

# Srednja poklicna in tehniška šola Murska Sobota

Šolsko naselje 12  
9000 Murska Sobota

## PREDVAJALNIK GLASBE V C#

Izdelek za četrti predmet poklicne mature

Dijak: Aleksander Kovač  
Mentor: mag. Igor Kutoš

## *Povzetek*

V tem je dokumentu je predstavljen izdelek za 4. predmet poklicne mature. Opraviti, ga moram če želim, zaključiti srednje strokovno izobraževanje po programu Tehnik računalništva.



## Kazalo vsebine

Povzetek.....	1
Predstavitev problema.....	3
Predvajalnik glasbe v jeziku C#.....	4
Načrtovanje.....	4
Analiza problema .....	4
Ideja.....	5
Diagram poteka .....	6
Orodja za izdelavo naloge .....	8
Testiranje.....	21
Navodila.....	22
Začetek.....	22
Predvajanje datotek.....	22
O programu.....	23
Izhod iz programa .....	23
Literatura.....	24
Bibliografija.....	24
Svetovni splet .....	24
Kazalo slik in kazalo tabel.....	25
Kazalo slik.....	25
Kazalo tabel.....	26
Priloga: programska koda.....	26
Form1.cs (Glavni razred).....	26
Metapodatki.cs (Razred za obdelavo metapodatkov) .....	34
SQLite.cs (Razred za delo s podatkovno bazo) .....	37
Predvajanje.cs (Razred za predvajanje glasbe) .....	39
Zahvala mentorju in ostalim .....	40

## *Predstavitev problema*

V današnjem času lahko preko računalnika naredimo že marsikaj. Lahko naročamo preko njega, dostopamo do gradiv za katere smo še pred 20 leti morali v knjižnico, si lahko ogledamo na milijone video posnetkov. Računalnik lahko tudi nadomesti marsikatero napravo. Ena od takih je predvajalnik glasbe.

Ker rad poslušam glasbo sem imel na računalniku nameščen program VLC, (ki je med drugim tudi program za predvajanje filmov. Ker bi pa rad prihranil prostor bi rad imel le predvajalnik, ki bo predvajal glasbo sem odstranil VLC in začel iskati alternativo, ki bi predvajala samo audio datoteke formate .mp3, ob enem pa bila čisto preprosta za uporabo. Na žalost sem prišel do ugotovitve, da takega programa ni. Zato se mi je porodila ideja, da bi sam izdelal predvajalnik, glasbe, ki bo predvajal glasbene datoteke formatov mp3, flac in wav na operacijskem sistemu Windows 10, ob enem pa vseboval res tiste najnujnejše gradnike, ki jih ima vsak predvajalnik glasbe. Ker smo se v šoli učili programiranja v programskem jeziku C# sem se odločil, da ga sam napišem v tem programskem jeziku.



# *Predvajalnik glasbe v jeziku C#*

## *Načrtovanje*

### *Analiza problema*

Potrebno je bilo je analizirati kaj vse potrebujemo za izdelavo aplikacije, ki bo predvajala glasbo. Preden sem začel z delom sem si postavil nekaj osnovnih vprašanj:

#### *KAJ konkretno želim narediti?*

Želim narediti enostaven predvajalnik glasbe, ki bo predvajal mp3 audio datoteke in hkrati zasedel čim manj prostora.

#### *KAKO bom to realiziral?*

To bom realiziral s pomočjo znanja, ki sem ga pridobil v zadnjih 4 letih.

#### *KATERA orodja bom uporabil?*

Uporabil bom:

- Programski jezik C# (izg. si šarp) za realiziranje rešitve danega problema
- Programske knjižnice C# za lažje delo s C#
- Adobe Illustrator CS6 za risanje ikon gumbov in diagrama poteka
- Razvojno okolje Visual Studio 2015, v katerem je bil program napisan
- Program za dokumentiranje projektne naloge Microsoft Office Word 2016

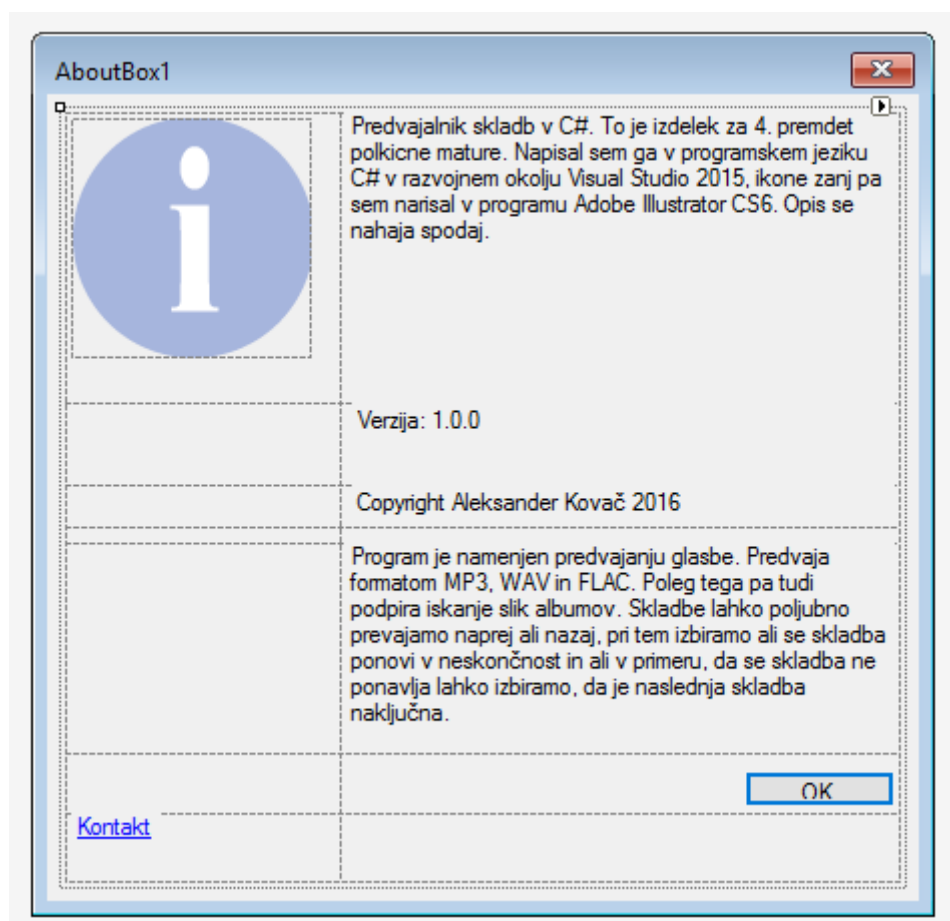
#### *KAKO bom to predstavil?*

Izdelek bom predstavil s pomočjo programa Microsoft Office Power Point 2016

# Ideja



Slika 1 Idejni izgled skladbe

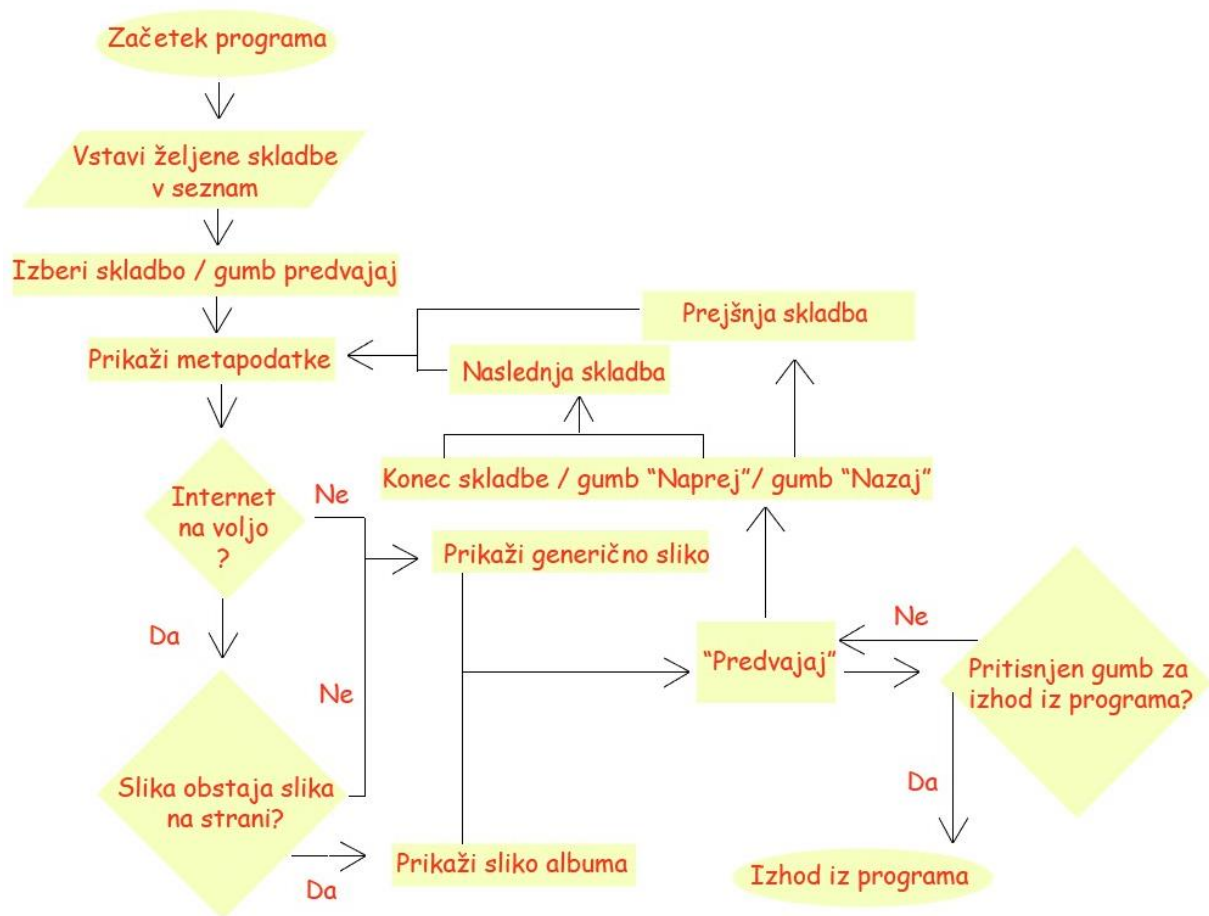


Slika 2 Izgled okna "O programu".



