**ProvisonSample.yaml**

- name: Quickstart Building

type: Venue

resources:

- type: IoTHub

spaces:

- name: Floor 3

type: Floor

spaces:

- name: Focus Room 30

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 30

hardwareId: 123456789030

sensors:

- dataType: Light

hardwareId: MAC000030

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 32

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 32

hardwareId: 123456789032

sensors:

- dataType: Light

hardwareId: MAC000032

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 33

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 33

hardwareId: 123456789033

sensors:

- dataType: Light

hardwareId: MAC000033

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 34

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 34

hardwareId: 123456789034

sensors:

- dataType: Light

hardwareId: MAC000034

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 35

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 35

hardwareId: 123456789035

sensors:

- dataType: Light

hardwareId: MAC000035

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 36

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 36

hardwareId: 123456789036

sensors:

- dataType: Light

hardwareId: MAC000036

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Floor 2

type: Floor

spaces:

- name: Focus Room 20

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 20

hardwareId: 123456789020

sensors:

- dataType: Light

hardwareId: MAC000020

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 21

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 21

hardwareId: 123456789021

sensors:

- dataType: Light

hardwareId: MAC000021

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 22

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 22

hardwareId: 123456789022

sensors:

- dataType: Light

hardwareId: MAC000022

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 23

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 23

hardwareId: 123456789023

sensors:

- dataType: Light

hardwareId: MAC000023

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 24

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 24

hardwareId: 123456789024

sensors:

- dataType: Light

hardwareId: MAC000024

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 25

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 25

hardwareId: 123456789025

sensors:

- dataType: Light

hardwareId: MAC000025

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 26

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 26

hardwareId: 123456789026

sensors:

- dataType: Light

hardwareId: MAC000026

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 27

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 27

hardwareId: 123456789027

sensors:

- dataType: Light

hardwareId: MAC000027

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 28

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 28

hardwareId: 123456789028

sensors:

- dataType: Light

hardwareId: MAC000028

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 29

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 29

hardwareId: 123456789029

sensors:

- dataType: Light

hardwareId: MAC000029

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Floor 0

type: Floor

spaces:

- name: Focus Room 02

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 02

hardwareId: 123456789002

sensors:

- dataType: Light

hardwareId: MAC000002

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 03

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 03

hardwareId: 123456789003

sensors:

- dataType: Light

hardwareId: MAC000003

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 04

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 04

hardwareId: 123456789004

sensors:

- dataType: Light

hardwareId: MAC000004

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 06

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 06

hardwareId: 123456789006

sensors:

- dataType: Light

hardwareId: MAC000006

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 07

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 07

hardwareId: 123456789007

sensors:

- dataType: Light

hardwareId: MAC000007

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 08

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 08

hardwareId: 123456789008

sensors:

- dataType: Light

hardwareId: MAC000008

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 09

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 09

hardwareId: 123456789009

sensors:

- dataType: Light

hardwareId: MAC000009

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Floor 1

type: Floor

spaces:

- name: Focus Room 10

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 10

hardwareId: 123456789010

sensors:

- dataType: Light

hardwareId: MAC000010

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 11

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 11

hardwareId: 123456789011

sensors:

- dataType: Light

hardwareId: MAC000011

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 12

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 12

hardwareId: 123456789012

sensors:

- dataType: Light

hardwareId: MAC000012

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 13

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 13

hardwareId: 123456789013

sensors:

- dataType: Light

hardwareId: MAC000013

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 16

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 16

hardwareId: 123456789016

sensors:

- dataType: Light

hardwareId: MAC000016

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 18

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 18

hardwareId: 123456789018

sensors:

- dataType: Light

hardwareId: MAC000018

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Focus Room 19

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 19

hardwareId: 123456789019

sensors:

- dataType: Light

hardwareId: MAC000019

matchers:

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

- name: Area A

type: Area

- name: Conference Room 11

type: Room

subType: ConferenceRoom

- name: Focus Room A1

type: Room

subType: FocusRoom

devices:

- name: Raspberry Pi 3 A1

hardwareId: 1234567890AB

sensors:

- dataType: Motion

hardwareId: SAMPLE\_SENSOR\_MOTION

- dataType: CarbonDioxide

hardwareId: SAMPLE\_SENSOR\_CARBONDIOXIDE

- dataType: Light

hardwareId: MAC000002

- dataType: Temperature

hardwareId: SAMPLE\_SENSOR\_TEMPERATURE

matchers:

