УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

CLOUD COMPUTING U ELEKTROENERGETSKIM SISTEMIMA

Osnove *Microsoft WindowsAzure* servisa

-SKRIPTA-

Novi Sad, 2022

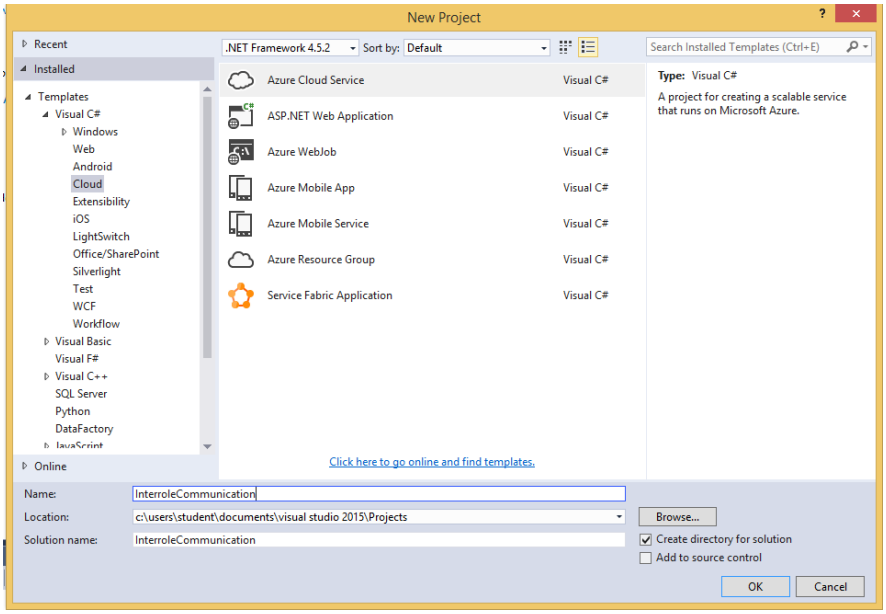
Vežba 1 – *Windows Communication Foundation (WCF)*

Zadatak je implementirati *Health Monitoring* system koji se sastoji iz servera i klijenta. Klijent predstavlja worker role tip procesa koji treba na svakih 5 sekundi da se javlja serveru čime potvrđuje da je živ. Preporuka je da se zadatak sastoji iz sledećih projekata:

- *WCFServer* (console application)
- *WCFHealthMonitoring* (Azure Cloud Service)
- *WCFContracts* (class library) – sadrži sve interfejse

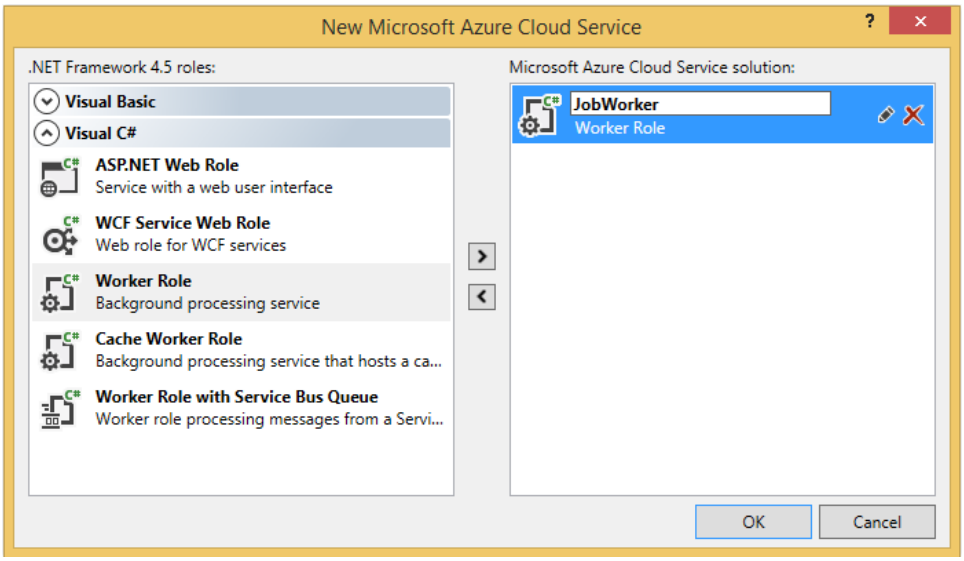
Prvo se treba kreirati C*loud* projekat *WCFHealthMonitoring* koristeći šablon koji dolazi uz instalirani SDK. Postupak za kreiranje *cloud* projekta:

