Gimnazija Vič

Tržaška c. 72, 1000 Ljubljana

Maturitetna seminarska naloga pri predmetu informatika

Programiranje za mobilne platforme iOS

Avtor:
Vid Drobnič

Mentor:
prof. Klemen Bajec

februar 2017

Povzetek

Projektna naloga se ukvarja z ravojem aplikacije GimVic za mobilne naprave z operacijskim sistemom iOS. Namen aplikacije je dijakom in profesorjem Gimnazije Vič olajšati pregledovanje urnika in suplenc. Projektna naloga ima tri dele. Prvi del seminarske naloge se ukvarja z operacijskim sistemom iOS in razvojem aplikacij zanj. Drugi del se ukvarja s samo izdelavo GimVic aplikacije. Ker je aplikacija razmeroma obsežna, ni primerno, da bi bila predstavljena v celoti, zato so podrobneje razloženi samo nekateri pomembnejši deli. Tretji del opisuje objavo aplikacije na spletni trgovini AppStore in odziv uporabnikov.

Ključne besede: Swift, iOS, AppStore, aplikacija

Kazalo

1 Uvod

Čez celo šolsko leto imamo dijaki veliko nadomeščanj, ki so dostopna na spletni učilnici. Zaradi nepreglednosti spletne učilnice je pregledovanje nadomeščanj zelo dolgotrajno opravilo. Zato sem se odločil napisati aplikacijo za mobilne naprave z operacijskim sistemom iOS[6].

Pred začetkom pisanja aplikacije sem si zadal določene cilje. Poleg prikazovanja urnika z nadomeščanji, sem si kot cilj zastavil tudi prikazovanje jedilnika. Vse zastavljene cilje sem tudi dosegel. Na koncu sem preveril še priljubljenost aplikacije s pomočjo statističnih podatkov, ki sem jih pridobil na Apple-ovi spletni strani za razvijalce.

2 Operacijski sistem iOS

iOS[6] je Applov operacijski sistem a mobilne naprave. V osnovi je bil zasnovan za iPhone, danes pa ga uporabljajo tudi iPad, iPod Touch in Apple TV. Prva verzija je bila izdana leta 2007. Zasnovan je na Unix-ovem[14] jedru.

2.1 Programiranje za iOS platformo

Apple je za iOS razvil razvijalsko orodje Xcode[15], ki je na voljo za računalnike z operacijskim sistemom macOS. Naredili so tudi primerno dokumentacijo[1] za programiranje za iOS in spletno trgovino imenovane App Store[3], kamor razvijalci objavimo svoje aplikacije.

2.1.1 Jeziki

Čez leta se je zamenjalo veliko jezikov za pisanje aplikacij za iOS platformo. Prvotno so se aplikacije pisale v C++[4]. Kasneje ga je zamenjal programski jezik Objective-C[10], ki je bil uporabljen samo na Applovih platformah. Leta 2014 je Apple predstavil svoj odprtokodni programski jezik imenovan Swift[13], ki naj bi zamenjal sedaj nekoliko zastarel Objective-C[10].

2.1.2 Zgradba iOS Aplikacije

Aplikacije za iOS se shranjujejo v Xcode projektih. To je skupina map in datotek z določeno strukturo. Prevladujejo Swift oziromva Objective-C (odvisno od tega kater jezik si izberemo za razvoj), .storyboard, .xib in .plist datoteke. V .storyboard in .xib datotekah se nahaja zgradba uporabniškega

vmesnika, v .plist datotekah se nahaja konfiguracija aplikacije (ime aplikacije, ime razvijalca, ...), v Swift oziroma Objective-C datotekah pa se nahaja koda, ki se kasneje izvaja na napravi.

Aplikacije je sestvaljena na modelu MVC[8] (Model-View-Controller). Model nam predpiše tri glavne komponente aplikacije. Komponenta Model skrbi za pridobivanje, shranjevanje in obdelavo podatkov, komponenta View skrbi za prikazovanje grafičnega vmesnika, komponenta Controller pa povezuje Model in View med seboj. Ko uporabnik pritisne na določeno del uporabniškega vmesnika, gre Controller po podatke v Model objekt in nato pove View objektu katere podatke naj prikaže.

