Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tine Mislej

Prototip uporabniškega vmesnika za scenarijsko usmerjene mobilne aplikacije

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00390/2013
Datum: 04.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **TINE MISLEJ**

Naslov: **PROTOTIP UPORABNIŠKEGA VMESNIKA ZA SCENARIJSKO USMERJENE MOBILNE APLIKACIJE**

PROTOTYPE OF USER INTERFACE FOR SCENARIO ORIENTED MOBILE APPLICATIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Za poslovne procese z večjim številom aktivnosti so mobilne aplikacije s klasičnim uporabniškim vmesnikom okorne za uporabo. To nakazuje na potrebo po raziskavi konceptov uporabniških vmesnikov za tovrstne poslovne proces. Koncepte za tovrstne uporabniške vmesnike bi lahko poimenovali tudi scenarijsko usmerjeni koncepti in posledično lahko govorimo o scenarijsko usmerjenih mobilnih aplikacijah.
Izvedite raziskavo konceptov scenarijsko usmerjenih mobilnih aplikacij. Opredelite 5 različnih tipov prototipov uporabniških vmesnikov, ki se med seboj razlikujejo po načinu uporabe. Za vsak tip uporabniškega vmesnika razvijte ustrezno mobilno aplikacijo za tablične računalnike Android platforme.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tine Mislej, z vpisno številko **63100094**, sem avtor diplomskega dela z naslovom:

Prototip uporabniškega vmesnika za scenarijsko usmerjene mobilne aplikacije

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 11. septembra 2013 Podpis avtorja:

Rad bi se zahvalil vsem, ki so mi skozi študijska leta stali ob strani, še posebej družini Mislej. Velika zahvala za mentorstvo gre tudi doc. dr. Roku Rupniku.

Kazalo

Seznam uporabljenih kratic in simbolov

API - Application Programming Interface; Programski vmesnik, ki določa, kako nekatere programske komponente delujejo med seboj.

IDE - Integrated Development Environment; Integrirano razvojno okolje.

JVM - Java Virtual Machine; Javanski navidezni stroj.

V/I - Vhod/Izhod.

DVM - Dalvik Virtual Machine; programska oprema, ki poganja Android aplikacije.

DSLV - DragSortListView.

XML - Extensible Markup Language; označevalni jezik.

MIT - Massachusetts Institute of Technology.

ADT - Android Development Tools; razvojna orodja za Android.

Povzetek

Diplomsko delo opisuje razvoj petih scenarijsko usmerjenih mobilnih uporabniških vmesnikov, ki so bili načrtovani za mobilno aplikacijo za podporo delu na kmetiji. V delu je podana tudi analiza izdelanih vmesnikov ter opis zadovoljevanja navodil in principov za načrtovanje uporabniških vmesnikov. Realizirani uporabniški vmesniki niso končane, delujoče aplikacije, ampak le delujoči prototipi. Pri razvoju smo upoštevali že uveljavljena vodila in principe (npr. Normanove in Mandellove) kot tudi novejša navodila za oblikovanje uporabniških vmesnikov.
V začetku dela so predstavljene tehnologije in orodja, ki smo jih uporabili pri razvoju, v nadaljevanju pa navodila za načrtovanje uporabniških vmesnikov. V končnem delu je predstavljen proces razvoja ter analiza realiziranih uporabniških vmesnikov.

Ključne besede: Android, uporabniški vmesnik, scenarijsko usmerjeni uporabniški vmesniki, prototip

Abstract

The thesis deals with the development process of five scenario oriented mobile user interfaces for operating system Android which have been designed for a mobile application to support farm work. In addition to the description of the development process, a description of satisfying design guidelines and principles is also presented along with the analysis of designed user interfaces. Implemented user interfaces are not fully functioned applications, but rather functioning prototypes. Many established guidelines and principles (e.g. Norman's and Mandell's) as well as some newer guidelines for designing user interfaces for mobile applications have been taken into account in the development process. In the first part of the thesis several technologies and development tools are presented. In the second one, process of five user interfaces, also analysed in the thesis, is described.

Keywords: Android, user interface, scenario oriented user interfaces, prototype

Poglavje 1

Uvod

V zadnjih letih smo priča zelo hitremu razvoju na področju mobilnih naprav. Vsakodnevno v naših žepih nosimo naprave, ki niso zmožne le klicanja in pisanja sporočil, ampak zmorejo veliko več. Z njimi se lahko povezujemo v svetovni splet, na daljavo upravljamo druge naprave, uporabljamo jih lahko kot navigacijske naprave, merimo temperaturo in globino, ipd. Menim, da se je koncept mobilnih naprave povsem spremenil. Te niso več le pripomoček za lažjo komunikacijo, ampak so postale nujna "razširitev" človeka v 21. stoletju. Danes je ključno, da so prave informacije na voljo v trenutku, ko jih potrebujemo.
Že v tem kratkem delu uvoda sem večkrat uporabil besedo "mobilno". V knjigi Designing Mobile Interfaces avtorja Steven Hoober in Eric Berkman ugotavljata, da evolucijo mobilne telefonije oz. mobilnih naprav lahko razdelimo v naslednja štiri obdobja [1]:

- Glas (ang. "Voice")
- Pozivanje in tekst (ang. "Paging and text")
- Razširjena omrežna povezljivost (ang. "Pervasive network connectivity")
- Splošne računske naprave (ang. "General computing devices")

Če današnji povprečen mobilni telefon smatramo kot napravo iz zadnjega, četrtega obdobja, ugotovimo, da ima naslednje lastnosti [1]:

Majhen. Je dovolj majhen, da ga lahko vedno nosimo s seboj, najraje v žepu.

Prenosljiv. Napaja ga baterija in je neodvisen od zunanjega sveta. Tako ni potrebno, da je priključen v elektriko oz. večkrat pregledan.

Povezan. Je brezžično povezan; kadar je mogoče, je povezan v različna omrežja (npr. podatkovna in glasovna).

Interaktiven. Je že sam po sebi interaktiven. Za razliko od npr. ure, ki nudi ožjo množico interakcij, pametni telefon ponuja širšo množico poljubnih interakcij, kot npr. vnos teksta ali iskanje po ključnih besedah.

Zaveda se konteksta. Sam oz. storitve, na katere je povezan, uporabljajo sposobnosti razumevanja omrežja, na katerega je naprava povezana, prav tako se zaveda svojih senzorjev, ki uporabniku pomagajo pri delu in samoiniciativno zbirajo informacije.

Mobilne naprave se precej razlikujejo od npr. namiznih računalnikov. So manjše, varčne, računska moč je navadno ustrezno manjša. Prav tako kot se te naprave razlikujejo med seboj, se med seboj zelo razlikujejo tudi uporabniški vmesniki aplikacij za posamezno vrsto naprav. Namizni računalniki imajo navadno večji zaslon, za interakcijo pa veliko uporabljamo miško. Uporabniški vmesniki so temu primerni. So večji, prostora za prikaz informacije je več, interakcija je natančnejša (npr. klik miške na določen del na zaslonu). Uporabniški vmesniki mobilnih aplikacij pa so tu zelo drugačni. Zasloni mobilnih naprav so znatno manjši, torej je prostora za prikaz informacij precej manj. Interakcija je manj natančna, mišk navadno ne uporabljamo, v večini uporabljamo kar svoje prste, zato smo pri klikanju precej manj natančni. Na tem področju torej razvijalca čaka veliko izzivov, s katerimi se v diplomskem delu srečujem tudi jaz. Izdelali smo pet različnih scenarijsko usmerjenih uporabniških vmesnikov, namenjenih aplikacijam za podporo pri delu na kmetiji.

Realizirani so za operacijski sistem Android. V delu bom predstavil proces razvoja in jih analiziral z različnih vidikov.

