Šolski center Celje,
srednja šola za kemijo, elektrotehniko in računalništvo

Nadzorni vmesnik

za operacijski sistem Windows

Raziskovalna naloga

Avtorja:

Marko GLUHAK

Dejan GREGORC

Mentor:

Gorazd Breznik

Celje, 2017

IZJAVA*

Mentor (-ica) , GORAZD BREZNIK , v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi naslovom
NADZORNI VMESNIK ZA OPERACIJSKI SISTEM WINDOWS,
katere avtorji (-ice) so MARKO GLUHAK, DEJAN GREGORC, _____ _____ :

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo (-ičino) dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje

Celje, 13.3.2017

žig šole

Podpis mentorja(-ice)

Podpis odgovorne osebe

*

POJASNILO

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja(-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja(-ice) fotografskega gradiva, katerega ni avtor(-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

Kazalo

Kazalo slik

Zahvala

Najprej bi se rada zahvalila najinemu mentorju, ki nama je s svojim znanjem in izkušnjami pomagal pri pripravi naloge, da sva lahko pripravila uspešen izdelek. Prav tako bi se rada zahvalila prijatelju, ki nama je dal idejo, kako urejati in montirati slike, saj brez njegove pomoči ne bi nikoli potreboval tega programa, ki je sedaj del vsakdana.

Povzetek

V raziskovalni nalogi smo se odločili narediti okensko aplikacijo za Windows v Visual Studiu 2013, ki uporablja programski jezik C#. Aplikacija bo poenostavila pregled aplikacij, v operacijskem sistemu. Program nudi med drugimi tudi možnosti odjave, zaklepa, ponovnega zagona in zaustavitve sistema. Cilji programa so, poenostavljen dostop do aplikacij v Windows okolju, uporabnika vključiti v oblikovanje izgleda grafičnega vmesnika (GUI – Graphic User Interface) in pospešiti zagon računalnika, ki ima na voljo manj sredstev za pogon sistema.

1. Uvod

Ste si kdaj zaželeli, da bi se vaš osebni računalnik zagnal hitreje, brez vseh nepotrebnosti, ki upočasnjujejo njegov zagon? Ste kdaj želeli odstraniti popolnoma vse ikone iz namizja? Ste kdaj imeli prelepo ozadje, ki ga niste mogli obdržati zaradi prevelikega števila ikon? Ste želeli imeti pregled nad vsemi vašimi programi, ki jih uporabljate in jih spraviti v predalčke, da jih boste lažje našli?

Starejši računalniki se običajno počasneje zaganjajo, sploh če nimajo na voljo sposobnega procesorja oziroma SSD naprave, ki bi omogočala optimalen zagon. Ikone, bližnjice in mape na namizju je zelo lepo videti, vendar ko izberemo nekaj na namizju, ni to nič posebnega. Ljudje uživamo, ko vemo, da uporabljamo nekaj, kar je bilo narejeno za nas in lahko tudi sami vplivamo na podobo okolja v katerem delamo, se zabavamo ali kaj podobnega. Na osnovi glavne ideje, da človek z večjim užitkom upravlja z nečem, kar je prilagojeno njemu, je nastala ideja za najin program, ki sva ga poimenovala, »Command Centre.« Veliko ljudi je vizualnih, zato je grafični vmesnik za večino uporabnikov odločilni faktor, ali jim je neka aplikacija všeč, ali ne, zato smo se odločili, da je uporabnikom na voljo toliko različnih možnosti, da dodajo našem programu občutek lastne osebnosti.

Problem se pojavi, če onemogočimo vse aplikacije, ki upočasnjujejo zagon računalnika pri ponovnem izboru teh aplikacij, saj če jih imamo veliko, lahko nastane velika zmeda in naenkrat ne vemo, katere je potrebno najprej zagnati. Zato je pomembna tudi možnost, da zaganjamo programe v zaporedju, ki je za to najboljše in čim hitrejše.

Eden od možnih problemov nastane, ko smo ljudje brezskrbni in ne vemo kam smo katero aplikacijo shranili in je potem ne najdemo, ko čistimo namizje, zbrišemo bližnjico in aplikacija tone v pozabo. Tudi ta problem bi odpravil naš program, če je njegov uporabnik dosleden do te mere, da vse programe, ki jih namesti v sistem, namesti tudi v aplikacijo. Tako so mu vedno znane poti do teh aplikacij, če jih pozabi, namizje ostane čisto in aplikacijo pa lahko požene kar iz zbirke ostalih aplikacij in programov, ki jih redno uporablja v svojem vsakdanu.

1.1. Hipoteza naloge

Hipoteze zastavljene so naslednje:

- Dostop do aplikacij je bolj oseben
- Program pripomore k zagonu
- »Nadzorni vmesnik je lahkotno integriran kamorkoli«

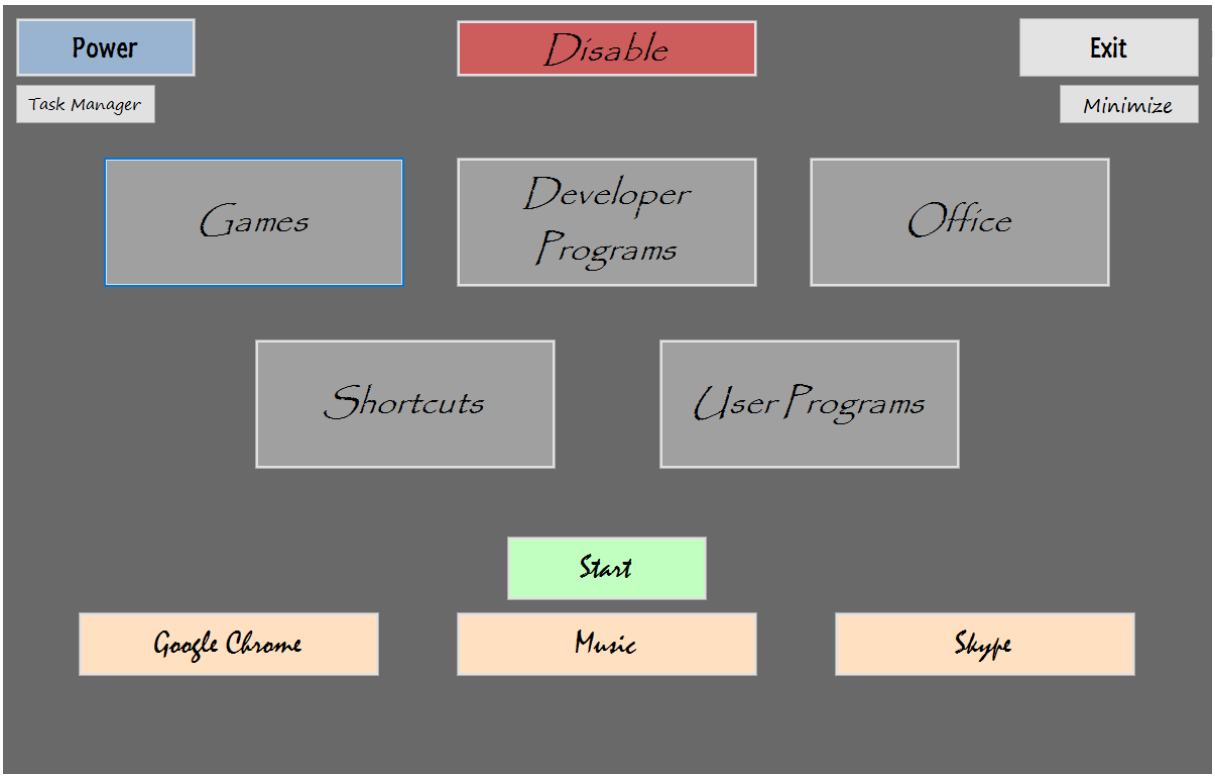
1.2. Opis raziskave

Izdelala sva anketo ki so jo reševali naključni izbranci, ki so spadali v različne skupine ljudi, starostne, družbene in uporabnikov različnih operacijskih sistemov, saj sva le tako lahko uspešno potrdila da bi bilo to aplikacijo sploh vredno vpeljati za vsakdanje uporabnike. Najbolj smiselna se je aplikacija zdela starejšim ljudem, ki imajo ponavadi težavo pri iskanju po sistemu, ki je lahko na trenutke zapleten. Mlade je pritegnila ideja o hitrejšem zagonu računalnika, saj je večina z željo po aplikaciji tudi obkrožila, da ima več kot en računalnik, sklepamo da je ta manj zmogljiv. Ideja, da bi bila aplikacija primerna za v delovno okolje se je izjalovila, saj nihče od anketiranih ni izrazilo pozitivnega odnosa za možnost vpeljave na delovno postajo. Anketirancem je bilo po veliki večini všeč to, da bi lahko sami upravljajo in urejajo izgled in odzivanje aplikacije, glede na njihove preference.

2. Uporaba programa

2.1. Namen programa

Program je namenjen osebni uporabi in daje uporabnikom občutek kot da je pripravljen posebej za njih in da je prilagojen njihovim potrebam. Na gumbih piše točno to kar si želijo in jih vodijo tam, kamor je nastavljeno oz. želijo. Na sliki je primer ene možne izvedbe.

