Tako npr. za iPhone definiramo datoteku Safari.css, dok bi za Androide definirali datoteku Android.css. Iako bismo, s obzirom na veliku količinu međusobno ipak različitih Android uređaja, trebali imati više različitih Android.css datoteka, te razlike nisu tolike da bi se te Android.css datoteke puno razlikovale.

B. *Podaci o artiklima*

Artikli su naravno, spremljeni u bazi podataka na web serveru i iako ih kao takve možemo jednostavno promijeniti pristupom MySQL administraciji na PC-u naša aplikacija mora osigurati da korisnik može i iz mobilne aplikacije dodati novi artikal (sl.7) ili npr. promijeniti cijenu postojećem (sl.8).



Slika 7: Unos artikla



Slika 8: Promjena artikla

Opet, kao i kod računa moramo povezati klijentsku stranu, odnosno podatke unesene u klijentskoj HTML formi i PHP skriptu na web serveru koja će te podatke zapisati u bazu podataka. To ćemo opet uraditi prosljeđivanjem s GET metodom, ali ovaj put uz pomoć AJAX tehnologije[5] tek da izbjegnemo klasični *postback* i omogućimo korisniku brži i elegantniji unos i osobito promjenu podataka (npr. ako korisnik hoće svim artiklima promijeniti cijenu, tada će to puno brže uraditi uz pomoć AJAX-a nego da nakon svake promjene cijene radi *postback* prema serveru). Gotovo da će korisnik imati dojam da radi s nativnom aplikacijom.

```
<input type="button" class="b5" value="Spremi" onclick="AjaxFunction()"/>
function AjaxFunction() {
   url = Artikli.php + "?ind=" + (r1.checked == true ? "1" : "0") +
                       "&id="     + art_id.value +
                       "&naziv="  + art_naziv.value +
                       "&cijena=" + art_cijena.value +
```

```
                        "&pdv="    + art_pdv.value;
   xmlHttp.onreadystatechange = stateChanged;
   xmlHttp.open("GET", url, true);
   xmlHttp.send(null);
}
```

Što se tiče samog spremanja novih i izmjene postojećih artikala, sve prepuštamo PHP-u koji će najprije pročitati varijable poslane s GET metodom:

```
$ind = $HTTP_GET_VARS['ind'];   // $ind=0 -> UPDATE, $ind=1 -> INSERT
$id = $HTTP_GET_VARS['id'];
$naziv = $HTTP_GET_VARS['naziv'];
$cijena = $HTTP_GET_VARS['cijena'];
$pdv = $HTTP_GET_VARS['pdv'];
```

i zatim ih zapisati u bazu podataka pozivom SQL *insert* ili *update* naredbe:

```
$sql = sprintf("insert into Artikli(Naziv, Cijena, GrupaID)
                          values(\"%s\",%lf,%d)", $naziv, $cijena, $pdv);
$sql = sprintf("update Artikli set Cijena=%lf, GrupaID=%d where ID=%d",
                           $cijena, $pdv, $id);
$res = @mysql_query($sql);
```

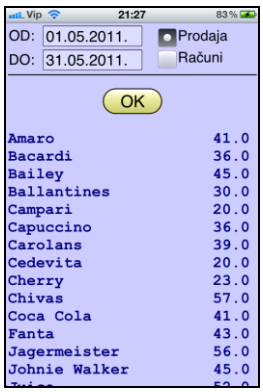
C. *Pregled prodaje i računa*

Treća tipka glavne forme omogućuje pregled ostvarene prodaje (sl.9) i izrađenih računa (sl.10) i mora biti neizostavni dio ovakve aplikacije. Opet ćemo to ostvariti kombinacijom klijentskog i serverskog kôda. Na klijentskoj strani ćemo definirati izgled formi, njihovih objekata i ponuđenih vrijednosti dok će serverska strana na osnovi unesenih vrijednosti poslanih serveru izdvojiti i izračunati iz baze podataka tražene podatke i poslati ih natrag klijentu u obliku HTML kôda. Vezu između klijenta i servera opet ćemo uraditi s AJAX-om tako da korisnik gotovo da i nema dojam da dolazi do razmjene podataka između web servera i mobilnog pretraživača, a AJAX nam olakšava i izradu serverske skripte koja, budući da nema klasičnog *postbacka*, ne mora klijentu slati HTML kôd cijele web stranice nego samo dio koji se mijenja, a to je samo jedan element čiji će sadržaj biti izračunato stanje prodaje po artiklima ili izlist traženih računa.

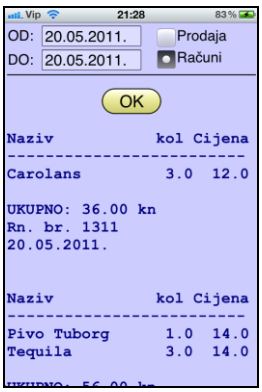
```
<div id="pregled" class="d3"></div>
```

i u funkciji *stateChanged()* definiramo:

```
if(xmlHttp.readyState == 4)
   document.getElementById("pregled").innerHTML = xmlHttp.responseText;
```



Slika 9: Pregled prodaje



Slika 10: Pregled računa

S obzirom da se izračun pregleda prodaje i izračun izdanih računa prilično razlikuju kako u sadržaju navedenog <*div*> taga tako i u SQL sintaksi potrebnoj da bi se iz baze izdvojili traženi podaci, najjednostavnije nam je razdvojiti ih svakog u svoju PHP skriptu.