Poglavje 2

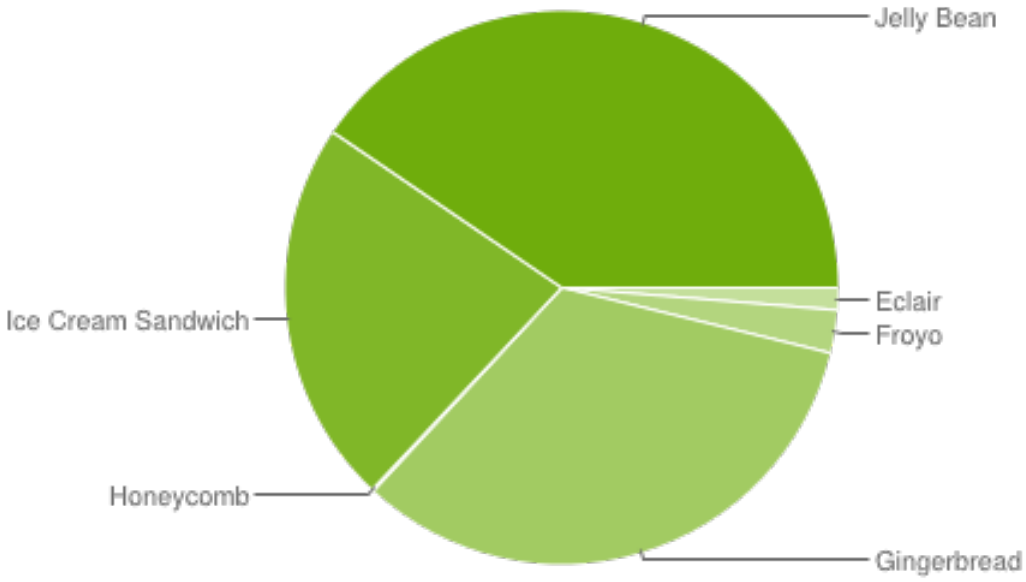
Uporabljena orodja in tehnologije

2.1 Android

Android je operacijski sistem, v glavnem namenjen mobilnim napravam, ki temelji na Linux jedru. Razvilo ga je podjetje Android Inc., ki ga je Google sprva finančno podprl, leta 2005 pa kupil [7]. Prva uradna verzija je izšla leta 2007, izdana je bila pod imenom Donut (glej tabelo 2.1). Po podatkih spletne strani time.com je danes najbolj razširjen operacijski sistem na pametnih telefonih prav Android, in sicer zavzema 51.2 % celotnega trga. Sledi mu iOS s 43.5 %, medtem ko 5.3 % zavzemajo ostali operacijski sistemi.

Verzija	Ime	API	Delež
1.6	Donut	4	0.1 %
2.1	Eclair	7	1.2 %
2.2	Froyo	8	2.5 %
2.3 - 2.3.2	Gingerbread	9	0.1 %
2.3.3 - 2.3.7	Gingerbread	10	33.0 %
3.2	Honeycomb	13	0.1 %
4.0.3 - 4.0.4	Ice Cream Sandwich	15	22.5 %
4.1.x	Jelly Bean	16	34.0 %
4.2.x	Jelly Bean	17	6.5 %

Tabela 2.1: Android verzije na dan 1. avgusta 2013.



Slika 2.1: Android verzije.

2.2 Arhitektura sistema Android

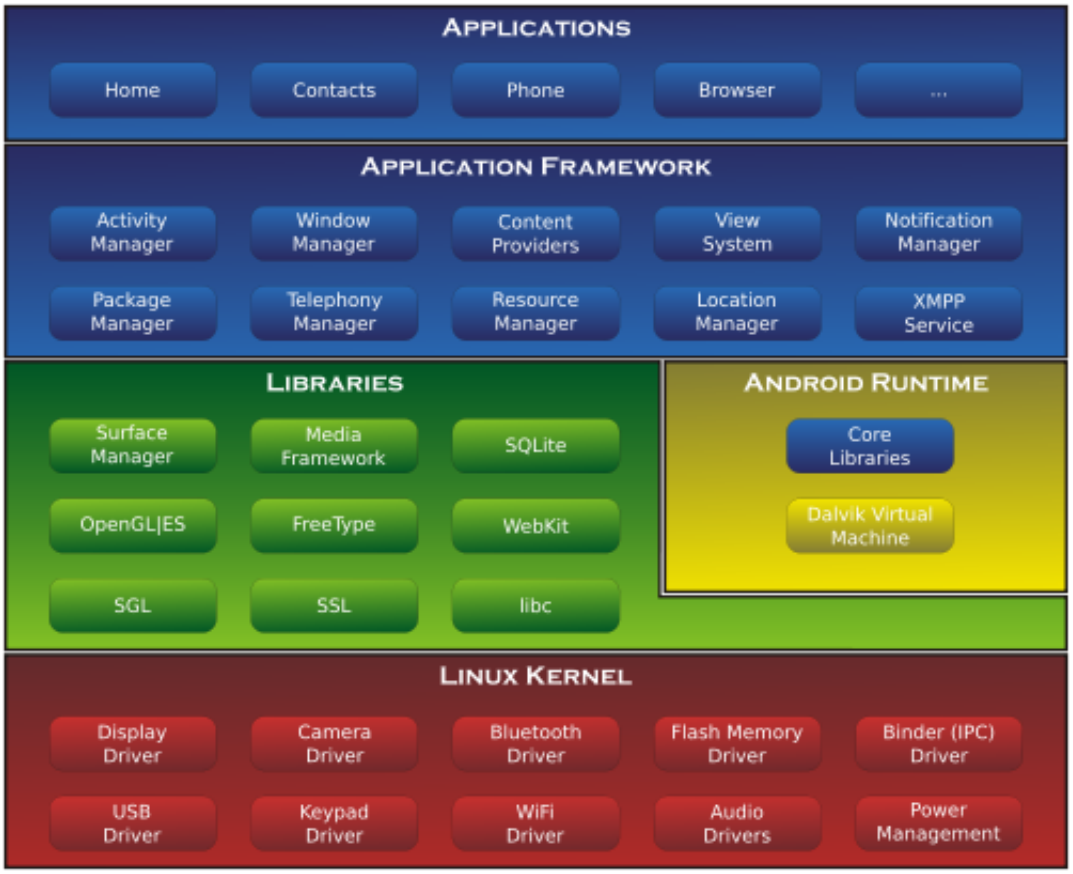
Arhitekturo operacijskega sistema Android lahko prikažemo kot sklad različnih plasti (glej sliko 2.2). Vsaka plast nudi storitve plasti nad njo.

Temelj arhitekture predstavlja Linux jedro verzije 3.x oz. 2.6 pred Ice Cream Sandwich verzijo sistema [7]. Jedro je vmesna plast med strojno in programsko opremo, ki upravlja V/I zahteve (komunikacija programske opreme s strojno).

Nad jedrom se nahaja plast s knjižnicami, napisanimi v programskem jeziku C/C++. Na tej plasti se nahaja še Android izvajalno okolje (Android Runtime). Sestavljajo ga Java knjižnice in Dalvik navidezni stroj (DVM). Te knjižnice se razlikujejo od običajnih Java SE in Java ME knjižnic, a priskrbijo večino funkcionalnosti, ki so definirane v teh. DVM je tip Java navideznega stroja (JVM), ki je v Android sistemu uporabljen za poganjanje aplikacij. Optimiziran je za nižjo računsko moč in okolja z manj pomnilnika. Za razliko od JVM, ki poganja .class datoteke, DVM poganja .dex datoteke. Te so zgrajene iz .class datotek in nudijo višjo stopnjo učinkovitosti v okoljih, kjer je sistemskih virov manj [2].

Plast višje se nahaja aplikacijsko ogrodje (ang. Application Framework). To je programska oprema, s katero so aplikacije v stalni interakciji. Nudi veliko osnovnih funkcionalnosti, na primer upravljanje z viri, ki jih uporablja aplikacija, klici, ipd.

V najvišji plasti se nahajajo Android aplikacije, tako prednameščene (aplikacija za SMS sporočanje, klicanje, ...) kot tudi tiste, ki jih kasneje sami namestimo.



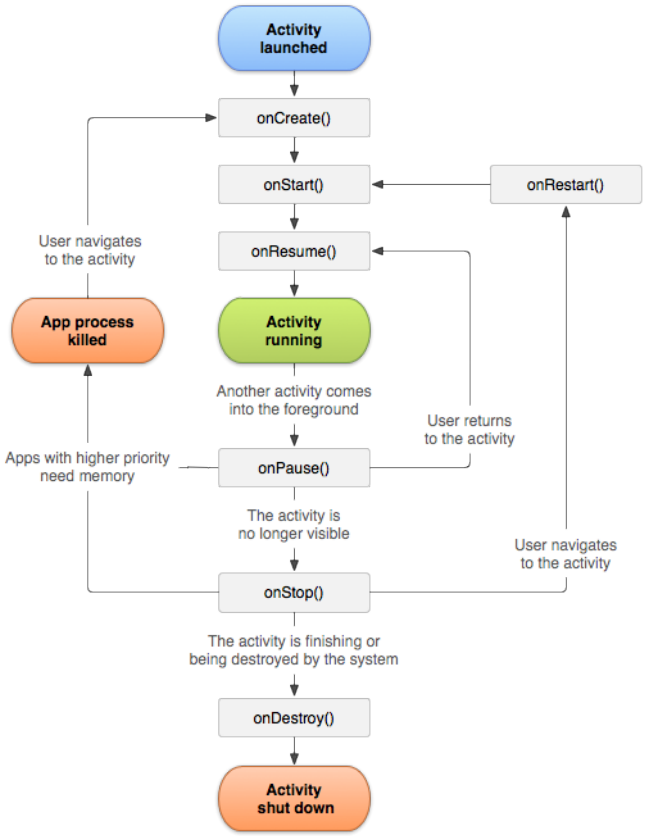
Slika 2.2: Arhitektura sistema Android.

