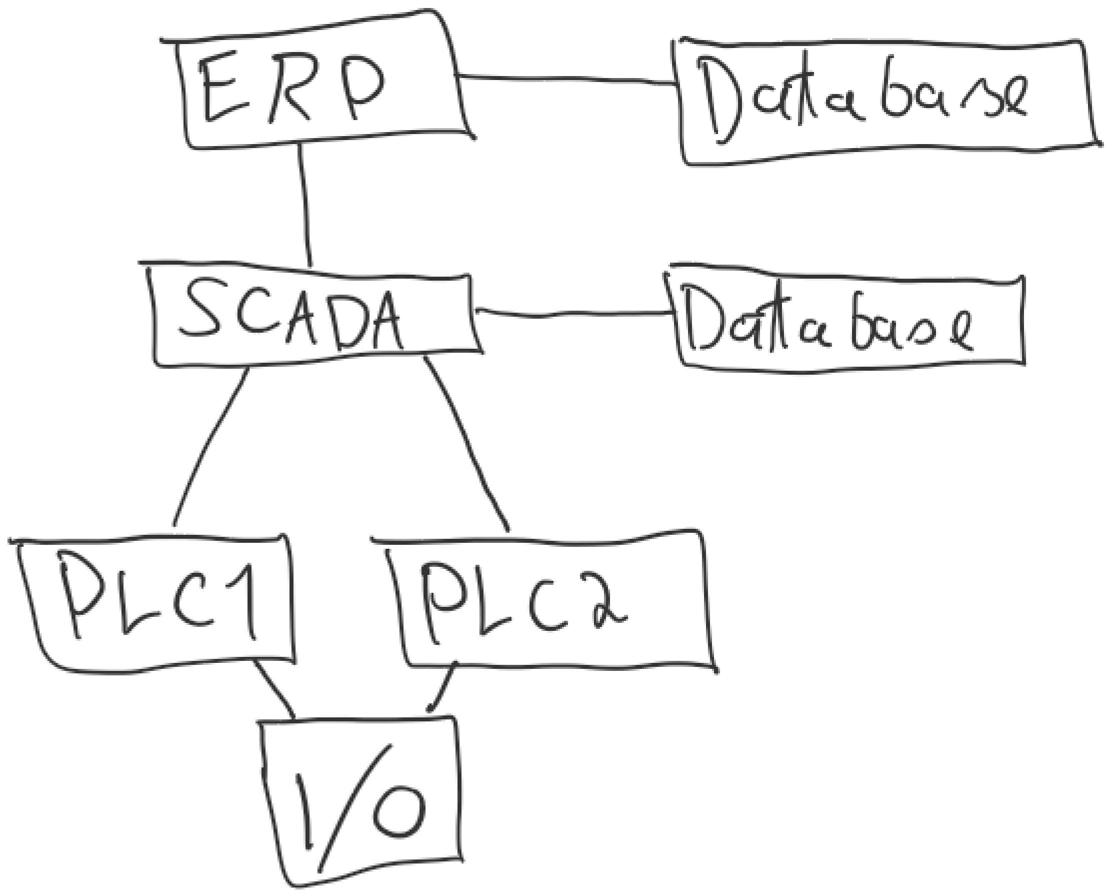
Exam 2022

Part 1

1. Introduction
    a. Drawn in OneNote



    b. The task asks us to describe the data flow between the modules, and the function of the connection between the modules. Between IO and PLC data is exchanged trough a bus protocoll, the
    c. Data Security
2. Analysis and design
    a. Four functional requirements would be: Predicting the need for maintenance, Monitor the system, Store the data, Adjust Controller output parameters.
    b. Two Non-functional requirements would be: Easy access of logged data, Intuitive SCADA system.
    c. This system could be autonomous. One should consider verifying the changes the monitoring system does to the controller parameters. But it could be done autonomously, as the main operation is analyzing the data, which is autonomous. There might be a challenge for the system to identify deviations that could have been avoided by predictive maintenance.
3. Digitalization

a. This is because there is to much data generated and consumed. The database should be used for data that's already processed. If ML queried too often, or asked for too much data, the SQL server could crash.
b. To clarify the data entry in the CSV file, the header will also be added.

```
TimeStamp, Instrument1, Instrument2, Instrument3, Instrument4, Instrument5
2022-05-19 09:00.000, 4.36, -16.86, 234.5, 1014.5, -0.5
```

c. When adding samples to a CSV file, its important to append to the existing one. SampleTemplate is the format for the data, ensures the right format for data being appended. For now only two input and two parameter outputs are added, to show the strucure and data format.

```
public void csvAppend(list<SampleTemplate> records)
            {
             string filepath = $"path\to\\file.csv";
            var config = new CSVConfig()
            {
                //Doesnt write header when appending
                WriteHeader = false;
            };
            //Sets mode to appending when opening file
            var stream = FileOpen(Mode = Append);
            //Creates stream for writing samples
            var writer = new Streamwriter(stream);
            //Appends samples, config is used to skip header.
            var csv = CsvWrite(writer, config)
            {
            csv.write(records);
            };

            };
```

```
public class SampleTemplate
    {
        string Timestamp { get; set; }
        double I_1 { get; set; }
        double I_2 { get; set; }
        double Parameter1 { get; set; }
        double Parameter2 { get; set; }

           }
```

4. Communication

a. Both the data link layer and the network layer is used to redirect data. The data link layer used the mac address of a unit to send data over through physical wires within a single network. While the network layer uses the IP addres to locate the the unit across networks when sending data.

b. that would not be possible. A switch, or router contains a table with ip addresses and its cooresponding MAC addresses of the devices connected to that network. The MAC adress is needed to identify each device in a network.

c. A Iot Device is most of the time lacking the presentation and application layer. A common application and and presentation layer is therefore used in the form of a cloud service for several IoT devices.

