Zbirka nalog iz načrtovanja uporabniških vmesnikov

Aleš Smrdel, Ciril Bohak, Miha Amon, Franc Jager

2017

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID=292758784

ISBN 978-961-6209-95-3 (pdf)

Elektronska izdaja knjige je na voljo na URL:
http://zalozba.fri.uni-lj.si/smrdel2017.pdf

Recenzenta: viš. pred. dr. Borut Batagelj, viš. pred. dr. Alenka Kavčič
Založnik: Založba UL FRI, Ljubljana
Izdajatelj: UL Fakulteta za računalništvo in informatiko, Ljubljana
1. izdaja, 2017
Urednik: prof. dr. Franc Solina

Kazalo

1 Programska arhitektura uporabniških vmesnikov

1. Kaj je podoba (angl. "widget") uporabniškega vmesnika? Kaj podobe omogočajo?

   Rešitev: Podoba je komponenta uporabniškega vmesnika, na primer meni, drsnik, vnosno polje, gumb oziroma drug element, s katerimi lahko gradimo uporabniške vmesnike. Podobe omogočajo visokonivojsko abstrakcijo za orodja za načrtovanje uporabniških vmesnikov. Orodja so tako zbirke podob, ki omogočajo enostavnejšo in lažjo gradnjo uporabniških vmesnikov.

2. Razložite razliko med AWT in Swing s stališča grafičnega prikaza gradnikov (podob).

   Rešitev: AWT je prilagodljivo in prenosljivo ogrodje razredov. Podprtim gradnikom so poiskani enakovredni sorodniki v operacijskih sistemih. Gradnike izriše gostujoči operacijski sistem. Aplikacije, ki so napisane v AWT, se razlikujejo med operacijskimi sistemi.

   Swing vsebuje razrede, izpeljane iz hierarhije AWT, kjer je osnovni gradnik *JComponent*. Gostujoči operacijski sistem prispeva pravokotno risalno površino. Javanski navidezni stroj nariše lahke gradnike v stanju, v kakršnem so se znašli. Težki gradniki, kot so na primer *JFrame*, *JWindow* in *JDialog*, so gradniki, ki so povezani z izvornimi gradniki operacijskega sistema. Za njihov izris je zadolžen gostujoči operacijski sistem.

3. Naštejte nekaj lahkih vsebovalnikov okolja Swing, ki jih najdete tudi v generatorju vmesnikov NetBeans.

   Rešitev: Lahki vsebovalniki okolja Swing so:

   a) lahki generični vsebovalnik plošča (tudi "Panel" oziroma *JPanel*);
   b) vsebovalnik večstranska površina z možnostjo izbire (tudi "Tabbed Pane" oziroma *JTabbedPane*);

c) vsebovalnik razcepljena površina (tudi "Split Pane" oziroma *JSplitPane*);

d) drsna površina oziroma vsebovalnik lahkih komponent z drsniki (tudi "Scroll Pane" oziroma *JScrollPane*);

e) orodna vrstica (tudi "Tool Bar" oziroma *JToolBar*);

f) vsebovalnik za notranja okna (tudi "Desktop Pane" oziroma *JDesktopPane*);

g) notranje okno oziroma lahki notranji vsebovalnik okvir (tudi "Internal Frame" oziroma *JInternalFrame*);

h) vsebovalnik po globini (tudi "Layered Pane" oziroma *JLayeredPane*).

4. Naštejte nekaj načinov razvrščanja komponent v vsebovalnikih okolja AWT oziroma v vsebovalnikih okolja Swing.

   Rešitev: Nekateri načini razvrščanja so:

   a) programer nastavlja koordinate in dimenzije komponent (ali "Absolute Positioning" oziroma NULL);

   b) od leve proti desni, od zgoraj navzdol (ali tekoče razvrščanje oziroma *FlowLayout*);

   c) razvrščanje v mrežo (ali *GridLayout*);

   d) robno razvrščanje (ali *BorderLayout*);

   e) razvrščanje ena naenkrat (ali *CardLayout*);

   f) razvrščanje v mrežo različno velikih celic (ali *GridBagLayout*);

   g) razvrščanje po vertikali ali horizontali (ali *BoxLayout*);

   h) razvrščanje z vzmetmi (ali *SpringLayout*);

   i) skupinsko razvrščanje (ali "Free Design" oziroma *GroupLayout).*

5. Napišite aplikacijo (program), ki se bo lahko izvajala na oddaljenem računalniku, okna pa se bodo prikazovala na lokalnem računalniku. Aplikacija naj vsebuje okno velikosti 400×300 pikslov. Aplikacija naj v zanki čaka na dogodke, ki jih proži uporabnik. V primeru pritiska tipke na tipkovnici naj se v terminalu izpiše tipka, ki je bila pritisnjena. Če pa je bila pritisnjena katera izmed tipk "b", "c", "r", "z" ali "m", pa naj se nastavi še barva ospredja okna (barva, s katero rišemo) na belo, črno, rdečo zeleno oziroma modro barvo. V primeru pritiska levega gumba na miški naj se v oknu izriše zapolnjen pravokotnik v barvi ospredja. Prvi pritisk naj samo določi položaj zgornjega levega oglišča pravokotnika, medtem ko drugi pritisk določi spodnje desno oglišče pravokotnika in povzroči izris. Ob pritisku srednjega gumba na miški naj se vsebina okna izbriše, ob pritisku desnega gumba na miški pa naj se okno (aplikacija) zapre. Napišite tudi ukaz za prevajanje te aplikacije.

Rešitev: Pri tej rešitvi smo se odločili za uporabo okenskega sistema X11 in programskega jezika C. Aplikacijo bomo razdelili na nekaj funkcij. Najprej napišemo funkcijo (koda je prikazana v listingu 1.1), ki zgradi in prikaže okno, pri čemer predpostavimo, da so spremenljivke, ki jih uporabljamo v tej funkciji in niso najavljene (angl. "declared") znotraj te funkcije, najavljene globalno.

```
1  void inicializirajX() {
2    unsigned long crna,bela;
3
4    prikaz = XOpenDisplay((char *)0);
5    zaslon = DefaultScreen(prikaz);
6    barve = DefaultColormap(prikaz, zaslon);
7    crna = BlackPixel(prikaz,zaslon);
8    bela = WhitePixel(prikaz, zaslon);
9
10   okno=XCreateSimpleWindow(prikaz, DefaultRootWindow(prikaz
       ), 0, 0, 400, 300, 5, crna, bela);
11   XSetStandardProperties(prikaz, okno, "Naslov okna", "
       Naslov minimiziranega okna", None, NULL, 0,NULL);
12   XSelectInput(prikaz, okno, ExposureMask|ButtonPressMask|
       KeyPressMask);
13
14   kontekst=XCreateGC(prikaz, okno, 0, 0);
15   XSetBackground(prikaz, kontekst, bela);
16   XSetForeground(prikaz, kontekst, crna);
17
18   XClearWindow(prikaz, okno);
19   XMapRaised(prikaz, okno);
20 }
```

Listing 1.1 Inicializacija povezave s strežnikom X, kreiranje okna in prikaz okna.

S kodo funkcije, ki je prikazana v listingu 1.1, najprej odpremo povezavo s strežnikom X, dobimo številko zaslona, privzeto barvno paleto ter vrednosti, ki določata črni in beli piksel (vrstice od 4 do 8). Nato kreiramo okno, mu določimo osnovne značilnosti in določimo, katere dogodke naj strežnik X javlja oziroma kateri dogodki so zanimivi (vrstice od 10 do 12). Nato določimo grafični kontekst in barvo ozadja ter ospredja (vrstice od 14 do 16). Na koncu okno še počistimo in ga prikažemo (vrstici 18 in 19).

Sledi funkcija, s katero lahko zapremo okno, ki jo prikazuje koda v listingu 1.2.

```
1 void zapriX() {
2   XFreeGC(prikaz, kontekst);
3   XDestroyWindow(prikaz, okno);
4   XCloseDisplay(prikaz);
5   exit(1);
6 }
```

Listing 1.2 Razkrojitev grafičnega konteksta, uničenje okna in prekinitev povezave s strežnikom X.

Pri zapiranju okna najprej razkrojimo grafični kontekst (vrstica 2), nato razkrojimo okno in vsa podokna (vrstica 3), na koncu pa še zapremo povezavo s strežnikom X, s čimer razkrojimo vse resurse, ki jih je odjemalec ustvaril na strežniku (vrstica 4), in končamo z izvajanjem (vrstica 5).

Za tem napišemo še funkcijo, v kateri čakamo na dogodke, ki jih sproži uporabnik aplikacije (koda v listingu 1.3). Dogodki, ki jih aplikacija zazna, so dogodki ob pritisku na tipko, ob pritisku gumba na miški in ob prikazu okna.

```
1  void zankaDogodkov(){
2    XEvent dogodek;
3    KeySym kljuc;
4    char tekst[255];
5    int pritisk=0;
6    int zacX, zacY;
7
8    while(1) {
9      XNextEvent(prikaz, &dogodek);
10     if (dogodek.type==KeyPress && XLookupString(&dogodek.
         xkey,tekst,255,&kljuc,0)==1){
11       if (tekst[0]=='b') {
12         barva.red = 65535; barva.green = 65535; barva.blue
             = 65535;
13       }
14       if (tekst[0]=='c') {
15         barva.red = 0; barva.green = 0; barva.blue = 0;
16       }
17       if (tekst[0]=='r') {
18         barva.red = 65535; barva.green = 0; barva.blue = 0;
19       }
20       if (tekst[0]=='z') {
21         barva.red = 0; barva.green = 65535; barva.blue = 0;
```

```
22          }
23          if (tekst[0]=='m') {
24            barva.red = 0; barva.green = 0; barva.blue = 65000;
25          }
26          barva.flags = DoRed | DoGreen | DoBlue;
27          XAllocColor(prikaz, barve, &barva);
28          XSetForeground(prikaz, kontekst, barva.pixel);
29          printf("Pritisnili ste tipko %c!\n",tekst[0]);
30        }
31
32        if (dogodek.type==ButtonPress) {
33          if (dogodek.xbutton.button==1){
34            if (pritisk==0){
35              zacX=dogodek.xbutton.x;
36              zacY=dogodek.xbutton.y;
37            }
38            else{
39              if (zacX<dogodek.xbutton.x && zacY<dogodek.
                    xbutton.y)
40                XFillRectangle(prikaz, okno, kontekst, zacX,
                      zacY, dogodek.xbutton.x-zacX, dogodek.
                      xbutton.y-zacY);
41            }
42            pritisk=(pritisk+1)%2;
43          }
44          else if (dogodek.xbutton.button==2)
45            XClearWindow(prikaz, okno);
46          else if (dogodek.xbutton.button==3)
47            zapriX();
48        }
49      }
50  }
```

Listing 1.3 Čakanje na dogodke ob pritisku tipke na tipkovnici, dogodke ob pritisku gumba na miški in dogodke ob prikazu okna.

Koda v listingu 1.3 prikazuje zanko dogodkov, ki jo moramo v sistemu X11 implementirati sami. Na dogodke čakamo v neskončni zanki (vrstice 8 do 49), v kateri procesiramo vse dogodke, ki jih javi strežnik X in jih jemljemo iz vrste dogodkov (vrstica 9). V primeru, da je bil dogodek s tipkovnice (vrstica 10), pogledamo, če smo pritisnili katerega izmed gumbov “b”, “c”, “r”, “z” ali “m”, in ustrezno določimo vrednosti posameznih barvnih komponent (vrstice od 11 do 25). Nato določimo, katere barvne komponente bomo uporabili za določanje barve, alociramo ustrezno barvo iz barvne palete in nastavimo

barvo ospredja (vrstice od 26 do 28). Na koncu še izpišemo pritisnjeno tipko (vrstica 29). V primeru, da je bil dogodek z miške (vrstica 32), pa pogledamo, kateri gumb je bil pritisnjen. Če je bil pritisnjen levi miškin gumb (vrstica 33), si zapomnimo položaj v primeru, da je bil to prvi pritisk (števec pritiskov je enak 0, vrstice od 34 do 37), če je bil to drugi pritisk (števec pritiskov je takrat enak 1), pa narišemo pravokotnik v izbrani barvi (vrstice od 38 do 41). Na koncu še ustrezno popravimo števec pritiskov, ki hrani vrednosti 0 ali 1 (vrstica 42). V primeru pritisnjenega srednjega miškinega gumba (vrstica 44) pobrišemo okno (vrstica 45), v primeru pritisnjenega desnega miškinega gumba (vrstica 46) pa kličemo funkcijo, ki zapre okno (vrstica 47, koda v listingu 1.2).

Preostanejo še vključitev potrebnih zaglavnih datotek, deklaracija globalnih spremenljivk in funkcij ter glavna funkcija programa, kar prikazuje koda v listingu 1.4.

```
1  #include <X11/Xlib.h>
2  #include <X11/Xutil.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  Display *prikaz;
7  int zaslon;
8  Window okno;
9  Colormap barve;
10 XColor barva;
11 GC kontekst;
12 void inicializirajX();
13 void zapriX();
14 void zankaDogodkov();
15
16 int main(int argc, char *argv[]){
17   inicializirajX();
18   zankaDogodkov();
19 }
```

Listing 1.4 Glavni program.

Program lahko prevedemo z GNU prevajalnikom za programski jezik C. Pri prevajanju aplikacije moramo dinamično povezati v naš program tudi knjižnico libX11. Tako lahko to aplikacijo prevedemo z ukazom (predpostavimo, da se celoten program nahaja v datoteki EnostavnoOknoX11.c):

```
gcc -o EnostavnoOknoX11 EnostavnoOknoX11.c -lX11
```

Prevedeni program pa izvedemo z ukazom:

```
./EnostavnoOknoX11
```

Pri izvajanju programa pa moramo imeti nameščen okenski sistem X na strežniku in na odjemalcu. Operacijski sistemi Linux imajo ponavadi že nameščen okenski sistem X, pri Windows operacijskih sistemih pa je potrebno namestiti okolje *cygwin* za to, da lahko poganjamo aplikacije X11. Za operacijske sisteme Mac OS X pa obstaja projekt *XQuartz*, s pomočjo katerega je mogoče poganjati aplikacije X11.

6. Napišite aplikacijo (program) v programskem jeziku Java z uporabo knjižnice Swing. Aplikacija naj vsebuje okno velikosti 400×300 pikslov. Aplikacija naj čaka na dogodke, ki jih proži uporabnik. V primeru izbire tipke na tipkovnici naj se v terminalu izpiše tipka, ki jo je uporabnik pritisnil. V primeru, da je bila pritisnjena katera izmed tipk "b", "c", "r", "z" ali "m", pa naj se še nastavi barva ospredja okna (barva, s katero rišemo) na belo, črno, rdečo zeleno oziroma modro barvo. V primeru klika levega gumba na miški naj se v oknu izriše zapolnjen pravokotnik v barvi ospredja. Prvi klik naj samo določi položaj zgornjega levega oglišča pravokotnika, medtem ko drugi klik določi spodnje desno oglišče pravokotnika in povzroči izris. Ob kliku srednjega gumba na miški naj se vsebina okna izbriše, ob kliku desnega gumba na miški pa naj se okno (aplikacija) zapre.

   Rešitev: Naloga je podobna prejšnji nalogi, vendar je rešitev nekoliko krajša in lažja, ne omogoča pa izvajanja aplikacije na oddaljenem računalniku in prikaza na lokalnem računalniku. Najprej je potrebno narediti razred, ki bo razširjal razred *JFrame* in bo služil kot glavno okno aplikacije. Javanska koda, ki naredi potrebne korake, je prikazana v listingu 1.5.

```
1  import javax.swing.JFrame;
2  import javax.swing.SwingUtilities;
3  import java.awt.BorderLayout;
4  import java.awt.Dimension;
5
6  public class EnostavnoOkno extends JFrame{
7    private EnostavnaPlosca plosca;
8    public void inicializirajOkno(){
9      this.setSize(new Dimension(400, 300));
10     this.setTitle("Enostavno okno");
```

```
11      this.setLayout(new BorderLayout());
12      plosca = new EnostavnaPlosca();
13      this.add(plosca, BorderLayout.CENTER);
14    }
15
16    public static void main(String[] args){
17      SwingUtilities.invokeLater(new Runnable(){
18        public void run(){
19          EnostavnoOkno okno = new EnostavnoOkno();
20          okno.inicializirajOkno();
21          okno.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
                ;
22          okno.setVisible(true);
23        }
24      });
25    }
26  }
```

Listing 1.5 Okno aplikacije, ki razširja okvir (razred *JFrame*) skupaj z metodo *main*, ki ustvari in prikaže to okno.

Ta razred vsebuje metodo, s katero oknu določimo velikost, naslov, razvrščevalnik, kreiramo komponento, v kateri se bo izvajal izris, in to komponento dodamo oknu (vrstice od 8 do 14). Okvir ima že privzeto nastavljeno robno razvrščanje ("BorderLayout"), tako da je potrebno samo še določiti predel, v katerega bomo dodali komponento. Ta predel je v našem primeru kar center. Razred pa vsebuje tudi metodo *main* (vrstice od 16 do 21), s katero kreiramo okno aplikacije. Ker večina komponent Swing ni nitno varnih, kreiramo okno aplikacije znotraj niti za dogodke, s katerimi kličemo metodo za inicializacijo komponent vmesnika in okno tudi prikažemo.

Nato naredimo še razred za komponento, v kateri se bo izvajal izris. Ta komponenta bo tudi zaznavala dogodke, ki jih bo sprožil uporabnik. Koda za to komponento je prikazana v lisingu 1.6.

```
1  import javax.swing.JPanel;
2  import java.awt.Color;
3  import java.awt.Graphics;
4  import java.awt.event.MouseEvent;
5  import java.awt.event.MouseListener;
6  import java.awt.event.KeyEvent;
7  import java.awt.event.KeyListener;
8
9  class EnostavnaPlosca extends JPanel implements KeyListener
       ,MouseListener {
```

```
10    private Color barva=Color.black;
11    private int pritisk=0;
12    private int zacX, zacY;
13
14    public EnostavnaPlosca(){
15      this.addKeyListener(this);
16      this.addMouseListener(this);
17      this.setBackground(Color.white);
18      this.setFocusable(true);
19    }
20
21    private void risi(int x, int y, int d, int s){
22      Graphics g=this.getGraphics();
23      g.setColor(barva);
24      g.fillRect(x, y, d, s);
25    }
26
27    public void keyTyped(KeyEvent e) {
28      switch (e.getKeyChar()){
29        case 'b': barva=Color.white;break;
30        case 'c': barva=Color.black;break;
31        case 'r': barva=Color.red;break;
32        case 'z': barva=Color.green;break;
33        case 'm': barva=Color.blue;break;
34      }
35      System.out.println("Pritisnili ste tipko " + e.
          getKeyChar()+"!");
36    }
37
38    public void mouseClicked(MouseEvent e) {
39      if (e.getButton()==1){
40        if (pritisk==0){
41          zacX=e.getX();
42          zacY=e.getY();
43        }
44        if (pritisk==1){
45          if (zacX<e.getX() && zacY<e.getY()){
46            risi(zacX, zacY, e.getX()-zacX,e.getY()-zacY);
47          }
48        }
49        pritisk=(pritisk+1)%2;
50      }
51      else if (e.getButton()==2){
52        repaint();
```

```
53      }
54      else if (e.getButton()==3){
55        System.exit(0);
56      }
57    }
58
59    public void keyPressed(KeyEvent e) {     }
60    public void keyReleased(KeyEvent e) {     }
61    public void mousePressed(MouseEvent e) {     }
62    public void mouseReleased(MouseEvent e) {     }
63    public void mouseEntered(MouseEvent e) {     }
64    public void mouseExited(MouseEvent e) {     }
65  }
```

Listing 1.6 Razred, v katerem se bo izvajal izris v okno in ki razširja ploščo (razred *JPanel*) ter implementira poslušalce dogodkov s tipkovnice (razred *KeyListener*) in poslušalce dogodkov z miške (razred *MouseListener*).

Razred, ki smo ga napisali, razširja razred *JPanel* (plošča) in implementira poslušalca za dogodke s tipkovnice in miške (*KeyListener* in *MouseListener*), ki jih sproži uporabnik. Ker implementira ta dva vmesnika, moramo implementirati tudi vse abstraktne metode, ki so definirane v teh dveh vmesnikih. Najprej definiramo nekaj globalnih spremenljivk v tem razredu, nato definiramo konstruktor tega razreda (vrstice od 14 do 19), kjer dodamo poslušalca (vrstici 15 in 16), določimo barvo ozadja in določimo, da lahko sprejema fokus (vrstici 17 in 18). Nato napišemo metodo (vrstice od 21 do 25), s katero izrišemo kvadrat v ustrezni barvi (trenutno izbrana barva ospredja) in na ustreznem položaju. Nato implementiramo metodo (vrstice od 27 do 36), ki se odziva na natipkano črko z uporabo tipkovnice (dogodek natipkana črka se zgodi, ko tipko pritisnemo in spustimo, kar lahko tudi razdelimo na dva dogodka, za katera lahko realiziramo svoji metodi). S to metodo preberemo, kateri znak je uporabnik natipkal, in, če je potrebno, ustrezno nastavimo barvo ospredja, nato pa še implementiramo metodo (vrstice od 38 do 57), ki se odziva na klik gumbov na miški (dogodek klik gumbov na miški se zgodi takrat, ko pritisnemo in spustimo gumb na miški, kar lahko tudi razdelimo na dva dogodka, za katera lahko realiziramo svoji metodi). V primeru klika levega gumba (vrstice od 39 do 50) na miški si zapomnimo položaj klika v primeru prvega klika (vrednost števca je 0) oziroma kličemo metodo za izris pravokotnika v primeru drugega klika (vrednost števca je 1). V primeru klika srednjega gumba na miški (vrstice od 51 do 53) pobrišemo vsebino okna. V primeru klika desnega gumba na miški (vrstice od 54 do 56) pa končamo z aplikacijo. Na koncu pa so še definirane metode, ki jih moramo implementirati zaradi zahtev jezika (vrstice od 59 do 64). Naš razred

namreč implementira abstraktna vmesnika *KeyListner* in *MouseListener*, ki vsebujeta nekaj abstraktnih metod. Da lahko naredimo primerek takega razreda, pa morajo biti implementirane vse abstraktne metode, tudi tiste, ki jih ne potrebujemo.

Program prevedemo z naslednjim ukazom:

```
javac EnostavnoOkno.java
```

Prevedeni program pa izvedemo z ukazom:

```
java EnostavnoOkno
```

7. Program, predstavljen v nalogi 6, popravite tako, da se pri izrisu kvadrata za zgornje levo oglišče pravokotnika uporabi položaj, kjer se je zgodil pritisk gumba na miški, za spodnje desno oglišče pa se uporabi položaj, kjer se je gumb na miški izpustil.

   Rešitev: Za to je potrebno spremeniti ustrezne tri metode poslušalca za dogodke z miške (dogodek ob pritisku, dogodek ob izpustitvi in dogodek ob kliku), medtem ko ostanejo druge metode in definicije nespremenjene. Koda, ki vsebuje potrebne spremembe, je prikazana v listingu 1.7.

```
1  public void mouseClicked(MouseEvent e) {
2    if (e.getButton()==2){
3      repaint();
4    }
5    else if (e.getButton()==3){
6      System.exit(0);
7    }
8  }
9
10 public void mousePressed(MouseEvent e) {
11   if (e.getButton()==1){
12     zacX=e.getX();
13     zacY=e.getY();
14   }
15 }
16
17 public void mouseReleased(MouseEvent e) {
18   if (e.getButton()==1){
19     if (zacX<e.getX() && zacY<e.getY()){
20       risi(zacX, zacY, e.getX()-zacX,e.getY()-zacY);
```

```
21            }
22        }
23    }
```

Listing 1.7 Implementacija poslušalcev dogodkov z miške, ki omogočajo določitev pravokotnika s pritiskom in izpustitvijo gumba na miški (abstraktne metode, definirane v razredu *MouseListener*).

V tem primeru se aplikacija odziva na klik gumba na miški samo v primeru klika na srednji ali na desni gumb na miški, kjer je koda nespremenjena (vrstice od 1 do 8). Zgornje levo oglišče pravokotnika se določi z metodo, ki posluša za dogodke ob pritisku gumba na miški (vrstice od 10 do 15), medtem ko se položaj spodnjega desnega oglišča pravokotnika in izris zgodita pri metodi, ki posluša za dogodke ob izpustitvi gumba na miški (vrstice od 17 do 23).

8. Napišite aplikacijo (program) z uporabo grafične knjižnice *GTK+*. Aplikacija naj vsebuje okno velikosti 400×300 pikslov. Aplikacija naj čaka na dogodke, ki jih proži uporabnik. V primeru izbire tipke na tipkovnici naj se v terminalu izpiše natipkana tipka. V primeru, da je bila natipkana katera izmed tipk "b", "c", "r", "z" ali "m", pa naj se še nastavi barva ospredja okna na belo, črno, rdečo, zeleno oziroma modro barvo. V primeru klika levega gumba na miški naj se v oknu izriše zapolnjen pravokotnik v barvi ospredja. Prvi klik naj samo določi položaj zgornjega levega oglišča pravokotnika, medtem ko drugi klik določi spodnje desno oglišče pravokotnika in povzroči izris. Ob kliku srednjega gumba na miški naj se vsebina okna izbriše. Ob kliku desnega gumba na miški naj se okno (aplikacija) zapre.

   Rešitev: Pri izdelavi vmesnika z uporabo grafične knjižnice *GTK+* lahko uporabimo različne programske jezike, ki imajo implementirane klice za knjižnico *GTK+*. Pri tej implementaciji aplikacije bomo uporabili programski jezik C. Poleg tega pa je mogoča uporaba knjižnic *GTK+* različnih različic (1, 2 ali 3, ki pa imajo vse tudi še svoje podrazličice), ki se med seboj razlikujejo. Knjižnica *GTK+* različice 1 je že zelo stara in jo uporablja le še malo aplikacij. Knjižnica *GTK+* različice 2 je nekoliko novejša in obstaja kar nekaj aplikacij, ki uporabljajo to knjižnico, tako da bi bilo mogoče uporabiti tudi to različico knjižnice *GTK+*. Različne inačice Linuxa še vedno omogočajo poganjanje aplikacij, pisanih s starejšimi različicami knjižnice, vendar pa je knjižnica *GTK+* različice 3 v glavnem že prevzela primat in je tudi dovolj stabilna, tako da bomo to aplikacijo razvili z uporabo knjižnice *GTK+* različice 3. Ena izmed posebnosti te knjižnice pa je, da je za risanje zaželena uporaba knjižnice za delo z vektorsko grafiko *cairo* in je tudi zaradi tega nekoliko zahtevnejša za uporabo.

Pri izdelavi aplikacije bomo najprej napisali inicializacijsko funkcijo, ki bo zgradila okno aplikacije, omogočila posredovanje želenih uporabnikovih dogodkov aplikaciji, povezala dogodke in signale z ustreznimi odzivnimi funkcijami ter na koncu še prikazala okno aplikacije. Koda za inicializacijo je prikazana v listingu 1.8.

```
1  void inicializacijaGTK(){
2    okno = gtk_window_new(GTK_WINDOW_TOPLEVEL);
3    gtk_window_set_title(GTK_WINDOW(okno), "Aplikacija");
4    gtk_window_set_default_size(GTK_WINDOW(okno), 400, 300);
5    g_signal_connect(okno, "destroy", G_CALLBACK(
         gtk_main_quit), NULL);
6
7    mreza = gtk_grid_new();
8    gtk_container_add(GTK_CONTAINER(okno), mreza);
9
10   risalo = gtk_drawing_area_new();
11   gtk_widget_set_can_focus(risalo, TRUE);
12   gtk_widget_set_size_request(risalo, 400, 300);
13   gtk_grid_attach(GTK_GRID(mreza), risalo, 0, 0, 1, 1);
14
15   gtk_widget_set_events (risalo, gtk_widget_get_events(
         risalo) | GDK_BUTTON_PRESS_MASK | GDK_KEY_PRESS_MASK)
         ;
16   g_signal_connect(G_OBJECT(risalo), "configure_event",
       G_CALLBACK (konfiguriraj), NULL);
17   g_signal_connect(G_OBJECT(risalo), "draw", G_CALLBACK(
         narisi), NULL);
18   g_signal_connect(G_OBJECT(risalo), "button_press_event",
       G_CALLBACK(miska), NULL);
19   g_signal_connect(G_OBJECT(risalo), "key_press_event",
       G_CALLBACK(tipkovnica), NULL);
20   gtk_widget_show_all(okno);
21 }
```

Listing 1.8 Kreiranje okna, povezava signalov in odzivnih funkcij ter prikaz okna.

V kodi, prikazani v listingu 1.8 najprej naredimo okno aplikacije, mu določimo začetno velikost in povežemo signal za zaprtje okna s funkcijo, ki izstopi iz zanke dogodkov (vrstice od 2 do 5). Pri tem predpostavimo, da so spremenljivke, na katere se sklicujemo v funkciji in niso definirane v tej funkciji, definirane globalno. Nato naredimo mrežni vsebovalnik (vsebovalnik, ki lahko vsebuje več podob in pri katerem lahko vsaka podoba zavzame poljubno število vrstic in stolpcev) in ga dodamo oknu aplikacije (vrstici

7 in 8). *GTK+* vsebuje podobo, ki je namenjena risanju, to je podoba risalna površina. Kreiramo podobo risalna površina in ji določimo, da lahko sprejema fokus (vrstici 10 in 11), zato da lahko prestrezamo dogodke s tipkovnice. Nato ji določimo velikost in jo dodamo v vsebovalnik (vrstici 12 in 13). Risalna površina privzeto ne posluša za vse dogodke, ki jih želimo uporabiti v aplikaciji. Tako moramo risalni površini določiti vse tiste dogodke, za katere želimo, da jih dejansko prestreže (vrstica 15). V *GTK+* so nekateri dogodki že privzeto določeni, da se prestrežejo, nekatere dogodke pa moramo dodati sami. Tako smo dodali dogodke ob pritisku gumba na miški oziroma ob pritisku tipke. Nato povežemo dogodek ob konfiguraciji risalne površine, signal za izris, dogodek ob pritisku gumba na miški in dogodek ob pritisku gumba na tipkovnici z ustreznimi odzivnimi funkcijami (vrstice 16-19). Na koncu prikažemo okno aplikacije (vrstica 20).

Koda, ki jo uporabljamo pri konfiguraciji risalne površine, je prikazana v listingu 1.9.

```
1  static void zbrisiPovrsino (){
2    cairo_t *cr;
3    cr = cairo_create (povrsina);
4    cairo_set_source_rgb (cr, 1, 1, 1);
5    cairo_paint (cr);
6    cairo_destroy (cr);
7  }
8
9  static gint konfiguriraj (GtkWidget *widget, GdkEventButton
       *event){
10   if (povrsina){
11     cairo_surface_destroy (povrsina);
12   }
13   povrsina = gdk_window_create_similar_surface(
         gtk_widget_get_window(widget), CAIRO_CONTENT_COLOR,
         gtk_widget_get_allocated_width(widget),
         gtk_widget_get_allocated_height(widget));
14   zbrisiPovrsino();
15   return TRUE;
16 }
```

Listing 1.9 Brisanje vsebine risalne površine in konfiguracija risalne površine.

Prva funkcija, ki jo prikazuje koda v listingu 1.9, je namenjena izbrisu trenutne vsebine risalne površine. Risalna površina v *GTK+* omogoča risanje vektorskih slik z uporabo vektorske grafične knjižnice *cairo*. Da lahko rišemo v risalno površino, moramo najprej za to površino pridobiti kontekst *cairo*, ki

vsebuje trenutno stanje površine, v katero želimo risati (vrstici 2 in 3). Nato nastavimo barvo, ki jo bomo uporabili za risanje, tako, da določimo vrednost za rdečo, za zeleno in za modro barvno komponento na intervalu od 0 do 1 (vrstica 4). Nato s to barvo zapolnimo celotno izbrano območje v našem oknu (vrstica 5). Če ne izberemo območja, je to območje enako kar celotnemu prikazanemu oknu. Po izbrisu vsebine oziroma barvanju risalne površine lahko izbrišemo referenco na risalno površino, kar predstavlja kontekst *cairo* (vrstica 6). Druga funkcija pa je funkcija, ki je namenjena konfiguraciji risalne površine. Ta funkcija se kliče vedno, ko se sproži dogodek ob spremembi velikosti objekta (tudi ob kreiranju), torej risalne površine. V tej funkciji najprej pogledamo, če že obstaja risalna površina, in če obstaja, zbrišemo referenco nanjo (vrstice od 10 do 12). Nato površino ustvarimo, in sicer tako, da je kar se da združljiva s podanim oknom (vrstica 13). Površino nato še zbrišemo s klicem zgoraj opisane funkcije (vrstica 14). Funkcijo zaključimo z vrnitvijo vrednosti TRUE (vrstica 15), s čimer nakažemo, da ni potrebno nadaljnje procesiranje oziroma servisiranje tega dogodka.

Sedaj sledi koda funkcij za obravnavanje dogodkov z miške in risanje, prikazana v listingu 1.10.

```
1  static gint miska (GtkWidget *widget, GdkEventButton *event
       ){
2    cairo_t *cr;
3    static int x, y;
4    static int pritisk=0;
5
6    if (event->button == 1){
7      if (pritisk==0){
8        x=event->x;
9        y=event->y;
10     }
11     else{
12       cr = cairo_create(povrsina);
13       cairo_set_source_rgb(cr, r, z, m);
14       cairo_rectangle(cr, x, y, event->x-x, event->y-y);
15       cairo_fill(cr);
16       cairo_destroy(cr);
17       gtk_widget_queue_draw(widget);
18     }
19     pritisk=(pritisk+1)%2;
20   }
21   else if (event->button==2){
22     zbrisiPovrsino();
23     gtk_widget_queue_draw(widget);
```

```
24    }
25    else if (event->button==3){
26      if (povrsina)
27      cairo_surface_destroy (povrsina);
28      gtk_main_quit ();
29    }
30    return TRUE;
31  }
32
33  static gboolean narisi (GtkWidget *widget, cairo_t   *cr,
       gpointer   data){
34    cairo_set_source_surface (cr, povrsina, 0, 0);
35    cairo_paint (cr);
36    return FALSE;
37  }
```

Listing 1.10 Obravnava dogodkov, povzročenih z miško, in risanje v risalno površino.

Funkcija, ki jo prikazuje koda v listingu 1.10, obravnava dogodke z miške. V primeru, da se je zgodil pritisk levega gumba na miški (vrstice od 6 do 20), se v primeru prvega pritiska (vrednost števca pritiskov je enaka 0) shranijo koordinate tega pritiska (vrstice od 7 do 10), v primeru drugega pritiska (vrednost števca pritiskov je v tem primeru 1) pa se nastavi barva za risanje in nariše pravokotnik z začetnimi koordinatami prvega pritiska ter širino in višino, izračunano na podlagi razlike koordinat drugega in prvega pritiska (vrstice od 11 do 18). Pri tem lahko rišemo tudi z negativnimi vrednostmi za višino oziroma širino. V tem primeru se riše negativna stranica v obratno smer (v primeru negativne širine se bo pravokotnik risal od položaja prvega klika proti levi in ne proti desni). Po vsakem izrisu moramo izrisano tudi prikazati v oknu. To dosežemo tako, da sprožimo signal za risanje ("draw"), ki smo ga povezali z ustrezno odzivno funkcijo. Na ta način ročno sprožimo signal za ponoven izris okna ali le dela okna, ki ga je potrebno izrisati. Ob sprožitvi tega signala se bo klicala odzivna funkcija, ki je pridružena temu signalu. Nato še spremenimo stanje števca pritiskov levega miškinega gumba (vrstica 19). V primeru, da se je zgodil pritisk srednjega gumba na miški (vrstice od 21 do 24), se izbriše vsebina risalne površine, kjer se kliče funkcija, ki se uporablja tudi pri konfiguraciji risalne površine. V primeru, da se je zgodil pritisk desnega gumba na miški (vrstice od 25 do 29) se zbriše referenca na risalno površino in zaključi z zanko dogodkov. Druga funkcija (vrstice od 33 do 37) je odzivna funkcija, ki se kliče, kadar je sprožen signal za risanje. Pri knjižnici *GTK+* verzije 3 je parameter, ki se prenese v odzivno funkcijo, grafični kontekst za risanje in ne več dogodek, kot je to bilo pri

knjižnici *GTK+* verzije 2 (npr. GdkEventExpose ob zahtevku za izris dela površine okna). Funkcija povzroči, da se ponovno nariše del vsebine okna. Iz risalne površine se ustvari vzorec ("pattern"), ki se določi kot izvor ("source") v kontekstu *cairo* (vrstica 34), nato pa se ta vzorec nariše v trenutnem oknu (vrstica 35). Funkcijo zaključimo s tem, da vrnemo vrednost FALSE (vrstica 36), s čimer omogočimo nadaljnje obravnavanje dogodka.

Sledi še funkcija, prikazana v listingu 1.11, ki obravnava dogodke s tipkovnice.

```
1  static gint tipkovnica (GtkWidget *widget, GdkEventKey *
       event){
2    if (strlen(event->string)<=0)
3    return -1;
4    switch (event->string[0]){
5      case 'b': r=z=m=1; break;
6      case 'c': r=z=m=0; break;
7      case 'r': r=1;z=m=0; break;
8      case 'z': r=m=0;z=1; break;
9      case 'm': r=z=0;m=1; break;
10   }
11   printf("Pritisnili ste tipko: %c!\n", event->string[0]);
12   return 0;
13 }
```

Listing 1.11 Obravnavanje dogodkov s tipkovnice.

Koda, ki je prikazana v listingu 1.11, prebere, katera tipka na tipkovnici je bila pritisnjena. Najprej se preveri, če je bila pritisnjena katera izmed tipk, s katero določamo barvo, in glede na to ustrezno spremenimo vrednosti za rdečo, modro in zeleno barvno komponento (vrstice od 4 do 10). Na koncu še v terminal izpišemo pritisnjeno tipko (vrstica 11).

Sledijo še vključevanje potrebnih zaglavnih datotek, deklaracija globalnih spremenljivk ter glavna ("main") funkcija programa, kar prikazuje koda v listingu 1.12 .

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <gtk/gtk.h>
5
6 GtkWidget *okno;
7 GtkWidget *risalo;
8 GtkWidget *mreza;
9 static cairo_surface_t *povrsina = NULL;
```

```
10 int r=0,z=0,m=0;
11
12 int main(int argc, char *argv[]){
13   gtk_init(&argc, &argv);
14   inicializacijaGTK();
15   gtk_main();
16   return 0;
17 }
```

Listing 1.12 Inicializacija grafične knjižnice, klic funkcije za izgradnjo okna in vstop v zanko dogodkov.

V kodo listinga 1.12 najprej vključimo vse potrebne zaglavne datoteke (vrstice od 1 do 4), nato deklariramo globalne spremenljivke (vrstice od 6 do 10), katerim, kjer je potrebno, tudi določimo začetne vrednosti. Na koncu programa (z vsemi funkcijami) se nahaja še funkcija *main* (vrstice od 12 do 17), ki se kliče ob zagonu programa. V tej funkciji se najprej inicializira knjižnica *GTK+* (vrstica 13). Inicializaciji knjižnice *GTK+* sledi klic funkcije, ki kreira okno aplikacije in poveže signale ter dogodke z odzivnimi funkcijami (vrstica 14), nato pa vstopimo v zanko dogodkov (vrstica 15). Program se konča tako, da po izstopu iz zanke dogodkov vrnemo vrednost nič (vrstica 16), kar predstavlja normalno končanje programa.

Celoten program lahko prevedemo z uporabo GNU prevajalnika za programski jezik C. Pri tem moramo paziti na vrstni red pisanja funkcij v datoteko s programom, saj prevajalnik zahteva, da so vse funkcije deklarirane, preden so vidne v kodi (funkcija *inicializacijaGTK* mora biti v datoteki deklarirana za vsemi odzivnimi funkcijami, ki jih v tej funkciji povezujemo s signali, torej za funkcijami: *konfiguriraj*, *narisi*, *miska* in *tipkovnica*), sicer pride do napake pri prevajanju. Celotno datoteko, za katero predpostavimo, da je v datoteki risiGTK.c, lahko prevedemo z naslednjim ukazom:

```
gcc -o risiGTK risiGTK.c $(pkg-config --libs --cflags gtk+-3.0)
```

S tem ukazom bomo izvorno kodo, ki se nahaja v datoteki risiGTK.c, prevedli v izvršljivi program *risiGTK*, ukaz pkg-config v zgornjem ukazu za prevajanje pa za knjižnico *GTK+* verzije 3 nastavi vsa stikala, ki so potrebna za prevajanje programov. Preveden program izvedemo z ukazom:

```
./risiGTK
```

Ekvivalenten uporabniški vmesnik lahko seveda ustvarimo tudi s programskim jezikom Python. Program, napisan v Pythonu, prikazuje koda v listingu 1.13. Opis kode je zelo podoben opisu programa, ki je napisan v programskem jeziku C, zato je izpuščen.