2.3 Aplikacijske komponente v Androidu

Aplikacijske komponente so osnovni gradniki aplikacije v sistemu Android. Preko njih aplikacija na različne načine komunicira s sistemom. Obstajajo štirje tipi aplikacijskih komponent, vsaka ima svoj namen in življenjski cikel, ki določa, kdaj se komponenta ustvari in uniči:

- **Aktivnosti (ang. Activities)**
  Aktivnost je, splošno rečeno, kar je v danem trenutku uporabniku vidno na zaslonu. Prva aktivnost v aplikaciji za pošiljanje SMS sporočil je lahko pregled sporočil, druga pa, kjer pišemo odgovor. Aktivnosti so med seboj neodvisne, zato lahko aplikacija požene tudi aktivnost druge

aplikacije, če ji le-ta akcijo dovoli. Vsaka aktivnost ima svoj življenjski cikel (glej sliko 2.3).



Slika 2.3: Življenjski cikel aktivnosti.

- **Storitve (ang. Services)**
  Storitev je komponenta, ki teče v ozadju. Navadno jo zažene določena aktivnost, storitev pa se lahko veže še na druge aktivnosti. Primer storitve je recimo predvajanje glasbe v ozadju.

- **Ponudniki vsebine (ang. Content providers)**
  Ponudnik vsebine upravlja z množicami med aplikacijami deljenih podatkov. Le-te lahko shranimo v podatkovno bazo, na datotečni sistem, splet, ipd. Aplikacija lahko preko ponudnika vsebine dostopa in tudi ureja te podatke.

- **Prejemniki oddajanja (ang. Broadcast receivers)**
  Prejemnik oddajanja je komponenta, ki se odziva na sistemske signale. Tak signal je lahko na primer signal, ki ga sistem pošlje vsem aplikacijam o stanju baterije. Prejemnik oddajanja nima lastnega uporabniškega vmesnika, lahko pa generira obvestila.

2.4 Android Studio

Android Studio je novo razvojno okolje, ki temelji na IntelliJ IDEA okolju. Podobno kot Eclipse z ADT vtičnikom, razvijalcu nudi potrebna orodja za razvoj in iskanje napak v kodi [4]. Trenutno je še v razvojni fazi, a razvijalci si že lahko namestijo predogledno verzijo. Prva verzija je bila predstavljena na konferenci Google I/O 2013.

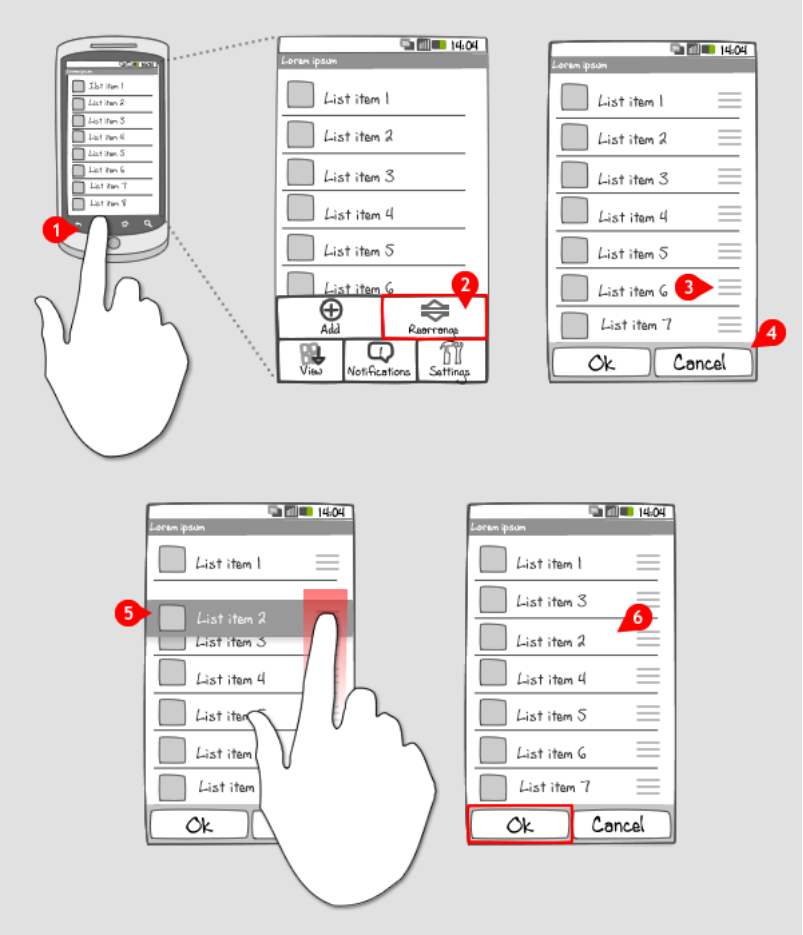
2.5 Syncfusion Metro Studio

Za izdelavo ikon v uporabniškem vmesniku sem se poslužil programa Metro Studio. Metro Studio je zbirka več kot 2000 ikon, ki jih lahko poljubno prilagajamo (barva, velikost, ozadje, ipd.). Ikone so zastonjske in jih lahko svobodno uporabimo tako v odprtokodnih projektih kot tudi v komercialne namene.

2.6 DragSortListView

DragSortListView (DSLV) je razširitev objekta ListView. ListView je eden izmed gradnikov uporabniškega vmesnika v Androidu, ki omogoča prikazovanje poljubnih podatkov v seznamu. DSLV osnovne funkcionalnosti ListViewa razširi še s podporo za preureditev podatkov v seznamu, s pomočjo akcije povleci - spusti, podobno kot prikazuje slika 2.4. Poleg tega omogoča še brisanje podatkov iz seznama z drsenjem prsta na določenem elementu v seznamu.

Avtor DSLV-ja je Carl Bauer, razširitev pa je pod Apache 2.0 licenco prosto dostopna na spletni strani github.com.



Slika 2.4: Drag & drop vzorec.

2.7 Horizontal ListView

Horizontal ListView je implementacija ListViewa, ki podatke prikaže v vodoravnem seznamu. ListView horizontalnega pogleda na podatke ne omogoča, zato smo v enem izmed realiziranih vmesnikov uporabili to razširitev. Je prav tako prostodostopen na spletni strani github.com, pod licenco MIT.

2.8 Java

Java je splošno namenski, objektno orientiran programski jezik. Deluje na vseh platformah z nameščenim Java navideznim strojem (ang. JVM - Java Virtual Machine). Je najbolj razširjen in priporočen programski jezik za razvoj Android aplikacij.

Poglavje 3

Načrtovanje uporabniških vmesnikov

Načrtovanje uporabniških vmesnikov je ena izmed najpomembnejših dejavnosti v procesu razvoja programske opreme. Tu se poslužujemo različnih tehnik, ki temeljijo tako na računalniški kot tudi kognitivni znanosti, da izdelamo interaktivne sisteme, ki zadovoljujejo določene zahteve. Največkrat se pri načrtovanju uporabniških vmesnikov poslužujemo uveljavljenih principov in navodil.

3.1 Principi za načrtovanje uporabniških vmesnikov

Principi za načrtovanje uporabniških vmesnikov so visokonivojski koncepti, namenjeni izboljšavi procesa načrtovanja uporabniških vmesnikov. Na osnovi principov nastanejo konkretna navodila, ki pa so dejanski napotki za gradnjo uporabniškega vmesnika. Le-ta se razlikujejo od sistema do sistema (npr. za Android lahko najdemo drugačna navodila, kot jih najdemo za Applov mobilni operacijski sistem iOS). Za razliko od navodil so torej principi splošnejši in jih lahko uporabimo kot vodila pri oblikovanju poljubnih uporabniških vmesnikov, tudi tistih za mobilne naprave.

Med principi je še posebej znanih 10 Nielsenovih principov [5]:

- **Vidljivost statusa sistema**
  Akcije in ukazi morajo biti vidni, odzivni časi pa čim krajši. Ob daljših odzivnih časih je potrebno uporabniku dati iluzijo napredka, npr. indikator napredka (ang. Progress bar).

- **Preslikave med sistemom in realnim svetom**
  Informacije morajo biti podane v naravnem in logičnem zaporedju. Izogibati se moramo žargonu in uporabljati splošne, uporabniku razumljive besede oz. besedne zveze.

