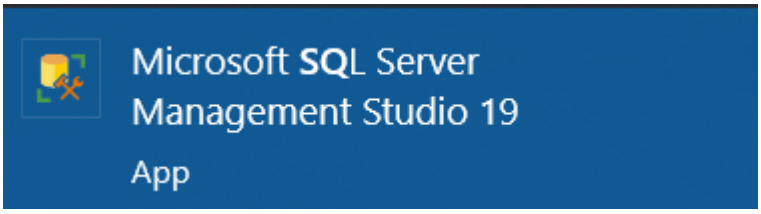
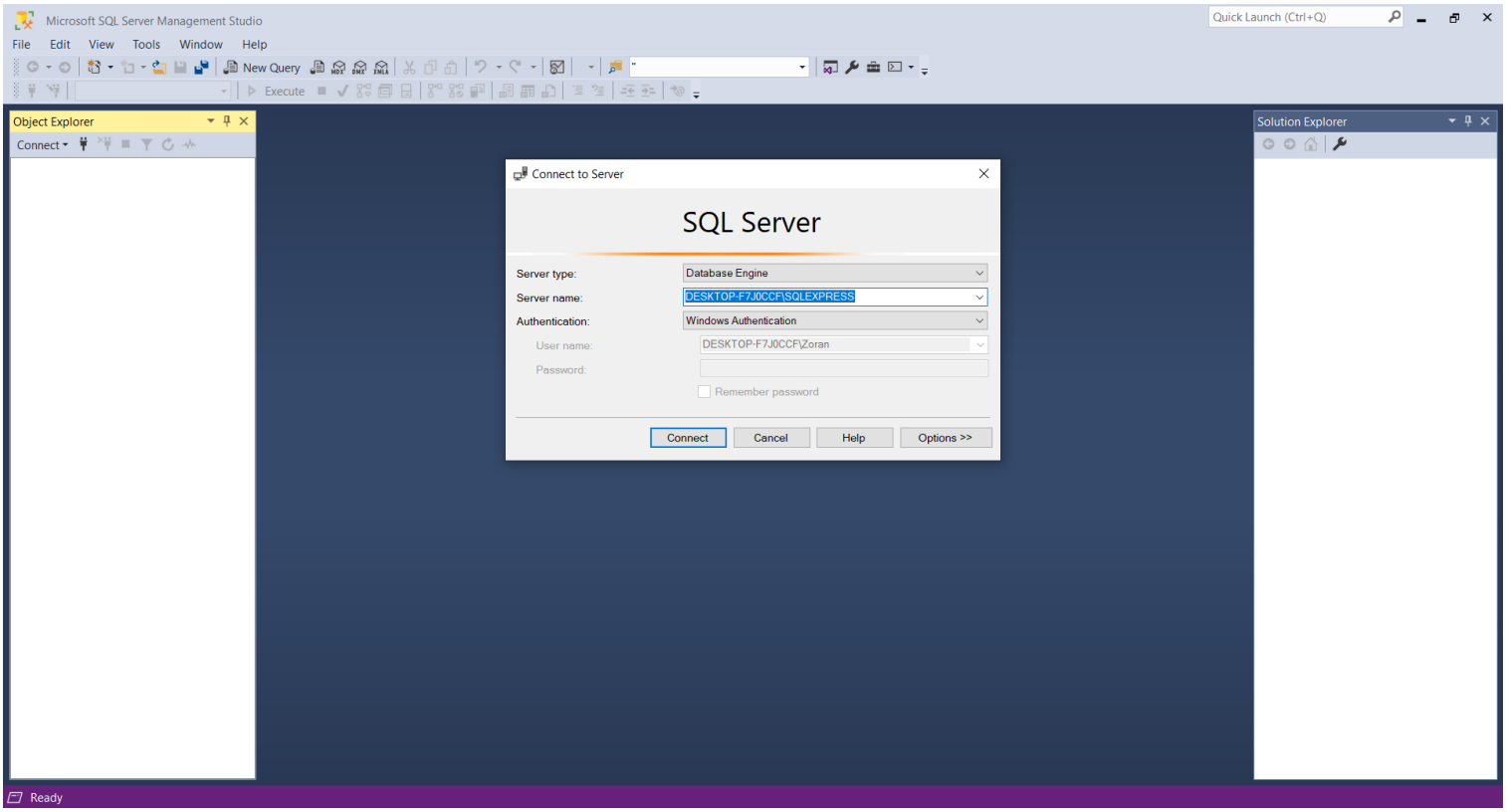
RAD SA BAZAMA PODATAKA