Pregled prodaje po pojedinom artiklu u određenom razdoblju izračunat ćemo uz pomoć SQL *Group By* naredbe:

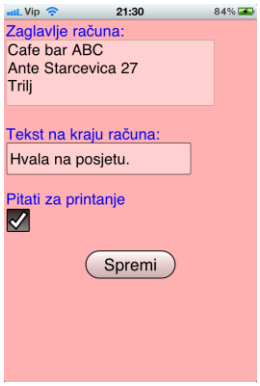
```
$sql = sprintf("
select Artikli.Naziv,Sum(Racuni.Kolicina),sum(Racuni.Cijena*Racuni.Kolicina)
from Racuni, Artikli
where Artikli.ID = Racuni.ArtikalID  &&
                   Racuni.Datum >= '%d-%d-%d'  &&
                   Rn.Datum <= '%d-%d-%d'
group by Art.Naziv", $y1, $m1, $d1, $y2, $m2, $d2);
$res = @mysql_query($sql);
```

Vrijednosti početnog i krajnjeg datuma razdoblja korisnik unosi u klijentskoj formi te ih PHP čita kao sadržaj $_GET varijable.

Što se tiče pregleda izdanih računa, posao je olakšan jer ne trebamo nikakvo grupiranje i dovoljna nam je obična *select/where* naredba:

```
$sql = sprintf("
select Racuni.Broj, Artikli.Naziv, Racuni.Kolicina,
       Racuni.Cijena, Racuni.Datum
from Racuni, Artikli
where Art.ID = Rn.ArtID  &&  Rn.Datum >= '%d-%d-%d'  &&
               Rn.Datum <= '%d-%d-%d'", $y1, $m1, $d1, $y2, $m2, $d2);
$res = @mysql_query($sql);
```

D. ***Postavke***



Slika 11

Četvrta tipka u glavnoj formi je tipka *Postavke* koja služi za definiranje i spremanje nekih postavki na razini pojedinog korisnika aplikacije. Posebnost je u tome što ćemo spremanje uraditi na klijentskoj strani bez pomoći serverske baze podataka i za to će nam poslužiti HTML5 objekt *localstorage*. Kao što se vidi na slici, spremamo neke osnovne podatke kao što su: zaglavlje računa, tekst koji se tiska na kraju svakog računa, a možemo vidjeti i jedan *input* element tipa *checkbox* koji nam definira korisnikovu želju hoće li da ga program pita za potvrdu prije tiskanja svakog računa (sl.11). S obzirom da takva pitanja znaju biti pomalo iritirajuća ostavljeno je korisniku da to sam sebi definira.

Na kraju svega možete se još jedino potruditi da vam ovakva nenativna (web) aplikacija izgleda što sličnije nativnoj aplikaciji, a ono što će svaki korisnik sigurno tražiti je prelazak na sljedeći i prethodni zaslon pomicanjem prsta po zaslonu iako se isto može postići već ugrađenim, dobro poznatim funkcionalnostima pretraživača, a to su tipke Natrag (*Back*) i Naprijed (*Forward*). To ćemo riješiti tako da na svakom HTML dokumentu naše aplikacije definiramo i obradimo događaje *ontouchstart* i *ontouchmove*.

```
<body class="b4" ontouchstart="javascript: KoordX = 10000;"
                 ontouchmove="fPomak(event)" >
```

Događaje definiramo nad elementom <*body*> jer želimo omogućiti da korisnik može pomaknuti prstom bilo gdje na zaslonu pretraživača.

Pomak prstom generira najprije jedan događaj *ontouchstart*, a zatim i niz događaja *ontouchmove*. Zato ćemo kao reakciju na događaj *ontouchstart* varijabli *KoordX* dodijeliti neku vrijednost koja je sigurno veća od najdesnije koordinate zaslona (npr. 10000) i onda kao reakciju na događaj *ontouchmove* pozivamo funkciju *fPomak()* u kojoj preko čitanja trenutne koordinate po osi X provjeravamo je li ostvaren pomak prstom udesno.

```
function fPomak(e) {
  var Dogadjaj =  e.touches.item(0);
  if(Dogadjaj.clientX - 100 > KoordX)
     window.open("index.html", "_self");
  else
     KoordX = Dogadjaj.clientX;
}
```

U *if* dijelu provjeravamo je li prst pomaknut prema desno, a zbog početne vrijednosti *KoordX* u *else* dijelu ćemo zabilježiti ostvarenje prvog generiranog *ontouchmove* događaja te memoriramo X koordinatu da bismo je mogli usporediti u sljedećem generiranom *ontouchmove* događaju.

III. Zaključak

Nativne ili nenativne aplikacije?

Prednosti nativnih aplikacija su bolja prilagodljivost pojedinoj platformi, pristup hardveru uređaja, općenito ugodniji i brži rad s aplikacijom, dok je prednost nenativnih aplikacija lakša prenosivost između različitih uređaja i njihovih operacijskih sustava, lakše održavanje, brži pristup većem broju klijenata, neovisnost o možebitnoj promjeni strateških planova Applea, Microsofta ili nekog trećeg kojima se možda neće biti lako prilagoditi. Većina prednosti jednih su uglavnom i nedostaci drugih.

Na kraju, pitanje jesu li bolje nativne ili nenativne aplikacije je suvišno jer su nativne aplikacije u svakom slučaju bolje, ali ako postavimo pitanje koje su aplikacije isplativije tada je već teže dati točan odgovor. Pravi odgovor je da na to pitanje nema točnog odgovora i sve se svodi na to je li vam više znače prednosti jednih ili vas više ograničavaju nedostaci drugih.

Popis literature

1. Bruce Lawson, Introducing HTML5, O'Reilly, 2011.
2. Official PHP documentation http://php.net/docs.php
3. Official MySQL documentation http://dev.mysql.com/doc
4. Maximiliano Firtman, Programming the Mobile Web, O'Reilly, 2010.
5. Nathaniel T. Schutta, Ryan Asleson, Foundations of Ajax, Apress, 2006.