Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Hribar

Razvoj medplatformne knjižnice za uporabo v mobilnih in spletnih aplikacijah

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Dejan Lavbič
MENTOR

Ljubljana, 2014

Namesto te strani se vstavi *original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!*

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom Dejana Lavbiča,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Miha Hribar, Ljubljana, maj 2014.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Hribar

Razvoj medplatformne knjižnice za uporabo v mobilnih in spletnih aplikacijah

POVZETEK

Razvoj aplikacij za več različnih platform je težaven. Odpira veliko možnosti za napake, oteži testiranje in odpravljanje napak, ter skoraj onemogoči sočasno nadgrajevanje aplikacij. Rezultat so dolgotrajni razvojni cikli in počasno dodajanje funkcionalnosti, kar v današnjem "startup" svetu ni zaželjeno.

Kljub različnosti med posameznimi platformami je ponavadi veliko kode z identično funkcionalnostjo, ki jo je potrebno razviti za vsako platformo posebej. Velikokrat je v podjetju za vsako od platform zadolžen drug razvijalec, še bolj pogosto pa razvoj na različnih platformah ne poteka sočasno. Rešitev iz te zagate je razvoj medplatformne knjižnice.

Cilj diplomske naloge je razvoj knjižnice za RRULE RFC5545[2] specifikacije, ki omogoča generiranje ponavljajočih se koledarskih dogodkov in jo je možno uporabiti v spletni, iOS, Android in Windows Phone aplikaciji. Našteli bomo možne pristope, navedli prednosti in slabosti, ter na koncu izbrali najbolj primerno rešitev za implementacijo knjižnice.

Kljucne besede: medplatformna knjižnica, iOS, Android, Windows Phone, JavaScript, Emscripten, LLVM

[2] http://tools.ietf.org/html/rfc5545#section-3.3.10

University of Ljubljana
Faculty of Computer and Information Science

Miha Hribar

Developing a cross platform library for use in mobile and web applications

ABSTRACT

Developing applications for different platforms is complicated. It opens a lot of avenues for mistakes, complicates testing and bugfixing, while almost completely destroys any chance of simultaneous application upgrade. The result of this are prolonged development cycles and slow feature creep, which in todays "startup" world is not an option.

Despite the differences between different plaftorms, they most likely share a lot of functionality which has to be developed for each platform. Most of the time each platform is handled by a different developer and usually not simultaneously with other applications. The solution to this problem is to develop a cross-platform library.

The goal of the thesis is to develop a library for the RRULE RFC5545[3] specification, which enables applications to schedule and display recurring events. The library will then be used in an web, iOS, Android and Windows Phone application. We will outline different approaches to writing the shared library, list the pros and cons and in the end decide on the best approach.

Key words: cross platform library, iOS, Android, Windows Phone, JavaScript, Emscripten, LLVM

[3] http://tools.ietf.org/html/rfc5545#section-3.3.10

ZAHVALA

Rad bi se zahvalil mentorju doc. dr. Dejanu Lavbiču za strokovno svetovanje in predvsem za potrpežjivost pri nastanku diplomskega dela. Hvala družini za vso podporo in finančno pomoč pri študiju. Hvala vsem ostalim, ki ste mi stali ob strani.

In seveda hvala Petri, ker mi pustiš da sem to kar sem.

— Miha Hribar, Ljubljana, maj 2014.

KAZALO

1 Uvod

Dandanes uporabljamo več različnih naprav sočasno. V lasti imamo najverjetneje prenosni računalnik, pametni telefon in po možnosti še tablico. Ko najdemo aplikacijo, ki nam je všeč, od te pričakujemo brezhibno delovanje na vseh naših napravah.

To je seveda zelo težko doseči, sploh z majhno ekipo. Dodatno se stvari zakomplicirajo, če so vse te naprave na različnih operacijskih sistemih. Tako imam lahko Windows prenosnik, Android telefon in Apple tablico. Kot razvajen uporabnik pričakujem, da je izbrana aplikacija na voljo na vse naštetih platformah.