3 Razvoj aplikacije

Da bi bila aplikacija čim bolj prijazna uporabniku je šla čez veliko različic in popravkov. V nadaljevanju je opisano kako deluje trenutno najnovejša različica 3.0. Izvorna koda je dostopna na spletnem portalu GitHub na naslovu `https://github.com/GimVic-app/gimvic-ios`.

3.1 Struktura

Aplikacija je sestavljena iz dveh delov. Prvi del se ukvarja s pridobivanjem podatkov iz strežnika, drugi del pa se ukvarja s predstavitvijo podatkov uporabniku. Del aplikacije, ki se ukvarja s pridobivanjem in shranjevanjem podatkov je narejen kot knjižnica (Framework), ki jo aplikacija uporablja. S tem lahko kasneje dodam še druge načine prikazovanja urnika uporabniku (na primer gradnik (*ang.* widget)), brez da bi ponovno pisal pridobivanje podatkov.

3.2 Podatki

Podatki, ki jih aplikacija potrebuje, so servirani iz različnih strežnikov v različnih oblikah zapisa. Urnik je na voljo v JavaScript Array[16] obliki, nadomeščanja pa v JSON[7]. Jedilnik za šolsko malico in kosilo se naloži na strežnik v csv[5] obliki. Odločil sem se, da bo za podatke skrbel poseben strežnik, ki bo vse od zgoraj naštetega združil v enovito JSON obliko, ker bo tako prikazovanje na telefonu hitrejše in bolj učinkovito. Strežnik si podatke shranjuje v MySQL[9] podatkovno bazo[2] in jih nato servira v JSON obliki.

Pri dostopu do podatkov strežniku z URL parametri[12] podaš ali želiš podatke za profesorja, ali pa za dijaka. Pri profesorju strežniku kot parameter podaš ime profesorja, pri dijaku pa razred in izbirne ozroma maturitetne predmete. Strežnik nato vrne seznam, kjer vsak element v tem seznamu predstavlaj podatke za določen dan v tednu. Vsak dan je predstavljen z seznamom v katerem vsak element predstavlja uro.

Primer URL-ja:

`http://gimvicapp.404.si/data?addSubstitutions=true&classes[]=4B&snackType=navadna&lunchType=navadno`

3.2.1 Upravitelj podatkov

Glaven razred v knjižnici za delo s podatki je upravitelj podatkov, ki skrbi za pridobivanje podatkov iz strežnika, njihovo obdelovanje in shranjevanje. Zažene se vsakič ko se aplikacija odpre in najprej preveri starost trenutno shranjenih podatkov. Če so podatki prestari gre na strežnik po nove podatke, ki jih pretvori v primerne podatkovne strukture in shrani na pomnilnik telefona. Nato o novih podatkih obvesti glavno aktivnost, ki osveži grafični vmesnik z novimi podatki.

Če upravitelj podatkov ugotovi, da naprava nima internetne povezave uporabi podatke, ki so že shranjeni na napravi. Uporabnik lahko podatke osveži ročno. V tem primeru glavna aktivnost zahteva od upravitelja podatkov, da gre po nove podatke.

Upravitelj podatkov je `singleton`, kar pomeni, da je inicializiran samo en objekt, ki ga uporabljajo vsi drugi objekti. S tem se izognemo podvojevanju podatkov in nesočasnega osveževanja, zaradi česar aplikacija porabi manj spomina in prenese manj podatkov.

Vsak objekt, ki želi od upravitelja podatkov prejemati sporočila o spremembah podatkov, se mora dodati kot delegat (*ang.* delegate). Upravitelj podatkov ima za shranjevanje delegatov mapo (*ang.* dictionary). Vsak delegat je shranjen v mapi pod svojim imenom, ki predstavlja ključ. S tem se lahko kasneje objekt pri zbrisevanju odstrani iz upravitelja podatkov.

Inicializacija singleton objekta in shranjevanje delegatov:

```
public final class TimetableData {
  public static let sharedInstance = TimetableData()

  public var delegates = [DelegateID: TimetableDataDelegate]()
  ...
}
```

Ker morajo biti vsi objekti v mapi istega tipa, izkoristimo protokole. V Swiftu s protokolom definiramo, katere funkcije in spremenljivke mora objekt imeti. Protokol se nato obnaša kot tip spremenljivke. S tem zagovotimo da imajo vsi shranjeni objekti na voljo funkcijo, ki jo kličemo, ko se spremenijo podatki.