```
1  import gi;
2  gi.require_version('Gtk', '3.0')
3  from gi.repository import Gtk;
4  from gi.repository import Gdk;
5  import cairo;
6  povrsina=None;
7  x=0;
8  y=0;
9
10 class MojeOkno(Gtk.Window):
11     pritisk=0;
12     r=z=m=0;
13     def __init__(self):
14         Gtk.Window.__init__(self);
15         self.set_title("Aplikacija");
16         self.set_default_size(400, 300);
17         mreza = Gtk.Grid();
18         self.add(mreza);
19         risalo = Gtk.DrawingArea();
20         risalo.set_can_focus(True);
21         risalo.set_size_request(400, 300);
22         mreza.attach(risalo, 0, 0, 1, 1);
23         risalo.set_events(risalo.get_events() | Gdk.
               EventMask.BUTTON_PRESS_MASK | Gdk.EventMask.
               KEY_PRESS_MASK);
24         risalo.connect("configure_event", self.konfiguriraj
               );
25         risalo.connect("draw", self.narisi);
26         risalo.connect("button_press_event", self.miska);
27         risalo.connect("key_press_event", self.tipkovnica);
28
29     def zbrisiPovrsino(self):
30         global povrsina;
31         cr = cairo.Context(povrsina)
32         cr.set_source_rgb(1,1,1)
33         cr.paint()
34         del cr
35
36     def konfiguriraj(self, *args):
37         global povrsina;
```

```
38          if povrsina is not None:
39              del povrsina;
40              povrsina = None;
41          w = self.get_allocated_width();
42          h = self.get_allocated_height();
43          povrsina = self.get_window().create_similar_surface
                (cairo.CONTENT_COLOR,w, h);
44          self.zbrisiPovrsino();
45
46      def narisi(self, *args):
47          global povrsina;
48          args[1].set_source_surface(povrsina, 0, 0);
49          args[1].paint()
50          return False;
51
52      def miska(self, *args):
53          global x, y, povrsina;
54          if (args[1].button==1):
55              if (self.pritisk==0):
56                  x=args[1].x;
57                  y=args[1].y;
58              else:
59                  cr = cairo.Context(povrsina);
60                  cr.set_source_rgb(self.r, self.z, self.m);
61                  cr.rectangle(x,y,args[1].x-x,args[1].y-y);
62                  cr.fill();
63                  del cr;
64                  self.queue_draw();
65              self.pritisk=(self.pritisk+1)%2;
66          elif (args[1].button==2):
67              self.zbrisiPovrsino();
68              self.queue_draw();
69          elif (args[1].button==3):
70              if povrsina is not None:
71                  del povrsina;
72              Gtk.main_quit();
73
74      def tipkovnica(self, *args):
75          if (len(args[1].string)<=0):
76              return;
77          if args[1].string == "b":
78              self.r=self.z=self.m=1;
79          elif args[1].string == "c":
80              self.r=self.z=self.m=0;
```

```
81            elif args[1].string == "r":
82                self.r=1;self.z=self.m=0;
83            elif args[1].string == "z":
84                self.r=self.m=0;self.z=1;
85            elif args[1].string == "m":
86                self.r=self.z=0;self.m=1;
87            print ("Pritisnili ste tipko " + args[1].string +"!
                  ");
88
89    okno = MojeOkno();
90    okno.connect("destroy", Gtk.main_quit)
91    okno.show_all();
92    Gtk.main()
```

Listing 1.13 Grafični uporabniški vmesnik, ki uporablja knjižnico *GTK+*, napisan v programskem jeziku Python.

Pod predpostavko, da smo zgornjo kodo zapisali v datoteko z imenom risiGTK.py, lahko ta program izvedemo z ukazom:

```
python risiGTK.py
```

9. Napišite spletno aplikacijo (program) za enostavno risanje z uporabo tehnologij, ki se prikazujejo oziroma izvajajo v brskalniku. Aplikacija naj vsebuje risalno okno velikosti 400×300 pikslov. Aplikacija naj čaka na dogodke, ki jih proži uporabnik. V primeru izbire tipke na tipkovnici naj se na zaslonu izpiše natipkana tipka. V primeru, da je bila natipkana katera izmed tipk "b", "c", "r", "z" ali "m", pa naj se še nastavi barva ospredja okna na belo, črno, rdečo, zeleno oziroma modro barvo. V primeru klika levega gumba na miški naj se v oknu izriše zapolnjen pravokotnik v barvi ospredja. Prvi klik naj samo določi položaj zgornjega levega oglišča pravokotnika, medtem ko drugi klik določi spodnje desno oglišče pravokotnika in povzroči izris. Ob kliku srednjega gumba na miški naj se vsebina okna izbriše, ob kliku desnega gumba na miški pa naj se na zaslonu izpišejo navodila za končanje aplikacije.

   Rešitev: Pri izdelavi aplikacije bomo uporabili označevalni jezik HTML5, ki je namenjen določanju strukture dokumentov HTML, jezik za prekrivne sloge CSS, ki je namenjen določanju izgleda aplikacije, ter jezik JavaScript, ki se predvsem uporablja za dodajanje dinamičnosti spletnim dokumentom na strani odjemalca (brskalnika). V jeziku HTML bomo določili strukturo dokumenta HTML, z jezikom za prekrivne sloge bomo določili izgled elementov, definiranih z jezikom HTML, z jezikom JavaScript pa bomo uporabniku omogočili risanje.

Koda v listingu 1.14 prikazuje kodo, napisano v HTML in CSS, s pomočjo katere zgradimo strukturo spletne aplikacije. Ker se za prikaz spletnih aplikacij uporabljajo brskalniki, je za strukturo uporabniškega vmesnika potrebne relativno malo kode.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Risanje likov na platno</title>
5      <meta charset="utf-8">
6      <style>
7        #mojePlatno {
8          border: 2px solid #000000;
9        }
10     </style>
11   </head>
12
13   <body>
14     <canvas id="mojePlatno" width="400" height="300"
         tabindex="1">
15       Brskalnik ne podpira HTML5 kanvasa
16     </canvas>
17     <div id="obvestilo"> </div>
18   </body>
19 </html>
```

Listing 1.14 Grafični uporabniški vmesnik, napisan v jeziku HTML in oblikovan z jezikom CSS.

V kodi v listingu 1.14 najprej definiramo, da je to dokument HTML verzije 5 (vrstica 1). Nato sledi definicija strukture dokumenta HTML (vrstice 2 do 19). Struktura dokumenta HTML je deklarirana znotraj značke *<html>* in je sestavljena iz dveh delov: glave dokumenta (vrstice 3 do 11) in telesa dokumenta (vrstice 13 do 18). Glava dokumenta je vsebovalnik za metapodatke, kot so na primer naslov, nabor znakov, slogi za dokument, razne skripte, ter niso prikazani. V listingu 1.14 je tako definiran naslov dokumenta HTML oziroma strani (vrstica 4), ki se prikaže v zavihku, v katerem je stran prikazana, in v naslovni vrstici brskalnika, če je zavihek aktiven. Nato so definirani uporabljen nabor znakov (vrstica 5) ter slogi za gradnike strani (vrstice 6 do 10), kjer smo samo definirali slog obrobe okoli risalne površine. Telo dokumenta HTML služi definiciji telesa oziroma strukture dokumenta HTML in hrani vsebino dokumenta HTML oziroma strani, kot so na primer besedilo, povezave, slike, table in drugi elementi HTML, vsebuje pa lahko tudi skripte. V telesu strani v listingu 1.14 je najprej definicija risalne površina *<canvas>*

(vrstice 14 do 16). Pri definiciji risalne površine smo dodali parametre za identifikacijo (*id*), s pomočjo katere bomo v kodi v jeziku JavaScript dostopali do tega elementa, višino (*width*) in širino (*height*) risalne površine ter za fokusiranje (*tabindex*), ki omogoča preusmeritev fokusa na risalno površino. V primeru, da brskalnik ne podpira risalne površine, se uporabniku namesto risalne površine prikaže obvestilo (vrstica 15). Na koncu (vrstica 17) sledi še definicija elementa za obvestila (<*div*>), ki je namenjen izpisu pritisnjenih tipk na tipkovnici ter prostoru za obvestilo uporabniku.

Da bo stran odgovarjala na uporabniške akcije, je potrebno še registrirati rokovalnike dogodkov, kar prikazuje koda JavaScript v listingu 1.15. Ta koda se doda v telo dokumenta HTML (listing 1.14) takoj za definicijo risalne površine.

```
1  <script type="text/javascript">
2    var cnv=document.getElementById("mojePlatno");
3    var ctx=cnv.getContext('2d');
4    var obvestilo=document.getElementById("obvestilo");
5    var barva="black", pritisk=0, zacX, zacY;
6    cnv.addEventListener("mousedown", gumbPritisnjen,
       false);
7    cnv.addEventListener("keydown", tipkaPritisnjena,
       false);
8    cnv.addEventListener("contextmenu", function(event){
       event.preventDefault();}, false);
9  </script>
```

Listing 1.15 Koda v jeziku JavaScript (znotraj značk *script* jezika HTML) za inicializacijo spremenljivk ter za registracijo rokovalnikov dogodkov.

V kodi, prikazani v listingu 1.14, se najprej pridobi referenca na risalno površino (vrstica 2). Ta referenca je potrebna zato, da lahko za risalno površino registriramo želene rokovalnike dogodkov. Nato se pridobi še referenca na grafični kontekst risalne površine (vrstica 3), v katerem se bo izvajalo risanje, ter na element za izpis pritisnjenih črk oziroma obvestila (vrstica 4). Nato se definirajo globalne spremenljivke (vrstica 5) ter rokovalnik dogodkov za pritisk miškinega gumba (vrstica 6) in za pritisk tipke na tipkovnici (vrstica 7). Na koncu moramo dodati še rokovalnik dogodkov za dogodek "kontekstni meni" (vrstica 8). To je dogodek, ki se na elementih zgodi ob kliku desnega miškinega gumba. Na risalni površini HTML (<*canvas*>) se ob tem dogodku privzeto prikaže kotekstni oziroma pojavni meni. V primeru te aplikacije je to nezaželeno obnašanje. Prikaz pojavnega menija ob pritisku desnega miškinega gumba preprečimo tako, da definiramo lasten rokovalnik dogodkov, v katerem samo določimo, da se privzeta akcija (prikaz pojavnega menija)

ne izvede. Ob pritisku desnega miškinega gumba se tako najprej izvede ta rokovalnik dogodkov, prikaz pojavnega menija pa se dejansko prekliče.

Sledi še definicija rokovalnikov dogodkov, ki je prikazana v listingu 1.16. Koda v tem listingu se doda na primer v glavo dokumenta HTML za značko *<style>*.

```
<script type="text/javascript">
  function gumbPritisnjen(event){
    switch (event.button){
      case 0:
      if (pritisk==0){
        zacX=event.offsetX;
        zacY=event.offsetY;
        pritisk=pritisk+1;
      }
      else{
        ctx.fillStyle=barva;
        ctx.fillRect(zacX, zacY, event.offsetX-zacX,
            event.offsetY-zacY);
        pritisk=0;
      }
      break;
      case 1:
      ctx.clearRect(0, 0, cnv.width, cnv.height);
      break;
      case 2:
      obvestilo.innerHTML="<h2>Za konec aplikacije
          morate zapreti zavihek.</h2>";
      break;
    }
  }

  function tipkaPritisnjena(event){
    obvestilo.innerHTML="<h2>"+event.key+"</h2>";
    switch (event.key){
      case 'b': barva="white";break;
      case 'c': barva="black";break;
      case 'r': barva="red";break;
      case 'z': barva="green";break;
      case 'm': barva="blue";break;
    }
  }
</script>
```

Listing 1.16 Rokovalniki dogodkov v jeziku JavaScript (znotraj značk *script* jezika HTML).

Prvi rokovalnik dogodkov v listingu 1.16 obravnava dogodke ob pritisku miškinega gumba (vrstice 2 do 23). V primeru, da je bil pritisnjen levi miškin gumb (vrstice 4 do 15), se v primeru prvega pritiska (vrednost števca pritiskov je 0) shrani položaj, kjer se je dogodek zgodil, in popravi števec pritiskov (vrstice 5 do 9), v primeru drugega pritiska (vrednost števca pritiskov je 1) pa se nastavi barva za izris, izriše kvadrat v želeni velikosti in nastavi števec pritiskov (vrstice 10 do 15). V primeru, da je bil pritisnjen srednji miškin gumb (vrstice 16 do 18), se pobriše celotna risalna površina, v primeru pritiska desnega miškinega gumba (vrstice 19 do 21) pa se v element za obvestilo zapiše navodilo uporabniku, kako naj konča aplikacijo. To obvestilo je nadomestilo za funkcionalnost zapiranja aplikacije oziroma zapiranja okna, saj neposredno v aplikaciji okna oziroma zavihka ne moremo zapreti. Razlog za to je, da brskalniki ne dovoljujejo skriptam, da bi same zapirale okna, ki jih niso odprle. To mora narediti uporabnik sam.

Drugi rokovalnik dogodkov v listingu 1.16 posluša za dogodke s tipkovnice (vrstice 25 do 34). Ko se zgodi dogodek s tipkovnice na risalni površini, se najprej v element za sporočila izpiše pritisnjena tipka (vrstica 26), nato pa se v primeru tipk “b”, “c”, “r”, “z” oziroma “m” ustrezno nastavi še barva za izris. Da lahko risalna površina zazna dogodke s tipkovnice, mora biti v fokusu, kar dosežemo tako, da kliknemo z enim izmed miškinih gumbov na to površino. Pri tem mora biti obvezno nastavljena lastnost za fokusiranje na risalni površini, sicer fokus ni mogoč (listing 1.14, vrstica 14).

2 Uporabnost

10. Kaj je uporabnost? Naštejte dimenzije uporabnosti.

    Rešitev: Uporabnost podaja, kako dobro lahko uporabniki uporabljajo funkcionalnosti sistema. Uporabnost sestavljajo različne dimenzije: naučljivost, učinkovitost in varnost, poleg tega pa sta pri uporabniških vmesnikih pomembni tudi dimenziji, ki povesta, ali je orodje prijetno za uporabo. Ti dimenziji sta estetika in ergonomija.

11. Razložite, zakaj relativno kratke, a nepredvidljive zamude povzročajo večje probleme v uporabnosti kot dolge zamude, ki jih lahko uporabnik predvidi.

    Rešitev: Če uporabnik ve, da bo določena naloga trajala dolgo časa, lahko predvidi ustrezen časovni termin zanjo. Na primer, prenos podatkov za nadgradnjo programske opreme lahko vzame precej časa, zato se lahko izvede ponoči. Če je prenos nepredvidljiv, je to zelo moteče. To je tipično večji problem, če naloga vzame več časa, kot je bilo predvideno, lahko pa je tudi problem, če se daljša naloga zaključi prej, kot je uporabnik načrtoval.

12. Zakaj je pri razvoju aplikacij uporabniški vmesnik pomemben?

    Rešitev: Uporabniški vmesnik ponavadi predstavlja prvi stik uporabnika z aplikacijo. Poleg tega je uporaba funkcionalnosti, ki jih nudi aplikacija, odvisna od uporabniškega vmesnika. Če je ta slab, potem se tudi funkcionalnosti ne bodo mogle uporabljati. Če pa je dober, bo omogočal enostavno in hitro delo, s tem pa bo tudi omogočal boljšo sprejetost aplikacij.

13. Zakaj je težko načrtati primeren uporabniški vmesnik?

    Rešitev: Poglavitna težava je ta, da je potrebno načrtati uporabniški vmesnik za uporabnika, ki ni načrtovalec oziroma ne ve, kako uporabniški vmesnik deluje. Pri tem je potrebno v razvoj vključiti tudi uporabnike in z njimi komunicirati. Ta komunikacija je pogosto težavna zaradi različnih predstav

in računalniškega predznanja načrtovalca in uporabnika.

14. Uporabnost je ena izmed lastnosti sistema. Naštejte še nekaj lastnosti, na katere mora biti razvijalec programske opreme pozoren. Ali je mogoče optimizirati vse te lastnosti?

    Rešitev: Poleg uporabnosti je potrebno pri razvoju programske opreme upoštevati še funkcionalnost, zmogljivost, zanesljivost, zaščito, ceno in standarde. Pri razvoju programske opreme smo omejeni tako s časom, kot tudi s ceno. Ker optimiziranje posamezne lastnosti pomeni večji strošek in več časa, potrebnega za razvoj, je ponavadi potrebno narediti kompromis pri optimiziranju lastnosti.

3 Uporabniško usmerjeno načrtovanje

15. Naštejte korake slapovnega modela razvoja programske opreme.

    Rešitev: Koraki slapovnega modela so:

    a) korak postavljanja zahtev,
    b) korak načrtovanja,
    c) korak izvedbe,
    d) korak integracije/vpeljave,
    e) korak končnega testiranja,
    f) korak izdaje.

16. Razložite, kako poteka razvoj programske opreme z uporabo slapovnega modela.

    Rešitev: Pri razvoju programske opreme z uporabo slapovnega modela si koraki (postavljanje zahtev, načrtovanje, izvedba, integracija, testiranje, izdaja) sledijo zaporedno od začetka do konca, brez vračanja nazaj. Pri razvoju programske opreme je to lahko problematično, zato so nastale različne modifikacije slapovnega modela. Tako imamo lahko pri modificiranem slapovnem modelu povratne zanke v predhodni korak, kar poveča njegovo uporabnost pri razvoju programske opreme.

17. V katere korake slapovnega modela razvoja programske opreme so vključeni uporabniki? Zakaj lahko to predstavlja problem pri razvoju programske opreme?

    Rešitev: Uporabniki so vključeni v koraku postavljanja zahtev (angl. "requirements") in v koraku končnega testiranja ali izdaje (angl. "acceptance" ali angl. "release").
    Vključitev uporabnikov v prvo fazo in nato šele v končne faze je lahko problematična, saj se lahko zgodi, da uporabnik z izdelkom ne bo zadovoljen. Tako je mogoče, da bo veliko opravljenega dela (načrtovanje, izvedba, integracija) potrebno ponovno opraviti. Uporabnikove želje so lahko take, da jih

samo s kozmetičnimi popravki ni mogoče upoštevati v obstoječem izdelku, kar pomeni, da bo potrebno ponovno izvesti vse vmesne korake in del že opravljenega dela zavreči.

18. Naštejte korake iterativnega načrtovanja uporabniških vmesnikov.

    Rešitev: Koraki iterativnega načrtovanja uporabniških vmesnikov so:

    - načrtuj,
    - implementiraj,
    - vrednoti.

    Ti trije koraki so podobni slapovnemu modelu, pri iterativnem modelu pa te korake še ciklično ponavljamo.

19. Razložite, kateri bi bil napačen pristop pri iterativnem načrtovanju in zakaj.

    Rešitev: Napačen pristop bi bil, če bi vsaka iteracija vključevala tudi izdajo programske opreme, saj bi to pomenilo dražje načrtovanje. Poleg tega ni vsaka iteracija primerna za izdajo, saj lahko pride do pomanjkljivosti, ki bi lahko negativno vplivale na uporabnike programske opreme in s tem tudi na sprejetost programske opreme.

20. Kateri so koraki spiralnega modela načrtovanja uporabniških vmesnikov? V čem je razlika med spiralnim modelom in iterativnim modelom?

    Rešitev: Koraki spiralnega modela so:

    - načrtuj,
    - implementiraj,
    - vrednoti.

    Spiralni model je podoben iterativnemu modelu, saj tako kot iterativni tudi spiralni model predvideva ponavljanje teh treh korakov. Glavna razlika med obema modeloma je ta, da pri začetnih iteracijah, ko je še velika verjetnost, da bo šlo kaj narobe, spiralni model uporablja poceni prototipe. To zagotavlja, da tudi če so potrebne spremembe ali celo nov začetek, to ni tako zelo drago.

21. Kaj je mišljeno pod pojmom uporabniško usmerjeno načrtovanje?

    Rešitev: Pri uporabniško usmerjenem načrtovanju imamo v mislih to, da se osredotočamo na uporabnika in ne na, recimo, funkcionalnosti. Tako je fokus

usmerjen na potrebe ljudi in ne na sistem oziroma tehnologije. Uporabniško usmerjeno načrtovanje zahteva pogovor z ljudmi in razumevanje njihovih potreb.

22. Za uporabniško usmerjeno načrtovanje so tipične tri značilnosti. Katere so te tri značilnosti?

    Rešitev: Značilnosti uporabniško usmerjenega načrtovanja so:

    - zgodaj se je potrebno osredotočiti na uporabnike in njihove naloge; potrebno je analizirati uporabnike in njihove naloge;
    - potrebno je uporabljati iterativno načrtovanje;
    - potrebno je stalno vrednotenje.

23. Zamislite si, da intervjuvate načrtovalca uporabniškega vmesnika, za katerega predpostavljate, da ve, kaj je uporabniško usmerjeno načrtovanje. Katera specifična vprašanja bi mu postavili, da bi potrdili svojo predpostavko, in kakšni so načrtovalčevi pravilni odgovori?

    Rešitev: Preveriti moramo, če pozna značilnosti uporabniško usmerjenega načrtovanja uporabniških vmesnikov:

    - Kaj naredite najprej?
      Najprej se osredotočim na uporabnike in njihove naloge. Analiziram uporabnike in njihove naloge.
    - Kako izgleda proces načrtovanja uporabniškega vmesnika?
      Uporabljam iterativno načrtovanje. Pri tem načrtovanju iterativno ponavljam korake načrtovanja prototipa, implementacije prototipa in vrednotenja prototipa.
    - Ali so uporabniki vključeni v proces načrtovanja in če so, kako so vključeni v ta proces?
      Da, uporabniki so vključeni v proces načrtovanja. V vsaki iteraciji načrtovanja moramo izvesti vrednotenje prototipa, v katero moramo vključiti tudi uporabnike.

24. Kje je ovrednotenje umeščeno v cikel uporabniško usmerjenega načrtovanja?

    Rešitev: Značilnost uporabniško usmerjenega načrtovanja je tudi ta, da se uporablja iterativno načrtovanje. Tako je v uporabniško usmerjenem načrtovanju vrednotenje tretji korak v vsaki iteraciji načrtovanja.

25. Kaj je namen vrednotenja? Kdo bi moral uporabljati vrednotenje?

Rešitev: Namen je razumeti potrebe uporabnika; preveriti sistem oziroma zamisli, če zadovoljujejo pričakovanja (delajo tisto, kar uporabnik pričakuje). Vrednotenje bi morali uporabljati načrtovalci, specialisti za uporabniške vmesnike oziroma tisti, ki izvajajo teste.

4 Sposobnosti človeka

26. Naštejete razlike med kratkotrajnim (delovnim) in dolgotrajnim (dolgoročnim) spominom.

    Rešitev: Dolgotrajni oziroma dolgoročni spomin je zelo asociativen in obstojen (persistenten). Nekatere študije celo nakazujejo, da ni nič, kar si je človek shranil v dolgotrajni spomin, pozabljeno s časom, le poti do shranjene informacije izginejo. Kratkotrajni spomin pa ni obstojen in je tudi bolj podvržen interferenci.

27. S katerimi človekovimi procesorji in s katerim človekovim spominom je povezana človekova pozornost med obdelavo informacij?

    Rešitev: Človekova pozornost je povezana s:

    - procesorjem za zaznavanje,
    - procesorjem za razumevanje,
    - motoričnimi procesorji,
    - delovnim spominom.

28. Kaj je zlivanje med zaznavanjem? Kdaj pride do zlivanja in kakšne so njegove posledice?

    Rešitev: Zlivanje med zaznavanjem je dogodek, do katerega pride, ko prispeta dva dražljaja v zelo kratkem časovnem obdobju. Procesor za zaznavanje obdela ta dva dogodka znotraj enega cikla, zaradi česar se zdita dražljaja zlita. Do zlivanja pride, ko se zgodita dva dražljaja znotraj cikla procesorja za zaznavanje, ki je $T_p \approx 100$ ms. Posledica zlivanja je, da je $1/T_p = 10$ slik na sekundo še dovolj za sprejem gibajoče se slike. Ker je to povprečen čas velja, da je v splošnem za sprejem gibajoče se slike potrebno minimalno 16 slik na sekundo. Druga posledica pa je, da se računalniški odziv, ki je krajši kot T_p, zdi takojšen, prav tako pa zlivanje vpliva na dojemanje kavzalnosti.

29. Ko govorimo o človekovi obdelavi informacij se pojavlja pojem "kos" (angl. "chunk"). Kaj so "kosi"? Od česa je odvisna gradnja "kosov"?

    Rešitev: "Kos" je enota zaznavanja ali spomina. Kos je definiran oziroma prepoznan simbol in predstavlja aktivacijo izkušnje iz preteklosti. Gradnja kosov je odvisna od predstavitve (števila znakov) in od že prej znanega (nekatere kose je lažje prepoznati kot druge). Najprimernejša dolžina kosov za kodiranje nepovezanih znakov je 3-4 znake (telefonske številke, na primer, so tipično razdeljene v skupine treh oziroma štirih znakov).

30. Kaj je naloga procesorja za zaznavanje med človekovo obdelavo informacij? S katerimi procesorji ter spomini sodeluje in kako?

    Rešitev: Naloga procesorja za zaznavanje med človekovo obdelavo informacij je, da iz simbolov, ki so lahko vizualni (kot so na primer črke, besede, ikone) ali akustični (na primer fonemi), in "kosov" gradi nove kose.
    Sodeluje s senzornim spominom, dolgotrajnim spominom in procesorjem za razumevanje, in sicer:

    - od senzornega spomina sprejema simbole;
    - od dolgotrajnega spomina sprejema simbole in kose;
    - simbole in kose posreduje procesorju za razumevanje.

31. Kaj je naloga procesorja za razumevanje med človekovo obdelavo informacij? S katerimi procesorji ter spomini sodeluje in kako?

    Rešitev: Naloga procesorja za razumevanje med človekovo obdelavo informacij je, da primerja dražljaje, ki jih dobi od procesorja za zaznavanje, ki so lahko simboli in "kosi", ter da izbere odziv ali odločitev. Odziv oziroma odločitev pa lahko izbere na osnovi izkušenj, na osnovi pravil ali na osnovi znanja.
    Sodeluje s procesorjem za zaznavanje, z motoričnim procesorjem in delovnim spominom, in sicer:

    - od procesorja za zaznavanje sprejema simbole in kose;
    - motoričnemu procesorju posreduje odločitve;
    - iz delovnega spomina črpa simbole in kose;
    - v delovni spomin shranjuje simbole in kose.

32. Na katere človekove spomine in procesorje je usmerjena človekova pozornost med obdelavo informacij? Kaj so lastnosti tistega spomina, kjer se nahaja človekova zavest?

Rešitev: Pozornost je usmerjena na: procesor za zaznavanje, procesor za razumevanje, na motorični procesor in na delovni spomin.
Lastnosti delovnega spomina so:

- majhna kapaciteta: $\sim 4 \pm 1$ "kosov" (dolgo je veljalo, da je kapaciteta $\sim 5 \pm 2$, Cowan pa je v članku [2] predlagal in tudi utemeljil nekoliko manjšo kapaciteto);
- hitro pozabljanje: tipično približno od 10 do 15 sekund, včasih do ene minute;
- vztrajno ponavljanje izniči pozabljanje, a zahteva pozornost;
- interferenca konfliktnih kosov povzroči hitrejše pozabljanje.

33. Pri motoričnem procesiranju obstajata dve vrsti procesiranja. Kateri sta ti dve vrsti in kako se razlikujeta?

    Rešitev: Pri motoričnem procesiranju imamo lahko procesiranje z odprto zanko ter procesiranje z zaprto zanko. Pri procesiranju z odprto zanko motorični procesor deluje samostojno, brez povratne informacije. Ker ni potrebno čakati na povratno informacijo, je tudi čas cikla bistveno krajši in je enak ciklu motoričnega procesorja, $T_m \sim 70$ ms. Pri nadzoru z zaprto zanko pa se premik mišic oziroma rezultat premika mišic zazna in se primerja z želenim rezultatom. V tem primeru je čas cikla odvisen ne samo od motoričnega procesorja, temveč tudi od procesorja za zaznavanje in procesorja za razumevanje, $T_p + T_c + T_m \sim 240$ ms.

34. Kaj od naštetega se nanaša na delovni spomin? (izberite le en odgovor):

    a) zlivanje med zaznavanjem,

    b) motorični procesorji,

    c) gradnja "kosov".

    Rešitev: Pravilen je odgovor c), gradnja "kosov".

35. Kaj od naštetega se nanaša na procesor za razumevanje? (izberite le en odgovor):

    a) delovni spomin,

    b) hitro pozabljanje,

    c) gradnja "kosov",

    d) izbiranje odziva oziroma odločitve.

Rešitev: Pravilen je odgovor d), izbiranje odziva oziroma odločitve.

36. V čem je razlika med prepoznavanjem in pomnjenjem? Kaj je lažje, prepoznavanje ali pomnjenje? Podajte po en primer uporabniškega vmesnika, ki zahteva pomnjenje oziroma prepoznavanje.

    Rešitev: Prepoznavanje se nanaša na človekovo zmožnost priklica določene informacije na podlagi vizualnega ali kakšnega drugega dražljaja oziroma namiga. Pomnjenje pa se nanaša na priklic določene informacije iz spomina brez kakršnega koli zunanjega dražljaja ali namiga. Bistveno lažje je prepoznavanje. Uporabniški vmesniki, ki zahtevajo pomnjenje, so vmesniki brez vizualnih namigov, na primer ukazna vrstica, kjer se mora uporabnik spomniti ukaza in njegove sintakse zato, da lahko izvede neko akcijo. Vmesniki, ki zahtevajo prepoznavanje, pa so na primer vmesniki, ki vsebujejo menije ali sezname.

37. Napišite definicijo metafore.

    Rešitev: Metafora je prenašanje lastnosti objektov ali akcij realnega sveta na neke druge objekte z namenom sugeriranja podobnosti ali analogije med njimi.

38. Naštejte težave, na katere lahko naletimo, če želimo uporabljati metafore.

    Rešitev: Težave, na katere lahko naletimo pri uporabi metafor, so:

    - težko je najti metafore,
    - metafore niso vedno razumljive,
    - metafore včasih varajo,
    - metafore so omejene.

39. Pri uporabi metafor lahko naletimo na težave. Ena izmed težav je tudi, da metafore včasih varajo. Kaj pomeni, da "metafore včasih varajo"?

    Rešitev: Z metaforo poskušamo prenašati lastnosti objektov realnega sveta na druge objekte zato, da bi na podlagi podobnosti uporabniki znali uporabljati ta objekt. S tem lahko prenesemo tudi nekatere lastnosti, ki jih fizični objekt ima, ne pa nujno tudi "naš" objekt, ki ga predstavljamo s to metaforo.

40. Razložite metaforo iskalnega žarometa, ki opisuje obnašanje človekove pozornosti med zaznavanjem.

    Rešitev:

    - Pozornost je v danem trenutku usmerjena le na en vhodni kanal, recimo na vizualni kanal ali pa na zvočni kanal.

- Pozornost se spreminja serijsko od enega vhodnega kanala do drugega. V nekem trenutku je pozornost usmerjena na primer samo na vizualni kanal, če pa želimo slediti pogovoru, moramo pozornost preusmeriti na zvočni kanal.
- Prevladuje vizualna dominanca oziroma lažje je slediti vizualnim kanalom kot zvočnim kanalom.
- Čeprav imamo lahko pozornost v nekem trenutku usmerjeno samo na en kanal, pa lahko znotraj tega kanala procesiramo več dražljajev vzporedno.

41. Ko govorimo o človekovem zaznavanju, se srečamo s principom kavzalnosti. Razložite ta princip, razložite pa tudi navidezno kavzalnost. Podajte tudi kakšen primer kavzalnosti.

    Rešitev: Kavzalnost je relacija med dvema dogodkoma, vzrokom in posledico, kjer se drugi dogodek zgodi kot odziv na prvega. O navidezni kavzalnosti pa govorimo takrat, ko se nek dogodek zgodi neposredno po neki akciji in izgleda, kot da je posledica te akcije. V tem primeru povezujemo posledico z *navideznim* vzrokom, čeprav ni bil nujno vzrok posledice. Primer kavzalnosti je na primer oblika gumba ob pritisku gumba z miško, ki se ob pritisku spremeni in se vrne v prvotno stanje, ko ta gumb spustimo.

42. Pri uporabi sistema lahko uporabnika zmede napačna kavzalnost. Kakšne vrste napačne kavzalnosti poznamo? Podajte kakšen primer napačne kavzalnosti.

    Rešitev: O napačni kavzalnosti lahko govorimo v primeru navidezne kavzalnosti, ko se je nekaj zgodilo s sistemom neposredno po naši interakciji s sistemom, čeprav ta odziv ni bil rezultat prvotne akcije. Tako lahko uporabnik na primer povezuje sesutje aplikacije ali sistema z neko akcijo, ki jo je zahteval neposredno pred sesutjem. Drugi primer napačne kavzalnosti pa je nevidna kavzalnost. Pri nevidni kavzalnosti neka akcija nima vidne posledice, kar lahko uporabnika vodi v zmoten zaključek, da akcija ni bila izvedena, zaradi česar lahko akcijo ponovi. Akcije, od katerih se pričakuje odziv, morajo imeti viden odziv.

43. Razložite Hick-Hymanov zakon o reakcijskem času.

    Rešitev: Hick-Hymanov zakon pravi, da je enostaven reakcijski čas enak času ciklov človekovih procesorjev za obdelavo informacije: $RT = T_p + T_c + T_m$. To je čas, ki je potreben za sprejem enega dražljaja in izdajo enega odziva. Če pa se mora uporabnik odločiti, na primer izbrati drugačno odločitev za

vsak vhodni dražljaj, potem je reakcijski čas odvisen tudi od informacijske vsebine dražljaja. Število ciklov, ki jih zahteva procesor za razumevanje, je sorazmerno količini informacije dražljaja. Tako potrebuje procesor za razumevanje $log(N)$ ciklov za razpoznavo in primeren odziv na dražljaje, če imamo N enako verjetnih dražljajev.

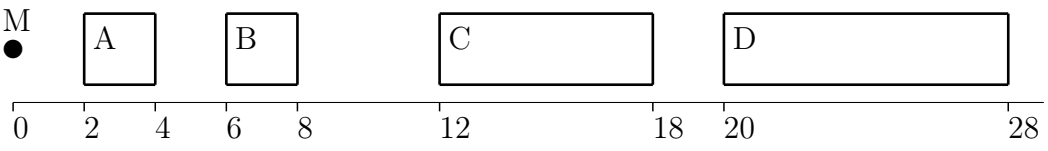
44. Na zaslonu so štiri tarče. Oddaljenosti centrov tarč od trenutnega položaja miške in širine tarč so: A: 8 cm (širina 5 cm), B: 6 cm (širina 2 cm), C: 12 cm (širina 9 cm) in D: 9 cm (širina 6 cm). Za vsako od teh štirih tarč napišite indeks težavnosti za premik roke do tarče. Katero tarčo dosežemo z miško najhitreje in katero najkasneje? Privzemite, da lahko premikate roko enako dobro v vseh smereh in da nobena od tarč ni blizu roba zaslona.

Rešitev: Imamo tarče, do katerih lahko neovirano dostopamo. Shannonova formulacija Fittovega zakona pravi, da je indeks težavnosti za premikanje do tarče enak $log_2(D/S+1)$, kjer je D razdalja med položajem miške in centrom tarče, S pa širina tarče. Tako lahko potem za vse štiri tarče zapišemo indeks težavnosti in ga tudi izračunamo:

A: $log_2(D/S+1) = log_2(13/5) = log_2(2.6)$;
B: $log_2(D/S+1) = log_2(8/2) = log_2(4)$;
C: $log_2(D/S+1) = log_2(7/3) = log_2(2.33)$;
D: $log_2(D/S+1) = log_2(5/2) = log_2(2.5)$.

Na podlagi zgornjih štirih izračunov vidimo, da najhitreje dosežemo tarčo C in najkasneje tarčo B.

45. Za vsako od teh štirih tarč napišite indeks težavnosti za premik roke do tarče. Katero tarčo zadenemo z miško najhitreje in katero najkasneje, če je začetni položaj miške v točki M? Privzemite, da lahko premikate roko enako dobro v vseh smereh in da nobena od tarč ni blizu roba zaslona.



Rešitev: Shannonova formulacija Fittovega zakona pravi, da je indeks tažavnosti za pomik roke enak $log_2(D/S+1)$, kjer je D razdalja od položaja

miške do centra tarče, S pa širina tarče. Na podlagi slike za vsako tarčo izračunamo razdaljo od položaja miške do centra posamezne tarče, širine tarče so na voljo, tako da lahko zapišemo indekse težavnosti:

A: $log_2(D/S + 1) = log_2(5/2) = log_2(2.5)$;

B: $log_2(D/S + 1) = log_2(9/2) = log_2(4.5)$;

C: $log_2(D/S + 1) = log_2(21/6) = log_2(3.5)$;

D: $log_2(D/S + 1) = log_2(32/8) = log_2(4)$.

Najhitreje zadenemo tarčo A in najkasneje tarčo B.

46. Napišite relacijo med časom (T), razdaljo (D) in velikostjo (S) za nalogo premika roke do tarče velikosti S na razdalji D in za nalogo premika roke skozi tunel dolžine D in širine S. Katera naloga je težja?

Rešitev:

- Premik roke do tarče: $T \sim log_2(D/S + 1)$;
- premik roke skozi tunel: $T \sim D/S$;
- težja je naloga premika roke skozi tunel.

47. Napišite indeks težavnosti za nalogo premikanja roke do tarče širine S na razdalji D in indeks težavnosti za nalogo premikanja roke skozi tunel dolžine D ter širine S. Na zaslonu sta dve tarči, tarča A in tarča B. Oddaljenost centra tarče A od trenutnega položaja miške je 8 cm. Širina tarče A je 4 cm. Pot do tarče A je prosta, medtem ko se tarča B nahaja na koncu tunela dolžine 6 cm. Kakšna mora biti širina tunela, da dosežemo tarčo B enako hitro kot tarčo A?

Rešitev: Tukaj imamo nalogo premikanja in nalogo vodenja. Naloga vodenja je težja, kot je naloga premikanja. Indeksa težavnosti sta:

- premik roke do tarče: $log_2(D/S + 1)$;
- premik roke skozi tunel: D/S.

Na podlagi indeksov lahko izračunamo širino tunela:

- A: $log_2(D/S + 1) = log_2(8/4 + 1) = log_2(3)$;
- B: $log_2(3) = D/S = 6$ cm$/S$; $S = 6$ cm$/log_2(3)$.

Da dosežemo obe tarči enako hitro, mora biti širina tunela enaka (6 cm$/log_2(3)$).

