Grafičko oblikovanje sučelja u programskom jeziku Visual Basic

Rajič, Blaž

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Blaž Rajič

Grafičko oblikovanje sučelja u programskom jeziku Visual Basic

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Blaž Rajič

Matični broj: 0016130409

Studij: Informacijski sustavi

Grafičko oblikovanje sučelja u programskom jeziku Visual Basic

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Mario Konecki

Varaždin, kolovoz 2020.

Blaž Rajič

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome radu baviti ću se analizom programskog jezika Visual Basic.NET te koje su nam sve mogućnosti za izradu grafičkih programa unutar programa Visual Studio. U početku opisati ću povijest iza Visual Basic programskog jezika, koji su njegovi početci te njegove inačice kroz povijest. Usporediti ću razlike između klasičnog Visual Basic programa i Visual Basic.NET programskog jezika te nakon toga unutar Visual Studio programa prikazati ću koje sve mogućnosti su nam dostupne kada programiramo u Visual Basic.NET jeziku. Objasniti ću koje su prednosti svake metode i njezine nedostatke, te za što se svaka metoda koristi u današnjem svijetu. Za kraj opisati ću praktični dio ovog rada te ukratko opisati njegovu funkcionalnost.

Ključne riječi: Visual Basic, Visual Basic.NET, .NET, grafičko sučelje, Visual Studio

Sadržaj

1. Uvod

Tema ovog rada je izrada grafičkog sučelja u programskom jeziku Visual Basic. Prema definiciji pojam sučelje je općenito skup sredstava koja omogućuje interakciju između korisnika i nekakvog mehaničkog stroja ili drugog složenog sustava.[1] Pojam grafičko sučelje bi mogli opisati onda kao vrsta sučelja koja koristi grafičke elemente koji olakšavaju interakciju između korisnika i računala kako bi ona bila što intuitivnija.

Visual Basic kao programski jezik ima bogatu povijest, kao programski jezik iskazao se je kao vrlo sposobnim pogotovo u funkciji izrade grafičkog sučelja. Izrada grafičkih sučelja je bila jednostavna te nije bilo potrebno puno programskog koda kako bi se napravile određene osnovne funkcije za rad programa. Njegova jednostavnost privukla je mnogo novih programera koji su sa lakoćom mogli napraviti programe, ali nisu samo novi nego i stariji programeri su također koristili baš zbog njegove lakoće korištenja. Prenošenjem unutar .NET frameworka Visual Basic je dobio nove mogućnosti koje programer može iskoristiti za razvoj novih i modernih programa.

Prije pisanja ovog rada nisam znao puno o Visual Basic programskom jeziku, htio sam naučiti više o njemu pošto sam čuo od drugih starijih programera kako su oni naučili programirati baš na Visual Basic jeziku. I dok u Visual Studio programu jesam radio imao sam želju proširiti svoje znanje o drugim načinima izrade grafičkih sučelja jer svaka metoda ima svoje upotrebe u današnjem svijetu informatike.

2. Metode i tehnike rada

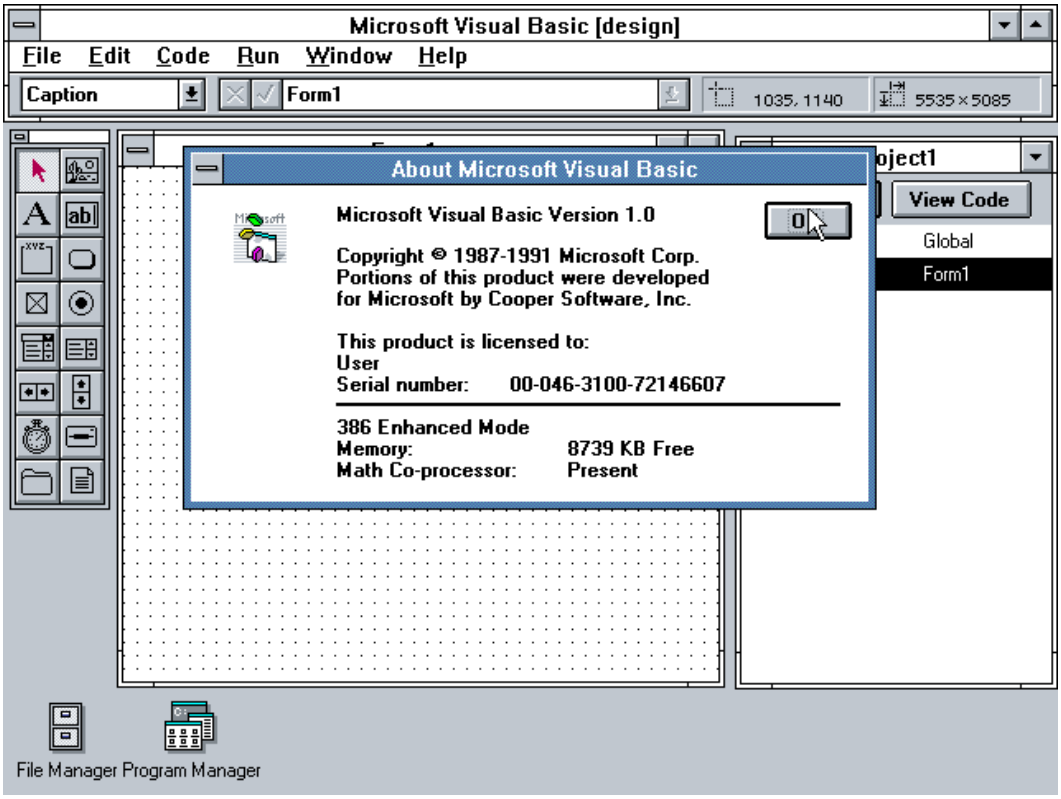
Pošto u ovom radu koristim Visual Basic programski jezik, odlučio sam prvo opisati sam programski jezik. Krenuo bih sa njegovom povijesti, zašto je nastao, kakva je bila njegova uloga i je li bio uspješan u tome. Nastavio bih zašto je Visual Basic postao popularan, što se je svidjelo programerima kod njega te koje funkcionalnosti je Visual Basic nudio a da ih nema kod drugih programskih jezika. Za kraj ovog dijela planirao sam opisati ulogu Microsofta i koje su strategije bile sa Visual Basic jezikom i kako ga je odlučio izbaciti na tržište.

Nakon uvoda u sam programski jezik, planirao sam opisati koje su mogućnosti izrade grafičkih sučelja u Visual Basic.NET programu u današnjem vremenu na današnjem tržištu. Pošto postoji više odabira za grafička sučelja prošao bih kroz najvažnija i objasnio prednosti i nedostatke te koja je najbolja funkcija za takvo sučelje.

Za izradu ovog rada koristio sam Visual Studio Community 2019 te sam programirao u Visual Basic.NET i XAML programskom jeziku.

3. Povijest Visual Basic programskog jezika

Prije ulaska u samu temu rada, htio bi proći kroz povijest programskog jezika Visual Basic i sve njegove inačice. Alan Cooper, također znan kao „otac“ Visual Basic-a, izrađuje *drag-and-drop* prototip zvan Tripod te ga prezentira vlasniku Microsofta Billu Gatesu. Microsoft kupuje koncept te ga naziva Ruby(nema veze sa programskim jezikom Ruby). Ruby nije imao nikakav programski jezik, te je Microsoft odlučio spojiti koncept zajedno sa programskim jezikom QuickBasic čime je nastao Visual Basic 1.0 u 1991 godini. [2]



Slika 1: Visual Basic 1.0 [3]

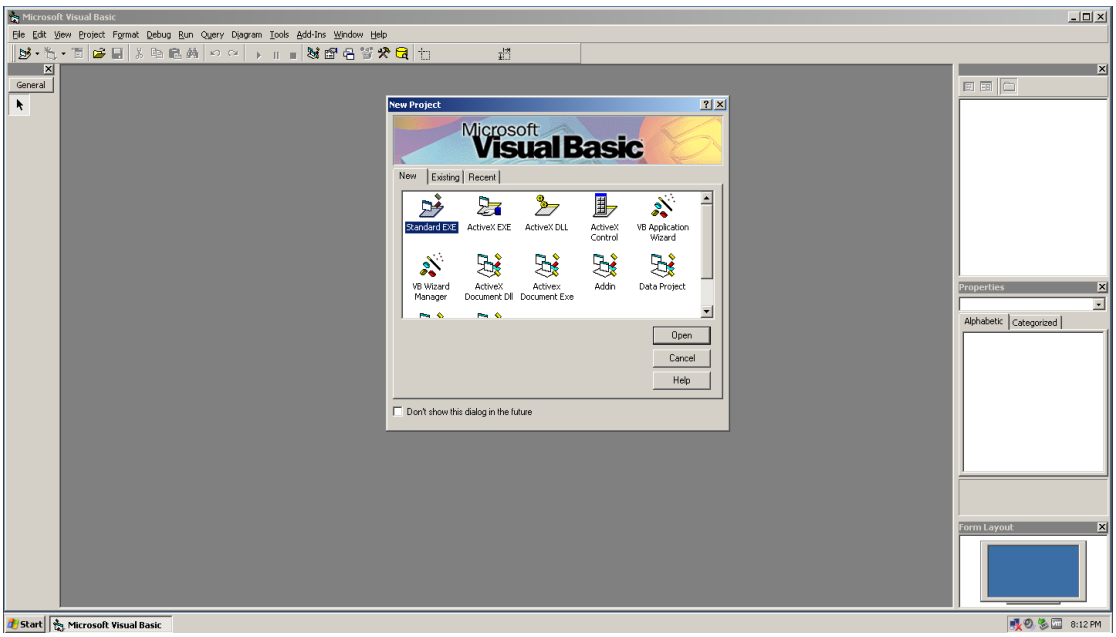
Godinu kasnije u 1992. godini Microsoft izbacuje na tržište Visual Basic 2.0 Toolkit, ovakav program je integrirao nekoliko alata sa treće strane(eng. *third-party tools*) u jedan konkretan paket koji je omogućio veliku kontrolu korisnicima. Ovakav program se je pokazao važnim čimbenikom u povećavanju sveukupne podrške Visual Basic-a od krajnjih korisnika.[2]

Iste godine izbačen je i Visual Basic 2.0, osim nekih općenitih poboljšanja i optimizacija, dodane su određene nove funkcionalnosti kao MDI obrasci i objektne varijable. Također ovo je i prvi Visual Basic koji nudi i Professional inačicu programa. Spomenuo bih i Microsoft Access koji je u isto vrijeme izbačen. Access koristi dijelove iz programskog koda

VB 2.0 te je iskoristio jednostavnost korištenja Visual Basica i prenio na relacijske baze podataka.[2]

U sljedećih 5 godina, Microsoft je izbacio 3 inačice Visual Basica, u međuvremenu dodane su nove funkcionalnosti kao VBScript, programski jezik dizajniran za izradu web aplikacija unutar Microsoftovog web preglednika Internet Explorer. Dodana je mogućnost za izradu 32-bitnih programa umjesto za dosadašnjih 16-bitnih programa a dodana je i mogućnost izrade ActiveX kontrola.[2]