5. Multitasking

a. In a multitasking operating system, different tasks can either be running, stopped, or waiting for resources. A resource locked by a mutex is a reason a task is waiting to run. When the scheduler runs through the task states, it queues them if there are severeal tasks waiting for the same resource.

b. A task could be waiting for a resource to be freed before it runs. The resource could be locked by a mutex. When a task gets a hold of the resource its needs, it locks the resource with a mutex. If however, the task locks the resource with a mutex and then waits for the resource to be freed. It ends up in a deadlock.

c.
For predicting the need for maintenance, a module is needed that continuously analyzes the data that’s being produced by the system. Monitoring the system needs a module, this module continuously gathers the relevant information and formats the data before its being used for analysis. Adjusting the controller is also needed as a separate module, it needs to looks for parameter changes made by the other modules, and adjust them in the system at the right time. Storing the data is also needed as a separate module, this module writes data to a CSV file with a timestamp.

d. For a buffer like this, a queue would be the best option. When data is produced it gets added to the queue, when another task consumes this data it consumes the first entry, which removes the item from the queue, bringing other items forward.

```
public void Task1()
            {
            float Data;
            Data = Calculation();
            Queue.Add(Data);

            }

        public void Task2()
        {
            float calculatedData;
            if (Queue not NULL)
           {
            calculatedData = Queue.Use();
           }
        }

        }
```

6. Alarm system

a. The monitoring system should have an alarm system, notifying the operator when predictive maintenance is needed, as this maintenance often means action needed from an operator. Additionally there should be alarms for status of the ICS system, there could be problems with data links that needs to be highlighted.

b. The alarm suppression module uses logic to decide which alarms that are going to be displayed for the operator. The Alarm shelving module moves alarms to the shelved list. This is done manually by the operator.

c. Because this monitor system, is a seperate system, only used for predictive maintenance, theres no need for shelving alarms. An overview and detailed display is hovewer needed. This will not change the design drasticly, but seperate tables for each of the lists is going to be added.

d.

A method for high alarms only is created. The method checks the config method, if there is a high alarm attached to it. If so, and the high level is true. A storage procedyre is executed in SQL.

```
     public void HighAlarm(int tagId, float value)
     {
         var config = new alarmConfig();
         if (config.ContainsHighAlarm(tagId))
         {
             if (value > config.HighLimit(tagId))
             {
                 //Adds alarm to database using storage procedure
                 string query = $"Exec AlarmInsert @tagId = {tagId}
@Value = {value}";
                 ExecuteQuery(query);
             }
         }
     }
```

Part 2

Task 7

The tank function and low pass filter is created as a seperate VI.