- name: Matcher Motion A1

dataTypeValue: Motion

- name: Matcher CarbonDioxide A1

dataTypeValue: CarbonDioxide

- name: Matcher Electricity A1

dataTypeValue: None

- name: Matcher Temperature

dataTypeValue: Temperature

- name: Matcher Light

dataTypeValue: Light

userdefinedfunctions:

- name: Motion Processor

matcherNames:

- Matcher Motion A1

- Matcher CarbonDioxide A1

- Matcher Electricity A1

- Matcher Temperature

- Matcher Light

script: actions/userDefinedFunctions/availability.js

roleassignments:

- roleId: 98e44ad7-28d4-4007-853b-b9968ad132d1 # System Role: SpaceAdministrator

objectName: Motion Processor

objectIdType: UserDefinedFunctionId

**Database.sql**

CREATE TABLE [Power] (

id bigint(50) NOT NULL,

LCLid varchar(50) NOT NULL,

stdorToU varchar(50) NOT NULL,

DateTime varchar(50) NOT NULL,

KWH hh varchar(50) NOT NULL,

Acorn varchar(50) NOT NULL,

Acorn Grouped varchar(50) NOT NULL,

CONSTRAINT [PK\_POWER] PRIMARY KEY CLUSTERED

(

[id] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [registration] (

login varchar(50) NOT NULL,

password varchar(50) NOT NULL,

CONSTRAINT [PK\_REGISTRATION] PRIMARY KEY CLUSTERED

(

[login] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

CREATE TABLE [spaces] (

spaceId varchar(50) NOT NULL,

sensorId varchar(50) NOT NULL,

sensorName varchar(50) NOT NULL,

CONSTRAINT [PK\_SPACES] PRIMARY KEY CLUSTERED

(

[spaceId] ASC

) WITH (IGNORE\_DUP\_KEY = OFF)

)

GO

ALTER TABLE [registration] WITH CHECK ADD CONSTRAINT [registration\_fk0] FOREIGN KEY ([login]) REFERENCES [Power]([LCLid])

ON UPDATE CASCADE

GO

ALTER TABLE [registration] CHECK CONSTRAINT [registration\_fk0]

GO

ALTER TABLE [spaces] WITH CHECK ADD CONSTRAINT [spaces\_fk0] FOREIGN KEY ([sensorName]) REFERENCES [Power]([LCLid])

ON UPDATE CASCADE

GO

ALTER TABLE [spaces] CHECK CONSTRAINT [spaces\_fk0]

GO

**Availability.js**

var Type = "Light";

var StatusOk = "OK";

var ElectricityThreshold = 2.0;

function process(telemetry, executionContext) {

try {

log(`Sensor ID: ${telemetry.SensorId}. `);

log(`Sensor value: ${JSON.stringify(telemetry.Message)}.`);

var sensor = getSensorMetadata(telemetry.SensorId);

var parseReading = JSON.parse(telemetry.Message);

setSensorValue(telemetry.SensorId,sensor.DataType, parseRead-ing.SensorValue);

var parentSpace = sensor.Space();

var otherSensors = parentSpace.ChildSensors();

var ElectricitySensor = otherSensors.find(function(element) {

return element.DataType === motionType;

});

var ElectricityValue = getFloat-Value(ElectricitySensor.Value().Value);

var roomIsOk = "Room is OK";

var roomIsNotOk = "Room is not OK";

if(ElectricityValue < ElectricityThreshold) {

log(`${roomIsOk}. Electricity: ${ElectricityValue}.`);

setSpaceValue(parentSpace.Id, StatusOk, roomIsOk);

}

else {

log(`${roomIsNotOk}. Electricity: ${ElectricityValue}.`);

setSpaceValue(parentSpace.Id, StatusOk, roomIsNotOk);

parentSpace.Notify(JSON.stringify(roomIsNotOk));

}

}

catch (error)

{

log(`error: ${error.name} Message ${error.message}.`);

}

}

function getFloatValue(str) {

if(!str) {

return null;

}

return parseFloat(str);

}

**DeviceConnectivity.cs**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Net.NetworkInformation;

using System.Runtime.Serialization.Json;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Microsoft.Azure.Devices.Client;

using Microsoft.Azure.DigitalTwins.Samples.Models;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Configuration.Binder;

using System.Data.SqlClient;

using System.Data;

namespace Microsoft.Azure.DigitalTwins.Samples

{

public static class DBConnector

{

public static int GetShift(Dictionary<string, int> dict, string id)

{

int index = 0;

foreach (var el in dict)

{

if (el.Key == id)

