Listing glavnih delova implementacije čuvanja primarne instance

Listing 1 – Phone class

```
using Microsoft.WindowsAzure.Storage.Table;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CloudService_Data
{
    public class Phone : TableEntity
    {
        public string Brand { get; set; }
        public string Model { get; set; }
        public double Price { get; set; }

        public Phone(string phoneNo)
        {
            PartitionKey = "Phone";
            RowKey = phoneNo;
        }

        public Phone() { }
    }
}
```

Listing 2 – PhoneStoreDataRepository class

```
using Microsoft.Azure;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Table;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CloudService_Data
{
    public class PhoneStoreDataRepository
    {
        private CloudStorageAccount _storageAccount;
        private CloudTable _table;

        public PhoneStoreDataRepository()
        {
            _storageAccount =

CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
            CloudTableClient tableClient = new CloudTableClient(new
            Uri(_storageAccount.TableEndpoint.AbsoluteUri), _storageAccount.Credentials);
            _table = tableClient.GetTableReference("PhoneStoreTable");
```

```
            _table.CreateIfNotExists();
        }

        #region Phone Operations

        public void AddPhone(Phone newPhone)
        {
            TableOperation insertOperation = TableOperation.InsertOrReplace(newPhone);
            _table.Execute(insertOperation);
        }

        public void AddOrReplacePhone(Phone newPhone) //This will be the edit CRUD
operation, later..
        {
            TableOperation insertOperation = TableOperation.InsertOrReplace(newPhone);
            _table.Execute(insertOperation);
        }
```

Listing 3 – Create and Edit Actions in Web Role

```
public ActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Create(Phone phone)
        {
            List<EndpointAddress> internalEndpoints = RetreiveAllInstances();
            IEntityHandler proxy = new ChannelFactory<IEntityHandler>(new
NetTcpBinding(), internalEndpoints[0]).CreateChannel();

            proxy.AddPhone(phone.RowKey, phone.Brand, phone.Model, phone.Price);

            return RedirectToAction("Index");
        }

        // Get Edit
        public ActionResult Edit(string id)
        {
            List<EndpointAddress> internalEndpoints = RetreiveAllInstances();
            IEntityHandler proxy = new ChannelFactory<IEntityHandler>(new
NetTcpBinding(), internalEndpoints[0]).CreateChannel();

            Phone phone = proxy.GetPhone(id);

            return View(phone);
        }

        [HttpPost]
        public ActionResult Edit(Phone phone)
        {
            List<EndpointAddress> internalEndpoints = RetreiveAllInstances();
            IEntityHandler proxy = new ChannelFactory<IEntityHandler>(new
NetTcpBinding(), internalEndpoints[0]).CreateChannel();

            proxy.AddOrReplacePhone(phone.RowKey, phone.Brand, phone.Model, phone.Price);
```

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

Listing 4 – IentityHandler Interface – Database writing contracts

```
using CloudService_Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;
using System.Text;
using System.Threading.Tasks;

namespace Common
{
    [ServiceContract]
    public interface IEntityHandler
    {
        [OperationContract]
        IQueryable<Phone> RetrieveAllPhones();

        [OperationContract]
        void AddPhone(string id, string brand, string model, double price);

        [OperationContract]
        void DeletePhone(string phoneID);

        [OperationContract]
        Phone GetPhone(string phoneID);

        [OperationContract]
        void AddOrReplacePhone(string id, string brand, string model, double price);

        [OperationContract]
        IQueryable<Store> RetrieveAllStores();

        [OperationContract]
        void AddStore(string id, string storeName, string storeAddress, string
phoneModel, double phonePrice);

        [OperationContract]
        void DeleteStore(string storeID);

        [OperationContract]
        Store GetStore(string storeID);

        [OperationContract]
        void AddOrReplaceStore(string id, string storeName, string storeAddress, string
phoneModel, double phonePrice);
        [OperationContract]
        bool IsAlive();
```

```
    }
}
```

Listing 5 – EntityHandler_JobServer – Contract methods implementation

```
public void AddPhone(string id, string brand, string model, double price)
        {
            Phone newPhone = new Phone(id)
            {
                Brand = brand,
                Model = model,
                Price = price
            };
            repository.AddPhone(newPhone);
        }


public void AddOrReplacePhone(string id, string brand, string model, double price)
        {
            Phone phone = new Phone(id)
            {
                Brand = brand,
                Model = model,
                Price = price
            };
            repository.AddOrReplacePhone(phone);
        } //EDIT Phone
```

Navesti nazive resursa u Cloud-u (nazive tabela, kontejnera i sl.), parametri u okviru Queue-a

- Table, naziv tebele u bazi jestes: PhoneStoreTable, Služi za smeštanje entiteta koji se dodaju iz web role nakon izvršenja osnovnih CRUD operacija.. entity handler ih obrađuje I piše u table.
- Blob, naziv bloba u bazi jeste: blokkontejner, služi da bi entity handler smestio vrednost (da li je instanca živa ili nije) u kontejner za određeni dan.

- Queue, naziv u bazi jeste: healthmonitoring, služi da bi health monitoring prosledio vrednost (da li je instanca živa ili nije) notifier worker roli, tako što ona uzima iz queuq-a vrednost I dalje prosleđuje healthmonitoring konzolnoj app. I u bolob.

Primer scenaria rada I test slučja:

Scenario rada bi bio sledeci:

Prvo treba pokrenuti aplikaciju koja gadja default rutu,

```
            routes.MapRoute(
                name: "Phone",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Phone", action = "Index", id =
UrlParameter.Optional }
            );
```

Default ruta je: Phone/Index, zatim je potrebno pokrenuti, preko debug-start new isntance, HealtMonitoringApp projekat u koji ce da pristizu informacije da li je rola 1 aktivna tako I samostalna provera ostale 2 role da li su aktivne.. Kada se pokrene aplikacija otvorice se browser na toj ruti, odmah je moguć prikaz svih telefona koji su na stanju kao I u taskbaru je moguće odabrati I izlistavanje svih Radnji.

Ako se želi dodati novi telefon, onda se ide na create new opciju, zatim se popune podaci potrebi za telefon I to gadja prvu ili treću instancu u entity handler worker roli koja je zadužena za obrađivannje crud operacija primarnih entitet, za radnju je operacija identicna, samo što nju obradjuje 2 instanca entity handler worker role, jer je ona zadužena za sekundarne isntance.

Ako se želi obrisati neki telefon/radnja može se ići na operaciju Delete, za edit je potrebno kliknuti na operaciju edit. Instance koje obradjuju telefone (primarnu klasu) su : prva I druga, dok treća obrađuje sekundarnu klasu tj. Radnju

Za potrebne upise u bazu možete pogledati server explorel -> azure -> storage, I tada već imate pristup određenoj tabeli/kontejneru/redu.