48. Napišite indeks težavnosti za nalogo premikanja roke do tarče širine S na razdalji D in indeks težavnosti za nalogo premikanja roke skozi tunel dolžine D ter širine S. Na zaslonu sta dve tarči, tarča A in tarča B. Oddaljenost centra tarče A od trenutnega položaja miške je 12 cm. Širina tarče A je 3 cm. Pot do tarče A je prosta, medtem ko se tarča B nahaja na koncu tunela dolžine 8 cm. Kakšna mora biti širina tunela, da dosežemo tarčo B hitreje kot tarčo A?

    Rešitev: Tukaj imamo nalogo premikanja in nalogo vodenja. Naloga vodenja je težja, kot je naloga premikanja. Indeksa težavnosti sta:

    - premik roke do tarče: $log_2(D/S + 1)$;
    - premik roke skozi tunel: D/S.

    Na podlagi indeksov lahko izračunamo minimalno širino tunela:

    - A: $log_2(D/S + 1) = log_2(12/3 + 1) = log_2(5)$;
    - B: $log_2(5) > D/S$; $log_2(5) > 8$ cm$/S$; $S > 8$ cm$/log_2(5)$.

    Da dosežemo tarčo B hitreje kot tarčo A, mora biti širina tunela večja od (8 cm$/log_2(5)$).

49. Napišite indeks težavnosti za nalogo premikanja roke do tarče širine S na razdalji D in indeks težavnosti za nalogo premikanja roke skozi tunel dolžine D ter širine S. Na zaslonu sta dve tarči, tarča A in tarča B. Oddaljenost centra tarče A od trenutnega položaja miške je 10 cm. Širina tarče A je 5 cm. Pot do tarče A je prosta, medtem ko se tarča B nahaja na koncu tunela širine 7 cm. Kolikšna je lahko največja dolžina tunela, da dosežemo tarčo B hitreje kot tarčo A?

    Rešitev: Tukaj imamo nalogo premikanja in nalogo vodenja. Naloga vodenja je težja, kot je naloga premikanja. Indeksa težavnosti sta:

    - premik roke do tarče: $log_2(D/S + 1)$;
    - premik roke skozi tunel: D/S.

    Na podlagi indeksov lahko izračunamo največjo dolžino tunela:

    - A: $log_2(D/S + 1) = log_2(10/5 + 1) = log_2(3)$;
    - B: $log_2(3) > D/S = D/7$ cm; $D < 7$ cm $\cdot\, log_2(3)$;

    Da dosežemo tarčo B hitreje kot tarčo A, mora biti dolžina tunela manjša od 7 cm $\cdot\, log_2(3)$.

50. Ob načrtovanju uporabniškega vmesnika ste se odločili za krožne menije in ne za linearne dvižne (pojavne) menije. Napišite indeks težavnosti za nalogo premikanja roke do želene opcije v krožnem meniju in indeks težavnosti za nalogo premikanja roke do želene opcije v linearnem dvižnem meniju. Z besedami napišite, kaj predstavlja vsaka od spremenljivk v obeh indeksih težavnosti. Privzemite, da se ob aktivaciji krožnega menija kurzor miške pojavi v centru menija, ob aktivaciji linearnega dvižnega menija pa na njegovem vrhu. Kateri meni v splošnem zagotavlja hitrejše delo? Svoj odgovor utemeljite.

Rešitev:

- Premik roke do opcije v krožnem meniju: $log_2(D/S + 1)$;
  D - razdalja do centra opcije od začetnega položaja miške;
  S - širina opcije.
- Premik roke do opcije v linearnem dvižnem meniju: D/S;
  D - razdalja do opcije od začetnega položaja miške;
  S - širina menija (tunela).
- Hitrejše delo zagotavlja krožni meni.
- Utemeljitev: Krožni meni ima enak D za vsako opcijo in enak S za vsako tarčo, saj je položaj miške v centru menija; linearni dvižni meni ima za bolj oddaljene opcije večji D in konstanten S. Ker so v linearnem dvižnem meniju bolj oddaljene tarče težje dostopne (večji D), je v povprečju krožni meni hitrejši (od 15 % do 20 %).

51. Imamo krožni meni in linearni dvižni meni. Napišite indeks težavnosti za izbiro v vsakem izmed teh dveh menijev. Katero tarčo dosežemo hitreje, tarčo A v krožnem meniju (širina tarče je 3 cm, oddaljenost centra tarče od položaja miške je 9 cm) ali tarčo B v linearnem dvižnem meniju (širina tarče je 3 cm, oddaljenost tarče od položaja miške je 9 cm).

Rešitev:

- Premik roke do končne izbire v krožnem meniju je: $T = log_2(D/S + 1)$;
- premik roke do končne izbire v linearnem dvižnem meniju je: $T = (D/S)$;
- indeks težavnosti za izbiro v krožnem meniju je: $T = log_2(D/S + 1) = log_2(9\text{ cm}/3\text{ cm} + 1) = log_2(4) = 2$;
- indeks težavnosti za izbiro v linearnem dvižnem meniju je: $T = (D/S) = (9\text{ cm}/3\text{ cm}) = 3$.

Hitreje dosežemo tarčo A, ki se nahaja v krožnem meniju.

52. Na zaslonu imamo štiri tarče, tarčo A, tarčo B, tarčo C in tarčo D. Tarče si sledjo zaporedoma. Razmik med vsemi tarčami je enak in znaša 3 cm. Tarča A je široka 2 cm, tarča B je široka 3 cm, tarča C je široka 4 cm in tarča D je široka 6 cm. Katero izmed tarč dosežemo najhitreje in katero najkasneje, če je trenutni položaj miške 10 cm pred tarčo A? Za vsako od teh štirih tarč napišite tudi indeks težavnosti za premik roke do tarče.

Rešitev:

A: $T = log_2(11/2 + 1) = log_2(13/2) = log_2(6.5)$;

B: $T = log_2(16.5/3 + 1) = log_2(19.5/3) = log_2(6.5)$;

C: $T = log_2(23/4 + 1) = log_2(27/4) = log_2(6.75)$;

D: $T = log_2(31/6 + 1) = log_2(37/6) = log_2(6.17)$.

Najhitreje dosežemo tarčo D, najkasneje pa tarčo C.

53. Na zaslonu imamo štiri tarče, tarčo A, tarčo B, tarčo C in tarčo D. Tarče si sledjo zaporedoma. Razmak med tarčama A in B znaša 3 cm, med tarčama B in C znaša 4 cm ter med tarčama C in D 11 cm. Tarča A je široka 2 cm, tarča B je široka 4 cm, tarča C je široka 6 cm. Za vsako od teh štirih tarč napišite indeks težavnosti za premik roke do tarče in izračunajte, koliko mora biti širina tarče D, da:

a) tarčo D dosežemo najhitreje;

b) tarčo D dosežemo najpočasneje.

Pri tem predpostavite, da je trenutni položaj miške 5 cm pred tarčo A.

Rešitev:

A: $T = log_2(6/2 + 1) = log_2(4)$;

B: $T = log_2(12/4 + 1) = log_2(4)$;

C: $T = log_2(21/6 + 1) = log_2(4.5)$;

D: $T = log_2((35\text{ cm} + S/2)/S + 1)$.

Da tarčo D dosežemo najhitreje, mora biti indeks težavnosti manjši od vseh preostalih indeksov, torej mora veljati, da je $T = log_2((35\text{ cm} + S/2)/S + 1) < log_2(4)$, torej da je indeks težavnosti manjši od najmanjšega indeksa (za tarčo A oziroma B). Iz tega sledi, da mora biti širina S tarče D večja od 14 cm, da dosežemo tarčo D najhitreje.
Podobno lahko izpeljemo tudi v primeru, ko želimo, da tarčo D dosežemo najpočasneje. V tem primeru mora veljati, da je $T = log_2((35\text{ cm} + S/2)/S + 1) >$

$log_2(4.5)$, torej da je indeks težavnosti večji od največjega indeksa (za tarčo C). Iz tega sledi, da mora biti širina S tarče D manjša od 35 cm/3 cm oziroma 11.67 cm, da dosežemo tarčo D najpočasneje.

5 Interakcije

54. Naštejte nekaj stilov interakcije med človekom in računalnikom.

    Rešitev: Stili interakcije med človekom in računalnikom so:

    - ukazna vrstica,
    - forma,
    - meni,
    - direktna manipulacija.

55. Naštejte lastnosti koncepta direktne manipulacije.

    Rešitev: Lastnosti koncepta direktne manipulacije so:

    - sistem je predstavljen kot podaljšek realnega sveta,
    - zvezna vidljivost objektov in akcij (koncept WYSIWYG),
    - fizične interakcije,
    - uporablja "prepoznavanje",
    - hitri in vidni rezultati akcij,
    - reverzibilni rezultati akcij,
    - obstaja akcija povleci in akcija povleci - spusti.

56. Vmesniki, ki temeljijo na direktni manipulaciji, uporabljajo Normanove namige. Kaj zagotovijo Normanovi namigi vmesniku? Naštejte Normanove namige. Katera sta osnovna načina namiga *pomagljivost*, ki povesta uporabniku, kako uporabljati spletne strani?

    Rešitev: Normanovi namigi zagotavljajo boljšo komunikacijo med uporabnikom in sistemom.
    Namigi so:

    - pomagljivost,
    - oznake,
    - omejitve,

- naravne preslikave,
- vidljivost,
- povratna informacija.

Osnovna načina namiga *pomagljivosti*, ki povesta uporabniku, kako uporabljati spletno stran, sta:

- povezave so realizirane kot obarvan podčrtan tekst;
- oblika kurzorja miške se spremeni, če ta prekrije povezavo.

57. Eden izmed Normanovih namigov je *pomagljivost*. Kako namig *pomagljivost* pomaga pri izboljšanju komunikacije med uporabnikom in sistemom. Podajte tudi kakšen konkreten primer za ta namig.

Rešitev: Namig *pomagljivost* izboljšuje komunikacijo tako, da uporabniku sugerira različne akcije, ki jih sistem omogoča. Primer pri uporabniških vmesnikih je kurzor, ki spremeni obliko, ko se premaknemo nad objekt, ki omogoča neko akcijo (ko se z miško premaknemo nad besedilo v urejevalniku, ki ga lahko urejamo, se oblika kurzorja spremeni, prav tako se spremeni oblika tudi, če se premaknemo nad povezavo, ki jo lahko kliknemo).

58. Eden izmed Normanovih namigov je *pomagljivost*. *Pomagljivost* pa se lahko pojavlja v dveh oblikah. Povejte, kateri obliki sta to, ju na kratko opišite in napišite, v kakšnem odnosu sta.

Rešitev: *Pomagljivost* je lahko zaznavna in dejanska. Pri zaznavni *pomagljivosti* uporabnik zazna neko akcijo kot možno, pri dejanski *pomagljivosti* pa je neka akcija dejansko mogoča. Množici zaznavnih in dejanskih pomagljivosti sta lahko enaki, in sicer v primeru, ko uporabnik zazna akcije, ki jih objekt omogoča, lahko pa se razlikujeta v primeru, da uporabnik ne zazna mogoče akcije (sicer v tem primeru ne moremo govoriti o pomagljivosti, ker ne obstaja) ali uporabnik napačno zazna akcijo, ki je vmesnik ne omogoča.

59. Pri Normanovem namigu *pomagljivost* govorimo o zaznavni in dejanski *pomagljivosti*. Opišite obe vrsti *pomagljivosti* in podajte primer, ko se zaznavna in dejanska *pomagljivost* razlikujeta.

Rešitev: *Pomagljivost* daje namige o tem, kako lahko nek objekt uporabljamo. Pri zaznavni *pomagljivosti* gre za namig o uporabi, kjer uporabnik neko akcijo zazna kot mogočo, četudi dejansko ni mogoča. Pri dejanski pomagljivosti pa gre za akcijo, ki jo uporabnik zazna kot mogočo, objekt pa to akcijo tudi dejansko omogoča. Primer zaznavne *pomagljivosti*, ki ni tudi

dejanska pomagljivost, je na primer podčrtano besedilo modre barve, ki ga prikažemo v spletnem brskalniku in obenem omogočimo tudi spremembo oblike kurzorja, če se z miško premaknemo nad to besedilo. V tem primeru je to navadno besedilo in ne povezava, zato klik na to besedilo ne služi nobeni akciji.

60. Naštejte tri pristope, ki sledijo Normanovemu namigu *vidljivost* in zagotovijo uporabniku vidljivost stanja med navigacijo po vmesniku, ki temelji na direktni manipulaciji, kot je na primer spletna stran.

    Rešitev: Pristopi, ki zagotavljajo vidljivost stanja med navigacijo po spletni strani, so:

    - prikazana je pot navigacije;
    - prikazane so številke strani;
    - jezički (angl. "tabs"), metafora indeksiranih kartic.

61. Naštejte nekaj načinov Normanovega namiga *pomagljivost*, ki zagotovijo uporabniku vidljivost akcij med direktno manipulacijo.

    Rešitev: Namigi pomagljivosti, ki zagotovijo uporabniku vidljivost akcij, so:

    - 3D oblika gumbov,
    - gumbi in povezave,
    - obarvani in podčrtani teksti,
    - značke za izvlečne menije,
    - teksture,
    - oblika kurzorja miške,
    - poudarjanje gradnika pod miško,
    - ikona med akcijo povleci - spusti oziroma akcija povleci - spusti.

62. Eden izmed Normanovih namigov so tudi *naravne preslikave*. Razložite, kako namig *naravne preslikave* omogoča lažjo uporabo sistema.

    Rešitev: Namig *naravne preslikave* izkorišča primerne in predvsem naravne povezave med kontrolami nekega sistema in učinkom, ki jih imajo te kontrole na sistem. Funkcija naravnih preslikav je v tem, da omogočajo zmanjšanje potrebnosti informacij iz uporabnikovega spomina za opravilo neke naloge. Pri fizičnih objektih gre za fizičen učinek, na primer levo stikalo za prižig luči na levi strani, desno stikalo za prižig luči na desni strani. Pri uporabniških vmesnikih pa imamo podobne učinke na komponente vmesnika, na primer pri drsniku pritisk na puščico gor ali dol povzroči premik vidne odprtine gor ali dol.

63. Eden izmed Normanovih namigov so *naravne preslikave*. Ko govorimo o namigu *naravne preslikave*, pa se pojavi tudi pojem preslikave. Razložite razlike oziroma podobnosti med pojmoma naravne preslikave in preslikave.

    Rešitev: Oba pojma sta podobna z vidika, da govorita o relacijah med kontrolami, njihovim premikanjem oziroma interakcijo z njimi ter njihovim dejanskim učinkom na sistem. Edina razlika med obema pojmoma je, da *naravne preslikave* zagotavljajo uporabniku pravilno organizirane kontrole, s pomočjo katerih uporabnik lahko takoj razume, kakšen učinek ima kontrola na sistem oziroma s katero kontrolo bo dosegel želen učinek. Pri *preslikavah* pa to ni nujno res, saj preslikava ne posreduje nujno vizualne informacije o učinku (na primer, premik puščice navzdol v vsebovalniku z drsniki bi povzročil premik vidne odprtine levo, če bi bilo to mogoče).

64. Naštejte nekaj poti, po katerih podoba drsnik demonstrira Normanov namig *naravne preslikave*.

    Rešitev: Podoba drsnik demonstrira *naravne preslikave*:

    - puščici gor in dol, ki se nahajata nad oziroma pod drsnikom;
    - pritisk na puščico gor oziroma dol povzroči premik vidne odprtine znotraj dokumenta gor oziroma dol;
    - pomik gumba gor oziroma dol povzroči premik vidne odprtine znotraj dokumenta gor oziroma dol;
    - višina gumba v drsniku v primerjavi z višino celotnega drsnika je sorazmerna višini vidne odprtine v primerjavi z višino celotnega dokumenta;
    - klik v drsniku nad gumbom premakne vidno odprtino gor, klik v drsniku pod gumbom pa premakne vidno odprtino dol.

65. Eden izmed Normanovih namigov so tudi *oznake*. Kaj je ta namig in zakaj se uporablja?

    Rešitev: Normanov namig *oznake* je namig, ki se razlikuje glede na fizični svet in grafične uporabniške vmesnike. *Oznake* so tekstovni opisi, ki so dodani kontrolam, s katerimi izvajamo akcije. V fizičnem svetu ponavadi velja, da če je pri preprostih vmesnikih potrebno uporabiti *oznake*, potem je napaka v načrtu takega vmesnika. Pri grafičnih uporabniških vmesnikih pa se *oznake* uporabljajo zato, da omogočajo lažjo in bolj intuitivno uporabo vmesnika in so pogosto zaželene in potrebne, njihova uporaba pa ni nujno posledica slabo načrtanega sistema.

66. Eden izmed Normanovih namigov so tudi *omejitve*. Kaj je ta namig in kako ter zakaj se uporablja? Podajte tudi en primer *omejitve* v fizičnem svetu in enega v "navideznem" svetu.

    Rešitev: Težavnost prilagajanja na novo situacijo oziroma uporabo sistema ali aplikacije je neposredno odvisna od števila možnosti, ki jih ima uporabnik na razpolago. Z omejevanjem števila možnosti olajšamo prilagajanje oziroma uporabo sistema. Namig *omejitve* omogoča zmanjševanje števila možnosti. S *pomagljivostjo* uporabniku sugeriramo možne akcije, z *omejitvami* pa jih omejimo. S tem uporabniku olajšamo delo; tako na primer priključka VGA ne moremo vstaviti v priključek HDMI, prav tako pa drsnika ne moremo premakniti preko meje vsebine dokumenta.

67. Eden izmed Normanovih namigov so tudi *omejitve*. *Omejitve* pa lahko delimo na različne kategorije. Naštejte te kategorije in podajte primer za posamezno kategorijo *omejitve*.

    Rešitev: Namig *omejitve* lahko razdelimo na naslednje kategorije:

    a) fizične omejitve: kako lahko nek objekt uporabljamo; drsnika, recimo, ne moremo premakniti preko roba dokumenta;

    b) semantične omejitve: kako se objekt uporablja glede na dano situacijo; pri uporabi sistema imamo namizje prikazano/postavljeno od zgoraj navzdol;

    c) kulturne omejitve: kako se objekt prikaže glede na kulturo, v kateri se uporablja; besedilo v oknu je prikazano od zgoraj navzdol in od leve proti desni;

    d) logične omejitve: kjer se upoštevajo logične relacije, na primer za brisanje datoteke se uporablja koš za smeti.

68. Kaj so konvencije? Kako so povezane z Normanovim namigom *omejitve*?

    Rešitev: Konvencije so kulturne *omejitve*. Na začetku so konvencije arbitrarne oziroma poljubne, vendar se s časom razvijejo in postanejo sprejete. Ker so konvencije kulturne *omejitve*, se lahko med različnimi kulturami tudi razlikujejo. Še posebej je to pomembno pri izbiri barve, ki ima v različnih kulturah lahko izrazito drugačen pomen. Tako ima na primer rdeča barva v zahodnem svetu pomen nevarnosti, v Indiji in na Japonskem predstavlja življenje, na Kitajskem pa srečo.

69. Vmesniki, ki temeljijo na direktni manipulaciji, uporabljajo Normanove namige, prav tako pa so vmesniki načrtani na osnovi principov uporabnosti

in na osnovi (na primer) Nielsenovih principov načrtovanja uporabniških vmesnikov. Spodaj so našteti nekateri Normanovi namigi, nekateri principi uporabnosti in nekateri Nielsonovi principi. Z besedami napišite, kateri štirje so Normanovi namigi.

- Konsistentnost in standardi
- Učinkovitost
- Naravne preslikave
- Naučljivost
- Oznake
- Vidljivost
- Izogibanje napakam
- Uporabnikov nadzor in svoboda
- Pomagljivost
- Manj je več

Rešitev: Med zgornjimi pojmi so naslednji Normanovi namigi: *vidljivost*, *naravne preslikave*, *pomagljivost* in *oznake*.

70. Napišite tri dejstva, ki razlagajo, da Unixov ukaz *cp* (kopiranje datotek) oziroma DOS-ov ukaz *copy* (kopiranje datotek) ne dosegata definicije direktne manipulacije.

    Rešitev: Dejstva, ki razlagajo, da ta ukaza ne dosegata definicije direktne manipulacije:

    a) ni zvezne vidljivosti objektov in akcij: za to, da vidmo, katere datoteke obstajajo, moramo izvršiti ukaz *ls* oziroma *dir*, sicer objekti niso vidni;

    b) ne uporablja prepoznavanja pač pa pomnjenje: ukaz za kopiranje moramo poznati na pamet skupaj s parametri;

    c) rezultati niso takoj vidni: ko izvršimo ukaz za kopiranje, moramo ponovno izvršiti ukaz *ls* oziroma *dir* zato, da vidimo rezultat.

71. Naštejte vsaj eno prednost in vsaj eno slabost vsakega od naslednjih stilov interakcije med človekom in računalnikom: ukazna vrstica, forma (polje za vnos), meni in direktna manipulacija.

    Rešitev: Prednosti (+) in slabosti (-) so:

    - **Ukazna vrstica:**
        + fleksibilnost (omogoča na primer uporabo stikal za prilagajanje osnovnega ukaza, veriženje ukazov);
        + privlačna za izkušene uporabnike;
        + bližnjice za pogoste operacije preko skript;
        - zahteva veliko vaje;
        - pomnjenje ukazov.

- **Forma:**
  + preprost vnos podatkov;
  + ne zahteva veliko vaje;
  + pomaga uporabniku s pričakovanimi vhodi;
  - zahteva veliko prostora.
- **Meni:**
  + razbremenjuje spomin;
  + učenje uporabe je enostavno;
  + hitro delo;
  + strukturiranost opcij;
  + enostavna uporaba za neizkušene uporabnike;
  - nevarnost preveč kompleksnih menijev;
  - počasno delo za izkušene uporabnike, če ni bližnjic.
- **Direktna manipulacija:**
  + koncepti opravil so predstavljeni vizualno z objekti;
  + ni se je težko naučiti;
  + učenje uporabe je enostavno;
  + omogoča reševanje iz napak;
  + vzpodbuja preiskovanje;
  - zahteva grafični prikaz;
  - zahteva uporabo zveznih vhodnih naprav (miško, zaslon na dotik);
  - ikone in metafore imajo lahko različen pomen za različne skupine uporabnikov.

72. Kaj je konceptualni model? Kaj lahko vpliva na izgradnjo uporabnikovega konceptualnega modela?

    Rešitev: Konceptualni model je mentalni model, ki razlaga, kako neka stvar deluje. Konceptualni model se formira v glavi osebe. Na izgradnjo uporabnikovega konceptualnega modela lahko vplivajo:

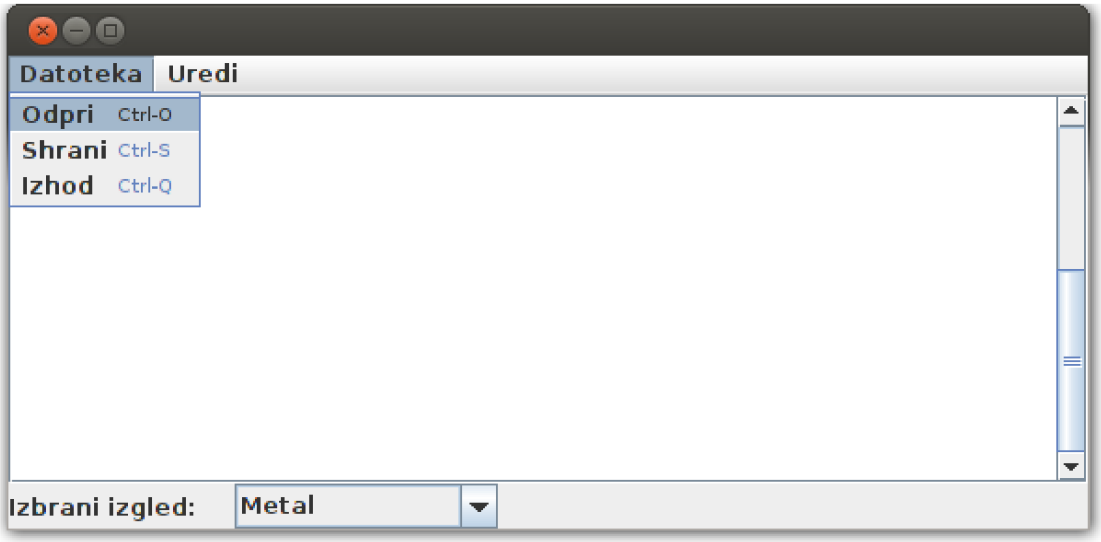
    - poznavanje podobnih naprav ali programov,
    - pomagljivost,
    - naravne preslikave,
    - omejitve,
    - kavzalnost,
    - navodila,

- interakcija z napravo ali programom.

73. Razložite, zakaj se bo model načrtovalca razlikoval od modela končnega uporabnika.

    Rešitev: Načrtovalci imajo v glavi svoj konceptualni model, model sistema. Zaradi tega načrtovalci ponavadi razumejo, na kakšen način je sistem narejen. Prav tako je zelo verjetno, da imajo več znanja in izkušenj z velikim številom informacijskih tehnologij in tehnik za interakcijo. Zaradi tega se lahko načrtovalci zelo težko postavijo v vlogo novega uporabnika interaktivnega sistema. To lahko povzroči, da načrtovalci neprimerno ocenijo delovne prakse, ki jih mora podpirati programska oprema.

74. Na uporabniškem vmesniku, ki je prikazan na sliki 5.1, prepoznajte in označite vse vidne namige *pomagljivosti*.



Slika 5.1 Slika uporabniškega vmesnika z nekaterimi vidnimi namigi *pomagljivosti*.

Rešitev: Prikazani so naslednji vidni namigi *pomagljivosti*:

- slika 5.2(a): označen meni; vidimo lahko, da je v vrstičnem meniju označen gumb, na katerega je uporabnik pritisnil, da se je prikazal meni, v tem meniju pa je označena tudi trenutna izbira;
- slika 5.2(b): značka za izvlečni meni; vidimo, da je pri opciji, ki omogoča izbiro izgleda, dodan gumb z oznako (puščica), ki nakazuje na to, da obstaja možnost prikaza dodatnih izbir;

- slika 5.2(c): tekstura drsnika, ki spominja na fizične drsnike s hrapavo ali nazobčano površino, ki omogoča lažje upravljanje.

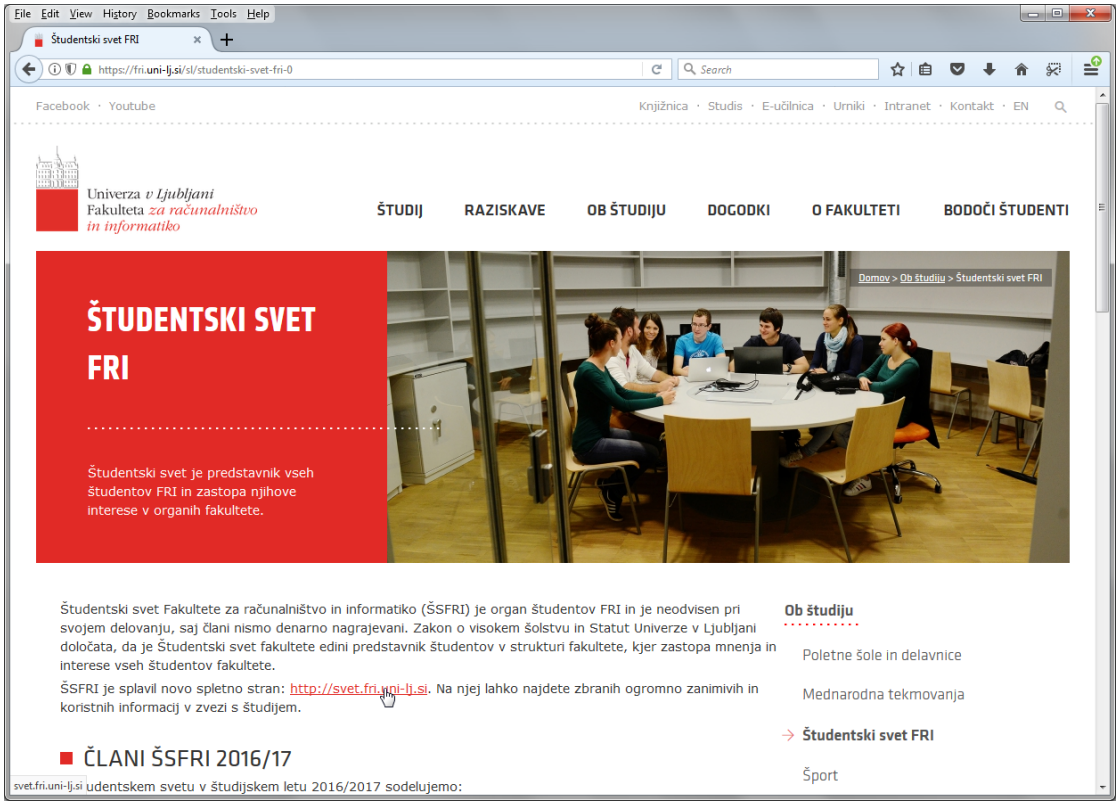
(a) (b) (c)

Slika 5.2 Detajlni prikaz naslednjih namigov *pomagljivosti*: (a) označen meni, (b) značka za izvlečni meni in (c) tekstura drsnika.

75. Na sliki 5.3, ki prikazuje spletno stran, prikazano v brskalniku, prepoznajte in označite vse vidne namige *pomagljivosti*.

    Rešitev: Na sliki 5.4 so prikazani naslednji vidni namigi *pomagljivosti*:

    - slika 5.4(a): barva povezav, ki se loči od barve besedila;
    - slika 5.4(b): oblika kurzorja in podčrtana povezava ob prehodu čez povezavo;
    - slika 5.4(c): tekstura drsnika, ki spominja na fizične drsnike s hrapavo ali nazobčano površino, ki omogoča lažje upravljanje.

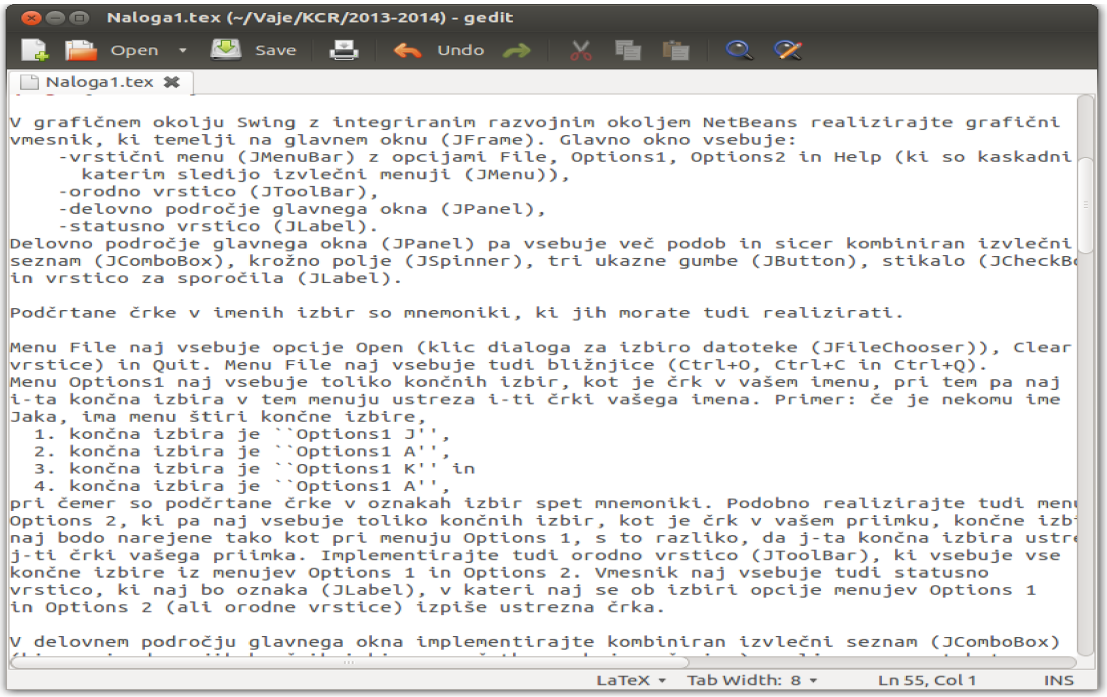


Slika 5.3 Slika brskalnika z naloženo spletno stranjo, v katerem so prikazani nekateri namigi *pomagljivosti*.

(a) (b) (c)

Slika 5.4 Detajlni prikaz naslednjih namigov *pomagljivosti*: (a) barva povezav, (b) oblika kurzorja in (c) tekstura drsnika.

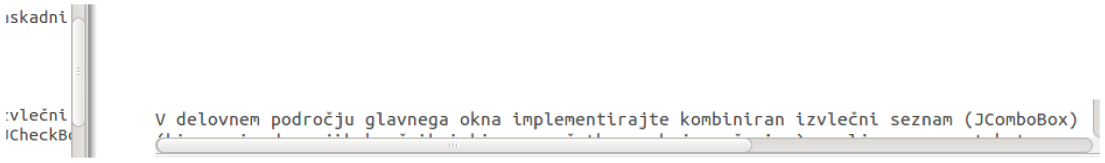
76. Na uporabniškem vmesniku, ki se nahaja na sliki 5.5, prepoznajte, označite in opišite vse vidne namige *vidljivosti*, ki pomagajo pri navigaciji po dokumentu.



Slika 5.5 Slika vmesnika za urejanje tekstovnih datotek, kjer so prikazani nekateri namigi *vidljivosti*.

Rešitev: Namigi *vidljivosti*, ki so vidni na sliki 5.5, so:

- slika 5.6(a): drsnik, ki po obliki in izgledu namiguje na to, da ga je možno prijeti in premikati;
- slika 5.6(b): velikost in položaj drsnika namigujeta na količino in na to, kateri del besedila je prikazan glede na celotno besedilo.

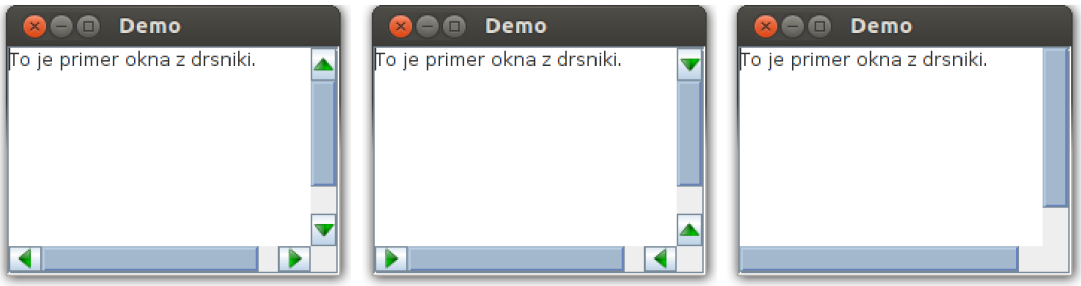


(a)

Slika 5.6 Detajlni prikaz naslednjih namigov vidljivost: (a) drsnik in (b) oblika ter položaj drsnika.

.

77. Na sliki 5.7 so prikazane tri implementacije drsnikov. Prva implementacija (slika 5.7(a)) ima puščice pri drsnikih, ki so obrnjene navzven in prikazujejo smer premikanja drsnika. Druga implementacija (slika 5.7(b)) ima drsnike, ki so obrnjeni navznoter in prikazujejo smer premikanja besedila v oknu. Tretja implementacija (slika 5.7(c)) pa nima puščic pri drsnikih. Izberite najprimernejšo implementacijo glede na naravno preslikavo.



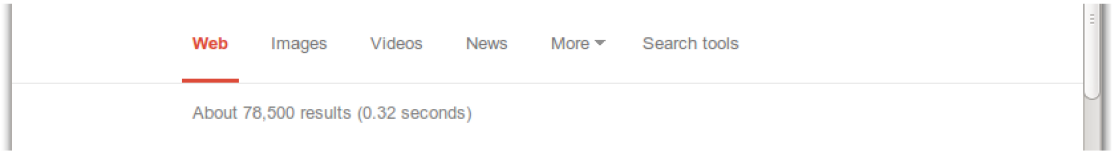
(a) (b) (c)

Slika 5.7 Slika prikazuje tri okna z implementacijo drsnikov, kjer: (a) so puščice pri drsnikih obrnjene navzven, (b) so puščice pri drsnikih obrnjene navznoter in (c) ni puščic pri drsnikih.

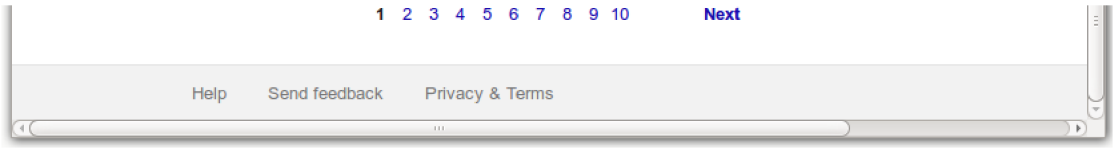
Rešitev: Vmesnika na slikah 5.7(a) in 5.7(b) vsebujeta puščice, ki dajo uporabniku vedeti, da lahko premika drsnik ter tudi vsebino okna levo in desno oziroma gor in dol, vmesnik na sliki 5.7(c) pa tega nima. Vmesnika na slikah 5.7(a) in 5.7(b) ponujata naravno preslikavo v obliki puščic, zaradi tega sta primernejša od vmesnika na sliki 5.7(c). Vmesnika na slikah 5.7(a) in 5.7(b) imata puščice, s katerimi lahko premikamo drsnik oziroma se lahko premikamo po vsebini okna. Zamenjana pa je usmeritev puščic. Glede na to, da imajo uporabniški vmesniki puščice realizirane tako kot na vmesniku na sliki 5.7(a) in so uporabniki tudi navajeni na tako razporeditev, je vmesnik na sliki 5.7(a) primernejši. Razporeditev, kot jo ima vmesnik na sliki 5.7(b), je bila pri prvotnem uporabniškem vmesniku *Star*, ki so ga razvijali v *Xerox* parku. Testirali so obe možnosti, kjer imajo posamezne puščice nekoliko različen pomen. Pri vmesniku na sliki 5.7(a) so puščice obrnjene navzven, kjer pritisk na puščico povzroči premik drsnika v ustrezno smer (levo/desno ali gor/dol), obenem pa se vsebina okna (na primer besedilo) premakne v nasprotno smer (če pritisnemo puščico, ki kaže navzdol, se bo drsnik premaknil navzdol, besedilo pa se bo premaknilo navzgor). Pri vmesniku na sliki 5.7(b) pa so puščice obrnjene navznoter, kjer pritisk na puščico povzroči premik vsebine okna (na primer besedila) v ustrezno smer (levo/desno ali gor/dol), obenem pa se drsnik premakne v nasprotno smer (če pritisnemo puščico, ki kaže navzdol, se bo besedilo premaknilo navzdol, drsnik pa se bo

premaknil navzgor). Obe možnosti nudita ustrezno funkcionalnost, vendar so se pri prenosu drsnikov v vmesnik za računalnike *Lisa Macintosh* odločili za razporeditev, kot jo ima vmesnik na sliki 5.7(a), ki je nato tudi obveljala pri ostalih vmesnikih.

78. Slike 5.8(a), 5.8(b) in 5.8(c) prikazujejo izseke spletnih strani, kjer so dodane komponente, ki prikazujejo vidljivost stanja med navigacijo. Vaša naloga je, da v vsakem izseku spletne strani poiščete komponento, ki prikazuje vidljivost stanja.



(a)



(b)

(c)

Slika 5.8 Slika prikazuje različne izseke spletnih strani, ki uporabniku olajšajo navigacijo po spletni strani.

Rešitev: Slika 5.8(a) prikazuje zavihke na spletni strani, preko katerih lahko dostopamo do vsebin, ki so razdeljene po posameznih tematskih sklopih. Slika 5.8(b) prikazuje številčenje strani, kjer lahko izberemo posamezno številko strani in s tem dostopamo do vsebine, ki je prikazana na posamezni strani. Slika 5.8(c) pa prikazuje pot navigacije (sredina zgoraj), s pomočjo katere se lažje znajdemo na strani, obenem pa omogoča tudi vračanje na posamezen del obiskane poti.