{

index = el.Value;

return index;

}

}

return index;

}

public static int GetIndex(DataTable dt, string id)

{

for (int i = 0; i < dt.Rows.Count; i++)

if (dt.Rows[i][0].ToString() == id)

{

return i;

}

return -1;

}

public static DataTable Connect(string sql, string connection-String)

{

SqlConnection connection = new SqlConnec-tion(connectionString);

DataTable dt = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

using (connection)

{

connection.Open();

DataTable ds = new DataTable();

new SqlDataAdapter(sql, connection).Fill(ds);

dt = ds;

}

return dt;

}

}

class Program

{

static Dictionary<string, int> dict = new Dictionary<string, int>();

static string sql\_dt = "select \* from [Power-Networks-LCL-June2015(withAcornGps)v2\_1] as tbl order by tbl.LCLid asc, tbl.DateTime asc";

static string connectionString\_dt = @"Data Source=.\SQLEXPRESS;Initial Catalog=Electricity;Integrated Securi-ty=True";

private static DataTable dt = DBConnector.Connect(sql\_dt, connec-tionString\_dt);

static string sql\_reg = "select \* from [registration] as reg or-der by reg.login asc";

static string connectionString\_reg = @"Data Source=.\SQLEXPRESS;Initial Catalog=Electricity;Integrated Securi-ty=True";

private static DataTable reg = DBConnector.Connect(sql\_reg, con-nectionString\_reg);

private static IConfigurationSection settings;

static void Main(string[] args)

{

foreach (var el in dict)

Console.WriteLine(el);

for (int i = 0; i < reg.Rows.Count; i++)

dict.Add(reg.Rows[i][1].ToString(), DBConnector.GetIndex(dt, reg.Rows[i][1].ToString()));

settings = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build()

.GetSection("Settings");

try

{

DeviceClient deviceClient = DeviceCli-ent.CreateFromConnectionString(settings["DeviceConnectionString"]);

if (deviceClient == null)

{

Console.WriteLine("ERROR: Failed to create DeviceClient!");

return;

}

SendEvent(deviceClient).Wait();

}

catch (Exception ex)

{

Console.WriteLine("EXIT: Unexpected error: {0}", ex.Message);

}

}

static Func<string> CreateGetRandomSensorReading(int iteration, int shift)

{

string temp = dt.Rows[iteration + shift][3].ToString();

return () => temp.ToString(CultureInfo.InvariantCulture);

}

static async Task SendEvent(DeviceClient deviceClient)

{

var serializer = new DataContractJsonSerializ-er(typeof(CustomTelemetryMessage));

var sensors = settings.GetSection("Sensors").Get<Sensor[]>();

var delayPerMessageSend = int.Parse(settings["MessageIntervalInSeconds"]);

var countOfSendsPerIteration = sensors.Length;

var maxSecondsToRun = 15 \* 60;

var maxIterations = maxSecondsToRun / countOfSendsPerItera-tion / delayPerMessageSend;

var curIteration = 0;

do {

foreach (var sensor in sensors)

{

var getRandomSensorReading = CreateGetRandomSen-sorReading(curIteration, DBConnector.GetShift(dict, sen-sor.HardwareId.ToString()));

var telemetryMessage = new CustomTelemetryMessage()

{

SensorValue = getRandomSensorReading(),

};

using (var stream = new MemoryStream())

{

serializer.WriteObject(stream, telemetryMessage);

var byteArray = stream.ToArray();

Message eventMessage = new Message(byteArray);

eventMessage.Properties.Add("DigitalTwins-Telemetry", "1.0");

eventMessage.Properties.Add("DigitalTwins-SensorHardwareId", $"{sensor.HardwareId}");

eventMessage.Properties.Add("CreationTimeUtc", DateTime.UtcNow.ToString("o"));

eventMessage.Properties.Add("x-ms-client-request-id", Guid.NewGuid().ToString());

Con-sole.WriteLine($"\t{DateTime.UtcNow.ToLocalTime()}> Sending message: {En-coding.UTF8.GetString(eventMessage.GetBytes())} Properties: {{ {eventMes-sage.Properties.Aggregate(new StringBuilder(), (sb, x) => sb.Append($"'{x.Key}': '{x.Value}',"), sb => sb.ToString())} }}");

await deviceClient.SendEventAsync(eventMessage);

}

}

await Task.Delay(TimeSpan.FromSeconds(delayPerMessageSend));

} while (++curIteration < maxIterations);

Console.WriteLine($"Finished sending {curIteration} events (per sensor type)");

