Rad s bazama podataka u programskom jeziku Visual Basicu.NET

Hegedušić, Mario

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:858987

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mario Hegedušić

Rad s bazama podataka u Visual Basic.NET

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mario Hegedušić

Matični broj: 46296

Studij: Primjena informacijske tehnologije u poslovanju

Rad s bazama podataka u Visal Basic.NET

ZAVRŠNI/DIPLOMSKI RAD

Mentor/Mentorica:

Doc. dr. sc. Mario Konecki

Varaždin, rujan 2021.

Mario Hegedušić

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

Ova tema bavi se razvojem Visual Basic.NET aplikacija i njihovim spajanjem na baze podataka. Polazi se od upoznavanja sa programskim jezikom, koje su njegove glavne karakteristike i njegova sintaksa. Pokazuju se primjeri pomoću kojih možemo shvatiti kako funkcionira povezivanje baza podataka sa aplikacijom izrađenom unutar Visual Studio 2019 programa i Visual Basic.NET programskog jezika. Prikazuju se glavne karakteristike ADO.NET arhitekture i ona se povezuje s programskim jezikom koji se obrađuje.

Ključne riječi: Visual Basic.NET, SQL, Baza podataka, ADO.NET, arhitektura, metode, klase, povezivanje, izvor podataka, podaci, Access.

Sadržaj

1. Uvod

Tema ovog završnog rada je Visual Basic.NET, konkretnije rad s bazama podataka unutar tog programskog jezika. Visual Basic.NET je program koji se sastoji od više paradigmi i on je objektno orijentirani programski jezik koji je implementiran na .NET Coreu i .NET Frameworku. Ovaj programski jezik izdan je 2002.godine i on je nasljednik originalnog Visual Basic programskog jezika. Integrirano razvojno okruženje koje se koristi za razvoj aplikacija u Visual Basic.NET je Visual Studio. Ovaj programski jezik je uz C# i F# jedan od 3 glavna jezika koji se temelje na .NET Frameworku. Također Microsoft je objavio kako je razvoj VB.NETa završio sa 11. Ožujkom 2020.

U ovom završnom radu ćemo još dati detaljne primjere kako se sve mogu kreirati baze podataka u ovom programskom jeziku te koji su sve načini spajanja s drugim bazama podataka, odnosno s bazama podataka napravljenim u drugim okruženjima.

Motivacija i razlog zbog kojeg sam odabrao baš ovu temu za svoj završni rad je taj što želim naučiti više o samim bazama podataka, vidjeti koje su sve mogućnosti s njihovim radom te kako se sve možemo spojiti i koristiti sa bazom podataka te ista ta saznanja prenijeti i Vama.

2. Visual Basic.NET

VB.NET često se koristi zajedno s Grafičkim korisničkim sučeljem unutar sustava windows i njegovih formi za izradu računalnih aplikacija za windows. Programiranje takvih aplikacija unutar VB.NET podrazumijeva „drag and drop“ metodu izrade pomoću koje se unutar programa samo postavljaju kontrole i forme.

VB.NET koristi se naredbama kako bi se odredile akcije koje se žele pokrenuti. Najčešći naredbe koje se pojavljuju su izrazi kojima se dodjeljuje neka vrijednost. Dodjeljivanje vrijednosti nekom izrazu može se postići tako da se pozove neki potprogram ili neka funkcija. Također izrazu se može direktno dodijeliti neka vrijednost upisivanjem iste nakon znaka „=“. U ovom programskom jeziku također postoje i neke naredbe kojima postavljamo uvjete i određujemo smjer kojim se redoslijedom programski kod mora izvršavati, gdje će taj programski kod završiti, kao što su „If...Then...Else...End If“ i „Select Case....Case...End Select“. Ovdje možemo pronaći sličnost sa nekim programskim jezicima kao što su C#, C++, Python. Također postoje i petlje kojima se određuje koliko puta će se neki dio koda izvršiti. Imamo tri vrste iteracija u VB.NET, a to su „Do....Loop“, „For....To“ i „For Each“. Naredba Do .... Loop mora u sebi imati odjeljak za inicijalizaciju gdje određujemo od kojeg podatka će petlja krenuti, te također mora imati odjeljak za ispitivanje gdje će se odrediti do kojeg podatka petlja mora stići kako bi završila. Oba dva ta podatka moraju biti prisutna u petlji, dok petlja For Each prolazi kroz svaki podatak unutar neke liste.

Unutar VB.NET ne postoje neki znakovi kojim bismo određivali gdje započinje koji blok, a gdje prestaje, već se u ovom programskom jeziku blokovi započinju imenom naredbe kao što je npr. „If“, „Sub“ i sl., a završavaju se ključnim riječima kao što su „End If“, „End Sub“. Naredbe unutar VB.NET se završavaju sa dvotočkom („:“) ili se samo sa krajem retka podrazumijeva i kraj naredbe. Ako želimo više naredbi napisati u jedan red onda se na kraju naredbe dodaje „_“, međutim potreba za ovakvim pisanjem naredbi se uvelike maknula u verzijama 10 i više. Znak jednako „=“ se može koristiti u više situacija, tj. on može biti interpretiran kao znak pridruživanja i kao znak uspoređivanja. Ovdje vidimo kako se taj znak razlikuje od drugih programskih jezika gdje se za znak pridruživanja također koristi „=“, ali za znak uspoređivanja se koriste dva znaka jednako ili čak tri. Nadalje unutar sintakse ovog programskog jezika imamo i zagrade. Okrugle zagrade koriste se kao polja, kao deklaracija nekog polja, ali se koriste i kod pretraživanja polja preko indeksa. Također te zagrade se koriste i kod definiranja potprograma i funkcija. Još moramo spomenuti i jednostruke navodnike koji se koriste kako bi se u programu naznačio komentar.

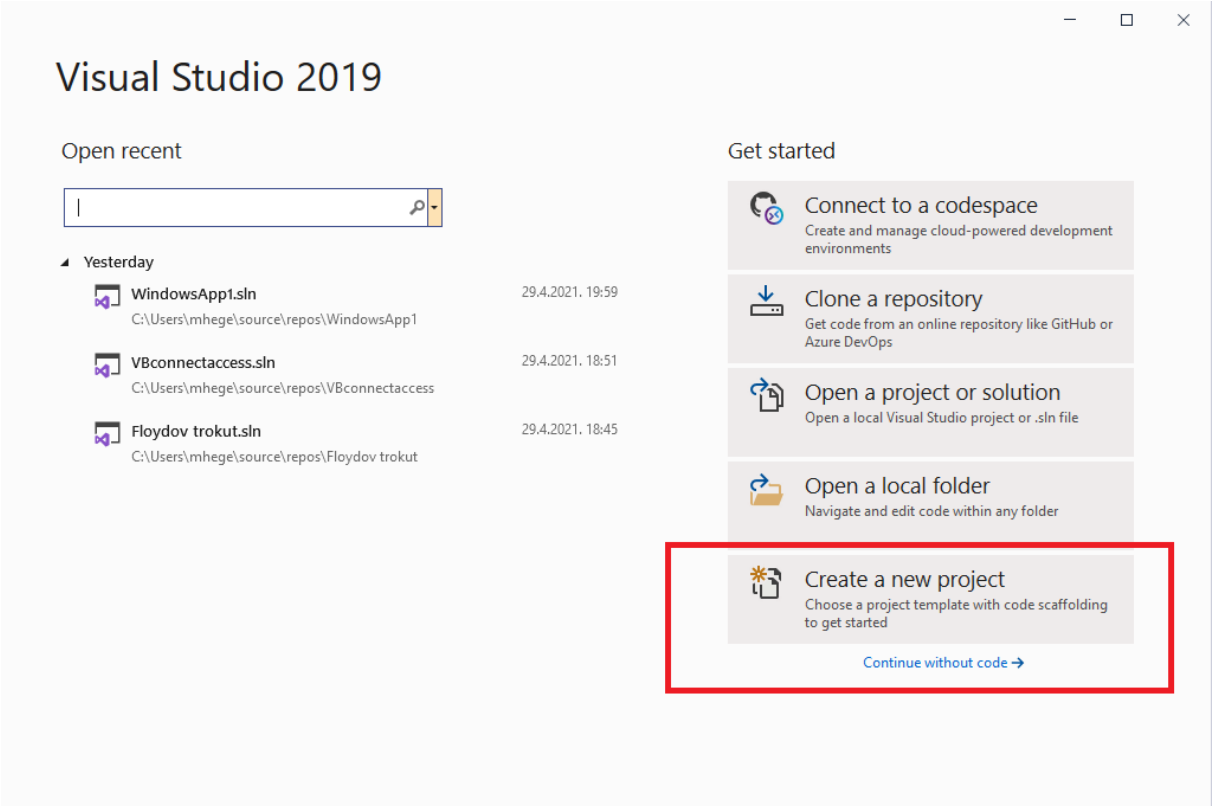
```
Imports System.Console

Module Program

    Sub Main()
        Dim rows As Integer

        ' Input validation.
        Do Until Integer.TryParse(ReadLine("Enter a value for how many rows to be displayed: " & vbcrlf), rows) AndAlso rows >= 1
            WriteLine("Allowed range is 1 and {0}", Integer.MaxValue)
        Loop

        ' Output of Floyd's Triangle
        Dim current As Integer = 1
        Dim row As Integer
        Dim column As Integer
        For row = 1 To rows
            For column = 1 To row
                Write("{0,-2} ", current)
                current += 1
            Next

            WriteLine()
        Next
    End Sub
```

```
    ''' <summary>
    ''' Like Console.ReadLine but takes a prompt string.
    ''' </summary>
    Function ReadLine(Optional prompt As String = Nothing) As String
        If prompt IsNot Nothing Then
            Write(prompt)
        End If

        Return Console.ReadLine()
    End Function

End Module
```

Ovo je primjer jednog programa izrađenog u VB.NET koji nam pokazuje sve naredbe koje smo do sada spomenuli i on izračunava i ispisuje brojeve Floydovog trokuta od proizvoljnog broja redova. Kao što vidimo skroz na početku ovog programa imamo petlju Do ... Loop koja nam osigurava da broj redova koji unesemo mora biti cijeli broj i mora biti jednak ili veći od 1, i ta petlja se tako dugo izvršava dok se ne ispune ti uvjeti. Također ovdje vidimo i kako se definiraju pojedine varijable, a to je sa naredbom „Dim“. Naredba „As“ osigurava nam da odredimo vrstu podataka koja će se upisivati u tu varijablu (brojevi, slova, decimalni brojevi...). „Sub“ nam označava jedan blok programa. Također ovdje još vidimo i tri jednostruka navodna znaka u jednom retku, što označava taj cijeli redak kao komentar.

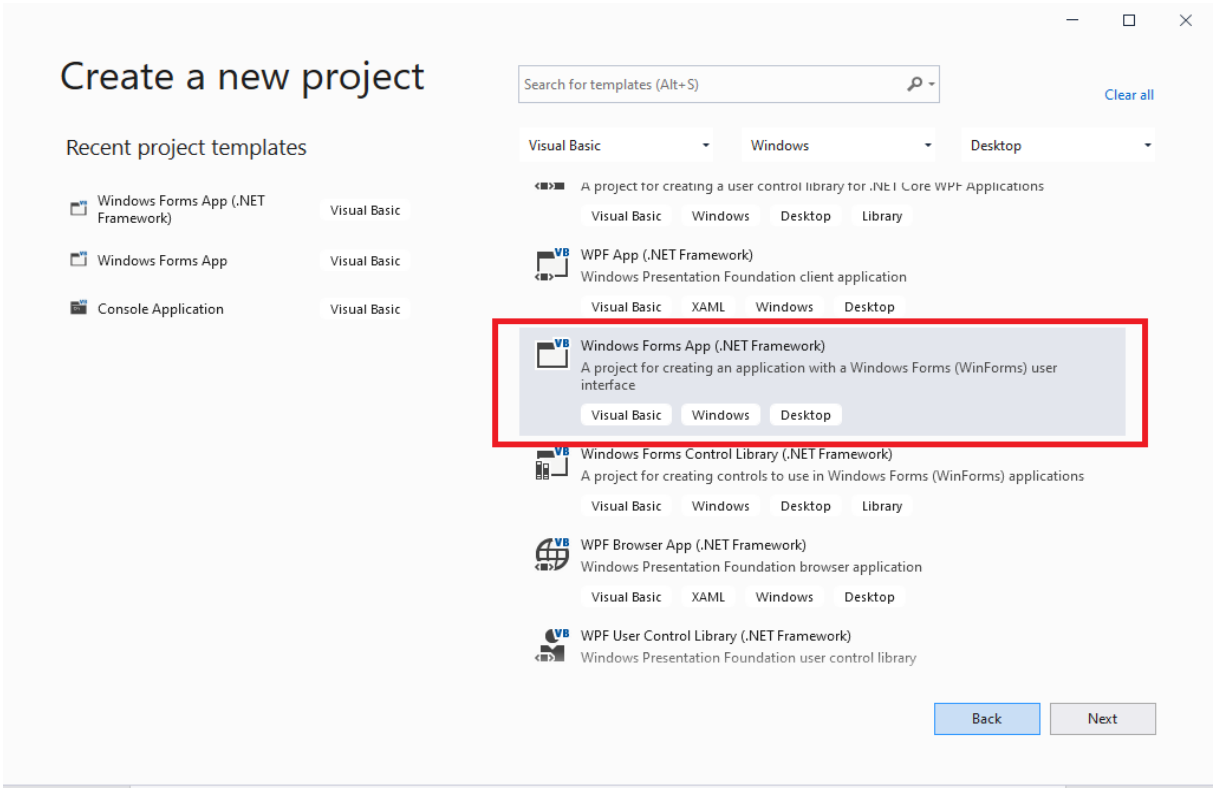
2.1. Povezivanje Access baze podataka

U ovom poglavlju govoriti ću o mogućim povezivanjima access baze sa Visual Basic.NET programskim jezikom. Prvi način koji ću spomenuti je način da napravimo formu unutar Visual Basic programa te preko forme dodajemo zapise koji se nalaze unutar baze s kojom ćemo se povezati. Kako bismo to napravili, prilikom otvaranja Visual Studio programa moramo odabrati da želimo kreirati novi projekt te krenuti bez već izrađenih obrazaca.



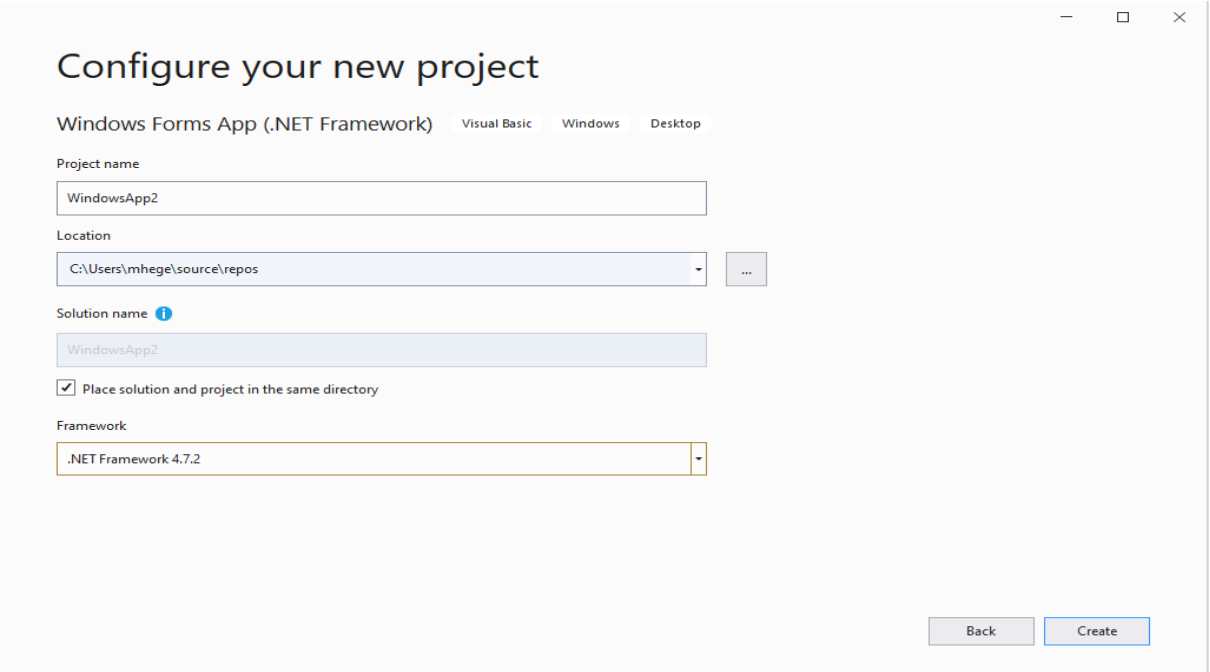
Slika 1. Kreiranje novog projekta

Kada smo odabrali da želimo kreirati novi projekt (vidljivo na slici 1.), otvara nam se novi prozor u kojem odabiremo između više ponuđenih obrazaca kakav želimo mi kreirati. U slučaju kada želimo povezati bazu podataka sa svojom formom koju ćemo napraviti moramo odabrati Windows Form App(.NET Framework). Kada bismo odabrali samo Windows Form App tada ne bismo mogli povezati svoju bazu podataka jer takva forma ne podržava povezivanje podataka između baze i forme. [5.]



