I. gimnazija v Celju

Razvoj mobilne aplikacije za urejanje prijav na malico

Avtor:
Aljaž Pirc, 4.b

Mentorica:
Mojca Plevnik Žnidarec, univ. dipl. inž. kem. teh.

Področje: Računalništvo - storitev z izdelkom

Mestna občina Celje, Mladi za Celje
Celje, 2016

Kazalo

Kazalo slik

Zahvala

Rad bi se zahvalil svoji mentorici za vso pomoč in podporo pri mojem projektu, kakor tudi pri izdelavi raziskovalne naloge. Prav tako se želim zahvaliti vsem ostalim, ki so mi na kakršen koli način pomagali in me podpirali pri raziskovanju.

Povzetek

Za izdelavo te raziskovalne naloge sem se odločil, ker menim, da se je potrebno prilagoditi modernim tendencam vedno večje uporabe mobilnih naprav. Poleg nekaterih negativnih lastnosti, ki so velikokrat rade izpostavljene in žal tudi obstajajo, imajo mobilne naprave ogromno prednosti. Odločil sem se razviti aplikacijo, ki bo dijakom naše gimnazije, in še morda kdaj kakšne druge šole, poenostavila proces prijave in odjave na malico ter zamenjavo naročenih obrokov. Razvil sem dve verziji aplikacije in sicer za operacijska sistema iOS in Android. Med njima je sicer nekoliko razlik, vendar sta se obe verziji izkazali kot zelo uporabni. Poleg tega sem si zamislil in si ustvaril koncept aplikacije, ki bi poleg urejanja malic omogočala še nekatere druge funkcije, ki postajajo v današnjem času, v šoli, vedno pomembnejše.

1 Uvod

V današnjem času ima v našem življenju vse večjo vlogo internet, ki nam olajšuje medsebojno komunikacijo. V povezavi z internetom so v vzponu ultra majhni, ultra prenosni računalniki, kot so pametni telefoni in tablice, ki jih ima dandanes že prav vsak. Tem novim trendom pa bi morale slediti tudi vse organizacije, ne samo podjetja, ki se ukvarjajo s proizvodnjo le-teh, temveč tudi šole, bolnišnice, uradi in tako naprej.
Kot dijak gimnazije, lahko poskusim prispevati nekaj v povezavi s šolo, saj se vsakodnevno srečujem z morebitnimi težavami na tem področju. Osredotočil sem se na področje, ki ima po mojem mnenju manj prilagojen sistem sodobnim napravam, to je naročanje malic na naši šoli. o sedaj sta bili na voljo dve možnosti. Prva je uporaba terminala na šoli, ki je najbolj priljubljena med dijaki, druga pa preko spleta, ki pa ni priljubljena, saj je zelo zapletena.
V svoji rešitvi sem želel združiti preprostost, kot tudi priročnost in kaj to ponuja bolje, kot mobilne aplikacije. Pri izdelavi teh, je najpomembnejša izvrstna uporabniška izkušnja, ki se jo doseže s preprostim in estetsko oblikovanim ter izjemno uporabnim uporabniškim vmesnikom. Aplikacijo sem prilagodil za najbolj razširjena mobilna operacijska sistema, iOS in Android.
Prva ideja za takšno aplikacijo se mi je porodila ob sodelovanju pri projektu Junaki Prihodnosti, katerega cilj je bil trajnostno rešiti neko težavo v lokalnem okolju. V okviru tega projekta je bila tudi izvedena anketa, ki je vključevala vprašanje o zainteresiranosti dijakov za takšno aplikacijo. Ob več kot 90 odstotnem pozitivnem odzivu skorajda nisem več imel druge možnosti kot lotiti se izdelave te aplikacije.

1.1 Opredelitev problema

Cilj te raziskovalne naloge je razviti dve aplikaciji, ki bosta namenjeni odjavi in prijavi na malico na naši šoli, kasneje morda še kje drugje in bosta prilagojeni vsaka na svoj operacijski sistem. Pomembno spoznanje, je bil namreč podatek, pridobljen v projektu Junaki Prihodnosti o veliki količini neprevzetih malic, saj se dijaki očitno zaradi omejitev obstoječega sistema preprosto ne odjavljajo. Zanimalo me je, kako se lahko ta rešitev obnese na mobilnih napravah in kako priročna je. Hkrati me je zanimala primerjava s starim sistemom, ki je narejen kot spletna storitev in ni niti vizualno prilagojen na mobilne brskalnike kar pomeni, da je lahko njegova uporaba v tem okolju prav neprijetna.

1.2 Razlaga uporabljenih izrazov

- iOS - operacijski sistem, razvit s strani podjetja Apple, ki se uporablja na napravah iPhone, iPad in iPod touch.

- Xcode - razvojno okolje, razvito s strani Appla, ki se uporablja za razvoj aplikacij za iOS, Mac OS X, tvOS in watchOS. Na voljo je samo za Mac OS X. Ima llmv compiler za programska jezika Objective-C in swift oziroma novi swift 2.0. Omogoča tudi uporabo drugih compilerjev.

- Android - odprtokodni operacijski sistem podjetja Google, ki je uporabljen na večini mobilnih naprav, velikokrat v nekoliko modificirani verziji.

- Android studio - razvojno okolje, razvito s strani podjetja Google, ki je namenjena razvoju aplikacij za razvoj aplikacij za operacijski sistem Android. Na voljo je za operacijske sisteme Linux, Mac OS X in Windows. Programski jezik, ki se uporablja je Java, za prevajanje pa se uporablja javac compiler, razvit s strani podjetja Oracle.

- Compiler - računalniški program, ki prevede izvorno kodo v strojno kodo, ki jo računalniški procesorji lahko izvedejo.

- Aplikacija - skupek izvedljivih in ostalih pomožnih datotek (slike…), ki tvorijo računalniški program.

- XML - preprost računalniški (označevalni) jezik, ki se uporablja za opisovanje strukturiranih podatkov.

- Razred - programski element v objektno orientiranih programskih jezikih, ki opisuje obnašanje in lastnosti nekega objekta, ki je del programa.

- UIViewController tudi controller - sistemski razred v razvojnem okolju za iOS, ki poskrbi za strukturo za urejanje pogledov v iOS aplikaciji. Ureja skupino pogledov, ki tvorijo del uporabniškega vmesnika aplikacije.

- Activity tudi aktivnost - je vsebina aplikacije operacijskega sistema Adnroid, ki priskrbi zaslon na katerem lahko uporabnik izvede neko dejanje.

- Fragment - predstavlja del uporabniškega vmesnika v aktivnosti. V eni aktivnosti se lahko nahaja več fragmentov hkrati.

- Linear Layout - skupina pogledov, ki razporedi vse elemente, ki jih vsebuje vertikalno ali horizontalno.

- Auto Layout - tehnnologija razvojnega okolja za iOS, ki dinamično (s samodejnimi prilagoditvami na spremembe, kot so zasuk zaslona) izračuna dimenzije pogledov in jih nato postavi v hierarhijo na zaslon. Izračun je narejen na osnovi pravil, ki so podana v vizualnem vmesniku ali v programski kodi s pomočjo jezika Visual Format Language.

1.3 Hipoteze

- Predvidevam, da je mobilno okolje veliko bolj primerno za preprosta dejanja, kot je odjava malice, kot pa namizno. Prižiganje namiznega računalnika je lahko daljše kot dejanska uporaba, zato so bolj primerne naprave, ki so prižgane ves čas.

- Predvidevam, da lahko aplikacija uporabniku omogoči prijaznejši način upravljanja dejavnosti, kot pa spletna storitev, zaradi popolne prilagojenosti na operacijski sistem.