- Klikom na *File* meni, odabrati *New/Project*
- U *New Project* dijalogu, proširiti Visual C# u listi *Installed Templates* i selektovati *Cloud*, kao što je prikazano na slici 2. Odabrati šablon *Azure Cloud Service*. Uneti ime projekta *WCFHealthMonitoring*. Kliknuti na OK.



Slika 1 Kreiranje cloud projekta

U *New Windows Azure Project* dijalogu, u okviru *Roles* panela, otvoriti tab Visual C#. Selektovati *Worker Role* i dati mu naziv *JobWorker* (Slika 3).



Slika 2 Dodavanje worker role

Dodati *Console Application* projekat po nazivu *WCFServer*. U okviru projekta kreirati *Server* kao posebnu klasu (*JobServer*) u okviru koje će biti metode za otvaranje i zatvaranje konekcije servisa (primer metoda dat u listingu 2). Koristiti *NetTcpBinding*. U implementaciji servisa (*HealthMonitoring* klasa koja implementira *IHealthMonitoring* interfejs) potrebno je ispisivati poruku "Worker role checked in.".

Kreirati *Class Library* projekat koji će sadržati interfejs za *WCF* servise. Nazvati projekat "*Contracts*". Dodati interfejs *IHealthMonitoring* u biblioteku. Definicija interfejsa je data u listingu 1.

Listing 1 – *IHealthMonitoring* interfejs.

```
[ServiceContract]
public interface IHealthMonitoring
{
	[OperationContract]
	void IAmAlive();
}
```

U implementaciji *OnStart()* metode *Worker role* pozvati otvaranje konekcije ka servisu (metoda *Connect()* prikazana u listingu 3). Zatim je potrebno u metodi *RunAsync* unutar *while* petlje pozivati metodu *IAmAlive* i ispisivati poruku "I am alive." u *Compute emulator*. Za ispisivanje poruka u *Compute emulator* koristiti: `Trace.WriteLine("tekst_poruke", "Information");`

Listing 2 – pokretanje i zatvaranje konekcije servisa

```
private ServiceHost serviceHost;

public JobServer()
{
	Start();
}

public void Start()
{
	serviceHost = new ServiceHost(typeof(HealthMonitoring));
	NetTcpBinding binding = new NetTcpBinding();
	serviceHost.AddServiceEndpoint(typeof(IHealthMonitoring), binding, new
		Uri("net.tcp://localhost:6000/HealthMonitoring"));
	serviceHost.Open();
	Console.WriteLine("Server ready and waiting for requests.");
}

public void Stop()
{
	serviceHost.Close();
	Console.WriteLine("Server stopped.");
}
```