Slika 3. Odabir forme

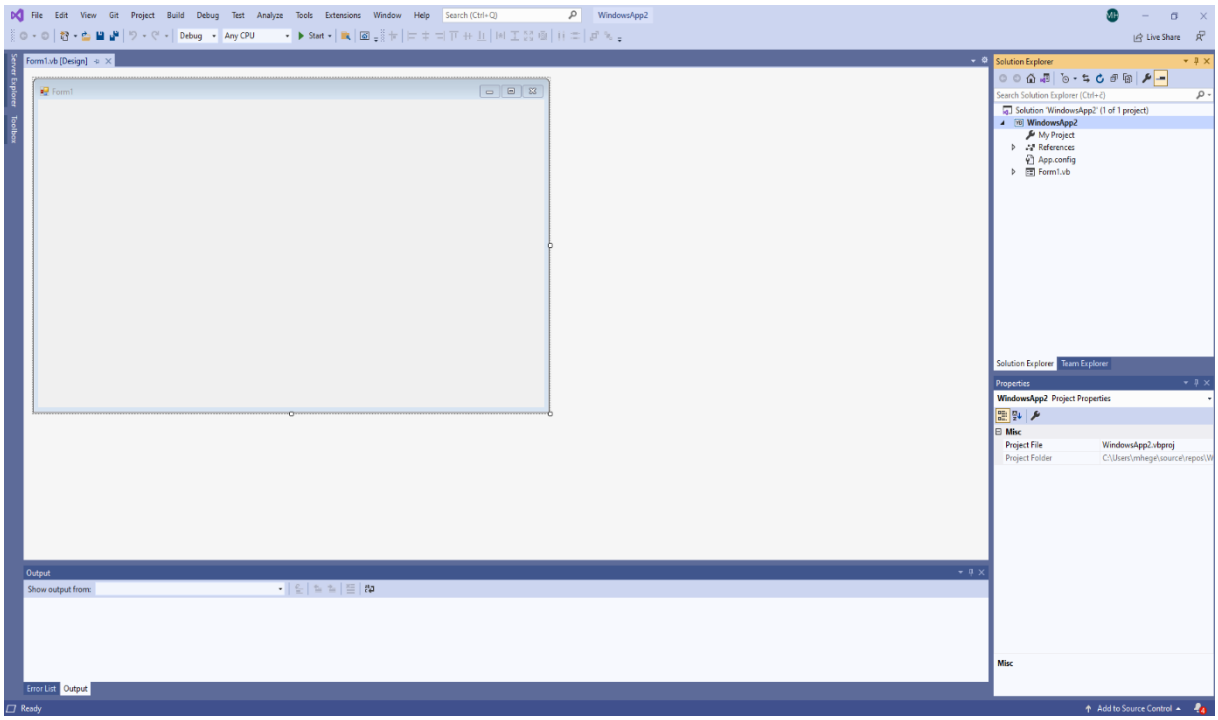
Dalje, nakon što odaberemo vrstu forme koju želimo izraditi upisujemo naziv forme te mjesto na kojem će se naš projekt spremiti. Isto tako moramo odabrati i verziju .NET



Slika 2. Završna konfiguracija projekta

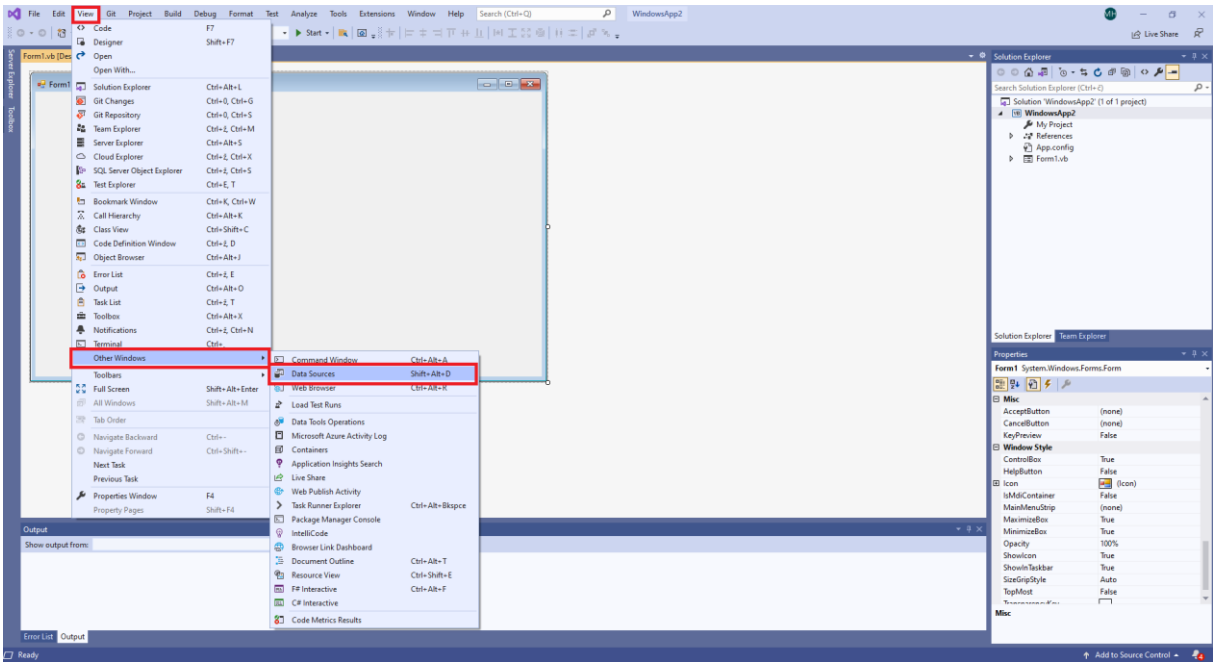
frameworka koju ćemo koristiti (Visual Studio nam automatski označava najnoviju verziju) te pritisnemo gumb „Create“. [5.]

Kada smo napravili sve korake do sada i konfigurirali svoj projekt te pritisnuli gumb „Create“, Visual Studio će pokrenuti izradu našeg projekta i naše forme. Nakon što je projekt kreiran pojaviti će se ekran na kojem ćemo vidjeti izgled naše forme, koja je trenutno prazna, te izgled grafičkog sučelja Visual Studio programa u kojem ćemo dalje raditi.



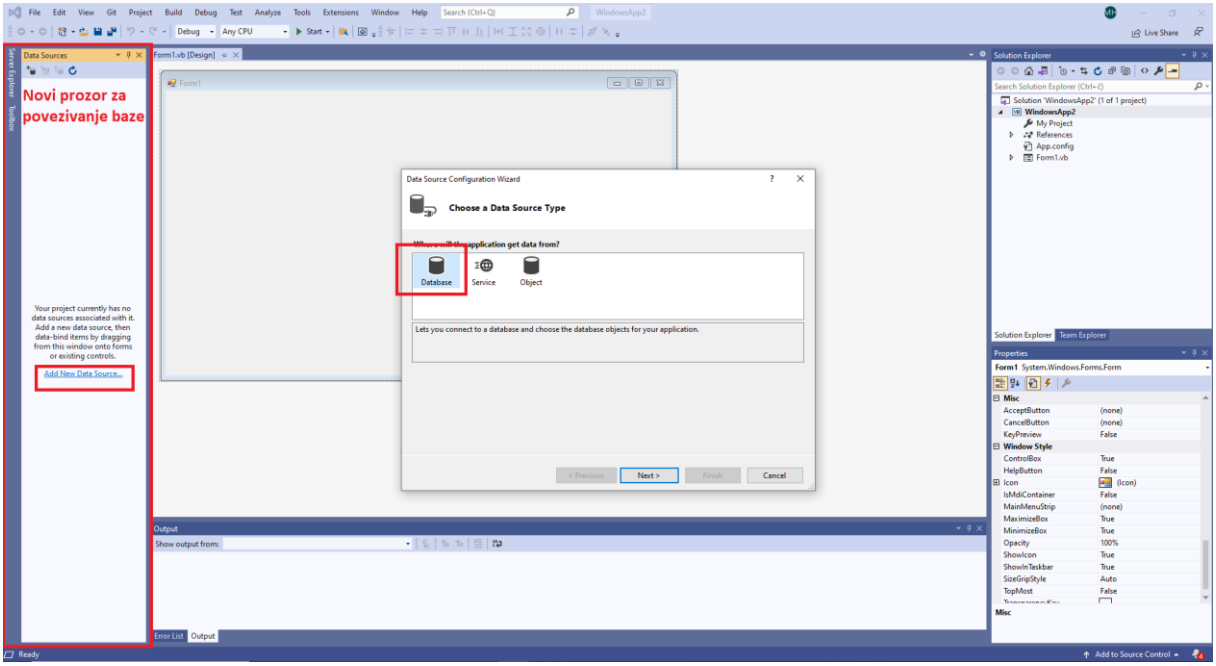
Slika 4. Izgled forme i sučelja VSa

Unutar forme koja je prikazana na slici 4. dodavat ćemo zapise koji se nalaze unutar naše baze podataka. Kako bismo mogli dodavati te podatke, najprije moramo bazu podataka povezati s našim projektom. Povezivanje baze s projektom radi se na sljedeći način: prvo uključimo prozor koji nam omogućava dodavanje novog resursa podataka. To ćemo učiniti tako da u glavnom izborniku Visual Studia odaberemo padajući izbornik „View“ te unutar tog izbornika odaberemo „Other windows“ i unutar njega „Data Sources“. [5.]

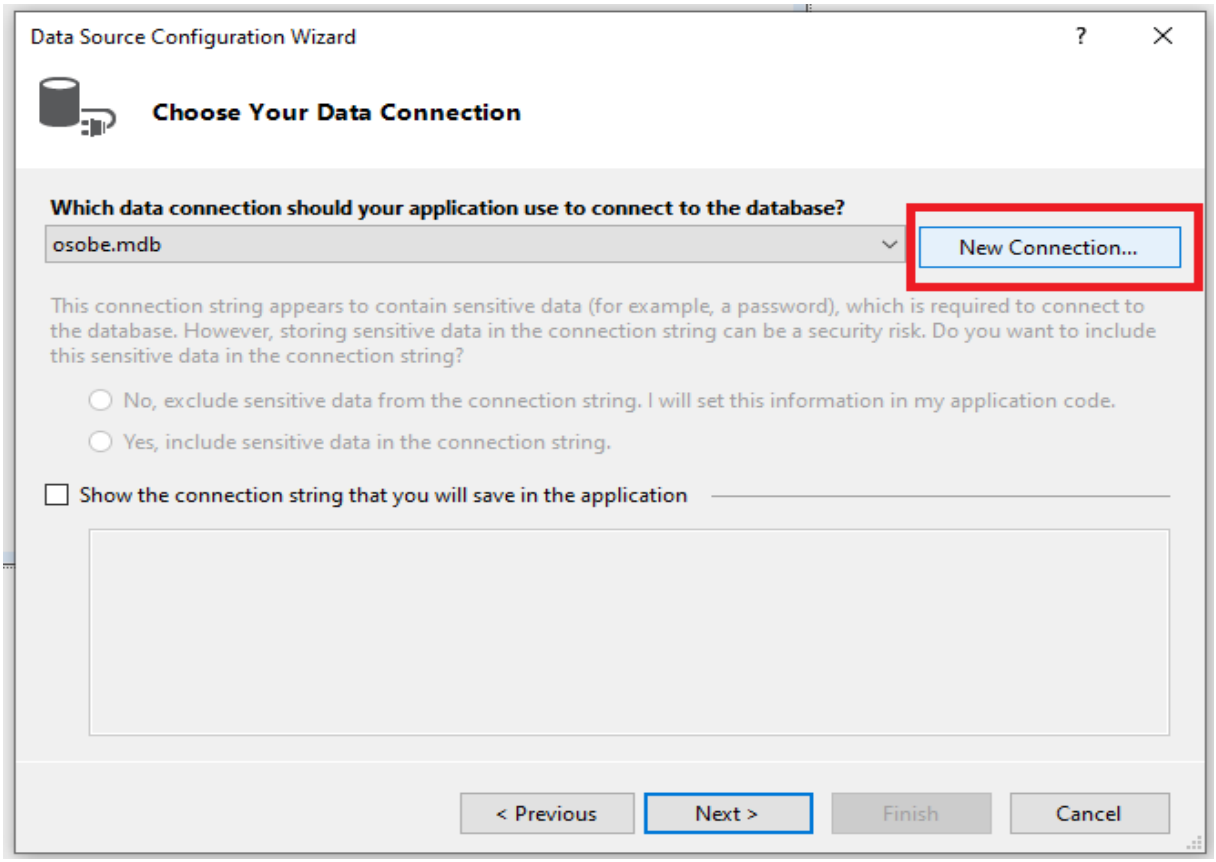


Slika 6. Odabir prozora za dodavanje podataka iz baze

Sljedeći korak povezivanja baze je taj da na novo otvorenom prozoru odaberemo „Add new data source...“. Sljedeće što nam se pojavljuje na ekranu je odabir vrste izvora podataka, gdje odabiremo „Database“ (bazu podataka), te nas čarobnjak za izradu veze vodi do odabira baze podataka koju želimo povezati. (Slika 5. i slika 6.) [5.]