# Diagram poteka

Naslednja slika prikazuje diagram poteka programa.



Slika 3 Diagram poteka izvajanja programa

## Pojmi

**Algoiritem** je navodilo, s katerim rešujemo nek problem. Običajno je zapisan kot seznam korakov, ki nas pripeljejo do rešitve problema. Kako podrobno razdelamo korake, je odvisno od tega, kdo izvaja algoritem (človek, računalnik). Če algoritem izvaja računalnik, potem govorimo o računalniškem programu. Primer algoritma iz vsakdanjega življenja je kuharski recept.

**Diagram poteka** uporabljamo za raševanje problema in je vizualna predstavitev toka podatkov. Diagram se riše s standardnimi elementi

**IDE** ali vgrajeno razvojno okolje je skupek programov s katerimi razvijamo programsko opremo. Razvojno okolje običajno sestavljajo urejevalnik teksta, prevajalnik oziroma interpreter ter povezovalnik.

**MP3** je priljubljena digitalna oblika zapisa glasbenih datotek, ki je bila razvita z namenom zmanjšanja velikosti slednjih. V ne stisnjeni obliki le-te zasedajo veliko prostora. Je oblika zapisa glasbe z izgubami. Omogoča predstavitev analognega signala na racionalnejši način.

**Metapodatek** (opodatek) je podatek, ki vsebuje informacije o nekem podatku, a ni del le-tega. Primer metapodatkov so dimenzije fotografije, vendar niso del slike.

Metapodatki vsebujejo podatke o podatkih; obsegajo podatke, ki se nanašajo na vsebino, strukturo, kvaliteto, lastništvo, distribucijo, tehnologijo, namen, uporabnost in druge elemente, ki so pomembni za pravilno interpretacijo oziroma uporabo podatkov.

**Podatkovna baza** je posplošena združena zbirka podatkov skupaj z njenim opisom, ki jo uporabljamo tako, da zmore zadostiti vsem različnim potrebam uporabnikov.

Je zbirka:

- shranjenih delovnih podatkov, ki jih uporabljajo podsistemi opazovane organizacijske enote.
- Med seboj povezanih podatkov o organiziranem delovno zaključenem sistemu, ki so namenjeni različnim uporabnikom.
- povezanih podatkov, pri čemer so podatki dejstva, ki jih lahko zabeležimo in imajo nedvoumen pomen.
- med seboj pomensko povezanih podatkov, ki so shranjeni v računalniškem sistemu, dostop do njih je centraliziran in omogočen s pomočjo sistema za upravljanje s podatkovno zbirko.

Je mehanizirana, večuporabniško formalno definirana in centralno nadzirana zbirka podatkov.

**Vektorska grafika**. Slika na računalniku je predstavljena z nizom predmetov, od katerih je v pomnilniku zapisan z matematično enačbo. Vsak predmet ima lastnosti, ki ga povsem določajo točka (koordinati, barva) in ravna črta (začetna in končna točka, barva, debelina). Vektorska grafika zasede relativno malo prostora. Formati v katerih je vektorska grafika shranjena so npr. : .ai (Adobe Illustrator), .dwg (AutoCAD).





## Orodja za izdelavo naloge

Za vsak izdelek potrebujemo tudi primerno orodje, da ga lahko naredimo. Pri izdelovanju programa za predvajanje glasbenih datotek s končnico .mp3 in sem uporabljal orodja omenjena v poglavju »Načrtovanje«.

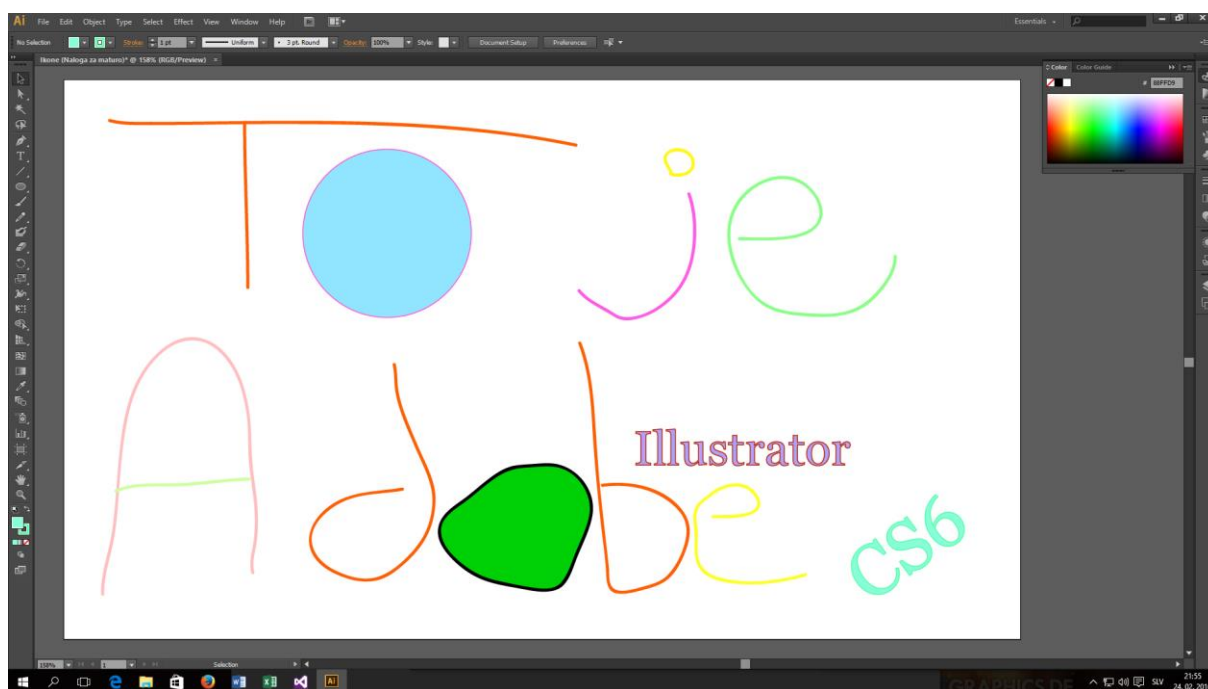
### Adobe Illustrator CS6

Potreba po uporabi programa je nastala, ker sem se odločil program opremiti z ikonami. Le-te sem poskušal sneti s spletnih strani, vendar se niso ujemale, tako v slogu, kot v barvi in velikosti. Zato sem sklenil, da bo najbolje, da jih narišem sam s pomočjo s programov za vektorsko grafiko. Izbral sem program Adobe Illustrator zbirke Creative Suite različice 6.

Adobe Illustrator je profesionalni grafični računalniški program za risanje vektorskih grafik, razvit od podjetja Adobe Systems.

Navodila za uporabo programa so naslednja:

Ko zaženemo program v zavihku »File« izberemo možno »New«. Dokument poimenujemo in mu dodamo profil. Lahko izbiramo med različnimi profili, kot so na primer dokument za splet, tisk, naprave ali video in film, lahko pa izdelamo tudi svoj profil dokumenta. Temu lahko določimo različne parametre kot so mere dokumenta v različnih enotah, orientacijo, število strani in tako naprej. Ko smo nastavitve izbrali, kliknemo na tipko »OK«. Izbiranje oblik nam na levi strani omogoča gumb v obliki pravokotnika, besedilo dodamo s pomočjo gumba »T«, lahko rišemo krivulje s čopičem pa tudi z barvno paletto objektom dodajamo barve. To so tudi funkcije, ki sem jih sam uporabil za risanje. Po končanem oblikovanju v zgoraj omenjenem zavihku poiščemo možnost »Save As..« in shranimo v formatu datoteke, ki je se nam zdi najprimernejši.



Slika 4 Vmesnik programa Adobe Illustrator CS6

Izdelati je bilo potrebno naslednje ikone:

- Ikone vmesnika:

- Predvajaj



*Slika 5 Ikona za predvajanje*

- Ustavi



*Slika 6 Ikona za ustavitev predvajanja*

- Prejšnja skladba



*Slika 7 Ikona gumba za prejšnjo skladbo*

- Naslednja skladba



- Ikona programa



*Slika 8 Ikona programa*

- Slika mp3 skladbe ne obstaja



*Slika 9 Slika albuma MP3 skladbe manjka*



- Slika wav skladbe ne obstaja

\*.wav

- Slika 10 Slika albuma WAV skladbe ni mogoče pridobiti

- Slika wav skladbe ne obstaja

\*.flac

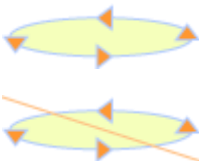
Slika 11 Slika albuma FLAC skladbe ni mogoče pridobiti

- Slika okvirčka, ko se program naloži



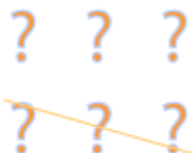
Slika 12 Slika okvirčka, ko se program naloži

- Slikici, ki nam sporočata ali se skladba ponavlja



Slika 13 Ponavljanje skladbe vklop / izklop

- Slikici, ki nam sporočata ali se predvaja naključno izbrana skladba



Slika 14 Naključno predvajanje skladb

Vse slike razen ikone programa, so prikazujejo le osnovne gumbe, ikona programa pa je vendar ikona programa, zato sem jo tudi vizualno drugače naredil.

## Programski jezik C#

C# je objektno orientiran programski jezik ki ga je razvil Microsoft. Jezik izvira pretežno iz programskih jezika C++. Trenutno obstaja v različici 6. Namenjen je pisanju programov v okolju Windows in je primeren za razvoj programske opreme. Poleg tega je zasnovan na tak način, da ga je mogoče preprosto izboljševati in razširjati, brez nevarnosti, da bi s tem izgubili združljivost z obstoječimi programi. C# je bil oblikovan za delo z Microsoftovo .NET platformo (.NET Framework). To je platforma, ki predstavlja ogrodje za vsa .NET orientirana programska orodja in aplikacije za osebne računalnike, dlančnike, pametne telefone. Platforma omogoča gradnjo uporabniških vmesnikov, dostopanje do podatkov, razvoj aplikacij, ki za delovanje potrebujejo internet ter podpira številne algoritme.