SQL Server Management Studio 19 – početni koraci

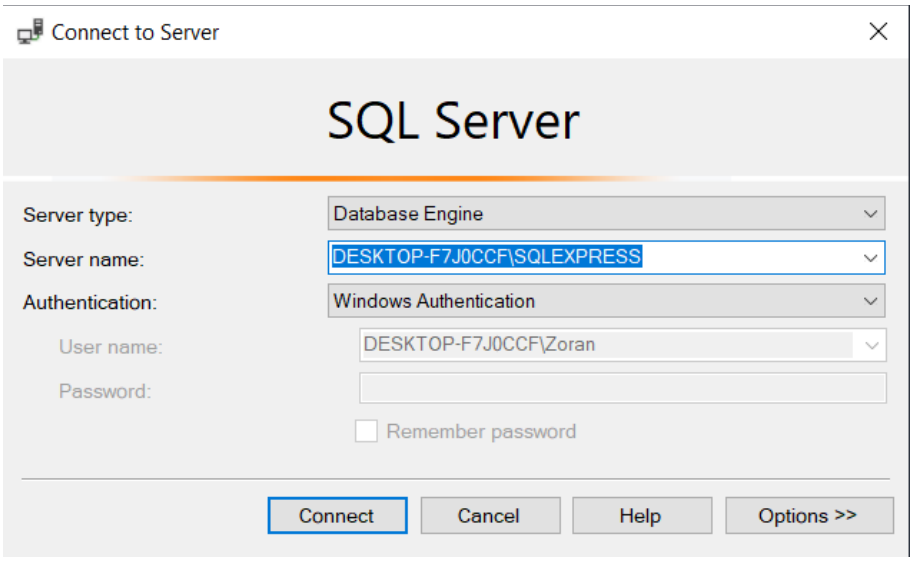
Pokretanje *Microsoft SQL Server Management Studio 19* okruženja :



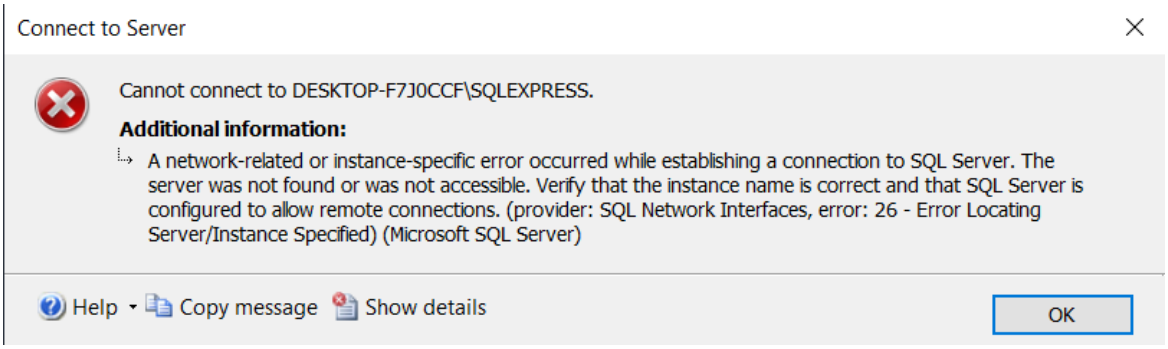
Nakon pokretanja programa dobija se radno okruženje i otvoren prozor *Connect to Server* :



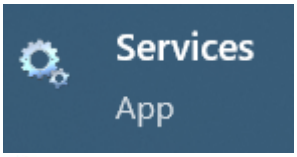
U prozoru *Connect to Server* potrebno je izabrati opciju *Connect*. Na mestu *Server name* će stojati ime_računara\SQLEXPRESS ako prilikom instalacije nije drugačije naznačeno.