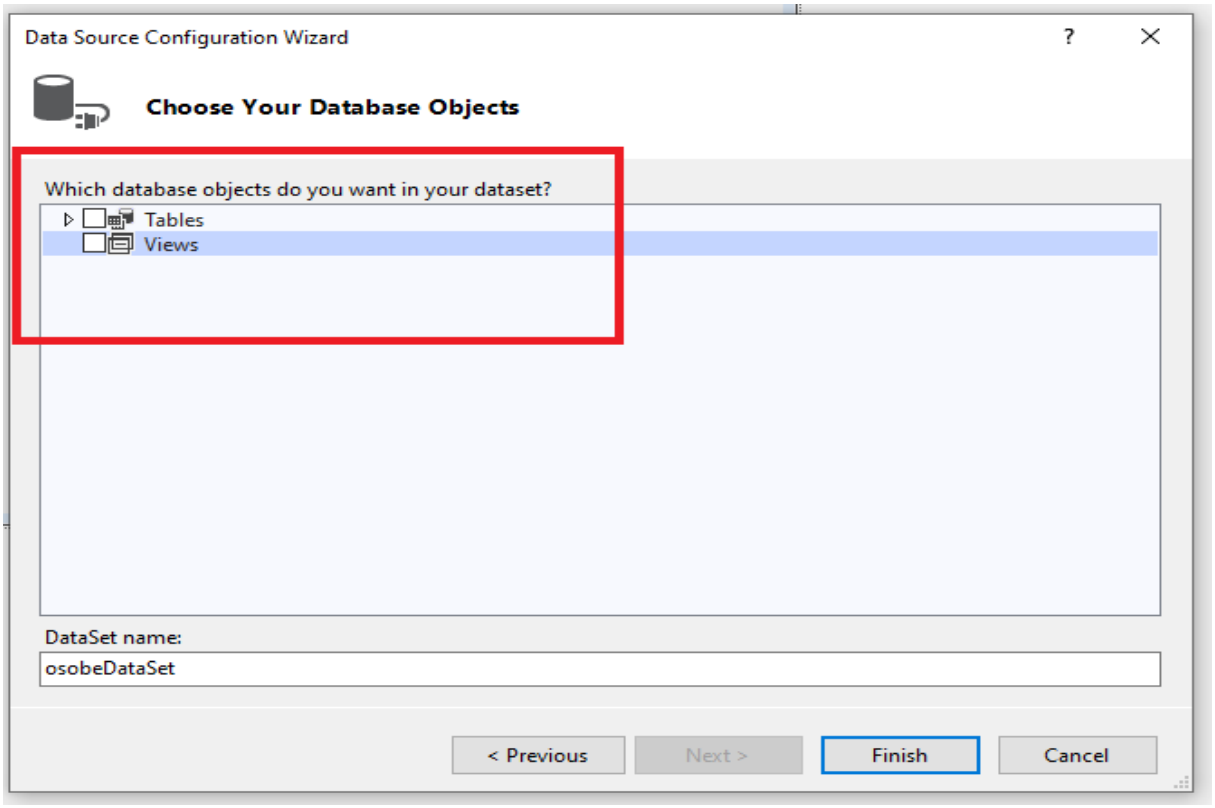


Slika 5. Dodavanje novog izvora podataka



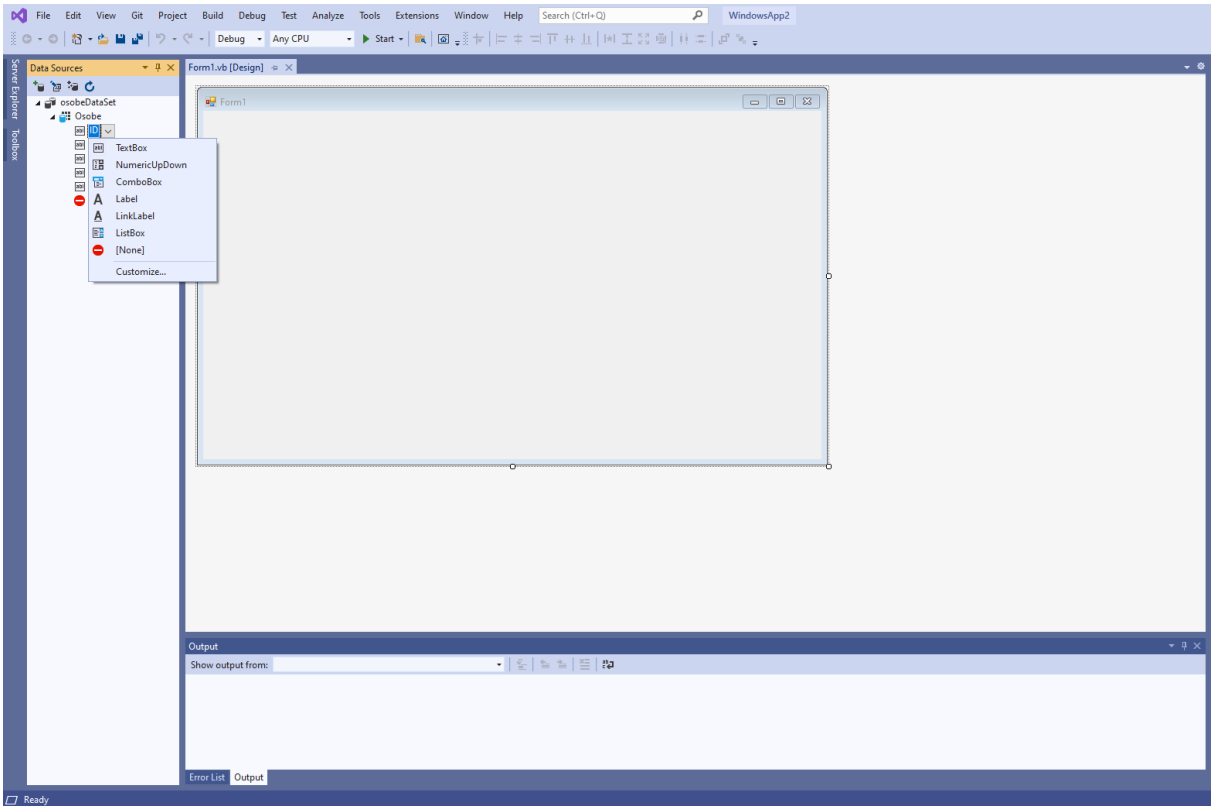
Slika 7. Odabir baze podataka

Nakon odabira baze s kojom se želimo povezati, i dođemo do kraja čarobnjaka za



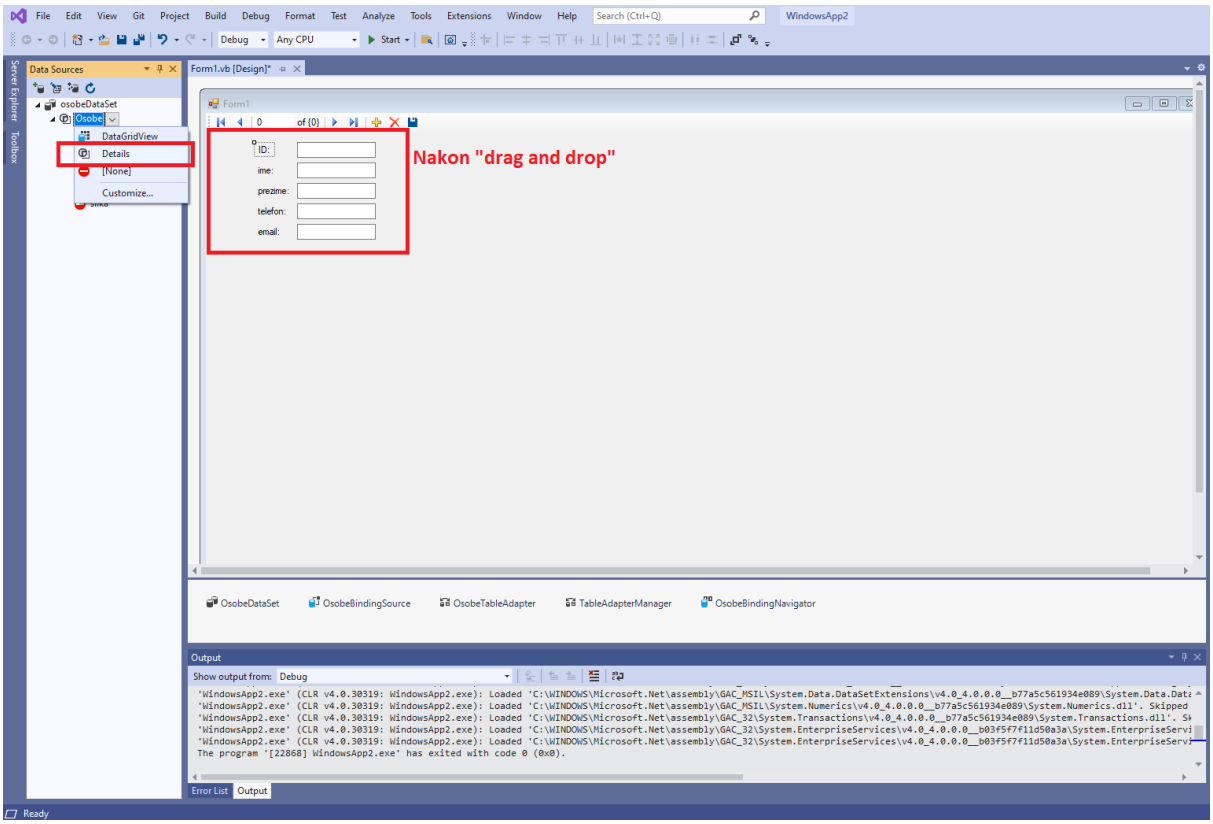
izradu veze, moramo odabrati što sve želimo uključiti , tj. koje sve objekte želimo prikazati.

U našem konkretnom slučaju odabrat ćemo tablice. Kada završimo s izradom i povezivanjem baze podataka sa našim projektom, u prijašnjem novootvorenom prozoru gdje smo odabrali da želimo dodati novi izvor podataka, otvoriti će nam se podaci naše baze te tablice koje smo odabrali da želimo vidjeti. Kada detaljnije pogledamo te podatke, možemo vidjeti da je naša baza podataka prikazano detaljno, što znači da imamo prikazani svaki dio tablice, svaki stupac koji smo napravili, te vrstu izgleda kako će ti podaci biti prikazani na formi.



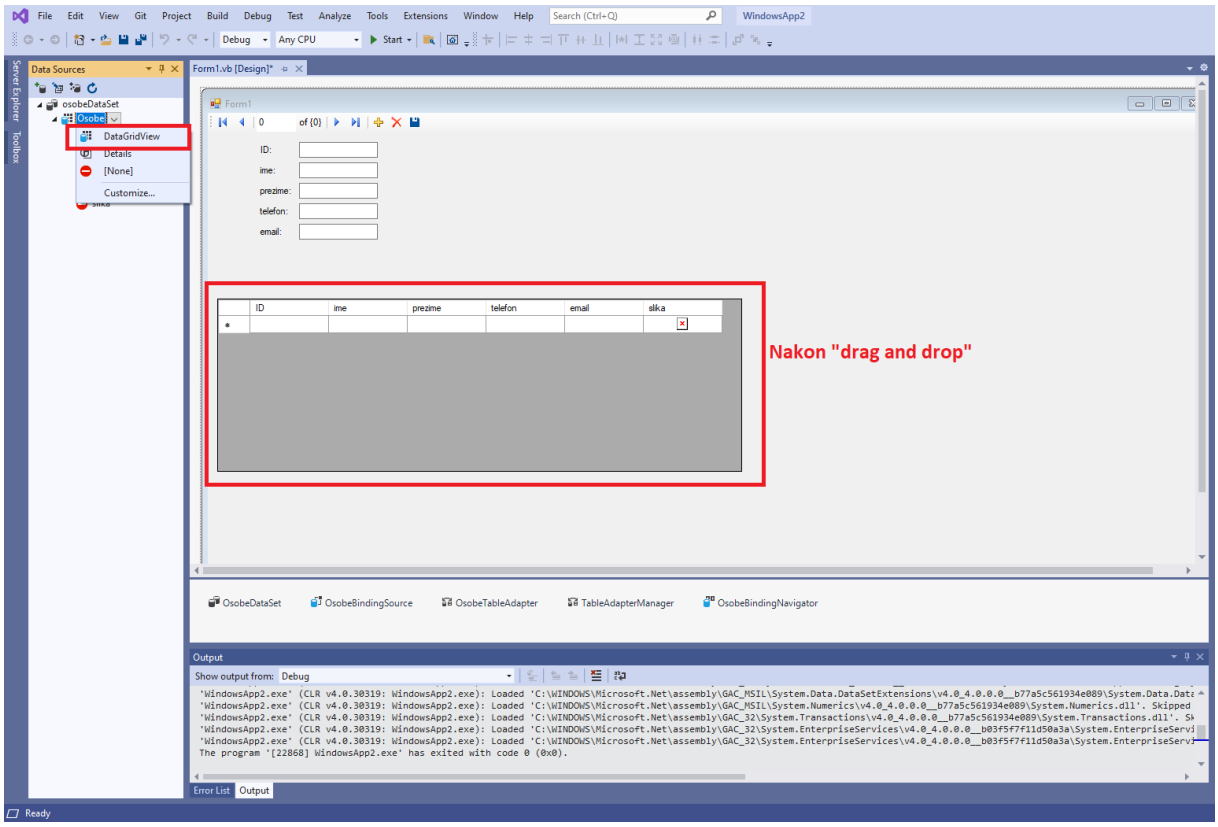
Slika 8. Prikaz tablice u VSu

Ako podatke na našoj formi želimo vidjet i pretraživati pojedinačno, onda ćemo u prozoru „Data Sources“ odabrati naziv naše tablice koju želimo prikazati, otvoriti padajući izbornik te odabrati „Details“. Nakon toga vrlo je lako dodati podatke na formu, i to se radi pomoću „Drag and drop“ metode, tj. prikvačimo mišom naziv tablice koju želimo dodati na formu te ju samo povučemo na formu. [5.]



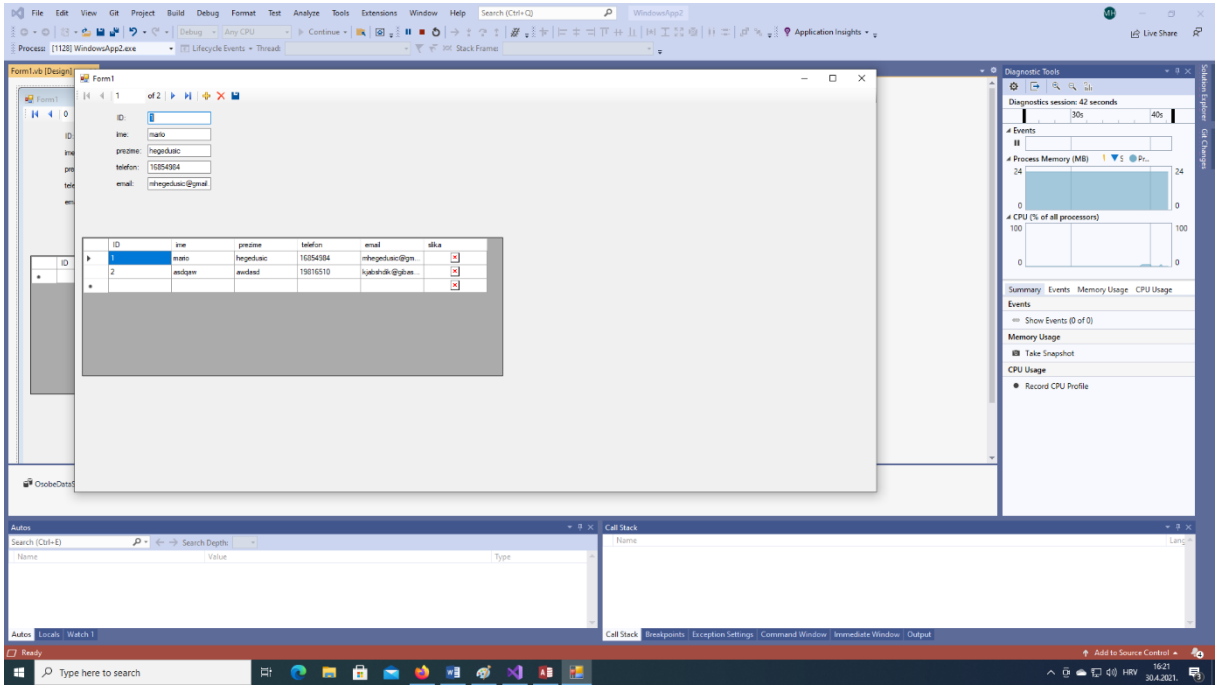
Slika 9. Stavljanje podataka u formu

Sada ćemo na istu ovu formu dodati te podatke prikazane u mreži i ti podaci biti će svi prikazani od jednom. To ćemo napraviti tako da iz padajućeg izbornika, pritiskom na tablicu koju želimo dodati, odaberemo „Data grid view“ te opet uz pomoć „Drag and drop“ metode dovučemo tablicu na formu. Nakon toga pokrenuti ćemo formu i pokazati kako će se podaci prikazivati na formi. (Slika 10. i slika 11.) [6.]



Slika 11. Izgled forme pomoću mreže

Pritiskom na tipku F5 ili na gumb Start u sučelju Visual Studia pokrećemo ovaj projekt.

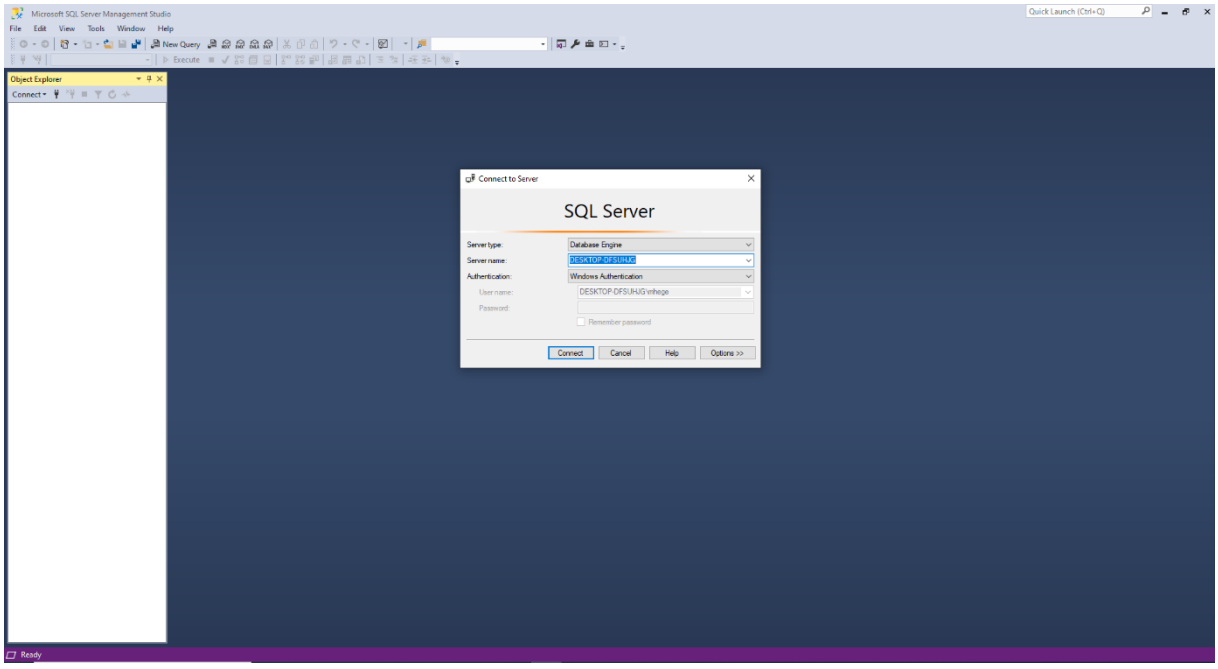


Slika 10. Pokrenuti projekt

Također ovdje još možemo vidjet kako nam se dodaje i navigacijski dio u formu. On se dodaje automatski prilikom dodavanja zapisa na formu i pomoću njega možemo pretraživati zapise, dodavati nove, brisati postojeće te spremati promjene. Ako želimo vidjeti kako izgleda programski kod naše forme koju smo kreirali to možemo napraviti na način da pritisnemo tipku F7 ili sa desnim klikom miša pritisnemo unutar forme i odaberemo „View code“. [6.]

