Dokumentacija za projektni zadatak

1. Listing glavnih delova čuvanja primarne instance:

HomeController metoda *Dodavanje*

```
[HttpPost]
 public ActionResult Dodavanje(String index, String ime, String prezime)
 {


     Random rand = new Random();


     List<RoleInstance> pomoc = new List<RoleInstance>();

     foreach (RoleInstance role in RoleEnvironment.Roles["WriterRole"].Instances)
     {
         pomoc.Add(role);
     }

     bool uslov = false;

     while (!uslov)
     {
         int instanca = rand.Next(3);
         ChannelFactory<IWriter> factory = new ChannelFactory<IWriter>(new
NetTcpBinding(), new EndpointAddress(String.Format("net.tcp://{0}/{1}",
pomoc[instanca].InstanceEndpoints["InternalRequest"].IPEndpoint, "InternalRequest")));
         IWriter proxy = factory.CreateChannel();

         uslov = proxy.DodajStudenta(index, ime, prezime);
     }




         return RedirectToAction("Index");
 }
```

Metoda *DodajStudenta* koja je implementirana u WorkerRole, ServerProvider

```
public bool DodajStudenta(string index, string ime, string prezime)
        {

            int pomInd = GetIndex(RoleEnvironment.CurrentRoleInstance.Id);
            if (pomInd != 0 && pomInd != 2)
                return false;


            ChannelFactory<IReader> factory = new ChannelFactory<IReader>(new
NetTcpBinding(), new EndpointAddress("net.tcp://localhost:10100/InputRequest"));
            IReader proxy = factory.CreateChannel();



            Student noviStudent = new Student(ime,prezime,index);

            Student student = proxy.PronadjiStudenta(noviStudent.Index);
            if (student == null)
            {
                writerRepo.DodajStudenta(noviStudent);
                //Trace.WriteLine($"DOODAT STUDENT:
{RoleEnvironment.CurrentRoleInstance.Id}");

                CloudQueue queue = QueueKreiranje.GetQueueReference("zadatakqueue");
                string poruka = $" [{DateTime.Now}] Dodavanje studenta: Index:
{noviStudent.Index} Ime: {noviStudent.Ime} Prezime: {noviStudent.Prezime}";
                CloudQueueMessage queue_poruka = new CloudQueueMessage(poruka);
                queue.AddMessage(queue_poruka, null, null);

            }
            else
            {
                CloudQueue queue = QueueKreiranje.GetQueueReference("zadatakqueue");
                string poruka = $" [{DateTime.Now}] Vec postoji student sa ovim indeksom:
{noviStudent.Index}";
                CloudQueueMessage queue_poruka = new CloudQueueMessage(poruka);
                queue.AddMessage(queue_poruka, null, null);

                Trace.WriteLine($"Vec postoji student sa ovim indeksom:
{noviStudent.Index}");
            }

            return true;
        }
```

`writerRepo.DodajStudenta(noviStudent)` - metoda koja se poziva u okviru Metode *DodajStudenta*

```
public void DodajStudenta(Student noviStudent)
{

     TableOperation insertOperation = TableOperation.Insert(noviStudent);
     _table.Execute(insertOperation);
 }
```

Sledi pojašnjenje pojedinih delova: Kada klijent putem korisničkog interfejsa pomoću odgovarajućeg View – a pogodi akciju kontrolera (u ovom slučaju Dodavanje u okviru HomeController) tada se započne proces dodavanja studenta.WebRole i WriterRole komuniciraju preko internal endpoint - a pa se iz tog razloga vrši otvaranje konekcije ka jednoj od instanci WriterRole – a.Prilikom ovog poziva `proxy.DodajStudenta(index, ime, prezime);` vrši se provera da li je pogođena odgovarajuća instanca WriterRole (jedna od onih koje su zadužene za manipulaciju primarnim entitetima).Ukoliko nije ostvarena konekcija sa tom instancom vracaća se false i vrši se ponovno rekreiranje ChannelFactory – a dok se uslov ne zadovolji.

```
bool uslov = false;

while (!uslov)
{
    int instanca = rand.Next(3);
    ChannelFactory<IWriter> factory = new ChannelFactory<IWriter>(new
    IWriter proxy = factory.CreateChannel();

    uslov = proxy.DodajStudenta(index, ime, prezime);
}
```

uslov je zadovoljen tek kada se pogodi odgovarajuća instanca

```
int pomInd = GetIndex(RoleEnvironment.CurrentRoleInstance.Id);
if (pomInd != 0 && pomInd != 2)
    return false;
```

Provera da li je pogođena odgovarajuća instanca WriterRole - e

Metoda *DodajStudenta* koja je implementirana u WorkerRole, ServerProvider sadrži interrole komunikaciju sa ReaderRole (ona je zadužena za čitanje, ne sme WriterRole da čita) kako bi se utvrdilo da ne postoji student sa zadatim indeksom.Ako ne postoji entitet se dodaje u tabelu, u suprotnom se dobija informacija o grešci.Takođe nakon operacije dodavanja u queue se dodaje poruka koja sadrži informaciju o entitetu koji je dodat.Sličan je princip i prilikom ažuriranja.