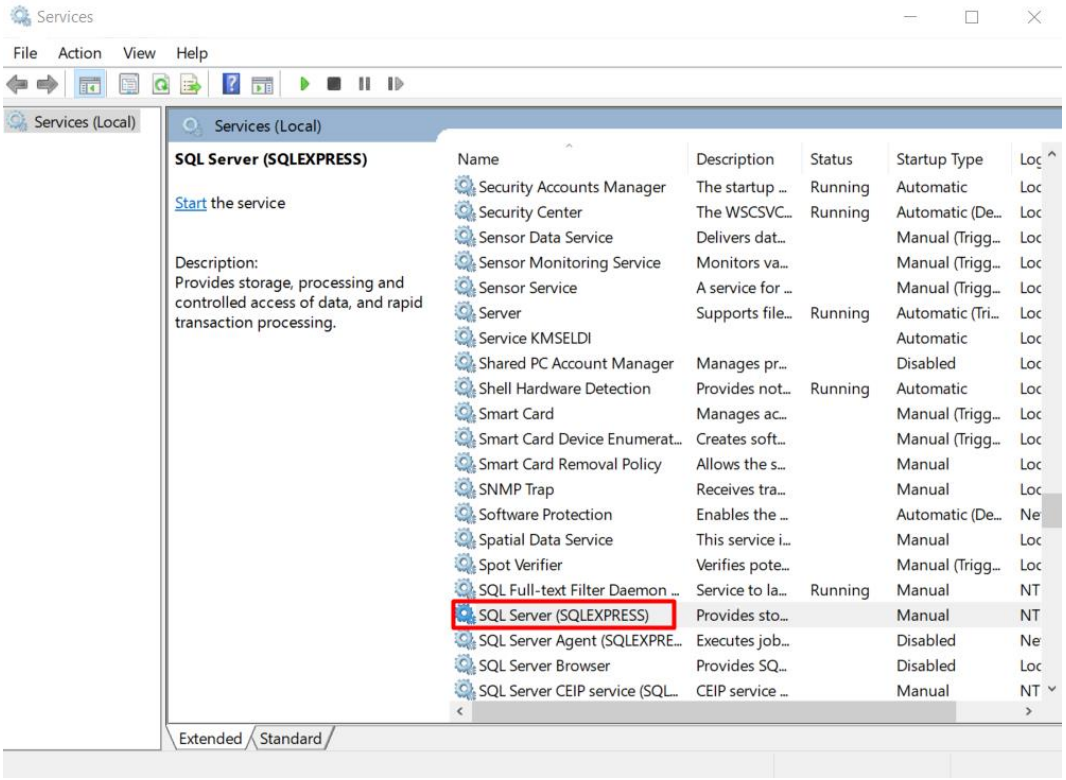


Ukoliko prilikom konektovanja na server dođe do greške prikazane na slici ispod, to je znak da SQL Server nije pokrenut na računaru.