79. Za vsakega izmed Normanovih namigov (*pomagljivost*, *omejitve*, *naravne preslikave*, *vidljivost*, *povratna informacija*) povejte, kako ga vertikalni drsnik

uporablja za učinkovit dizajn.

Rešitev: Uporaba Normanovih namigov pri vertikalnih drsnikih:

- *pomagljivost*: drsnik, ki ga lahko vlečemo (drsimo z njim);
- *omejitve*: drsnik lahko premikamo samo vertikalno, ne pa tudi horizontalno, drsnika ne moremo premakniti čez meje vsebine dokumenta;
- *naravne preslikave*: puščice, s katerimi se lahko premikamo po vsebini okna;
- *vidljivost*: velikost/položaj drsnika glede na pozicijo/velikost okna;
- *povratna informacija*: ob premiku drsnika se premakne vsebina okna.

80. Opišite fizične omejitve ter namig *omejitve*.

Rešitev: Fizične omejitve se nanašajo na način, na katerega fizični objekti omejujejo premikanje stvari. Na primer, način, na katerega se lahko ključ USB vstavi v računalnik, je omejen z obliko in velikostjo (ali pa konektor DVI na primer). Namig *omejitve* pa na podoben način postavlja omejitve pri uporabi podob. Tako na primer drsnika ne moremo premakniti čez meje vsebine dokumenta.

81. Načrtovanje uporabniškega vmesnika je osredotočeno tudi na določanje ustreznih metafor za računalniški sistem. Tako nekateri sistemi vključujejo metafore za predstavitev delovanja računalniških diskov, zelo prepoznavna pa je na primer tudi metafora koša za odpadke, ki pri večini operacijskih sistemov zagotavlja abstrakcijo funkcije za brisanje. Pri tej metafori se datoteka izbriše (dejansko se premakne v mapo za recikliranje), ko jo premaknemo v koš za smeti. Kje se skrivajo nevarnosti, povezane s skrivanjem kompleksnosti računalniškega sistema na način z metaforami?

Rešitev: Najočitnejša nevarnost je ta, da uporabniki ne bodo vedeli, kaj naj naredijo, ko določena metafora odpove. Na primer, če pride do napake na disku, bodo morda imeli premalo vpogleda, da bi diagnosticirali opozorilo o formatu datoteke ali o napaki diska. Druga nevarnost pa je ta, da lahko razširijo metaforo na neprimerne načine. Na primer, uporabniki lahko dodajajo datoteke v koš za smeti, ne da bi ga izpraznili. Fizični objekt podpira obliko obnašanja, ki nima veliko smisla v navideznem dvojniku oziroma navidezni koš (metafora) nima nekega veljavnega vizualnega mehanizma, ki bi uporabnika vzpodbudil k temu, da bi koš izpraznil. Ponavadi sicer tovrstne metafore ponujajo vizualne namige, ki uporabniku nakazujejo, da so datoteke v košu, ne zagotavljajo pa informacije o količini, ki bi uporabnika lahko opozarjala na to, da bi bilo dobro koš tudi sprazniti.

6 Principi načrtovanja uporabniških vmesnikov

82. Naštejte tri osnovne (Mandelove) principe načrtovanja uporabniških vmesnikov.

    Rešitev: Osnovni principi načrtovanja uporabniških vmesnikov so:

    1) zagotovi nadzor uporabnika;
    2) reduciraj obremenitev uporabnikovega spomina;
    3) zagotovi konsistentnost.

83. Naštejte deset Nielsenovih principov načrtovanja uporabniških vmesnikov.

    Rešitev: Nielsenovi principi načrtovanja uporabniških vmesnikov so:

    1) prilagodi se realnemu svetu/uporabljaj splošne besede, ne tehnični žargon;
    2) konsistentnost in standardi/princip najmanjšega presenečenja;
    3) pomoč in dokumentacija/uporabniki ne berejo priročnikov;
    4) uporabnikov nadzor in svoboda/dobro označeni izhodi/uporabnik naj ne bo ujetnik;
    5) vidljivost statusa sistema/iluzija napredka/uporabnik mora biti ves čas obveščen;
    6) fleksibilnost in učinkovitost/zagotovi lahko naučljive bližnjice za pogoste operacije;
    7) izogibanje napakam/izbira je manj podvržena napakam od tipkanja;
    8) raje prepoznaj, kot si zapomni/uporabljaj menije, ne ukazni jezik;
    9) javljanje napak, diagnoza, reševanje/bodi natančen, daj dobro obvestilo o napaki;
    10) estetika in minimalistično načrtovanje/manj je več/preprostost/gradniki naj imajo večkratno vlogo/dobro grafično načrtuj.

84. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Prilagodi se realnemu svetu". Razložite ta princip.

Rešitev: Ta princip govori o tem, da moramo pri načrtovanju uporabniškega vmesnika poskrbeti za to, da je vmesnik načrtan v skladu z znanjem in pričakovanji tipičnega uporabnika vmesnika. Zaradi tega moramo uporabljati jezik, ki ga bo uporabnik razumel. Ne smemo uporabljati tehničnega žargona iz računalniške domene, če to ni domena uporabnika. Lahko pa se uporabljajo specifični izrazi iz uporabnikove domene, če to pripomore k razumljivosti in lažji uporabi vmesnika. Uporabnika se ne sme omejevati pri določanju imen oziroma izbiri nizov, ki jih sistem zahteva, če to ni potrebno. Prav tako pa naj se omogoči uporaba okrajšav, na primer pri ukaznih jezikih, oziroma prilagajanje terminologije, če to olajša delo uporabniku.

85. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Konsistentnost in standardi". Zakaj je ta princip pomemben in povejte, kako lahko zadostimo temu principu.

Rešitev: Ta princip je pomemben, ker omogoča hitro prilagajanje uporabnika na nov vmesnik. Znanje, ki ga je uporabnik pridobil pri uporabi nekega vmesnika, se lahko prenese tudi na drug vmesnik. Pri načrtovanju uporabniškega vmesnika naj podobne stvari izgledajo in se obnašajo podobno, različne pa različno. Potrebno je paziti na terminologijo (vedno isti izraz v vmesniku), velikost, položaj in barvo. Pri načrtovanju vmesnikov je priporočeno slediti podobnim vmesnikom in navodilom za posamezno platformo.

86. Povejte, kaj je notranja, zunanja in kaj metaforična konsistentnost.

Rešitev:

- Notranja konsistentnost je konsistentnost z drugimi deli vmesnika znotraj iste aplikacije.
- Zunanja konsistentnost je konsistentnost z drugimi vmesniki na isti platformi.
- Metaforična konsistentnost je konsistentnost z metaforo realnega sveta.

87. Kaj je notranja konsistentnost?

Rešitev: Notranja konsistentnost se nanaša na način izvajanja podobnih akcij in predstavitve podobnih informacij znotraj enega vmesnika. Podobne operacije naj bi se izvajale na podoben način, podobne informacije pa naj bi bile predstavljene na podoben način. Notranja konsistentnost se nanaša na podobnosti znotraj istega vmesnika. Notranjo konsistentnost je lažje zagotoviti kot zunanjo konsistentnost, saj je težko zagotoviti, da bi različne razvojne

ekipe za različne aplikacije sledile istim oziroma podobnim smernicam pri implementaciji akcij in predstavitvi podatkov.

88. Kaj je zunanja (eksterna) konsistentnost? Kako se notranja konsistentnost razlikuje od zunanje?

Rešitev: Zunanja konsistentnost se nanaša na način, kako se podobne akcije izvajajo v različnih aplikacijah. To pomaga uporabnikom pri prenašanju veščin, naučenih na eni aplikaciji, na druge aplikacije, ki jih bodo mogoče uporabljali v prihodnosti, zunanja konsistentnost pa je posledično izjemno pomembna pri različnih verzijah istega proizvoda. Primeri zunanje konsistentnosti so na primer bližnjice za pogosta opravila v raznih urejevalnikih besedila (kopiraj, prilepi), ki so med različnimi urejevalniki pogosto enake. To je v kontrast notranji konsistentnosti, ki lahko zagotovi, da se podobne akcije opravljajo na podoben način oziroma da se podatki predstavljajo na podoben načine skozi celotno aplikacijo, ne zagotavlja pa podobnosti izvajanja akcij in predstavitve podatkov tudi v drugih aplikacijah.

89. Pri uporabi metafor je pomembna tudi metaforična konsistentnost. Kaj je to in ali se jo lahko kdaj tudi prelomi?

Rešitev: Metaforična konsistentnost se nanaša na izgled in obnašanje objekta, ki sta konsistentna z metaforo, uporabljeno za predstavitev. Problem pri uporabi metafor je, da se lahko pogosto zgodi, da metafora ni izbrana primerno, zato objekt že v osnovi ne more vsebovati lastnosti, ki jih metafora komunicira uporabniku (npr. škatla za CD kot metafora za predvajalnik glasbe), pogosto pa lahko objekt, ki je predstavljen z metaforo, ponuja več zmožnosti (npr. pisalni stroj kot metafora za urejevalnik besedila). Vztrajanje pri metaforični konsistentnosti je tako smiselno do neke točke, pogosto pa se zgodi, da je potreben prelom pri tej konsistentnosti.

90. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Pomoč in dokumentacija". Zakaj je ta princip pomemben, če vemo, da uporabniki ne berejo priročnikov.

Rešitev: Pri uporabi vmesnikov je potrebno upoštevati, da uporabniki ne berejo priročnikov, zato je potrebno vmesnik izboljšati v smeri njihovih opravil. Vendar pa se lahko kljub dobremu uporabniškemu vmesniku zgodi, da se uporabnik v nekem trenutku ne znajde ali pa išče specifične odgovore. V takem primeru sta takojšnja pomoč znotraj aplikacije in priročnik nujna. Pomoč mora omogočati lahko iskanje, biti mora v kontekstu opravila, ki ga uporabnik izvaja, opisati mora naloge, ki jih mora uporabnik izvesti, in biti

mora konkretna ter kratka.

91. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Uporabnikov nadzor in svoboda". Razložite ta princip in podajte primere upoštevanja tega principa za različne podobe ali akcije.

    Rešitev: Ta princip govori o tem, da mora biti vmesnik načrtan tako, da uporabnik določa, kaj in kdaj se bo zgodilo. Ker se v grafičnih uporabniških vmesnikih akcije oziroma operacije zgodijo kot posledica uporabnikove interakcije, se ta princip nanaša predvsem na to, da morajo biti operacije prekinljive. Tako mora biti vedno viden izhod iz vmesnika oziroma trenutne operacije, ki se izvaja v vmesniku. V okviru dialogov to pomeni, da morajo imeti vsi dialogi, pri katerih je to smiselno, možnost preklica akcije, ki je povzročila pojavitev dialoga. V okviru akcij, ki ne povzročijo pojavitve dialoga, pa je potrebno implementirati možnost razveljavitve akcije (nekaterih akcij se sicer ne da razveljaviti). Prav tako pa naj bo mogoče prekiniti dolge akcije brez posledic za integriteto podatkov.

92. Eden izmed Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Vidljivosti statusa sistema". Na katere aspekte uporabniškega vmesnika pa se nanaša ta princip?

    Rešitev: Ta princip se nanaša na:

    - tekoče spremembe, ki se morajo pojaviti kot neposreden odziv na neko uporabnikovo akcijo, kot je na primer klik na določen gumb, prikaz uporabnikovega izbora, prikaz informacije v statusni vrstici, povezane z nalogo;
    - vidljivost akcij, ki so v nekem trenutku možne, in vključuje na primer značke za izvlečne menije, teksture, ki omogočajo "prijemanje", različne oblike kurzorja miške, poudarjanje gradnikov, ki so pod miško, ter akcijo povleci-spusti;
    - vidljivost ukazov v obliki menijev, namigov, oziroma samorazkritij;
    - vidljivost načinov, pri katerih imajo akcije drugačen pomen;
    - vidljivost stanja med navigacijo, ki omogoča prikaz trenutnega položaja in mogoče poti.

93. Eden izmed Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Vidljivosti statusa sistema". En aspekt tega principa pa je tudi povratna informacija. Kaj ta aspekt predstavlja in kako ga lahko vidimo v tipičnem vmesniku? Kakšne vrste povratne informacije poznamo?

Rešitev: Ta aspekt predstavlja, kako se sistem odzove na uporabnikovo interakcijo z njim. Ko uporabnik manipulira s sistemom, se mora ta sistem tudi temu primerno odzvati. Pri pritisku na gumb, se izgled gumba spremeni tako, da je videti, kot da bi bil gumb pritisnjen in nato spuščen. V primeru manipulacije z drsniki se položaj drsnika spreminja. Ob pritisku na tipko se izpiše ustrezna črka v besedilnem območju. Obstajata nizkonivojska povratna informacija, kjer podoba sama poskrbi za povratno informacijo (pritisk na gumb), in visokonivojska povratna informacija, ki predstavlja končni rezultat uporabnikove interakcije (na primer naloženo novo spletno stran).

94. Eden izmed Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Vidljivosti statusa sistema". Zakaj je pomembno, da je uporabnik obveščen o stanju sistema?

Rešitev: Pogosto se zgodi, da je človekova pozornost obrnjena vstran od področij, ki se uporabljajo za obvestila o stanju, vendar je kljub temu potrebno obveščanje, saj lahko to prepreči napake ali pa jih vsaj omili. Pogosti napaki, do katerih pride med vnašanjem besedila, se nanašata na spremenjen način vnašanja besedila: velike/male črke in pa vstavljanje/prepisovanje. Čeprav uporabnikova pozornost ni direktno usmerjena v iskanje sporočil o statusu sistema, pa morajo ta obstajati zato, da lahko uporabnik vedno in hitro najde informacijo o stanju.

95. Eden izmed Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Vidljivosti statusa sistema". Vidik o povratni informaciji znotraj tega principa govori, da je potrebno zagotoviti povratno informacijo. Zakaj je potrebno zagotoviti povratno informacijo? Kakšna povratna informacija je potrebna pri različnih odzivnih časih sistema?

Rešitev: Povratno informacijo je potrebno zagotoviti zato, da je uporabnik obveščen o tem, da je vmesnik zaznal uporabnikov vnos in da izvaja procesiranje na podlagi tega vnosa. V primeru, da to ne bi bilo narejeno, uporabnik ne bi vedel, ali se operacija izvaja ali ne. Zaradi tega bi lahko ponovno zahteval isto akcijo, kar bi bilo lahko zamudno, lahko pa bi vodilo tudi v napake.
Povratna informacija, ki se mora uporabniku posredovati, se razlikuje od pričakovanega časa, potrebnega za odziv:

- < 0.1 s: uporabnik ne opazi zamika, če se rezultat akcije prikaže v uporabnikovem fokusu; v tem primeru ni potrebna nobena dodatna povratna informacija;
- $0.1 - 1$ s: uporabnik opazi zamik; v tem primeru zadostuje nizkonivojska povratna informacija, na primer sprememba izgleda gumba;

- $1-5$ s: zamik je že opazen; v tem primeru je potrebna dodatna povratna informacija, ki je lahko v obliki indikatorja zasedenosti;
- > 5 s: zamik je izrazit; v tem primeru je potrebna dodatna povratna informacija, ki uporabnika obvesti o napredku naloge, na primer v obliki indikatorja napredka.

96. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je "Fleksibilnost in učinkovitost". Naštejte pristope, s katerimi lahko zadostimo temu principu.

Rešitev: Pristopi, s katerimi omogočimo fleksibilnost in učinkovitost sistema, so:

- lahko naučljive bližnjice za pogoste ukaze: mnemoniki in pospeševalniki; okrajšave za ukaze;
- kopičenje ukazov: uporaba skriptnih datotek; združevanje elementov s podobnimi lastnostmi (primer: družine slogov v urejevalnikih besedil);
- zagotovitev predvidevanja uporabnikovih potreb: privzete nastavitve; zgodovina; pričakovanja.

97. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Izogibanje napakam". Razložite ta princip in podajte pristope, ki jih lahko uporabite za izpolnitev tega principa.

Rešitev: Pri uporabi vmesnikov je mogoče pričakovati, da bodo uporabniki delali napake. Namesto da bi od uporabnikov zahtevali, naj ne delajo napak, moramo vmesnik načrtati tako, da jih uporabniki ne morejo delati. Tako je smiselno, da se vnosna polja, ki zahtevajo vnos preko tipkovnice, zamenjajo s polji za izbiro, ki so manj podvržena napakam. Namesto ukaznega jezika je boljša uporaba menijev in form. Ukazi, ki v nekem trenutku niso mogoči, naj bodo onemogočeni. Dobro je vgraditi funkcionalnosti, ki omogočajo zaščito uporabnikovega dela, kot sta na primer "razveljavi" in pa "avtomatsko shranjevanje".

98. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Izogibanje napakam". Naštejte tri tipične vrste napak.

Rešitev: Tipične vrste napak so:

a) napaka v opisu, kjer pride do napake zaradi napačnega ali nepopolnega opisa posamezne akcije;

b) napaka zaradi istega začetka akcij, kjer pride do napake zato, ker imata dve različni akciji enak začetek (enako predpono);

c) napaka zaradi načina, kjer pride do napake, ker ima ista akcija v drugem načinu drugačen učinek, npr. gumb za velike črke ali pa način vstavljanja oziroma prepisovanja.

99. Naštejte pristope pri gradnji uporabniških vmesnikov, ki podpirajo princip "Raje prepoznaj, kot si zapomni".

Rešitev: Pristopi, ki podpirajo princip "Raje prepoznaj, kot si zapomni", so:

- uporabljaj menije, ne ukazni jezik;
- uporabljaj kombinirane izvlečne liste, ne tekstovna polja;
- uporabljaj generične ukaze, kjer je to mogoče, kot so na primer: Odpri ("Open"), Shrani ("Save"), Kopiraj ("Copy") in Prilepi ("Paste");
- vse potrebne informacije naj bodo vidne.

100. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Javljanje napak, diagnoza, reševanje". Naštejte nekaj vidikov, na katere moramo biti pozorni pri tem principu.

Rešitev: Pri tem principu moramo biti pozorni na:

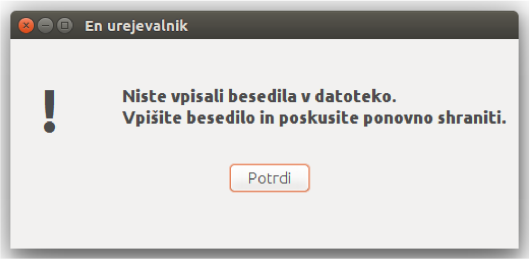
- bodi natančen, daj dobra obvestila o napakah;
- skrivaj tehnične podrobnosti (razen na željo);
- daj konstruktivno pomoč;
- bodi prijazen.

101. Eden od Nielsenovih principov načrtovanja uporabniških vmesnikov je tudi "Estetika in minimalistično načrtovanje". Za dosego preprostosti vmesnika se lahko uporablja tudi pristop večkratne vloge gradnikov. Razložite ta pristop in podajte kakšen primer.

Rešitev: Večkratna vloga gradnikov je pristop, pri katerem ima gradnik poleg svoje primarne vloge še neko sekundarno vlogo. S tem pristopom lahko zagotovimo potrebno funkcionalnost vmesnika z manj gradniki, kot bi jih sicer potrebovali. Primeri večkratne vloge gradnika vključujejo razne podobe orodij za gradnjo vmesnika, na primer drsnik, ki služi tako za premikanje vidne odprtine kot tudi za indikator položaja in velikosti vidne odprtine. Še en primer je vnosno polje, ki služi za vnos informacije, uporabimo pa ga lahko tudi kot polje, kjer se v primeru, če uporabnik še ni nič vnesel, nahaja na primer opis namembnosti polja ali pa opis formata podatkov za vnos.

102. Dialog na sliki 6.1 prikazuje primer obvestila, ki se je prikazalo v nekem urejevalniku starejše verzije. Ta dialog uporabnika obvešča, da ne more

shraniti prazne datoteke. Katerega izmed Nielsenovih principov načrtovanja krši ta dialog in kako to odpraviti?



Slika 6.1 Primer dialoga z obvestilom, ki se je prikazal uporabnikom urejevalnika starejše verzije.

Rešitev: Ta dialog krši Nielsenov princip "Uporabnikov nadzor in svoboda", saj ne dovoljuje uporabniku shraniti prazne datoteke, kar je popolnoma veljavna in varna operacija. Ena možna rešitev je, da se tak dialog odstrani. Druga možna rešitev pa je, da se dialog spremeni tako, da uporabnika opozori na to, da shranjuje prazno datoteko, vendar mu kljub temu dovoli nadaljevati s to operacijo (doda se še en gumb, s katerim se omogoči nadaljevanje zahtevane akcije).

103. Izberite najboljšega izmed odgovorov.
Primeri aplikacij, ki pomagajo reducirati uporabo spomina, so:

a) koledarji,

b) kalkulatorji/računala,

c) programi za vizualizacijo informacij,

d) vse zgoraj našteto.

Rešitev: Pravilen odgovor je odgovor d) "Vse zgoraj našteto". Vsi ti programi uporabniku nudijo vizualno predstavitev in manipulacijo s podatki. S tem ti programi tudi sledijo Nielsenovemu principu "Raje prepoznaj, kot si zapomni".

104. Izberite najboljšega izmed odgovorov.
Da si bo uporabnik lahko zapomnil dizajn vašega vmesnika, morate:

a) zagotoviti priročnike, ki so lahko razumljivi;

b) zagotoviti priročne bližnjice za pomembna opravila;

c) dizajnirati za prepoznavanje;

d) nič od zgoraj naštetega.

Rešitev: Pravilen odgovor je odgovor c) "dizajnirati za prepoznavanje". Priročniki so lahko zelo koristni, vendar jih uporabniki le redko uporabijo, in še to takrat, ko so v stiski. Bližnjice so pomembne, saj omogočajo hitro izvedbo akcij za izkušene uporabnike, vendar pa jih povprečni uporabniki le redko uporabljajo. Če uspe načrtovalcu narediti vmesnik, ki omogoča uporabo prepoznavanja, torej izbiro in ne tipkanja, pa bo uporabniku to omogočalo tudi enostavno pomnjenje uporabniškega vmesnika.

105. Izberite najboljšega izmed odgovorov.
Uporabniki postanejo nezadovoljni z vašo aplikacijo, ker:

a) aplikacija ne deluje tako, kot jo uporabnik dojema;

b) pojavlja se preveč sporočil o napakah;

c) izgled vmesnika je pokroviteljski;

d) vse zgoraj našteto.

Rešitev: Pravilen odgovor je odgovor d) "Vse zgoraj našteto". Če je avtor uporabil napačno metaforo oziroma jo uporabnik ne razume, lahko to privede do tega, da si bo uporabnik napačno predstavljal delovanje aplikacije in je ne bo znal uporabljati, zaradi česar bo nezadovoljen. Prav tako bo uporabnik nezadovoljen tudi zaradi preveč sporočil o napakah, bodisi upravičenih, kar pomeni, da je z aplikacijo nekaj narobe, bodisi nepotrebnih oziroma neupravičenih zaradi napak, ki jih lahko obdela aplikacija brez obveščanja uporabnika, saj to odvrača pozornost in zmanjšuje učinkovitost. Pri vmesnikih pa je pomembno, da so preprosti za uporabo, vendar ne smejo izgledati in se obnašati pokroviteljsko do uporabnika. To je do uporabnika žaljivo, poleg tega pa lahko še povečuje uporabnikovo frustracijo ob morebitni težavi pri ravnanju z uporabniškim vmesnikom.

106. Izberite najboljšega izmed odgovorov:
Da uporabniku pomagamo prepoznati vmesnik, namesto da bi si ga moral zapomniti, lahko:

a) zagotovimo podroben priročnik z vizualnimi primeri;

b) zagotovimo veliko interaktivnih možnosti, ki jih uporabnik želi v sistemu;

c) zagotovimo seznam bližnjic, ki jih uporabniki lahko uporabijo med delom s sistemom;

d) nič od zgoraj naštetega.

Rešitev: Pravilen odgovor je odgovor d) "nič od zgoraj naštetega". Tudi če zagotovimo podroben priročnik, je to za uporabnika izhod v sili, ki mu ne bo pomagal pri prepoznavanju in uporabi vmesnika. Interaktivne možnosti so sicer zaželene v aplikacijah, vendar ravno tako ne olajšajo prepoznavanja vmesnika. Bližnjice pa so idealne za izkušenega uporabnika, ki veliko dela z vmesnikom, vendar so bližnjice olajšava za uporabnikovo delo, ki se je mora naučiti.

7 Navodila načrtovanja uporabniških vmesnikov

107. Kaj so navodila za načrtovanje uporabniških vmesnikov?

Rešitev: To so pravila (ali vsaj močni namigi), ki urejajo večino vidikov načrta uporabniškega vmesnika. Navodila so bila testirana in ovrednotena, za njih pa je bilo tudi ugotovljeno, da zagotavljajo konsistenten in učinkovit rezultat.

108. Razložite razliko med principi in navodili za načrtovanje uporabniških vmesnikov.

Rešitev:

- Principi so visokonivojski koncepti za načrtovanje uporabniških vmesnikov. Nastali so na osnovi človekovega mentalnega modela ter njegovih fizičnih in psihičnih lastnosti.
- Navodila izhajajo iz principov in so dejanski napotki oziroma pravila, kako graditi uporabniške vmesnike.

109. Naštejte dele vzorca model-pogled-nadzornik (angl. "Model-View-Controller", MVC) in podajte osnovno nalogo vsakega.

Rešitev:

- Model: opisuje podatkovni in procesni vidik aplikacije.
- Pogled: zadolžen je za grafično upodabljanje gradnika in osveževanje.
- Nadzornik: interpretira zahteve uporabnika (interpretira dogodke) in preko modela poskrbi za izbiro ustreznega pogleda.

110. Kaj opisujejo gestalt principi zaznavanja?

Rešitev: Gestalt principi zaznavanja opisujejo, kako opazovalci različne predmete, ki se nahajajo v vizualnem polju, združujejo v skupine ali v celoto, če so posamezni opazovani elementi organizirani na določen način.

111. Naštejte in na kratko opišite gestalt principe zaznavanja za grupiranje elementov v vizualnem polju.

Rešitev: Gestalt principi so:

- bližina/grupiranje - elemente ki so bližje skupaj, se zazna kot grupirane;
- podobnost - elemente, ki so podobni oziroma si delijo nekatere vizualne karakteristike, se zazna kot, da spadajo skupaj;
- zveznost - pri elementih, ki so sestavljeni iz posameznih segmentov, se posamezne segmente, ki si zvezno sledijo, zazna kot en sam zvezen segment;
- zaprtje - elemente, ki niso v celoti narisani, se zazna kot celoto;
- površina - pri prekrivanju dveh elementov bo manjši zaznan kot slika na večjem elementu; celoten konstrukt se ne zazna kot element z luknjo;
- simetrija - če obstaja več različnih možnih razlag oziroma opisov postavitve elementov, se bo zaznala tista, ki ponuja večjo simetrijo.

112. Kaj je haptična interakcija?

Rešitev: Haptična interakcija je za razliko od slišne ali vizualne interakcije osredotočena na čutilo tipa. Nanaša se na uporabo računalnika s taktilnimi (čutilnimi) metodami, ki vključujejo naprave, ki zaznavajo premike telesa oziroma uporabnikov vnos in podajajo čutni odgovor uporabniku.

113. Kaj so zvezne in kaj so diskretne vhodne naprave?

Rešitev: Zvezne vhodne naprave so naprave, ki omogočajo zvezen vnos, kot je na primer miška (premik miške, v vsakem trenutku lahko miško zvezno premaknemo na nov položaj) ali sledilna ploščica ali zaslon na dotik. V nasprotju s tem pa so diskretne naprave tiste, kjer je mogoč diskreten vnos vrednosti (da/ne), kot je na primer tipkovnica, kjer lahko pritisnemo neko tipko.

114. Kaj so naprave za kazanje? Razložite sledilno kroglo kot napravo za kazanje.

Rešitev: Naprave za kazanje so vhodne naprave, ki se uporabljajo za določanje točke ali poti v 1-, 2- ali 3-dimenzionalnem prostoru, katerih karakteristike morajo biti upoštevane v povezavi s potrebami načrta ali uporabniškega vmesnika. Naprave za kazanje so: miška, sledilna ploščica, sledilna krogla, igralna palica, zaslon na dotik, sistem za sledenje pogleda z očmi.

Sledilna krogla je ena izmed možnih sledilnih naprav. Uporabnik lahko upravlja kurzor na zaslonu s premikanjem sledilne krogle, ki je podobna na glavo

obrnjeni miški. Prednost sledilne krogle je ta, da ne potrebujemo veliko prostora za premikanje, poleg tega tudi ni občutljiva na nenadne tresljaje. Slabost pa je, da je težja za uporabo kot na primer miška.

115. V smislu hitrosti in natančnosti primerjajte naslednji napravi za interakcijo: miško in zaslon na dotik.

Rešitev: Zaslon na dotik omogoča uporabniku večjo hitrost, saj je lažje premakniti prst roke na želeno lokacijo kot pa zapeljati kurzor z uporabo miške. Omogoča pa miška večjo natančnost, saj je lažje natančno pozicionirati kurzor z uporabo miške, kot pa pritisniti del zaslona na piksel natančno.

116. Opišite situacije, kjer bi morda raje uporabljali sledilno kroglo kot miško.

Rešitev: Sledilno kroglo bi raje uporabili na omejenem ali neurejenem prostoru, ker je ni potrebno premikati. Potreben prostor za sledilno kroglo je manjši kot za miško. Prav tako je njena uporaba primernejša na področjih, kjer so vibracije ali premiki, ki bi lahko premaknili miško.

117. Opišite nekaj situacij, ko ne bi uporabili menija v uporabniškem vmesniku za uporabnikov vnos ali izbiro.

Rešitev: Menija ne bi uporabili v primeru, ko je nemogoče oceniti vse vrednosti, ki bi jih lahko uporabnik vnesel. Meniji tudi niso primerni, ko podpirajo veliko število elementov, saj je obremenitev zaradi uporabe drsnikov prevelika. Meniji tudi niso primerni takrat, ko uporabnik nima dostopa do naprave za kazanje.

118. Opišite primere, kjer je uporaba miške priporočljivejša kot uporaba igralne palice.

Rešitev: Uporaba miške je priporočljivejša v primerih, kjer je potrebno natančno absolutno pozicioniranje. Igralna palica pa omogoča hitre relativne premike glede na trenuten položaj kurzorja. V praksi pa je miška primernejša za večino pisarniških aplikacij, medtem ko je igralna palica primernejša za igre in druge kontrolirane situacije, kot sta letalstvo in robotika.

119. Grafični uporabniški vmesniki so sestavljeni iz oken. Kaj pa omogočajo okna v grafičnih uporabniških vmesnikih?

Rešitev: Okna v grafičnih uporabniških vmesnikih omogočajo:

- prikaz različnih vrst informacij;
- prikaz različnih nivojev informacij;
- zaporednost opravil;
- vzporednost opravil;
- razbremenitev kratkotrajnega spomina;
- preglednost;
- večkratno predstavitev istega opravila.

120. Grafični uporabniški vmesniki so sestavljeni iz oken. Napišite, kako lahko okna razdelimo glede na vrsto opravila. Opišite posamezna okna.

Rešitev: Glede na vrsto opravila lahko okna razdelimo na:

- primarno okno oziroma okno aplikacije: to je okno, v katerem lahko uporabnik vedno dostopa do aktivnosti, ki so skupne za različna okna;
- sekundarna okna so okna, s katerimi lahko razširimo funkcionalnosti aplikacije: nimajo vrstičnega menija; obravnavajo en objekt aplikacije; sporočilna okna in dialogi (akcije z omejenim kontekstom).

121. Kateri so vidiki načrtovanja menijev?

Rešitev: Vidiki načrtovanja menijev so:

- prikaz opcij,
- organizacija opcij,
- grupiranje opcij,
- vrstni red opcij.

122. Naštejte vrste menijev.

Rešitev: Vrste menijev so:

- vrstični meni;
- izvlečni meni;
- kaskadni meni;
- dvižni (pojavni) meni;
- opcijski meni;
- orodna vrstica;
- meni z ikonami;
- krožni meni.

123. Za vsako vrsto menija napišite, kdaj bi ga izbrali glede na število izbir, glede na pogostost uporabe in sprememb, glede na vsebino menija in glede na prostor?

Rešitev: Izbira vrste menijev:

- vrstični meni: pogoste akcije, skupne akcije za sekundarna okna;
- izvlečni meni: pogoste akcije, akcije, opisane z besedilom, redke spremembe, 5-10 končnih izbir;
- kaskadni meni: preprostejši in preglednejši meni, končne izbire se medsebojno izključujejo, 1-2 kaskadi;
- dvižni (pojavni) meni: pogosta uporaba akcij v kontekstu, hitrejša uporaba, redke spremembe, malo prostora, 5-10 končnih izbir;
- meni s ponujeno izbiro (opcijski meni): pogosta raba ene akcije v kontekstu, druge izbire so redke, malo prostora, 5-10 končnih izbir;
- orodna vrstica: ikone, pogosto uporabljene končne izbire menijev;
- meni z ikonami: izbira aplikacije, za dosego posebnih funkcij aplikacije, akcije opisane z ikonami;
- krožni meni: do 10 končnih izbir, izbira aplikacije, verjetnosti uporabe za končne izbire so približno enake, najpogosteje za računalniške igre, manj pogosto za ostale aplikacije: je učinkovitejši kot na primer izvlečni ali dvižni meni, vendar razbija zunanjo konsistentnost.

124. Kaj je pri načrtovanju menijev priporočljivo vključevati za lažje razumevanje in enostavnost uporabe?

Rešitev: Priporočljivo je vključevati:

- navigacijo,
- ločevanje izbora in aktivacije,
- aktivacije izbir z uporabo tipkovnice (mnemoniki in pospeševlniki),
- indikatorje naslednikov (podmenijev),
- ločevalnike za grupiranje.

125. Kaj so gradniki uporabniškega vmesnika in kaj omogočajo? Kateri so sestavni deli gradnikov?

Rešitev: Gradniki uporabniškega vmesnika so vizualni objekti, s katerimi lahko uporabnik izvaja interakcijo. Omogočajo komunikacijo uporabnika z vmesnikom in manipulacijo podatkov. Sestavni deli gradnikov so:

- oznaka za identifikacijo;

- vsebina, ki vključuje podatke in drugo relevantno informacijo;
- lahko vsebuje kontrole za manipulacijo.

126. Primerjate med sabo množico stikal za več izbir in seznam. Za kaj se uporabljajo ti gradniki?

Rešitev: Množica stikal in seznam se uporabljata za nastavitev oziroma izbiro več možnosti. Prednosti množice stikal so: 1) da omogoča lahek dostop, 2) omogoča primerjavo in 3) je razumljiva. Zasede pa veliko prostora, zaradi tega lahko z njo predstavimo omejeno število možnosti. Prednosti seznama so: 1) da omogoča neomejeno število možnosti (prikaz drsnika), 2) omogoča primerjavo in 3) omogoča stalen prikaz. Slabosti pa so: 1) da zasede veliko prostora, 2) iskanje možnosti je zamudno, 3) zahteva veliko privajanja in 4) ima "krhko" izbiro.

127. Primerjate med sabo seznam in dvižni seznam. Za kaj se uporabljata ta gradnika?

Rešitev: Seznam in dvižni seznam se uporabljata za izbiro ene ali več možnosti. Gradnika sta si zelo podobna, s to razliko, da seznam omogoča stalen prikaz, medtem ko dvižni seznam omogoča prikaz na zahtevo. Dobre lastnosti obeh podob so, da omogočata neomejeno število možnosti ter prepoznavanje. Slabe lastnosti pa so, da podobi zahtevata veliko privajanja, iskanje možnosti je zamudno, prav tako pa imata obe podobi "krhko" izbiro. Dobra lastnost dvižnega seznama je, da ne zaseda veliko prostora, slaba pa je ta, da zahteva dodaten korak za prikaz seznama. Seznam ne zahteva dodatnega koraka, zato pa zasede veliko prostora.

128. Kdaj bi uporabili gumbe za izključujočo izbiro in kdaj gumbe za izbiro?

Rešitev: Gumbi za izključujočo izbiro se lahko uporabijo za menjavanje med dvema ali več stanji, ki so medsebojno izključujoča, lahko pa včasih tudi za prikaz in predstavljanje za statusno informacijo. Gumbe za izbiro lahko uporabimo takrat, ko izbire niso med seboj izključujoče.

129. Naštejte vidike, ki jih je potrebno upoštevati, ko se odločate med meniji oziroma gumbi.

Rešitev: Vidiki, ki jih je potrebno upoštevati, so:

- število ukazov: več kot 7 zahteva menije;
- kompleksnost ukazov: hierarhija zahteva menije;

- pogostost ukazov: pogoste akcije zahtevajo gumbe;
- povezanost ukaza z drugim ukazom: povezanost zahteva gumbe.

130. Izbirate gradnike za svojo aplikacijo. Za vnos nekega podatka ste izbrali krožno polje. Katere značilnosti vodijo v izbiro takega gradnika?

Rešitev: Krožno polje za vnos podatkov izberemo, ko od uporabnika zahtevamo vnos med seboj izključujočih se alternativ, pri tem pa so podatki, med katerimi uporabnik izbira, znani, zaporedni in napovedljivi. Krožno polje je primerno tudi takrat, ko želimo uporabniku omogočiti tudi tipkan vnos. Primeren je tudi, če nimamo dovolj prostora, pri tem pa so spremembe in uporaba redki.

131. Zakaj bi uporabili pisave brez serifov (angl. "sans-serif")?

Rešitev: Sans-serifne pisave ne vsebujejo pismenk ali dodatnih serifov oziroma zavihkov, katerih namen je vodenje očesa bralca vzdolž linije besedila. Posledica tega je, da besedilo dobro izstopa v diskretnih blokih, zato so te pisave primerne za uporabo v naslovih. Primere so tudi za uporabo v imenih ukazov in opcijah menijev v uporabniških vmesnikih, saj morajo ta besedila izstopati.

132. S stališča modela "look" (količina informacije, prikaz informacije) in s stališča modela "feel" (dinamičnost, manipulacija s tekstom) primerjaj naslednje gradnike uporabniških vmesnikov: meni, besedilno območje, oznaka in seznam. Napišite tudi vrstni red podob glede na količino informacij pri posamezni podobi ter na dinamičnost posamezne podobe.

Rešitev: Vrstni red podob glede na količino informacije in glede na dinamičnost podobe:

- "look": količina informacije raste v smeri: oznaka, meni, seznam, besedilno območje;
- "feel": dinamičnost raste v smeri: oznaka, meni, seznam, besedilno območje.

Primerjava podob glede na modela "look" in "feel":

- **Oznaka:**
  - "look" - prikaz informacije: tekst, ikona, permanenten prikaz;
  - "feel" - manipulacija s tekstom: ni manipulacije nad tekstom.
- **Meni:**

  - "look" - prikaz informacije: tekst, več opcij, permanenten prikaz ali prikaz na zahtevo;
  - "feel" - manipulacija s tekstom: izbira, dinamičnost.
- **Seznam:**
  - "look" - prikaz informacije: tekst, več opcij, permanenten prikaz ali prikaz na zahtevo, lahko drsniki;
  - "feel" - manipulacija s tekstom: izbira, dinamičnost, dodam, odvzamem vrstico.
- **Besedilno območje:**
  - "look" - prikaz informacije: tekst, permanenten prikaz, drsniki;
  - "feel" - manipulacija s tekstom: polna manipulacija, dodam, odvzamem.

133. Naštejte vidike aranžiranja gradnikov za interakcijo, ki izhajajo iz principa načrtovanja "Estetika in minimalistično načrtovanje".