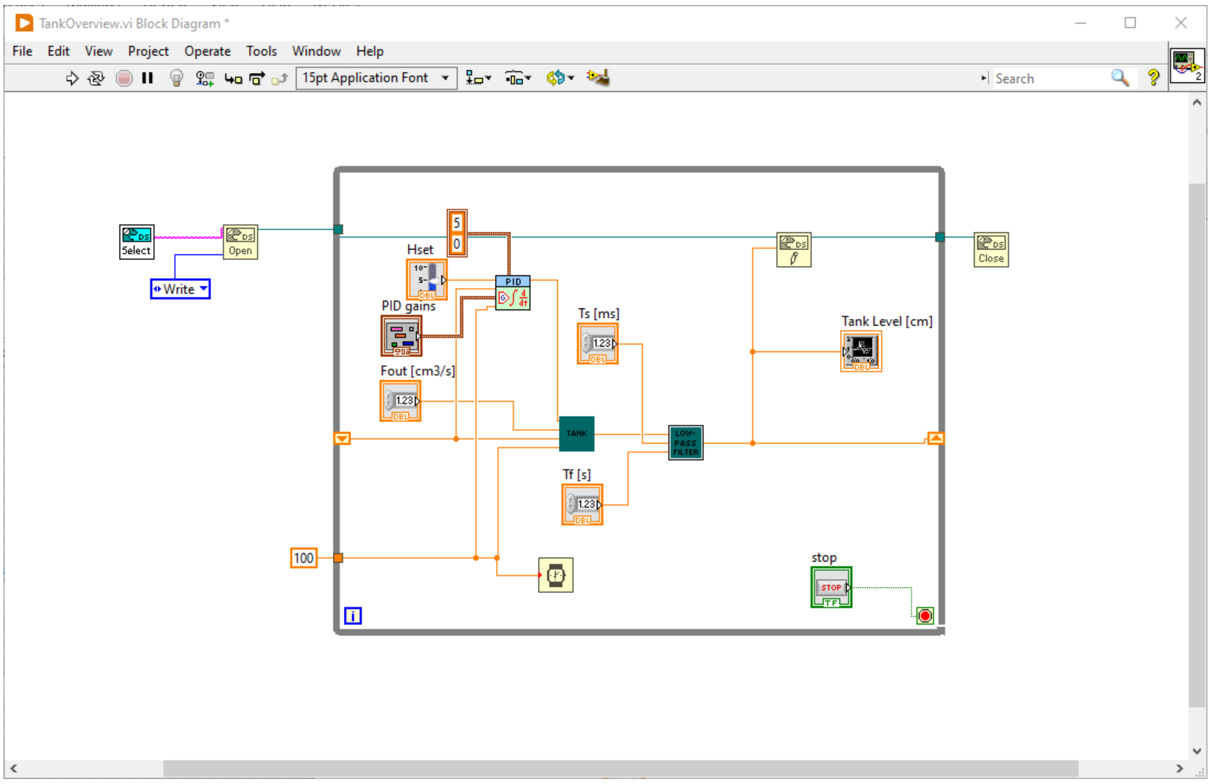


Figure 0-1: Code for Uploading tag data to OPC server

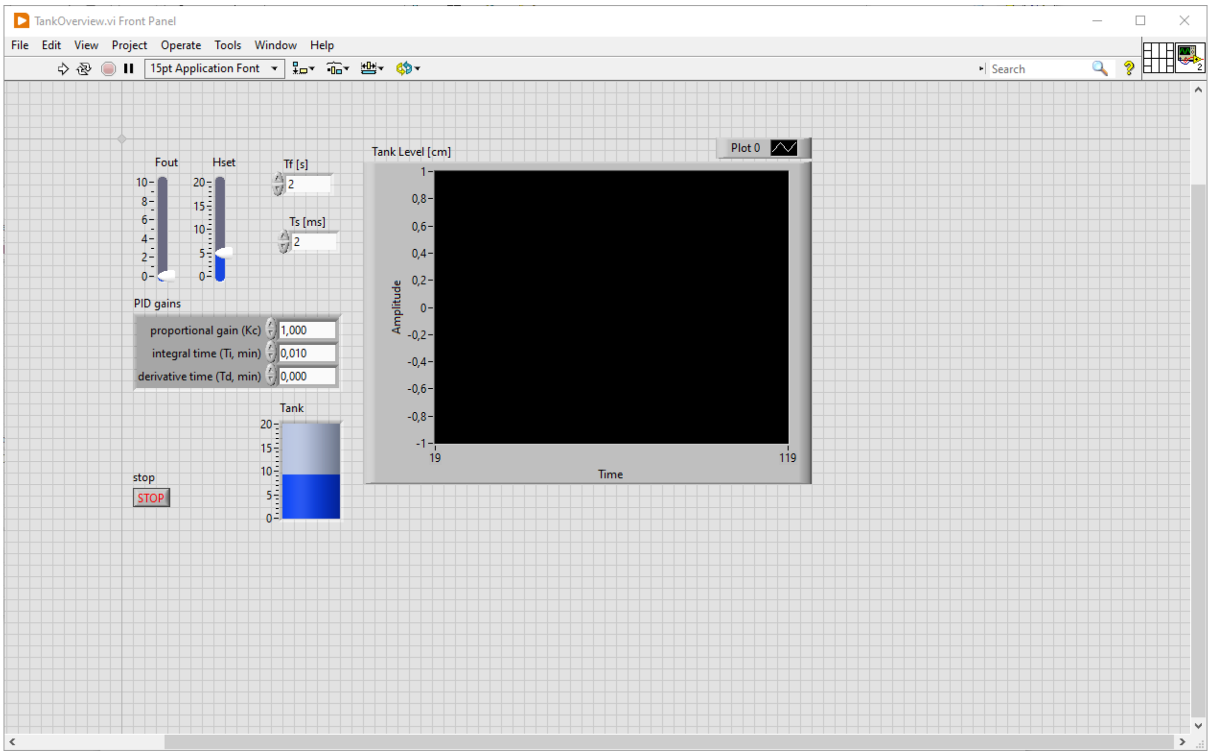

TankOverview.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Fout
Hset
Tf [s]
2
Ts [ms]
2
PID gains
proportional gain (Kc) 1,000
integral time (Ti, min) 0,010
derivative time (Td, min) 0,000
Tank
stop
STOP
Tank Level [cm]
Plot 0
Amplitude
Time
19
119

Figure 0-2: GUI for tank control

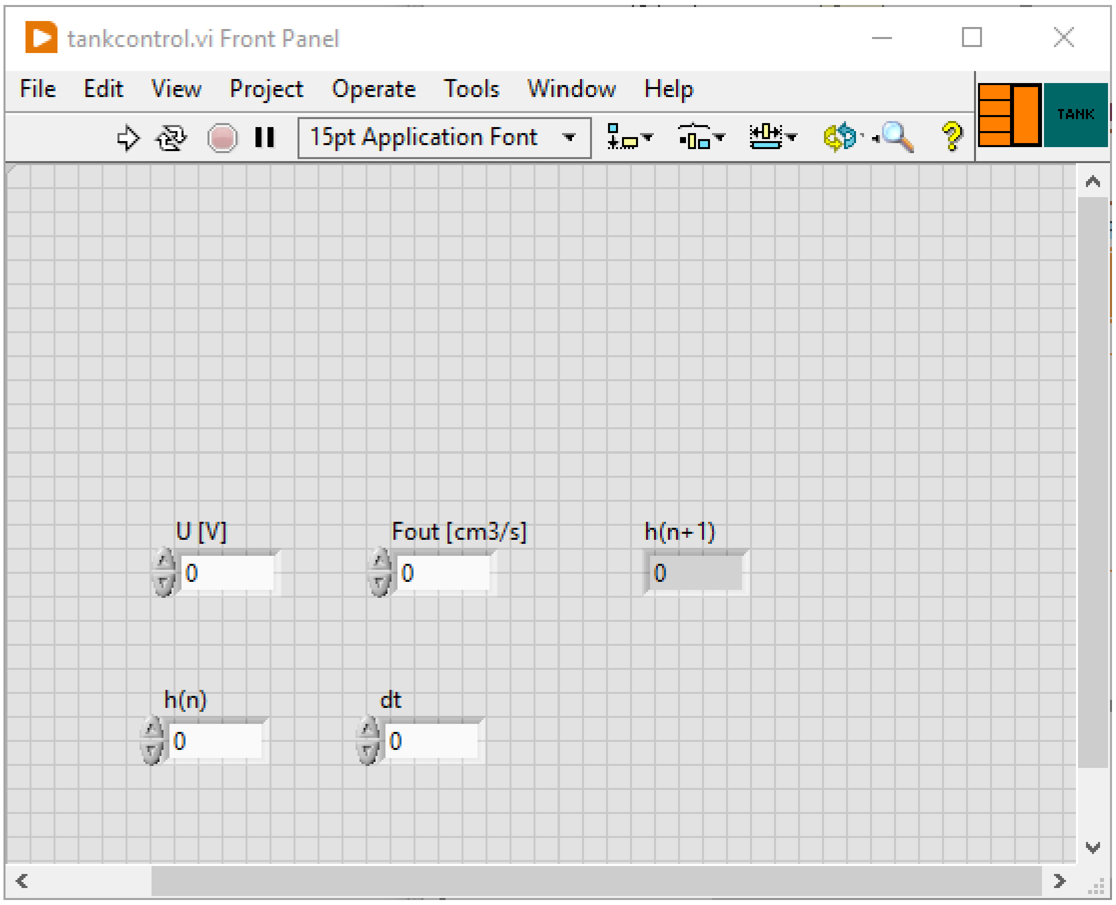
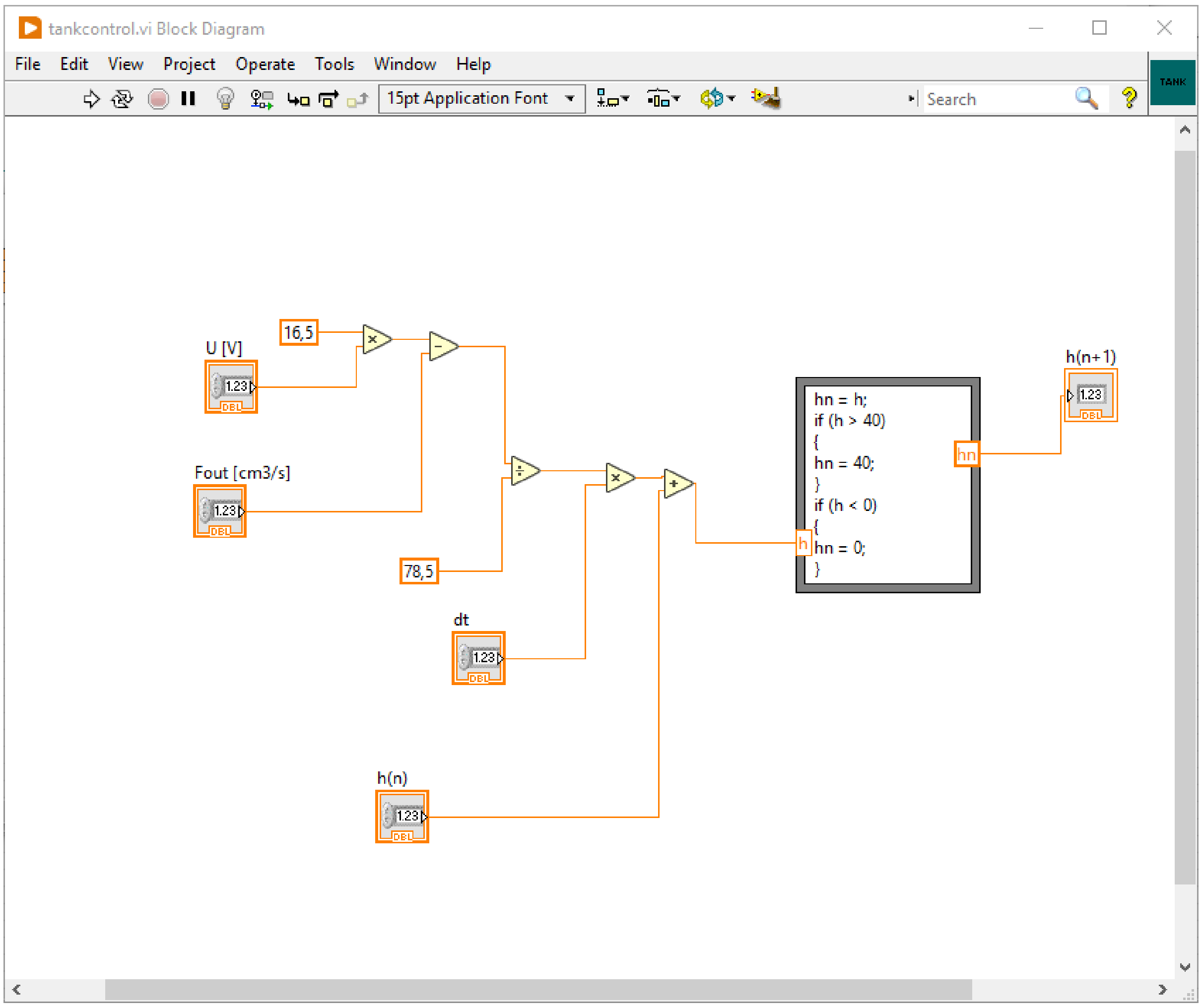