2 Listing implementacije komunikacije Writer instanci i Logger-a

Primeri u kojima se dodaje poruka u queue u okviru WriterRole – e

Primer1

```
writerRepo.DodajPredmet(noviPredmet);

                CloudQueue queue = QueueKreiranje.GetQueueReference("zadatakqueue");
                string poruka = $" [{DateTime.Now}] Dodavanje predmeta: Oznaka predmeta:
{noviPredmet.OznakaPredmeta} Naziv Predmeta: {noviPredmet.NazivPredmeta}";
                CloudQueueMessage queue_poruka = new CloudQueueMessage(poruka);

                queue.AddMessage(queue_poruka, null, null);
```

Primer2

```
writerRepo.ObrisiIspit(ispit);

            CloudQueue queue = QueueKreiranje.GetQueueReference("zadatakqueue");
            string poruka = $" [{DateTime.Now}] Brisanje ispita: IdIspita:
{ispit.IdIspita} Datun: {ispit.Datum} Polozen: {ispit.Polozen}";
            CloudQueueMessage queue_poruka = new CloudQueueMessage(poruka);
            queue.AddMessage(queue_poruka, null, null);
```

Prilikom svake operacije koja se izršava u oviru WorkerRole dodaje se poruka u queue.Iz tog razloga kreirana je klasa QueueKreiranje (nalazi se u okviru ClassLibrary projekta ManipulacijaPodacima) koja nam olakšava rad sa queue.

```
namespace ManipulacijaPodacima
{
    public class QueueKreiranje
    {

        public static CloudQueue GetQueueReference(String queue_naziv)
        {

            CloudStorageAccount storageAccount = CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

            CloudQueue queue = queueClient.GetQueueReference(queue_naziv);
            queue.CreateIfNotExists();

            return queue;

        }
    }
}
```

U okviru LoggerRole se vrši kontinualna provera saržaja queue i u zavisnosti od toga da li ima neka poruka preduzima se odgovarajuća akcija (upis u blob).Nakon preuzimanja poruke iz queue poruka se briše tj. obrađena je.

```
CloudQueueMessage poruka = queue.GetMessage();

if (poruka == null)
{
    Trace.WriteLine("Ne postoji ni jedna log poruka.");
}
else
{
    string preuzetaPoruka = poruka.AsString;
    Trace.WriteLine($"Preuzet log: {preuzetaPoruka}");



    string name_in_blob = "poruke";
    var storageAccount = CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
    CloudBlobClient blobStorage = storageAccount.CreateCloudBlobClient();
    CloudBlobContainer container = blobStorage.GetContainerReference("zadatakblob");

    CloudBlockBlob blob = container.GetBlockBlobReference(name_in_blob);

    string red = "";
    if (blob.Exists())
    {
        red = blob.DownloadText();
        red = red + "|";

    }

    red = red + preuzetaPoruka;
    blob.UploadText(red);
    queue.DeleteMessage(poruka);
```

3 Nazivi resursa

- Naziv tabele: ZadatakTabela (u okviru nje sam smeštao sve potrebne podatke tako što sam im stavljao različit PartitionKey i RowKey)
- Naziv queue: zadatakqueue
- Naziv bloba: zadatakblob

4 Test slučajevi

Koraci za testiranje dodavanja primarne instance:

1. Navigacijom u okviru web stranice pronaći view za dodavanje studenta
2. Popuniti polja i kliknuti na submit dugme (Dodaj studenta)
3. Logika za odredjivanje instance koja prima zahtev odrađena je pomoću Random – a I ne može se direktno uticati koja će se instance pogoditi
4. U pozadini se vrši sve ono što je već objašnjeno i prikazano u ranijim stvakama dokumentacije

5. Logger se salje odgovaraju informaciju
6. Logger upisuje tu informaciju u blob

Dodavanje sekundarnog entiteta funkcionise na intentičan način samo što se dodavanje vrši u okviru odgvorajuće instance (instance sa id 1)

Ažuriranje i brisanje mogu se izvršiti tako što se u okviru View a koji lista sve objekte klikne na ažuriraj ili obriši.

LoggerClientConsole ima mogućnost prikaza svih logova koji se nalaze u okviru bloba.Prikazan je meni koji jasno govori koje mogućnosti su u opticaju.Takođe LoggerClientConsole prima informacije o novonstalim logovima tako što putem input endpointa komunicira sa LoggerRole.

Napomen: Prilikom pokretanja programa neophodno je pokrenuti LoggerClientConsole posebno: desni klik na taj projekat -> debug -> start new instance