Kot razvojno okolje se predvsem uporablja Visual Studio, obstajajo pa tudi druga razvojna okolja kot je na primer SharpDevelop za Linux.

## Prvine jezika C#

V naslednjih poglavjih so opisani osnovni principi programskega jezika C#.

### Podatkovni tipi

Vsaka spremenljivka v jeziku C# pripada določenemu tipu. Tip spremenljivki določimo z njeno deklaracijo, s tem pa določimo tudi, katere operacije bomo lahko izvajali nad to spremenljivko.

Podatkovni tip deklariramo na naslednji način:

**podatkovni tip ime tipa (se ne sme začeti na številko/na ne angleški znak) = prirejena vrednost;**

V programu sem uporabil različne podatkovne tipe. Te sem tudi označil in dodal komentar. Nahajajo se v kodi od Form1.cs, ki se nahaja v prilogi tega dokumenta

### Pogojni stavki

Vsak programski jezik dobi pravo moč z uporabo krmilnih stavkov, kot so vejitve in zanke. Tudi tu C# ni izjema.

Stavek **if** se izvede takrat kadar drži pogoj (ima vrednost **true**), ki je zapisa v oklepaju.

Druga oblika stavka **if** je z uporabo stavka **else**. Stavki, ki so v **else**, se izvedejo takrat kadar pogoj ne drži (ima vrednost **false**).

(Horvat, september 2014)

Pogojne stavke tvorijo t. i. primerjalni operatorji, s katerimi primerjamo dva operanda. Rezultat primerjave je logična vrednost resnično (**true**) ali neresnično (**false**). Najpogosteje se z njimi srečujemo v zankah ali pogojnih stavkih. Primeri za te so je enako (**==**), ni enako (**!=**), je večje/manjše (ali enako).



Prav tako pa uporabljamo logične operatorje s katerimi izvajamo operacije nad logičnimi vrednostmi. Rezultat je lahko resničen (true) ali neresničen (false). Če je (a večji od b) in (c večji od d) potem izvedi ukaz.

Logični operator && vrne vrednost resnično, kadar so vsi pogoji resnični. Logični operator ali (||) vrne vrednost true, kadar je vsaj eden izmed pogojev resničen. Logični operator negacije(!=), ki negira vrednost operanda. Vrednost true tako postane false in nasprotno.

## Zanke

Z zankami lahko določeno zaporedje ukazov večkrat izvedemo. Poznamo štiri zanke **while**, **for**, **do-while** in **foreach**. Izmed vseh možnih sem se odločil za zanko foreach zaradi preprostosti dela z njo:

- Zanka foreach izvede sprehod skozi seznam elementov (npr. *polja*), pri tem pa sproti vrača vrednost *trenutnega elementa* ali *indeks in vrednost*. V programu sem jo uporabil, da sem obdelal več elementov naenkrat, brez potrebe implementacije funkcije, ki bi štela število elementov. Prednost te zanke, je da ni potrebno navesti število ponavljanj ampak se zanka samodejno ponavlja, dokler vsi izbrani elementi niso obdelani. Uporabil sem jo v Form1.cs pri polnjenju seznama.

(Horvat, september 2014)

## Funkcije

Tako kot C++, pozna tudi C# funkcije. Če določen del kode rabimo večkrat ga je smiselno dati v funkcijo. Primer: Z vsakim klikom na sliko, bi radi sliko povečali in zatemnili ozadje. Slik imamo 50. Da ne bi za vsako sliko pisali enake kode za zatemnitev ozadja in povečavo, lahko izdelamo funkcijo in jo pokličemo takrat, ko jo rabimo.

Funkcijo definiramo tako, da napišemo tip funkcije, njeno ime, ki je lahko poljubno, ter morebitne parametre, ki jih funkcija sprejme. Vse funkcije, ki niso tipa »void« (prazne) morajo ime »return« stavek, kjer vračajo vrednost takšni obliki, kot smo zapisali tip funkcije.

Funkcije sem napisal za številne stvari (npr. predvajanje glasbe) od ostalih stvari pa se razlikujejo po tem, da so definirane kot podatkovni tip, sledi oklepaj z izbirnimi vhodnim spremenljivkami in samo kodo med zavitim oklepajem.

(Horvat, september 2014)

## Razredi

Pri objektno usmerjenem programiranju se ukvarjamo z objekti. Objekt je nekakšna črna škatla, ki dobiva in pošilja sporočila. V tej črni škatli (objektu) se skriva tako koda (torej zaporedje programskih stavkov), kot tudi podatki (informacije nad katerimi se izvaja koda). V objektno usmerjenem programiranju (OO) pa sta koda in podatki združeni v nedeljivo celoto – objekt. To prinaša določene prednosti. Ko uporabljamo objekte, nam nikoli ni potrebno pogledati v sam objekt. To je dejansko prvo pravilo OO programiranja – uporabniku ni nikoli potrebno vedeti, kaj se dogaja znotraj objekta. Gre dejansko zato, da nad objektom izvajamo različne metode. In za vsak objekt se točno ve, katere metode zanj obstajajo in kakšne rezultate vračajo. Recimo, da imamo določen objekt z imenom `robotek`.

Vemo, da objekte take vrste, kot je `robotek`, lahko povprašamo o tem, koliko je njihov IQ. To storimo tako, da nad njimi izvedemo metodo `robotek.KolikoJeIQ()`;

Razred (class) je abstraktna definicija objekta (torej opis, kak o je naša škatla videti znotraj). Veliko razredov je že vgrajenih (so v standardnih knjižnicah) v C# in njegovo okolje, pogosto pa je za naše aplikacije potrebno napisati tudi kak svoj razred. Z razredom opišemo, kako je neka vrsta objektov videti. Če na primer sestavljamo razred zajec, opisujemo, katere so tiste lastnosti, ki določajo vse zajce.

V razredu zapišemo lastnosti, stanja in obnašanje objektov: zapišemo torej katere podatke bomo hranili v objektih te vrste in katere metode oz. odzive objektov na sporočila. Stanje objekta torej opišemo s spremenljivkami (rečemo jim tudi polja ali komponente), njihovo obnašanje oz. odzive pa z metodami. Razredi so vključeni v prilogo. To so `Form1.cs`, `SQLite.cs`, `Metapodatki.cs` ter `Predvajanje.cs`.

## Ostali programski stavki

### *Try {} catch {} finally {}*

Tako kot vsakdanjem življenju se tudi pri delovanju programov pojavljajo napake. Zelo neprijetno je, če se program sredi delovanja "sesuje" (neha delovati), ker na primer programer ni predvidel, da lahko v določenih izjemnih primerih pride do "čudne" situacije. C# pozna tako imenovan mehanizem izjem (exceptions). Ideja je v tem, da dele programa, ki so "kritični" obdamo z varovalnim blokom imenovanim »Try«. Če v tem varovalnem bloku pride do napake, se izvede lovilni del t.i. »Catch«, kjer lahko ob napakah ustrezno reagiramo. Del »Finally« pa se izvede ne glede na to ali se je napak zgodila ali ne. Uporabil sem pri branju vrednosti iz baze in vstavljanju datotek.

(Uranič & Lokar, Programski jezik C#: Izjeme, november 2008)

### *Generični podatkovni tipi*

Generični podatkovni tipi so tipi, kjer namesto konkretnega tipa podatka (npr. `int`, `string`, ...) zapišemo `T` in s tem prevajalniku povemo, naj sam ugotovi, kakšen tip podatka predstavlja `T` ob izvajanju. Pozor: Namesto `T` bi lahko uporabili kakšno drugo oznako, ki seveda ni



rezervirana. Pa si oglejmo kar primer, ko ustvarimo seznam List <>, ki mu moramo med < > podati tip podatka, zato je razred List <> generičen. Kar pomeni, da če bomo namesto T zapisali int, bo to seznam namenjen shrambi celih števil. Generične podatkovne tipe sem uporabil pri vstavljanju skladb. Generični podatkovni tip string sem uporabil pri nalaganju

### *Delo z datotekami*

Program, ki sem ga napisal podpira tudi delo z datotekami in imeniki. Če datoteke za podatkovno bazo SQLite jo mora program narediti, pred tem pa more pogledati, če obstaja glavni imenik programa in imenik v katerega se shranjujejo slike albumov, ter če je lokalno ta slika na voljo. Program med nalaganjem preveri razpoložljivost imenikov in datoteke podatkovne baze oziroma med samim izvajanjem preveri razpoložljivost slike albuma. Po potrebi kreira manjkajoče imenike / datoteke. Obstoj imenikov / datotek program preveri na začetku izvajanja.

Preverjal sem obstoj datotek/imenikov:

- Datoteka SQLite v kateri je podatkovna baza
- Imenik kjer sta datoteka SQLite in tudi imenik, ki hrani datoteke skladb.

Imenike programski jezik naredi sam, v odvisnosti kje se nahaja sistemska mapa »Dokumenti«.

### *Delo z nizi*

Ker spletna stran gracenote.com za iskanje slike albuma potrebuje naslov izvajalca in albuma in je niz Ime\_albuma\_[dodatne\_oznake] nesprejemljiv, moramo te dodatne oznake vključno z znaki »()«, »[]«, »Disc stevilka\_diska«. Zato se uporabil funkcije, ki te oznake izbrišejo iz metapodatka Album. Funkcije za delo z nizi se uporabljajo za odstranjevanje določenih znakov oziroma delov besedila. Delo z nizi je predstavljeno v prilogi v razredu Metapodatki.cs.

## Programske knjižnice

Programska knjižnica je v računalništvu zbirka podprogramov za pomoč pri izdelavi oziroma razvoju programske opreme. Knjižnice vsebujejo kodo in podatke, ki se jih da uporabiti v neodvisnih programih. Zaradi tega se programi izmenjujejo in spreminjajo modularno.