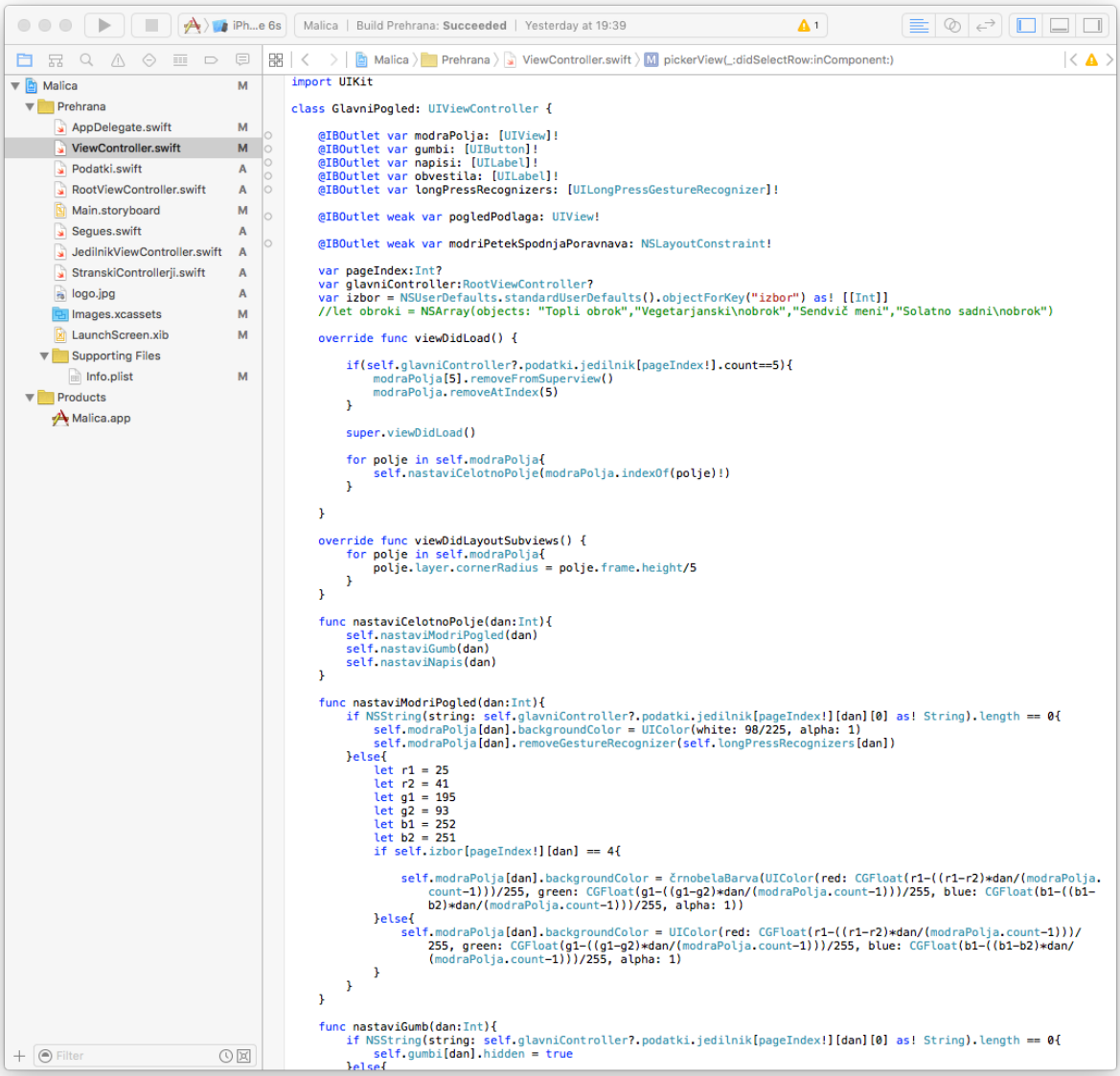
- Na osnovi prejšnjih hipotez sklepam, da bodo končen produkt, njegov namen in izvedba uporabniku primernejši in prijaznejši, kot je obstoječa rešitev.

1.4 Raziskovalne metode

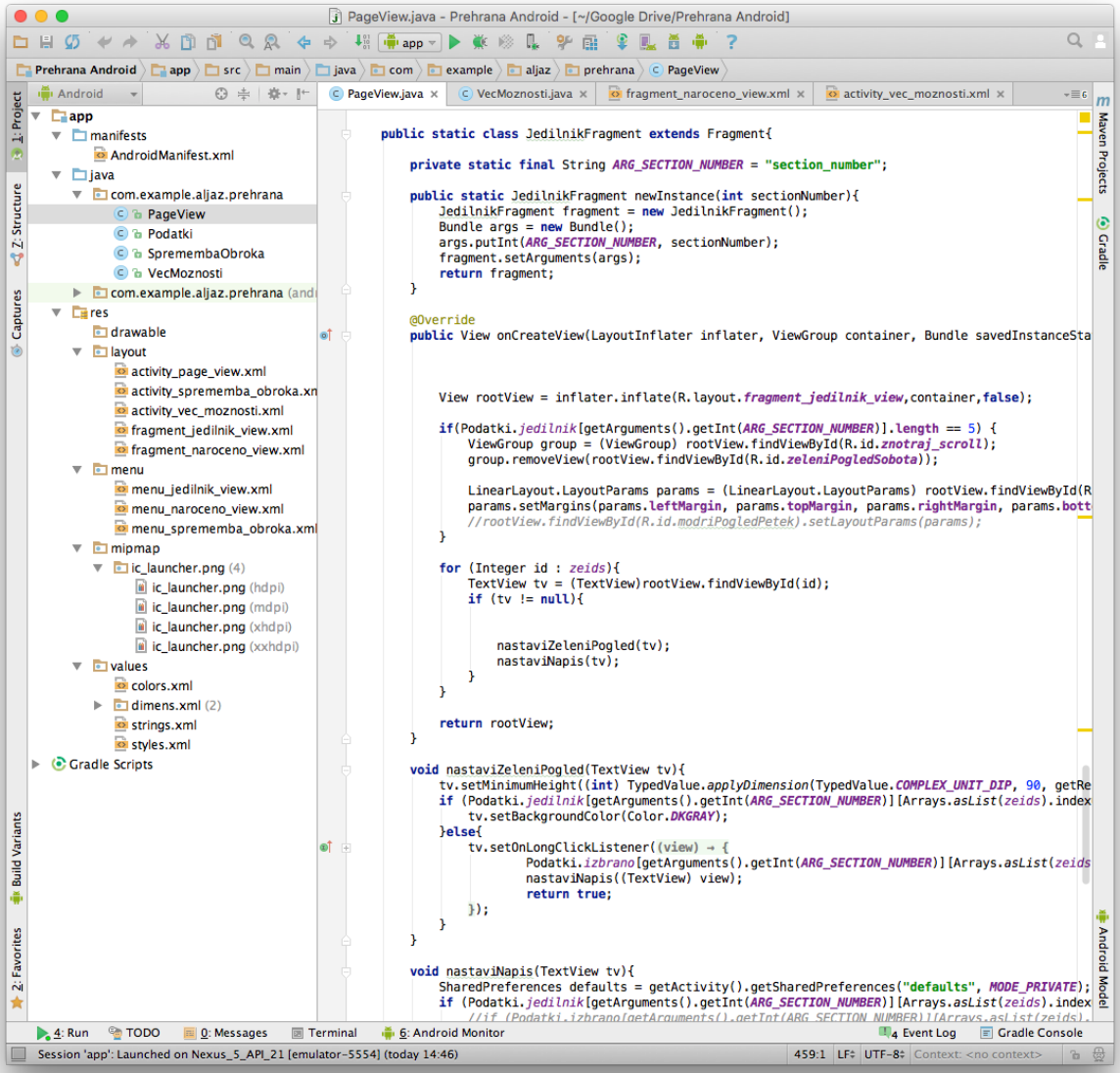
Najpomembnejši del te raziskovalne naloge pojasnj razvoj obeh verzij aplikacije, torej za operacijski sistem iOS in Android posebej. Ker imam z razvijanjem v teh dveh okoljih kar nekaj izkušenj, mislim, da mi je to dobro uspelo. Za vse dileme sem uporabil tehnične reference na spletnih straneh http://developer.apple.com in http://developer.android.com, kot še en pomožen izhod v sili pa lahko uprabljam teme na http://stackoverflow.com, kjer se najde praktično odgovor na vsako vprašanje, povezano s programiranjem. Prav tako sem poskušal doseči dogovor s šolo o dejanski izvedbi projekta in povezavi na strežnik, da lahko sistem začnemo uporabljati. Na koncu sem naredil še primerjavo z že obstoječim sistemom.

2 O razvojnih okoljih za operacijska sistema iOS in Android

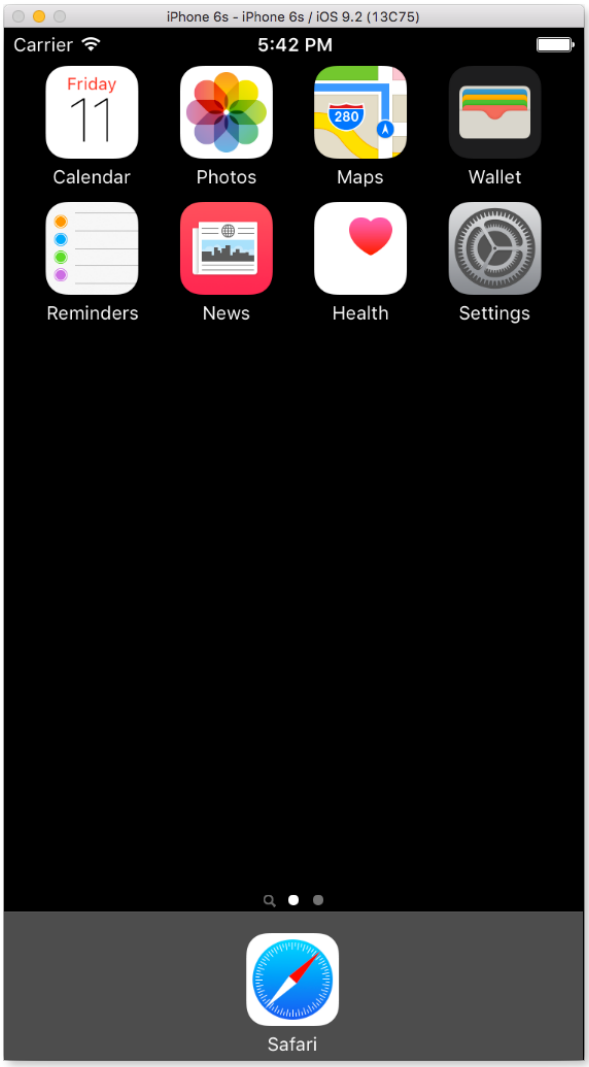
Razvijalski okolji za operacijska sistema iOS in Android podjetij, ki ju razvijata sta Xcode in Android Studio. V Applovem Xcode se priporoča uporaba novega programskega jezika Swift, v googlovem Android Studiu pa uporaba jezika Jave. Možna bi bila tudi uporaba kakšnega tretjega razvojnega okolja, kot sta na primer Xamarin Studio ali Microsoft Visual Studio, ki mi obe verziji aplikacije omogočata razvijati kot en projekt, a se za to nisem odločil, saj bi to še vseeno pomenilo razvoj dveh uporabniških vmesnikov in seveda dveh pripadajočih programskih kod z različnimi tehnologijami in pristopi v ozadju. Prav tako je uporabniški vmesnik bolje razviti v okolju, ki je prilagojeno na nek operacijski sistem, da se ta potem sklada z uporabniško izkušnjo, ki so jo uporabniki tega operacijskega sistema vajeni. Obe razvojni okolji imata tudi zelo dobre emulatorje mobilnih naprav, ki se lahko uporabljajo za testiranje in razhroščevanje končnih produktov.