}

}

public class Sensor

{

public string DataType { get; set; }

public string HardwareId { get; set; }

}

}

**Appsettings.json**

{

"Settings": {

"DeviceConnectionString": "HostName=ih-7e3ee1ee-25a7-46cf-b3f1-dbcfc95895a9-1.azure-devices.net;DeviceId=e5411336-72e0-490c-8c95-13d72ba4288a;SharedAccessKey=hJcFjn5TZa3FYkCDlS23Cfs403JXLK4DvT61K6qH5pc=",

"MessageIntervalInSeconds": 5,

"Sensors": [{

"DataType": "Motion",

"HardwareId": "SAMPLE\_SENSOR\_MOTION"

},{

"DataType": "CarbonDioxide",

"HardwareId": "SAMPLE\_SENSOR\_CARBONDIOXIDE"

},{

"DataType": "Light",

"HardwareId": "MAC000002"

},{

"DataType": "Light",

"HardwareId": "MAC000003"

},{

"DataType": "Light",

"HardwareId": "MAC000004"

},{

"DataType": "Light",

"HardwareId": "MAC000006"

},{

"DataType": "Light",

"HardwareId": "MAC000007"

},{

"DataType": "Light",

"HardwareId": "MAC000008"

},{

"DataType": "Light",

"HardwareId": "MAC000009"

},{

"DataType": "Light",

"HardwareId": "MAC000010"

},{

"DataType": "Light",

"HardwareId": "MAC000011"

},{

"DataType": "Light",

"HardwareId": "MAC000012"

},{

"DataType": "Light",

"HardwareId": "MAC000013"

},{

"DataType": "Light",

"HardwareId": "MAC000016"

},{

"DataType": "Light",

"HardwareId": "MAC000018"

},{

"DataType": "Light",

"HardwareId": "MAC000019"

},{

"DataType": "Light",

"HardwareId": "MAC000020"

},{

"DataType": "Light",

"HardwareId": "MAC000021"

},{

"DataType": "Light",

"HardwareId": "MAC000022"

},{

"DataType": "Light",

"HardwareId": "MAC000023"

},{

"DataType": "Light",

"HardwareId": "MAC000024"

},{

"DataType": "Light",

"HardwareId": "MAC000025"

},{

"DataType": "Light",

"HardwareId": "MAC000026"

},{

"DataType": "Light",

"HardwareId": "MAC000027"

},{

"DataType": "Light",

"HardwareId": "MAC000028"

},{

"DataType": "Light",

"HardwareId": "MAC000029"

},{

"DataType": "Light",

"HardwareId": "MAC000030"

},{

"DataType": "Light",

"HardwareId": "MAC000032"

},{

"DataType": "Light",

"HardwareId": "MAC000033"

},{

"DataType": "Light",

"HardwareId": "MAC000034"

},{

"DataType": "Light",

"HardwareId": "MAC000035"

},{

"DataType": "Light",

"HardwareId": "MAC000036"

}]

}

}

**Appsettings.json**

{

"AADInstance": "https://login.microsoftonline.com/",

"ClientId": "8d2fe09a-d7e5-4d57-94cd-43939041a97d",

"Tenant": "e7e81f3b-3ea8-4dc5-bb40-d99c786d2c9a",

"BaseUrl": "https://ELECTRICITYDIGITALTWIN.northeurope.azuresmartspaces.net/management/api/v1.0/"

}

**Test.cs**

using System;

using System.Linq;

using Xunit;

using Microsoft.Azure.DigitalTwins.Samples;

using System.Net.Http;

using System.Net;

using System.Threading.Tasks;

using Newtonsoft.Json;

using System.Collections.Generic;

using Moq;

using System.IO;

using System.Collections;

using YamlDotNet.Serialization;

using Microsoft.Extensions.Logging;

namespace Microsoft.Azure.DigitalTwins.Samples.Tests

{

public class ProvisionSampleSensorsTests

{

private static Serializer yamlSerializer = new Serializer();

private static Guid sensor1Guid = new Guid("00000000-0000-0000-0000-000000000001");

private static Guid sensor2Guid = new Guid("00000000-0000-0000-0000-000000000002");

[Fact]

public async Task GetProvisionSampleCreatesDescriptions()

{

var yaml = @"

- name: Test1

devices:

- name: Device1

hardwareId: DeviceHardwareId1

sensors:

- dataType: SensorType1

hardwareId: SensorHardwareId1

";

var expectedDescriptions = new [] { new SpaceDescription()

