Firefox OS: nov izziv za študentske projekte

Uroš Berglez, Robert Meolic

Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
E-pošta: uros.berglez@um.si, robert.meolic@um.si

Firefox OS: a new challenge for student projects

Telecommunications students must carry out a number of projects while studying, and many of them involve mobile devices. This paper suggests that Firefox OS, a new mobile operating system, is a good platform for such projects. One of its interesting benefits is a fast learning curve because it builds on standardised web technologies, which are already familiar to many students. Moreover, using an open platform is an ideal choice for academic environments. This paper shows that one can make an interesting project in a few relatively easy steps without installing and learning complex environments. Some important aspects of project planning are also discussed and an example of a student project is given.

1 Uvod

Na Fakulteti za elektrotehniko, računalništvo in informatiko študenti programa Telekomunikacije pri več predmetih izvajajo projektno delo. V zadnjem času so zaželeni projekti, pri katerih lahko vključijo lastne mobilne naprave. Taki projekti pri študentu in njegovi okolici poudarijo občutek, da se uči nekaj uporabnega in koristnega.

Organizacija *Mozilla Foundation* je v letu 2013 uradno predstavila mobilni operacijski sistem *Firefox OS*. V letu 2014 je luč sveta zagledala različica 1.3, ki je nameščena na zanimivem in v Evropi dosegljivem telefonu *ZTE Open C*.

V članku pokažemo, da je *Firefox OS* dobra platforma za izvajanje študentskih projektov. Na primeru, ki ne zahteva veliko predznanja, izpostavimo preprostost razvoja aplikacij, kar študentom omogoča hiter začetek in osredotočanje na cilje, ne pa na tehnologijo. Druge zanimive prednosti sistema *Firefox OS* so:

- sistem je zgrajen iz proste kode in to načelo tudi promovira, kar olajša izmenjavo idej med razvijalci in se tudi idealno poda v akademsko okolje,
- programira se z uporabo sorodnih tehnologij, kot se uporabljajo za izdelavo spletnih aplikacij, zato ima pridobljeno znanje še dodatno uporabnost,
- aplikacije so sorazmerno enostavno prenosljive na vse platforme, na katerih teče brskalnik *Mozilla Firefox*, zato imajo dolgoročno vrednost.

2 Izdelava preproste aplikacije

Ne glede na uporabljen operacijski sistem potrebujemo le spletni brskalnik *Mozilla Firefox*. V brskalnikovem meniju izberemo Orodja › Spletni razvoj › Upravitelj aplikacij. Upravitelj aplikacij nam omogoča zagon aplikacije v simulatorju in njen prenos na fizično napravo. Za razvoj bomo uporabljali oboje, zato namestimo vtičnika *Firefox OS Simulator* in *Adb Helper*. Uspeh namestitve si ogledamo tako, da v meniju izberemo Orodja › Dodatki › Razširitve. Na sliki 1 je prikazan telefon *ZTE Open C* ter zaslonski sliki prilagojene osnovne strani telefona in simulirane naprave.

Za prvi primer projekta pripravimo aplikacijo *Pozdravljen svet*, ki kot rezultat izpiše zgolj pozdravno sporočilo. Pripravili bomo *packaged app*, pri katerem so vse potrebne datoteke shranjenje v eni mapi. V prazni mapi tvorimo nekaj datotek, obvezna je `manifest.webapp`, ki vsebuje meta-podatke o aplikaciji.

```
{ "name": "Pozdrav",
  "launch_path": "/index.html",
  "icons": {"128": "/icons/icon.png"} }
```