Za razvijalca smo ravnokar opisali nočno moro. Da zadovolji potrebe uporabnikov, je primoran razviti isto aplikacijo za vsako od platform. Četudi omejimo razvoj na najbolj priljubljene platforme iOS, Android in Windows Phone, smo ravnokar našteli tri povsem različne tehnologije, tri različne jezike in s tem tri priložnosti za povsem različne težave pri implementaciji naše aplikacije. Veliko truda in energije je potrebno, da so te aplikacije poenotene in da skladno sledijo razvoju novih funckionalnosti.

Izkušen razvijalec bo pri predstavitvi problema takoj pomislil na medplatformni razvoj, ki si ga bomo ogledali v drugem poglavju. Omenili bomo tako imenovane “celotive”

metode, kot so Qt in Xamarain, “hibridne” kot sta recimo PhoneGap in Appcelerator Titanium, ter “deljene” metode npr. Lua, Haxe in C++.

V tretjem poglavju si bomo ogledali zakaj smo se odločili za razvoj knjižnice s pomočjo C++, ter jo tudi zgraditi. V četrtem poglavju jo bomo ovili v nekaj ovojev in uspešno uporabili v ObjC (iOS), Javi (Android) in C# (Windows Phone). Predstavili bomo tudi način kako C++ kodo prevesti neposredno v JavaScript s pomočjo orodja emscripten, ter na koncu knjižnico vključili tudi v spletno aplikacijo.

Pa začnimo.

2 Pregled metod medplaformnega razvoja

2.1 Celovit

Celovita metoda za razvoj uporablja eno samo okolje, s pomočjo katerega nato aplikacijo izvozimo za različne platforme - tako imenovani princip "piši enkrat, uporabljaj povsod", ki nam je znan iz programskega jezika Java. Zajema tako orodja za grafični vmesnik, kot prenosljivost programske kode.

2.1.1 Qt

Qt[1] je ogrodje za grafično programiranje za več platform s pomočjo jezika C++ in QML[2]. Omogoča nam sočasni razvoj za platforme OSX, Linux, Windows, Android in iOS. Podpira tudi uporabo HTML5[3] namesto QML, kar pomeni, da spletni razvijalci lahko uporabijo že obstoječe znanje in učenje novega jezika ni potrebno.

Qt projekt je povsem odprtokoden in dovoljuje uporabo v skladu z licencama GPL

[1] http://qt-project.org
[2] Qt Meta Language ali Qt Modeling Language, vir Wikipedia
[3] http://en.wikipedia.org/wiki/HTML5

v3[4] in LGPL v2.1[5], a če želite orodje uporabiti za razvoj mobilne aplikacije, boste morali za to odšteti 149$ mesečno.

Glavne slabosti Qt so neskladnost z izgledom ostalih aplikacij na mobilnih platformah, plačljiva licenca za razvoj mobilnih aplikacij ter končna velikost samih programov.

2.1.2 Xamarin

Xamarin[6] je ogrodje za sočasen razvoj aplikacij za platforme iOS, Android, Mac in Windows v jeziku C#. Izjaha iz projekta Mono[7], ki omogoča uporabo ogrodja .NET na različnih platformah. Cenovno ni ravno ugoden, saj se cene začnejo pri 299$/mesec za vsakega razvijalca in vsako platformo.

Ogrodje omogoča razvoj aplikacij, katerih izgled je skladen z ostalimi aplikacijami na izbrani platformi. Kot primer si lahko ogledamo aplikacijo za poslušanje glasbe Rdio[8], ki je na voljo za iOS, Android in Windows Phone.

Glavna slabost ogrodja Xamarin je cena. Za majhno ekipo je začetna cena enostavno previsoka. Vprašljiva je hitrost dodajanja funckionalnosti posameznih platform, ko se te nadgradijo, določen riziko predstavlja tudi muhavost posameznih platform pri omejitvah uporabe tega ogrodja.

2.1.3 Adobe Air

Adobe Air je brezplačno ogrodje, ki omogoča zagon iste aplikacije na platformah iOS, Android, Mac, Windows in Linux. Čeprav za razvoj namiznih aplikacij omogoča uporabo HTML in Javascript, je za razvoj mobilnih aplikacij omejen na uporabo jezika ActionScript. V času pisanja diplomske naloge ogrodje ne omogoča zagon na platformi Windows Phone.