Listing 3 – konektovanje klijenta na servis

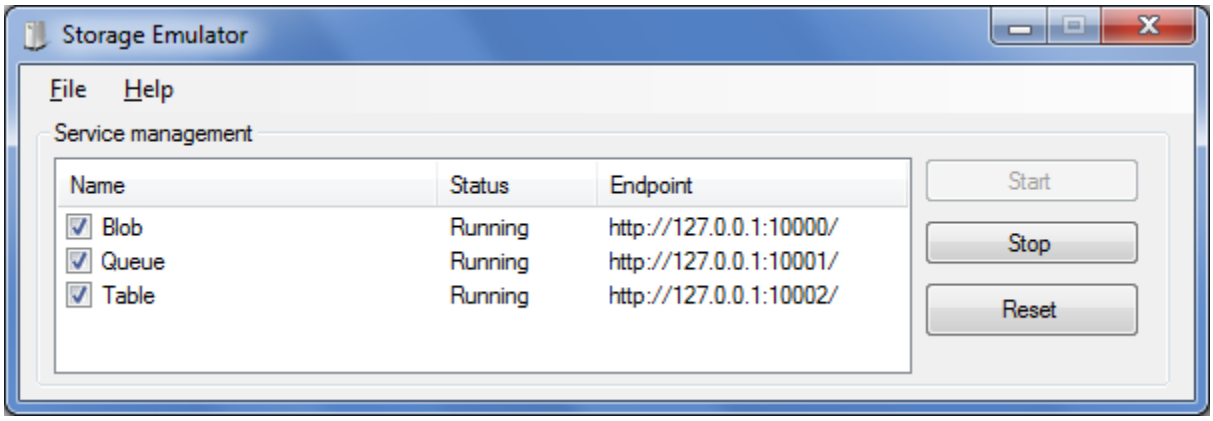
```
private IHealthMonitoring proxy;

public void Connect()
{
	var binding = new NetTcpBinding();
	ChannelFactory<IHealthMonitoring> factory = new
		ChannelFactory<IHealthMonitoring>(binding, new
		EndpointAddress("net.tcp://localhost:6000/HealthMonitoring"));
	proxy = factory.CreateChannel();

	//dalje se koristi proxy.IAmAlive() u RunAsync metodi

}
```

Rad sa Compute emulatorom i Storage emulatorom

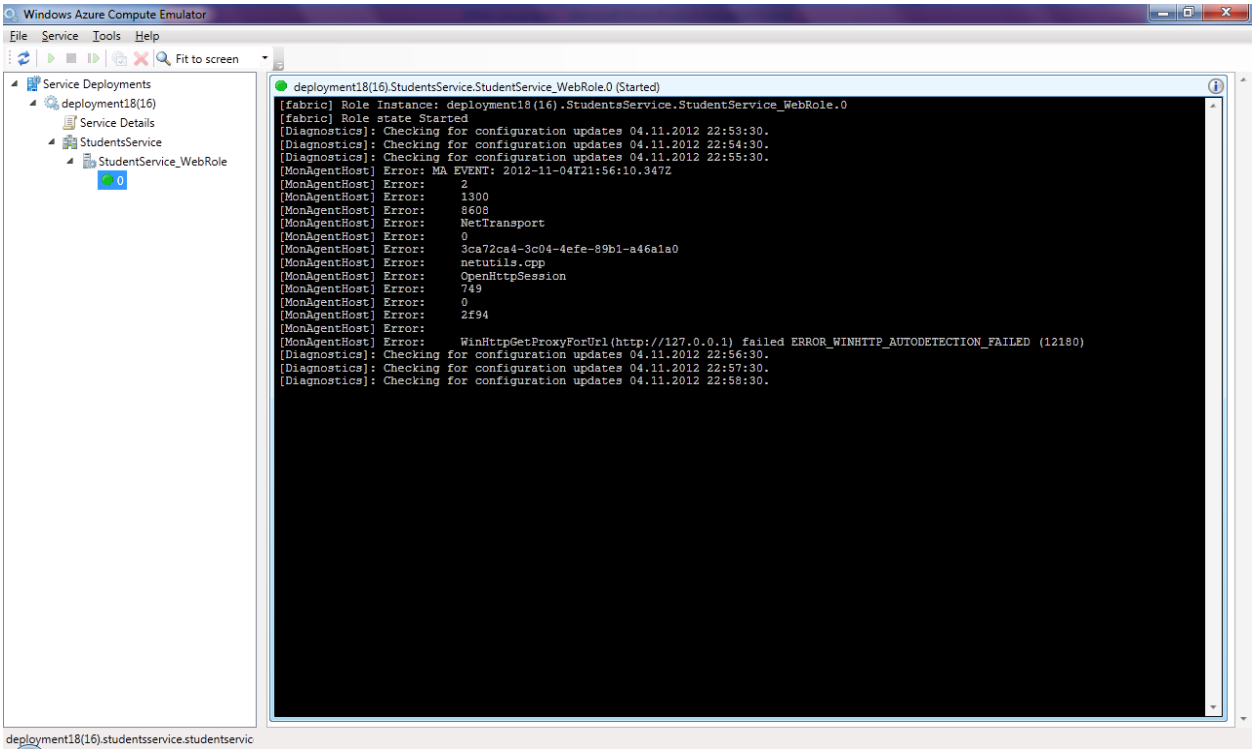
Nakon pokretanja *cloud* projekta, prvo se pokreću *Compute* emulator i *Storage* emulator. U okviru *system tray* (*notification area*) u *Windows OS* okruženju, koji se nalazi pored sata, pojavi se *Microsoft logo*, koji predstavlja *WindowsAzure* emulator. Desnim klikom na logo, dobija se pet opcija: *Shutdown Compute* emulator, *Shutdown Storage* emulator, *Show Compute UI*, *Show Storage UI* i *exit*.