Inačica Visual Basic 6 je izašla 1998. godine, do tada programski jezik je postao vrlo popularan i procjenjuje se kako je bilo deset puta više programera koji su koristili Visual Basic nego drugi programski jezik C++. Do tada VB se je dokazao kao valjan programski jezik koji je mogao zadovoljiti sve potrebe individualnih programera i poslovnih korisnika.[4]



Slika 2: Visual Basic 6 [4]

U 2002. godini, Microsoft je htio uvesti .NET tehnologiju, to je bila skroz nova programska podrška te kako bi Visual Basic bio kompatibilan sa njom morao je proći kroz velike preinake. Krajnji rezultat je novi takozvani Visual Basic.NET koji nije više bio kompatibilan sa prošlim inačicama Visual Basica. Sve do danas Microsoft je postepeno održavao VB.NET i dodavao nove funkcionalnosti unutar .NET frameworka.[5]

4. Općenito o programskom jeziku

4.1 Klasični Visual Basic

Zajedno sa operacijskim sustavom Windows, Microsoft se je prebacio u svijet kliktanja mišem te shodno tome trebao je programski jezik čija je glavna svrha bila iskorištavanje punog potencijala upravo toga. Kada je Visual Basic došao na scenu bio je nešto novo i drugačije od ostalih programskih jezika, nije više bilo potrebno sjediti za velikom količinom kriptičnog programskog koda nego je bilo moguće nacrtati grafičke elemente koje je korisnik htio i kasnije ih sve povezati programskim kodom.[5]

Visual Basic je također općenito vrlo praktičan, jako lagano i brzo se je moglo napraviti cijelo korisničko sučelje u odnosu druge programske jezike ali ono što je dodatno istaknulo VB je bila takozvana funkcionalnost *edit-and-continue*. Takva funkcionalnost dozvoljavala je korisnicima da kada pokrenu svoj program te primijete neku pogrešku, moguće je pauzirati izvođenje, promijeniti kod programa te nakon toga samo nastaviti izvođenje. Ova funkcionalnost je smanjila sveukupno potrebno vrijeme programiranja jer ako pogledamo druge programske jezike, kada dođe do mijenjanja programskog koda, potrebno je ponovno kompilirati cijeli kod što troši dodatno vrijeme.[5]

Kako neki kažu, Visual Basic je bio jedan od najlakših, najbržih i ujedno i najzabavnijih načina za izradu aplikacija u Windowsima 95,98 i NT. Naravno, uspješnost Visual Basica nije prošla neprimijećeno te je konkurencija pokušala implementirati funkcionalnosti VB-a u svoje programske jezike. Takvi jezici zajedno sa VB se nazivaju RAD, skraćeno za *rapid application development* zbog njihove lakoće izrada aplikacija.[5]

Osnovni koncept programskog jezika Visual Basic je takozvano *event-driven* programiranje, sa takvim konceptom programiranja u aplikaciji sve što radimo su objekti(eng. *object*). Svaki objekt ima svoja određena svojstva(eng. *properties*) koja određuju kako će taj objekt izgledati i ponašati. Svaki objekt ima svojstvo odgovarati na specifične događaje(eng. *events*) koji se događaju na tom specifičnom objektu. Postoji mnogo vrsta događaja i svaki objekt je dizajniran da može odgovarati na događaje koji su njemu potrebni. Objekti također podržavaju i metode(eng. *methods*), metode su nešto što definira kako objekt odrađuje neki zadatak, to za primjer može biti objekt za povezivanje na bazu podataka te on mora imati određenu funkciju koja inicijalno stvara vezu na željenu bazu.[6]

Postojale su razne vrste klasičnog Visual Basica, Microsoft je svaku vrstu prodavao određenim vrstama klijenata ovisno o njihovim potrebama, a razlikujemo ove vrste:[6]

- **Control Creation Edition:** Ova vrsta bila je orijentirana brzom i laganom razvoju ActiveX kontrola, ActiveX je bio set tehnologija koje integriraju programsku podršku u web okruženje, omogućavajući laganu izradu interaktivnih aplikacija i web stranica. Zadnja inačica ovakve vrste je došla zajedno sa Visual Basic 5.0
- **Learning Edition:** Kako i sam naslov kaže ova vrsta je bila namijenjena početnicima, osim samog programa korisnici su dobili i isprintanu dokumentaciju za korištenje programa.
- **Working Model Edition:** Ova vrsta je bila ciljana prema korisnicima koji nisu htjeli puno platiti za ostale vrste programa, WME nudi ista okruženja za razvoj programa kao i learning edition. Određeni nedostaci ove vrste su bili nemogućnost izrade Web stranica te ActiveX kontrola.
- **Professional Edition:** Ova vrsta je sadržavala sve što je u Learning Edition, uz dodatak određenih kontrola te dodatne alate za Web i baze podataka.
- **Enterprise Edition:** Najbolja i najveća vrsta što se tiče tehnologija koje podržava, ujedno i najskuplja u odnosu na druge vrste.

Slika 3: Primjer vrste Visual Basic 6.0 Learning Editon[7]

Osim samih vrsta Visual Basica, Microsoft je napravio i određene programe koji su bazirani na VB tehnologiji ali služe za drugačije svrhe. Cilj ovoga je bila ideja da kada korisnik nauči VB programski jezik, sa lakoćom se može prebaciti i na druge tehnologije koje mu proširuju opseg svega što sve može napraviti. Razlikujemo: [6]

- **Visual Basic Scripting Editon:** Također zvan i VBScript, ovaj programski jezik bio je namijenjen pisanju skripti koje pružaju interaktivnost Web stranicama. VBScript bi se

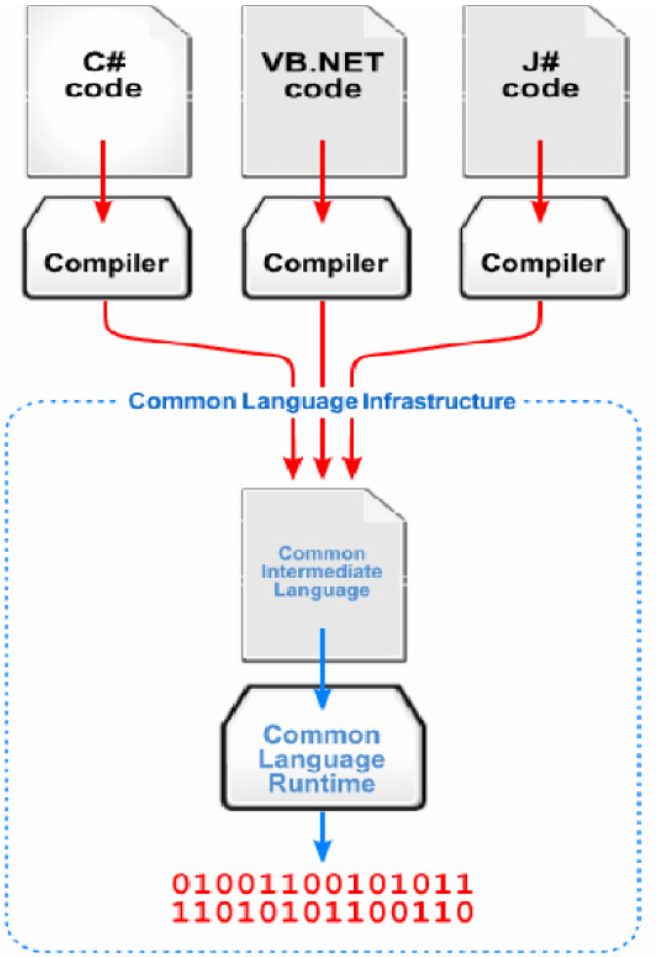
mogao povezati zajedno sa HTML kodom te bi pružio funkcionalnosti koje ne bi bile moguće sa samim HTML kodom.

- **Visual Basic Applications Editon(VBA):** Ovaj program bio je namijenjen pisanju ekstenzija za druge programe koje bi dodale nove funkcionalnosti ovisno o korisnikovim potrebama. Inače se je koristio za pisanje ekstenzija za druge Microsoftove proizvode kao Microsoft Word i Excel.
- **Windows CE Toolkit for Visual Basic:** Ideja iza ovog programa bila je mogućnost pisanja aplikacija za Windows CE, posebna inačica Windowsa dizajnirana za male prenosive uređaje.

I dok je klasični Visual Basic bio veliki uspjeh, imao je par nedostataka. Bio je sporiji u izvođenju u odnosu na neke druge programske jezike te VB nije bio najefikasniji za programe koji bi koristili puno računalnih resursa kao na primjer video igrice. Dodatno još što se nekim programerima također nije svidjelo je nedostatak podrške za nasljeđivanje(eng. inheritance).[5] Za kraj važno je napomenuti pošto je VB razvijen od tvrtke Microsoft programi koji su napisani u njemu će raditi samo na Windows operacijskim sustavima.[8]

4.2 Visual Basic.NET

U 2002. godini Microsoft je uveo .NET framework, to je bila nova iteracija Microsoftove platforme za razvoj aplikacija. Jedan od novih implementacija je mehanizam za učitavanje, pokretanje programa i upravljanje njihovih interakcija. Novi mehanizam zove se CLI, skraćeno od *Common Language Infrastructure,* CLI je omogućavao programima razmjenu podataka međusobno bez obzira u kojem su programskom jeziku napisane te je omogućavao izvršavanje programskog koda na drugim operacijskim sustavima bez re-kompilacije programskog koda.[9]



Slika 4: Prikaz CLI infrastrukture[10]

CLI je doduše samo specifikacija, te ju je potrebno implementirati na određeni način. Način na koji je implementirana takva specifikacija nazivamo *Common Language Runtime* ili skraćeno CLR. Ako neki jezični prevoditelj(eng. *compiler*) radi u skladu sa CLR, on prevodi izvorni kod u CIL, skraćeno od *Common Intermediate Language*. CIL je strojni jezik koji nije vezan ni za jedan specifični operacijski sustav, jer prema CLI specifikaciji, svaki stroj koji implementira CLR bi trebao biti u stanju interpretirati i prevesti CLI.[9]

Zajedno sa izlaskom .NET tehnologije, Microsoft je htio prilagoditi Visual Basic kako bi ju mogao podržati. Microsoft je morao napraviti drastične promjene, te jedna od glavnih razlika između novog i starog jezika je bila uvođenje kompletnog koncepta objektno orijentiranog programiranja. Sada svaka komponenta koja radi sa sustavom se naziva objekt(eng. *object*) dok je svaki objekt nastao iz određene klase. Klase mogu biti deklarirane od strane korisnika ili automatski kroz sami programski kod. [11]

Kako je došlo do velikih promjena u samom programskom jeziku, Visual Basic.Net više nije kompatibilan sa prošlim inačicama VB-a, te dok je to bila nužna promjena kako bi se sve moglo implementirati u .Net framework, bile su određene kritike od strane korisnika. Jedna od

kritika je bila kako više kompatibilnosti nije bilo između klasičnog VB i VB.Net, Microsoft se nije previše potrudio o prebacivanju sa stare inačice na novu, rezultirajući kako su sada programeri i programi starog VB-a u svome izoliranom svijetu.[5]

4.3 Razlike između klasičnog VB i VB.Net

I dok možemo reći kako su Visual Basic i Visual Basic.NET slični jezici, postoje velike razlike. Visual Basic kao stariji jezik ne podržava novije koncepte programiranja koji su došli sa vremenom dok Visual Basic.NET kao noviji jezik uveo takozvane koncepte objektno orijentiranog programiranja. Ostale važnije razlike možemo pronaći i u tablici.