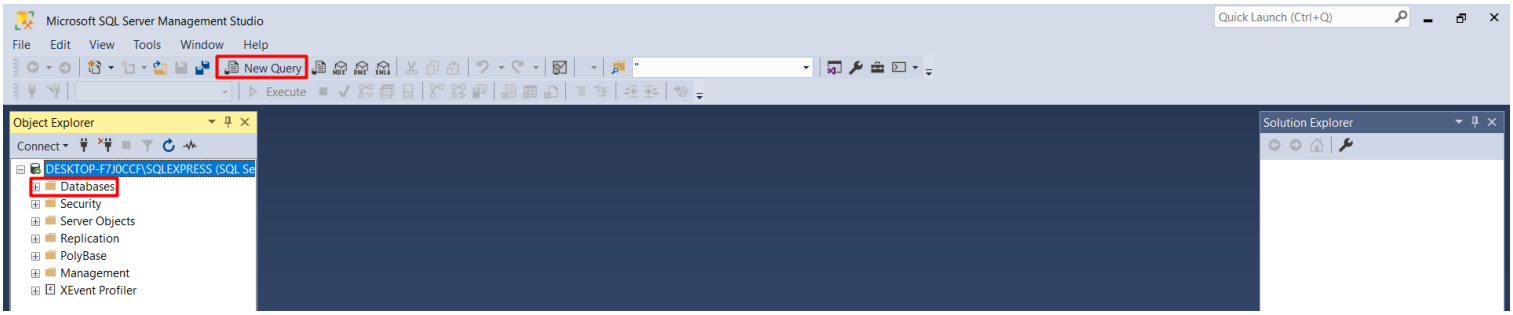


Da bi se SQL Server pokrenuo pristupa se servisima (*Services*) i pronalazi se servis *SQL Server (SQLEXPRESS)* i pokreće se desnim klikom na njega i izborom opcije *Start*.





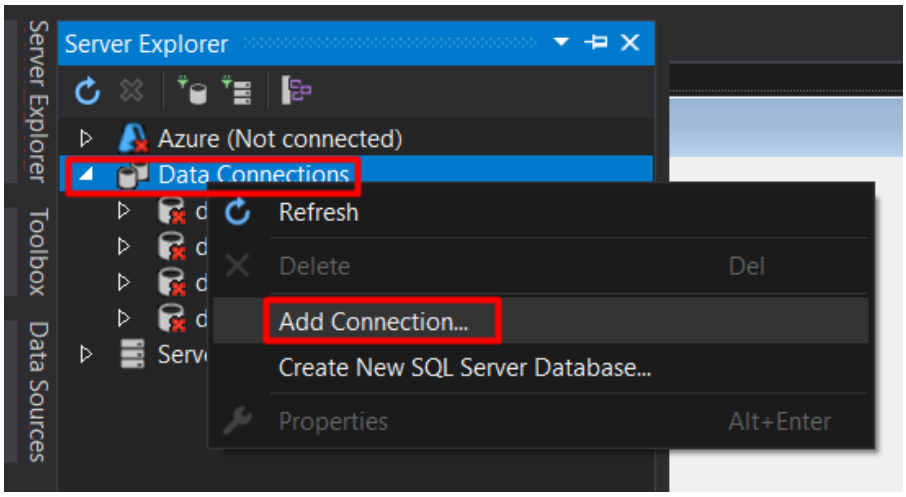
Nakon pokretanja potrebnog servisa ponovo biramo opciju *Connect* u prozoru *Connect to Server* i dobijamo radno okruženje sa *Object Explorer*-om na levoj, *Solution Explorer*-om na desnoj strani i praznim mestom u sredini. Upite kreiramo izborom opcije *New Query* iz gornje trake sa opcijama (ili kombinacijom tastera *CTRL* i *N*). Unutar *Object Explorer*-a će se nalaziti sve baze podataka koje kreirate, zajedno sa njihovim tabelama, pogledima itd.