Rešitev: Vidiki aranžiranja gradnikov za interakcijo, ki izhajajo iz principa "Estetika in minimalistično načrtovanje", so:

- velikost in število oken;
- logična organiziranost;
- količina informacij;
- balansiranje;
- grupiranje;
- orientacija in poravnavanje.

134. V uporabniškem vmesniku morate del gradnikov grupirati in jih ločiti od preostalih gradnikov vmesnika. Kaj lahko uporabite za grupiranje gradnikov vmesnika?

Rešitev: Elemente lahko grupiramo z uporabo:

- belih presledkov: prazen prostor med gradniki ali grupami (večji);
- okvirjev: vidni pravokotniki okoli posamezne grupe;
- glav: pojasnila za grupo;
- ločevalnikov: vidne navpične ali vodoravne črte med grupami;
- oznak: pojasnila za gradnik ali grupo.

135. Pri grupiranju elementov ste se odločili za uporabo belih presledkov. Razložite, kaj so beli presledki in zakaj ste se odločili za njihovo uporabo.

Rešitev: Beli presledki so prazen prostor okoli gradnika. Pravilna uporaba belih presledkov omogoča vizualno grupiranje gradnikov uporabniškega vmesnika. Beli presledki se lahko pogosto uporabljajo namesto ločevalnikov (črt) ali okvirjev za grupiranje gradnikov. Pri tem lahko z majhnimi belimi presledki med gradniki grupiramo te gradnike v grupo, z večjimi belimi presledki med grupami pa grupe ločimo med sabo. Zmanjševanje števila gradnikov (okvirji, ločevalniki) ima za posledico manj "natlačen" in bolj pregleden vmesnik. Tak vmesnik pa omogoča lažji dostop in hitrejše delo (Fittov zakon).

136. Napišite in razložite prednosti ter slabosti uporabe standardih podob grafičnih orodij pri implementaciji uporabniškega vmesnika.

Rešitev: Pri implementaciji uporabniškega vmesnika lahko uporabimo standardne podobe, ki jih nudijo razna orodja (npr. *Swing*, *JavaFX*, *GTK+*). Uporaba standardnih podob omogoča hitrejšo izgradnjo uporabniškega vmesnika, saj so že implementirane, tako da dodaten razvoj ni potreben. Standardne podobe tudi zagotavljajo večjo zanesljivost, saj so bile že testirane. Poleg tega so uporabniki standardnih podob in njihove uporabe že navajeni, tako da to ne zahteva dodatnega privajanja uporabnikov. Po drugi strani pa uporaba standardnih podob otežuje inovativnost vmesnika, saj se namesto novih podob uporabijo že narejene. Poleg tega je nabor standardnih podob omejen, uporaba takega nabora pa lahko vodi v manj uporaben vmesnik.

137. Naštejte in opišite vidika pri izbiri besedila v uporabniškem vmesniku.

Rešitev: Vidika sta:

- izbor besedila: preprosto, čisto in vljudno; kratke, pozitivne besede; besedila naj vsebujejo 40-60 znakov na vrstico; prazne vrstice med odstavki; alineje; kratki stavki, do 30 besed; vključuj primere, uporabljaj aktiv;
- aranžiranje besedila: uporabljaj robove, besedilo naj ne bo na robu; uporabljaj mrežno razvrščanje; poravnaj besedilo po levem robu.

138. Načrtujete alfanumerični prikaz. Naštejte barve ozadja, ki zadoščajo zahtevam po vidljivosti, kontrastu in harmoniji.

Rešitev: Barve, ki zadoščajo zahtevam po vidljivosti, kontrastu in harmoniji so črna, modra, sinja (angl. "cyan"), modro-rdeča (vijolična), (tudi bela).

139. Katere od svetlih barv so tudi hladne?

Rešitev: Svetle barve, ki so tudi hladne, so zelena, sinja (angl. "cyan"), modro-rdeča (vijolična).

140. Kaj je kontrast?

Rešitev: Kontrast se nanaša na zaznavne razlike vzdolž vizualnih dimenzij. To je informacija, ki je sprejeta na nivoju zaznavanja. Kontrast so nepravilnosti v načrtu, zaradi katerih nek element izstopa oziroma ki posredujejo neko informacijo. Večje nepravilnosti predstavljajo večji kontrast in zagotavljajo večjo ločljivost.

141. Naštejte vizualne spremenljivke, ki omogočajo dosego dobrega kontrasta.

Rešitev: Vizualne spremenljivke, ki omogočajo dosego dobrega kontrastam so:

- intenziteta,
- barva,
- oblika,
- položaj,
- orientacija,
- velikost.

142. Naštejte vizualne spremenljivke, ki omogočajo dosego dobrega kontrasta in nimajo naravno zaznavnega vrstnega reda vzdolž svojih vizualnih dimenzij.

Rešitev: Naravno zaznavnega vrstnega reda vzdolž svojih vizualnih dimenzij nimajo:

- barva,
- oblika,
- orientacija.

143. Naštejte pogovorna okna (dialoge) za obvestila.

Rešitev: Pogovorna okna za obvestila so:

- za potrditev,
- za informacijo,
- za opozorilo,

- za napake in kritične akcije,
- za vprašanja,
- delovni dialog.

144. Pri katerih aplikacijah je potreben zvok?

Rešitev: Aplikacije, pri katerih je potreben zvok, so:

- aplikacije, kjer so oči in pozornost vstran od zaslona;
- aplikacije, ki vključujejo nadzor nad procesom;
- aplikacije, namenjene uporabnikom s slabšim vidom.

145. Navodila načrtovanja uporabniških vmesnikov za aranžiranje grafičnih gradnikov za interakcijo sledijo principu preprostosti. Katere so tehnike za dosego preprostosti? Naštejte tudi navodila aranžiranja grafičnih gradnikov za interakcijo.

Rešitev: Tehnike za dosego preprostosti:

- redukcija,
- regularnost,
- večkratna vloga gradnikov.

Navodila za aranžiranja grafičnih gradnikov za interakcijo:

- balansiranje/enakomerna distribucija in simetrija,
- grupiranje,
- orientacija in poravnavanje.

146. Predpostavite, da načrtujete pogovorno okno (dialog). V kakšnih načinih (modalnost dialogov) je lahko dani dialog?

Rešitev: Načini dialogov (modalnost dialogov):

a) nemodalni način: nič na sistemu med življenjem dialoga ne zamrzne, uporabnik lahko nemoteno uporablja vse funkcionalnosti sistema;

b) modalnost okna: zamrzne okno, ki je generiralo dialog, uporabnik ne more uporabljati okna, ki je generiralo dialog, dokler ne odgovori na dialog, lahko pa uporablja preostala okna aplikacije in druge funkcionalnosti sistema;

c) modalnost aplikacije: zamrzne aplikacija, ki je generirala dialog, uporabnik je ne more uporabljati, dokler ne odgovori na dialog, lahko pa uporablja preostale funkcionalnosti sistema;

d) sistemski modalni način: zamrzne cel sistem, dokler dialog živi, dokler se ne odgovori na dialog, je možna samo interakcija z dialogom;

e) zaporedje modalnih dialogov: način čarovnika, možna je interakcija z aplikacijo preko zaporedja dialogov, druga interakcija z aplikacijo ni mogoča, mogoče pa je uporabljati preostale funkcionalnosti sistema.

147. Narediti morate meni, ki vsebuje možnosti "Nova", "Tiskaj", "Izhod", "Odpri", "Predogled" in "Shrani". Implementirajte meni, ki razločno in konsistentno predstavi te možnosti uporabniku.

Rešitev: Zaradi konsistentnosti z obstoječimi meniji je pomemben vrstni red možnosti, ki se pojavljajo v tem meniju. Tako je primeren vrstni red "Nova", "Odpri", "Shrani", "Predogled", "Tiskaj" in "Izhod". Poleg tega pa je zaradi preglednosti in zaradi grupiranja posameznih podobnih možnosti primerno, da uporabimo tudi ločevalnike, s katerimi ločimo možnosti za delo z datotekami ("Nova", "Odpri", "Shrani"), za ogled pred tiskanjem in tiskanje ("Predogled", "Tiskaj") ter za končanje programa ("Izhod"). Tak vrstni red tudi sovpada z vrstnim redom uporabe opcij. Primer takega menija je prikazan na sliki 7.1.

Slika 7.1 Prikaz možne implementacije menija, kot ga zahteva naloga.

148. Postavljeni ste pred nalogo, da naredite uporabniški vmesnik za vpisovanje dimnikarskih storitev. Ta vmesnik mora med drugim vsebovati tudi polje, v katerega mora uporabnik vnesti tiste mesece v letu, ko bi želel, da ga dimnikar obišče. Pri tem morate upoštevati, da zakon zahteva obisk dimnikarja vsaj dvakrat letno v primeru kurišča, ki uporablja biomaso (drva, sekanci, peleti in podobno), uporabnik pa seveda lahko naroči dimnikarja poljubno mnogokrat. Pri tem predpostavimo, da lahko dimnikar pride kvečjemu enkrat na mesec. Za realizacijo imate na voljo več podob: izvlečni meni, polje stikal, polje izključujočih stikal, kombiniran izvlečni seznam, vnosno polje in seznam. Za katero izmed naštetih podob bi se odločili?

Rešitev: Glede na to, da lahko pride dimnikar več kot enkrat letno, potrebujemo tako podobo, ki bo omogočala izbiro več kot ene možnosti. Zaradi tega ne moremo uporabiti podob, ki omogočajo izbiro samo ene možnosti, torej

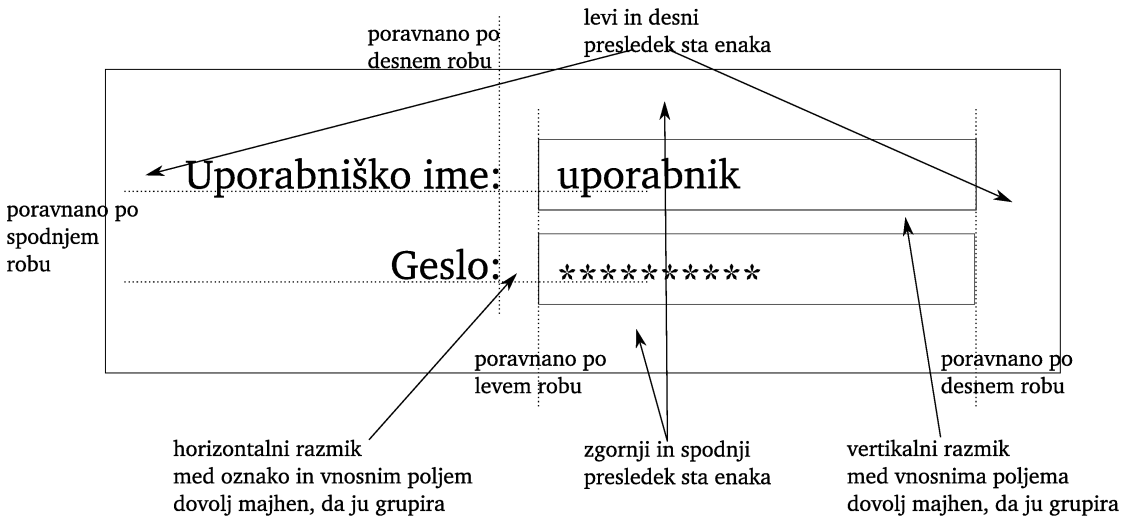
podob izvlečni meni, polje izključujočih stikal in kombiniran izvlečni seznam. Slabost vnosnega polja je ta, da je potrebno tipkanje, s čimer je mogoče vnašati napake, tako da ta podoba ni najprimernejša, prav tako pa bi lahko prišlo do težav pri različnih oblikah vnosa za mesec (npr. september, sept., 9) in pri ločevanju posameznih mesecev (uporaba podpičja, vejice, presledka). Polje stikal je primerna podoba, če imamo manj (do 10) izbir, drugače pa je najprimernejša podoba seznam, saj omogoča izbiro in ne zahteva tipkanja, je pa sicer zahtevnejši za uporabo.

149. Z uporabo navodil aranžiranja grafičnih gradnikov za interakcijo v okno spodaj načrtujte (narišite) dialog za prijavo, ki od uporabnika zahteva vnos uporabnikovega imena in njegovega gesla. Naštejte, katere podobe ste uporabili, ter utemeljite njihovo razvrstitev in velikost s stališča navodil aranžiranja grafičnih gradnikov za interakcijo, ki so: enakomerne distribucije in simetrije, grupiranja ter orientacije in poravnavanja.

Rešitev:

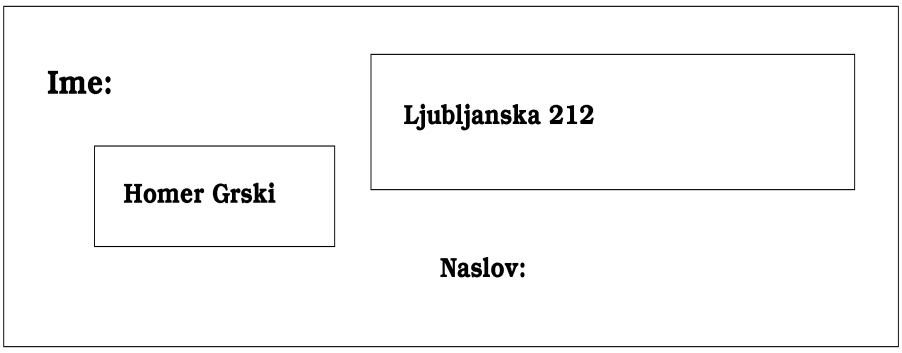
- Podobe: dve oznaki, polje za vnos, polje za vnos z masko;
- enakomerna distribucija in simetrija: levi in desni presledek sta enaka, zgornji in spodnji presledek sta enaka;
- grupiranje: oznaki sta levo, polji za vnos in vnos z masko desno, uporabniško ime zgoraj, geslo spodaj; beli presledki so levo in desno ter zgoraj in spodaj v pravem razmerju glede na vsebino dialoga; beli presledek med zgornjo in spodnjo vrstico je dovolj majhen in v pravem razmerju z okoliškimi belimi presledki, da grupira vrstici;
- orientacija: vertikalna;
  poravnavanje: v vsaki vrstici oznaka in vnosno polje oziroma polje za vnos z masko horizontalno glede na besedilo; oznake vertikalno levo, vnosno polje in vnosno polje z masko vertikalno levo in desno.

Možno aranžiranje takega dialoga je prikazano na sliki 7.2.

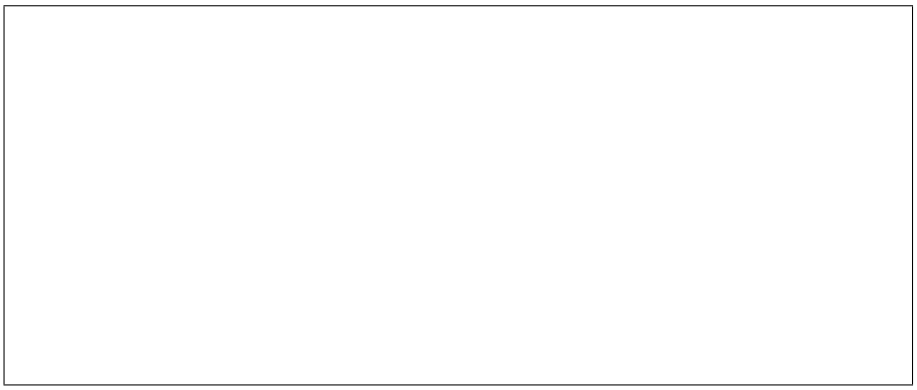


Slika 7.2 Možno aranžiranje dialoga za vnos uporabniškega imena in gesla.

150. Dialog, prikazan na sliki 7.3, ima štiri komponente. V prazen dialog spodaj pravilno aranžirajte te štiri komponente. Svojo razvrstitev utemeljite s stališča velikosti podob in s stališča navodil aranžiranja grafičnih gradnikov za interakcijo, ki so: enakomerne distribucije in simetrije, grupiranja ter orientacije in poravnavanja.



Slika 7.3 Dialog za vnos imena in naslova.



Rešitev:

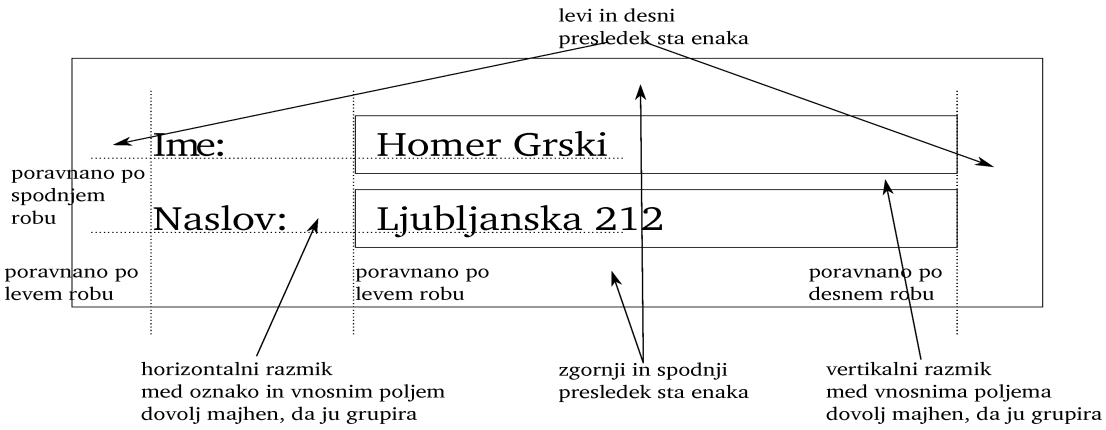
- Velikost: enaki velikosti polj za vnos, enaka velikost besedil (enaka pisava);
- enakomerna distribucija in simetrija: levi in desni presledek sta enaka, zgornji in spodnji presledek sta enaka;
- grupiranje: oznaki sta levo, polji za vnos desno, ime zgoraj, naslov spodaj; beli presledki so levo in desno ter zgoraj in spodaj v pravem razmerju glede na vsebino dialoga; beli presledek med zgornjo in spodnjo vrstico je dovolj majhen in v pravem razmerju z okoliškimi belimi presledki, da grupira vrstici;
- orientacija: vertikalna;
  poravnavanje: v vsaki vrstici oznaka in vnosno polje horizontalno glede na besedilo; oznake vertikalno levo, vnosni polji vertikalno levo in desno.

Možna rešitev za dialog za vnos imena in naslova uporabnika je prikazana na sliki 7.4.

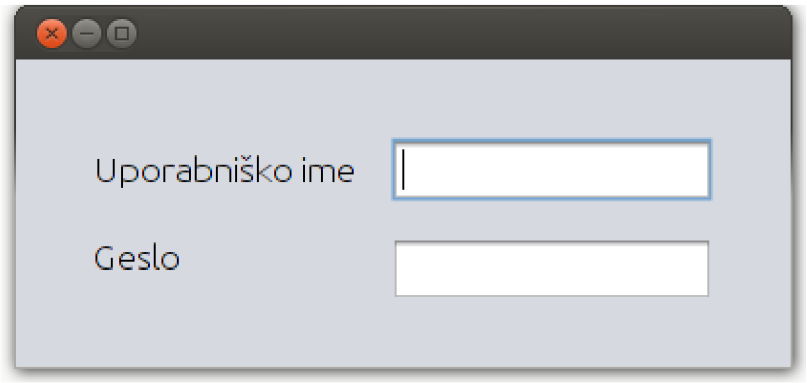
151. Z uporabo večkratne vloge gradnikov reducirajte spodnji dialog za vnos uporabniškega imena in gesla (slika 7.5).

Rešitev: Ena možnost reduciranja z uporabo dvojne vloge gradnikov je ta, da uporabimo vnosno polje za vnos imena in zakrito vnosno polje za vnos gesla hkrati tudi kot podobi, ki služita za prikaz vsebine oznak, ki sta prej pojasnjevali vnosni polji. V tem primeru imamo sedaj namesto štirih podob (dve oznaki, vnosno polje in zakrito vnosno polje) samo še dve podobi, in sicer vnosno polje ter zakrito vnosno polje. Ob tem pa vnosno polje in zakrito vnosno polje prevzameta tudi vlogo prikaza besedila v oznaki, tako kot je prikazano na sliki 7.6.

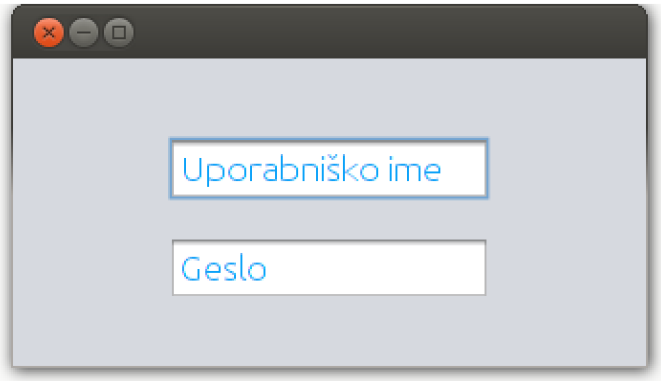
Uporaba vnosnega polja tako za prikaz besedila (namembnosti) ni problematična, saj je vnosno polje temu namenjeno. Pri tem bi vnosno polje implementirali tako, da hrani opis v polju toliko časa, dokler uporabnik



Slika 7.4 Možno aranžiranje dialoga za vnos imena in naslova uporabnika.



Slika 7.5 Prikaz dialoga za vnos uporabniškega imena in gesla.



Slika 7.6 Dialog za vnos uporabniškega imena in gesla, kjer je uporabljena večkratna vloga gradnikov.

ne začne vnašati besedila. Pri uporabi zakritega vnosnega polja za prikaz namigov pa se pojavi težava, saj zakrito vnosno polje prikazuje zakrite znake. Tako bi bilo tudi besedilo oziroma namembnost polja zakrita, kar pa ni tisto, kar bi želeli doseči. Nekatera novejša orodja že vsebujejo podobe, ki imajo med lastnostmi tudi možnost dodajanja pojasnjevalnega besedila (angl. "placeholder"), ki naj se prikaže, če ni še nič vnešeno. Veliko starejših orodij tega nima. Zaradi tega moramo v teh primerih to implementirati drugače oziroma moramo to funkcionalnost implementirati sami.

V Javi to za orodje *Swing* naredimo tako (koda, prikazana v listingu 7.1), da vnosno polje ali zakrito vnosno polje, ki ga uporabimo tudi za prikaz pojasnila, če ni uporabnik še nič vnesel, razširimo iz vnosnega polja oziroma iz zakritega vnosnega polja. Pri tem prepišemo metodo za risanje (*paintComponent(...)*) in dodamo metodo za nastavitev besedila, ki naj se prikaže, če ni uporabnik vnesel še nobenega besedila. S tem se izognemo prikazu besedila v vnosnem polju in s tem tudi zakrivanju znakov pojasnila, saj metoda *paintComponent(...)* riše preko vnosnega polja in ne vanj.

```
1  import java.awt.Color;
2  import java.awt.Graphics;
3  import javax.swing.JPasswordField;
4  import javax.swing.JTextField;
5
6  public class JPasswordPlaceholder extends JPasswordField{
7    private String pojasnilo;
8    private Color barva = new Color(11, 162, 249);
9
10   @Override
11   protected void paintComponent(Graphics g) {
12     super.paintComponent(g);
13     String geslo = new String(this.getPassword());
14     if (pojasnilo.length() == 0 || geslo.length() > 0){
15         return;
16     }
17     g.setColor(barva);
18     g.drawString(pojasnilo, getInsets().left, gd.
           getFontMetrics().getMaxAscent() + getInsets().top);
19   }
20
21   public void setPojasnilo(final String s) {
22     pojasnilo = s;
23   }
24 }
```

Listing 7.1 Razred, ki razširja zakrito vnosno polje in omogoča prikaz namigov.

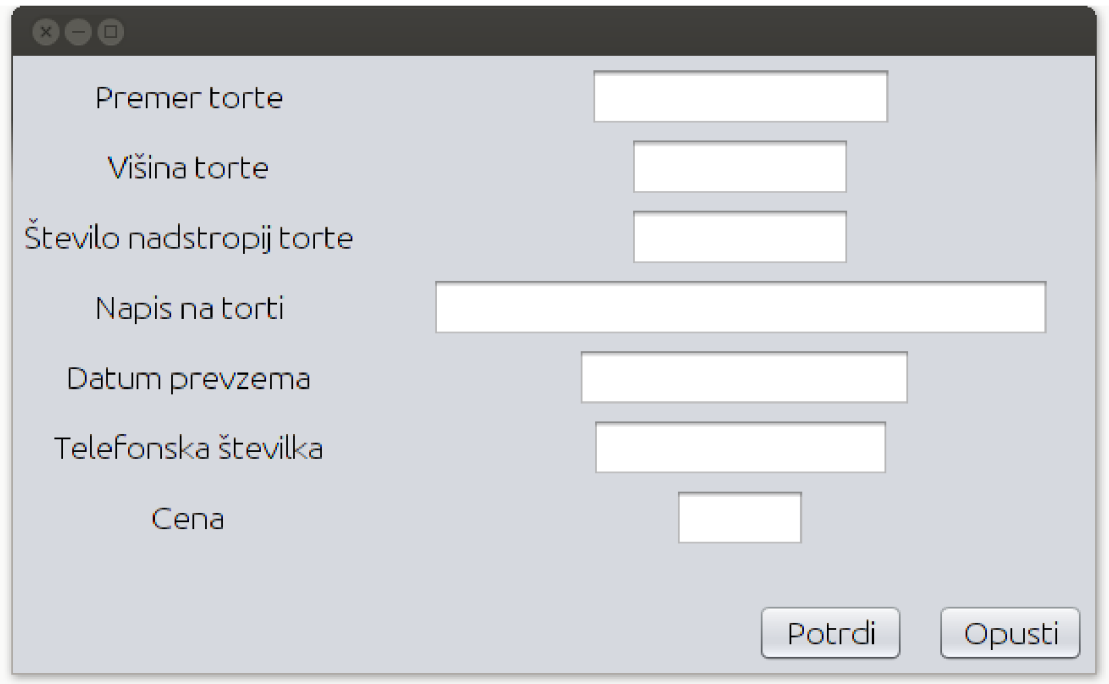
Ta razred razširja zakrito vnosno polje na tak način, da je v primeru, ko uporabnik ni še nič vnesel v polje, v podobi izpisano besedilo, ki smo ga nastavili z metodo *setPojasnilo(...)* (vrstice od 21 do 23). Pri tem ne pišemo neposredno v tekstovni del podobe, temveč besedilo izrišemo v grafični del komponente (vrstice od 10 do 19). Pojasnilo tudi poudarimo tako, da ga izrišemo v drugi barvi (v vrstici 8 definiramo barvo, ki jo uporabimo za izpis, v vrstici 17 nastavimo barvo pri izpisu, v vrstici 18 pa izpišemo besedilo v grafični del komponente). Konstruktorja, ki bi nastavil to besedilo, nismo implementirali, to lahko naredi bralec sam.

V primeru implementacije te funkcionalnosti za vnosno polje pa bi morali razširiti razred za vnosno polje (razred *JTextField*) in v metodi *paintComponent(...)* preveriti, če je vpisano besedilo dolgo nič znakov (metoda *getText()*, ki vrne niz, in ne metoda *getPassword()*, ki vrne polje znakov), vse ostalo pa je identično.

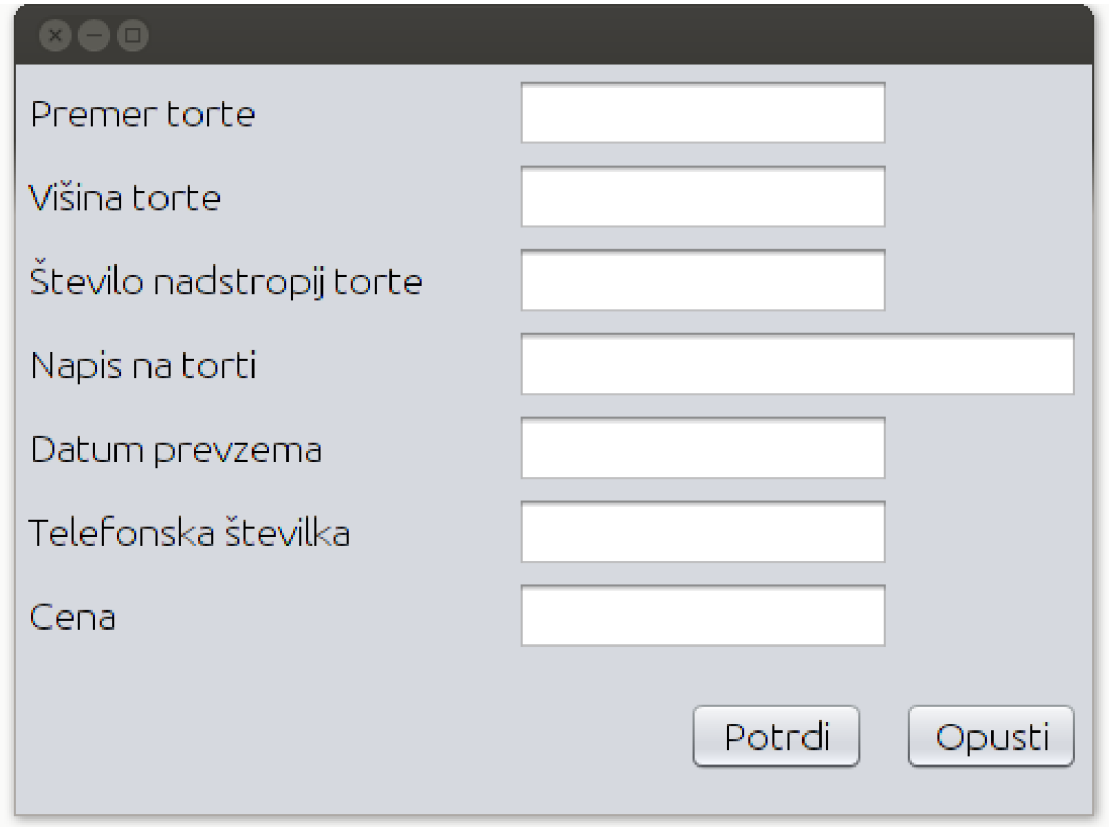
152. Za dialog na sliki 7.7 spremenite poravnave oznak tako, da bodo poravnane vertikalno levo, za vnosna polja pa spremenite dimenzije, kjer je to smiselno, in spremenite poravnave, da bodo poravnana vertikalno levo in desno, kjer je to mogoče.

Rešitev: Slika 7.8 prikazuje vmesnik, kjer so oznake vertikalno poravnane levo, vnosna polja vertikalno tako levo in desno, razen za vnos besedila na tori, kjer je besedilo lahko dolgo. V tem primeru ni smiselno preveč skrajševati te podobe, saj bi bilo zaradi tega besedilo lahko tudi težko berljivo.

Večino podob bi bilo sicer smiselno zamenjati za podobe, ki uporabniku omogočajo izbiro in ne prostega vnosa, saj tako onemogočimo neveljavne vnose, kot je na primer torta premera 25 metrov. Mogoče bi bilo tudi, da se vnosno polje zamenja za besedilno območje ali kakšno drugo podobo, ki omogoča vpis več kot samo ene vrstice besedila, vendar v primeru naročila tort takšna zamenjava ni smiselna, saj besedila zaradi omejitve prostora na tortah ne smejo biti predolga.

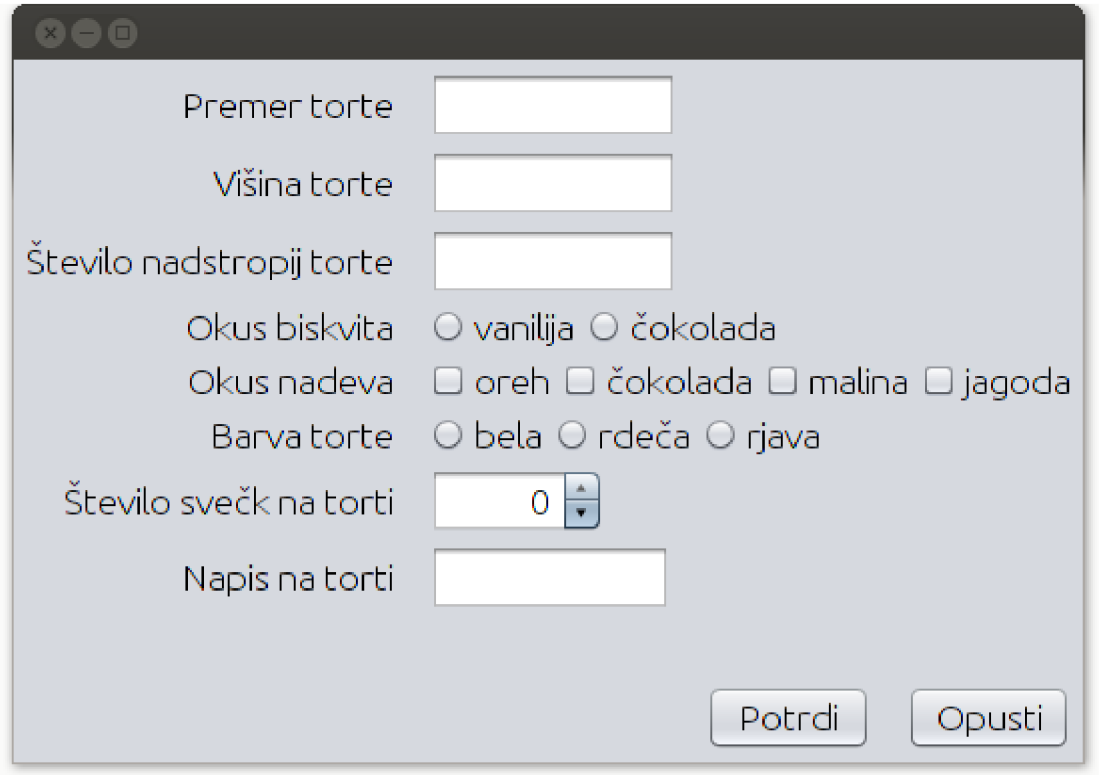


Slika 7.7 Prikaz dialoga za naročilo tort z neprimerno poravnanimi oznakami in vnosnimi polji.



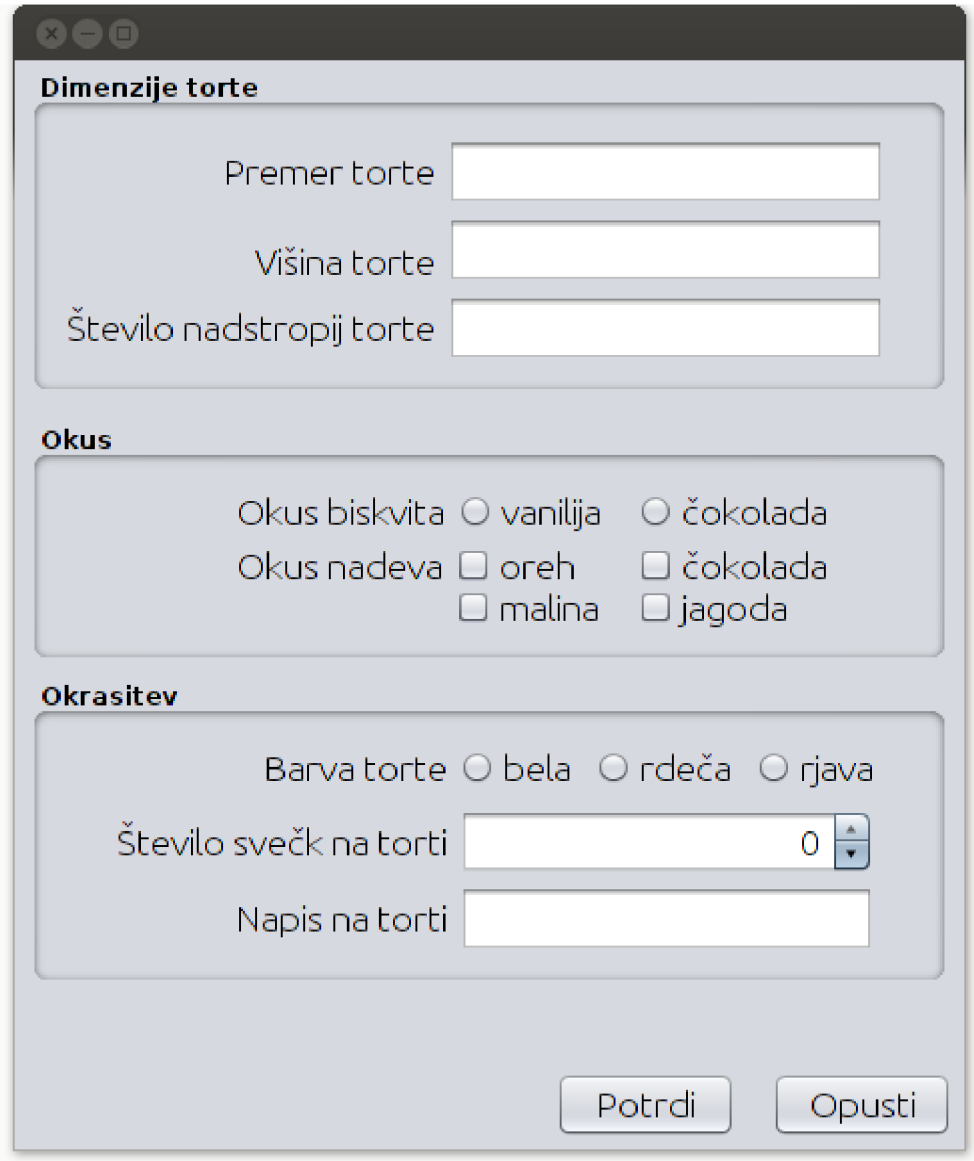
Slika 7.8 Prikaz dialoga za naročilo tort s poravnanimi oznakami in vnosnimi polji.

153. Z uporabo pravil grupiranja boljše načrtajte uporabniški vmesnik, ki je namenjen sprejemanju naročil za torto in je prikazan na sliki 7.9. Pri tem zanemarite dejstvo, da ponekod morda niso izbrane najprimernejše podobe.



Slika 7.9 Dialog za sprejemanje in vpisovanje naročil za torto.

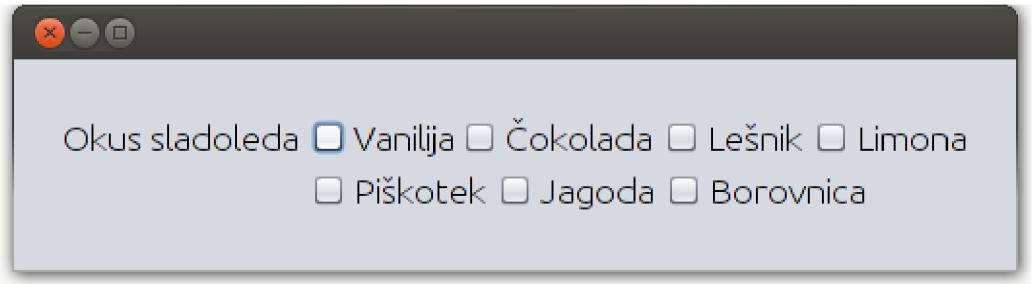
Rešitev: Pri prvotnem vmesniku so možnosti, ki jih ima uporabnik na izbiro, sicer v pravem vrstnem redu, vendar zaradi neuporabe belih presledkov in okvirjev niso dovolj pregledne. Za boljšo preglednost možnosti, ki jih ima uporabnik na voljo, bi bilo mogoče vse možnosti razdeliti na nekaj sklopov. Takih delitev je lahko več, ena pa je na primer: 1) dimenzija torte, kamor spadajo vse tiste možnosti, ki jih ima uporabnik pri določitvi fizičnih karakteristik torte; 2) okus, kamor spadajo vse tiste možnosti, pri katerih lahko uporabnik vpliva na okus torte, se pravi biskvita in nadeva; 3) okrasitev, kamor spadajo tiste možnosti, pri katerih lahko uporabnik vpliva na končni izgled torte. Vsak izmed teh sklopov pa se lahko loči od ostalih sklopov z uporabo okvirjev. Znotraj posameznega sklopa med posameznimi možnostmi dodamo enako količino belih presledkov. Primer takega vmesnika se nahaja na sliki 7.10.