- **Uporabnikov nadzor in svoboda**
  Uporabnik mora imeti možnost izhoda. Dialogi morajo vsebovati gumb za prekinitve, dovoliti moramo razveljavitev akcije. Brezizhodne situacije so nezaželjene. V primeru dolgih operacij (računanje nad podatki, ipd.) moramo ponuditi možnost prekinitve in razveljavitve akcije.

- **Konsistentnost in standardi**
  Konsistentnost omogoča uporabnikom, da svoje že obstoječe znanje prenesejo na uporabniški vmesnik, zato je pomembno, da zmanjšamo možnosti presenečenja. Elementi, ki izgledajo podobno, naj se tako tudi obnašajo. Elementi istega tipa, prioritete, ipd., naj izgledajo enako oz. podobno. Uporabljajo naj enako barvo, velikost ter druge atribute. Kolikor je le mogoče, moramo uporabljati navodila posameznih platform.

- **Preprečevanje napak**
  Zaželjeno je uporabiti mehanizme, ki preprečijo, da se napaka sploh zgodi. Npr. za vnos številčne vrednosti, namesto običajnega vnosnega polja, uporabimo polje, ki podpira samo številčni vnos v določenem obsegu (če ta obseg obstaja). Primer take komponente je krožno polje (ang. Spinner).

- **Prepoznavanje namesto pomnjenja**
  Uporabnikov spomin moramo čim bolj razbremeniti. Vse potrebne informacije morajo biti vidne, uporabniku naj ne bo potrebno pomniti informacij za uporabo na nekem drugem delu uporabniškega vmesnika.

- **Fleksibilnost in učinkovitost**
  Hitri ukazi (bližnjice) lahko pohitrijo interakcijo izkušenega uporabnika, a so začetniku običajno nepoznani. Uporabniku je zaželjeno omogočiti možnost prilagajanja bližnjic.

- **Estetika in minimalistično načrtovanje**
  Izogibati se moramo odvečnim informacijam, slikam, ipd. Če je le mogoče, naj imajo posamezni gradniki večkratno vlogo, paziti pa moramo, da ob tem njihove funkcionalnosti ostanejo jasne in predvidljive.

- **Javljanje napak, diagnoza in reševanje**
  Sporočila ob napaki naj bodo jasna, netehnična, uporabnikom naj v splošnem jeziku predstavijo problem in predlagajo rešitev.

- **Pomoč in dokumentacija**
  Uporabniški vmesnik moramo čimbolj izboljšati tako, da je za razumevanje potrebnega čim manj učenja oz. dokumentacije. Vseeno pa naj bo pomoč uporabniku na voljo v vsakem trenutku. Ta mora biti kratka, konstruktivna in v kontekstu.

Poglavje 4

Scenarijsko usmerjeni uporabniški vmesniki

Danes je delo na kmetiji še precej nepodprto s področja programske opreme, hiter razvoj mobilnih naprav pa je prav tu odprl veliko novih možnosti. V okviru praktičnega dela diplomske naloge smo realizirali pet različnih scenarijsko usmerjenih uporabniških vmesnikov za podporo delu na kmetiji.
Ideja scenarijsko usmerjenega uporabniškega vmesnika je, da podpira in se dobro obnaša v različnih situacijah, v katerih se lahko uporabnik oz. uporabnikova naloga nahajata.

Primer:
Možna uporabniška zgodba, ki jo podpirajo nekateri izmed realiziranih uporabniških vmesnikov:

"Uporabnik med naloge, ki jih mora opraviti danes, doda novo nalogo. Zaradi pomanjkanja časa jo hoče prestaviti med naloge, ki jih mora izvršiti v prihodnosti."

Scenarijsko usmerjen uporabniški vmesnik, mora take scenarije predvideti in jih uspešno obvladovati.

4.1 Proces realizacije

4.1.1 Zbiranje uporabniških zahtev

Dobro uporabniško izkušnjo največkrat lahko dosežemo le tako, da uporabnikom omogočimo, da na preprost in intuitiven način dosežejo svoj cilj. Za to pa je potrebno podrobno poznavanje uporabnikovih nalog in njegovih navad ter okolja, v katerem deluje. Uporabniške zahteve navadno delimo v naslednji dve skupini:

- **Funkcionalne zahteve**
  Funkcionalne zahteve so tiste, ki morajo biti implementirane, da lahko uporabnik doseže cilj oz. izpolni nalogo. V kontekstu uporabniških vmesnikov, ki smo jih izdelali, je uporabniška zahteva dodajanje nove naloge, urejanje obstoječe naloge, ipd.

- **Nefunkcionalne zahteve**
  Nefunkcionalne zahteve določajo kvalitete, ki jih mora sistem imeti, npr. varnost, uporabnost, interoperabilnost, ipd.

Uporabniške zahteve navadno zbiramo na naslednje načine [6]:

- **Intervjuji**
  Intervjuji z uporabniki so dobri, saj izzovejo veliko različnih scenarijev in prisilijo uporabnika, da razišče in premisli o problemih in vprašanjih.

- **Vprašalniki**
  Z vprašalniki navadno pridobivamo začetne odzive in širšo perspektivo na različnih temah. Vprašalnike lahko uporabimo tudi za pridobivanje mnenj in pogledov na različne predloge.

- **Direktno opazovanje**
  Direktno opazujemo udeležence v njihovem naravnem okolju, tako lažje razumemo njihove naloge in okolje, v katerem jih izvajajo. Včasih opazovanje izvršujejo trenirani opazovalci in opažanja posredujejo oblikovalski ekipi, včasih pa to naredijo kar opazovalci sami.

- **Indirektno opazovanje**
  Indirektno opazovanje se ne izvrši nad udeleženci v določenem okolju, ampak nad posameznimi zapisi (če obstajajo), ki se ustvarijo med preizkušanjem določene različice programske opreme.

- **Raziskovanje podobnih produktov**
  Z raziskovanjem in opazovanjem podobnih produktov lahko tudi določimo zahteve, ki jih mora izpolniti naš produkt.

- **Preučevanje dokumentacije**
  Tudi s preučevanjem dokumentacije produkta lahko določimo nekatere zahteve. Vedno pa moramo imeti v mislih, da je dokumentacija precej idealiziran dokument, zato ne sme biti edini vir za določanje zahtev oz. morajo imeti načini, kjer je uporabnik udeležen v proces zbiranja zahtev, večjo težo.

Ker zbiranje uporabniških zahtev zahteva precej časa in predvsem zainteresiranih udeležencev, sva za potrebe diplomskega dela z mentorjem zahteve analizirala in določila kar sama.

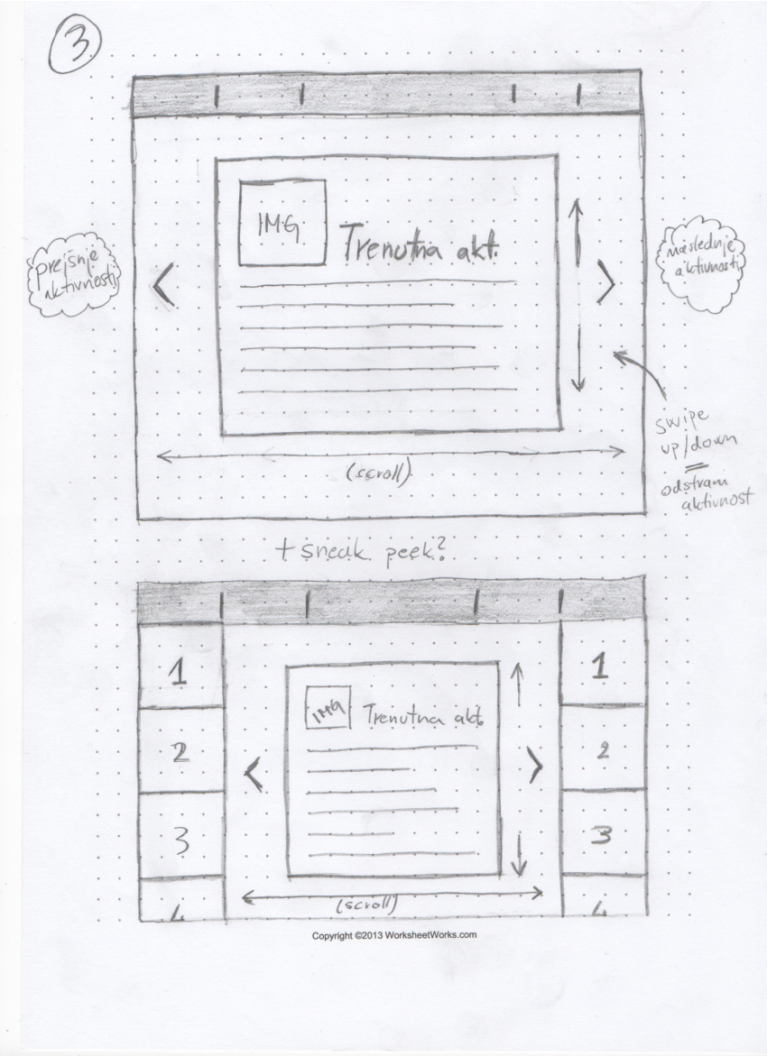
4.1.2 Papirnati prototipi

Prototipiranje na papir (slika 4.1) je na področju komunikacije človek-računalnik široko uporabljena metoda za načrtovanje uporabniških vmesnikov. Je zelo preprosta, a učinkovita metoda, saj že v zgodnji fazi razvoja priskrbi obilo uporabnih povratnih informacij in tako pripomore k boljši končni obliki produkta. Prototipiranje na papir je priporočljiva metoda, saj nam nudi veliko prednosti:

- Hitrejši razvoj, saj hitreje skiciramo, kot pa programiramo.
- Lažje spreminjamo, papirnati prototip lahko tudi zavržemo.
- Tudi ljudje brez tehničnega znanja so nam lahko v pomoč.