V programu sem uporabil več vrst knjižnic. Kar precej jih .NET ogrodje že vsebuje, sem pa uporabil tudi knjižnice, ki se v program povezujejo dinamično t. i. DLL-ji ter zunanje knjižnice, ki sem jih snel s spleta. Za predvajanje glasbe sem uporabil DLL sistema Windows 10, ki se imenuje winmm.dll. Omogoča predvajanje raznih audio formatov. Uporabil pa sem tudi zunanji knjižnici ta delo z metapodatki. To sta knjižnici Taglib za urejanje metapodatkov in Gracenote za prenos slike albuma. Sledi seznam knjižnic ogrodja .NET :

- System.Collections.Generic omogoča uporabo generičnih podatkovnih tipov
- System.Data.SQLite nam omogoča delo s podatkovno bazo SQLite
- System.IO – branje in pisanje v datoteke
- System.Runtime.InteropServices uporabljamo ko potrebujemo knjižnice, ki se v program povežejo dinamično moramo uporabiti to knjižnico
- System.Text knjižnica nam omogoča delo z nizi
- System.Windows.Forms nam omogoča delo z okni. Skrbi za prikaz okenske aplikacije, pa tudi delo z gradniki

## Taglib

Taglib je programska knjižnica je namenjena branju in urejanju metapodatkov različnih audio formatov. Trenutno podpira formate, kot so mp3, ogg, flac, aiff, mp4 ter druge pogostejše uporabljene formate.

Taglib knjižnico uvozimo v projekt tako, da jo snamemo s spleta (dosegljiva na <https://www.nuget.org/packages/taglib/>) ter v Package Manager Console (najdena pod zavihkom »Tools > Library Package Manager > Packet Manager Console«) vnesemo ukaz Install-Package taglib. Dobimo obvestilo, da se ja paket namestil uspešno.

## Gracenote

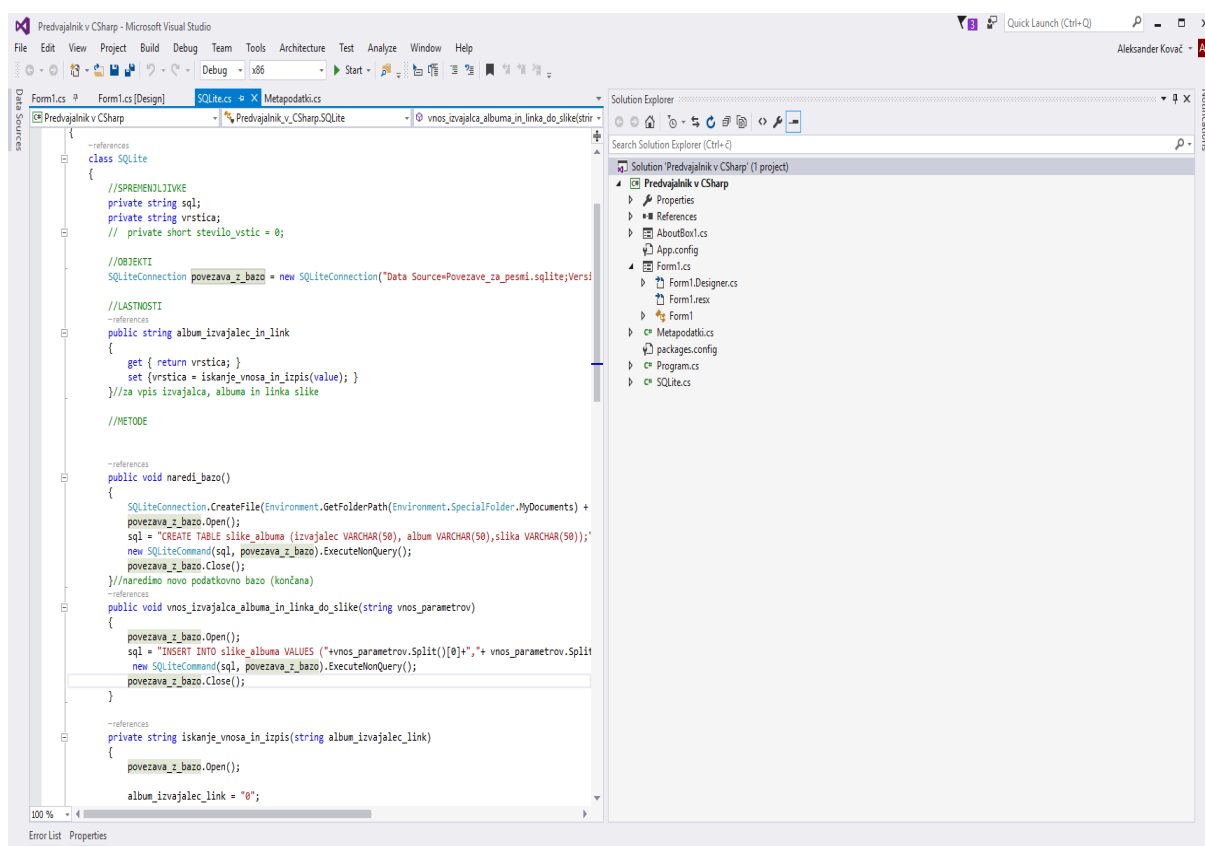
Omenjena knjižnica omogoča, da se povežemo na strežnik podjetja Park Square, kjer se hranijo slike albumov skladb. Z funkcijami, ki jih knjižnica omogoča, sem sliko poskusil poiskati in v primeru, da je bila najdena sem jo s pomočjo funkcij tudi snel s spleta in jo vstavil ob predvajanju skladbe. Uvozimo jo na enak način, kot knjižnico Taglib. Paket, ki pa nam omogoča uporabo te knjižni pa se imenuje ParkSquare.Gracenote.





## Visual studio 2015

Aplikacijo sem napisal v razvojnem okolju Visual Studio 2015, ki je bila v času nastajanja te aplikacije najnovejša različica razvojnega okolja Visual Studio. Razvilo ga je podjetje Microsoft. Izbira le-tega je temeljila na dejstvu, da je to okolje zelo kvalitetno, uporabniku prijazno in tudi Okna (Microsoft Windows oz. MS- Windows) so še zmeraj prevladujoč operacijski sistem na trgu. Združuje več programskih jezikov med katerimi sta tudi jezika C++ in C#, in prevajalnike zanje. Z njim je mogoče ustvarjati tudi spletne aplikacije, video igre, ter aplikacije, ki se dajo poganjati na vseh napravah s sistemom Windows 10.



Slika 15 Vmesnik razvojnega okolja Visual Studio 2015

Če želimo ustvariti svojo aplikacijo zaženemo program Visual Studio 2015, ter na pozdravni strani izberemo »New Project«. Odpre se nam okno za ustvarjanje novega projekta nakar sledi izbiranje ustreznega programskega jezika in tipa aplikacije, ki jo želimo narediti. Izberemo jo, dodamo ime aplikacije, ki je ponavadi hkrati ime rešitve, ter kam se naj rešitve shrani. Sledi nalaganje osnovnega okna. Ko izberemo Windows Forms Application se nam naloži okno. Na njega nato vlečemo gradnike iz orodjarne. Po desnem kliku kliknemo na »View code« in začnemo pisati kodo v izbranem programskem jeziku. Če želimo naš program preizkusiti, kliknemo na »Start« ali pritisnemo na gumb F5 na tipkovnici.

Čar take aplikacije so pa seveda gradniki. Te najdemo pod zavihkom »View>ToolBox« Tam izmed številnih gradnikov izberemo tiste. Uporabil sem naslednje gradnike:

- AboutBox gradnik služi za predstavitev osnovnih informacij o programu.
- Štoparica je gradnik, ki skrbi za štetje časa.
- Dialog za odpiranje datotek je gradnik, ki odpre eno ali več datotek, te iz njih izbere poti datotek. Prikaže se nam pojavno okno, kjer lahko s pomočjo filtra določimo katere vrste datotek bomo odpirali.
- Gumb je jedro okenske aplikacije. Z dvoklikom na gumbe, ki smo jih povlekli na okno začnemo pisati kodo za le-te.
- Label je gradnik, s pomočjo katerega na okensko aplikacijo dodamo oznake. Ponavadi jim prirejamo besedilo. Posebna različica »labela« je link label, s klikom nanj lahko pridemo do povezave, ki je prikazana.
- Listbox lahko vsebuje več predmetov tipa string, vsakega ločenega v svoji vrstici, gre za neke vrste seznama.
- Meni je gradnik s katerim delamo podobno kot z gumbi. Element izberemo v toolboxu, opcije pa vstavljamo v prazne pravokotnike, ki se nam samodejno izrisujejo. Z dvoklikom na opcijo v meniju začnemo pisati kodo za opcije v meniju.
- PictureBox – v njega vstavljamo želene/e slike/e.
- S pomočjo »trackbara« oziroma po slovensko sledilne črte dobimo vizualno predstavitev na katerem delu skladbe se nahajamo.
- Windows Form – je tipično okno v sistemu Windows. V oblikovalniku na njega povlečemo zgoraj omenjene elemente. Z dvoklikom na gumbe, ki smo jih povlekli na okno začnemo pisati kodo za le-te.



## Podatkovna Baza SQLite

SQLite je podatkovna baza, uporabna predvsem za lokalno shranjevanje podatkov, kjer ni potrebe po dodatnem podatkovnem strežniku. Je odprta podatkovna baza, ki pa je za razliko od ostalih podatkovnih baz sestavni del aplikacije, ki jo uporablja. Ima lastnosti običajnih podatkovnih baz, kot je na primer sintaksa SQL. Zaradi svojih odlik (npr. majhna poraba delovnega pomnilnika, integracija v samo aplikacijo, podpora za različne programske jezike, ...) je zelo priljubljena in se med drugim uporablja v spletnih brskalnikih, operacijskih sistemih. V svojem programu sem jo uporabil, kot shrambo lokacije slik albumov na podatkovnem nosilcu..

## Delo s podatkovno bazo

Tako kot ostali programski jeziki, ima tudi C# možnost dela s podatkovno bazo.

C# podpira številne podatkovne baze med drugim tudi podatkovno bazo SQLite. Z ukazom **SQLiteConnection.CreateFile(»poljubna lokacija«);** jo kreiramo na poljubni lokaciji ter ustvarimo povezo do nje z ukazom **SQLiteConnection povezava z bazo = new SQLiteConnection(»Data Source=Povezave za pesmi.sqlite;Version=3;«);**, ki jo potem pred operacijo odpremo in po njej zapremo.