Deklaracija protokola:

```
public protocol TimetableDataDelegate {
  func timetableDataDidUpdateWithStatus(_ status: DataGetterStatus)
}
```

Zelo moramo paziti, da se vsak objekt, ki se doda kot delegat, tudi odstrani. Drugače lahko do objekta dostopamo preko večeih kazalcev (*ang.* pointer), kar pomeni, da se objekt nikoli ne bo odstranil iz pomnilnika. S tem ustvarimo puščanje pomnilnika (*ang.* memory leak).

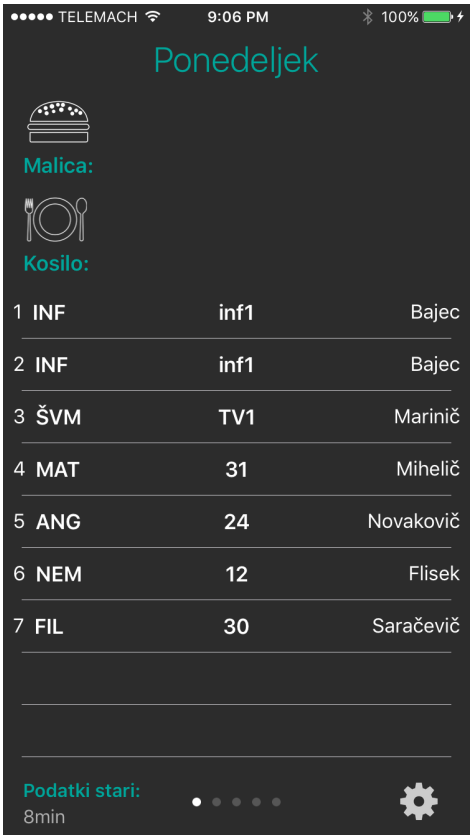
3.3 Predstavitev podatkov

Del aplikacije, ki skrbi za predstavitev podatkov uporabniku, je sestavljen iz večih komponent:

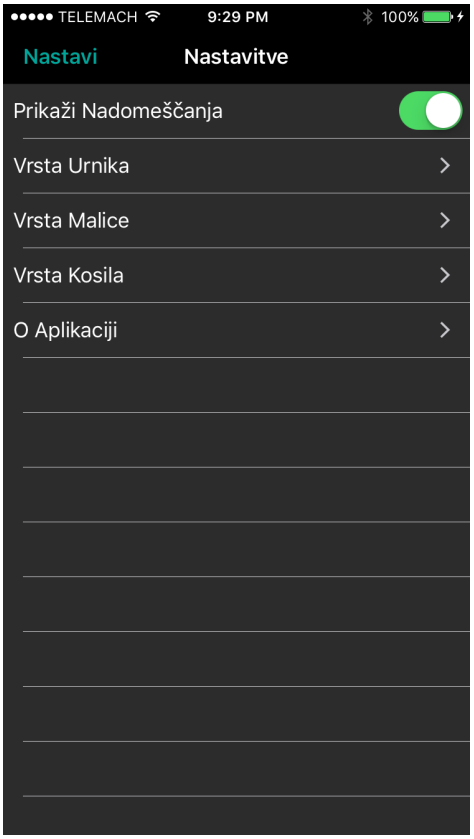
- glavna aktivnost (*ang.* root view controller) (slika 1)
- nastavitve (slika 2)
- o aplikaciji

3.3.1 Glavna aktivnost

Tu se odvija večino uporabniku pomembnih stvari. Ko uporabnik odpre aplikacijo se mu prikaže glavni pogled (*ang.* main view) (slika 1) v katerem je za vsak dan prikazan jedilnik in urnik. V glavni aktivnosti se nahaja tudi gumb za nastavitve.



Slika 1: Glavna aktivnost



Slika 2: Nastavitve

Ko se aplikacija odpre, najprej preveri če je odprta prvič. V tem primeru uporabnika popelje čez osnovne nastavitve, kjer nastavi razred oziroma ime profesorja, ter vrsto malice in kosila.