2.2. Metoda spajanja s lokalnom SQL bazom

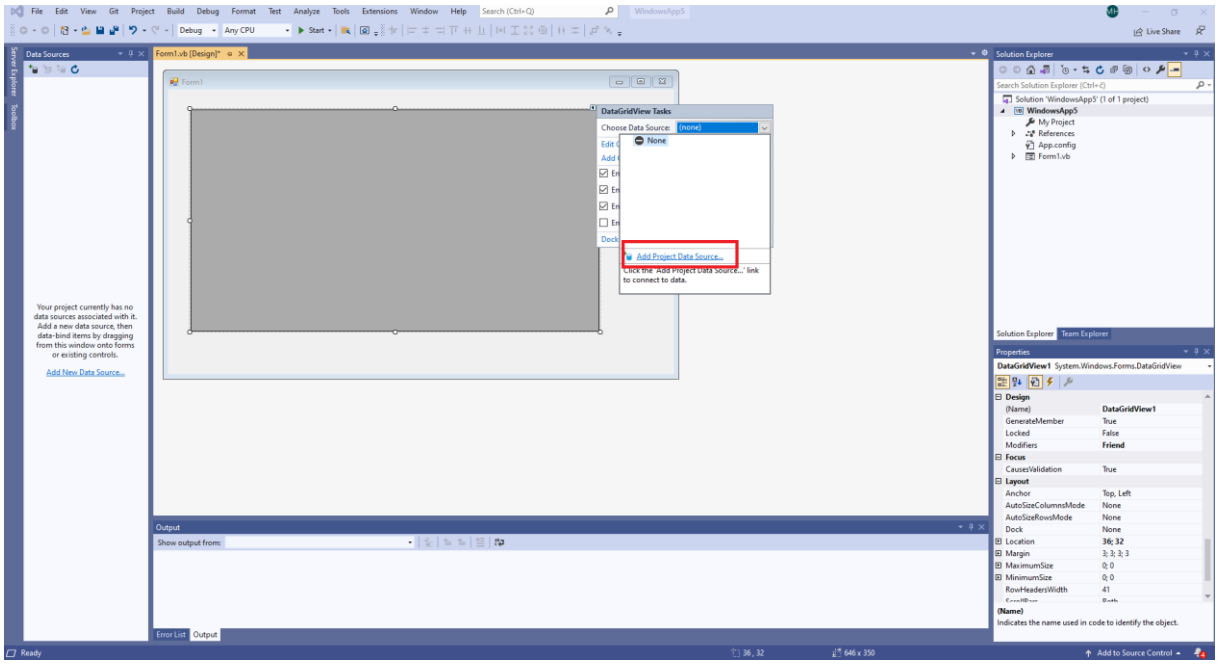
Sljedeća metoda spajanja koju ćemo pokazati u ovom radu je spajanje sa lokalnom SQL bazom podataka, tj. spajanje na lokalni SQL server na kojem se nalazi baza podataka čije podatke želimo vidjeti. Kako bismo to napravili prvo trebamo kreirati lokalni server i na njemu bazu podataka. Za to će nam trebati program Microsoft SQL Server Managment Studio. Kada otvorimo taj program, u ovom našem slučaju, kreirati ćemo i povezat ise na server koji odgovara imenu našeg računala, te ćemo se na taj server povezivati također kao što se povezujemo na svoje računalo, tj. s loginom od računala. [7.]



Slika 12. Postavljanje konfiguracije servera za bazu

Kada se povežemo na server, s desne strane ekrana pojaviti će nam se prozor s podacima o našem serveru, što sve postoji na serveru, koje baze podataka i slično. U tom prozoru pritisnuti ćemo desni klik na „Databases“ te odabrati da želimo kreirati novu bazu „New Database“. Sljedeći korak je imenovanje te baze podataka i njezino kreiranje. Kada smo kreirali

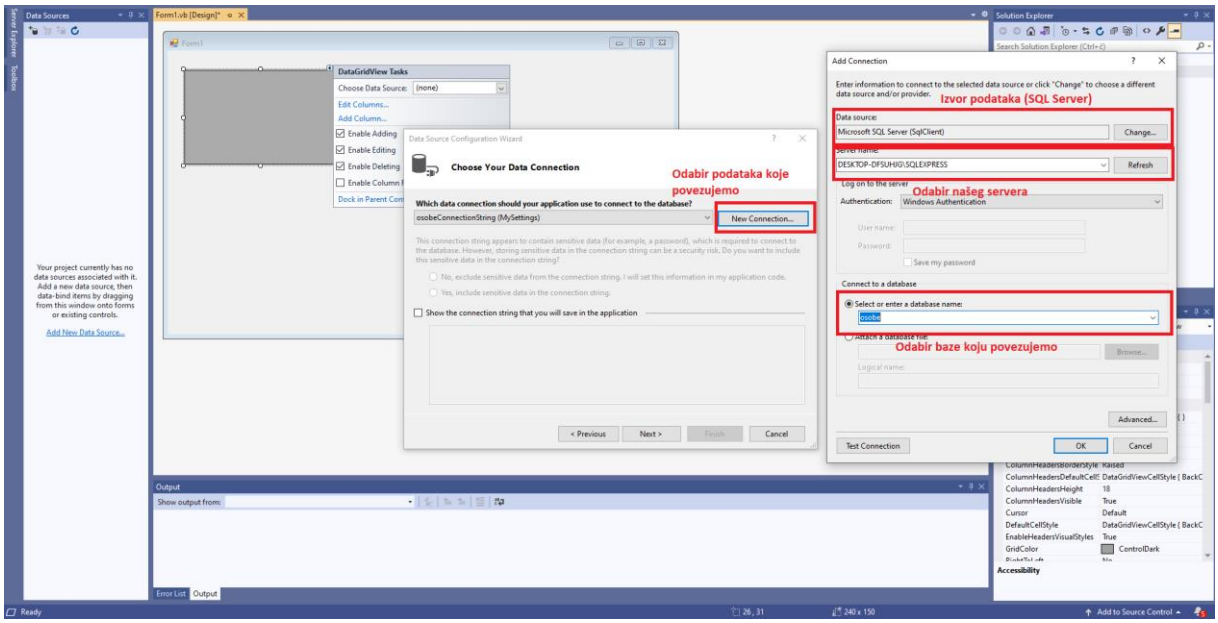
bazu, u nju moramo dodati neke tablice. Pa ćemo tako odabrati prvo bazu podataka koju smo kreirali te proširiti padajući izbornik iz kojega ćemo dalje odabrati „Tables“ i desnim klikom otvoriti izbornik u kojem odabiremo „New“ -> „Table“. Kada napravimo ovaj korak pojaviti će



Slika 13. Odabir izvora podataka

nam se novi prozor u kojem ćemo raditi stupce u tablici, tj. kreirat ćemo entitete u koje ćemo stavljati podatke. Kada kreiramo tablicu i njezine entitete sljedeći korak se događa unutar Visual Studio programa gdje moramo naš server i našu bazu povezati sa formom u koju ćemo dodati podatke. To ćemo napraviti tako da unutar forme, iz „Toolboxa“ dodamo alat „DataGridView“ te kod „Choose Data Source“ odaberemo „Add Project Data Source...“.

Sljedeće odabiremo „Database“ -> „Dataset“ te kada dođemo do odabira koje podatke želimo povezati odabiremo „New connection“ i pod padajućim izbornikom „Data source“ biramo „Microsoft SQL Server“. Znamo da smo prije toga napravili bazu podataka koja je spremljena na lokalnom serveru pa tako odabiremo ovaj izbor. Nadalje moramo odabrati ime servera, a to smo odabrali kada smo server kreirali te opet iz padajućeg izbornika odabiremo ponuđeni server, te iz zadnjeg padajućeg izbornika odabiremo koju bazu podataka želimo povezati iz tog našeg servera. [7.]



Slika 14. Biranje podataka, SQL servera i baze za povezivanje

Kada smo to napravili, na samom kraju čarobnjaka za povezivanje baze odabire tablice koje želimo prikazati na formi i završimo s izradom. Na našoj formi će se tada pojaviti svi stupci i redovi koje imamo spremljene u našoj tablici koju smo dodali. [7.]

2.3. ADO.NET objekti

ADO.NET je tehnologija koja je integrirana unutar .NET frameworka i omogućava nam dohvaćanje podataka i komunikaciju između različitih sustava. To je dakle skup komponenata koji programerima omogućava dohvaćanje podataka iz baza podataka. ADO.NET objekti odgovaraju operacijama koje se izvode prilikom pokretanja aplikacije koja dohvaća podatke iz baze. Prilikom pokretanja te aplikacije izvodi se sljedeće:

1. Ostvaruje se veza na bazu
2. Izvršavaju se naredbe nad bazom podataka
3. Mapiraju se svojstva baze podataka nad podacima
4. Pohranjuju se rezultati

Osnovni objekti ADO.NET arhitekture odgovaraju tim operacijama i sukladno tome tako su i dobili naziv Connection, Commad, i DataAdapter, a DataSet i DataReader oboje pohranjuju rezultate. Postoje dvije mogućnosti kada se želimo povezati na bazu, a to su korištenje namespacea System.Data.Sql.Client ili namespacea System.Data.OleDb, ovisno s kakvom bazom se želimo povezati. Ako ćemo koristiti bazu koja je napravljena preko SQL servera, onda ćemo koristiti namespace System.Data.Sql.Client zbog toga što su njegovi objekti

optimizirani za SQL baze, a ako ćemo koristiti baze koje su napravljenje u MS Access ili Oracle programima onda ćemo koristiti namespace System.Data.OleDb.

ADO.NET connection object predstavlja konekciju na bazu podataka. Nakon što se napravi objekt povezivanja, koristimo ConnectionString svojstvo kako bismo pružatelju podataka pokazali da želimo koristiti i zatim i pozvati metodu naziva Call() kako bismo napokon i ostvarili vezu na server. Postoje dvije mogućnosti kada radimo objekt za povezivanje. To možemo napraviti tako da iskoristimo već gotove vizualne alati koje nam nudi Visual Studio i SQL Server, ili možemo to napravili programski preko Connection Object.

Sljedeće što ćemo spomenuti je Command object. Ovaj objekt služi nam kako bismo izvršavali neke radnje i naredbe nad bazom na koju smo se prethodno povezali. Visual Studio nam omogućava da spremimo svoje naredbe ili blokove naredbi kada radimo Command object. On koristi graditelj upita koji je vrlo sličan onome u MS Accessu. Također ovdje imamo i pohranjene procedure, ato su aplikacije koje su napisane u programskom jeziku SQL Servera, T-SQL. Često u sebi sadrže neke argumente koji su poznati i kao parametri. Kada radimo sa više tablica, ovakav rad sa pohranom nekih procedura koje će nam trebati omogućavaju brže i lakše provođenje nekih radnji nad samom bazom podataka.

DataAdapterObject je sljedeći objekt kojeg ćemo upoznati. Kako bismo izvršili SQL naredbe ili već od prije pohranjene procedure, moramo najprije uspostaviti Command i DataAdapter objekte. Predaje se Command object SQL naredbi ili već prije pohranjenoj proceduri zajedno sa argumentima koji su nam potrebni za izvođenje. Ovdje još moramo spomenuti i metodu Fill() koja nam omogućava dohvaćanje podataka.

DataReader object je najvažniji dio ADO.NETa. On nam omogućava rad sa vrlo kompleksnim podacima i nije ograničen samo na forme podataka koje su pohranjene i spremljene sa tabularnim razmakom kao što je radio RecordSet. Odlika DataReader objecta je ta da se može direktno pristupiti bilo kojem retku želimo u tablici koristeći naredbu MoveNext(). Također DataReader samo prikazuje podatke, a ne dohvaća ih iz memorije kao što to radi DataSet.

Za kraj nam je ostao i DataSet object koji radi na principu unutarnje memorije baze podataka. DataSet object omogućava nam jaču manipulaciju podacima, ali ako koristimo vrlo veliku količinu podataka, za njega nam je potrebna i velika količina računalnih resursa, pa se stoga za veće količine podataka preporučuje korištenje DataReader objecta. [1.]

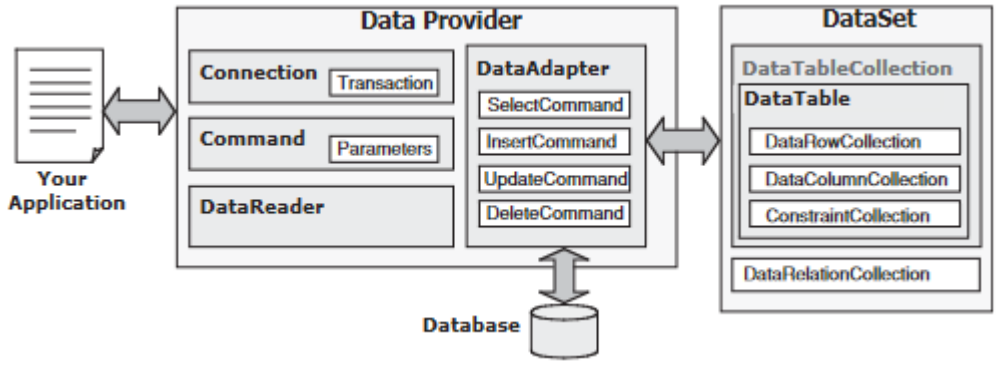
3. .NET Framework

.NET framework je najbitnija komponenta programa i programskog jezika koji smo koristili kako bismo preko Visual Studio programa povezivali svoje baze na aplikaciju. To je okruženje koje pri pokretanju izvršavanja aplikacije omogućuje aplikaciji da locira .NET framework na našem sustavu. Većinom sadrži uobičajene programske jezike koji nam pružaju lakše upravljanje memorijom i ostalim servisima našeg sustava, te opsežnu klasu biblioteka koja omogućava programeri prednost nad kodovima koji su robusni i pouzdani za sve velike grane razvoja aplikacija. Posljednja verzija .NET frameworka je 4.8 koja je startala od 18.4.2019. godine. Ovaj framework je servisiran svaki mjesec pa se tako svaki mjesec provjeravaju i ispravljaju njegove mane i nedostatci kao i njegova sigurnost.

Postoje dvije vrste uporabe .NET frameworka, a to su: .NET framework za korisnike i .NET framework za developere (programere). Ako ne programiramo neke aplikacije i ne pišemo kod, ali koristimo aplikacije koje zahtijevaju .NET framework potrebno ga je samo instalirati na svoje računalo kako bismo mogli pokretati takve aplikacije. Većina aplikacija koja zahtijeva ovaj framework, kod svojih instalacija, zatražiti će od vas instalaciju istog, te vam ponuditi njegovu instalaciju uz instalaciju tog programa. Ako se pak to ne dogodi i program traži zasebnu instalaciju frameworka ili ažuriranje na najnoviju verziju, potrebno je otići na web stranicu od .NET frameworka te ga preuzeti i instalirati samostalno. Većina novijih aplikacija i igara traži instalirani .NET framework na vašem računalo kako biste te iste aplikacije uopće mogli i pokrenuti. Različite verzije .NET frameworka mogu postojati na istom računalu u isto vrijeme, što znači da nije potrebno deinstalirati prethodne verzije kada želimo instalirati novu. S druge strane ako ste programer i želite koristiti ovaj framework za razvoj aplikacija, izaberite bilo koji jezik koji podržava .NET framework za kreiranje svojih aplikacija. Ovaj framework pruža nam neovisnost i interoperabilnost između programskih jezika, stoga možemo međusobno komunicirati i djelovati s drugim .NET framework aplikacijama i komponenatama bez obzira u kojem jeziku su one napisane. Kako bismo mogli programirati aplikacije sa ovim frameworkom prvo što je potrebno je instalirati verziju frameworka koju će naša aplikacija zahtijevati. Najbolje je uvijek instalirati zadnju verziju koja nam se pruža zato što većinom aplikacije u današnje vrijeme koriste najnovije značajke. Sljedeće bismo trebali odabrati jezik u kojem želimo isprogramirati svoju aplikaciju koju također taj framework podržava. Veliki broj programskim jezika je podržan što uključuje i Visual Basic. C#, F# te C++ od Microsofta. Zadnje što moramo napraviti je odabrati i instalirati okruženje u kojem ćemo raditi i pisati svoj programski kod za razvoj aplikacije koji podržava programski jezik koji odaberemo. Microsoftovo integrirano okruženje za razvoj aplikacija u .NET frameworku je aplikacija Visual

Studio. Ona je dostupna u više izdanja. Također valja napomenuti kako stariji operacijski sustavi neće podržati novije verzije ovog frameworka, a isto tako on ima i svoje zahtjeve koje je potrebno ispuniti u obliku sistemskim specifikacija poput brzine procesora, veličine RAM memorije te dostupno mjesto na hard disku. [9.]