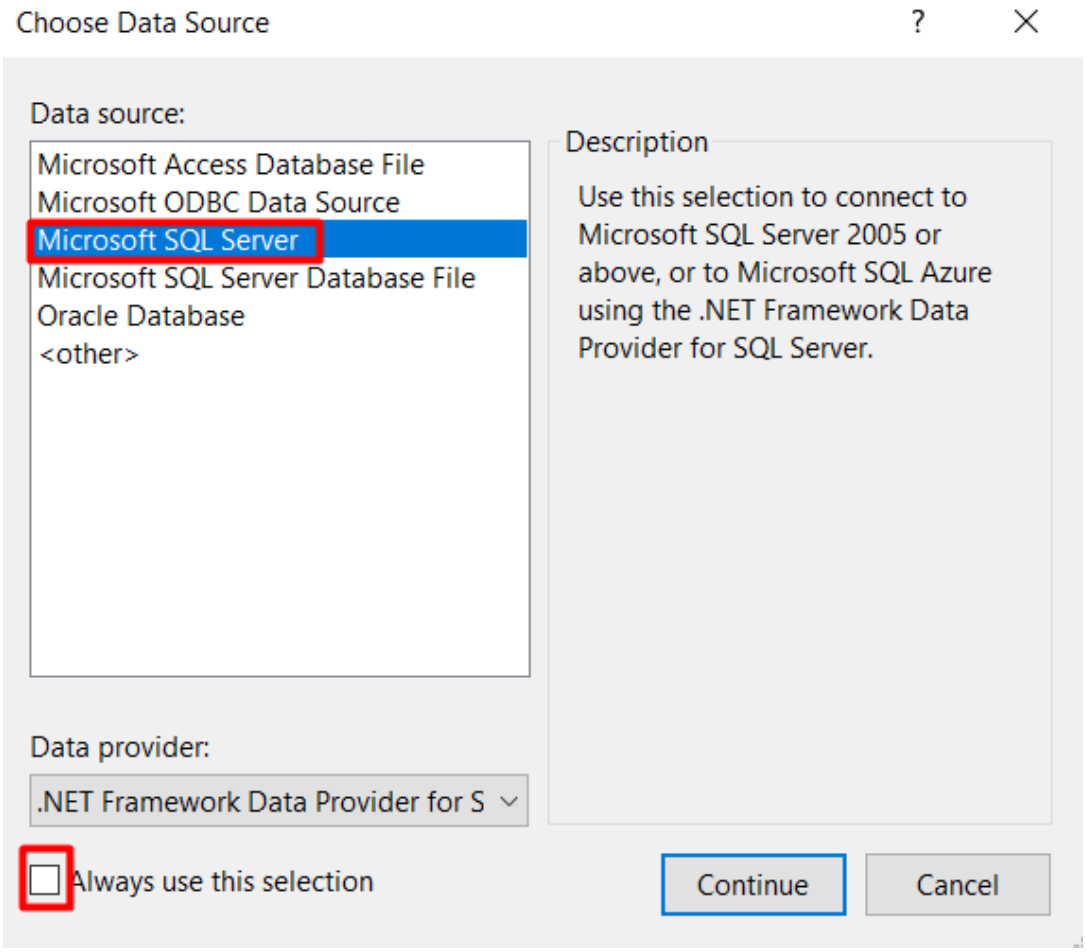
Nakon kreiranja novog upita možemo birati nad kojom bazom podataka želimo da ga izvršimo, izborom baze iz padajućeg menija u gornjoj traci sa alatima. Izborom opcije *Execute* izvršavamo kreirani upit.

Povezivanje baze podataka sa projektom – početni koraci

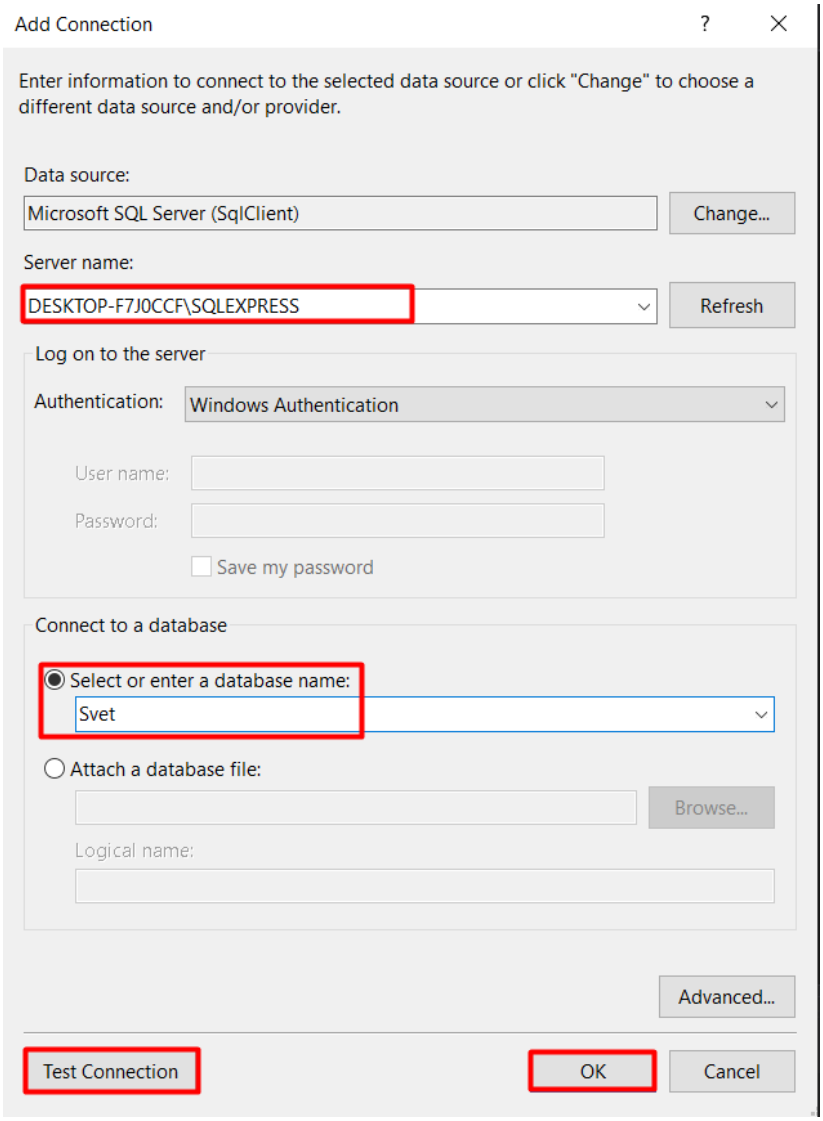
Unutar *Server Explorer*-a desnim klikom na *Data Connections* biramo opciju *Add Connection…* (ukoliko se *Server Explorer* ne nalazi na levoj strani radne površine otvaramo ga iz gornjeg menija biranjem opcija *View -> Server Explorer*.