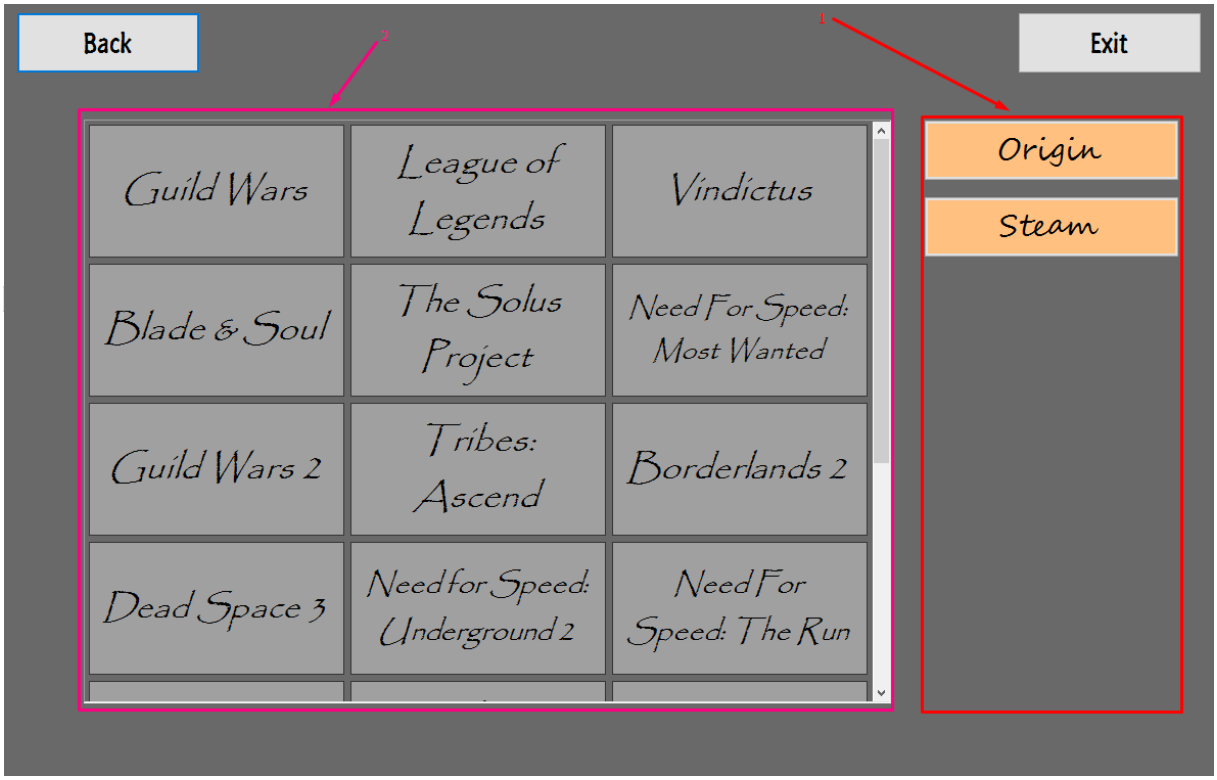


Slika 1: Primer prilagoditve programa

Seveda js stil, ki ga želi uporabnik, popolnoma neodvisen od tega ki ga zapišemo v programu, saj lahko uporabnik spreminja veliko večino kontrol. Ena od glavnih atrakcij programa je gotovo možnost simultanega pogona aplikacij, ki nadomestijo »startup« v Windows okolju, da je zagon računalnika nekoliko hitrejši.

2.2. Zmogljivost programa

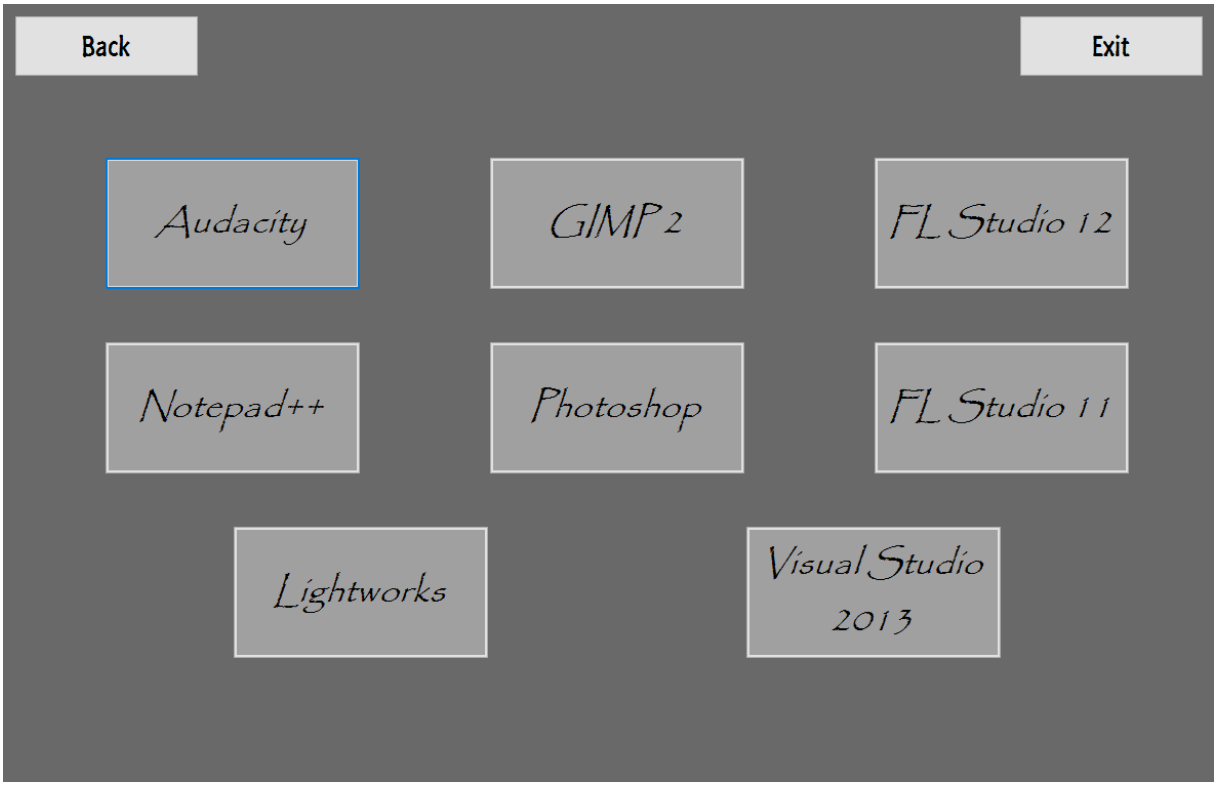
Program nudi med drugimi možnostmi tudi to, da lahko dodamo novo vrstico za posebne aplikacije, ki jih ponavadi uporabljamo vzporedno z izbrano zbirko, označeno na sliki »1«, ostali gumbi pa so tesno povezani skupaj, kot jih je razvrstil uporabnik. »2«.



Slika 2: Primer razvrščanja programov v skupine

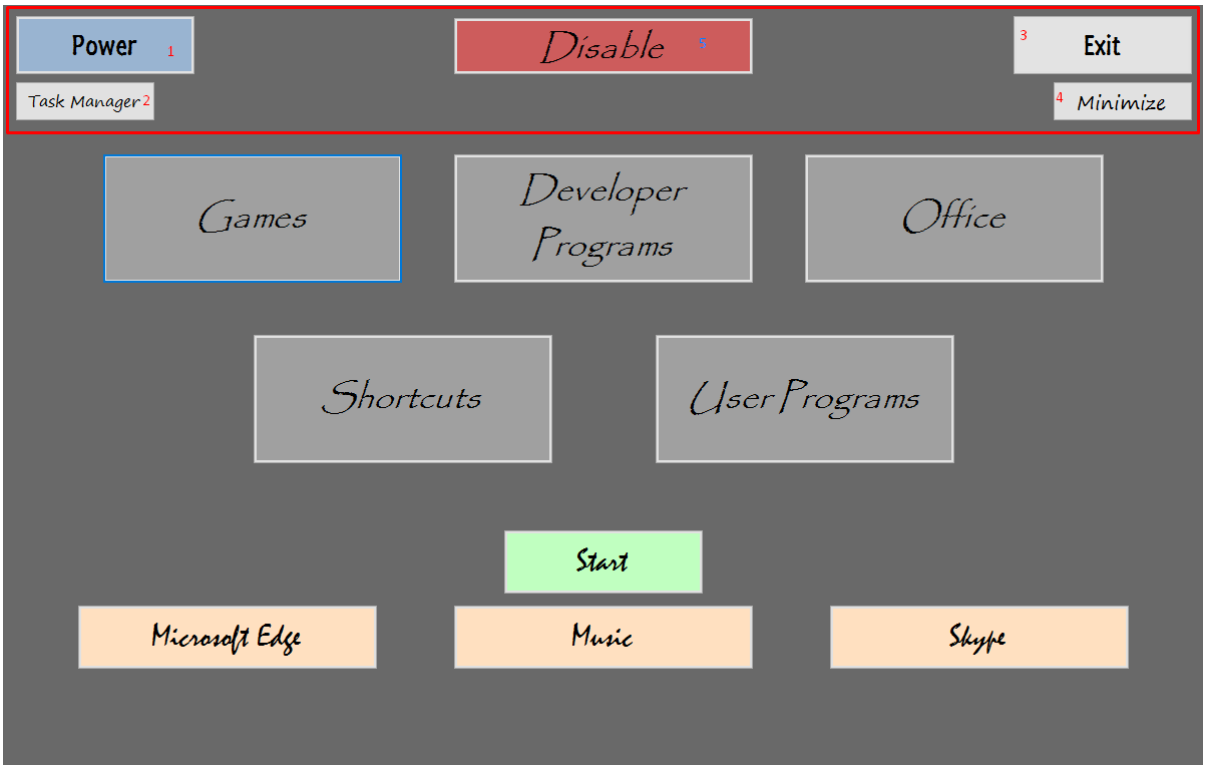
Imamo tudi možnost, da gumbe razvrstimo malo drugače, da so bolj razpoznavni, vendar to ni vedno priporočljivo, saj v primeru prevelikega števila aplikacij v eni zbirki, bi to otežilo slednji namen.