4. Klase povezivanja



Slika 15. Osnovna arhitektura ADO.NET

Na slici 15. možemo vidjeti kako izgledati osnovna arhitektura ADO.NETa. U ovom poglavlju osvrnuti ćemo se na Data Provider komponentu. Ona nam omogućava da povežemo svoje izvore podataka sa svojom aplikacijom, izvršimo različite metode kako bismo izveli upite i operacije dohvaćanja podataka između izvora podataka i aplikacije te odspajanje s izvorom podataka kada su sve operacije završene. Spomenuti ćemo i popularne verzije Data Providera, a one su: Open DataBase Connectivity (Odbc) Data Provider (ODBC.NET), Open Linking and Embeding DataBase (OleDb) Data Provider (OLEDB.net), SQL Server (Sql) Data Providet (SQL Server.NET), Oracle (Oracle) Data Provider (Oracle.NET). Svaki ovaj Data Provider može se pojednostavljeno zvati sa ključnim riječima, koje se nalaze napisane unutar prvih zagrada. Ako se koriste stariji Data Provideri moramo na kraj svake ključne riječi također dodati i sufiks .NET. [2.]

4.1. Connection class

Connection class je klasa koja nam pruža konekciju između naše aplikacije i baze podataka koju odaberemo. Kako bismo mogli koristiti ovu klasu za uspostavljanje konekcije, prvo moramo kreirati instancu ili objekt koji je temeljen na ovoj klasi. Može se kreirati globalna

instanca za konekciju za cijeli projekt ili samo lokalni objekti. Preporuča se globalna instanca za jednostavnije projekte. Data Provider pruža nam 4 različite connection class klase koje se razlikuju po tome koju vrstu baze podataka spajamo. Postoji i pojam koji se naziva connection string i on nam služi kako bismo unutar njega upisali sve važne informacije koje su nam potrebne za spajanje na bazu. On sadrži nekoliko parametara, a najvažniji su nam: davatelj usluge(provider), izvor podataka(data source), baza podataka(database), korisničko ime za spajanje(user ID), lozinka(password). Za različite baze podataka parametri su različiti, npr. za baze na SQL serveru umjesto davatelja usluge(provider) unositi ćemo riječ server. Svi parametri koji se unose osjetljivi su na velika i mala slova stoga moramo paziti kada unosimo imena servera, baze i slično. Tipična instanca konekcije može izgledati ovako:

Connection = NewxxxConnection("Provider = MyProvider;" &"Data Source = MyServer;" &"Database = MyDatabase;" &"User ID = MyUserID;" &"Password = MyPassWord;")

Na mjestu gdje piše „xxx" dolazi ime davatelja usluge koji ovisi o tome na kakvu se bazu spajamo. To može biti OleDb, Sql, Oracle. MyServer mijenjamo sa imenom svojeg servera, MyDatabase sa imenom baze na koju se želimo spojiti, MyUserID je korisničko ime s kojim se spajamo te MyPassWord je lozinka s kojom se spajamo.

OLE DB Data Provider for Microsoft Access Database:

```
Connection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" & _
                                 "Data Source=C:\database\CSE_DEPT.mdb;" & _
                                 "User ID=MyUserID;" & _
                                 "Password=MyPassWord;")
```

SQL Server Data Provider for SQL Server Database:

```
Connection = New SqlConnection("Server=localhost;" +_
                               "Data Source=Susan\SQLEXPRESS;" + _
                               "Database=CSE_DEPT;" + _
                               "Integrated Security=SSPI")
```

Oracle Data Provider for Oracle Database:

```
Connection = New OracleConnection("Data Source=XE;"+_
                                  "User ID=system;"+_
                                  "Password=reback")
```

Slika 16. Tipične instance spajanja za različite baze

Pored važnih svojstava connection class klase ona mora sadržavati i neke važne metode kao što su Open() i Close(). Open() metoda omogućava nam otvaranje konekcije sa izvorom podataka koji je određen sa postavkama svojstava u connection class. Najčešće se ova metoda koristi unutar petlje „Try...Catch“ kako bismo bili sigurni da smo se povezali na bazu i da je konekcija bila uspješna.

```
Dim strConnectionString As String = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
                                    "Data Source=C:\database\CSE_DEPT.mdb;"
accConnection = New OleDbConnection(strConnectionString)
Try
   accConnection.Open()
Catch OleDbExceptionErr As OleDbException
   MessageBox.Show(OleDbExceptionErr.Message, "Access Error")
Catch InvalidOperationExceptionErr As InvalidOperationException
   MessageBox.Show(InvalidOperationExceptionErr.Message,"Access Error")
End Try
If accConnection.State <> ConnectionState.Open Then
   MessageBox.Show("Database Connection is Failed")
   Exit Sub
End If
```

Slika 17. Pojednostavljeno otvaranje konekcije

Close() metoda je usko povezana sa Open() metodom i ona se koristi kada želimo zatvoriti konekciju koju smo prije otvorili. Ona se koristi kada napravimo sve željene radnje sa podacima unutar baze i ta konekcija se mora zatvoriti inače bismo mogli dobiti neke pogreške u daljnjem radu naše aplikacije ako bismo htjeli raditi neke nove konekcije. [2.]

5. LINQ u Visual Basic programu

Language-integrated query (LINQ) što u prijevodu znali „Jezično integrirani upiti“, dodaju mogućnosti stvaranja upita u Visual Basic jezik i daju nam jednostavne a vrlo moćne mogućnosti kada radimo s bilo kakvom vrstom podataka. Ovo nam uvelike pomaže u toj mjeri da upite možemo izvršavati unutar jezika Visual Basic umjesto da te iste upite izvršavamo unutar programa za rad s bazom podataka ili da koristimo drugačije sintakse za drugačije tipove podataka koje tražimo. On koristi ujedinjenu sintaksu neovisno o tipu podataka koji koristimo. LINQ nam omogućuje rad s upitima nam podacima koji dolaze iz različitih izvora, ato može biti: SQL server baza podataka, XML, polja koja se nalaze samo u memoriji, ADO.NET skupovi podataka ili bilo koji drugi podaci bili oni udaljeni s računala ili postavljeni lokalno na računalo i sve se to može odraditi sa uobičajenim elementima Visual Basic programa. Ako koristimo upite koji su pisani u Visual Basic programskom jeziku naši rezultati će biti prikazani i vraćeni kao strogo tipirani objekti. Takvi objekti omogućuju nam da brže i

lakše napišemo svoj kod te da otkrijemo svoje greške već unutar kompajliranja svojeg programskog koda umjesto pri pokretanju aplikacije. [10.]

5.1. LINQ pružatelji usluga

LINQ pružatelj usluga,naše upite postavljene u Visual Basic jeziku, mapira u izvor podataka nad kojim radimo upit. Kada pišemo LINQ upit, pružatelj te usluge nam prevodi taj upit u naredbe koje će naš izvor podataka prepoznati i koje će moći izvršiti. Također on će za nas i pretvoriti vrstu podataka iz izvora podataka na objekte koji će nam dati finalne rezultate našeg postavljenog upita. Na kraju pretvara objekte u podatke kada se šalju ažurirani podaci našem izvoru podataka.

Visual Basic uključuje sljedeće LINQ pružatelje usluga:

LINQ na objekte – ovaj pružatelj usluga omogućuje nam upite vezane uz kolekcije i nizove koji se nalaze u memoriji. Ako nam objekt podržava Ienumerable sučelje (razotkriva enumeratora koji podržava jednostavnu iteraciju nad zbirkom koja nije generička) ili Ienumerable<T> sučelje (razotkriva enumerator koji podržava jednostavnu iteraciju nad kolekcijom nekog određenog tipa).

LINQ na SQL – ovaj pružatelj usluga omogućuje nam izradu upita i modificiranje podataka u SQL Server bazi podataka. Visual Basic olakšava rad sa LINQ na SQL sa uključivanjem Dizajnera Relacijskih Objekata (O/R Designer).

LINQ na XML – ovaj pružatelj usluga omogućuje nam kreiranje i izvršavanje upita i modificiranje podataka XMLa. XML podaci mogu se mijenjati kada su pohranjeni u memoriji, a isto tako mogu se učitati te onda promijeniti i spremiti u datoteku. Dodatno, ovaj pružatelj omogućava nam značajke XML literale i XML os koji nam dalje omogućavaju da pišemo svoj XML direktno u Visual Basic kodu.

LINQ na DataSet – ovaj pružatelj usluga omogućuje nam kreiranje i izvršavanje upita u ADO.NET skupu podataka. Sa ovim pružateljem usluga dobijemo mogućnost pojednostavljenja, a i proširenja svojih mogućnosti za kreiranje upita, skupljanje i ažuriranje podataka u našem skupu podataka. [10.]

5.2. Struktura LINQ upita

LINQ upit vrlo često se referira kao „upitni izraz“ koji se sastoji od kombinacije klauzule upita koja identificira izvor podataka i iteracije varijable za upit. Također izraz može uključivati

i razne druge mogućnosti kao što su sortiranje, filtriranje, grupiranje i spajanje, ili različite izračune koji se mogu onda primijeniti na izvorne podatke. Pronaći ćemo vrlo veliku sličnost sa upitima koji se postavljaju u sintaksi SQLa.

Izraz upita počinje sa klauzulom FROM koja identificira izvorne podatke koje ćemo koristiti u upitu te varijable koje su korištene da razlikujemo svaki element izvornih podataka. Te varijable nazvane su „iteracijske varijable“. Sada ćemo vidjeti neke klauzule koje se koriste unutar Visual Basic LINQ postavljenih upita te isto tako vidjeti i sličnost sa SQL upitima.

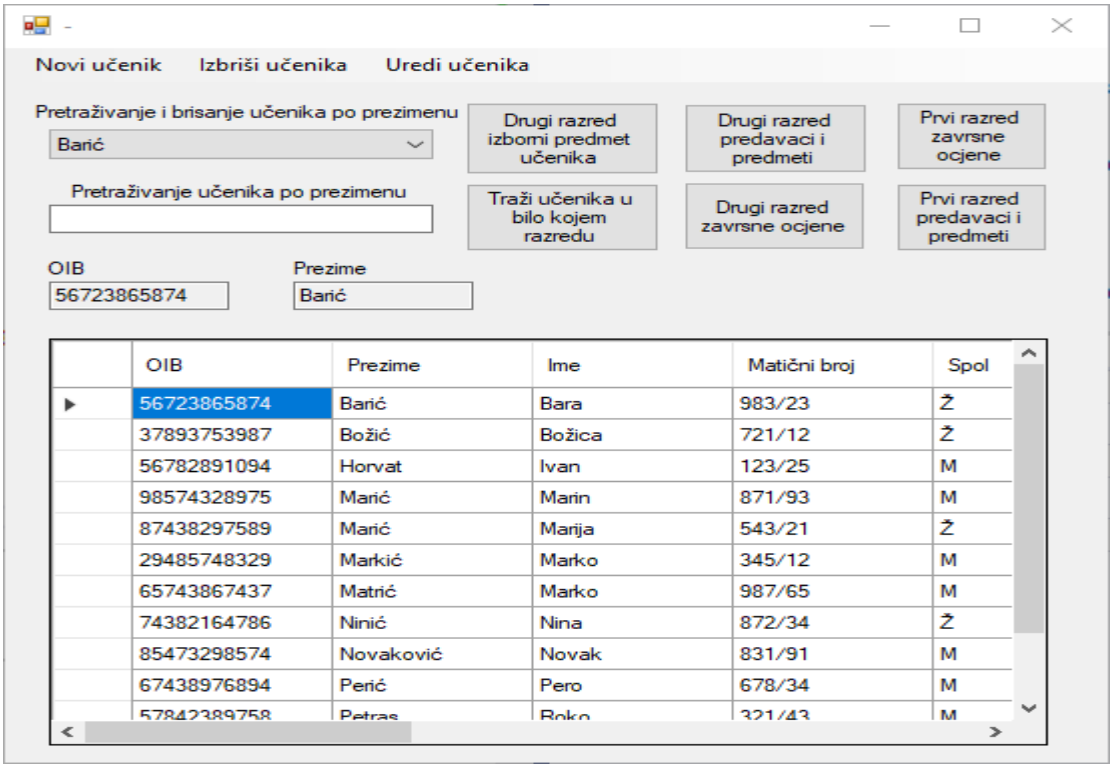
Prva klauzula koju čemo spomenuti, a i već smo je i spomenuli, je klauzula FROM koja nam je neophodna za kreiranje upita i njome počinje svaki upit kako bismo znali odakle tražimo podatke. Sljedeća klauzula koju ćemo spomenuti je SELECT i ona je opcionalna. Ona nam omogućuje da odaberemo podatke iz naših izvornih podataka. Možemo odabrati samo nekoliko podataka ili sve podatke. Dalje prelazimo na WHERE klauzulu koja je također opcionalna i ona nam omogućuje da filtriramo rezultate koje želimo prikazati, pa tako možemo odabrati samo rezultate koji počinju slovom A, ili podatke koji u sebi sadrže neki naziv ili neki podatak. Sljedeće klauzula je ORDER BY koja nam omogućuje sortiranje podataka, pa tako možemo sortirati svoje podatke prema bilo kojem stupcu ili kriteriju kojem želimo u svojim podacima. JOIN klauzula je također opcionalna te neobavezna i onda se koristi kada želimo kombinirati dvije kolekcije ili niza u jedan zajednički. GROUP BY kluzula omogućuje nam grupiranje elemenata upita u rezultatu, pa tako svoje rezultate možemo primjerice grupirati po datumu ili sl. zavisno o podacima koje imamo unutar svoje baze. AGGREGATE klauzula je obavezna ukoliko ne koristimo klauzulu FROM i njome se također započinje upit. Ova klauzula primjenjuje jednu ili više agregatnih funkcija nad kolekcijom. Primjerice ovu klauzulu možemo koristiti kako bismo izračunali sumu svih elemenata koji nam je upit vratio kao rezultat.

LET klauzula je opcionalna i ona izračunava neku vrijednost i nju dodjeljuje nekoj novoj varijabli unutar upita. DISTINCT klauzula je također opcionalna i ona ograničava vrijednosti trenutne iteracijske varijable kako bi se spriječile duple vrijednosti kod rezultata upita. SKIP klauzula preskače određeni broj elemenata koji mi odaberemo unutar niza i vraća ostali broj elemenata kao rezultat, ona je opcionalna. SKIP WHILE klauzula isto je opcionalna i onda preskače elemente niza dok god je uvjet unutar WHILE naredbe ispunjen, te tada vraća preostale elemente kao rezultat. TAKE clauzula je također opcionalna i vraća nam određeni broj sljednih elemenata od početka niza. TAKE WHILE klauzula radi na sličan način te je također opcionalna. Ona nam vraća podatke iz niza iz kojeg želimo toliko dugo dok je uvjet unutar WHILE naredbe istinit, a preostale elemente zanemaruje.

Ako želimo povezati bazu podataka koristeći LINQ na SQL, unutar Visual Basica identificiramo objekte SQL Server baze podataka, kao što su tablice, pogledi i spremljene

procedure kojima želimo pristupiti koristeći datoteku LINQ na SQL. Ta datoteka ima ekstenziju .dbml. Kada postoji valjana veza na SQL Server bazu podataka moežmo dodati preložak stavaka sa LINQ na SQL klasa. To će nam otvoriti dizajnera relacijskih objekata koji nam omogućuje da sa drag and drop metodom postavimo svoje podatke koje želimo prikazati na područje dizajnera,a podaci koje možemo prikazati bit će nam omogućeni na izborniku Server Explorer/Database Explorer. LINQ na SQL datoteka dodaje DataContext objekt na naš projekt koji nam omogućuje da vidimo svojstva svojih tablica i pogleda kojima želimo pristupiti i spremljene procedure koje želimo pozvati u svoj kod. Kada smo to napravili i kada smo spremili svoje promjere, ovim objektima možemo pristupiti bilo kada tako što ćemo se referencirati na DataContext objekt. Visual Basic također ima i neke značajke koje nam uvelike pomažu pri pojednostavljenju i smanjenju količine koda kojeg pišemo za izvršavanje upita. To uključuje sljedeće: **anonimni tipovi** koji nam omogućavaju kreiranje novog tipa koji je baziran na rezultati upita, **implicitne tipove varijabli** što nam omogućuje odgađanje određivanja vrste varijable i pušta kompajleru da zaključi tip podataka prema rezultatu upita, **metode proširenja** što nam omogućuje da proširimo već neki postojeći tip sa nekim svojim metodama bez modificiranja samog tipa. [10.]