Manifest med drugim določa glavno datoteko, ki se prikaže ob zagonu aplikacije. V prikazanem primeru je to datoteka `index.html`, ki vsebuje naslednjo kodo HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pozdravljen svet</title>
  </head>
  <body>
    <div>Pozdravljen svet!</div>
  </body>
</html>
```

Celotno aplikacijo razvijamo podobno kot navadno spletno stran, le da je ta popolnoma prilagojena za mobilne naprave. Pomembno je, da jo preverjamo na čimveč napravah, pri čemer si lahko do neke mere pomagamo tudi s simulatorjem. Z dobrim poznavanjem funkcionalnosti, vključenih v standard *HTML5* ter znanjem skriptnega jezika JavaScript lahko pripravimo zmogljivo ter do uporabnika prijazno aplikacijo [1, 2].

Preden se odločimo za izdelavo projekta, je treba premisliti in vsaj po delih preveriti njegovo izvedljivost. Za večje projekte je priporočljivo izdelati prototip, v katerem izvedemo najbolj kritične funkcionalnosti. Pri projektih, ki temeljijo na pridobivanju informacij z interneta,

Slika 1: Telefon *ZTE Open C*, prilagojena zaslonska slika osnovne strani in simulirana naprava.

zadošča, če s pomočjo brskalnika preverimo, da znamo in zmoremo pridobiti vse potrebne podatke. To je še posebej pomembno, če podatkov ne pridobivamo z lastnega strežnika, ampak se poslužujemo zunanjih virov.

Uporabniku mora aplikacija, ki si jo namesti, prinesti neko dodano vrednost, sicer je njen namen ničen. Zato pri načrtovanju hitro ugotovimo, da statično pripravljene vsebine in uporaba zgolj označevalnega jezika *HTML* ni dovolj za izvedbo zastavljenih funkcionalnosti. Potrebujemo dinamičnost aplikacije, ki jo dosežemo z dodajanjem kode JavaScript.

Za učinkovitejše delo lahko uporabimo eno ali več dodatnih knjižnic za JavaScript, ki implementirajo funcionalnosti, katere pripomorejo k lažji in hitrejši obdelavi objektov *DOM* ter lažjemu pridobivanju podatkov s strežnika s pomočjo t.i. zahtevkov *Ajax*. Tako kot pri knjižnicah za druge programske jezike je tudi tukaj ideja v tem, da razvijalec v svojih programih uporabi že narejene gradnike. Ločimo dva pristopa: uporaba knjižnic, ki JavaScript dopolnijo z dodatnimi gradniki ali pa uporaba celovitega ogrodja za strukturiran razvoj, ki sledi načrtovalskemu vzorcu *Model-View-Controller* oz. kakšni podobni tehniki. Če upoštevamo vse možnosti, je na voljo na desetine različnih rešitev. Najbolj razširjena knjižnica, ki ponuja dodatne gradnike za razvoj aplikacij JavaScript, je *jQuery* [3]. Kot alternativo lahko uporabimo izpeljanko *Zepto*, ki je na račun nepodpore starejšim sistemom učinkovitejša. Po potrebi v projekt dodamo še knjižnice *Backbone* (strukturirano načrtovanje), *Underscore* oz. *Lo-dash* (funkcijsko programiranje), *Modernizr* (prilagajanje napravam) in druge.

Kot drugi primer bomo v pripravljeni dokument vključili knjižnico jQuery, ki jo prenesemo z uradne spletne strani. Tako kot vse datoteke JavaScript je tudi ta v navadni tekstovni obliki. Za lažje upravljanje z datotekami si ustvarimo mapo `js`. V njej ustvarimo še prazno tekstovno datoteko z imenom `app.js`, v katero bomo v nadaljevanju dopisali kodo, s katero bomo definirali lastne funkcionalnosti. Nova struktura datotek je sedaj takšna:

```
index.html          |   icons/icon.png
js/app.js           |   js/jquery-latest.min.js
manifest.webapp
```

Dopolniti moramo tudi vsebino datoteke `index.html`, vpišemo poti do dodanih knjižnic in dva elementa `div`:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pozdravljen JavaScript!</title>
    <script src="js/jquery-latest.min.js"
            type="text/javascript"></script>
    <script src="js/app.js"
            type="text/javascript"></script>
  </head>
  <body>
    <div id="el1"></div><div id="el2"></div>
  </body>
</html>
```

Enostaven primer uporabe funkcij *jQuery*-ja dobimo, če v datoteko `js/app.js` dodamo naslednje vrstice:

```
$(document).ready(function(){
  $("div").text("Pozdravljen");
  $("#el2").html("<ul><li>Svet!</li></ul>");
});
```

jQuery je objekt, do katerega dostopamo preko spremenljivke `jQuery` ali `$`. Kot parameter mu nato podamo izbirnike (ang. selectors). V našem primeru je to `document`, ki predstavlja trenutno naložen dokument HTML. Z `$("div")` izberemo vse elemente `div` v dokumentu, če pa želimo izbrati element z atributom `id` in vrednostjo `el2`, uporabimo `$("#el2")`. V primeru, da bi imel element namesto identifikatorja definiran razred, bi namesto izbirnika `#el2` uporabili `.el2`. Za izbirnikom lahko napišemo ime metode, ki naj se izvrši nad najdenimi elementi. V našem primeru smo zapisali `$(document).ready()`, zato da brskalnik kodo, ki

se nahaja v parametru, izvrši šele, ko je dokument naložen do zadostne mere in je pripravljen celoten objekt DOM.