Slika 7.10 Urejen dialog za sprejemanje naročil za torto.

154. Slika 7.11 prikazuje del dialoga za izbiro okusa naročenega sladoleda, kjer poteka izbira okusov preko polja stikal. Izboljšajte aranžiranje polja stikal. Po potrebi lahko kakšno izmed podob tudi dodate ali jo zamenjate s kakšno, za katero menite, da je primernejša.

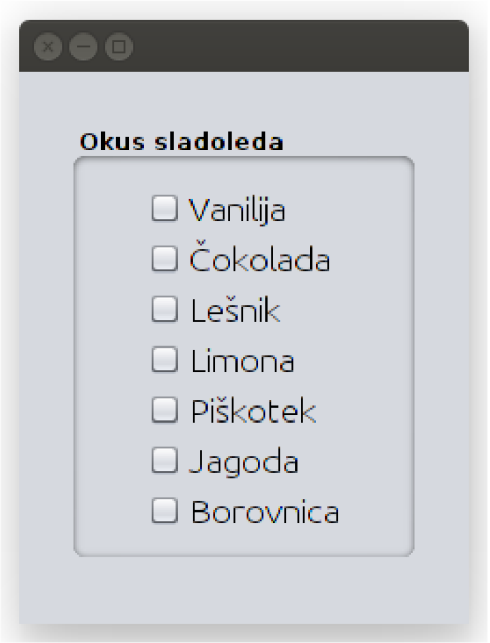
Rešitev: Najprej se je potrebno odločiti za orientacijo podob. Ker je možnih okusov še kar precej, se lahko odločimo za uporabo vertikalne orientacije in ne horizontalne, saj bi bil dialog zaradi tega lahko preširok. Tako se bodo posamezni okusi nahajali eden pod drugim. Za boljše grupiranje lahko oznako



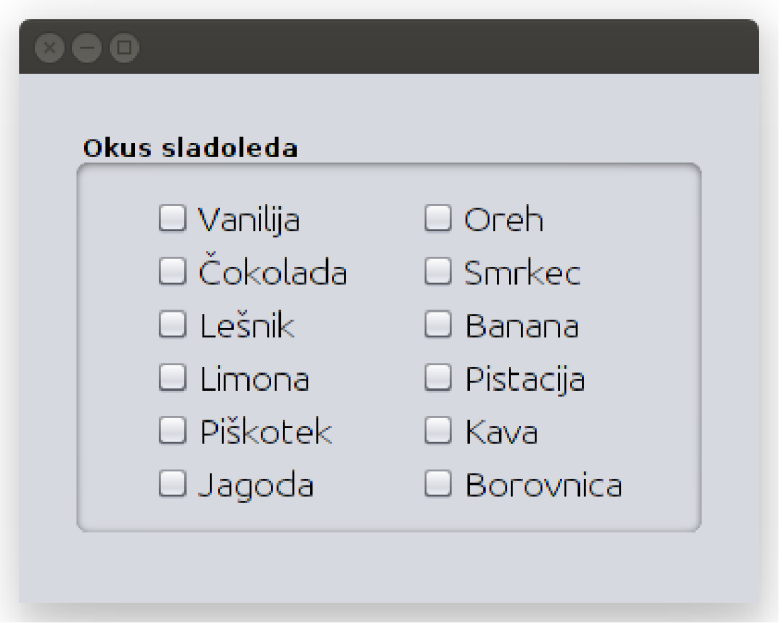
Slika 7.11 Del dialoga za izbiro sladoleda, kjer je prikazan del za izbiro okusa.

zamenjamo z okvirjem z naslovom. V okvirju pa so nato dodana stikala, s katerimi lahko izbiramo posamezne okuse. Primer takega dialoga se nahaja na sliki 7.12. V primeru velikega števila okusov bi lahko okuse razdelili tudi v dva stolpca, zato da dialog ne bi bil predolg. Primer dialoga z okusi v dveh stolpcih je prikazan na sliki 7.13.

Druga možnost bi bila, da bi namesto polja stikal uporabili seznam, ki tudi nudi možnost izbire več elementov. A ima seznam to slabost, da se ga je potrebno naučiti uporabljati, poleg tega pa ima "krhko" izbiro, kjer lahko napačen klik z miško odznači že izbrane možnosti. Zaradi tega je tudi uporaba takih seznamov pogosto problematična, saj uporabniki teh seznamov pogosto ne znajo uporabljati oziroma lahko hitro vodijo v neučinkovito delo (ob neprimernem kliku).

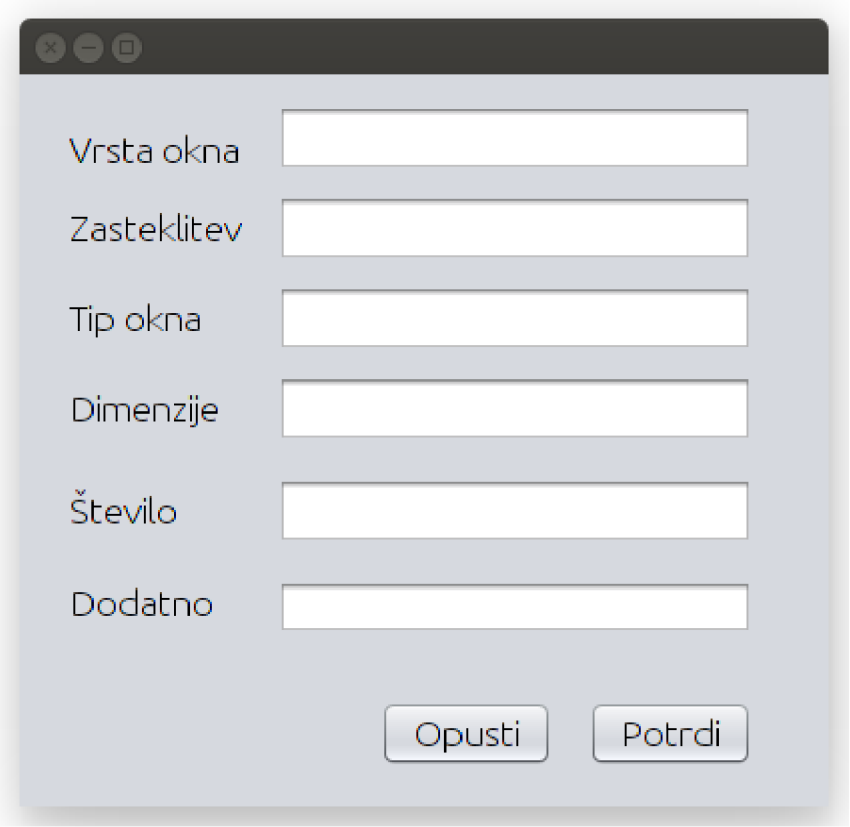


Slika 7.12 Spremenjen del dialoga za izbiro sladoleda, kjer je prikazan del za izbiro okusa z uporabo polja stikal v enem stolpcu.



Slika 7.13 Spremenjen del dialoga za izbiro sladoleda, kjer je prikazan del za izbiro okusa z uporabo polja stikal v dveh stolpcih.

155. Slika 7.14 prikazuje dialog za izbiro oken v okviru aplikacije za naročilo oken. V dialog lahko vnašate vrsto okna (PVC, lesena ali pa lesena in ALU okna), vrsto zasteklitve (dvoslojna, troslojna in troslojna s kriptonom), tip okna (enokrilno, dvokrilno, dvokrilno s pokončno letvijo, fiksno, nagibno, balkonska vrata, dvokrilna balkonska vrata, drsna stena), dimenzije okna in dodatne možnosti, kot so rolete, senčila in komarniki. Vaša naloga je, da za dani dialog izberete primernejše podobe in na novo načrtate dialog. Podobe, ki se nahajajo v dialogu na sliki 7.14, lahko zamenjate z drugimi podobami, prav tako pa lahko dodate tudi nove podobe, če mislite, da je to smiselno.



Slika 7.14 Dialog za izbiro enega tipa oken v okviru aplikacije za naročilo stavbnega pohištva.

Rešitev: Dialog za izbiro okna sicer vsebuje podobe, ki omogočajo naročilo želenih oken, vendar uporabniku ne nudijo dovolj namigov pri mogočih vrednostih oziroma izbirah pri posamezni opciji. Te podoba pa prav tako uporabniku ne preprečujejo vnosa nepravilnih ali neobstoječih vrednosti. Zaradi tega je dobro, če se nekatere podobe v oknu zamenjajo, kjer je to mogoče, s podobami, ki omogočajo izbiro. Pri izbiri vrste oken lahko

uporabnik izbira samo med možnostmi, ki so na voljo (PVC, lesena in lesena ter ALU okna), kjer pa uporabniku to tudi ni posredovano. Pri izbiri podobe za vnos vrste oken je na voljo več podob, ki uporabniku nudijo možnost izbire: polje izključujočih stikal, seznam, kombiniran izvlečni seznam, krožno polje; in pa podoba, ki omogoča vnos, to je obstoječe polje za vnos. Pri odločitvi za primerno podobo uporabimo algoritme za izbiro podob. Med odločitvijo za polje za vnos ali polje za izbiro se vprašamo:

- Ali so vhodni podatki omejeni po količini in vsebini?
  Odgovor je "Da", saj imamo natanko tri možnosti (ta odgovor še ne da končne rešitve, tako da so potrebna še nadaljnja vprašanja).
- Ali so podatki znani in razumljivi?
  Tudi tu je odgovor "Da"; podatki so znani in razumljivi, saj se ve, med katerimi vrstami oken lahko izbiramo in ni nejasnosti (tudi po tem odgovoru še nimamo dokončnega odgovora, zato je potrebno zastaviti nadaljnja vprašanja).
- Ali so mogoče tipkarske napake? Tudi tu je odgovor "Da"; če želimo vnesti eno izmed možnosti, je pri vnosu mogoče narediti tipkarsko napako, poleg tega pa je mogoče, da bi lahko različni uporabniki vnašali različne vrednost, na primer, nekdo bi lahko vnesel ALU okna, kdo drug pa aluminijasta okna, nekdo pa samo ALU (po tem odgovoru pa imamo tudi že končni odgovor, da je polje za izbiro primernejše od vnosnega polja).

Pri odločitvi med ponujenimi polji za izbiro pa vidimo:

- da imamo omejeno število možnosti;
- da ni sprememb;
- da je dovolj prostora.

Na podlagi tega se med potencialnimi podobami, ki omogočajo izbiro, odločimo za polje stikal. Zaradi samo treh možnosti prikaz vseh teh treh možnosti ni problematičen, poleg tega pa so za uporabnika taka polja tudi intuitivna in lahka za uporabo. Pri izbiri tipa zasteklitve velja podoben razmislek. Tudi tu imamo tri možnosti, kar ponovno ponuja isti nabor podob. Tudi tu se izkaže kot najprimernejša izbira polje izključujočih stikal, saj izključujoča stikala ne zasedejo preveč prostora, poleg tega so tudi intuitivna. Pri izbiri tipa oken pa uporabnik izbira med več možnostmi (osem možnosti, ki so ponovno omejene). Ponovno je na voljo več podob, ki nudijo ustrezno izbiro: polje izključujočih stikal, seznam, kombiniran izvlečni seznam, krožno polje in vnosno polje. Tudi v tem primeru imamo medsebojno izključujoče se

alternative, diskretne podatke, dve ali več alternativ, poleg tega pa ni dovolj prostora za izris vseh možnosti, kar da kot končno rešitev dvižni seznam.

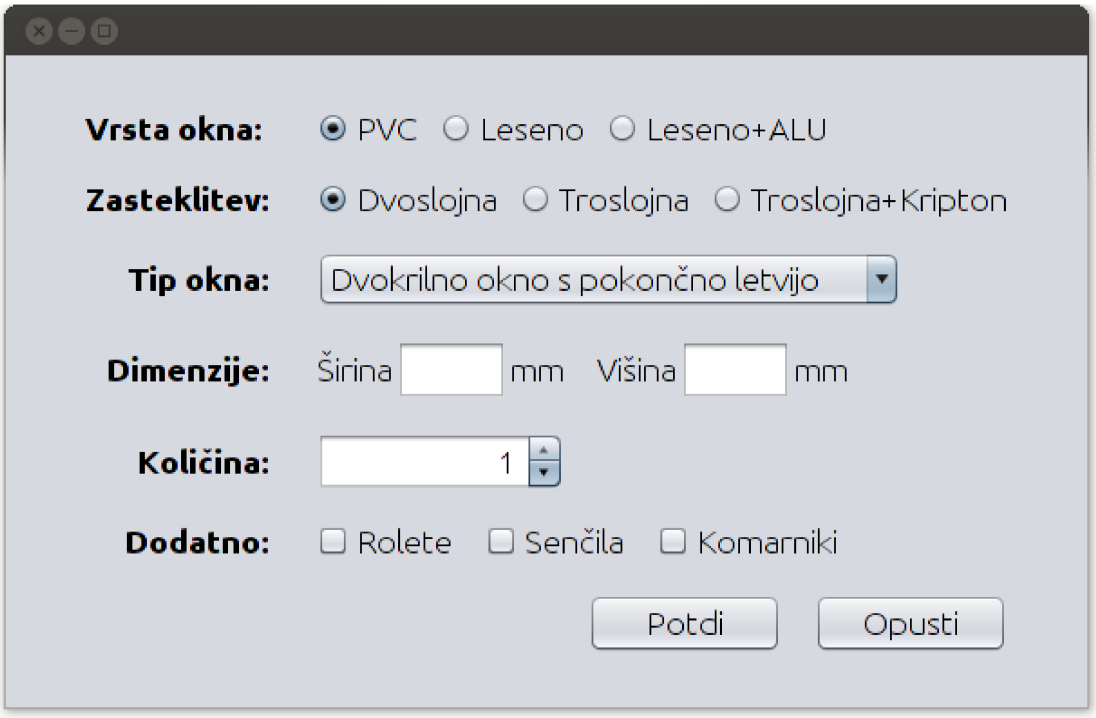
Pri izbiri podobe za vnos dimenzije okna se takoj pojavi dvoumnost pri vnosu, saj je potrebno vnesti dve vrednosti, to sta, širina in višina, nikjer pa ni določen format vnosa in pravilni vrstni red obeh dimenzij. Prav tako niso podane merske enote (npr. mm ali cm). Pri izbiri najprimernejše podobe vidimo, da podatki niso omejeni po količini in vsebini, tako da je najprimernejša podoba polje za vnos. Za formatiran vnos (npr. 1000×1200) bi lahko uporabili polje za formatiran vnos, vendar s tem še ni določeno, kateri podatek je širina, kateri pa višina okna. Zaradi tega je primernejše razdeliti podobo za vnos dimenzij na podobi za vnos širine in vnos višine, kjer vsaki podobi sledi še oznaka, ki podaja merske enote.

Pri izbiri števila oken vidimo, da imamo medsebojno izključujoče se alternative, znane in napovedljive zaporedne podatke, kjer je tipkan vhod včasih tudi zaželen (uporabniki po večini izbirajo enega do pet oken iste vrste, včasih pa se lahko zgodi, da se naroči večja količina oken enakih dimenzij, npr. novogradnja/obnova stanovanjskega bloka, kar zahteva tipkan vnos za večje vrednosti). Glede na te značilnosti je najprimernejša podoba krožno polje, ki omogoča uporabo miške za izbiranje med zaporednimi podatki, omogoča pa tudi tipkanje kakšnih izjemnih vrednosti.

Dodatne možnosti so tudi omejene po količini in vsebini (tri možnosti), predstavljajo diskretne podatke (da/ne), tipkanje podatkov ni potrebno, prav tako pa tudi ni sprememb in je dovolj prostora, iz česar dobimo kot primerno podobo skupino stikal. Primer mogočega novega aranžiranja tega dialoga je prikazan na sliki 7.15

156. Slika 7.16 prikazuje uporabniški vmesnik za sprejem in spremljanje izgubljenih živali v azilu za male živali. Azil sprejema samo nekaj vrst malih živali, za katere vodi evidenco o starosti, spolu, datumu sprejema, cepljenju in ostalih storitvah, ki so jih nudili posamezni živali. Poleg starosti imajo tudi posebno rubriko, iz katere je takoj razvidno, ali je žival morda mladiček, saj imajo mladički veliko večjo verjetnost, da jih bo kdo vzel iz azila domov. Pri izgradnji vmesnika se je zgodila manjša nepazljivost, tako da so podobe nekoliko premetane po oknu, poleg tega pa nekatere podobe v tem vmesniku tudi niso najboljše izbrane. Vaša naloga je, da izberete primernejše podobe, kjer je to potrebno, ter na novo načrtate ta dialog. Poleg tega boste morali na nekaterih delih tudi popraviti besedilo in uporabljeno pisavo.

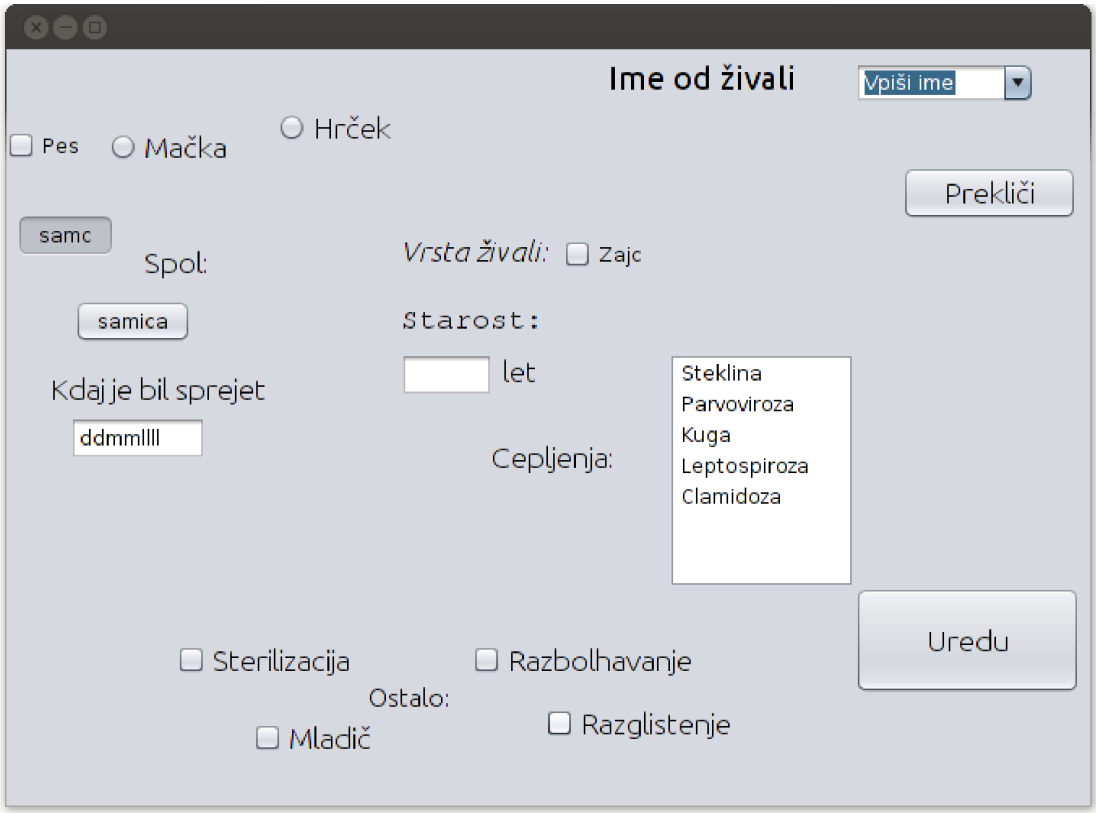
Rešitev: Ob pogledu na dialog za sprejem in spremljanje posamezne živali vidimo, da so podobe neprimerno aranžirane, neprimerno izbrane, prav tako pa so posamezna besedila neprimerna in nekonsistentno napisana. Za vnos imena imamo kombiniran izvlečni seznam, v katerega lahko pišemo. Zaradi



Slika 7.15 Primer mogočega aranžiranja dialoga za naročilo enega tipa oken v okviru aplikacije za naročanje stavbnega pohištva.

velikega števila mogočih imen je njegova uporaba nesmiselna, saj uporabniku ne pomaga z možnostjo izbire. Boljša podoba za vnos imena bi bilo vnosno polje. Za izbiro vrste živali je, glede na to, da azil sprejema samo štiri vrste živali, primerna uporaba polja izključujočih stikal, kjer morajo biti vse podobe konsistentno izbrane, se pravi, vse podobe morajo biti izključujoča stikala. Za vpis starosti je primernejše krožno polje kot vnosno polje, saj omogoča izbiro in s tem uporabniku pomaga, da ne dela napak s tipkanjem. Pri izbiri spola je primernejša uporaba polja izključujočih stikal kot uporaba zatičnega gumba, saj je polje izključujočih stikal bolj intuitivno za uporabo. Pri izbiri datuma sprejema obstajajo različne komponente, ki jih je mogoče uporabiti za vpis datuma in so primernejše kot vnosno polje. Tako bi lahko uporabili formatirano tekstovno polje za lažji vnos datuma ali pa kombinacijo izvlečnih seznamov za izbiro dneva, meseca in leta, mogoče pa je tudi uporabiti krožno polje, ki omogoča izbiro datuma. Pri izbiri opravljenih cepljenj je bil v prvotnem vmesniku izbran seznam, ki smo ga zamenjali s poljem stikal zaradi manjše porabe prostora in kompaktnejšega izgleda vmesnika. Pri izbiri ostalih storitev smo pustili prej izbrane komponente, saj so bile primerne.

Nato smo popravili napise. Napis “Ime od živali” ni v lepi slovenščini, zato

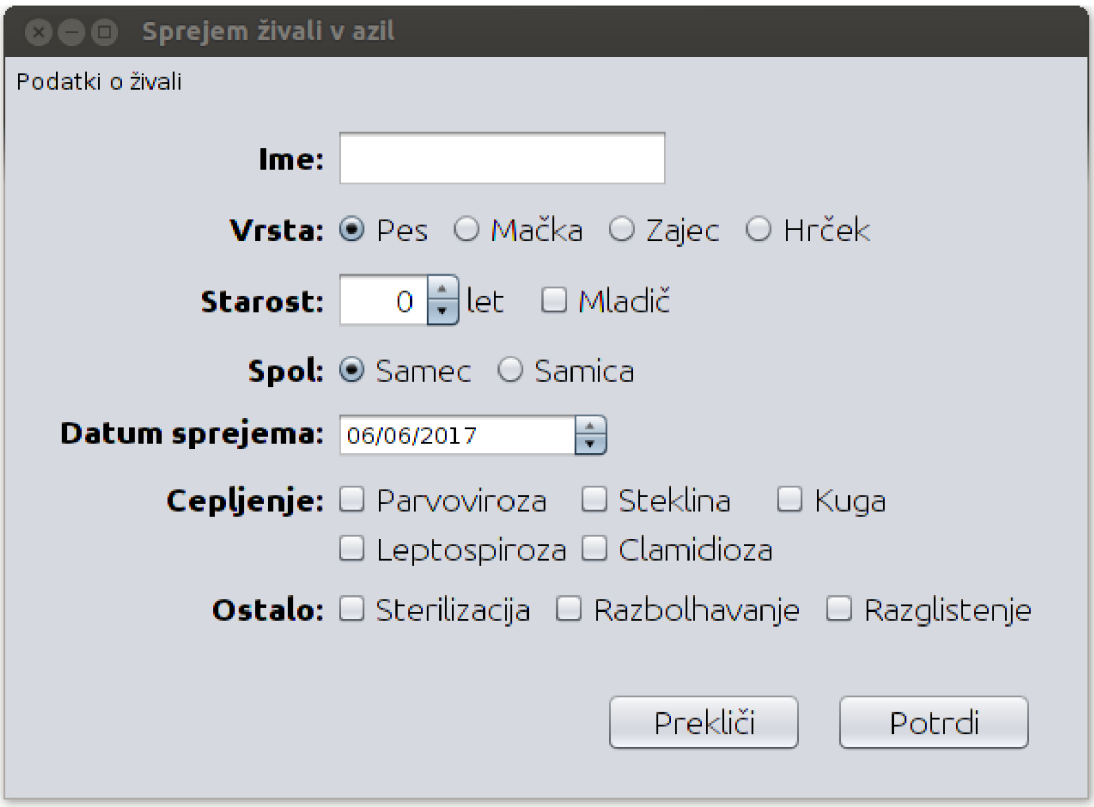


Slika 7.16 Uporabniški vmesnik programa za sprejem in spremljanje izgubljenih živali v azilu za male živali.

bi ga bilo smiselno popraviti v “Ime živali:”. Ker pa je ta napis poleg napisa “Vrsta živali” edini, ki posebej poudarja, da gre za žival, je potrebno zaradi konsistentnosti bodisi dodati “žival” v ostale napise ali pa to besedo odstraniti iz teh dveh napisov. Da bi dosegli enostavnejši in preglednejši vmesnik, smo se odločili za odstranitev besede “žival” iz napisov, ki to besedo vsebujejo. Namesto tega smo za zagotovitev nedvoumnosti vmesnika dodali na vrh vmesnika še oznako “Podatki o živali”. Spremenili smo tudi preostala neprimerno izbrana besedila: “Zajc” smo zapisali kot “Zajec”, “samc” kot “Samec”, “Kdaj je bil sprejet” v “Datum sprejema:”, “Cepljenja” v “Cepljenje:” in “Uredu” v “Potrdi”. Poleg tega smo tudi popravili nekonsistentnosti pri uporabi velikih in malih črk ter nekonsistentnosti pri uporabi dvopičja.

Nato smo izvedli še aranžiranje vmesnika. Vmesnik smo orientirali vertikalno, tako da si podobe sledijo od zgoraj navzdol. Na levi strani so oznake, ki so vertikalno poravnane po desnem robu, na desni strani pa so vnosna polja, ki so, kjer se to da, vertikalno poravnana po levem robu. Oznake in vnosna polja so horizontalno poravnana po spodnjem robu. Med posameznimi vnosnimi

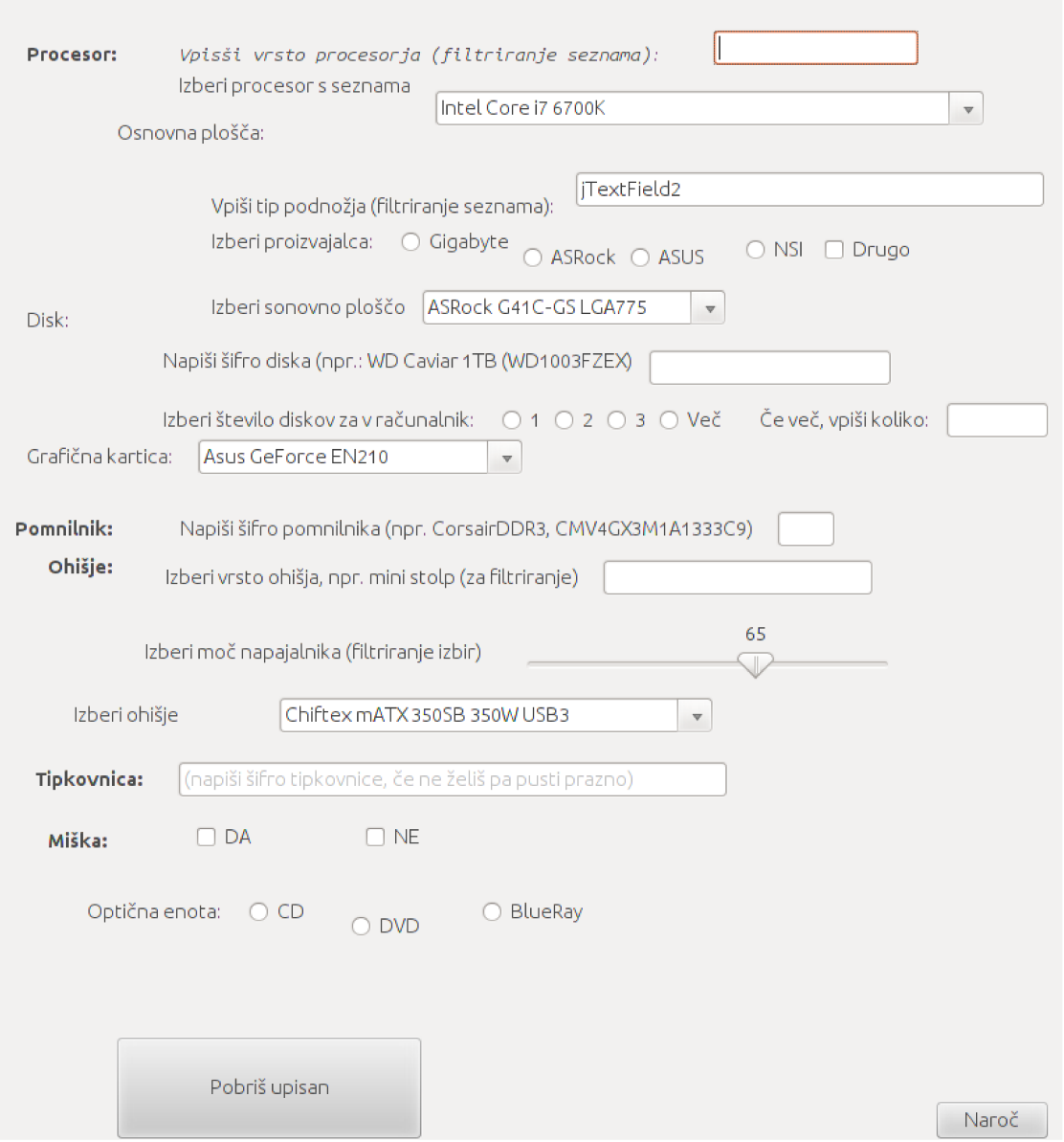
polji je enak bel presledek, tako da te komponente primerno grupira. Manjši bel presledek pa je med vrsticama stikal za cepljenje, da se stikala grupirajo. Oddaljenost podob od zgornjega in od spodnjega ter od levega in od desnega roba je enaka. Uporabili smo dve pisavi, eno za oznake (krepka) in eno za vnosna polja (normalna). Dobljeni dialog je prikazan na sliki 7.17.



Slika 7.17 Nov uporabniški vmesnik aplikacije za sprejem in spremljanje izgubljenih živali v azilu za male živali.

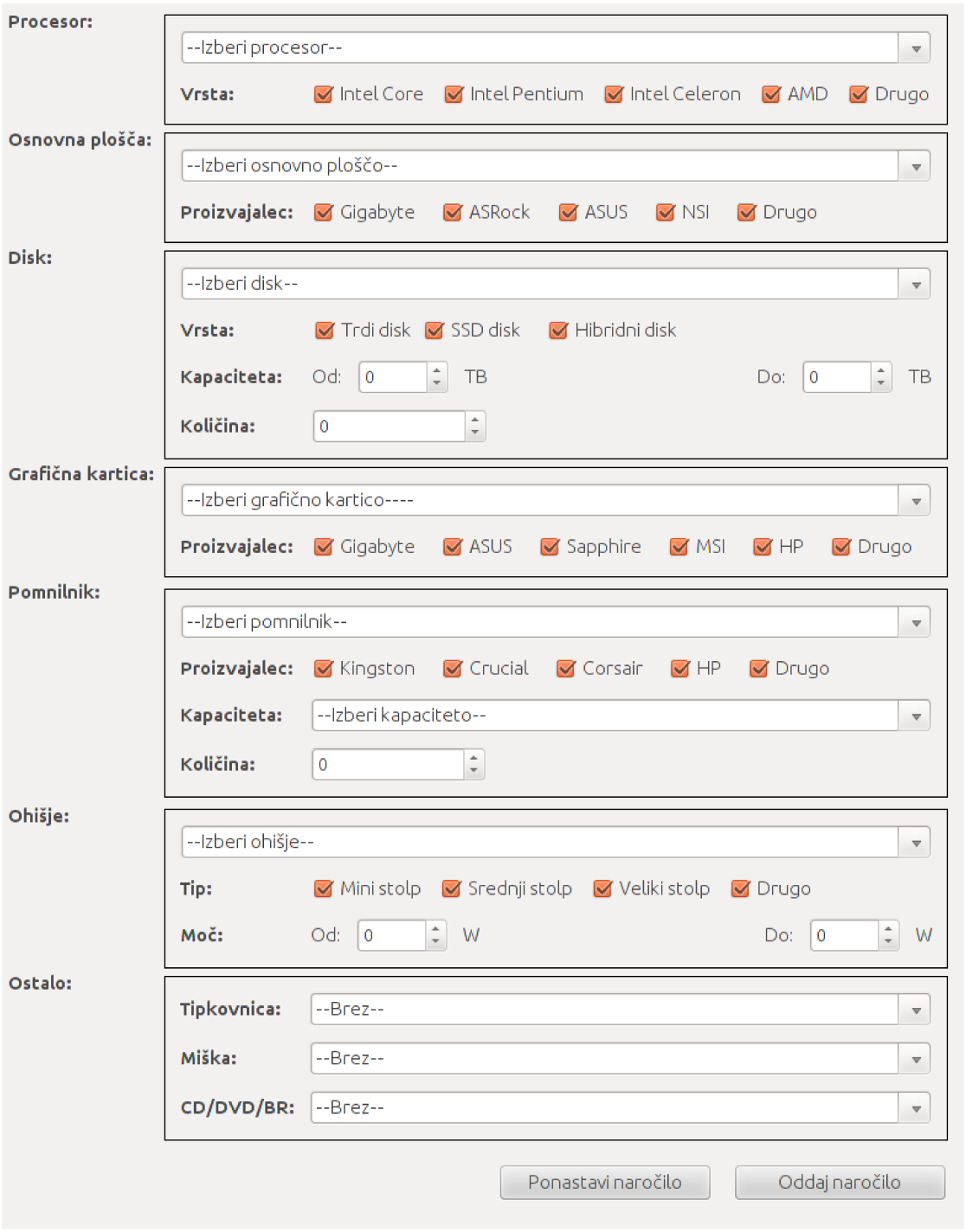
157. Slika 7.18 prikazuje dialog za naročilo računalnika po komponentah. Podobe v tem vmesniku niso najboljše izbrane in tudi ne najboljše aranžirane. Vaša naloga je, da popravite izbiro podob in aranžiranje tega dialoga.

Rešitev: Na sliki 7.19 je prikazan primer možne implementacije dialoga za naročilo računalnika po komponentah, kjer so komponente boljše izbrane in ki je boljše aranžiran, kot je bil osnovni dialog. Pri tem so možnosti za posamezne komponente izbrane tako, da omogočajo enostavno izbiro, vnos z uporabo tipkovnice pa ni več potreben. Za grupiranje možnosti v okviru posamezne komponente so bili uporabljeni okvirji, za pojasnilo vnosa pa je na levi strani dodana tudi oznaka, ki je z okvirjem poravnana horizontalno



Slika 7.18 Dialog za naročilo računalnika po komponentah.

po zgornjem robu. Namesto oznak bi bilo mogoče uporabiti tudi okvirje z naslovom, kjer bi naslov okvirja lahko nadomestil posamezno oznako. Vendar pa uporaba oznak omogoča realizacijo preglednejšega dialoga, saj so tako postavljene oznake vidnejše.



Slika 7.19 Nov dialog za naročilo računalnika po komponentah kot del aplikacije za naročanje računalnika.

158. Spodnji dialog prikazuje primer naročanja malice preko telefona. Vaša naloga je, da na podlagi tega dialoga načrtate uporabniški vmesnik, ki bo omogočal vnos vseh podatkov, kot jih zahteva dialog.

- **Dober dan. Poklicali ste "Halo malica".**
- Dober dan. Rad bi naročil malico.
- **Seveda, kaj pa boste naročili? Nudimo razne glavne jedi in sladice, poleg tega pa imamo še pice.**
- Kakšne glavne jedi pa imate na voljo?
- **Nudimo enolončnice, in sicer pasulj, segedin zelje, joto in golaž.**
- Ali lahko zraven naročim še kakšen dodatek?
- **Lahko vam ponudimo še klobaso, hrenovko ter seveda kruh.**
- Vzel bom joto s klobaso in kruhom. Kakšne sladice pa imate?
- **Na voljo imamo različne zavitke: jabolčni, sirov, marelični in češnjev zavitek. Zraven pa vam lahko damo še malo smetane, dodatno sladkamo in posujemo s čokolado.**
- Pa bi res vzel še en sirov zavitek s smetano in en češnjev zavitek.
- **Odlična izbira. Ali vas morda zanima še kaj drugega? Na voljo imamo še pice, in sicer Margerito, Morsko in Zelenjavno, vsaki pici pa po želji lahko dodate še koruzo, jajce in pa hren.**
- Ne hvala, to bi bilo vse.
- **V redu, komu in kam pa lahko to dostavimo?**
- Sem Janez Novak, in sicer živim na Novakovi 389 v Ljubljani.
- **Za telefonsko številko bi še prosil.**
- Moj telefon je 031 000 000.
- **Dostavili vam bomo v 30 minutah. Hvala za naročilo in lep dan vam želim.**

Rešitev: Na podlagi dialoga lahko razberemo potrebne komponente. Zaradi enostavnosti je za izbiro posameznih jedil (enolončnic, zavitkov in pic) najprimernejša uporaba polja stikal za izključujočo izbiro, za izbiro dodatkov pa polje stikal, ki ju lahko grupiramo z uporabo okvirjev za izbiro jedi in dodatka iz posamezne grupe. Iz pogovora je tudi razvidno, da lahko stranka želi več kot samo eno jed iste vrste (na primer dva zavitka), tako da je potrebno v vmesniku omogočiti tudi to. Pri realizaciji te možnosti je na voljo več možnosti. Ena izmed možnosti je, da vmesnik realiziramo s pomočjo okvirjev,

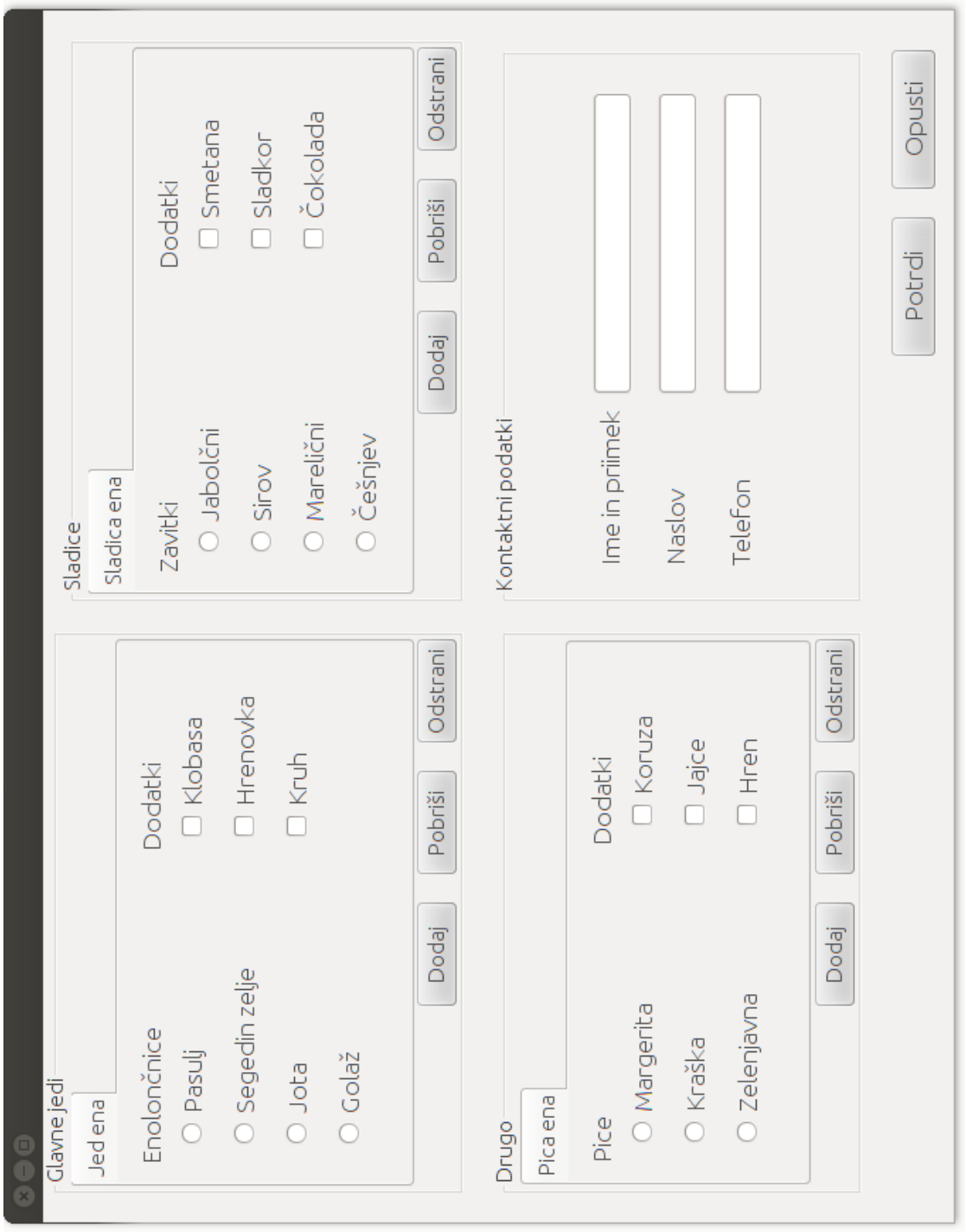
kjer za vsako jedilo (glavna jed, sladica, drugo) uporabimo svoj okvir, ki je na začetku prazen. V vsak okvir dodamo gumb "Dodaj", ki omogoča dodajanje nove jedi iste vrste, s tem da doda nov zavihek za to grupo. Poleg tega dodamo še gumba "Pobriši", ki zbriše oziroma inicializira posamezen vnos v primeru, da smo se pri vnosu zmotili in želimo začeti z vnosom ponovno, ter "Odstrani", s katerim lahko posamezno jed odstranimo iz naročila, če je naročnik ne želi več naročiti. Gumba "Pobriši" in "Odstrani" pa delujeta na trenutno izbranem/prikazanem zavihku.