6. Aplikacija

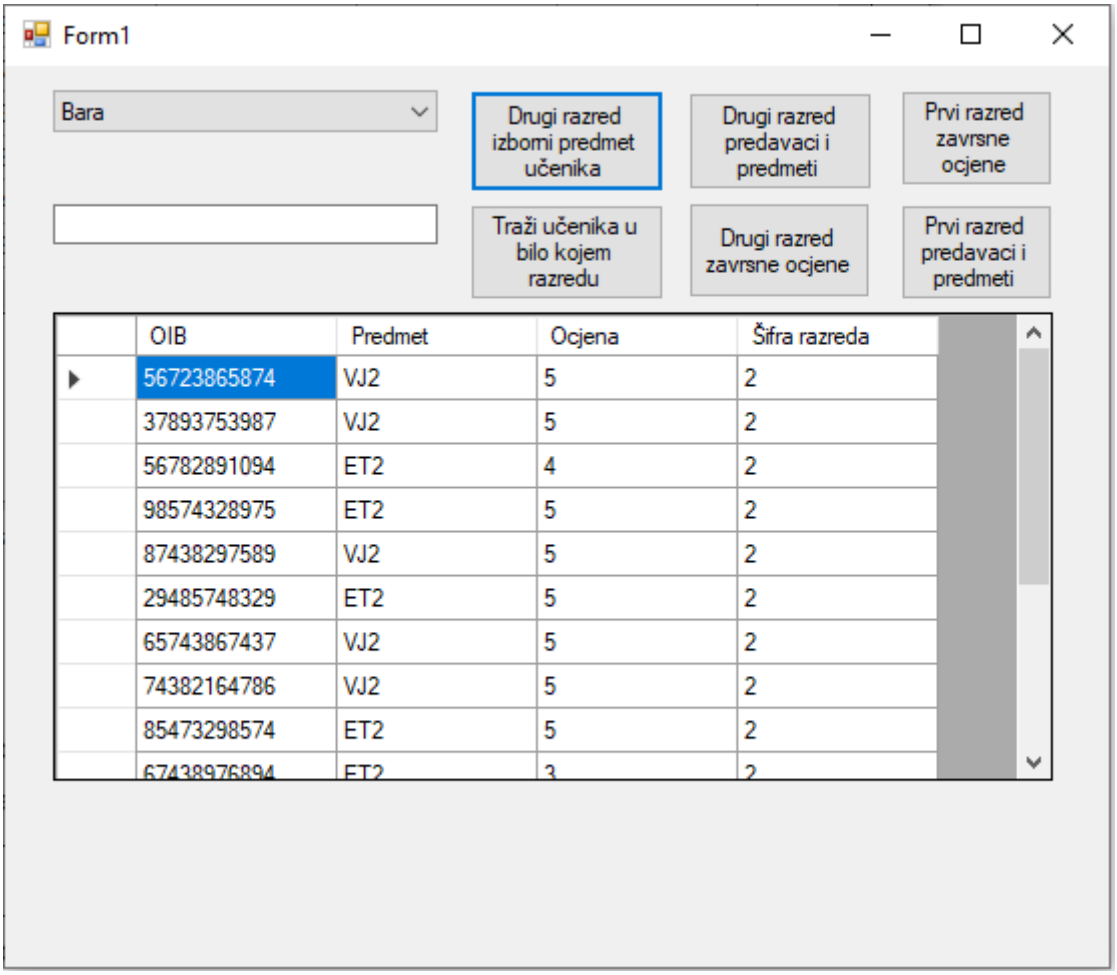


Slika 18.Izgled aplikacije pri pokretanju

U ovom poglavlju pokazat ću svoju aplikaciju napravljenu u Visual Basic.NET programu i okruženju pomoću baze podataka izrađene unutar MS Access te sve njeznie funkcionalnosti.

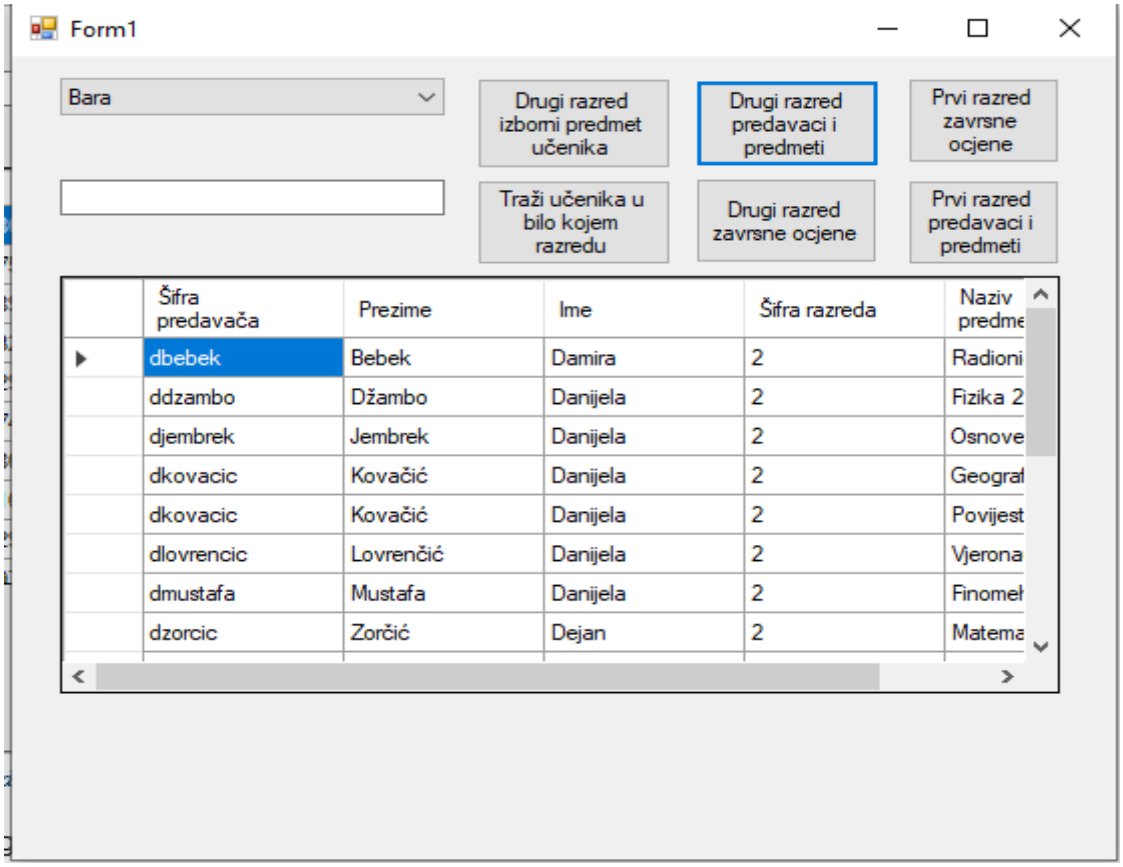
Baza je napravljena sa podacima o učenicima te predmetima, predavačima i ostalom što spada u školstvo.

Funkcionalnost ove aplikacije je da prikazuje podatke o učenicima iz tablica kreiranih u MS Access programu, te izrada različitih upita nad tablicama i prikaz njihovim rezultata također. Pomoću gumbova dodali smo jednostavnost aplikaciji pa se tako kroz njih pokazuju podaci koji se dobivaju izvršavanjem SQL upita i prikazuju na DataGridView prikazu. Također postoji jedna funkcionalost kojom možemo potražiti bilo kojeg učenika u sustavu pomoću gumba „Traži učenika u bilo kojem razredu“ i „search bara“ pored njega.

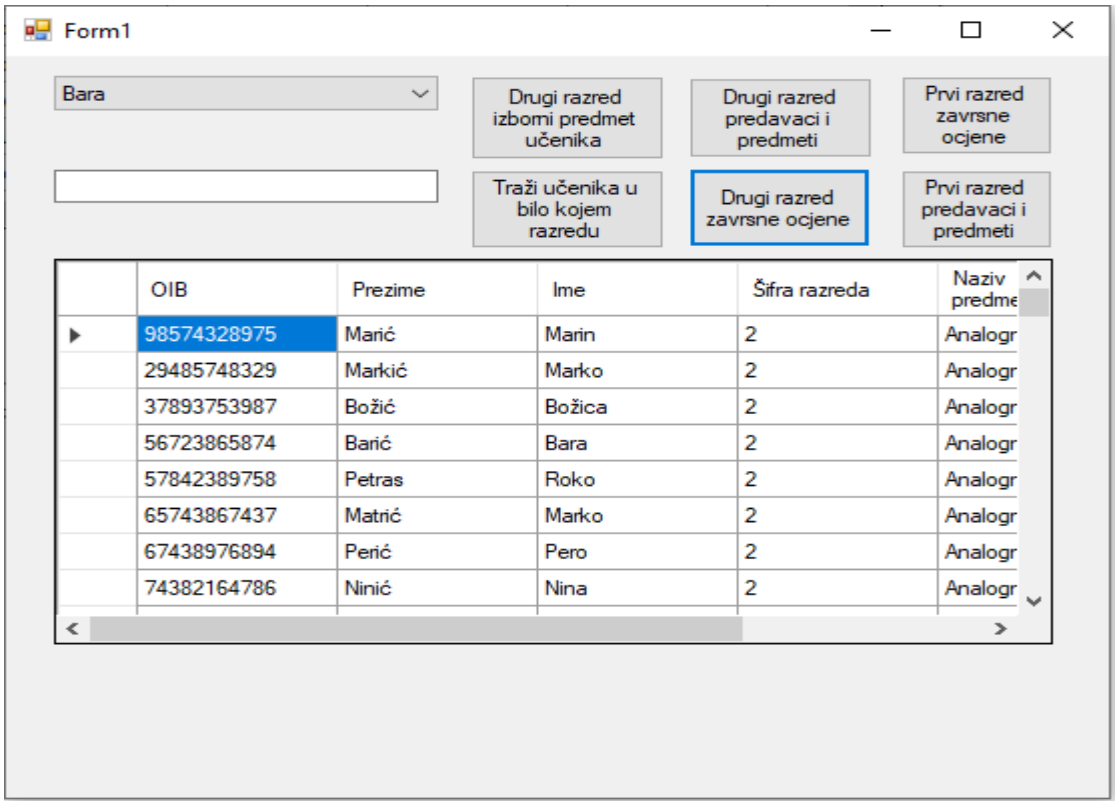


Slika 19. Drugi razred izborni predmet

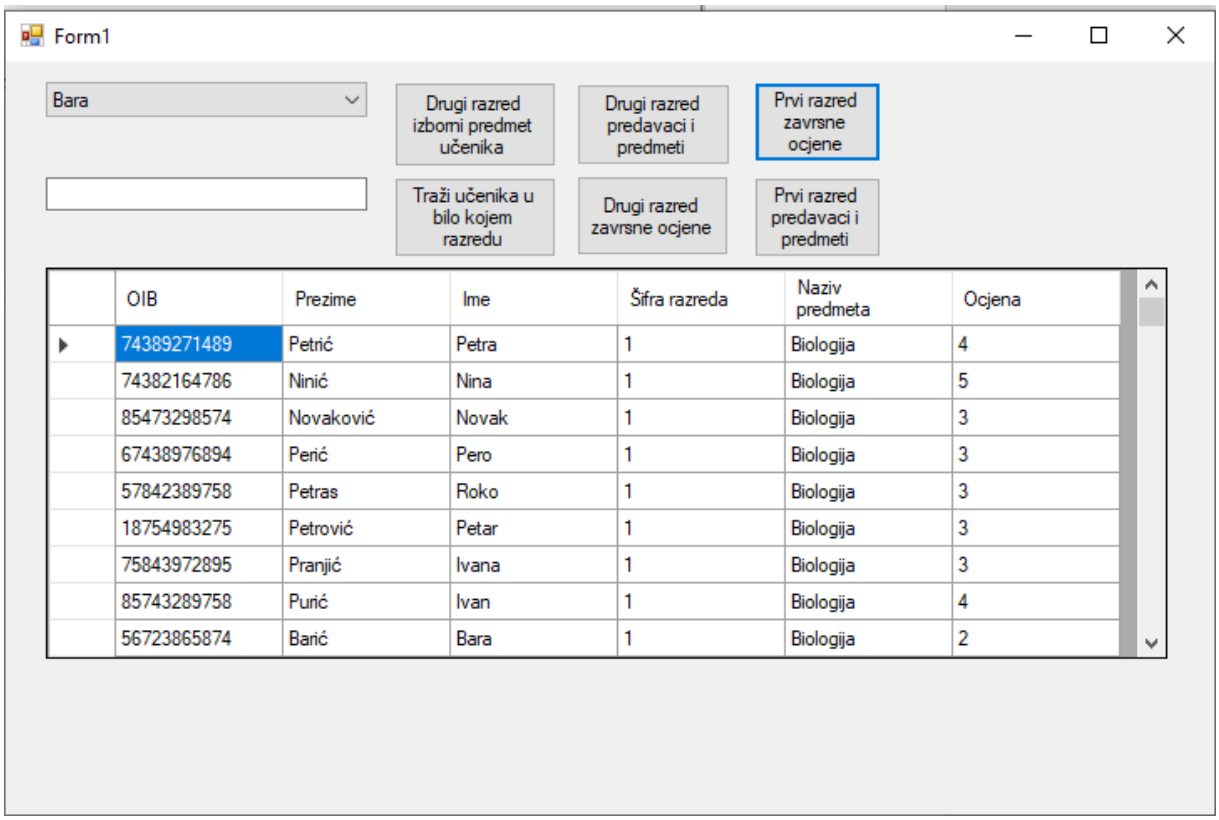
Pritiskom na gumb „Drugi razred izborni predmet učenika“ dobijemo prikaz svih učenika preko njihovih OIBa, te koju ocjenu oni imaju iz kojeg izbornog predmeta. Također to smo napravili i sa prvim razredom i njihovim učenicima. Pritiskom na „Drugi razred predavaci i predmeti“ dobijemo uvid u sve predavače i predmete koji se izvode u drugom razredu, tako i za prvo razred na sljedeći gumb itd. Sada ćemo sve to pokazati preko slika.



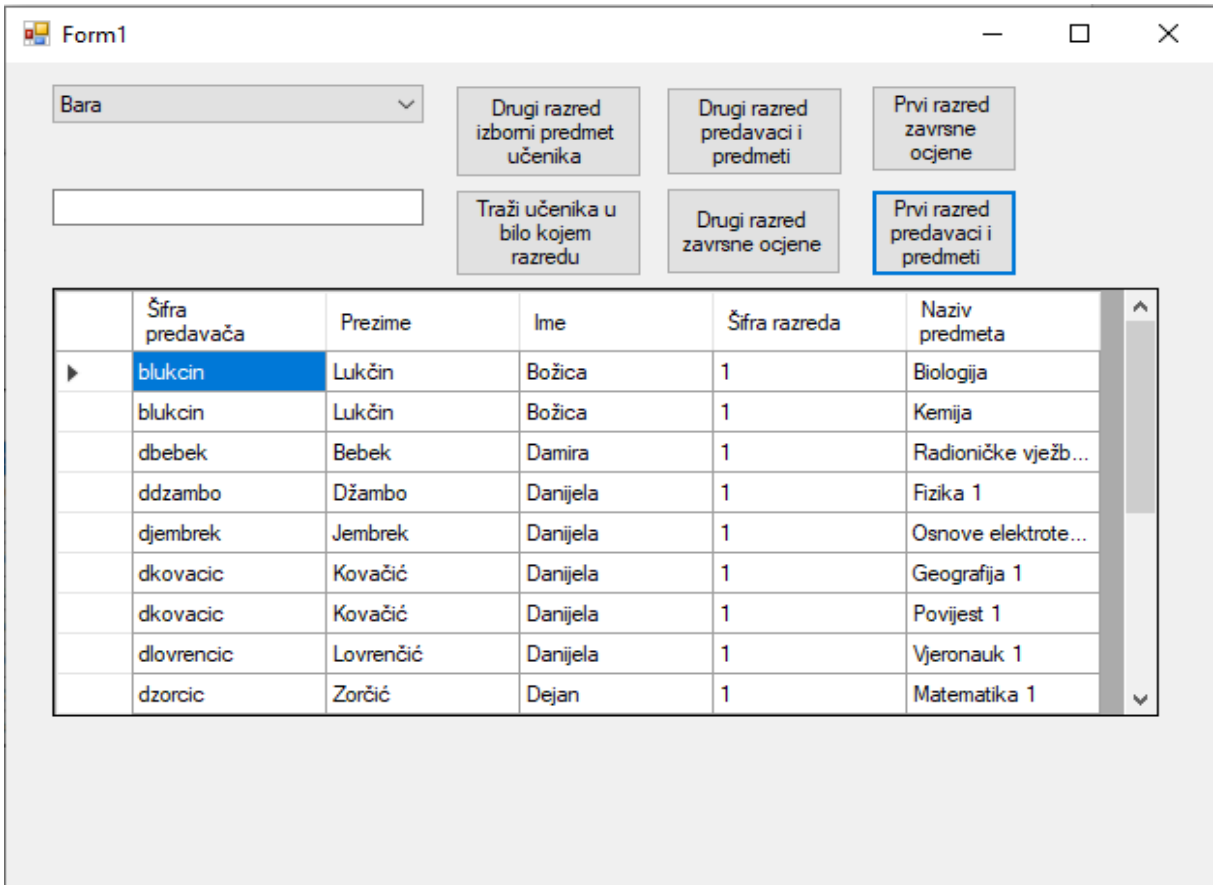
Slika 20. Drugi razred predavači i predmeti