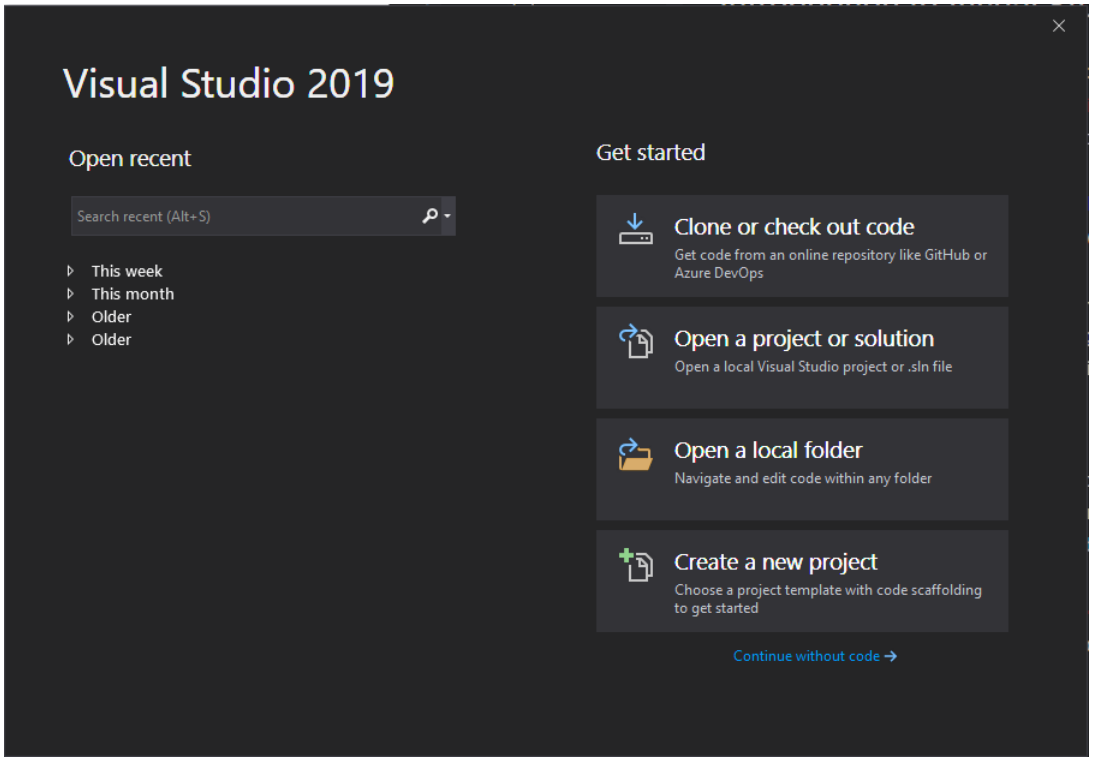
Tablica 1: Razlike između Visual Basic i Visual Basic.NET[12]

Visual Basic .Net	**Visual Basic**
Moderan, u potpunosti objektno orijentiran jezik koji je zamijenio VB6	Starija inačica koja nije u potpunosti objektno orijentirana te nije više održavana od strane Microsofta
Koristi *Common Language Runtime* (CLR) komponentu iz .Net okruženja, nudi više mogućnosti u odnosu na VB-Runtime	Koristi vlastito VB-Runtime okruženje
Kompilatorski programski jezik	Interpreterski programski jezik
Ne podržava prošle inačice VB-a	Podržava starije inačice VB-a
Type-safe programski jezik	Nije *type-safe* programski jezik
Podaci se obrađuju preko ADO.Net protokola	Podaci se mogu obrađivati kroz DAO, RDO i ADO*(ActiveX Data Object*) protokol
Podržava višedretvenost	Ne podržava višedretvenost

(Izvor: javaTpoint, „Difference Between VB.NET and Visual Basic“)

5.Vrste grafičkih sučelja unutar programskog jezika Visual Basic.NET

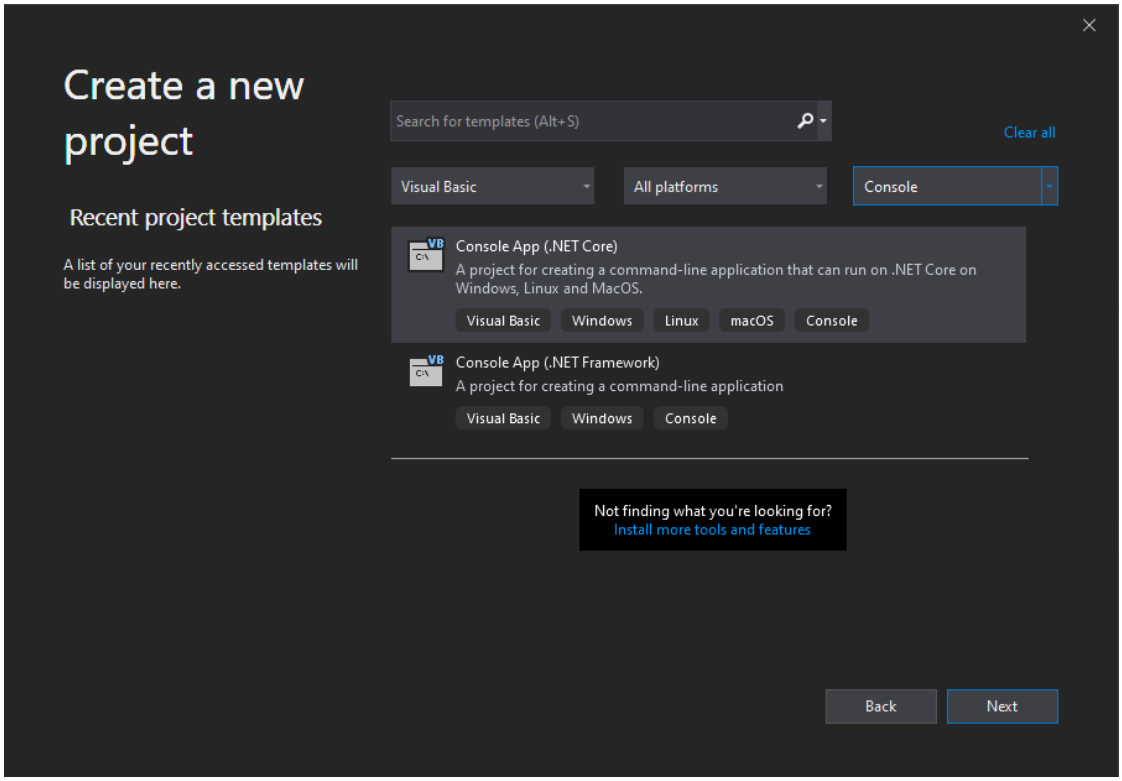
Kako bi napravio grafička sučelja, koristio sam Visual Studio program od tvrtke Microsoft. Visual Studio je IDE(eng. *Integrated Development Environment)* koji nam služi za razvoj programa, web aplikacija i servisa, mobilnih aplikacija te cloud servisa. Jedna od mogućnosti Visual Studia je i razvoj grafičkih korisničkih sučelja(eng. *grahical user interface*) što će nam biti potrebno za razvoj programa za ovaj rad. Visual Studio nam nudi mogućnosti za razvoj programa koristeći Visual Basic.NET a u nastavku ću navesti važnije i objasniti koje su prednosti svake i zašto bi koristili baš tu metodu za naše potrebe. Također sa strane budem i objasnio osnovne funkcionalnosti samog programskog jezika Visual Basic.NET koje nam trebaju kako bi ostvarili tražene funkcionalnosti.



Slika 5. Početni zaslon unutar Visual Studio 2019 (autorski rad)

5.1 Console aplikacije

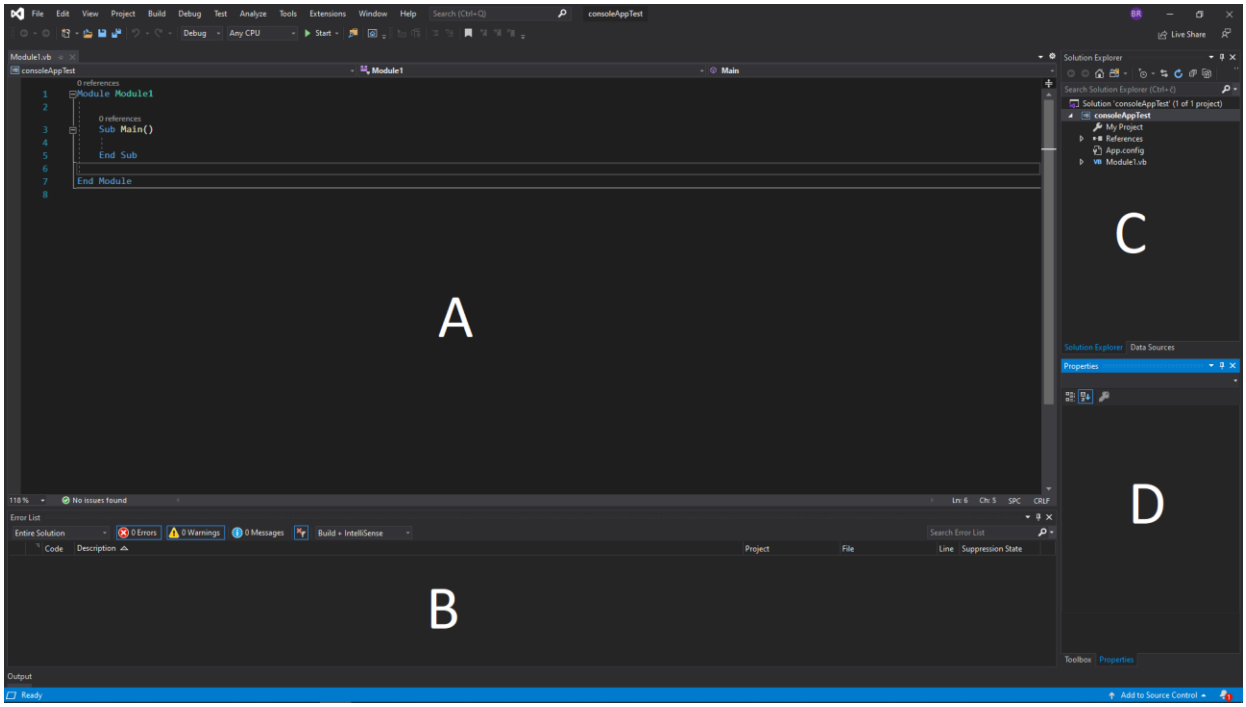
Konzolna aplikacija je prvo rješenje koje obrađujemo, kada odabiremo novi projekt unutar Visual Studia odabiremo *Console Application* za tip projekta, kao što na slici 6 vidimo ponuđena su nam dva tipa konzolne aplikacije: .NET Core i .NET Framework.