V prikazani kodi s pomočjo metode `text()` v oba elementa `div` vstavimo neoblikovano besedilo, z metodo `html()` takoj za tem pa v izbrani element ugnezdimo novo kodo HTML.

V tretjem primeru dodamo interakcijo z uporabnikom. Knjižnica jQuery brez dodatnih modulov ne podpira gest. Problem lahko rešimo tako, da npr. dodamo knjižnico *jQuery Mobile* ali pa knjižnico *Hammer*. Če nočemo biti odvisni od dodatnih knjižnic, pa lahko razpoznavanje gest realiziramo tudi na nižjem nivoju, saj jedro sistema vsebuje funcionalnosti specifikacije *Touch Events* (W3C, 2013, [4]).

Vzorčno datoteko `index.html` spremenimo, tako da v element `body` namesto dosedanje kode vstavimo:

```
<img id="slika" width="480" height="336">
```

Za implementacijo razpoznave gest moramo v datoteko `js/app.js` vpisati:

```
var imgNo = 1, swipeMin = 80;
var aT = ""; // "activeTouch"
function tS(evt) { //"touchstart"
  aT = evt.changedTouches[0];
}
function tM(evt) { //"touchmove"
  evt.preventDefault();
}
function tE(evt) { //"touchend"
  var pageX = evt.changedTouches[0].pageX;
  if (aT) {
    if (pageX - swipeMin > aT.pageX)
      imgNo = (imgNo == 1) ? 3 : imgNo - 1;
    else if (pageX + swipeMin < aT.pageX)
      imgNo = (imgNo == 3) ? 1 : imgNo + 1;
    $("#slika").attr("src", "img"+imgNo+".png");
}}
$(document).ready(function() {
  document.addEventListener("touchstart",tS);
  document.addEventListener("touchmove",tM);
  document.addEventListener("touchend",tE);
  $("#slika").attr("src","img1.png");
});
```

Dodali smo torej element <img> brez veljavnega atributa `src`, ki ga nato dinamično, glede na izvedeno gesto, spremenimo z JavaScriptom. Ko se uporabnik dotakne zaslona, si v globalno spremenljivko `aT` shranimo točko dotika. Ker nočemo ohraniti privzetega obnašanja ob potegu (npr. slika, ki ni v celoti prikazana, se ob potegih premika), prestrežemo vse dogodke *touchmove* in jim preprečimo vpliv na aplikacijo. Ko se prst odmakne, preverimo, ali je uporabnik potegnil dovolj daleč, da lahko to obravnavamo kot poteg. Za pravilno delovanje primera moramo v mapo seveda dodati slike `img1.png`, `img2.png` in `img3.png`.

V zadnjem primeru dodamo nalaganje vsebin z interneta. Pri tem moramo paziti na omejitev modernih brskalnikov, ki iz varnostnih razlogov ne dovolijo nalaganja dokumentov HTML z naslovov tretjih spletnih strani (angl. *same-origin policy*). Ta omejitev ne velja za nalaganje slik in nekaterih standardnih oblik izmenjave podatkov, prav tako ne vpliva na t. i. *native* aplikacije, ki lahko do vsebin na internetu dostopajo neposredno, brez pomoči brskalnika. V sistemu Firefox OS se brskalniku ne moremo izogniti, lahko pa v *manifestu* zahtevamo dovoljenje za nalaganje takšnih vsebin. Aplikacija s tem postane privilegirana in za njo veljajo strožja pravila glede objave na tržnici ter nameščanja na napravo.