- Pozornost je usmerjena na veliko sliko in ne na detajle v uporabniškem vmesniku.

Skiciramo lahko prikaze različnih oken v vmesniku, prikaze dialogov, ipd. Pri testiranju vmesnika oseba simulira računalnik oz. aplikacijo tako, da ob ustreznih dogodkih prikaže ustrezno skico.



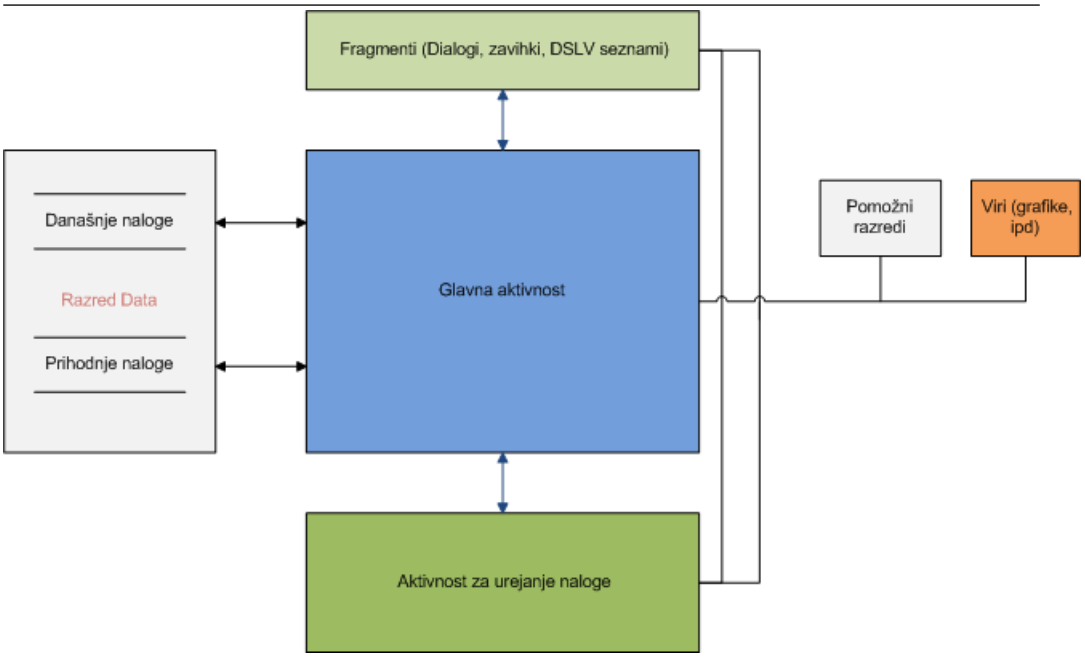
Slika 4.1: Primer papirnega prototipa za enega izmed realiziranih vmesnikov. Realiziran vmesnik nekoliko odstopa od začetnega prototipa.

4.1.3 Kodiranje

Fazi načrtovanja prototipov je sledila faza kodiranja posameznih vmesnikov. Implementirali smo pet različnih delujočih prototipov z vso potrebno logiko. Prototipi so medseboj povsem neodvisni. Za razvoj smo uporabili že prej omenjeno razvojno okolje Android Studio. Osnovno strukturo projekta nam okolje ob kreiranju projekta že samo ustvari. Glavne datoteke se nahajajo v mapah src in res. Mapa src vsebuje zgolj izvorno kodo, zapisano v java datotekah. V mapi res se nahajajo vsi potrebni viri (slike, zvočne datoteke, definicije barv, ipd.), ki jih aplikacija potrebuje. Uporabniške vmesnike v Androidu lahko definiramo v izvorni kodi (v java datotekah) ali v XML datotekah. Le-te se nahajajo v mapah layout in layout-land in tako skušajo ločiti uporabniški vmesnik od logike, a največkrat brez mešanja obeh pristopov ne gre.

Struktura aplikacije

Zaradi lažje predstavitve je v aplikaciji združenih vseh pet uporabniških vmesnikov, do katerih dostopamo v začetnem meniju. Vsak izmed vmesnikov pa je v realnosti mišljen kot vmesnik za eno, samostojno aplikacijo. Grobo strukturo posameznega vmesnika predstavlja slika 4.2, dejanska struktura pa z manjšimi razlikami variira od vmesnika do vmesnika.
Ob vstopu v aplikacijo se nam prikaže glavna aktivnost, ki v glavnem upravlja z vsemi dogodki v vmesniku. V začetku se zgradita DSLV fragmenta, ki prikazujeta seznama nalog, v vmesnikih Alfa, Beta in Delta pa še fragment, ki omogoča drsenje skozi oba seznama nalog. Podatke, t.j. naloge, fragmenta prejmeta preko razreda Data. Ta razred je vmesnik za dostop in delo s podatki. Omogoča operacije, kot so dostop do nalog, dodajanje le-teh, itd. Za potrebe diplomskega dela se podatki zgenerirajo ob zagonu aplikacije, v realnem svetu pa bi bil razred Data vmesnik za dostop do kateregakoli vira podatkov. To bi lahko bila podatkovna baza, določen vir v oblaku, ipd.



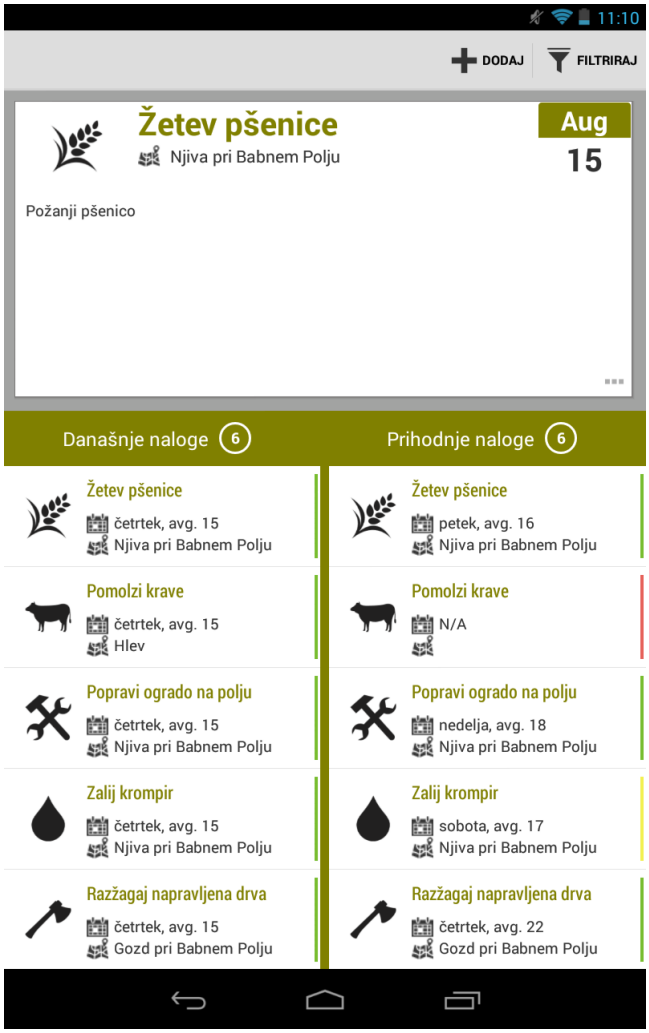
Slika 4.2: Struktura uporabniškega vmesnika.