Primer dobre uporabe takšne razvrstitve je na sledeči sliki:



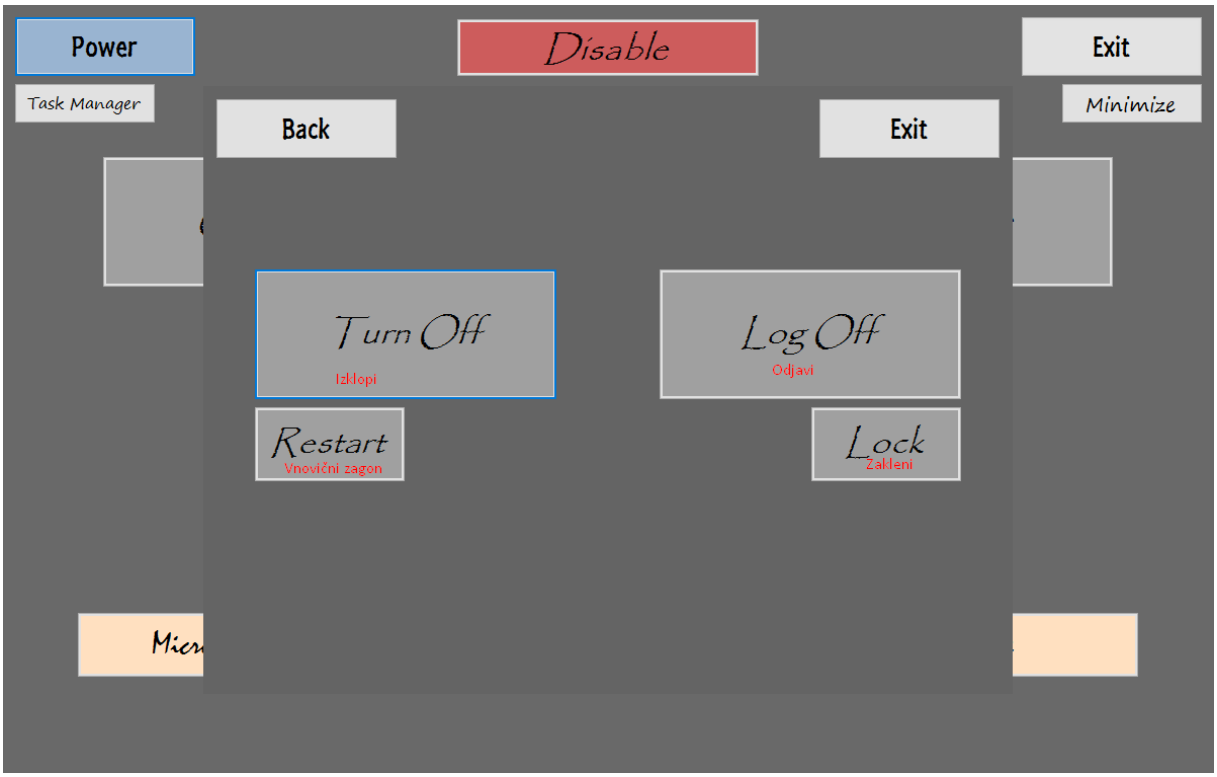
Slika 3: Primer razvrstitve

Naš program omogoča tudi naslednjo možnost, da si uporabnik sam nastavi na glavnem meniju zgornjo vrstico z elementi, ki jih izbere z našega nabora ali pa sam doda gumbe po želji, ki vodijo kamor sam želi. Uporabnikom smo dali možnost izbora med »Power« gumbom (1), ki odpre novo okno, ki da uporabnuku na voljo več funkcij, lahko si jih ogledate malo nižje, kjer jih bova podrobneje opisala. Na voljo je tudi »upravitelj opravil« (2), ta zažene windowsovo aplikacijo s to funkcijo. Na voljo sta še samoumevna gumba »Exit« (3) in pa »Minimize« (4).



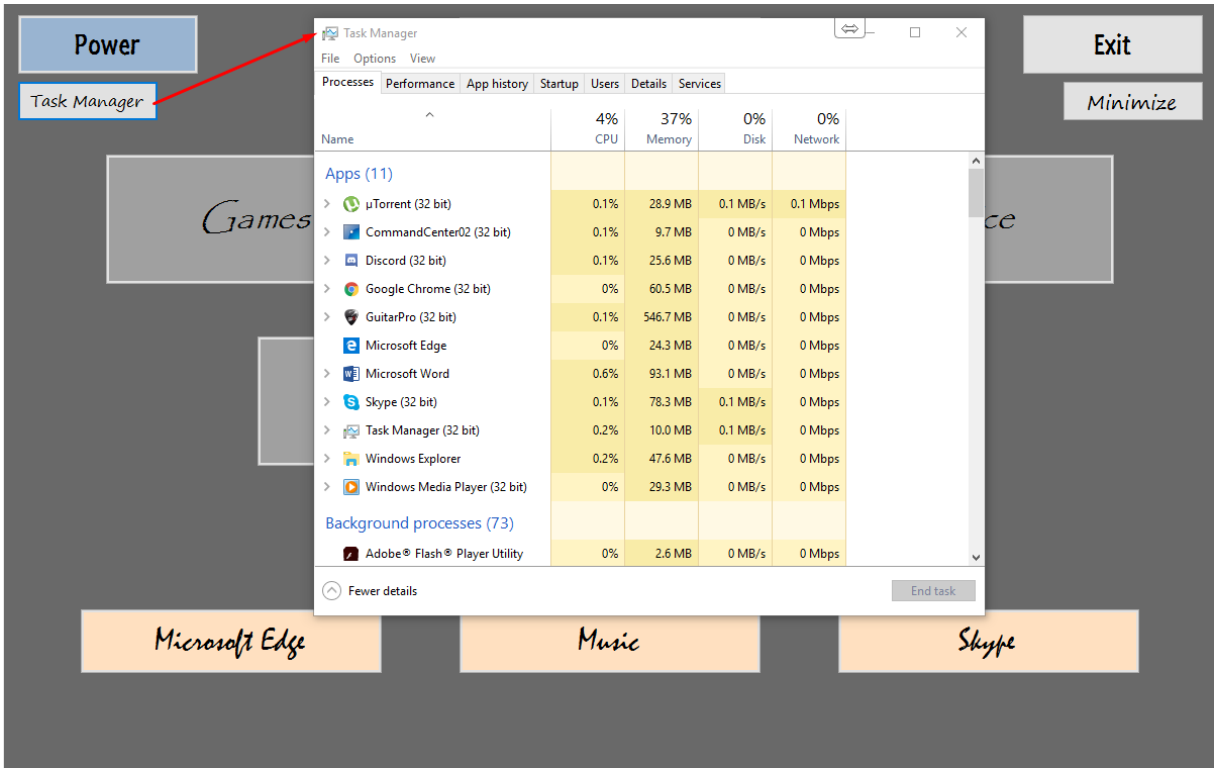
Slika 4: Funkcijski gumbi na vrhu

Vpeljujemo tudi dodatno funkcijo, ki bi odprla posebno okno narejeno kot windowsovo okno za vpis, trenutno je izvedena kot gumb »Disable« (5), vendar upamo, da ga bomo uspeli izvesti kot ohranjevalnik zaslona, da se bo sam pojavil v času mirovanja računalnika, ko ga bo nastavljal uporabnik. Še navedene slike za »Power« (1)