Slika 6. Odabir konzolne aplikacije(autorski rad)

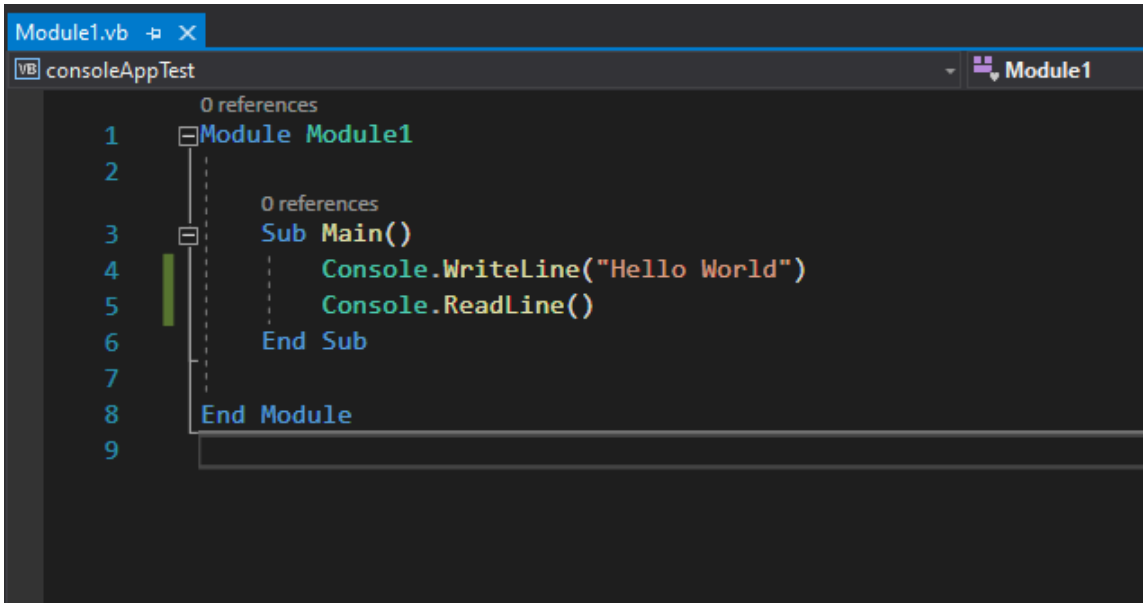
Hoćemo li izabrati .NET Core ili .NET Framework ovisi o našim potrebama i za što radimo našu aplikaciju. Prva implementacija .NET specifikacije je bila .NET framework koji radi samo na Windows operacijskom sustavu. Izvorni kod je vlasništvo Microsofta te nisu dopušteni doprinosi od strana korisnika. Ovaj framework doduše nudi mnoštvo mogućnosti za izradu desktop aplikacija te postoji mnogo vanjskih biblioteka ako su potrebne korisnicima. Sa druge strane .NET Core je noviji framework također razvijen od tvrtke Microsoft, jedna od većih razlika između je što je .NET Core podržan od strane drugih operacijskih sustava kao što su Linux i macOS. Druga razlika je kako je .NET Core *open-source*, gdje Microsoft prihvaća doprinose od korisnika. Dodatno dok se broj biblioteka se povećava sa vremenom, broj je i dalje manji u odnosu na .NET framework.[13]

Za našu konzolnu aplikaciju koristiti ćemo .NET framework, kada odredimo potrebne informacije kao naziv projekta i lokaciju gdje će nam se spremiti projekt otvara nam se glavni prozor za uređivanje naše aplikacije prikazane na slici 7.



Slika 7. Glavni prozor Visual Studio programa za konzolne aplikacije(autorski rad)

Kao što primjećujemo sveukupni prozor je podijeljen na više manjih prozora. Ako gledamo sliku 7, primijetiti ćemo kako prozor A zauzima najviše mjesta, to je takozvani *code editor* u kojemu pišemo svoj programski kod. Ispod njega trenutno se nalazi se error list koji nas obavještava o potencijalnim greškama i problemima. U desnom stupcu nalaze se još dva prozora, unutar prozora C nalazi se *solution explorer* koji pokazuje podatke na kojima korisnik trenutno radi. Zadnji prozor je prozor D koji nam pokazuje postavke za određenu stavku na kojoj trenutno radimo. Također važno je napomenuti kako raspored i koji prozori su prisutni je skroz opcionalno i korisnik može složiti raspored kako njemu odgovara. Započeti ćemo sa osnovnim primjerom za početak, cilj nam je ispisati Hello World korisniku. Na slici 8 je sav potreban programski kod za navedeni zadatak te rezultat koji smo dobili:



Slika 8 i 9: Programski kod za ispis Hello World te rezultat(autorski rad)

Unutar Visual Basic.Net programskog koda, postoji određen redoslijed naredbi koji trebamo poštivati kako bi program uspješno radio. Na početku započinjemo sa `Option` naredbama ako ih imamo, takve naredbe nam definiraju određena pravila za nadolazeći programski kod. Par primjera `Option` naredbi imamo `Option Explicit` naredbu koja programu govori jesu li sve varijable pravilno deklarirane i gramatički točne, uključivanjem ove naredbe smanjujemo potrošeno vrijeme u debuggingu. Drugi primjer ovog tipa naredbe je `Option Compare` naredba, koja definira način na koji Visual Basic uspoređuje varijable tipa string.[14]

Nakon `Option` naredbi dolaze `Imports` naredbe koje također su dodatne, neki programi ih ne trebaju imati kojima ne treba. Ukratko Imports naredba omogućuje našem programu „uvoz“ drugih klasa i tipova ako nam je to potrebno za određene funkcionalnosti koje nam trebaju.[14]

Nakon ova dva tipa dolaze `Namespace` naredbe te svi njihovi elementi. Ukratko, namespace bi mogli opisati kao organizacijski sistem koji nam olakšava prezentaciju

programskih elemenata unutar tog namespacea drugim programima. Nas trenutno ne zanima sama `Namespace` naredba nego svi elementi koji se nalaze na njezinoj razini a oni su:[14]

- **Class**: `Class` naredba definira novi tip podatka, klasa može sadržavati varijable, svojstva, procedure i događaje u određenom objektu. Objekt je instanca neke klase te može biti neograničeno mnogo objekata jedne klasa ovisno koliko je potrebno korisniku.
- **Structure**: `Structure` naredbu koristimo za stvaranje struktura, strukture su naziv za složeni tip podatka koji u sebi sadrži druge tipove podataka. Strukture su slične klasama ali postoje razlike, neke od njih su kako je struktura *value* tip dok su klase *reference* tip podataka te strukture ne podržavaju nasljeđivanje.
- **Module**: Module naredbu koristimo za pravljenje referentnog tipa podataka, ne možemo odrediti da neki element bude tipa module nego možemo ga samo referencirati. Modul kao i struktura je sličan klasi, ali sa bitnim razlikama. Glavna očita razlika je što modul ne možemo odrediti nekoj varijabli te modul ne podržava nasljeđivanje ni implementaciju sučelja.
- **Interface**: Interface ili sučelja možemo opisati kao definirani skup svojstva(eng. properties) i procedura koje klase ili strukture mogu implementirati. Valja naglasiti kako sučelja ne definiraju unutrašnji mehanizam navedenih svojstva i procedura, to je zadaća klase ili strukture koja je implementira.

Svaki gore navedeni tip sadrži dodatne elemente koji sadrže programski kod, koji element ćemo koristiti ovisi o našim potrebama, dolje navedeni neće biti svi ali neki od bitnijih elemenata:[14]

- **Function**: `Function` naredbu koristimo ukoliko bismo htjeli napraviti dio programskog koda koji bi odradio određenu funkciju te kada je gotov vratio rezultat gdjegod je ta procedura bila pozvana. Moguće je i postaviti ulazne podatke za funkciju ako je potrebno.
- **Sub**: Po načinu rada je isto kao `Function` ali `Sub` ne vraća nikakav rezultat
- **Property**: `Property` ili svojstvo koristimo ukoliko bismo htjeli spremiti ili dohvatiti određeni podatak iz svog elementa. Ako bismo htjeli dohvatiti podatke koristimo `Get` naredbu a ako je potrebno i postavljanje vrijednosti svojstva koristimo i `Set` naredbu

Vratimo se sada na naš Hello World program, unutar koda primjećujemo jedan modul `Module1`, a unutar navedenog modula nalazimo jednu `Sub` proceduru `Main()`, pošto nigdje nije potrebno vraćanje rezultata korištenje `Sub` naredbe ima smisla. Dodatno treba napomenuti

kako naziv `Main()` nije slučajan, svaki Visual Basic program treba imati `Main()` proceduru koja je polazna točka izvođenja programa.[14]

Za ispis koristili smo `Console.WriteLine()` naredbu, unutar zagrada stavljamo što bi htjeli ispisati na ekran dok `Console.ReadLine()` služi samo za pauziranje programa jer bez nje bi se odmah upalio i ugasio. Nakon pokretanja program nam je ispisao rezultat, ali primijetili smo korisničko sučelje ili bolje rečeno njegov nedostatak. Ovakav je izgled konzolne aplikacije, tekst i ništa više.

Prvo mišljenje na ovakvo grafičko sučelje reklo bi se je većinom negativno, rijetko koji korisnik u današnje vrijeme bi htio gledati u ovakav program. Te dok je to istina ne koristimo konzolnu aplikaciju u pravu svrhu za što je ona danas namijenjena.