Prav tako dodamo še vnosna polja za vnos imena in priimka, naslova ter telefonske številke, ki jih tudi grupiramo z uporabo okvirjev. Tudi tukaj bi lahko dodali gumb "Pobriši" tako kot pri okvirjih za izbiro različnih vrst jedi, vendar tega nismo realizirali predvsem zaradi majhnega števila vnosnih polj in ker nimamo dodatnih zavihkov, ki bi jih morda želeli spreminjati. Na dno vmesnika pa dodamo še gumba za potrditev naročila ("Potrdi") in preklic naročila ("Opusti").

Ko so podobe izbrane, jih je potrebno še primerno aranžirati. Znotraj posamezne grupe sta polje stikal za izključujočo izbiro in polje stikal orientirani vertikalno in malo zamaknjeni glede na opis. Ti polji stkal sta postavljeni v dveh stolpcih. Znotraj vsake skupine spodaj sta še dva gumba za dodajanje in za preklic posamezne izbrane jedi. Beli presledki med stikali v poljih stikal so taki, da jih grupirajo skupaj, med polji stikal pa so dovolj veliki, da jih ločijo. Posamezne grupe si sledijo horizontalno in so razporejene v dveh vrsticah zaradi večje preglednosti in kompaktnosti vmesnika. Možna bi bila tudi rešitev, kjer si grupe sledijo vertikalno (en stolpec) ali pa horizontalno (ena vrstica). Prav tako je možna tudi rešitev, kjer je vsaka grupa v svojem zavihku. Primer opisane rešitve se nahaja na sliki 7.20.

159. Spodnji pogovor prikazuje primer naročanja malega oglasa za najem ali oddajo stanovanjske enote preko telefona. Vaša naloga je, da na podlagi tega dialoga načrtate uporabniški vmesnik, ki bo omogočal vnos podatkov, kot jih zahteva dialog. Če se vam zdi potrebno ali smiselno, lahko kakšno podobo, ki ni zajeta v tem dialogu, tudi dodate.

- **Dober dan. Poklicali ste male oglase za najem ter oddajo stanovanjskih enot. Kako vam lahko pomagam?**
- Dober dan. Rad bi naročil mali oglas.
- **Seveda. Ali želite mali oglas za najem ali oddajo?**
- Rad bi oddajal.
- **V redu, ali lahko še poveste naslov in pošto te stanovanjske enote?**

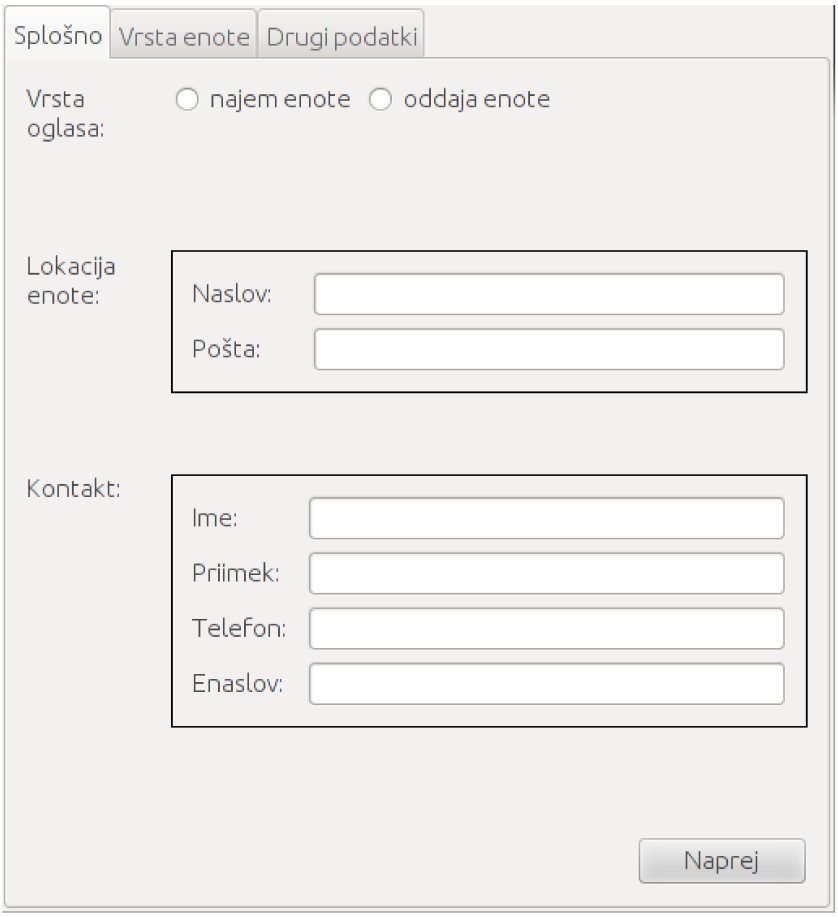


Slika 7.20 Uporabniški vmesnik programa za sprejemanje naročil za malico.

- Naslov je "Ulica za oddajo 20", pošta pa je 1000 Ljubljana.
- **Prosil bi še za vaše kontaktne informacije: ime in priimek, telefonsko številko in elektronski naslov, če ga imate.**
- Vsekakor, jaz sem Janez Novak, moja telefonska številka je 099 999 999, elektronskega naslova pa nimam.
- **Ali oddajate samo sobo, stanovanje ali pa morda celo hišo?**
- Rad bi oddal dve sobi.
- **Torej dve sobi? Ali to vključuje tudi kuhinjo in kopalnico?**
- Sobi vključujeta kopalnico s straniščem in souporabo kuhinje.
- **Ali ima soba tudi balkon?**
- Da, ima balkon, ki pa ni zastekljen.
- **Kolikšna pa je površina sob?**
- Sobi imata skupaj 25 m^2 in sta primerni za dve osebi.
- **Ali oddajate tudi garažo?**
- Ne, garaže ne oddajam, sobam pa pripada parkirišče.
- **Kako pa je z ogrevanjem? Ali imate svojo peč, skupinsko kotlovnico ali imate morda daljinsko ogrevanje?**
- Imamo daljinsko ogrevanje.
- **Ali so na voljo še kakšne druge stvari, ki bi jih izpostavili? Na primer telefon, satelitsko anteno, optični kabel, klima napravo, brezžični internet, dvigalo?**
- Da, v stanovanju je brezžični internet, prav tako pa ima tudi telefonski priključek. Ah da, skoraj bi pozabil, sobi sta tudi že popolnoma opremljeni.
- **V redu, mi lahko poveste še nadstropje te enote?**
- Seveda, to je stanovanje 111, ki se nahaja v tretjem nadstropju.
- **Kakšna pa je cena najema?**
- Cena je 350 evrov na mesec.
- **To bi bilo zaenkrat vse. Hvala za naročilo in na svidenje.**
- Na svidenje.

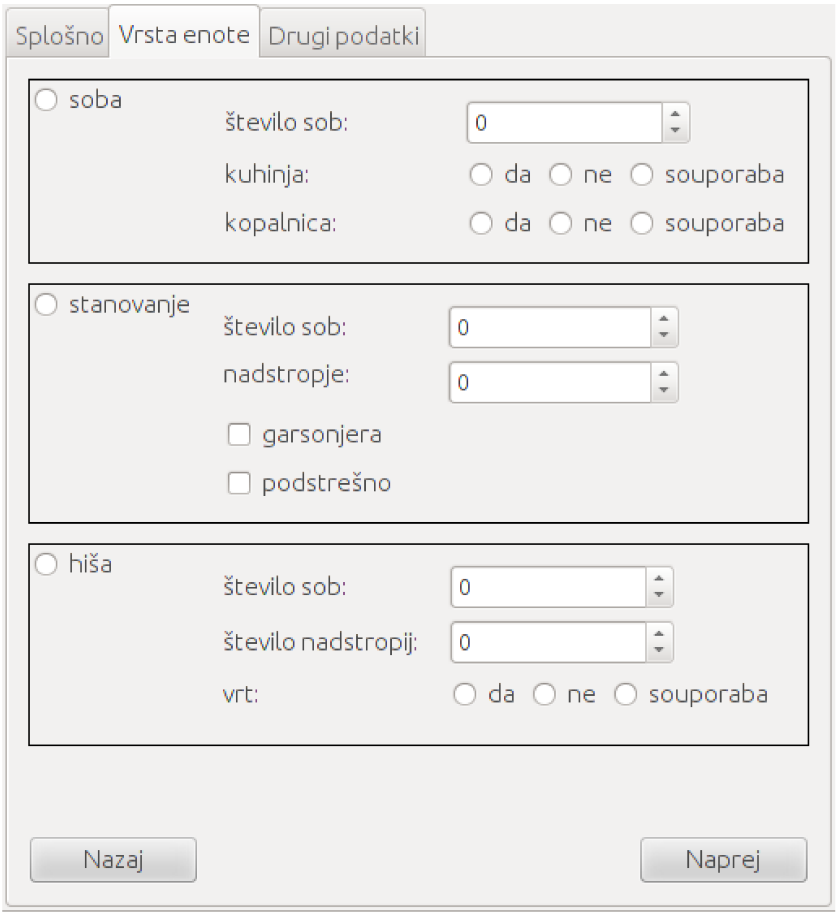
Rešitev: Na podlagi zgornjega dialoga je moč razbrati potrebne komponente. Za vnos podatkov o tipu oglasa je najprimernejša izbira stikal za izključujočo izbiro. Za vnos podatkov o naslovu stanovanjske enote in kontaktnih podatkov je edina možna izbira vnosno polje zaradi neomejene količine podatkov. Pri

tem lahko sicer uporabniku pomagamo tako, da pri polju za vnos telefonske številke in pri polju za vnos elektronskega naslova izberemo formatirano vnosno polje, ki sproti omejuje in opozarja uporabnika, če je prišlo do napačnega vnosa. Pri izbiri polj, ki zahtevajo vnos številčnih informacij, kot je na primer število sob ali pa nadstropje, smo izbrali krožno polje, ki omogoča vnos števil. Pri poljih, ki zahtevajo izbiro samo ene izmed možnosti, kot je to na primer možnost dostopa do kuhinje v primeru oddajanja sob, smo uporabili izključujoča stikala. Pri izbiri posameznih lastnosti, kot je na primer možnost dostopa do interneta, pa smo izbrali stikalo.



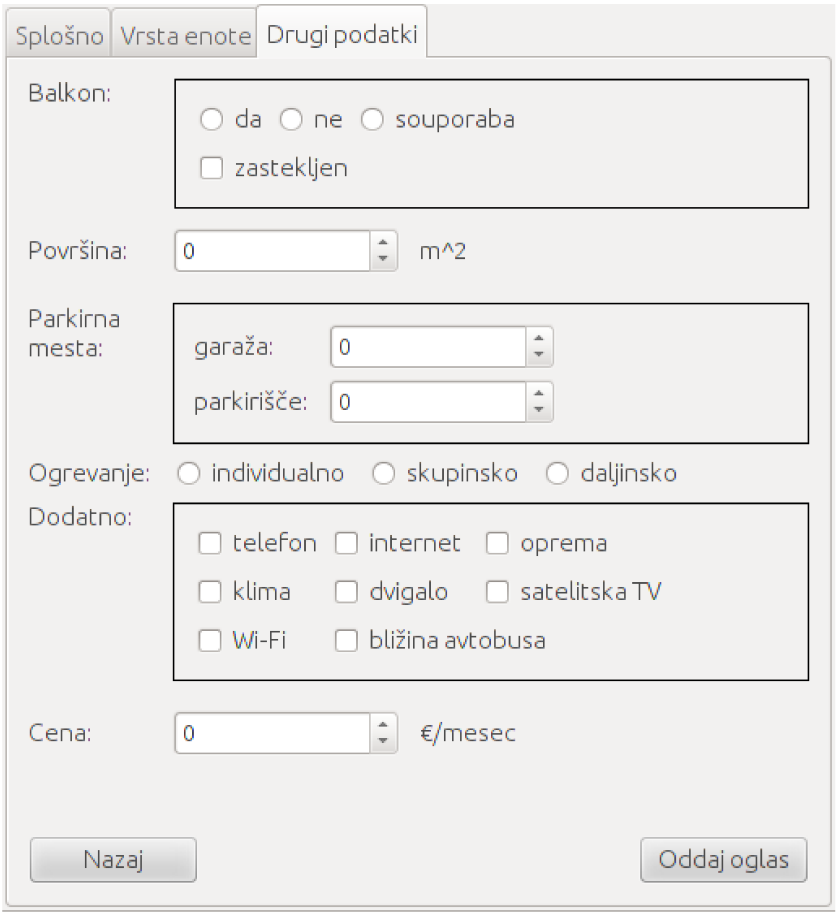
Slika 7.21 Prvi zavihek uporabniškega vmesnika aplikacije za sprejemanje naročil za oglas za najem ali oddajo stanovanjske enote.

Pri aranžiranju podob smo se odločili za grupiranje posameznih kategorij podatkov v okviru posamezne površine z zavihki. Tako smo celoten obrazec razdelili na več zavihkov, in sicer na zavihek s splošnimi informacijami (slika 7.21), zavihek z informacijami o vrsti stanovanjske enote (slika 7.22) ter na



Slika 7.22 Drugi zavihek uporabniškega vmesnika aplikacije za sprejemanje naročil za oglas za najem ali oddajo stanovanjske enote.

zavihek z drugim relevantnimi podatki (slika 7.23). Med posameznimi zavihki smo omogočili tudi možnost navigacije z uporabo gumbov. Pri aranžiranju posameznih zavihkov smo podatke, ki v nekem zavihku sodijo skupaj, zaradi boljše vidljivosti in grupiranja ločili z uporabo okvirjev. Pri aranžiranju smo se odločili za vertikalno orientacijo, kjer smo med posameznimi podobami znotraj grupe uporabili konsistentno manjše bele presledke za boljše vizualno grupiranje podob. Med posameznimi grupami smo uporabili tako okvirje kot tudi bele presledke, ki pa so večji od belih presledkov znotraj skupine in so sorazmerni glede na okolico. Pri tem so oznake na levi strani (razen pri opisih stikal, kjer so na desni strani), vnosna polja pa na desni strani. Oznake so vertikalno poravnane po levem robu, vnosna polja pa vertikalno po levem robu in po desnem robu, kjer je to mogoče oziroma smiselno. Oznake in vnosna polja so horizontalno poravnana po spodnjem robu. Podobe so uravnoteženo



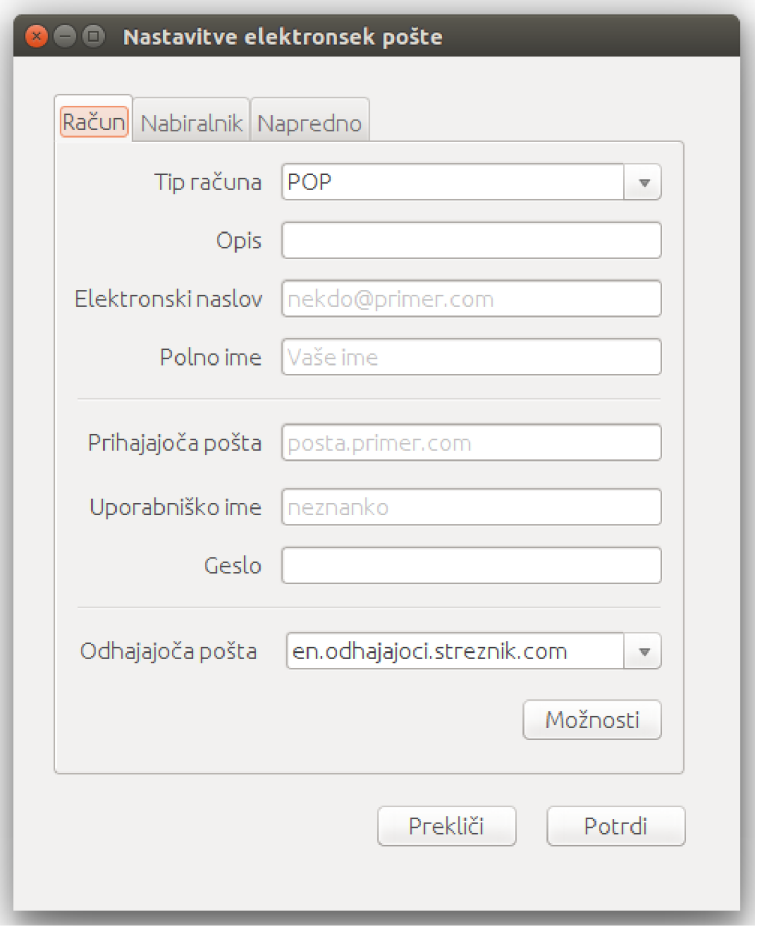
Slika 7.23 Tretji zavihek uporabniškega vmesnika aplikacije za sprejemanje naročil za oglas za najem ali oddajo stanovanjske enote.

razmeščene po celotni površini okna, tako da je vmesnik balansiran okoli vertikalne osi.

160. Spodnja slika 7.24 prikazuje dialog za nastavitve lastnosti elektronske pošte. Pozorno preglejte dialog in naštejte dobre in slabe lastnosti, ki jih opazite.

Rešitev: Dobre (+) in slabe (-) lastnosti dialoga so:

+ namigi, ki pomagajo uporabniku pri izpolnjevanju;
+ uporaba ločevalnikov za vizualno ločevanje posameznih logičnih skupin;
+ uporaba podob, ki omogočajo izbiro, kjer je to mogoče;
- ni gumba, ki bi omogočal shranjevanje vnesenih vrednosti.

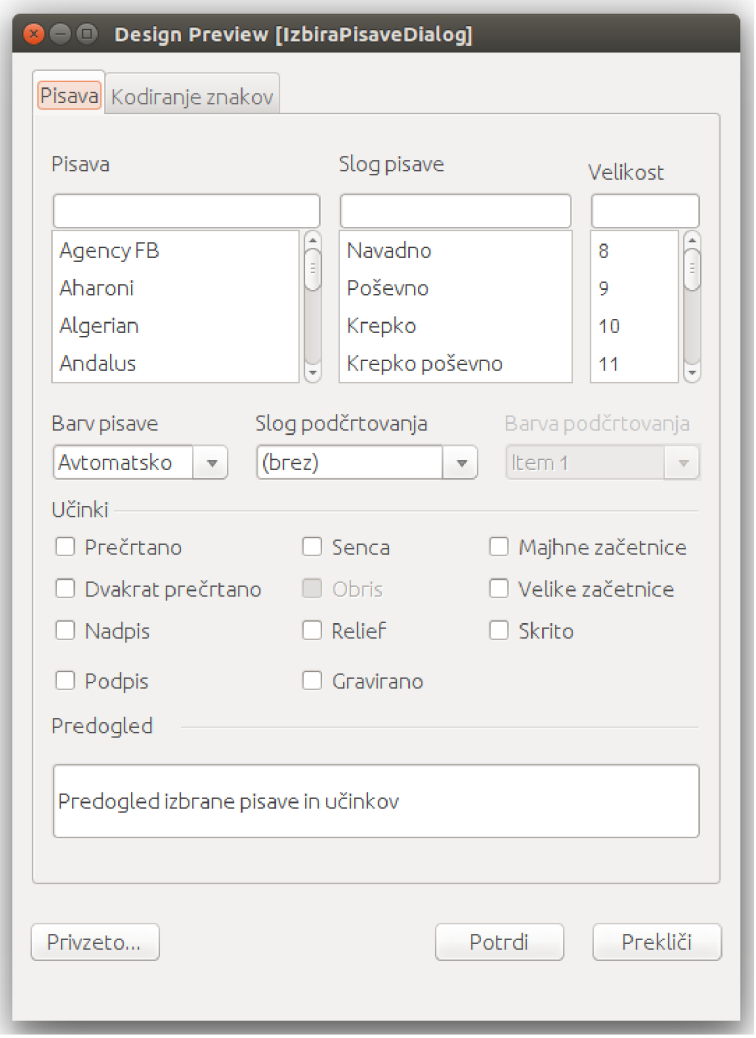


Slika 7.24 Dialog za nastavitev lastnosti elektronske pošte.

161. Slika 7.25 prikazuje primer dialoga za izbiro efektov pisave v nekem pisarniškem programu. Pozorno preglejte dialog in poiščite napake, ki jih najdete. Povejte tudi, kako bi lahko odpravili odkrite napake.

Rešitev: Problematična je pretirana uporaba stikal za izbiro. V tem dialogu je mogoče hkrati izbrati tudi možnosti, ki se medsebojno izključujejo:

- hkrati lahko izberemo prečrtano in dvakrat prečrtano pisavo;
- izberemo lahko pisavo, ki je hkrati podpis in nadpis;
- prav tako je lahko pisava v tem dialogu hkrati gravirana, samo obris in reliefna;
- črke so lahko hkrati velike tiskane črke v velikosti malih tiskanih črk (Majhne začetnice), velike tiskane črke (Velike začetnice) ter skrite.



Slika 7.25 Ponazoritev dialoga za izbiro efektov izbrane pisave, ki je bil uporabljen v nekem znanem pisarniškem programu.

Načrtovalci vmesnika so se sicer potrudili z grafičnim načrtom, saj so uporabnikom ponudili možnost prilagoditve velikega števila lastnosti pisav, pri tem pa so popolnoma pozabili na uporabnost. Vse zgoraj naštete izključujoče se možnosti bi morali realizirati z uporabo stikal za izključujočo izbiro. Določiti bi bilo potrebno učinke, ki so med seboj izključujoči, in jih dati v svojo skupino. Poleg tega pa bi bilo potrebno v vsako skupino dodati tudi možnost izključitve posameznega učinka (na primer "Brez"). V posamezni skupini bi tako lahko potem izbrali samo enega izmed učinkov, obenem pa bi lahko izbrali učinek v večih skupinah.

8 Načrtovanje in izbor ikon

162. Naštejte prednosti uporabe dobro načrtanih ikon.

Rešitev: Prednosti uporabe (dobro načrtanih) ikon so:

- privarčujejo prostor na zaslonu;
- se jih hitro prepozna v zasedenem okolju;
- se jih hitro zapomni;
- so v pomoč pri internacionalizaciji vmesnika.

163. Naštejte standardne dele ikone.

Rešitev: Standardni deli ikone so:

- obroba, ki loči ikono od okolice;
- slika, ki predstavlja pomen;
- ozadje, ki poudari sliko;
- oznaka, ki dodatno pojasnjuje ikono.

164. Naštejte principe načrtovanja ikon.

Rešitev: Principi načrtovanja ikon so:

a) skladnost;
b) čitljivost;
c) uporabljaj prepoznavanje in ne spomin;
d) uporabljaj barve konzervativno.

165. Eden izmed principov načrtovanja ikon je "Skladnost". Razložite ta princip.

Rešitev: Pri načrtovanju ikon moramo vse ikone načrtovati kot celoto. Ikone morajo biti vizualno balansirane, na primer vse ikone iz skupine vsebujejo malo grafike. Prav tako naj bi bile ikone skladne glede na velikost, barve, nivo abstrakcije in metafore. Vizualne razlike med ikonami pa naj imajo pomen in naj ne služijo samo kot okras.

166. Eden izmed principov načrtovanja ikon je "Čitljivost". Razložite ta princip.

Rešitev: Pri načrtovanju ikon je pomembno, da se, glede na razpoložljiv prostor, uporabljajo veliki objekti in poudarjene črte. Pri tem je potrebno upoštevati tako velikost zaslona kot tudi razdaljo gledanja. Za dobro čitljivost je potrebno zagotoviti dober kontrast med ospredjem in ozadjem. Priporočljivo se je izogibati krožnicam in poševnim črtam, ki povzročijo nazobčanost, prav tako pa je priporočljivo uporabljati silhuete, ki lahko posredujejo veliko informacij.

167. Eden izmed principov načrtovanja ikon je "Uporabljaj prepoznavanje in ne spomin". Razložite ta princip.

Rešitev: Pri načrtovanju ikon je pomembno, da so uporabniku razumljive. Zato je priporočljivo, da se, kjer je to mogoče, uporabljajo metafore, ki so blizu uporabniku. Priporočljiva je uporaba konkretnih objektov, saj je abstraktne koncepte težko upodobiti. Za vsak slučaj oziroma lažje prepoznavanje pa je pogosto primerno tudi zagotoviti oznake.

168. Eden izmed principov načrtovanja ikon je "Uporabljaj barve konzervativno". Razložite ta princip.

Rešitev: Pri načrtovanju ikon je pomembno, da so ikone razumljive na različnih medijih, med drugim tudi na papirju (želimo natisniti sliko vmesnika). Ob pretirani ali neprimerni uporabi barv se lahko zgodi, da postane taka ikona na nekem mediju neprepoznavna. Zaradi tega je potrebno načrtovati ikone najprej črno-belo, zato da je informacija, ki naj bi jo ikona posredovala, vidna oziroma posredovana uporabniku, šele nato pa se lahko postopoma doda barve. Pretirana uporaba barv tudi preobremeni uporabnika, zaradi tega je primernejša uporaba sivih tonov in ene ali dveh barv.

169. Ikone lahko pripomorejo k internacionalizaciji vmesnika. Na kaj pa je potrebno paziti pri izbiri ikon za internacionalizacijo?

Rešitev: Priporočljivo je, da se ne uporablja besedila ali črk, ki se lahko razlikujejo za različne jezike, za sliko znotraj ikone (v nasprotju z oznako v ikoni), saj se lahko zgodi, da bo potrebno za različne jezike narediti različne verzije ikon (primer take uporabe črk so ikone v nekaterih pisarniških paketih, ki se uporabljajo za predstavitev slogov za poševno, krepko oziroma podčrtano pisavo, ki morajo biti za različne jezike drugačne; pri teh ikonah se za posamezno črko namreč uporablja prva črka v imenu tega sloga, ki pa je lahko različna za različne jezike). Prav tako se lahko geste ali izrazi obraza v

različnih kulturah interpretirajo drugače, zato se jim je pri načrtovanju ikon dobro izogibati. Prav tako pa se je dobro izogibati uporabi metafor, ki so tipične za neko kulturo, drugje pa niso prepoznane.

170. Kaj je jezik ikon? Čemu je namenjen? Podajte kakšen primer takega jezika.

Rešitev: Jezik ikon je sistematičen način združevanja osnovnih simbolov v kompleksnejše ikone. Določata ga besedišče (osnovni simboli) in slovnica (pravila za združevanje simbolov). Namenjen je definiranju večjega števila pomensko povezanih ikon. Primer takega jezika je nabor ikon, ki so bile uporabljene pri nekaterih pisarniških paketih proizvajalca Microsoft.

171. Opišite življenjski cikel načrtovanja ikon.

Rešitev: Življenjski cikel je:

a) Začni s preprosto črno-belo skico na papirju.

b) Testiraj in ponovno načrtuj, dokler osnovni princip ne deluje.

c) Dodaj sivine in morda še kakšno barvo.

d) Prenesi v računalnik in natisni v dejanski velikosti.

e) Testiraj in ponovno načrtuj, dokler ikona ne deluje.

172. Slika 8.1 prikazuje približek ikone, ki se je pojavila, ko so uporabniki namestili gonilnike za diskovne naprave nekega proizvajalca tovrstnih naprav. Kaj ta ikona predstavlja? Možni odgovori so:

a) Disk s kapaciteto 250 GB je skoraj poln.

b) Disk s kapaciteto 250 GB bo kmalu odpovedal.

c) Disk s kapaciteto 250 GB deluje v načinu “turbo”.

Ali morda opazite kakšno težavo pri razumevanju te ikone?

Slika 8.1 Ikona, ki se namesti ob uporabi gonilnikov za diskovne naprave nekega proizvajalca.

Rešitev: Pravilni odgovor je odgovor c) "Disk s kapaciteto 250 GB deluje v načinu "turbo". Ikona naj bi pomenila, da disk deluje v hitrem načinu. Ikona naj bi bila podobna števcu obratov v avtomobilu - višji kot so obrati, hitrejše deluje. A glede na izvedeno anketo je le tretjina vprašanih (na Twitterju) odgovorila pravilno. Večina anketirancev je odgovorila, da bo disk kmalu odpovedal. Problem pri uporabljeni analogiji je ta, da ikona prikazuje kazalec v rdečem območju, to pa ponavadi predstavlja nevarnost. Zaradi tega uporabniki niso povezali ikone z boljšim delovanjem, temveč z nevarnostjo, da bo disk odpovedal.

9 Prototipi

173. Naštejte vrste prototipov v naraščajočem redu kompleksnosti in jih na kratko opišite.

Rešitev: Vrste prototipov so:

- verbalni prototip: opis možnosti in rezultatov;
- papirnati prototip: na roko narisane skice (nizka natančnost);
- računalniški prototipi: z računalnikom narejene dovršene (lahko tudi natisnjene) skice (višja natančnost);
- interaktivne skice: interaktivna kompozicija na roko ali z računalnikom narejenih skic, lahko tudi z operaterjem, ki v ozadju simulira odzive vmesnika;
- delujoči prototip: delujoča aplikacija s preprostimi algoritmi, izmišljenimi podatki.

174. Prototipi se med seboj razlikujejo tudi po natančnosti. Naštejte in na kratko opišite dimenzije natančnosti prototipov.

Rešitev: Natančnost prototipov ima naslednje dimenzije:

- širino: ta dimenzija je določena z odstotkom implementiranih značilnosti; prototip, ki je nenatančen po širini ima implementiranih zelo malo značilnosti; lahko so implementirane samo tiste, ki so nujno potrebne za izvedbo določenih akcij;
- globino: ta dimenzija pove, kako natančno je neka funkcionalnost implementirana; nizka natančnost po globini pomeni, da so implementirane samo osnovne značilnosti, omejena možnost izbire, fiksni odgovori, ni lovljenja napak in podobno;
- model "look": natančnost v smislu modela "look" pove, kako natančno prototip ponazarja končni izdelek; nizka natančnost v smislu modela "look" pomeni, da je prototip po izgledu le približek končnemu izdelku;

- model "feel": natančnost v smislu modela "feel" pove, kako natančno prototip omogoča interakcijo v primerjavi s končnim izdelkom; nizka natančnost v smislu modela "feel" pomeni, da se pri interakciji s prototipom uporabljajo povsem druge metode za interakcijo, na primer kazanje s prstom namesto uporabe miške.

175. Kakšno natančnost dosega papirni prototip v smislu modela "look", v smislu modela "feel" in v smislu globine?

Rešitev: Papirnati prototip dosega naslednjo natančnost:

- model "look": nizka natančnost;
- model "feel": nizka natančnost;
- globina: visoka natančnost.

176. Kaj od naštetega bo papirnati prototip najmanj verjetno razkril? Odgovor tudi utemeljite. (Izberite le en odgovor.)

a) Manjka potrditveni gumb na dialogu za shranjevanje.

b) Gumbi orodne vrstice so premajhni.

c) Dani dialog mora biti nemodalen, saj uporabnik potrebuje dostop tudi do drugih informacij pod njim.

d) Meni Help ni na pravem mestu.

Rešitev: Pravilen je odgovor b) "Gumbi orodne vrstice so premajhni". Glede na to, da je papirnati prototip nenatančen v smislu modelov "look" in "feel" ter da so ponavadi papirnati prototipi narejeni večje, kot bo dejanski vmesnik v praksi, je malo verjetno, da se bo opazilo, da so gumbi orodne vrstice premajhni.

177. Pri izdelavi prototipov se lahko poslužimo tudi računalniških prototipov. Katere so njihove prednosti in katere slabosti v primerjavi s papirnatimi prototipi? Katere tehnike računalniških prototipov poznate?

Rešitev: Prednosti računalniških prototipov so, da so natančni v smislu modela "look" in v smislu modela "feel": izgled je zelo podoben končnemu izgledu, omogočajo tudi interakcijo, ki se bo uporabljala tudi v končnem izdelku.
Slabosti pa so, da imajo nizko natančnost po globini: velikokrat je implementirana kvečjemu najosnovnejša funkcionalnost, ki omogoča izvedbo določene akcije; tipično so računalniški prototipi horizontalni, pokrivajo značilnosti, ne pa ozadja.

Tehnike računalniških prototipov so: zaporedje slik vmesnika, zaporedje povezanih slik vmesnika, generatorji vmesnikov.

178. Pri načrtovanju uporabniškega vmesnika ste se odločili za papirnati prototip namesto računalniškega prototipa. Naštejte argumente za takšno odločitev. Prijatelj je opazil vaše delo in vam razlaga, česa vse se ne boste mogli naučiti iz papirnatega prototipa. Naštejte, česa se ne boste mogli naučiti iz papirnatega prototipa. Povejte tudi, kateri od obeh prototipov (papirnati prototip ali računalniški prototip) dosega nižjo natančnost po globini.

Rešitev: Argumenti za papirnati prototip:

- hitrejši razvoj, mnogo hitrejše je nekaj skicirati na list papirja, kot pa implementirati računalniški prototip;
- lažje je spreminjati prototip, na primer zbrisati nek detajl in ga drugače implementirati;
- pozornost je usmerjena na "veliko sliko", zanima nas dejanski koncept aplikacije in interakcije z aplikacijo in ne drobni detajli, ki se lahko popravljajo kasneje;
- neprogramerji so lahko v pomoč, saj za izdelavo in testiranje tovrstnih prototipov ni potrebno računalniško predznanje.

Česa se ne moremo naučiti iz papirnatega prototipa:

- modela "look", to je barv, oblik pisav, belih presledkov, saj je vse skicirano na roko, tako da dejanskega končnega izgleda ni mogoče razbrati iz papirnatega prototipa;
- modela "feel", to je implikacij Fittovega zakona oziroma kako bo potekala dejanska interakcija, saj pri testiranju prototipov ne moremo testirati dejanske interakcije, ker računalnik nadomešča oseba, preko katere poteka interakcija;
- odzivnih časov, saj sistem simulira oseba, pri kateri dejanski odzivni časi niso primerljivi z odzivnim časom računalnika;
- ali bo uporabnik opažal drobne spremembe (slepota sprememb), saj je vsaka še najmanjša sprememba na prototipu povzročena s fizično akcijo osebe, ki simulira računalnik, tako da je to dobro vidno;
- ali bo uporabnik več preiskoval ali premišljal, saj so zaradi narave "računalnika" (dejanska oseba) uporabniki, ki testirajo prototip, bolj nagnjeni k razmišljanju in manj preiskujejo.

Računalniški prototip dosega nižjo natančnost po globini.

179. V prvi iteraciji načrtovanja uporabniškega vmesnika ste se odločili za papirnati prototip. Naštejte, kaj vse se lahko naučimo iz papirnatega prototipa.

Rešitev: Iz papirnatega prototipa se lahko naučimo:

- konceptualnega modela: ali uporabnik razume vmesnik in ga zna uporabljati;
- funkcionalnosti: ali so vsebovane vse potrebne značilnosti, ali katere manjkajo (ali vmesnik naredi, kar je potrebno);
- toka navigacije in nalog: ali se uporabniki znajdejo v vmesniku in najdejo želeno akcijo ali informacijo; ali imajo na voljo dovolj informacij za uspešno delo;
- terminologije, ki je uporabljena v vmesniku: ali uporabniki razumejo oznake in so te tudi smiselne;
- vsebine za zaslon: kaj mora biti prikazano.

10 Vrednotenje

180. Naštejte dva pristopa, s katerima se lahko lotimo vrednotenja oziroma testiranja uporabniškega vmesnika. Kateri od obeh pristopov je cenejši?

Rešitev: Testiranja uporabniškega vmesnika se lahko lotimo s hevrističnim vrednotenjem in s testiranjem uporabnikov. Cenejši pristop je hevristično vrednotenje.

181. Opišite probleme pri uporabi testiranja uporabnikov.

Rešitev: Testiranja uporabnikov so lahko draga in časovno potratna. Ponavadi zajemajo novačenje potencialnih uporabnikov za interakcijo s sistemom pod kontroliranimi pogoji. Čeprav to lahko pomaga pri statistični analizi rezultatov, pa lahko vodi v probleme pri zagotavljanju tega, da evalvacija omogoča veljaven vpogled v manj kontrolirane pogoje pri končni implementaciji oziroma uporabi.

182. Naštejte osebe, ki sodelujejo pri testiranju papirnatih prototipov in razložite njihove naloge.

Rešitev: Pri testiranju papirnatih prototipov sodelujejo:

- uporabnik: to je oseba, ki je izbrana tako, da predstavlja dejanskega končnega uporabnika, njegova naloga pa je, da poskuša rešite postavljene naloge;
- pomočnik: to je oseba, ki postavlja naloge uporabniku, poleg tega pa ga lahko spodbuja tudi h glasnemu razmišljanju, če testiranje to zahteva;
- "računalnik": to je oseba, ki pomaga pri simulaciji delovanja računalnika;
- opazovalec: to je oseba, ki spremlja testiranje, dela zapiske in ocenjuje testiranje.

183. Katerega izmed naslednjih problemov bomo najmanj verjetno našli med testiranjem uporabnikov, ki bodo uporabljali papirnati prototip vmesnika, in zakaj je najmanj verjeten?

a) Pomemben gumb orodne vrstice je premajhen in predaleč vstran.

b) Odzivni čas sistema je predolg.

c) Ena od ikon je nerazumljiva.

d) Ni dovolj prostora na zaslonu za vse informacije.

Rešitev: Glede na to, da so papirnati prototipi v smislu modelov "look" in "feel" nenatančni, se lahko zgodi, da se ne bo našlo nobenega izmed zgoraj naštetih problemov, čeprav se nekatere izmed teh problemov vseeno lahko zazna. Najmanj verjeten pa je odgovor b) "Odzivni čas sistema je predolg". Papirnati prototip je tipično vertikalen in dosega visoko natančnost po globini zaradi osebe "računalnik", ki simulira delovanje. Funkcionalnosti sistema bodo tako sicer izvedene, vendar odzivni čas ni realističen, ker je delovanje računalnika simulirano.