Preverjanje če je aplikacija odprta prvič:

```
override func viewDidAppear(_ animated: Bool) {
  super.viewDidAppear(animated)

  if UserDefaults().object(forKey:
                           UserSettings.
                           lastRefreshedTimetableData.
                           rawValue) as? Date == nil {
    let setupStoryboard = UIStoryboard(name: "Setup", bundle: nil)
    let viewController = setupStoryboard.instantiateInitialViewController()!
```

```
    present(viewController, animated: true, completion: nil)
  }
}
```

Aplikacija nato prikaže vse elemente uporabniška vmesnika. V osnovi je sestavljen iz `UIScrollView`-ja, ki skrbi zato, da se lahko premikamo levo in desno. `UIScrollView` ima več otrok, kjer vsak otrok predstavlja pogled(`UIView`) za en dan v tednu. Otrok ima eno tekstovno polje (`UILabel`) v katerem je zapisan dan v tednu in še dve tekstovni polji v katerih sta zapisana malica in kosilo. Vsak dnevni pogled vsebuje tudi tabelo (`UITableView`), v kateri je prikazan urnik.

V glavni aktivnosti sta prikazana tudi gumb (`UIButton`) za nastavitve in kako stari so podatki. Starost podatkov nastavimo, ko se aplikacija prikaže na zaslon. Nato naredimo nov merilnik časa (`NSTimer`), ki vsako minuto kliče določeno funkcijo, ki posodobi starost podatkov.

Funkcija za prikazovanje starosti podatkov:

```
func setDataAgeLabel() {
  let lastRefreshed = UserDefaults().
                      object(forKey:
                             UserSettings.
                             lastRefreshedTimetableData.
                             rawValue) as? Date
  if lastRefreshed == nil {
    dataAgeLabel.text = "N/A"
  } else {
    let ageText = FuzzyDate.timeSince(lastRefreshed!)
    dataAgeLabel.text = ageText
  }
}
```

3.3.2 Nastavitve

V nastavitvah (slika 2) lahko uprabnik nastavi ali je dijak, ali profesor in temu primeren filter za urnik. Nastavi lahko tudi vrsto malice in kosila na katerega je naročen. Poleg tega ima tudi možnost, da izklopi funkcijo za prikaz nadomeščanj, tako da je uporabniku prikazan samo urnik. V nastavitvah lahko uporabnik tudi pogleda podatke o aplikaciji.

Nastavitve se na zaslon prikažejo kot modalno okno (*ang.* Modal View). Ko uporabnik zapre nastavitve z pritiskom na gump *Nastavi*, nastavitve sporočijo upravitelju podatkov, naj gre po nove podatke.

Izhod iz nastavitev:

```
@IBAction func doneButtonPressed(_ sender: AnyObject) {
  UserDefaults().synchronize()
  TimetableData.sharedInstance.update()
  navigationController?.dismiss(animated: true, completion: nil)
}
```

3.3.3 O aplikaciji

V tem meniju se uporabniku prikažejo osnovne informacije o aplikaciji kot so avtor aplikacije, verzija in pod katero licenco[11] je aplikacija izdana. Prav tako je v tem meniju zapisan vir nekaterih ikon, ki so uporabljene, saj to predpisuje licenca pod katero so izdane te ikone.

4 Objava aplikacije

Ko sem aplikacijo dokončal, sem jo še objavil v spletni trgovini App Store. Gre virtualno trgovino po kateri uporabniki iOS pregledujejo različne aplikacije in jih naložijo na svoje naprave

Za objavo aplikacije na App Store potrebujemo razvijalski račun za Applove naprave, s katerim se prijavimo v njihov spletni portal. Tam naredimo novo aplikacijo in jim dodelimo unikatno ime, ki ga vidi samo razvijalec (*npr.* com.gimvic.ios). Aplikaciji dodamo tudi slike in opis, lahko pa naložimo tudi promocijski video.