Slika 21. Drugi razred završne ocjene



Slika 22. Prvi razred završne ocjene



Slika 23. Prvi razred predavači i predmeti

Prethodno smo prikazali sučelje i funkcionalnost svih gumbova koji postoje na aplikaciji. Sada ćemo pokazati kod koji se nalazi u pozadini aplikacije i gumbova koji nam omogućuje cijelu funkcionalnost same aplikacije.

```
Public Class Form1

    Private Access As New dbcontrol


    Private Function NotEmpty(text As String) As Boolean

        Return Not String.IsNullOrEmpty(text)

    End Function


    Private Sub Form1_Shown(sender As Object, e As EventArgs) Handles Me.Shown


        ' RUN QUERY '
```

```
    Access.ExecQuery("SELECT * FROM Učenik ORDER BY prezime ASC")
    If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

    ' FILL DATAGRID '
    dgvData.DataSource = Access.DBdataset

    ' FILL COMBOBOX '
    For Each R As DataRow In Access.DBdataset.Rows
        Users.Items.Add(R("ime"))
    Next

    ' DISPLAY FIRST NAME FOUND '
    If Access.RecordCount > 0 Then Users.SelectedIndex = 0
End Sub

Private Sub SearchMember(Prezime As String)
    ' ADD PARAMETERS & RUN OUR QUERY '
    Access.AddParam("@user", "%" & Prezime & "%")
    Access.ExecQuery("SELECT prezime, ime, OIB , kontakt  " &
                     "FROM Učenik " &
                     "WHERE prezime LIKE @user")

    ' REPORT AND ABORT ON ERRORS '
    If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

    ' FILL COMBOBOX '
    dgvData.DataSource = Access.DBdataset
End Sub

Private Sub cmdFind_Click(sender As Object, e As EventArgs) Handles cmdFind.Click
```

```
        SearchMember(Find.Text)
    End Sub

    Private Sub Izborni_predmet(Predmet As String)
        ' ADD PARAMETERS & RUN OUR QUERY '

        Access.ExecQuery("SELECT [Izborni predmet učenika].OIB, [Izborni predmet
učenika].Predmet, [Izborni predmet učenika].Ocjena, Predmet.[Šifra razreda]" &
        "From Predmet INNER Join [Izborni predmet učenika] On Predmet.[Šifra predmeta]
= [Izborni predmet učenika].[Predmet] Where (((Predmet.[Šifra razreda])='2'))")



        ' REPORT AND ABORT ON ERRORS '
        If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

        ' FILL COMBOBOX '
        dgvData.DataSource = Access.DBdataset
    End Sub

    Private Sub Dripu_Click(sender As Object, e As EventArgs) Handles Dripu.Click
        Izborni_predmet(Dripu.Text)
    End Sub

    Private Sub Predavaci_predmeti2(Predavač As String)
        ' ADD PARAMETERS & RUN OUR QUERY '

        Access.ExecQuery("SELECT Predavač.[Šifra predavača], Predavač.Prezime,
Predavač.Ime, Predmet.[Naziv predmeta], Predmet.[Šifra razreda]" &
        "From Predavač INNER Join Predmet On Predavač.[Šifra predavača] = Predmet.[Šifra
predavača] Where (((Predmet.[Šifra razreda])='2'))")
```

```
        ' REPORT AND ABORT ON ERRORS '
        If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

        ' FILL COMBOBOX '
        dgvData.DataSource = Access.DBdataset
    End Sub

    Private Sub Drpp_Click(sender As Object, e As EventArgs) Handles drpp.Click
        Predavaci_predmeti2(drpp.Text)
    End Sub
    Private Sub Zavrsne_ocjene2(Ocjena As String)
        ' ADD PARAMETERS & RUN OUR QUERY '

        Access.ExecQuery("SELECT Učenik.OIB, Učenik.Prezime, Učenik.Ime, [Završne
ocjene].Ocjena, Predmet.[Šifra razreda], Predmet.[Naziv predmeta]" &
        "From Učenik INNER Join (Predmet INNER Join [Završne ocjene] On Predmet.[Šifra
predmeta] = [Završne ocjene].Predmet) ON Učenik.OIB = [Završne ocjene].OIB Where
(((Predmet.[Šifra razreda])='2'))")

        ' REPORT AND ABORT ON ERRORS '
        If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

        ' FILL COMBOBOX '
```

```
        dgvData.DataSource = Access.DBdataset
    End Sub

    Private Sub drzo_Click(sender As Object, e As EventArgs) Handles drzo.Click
        Zavrsne_ocjene2(drzo.Text)
    End Sub

    Private Sub Zavrsne_ocjene1(Ocjena As String)
        ' ADD PARAMETERS & RUN OUR QUERY '

        Access.ExecQuery("SELECT Učenik.OIB, Učenik.Prezime, Učenik.Ime, [Završne
ocjene].Ocjena, Predmet.[Šifra razreda], Predmet.[Naziv predmeta]" &
        "From Učenik INNER Join (Predmet INNER Join [Završne ocjene] On Predmet.[Šifra
predmeta] = [Završne ocjene].Predmet) ON Učenik.OIB = [Završne ocjene].OIB Where
(((Predmet.[Šifra razreda])='1'))")




        ' REPORT AND ABORT ON ERRORS '
        If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

        ' FILL COMBOBOX '
        dgvData.DataSource = Access.DBdataset
    End Sub

    Private Sub przo_Click(sender As Object, e As EventArgs) Handles przo.Click
        Zavrsne_ocjene1(przo.Text)
    End Sub
    Private Sub Predavaci_predmeti1(Predmet As String)
```

```
        ' ADD PARAMETERS & RUN OUR QUERY '

        Access.ExecQuery("SELECT    Predavač.[Šifra    predavača],    Predavač.Prezime,
Predavač.Ime, Predmet.[Naziv predmeta], Predmet.[Šifra razreda]" &
        "From Predavač INNER Join Predmet On Predavač.[Šifra predavača] = Predmet.[Šifra
predavača] Where (((Predmet.[Šifra razreda])='1'))")




        ' REPORT AND ABORT ON ERRORS '
        If NotEmpty(Access.Exception) Then MsgBox(Access.Exception) : Exit Sub

        ' FILL COMBOBOX '
        dgvData.DataSource = Access.DBdataset
    End Sub

    Private Sub prpp_Click(sender As Object, e As EventArgs) Handles prpp.Click
        Predavaci_predmeti1(prpp.Text)
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub
End Class
```

Ovdje je prikazan kod koji sam koristio pri izgradnji ove aplikacije. Ovaj kod opisuje cijelu formu, od izgleda do izvršenja funkcija i upita. Većina koda koji je ovdje prikazan je generirana od strane same aplikacije VB.NET i ugrađenog jezika. Pa tako kada dvostruko kliknemo na samu formu ili na gumb programski jezik nam sam generira kod koji se nalazi iza toga. Unutar tog koda pišu se funkcije kako bi vaša aplikacija mogla izvršavati neku radnju, pa

se tako unutar glavne clase i forme „Form1“ rade novi blok u koji pišemo odvojeni skup koda (sub) u gdje sam upisivao upite i izvršavao te napravljene upite kako bismo rezultate prikazivali na ekranu prilikom pritiska gumba. Također postoji i još jedna skripta koja je napravljena unutar VB.NET programa i ona nam pomađe kako bismo se lakše spajali sa bazom i koristili njene funkcije, a ona izgleda ovako.

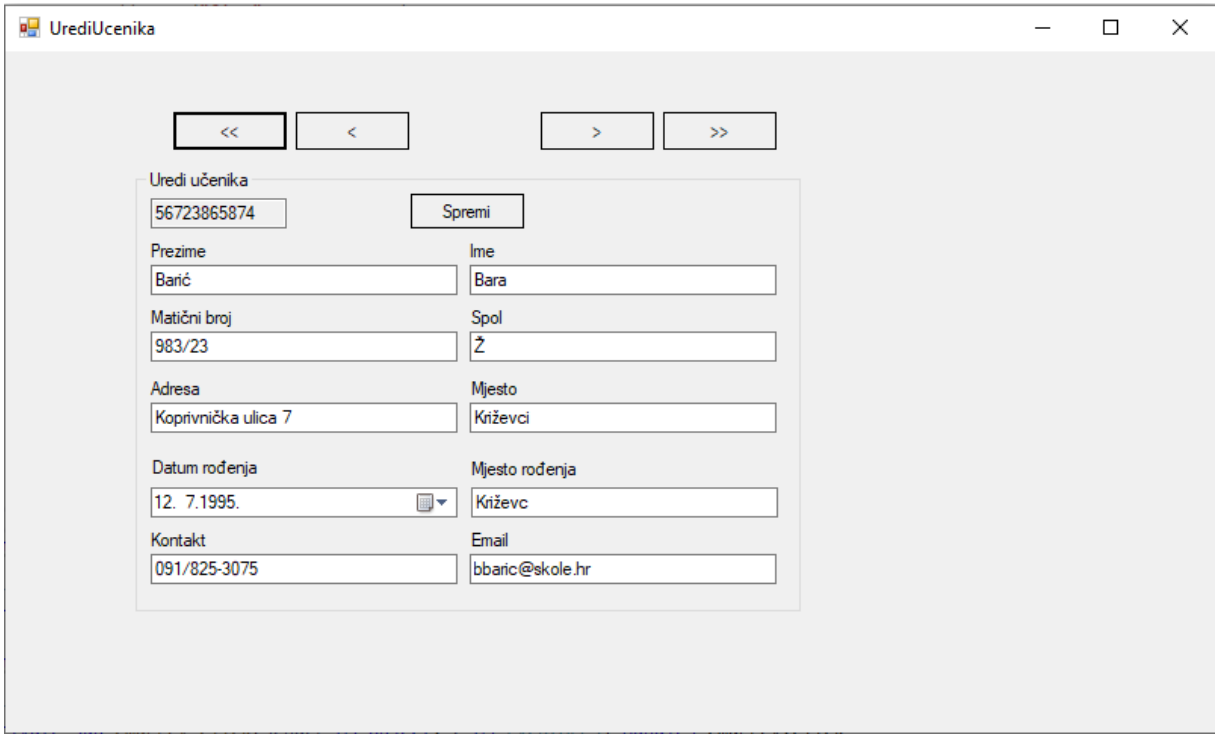
```
Imports System.Data.OleDb

Public Class dbcontrol
    ' CREATE YOUR DATABASE CONNECTION '
    Private DBcon As New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;" &
                                         "Data Source=Projekt.accdb;")

    ' PREPAE DB COMAND '
    Private DBcmd As OleDbCommand

    ' DB Data '

    Public DBdata As OleDbDataAdapter
    Public DBdataset As DataTable

    ' QUERY PARAMETERS '

    Public Params As New List(Of OleDbParameter)

    ' QUERY STATISTIC '

    Public RecordCount As Integer
    Public Exception As String
```

```
Public Sub ExecQuery(Query As String)
    ' RESET QUERY STATS '
    RecordCount = 0
    Exception = ""

    Try
        ' OPEN A CONNECTION '
        DBcon.Open()

        ' Create db command '
        DBcmd = New OleDbCommand(Query, DBcon)

        ' LOAD PARAMS INTO DB COMMAND '
        Params.ForEach(Sub(p) DBcmd.Parameters.Add(p))

        ' CLEAR PARAMS LIST '
        Params.Clear()

        ' EXECUTE DOMMAND & FILL DATATABLE '
        DBdataset = New DataTable
        DBdata = New OleDbDataAdapter(DBcmd)
        RecordCount = DBdata.Fill(DBdataset)

    Catch ex As Exception
        Exception = ex.Message

    End Try

    ' CLOSE YOUR CONNECTION '
    If DBcon.State = ConnectionState.Open Then DBcon.Close()
```

```
End Sub


' INCLUDE QUERY & COMMAND PARAMETERS '
Public Sub AddParam(Name As String, Value As Object)
    Dim NewParam As New OleDbParameter(Name, Value)
    Params.Add(NewParam)
End Sub


   End Class
```

Unutar ove skripte postavio sam parametre koje sam koristio u daljnjem radu sa aplikacijom i koja je uvelike olakšala moj rad. Napravljene su varijable u koje sam spremio podatke za konekciju na bazu te koje su nam potrebne da prikazujemo uopće podatke na formi (DataTable, OleDbDataAdapter, DBCon.open()...). Također na početku ove skripte naveli smo i izvor podataka, tj. našu bazu podataka.

Najglavniji dio ovog koda koji se nalazi u pozadini forme su upiti i kako njima baratati se nalazi u pod-formama i to su upiti. Access upite izvršavamo pomoću naredbe „Access.ExeQuery“ te nakon nje u zagradi se postavljaju upiti koje želimo izvršiti. Također postoji još jedna naredba s kojom možemo pretraživati naše rezultate, a to je `Access.AddParam("@user", "%" & Prezime & "%")`. Prisjetimo se da smo unutar skripte pomoću koje smo olakšali pristup bazi napravili i novi Sub (odvojeni skup koda) gdje smo generirali funkciju AddParam koji ovdje koristimo kako bismo odredili parametar po kojem želimo pretraživati naše rezultate. Funkcionalost gumbova riješili smo na način da smo dvostrukim klikom na gumb na formi otvorili jedan novi blok koda u koji je zadužen za radnje s tim gumbom, a tamo sam upisao naziv funkcije koju želim izvršiti te samim time i poziv te funkcije, da sam ju pozvao preko naziva gumba, što možemo promijeniti u svojstvima gumba



Slika 24. Izgled forme odabirom "Uredi ucenika" u izborniku na glavnoj formi

```
Public Class UrediUcenika

    Private Access As New dbcontrol


    Private CurrentRecord As Integer = 0


    Private Function NoErrors(Optional report As Boolean = False) As Boolean
        If Not String.IsNullOrEmpty(Access.Exception) Then
            If report = True Then MsgBox(Access.Exception) ' REPORT ERRORS '
            Return False
        Else
            Return True
        End If
    End Function


    Private Sub GetUsers()
```

```
        ' query users to fill data table
        Access.ExecQuery("SELECT * FROM učenik ORDER BY Prezime")

        ' report and abort on errors
        If NoErrors(True) = False OrElse Access.RecordCount < 1 Then Exit Sub

        ' get first record
        GetRecord()
    End Sub

    Private Sub GetRecord()
        ' fail if no records found or position is out of range
        If Access.DBdataset.Rows.Count < 1 OrElse CurrentRecord > Access.DBdataset.Rows.Count - 1 Then Exit Sub

        ' return first user found
        Dim r As DataRow = Access.DBdataset.Rows(CurrentRecord)

        ' populate fields
        txtOIB.Text = r("OIB").ToString
        txtPrezime.Text = r("Prezime").ToString
        txtIme.Text = r("Ime").ToString
        txtMb.Text = r("Matični broj").ToString
        txtSpol.Text = r("Spol").ToString
        txtAdresa.Text = r("Adresa").ToString
        txtMjesto.Text = r("Mjesto").ToString
        txtDr.Value = r("Datum rođenja")
        txtMr.Text = r("Mjesto rođenja").ToString
        txtKontakt.Text = r("Kontakt").ToString
        txtEmail.Text = r("E-mail").ToString
```

```
End Sub

Private Sub NextRecord(AddVal As Integer)
    CurrentRecord += AddVal
    If CurrentRecord > Access.DBdataset.Rows.Count - 1 Then CurrentRecord = 0
    If CurrentRecord < 0 Then CurrentRecord = Access.DBdataset.Rows.Count - 1

    ' update form
    GetRecord()
End Sub

Private Sub UpdateRecord()
    ' fail if no user is selected
    If String.IsNullOrEmpty(txtOIB.Text) Then Exit Sub

    'add parameters - order matters
    Access.AddParam("@prezime", txtPrezime.Text)
    Access.AddParam("@ime", txtIme.Text)
    Access.AddParam("@mb", txtMb.Text)
    Access.AddParam("@spol", txtSpol.Text)
    Access.AddParam("@adresa", txtAdresa.Text)
    Access.AddParam("@mjesto", txtMjesto.Text)
    Access.AddParam("@dr", txtDr.Value)
    Access.AddParam("@mr", txtMr.Text)
    Access.AddParam("@kontakt", txtKontakt.Text)
    Access.AddParam("@email", txtEmail.Text)
    Access.AddParam("@oib", txtOIB.Text)

    ' run command
    Access.ExecQuery("UPDATE Učenik " &
```

```
                        "SET    Prezime=@prezime,    Ime=@ime,    [Matični    broj]=@mb,
Spol=@spol, Adresa=@adresa, Mjesto=@mjesto, [Datum rođenja]=@dr, [Mjesto rođenja]=@mr,
Kontakt=@kontakt, [E-mail]=@email " &
                        "WHERE OIB=@oib")

        ' report and abort on errors
        If NoErrors(True) = False Then Exit Sub

        ' refresh the users datatableand fetch current record
        GetUsers()

    End Sub

    Private Sub UrediUcenika_Shown(sender As Object, e As EventArgs) Handles Me.Shown
        GetUsers()
    End Sub

    Private Sub cmdPrev_Click(sender As Object, e As EventArgs) Handles cmdPrev.Click
        NextRecord(-1)
    End Sub

    Private Sub cmdNext_Click(sender As Object, e As EventArgs) Handles cmdNext.Click
        NextRecord(1)
    End Sub

    Private Sub cmdFirst_Click(sender As Object, e As EventArgs) Handles cmdFirst.Click
        CurrentRecord = 0
        GetRecord()
    End Sub

    Private Sub cmdLast_Click(sender As Object, e As EventArgs) Handles cmdLast.Click
```