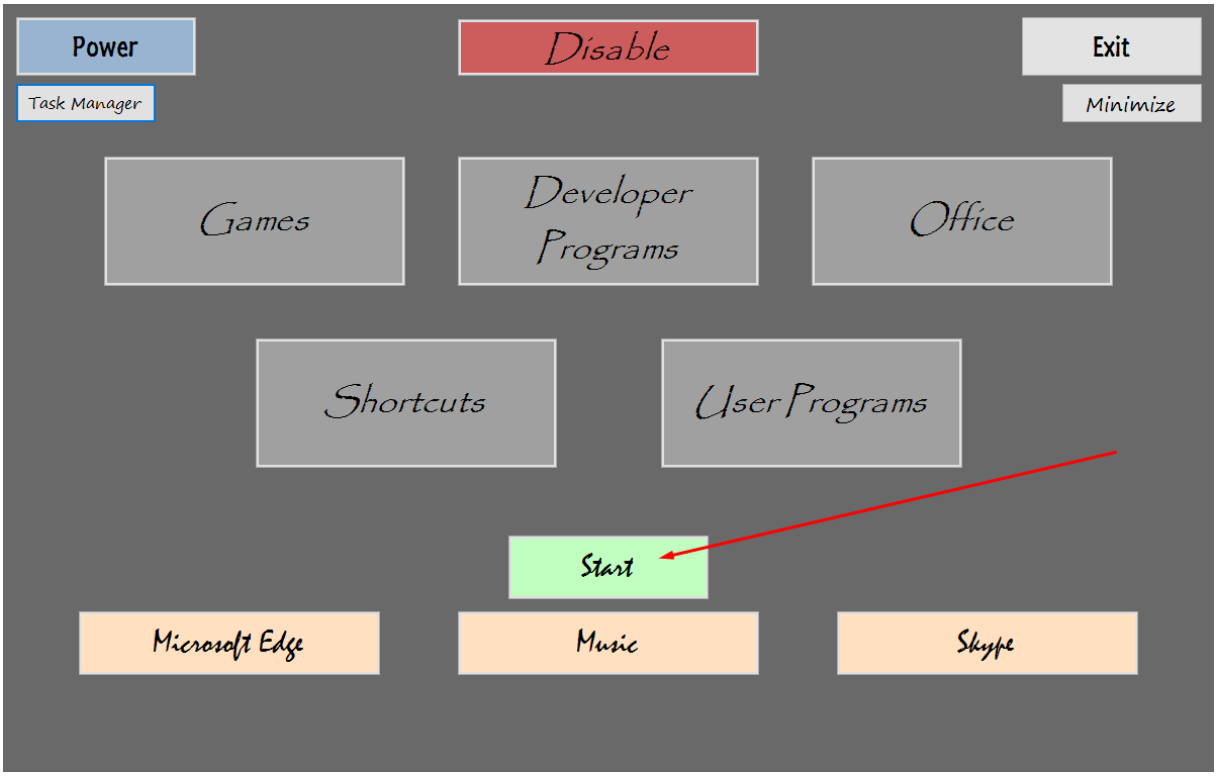


Slika 5: Gumb Power

In pa še prikaz delovanja gumba »TaskManager« (2)



Slika 6: Gumb Task manager



Slika 7: Gumb Start

Označeni »Start« gumb zažene vse programe vrstice priljubljenih hkrati.

3. O razvojnem okolju

Microsoft Visual Studio (VS) je integrirano razvojno okolje (IDE), ki ga je razvilo podjetje Microsoft. Njegov namen je razvoj programske opreme in aplikacij za operacijski sistem Windows, spletnih strani, spletnih aplikacij ter aplikacij ogrodja .NET.

Visual Studio uporablja razne razvojne platforme, na primer: Windows Forms, Console App, ASP.NET WEB App, Azure Cloud Service in mnoge druge. Omogoča produciranje strojne kode, kot pa tudi za uporabo višjih programskih jezikov.

Čeprav je okolje na voljo le na operacijskem sistemu Windows, se pogosto uporablja za razvoj programske opreme prenosljive med različnimi platformami. S integretacijo zunanjih urejevalnikov in drugih orodij ter vtičnikov lahko razvijamo programsko opremo tudi za druge platforme kot so na primer mobilne aplikacije za Android.

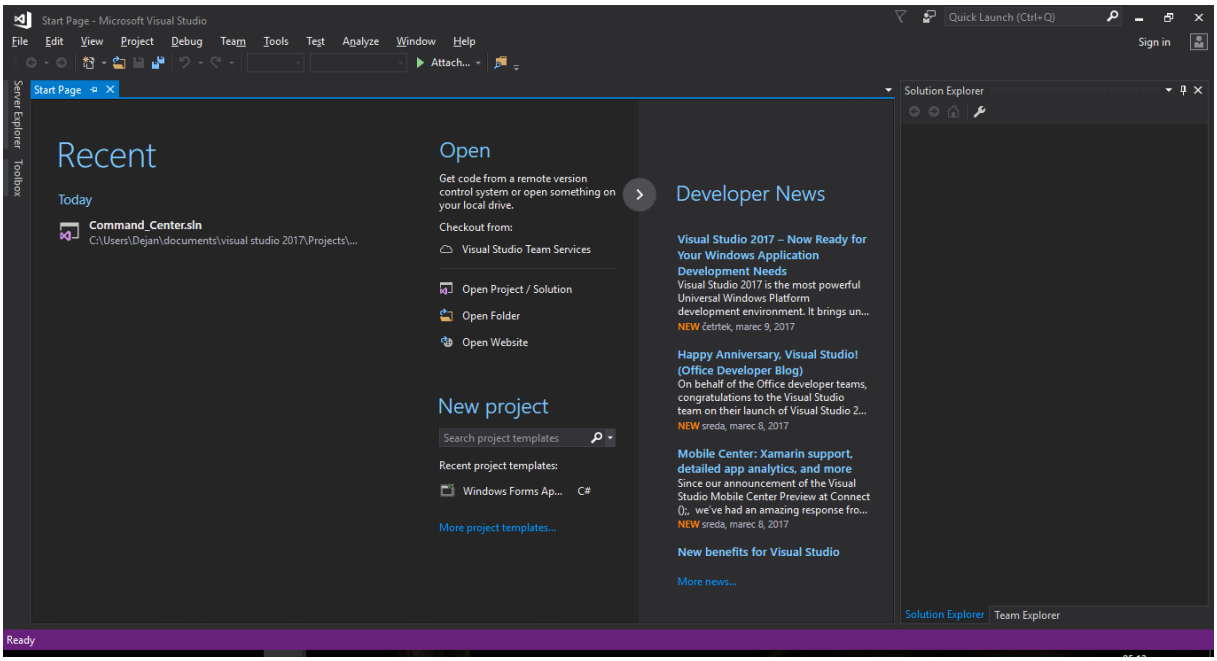
Urejevalnik izvorne kode podpira IntelliSense, komponento za avtomatično dopolnjevanje kode z razširjeno funkcionalnostjo.

Visual Studio je v okrnjeni različici Visual Studio Express je dostopen kot brezplačen program, ki pa ne podpira razširljivosti preko vtičnikov s strani neodvisnih razvijalcev. Različica Community Edition, ki podpira vtičnike, je brezplačna za osebno uporabo, akademsko uprabo in za uporabo v manjših skupinah. Visual Studio Professional in Visual Studio Enterprise sta namenjena večjim skupinam ali podjetjem in nudita večji izbor orodij za razvijalce.

3.1. Uporaba razvojnega okolja

Razvojno okolje Microsoft Visual Studio je na voljo na uradni spletni strani proizvajalca, https://www.visualstudio.com.

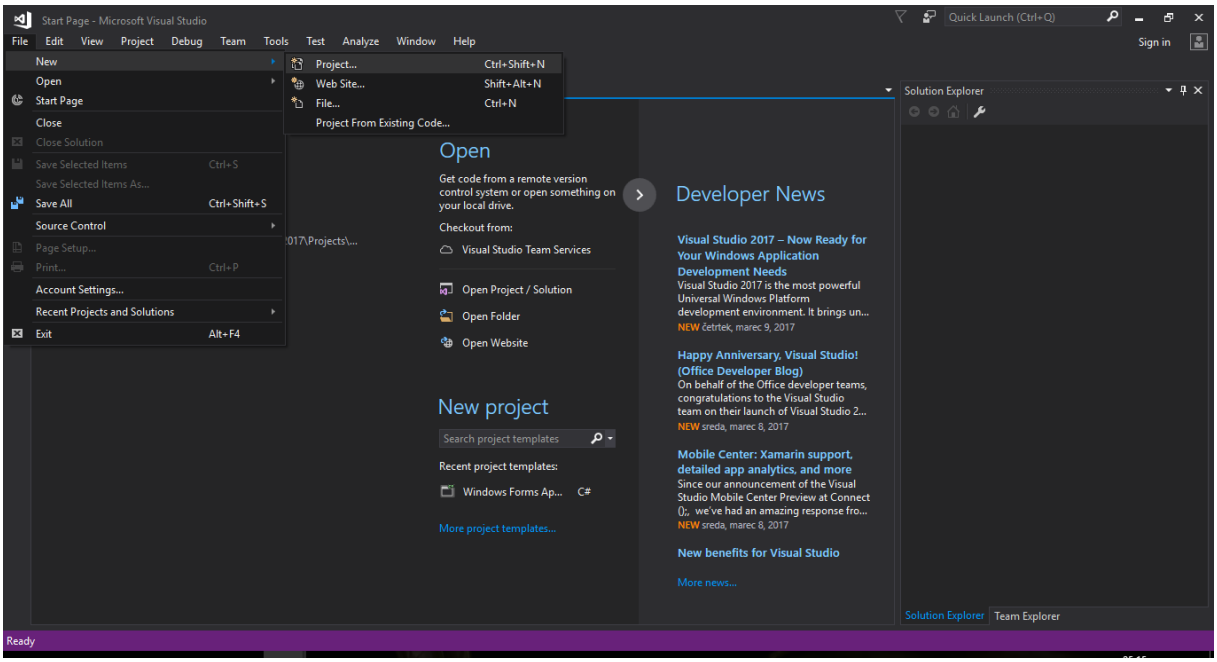
Ob zagonu se prikaže začetna stran Start Page na kateri so prikazani nedavni projekti, možnost odpiranja že obstoječih projektov ali začetek novega.