Pred objavo moramo na razvijalski portal naložiti tudi samo aplikacijo - to je datoteka, ki se dejansko izvaja na napravi. To naredimo preko razvijalskega orodja Xcode, v katerem smo aplikacijo tudi napisali. V Xcode moramo biti prijavljeni z razvijalskim računom, v konfiguraciji pa mora imeti aplikacija isto unikatno ime, kot smo ji ga dodelili v spletnem portalu. Nato lahko z gumbom *Publish* aplikacijo naložimo na portal.

5 Odziv uporabnikov

Applov spletni portal omogoča analizo statističnih podatkov o aplikiciji. To vkjučuje število namestitev na naprave, število aktivnih uporabnikov in pa tudi podatke bolj pomembne za ravijalce, kot so število zrušitev aplikacije.

Trenutno ima aplikacija 139 prenosov. Do velike spremembe bo prišlo na začetku šolskega leta, ko bodo novi dijaki naložili aplikacijo.

6 Zaključek

Aplikacijo GimVic za iOS uporabnike sem napisal z namenom, da bi profesorjem in djakom Gimnazije Vič olajšal pregledovanje urnika, nadomeščanj in jedilnika. Aplikacija iz strežnika dobiva realne podatke in jih nato prikaže uporabniku na prijazen in uporaben način. Dostopna je vsem uporabnikom iOS naprav na spletni trgovini App Store.

Glede na to, da si je aplikacijo preneslo večino profesorjev in dijakov z iOS napravo in jo tudi aktivno uporabljajo, sem dosegel zastavljen cilj. Prav tako sem zelo zadovoljen s pozitivnim odzivom uporabnikov aplikacije. Sama aplikacija ima še nekaj manjših napak, ki jih bom v prihodnosti popravil.

V prihodnosti se lahko aplikaciji dodajo tudi dodatne funkcije, kot so gradniki (*ang.* widgets) in obvestila (*ang.* notifications) o nadomeščanjih.

7 Literatura

[1] Apple. *Apple: iOS Documentation*. URL: https://developer.apple.com/reference (pridobljeno 2017).

[2] Andrej Brodnik, Luka Fürst, Alenka Krapež in sod. *Računalništvo in Informatika 1*. 2015. URL: lusy.fri.uni-lj.si/ucbenik/.

[3] Wikipedia The Free Encyclopedia. *App Store*. URL: https://en.wikipedia.org/wiki/App_Store_(iOS) (pridobljeno 2017).

[4] Wikipedia The Free Encyclopedia. *C++*. URL: https://en.wikipedia.org/wiki/C%2B%2B (pridobljeno 2017).

[5] Wikipedia The Free Encyclopedia. *csv*. URL: https://en.wikipedia.org/wiki/Comma-separated_values (pridobljeno 2017).

[6] Wikipedia The Free Encyclopedia. *iOS*. URL: https://en.wikipedia.org/wiki/IOS (pridobljeno 2017).

[7] Wikipedia The Free Encyclopedia. *JSON*. URL: https://en.wikipedia.org/wiki/JSON (pridobljeno 2017).

[8] Wikipedia The Free Encyclopedia. *MVC*. URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (pridobljeno 2017).

[9] Wikipedia The Free Encyclopedia. *MySQL*. URL: https://en.wikipedia.org/wiki/MySQL (pridobljeno 2017).

[10] Wikipedia The Free Encyclopedia. *Objective-C*. URL: https://en.wikipedia.org/wiki/Objective-C (pridobljeno 2017).

[11] Wikipedia The Free Encyclopedia. *Open-source license*. URL: https://en.wikipedia.org/wiki/Open-source_license (pridobljeno 2017).

[12] Wikipedia The Free Encyclopedia. *Query string*. URL: https://en.wikipedia.org/wiki/Query_string (pridobljeno 2017).

[13] Wikipedia The Free Encyclopedia. *Swift*. URL: https://en.wikipedia.org/wiki/Swift_(programming_language) (pridobljeno 2017).

[14] Wikipedia The Free Encyclopedia. *Unix*. URL: https://en.wikipedia.org/wiki/Unix (pridobljeno 2017).

[15] Wikipedia The Free Encyclopedia. *Xcode*. URL: https://en.wikipedia.org/wiki/Xcode (pridobljeno 2017).

[16] w3schools. *JavaScript Array*. URL: https://www.w3schools.com/js/js_arrays.asp (pridobljeno 2017).