Slika 1: Posnetek zaslona okna razvijalskega okolja Xcode 7.2.1 na Mac OS X 10.11.3 z odprtim projektom, ki vsebuje izvorno kodo za iOS verzijo aplikacije.



Slika 2: Posnetek zaslona okna razvijalskega okolja Android Studio 1.5.1 na Mac OS X 10.11.3 z odprtim projektom, ki vsebuje izvorno kodo za Android verzijo aplikacije.

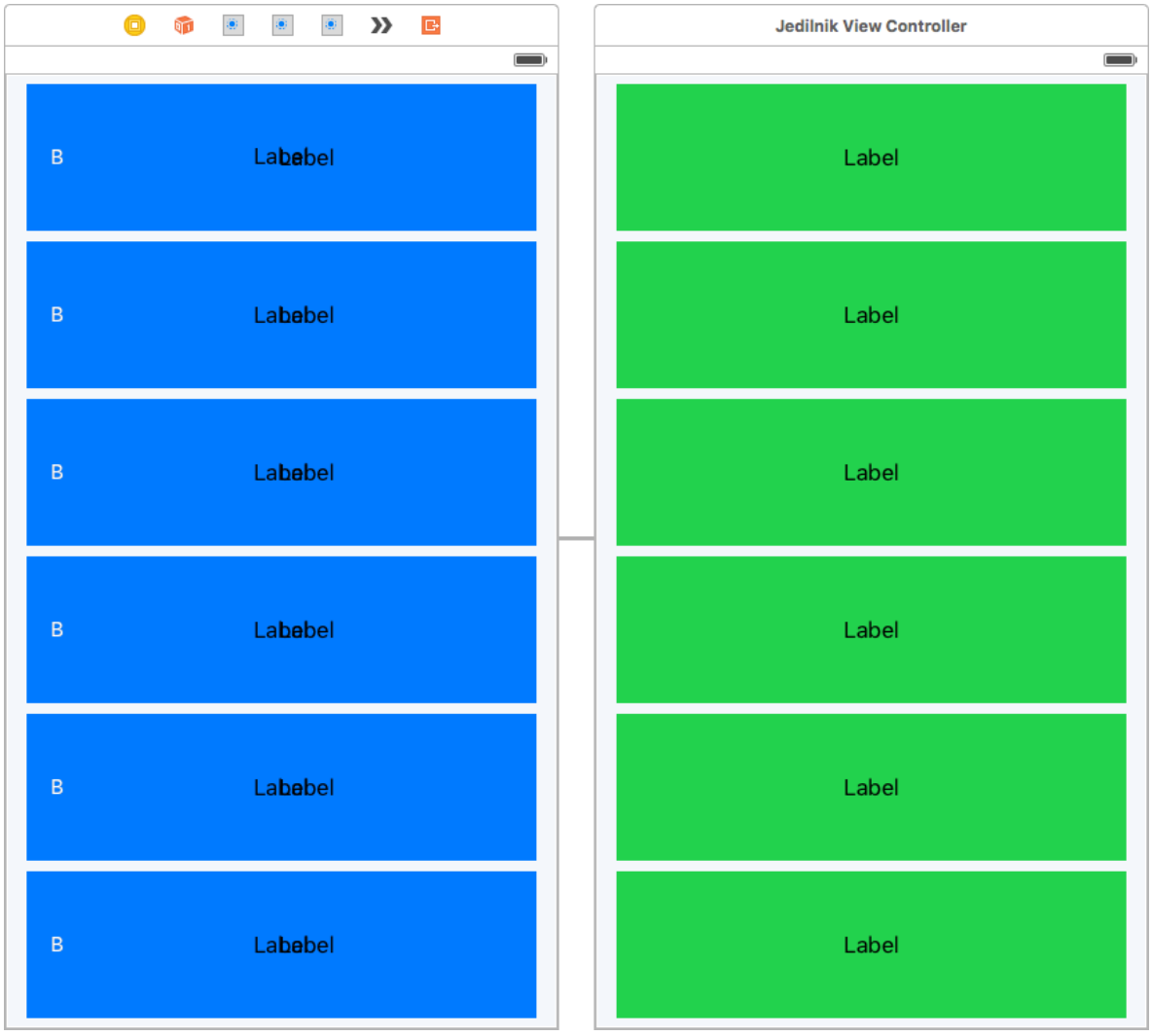


Slika 3, slika 4: Posnetka zaslona emulatorjev mobilnih naprav z operacijskima sistemoma iOS 9.2 (levo) in Android 5.0.2 (desno).

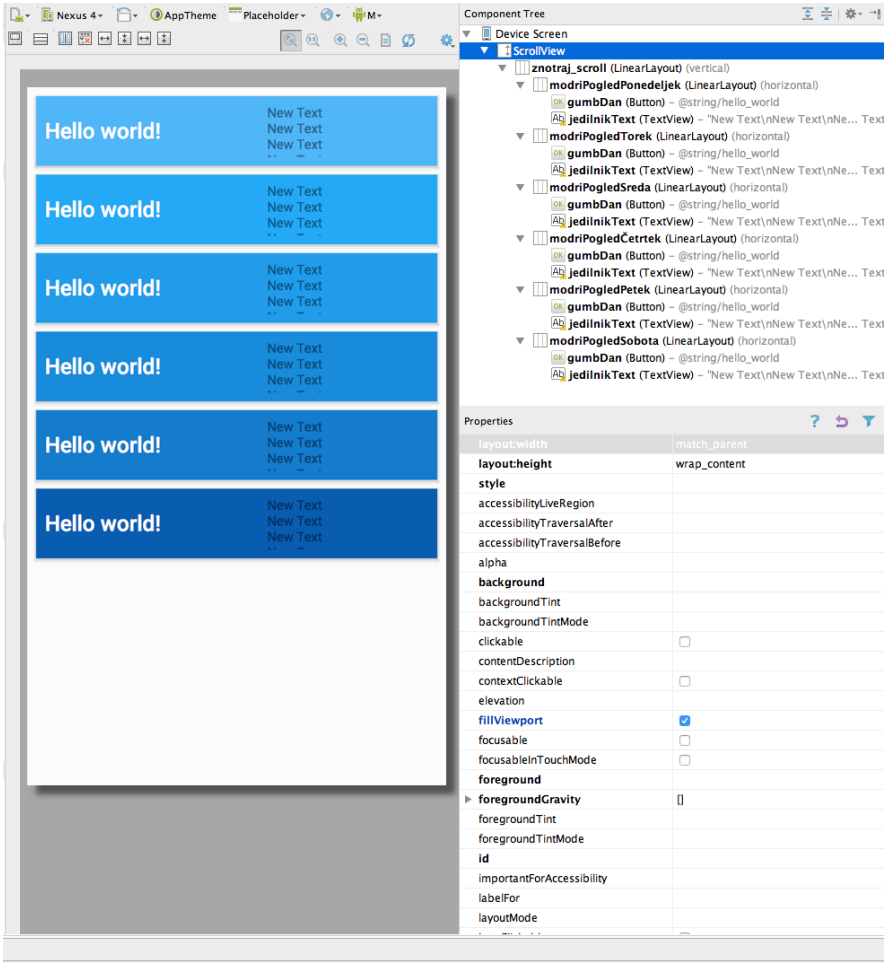
3 Načrtovanje in izvedba

3.1 Uporabniški vmesnik

Začel sem z osnovnim konceptom uporabniškega vmesnika, ki ga je potrebno prilagoditi različnim, a majhnim mobilnim zaslonom. Ne glede na majhnost pa mora biti funkcionalnost na visokem nivoju. Odločil sem se za tedensko razporeditev naročenih obrokov, s pripadajočim jedilnikom, kar uporabnikom omogoča odlično predstavo o vsem. Ime obroka je gumb, ki omogoča spremembo le-tega za pripadajoč dan.
Razvijalski okolji za operacijska sistema iOS in Android imata odlične možnosti oblikovanja grafičnega vmesnika s pomočjo sicer nekoliko različnih tehnologij. V iOSu se uporablja tehnologija Auto Layout, pri Androidu pa se preprosto uporabi kombinacija različnih elementov, kot sta Linear Layout in Relative Layout (v aplikaciji sem uporabil samo Linear Layout), znotraj katerih se potem postavi vidne elemente.
Celoten grafični vmesnik za iOS je opisan v datoteki s končnico .storyboard, pri Androidu pa se vsaka posamezena stran definira v svoji .xml datoteki. Oba tipa datotek sicer uporabljata jezik XML, razvojni okolji pa nudita tudi možnost grafičnega oblikovanja uporabniškega vmesnika. S pomočjo grafičnega oblikovanja sem zasnoval osnoven izgled vmesnika, torej postavitev posameznih elementov, nato pa sem jih s programsko kodo dokončno oblikoval in jim dodal funkcionalnost.