Za upravljanje z bazo uporabimo ustrezen SQL stavek ter ustvarimo objekt razred **SQLiteCommand.(sql stavek, povezava z bazo);** Metodi, ki sledita sta **.ExecuteNonQuery();** za izvedbo stavka oziroma izpis vrednosti ukaza v spremenljivko in sicer po sintaksi **spremenljivka=(tip spremenljivke) objekt.ExecuteScalar();**

Jezik Structured Query Language se deli na dva dela in sicer na jezik za definicijo tabele podatkovne baze in jezik za manipulacijo s podatki.

Stavki, ki sem jih uporabil pri naslednji:

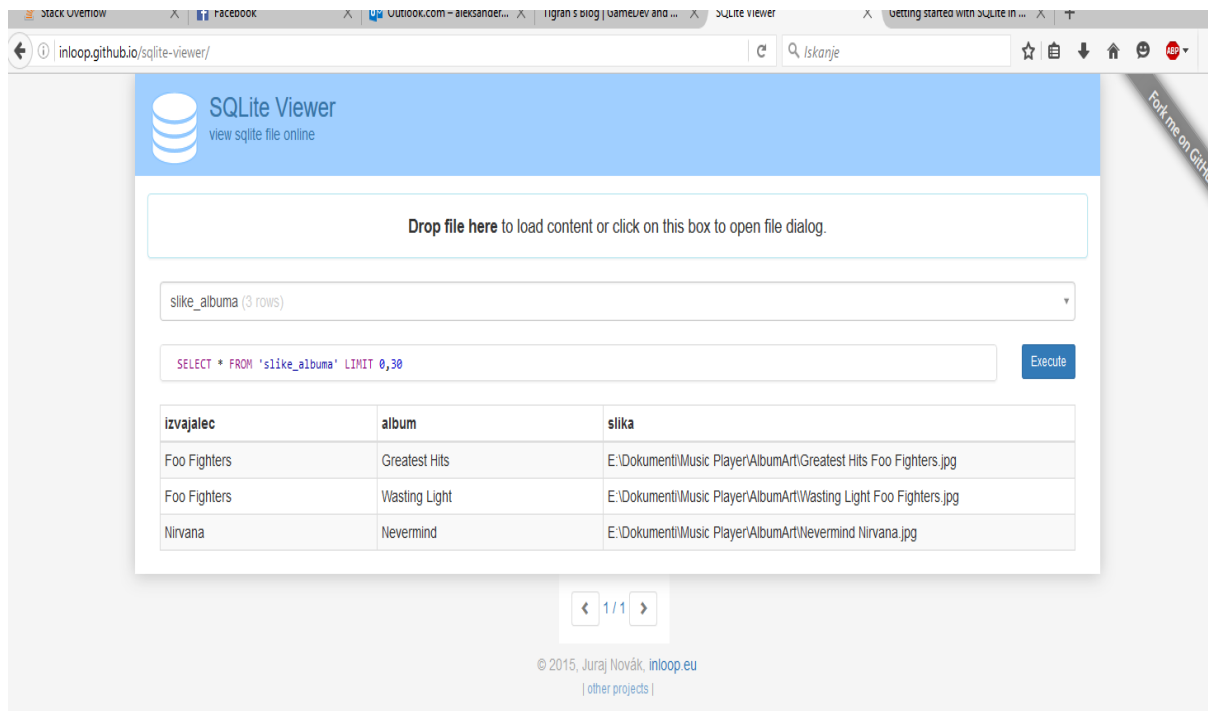
- CREATE TABLE – s tem stavkom ustvarimo tabelo in jo tudi definiramo.
- SELECT \* FROM – stavek izbire željen stolpec iz željene tabele pod določenimi pogoji.
- INSERT INTO – s tem stavkom vstavljamo vrednost v tabelo.

Sledi struktura tabele in primeri zapisov.

Slike albuma
Izvajalec (besedilo dolžine 25 znakov), Album (besedilo dolžine 20 znakov), Pot do slike (besedilo, dolžino 75);

Slika 16 Struktura podatkovne baze

S pomočjo spletne aplikacije »SQLite Viewer« (dosegljivo na: [inloop.github.io/sqlite-viewer/](http://inloop.github.io/sqlite-viewer/)) lahko vidimo vnose v tabeli `slike_albuma`:



The screenshot shows the SQLite Viewer web application interface. At the top, there's a blue header with the SQLite logo and the text "SQLite Viewer view sqlite file online". Below the header, there's a large white box with the text "Drop file here to load content or click on this box to open file dialog." Below this, there's a dropdown menu showing "slike\_albuma (3 rows)". Below the dropdown, there's a text input field containing the SQL query "SELECT \* FROM 'slike\_albuma' LIMIT 0,30" and a blue "Execute" button. Below the query input, there's a table with 3 columns: "izvajalec", "album", and "slika". The table contains 3 rows of data. At the bottom of the table, there's a pagination control showing "1 / 1". Below the pagination control, there's a footer with the text "© 2015, Juraj Novák, [inloop.eu](http://inloop.eu)" and a link to "other projects".

izvajalec	album	slika
Foo Fighters	Greatest Hits	E:\Dokumenti\Music Player\AlbumArt\Greatest Hits Foo Fighters.jpg
Foo Fighters	Wasting Light	E:\Dokumenti\Music Player\AlbumArt\Wasting Light Foo Fighters.jpg
Nirvana	Nevermind	E:\Dokumenti\Music Player\AlbumArt\Nevermind Nirvana.jpg

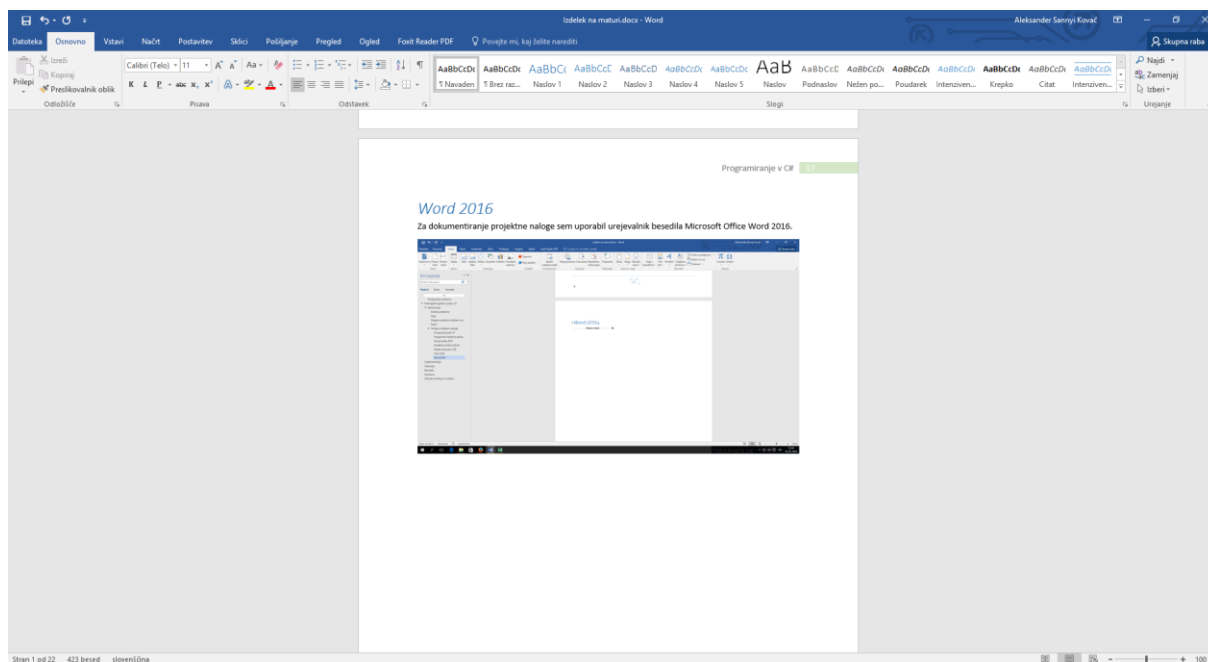
Slika 17 Tabela »slike\_albuma« v podatkovni bazi »Povezave\_za\_pesmi.sqlite«



## Microsoft Office Word 2016

Microsoft Office je zbirka pisarniških programov, ki jih razvija in trži podjetje Microsoft na Windows in Macintosh platformi. Microsoftov Office je trenutno najbolj razširjena zbirka pisarniških programov na svetu.

Microsoft Word je urejevalnik besedil in je del pisarniške zbirke Microsoft Office-a. Zaradi razširjenosti predstavlja Wordov format zapisa besedila, ki ga poznamo po podaljšku .doc standard med zapisi besedil, čeprav novejšje verzije Word-a omogočajo tudi zapis besedila v Microsoftovi različici XML, ki za razliko od standardne različice XML ni povsem odprta.



Slika 18 Vmesnik programa Microsoft Office Word 2016

## Testiranje

Testiranje programa sem opravil po naslednjih točkah. V tabeli sem z oznako »Uspešno« označil katere stvari delujejo tako kot je treba.