Kot glavno slabost ogrodja Adobe Air bi navedel upadanje zanimanja za orodje Flash. Špekuliramo lahko tudi o planih podjetja Adobe, saj so pred kratkim kupili podjetje Nitobi, ki je avtor ogrodja PhoneGap (katerega si ga bomo ogledali v nadaljevanju).

[4] http://www.gnu.org/copyleft/gpl.html
[5] https://www.gnu.org/licenses/old-licenses/lgpl-2.1.html
[6] https://xamarin.com
[7] http://www.mono-project.com
[8] https://www.rdio.com

2.2 Hibriden

Hibridna metoda za razvoj aplikacij uporablja spletne tehnologije v sožitju z kodo za posamezno platformo (t.i. premostitvena tehnika), ki omogoča dostop do glavnih funkcij naprav (kot so kamera, pospeškomer in podobno).

2.2.1 Apache Cordova / PhoneGap

Ogrodje Apache Cordova[9] je odprtokodni projekt, ki omogoča objavo spletnih aplikacij kot domorodne. V času pisanja diplomske naloge ogrodje podpira iOS, Android, Windows Phone, Blackberry, Palm WebOS, Bada in Symbian. Na vseh omenjenih platformah nam ogrodje Apache Cordova omogoča dostop do funkcij naprave, ko so naprimer kamera in pospeškomer.

Projekt PhoneGap[10] je dejansko samo ena od distribucij projekta Apache Cordova, ki poleg vseh obstoječih funkcionalnosti ponuja tudi razne storitve na katerih delajo v podjetju Adobe.

Za razvoj aplikacij razvijalci lahko uporabljajo spletne tehnologije HTML, CSS in JavaScript. S pomočjo ogrodij jQuery Mobile in Sencha Touch je možno izdelati aplikacije, katerih izgled je zelo lep približek ostalim aplikacijam na izbrani platformi. Če naletimo na funkcijo naprave, do katere nimamo dostopa, ali ugotovimo da je JavaScript za določene naloge premalo učinkovit, lahko preprosto spišemo lasten vtičnik, ki služi kot most med JavaScript in domorodno kodo.

Glavna prednost ogrodja Apache Cordova in predvsem distribucije PhoneGap je izredno nizka pregrada. Priporoča se predvsem za izdelavo prototipnih aplikacij, saj nam omogoča hiter razvoj in iteracijo.

Glavna slabost tega pristopa tiči v performanci in odzivnosti aplikacije, saj ta za prikazovanje izkorišča vgrajeno spletno okno. Trenutno je težko izdelati aplikacije, ki so grafično zahtevnejše, kar pomeni še toliko bolj pereč problem na napravah s slabšimi karakteristikami.

2.2.2 Appcelerator Titanium

Ogrodje Titanium nam omogoča izdelavo aplikacij za več platform hkrati s pomočjo JavaScript okolja, ki služi kot abstrakcijska plast med našo aplikacijo in domorodno

[9] http://cordova.apache.org
[10] http://phonegap.com

kodo. Aplikacijo gradimo s pomočjo jezika JavaScript, ki se med uporabo aplikacije izvaja s pomočjo V8 (Android) in JavaScriptCore (iOS) ali vgrajenega JavaScript okolja (če aplikacijo poganjamo v brskalniku). Za pravilen vizualen izgled skrbijo namestniški elementi, ki uporabljajo domorodne grafične elemente, kar pomeni da vizualno aplikacije ne ločimo od ostalih domorodnih aplikacij. V času pisanja diplomske naloge ogrodje podpira iOS, Android, Blackberry, Tizen in spletne aplikacije.

Glavna prednost ogrodja Titanium ni t.i. način piši enkrat, uporabljaj povosd; njegova prednost je da lahko celotno aplikacijo izdelamo v enem jeziku - JavaScript-u. Le redko se bomo srečali z domorodno kodo, saj ogrodje nudi široko paleto knjižnic.