tankcontrol.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
TANK
U [V]
0
Fout [cm3/s]
0
h(n+1)
0
h(n)
0
dt
0

Figure 0-3: GUI Sub VI for tank


tankcontrol.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
TANK
U [V]
16,5
Fout [cm3/s]
78,5
dt
h(n)
h
hn
h(n+1)

```
hn = h;
if (h > 40)
{
hn = 40;
}
if (h < 0)
{
hn = 0;
}
```

Figure 0-4:SubVi for tank

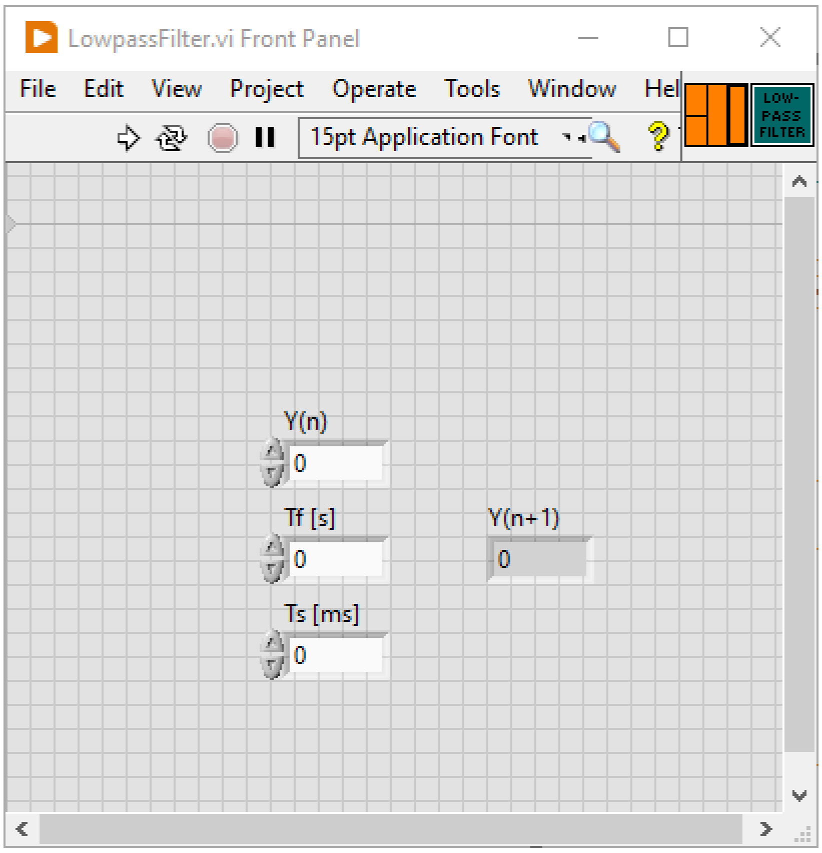
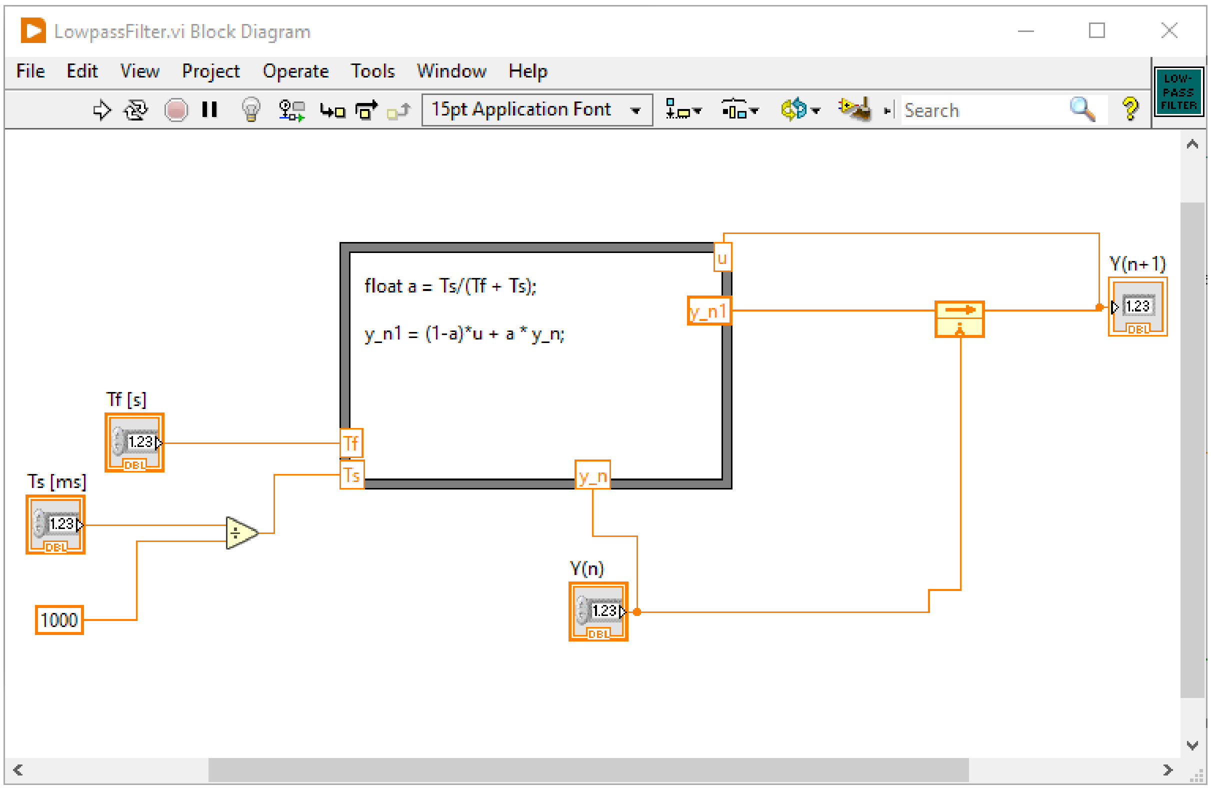