Iz glavne je možno preiti v aktivnost za urejanje izbrane naloge. Glavna aktivnost izbrano nalogo pošlje aktivnosti za urejanje, le-ta pa jo vrne nazaj, takoj ko uporabnik zaključi z delom. Glavna aktivnost nato urejeno nalogo pošlje razredu Data, ki jo uvrsti v pripadajoči seznam. Enako velja za dodajanje nove naloge, le da glavna aktivnost aktivnosti za urejanje ne pošlje nobenih podatkov. V zbirki, ki je na sliki 4.2 označena kot pomožni razredi, so zbrani različni razredi, potrebni za učinkovito delovanje aplikacije. Vire predstavlja mapa res, ki jo najdemo v projektu. Tu se nahajajo različne datoteke, ki sestavljajo oz. definirajo posamezne dele uporabniškega vmesnika.

Uporabniški vmesnik Alfa

Prvi uporabniški vmesnik smo poimenovali Alfa. Sestavljajo ga dve glavni komponenti (t.j. aktivnosti) ter trije fragmenti (pomožne komponente). Fragment predstavlja obnašanje dela uporabniškega vmesnika v določeni aktivnosti. Lahko si ga predstavljamo kot modularen del aktivnosti, ki ima

svoj življenjski cikel. Fragmenti lahko imajo svoj uporabniški vmesnik ali pa ne, navadno pa jih uporabimo, da lahko posodobimo le del celotnega uporabniškega vmesnika.

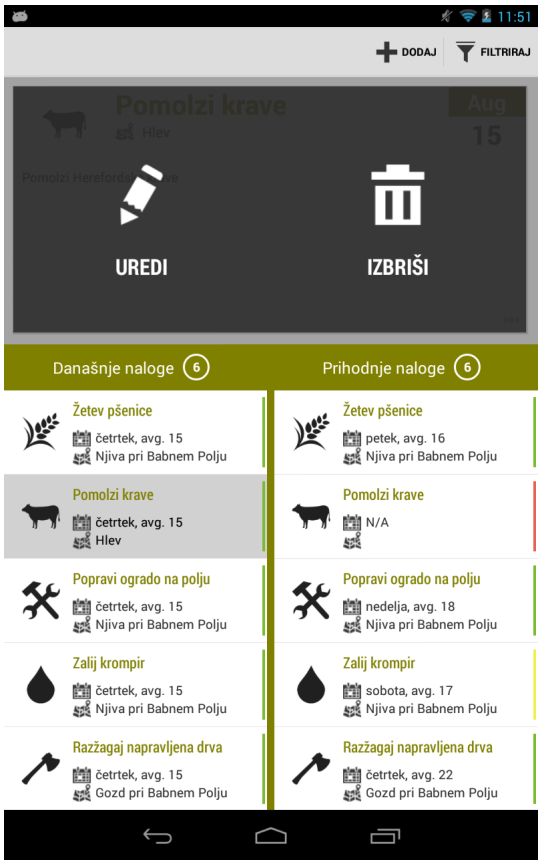


Slika 4.3: Uporabniški vmesnik Alfa.

Glavne komponente v uporabniškem vmesniku Alfa:

- **Glavna aktivnost**
  Glavna aktivnost je vse, kar vidimo na sliki 4.3. Ta aktivnost inicializira vse elemente uporabniškega vmesnika, vključno s potrebnimi fragmenti. Ti fragmenti sta oba seznama nalog (današnje in prihodnje) ter fragment na vrhu, kjer je prikazana trenutno izbrana naloga, ki pa hkrati omogoča še drsenje skozi vse današnje in prihodnje naloge.

- **Fragment za pregledovanje nalog**
  To je fragment, ki omogoča drsenje skozi vse naloge, privzeto pa prikazuje trenutno izbrano nalogo. Tukaj lahko tudi brišemo ali urejamo trenutno izbrano nalogo (glej sliko 4.4).
  Za drsenje skozi naloge smo uporabili komponento ViewPager, ki jo najdemo v podporni knjižnici v sistemu Android. ViewPager omogoča zelo hitro drsenje skozi seznam strani (posamezna stran predstavlja posamezno nalogo) z drsenjem prsta v levo ali desno smer. Na ta način se tudi znebimo morebitnih dodatnih komponent, kot sta recimo gumba 'naprej' in 'nazaj', se pa zaradi tega omejimo na naprave z zaslonom na dotik. Zaželjeno je, da ob uporabi take komponente vključimo tudi nekakšen indikator, ki bo uporabniku služil za predstavo, kje približno v seznamu se nahaja. Ker je v primeru uporabniškega vmesnika Alfa možno ogromno število nalog, smo tak indikator izpustili iz uporabniškega vmesnika, saj bi lahko pokvaril splošno uporabniško izkušnjo.



Slika 4.4: Možne akcije nad trenutno izbrano nalogo v vmesniku Alfa.

- **Fragmenta z nalogami**
  To sta fragmenta, ki se nahajata na levi oz. desni in vsebujeta seznam nalog. Za prikaz seznama smo uporabili razširitev DragSortListView, ki smo jo še dodatno razširili s svojo kodo, da smo dosegli željeno obnašanje. DragSortListView je razširitev Android komponente ListView. Ponuja iste funkcionalnosti, poleg tega pa še nudi možnost brisanja elementa (naloge) z drsenjem v levo oz. desno stran ter urejanja vrstnega reda podatkov (nalog) z drsenjem v željeni smeri (navzgor oz. navzdol). Tak vzorec je dober in zaželjen, saj je vrstni red nalog pomemben, možnost urejanja pa nujna. Je zelo preprost in intuitiven, saj so vsi premiki nalog vidni ter tako uporabniku nudijo vizualno pred-

stavo o trenutnem dogajanju. Žal pa ima tak vzorec tudi nekaj slabih lastnosti. Ena izmed teh je, da je možen le na napravah z zaslonom na dotik. Na napravah brez takega zaslona tak uporabniški vmesnik postane skoraj neuporaben. Druga slaba lastnost pa je, da so deli, ki prenos začnejo, sprva lahko nerazpoznavni. Za primer vzemimo prav vmesnik Alfa. Možnost prenosa naloge iz seznama v seznam se aktivira izključno, če uporabnik dalj časa (500 ms v Androidu) pritisne na sliko (ikono) naloge. V nasprotnem primeru, če isto akcijo izvrši nad preostalim vizualnim delom naloge, se aktivira možnost urejanja podatkov v seznamu.

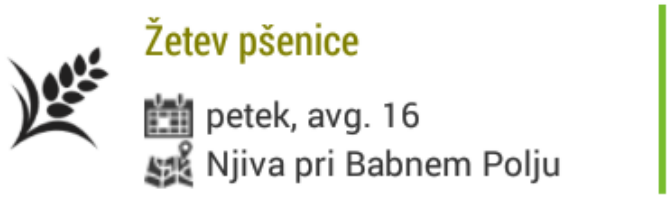
Med obema seznamoma lahko naloge tudi prenašamo z akcijo povleci - spusti (ang. Drag & Drop). Za prenos naloge moramo dalj časa pritisniti na njeno sliko oz. ikono ter jo prenesti v željen seznam. Podatki v nalogi se ob prenosu ne spremenijo, z izjemo datuma. Če nalogo prenesemo v seznam z današnjimi, se njen datum avtomatsko nastavi na današnjega, sicer pa na jutrišnjega. Vse podatke pa lahko po želji naknadno urejamo s klikom na nalogo. Dobra lastnost tega pristopa je, da gre za direktno manipulacijo nad objektom, saj tako uporabnik dobi občutek nadzora. Prostor, ki ga je prej zasedala prenašana naloga, nemudoma zasede njena naslednica. Na drugi strani pa je pomanjkljivost tega pristopa to, da mora uporabnik s poskušanjem ugotoviti, kaj in kako lahko prenaša. Kot sem že omenil, prenos naloge je možen le z daljšim klikom na njeno sliko.

Tipi nalog

V analizi uporabniških zahtev smo predvideli naslednje naloge:

- Molža,
- sečnja,
- žetev,
- zalivanje,
- popravilo.

V trenutni aplikaciji se glede na podatke ne razlikujejo (z izjemo ikon, ki so različne za vsak tip). Ideja je, da se v konkretni aplikaciji naloge za vsak tip prilagodijo glede na konkretne uporabniške zahteve.



Slika 4.5: Primer naloge v aplikaciji.