Ideja zadnjega primera je v tem, da slike naložimo s spletnega strežnika, pri čemer pa se informacija o tem, katera slika naj se prikaže, tudi pridobi z interneta. Datoteko `js/app.js` spremenimo tako, da vsebuje naslednjo kodo (manjkata metodi `tS` in `tM`, ki sta enaki kot v prejšnjem primeru):

```
var pageNo = "",  swipeMin = 80, aT = "";
var srv = "http://teletext.rtvslo.si/ttx/";
var rq = new XMLHttpRequest({mozSystem:true});
function tE(evt) {
  var pageX = evt.changedTouches[0].pageX;
  if (aT) {
    if (pageX - swipeMin > aT.pageX)
      makeRequest(srv + "applications/ttx?"+
      "cmd=prpage&page="+pageNo+"&subpage=1");
    else if (pageX + swipeMin < aT.pageX)
      makeRequest(srv+"applications/ttx?"+
      "cmd=nepage&page="+pageNo+"&subpage=1");
}}
function makeRequest(dataURL) {
  rq.open("GET",dataURL,true);
  rq.onreadystatechange = function() {
    if(rq.readyState == 4 && rq.status == 200) {
        var imgsrc = $(rq.response).
              find("img:first").attr("src");
        $("#slika").attr("src",srv+imgsrc);
        pageNo = imgsrc.split("_")[0];
  }};
  rq.send();
}
$(document).ready(function() {
  document.addEventListener("touchstart",tS);
  document.addEventListener("touchmove",tM);
  document.addEventListener("touchend",tE);
  makeRequest(srv+"applications/ttx?"+
              "cmd=load&page=100&subpage=1");
});
```

Datoteko `manifest.webapp` smo dopolnili takole:

```
{ "name": "TTX",
  "launch_path": "/index.html",
  "icons": { "128": "/icons/icon.png" },
  "fullscreen": "true",
  "orientation": "landscape",
  "type": "privileged",
  "permissions": { "systemXHR": {}} }
```

V metodi `makeRequest()` pridobimo dokument z naslova, ki je bil podan kot argument. V pridobljenem dokumentu (`rq.response`) s pomočjo izbirnikov poiščemo atribut `src` prve slike. Pridobljeni niz znakov uporabimo kot ime prikazane slike. V naslednji vrstici s pomočjo operacije `split` iz imena dobimo podniz do prvega podčrtaja in ga shranimo v globalno spremenljivko `pageNo`. To informacijo o trenutno prikazani sliki potrebujemo, ko od strežnika zahtevamo nov dokument.

Aplikacija na enak način kot v prejšnjem primeru razpozna gesti poteg levo in poteg desno, ob razpoznani gesti pa od strežnika zahteva nov dokument. V primeru smo uporabili strežnik za teletekst RTV Slovenija, uporabljene zahteve pa nam omogočajo, da dobimo informacijo o prejšnji in naslednji strani (ne obstaja stran za vsako številko in le s pomočjo poizvedb na strežniku se lahko izognemo luknjam).

3 Vzorčni projekt: Teletekst

Neposredni cilj študentskega projekta je mobilna aplikacija, vendar v skladu z modernimi standardi za kakovost (npr. ISO 9001:2008) ni pomemben le končni izdelek, pač pa tudi postopek, kako pridemo do izdelka. V opisih predmetov so navedena znanja, ki naj bi jih študenti pridobili z izdelavo projektov, in v grobem jih lahko razdelimo v dve skupini:

- razumevanje tehnologij, ki omogočajo razvoj mobilnih sistemov, naprav in storitev (strojna oprema, operacijski sistemi, programski jeziki, označevalni jeziki, podatkovne baze, varnostni pristopi),
- razumevanje razvojnega ciklusa programske opreme (planiranje, načrtovanje, izvedba, testiranje, dokumentiranje, vzdrževanje).