LowpassFilter.vi Front Panel
File Edit View Project Operate Tools Window Hel
15pt Application Font
LOW-PASS FILTER
Y(n)
0
Tf [s]
0
Ts [ms]
0
Y(n+1)
0

Figure 0-5: LowPass Filter GUI SubVI



```
float a = Ts/(Tf + Ts);

y_n1 = (1-a)*u + a * y_n;
```

Figure 0-6: Sub VI for filter

Task 8

Figure 0-7 shows the database. There is a table for tag configuration, with TagId as PK. It contains a description of the tag, the engineering unit, the location, and type of tag, for example a sensor tag.

The tag data table has a timestamp combined with the tagid as the primary key. Contains the value of the tag, and its engineering unit. The alarm configuration has AlarmId as PK, the table contains alarm type, the severity, and a description of the alarm. The Alarm data table combines the timestamp and alarmid as PK, if its acknowledged and shelving. The relationship between the configuration table and data table is one to zero, one or many. This means that for one configuration entry, there can be several datal entries of data.

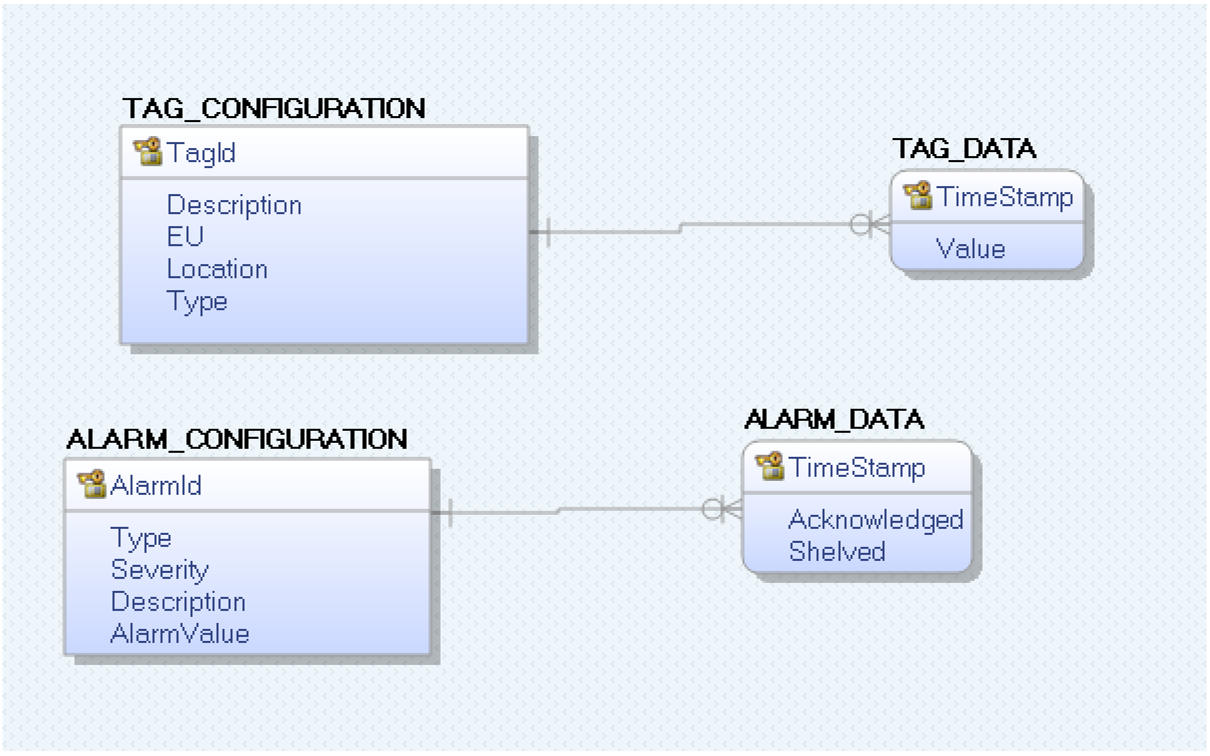


Figure 0-7: Database configuration for SQL

Part 9

When the storage procedure runs, it inserts data into the SQL table. With a timestamp being the time the storage procedure is being run.

```
CREATE PROCEDURE DataInsert @TagId INTEGER, @Value FLOAT
as

        INSERT INTO TAG_DATA (TimeStamp, TagId, Value)
        VALUES (GETDATE(),@TagId, @Value)
```