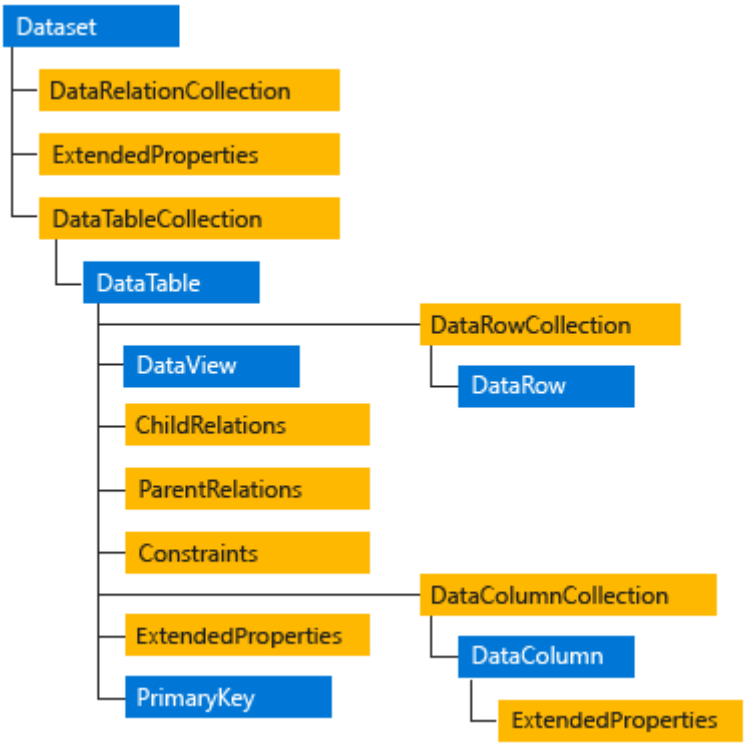
```
        CurrentRecord = Access.DBdataset.Rows.Count - 1
        GetRecord()
    End Sub

    Private Sub cmdSave_Click(sender As Object, e As EventArgs) Handles cmdSave.Click
        UpdateRecord()
    End Sub
End Class
```

Ovo je prikaz koda pomoću kojeg sam napravio funkcionalnost za uređivanje bilo kojeg učenika koji je upisan u sustav i bazu. Radio sam na isti način te se koristio nekim istim funkcijama kao i kod prošlih funkcionalnosti. Ova forma na početku selektira sve učenike koji se nalaze u bazi podataka te nam daje povratne informacije o njima. Svaki on njih je zapisan u polja koja smo napravili pomoću Toolboxa, tj. TextBoxova. U svaki textbox upisivali smo podatke koji se nalaze u našoj bazi podataka, a prije toga smo tim textboxovima dali nazive kako bismo se lakše snalazili u kodu i lakše im dodjeljivali vrijednosti. Također napravili smo i jednu novu varijablu koju smo nazvali „r“ pomoću koje sam dohvaćao svoje podatke, a onda dalje pomoću nje te podatke i ispisivao u textboxove. Svaku vrijednost koju sam dohvaćao sam pretvarao u string. Također izradio sam i gumbove pomoću kojih se možemo kretati kroz zapise, na način da se prvi zapis u tablici prikazuje odmah pri pokretanju, i to poredani po prezimenu, a svaki sljedeći zapis ili prethodni zapis se dohvaća pomoću gumbova. Napravio sam novi blok koda u kojem sam provjeravao na kojem se trenutno zapisu nalazimo, i ti zapisi su se provjeravali po primarnom ključu, što je u ovom slučaju „OIB“, te sam sa inkrementom prelazio na svaki sljedeći zapis. Također napravljeni su i gumbovi za odlazak na prvi, pa tako i na posljednji zapis u kojem go trenutku želimo. Nakon toga napravljen je blok koda u kojem ažuriram svoje podatke o učenicima. To je izvedeno tako što sam sa „Access.AddParam“ dodavao parametre svojim textboxovima te tim parametrima dalje dodavao vrijednost koja je upisana u textbox. Preko parametara napravio sam SQL upit pomoću kojeg sam ažurirao svoju tablicu. Bilo je vrlo bitno dobrim redoslijedom napisati parametre u upitu zbog toga što je VB.NET osjetljiv na redoslijed i moglo bi doći do nejasnih rezultata.

7. ADO.NET

Već prije smo spomenuli da je ADO.NET tehnologija koja je integrirana unutar .NET frameworka i omogućava nam dohvaćanje podataka i komunikaciju između različitih sustava. On nam omogućava dosljedan pristup izvorima podataka i podacima kao što je SQL Server i izovrima koji su izloženi preko OLE DB i ODBC. ADO.NET korisnik može koristiti i koristi za povezivanje na ta izvore podataka i dohvaćanje i manipuliranje podacima. On odvaja pristup podacima od manipuliranja podacima u dvije diskretne komponente koje mi možemo koristiti zajedno ili odvojeno. Također uključuje i .NET framework pružatelj podataka za spajanje na bazu, izvršavanje naredaba i dohvaćanje rezultata. Ovdje moramo spomenuti i DataSet objekt koji rezultate procesira direktno ili se može koristiti neovisno o .NET framework pružatelju podataka kako bi upravljao podatke lokalno u aplikaciji ili s XML izvora. Klase koje koristi ADO.NET mogu se pronaći u SystemData.dll datoteci i integrirane su sa datotekom System.Xml.dll. ADO.NET nam pruža najdirektinij pristup podacima unutar .NET frameworka.

Dvije glavne komponente ADO.NETa su .NET Framework Data Providers i DataSet. Prvi od njih su komponente koje su dizajnirane za manipulaciju podacima. Tako ovdje imamo Connection objekt koji nam pruža vezu sa izvorom podataka. Command objekt nam omogućava pristup naredbama nad bazom podataka s kojima možemo onda manipulirati podacima, prikazivati ih i izvršavati upite i slično. DataReader omogućuje nam brzi tok podataka s izvora podataka, a na kraju DataAdapter nam omogućuje most između DataSet objekta i izvora podataka. Kako bismo mogli napraviti promjene u originalnoj bazi podataka koju smo povezali sa aplikacijom potreban nam je DataAdapter koji korisnik Command objekt za izvršavanje SQL naredaba prema izvoru podataka za učitavanje DataSeta sa podacima i usklađivanje promjena koje su se dogodile nazad prema glavnom izvoru podataka.



Slika 25. ADO.NET arhitektura - DataSet

DataSet objekt unutar ADO.NET arhiktekture eksplicitno je dizajniran za pristup podacima neovisno kakav je izvor podataka. Kao rezultat toga on može biti korišten sa više različitih izvora podataka, XML podacima ili samo za loklanu uporabu u aplikaciji. On u sebi ima pohranjeno jednan ili više DataTable objekt koji se sastoji od redova i stupaca podataka te također primarnog ključa, vanjskog, ograničenja i informacija o vezama između podataka u DataTable objektima. [8.]

7.1. ADO.NET izvori podataka

Prvo ću pokazati primjer povezivanja na SQL klijenta. Ovaj kod kreira SQL komande koje selektiraju redove iz tablice Products, dodaje se parametar koji ograničava rezultat na redove koji imaju UnitPrice veću od određene upisane vrijednosti. SQL veza otvorena je unutar „using“ bloka naredbi što nam omogućava da se resursi zatvoreni i raspoloživi kad se završi blok naredbi.

```
using System;
using System.Data;
using System.Data.SqlClient;
```

```
class Program
{
    static void Main()
    {
        string connectionString =
            "Data Source=(local);Initial Catalog=Northwind;"
            + "Integrated Security=true";

        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from dbo.products "
                + "WHERE UnitPrice > @pricePoint "
                + "ORDER BY UnitPrice DESC;";

        // Specify the parameter value.
        int paramValue = 5;

        // Create and open the connection in a using block. This
        // ensures that all resources will be closed and disposed
        // when the code exits.
        using (SqlConnection connection =
            new SqlConnection(connectionString))
        {
            // Create the Command and Parameter objects.
            SqlCommand command = new SqlCommand(queryString, connection);
            command.Parameters.AddWithValue("@pricePoint", paramValue);
```

```
        // Open the connection in a try/catch block.
        // Create and execute the DataReader, writing the result
        // set to the console window.
        try
        {
            connection.Open();
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                Console.WriteLine("\t{0}\t{1}\t{2}",
                    reader[0], reader[1], reader[2]);
            }
            reader.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        Console.ReadLine();
    }
}
```

Sljedeći kod koji ćemo pokazati će biti prikaz kako se spaja OleDb baza podataka sa ADO.NET objektima. Primjer se ostvaruje na istim podacima i bazi kao i prošli primjer.

```
using System;
using System.Data;
using System.Data.OleDb;
```

```
class Program
{
    static void Main()
    {
        // The connection string assumes that the Access
        // Northwind.mdb is located in the c:\Data folder.
        string connectionString =
        "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
            + "c:\\Data\\Northwind.mdb;User Id=admin;Password=;";

        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from products "
                + "WHERE UnitPrice > ? "
                + "ORDER BY UnitPrice DESC;";

        // Specify the parameter value.
        int paramValue = 5;

        // Create and open the connection in a using block. This
        // ensures that all resources will be closed and disposed
        // when the code exits.
        using (OleDbConnection connection =
            new OleDbConnection(connectionString))
        {
            // Create the Command and Parameter objects.
```

```
        OleDbCommand command = new OleDbCommand(queryString, connection);
        command.Parameters.AddWithValue("@pricePoint", paramValue);

        // Open the connection in a try/catch block.
        // Create and execute the DataReader, writing the result
        // set to the console window.
        try
        {
            connection.Open();
            OleDbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                Console.WriteLine("\t{0}\t{1}\t{2}",
                    reader[0], reader[1], reader[2]);
            }
            reader.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        Console.ReadLine();
    }
}
```

Sljedeći kod koji ćemo pokazati će biti prikaz kako se spaja Odbc baza podataka sa ADO.NET objektima. Primjer se ostvaruje na istim podacima i bazi kao i prošli primjer.

```
using System;
```

```
using System.Data;
using System.Data.Odbc;

class Program
{
    static void Main()
    {
        // The connection string assumes that the Access
        // Northwind.mdb is located in the c:\Data folder.
        string connectionString =
            "Driver={Microsoft Access Driver (*.mdb)};"
            + "Dbq=c:\\Data\\Northwind.mdb;Uid=Admin;Pwd=;";

        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from products "
                + "WHERE UnitPrice > ? "
                + "ORDER BY UnitPrice DESC;";

        // Specify the parameter value.
        int paramValue = 5;

        // Create and open the connection in a using block. This
        // ensures that all resources will be closed and disposed
        // when the code exits.
        using (OdbcConnection connection =
            new OdbcConnection(connectionString))
```

```
        {
            // Create the Command and Parameter objects.
            OdbcCommand command = new OdbcCommand(queryString, connection);
            command.Parameters.AddWithValue("@pricePoint", paramValue);

            // Open the connection in a try/catch block.
            // Create and execute the DataReader, writing the result
            // set to the console window.
            try
            {
                connection.Open();
                OdbcDataReader reader = command.ExecuteReader();
                while (reader.Read())
                {
                    Console.WriteLine("\t{0}\t{1}\t{2}",
                        reader[0], reader[1], reader[2]);
                }
                reader.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            Console.ReadLine();
        }
    }
}
```

Sljedeći kod koji ćemo pokazati će biti prikaz kako se spaja Oracle baza podataka sa ADO.NET objektima. Ovaj kod prikazuje nam vezu na DEMO.CUSTOMER na Oracle serveru. Na samom početlu koda ovdje moramo dodati referencu System.Data.OracleClient.dll zbog komandi koje se koriste.

```
        using System;
using System.Data;
using System.Data.OracleClient;

class Program
{
   static void Main()
   {
      string connectionString =
         "Data Source=ThisOracleServer;Integrated Security=yes;";
      string queryString =
         "SELECT CUSTOMER_ID, NAME FROM DEMO.CUSTOMER";
      using (OracleConnection connection =
               new OracleConnection(connectionString))
      {
         OracleCommand command = connection.CreateCommand();
         command.CommandText = queryString;

         try
         {
            connection.Open();
```

```
            OracleDataReader reader = command.ExecuteReader();


            while (reader.Read())
            {
                Console.WriteLine("\t{0}\t{1}",
                    reader[0], reader[1]);
            }
            reader.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
  }
```

[8.]

8. Ažuriranje i brisanje podataka pomoću Visual Basic.NET

U ovom poglavlju govoriti ću o nekim metodama pomoći kojih možemo ažurirati podatke te brisati iste one koji se nalaze u našoj bazi podataka koju smo spojili sa našom aplikacijom izrađenom u VB.NETu. Takve metode uvelike nam pomažu pri radu s bazom zbog toga što ne moramo sve promjene raditi u samoj bazi, već to radimo pomoću sučelja u aplikaciji.

Tri su popularne metode koju su implementirane unutar programa VB.NET:

Uporabom TableAdapterDBDDirect metoda, kao što su TableAdapter.Update() i TableAdapter.Delete()

Uporabom TableAdapter.Update() metodom za ažuriranje i izvršavanje pripadajućih svojstava od TableAdaptera, kao što su UpdateCommand ili DeleteCommand, za spremanje promjena napravljenih za tablicu u DataSet tablici naše baze podataka

Uporabom „runtime“ objektne metode za kreiranje i izvršavanje Command metode ExecuteNonQuery() za ažuriranje i brisanje podataka prema bazi direktnim putem

Prve dvije metode koriste Visual Studio.NET gotove alate i čarobnjake pri izradi i konfiguraciji odgovarajućeg TableAdaptera, grade odgovarajuće upite pomoću Query Buildera i pozivaju te upite iz aplikacija napravljenih u VB.NETu. Razlika između prve dvije metode je ta što s pomoću prve metode možemo direktno ažurirati i brisati podatke naspram tablice u jednom koraku, a s drugom metodom potrebna su nam dva koraka kako bismo to učinili. U drugoj metode prvi korak je taj da se podaci ažuriraju i brišu samo na tablicama koje su odgovarajućeg DataSetu, a nakon toga se ažuriranje i brisanje odvija nad tablicama u bazi podataka pomoću TableAdapter.Update() metodom. [2.]

9. Zaključak

Kao zaključak ove teme sažeti ću cijeli rad o kojem sam dosad govorio. Visual Basic.NET pruža nam velike mogućnosti rada sa bazama podataka. Kao što smo mogli vidjeti možemo povezati mnogo različitih baza podataka sa ovim programskim jezikom. Metode povezivanja su razne. Visual Basic.NET kao programski jezik pruža nam posrednika između servera na kojem je naša baza podataka s kojom želimo uspostaviti konekciju i nas samih. U ovom radu koristio sam programski alat Visual Studio 2019 kako bih pokazao načine spajanja na baze podataka. Unutar samo programskog alata možemo raditi aplikacije koje se temelje na različitim programskim jezicima. Tako sam u svojim primjerima koristio i SQL i Visual Basic.NET kako bih mogao povezati podatke sa svojim primjerima. Vidjeli smo kako Visual Basic.NET i njegov rad s bazama podataka ima dosta složenu arhitekturu kako bi mogao pristupati podacima. Ima dva logička dijela a to su izvršni dio i dohvaćanje podataka. ADO.NET je glavni dio ovog programskog jezika zbog toga što nam on omogućuje manipulaciju podacima i bazom. On nam pruža konekciju te sve radnje koje izvršavamo. Također unutar ADO.NET arhitektura postoje i klase pomoću kojih lakše kreiramo i održavamo vezu sa serverom i bazom, a olakšavaju nam i sam rad aplikacije. Vidjeli smo isto tako i da se za različite baze podataka koriste različite instance konekcije te je ovaj programski jezik unutar tih instanci osjetljiv na velika i mala slova. Zaključno sa ovim radom smatram da je Visual Basic.NET jedan od naprednih programskih jezika koji uvelike olakšava rad sa svojim grafičkim sučeljem i pojednostavljenim načinima za spajanje preko svojih čarobnjaka, te nam još olakšava i manipulaciju podacima preko svojih alata.

Popis literature

1. Mastering Visual Basic .NET Database Programming, Evangelos P., Asli B., Sybex, 2002.
2. Practical database programming with Visual basic.NET, Ying B., Cambridge Univesrity Press, 2009.
3. https://www.homeandlearn.co.uk/NET/nets12p4.html (Ken Carney, Home and Learn)
4. https://www.tutorialspoint.com/vb.net/vb.net_database_access.htm (Tutorials Point India Limited)
5. https://www.youtube.com/watch?v=t68bCOPUzyM (iBasskung)
6. https://www.youtube.com/watch?v=cwDqjmSmtMQ (iBasskung)
7. https://www.youtube.com/watch?v=71ZwwzqzPeA (Programming for Everybody)
8. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/
9. https://docs.microsoft.com/en-us/dotnet/framework/get-started/
10. https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/linq/introduction-to-linq

Popis slika