Konzolne aplikacije su najbolje za svrhu automatizacije ili takozvani *batch* program, kada je potreban program koji se sam pokreće u točno traženo vrijeme i odradi određeni zadatak koji je predvidiv i neće se mijenjati kroz vrijeme te sve to bez ikakve korisničke interakcije. Dakle, konzolne aplikacije su dizajnirane da bi ih pokretale druge aplikacije, ako je potrebno napraviti neki zadatak bez korisničkog utjecaja ovakve aplikacije su odlične za tu svrhu te imaju veliku moć za obavljanje takvih zadataka.[15]

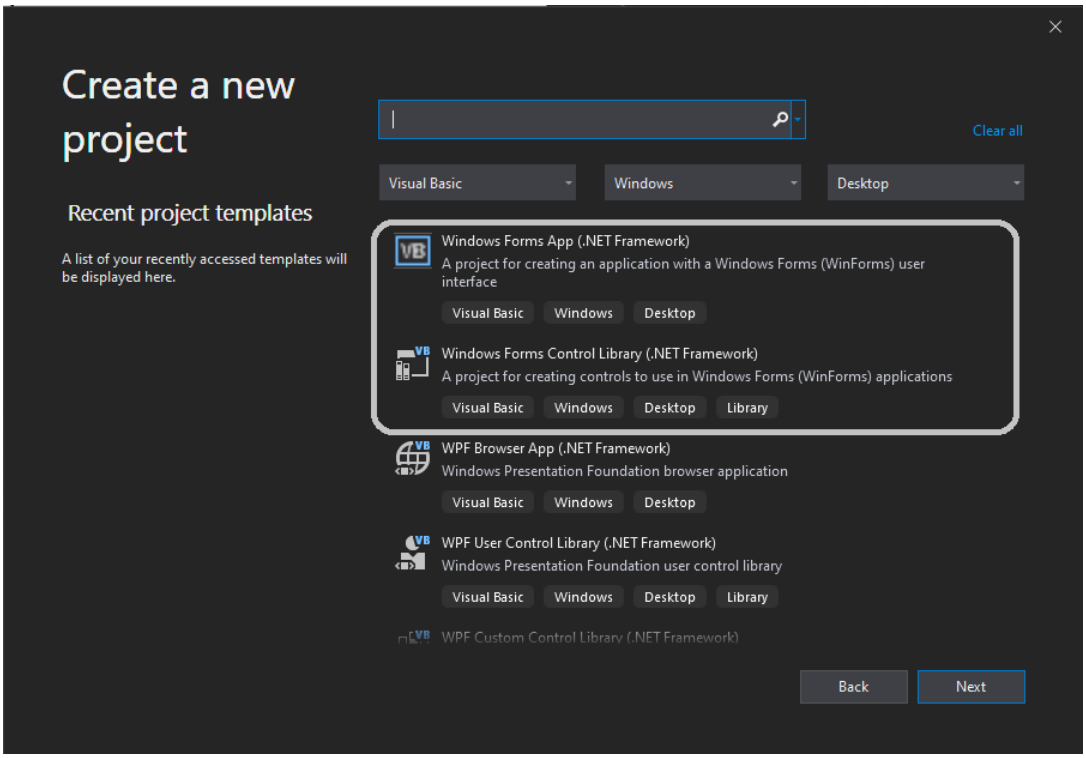
Prednosti ovakvih aplikacija su kao što smo vidjeli u primjeru jednostavnost, nije potrebno puno linija koda da bi samo pokrenuli program i dobili neke osnovne funkcionalnosti. Drugo je da su ovakvi tipovi aplikacija stabilni, konzolne aplikacije su prisutne već dugo vremena i malo toga može poći po zlu. Dodatna velika prednost je što su konzolne aplikacije jedina vrsta aplikacije koja je podržana i na drugim operacijskim sustavima(*cross-platform*) dokle god je napravljena unutar .NET Core frameworka.[15]

Glavni nedostatak je i naj očitiji, ovakav tip nije najbolji izbor za grafičko sučelje što se prosječnog korisnika tiče, puno je lakše koristiti obrazac i jednostavno u njega popuniti sve potrebne podatke.

5.2 Windows Forms

Prelazimo na drugi tip korisničkog sučelja zvan Windows Forms, Windows Forms je grafičko korisničko sučelje(GUI) koje je dio .NET Frameworka. Njegova glavna svrha je ponuda programeru sučelje za razvoj aplikacija koje su orijentirane prema korisnicima. Windows Forms ili skraćeno WinForms se sastoji od obrazaca(eng. *forms*) koje popunjavamo sa grafičkim elementima kako bi interakcija između računala i njegovog korisnika bila brža i intuitivnija. Neke od tih grafičkih elemenata ću prikazati u nastavku u aplikaciji. Kako bi

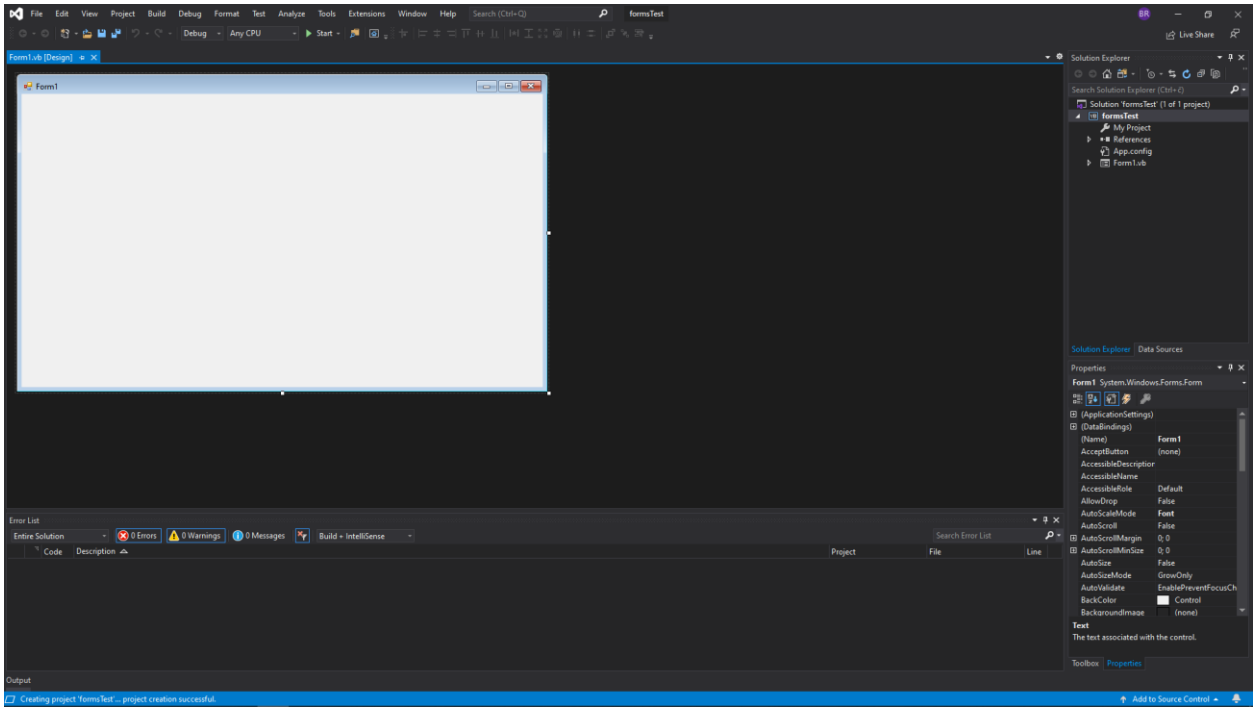
napravili WinForms program unutar Visual Studia potrebno je napraviti novi projekt gdje je potrebno odabrati WinForms tip projekta kao što je prikazano na slici 10.



Slika 10. Odabir WinForms aplikacije(autorski rad)

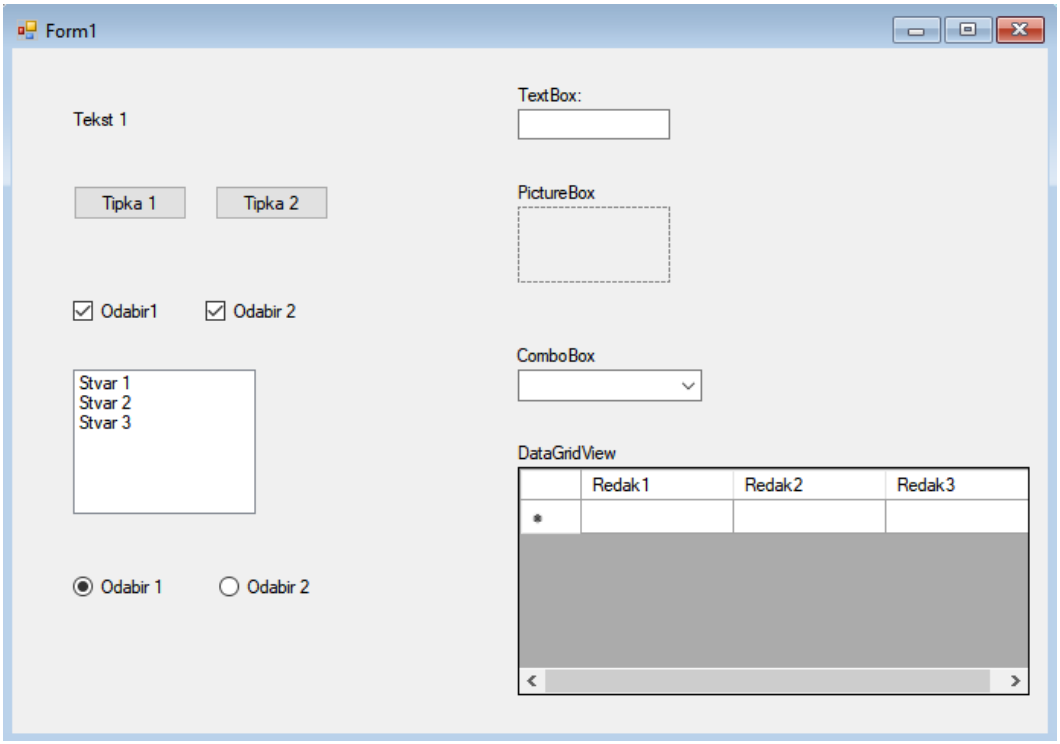
Kao što vidimo, sve vezano za Windows Forms je unutar sivog pravokutnika. Ako bismo htjeli napraviti WinForms aplikaciju potrebno je uzeti Windows Forms App(.NET Framework), kada dodamo osnovne informacije za naš projekt dobijemo glavni zaslon za uređivanje ovakvog tipa programa prikazanog na slici 11. Ako bismo htjeli napraviti vlastite kontrole za našu WinForms aplikaciju tada je potrebno napraviti WinForms Control Library.

Nakon što otvorimo naš projekt prikazati će nam se glavni prozor za naš projekt, ako usporedimo sa prozorom koji smo dobili unutar projekta za konzolne aplikacije, primijetiti ćemo kako najveći prozor više nije uređivač programskog koda. Sada glavni prozor je dizajner našeg obrasca, on nam omogućava prikaz kako bi prozor izgledao kada bi korisnik upalio aplikaciju.