{

name = "Test1",

devices = new [] {

new DeviceDescription()

{

name = "Device1",

hardwareId = "DeviceHardwareId1",

sensors = new [] {

new SensorDescription()

{

dataType = "SensorType1",

hardwareId = "SensorHardwareId1",

}

},

},

},

}};

var actualDescriptions = await Ac-tions.GetProvisionSampleTopology(new StringReader(yaml));

Assert.Equal(yamlSerializer.Serialize(expectedDescriptions), yamlSerializer.Serialize(actualDescriptions));

}

[Fact]

public async Task CreateTwoSensors()

{

(var httpClient, var httpHandler) = FakeDigitalTwinsHttpCli-ent.CreateWithDevice(

postResponseGuids: new [] { sensor1Guid, sensor2Guid },

getResponses: Enumerable.Repeat(Responses.NotFound, 2)

);

var descriptions = new [] { new SpaceDescription()

{

name = FakeDigitalTwinsHttpClient.Space.Name,

devices = new [] {

new DeviceDescription()

{

name = FakeDigitalTwinsHttpClient.Device.Name,

hardwareId = FakeDigitalTwinsHttpCli-ent.Device.HardwareId,

sensors = new [] {

new SensorDescription()

{

dataType = "SensorType1",

hardwareId = "SensorHardwareId1",

},

new SensorDescription()

{

dataType = "SensorType2",

hardwareId = "SensorHardwareId2",

}

}

}

},

}};

await Actions.CreateSpaces(httpClient, Loggers.SilentLogger, descriptions, Guid.Empty);

Assert.Equal(2, httpHandler.PostRequests["sensors"].Count);

}

}

}

**Endpoints.yaml**

- type: EventGrid

eventTypes:

- SensorChange

- SpaceChange

- TopologyOperation

- UdfCustom

connectionString: KbL4ssiPKdnXyP3t3hJ9y+1bfcQEphIB9wZdJuzBzgA=

secondaryConnectionString: lDXA0Cx09tX3kybaNjUdVGwaHJ69ie6o+qZpm2J/vgY=

path: notification.northeurope-1.eventgrid.azure.net

**GetSpaces.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Threading.Tasks;

using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

namespace Microsoft.Azure.DigitalTwins.Samples

{

public static partial class Actions

{

public static async Task GetSpaces(HttpClient httpClient)

{

Console.WriteLine("Polling spaces with 'ElectricityIsOk' value type");

var maxGets = 30;

for (var curGets = 0; curGets < maxGets; ++curGets)

{

var (spaces, response) = await GetManagementItemsAsync<Models.Space>(httpClient, "spaces", "includes=values");

if (spaces == null)

{

var content = await response.Content?.ReadAsStringAsync();

Console.WriteLine($"ERROR: GET spaces?includes=values failed with: {(int)response.StatusCode}, {response.StatusCode} {content}");

break;

}

var availableSpaces = spaces.Where(s => s.Values != null && s.Values.Any(v => v.Type == "ElectricityIsOk"));

if (availableSpaces.Any())

{

var availableDisplay = availableSpaces

.Select(s => GetDisplayValues(s))

.Aggregate((acc, cur) => acc + "\n" + cur);

Console.WriteLine($"{availableDisplay}");

}

else

{

Console.WriteLine("Unable to find a space with value type 'ElectricityIsOk'");

}

await Task.Delay(TimeSpan.FromSeconds(4));

}

}

private static async Task<(IEnumerable<T>, HttpResponseMessage)> GetManagementItemsAsync<T>(

HttpClient httpClient,

string queryItem,

string queryParams)

{

var response = await httpClient.GetAsync($"{queryItem}?{queryParams}");

if (response.IsSuccessStatusCode)

{

var content = await response.Content.ReadAsStringAsync();

var objects = JsonConvert.DeserializeObject<IEnumerable<T>>(content);

return (objects, response);

}

return (null, response);

}

private static string GetDisplayValues(Models.Space space)

{

var spaceValue = space.Values.First(v => v.Type == "ElectricityIsOk");

return $"\nId: {space.Id}\nTimestamp: {spaceValue.Timestamp}\nValue: {spaceValue.Value}\n}";

}

}

}

**AdminAPI.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Data;

using RestSharp;

using Newtonsoft.Json.Linq;

using Newtonsoft.Json;

public partial class AdminAPI : System.Web.UI.Page

{

public class Sensor

{

public int pollRate { get; set; }

public string spaceId { get; set; }

public string deviceId { get; set; }

public string id { get; set; }

public int portTypeId { get; set; }

public int dataUnitTypeId { get; set; }

public int dataTypeId { get; set; }

public int dataSubtypeId { get; set; }

public int typeId { get; set; }

public string hardwareId { get; set; }

}

public IRestResponse CreateSpace(string name, string id)