U prozoru *Choose Data Source* biramo opciju *Microsoft SQL Server*, a polje *Always use this selection* ostavljamo nečekirano. Zatim biramo opciju *Continue*.

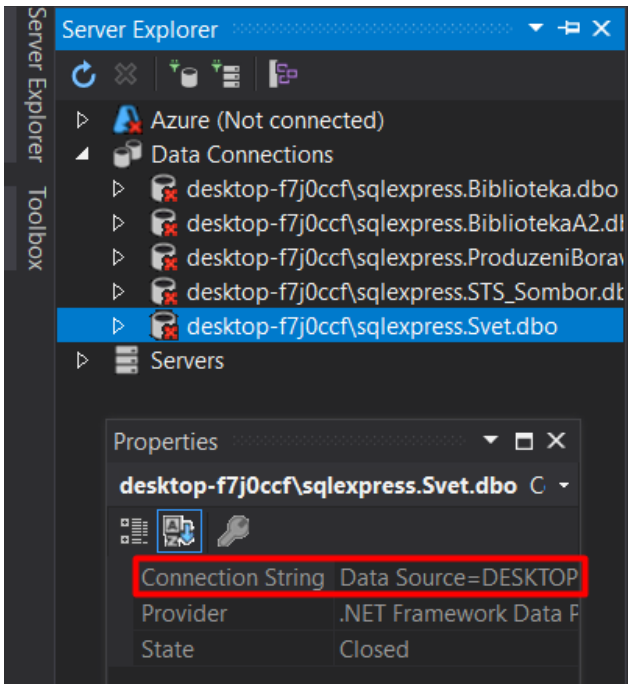


U polje *Server name:* upisujemo ime servera (isto koje je i u SQL Management Studio-u). Zatim u padajućem meniju biramo bazu podataka za koju želimo da dodamo konekciju. Nakon toga pritiskom na dugme *Test Connection* dobijamo povratnu informaciju da li je konekcija uspešna (*Test connection succeeded* ako jeste). Ukoliko je konekcija uspešna biramo dugme *OK*.



Konekcioni string

Konekcioni string koji nam je kasnije potreban za povezivanje sa bazom možemo naći u Properties-u od kreirane konekcije na bazu podataka.



Konekcioni string nam koristi da bi projekat znao na koji server i koju bazu podataka je potrebno da se poveže.

Kopirani tekst konekcionog stringa čuvamo unutar tekstualne promenljive proizvoljnog naziva (u primerima i priručniku promenljiva će biti nazivana *konekcioniString*).

```
string konekcioniString = @"Data Source=DESKTOP-F7J0CCF\SQLEXPRESS;Initial Catalog=Biblioteka;Integrated Security=True";
```

Klase za rad sa bazama podataka

Da bi koristili klase i metode za rad sa SQL Server bazom podataka na vrhu našeg projekta moramo dodati biblioteku *System.Data.SqlClient*.