Rezultat	Funkcija	Opis
Uspešno	Nalaganje datotek	Odprtje dialoga in nalaganje skladb v program <ul style="list-style-type: none"> <li>• preko gumba predvajaj,</li> <li>• s klikom na prazen seznam,</li> <li>• s klikom na zavihek Datoteka&gt;Odpri&gt;Audio datoteka.</li> </ul>
Uspešno	Predvajanje	Predvajamo glasbo: <ul style="list-style-type: none"> <li>• s klikom na gumb predvajaj (lihi kliki).</li> <li>• s klikom na ime skladbe v seznamu.</li> </ul>
Uspešno	Ustavitev	Ustavimo glasbo: <ul style="list-style-type: none"> <li>• s klikom na gumb predvajaj (sodi kliki).</li> </ul>
Uspešno	Naslednja skladba	Ob pritisku naslednjo skladbo: <ul style="list-style-type: none"> <li>• predvajanje naslednje skladbe,</li> <li>• če je skladba, ki se predvaja, zadnja na seznamu, skoči na prvo skladbo na seznamu.</li> </ul>
Uspešno	Prejšnja skladba	Ob pritisku gumba: <ul style="list-style-type: none"> <li>• predvajanje prejšnje skladbe,</li> <li>• če je skladba, ki se predvaja, prva na seznamu, skoči na zadnjo skladbo na seznamu.</li> </ul>
Uspešno	Metapodatki	Metapodatki: <ul style="list-style-type: none"> <li>• se ob naložitvi skladbe pravilno prikažejo.</li> </ul>
Uspešno	Prikaz pravilne slike	Ko se skladba začne predvajati: <ul style="list-style-type: none"> <li>• se naloži slika albuma, če je povezava z internetom vzpostavljena in slika albuma obstaja v podatkovni bazi spletne strani gracenote.com</li> <li>• se naloži slika s končnico datotek, ki sporoča, da slika albuma ni najdena, ker ni povezave z internetom ali ker slika na zgoraj omenjeni spletni strani ni bila najdena.</li> </ul>
Uspešno	Prikaz informacij o programu	Ob kliku na zavihek »O programu« se prikaže okno »O programu«.
Uspešno	Delovanje gumbov »Naključno« in »Ponovi«	Ob kliku se naslednja skladba izbere naključno. Ob kliku ponovi, se trenutna skladba ponavlja v neskončnost.

Tabela 1 Testiranje programa



## Navodila

Hvala, ker ste se odločili uporabljati moj predvajalnik glasbe napisan v programskem jeziku C#. V naslednjih nekaj stavkih, Vam bom predstavil uporabo tega programa.

## Začetek

Za začetek, kliknimo na izvršljivo datoteko »Predvajalnik v CSharp«. Da bi glasbo poslušali, izberemo nekaj glasbenih datotek. Da naložimo glasbene datoteke v program izberemo eno od naslednjih možnosti:

- Premaknemo se na zavihek »Datoteka«, pod zavihek »Odpri« ter izberemo gumb »Skladbe«.
- Pritisnemo prvič na gumb »Predvajaj«.
- Kliknemo na prazno polje v programu.
- Kliknemo na gumba »Nazaj« ali »Naprej«.

Odpre se dialog, v katerem bomo izbiramo skladbe v zapisu mp3. Izberemo poljubne skladbe in kliknemo »Odpri«. Imena glasbenih datotek se bodo pojavila v prej praznem polju . Če bomo v dialogu izbrali »Prekliči«, glasba ne bo naložena, to obvestilo, pa se bo tudi izpisalo.

## Predvajanje datotek

Ko smo datoteke naložili v seznam, jih lahko predvajamo. To storimo tako, da:

- kliknemo na gumb »Predvajaj«. Ob lihem kliku na gumb, ko so datoteke naložene, se bodo skladbe predvajale od začetka, ob sodem, pa se bo predvajanje zaustavilo. Predvajala se bo prva skladba naložena v seznamu.
- kliknemo na gumb »Naprej«. Če ta gumb kliknemo prvič, se vam bo predvajala druga skladba po vrsti. S tem gumbom prekinemo izvajanje trenutne skladbe in začnemo predvajanje naslednje, če se predvaja zadnja skladba in kliknemo omenjen gumb se začne predvajati prva skladba v seznamu.
- kliknemo na gumb »Nazaj«. Če ta gumb kliknemo prvič, se vam bo predvajala zadnja skladba po vrsti. S tem gumbom prekinemo izvajanje trenutne skladbe in začnete predvajanje prejšnje, če se predvaja prva skladba in kliknete omenjen gumb se začne predvajati zadnja skladba v seznamu.
- kliknemo na ime skladbe. Če to storite prekinete predvajanje trenutne skladbe, če se seveda kaj predvaja. Začne se predvajanje skladbe, na katero ste kliknili.

## *O programu*

Lahko si tudi ogledamo informacije o programu. To naredimo tako, da kliknemo na zavihek »O programu«. Prikaže se okno »O programu«, kjer je naslov programa, njegov opis navedena trenutna različica programa ter e-pošto, preko katere sem dosegljiv za morebitna vprašanja glede samega programa.

## *Izhod iz programa*

Če hočemo program končati, kliknemo na rdeči gumb »X« ali pa se pomaknemo na zavihek »Datoteka« in izberemo možnost izhod.





# Literatura

## Bibliografija

Horvat, S. (september 2014). *Načrtovanje in razvoj spletnih aplikacij*. Murska Sobota, Slovenija.

Uranič, S., & Lokar, M. (april 2009). *Programski jezik C#*. Kranj. Pridobljeno iz [http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/C%23\\_objektnoSreco.pdf](http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/C%23_objektnoSreco.pdf)

Uranič, S., & Lokar, M. (november 2008). *Programski jezik C#: Izjeme*. Kranj. Pridobljeno iz [http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/C%23\\_izjeme.pdf](http://uranic.tsckr.si/VI%C5%A0JA%20%C5%A0OLA/Programiranje%20I/C%23_izjeme.pdf)

## Svetovni splet

<https://taglib.github.io/>  
Taglib: Opis knjižnice [05.02.2016]

[https://de.wikipedia.org/wiki/Integrierte\\_Entwicklungsumgebung](https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung)  
Wikipedija: IDE ali razvojno okolje [24.02.16]

[https://sl.wikipedia.org/wiki/Podatkovna\\_zbirka#Definicije](https://sl.wikipedia.org/wiki/Podatkovna_zbirka#Definicije) [24.02.16]

[https://sl.wikipedia.org/wiki/Programski\\_jezik](https://sl.wikipedia.org/wiki/Programski_jezik)  
Wikipedija: Definicija programskega jezika [24.02.2016]

<https://sites.google.com/site/romanavogrincic/Home/gradiva-za-pouk/slikovna-predstavitev>  
Romana Vogrinčič: Slikovna predstavitev informacije. [24.02.2016]

<https://sl.wikipedia.org/wiki/MP3>  
Wikipedija: Datotečni format MP3 [24.2.2016]

[https://sl.wikipedia.org/wiki/Knji%C5%BEnica\\_%28ra%C4%8Dunalni%C5%A1tvo%29](https://sl.wikipedia.org/wiki/Knji%C5%BEnica_%28ra%C4%8Dunalni%C5%A1tvo%29)  
Wikipedija: Programska knjižnica [24.02.2016]

<https://sl.wikipedia.org/wiki/Metapodatek>  
Wikipedija: Opis metapodatka [24.02.2016]

<https://sites.google.com/site/romanavogrincic/Home/gradiva-za-pouk/zapis-algoritma>,  
Romana Vogrinčič: Zapis algoritma – diagram poteka . (diapozitiv 5) [24.02.2016]

<http://blog.tigrangasparian.com/2012/02/09/getting-started-with-sqlite-in-c-part-one/>,  
Tigran Gasparian: Osnovno delo s podatkovno bazo SQLite. [25.02.2016]

[http://hvasti.tsckr.si/APJ/podatkovni\\_tipi.htm](http://hvasti.tsckr.si/APJ/podatkovni_tipi.htm)  
 Tehnološki šolski center Kranj: Podatkovni tipi [25.02.2016]

[http://www.w3schools.com/sql/sql\\_select.asp](http://www.w3schools.com/sql/sql_select.asp),  
 W3Schools: Stavek Select. [26.02.2016]

[wiki.fmf.uni-lj.si/wiki/Operator](http://wiki.fmf.uni-lj.si/wiki/Operator)  
 Jože Premru: Operatorji [26.02.2016]

[http://www.w3schools.com/sql/sql\\_create\\_table.asp](http://www.w3schools.com/sql/sql_create_table.asp)  
 W3Schools: Stavek CREATE Table [26.02.2016]

<https://www.sqlite.org/serverless.html>  
 SQLite: Delovanje podatkovne baze SQLite[27.02.2016]

[https://sl.wikipedia.org/wiki/Microsoft\\_Office](https://sl.wikipedia.org/wiki/Microsoft_Office)  
 Wikipedija: Microsoft Office 2016 [27.02.2016]

<http://www2.nauk.si/materials/728/out-673531/index.html#state=84>,  
 Anita Kolaržik: Generični podatkovni tipi [27.02.2016]

<https://sl.wikipedia.org/wiki/Algoritem>  
 Wikipedija: Definicija algoritma [28.02.2016]

## *Kazalo slik in kazalo tabel*

### *Kazalo slik*

SLIKA 1 IDEJNI IZGLED SKLADBE .....	5
SLIKA 2 IZGLED OKNA "O PROGRAMU" .....	5
SLIKA 3 DIAGRAM POTEKA IZVAJANJA PROGRAMA .....	6
SLIKA 4 VMESNIK PROGRAMA ADOBE ILLUSTRATOR CS6 .....	8
SLIKA 5 IKONA ZA PREDVAJANJE .....	9
SLIKA 6 IKONA ZA USTAVITEV PREDVAJANJA .....	9
SLIKA 7 IKONA GUMBA ZA PREJŠNJO SKLADBO .....	9
SLIKA 8 IKONA PROGRAMA .....	9
SLIKA 9 SLIKA ALBUMA MP3 SKLADBE MANJKA .....	9
SLIKA 10 SLIKA ALBUMA WAV SKLADBE NI MOGOČE PRIDOBITI .....	10
SLIKA 11 SLIKA ALBUMA FLAC SKLADBE NI MOGOČE PRIDOBITI .....	10
SLIKA 12 SLIKA OKVIRČKA, KO SE PROGRAM NALOŽI .....	10
SLIKA 13 PONAVLJANJE SKLADBE VKLOP / IZKLOP .....	10
SLIKA 14 NAKLJUČNO PREDVAJANJE SKLADB .....	10
SLIKA 15 VMESNIK RAZVOJNEGA OKOLJA VISUAL STUDIO 2015 .....	16
SLIKA 16 STRUKTURA PODATKOVNE BAZE .....	18
SLIKA 17 TABELA »SLIKE_ALBUMA« V PODATKOVNI BAZI »POVEZAVE_ZA_PESMI.SQLITE« .....	19
SLIKA 18 VMESNIK PROGRAMA MICROSOFT OFFICE WORD 2016 .....	20