{

string bearer = (string)this.Page.Session["bearer"];

var yourobject = new Space

{

id = id,

name = name,

typeId = 14,

subtypeId = 13,

statusId = 12,

};

var json = JsonConvert.SerializeObject(yourobject);

var client = new RestClient("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/spaces/");

var request = new RestRequest(Method.POST);

request.AddParameter("application/json; charset=utf-8", json, ParameterType.RequestBody);

request.RequestFormat = DataFormat.Json;

request.AddHeader("Authorization", "Bearer " + bearer);

var response = client.Execute(request);

return response;

}

public IRestResponse CreateDevice(string name, string hardware, string id)

{

string bearer = (string)this.Page.Session["bearer"];

var yourobject = new Device

{

name = name,

typeId = 2,

subtypeId = 1,

hardwareId = hardware,

status = "Provisioned",

id = id

};

var json = JsonConvert.SerializeObject(yourobject);

var client = new RestClient("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/devices/");

var request = new RestRequest(Method.POST);

request.AddParameter("application/json; charset=utf-8", json, ParameterType.RequestBody);

request.RequestFormat = DataFormat.Json;

request.AddHeader("Authorization", "Bearer " + bearer);

var response = client.Execute(request);

return response;

}

public IRestResponse CreateSensor(string name, string id, string hardware, string devid)

{

string bearer = (string)this.Page.Session["bearer"];

var yourobject = new Sensor

{

pollRate = 0,

deviceId = devid,

id = id,

portTypeId = 8,

dataUnitTypeId = 7,

dataTypeId = 6,

dataSubtypeId = 5,

typeId = 9,

hardwareId = hardware

};

var json = JsonConvert.SerializeObject(yourobject);

var client = new RestClient("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/");

var request = new RestRequest(Method.POST);

request.AddParameter("application/json; charset=utf-8", json, ParameterType.RequestBody);

request.RequestFormat = DataFormat.Json;

request.AddHeader("Authorization", "Bearer " + bearer);

var response = client.Execute(request);

return response;

}

public class Device

{

public string name { get; set; }

public int typeId { get; set; }

public int subtypeId { get; set; }

public string hardwareId { get; set; }

public string spaceId { get; set; }

public string status { get; set; }

public string id { get; set; }

}

public class Space

{

public string id { get; set; }

public string name { get; set; }

public int typeId { get; set; }

public string parentSpaceId { get; set; }

public int subtypeId { get; set; }

public int statusId { get; set; }

}

protected static JObject Connector(string url, string id, string val, string bearer)

{

var client = new RestClient(url + id + "/" + val);

var request = new RestRequest(Method.GET);

request.AddHeader("cache-control", "no-cache");

request.AddHeader("Connection", "keep-alive");

request.AddHeader("accept-encoding", "gzip, deflate");

request.AddHeader("Host", "electricitydigitaltwin.northeurope.azuresmartspaces.net");

request.AddHeader("Cache-Control", "no-cache");

request.AddHeader("Accept", "\*/\*");

request.AddHeader("User-Agent", "PostmanRuntime/7.11.0");

request.AddHeader("Authorization", "Bearer " + bearer);

IRestResponse response = client.Execute(request);

string jsonString = response.Content;

JObject jObject = JObject.Parse(jsonString);

return jObject;

}

public string getSensorId(string sensorName)

{

string sensorId = "";

string spaceId = "";

string sensql = "select \* from [sensors] as sen order by sen.sensorName asc";

string regsql = "select \* from [registration] as reg order by reg.login asc";

string con = @"Data Source =.\SQLEXPRESS; Initial Catalog = Electricity; Integrated Security = True";

DataTable regdt = Connect(regsql, con);

DataTable sendt = Connect(sensql, con);

for (int i = 0; i < sendt.Rows.Count; i++)

if (sendt.Rows[i][0].ToString() == sensorName)

{

sensorId = sendt.Rows[i][1].ToString();

spaceId = sendt.Rows[i][2].ToString();

break;

}

return sensorId;

}

public static DataTable Connect(string sql, string connString)

{

SqlConnection connection = new SqlConnection(connString);

DataTable dt = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

using (connection)

{

connection.Open();

DataTable ds = new DataTable();

new SqlDataAdapter(sql, connection).Fill(ds);

dt = ds;