Shutdown opcije će zaustaviti servise emulatora i aplikacije neće moći da se izvršavaju u emuliranom okruženju. Opcija *exit* će uraditi *shutdown* nad obe vrste emulator servisa. Opcije *Show Compute UI* i *Show Storage UI* prikazuju korisnički interfejs za ova dva servisa.

Storage UI je jednostavan interfejs koji je prikazan na slici 6. Interfejs prikazuje statuse servisa, nudi opcije pokretanja, restartovanja i zaustavljanja servisa. Takođe, pokazuje i *Endpoint* adresu na kojoj možemo pristupiti resursima. Na osnovu ovih adresa, moguće je pristupiti resursima i putem *browser*-a.



Slika 3 Storage emulator interfejs

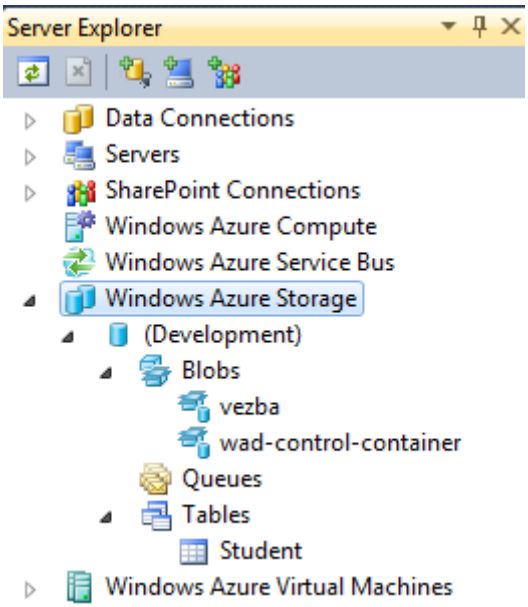


Slika 4 Compute emulator interfejs

Compute UI je češće korišćeni interfejs iz razloga što omogućava praćenje životnog ciklusa *WindowsAzure* servisa koji se izvršava u kontekstu emuliranog *compute* servisa. *Compute UI* je prikazan na slici 7.

Na osnovu stabla u levom delu interfejsa, moguće je izlistati sve zasebne kontekste izvršavanja, i okviru svakog postoje servisi i detalji servisa. U okviru servisa se uvek nalaze sve *web* i *worker* role. Proširenjem određene *role* možemo videti koliko je procesa pokrenuto. Na slici 17 se vidi da postoji samo jedan pokrenut *web role* proces. Klikom na određenu instancu, dobijaju se informacije o izvršavanju određene insance. Ovo je veoma pogodno u fazi rezvoja, jer je praktično omogućen konzolni prikaz za svaku instancu posebno. Uz pomoć *Trace* klase, moguće je ispisivati razne informacije na konzolu što olakšava proces razvijanja aplikacija.

WindowsAzure Storage servisu je moguće pristupiti direktno iz *Visual Studio* razvojnog okruženja. Iz menija *View*, kliknuti na *ServerExplorer* ukoliko već nije otvoren. Kada se otvori Server explorer, u njemu se nalazi *Windows Azure Storage* što je prikazano na slici 8.



Slika 5 Visual Studio Server Explorer

Iz *Visual Studio Server Explorer* komponente, može se vršiti navigacija po sve tri vrste servisa u okviru *Windows Azure Storage* servisa. Dvoklikom na tabele, moguće je izlistati sve entitete. Dvoklikom na *BLOB* kontejner, moguće je izlistati sve binarne objekte sadržane u okviru određenog kontejnera.