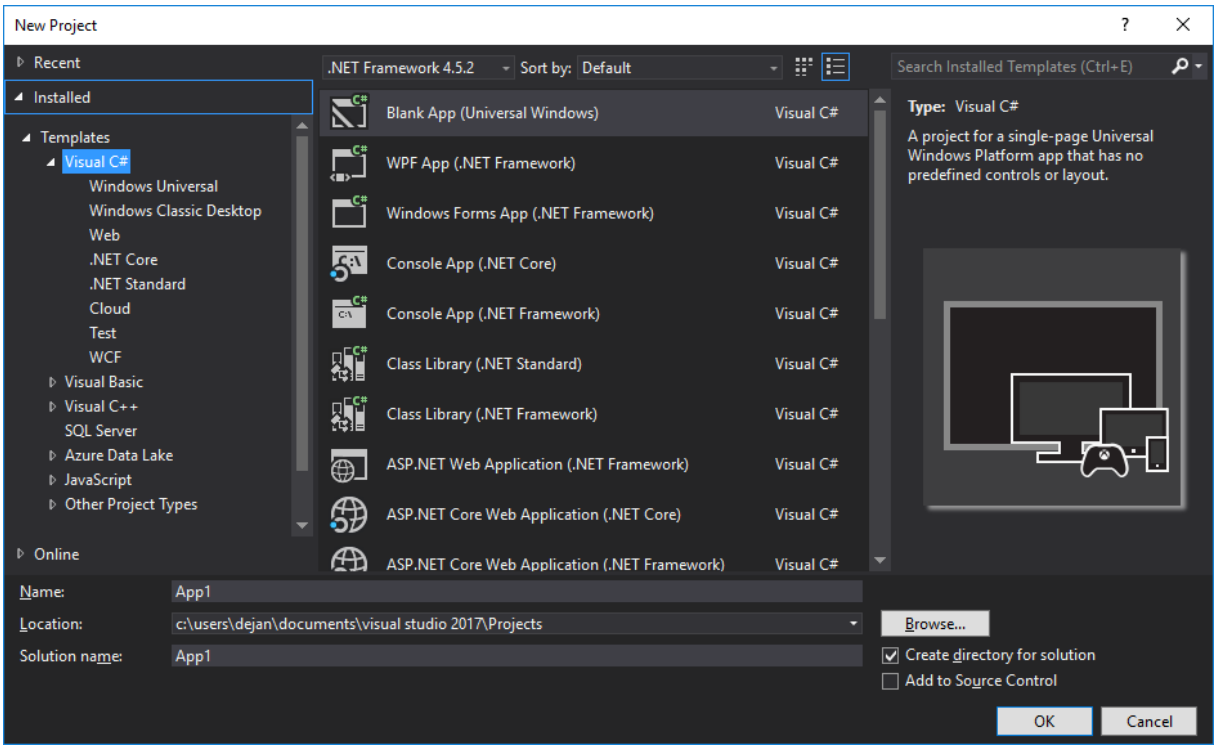
Slika 8: Začetni zaslon Microsoft Visual Studio

5.2.1. Začetek novega projekta

Nov projekt preprosto ustvarimo tako, da gremo v File->New->Project ali z bližnjico gumbov Ctrl+Shift+N.

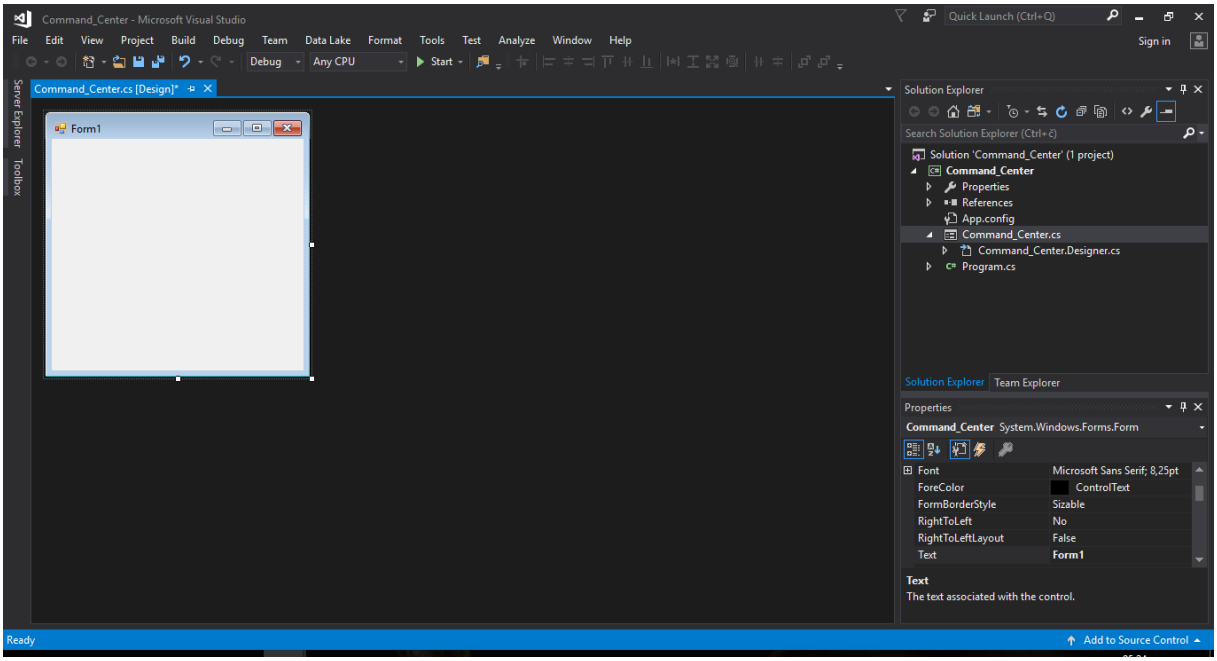


Odpre se nam okno New Project, kjer izberemo programski jezik in predlogo(template) ki jo želimo uporabiti.



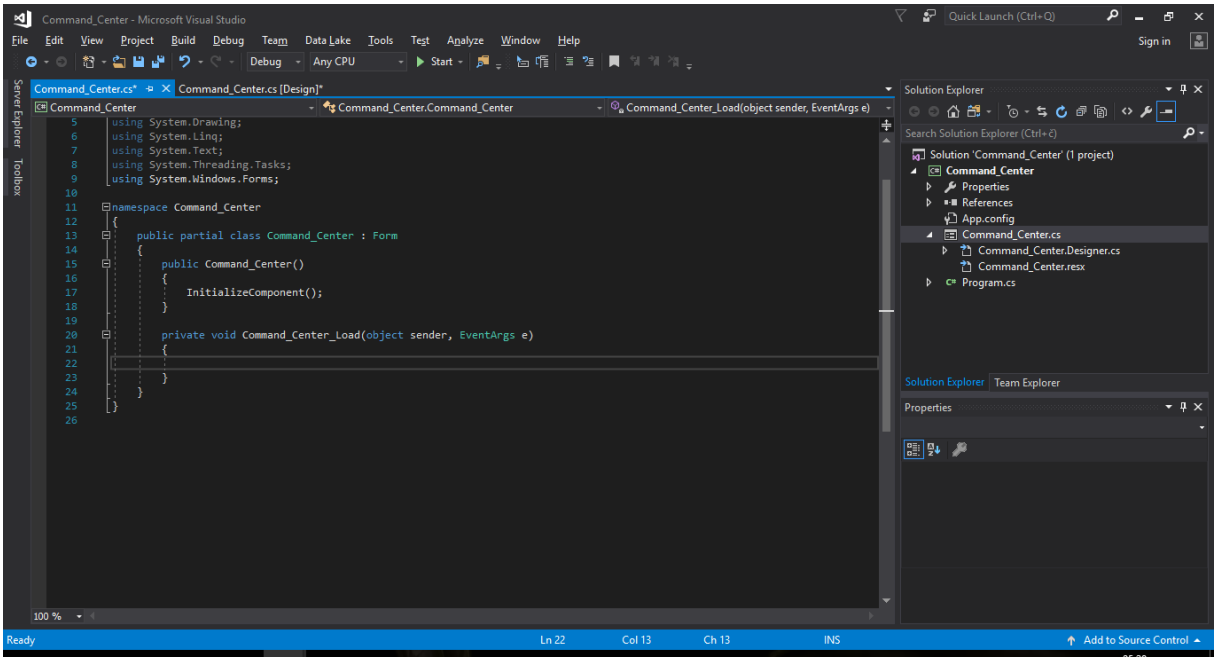
Slika 9: Nov projekt

Ko izberemo programski jezik, z katerim želimo pisati kodo, nam na desni prikaže vse predloge, ki so nam na voljo. Program poimenujemo in pritisnemo gumb OK, ki zažene projekt. V tem primeru sem izbral Windows Forms Apps z programskim jezikom C#, s katerim je napisan tudi projekt te raziskovalne naloge, Nadzorni vmesnik za operacijski sistem Windows.