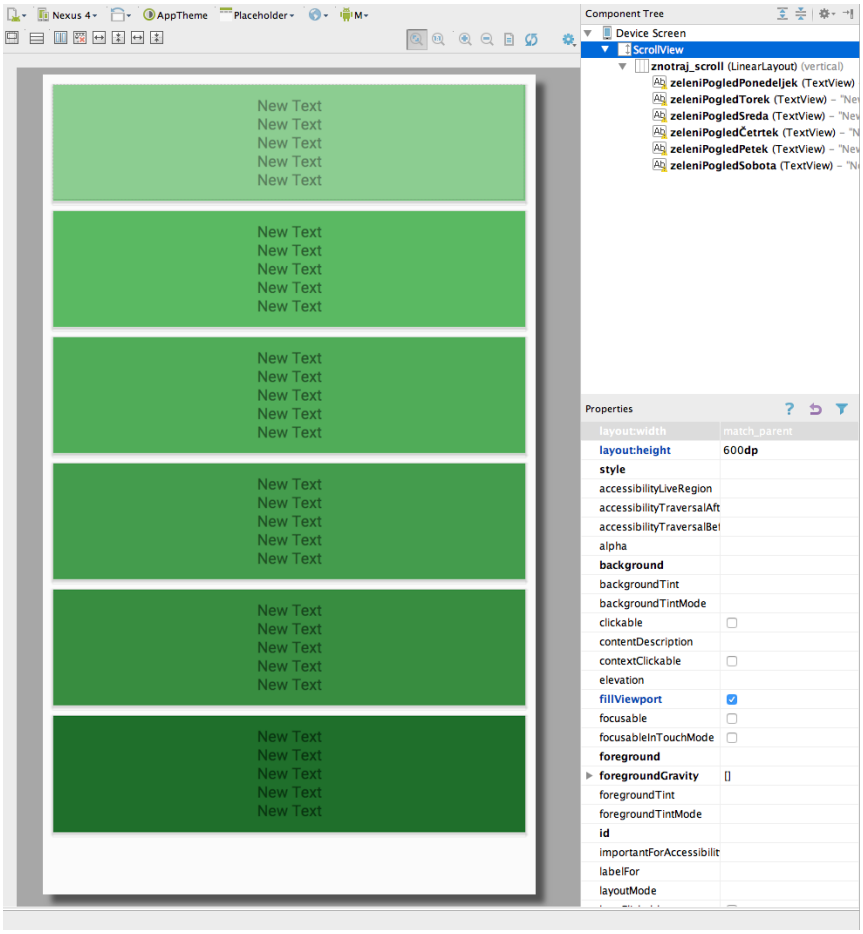


Slika 5: Najpomembnejša elementa uporabniškega vmesnika (iOS), pregled naročenih obrokov in pregled jedilnika (ena stran), definirana v datoteki Main.storyboard, odprta v Xcode.



Slika 6: Datoteka fragment_naroceno_view.xml v kateri je definirana ena stran pregleda naročenih obrokov na operacijskem sistemu Android.

Slika 7: Datoteka fragment_jedilnik_view.xml v kateri je definirana ena stran pregleda jedilnika na operacijskem sistemu Android.



3.1.1 Meniji

Poseben del uporabniškega vmesnika na mobilnih napravah so meniji. Na iOSu so sestavljeni iz statusne vrstice, ki prikazuje čas in stanje naprave, ki naj bi bila po priporočilih pri vseh aplikacijah enaka (uporaba sistemske) in vrstice z navigacijo, ki služi navigaciji nazaj, kot prostor za gumbe z dodatnimi možnostmi in kot prostor za naslov trenutnega dejanja. Te menije upravlja controller UINavigationViewController, ki skrbi tudi za postavitev in prehode med pogledi, ki jih upravljajo ostali controllerji.
Meniji na Androidu so sestavljeni tudi iz statusne vrstice in pa vrstice z navigacijo, naslovom prikazane aktivnosti in gumba Menu z dodatnimi možnostmi. Naprave, ki imajo Operacijski sistem Android verzije nižje kot 5.0, namesto tega gumba za dostop do dodatnih možnosti uporabljajo fizičen gumb, ki se nahaja ponavadi levo ali desno od gumba domov. Meniji so definirani v obliki xml, za eno aktivnost pa je možno menije tudi dinamično spreminjati.

3.2 Razvoj komponent za iOS

Razvoj aplikacij za operacijski sistem iOS poteka v objektno orientiranem programskem okolju. To je zelo dobra možnost za preprosto dodajanje kompleksnosti, ki je pri modernih operacijskih sistemih prisotna v veliki meri. Objekti se v programskih jezikih imenujejo razredi. Najpogostejši pristop k razvoju aplikacije v okolju Xcode je najprej organizacija uporabniškega vmesnika v datoteki s končnico .storyboard in nato definicija posameznih elementov le-tega, kot objektov v programski kodi. Ta koda opisuje obnašanje aplikacije. V nadaljevanju sem opisal vse programske razrede v aplikaciji, za imenom razreda in dvopičjem sledi še ime nadrazreda opisanega razreda.

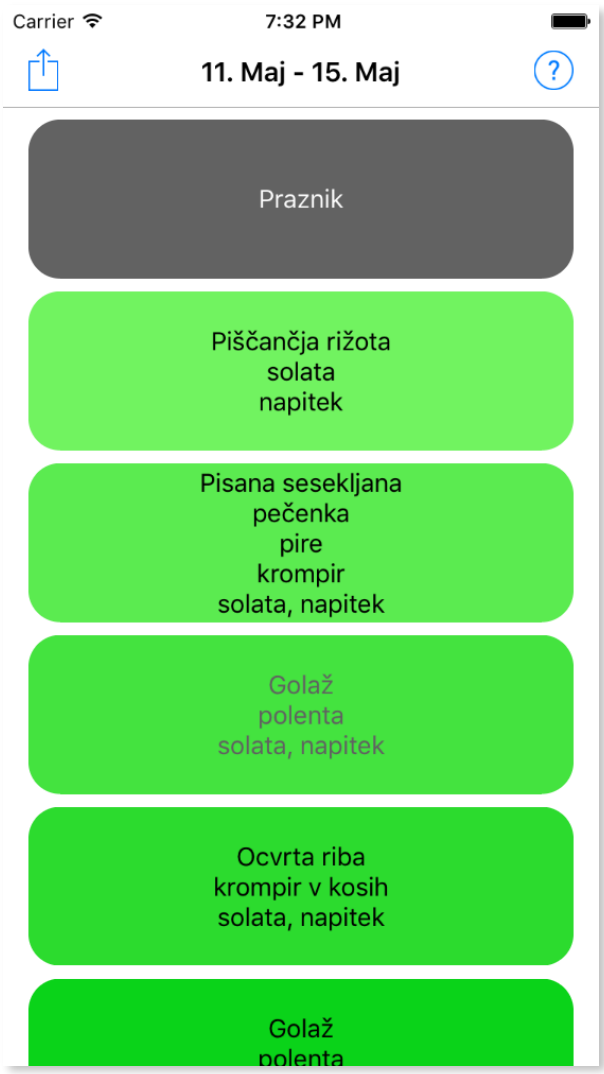
3.2.1 GlavniPogled : UIViewController

Naloga tega controllerja je urediti pogled na katerem je tedenski pregled naročenih obrokov. Prav tako je tudi odgovoren za odziv na interakcije uporabnika na tem pogledu in sicer dolg pritisk na polje za posamezen dan in izvedba odjave oziroma prijave ter pritisk na gumb, ki uporabnika pošlje na pogled za spremembo menija za določen dan, ki ga ureja razred MenjavaMenija.

Slika 8, slika 9: Posnetka zaslona ko je prikazan glavni pogled. Kot lahko vidimo na desni sliki, je aplikacija pripravljena tudi na situacije, ko malice ni mogoče naročiti in na sobote, na katere poteka pouk.

3.2.2 JedilnikViewController : UIViewController

Ta controller ureja pogled, ki prikazuje tedenski jedilnik za posamezen obrok. V meniju, dostopnem z gumbom zgoraj levo, je možno spremeniti prikazan obrok. Iz barve pisave je razvidno, ali smo na izbrani obrok za določen prijavljeni. V primeru, da smo odjavljeni, je pisava svetlejša. Če se uporabnik želi naročiti na izbrani obrok na dan, ko naj ni naročen, lahko to stori z interakcijo dolg pritisk. Pregled jedilnika je za razliko od pogleda naročenega dostopen tudi tistim uporabnikom, ki v aplikacijo niso prijavljeni.