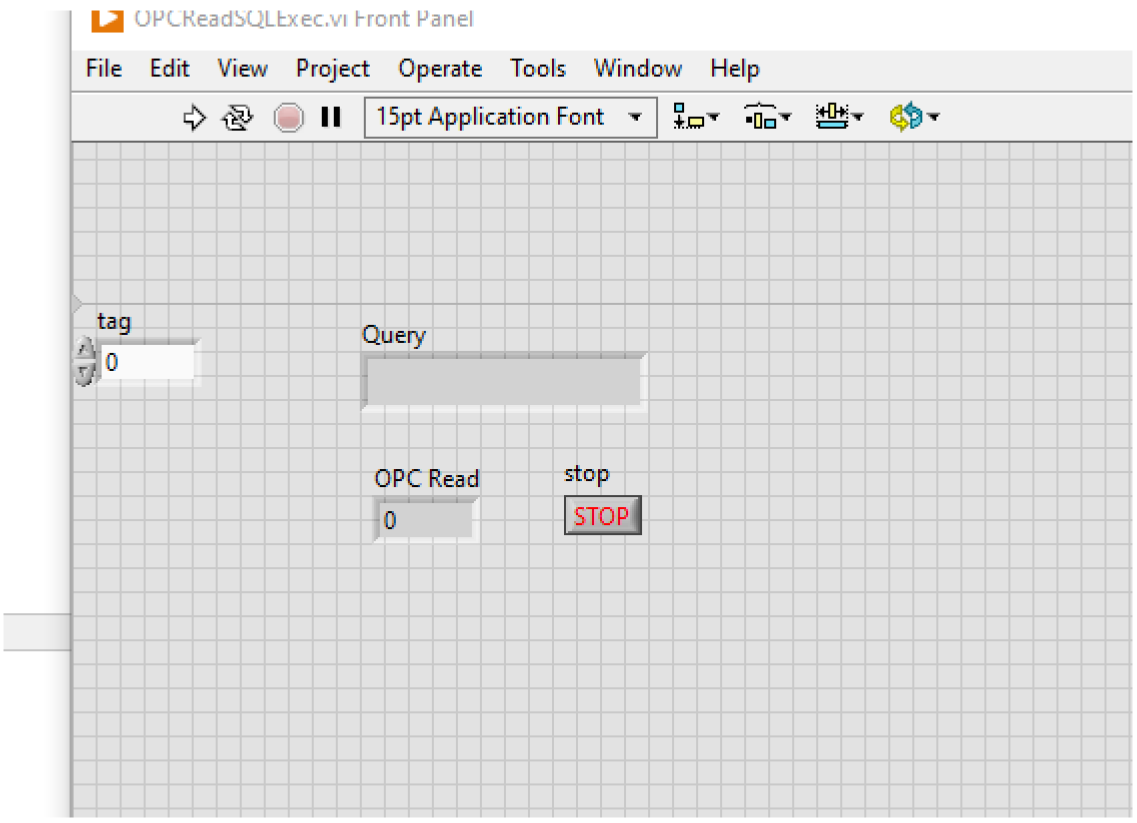

OPCReadSQLExec.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
tag
0
Query
OPC Read
0
stop
STOP