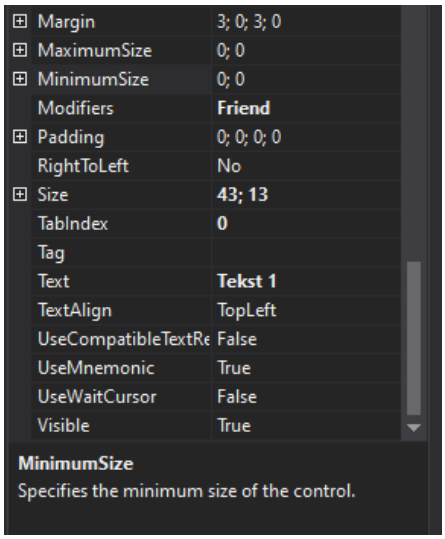
Slika 11. Glavni prozor za uređivanje WinForms aplikacija(autorski rad)

Primijetiti ćemo kako je naš obrazac prazan, to je zato što nismo dodali nikakve grafičke elemente, ponuda elemenata je velika tako da ću ovdje prikazati samo neke koji su više korišteni. Unutar WinForms programa svaki grafički element ima svoja svojstva(eng. *properties*) koja određuju osnovni izgled i funkcionalnost tog elementa. Primjer recimo za tekstualni element on će imati svojstva kao njegov tekstualni sadržaj, font, izgled teksta i pozicija gdje se nalazi. Drugi elementi mogu imati drugačija svojstva ovisno o njihovoj funkcionalnosti.



Slika 12. Primjer grafičkih elemenata unutar obrasca(autorski rad)

Unutar obrasca Form1 primjećujemo određene grafičke elemente koji nam služe za interakciju sa korisnikom, svaki na svoj način. Ako bi išli po redu od gornjeg lijevog kuta prvi na redu je takozvani label element, takav element nam nudi funkcionalnost prikaza teksta bilo gdje na obrascu. U svojstvima label elementa imamo Text svojstvo koje popunimo ovisno kakav tekst bi htjeli prikazati. Neka svojstva label elementa su prikazani i na slici 13, primijetite Text svojstvo i njegovu vrijednost.



Slika 13. Svojstva label elementa unutar WinForms programa (autorski rad)

Sljedeći element na redu je Button element, takav element bi mogli opisati kao tipku(ili gumb) koju kada korisnik klikne dogodi se određena radnja. Radnja koja će se dogoditi

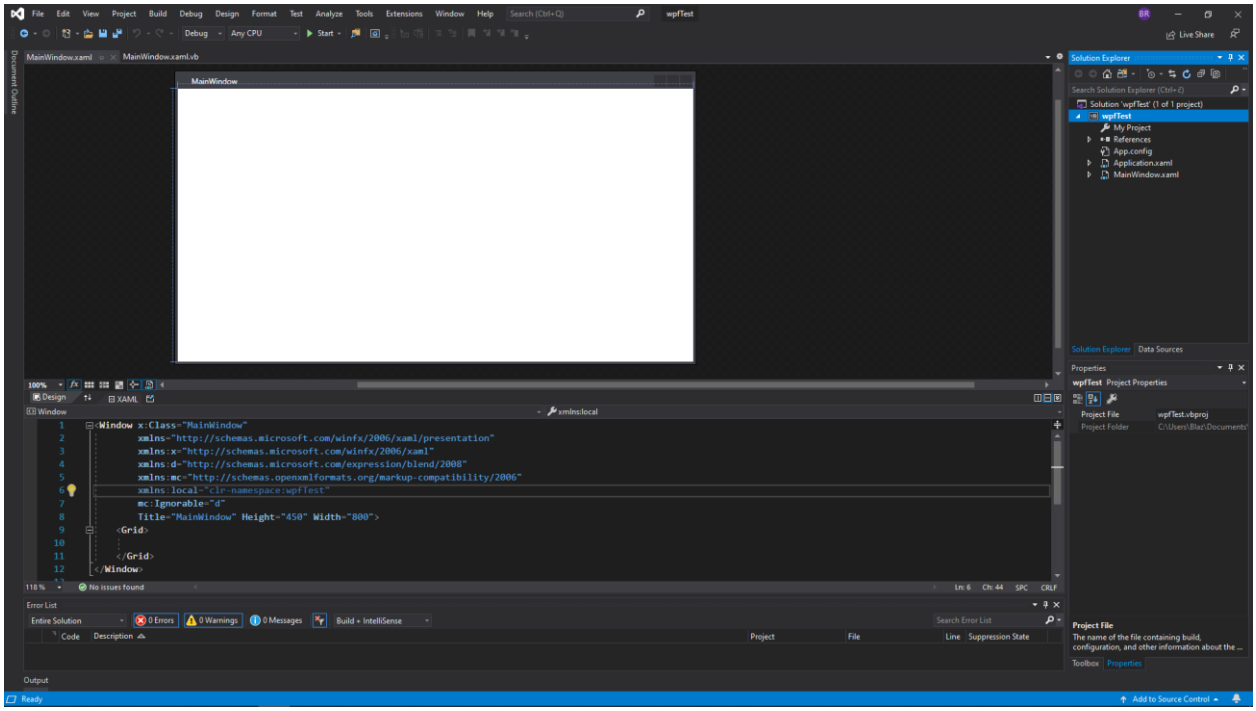
određuje programer u programskom kodu, to može biti spremanje trenutnih podataka ili recimo otvaranje drugog obrasca sa drugim elementima za drugu funkciju. Ispod Button elemenata vidimo CheckBox, takvi elementi su zaduženi za odabir i mogu biti isključeni ili uključeni. Također, oba elementa u isto vrijeme mogu imati istu vrijednost. U nastavku imamo ListBox element, takav element koristimo kada želimo pokazati jednostavni popis podataka, korisnik također može mišem označiti jedan ili više stavaka sa popisa ukoliko je to potrebno. Za zadnji element u prvom stupcu imamo takozvane RadioButton elemente. Ovakvi elementi se mogu usporediti sa prijašnje navedenim CheckBox elementima ali sa jednom bitnom razlikom a to je kako je moguće samo odrediti jedan RadioButton dok kod CheckBox elemenata nema takvog ograničenja.

Prelazimo u drugi stupac na vrhu gdje se nalazi TextBox element, ovakav element nam je najkorisniji kada nam je potrebno da korisnik upiše određene podatke u aplikaciju, bilo tekst ili brojeve. Ispod TextBox elementa nalazi se PictureBox, ovakav element koristimo kako i samo ime kaže ako bismo htjeli prikazati sliku unutar našeg obrasca. Trenutno ovaj element je prazan ali svrha je jasna. U nastavku nalazi se ComboBox, ovakav element vizualno bi mogli usporediti i sa ListBox elementom, oba elementa prikazuju popis određenih podataka koje je programer unaprijed postavio. ComboBox doduše ne pokazuje sve elemente cijelo vrijeme nego samo trenutno odabranog, a ako bi korisnik htio vidjeti sve elemente za to bi trebao prvo kliknuti na sami element. Nakon ComboBox elementa dolazimo do zadnjeg elementa na ovom obrascu a to je DataGridView. Ako bi htjeli prikazati složeni skup podataka kojih možda ima u velikom broju ili je broj informacija u jednom podatku veliki, za to nam najbolje služi DataGridView. Ovaj element također nudi složenije funkcionalnosti za interakciju sa korisnikom, moguće je označiti svake elemente posebno ili ih mijenjati ovisno o korisnikovim radnjama.

Što se tiče zašto koristiti WinForms, prednost je jasna u odnosu na konzolne aplikacije, grafičko sučelje je intuitivno i jednostavno za krajnjeg korisnika. Druge prednosti uključuju i jednostavnost pravljenja same aplikacije, princip slaganja grafičkih elemenata je jednostavan i povezivanje elemenata u programskom kodu je također lagano. WinForms kao način pravljenja aplikacija već duže vrijeme prisutno na tržištu i to donosi određene prednosti. Jedna od prednosti je kako su programi vrlo stabilni, a druga je i dalje velika rasprostranjenost ovakvih programa. Jedan od nedostataka je kako WinForms programi koriste samo snagu računalnog procesora, u današnje vrijeme došlo je do velikih napredaka u grafičkoj tehnologiji i korištenje samo procesorske snage na računalu može doći do neželjenih usporavanja rada računala.[15]

5.3 Windows Presentation Foundation

Windows Presentation Foundation ili skraćeno samo WPF je UI framework za izradu desktop aplikacija. WPF se nalazi unutar .NET frameworka te njegova svrha je iskorištavanje modernog grafičkog sklopovlja(eng. *hardware*). Nakon kreiranja WPF projekta otvara nam se glavni prozor za kreiranje WPF projekata koji je prikazan na slici 14.



Slika 14. Glavni prozor za uređivanje WPF projekta (autorski rad)

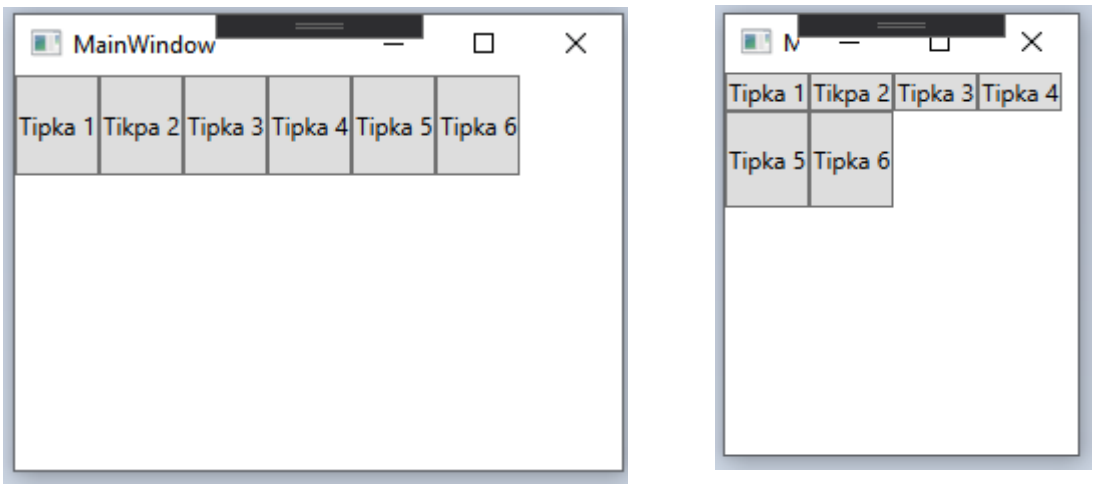
Ako pogledamo prozor za uređivanje, primijetiti ćemo kako je prozor identičan editoru za WinForms projekt ali sa jednim dodatkom. Primijetiti ćemo dodatni prozor za uređivanje programskog koda ispod samog izgleda trenutnog obrasca. Programski kod koji se ovdje nalazi nije Visual Basic, nego *Extensible Application Markup Language* ili skraćeno XAML. Ovaj dodani programski jezik nam omogućuje puno više mogućnosti i slobode nego što smo imali unutar WinForms programa što se tiče samog izgleda i ponašanja grafičkih elemenata. Cilj uvođenja XAML jezika je bilo odvajanje koda za izgled aplikacije i koda za samu funkcionalnost, Visual Basic je i dalje zadužen za funkcionalnost i u WPF projektima i to se naziva *Code-behind*.[16]

Za demonstraciju prikazati ću moguće načine rasporeda elemenata unutar prozora kada radimo sa WPF projektom. Koristiti ćemo takozvane WPF panels, takozvani paneli određuju raspored elemenata u prozoru, koliko svaki element može zauzimati prostora i kako će se mijenjati raspored ako dođe do promijene dimenzija prozora.