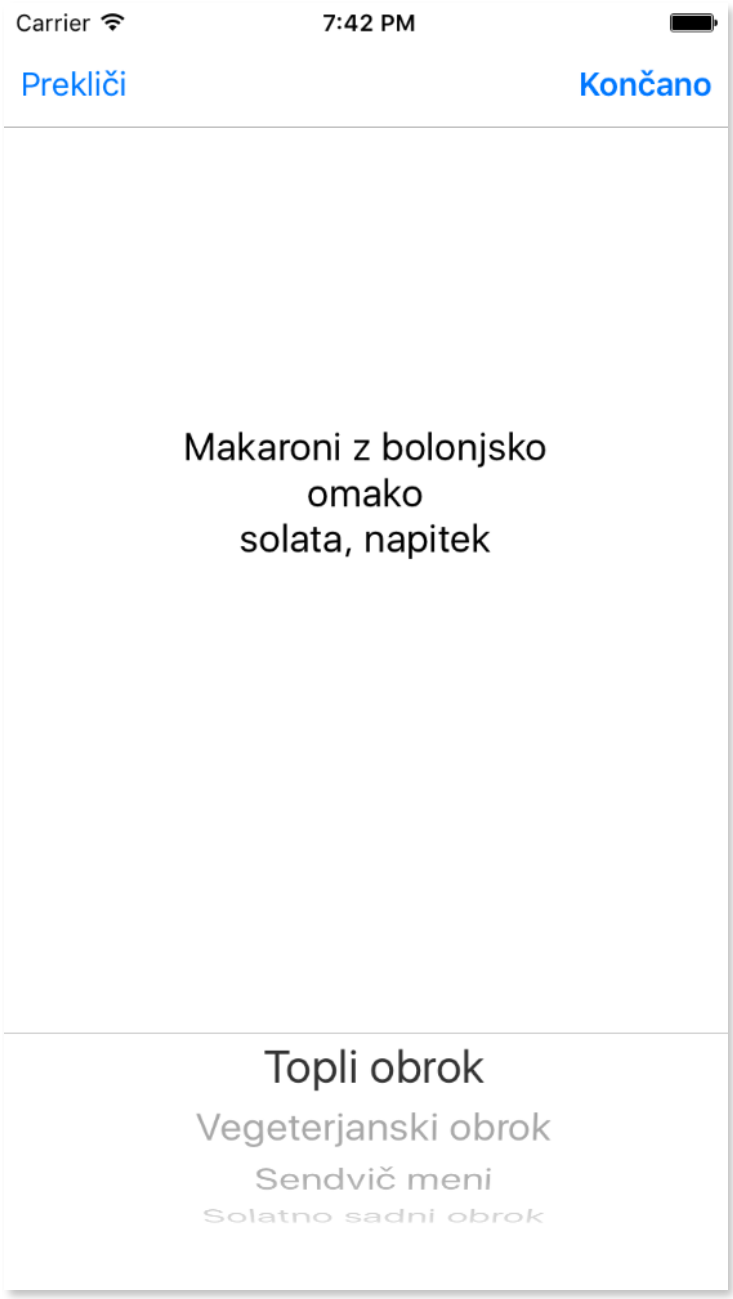
Slika 10, slika 11: Posnetek zaslona, ko je prikazan pogled jedilnika. Na desni sliki v četrem polju (četrtek) vidimo situacijo, ko je uporabnik naročen na drug obrok ali odjavljen od malice. Ta funkcija je na voljo le za prijavljene uporabnike.

3.2.3 RootViewController : UIViewController

Ta controller na zaslon postavlja poglede, ki jih urejata GlavniPogled in JedilnikViewController. To stori tako, da jih postavlja kot strani med katerimi se je preprosto premikati z drsanjem prsta po zaslonu. Tukaj v ozadju je sistemski razred, UIPageViewController, kateremu se pošljejo pripravljene strani, ta pa jih postavi na zaslon.

3.2.4 MenjavaMenija : UIViewController

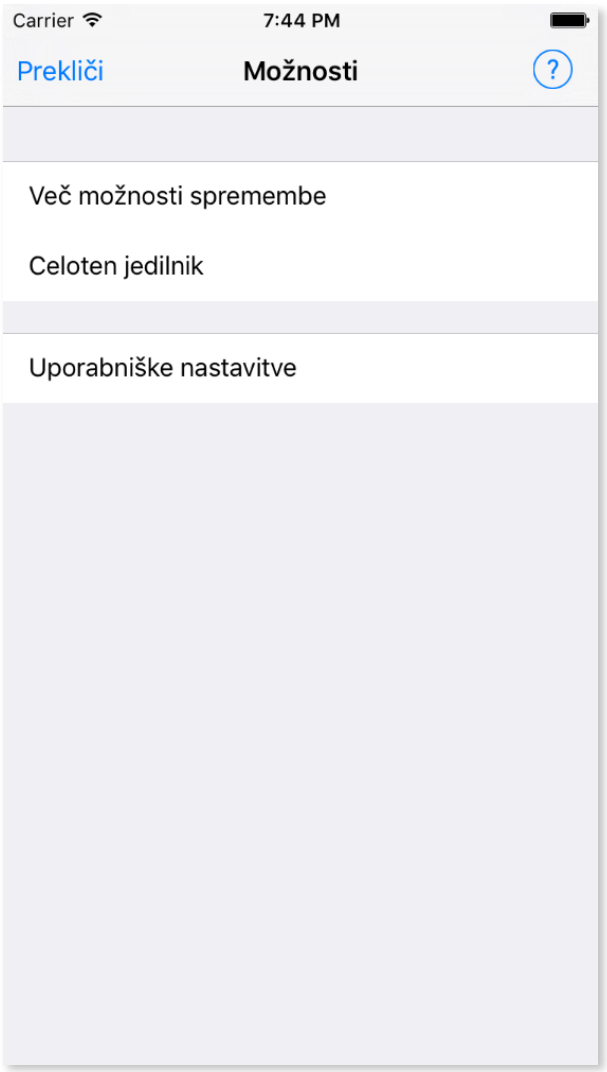
Pogled, ki ga nadzoruje ta controller, se odpre s pritiskom imena naročenega obroka v controllerju GlavniPogled. Ta pošlje podatek o tem, kateri dan je bil izbran in na osnovi tega se prikaže še jedilnik za ustrezen dan. Takoj, ko uporabnik spremeni svojo izbiro obroka s pomočjo elementa UIPickerView, se tudi jedilnik temu primerno spremeni. Sprememba se lahko prekliče z gumbom Prekliči ali pa potrdi z gumbom Končano.



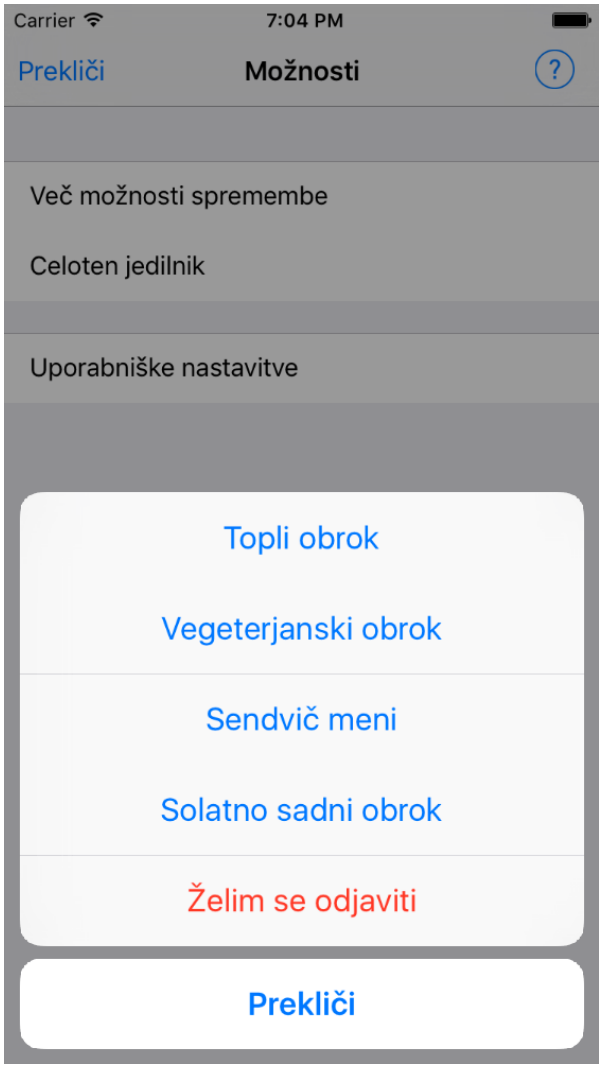
Slika 12: MenjavaMenija

3.2.5 MožnostiPriMenjavi : UITableViewController

V pogledu tega controllerja je tabela (UITableView), ki je pogosto uporabljen grafični element na operacijskem sistemu iOS. Do tega pogleda pridemo iz elementa GlavniPogled, s pritiskom na gumb action, v zgornjem levem robu pri elementu GlavniPogled (znotraj navigacijske vrstice menija) in vsebuje vse dodatne relevantne možnosti v povezavi s tem pogledom. Prva je možnost spremembe prijavljenega obroka za cel teden hkrati, druga prikaz jedilnika, tretja pa prikaže uporabniške nastavitve. Na desni sliki vidimo nad tem controllerjem prikazan element UIAlertController na iOS 8.0 in višje oziroma UIActionSheet na starejših verzijah. Ta elementa sta ustvarjena sproti in se prikažeta po izbiri ”Več možnosti spremembe.“ Služita temu, da uporabnik izbere, na kateri obrok se želi v naslednjem koraku prijaviti.