Figure 0-8: GUI for opc read and SQL query

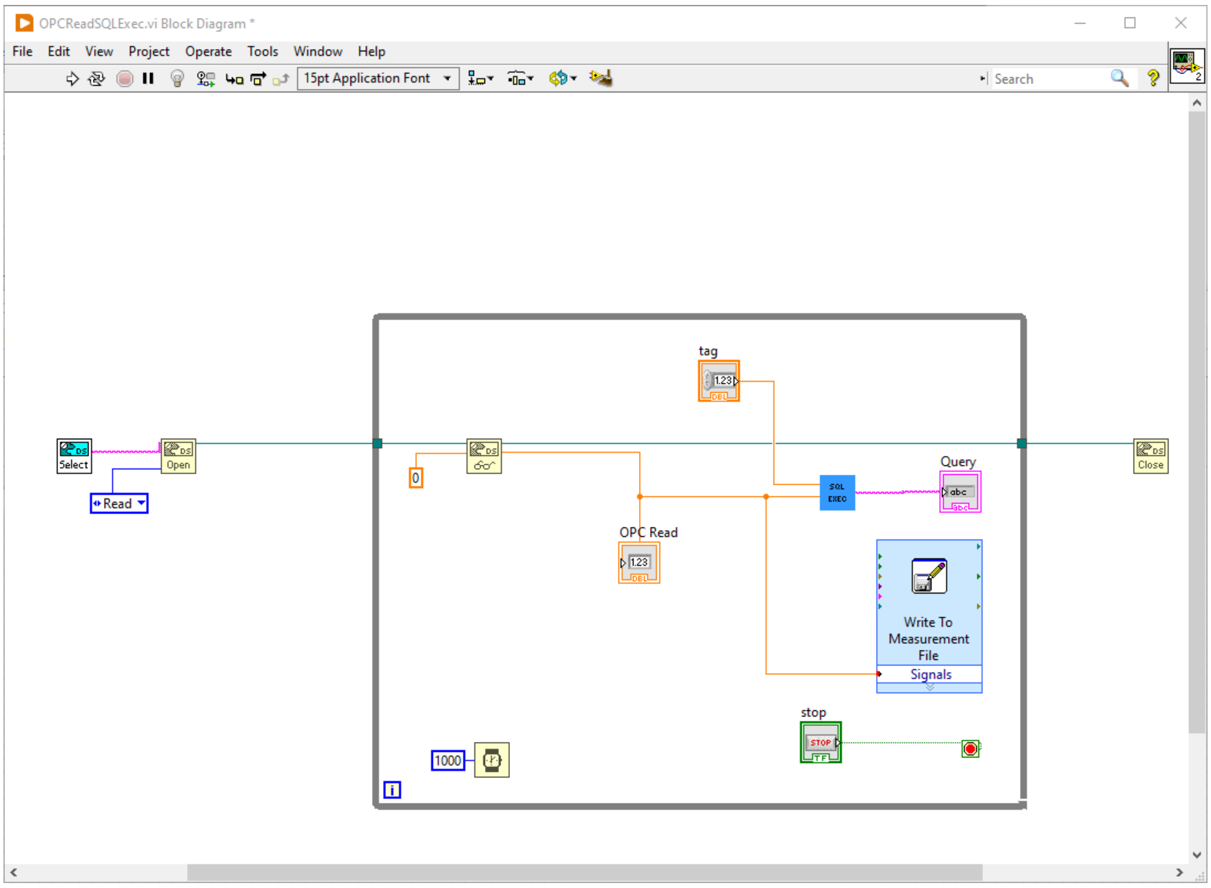
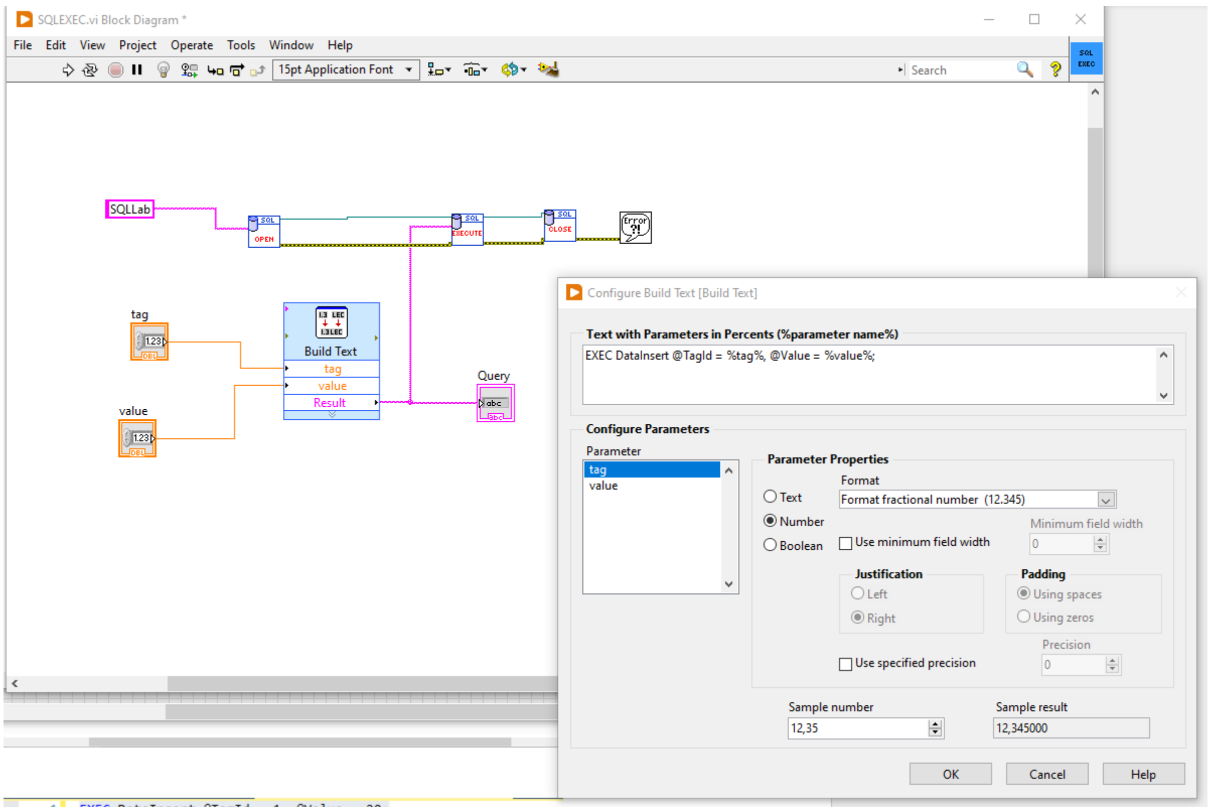

OPCReadSQLExec.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Select
Open
Read
tag
0
OPC Read
SQL EXEC
Query
Write To Measurement File
Signals
stop
1000
Close

Figure 0-9: OPC Read and SQL SubVI


SQLEXEC.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
SQLLab
OPEN
EXECUTE
CLOSE
tag
value
Build Text
Result
Query
Configure Build Text [Build Text]

Text with Parameters in Percents (%parameter name%)

```
EXEC DataInsert @TagId = %tag%, @Value = %value%;
```

Configure Parameters

Parameter: [✓ tag] value

Parameter Properties

Text [✓ Number] Boolean

Format: Format fractional number (12.345)

Use minimum field width ☐ — Minimum field width: 0

Justification: Left [✓ Right]

Padding: [✓ Using spaces] Using zeros

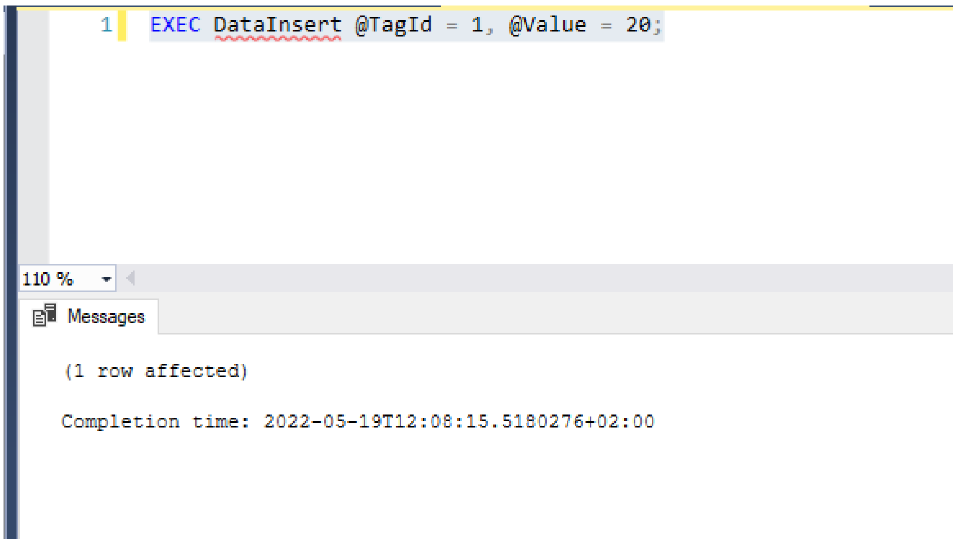
Use specified precision ☐ — Precision: 0