Slika 10: Nov Windows Forms projekt

S dvojnim klikom na okno Form1 se odpre del kode, kjer lahko začnemo programirati.



Slika 11: Kodni del novega projekta

6. O programskem jeziku C#

C# (C sharp) je objektno orientiran programski jezik, ki ga je razvil Microsoft v okviru razvoja ogrodja .NET. Sintaksa jezika se zgleduje po drugih programskih jezikih: svojima predhodnikoma C in C++, najdejo se pa tudi podobnosti z programskim jezikom Javo. Prva različica se je pojavila v začetku leta 2002. Nekateri tudi menijo, da je C# Microsoftov odgovor na Javo. Jezik sta kot standard odobrli organizaciji ECMA (ECMA-334) in ISO (ISO/IEC 23270:2006).

C# je programski jezik, ki cilja na izvajalsko okolje CLI(Common language infrastructure) in s tem ogrodje .NET ter njegove različice kot je Mono. Programi v C# se izvajajo v posebnem izvajalskem okolju v katerem se ta programska koda šteje za upravljalno. Dele programov napisane v C# je zato enostavno povezovati z deli, ki so napisani v kateremkoli drugem programskem jeziku skladnem s CLI, kar daje programskemu jeziku dobro prenosljivost med različnimi platformami. Zaradi točne specifikacije programskega jezika in vmesnega jezika CIL(Common intermediate language), se lahko program napisan v C# neposredno prevede in izvede na drugih okoljih, ne da bi bilo treba za prenosljivost program spreminjati. Te značilnosti, skupaj z jasnimi in dodelanimi skladnjami je C# eden najbolj priljubljenih programskih jezikov.

7. RAZVOJ APLIKACIJE

7.2. Začetek ideje

Razvoj aplikacije se je začel že dolgo pred raziskovalno nalogo, ker sem sam želel nekaj da počisti moj »startup« in sprazni namizje, saj sem človek ki zna ceniti lepe slike in fotografije. Ideja da bi tak program potrebovali tudi drugi se je prvič pojavila ko so nekateri ki so videli program spraševali kje bi se ga dalo dobiti, pa takrat še ni bilo nobene izvedbe za kaj podobnega na internetu, takrat se je porodila zamisel o programu ki je hkrati praktičen in estetičen.

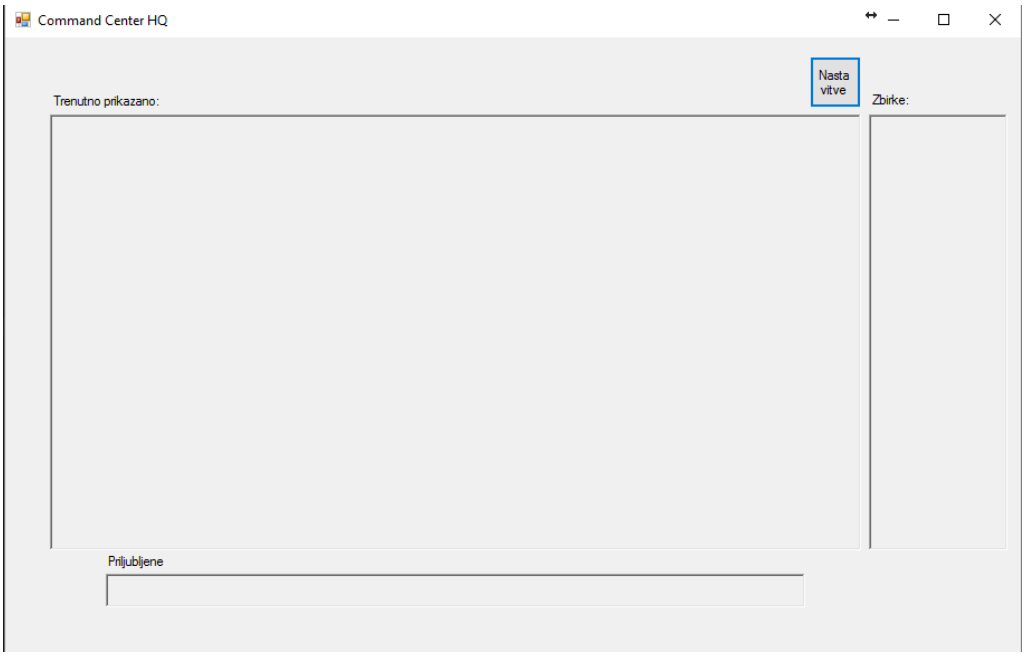
7.3. Prvi koraki

Za začetek je največji problem bil dinamično dodajanje gumbov, odpiranje in zapiranje form oken v Visual Studiu 2013, to je izvedeno nekako tako, saj smo uporabili za bolj dinamično posodabljanje programa funkcijo v razvojnem okolju »User Control«. To nam tudi omogoča da spreminjamo samo prikaz enega okna in zadeva steče bolj tekoče.

```
1 reference
private void bSettings_Click(object sender, EventArgs e)
{
    Settings settings = new Settings();
    if (flpMain.Controls.Count != 1)
    {
        settings.Height = flpMain.Height;
        settings.Width = flpMain.Width;
        flpMain.Controls.Add(settings);
    }
    else flpMain.Controls.RemoveAt(0);
}
```

Slika 12: Začetek pisanja kode

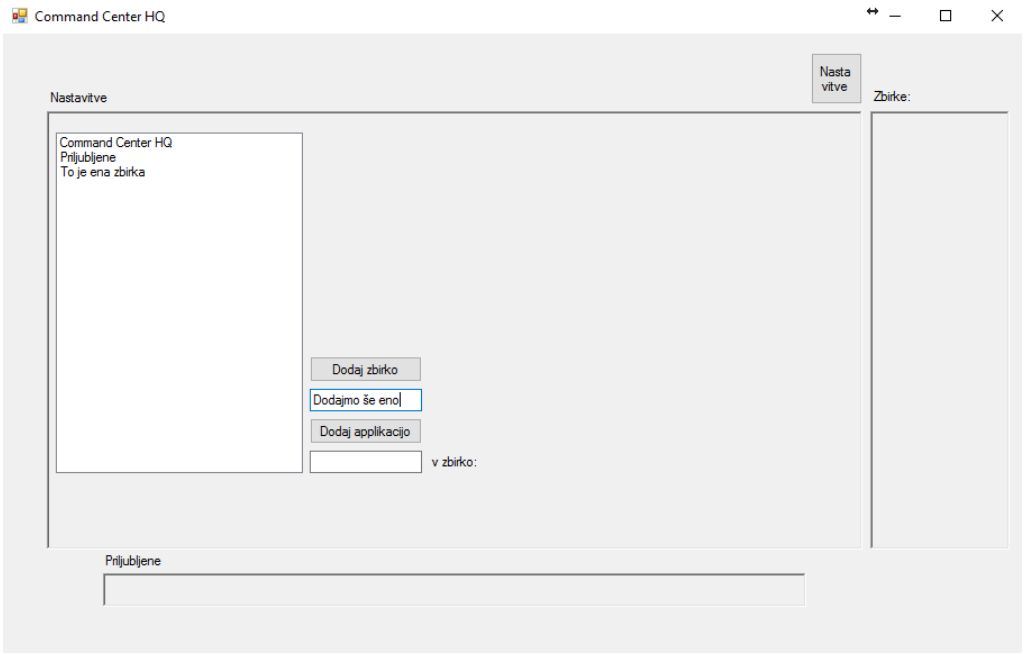
Na naslednji sliki si lahko ogledamo kaj to za naš program sploh pomeni, kaj se zgodi ob pritisku »Nastavitve« in zakaj je to dobro. Naslednja sliak prikazuje stanje programa pred pritiskom gumba.



Slika 13: pred pritiskom gumba Nastavitve

Pa poglejmo še kaj se zgodi ko se gumb pritisne in katere spremembe so možne.

Vidimo spremembe v napisu nad oknom prikaza in tudi to, da se kontrole ki prikazujejo »zbirke« in »priljubljene« še vedno vidne, to je ena od prednosti zaradi katerih sma se odločila za »User Control« namesto aplikacije z večmi okni.