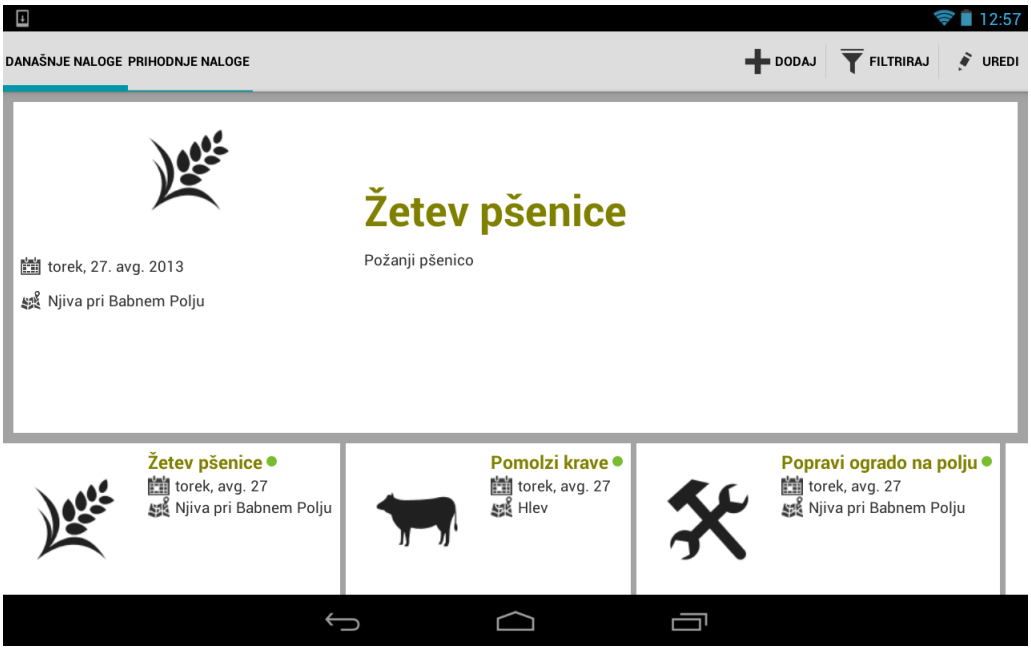
Slika 4.5 prikazuje primer prikaza ene naloge v seznamu. Levi del sestavlja ikona, ki označuje tip naloge. Ikona nudi grafično informacijo in tako omogoča uporabniku takojšnjo prepoznavo naloge v celotnem seznamu. Sredinski del prikaza sestavljajo najosnovnejši podatki, ki so skupni vsem tipom nalog. Čisto na vrhu imamo naslov oz. ime naloge. Ker je poleg slike, ki nakazuje tip naloge, njeno ime bolj pomemben podatek, je pisava v tem delu nekoliko večja in obarvana. Na ta način ime in ikona skupaj najbolj pritegneta uporabnikovo pozornost in omogočata hitro razločevanje med nalogami. Pod imenom se nahajata še datum naloge in lokacija. Na njuni levi strani sta ikoni, ki nakazujeta tip posamezne naloge. Skrajno desno se nahaja

indikator stanja naloge. Stanje naloge določa celovitost vseh podatkov posamezne naloge. Če ima naloga vnešene vse podatke, je njeno stanje popolno, kar nakazuje indikator zelene barve. Indikator rumene barve nakazuje delno vnešeno stanje, kar pomeni, da ima naloga poleg imena vnešene še nekatere druge podatke, ne pa vseh zahtevanih. Naloga, ki ima samo ime, je smatrana kot nepopolna in je označena z indikatorjem rdeče barve.

V vseh vmesnikih lahko naloge tudi filtriramo po njihovih stanjih.

Uporabniški vmesnik Beta

Uporabniški vmesnik Beta je prvi uporabniški vmesnik, ki je horizontalno orientiran, in edini, ki uporablja zavihke. Zavihka sta dva in se nahajata na levi strani v ActionBaru (vrstica na vrhu uporabniškega vmesnika). Zavihki omogočajo hitro visoko nivojsko navigacijo oz. brskanje med različnimi kategorijami podatkov. V našem vmesniku sta dve kategoriji, in sicer današnje in prihodnje naloge.



Slika 4.6: Uporabniški vmesnik Beta.

V vsakem zavihku imamo pregled naloge, ki zavzema večji del razpoložljivega prostora. V njem so vidni vsi podatki o posamezni nalogi. Pod pregledom naloge se nahaja vodoraven seznam z vsemi nalogami izbrane kategorije. Vodoraven seznam je v tem vmesniku boljša izbira kot navpičen, saj je tudi sam vmesnik vodoravno orientiran in posledično je v vodoravnem seznamu več prostora za predstavitev potrebnih podatkov posamezne naloge kot v navpičnem. V primerjavi z navpičnim seznamom vodoravni daje pristnejši občutek časovnega traku, ki je sploh v našem primeru zaželjen, saj je čas oz. datum eden izmed pomembnejših podatkov v posamezni nalogi.
Večja pomanjkljivost tega vodoravnega seznama je nepodprtost urejanja nalog z akcijo povleci - spusti. Ker Android ne omogoča prikaza podatkov v vodoravnem seznamu, smo za prikaz nalog v vodoravni ureditvi uporabili knjižnico HorizontalListView [3]. Le-ta pa ne omogoča urejanja, zato smo implementirali dodatno aktivnost (slika 4.7), v kateri lahko podatke urejamo z akcijo povleci - spusti. Do aktivnosti pridemo s klikom na gumb uredi, ki se nahaja v ActionBaru. Poleg urejanja vrstnega reda nalog ta aktivnost omogoča tudi urejanje podatkov posamezne naloge.
V začetku so v tej aktivnosti prikazani podatki o prvi današnji nalogi. Z drsenjem v desno se ta pogled umakne in prikažeta se seznama nalog, kot kaže slika 4.7. S klikom na posamezno nalogo se seznama ponovno skrijeta, izbrana naloga pa je pripravljena na urejanje. V teh seznamih lahko tudi urejamo vrstni red nalog z akcijo povleci - spusti.
V zgornjem desnem kotu se nahajajo še gumbi shrani, zavrzi in nazaj. Prva dva shranita oz. zavržeta urejene podatke v trenutno izbrani nalogi, zadnji pa zapre trenutno aktivnost in nas prestavi na začetno aktivnost (slika 4.6).

Slika 4.7: Aktivnost za urejanje nalog v uporabniškem vmesniku Beta.

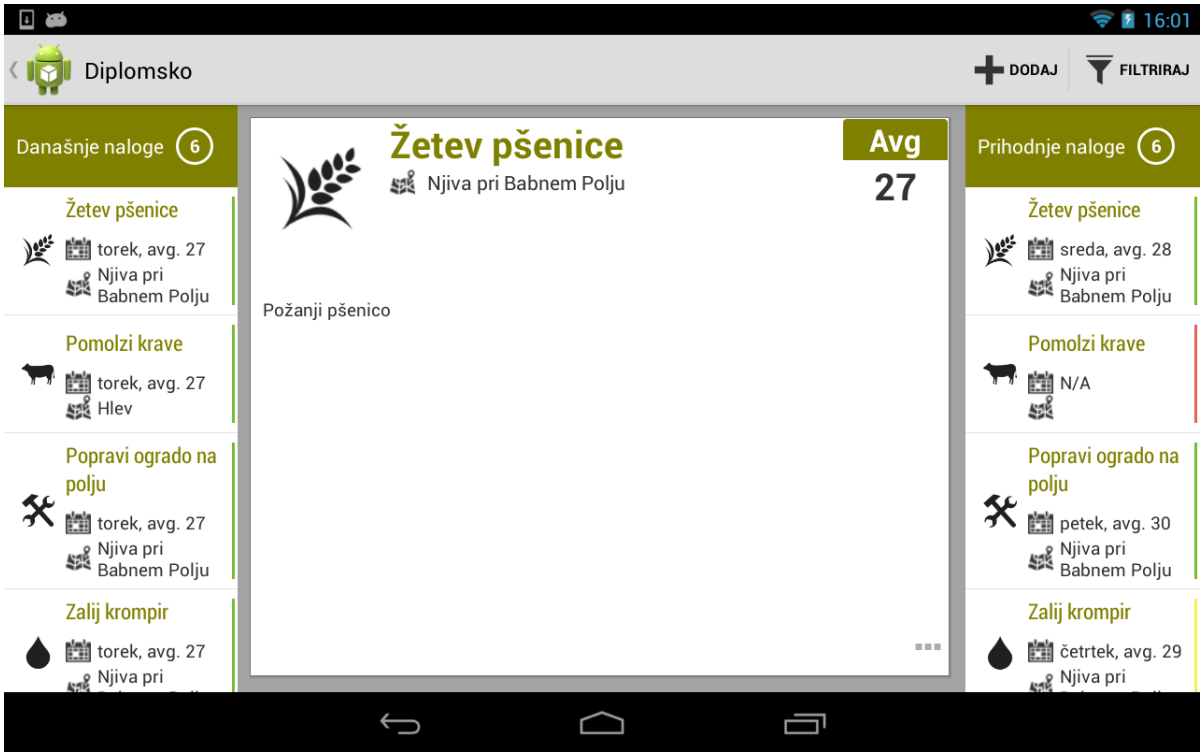
Na desni strani ActionBara se nahajajo gumbi za tri akcije, ki jih lahko izvajamo nad podatki v tem vmesniku. To so dodajanje nove naloge ter filtriranje in urejanje obstoječih nalog. Filtriranje naloge deluje tako kot v ostalih vmesnikih, in sicer glede na stanje naloge.