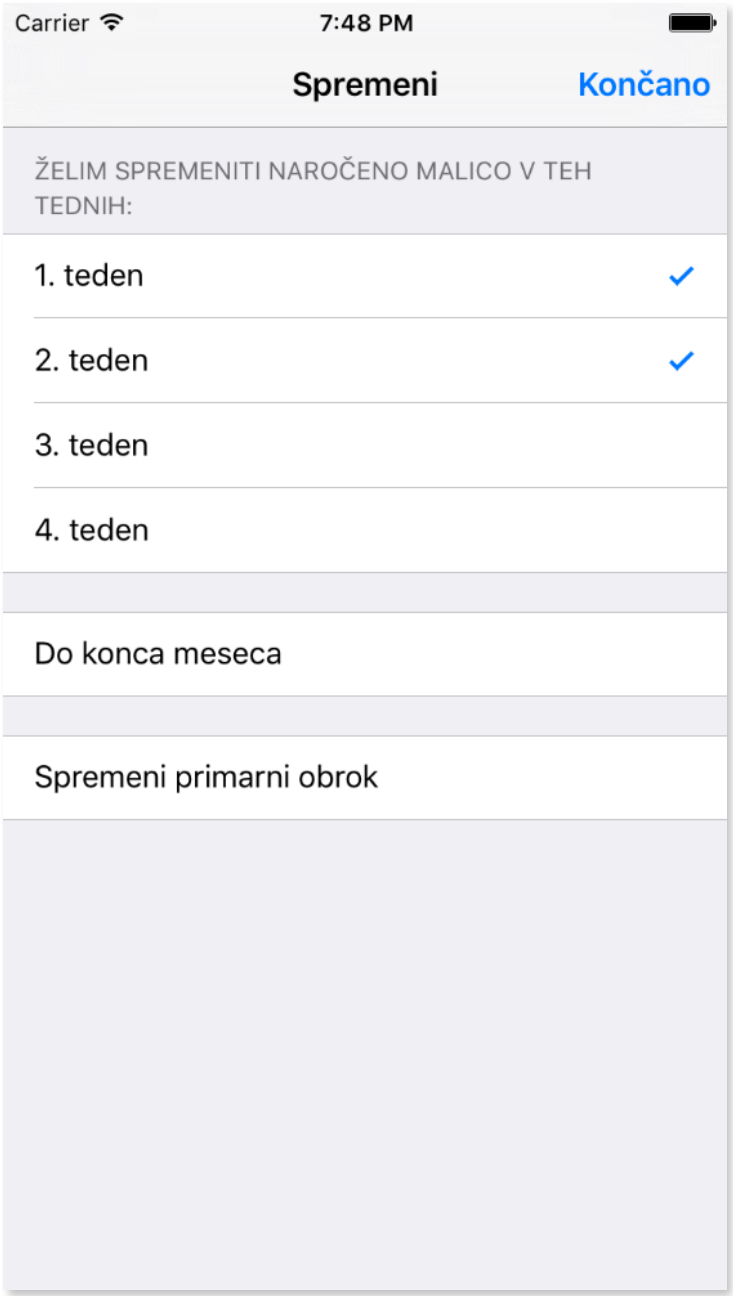
Slika 13: MožnostiPriMenjavi



Slika 14: UIAlertController preko MožnostiPriMenjavi

3.2.6 OdjavaDaljšiČas : UITableViewController

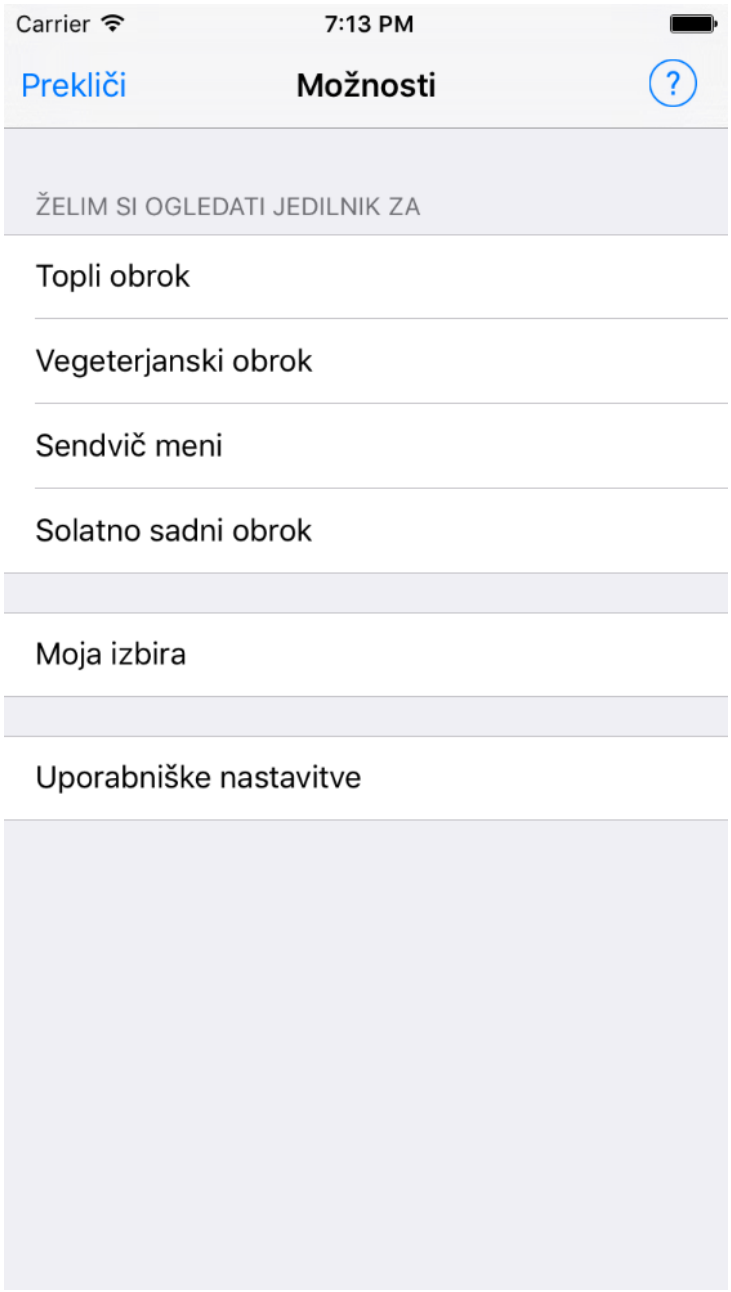
V tem meniju se po izbiri obroka izvede menjava nanj. Ponujene so možnosti spremembe za vsak teden posebej in pa do konca tekočega meseca. Lahko se spremeni tudi primarni obrok, to je tisti na katerega je uporabnik naročen samodejno ob začetku vsakega novega meseca, kakor je to že do sedaj bilo izvedeno s strani šolskega strežnika. Sprememba tega pa je bila skoraj nemogoča in ta funkcija bo res olajšala uporabo celotnega sistema. Po zaključeni izbiri pritisnemo gumb ”Končano“ in aplikacija uredi vse za nas.



Slika 15: OdjavaDaljšiČas

3.2.7 MožnostiPriJedilniku : UITableViewController

Tako kot v prej opisanem meniju MožnostiPriMenjavi, so tukaj le še dodatne relevantne možnosti glede na pogled prikazan v RootViewController. Tako kot tisti, je tudi ta pogled grajen na osnovi tabele (UITableView). Vsebuje možnosti za spremembo prikazanega obroka na jedilniku, preklop na pogled naročenega in še uporabniške nastavitve.



Slika 16: MožnostiPriJedilniku

3.2.8 Podatki

Podatki je razred, ki vse elemente aplikacije oskrbuje z informacijami o uporabnikovih naročenih obrokih, z jedilniki za vsak obrok in morebitnimi obvestili. Ta razred bo ob dokončani implementaciji s strežnikom tudi skrbel za pridobivanje teh podatkov od tam. Tak koncept za shranjevanje in pridobivanje podatkov iz strežnika je zelo primeren in tudi priporočen s strani Appla in Googla. Omogoča zelo hitro in enostavno, hkrati pa zanesljivo implementacijo s strežniškim delom. Pri opisu razredov verzije aplikacije za Android ni opisan, saj ima popolnoma identično funkcionalnost.

3.3 Razvoj komponent za Android

Tako kot pri iOS, je tudi pri Androidu razvoj aplikacij objektno orientiran. Tukaj se uporablja programski jezik Java, za razvoj aplikacije na tej platformi pa sem uporabil okolje Android studio. Osnova vsaki aplikaciji je datoteka AndroidManifest.xml, v kateri so opisane osnovne tehnične lastnosti aplikacije, glavna aktivnost, strojne zahteve in podobno. Tudi tukaj so opisani vsi posamezni razredi na enak način kot pri iOSu, torej z imenom nadrazreda.