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

SqlConnection

SqlConnection klasa u C# predstavlja vezu između aplikacije napisane u C# i baze podataka u kojoj se nalaze podaci koje aplikacija treba koristiti. Ova klasa omogućava otvaranje, zatvaranje i upravljanje vezom između aplikacije i baze podataka.

Da bi se koristila SqlConnection klasa, prvo se uključuje biblioteka za rad sa bazama podataka (*System.Data.SqlClient*). Zatim se kreira nova instanca SqlConnection klase i prosleđuje joj se string koji sadrži informacije o željenoj bazi podataka, kao što su naziv servera, naziv baze podataka, korisničko ime i lozinka (konekcioni string).

```
// Kreiranje instance klase SqlConnection
SqlConnection konekcija = new SqlConnection(konekcioniString);

// Otvaranje veze ka bazi podataka
konekcija.Open();

// Zatvaranje veze ka bazi podataka
konekcija.Close();
```

SqlCommand

SqlCommand klasa u C# je deo ADO.NET biblioteke i koristi se za izvršavanje SQL naredbi nad bazom podataka. SqlCommand klasa omogućuje da izvršimo različite vrste SQL naredbi, kao što su SELECT, INSERT, UPDATE, DELETE, itd.

```
/* Kreiranje instance klase SqlCommand
   (s tim da konekcija predstavlja instancu SqlConnection klase) */
SqlCommand komanda = new SqlCommand("SELECT * FROM Tabela", konekcija);

// SQL upit može da se čuva i unutar promenljive
string select = "SELECT * FROM Tabela";
SqlCommand komanda = new SqlCommand(select, konekcija);
```

Ukoliko se unutar upita komande nalaze i parametri njih dodajemo na sledeći način :

```
SqlCommand komanda = new SqlCommand(SQLupit, konekcija);

komanda.Parameters.AddWithValue("@ImeParametra", vrednost);
```

SqlDataReader

SqlDataReader klasa u C# omogućava programerima da pročitaju podatke iz SQL Server baze podataka. Ova klasa se koristi u kombinaciji sa SqlCommand klasom koja se koristi za izvršavanje SQL upita na bazi podataka. Kada se izvrši SQL upit korištenjem SqlCommand klase, SqlDataReader klasa se koristi za čitanje podataka iz rezultujućeg skupa podataka. SqlDataReader klasa čita podatke u jednom smeru, red po red, tako da omogućava brzo i efikasno čitanje velikih skupova podataka.

```
            string select = "SELECT * FROM Tabela";
            SqlCommand komanda = new SqlCommand(select, konekcija);

            // Izvršavanje čitanja podataka nad upitom iz komande
            SqlDataReader citac = komanda.ExecuteReader();

            while (citac.Read())
            {
                // Kod koji se izvršava dok se čitaju podaci
            }

            // Zatvaranje čitača - prekid čitanja podataka iz komande
            citac.Close();
```

SqlDataAdapter

SqlDataAdapter klasa u C# se koristi za preuzimanje podataka iz baze podataka i njihovo skladištenje u *DataSet* ili *DataTable* objekat. *DataSet* objekat može sadržati više *DataTable* objekata koji su usklađeni s tabelama u bazi podataka. Nakon što su podaci učitani u *DataSet* ili *DataTable* objekat, aplikacija ih može koristiti na razne načine.

```
            // Kreiranje instance klase SqlDataAdapter
            SqlDataAdapter adapter = new SqlDataAdapter(SQLupit, konekcija);

            // Kreiranje tabele - instance klase DataTable
            DataTable podaci = new DataTable();

            // Popunjavanje kreirane tabele podacima iz adapter-ovog upita
            adapter.Fill(podaci);
```

Rad sa ListView-om

ListView kontrola se na formu dodaje preuzimanjem iz *Toolbox*-a.