184. Kaj je hevristično vrednotenje?

Rešitev: Hevristično vrednotenje je preiskovanje uporabniškega vmesnika na podlagi množice pravil, ki se lahko aplicirajo pri razvoju uporabniških vmesnikov. Lahko se uporabljajo v celotnem razvojnem ciklu, vendar pogosto ne vrnejo tako dobrih rezultatov, kot jih vrne testiranje uporabnikov. Načrtovalci na primer težko dobijo uporabnikovo perspektivo vmesnika z uporabo množice pravil na vmesniku.

185. Kako poteka hevristično vrednotenje?

Rešitev: Hevristično vrednotenje ocenjuje načrt. Načrt je lahko verbalni opis, papirnati prototip, delujoč prototip ali tudi delujoč sistem. Koraki pri hevrističnem vrednotenju so:

a) Zagotovi se seznam hevristik.

b) Ocenjevalcem se omogoči dodatno usposabljanje, če je smiselno.

c) Vsak ocenjevalec dela sam od eno do dve uri.

d) Ocenjevalec preišče vmesnik v dveh prehodih.

e) Ocenjevalec dela zapiske.

f) Ocenjevalec sestavi seznam težav in dobrih lastnosti.

g) Ocenjevalec naredi zaslonske slike težav.

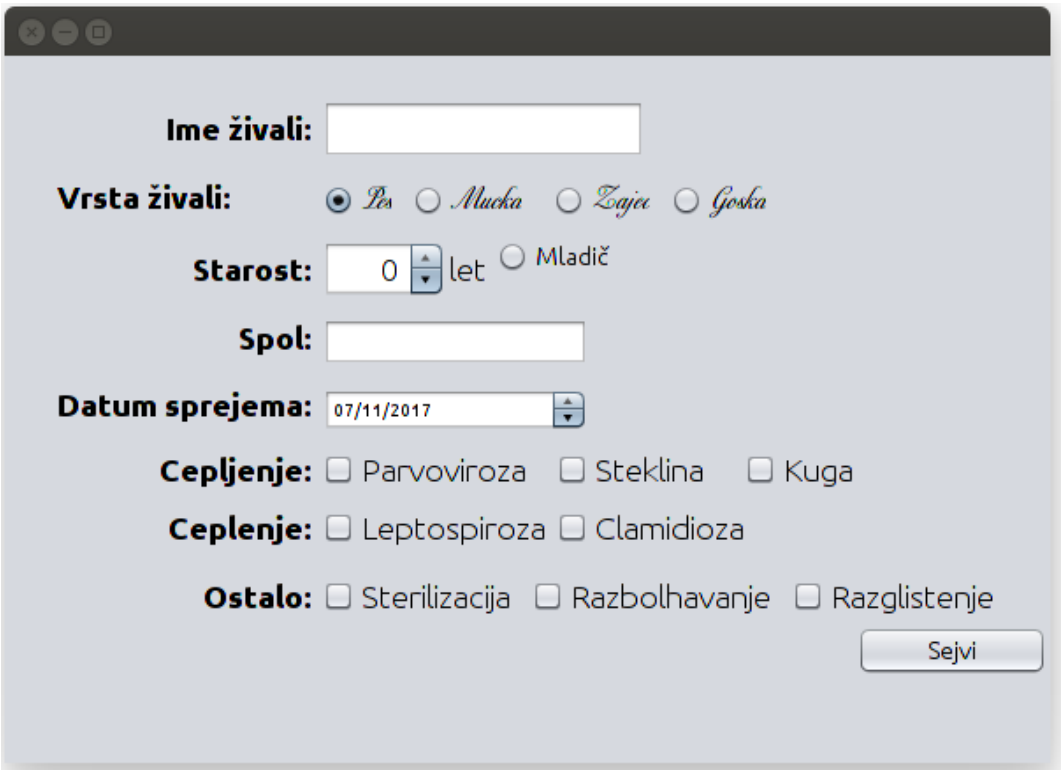
h) Težave, ki so jih našli ocenjevalci, se združijo.

i) Vsak ocenjevalec oceni resnost za vsako težavo v združenem seznamu.

j) Ocene resnosti se povprečijo, težave se uredijo po padajoči resnosti.

k) Organizira se sestanek za podajo predlogov o ponovnem načrtovanju.

186. Na sliki spodaj (slika 10.1) je prikazan primer uporabniškega vmesnika za sprejem živali v azil. Hevristično ovrednotite ta vmesnik na podlagi Nielsenovih principov načrtovanja uporabniških vmesnikov in sestavite seznam težav, kjer vključite tudi opis, razlago težave in mogoč način za odpravo težave.



Slika 10.1 Slika uporabniškega vmesnika za sprejem živali v azil.

Rešitev: Pri hevrističnem vrednotenju vmesnika smo naleteli na naslednje težave:

- Vmesnik nima naslova: naslova aplikacije v "sistemski" vrstici ni, kar otežuje prepoznavanje aplikacije. Pri velikem številu odprtih oken bo uporabnik težko našel želeno okno, če je zakrito. Prav tako omogoča naslov aplikacije tudi lažje iskanje minimizirane aplikacije, saj se na nekaterih sistemih prikaže naslov aplikacije, če se uporabnik s kurzorjem miške postavi nad ikono.
  Težavo se odpravi tako, da se doda naslov aplikacije.
- Vmesnik ne sporoča, čemu je namenjen: manjka pojasnilo, ki bi uporabniku nedvoumno povedalo, kakšen je namen aplikacije in čemu služijo

posamezna vnosna polja. Novi uporabniki se morajo navaditi aplikacije in njene uporabe. Ker ni pojasnila, čemu je aplikacija namenjena, imajo lahko novi uporabniki težave pri uporabi, obenem pa je lahko kakšno vnosno polje tudi dvoumno.
Težavo se lahko odpravi tako, da se doda na vrh delovnega področja aplikacije oznaka s pojasnilom namena aplikacije, na primer "Vnos živali".

- Oznaka "Ime živali": beseda "žival" sicer zagotavlja nedvoumno pojasnilo za zahtevani vnos, vendar je nekonsistentna z ostalimi oznakami, ki te besede nimajo.
  Težavo se lahko odpravi z dodano oznako za pojasnilo namena aplikacije in zamenjavo oznake "Ime živali" z oznako "Ime".

- Oznaka "Vrsta živali": beseda "žival" sicer zagotavlja nedvoumno pojasnilo za zahtevani vnos, vendar je nekonsistentna z ostalimi oznakami, ki te besede nimajo.
  Težavo se lahko odpravi z dodano oznako za pojasnilo namena aplikacije in zamenjavo oznake "Vrsta živali" z oznako "Vrsta".

- Oznaka "Vrsta živali": vertikalna poravnava oznake je po levem robu z oznako "Datum sprejema" in ni v skladu s poravnavami ostalih oznak, ki so vertikalno poravnane po desnem robu.
  Težavo se odpravi tako, da se oznaka "Vrsta živali" poravna po desnem robu z ostalimi oznakami.

- Polje stikal za izključujočo izbiro za vnos vrste živali: pisava pri polju stikal za izključujočo izbiro je drugačna, kot so pisave pri drugih vnosnih poljih.
  Težavo se odpravi tako, da se uporabi enako pisavo kot pri ostalih vnosnih poljih.

- Polje stikal za izključujočo izbiro za vnos vrste živali: drugo stikalo ima pridružen napis "Mucka", kar je pomanjševalnica, poleg tega pa se pogosto uporablja za definiranje mlade mačke. Napis je nekonsistenten z ostalimi napisi v tem polju stikal. Zaradi pomanjševalnice pa tudi ni očitno, kako lahko uporabnik aplikacije vnese starejšo mačko.
  Težavo se odpravi tako, da se namesto pomanjševalnice "Mucka" uporabi besedo "Mačka".

- Stikalo za izključujočo izbiro "Mladič": izbira podobe ni primerna. Podoba je namenjena samo izbiri oziroma neizbiri, za kar je primerna podoba stikalo, med tem ko je stikalo za izključujočo izbiro namenjeno izbiri med alternativami.
  Težavo se odpravi tako, da se stikalo za izključujočo izbiro zamenja s stikalom.

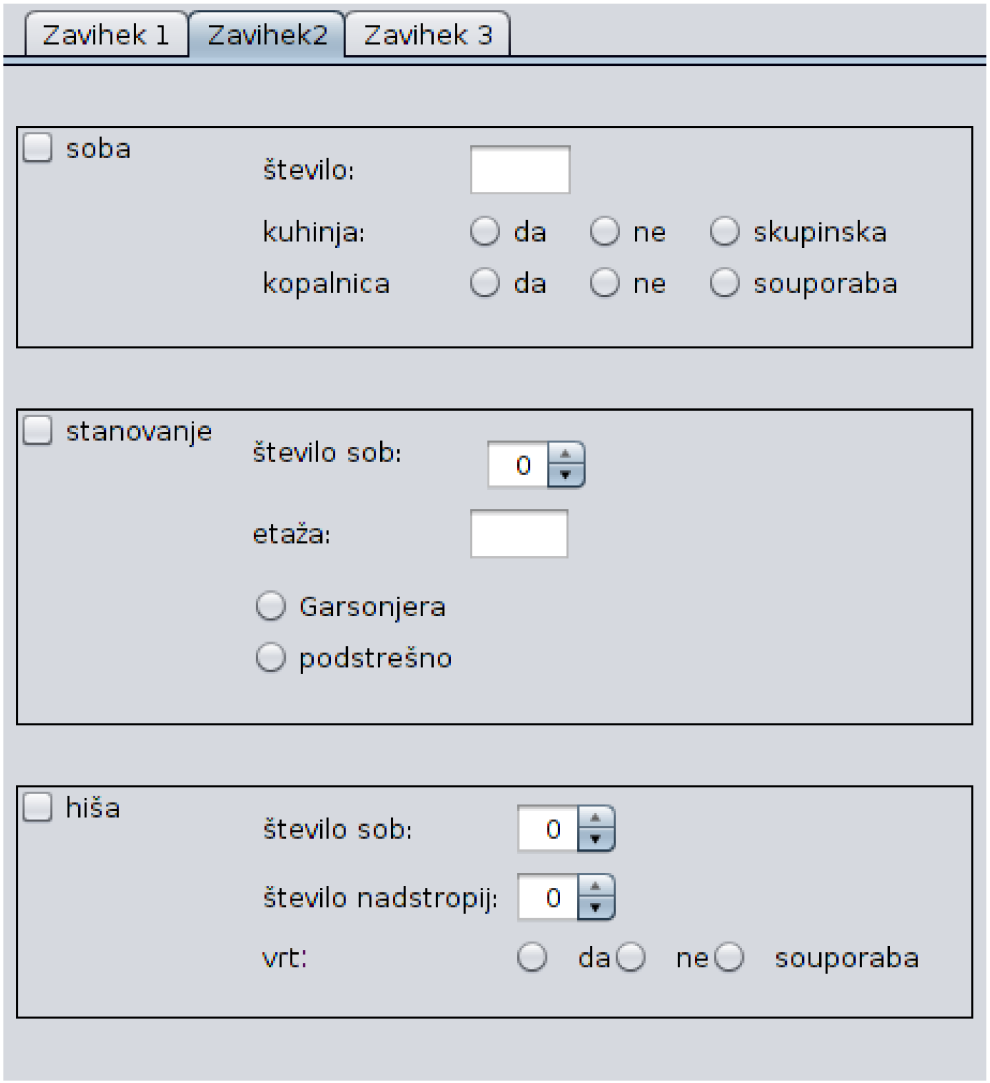
- Stikalo za izključujočo izbiro "Mladič": velikost pisave pri tem stikalu je manjša, kot je pri ostalih vnosnih poljih.
  Težavo se odpravi tako, da se uporabi velikost pisave kot pri ostalih vnosnih poljih.
- Stikalo za izključujočo izbiro "Mladič": horizontalna poravnava stikala za izključujočo izbiro s krožnim poljem in s pripadajočo oznako je po zgornjem robu, kar je nekonsistentno z ostalimi horizontalnimi poravnavami.
  Težavo se odpravi tako, da se stikalo za izključujočo izbiro poravna po spodnjem robu s krožnim poljem in s pripadajočo oznako.
- Vnosno polje za spol: izbira podobe za vnosno polje spola za sprejeto žival je neprimerna. Uporabniku omogoča vnašanje poljubnega niza.
  Težavo se odpravi tako, da se vnosno polje vrstica teksta zamenja s podobo, ki omogoča izbiro, na primer polje stikal ali pa seznam.
- Krožno polje za vnos datuma sprejema: pisava je premajhna. Izbrana pisava v krožnem polju je tako majhna, da je težko razbrati, kateri datum je vnešen.
  Težavo se reši z izbiro večje pisave, primerljive z ostalimi vnosnimi polji.
- Oznaka "Ceplenje": napis v oznaki je napisan napačno in nekonsistentno z oznako višje, kjer se nahaja napis "Cepljenje".
  Težavo se odpravi tako, da se napis "Ceplenje" zamenja z napisom "Cepljenje".
- Oznaka "Ceplenje": nepotrebno podvajanje oznake.
  Težavo se odpravi tako, da se oznaka odstrani, da ne pride do nejasnosti. Ker gre za enako akcijo, kot je v vrstici višje ("Cepljenje"), vnosna polja sodijo k zgornjim vnosnim poljem.
- Vrstica z oznako "Ceplenje": vrstica sodi k zgornji vrstici ("Cepljenje"). Razmak med obema vrsticama je enak kot med ostalimi vrsticami. Zato da ne pride do nejasnosti glede grupiranja oziroma zato da se poudari, da vrstici sodita skupaj, morata biti vrstici vizualno grupirani skupaj.
  Težavo se odpravi tako, da se med obema vrsticama uporabijo manjši beli presledki kot sicer, s čimer vizualno grupiramo vrstici skupaj.
- Vrstica "Ostalo": razmik do zgornje vrstice je nekonsistenten in večji, kot je razmik med drugimi vrsticami. Tako je vrstica brez razloga vizualno ločena od ostalih vrstic.
  Težavo se odpravi tako, da se uporabijo konsistentni beli presledki, kot so med ostalimi vrsticami.
- Gumb "Sejvi": napis "Sejvi" v gumbu ni primeren. Napis je v slengu, ki za vmesnik ni primeren.
  Težavo se odpravi tako, da se napis zamenja z napisom "Shrani".

- Gumb "Sejvi" je preblizu zgornje vrstice in preblizu roba. Glede na bližino zgornje vrstice izgleda tako, kot da je gumb namenjen samo za shranjevanje podatkov, ki se nanašajo na ostale storitve ("Sterilizacija", "Razbolhavanje", "Razglistenje").
  Težavo se odpravi tako, da se gumb prestavi nižje in stran od roba, tako da je vizualno ločen od vrstice "Ostalo".
- Manjka gumb, ki bi uporabniku omogočal preklic vnosa podatkov.
  Težavo se odpravi tako, da se doda gumb "Prekliči", s katerim lahko zahtevamo preklic vnosa podatkov.
- Manjka statusna vrstica, kjer bi uporabnik lahko dobil statusno informacijo.
  Težavo se odpravi tako, da se doda statusna vrstica v spodnji del uporabniškega vmesnika.

187. Na sliki spodaj (slika 10.2) je prikazan primer dela uporabniškega vmesnika za oddajo stanovanjske enote. Hevristično ovrednotite ta vmesnik na podlagi Nielsenovih principov načrtovanja uporabniških vmesnikov in sestavite seznam težav, kjer vključite tudi opis, razlago težave in mogoč način za odpravo težave.

Rešitev: Pri hevrističnem vrednotenju smo naleteli na naslednje težave:

- Naslov drugega zavihka ("Zavihek2"): naslov je napisan nekonsistentno glede na ostala zavihka, med besedico "Zavihek" in številko zavihka tukaj ni presledka, pri ostalih dveh pa presledek je.
  Težavo se odpravi tako, da se doda presledek med "Zavihek" in "2".
- Naslov prvega, drugega oziroma tretjega zavihka ("Zavihek 1", "Zavihek2" oziroma "Zavihek 3"): naslov uporabniku ne pove nič o vsebini zavihka in je kot tak nepotreben oziroma neprimeren.
  Težavo se odpravi tako, da se namesto naslovov "Zavihek 1", "Zavihek2" oziroma "Zavihek 3" izberejo imena, ki uporabniku nedvoumno povedo, čemu je zavihek namenjen.
- Stikala "soba", "stanovanje", "hiša": neprimerna izbira podob. Aplikacija je namenjena oddaji oglasa za enostanovanjsko enoto, torej ali sobo ali stanovanje ali hišo, ne pa več različnih enot, kar je z izbiro stikala veljavna možnost.
  Težavo se odpravi tako, da se podobe stikala zamenjajo s podobami stikala za izključujočo izbiro.
- Vnosno polje za vnos števila sob: neprimerna izbira vnosnega polja. Izbira vnosnega polja vrstica teksta omogoča uporabniku vnos poljubnega niza. Poleg tega je izbira vnosnega polja tudi nekonsistentna z izbiro vnosa števila sob pri stanovanju.



Slika 10.2 Slika dela uporabniškega vmesnika za oddajo stanovanjske enote.

Težavo se odpravi tako, da se vnosno polje vrstica teksta zamenja s krožnim poljem.

- Napis "skupinska" pri stikalu za izključujočo izbiro v polju stikal za kuhinjo v okvirju "soba": nekonsistentna uporaba terminologije. Napis je nekonsistenten z napisom "souporaba" (za souporabo kopalnice v okvirju "soba" oziroma vrta v okvirju "hiša").
  Težavo se odpravi tako, da se napis "skupinska" zamenja z napisom "souporaba".
- Oznaka "kopalnica" v okvirju "soba": napis v oznaki je napisan nekonsistentno, saj je pri vseh ostalih napisih v oznakah dodano dvopičje na

koncu.
Težavo se odpravi tako, da se napisu "kopalnica" doda dvopičje.

- Krožno polje za vnos števila sob v okvirju stanovanje: krožno polje je vertikalno neporavnano z vnosnim poljem za vnos številke etaže.
  Težavo se odpravi tako, da se krožno polje poravna vertikalno na levi in desni strani z vnosnim poljem za vnos številke etaže. Po potrebi se lahko spremeni tudi velikost krožnega polja.
- Oznaka "etaža" v okvirju stanovanje: nekonsistentna uporaba terminologije. V okvirju "hiša" se uporablja izraz "nadstropje". Čeprav "etaža" in "nadstropje" nista sopomenki, se pri oglasih uporablja izraz "nadstropje", saj se iz številke etaže težko ugotovi, za katero nadstropje dejansko gre.
  Težavo se odpravi tako, da se "etaža" nadomesti z "nadstropje".
- Vnosno polje za vnos "etaže": neprimerna izbira vnosnega polja. Izbira vnosnega polja vrstica teksta omogoča uporabniku vnos poljubnega niza. Poleg tega je izbira vnosnega polja tudi nekonsistentna z izbiro vnosa nadstropja v okvirju "hiša".
  Težavo se odpravi tako, da se vnosno polje vrstica teksta zamenja s krožnim poljem.
- Polje stikal za izključujočo izbiro "Garsonjera" in "podstrešno" v okvirju "stanovanje": neprimerna izbira podob. Možnosti garsonjera in podstrešno se ne izključujeta, čeprav sta v vmesniku prikazani na tak način.
  Težavo se odpravi tako, da se polje stikal za izključujočo izbiro zamenja s poljem stikal.
- Napis "Garsonjera" v polju stikal za izključujočo izbiro v okvirju "stanovanje": napis je nekonsistenten. Za razliko od vseh ostalih besedil je to besedilo napisano z veliko začetnico.
  Težavo se odpravi tako, da se besedilo "Garsonjera" zamenja z napisom "garsonjera".
- Polje stikal za izključujočo izbiro v vrstici za "vrt" v okvirju "hiša": neprimerno aranžiranje stikal za izključujočo izbiro. Stikala za izključujočo izbiro so preveč blizu skupaj, tako da so napisi pri posameznem stikalu za izključujočo izbiro bližje sosednjemu stikalu za izključujočo izbiro. Zaradi tega je dvoumno, h kateremu stikalu za izključujočo izbiro kateri napis pripada.
  Težavo se odpravi tako, da se doda več belih presledkov med stikala za izključujočo izbiro in manj med napis in stikalo za izključujočo izbiro, s čimer se bo doseglo primerno grupiranje.
- Manjkajoče podobe za navigacijo med zavihki: v vmesniku manjkata dve podobi, ki bi uporabniku omogočali enostavno prehajanje med sosednjimi zavihki (nazaj in naprej).

Težavo se odpravi tako, da se doda dva gumba pod spodnji okvir, s katerimi se omogoči navigacija med sosednjimi zavihki.

188. Naštejte bistvene razlike med hevrističnim vrednotenjem in testiranjem uporabnikov.

Rešitev: Razlike med hevrističnim vrednotenjem in testiranjem uporabnikov so:

- hevristično vrednotenje vršijo eksperti uporabnosti, pri testiranju uporabnikov pa razvijalci najamejo reprezentativne uporabnike;
- hevristično vrednotenje je metoda preiskovanja vmesnika v smislu principov in navodil načrtovanja, testiranje uporabnikov je ocenjevanje uporabnikov pri delu;
- hevristično vrednotenje lahko najde probleme, ki jih pri testiranju uporabnikov najdemo veliko težje, npr: konsistentnost pisav, implikacije Fittovega zakona.

189. Naštejte merljive principe (dimenzije) uporabnosti.

Rešitev: Merljivi principi oziroma dimenzije uporabnosti so:

- naučljivost,
- učinkovitost,
- varnost,
- estetika in ergonomija/zadovoljstvo in udobje ter utrujenost.

190. Za vsakega izmed merljivih principov (dimenzij) uporabnosti napišite, kaj pojasnjuje.

Rešitev: Merljivi principi uporabnosti:

- Naučljivost pojasnjuje oziroma odgovarja na vprašanje: "Ali se je lahko za naučiti?"
- Učinkovitost odgovarja na vprašanje: "Potem, ko je naučeno, ali se da hitro uporabljati?"
- Varnost odgovarja na vprašanje: "Ali so napake redke in popravljive?"
- Princip "estetika in ergonomija" pa odgovarja na vprašanje: "Ali je prijetno za uporabo?"

191. Naštejte merljive principe (dimenzije) uporabnosti, nato pa za vsak princip uporabnosti navedite vsaj en Nielsenov princip načrtovanja uporabniških vmesnikov, ki pomaga izboljšati ta princip uporabnosti.

Rešitev: Merljivi principi uporabnosti:

- naučljivost: prilagodi se realnemu svetu; konsistentnost in standardi; estetika in minimalistično načrtovanje; raje prepoznaj, kot si zapomni;
- učinkovitost: fleksibilnost in učinkovitost; raje prepoznaj, kot si zapomni; konsistentnost in standardi;
- varnost: izogibanje napakam; javljanje napak, diagnoza, reševanje; raje prepoznaj, kot si zapomni; konsistentnost in standardi;
- estetika in ergonomija/zadovoljstvo in udobje ter utrujenost: estetika in minimalistično načrtovanje; raje prepoznaj, kot si zapomni; konsistentnost in standardi.

192. Razložite faktorja uporabnosti (meri zmogljivosti) zadovoljstvo in naučljivost.

Rešitev: Zadovoljstvo poda oceno odstotka uporabnikov, ki po N danih nalogah podajo oceno zadovoljstva O. Pove uporabnikovo oceno glede na njegovo zadovoljstvo oziroma ali je vmesnik prijeten za uporabo.
Naučljivost poda oceno odstotka uporabnikov, ki po N danih nalogah in po M urah učenja z danim predznanjem uspešno osvojijo znanje. Pove, ali se uporabniki naučijo uporabe vmesnika oziroma ali je vmesnik intuitiven in enostaven za naučiti (konsistentnost z ostalimi podobnimi vmesniki).

193. Razložite faktorja uporabnosti (meri zmogljivosti) varnost in učinkovitost.

Rešitev: Varnost poda oceno odstotka uporabnikov, ki po N danih nalogah podajo oceno varnosti V. Pove, ali uporabniki uporabljajo vmesnik, ne da bi delali manjše (lahko popravljive) oziroma večje (težko popravljive/nepopravljive) napake oziroma ali je vmesnik varen za uporabo.
Učinkovitost poda oceno odstotka uporabnikov, ki uspešno izvršijo N danih nalog v T minutah. Pove, kako uspešno uporabnik opravlja svoja opravila oziroma ali omogoča učinkovito delo uporabnika, ko ta že pozna vmesnik.

194. Kakšna je razlika med faktorjema uporabnosti (merama zmogljivosti) naučljivost in učinkovitost?

Rešitev: Pri učinkovitosti je čas za izvedbo nalog omejen, medtem ko naučljivost govori samo o tem, ali po določenem času uporabnik lahko opravi nalogo. Učinkovitost pove, kako uspešno uporabnik opravlja svoja opravila. Naučljivost pa pove, do kakšne mere se je vmesnika na podlagi predznanja mogoče naučiti.

195. Naštejte faktorje uporabnosti in za vsakega od njih povejte, kaj oceni ob testiranju uporabnikov.

Rešitev: Faktorji uporabnosti so:

a) naučljivost: pove, ali se uporabniki naučijo uporabe vmesnika;

b) učinkovitost: pove, kako uspešno uporabnik opravlja opravila;

c) varnost: pove, kako dobro omogoča vmesnik izogibanje napakam in reševanje iz napak;

d) zadovoljstvo: pove uporabnikovo oceno glede na njegovo zadovoljstvo.

11 Načrtovanje spletnih in mobilnih vmesnikov

196. Naštejte navodila za načrtovanje spletnih strani.

Rešitev: Navodila za načrtovanje spletnih strani so:

- Predvidevaj profil obiskovalcev.
- Definiraj uporabnikove naloge.
- Pomagaj uporabnikom pri navigaciji in iskanju.
- Omogoči uporabnikom hitro delo.
- Zagotovi konsistentnost.

197. Naštejte lastnosti, ki jih je potrebno upoštevati pri izdelavi spletnih strani, ker pritegnejo uporabnika.

Rešitev: Za zagotavljanje atraktivnosti spletnih strani za uporabnike morajo biti izpolnjeni sledeči kriteriji (angleški akronim je "HOMERUN"):

H kvalitetna vsebina (angl. "**H**igh quality content");

O pogosto posodabljanje (angl. "**O**ften updated");

M minimalen čas prenosa (angl. "**M**inimal download time");

E enostavnost uporabe (angl. "**E**ase of use");

R relevantno za uporabnike (angl. "**R**elevent to users' needs");

U edinstven za spletni medij (angl. "**U**nique to the online medium);

N spletno usmerjena kultura podjetja oziroma spletne strani niso samo še en dodatek normalnemu poslovanju (angl. "**N**et-centric corporate culture").

198. Naštejte vidike, ki jih je potrebno upoštevati pri razvoju spletnih strani.

Rešitev: Vidiki, ki jih je potrebno upoštevati, so:

- Uporabnik nima potrpljenja s slabimi stranmi.

- Uporabnik ne želi uporabljati drsnikov pri pregledu in uporabi strani.
- Uporabniki na spletu ne želijo veliko brati.

199. Pri načrtovanju spletnih strani je pomembno, da zmanjšamo potreben čas prenosa spletne strani. Zakaj je to pomembno?

Rešitev: Zmanjševanje časa je potrebno iz več razlogov. Eden izmed razlogov je ta, da imajo lahko uporabniki slabo povezavo, kar lahko posledično pomeni dolgotrajno prenašanje vsebine. Če se stran prenaša predolgo, to povzroči slabo uporabniško izkušnjo. Pomemben je tudi čas med začetkom akcije in dejanskim odzivom. Če je ta čas zadosti majhen, potem ga uporabnik sploh ne bo opazil in bo spletna stran delovala zelo odzivno. Če je ta čas dolg, pa se lahko zgodi, da bo uporabnik zapustil spletno stran.

200. Pri načrtovanju spletnih strani je pomembno, da zmanjšamo potreben čas prenosa spletne strani. Kako lahko to naredimo?

Rešitev: To lahko naredimo tako, da zmanjšamo količino podatkov, potrebnih za prenos. Če uporabljamo razne knjižnice, je tako primerno, da uporabimo njihove minificirane verzije in ne prenašamo knjižnic, ki jih sploh ne potrebujemo. Prav tako ne uporabljamo multimedijskih vsebin, ki nimajo uporabne vrednosti za spletno stran. Pri uporabi slik prenašamo slike, ki so primerne velikosti za stran, večje slike pa lahko ponudimo uporabniku kot alternativen prenos. Prav tako lahko zmanjšamo čas, ki je potreben za prenos, in sicer tako, da uporabljamo knjižnice, ki so na voljo na omrežjih za zagotavljanje vsebin (angl. "Content Delivery Network", CDN), če so tam na voljo, saj ta omrežja omogočajo prenos podatkov s strežnikov, ki so blizu uporabniku.

201. Pri izdelavi spletnih strani lahko implementirate odzivni dizajn. Kaj je to odzivni dizajn?

Rešitev: Odzivni dizajn je rezultat pristopa k načrtovanju spletnih vmesnikov oziroma spletnih strani, kjer se prikaz vsebine oziroma strani prilagaja napravi, s katero dostopamo do spletnih strani. Pri tem se spremeni izgled (postavitev gradnikov) za različne velikosti zaslonov (tipično pri dveh fiksno postavljenih dimenzijah zaslona oziroma točkah preloma: angl. "breakpoints"). Za zagotavljanje boljše uporabniške izkušnje pa se lahko spremenijo tudi multimedijske datoteke, vendar tako, da so še vedno posredovane vse potrebne informacije (slike manjših velikosti, bolj stisnjene avdio ali video datoteke in podobno).

202. S pojavom različnih prenosnih naprav, kot so pametni telefoni in tablice, se je pojavila možnost za dostopanje do spletnih vmesnikov tudi preko teh naprav. Naštejte in na kratko opišite nekatere pristope (na strani spletnih strani ali mobilnih naprav), ki se uporabljajo za prikaz spletnih vmesnikov na teh napravah.

Rešitev: Prvi pristop za prikaz spletnih vmesnikov, ki je tudi najstarejši, je pristop za spletne strani oziroma spletišča, ki se ne prilagajajo napravi, s katero dostopamo. Pri tem pristopu se prikaže spletna stran na mobilni napravi, ki jo je pa težko uporabljati. Pri tem pristopu se uporabljajo razne geste (na primer: angl. "pinch-to-zoom"), s pomočjo katerih lahko del strani približamo in prikažemo. Izmed vseh pristopov je ta pristop za uporabnika najmanj primeren, saj zahteva dodatno manipulacijo. Drugi pristop je pristop z mobilnimi spletnimi stranmi ("m.spletnastran"), to je stranmi, ki so prilagojene prikazu na mobilnih napravah. Pri tem pristopu strežnik zahtevek za iskano stran preusmeri na stran, ki je prilagojena mobilnim napravam in naj bi zagotavljala enako vsebino kot originalna stran. Pomanjkljivost tega pristopa je, da zahteva dodatno delo za ustvarjanje mobilnih spletnih strani. Tretji pristop je, da se od uporabnika zahteva nalaganje mobilne aplikacije, s katero lahko potem dostopa do vsebin na nekem spletišču. Pomanjkljivost tega pristopa je, da ima lahko uporabnik veliko število aplikacij, vsako za svoje spletišče. Poleg tega pa je potrebno zagotoviti aplikacije za različne platforme, s katerimi lahko uporabnik dostopa do spletnih strani. Četrti pristop pa je odzivni dizajn, kjer imamo samo eno spletno stran, katere izgled se prilagaja napravi, s katero dostopamo do spletnih strani.

203. Pri dizajnu spletnih strani za različne naprave se pojavita pojma prilagodljivi (angl. "scalable") dizajn ter odzivni (angl. "responsive") dizajn. Kako se ta dva pojma razlikujeta?

Rešitev: Pri prilagodljivem dizajnu gre za to, da se vsebina na spletni strani prilagaja glede na velikost zaslona s skaliranjem, pri tem pa ostane osnovni izgled strani nespremenjen, ne glede na velikost zaslona naprave. Pri tem pristopu se elementu na spletni strani določi dimenzija v relativnih merah, na primer v odstotkih velikosti okna ali nadrejenega elementa, tako da se dimenzija elementa v pikslih spreminja relativno glede na velikost okna oziroma nadrejenega elementa, za katerega veljajo enaka pravila.
Pri odzivnem dizajnu pa se stran prilagaja dimenziji zaslonov v nekaj korakih glede na območje dimenzije zaslona (na primer za majhne širine zaslonov, srednje širine zaslonov oziroma velike širine zaslonov), poleg tega pa lahko znotraj posameznega območja dimenzije poteka še prilagajanje trenutni dimenziji zaslona (prilagodljiv dizajn). Če imamo v trenutni strukturi spletne

strani prilagodljiv vsebovalnik, se bo njegova širina spreminjala zvezno, tako kot tudi celotna struktura, če pa imamo odzivni vsebovalnik, se bo njegova širina spreminjala zvezno glede na širino strani in trenutno obliko strani, ki se lahko spreminja po korakih.

204. Katere tehnologije se uporabljajo za dosego odzivnega dizajna?

Rešitev: Za osnovno strukturo spletne strani moramo uporabiti HTML. Za implementacijo odzivnega dizajna moramo uporabiti medijske poizvedbe v okviru prekrivnih slogov (kaskadne slogovne predloge), s pomočjo katerih lahko ugotovimo, za kakšno napravo gre, potem pa na podlagi teh medijskih poizvedb posredujemo ustrezen prekrivni slog za napravo. Pri tem si lahko pomagamo še s programskim jezikom JavaScript.

205. Pri odzivnem dizajnu se pojavlja pojem "najprej mobilno". Razložite ta pojem.

Rešitev: Najprej mobilno ne pomeni da, ko začnemo načrtovati spletno stran, najprej naredimo dizajn za mobilne naprave, temveč pomeni, da medijske poizvedbe med pisanjem uredimo po velikosti zaslonov posameznih naprav. Pri tem najprej poskrbimo za najmanjše naprave, ki so ponavadi mobilne naprave, to pa tudi sovpada z zmogljivostmi naprave, povezano pa je ponavadi tudi s količino podatkov, ki jih moramo prenesti.

206. Zakaj je pri odzivnem dizajnu pomembno, da napišemo medijske poizvedbe najprej za mobilne naprave in šele nato za večje naprave?

Rešitev: Pri odzivnem dizajnu je vsebina, ki se prikaže na napravi, prilagojena tej napravi, vendar moramo za vsako spletno stran definirati privzeti prikaz, to je prikaz, ki se prikaže v primeru, ko ne obstaja prilagoditev za določeno napravo. Nato pa definiramo še posamezne posebne nastavitve, ki privzeti prikaz na novo definirajo za neko napravo. Zaželeno je, da je privzeti prikaz tak, da omogoča osnoven prikaz. V obratnem primeru se lahko namreč zgodi, da se prenašajo vsebine, ki jih uporabnik sploh ne vidi. Tako bi se lahko zgodilo, da se slika ozadja, ki morda za vsebino ni pomembna, nadomesti z nevtralno barvo ozadja pri manjših napravah. Prav tako so pogosto multimedijske vsebine prilagojene manjšim napravam, na primer lahko imamo manjše slike. Če najprej prenesemo vsebine, ki so namenjene mobilnim napravam (na primer barvo ozadja namesto slike) in šele nato vsebine, namenjene večjim naprava (sliko ozadja), bo nova vsebina nadomestila prvotno (slika čez barvo ozadja). V obratnem vrstnem redu pa bi se najprej prenesla vsebina za večje naprave (slika), nato pa še vsebina za manjše naprave (samo barva ozadja),

ki bi nadomestila prvotno prenešeno vsebino, kar bi bilo potratno.

207. Kaj so prednosti odzivnega dizajna spletnih vmesnikov v primerjavi s klasičnimi spletnimi vmesniki?

Rešitev: Pri klasičnih spletnih vmesnikih je bil dizajn prilagojen enemu tipu zaslona, na primer namiznim računalnikom. Če smo hoteli do posameznih strani dostopati z drugimi napravami, kot so mobilne naprave, ki imajo tipično manjše zaslone, je bilo potrebno za ogled uporabljati razne geste oziroma kretnje, s katerimi je bilo mogoče povečati in primerno prikazati želeno vsebino. Pri odzivnem dizajnu pa se prikaz vsebine prilagodi napravi, s katero dostopamo do spletnega vmesnika, tako da je dostop do vsebine bistveno lažji.

208. Pri izdelavi mobilnih aplikacij se lahko poslužimo več pristopov. Naštejte tri in jih na kratko opišite.

Rešitev: Nekateri pristopi, ki jih lahko uporabimo pri izdelavi mobilnih aplikacij:

- Izvorna mobilna aplikacija je aplikacija, ki je napisana namensko za neko mobilno platformo in je tej platformi tudi prilagojena. Prednost tega pristopa je, da lahko aplikacije izkoristijo funkcionalnosti, ki jih nudi posamezna platforma. Slabost je, da če želimo imeti aplikacijo na več platformah, potem moramo razvijati aplikacijo za vsako platformo posebej. Aplikacijo naložimo na sistem iz repozitorija.
- Spletna aplikacija je aplikacija, do katere dostopamo z brskalnikom, v katerem se tudi izvaja. Teh aplikacij ni potrebno namestiti na sistem. Prednost tega pristopa je, da se aplikacije lahko izvajajo na poljubni platformi. Za to, da lahko do aplikacije dostopamo, pa moramo imeti povezavo z internetom. Te aplikacije tudi ne morejo izkoriščati vseh funkcionalnosti, ki jih ponuja posamezna mobilna platforma.
- Hibridne aplikacije pa so aplikacije, ki so mešanica mobilnih in spletnih aplikacij. So aplikacije, napisane v spletnih tehnologijah (HTML, CSS, JavaScript), vendar se izvajajo v namenskih vsebovalnikih. Namestimo jih na sistem iz repozitorija, izvajajo se v vmesniku, ki je vdelan v aplikacijo, lahko pa tudi dostopajo do funkcionalnosti platforme. Prednost hibridnih aplikacij je tudi ta, da omogočajo večplatformni razvoj.

Literatura

[1] Andrews K. (2017): 'Human-Computer Interaction, Course notes', *Technical University of Graz*. Dosegljivo: **http://courses.iicm.tugraz.at/hci/hci.pdf**.

[2] Cowan N. (2001): 'The magical number 4 in short-term memory: a reconsideration of mental storage capacity', *Behav Brain Sci*, 24(1):87-114.

[3] Galitz W. O. (1994): 'It's Time To Clean Your Windows: Designing GUIs That Work', John Wiley & Sons, New York, USA.

[4] Galitz W. O. (2002): 'The Essential Guide to User Interface Design, Second Edition', John Wiley & Sons, New York, USA

[5] Körner C. (2016): 'Learning Responsive Data Visualization', Packt Publishing, Birmingham, UK.

[6] Mandel T. (1997): 'The Elements of User Interface Design', John Wiley & Sons, Inc., New York, USA.

[7] Miller R. (2017): 'Course 6.831: User Interface Design & Implementation, Readings', *Massachusetts Institute of Technology*. Dosegljivo: **http://web.mit.edu/6.813/www/sp17/**.

[8] Norman D. A. (2002): 'The Design of Everyday Things', Basic Books, New York, USA.

[9] Reimann R., Cronin D. (2007): 'About Face 3: The Essentials of Interaction Design', Wiley Publishing, Indianapolis, USA.

[10] Schneiderman B., Plaisant C. (2010): 'Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5th edition', Addison Wesly, Upper Saddle River, USA.

[11] Sebesta R. W. (2015): 'Programming the World Wide Web, 8th edition', Pearson Education, New Jersey, USA.

[12] Stone D., Jarrett C., Woodroffe M., Minocha, S. (2005): 'User Interface Design and Evaluation', Morgan-Kaufman, San Francisco, USA.