3.3.1 PageView : ActionBarActivity

Ta aktivnost je osnova za tedenski pregled naročenih obrokov. Znotraj nje postavlja razred SectionsPagerAdapter posamezne strani razredov NaročenoFragment in JedilnikFragment. Na zgornjem delu zaslona, je element ActionBar, ki prikazuje ime aplikacije, čas trajanja prikazanega tedna, ki bo vgrajen po uspešni implementaciji strežnika, zaradi časovne usklajenosti med aplikacijo in strežnikom in pa meni z dodatnimi možnostmi, ki je dostopen z gumbom na desnem delu elementa. V meniju so tako, kot pri iOSu relevantne možnosti za vsebino elementa PageView.

3.3.2 SectionsPagerAdapter : FragmentPagerAdapter

Razred uporabljen znotraj PageView. Njegova naloga je iz datoteke fragment_jedilnik_view.xml oziroma fragment_naroceno_view.xml ustvariti fragment, ki ga nato postavi v PageView kot eno od strani za prikaz izbranih obrokov oziroma jedilnika. Vsak fragment posebej predstavlja eno stran v PageView oziroma en teden.

3.3.3 NaročenoFragment : Fragment

To je podrazred razreda Fragment, ki se uporablja pri postavitvi več strani. Ta fragment je stran, na kateri uporabnik vidi svoje naročene obroke in jih z dotikom gumba z imenom obroka spremeni. Z gesto dolg pritisk, se je možno tudi prijavljati in odjavljati. V primeru, da se na malico v določenem dnevu ni mogoče prijaviti, na primer zaradi praznika ali športnega dne, je to jasno označeno, poleg tega pa je še napisano pojasnilo, kot je to na desni sliki za ponedeljek. Če tega obvestila ni podanega, se samodejno izpiše sporočilo ”Prijava ni mogoča.“ Fragment je prilagojen tudi na morebitne daljše delovne tedne, ki vključujejo sobote, kot vidimo pri desni sliki spodaj. Zaradi majhnega zaslona, je ta del dostopen tako, da s prstom podrsamo po zaslonu navzgor, kot je večina uporabnikov pri uporabi pametnih telefonov že navajena. Pogled za naslednji teden lahko dosežemo z isto interakcijo v levo.

Slika 17, slika 18: NaročenoFragment

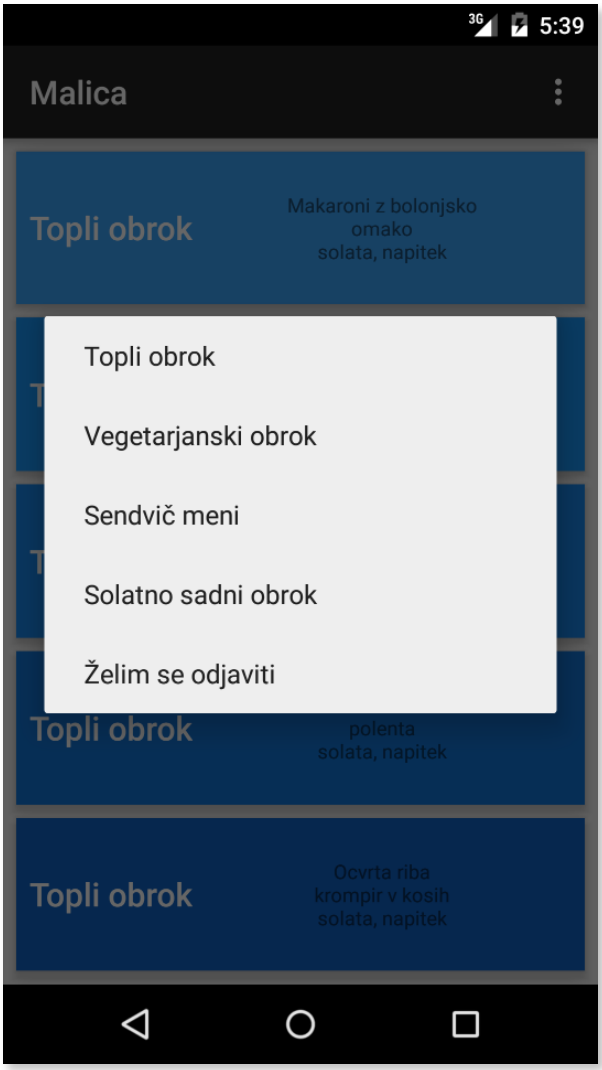
3.3.4 JedilnikFragment : Fragment

Ta podrazred Fragmenta omogoča ogled jedilnika za posamezen obrok in je dostopen tudi za neprijavljene uporabnike. Če je uporabnik prijavljen, je glede na barvo besedila razvidno, ali je na nek obrok za nek dan prijavljen in sicer črna barva pomeni prijavljen, siva pa neprijavljen, kar vidimo na desni sliki v pogledu za četrtek. Siva barva pomeni, da uporabnik ni prijavljen na ta obrok, torej da je lahko odjavljen od malice ali pa prijavljen na katerega drugega. Za dan, za katerega ta obrok ni izbran, je mogoča enostavna prijava z gesto dolg pritisk.

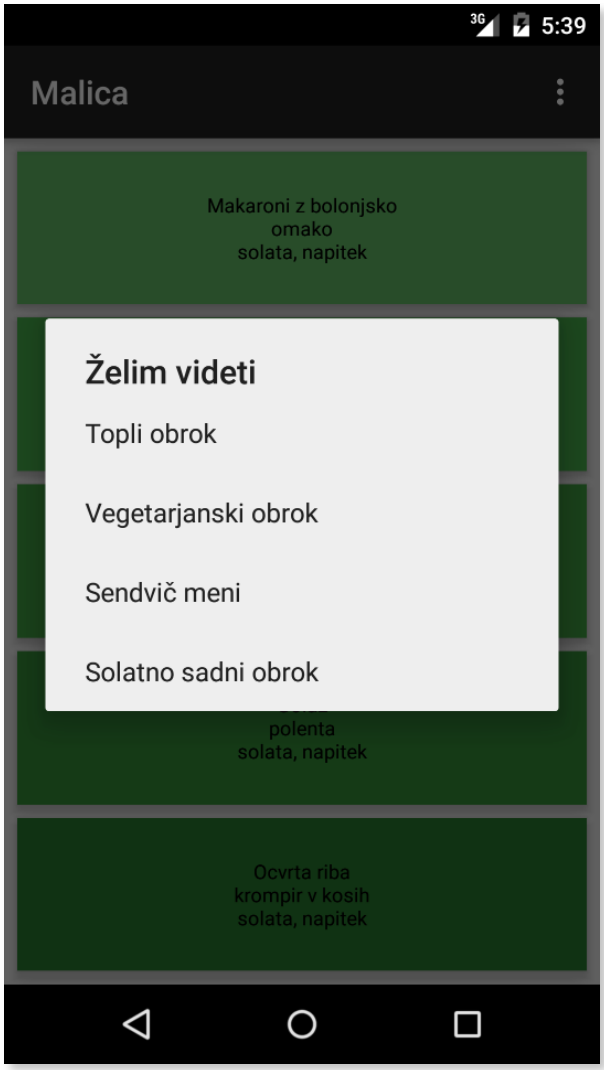
Slika 19, slika 20: JedilnikFragment

3.3.5 IzberiObrokZaSpremenitiFragment : android.support.v4.app.DialogFragment in IzberiObrokZaPrikazatFragment : android.support.v4.app.DialogFragment

Fragment IzberiObrokZaSpremenitiFragment je pomožen element pri celotni funkciji spreminjanja prijave na malico za daljši čas (po en teden skupaj). Dostopen je v meniju v elementu ActionBar (zgoraj desno), ki ima identične možnosti kot MožnostiPriMenjavi pri iOSu.
Fragment IzberiObrokZaPrikazatFragment se pojavi, ko si uporabnik želi ogledati jedilnik za drug obrok v pogledu jedilnika. Dostopen je prav tako v meniju elementa ActionBar (zgoraj desno), ki pa ima identične možnosti elementu MožnostiPriJedilniku na iOSu.
Tu gre za elementa z istim nadrazredom in podobnim namenom. V glavnem se vizualno razlikujeta le po postavitvi. IzberiObrokZaSpremenitiFragment po izbiri obroka nadomesti element VecMoznosti, ki je po namembnosti identičen controllerju OdjavaDaljšiČas na iOSu. IzberiObrokZaPrikazatiFragment pa le sporoči elementu SectionsPagerAdapter, da uporabnik želi menjavo prikazanega obroka.