Za prvu vrstu panela imamo WrapPanel, u našem primjeru XAML kod izgleda ovako:

```
<WrapPanel>
        <Button>Tipka 1</Button>
        <Button>Tikpa 2</Button>
        <Button>Tipka 3</Button>
        <Button>Tipka 4</Button>
        <Button>Tipka 5</Button>
        <Button Height="50">Tipka 6</Button>
</WrapPanel>
```

Primjećujemo 6 tipki unutar jednog panela, kada pokrenemo program izgledati će ovako:



Slika 15 i 16. WPF program koristeći WrapPanel(autorski rad)

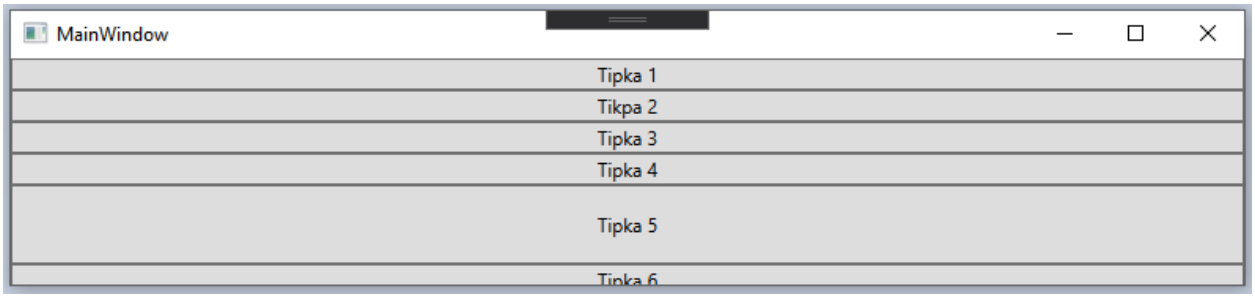
Primjećujemo na lijevoj slici kako su sve tipke u jednom redu, također unutar XAML koda postavili smo kako je tipka 6 visine 50 piksela, te automatski i cijeli red će biti postavljen na takvu visinu. Ali što se dogodi kada smanjimo prozor? Na desnoj slici kada smo smanjili prozor primijetili smo kako tipke koje više ne stanu su se prebacile u novi red, ovo je jedno od svojstva ovakvog panela. Također primijetili smo kako tipke 5 i 6 su veće od ostalih, to je razlog zato što kako smo rekli da na tipki 6 smo postavili visinu 50 piksela dok drugi red nema takvo svojstvo te takve tipke će biti najmanje što mogu biti.

Sljedeća vrsta panela je StackPanel, ovakva vrsta panela jednostavno slaže jedan element ispod drugog. Želimo li odrediti orijentaciju u kojem želimo slaganje elemenata možemo postaviti sa naredbom `Orientation`, u nastavku je primjer sa ovim panelom.

```
<StackPanel>
        <Button>Tipka 1</Button>
        <Button>Tikpa 2</Button>
        <Button>Tipka 3</Button>
        <Button>Tipka 4</Button>
```

```
        <Button Height="50">Tipka 5</Button>
        <Button >Tipka 6</Button>
</StackPanel>
```

Kada ćemo pokrenuti program dobiti ćemo izgled:

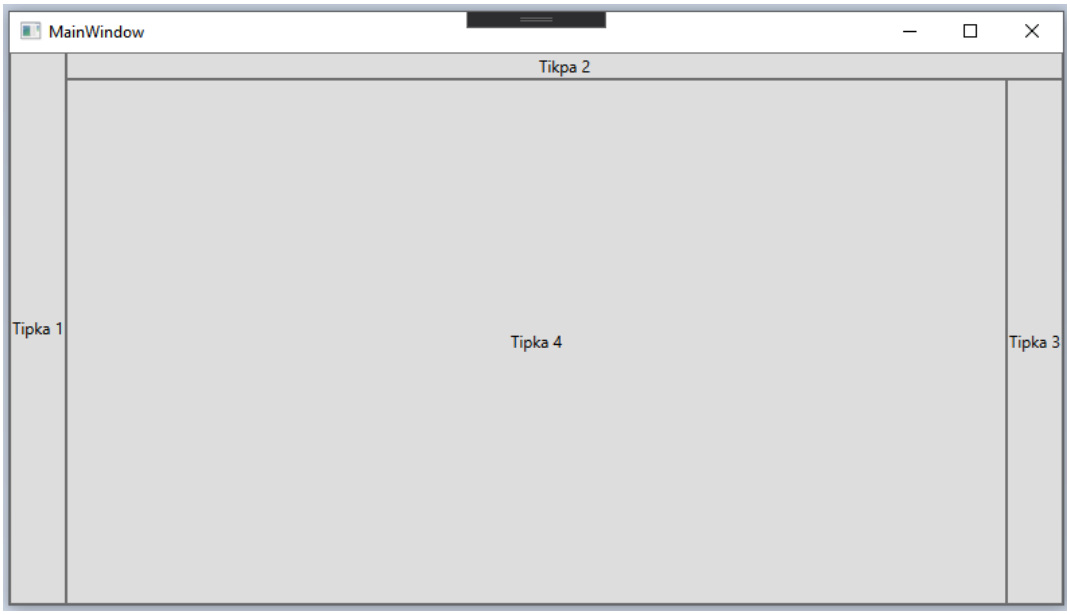


Slika 17. WPF program koristeći StackPanel(autorski rad)

Naš program je prikazao sve tipke jednu ispod druge, svaka tipka zauzima sav prostor jer je to trenutno svojstvo StackPanel panela, tipka 5 je veća od drugih jer smo na njoj definirali svojstvo `Height` od 50 piksela.

Sljedeći panel na redu je DockPanel, ovakav panel je malo složeniji u odnosu na prošle te je više moguće napraviti sa njim. DockPanel radi na principu povezivanja elemenata sa jednom od strana prozora, postoje četiri strane prozora: top, right, bottom i left. U nastavku je primjer sa ovakvim panelom te njegov rezultat.

```
<DockPanel>
        <Button>Tipka 1</Button>
        <Button DockPanel.Dock="Top">Tikpa 2</Button>
        <Button DockPanel.Dock="Right">Tipka 3</Button>
        <Button>Tipka 4</Button>
</DockPanel>
```



Slika 18. WPF program koristeći DockPanel(autorski rad)

Razlog zašto je ovakav raspored tipki je iz razloga što smo ih povezali na druge strane prozora. Koristili smo naredbu `DockPanel.Dock` kako bi odredili svojstvo gdje se povezuju određene tipke. Za tipku 1 nismo specifično postavili na koju stranu se veže element te WPF automatski je postavlja na lijevu stranu prozora. Tipka 2 primjećujemo je pri vrhu jer smo postavili svojstvo na Top odnosno vrh prozora, sad doduše dolazimo do pitanja u gornjem lijevom kutu koja tipka će „osvojiti“ ovaj prostor, rješenje je jednostavno te koja god tipka je iznad u programskom kodu ona dobiva prostor. Za tipku 3 nam je jasno, ali tipka 4 primjećujemo zauzima sav slobodan prostor koji je ostao na prozoru. Ovo je svojstvo samog panela te ako ne želimo ovakvo popunjavanje prostora trebamo postaviti svojstvo `LastChildFill` na vrijednost `False`.

WPF kao grafičko rješenje je jako sposobno, Microsoft je napravio WPF sa ciljem korištenja novog računalnog grafičkog sklopovlja koje je bilo sve jače. Ovakav framework ima mnogo prednosti u odnosu na WinForms, sa WPF-om dolazi moderan dizajn koji nudi puno više grafičkih mogućnosti koje više nisu samo ograničene na moć računalnog procesora. Druga prednost bi bilo razdvajanje programskog koda za izgled i programskog koda za funkcionalnost, ovakva podjela omogućuje recimo programskom dizajneru nesmetani rad gdje on može u potpunosti napraviti samo izgled aplikacije te zatim drugi programer može ispuniti svu funkcionalnost programa.[15]

Nedostaci bi se mogli povezati direktno sa prednostima, kako imamo puno više mogućnosti potrebno je više vremena za razvoj samog programa u odnosu na WinForms gdje je samo potrebno postaviti mišem grafički element na obrazac. Drugi nedostatak je kako bi

mogli napraviti sve nove grafičke mogućnosti Microsoft je uveo novi programski jezik XAML te time samo znanje Visual Basica nije dovoljno za razvoj WPF programa.

5.4 Universal Windows Platform

Universal Windows Platform ili skraćeno UWP je Microsoftov novi način izrade aplikacija za Windows operacijski sustav. Naziv *universal* dolazi iz svrhe za izradom univerzalnih aplikacija koje bi radile na drugim Microsoft uređajima kao što su Windows 10 Mobile, Xbox One i HoloLens. UWP aplikacije se i dalje mogu razviti pomoću VB.NET jezika te uz pomoć jezika za korisničko sučelje kao XAML ili HTML.

UWP aplikacije imaju pojačanu sigurnost u odnosu na prošle metode, aplikacija kada je pokrenuta ima ograničene resurse te joj nije dopušten pristup ostalim resursima. Ukoliko aplikacija zahtjeva nove resurse prvo je potrebno zatražiti korisnikovo odobrenje. Ovom zaštitom korisnik ne treba previše brinuti o sigurnosti aplikacije koje preuzima jer su aplikacije vrlo ograničene. [15]