Uporabniški vmesnik Gama

Tretji uporabniški vmesnik, imenovan Gama (slika 4.8), je dejansko vodoravna postavitev uporabniškega vmesnika Alfa. Seznama z današnjimi oz. prihodnjimi nalogami sta postavljena skrajno desno oz. levo. V sredini se nahaja fragment, ki prikazuje trenutno izbrano nalogo. Posledično Gama podeduje vse prednosti in slabosti Alfe, doda pa tudi svoje prednosti:

- Komponenta, ki prikazuje trenutno nalogo, je večja. Posledično imamo za prikaz potrebnih podatkov več prostora, kar pripomore k večji preglednosti naloge.

- Seznama pri strani sta po višini krajša, torej je hkrati vidnih manj nalog. To je za uporabnika dobro, saj lahko seznam z veliko vidnimi podatki da občutek prenatrpanosti in bremena.



Slika 4.8: Uporabniški vmesnik Gama.

Tudi v tem vmesniku lahko uporabnik z akcijo povleci - spusti prenaša naloge med seznamoma. Tako mu je omogočena večja stopnja nadzora, saj gre za direktno manipulacijo nad nalogo. Poleg prenosa je z akcijo povleci - spusti podprto tudi urejanje vrstnega reda nalog v določenem seznamu. Fragment v sredini privzeto prikazuje treutno izbrano nalogo, sicer pa kot v Alfi podpira tudi brskanje med vsemi nalogami.

Uporabniški vmesnik Delta

Pri uporabniškem vmesniku Delta so bile v obzir vzete tudi naprave z manjšim zaslonom. Seznama z nalogami sta privzeto skrita, vidimo jih pa lahko s potegom iz leve (današnje naloge) oz. desne strani zaslona (prihodnje naloge). Privzeto je vidna le trenutno izbrana naloga, z vleko v levo ali desno pa prikažemo prejšnjo oz. naslednjo nalogo. Tudi tukaj so naloge urejene kronološko - po vrsti. Skrajno levo je prva izmed današnjih nalog, skrajno desno pa zadnja izmed prihodnjih nalog. Na ta način uporabniku zagotovimo občutek časovnega traku.

Slika 4.9: Dodatni menu v uporabniškem vmesniku Delta.

Pomanjkljivost tega vmesnika je, da ne podpira povleci - spusti prenosa iz enega seznama v drugega, saj je lahko v danem trenutku odprt samo en

seznam. V dodatni meni, ki ga nakazujejo tri pikice, smo dodali izbiro za prenos (glej sliko 4.9). Omenjena izbira zamenja izvor naloge glede na njen trenutni izvor. Primer: če izbrana naloga spada med današnje, jo klik na to izbiro prenese med prihodnje. Nalogi se ob prenosu spremeni datum. Le-ta se nastavi na jutrišnjega, kasneje pa ga lahko spremenimo v poljubnega. Na ta način smo v vmesniku preprečili možnost, da bi se v seznamu nahajala naloga z napačnim oz. neveljavnim datumom.

Uporabniški vmesnik Epsilon

Uporabniški vmesnik Epsilon (glej sliko 4.10) je tako kot Delta primeren tudi za naprave z manjšim zaslonom, saj je seznam z nalogami skrit, prikažemo pa ga z drsenjem v desno. Tako imamo za prikaz trenutno izbrane naloge na voljo skoraj ves zaslon. Bistvena sprememba v tem vmesniku je, da naloge niso več razvrščene med današnje in prihodnje naloge, pač pa so vse naloge v enem seznamu urejene kronološko, od prve do zadnje. Poleg filtriranja imamo dodano možnost sortiranja nalog po datumu naraščajoče ali padajoče.
Za seznam smo zopet uporabili DSLV, ki kot vir nalog uporablja seznam današnjih in prihodnjih. Skrivanje seznama nalog smo implementirali z uporabo SlidingPaneLayout-a. Tako smo realizirali nekakšen Master-Detail vmesnik, kjer lahko seznam skrijemo, podatki o nalogi pa ostanejo vidni na ekranu.
SlidingPaneLayout vzame prvo komponento (prvega otroka) v tej postavitvi kot ploščo, ki jo lahko skrijemo oz. povlečemo iz leve strani. V tem vmesniku je to fragment, ki prikazuje seznam nalog. Drugega otroka pa vzame kot glavno komponento za prikazovanje podrobnih informacij. Ta komponenta je v tem vmesniku prikaz podatkov o posamezni nalogi in zasega ves razpoložljiv prostor na ekranu. SlidingPaneLayout je zelo uporabna komponenta, hkrati pa je preprosta za uporabo in ima velik doprinos k dobri uporabniški izkušnji.
Zato ker lahko seznam skrivamo, je ta vmesnik primeren tudi za naprave z manjšim zaslonom. Sicer bi bil uporabniški vmesnik prenatrpan, upo-

rabniška izkušnja pa znatno slabša. Ker v tem uporabniškem vmesniku ni več ločitve naloge na današnje in prihodnje, je bila nujna implementacija možnosti filtriranja in sortiranja nalog. Tako smo v tem vmesniku poleg filtriranja glede na stanje naloge dodali še možnost sortiranja po datumu padajoče ali naraščajoče. Uporabnik lahko tako le v nekaj trenutkih najde potrebne informacije o določenih nalogah.

Slika 4.10: Uporabniški vmesnik Epsilon z odprtim seznamom nalog.

Poglavje 5

Zaključek

V okviru diplomske naloge smo razvili pet različnih prototipov scenarijsko usmerjenih uporabniških vmesnikov. Ker je izdelati splošen uporabniški vmesnik skoraj nemogoče, smo se usmerili v izdelavo uporabniškega vmesnika za aplikacijo za podporo delu na kmetiji.
Uporabniški vmesniki niso dokončani, saj je za to potrebno poznati konkreten namen in zahteve aplikacije, so pa dobro izhodišče za izdelavo take aplikacije. Realizirali smo jih za operacijski sistem Android 4.0 ali več, saj so nekatere komponente na voljo šele v teh različicah in so ključne za zagotavljanje dobre uporabniške izkušnje.
Pri načrtovanju in razvoju uporabniških vmesnikov smo v veliki meri sledili splošnim principom in navodilom za razvoj uporabniških vmesnikov za operacijski sistem Android. Vmesniki so prilagojeni za tablične računalnike, z izjemo vmesnikov Delta in Epsilon, ki sta primerna tudi za uporabo na mobilnih napravah z manjšim zaslonom (pod 7 palcev). V prihodnosti bi bilo dobro na teh uporabniških vmesnikih postoriti naslednje reči:

- Vmesnike Alfa, Beta in Gama bi bilo dobro prilagoditi še za manjše mobilne naprave, saj je trenutna razvrstitev komponent neprimerna za manjše zaslone.
- V vseh vmesnikih je dobrodošla možnost razveljavitve akcije, še posebej brisanja posamezne naloge.

- V uporabniškem vmesniku Beta bi bilo dobro podpreti urejanje nalog z akcijo povleci - spusti. Potrebna je torej razširitev ListView-a, ki prikazuje podatke v vodoravni ureditvi in omogoča tako urejanje.

Menim, da je delo na kmetiji še precej slabo podprto s strani informacijske tehnologije, možnosti na tem področju pa so velike. Izdelani vmesniki so lahko dobro izhodišče za mobilne aplikacije, ki bodo za podporo delu na kmetiji nastajale v prihodnosti.

Slike

Tabele

Literatura

[1] S. Hoober, E. Berkman *Designing Mobile Interfaces*. O'Reilly, 2011, str. xiii-xviii.

[2] Android-apps (2013). Dostopno na: http://www.android-app-market.com/android-architecture.html

[3] Dev-Smart (2013). Dostopno na: http://www.dev-smart.com/archives/34

[4] Google Inc. (2013). Dostopno na: http://developer.android.com/sdk/installing/studio.html

[5] Jakob Nielsen (2005). Dostopno na: http://www.nngroup.com/articles/ten-usability-heuristics/

[6] Usability Geek (2013). Dostopno na: http://usabilitygeek.com/requirements-gathering-user-experience-pt1/

[7] Wikipedia (2013). Dostopno na: http://en.wikipedia.org/wiki/Android_(operating_system))