V nadaljevanju podajamo primer projektne naloge v obliki naročnikovih zahtev. Naročnik potrebuje aplikacijo za pregledovanje teleteksta RTV Slovenija na mobilni napravi s sistemom Firefox OS. Teletekst spada med storitve iz prejšnjih časov, a v poplavi naprednih tehnologij ostaja priljubljen, ker je za uporabnika preprosta storitev. Nekako tako kot stari dobri SMS-ji, malo novejši Twitter in še zelo svež Yo. V osnovni obliki ima teletekst problem predvsem glede nedosegljivosti za mobilne uporabnike, saj sprejem televizijskega signala ni običajna funkcija mobilnih naprav. RTV Slovenija je zato tako kot mnogi drugi ponudniki omogočila pregledovanje teleteksta tudi s pomočjo spletne aplikacije. Problem dostopa preko brskalnika pa je, da uporabniški vmesnik ni prilagojen mobilnim napravam z majhnim zaslonom. Na sistemu Firefox OS je izrazita tudi težava z vnosom številk, vnosno polje je majhno, virtualna tipkovnica pa zakriva večji del vsebine. Omeniti velja, da na tržnici Firefox Marketplace že obstaja namenska aplikacija za dostop do teleteksta RTV Slovenija, ki pa po prijaznosti in funkcionalnosti zaostaja celo za spletno aplikacijo.

Od aplikacije, ki bo rezultat projekta, se pričakujejo naslednje funkcionalnosti:

- posodabljanje vsebine s strežnika RTV Slovenija,
- prikazovanje vsebine in številke trenutne strani na uporabniku prijazen način,
- aktivni odziv na dotik prikazane številke,
- uporaba gest za premikanje na prejšnjo in naslednjo stran ter prejšnjo in naslednjo podstran,
- bližnjice za dostop do pogosto obiskanih strani,
- namenska tipkovnica, ki je prikazana le na zahtevo.

Od izvajalca se pričakuje, da najprej pripravi plan projekta in specifikacijo izvedbe. Po odobritvi s strani naročnika je treba aplikacijo pripraviti v načrtovanem roku. Naročnik bo aplikacijo ovrednotil na napravi ZTE Open C. Sestavni del projekta sta razvijalska dokumentacija in uporabniški priročnik.

Na sliki 2 je primer uporabniškega vmesnika, ki se dobro prilega napravi ZTE Open C. Ločljivost zaslona tega telefona je 800x480, pri čemer pa operacijski sistem to resolucijo še zmanjša za faktor 1,5. Uporabnik ima tako na voljo skromno resolucijo 533x320, a ker so slike s strežnika velikosti 480x336, se v ležečem načinu dobro prilagajajo napravi tudi brez preoblikovanja.

Slika 2: Primer pregledovalnika za teletekst.

4 Zaključek

Projektno delo je pomemben element študijskega procesa. Na področju telekomunikacij, pa tudi na sorodnih področjih, vedno večji pomen pridobiva programska oprema, kar narekuje tudi vsebino projektov. Članek obravnava gradnjo aplikacij za eno novejših mobilnih platform, Firefox OS. Predstavljena je ideja za vzorčni projekt, hkrati pa je podanih dovolj napotkov in primerov, da lahko tudi študent brez predznanja začne z reševanjem. Vse skupaj na le štirih straneh, s čimer je poudarjena velika uporabnost sistema Firefox OS za študentske projekte. Omeniti velja tudi, da je za solidno rešitev vzorčne naloge vsega skupaj dovolj le okoli 200 vrstic kode.

Za zahtevnejše projekte so na voljo številne funkcionalnosti, ki jih v članku nismo omenili, npr. vnaprejšnje nalaganje vsebine, sinhronizacija podatkov, obdelava slik, multimedijski dodatki, registracija uporabnika itd. Tudi pri uporabniškem vmesniku nismo omejeni, uporabimo lahko npr. gradnike iz knjižnic Firefox UI Building Blocks, Imperavi Kube, Twitter Bootstrap, itd.

Zaključujemo z opozorilom, da je pri uporabi poizvedovanj v strežnikih na internetu (v članku smo uporabili strežnik RTV Slovenija) treba zagotoviti, da ne pride do zlorabe oz. do kršenja pravil uporabe.

Literatura

[1] MDN App Center. https://developer.mozilla.org/en-US/Apps *(13.7.2014)*

[2] Firefox OS – The Mobile Web Platform That Needs To Succeed. http://rominirani.com/2013/07/23/firefox-os-the-mobile-web-platform-that-needs-to-succeed/ *(13.7.2014)*

[3] David Sawyer McFarland. JavaScript & jQuery: The Missing Manual. O'Reilly Media, 2011, 2014.

[4] W3C Recommendation: Touch events. http://www.w3.org/TR/touch-events/ *(13.7.2014)*