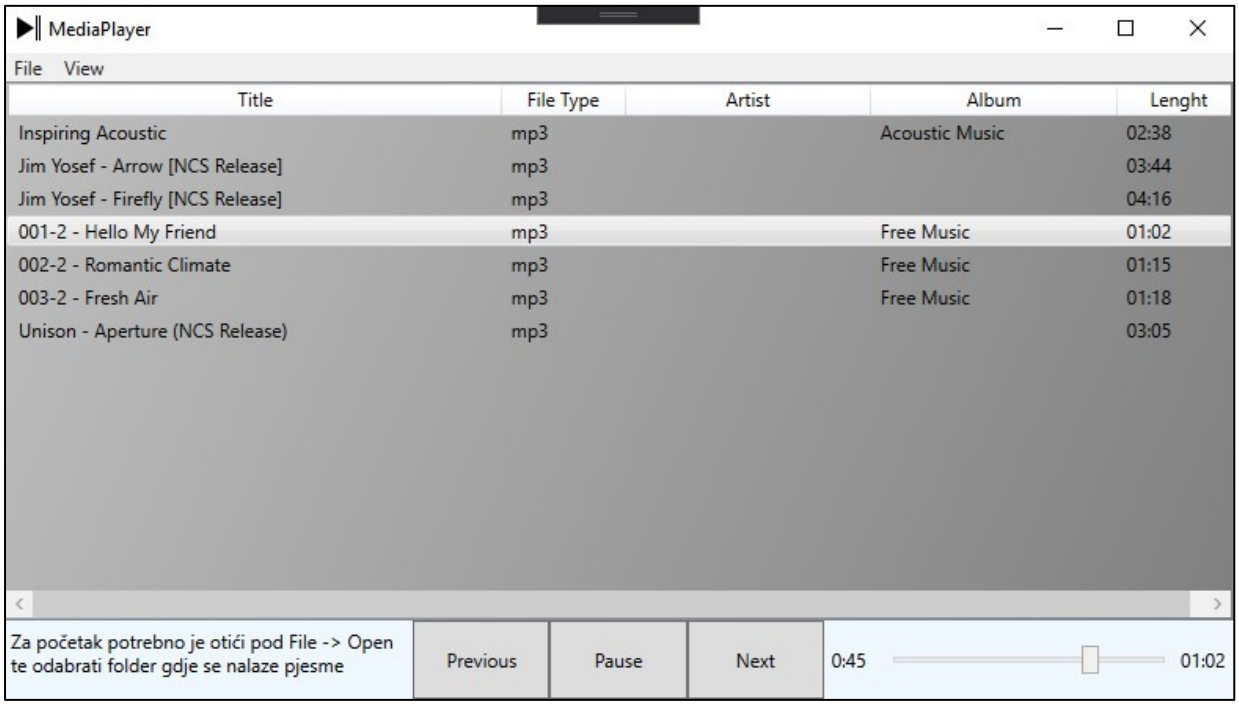
Uvođenjem UWP-a Microsoft je također napravio novu aplikaciju za distribuciju takvih aplikacija koja je nazvana Microsoft Store. Ovakva trgovina aplikacijama omogućava olakšanu distribuciju aplikacija na Windows operacijskim sustavima, druga prednost izbacivanjem aplikacije na Microsoft Store je što je Microsoft zadužen za naplaćivanje bilo kakve funkcionalnosti što je developer zadao, značajno olakšavajući samu kupovinu aplikacija.[15]

Sa druge strane, dok je Microsoft Store bila aplikacija koja bi uvelike olakšala distribuciju aplikacija problem je kako nije uvelike zaživjela. Store je bio jedini način distribucije UWP aplikacija te samim time broj korisnika koji su koristili ovakvu vrstu aplikacija nije bio veliki. Kao što je gore navedeno pojačana sigurnost same aplikacije je i nedostatak za sam razvoj navedene aplikacije. Zbog dodanih ograničenja programeri su potrebni uložiti puno više truda kako bi napravili određene funkcionalnosti koje bi na drugim metodama razvoja bile puno lakše za implementirati.[15]

Za kraj, i dok UWP nije zaživio kako je Microsoft na početku zamislio, napravljeni su određeni koraci kako bi se krenulo u pravom smjeru. UWP nudi nove i napredne funkcionalnosti za razvoj aplikacija te samo je pitanje kada će postati sposobna metoda za razvoj aplikacija na Windows operacijskim sustavima.

6.Praktični dio: svirač glazbe

Za praktični dio ovog rada odlučio sam napraviti svirač glazbe, svirač sam napravio kao WPF projekt i za izradu projekta koristio sam Visual Basic.NET i XAML programski kod.



Slika 19. Svirač glazbe-prikaz tablice(autorski rad)

Svrha ovog programa je reprodukcija glazbe koja se nalazi na korisničkom računalu. Pri početku rada potrebno je odabrati mapu gdje se nalaze pjesme. Radnja otvaranja je povezana sa već ugrađenom naredbom(eng. *command*) za otvaranje novog elementa, ovakvo povezivanje nam omogućuje pozivanje iste naredbe iz više različitih događaja. U ovoj aplikaciji događaji koji otvaraju odabir mape je ako korisnik klikne na tipku za otvaranje unutar glavnog izbornika ili ako pritisne prečac na tipkovnici Ctrl + O.

Nakon što je odabrana željena mapa unutar dijaloga za odabir počinje proces čitanja svakog elementa unutar mape i provjere je li podržan format koji je moguće zvučno reproducirati. Posebna klasa nazvana MediaElement je zadužena za spremanje podataka o svakoj pojedinačnoj pjesmi. Obavezni podaci koji se spremaju za svaku pjesmu su puna putanja pjesme na korisničkom mediju, tip podataka pjesme, duljina te naziv. Ukoliko pjesma sadrži i podatke o autoru pjesme, albumu te slici ti podaci se također spremaju.

U glavnom izborniku, korisnik ima mogućnosti dva prikaza popisa pjesama. Prvi prikaz je prikaz pomoću tablice, za ostvarenje ovog prikaza koristio sam XAML element ListView za

koji je izvor podataka skup svih učitanih pjesama. Druga metoda je prikaz takozvanih „pločica“ gdje je glavni vizualni element slika pjesme koja se koristi za interakciju sa korisnikom.

Sada, za početak rada potrebno je u gornjem lijevom kutu otići u postavke pod File pa zatim Open. Otvoriti će nam se prozor kojim odabiremo folder u kojemu se nalazi naša glazba, nakon odabira lokacije čitač će učitati sve podatke koji se nalaze u tom folderu i prikazati ih sviraču ako su podržani format. Svirač nam pokazuje osnovne informacije kao naziv pjesme, koji je format i trajanje pjesme. Ukoliko pjesma sadrži i ID3 podatke pokazati će se i informacije kao album u kojemu se pjesma nalazi i vlasnik pjesme.

Reprodukciju glazbe možemo pokrenuti klikom na željenu pjesmu, vizualno će se označiti pjesma koja je trenutno u reprodukciji te će tipke Previous, Pause i Next biti dostupne. Također desno od tipki nalazi se klizač koji nam pokazuje trenutni postotak prelaska pjesme. Korisnik također može pomaknuti klizač na mjesto gdje bi htio reproducirati pjesmu.

7.Zaključak

Izrada cijele aplikacije činio se kako izazov kada sam tek počeo raditi ovaj projekt. Sa vremenom sam savladao osnove kreiranja grafičkih aplikacija te izazov više nije bio kompliciran. Naučio sam vrste grafičkih sučelja i njihov način kako rade, koji je njihov cilj i zašto se koriste.

Visual Basic kao programski jezik je bio zanimljiv, njegova mogućnost laganog kreiranja grafičkih aplikacija mi se je činila kao velika prednost pogotovo kada je prvi put izašao na tržište. Kada je Microsoft odlučio prenijeti Visual Basic unutar .NET frameworka nekima se je činio kao loš potez. Programeri su bili skeptični kako će izgledati novi Visual Basic unutar .NET sustava i dok nažalost programski jezik je morao proći kroz velike preinake, krajnji rezultat se je činio zadovoljavajuć. Do dan danas Visual Basic.NET je unutar .NET frameworka te i dalje je aktivno podržan od strane Microsofta zajedno sa jezicima C# i F#. Nadam se da će i u budućnosti tako ostati.

Sve u svemu, cijeli projekt mi je ukazao na sve vrste grafičkih sučelja koje su na izbor programeru i dobio sam detaljan uvid u njihovu svrhu u današnjem svijetu programiranja a samim time dobio sam iskustvo za njihovo korištenje u budućnosti ukoliko će njihova primjena biti potrebna.

Popis literature

[1] Leksikografski zavod Miroslav Krleža, „sučelje“, [Na internetu]. Dostupno na: https://www.enciklopedija.hr/natuknica.aspx?ID=58610 [pristupljeno kol. 29, 2020].

[2] J. Smiley, „Visual Basic History“, [Na internetu]. Dostupno na: http://johnsmiley.com/visualbasic/vbhistory.htm [pristupljeno kol. 18, 2020].

[3] „Microsoft Visual Basic 1.0“, [Na internetu]. Dostupno na: https://winworldpc.com/screenshot/40c3942c-c281-2230-11c3-a4e284a2c3a5/3d4d42c6-9257-c3b1-11c3-a4c2a90f7054 [pristupljeno kol. 18, 2020].

[4] „Microsoft Visual Basic 6.0“, [Na internetu]. Dostupno na: https://winworldpc.com/product/microsoft-visual-bas/60 [pristupljeno kol. 19, 2020].

[5] M. MacDonald, „The Rise and Fall of Visual Basic“, [Na internetu]. Dostupno na: https://medium.com/young-coder/the-rise-and-fall-of-visual-basic-f422252349a6 [pristupljeno kol. 19, 2020].

[6] P. Thurrott, „20 Years of Visual Studio: Visual Basic“, [Na internetu]. Dostupno na: https://www.thurrott.com/dev/106091/20-years-visual-studio-visual-basic [pristupljeno kol. 20, 2020].

[7] „Visual Basic - Microsoft Visual Basic 6.0 Deluxe Learning Edition“, [Na internetu]. Dostupno na: https://www.lawebdelprogramador.com/foros/Visual-Basic/1582025-Microsoft-Visual-Basic-6.0-Deluxe-Learning-Edition.html [pristupljeno kol. 20, 2020].

[8] H. Soffar, „Visual Basics programming language advantages and disadvantages“, [Na internetu]. Dostupno na: https://www.online-sciences.com/programming/visual-basics-programming-language-advantages-and-disadvantages/ [pristupljeno kol. 20, 2020].

[9] D. Grundgeiger, *Programming Visual Basic .NET*. O'Reilly, 2002.

[10] J. Piiroinen, „Visual overview of the Common Language Infrastructure (CLI)“, [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/.NET_Framework#/media/File:Overview_of_the_Common_Language_Infrastructure.svg [pristupljeno kol. 21, 2020].

[11] „Difference Between Visual Basic and Visual Basic.Net (VB6 and VB.net)“, [Na internetu]. Dostupno na: https://www.differencebetween.com/difference-between-visual-basic-and-vs-visual-basic-net-vb6-and-vs-vb-net/ [pristupljeno kol. 20, 2020].

[12] „Difference Between VB.NET and Visual Basic“, [Na internetu]. Dostupno na: https://www.javatpoint.com/vb-net-vs-visual-basic [pristupljeno kol. 19, 2020].

[13] „Differences Between .NET Framework, .NET Core, and .NET Standard“, [Na internetu]. Dostupno na: https://code-maze.com/differences-between-net-framework-net-core-and-net-standard/ [pristupljeno kol. 21, 2020].

[14] „Structure of a Visual Basic Program“, [Na internetu]. Dostupno na: https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/program-structure/structure-of-a-visual-basic-program#file-level-programming-elements [pristupljeno kol. 22, 2020].

[15] T. Corey, „WinForm vs WPF vs UWP vs Console - The C# Desktop UI Showdown (and the future with .NET 5)", *YouTube*, 2019, [Na internetu]. Dostupno na: https://www.youtube.com/watch?v=yq0dSkA1vpM [pristupljeno kol. 22, 2020].

[16] „WPF overview“, [Na internetu]. Dostupno na: https://docs.microsoft.com/en-us/dotnet/framework/wpf/introduction-to-wpf [pristupljeno kol. 22, 2020].