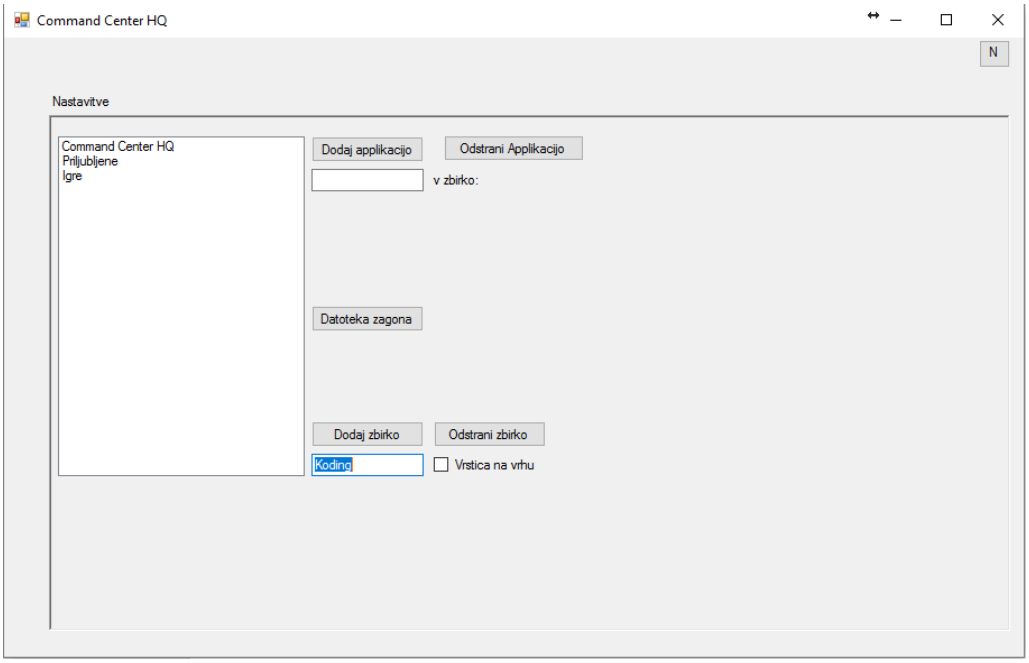
Slika 14: Možnosti gumba Nastavitve

7.4. Zanimivi problemi

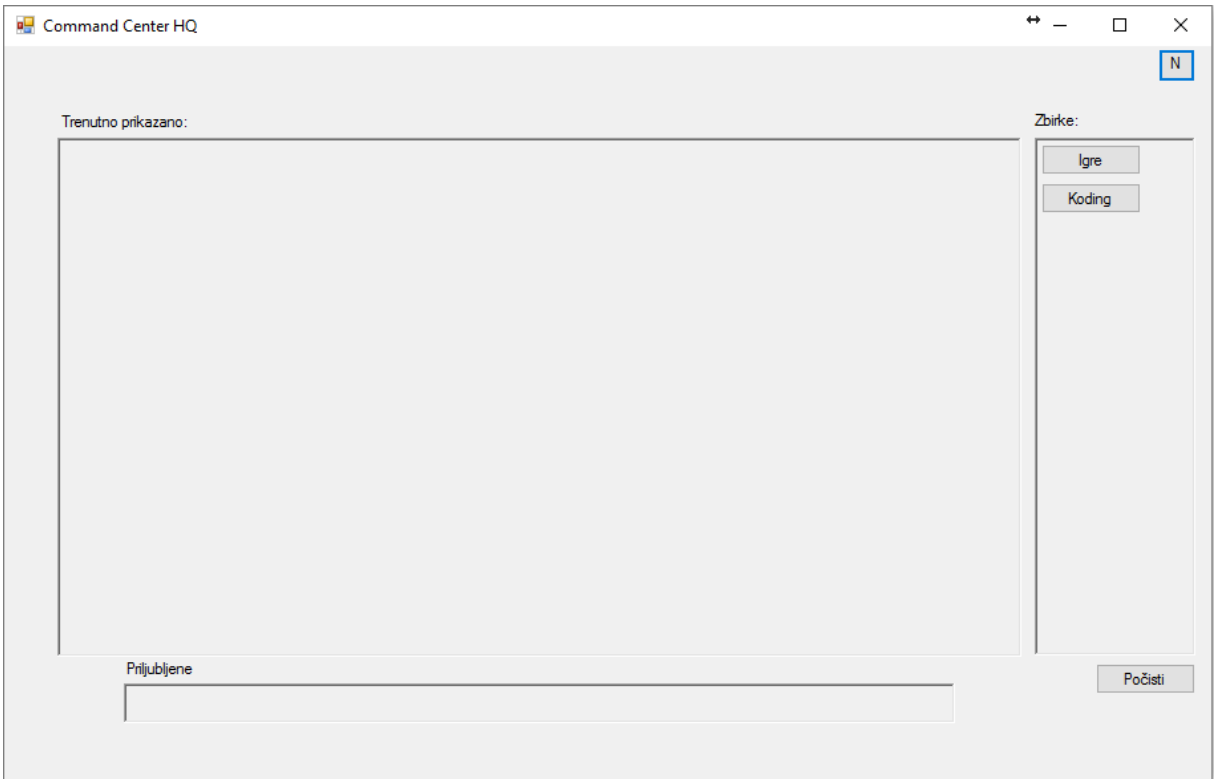
Prvi problem s katerim sva se srečala so bili dinamične kontrole na formi aplikacije, ter kako priklicati nova okna in jih postaviti nanje. Rešila sva ga tako, da sva uporabila znanje iz objektov in ustvarila nov gumb, ter mu pripisala lastnosti takšne kot jih je želel uporabnik. Gumbe sma zapisala v list objektov ter jih tako priklicala v obstoj natem oknu kot gumb in zbirko.

```
for (int i = 0; i < SeznamZbirk.getAll().Count; i++)
{
    if (SeznamZbirk.GetItem(i).Naziv == naziv)
    {
        zbirka = SeznamZbirk.GetItem(i);

    }
}

if(zbirka != null)
{
    if (zbirka.NaVrhu == true)
    {
        Label header = new Label();
        header.Text = zbirka.Naziv;
        flowLayoutPanel2.Controls.Add(header);
    }
    else
    {
        flowLayoutPanel2.Visible = false;
    }

    for(int i = 0; i < zbirka.Aplikacije.Count; i++)
    {
        Button b = new Button();
        b.Text = zbirka.Aplikacije[i].Naziv;

        flowLayoutPanel1.Controls.Add(b);
    }
```

Slika 15 Dinamično dodajanje gumbov



Slika 16 Vpis zbirke



Slika 17 Zbirka se pojavi kot nov gumb

Naslednji problem, ki je povzročal preglavice je bil sledeči, dodajanje novih oken glede na zbirko. Na ta problem sva odgovorila z uporabo C#-ovih »User Control«, ki se pojavijo v okvirju »trenutno prikazano:«

```
2 references
private void C_Click(object sender, EventArgs e)
{
    //System.Diagnostics.Process.Start(Aplikacija.pot);
    string s = (sender as Button).Text;
    userCustomized = new UserCustomized(s);

    if (flpMain.Controls.Count == 0)
    {
        settings.Height = flpMain.Height;
        settings.Width = flpMain.Width;
        flpMain.Controls.Add(userCustomized);
    }
    else
    {
        flpMain.Controls.RemoveAt(0);
        settings.Height = flpMain.Height;
        settings.Width = flpMain.Width;
        flpMain.Controls.Add(userCustomized);
    }
}
```

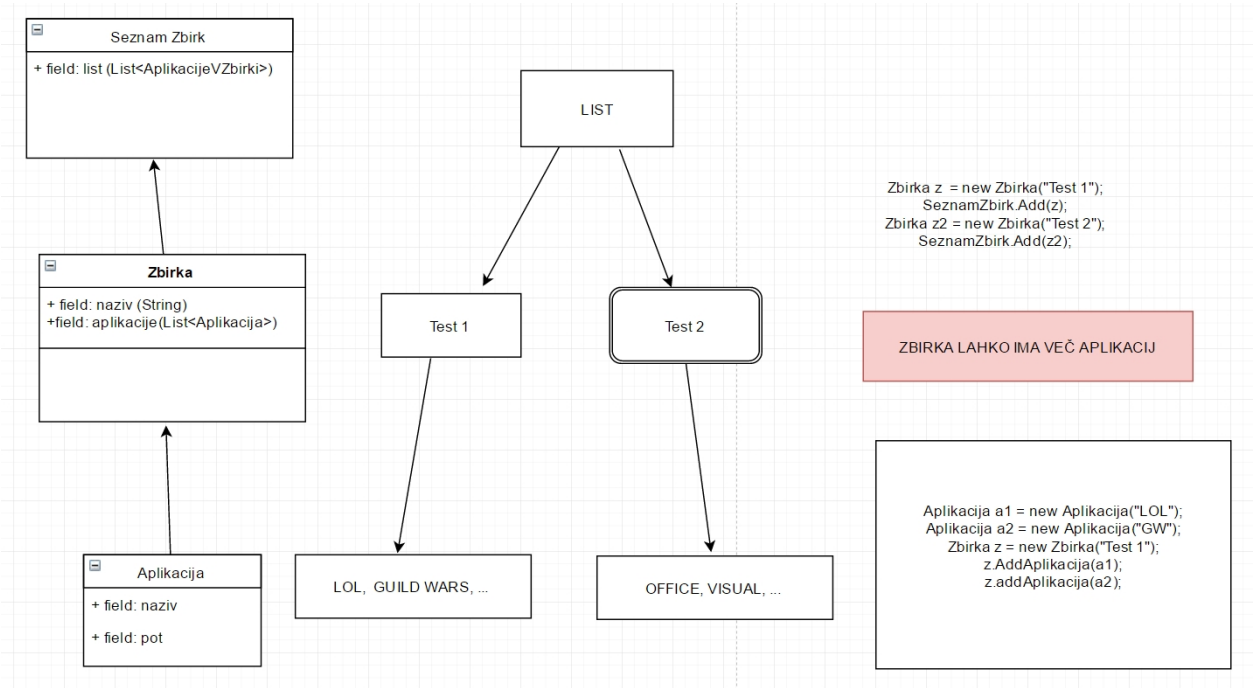
Slika 18 Izvedba User Control

Eden glavnih problemo pri prenašanju zbirk v gumbe je bilo podajanje objektov med okni, to sva rešila z implementacijo novih razredov, na sliki je prva rešitev izvedena, ta je list objektov, ki shranjuje gumbe oz. zbirke kot njegov atribut.