}

return dt;

}

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

string regsql = "select \* from [sensors] as reg order by reg.sensorName asc";

string con = @"Data Source =.\SQLEXPRESS; Initial Catalog = Electricity; Integrated Security = True";

DataTable regdt = Connect(regsql, con);

List<string> list = new List<string>();

for (int i = 0; i < regdt.Rows.Count; i++)

{

list.Add(regdt.Rows[i][0].ToString());

}

DropDownList1.DataSource = list;

DropDownList1.DataBind();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

string did = TextBox7.Text;

string dname = TextBox8.Text;

string sid = TextBox9.Text;

string sname = TextBox10.Text;

var dcontent = CreateDevice(dname,"DEV", did).Content;

TextBox1.Text = "Device with id:" + dcontent + " is successfully created";

var scontent = CreateSensor(sname, sid, "SEN", dcontent.ToString()).Content;

TextBox11.Text = "Sensor with id:" + scontent + " is successfully created";

}

protected void Button3\_Click(object sender, EventArgs e) //getvalue

{

string bearer = (string)this.Page.Session["bearer"];

string sensorName = DropDownList1.SelectedItem.Value;

string sensorId = getSensorId(sensorName);

TextBox3.Text = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("value");

}

protected void Button4\_Click(object sender, EventArgs e) //historical

{

string bearer = (string)this.Page.Session["bearer"];

string sensorName = DropDownList1.SelectedItem.Value;

string sensorId = getSensorId(sensorName);

TextBox4.Text = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[0].SelectToken("value").ToString() +

(string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[1].SelectToken("value").ToString() +

(string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[2].SelectToken("value").ToString();

}

protected void Button2\_Click(object sender, EventArgs e)

{

string id = TextBox5.Text;

string name = TextBox6.Text;

var content = CreateSpace(name, id).Content;

TextBox2.Text = "Room with id:" + content + " is successfully created";

}

}

**API.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using RestSharp;

using Newtonsoft.Json.Linq;

using System.Data.SqlClient;

using System.Data;

using System.Globalization;

public partial class API : System.Web.UI.Page

{

public string getSensorId()

{

string sensorName = (string)this.Page.Session["sensor"];

string sensorId = "";

string spaceId = "";

string sensql = "select \* from [sensors] as sen order by sen.sensorName asc";

string regsql = "select \* from [registration] as reg order by reg.login asc";

string con = @"Data Source =.\SQLEXPRESS; Initial Catalog = Electricity; Integrated Security = True";

DataTable regdt = Connect(regsql, con);

DataTable sendt = Connect(sensql, con);

for (int i = 0; i < sendt.Rows.Count; i++)

if (sendt.Rows[i][0].ToString() == sensorName)

{

sensorId = sendt.Rows[i][1].ToString();

spaceId = sendt.Rows[i][2].ToString();

break;

}

return sensorId;

}

protected void Page\_Load(object sender, EventArgs e)

{

}

public static DataTable Connect(string sql, string connString)

{

SqlConnection connection = new SqlConnection(connString);

DataTable dt = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

using (connection)

{

connection.Open();

DataTable ds = new DataTable();

new SqlDataAdapter(sql, connection).Fill(ds);

dt = ds;

}

return dt;

}

public float getDaily(string value1, string value2, string sensorName, DataTable sendt)

{

float result = 0.0f;

int sensorFirstIndex = 0;

int first = 0;

int second = 0;

for (int i = 0; i < sendt.Rows.Count; i++)

if (sendt.Rows[i][0].ToString() == sensorName)

{

sensorFirstIndex = i;

break;

}

for (int i = sensorFirstIndex; i < sendt.Rows.Count; i++)

if (sendt.Rows[i][3].ToString() == value1)

if (sendt.Rows[i+1][3].ToString() == value2)

{

first = i;

second = i + 1;

}

if (second - sensorFirstIndex < 48)

{

float sum = 0;

for (int i = sensorFirstIndex; i <= second; i++)

sum += float.Parse(string value2 = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[i].SelectToken("value").ToString());

return result = sum / (second - sensorFirstIndex);

}

else

{

float sum = 0;

for (int i = second - 48; i <= second; i++)

sum += float.Parse(string value2 = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[i].SelectToken("value").ToString());

return result = sum / 48;

}

}

protected static JObject Connector(string url, string id, string val, string bearer)