## Priloga: programska koda

### Form1.cs (Glavni razred)

```
using System;
using System.Drawing;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Windows.Forms;

namespace Predvajalnik_glasbe_v_CSharpu
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            pictureBox1.Image = Resource1.IKONA;
            timer1.Interval = 1000;
            trackBar1.TickFrequency = 1;
            trackBar1.Minimum = 0;
            trackBar1.Maximum = 2;
            trackBar1.Visible = false;
            if
            (!Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) +
            @"\Music Player"))//preveri ce obstaja imenik z imenom Media Player
            {

                Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\Music Player"); //naredimo imenik za media player

            }
            if
            (!Directory.Exists(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) +
            @"\Music Player\AlbumArt"))
            {

                Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\Music Player\AlbumArt");
            }

            if
            (!File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) +
            @"\Music Player\Povezave_za_pesmi.sqlite"))// preverimo,če na tej lokaciji obstaja že
            kaka baza
            {
                new SQLite().naredi_bazo(); //ustvarimo podatkovno bazo
            }

            List<string> skladba = new List<string>(); // seznam v katerega vstavimo poti
            audio datotek,ki jih bomo predvajali
        }
    }
}
```

```

        Metapodatki metapodatki = new Metapodatki(); //inicializiramo nov objekt
        razreda Metapodatki
        SQLite poizvedba = new SQLite(); //inicializiramo nov objekt razreda poizvedba
        Predvajanje glasba = new Predvajanje(); // inicializiramo nov objekt razreda
        Predvajanje
        private string globalni_string = ""; //niz besedila
        bool predvajanje = false;
        bool ponovi = false;
        bool nakljucno = false;
        short klik = 0, sekunde = 0, index = 0, s_nakljucno=0, s_ponovi=0;

        //Gumbi
        private void skladbeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            odpri();
        }
        private void izhodToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            TimeSpan cas = TimeSpan.FromSeconds(trackBar1.Value);
            globalni_string = cas.ToString(@"hh\:mm\:ss");
            if (globalni_string!=dolzina.Text)
            {
                sekunde++;
                trackBar1.Value++;
            }
            else
            {
                glasba.ustavi();
                timer1.Stop();
                trackBar1.Value = 0;

                sekunde = 0;
                if (ponovi==true)
                {
                    predvajaj(skladba[index]);
                }
                else
                {
                    if (nakljucno == true)
                    {
                        index= Convert.ToInt16(new Random().Next(0,
listBox1.Items.Count));
                    }
                    else
                    {
                        if ((index + 1) == listBox1.Items.Count)
                        {
                            index = 0;
                        }
                        else
                        {
                            index++;
                        }
                    }
                    predvajaj(skladba[index]);
                }
            }
        }
    }

```



```

    }

    cas = TimeSpan.FromSeconds(sekunde);
    p_cas.Text = cas.ToString(@"hh\:mm\:ss");
}
private void button3_Click(object sender, EventArgs e)
{
    s_ponovi++;
    if (s_ponovi % 2 == 1)
    {
        ponovi = true;
        button3.Image = Resource1.ponovi_ne;
    }
    else
    {
        button3.Image = Resource1.ponovi;
        ponovi = false;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    Funkcije_gumbov("Nazaj");
}

private void button2_Click(object sender, EventArgs e)
{
    Funkcije_gumbov("Naprej");
}

private void button4_Click(object sender, EventArgs e)
{
    if (listBox1.Items.Count == 0)
    {
        odpri();
    }
    else
    {
        if (!predvajanje)
        {
            predvajaj(skladba[index]);
        }
        else
        {
            glasba.ustavi();
            timer1.Stop();
            predvajanje = false;
            button4.Image = Resource1.predvajaj;
        }
    }
}

private void oProgramuToolStripMenuItem_Click(object sender, EventArgs e)
{
    new AboutBox1().Show();
}

private void listBox1_MouseDoubleClick(object sender, MouseEventArgs e)
{
    trackBar1.Value = 0;
    Funkcije_gumbov("Listbox");
}

private void trackBar1_Scroll(object sender, EventArgs e)
{

```

```

        p_cas.Text = "";
        sekunde = Convert.ToInt16(trackBar1.Value);
        p_cas.Text = sekunde.ToString(@"hh\:mm\:ss");
        glasba.isci(sekunde*1000);
    }
    private void button5_Click(object sender, EventArgs e)
    {
        s_nakljucno++;
        if (s_nakljucno % 2 == 1)
        {
            nakljucno = true;
            button5.Image = Resource1.nakljucno_ne;
        }
        else
        {
            button5.Image = Resource1.nakljucno;
            nakljucno = false;
        }
    }

    //Funkcije
    private void odpri()
    {
        string[] a_datoteke;

        openFileDialog1.FileName = "";
        openFileDialog1.Filter = "MP3|*.mp3|WAV|*.wav|FLAC|*.flac|Vse
datoteke|*.*";
        openFileDialog1.Title = "Izberite več audio datotek (Vsaj 2 datoteki).";
        openFileDialog1.Multiselect = true;
        openFileDialog1.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyMusic);

        if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
        {
            MessageBox.Show("Izbiranje glasbe preklicano!", "Preklicano!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            if (listBox1.Items.Count == 0)
            {
                album.Text = izvajalec.Text = naslov.Text = "Glasba ni naložena!";
            }
        }
        else
        {
            try
            {
                a_datoteke = openFileDialog1.FileNames;
                skladba.AddRange(a_datoteke);
                predvajaj(skladba[0]);
                listBox1.Items.Clear();
                Napolni_lisbox(skladba);
                Array.Clear(a_datoteke, 0, a_datoteke.Length);

                klik++;
                trackBar1.Visible = true;
            }
        }
    }

```



```

        catch (Exception izjema)
        {
            MessageBox.Show("Izbrane datoteke ni mogoče odpreti" +
Environment.NewLine + "Razlog: " + izjema.ToString(), "Napaka!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    ///funkcija odpre eno ali več audio datotek

    private void metapodatek(string datoteka_za_metapodatke)
    {
        Bitmap slika;
        metapodatki.Meta_podatki = datoteka_za_metapodatke;
        datoteka_za_metapodatke = metapodatki.Meta_podatki;
        ushort id = 0;
        Label[] oznake = new Label[] { naslov, izvajalec, album, dolzina };

        foreach (Label oznaka in oznake)
        {
            oznaka.Text = datoteka_za_metapodatke.Split(',')[id];
            id++;
        }
        poizvedba.iskanje_vnosa = album.Text + "," + izvajalec.Text;

        if (poizvedba.iskanje_vnosa != "0")
        {
            slika = new
Bitmap(Image.FromFile(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)
+ @"\Music Player\AlbumArt\" + album.Text + " " + izvajalec.Text + ".jpg"), new
Size(120, 120));
            pictureBox1.Image = slika;
            MessageBox.Show("Dela");
        }
        else if (poizvedba.iskanje_vnosa == "0")
        {
            if (Preveri_povezavo() == true && album.Text != "Neznano" &&
izvajalec.Text != "Neznano")
            {
                metapodatki.Album_art = album.Text + "," + izvajalec.Text + "," +
naslov.Text;
                if (metapodatki.Album_art != "Privzeto")
                {
                    slika = new Bitmap(Image.FromFile(metapodatki.Album_art), new
Size(120, 120));
                    pictureBox1.Image = slika;
                }
                else
                {
                    error_file(skladba[index]);
                }
            }
            else
            {
                if (!Preveri_povezavo())
                {
                    MessageBox.Show("Slike albuma nisem uspel pridobiti, ker ni
povezave do interneta.\nPreverite dostop do interneta.", "Ni interneta",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }

```

```

        error_file(skladba[index]);
    }
}
}
private void predvajaj(string audio_file)
{
    trackBar1.Value = 0;
    sekunde = 0; //sekunde postavimo na nič
    p_cas.Text = "00:00:00"; // posravimo na nula

    predvajanje = true;
    if (!File.Exists(skladba[index]))
    {
        MessageBox.Show("Skladba s tem imenom, ne obstaja, preverite, če se  
datoteka nahaja na tem mestu, če ne ste jo morda izbrisali ", "Ne obstaja!");
        skladba.Remove(skladba[index]);
        Napolni_lisbox(skladba);

        if (index >= listBox1.Items.Count)
        {
            index--;
        }
        audio_file = skladba[index];
    }
    metapodatek(skladba[index]);
    int cas = Convert.ToInt16(TimeSpan.Parse(dolzina.Text).TotalSeconds);

    trackBar1.Maximum = cas;
    timer1.Start(); // začnemo s štetjem
    glasba.odpri_skladbo(skladba[index]);
    glasba.predvajaj();
    poizvedba.vnos_slike(izvajalec.Text + "," + album.Text + "," +
metapodatki.Album_art);
    button4.Image = Resource1.ustavi;
}

private bool Preveri_povezavo()
{
    try
    {
        using (var klient = new WebClient())
        {
            using (var probaj_odpreti = klient.OpenRead("https://google.com"))
            {
                return true;
            }
        }
    }
    catch
    {
        return false;
    }
}

private void Form1_Load_1(object sender, EventArgs e)
{
    if (listBox1.Items.Count < 2)
    {
        odpri();
    }
}

```





```

    }
}

private void Napolni_lisbox(List<string> seznam)
{
    listBox1.Items.Clear();
    foreach (string napolni_listbox in seznam)
    {
        metapodatki.Meta_podatki = napolni_listbox;
        globalni_string = metapodatki.Meta_podatki.Split(',')[0];
        listBox1.Items.Add(globalni_string);
    }
} //funkcija za fillanje lisbox-a

private void Funkcije_gumbov(string Funkcija)
{
    if (listBox1.Items.Count < 2)
    {
        odpri();
    }
    else
    {
        glasba.ustavi();
        if (Funkcija == "Naprej")
        {
            trackBar1.Value = 0;
            if (nakljucno==true)
            {
                index = Convert.ToInt16(new Random().Next(0,
listBox1.Items.Count));
            }
            else
            {
                if ((index + 1) == listBox1.Items.Count)
                {
                    index = 0;
                }
                else
                {
                    index++;
                }
            }
        }
        else if (Funkcija == "Listbox")
        {
            index = Convert.ToInt16(listBox1.SelectedIndex);
            if (index == -1)
            {
                index = 0;
            }
        }
        else if (Funkcija == "Nazaj")
        {
            if (nakljucno == true)
            {
                index = Convert.ToInt16(new Random().Next(0,
listBox1.Items.Count));
            }
            else
            {
                if ((index - 1) == -1)