- ***Dodavanje kolona*** - Kolone se unutar ListView-a mogu dodavati ili preko svojstva **Columns** unutar *Properties*-a ili unutar koda forme (imeListView.Columns.Add("ImeKolone");).
- ***Prikaz kolona*** – Da bi videli naslove i podatke u kreiranim kolonama svojstvo **View** menjamo na vrednost **Details** (takođe ili preko *Properties*-a ili unutar koda (imeListView.View = View.Details;).
- ***Selektovanje celog reda*** – Ukoliko želimo da se klikom na red u ListView-u selektuju sve kolone tog reda menjamo svojstvo **FullRowSelect** u vrednost **True**.
- ***Selektovanje većeg broja redova*** – Ukoliko želimo da omogućimo/onemogućimo selektovanje većeg broja redova unutar ListView-a menjamo svojstvo **MultiSelect** u vrednost **True/False**.

Ukoliko želimo da određeni red bude selektovan :

```
imeListViewa.SelectedIndices.Add(indeks od željenog reda);
```

Ukoliko želimo da određeni red bude vidljiv (ukoliko je lista dugačka i potrebno je skrolovanje metoda *EnsureVisible()* će automatski dovesti do željenog reda) :

```
imeListViewa.Items[indeks od željenog reda].EnsureVisible();
```

Popunjavanje kolona podacima

Da bi popunili kolone podacima koristimo ranije pomenute klase za rad sa bazama podataka.

Primer :

```
// Dok čitač prolazi kroz podatke na osnovu prosleđenog upita kod unutar petlje će se izvršavati
            while (citac.Read())
            {
                /* Kreira se stavka ListView-a, a u konstruktor se postavlja prva željena vrednost u
tekstualnoj vrednosti */
                ListViewItem prikaz = new ListViewItem(citac[0].ToString());
```

```
                /* Na kreiranu stavku se dodaju SubItem-i, onoliko njih koliko podataka želimo da
izvučemo iz upita
                   Broj polja ne sme biti veći od broja polja koje vraća upit */
                prikaz.SubItems.Add(citac[1].ToString());
                prikaz.SubItems.Add(citac[2].ToString());
                prikaz.SubItems.Add(citac[3].ToString());
                prikaz.SubItems.Add(citac[4].ToString());
                // Kreirana stavka sa svojim SubItem-ima se dodaje ListView-u
                imeListViewa.Items.Add(prikaz);
            }
```

Rad sa ComboBox-om

Popunjavanje *ComboBox*-a podacima iz baze podataka se vrši pomoću korišćenja klasa *SqlDataAdapter* i *DataTable*.

```
            SqlDataAdapter adapter = new SqlDataAdapter(SQLupit, konekcija);

            DataTable podaci = new DataTable();
            adapter.Fill(podaci);

            // Popunjena tabela se dodaje kao DataSource
            imeComboBoxa.DataSource = podaci;

            // Za ValueMember se uzima unikatna vrednost iz upita
            imeComboBoxa.ValueMember = "UnikatnoPoljeIzTabele";

            // Za DisplayMember se uzima polje čija vrednost treba da se prikaže u ComboBox-u
            imeComboBoxa.DisplayMember = "PoljeIzTabele";
```

Rad sa DataGridView-om

DataGridView kontrola se na formu dodaje preuzimanjem iz *Toolbox*-a.

```
            SqlDataAdapter dataAdapter = new SqlDataAdapter(SQLupit);

            DataTable podaci = new DataTable();
            dataAdapter.Fill(podaci);

            // Podatke iz tabele dodeljujemo DataGridView-u
            imeDataGridViewa.DataSource = podaci;
```

Preko svojstva *ReadOnly* se omogućava/onemogućava da se vrši unos u polja *DataGridView*-a.

Rad sa Chart-om

Chart kontrola se na formu dodaje preuzimanjem iz *Toolbox*-a.

```
            // Dodeljujemo izvor podataka našem chart-u
            imeCharta.DataSource = dataAdapter;
            // Na x osu postavljamo kolonu po kojoj želimo da se mere podaci
            imeCharta.Series[0].XValueMember = "Kolona X ose";
            // Na y osu postavljamo kolonu koja po kojoj će se meriti podaci sa x ose
            imeCharta.Series[0].YValueMembers = "Kolona Y ose";
            // Unosimo podatke unutar chart-a
            imeCharta.DataBind();
```