{

var client = new RestClient(url + id + "/" + val);

var request = new RestRequest(Method.GET);

request.AddHeader("cache-control", "no-cache");

request.AddHeader("Connection", "keep-alive");

request.AddHeader("accept-encoding", "gzip, deflate");

request.AddHeader("Host", "electricitydigitaltwin.northeurope.azuresmartspaces.net");

request.AddHeader("Cache-Control", "no-cache");

request.AddHeader("Accept", "\*/\*");

request.AddHeader("User-Agent", "PostmanRuntime/7.11.0");

request.AddHeader("Authorization", "Bearer " + bearer);

IRestResponse response = client.Execute(request);

string jsonString = response.Content;

JObject jObject = JObject.Parse(jsonString);

return jObject;

}

protected void Button1\_Click(object sender, EventArgs e)

{

string bearer = (string)this.Page.Session["bearer"];

string sensorId = getSensorId();

TextBox1.Text = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("value");

}

protected void Button2\_Click(object sender, EventArgs e)

{

string bearer = (string)this.Page.Session["bearer"];

string sensorId = getSensorId();

string sql = "select \* from [Power-Networks-LCL-June2015(withAcornGps)v2\_1] as tbl order by tbl.LCLid asc";

string con = @"Data Source =.\SQLEXPRESS; Initial Catalog = Electricity; Integrated Security = True";

DataTable dt = Connect(sql, con);

string value1 = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[0].SelectToken("value").ToString();

string value2 = (string)Connector("https://electricitydigitaltwin.northeurope.azuresmartspaces.net/management/api/v1.0/sensors/", sensorId, "value", bearer).SelectToken("historicalValues")[1].SelectToken("value").ToString();

TextBox2.Text = getDaily(value1, value1, sensorId, dt).ToString();

}

protected void Button3\_Click(object sender, EventArgs e)

{

string bearer = (string)this.Page.Session["bearer"];

string sensorName = (string)this.Page.Session["sensor"];

string sensorId = "";

string spaceId = "";

string sensql = "select \* from [sensors] as sen order by sen.sensorName asc";

string regsql = "select \* from [registration] as reg order by reg.login asc";

string con = @"Data Source =.\SQLEXPRESS; Initial Catalog = Electricity; Integrated Security = True";

DataTable regdt = Connect(regsql, con);

DataTable sendt = Connect(sensql, con);

for (int i = 0; i < sendt.Rows.Count; i++)

if (sendt.Rows[i][0].ToString() == sensorName)

{

sensorId = sendt.Rows[i][1].ToString();

spaceId = sendt.Rows[i][2].ToString();

break;

}

}

}

**Registration.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.SqlClient;

using System.Data;

using System.Runtime.Serialization.Json;

using System.Runtime.Serialization;

public partial class RegistrationForm : System.Web.UI.Page

{

string sensorName;

string bearer = "";

public static DataTable Connect()

{

string sql = "select \* from [registration] as reg order by reg.login asc";

const string connectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog=Electricity;Integrated Security=True";

SqlConnection connection = new SqlConnection(connectionString);

DataTable dt = new DataTable();

SqlDataAdapter adapter = new SqlDataAdapter(sql, connection);

using (connection)

{

connection.Open();

DataTable ds = new DataTable();

new SqlDataAdapter(sql, connection).Fill(ds);

dt = ds;

}

return dt;

}

protected void Page\_Load(object sender, EventArgs e)

{

}

public bool isValid(Dictionary<string,string> dict, string login, string password)

{

foreach (var el in dict)

{

if (el.Key == login && el.Value == password)

return true;

}

return false;

}

public bool isAdmin(string login, string password)

{

if (login == "admin" && password == "admin")

return true;

else

return false;

}

protected void Button1\_Click(object sender, EventArgs e)

{

string login = TextBox1.Text; ;

string password = TextBox2.Text;

DataTable dt = Connect();

Dictionary<string, string> dict = new Dictionary<string, string>();

if (isAdmin(login, password))

Server.Transfer("AdminAPI.aspx");

for (int i = 2; i < dt.Rows.Count; i++)

{

dict.Add(dt.Rows[i][1].ToString(), dt.Rows[i][2].ToString());

}

if (dict.ContainsKey(login))

{

if (isValid(dict, login, password))

{

sensorName = login;

this.Page.Session["sensor"] = sensorName;

Server.Transfer("API.aspx");

}

else

Label3.Text = "password isn't correct";

}

else

{

Label3.Text = "login does not exist";

}

}

protected void TextBox3\_TextChanged(object sender, EventArgs e)

{

bearer = TextBox3.Text;

this.Page.Session["bearer"] = bearer;

}

}