```

```
        {
            index = Convert.ToInt16(listBox1.Items.Count - 1);
        }
        else
        {
            index--;
        }
    }
}
predvajaj(skladba[index]);
predvajanje = true;
}
}
private void error_file(string vrsta)
{
    if (vrsta.Contains(".wav"))
    {
        pictureBox1.Image = Resource1.wav;
    }
    else if (vrsta.Contains(".mp3"))
    {
        pictureBox1.Image = Resource1.mp3;
    }
    else if (vrsta.Contains(".flac"))
    {
        pictureBox1.Image = Resource1.flac;
    }
}
}
}
```



## *Metapodatki.cs (Razred za obdelavo metapodatkov)*

```
using System;
using System.IO;
using System.Linq;
using ParkSquare.Gracenote;

namespace Predvajalnik_glasbe_v_CSharpu
{
    class Metapodatki
    {
        //SPREMENLJIVKE
        private string audio_file;
        //LASTNOSTI
        public string Meta_podatki
        {
            get { return audio_file; }
            set { audio_file = metapodatki(value); }
        }
        public string Album_art
        {
            get { return audio_file; }
            set { audio_file = Prenos_slike(value); }
        }
        //METODE
        private string metapodatki(string datoteka)
        {
            TagLib.File a_dat = TagLib.File.Create(datoteka);

            if (a_dat.Tag.Title != null)
            {
                datoteka = a_dat.Tag.Title + ",";
            }
            if (datoteka.Contains(".wav") || a_dat.Tag.Title == null)
            {
                datoteka = trim_albuma(Path.GetFileName(datoteka), "pot")+",";
            }

            if (a_dat.Tag.FirstPerformer != null)
            {
                datoteka += a_dat.Tag.FirstPerformer + ",";
            }
            else
            {
                datoteka += "Neznano,";
            }
            if (a_dat.Tag.Album != null)
            {
                datoteka += trim_albuma(a_dat.Tag.Album, "album") + ",";
            }
            else
            {
                datoteka += "Neznano,";
            }
            if (a_dat.Properties.Duration.ToString(@"hh\:mm\:ss") != null)
            {
                datoteka += a_dat.Properties.Duration.ToString(@"hh\:mm\:ss");
            }
        }
    }
}
```

```

        return datoteka;
    } //funkcija za vračanje metapodatkov
    private string trim_albuma(string trimm, string type)
    {
        if (trimm.Contains("["))
        {
            trimm = trimm.Remove(trimm.IndexOf('['), (trimm.IndexOf(']') - trimm.IndexOf('[') + 1));
        }
        if (trimm.Contains("("))
        {
            trimm = trimm.Remove(trimm.IndexOf('('), (trimm.IndexOf(')') - trimm.IndexOf('(') + 1));
        }
        if (trimm.Contains("Disc"))
        {
            trimm = trimm.Remove(trimm.Length - 7);
        }
        if (trimm.Contains("CD"))
        {
            trimm = trimm.Remove(trimm.Length - 5);
        }
        if (trimm.Contains(".wav") || trimm.Contains(".mp3"))
        {
            trimm = trimm.Remove(trimm.IndexOf('.'), 4);
        }
        if (trimm.Contains(',') && type != "pot")
        {
            trimm = trimm.Replace(",", "");
        }
        return trimm;
    } //funkcija za odstranjevanje znakov
    private string Prenos_slike(string datoteka)
    {
        string album_meta = datoteka.Split(',')[0];
        string izvajalec_meta = datoteka.Split(',')[1];
        string ime_skladbe_meta = datoteka.Split(',')[2];
        string pot =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\Music
Player\AlbumArt\" + album_meta + " " + izvajalec_meta + ".jpg";

        if (!File.Exists(pot))
        {
            try
            {
                var odjemalec = new GracenoteClient("962650182-
16615324626BA4A3EC0A5EADD71428E5");
                var Slika = odjemalec.Search(new SearchCriteria
                {
                    TrackTitle = ime_skladbe_meta,
                    AlbumTitle = album_meta,
                    Artist = izvajalec_meta,
                    SearchMode = SearchMode.BestMatchWithCoverArt,
                    SearchOptions = SearchOptions.ArtistImage
                });
                Slika.Albums.First().Artwork.First().Download(pot); //0 album, 1
artist, 3 je lokacija
            }
        }
    }

```



```
        catch
        {
            pot = "Privzeto";
        }
    }

    return pot;
} //funkcija za prenos slike albuma
}
```

## SQLite.cs (Razred za delo s podatkovno bazo)

```
using System;
using System.Data.SQLite;

namespace Predvajalnik_glasbe_v_CSharpu
{
    class SQLite
    {
        SQLiteConnection povezava_z_bazo;
        //SPREMENJLJIVKE
        private string sql;
        private string vrstica;

        //OBJEKTI

        //LASTNOSTI
        public string iskanje_vnosa
        {
            get { return vrstica; }
            set { vrstica = iskanje_vnosa_in_izpis(value); }
        } //za vpis izvajalca, albuma in linka slike

        //METODE
        private void povezi()
        {
            povezava_z_bazo = new SQLiteConnection("Data Source=" +
            Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + @"\Music
            Player\Povezave_za_pesmi.sqlite" + "; Version=3;"); // ustvarimo povezavo z bazo
            povezava_z_bazo.Open();
        }
        public void naredi_bazo()
        {
            SQLiteConnection.CreateFile(Environment.GetFolderPath(Environment.SpecialFolder.MyDocu
            ments) + @"\Music Player\Povezave_za_pesmi.sqlite");

            povezi();
            sql = "CREATE TABLE slike_albuma (izvajalec VARCHAR(50) NOT NULL, album
            VARCHAR(50),slika VARCHAR(75))";

            SQLiteCommand kreiraj = new SQLiteCommand(sql, povezava_z_bazo);

            kreiraj.ExecuteNonQuery();
            povezava_z_bazo.Close();
        } //naredimo novo podatkovno bazo
        public void vnos_slike(string vnos_parametrov)
        {
            povezi();

            sql = "INSERT INTO slike_albuma (izvajalec, album, slika) VALUES ('" +
            vnos_parametrov.Split(',')[0] + "','" + vnos_parametrov.Split(',')[1] + "','" +
            vnos_parametrov.Split(',')[2] + "')";
            try
            {
                SQLiteCommand vnos = new SQLiteCommand(sql, povezava_z_bazo);
                vnos.ExecuteNonQuery();
                povezava_z_bazo.Close();
            }
        }
    }
}
```



```

    }
    catch
    {

    }

} //vnos slike v podatkovno bazo

private string iskanje_vnosa_in_izpis(string album_izvajalec)
{
    povezi();

    album_izvajalec = "0";
    try
    {
        sql = "SELECT count(slika) FROM slike_albuma WHERE album='" +
album_izvajalec.Split(',')[0] + "' and izvajalec='" + album_izvajalec.Split(',')[1] +
""";
        SQLiteCommand iskanje = new SQLiteCommand(sql, povezava_z_bazo);
        if ((short)iskanje.ExecuteScalar() == 1)
        {
            try
            {
                sql = "SELECT slika FROM slike_albuma WHERE album='" +
album_izvajalec.Split(',')[1] + "' and izvajalec='" + album_izvajalec.Split(',')[0] +
""";
                iskanje = new SQLiteCommand(sql, povezava_z_bazo);
                album_izvajalec = (string)iskanje.ExecuteScalar();
            }
            catch
            {
                album_izvajalec = "0";
            }
        }
        else
        {
            album_izvajalec = "0";
        }
    }
    catch
    {
        album_izvajalec = "0";
    }
    finally
    {
        povezava_z_bazo.Close();
    }
    return album_izvajalec;
} //funkcija preveri, če je v bazi link od slike in če ja ga vrne
}

```

## *Predvajanje.cs (Razred za predvajanje glasbe)*

```
using System.Runtime.InteropServices;
using System.Text;

namespace Predvajalnik_glasbe_v_CSharpu
{
    class Predvajanje
    {
        private string ukaz; //command
        [DllImport("winmm.dll")] // v program vključimo DLL za delo z multimedijo
        private static extern long mciSendString(string lpstrCommand, StringBuilder
lpstrReturnString, int uReturnLength, int hwdCallback); //funkcija za delo multimedijo
        //sklicuje se na winmm.dll

        public void odpri_skladbo(string datoteka)
        {
            ukaz = "open \"" + datoteka + "\" type MPEGVideo alias MUSIC";
            mciSendString(ukaz, null, 0, 0);
        } //funkcija odpre skladbo za predvajanje
        public void predvajaj()
        {
            ukaz = "play MUSIC";
            mciSendString(ukaz, null, 0, 0);
        } //funkcija začne predvajati skladbo
        public void ustavi()
        {
            ukaz = "stop MUSIC";
            mciSendString(ukaz, null, 0, 0);
            ukaz = "close MUSIC";
            mciSendString(ukaz, null, 0, 0);
        } //funkcija ustavi predvajanje skladbe
        public void ponovi()
        {
            ukaz = "seek MUSIC to start";
            mciSendString(ukaz, null, 0, 0);
            predvajaj();
        }
        public void isci(int cas)
        {
            ukaz = "seek MUSIC to " + cas;
            mciSendString(ukaz, null, 0, 0);
            predvajaj();
        }
    }
}
```





## *Zahvala mentorju in ostalim*

Za zaključek bi se rad zahvalil vsem, ki so mi pri izdelavi programa stali ob strani. Zahvala je namenjena predvsem mentorju Igorju Kutošu, ki me je usmerjal na poti pri izdelavi programa za predvajanje glasbe v programskem jeziku C#, pa tudi staršem, ki so mi stali ob strani in mi kuhali kavo, da sem lažje prestajal, noči v katerih sem svoje zbrano štiriletno znanje prelil v programsko kodo, ki mi bo služila kot referenca za mojo nadaljnjo kariero prav tako, pa se bo v delu opazilo znanje, ki so mi ga posredovali odlični profesorji.

Aleksander Kovač<sup>4R1</sup>