Zadatak B – Rešenje.

Listing 1 – Implementacija *ResizeImage* metode (Zavisi od listinga 2 i listinga 3).

```
        /// <summary>
        /// Metoda pronalazi studenta po prosledjenom broju indeksa. Ukoliko student ne
postoji, ispisuje poruku o tome u compute emulatoru.
        /// Ukoliko student postoji, preuzima sliku, konvertuje je u manju sliku i vrsi
upload manje slike.
        /// cuva url do manje slike na mesto ThumbnailUrl.
        /// </summary>
        /// <param name="indexNo">broj indeksa studenta</param>
        public void ResizeImage(String indexNo)
        {
            StudentDataRepository sdr = new StudentDataRepository();
            Student student = sdr.GetStudent(indexNo);
            if (student == null)
            {
                Trace.TraceInformation(String.Format("Student sa brojem indeksa {0} ne
postoji!", indexNo), "Information");
                return;
            }

            BlobHelper blobHelper = new BlobHelper();
            string uniqueBlobName = string.Format("image_{0}", student.RowKey);


            Image image = blobHelper.DownloadImage("vezba", uniqueBlobName);
            image = ImageConvertes.ConvertImage(image);
            string thumbnailUrl = blobHelper.UploadImage(image, "vezba", uniqueBlobName +
"thumb");

            student.ThumbnailUrl = thumbnailUrl;
            sdr.UpdateStudent(student);
        }
```

Listing 2 – Dodatne metode u klasi *StudentDataRepository*.

```
public Student GetStudent(string index)
        {
            return RetrieveAllStudents().Where(p => p.RowKey == index).FirstOrDefault();
        }

        public void UpdateStudent(Student student)
        {
            TableOperation updateOperation = TableOperation.Replace(student);
            _table.Execute(updateOperation);        }
```

Listing 3 – Implementacija *BlobHelper* klase.

```
public class BlobHelper
    {
        CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting("DataConnectionString"));
```

```
        CloudBlobClient blobStorage;

        public BlobHelper()
        {
            blobStorage = storageAccount.CreateCloudBlobClient();
        }

        internal Image DownloadImage(String containerName, String blobName)
        {
            CloudBlobContainer container =
blobStorage.GetContainerReference(containerName);
            CloudBlockBlob blob = container.GetBlockBlobReference(blobName);

            using (MemoryStream ms = new MemoryStream())
            {
                blob.DownloadToStream(ms);
                return new Bitmap(ms);
            }
        }

        internal string UploadImage(Image image, String containerName, String blobName)
        {
            CloudBlobContainer container =
blobStorage.GetContainerReference(containerName);
            CloudBlockBlob blob = container.GetBlockBlobReference(blobName);

            using (MemoryStream memoryStream = new MemoryStream())
            {
                image.Save(memoryStream, System.Drawing.Imaging.ImageFormat.Bmp);
                memoryStream.Position = 0;
                blob.Properties.ContentType = "image/bmp";
                blob.UploadFromStream(memoryStream);
                return blob.Uri.ToString();
            }
        }
    }
```