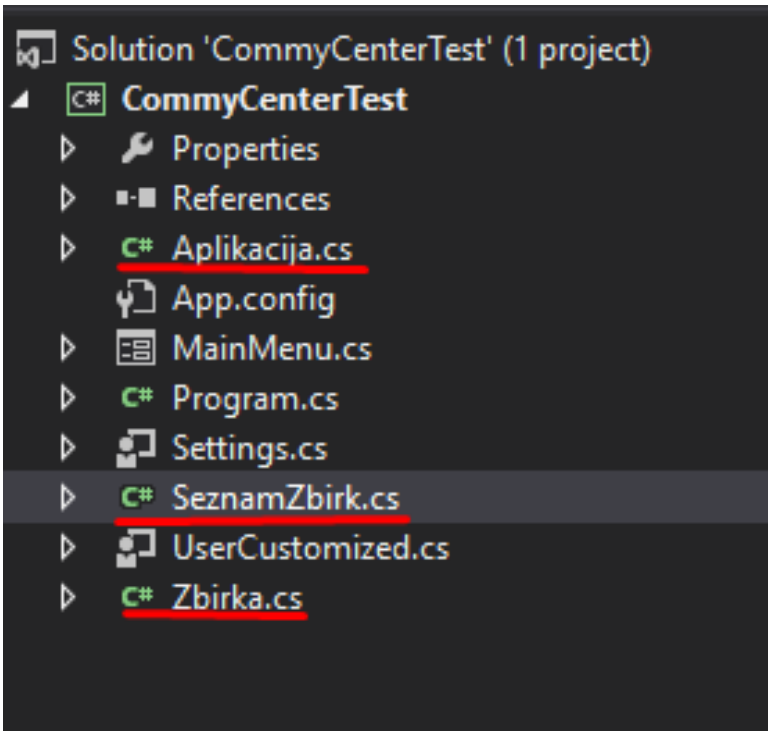
```
20 references
public class SeznamZbirk
{
    private static List<Zbirka> list = new List<Zbirka>();

    1 reference
    public static void Add(Zbirka z)
    {
        list.Add(z);
    }

    1 reference
    public static void RemoveAt(int index)
    {
        list.RemoveAt(index);
    }

    7 references
    public static Zbirka GetItem(int index)
    {
        return list.ElementAt(index);
    }

    7 references
    public static List<Zbirka> getAll()
    {
        return list;
    }
```

Slika 19 List zbirk

Za popolno odpravo problemov, je bila potrebna hierarhija razredov ki je prikazana na sliki 20, končni produkt tega načrtovanja pa je viden na sliki 21.



Slika 20 Diagram razredov



Slika 21 Seznam razredov

Razred »Seznam zbirk« ima shranjene vse zbirke in njihove nazive, da jih lahko potem prenašamo med okni lažje in bolj elegantno, saj so vsi zbrani na enem mestu, brez diskriminacije glede na njihovo ime. Razred »Zbirka« ima shranjene aplikacije in njihove nazive zaradi podobnih razlogov kot »SeznamZbirk«, aplikacije so shranjene kot list. Razred »Aplikacija« ima podatke o gumbih, ki so z objektom tega razreda povezani, njihv naziv in pa seveda pot s katere se naj izvajajo aplikacije.

Zaden večji problem ki sva ga imela je bila serializacija celotne aplikacije, saj se ničesar podobnega nismo učili, njen proces pa je kratek in zelo varljiv. Izvedla sva serializacijo atributa razreda »SeznamZbirk«, ki je imel v sebi shranjene vse podatke o zbirkah, ti pa o aplikacijah. Deserializacijo tega atributa sva si pripravila kot metodo v omenjenem razredu.

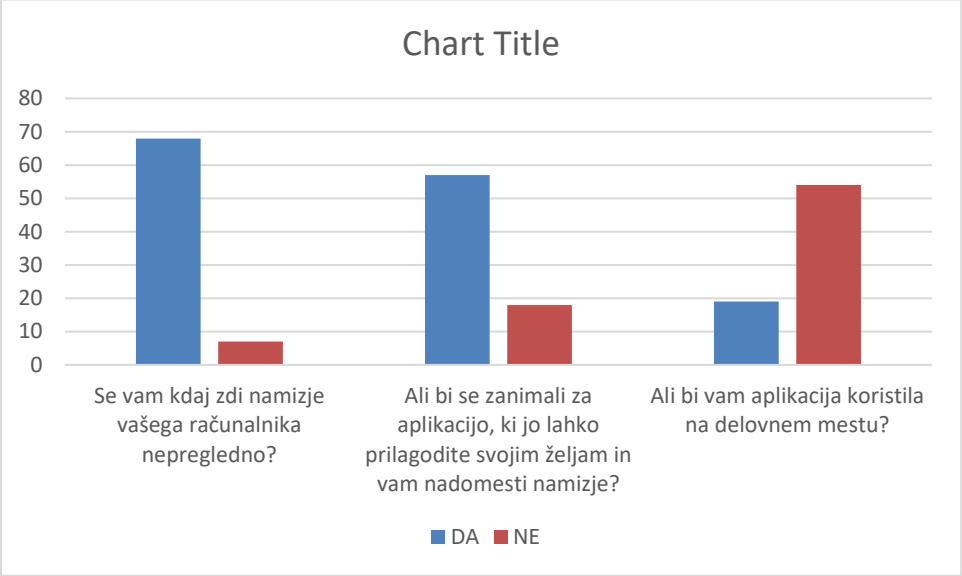
```
[Serializable()]
20 references
public class SeznamZbirk
{
    private static List<Zbirka> list = new List<Zbirka>();

    1 reference
    public static void Add(Zbirka z)
    {
        list.Add(z);
    }

    1 reference
    public static void RemoveAt(int index)
    {
        list.RemoveAt(index);
    }

    7 references
    public static Zbirka GetItem(int index)
    {
        return list.ElementAt(index);
    }

    7 references
    public static List<Zbirka> getAll()
    {
        return list;
    }

    // Serializacija - https://www.dotnetperls.com/serialize-list
    1 reference
    public static void Save()
    {
        try
        {
            using (Stream stream = File.Open("data.bin", FileMode.Create))
            {
                BinaryFormatter bin = new BinaryFormatter();
                bin.Serialize(stream, list);
            }
        }
        catch (IOException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    1 reference
    public static bool Open()
    {
        try
        {
            using (Stream stream = File.Open("data.bin", FileMode.Open))
            {
                BinaryFormatter bin = new BinaryFormatter();
                list = (List<Zbirka>)bin.Deserialize(stream);
            }
        }
        catch (IOException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }

        return true;
    }
```

Slika 22 Serializacija

8. Sklep

8.1. Rezultati ankete

Anketirala sva 75 ki so spadali v različne skupine ljudi, starostne, družbene in uporabnikov različnih operacijskih sistemov.



8.2 Teze ki sva si jih zastavila so bile sledče:

- Dostop do aplikacij je bolj oseben
- Program pripomore k zagonu
- »Nadzorni vmesnik je lahkotno integriran kamorkoli«

Prvi dve tezi sva potrdila, saj so bili uporabniki navdušeni nad programom in nadzorom ki jim ga je omogočil, da sestavijo svoj vmesnik, ki je hkrati pregleden in dober za zagonski čas sistema. Zadnjo sva morala ovreči, saj nisva premislila ob zastavitvi, da boma program razvijala samo za sistem Windows in s tem izpustila uporabnike ostalih sistemov.

8.3. Sklepna beseda

Skozi razvoj raziskovalne naloge in aplikacije sva naletela na več problemov, programskih kot človeških, saj sva morala poiskati rešitve ki so bile za naju izvedljive a ljudem še vedno vabljive. Problem je prišel na dan tudi zaradi nepoznavanja najinega področja, saj sva oba dinamične kontrole vpeljevala v Windows okensko aplikacijo prvič, sploh ko je imel uporabnik toliko modeliranja okolja na izbiro. Največji problem je predstavljala serializacija, ker je bila za naju popolnoma nov pristop do shranjevanja podatkov. Ugotovila sva da najina aplikacija ni primerna za uporabo v podjetju, je pa zaželjena med preprostimi uporabniki v domačih okoljih.

Viri

https://sl.wikipedia.org/wiki/Programski_jezik_C_sharp#Razli.C4.8Dice

https://msdn.microsoft.com/en-us/library/kx37x362.aspx

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

https://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx

http://stackoverflow.com/

Uranič, Srečo: Visual C# .NET

Gregor Jerše, Matija Lokar, Srečo Uranič: Programski jezik C# Delo z datotekami