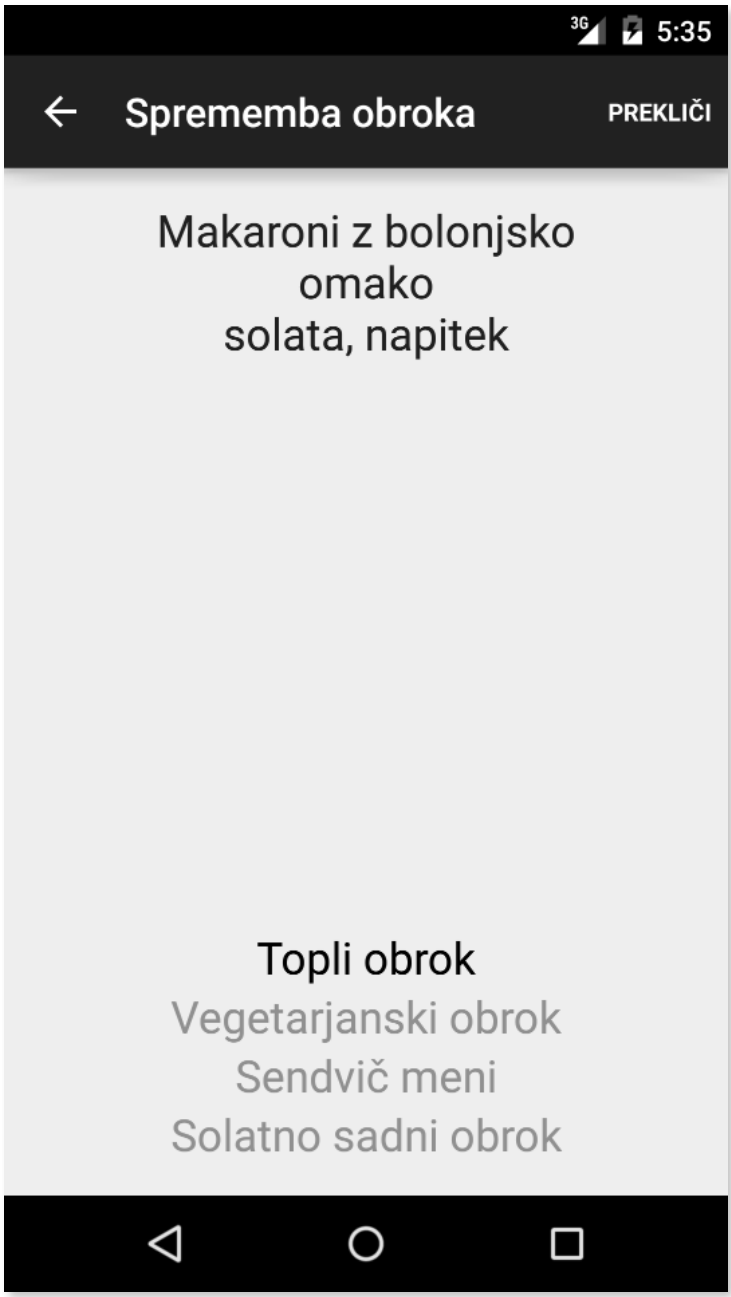
Slika 21: IzberiObrokZaSpremenitiFragment



Slika 22: IzberiObrokZaPrikazatFragment

3.3.6 SpremembaObroka : ActionBarActivity

Do tega activity elementa dostopamo s pritiskom na gumb z imenom obroka v fragmentu NaročenoFragment. Deluje podobno kot ekvivalenten UIViewController v iOSu, z eno ključno razliko. Tam je uporabljen element UIPickerView, ki je standarden element za izbiro več možnosti v tistem operacijskem sistemu. Ta element ima izgled kolesa pri igralnem avtomatu in je zelo dobro vizualno integriran v operacijski sistem. Pri Androidu pa je praksa uporabiti navadne menije, torej podoben element sploh ni sistemsko na voljo. Seveda bi se ga dalo izdelati, vendar bi to terjalo veliko časa, povrhu tega pa se ne bi vizualno skladal z uporabniško izkušnjo operacijskega sistema, ki so je uporabniki vajeni. Ostale lastnosti so zelo podobne verziji za iOS. Ob izbiri novega obroka, se jedilnik spremeni, izbiro pa potrdimo s sistemskim in tistim v meniju gumbom nazaj. Lahko jo tudi prekličemo z gumbom prekliči.



Slika 23: SpremembaObroka

3.4 Ostale komponente

Uporabniški del aplikacije je zaključen, dodelan in praktično pripravljen za uporabo. Ampak tukaj pride na vrsto še strežniški del aplikacije. Tu se stanje nekoliko zaplete, saj nimam sam nadzora nad celotnim potekom. Strežniški del je sestavljen iz že obstoječega strežnika, ki je že v uporabi za evidenco prijav in drugega dela, ki bo namenjen shranjevanju podatkov o jedilniku. Izvesti je torej potrebno še implementacijo prvega strežnika in mu dodati možnost shranjevanja jedilnikov. Povezal sem se tudi s skrbnikom računalniške opreme na šoli, ki me je povezal s programerji programske opreme našega sistema, vendar na končen odgovor in možnost uporabe še čakam. Vmes sem izdelal program v programskem jeziku Java z uporabo frameworka apache poi, ki datoteke .docx, v katerih je napisan jedilnik, pretvori v obliko podatkov, ki jih lahko pošlje strežniku in jih tam shrani. Žal tudi tu manjka samo strežniški del. Dodati moram še, da se lahko izvedba nalaganja podatkov o jedilniku na strežnik spremeni v skladu z dogovorom s podjetjem, ki dostavlja hrano na šolo. Pri implementaciji strežnika in uporabniškega dela aplikacije ne pričakujem večjih težav, saj bo za vse podatke poskrbel razred Podatki. Edina stvar poleg tega, ki jo bo tedaj potrebno še urediti, je časovna usklajenost aplikacije in strežnika, ki je pri določenih aspektih dokaj pomembna. Tu je nujno predvsem zavedanje do kdaj je možna zamenjava in odjava obrokov. Vseeno tudi tukaj ne pričakujem težav, saj mora že obstoječ sistem imeti mehanizem za časovno usklajeno delovanje.

4 Testiranje in analiza rezultatov

Testiranje delovanja produktov razvoja aplikacije navadno vedno teče vzporedno s samim razvojem, tako se namreč najbolj zanesljivo odpravi vse možne napake in aplikacijo optimizira in uporabi. Za ta izjemno pomemben del potrebujemo dovolj časa. Poleg testiranja samega delovanja, bo potrebna primerjava nove rešitve v obliki aplikacije s staro rešitvijo v obliki spletne strani. Aplikacija je na voljo skoraj vsak trenutek in je veliko bolj odzivna kot spletna stran. Razlog za to je prilagojenost na mobilni operacijski sistem in izdelava z mislijo na mobilno okolje. S tem pridobimo tudi lastnosti, kot so drsanje med stranmi in uporabo drugih podobnih interakcij, ki resnično pozitivno vplivajo na uporabniško izkušnjo.

NBa osnovi opravljenega lahko potrdim prvo hipotezo, saj sem ugotovil, da je veliko bolj priročno uporabiti mobilni telefon za takšno dejanje, kot je menjava naročenih obrokov. S tem, ko imamo vključene mobitele ves čas pri sebi, se čas dostopa do vmesnika za to dejanje občutno zmanjša, kar pomeni večjo priročnost.

Tudi druga hipoteza drži, saj je aplikacija resnično bolj odzivna kot spletna stran, poleg tega pa je tudi prilagojena na mobilni zaslon, kar pomeni kar največjo velikost gumbov in pisave brez nepotrebnega povečevanja in iskanja pravega elementa. Še ena izjemno uporabna lastnost je jedilnik, ki je prikazan ves čas, ko si uporabnik izbira obrok.

Na osnovi pravilnosti prvih dveh hipotez lahko sklepamo tudi na pravilnost tretje hipoteze, ki pravi, da je nova rešitev uporabniku bolj prijazna in prilagojena kot stara.

5 Zaključek

V tej raziskovalni nalogi sem ugotovil, da je eden najpriročnejših načinov za opravilo nekega preprostega dejanja, ki vključuje računalniški vmesnik, mobilna aplikacija. Z razvojem opisane aplikacije, sem priročnost in uporabnost tudi dokazal. Uporabljal sem razvojni okolji razvijalcev operacijskih sistemov, za katera sem jo prilagodil. Obe okolji imata svoje specifične lastnosti, a ko sem se obeh navadil, lahko rečem, da sta zelo dobro pripravljeni, kot se seveda za razvojni okolji tako velikih podjetij spodobi. Na začetku razvoja aplikacije, mi je bil sicer nekoliko bliže Xcode, ob uporabi Android Studia pa sem naletel na situacije, ki se v tem okolju rešujejo popolnoma drugače, vendar lahko sedaj rečem, da se mi oba pristopa zdita zelo logična. Edina razlika, ki je še vedno opazna, je odzivnost okolja, namreč v Xcode sem uporabljal nov, veliko bolj dinamičen jezik Swift, ki omogoča večje hitrosti kot Java, predvsem pri prevajanju programa.

Nad možnostmi razvoja v mobilnem okolju sem bil tako navdušen, da sem si zamislil še koncept aplikacije, ki bi poleg funkcionalnosti, predstavljene v tej raziskovalni nalogi, vsebovala še obvestila s strani šole, pregled urnikov in možnost prenosa učnega gradiva, kot to danes omogočajo tako imenovane spletne učilnice. Ta koncept sem ustvaril v razvojnem okolju Xcode, vsaka od dejavnosti, ki jo aplikacija nudi pa je ena stran elementa UITabBarController.
Mobilne aplikacije, zahvaljujoč modernim operacijskim sistemom, omogočajo zelo preprosto nadgradnjo in tako uporabniku kar se da hitro in preprosto ponudijo nove zmožnosti.

6 Viri

- https://sl.wikipedia.org/wiki/XML [Povzeto 10.3.2016]
- https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/DefiningClasses/DefiningClasses.html [Povzeto 10.3.2016]
- https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/ [Povzeto 10.3.2016]
- http://developer.android.com/guide/topics/ui/layout/linear.html [Povzeto 10.3.2016]
- https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853-CH7-SW1 [Povzeto 10.3.2016]