Glavna slabost ogrodja je počasno dodajanje novih platform zaradi obsežnosti dela, ki ga tak podvig zahteva. Določene knjižnice za delo z domorodnimi elementi tudi niso najbolj performančne, manjka pa tudi napovedana podpora platformi Windows Phone.

2.3 Deljen

Deljena metoda za razvoj aplikacij omogoča uporabo dela aplikacijske kode na vseh platformah za katere razvijamo. To lahko naredimo s pomočjo vgradnega skriptnega jezika (Lua), s pomočjo prevajanja iz izbranega programskega jezika v domorodnega (Haxe, XMLVM, emscripten) ali pa z uporabo programskega jezika C++ in programskimi ovoji, s katerimi pripravimo knjižnico za vgradnjo v druge platforme.

2.3.1 Lua

Lua[11] je preprost vgrandi skriptni jezik, ki ga odlikuje hitrost izvajanja in procesorska nezahtevnost, kar pomeni, da je kot nalašč za hitro izdelavo prototipov. Vgradimo ga lahko v platforme Android, iOS, Symbian in Windows Phone, z nekaj potrpljenja pa lahko isto kodo zaženemo tudi v spletni aplikaciji.

Čeprav je jezik Lua preprost za uporabo, se izkaže da za kompleksnejše knjižnice ni primeren. Manjka tudi Unicode podpora in boljša podpora rokovanju z napakami.

Standardna knjižnica?

[11] http://www.lua.org

2.3.2 Haxe

Haxe[12] zase pravi, da je večplatformski programski jezik. Razvijalec lahko svojo aplikacijo napiše v jeziku Haxe, nato pa jo s pomočjo prevajalnika prevede v JavaScript, C++, C# ali Javo.

Pri prevajanju aplikacije v ciljni jezik Haxe povzroči ne tako zanemarljivo napihjenje kode.

2.3.3 XMLVM

XMLVM[13] spada v isti razred kot Haxe - tako imenovanih prevajalcev iz enega jezika v drugega (ang. cross-compilers), a se XMLVM tega loti na drugačen način. Medtem ko Haxe prevaja na nivoju izvorne kode, XMLVM to počne na nivoju zlogovne kode (ang. byte code). Izvorna koda je lahko napisana za navidezne stroje (ang. virtual machine) JVM, .NET CLI ali Ruby YARV, medtem ko je rezultat delujoč program za JVM, .NET CLI, Javascript, Pyhton, Objective-C in C++.

// slika JVM, .NET CLI, Ruby YARV -¿ JVM, .NET CLI, JavaScript, Python, Objc, C++

Projekt izgleda zelo ambiciozen, a vse kaže da je šlo le za akademsko raziskavo, saj je v času pisanja diplome minilo že več ko leto dni odkar se je izvorna koda posodobila. Kljub temu se mi je projekt zdel zanimiv.

2.3.4 C++ in emscripten

V kolikor nobena od naštetih možnosti ne zadošča našim potrebam, želeli pa bi vseeno imeti deljeno knjižnico, ki bi jo bilo možno vgraditi v iOS, Android, Windows Phone in uporabiti v spletni aplikaciji, obstaja še ena možnost: C++ in emscripten.

C++[14] je eden izmed najbolj razširjenih programksih jezikov. V času pisanja diplomske naloge zaseda četrto mesto na lestvici najbolj popularnih jezikov[15].

Emscripten[16] je projekt Mozilinih laboratorijev, in omogoča prevajanje iz LLVM[17] zlogovne kode v skriptni jezik JavaScript.

[12] http://haxe.org
[13] http://xmlvm.org
[14] http://www.cplusplus.com
[15] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[16] https://github.com/kripken/emscripten
[17] Low Level Virtual Machine

3 Razvoj knjižnice

3.1 C++

3.2 Emscripten

3.3 Omejitve

4 Vključitev knjižnice v različne platforme

5 Ugotovitve

Literatura

A Appendix A

A.1 Intro section

All material unsuitable for the main part of the work (e.g. source code listings, extensive proofs, ...) should be placed in the Appendix.

A.2 ...