Sample number: 12,35

Sample result: 12,345000

OK Cancel Help

Figure 0-10: Sub VI for executing Storage procedure



```
EXEC DataInsert @TagId = 1, @Value = 20;
```

110 %

Messages

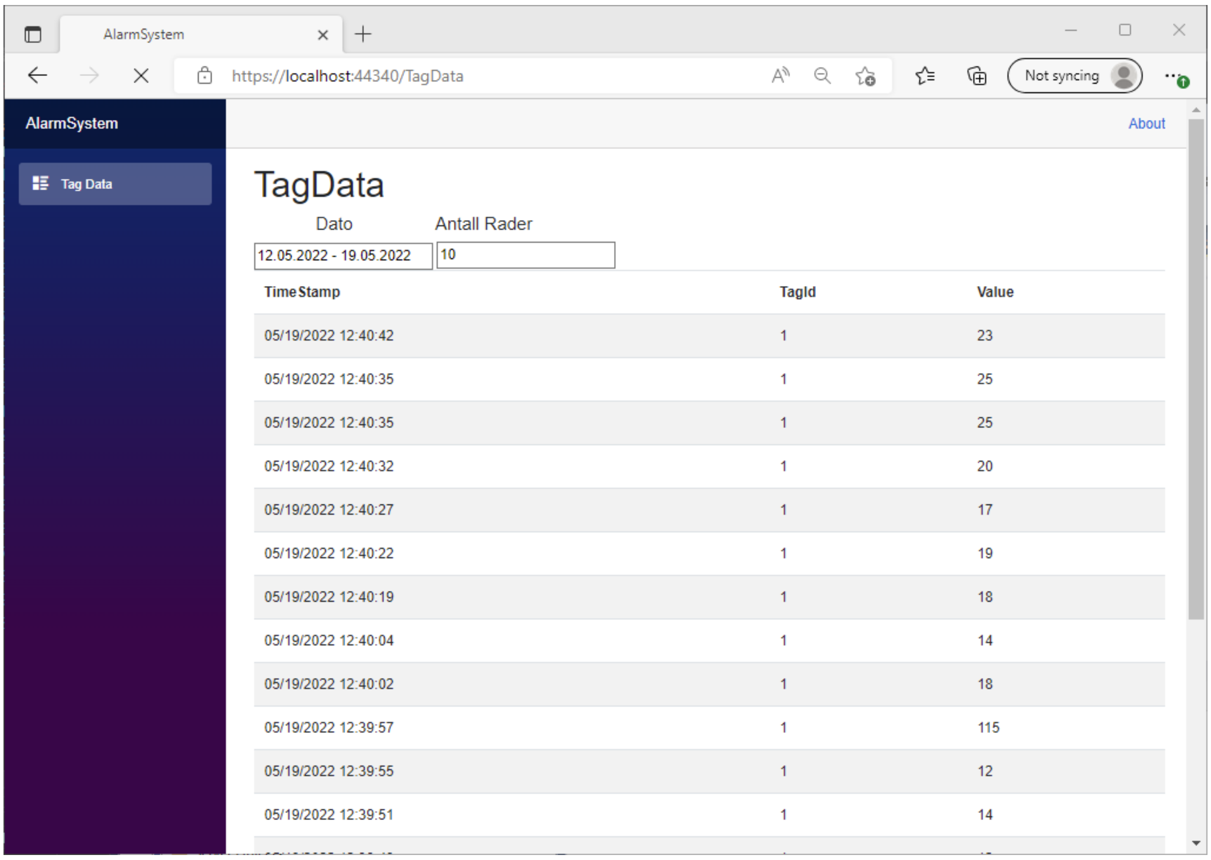
```
(1 row affected)

Completion time: 2022-05-19T12:08:15.5180276+02:00
```

Figure 0-11: Storage procedure tested

Part 10

The data is being displayed in a table in a blazor application. The top value is the current value. With the entries descending by timestamp down. The code for the web application is from previous SCADA project.



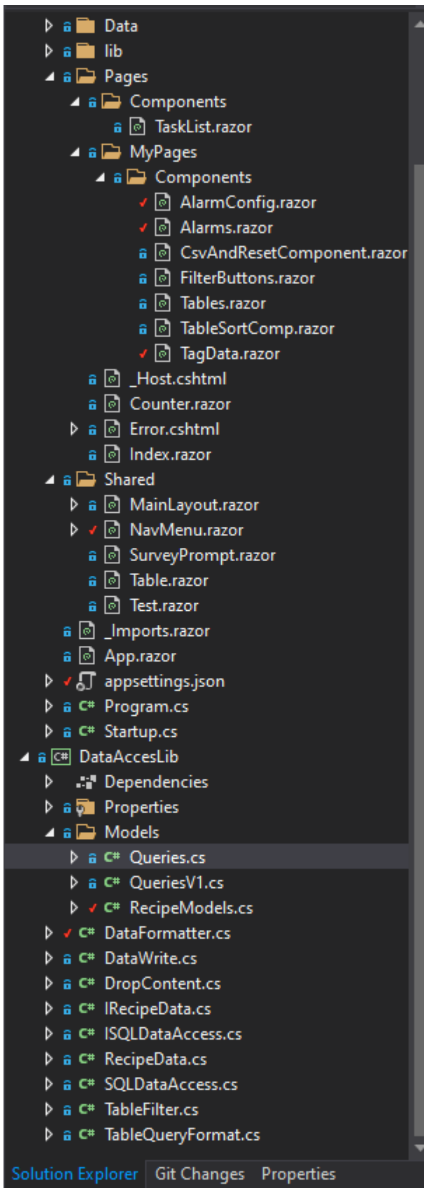
TagData

Dato	Antall Rader
12.05.2022 - 19.05.2022	10

TimeStamp	TagId	Value
05/19/2022 12:40:42	1	23
05/19/2022 12:40:35	1	25
05/19/2022 12:40:35	1	25
05/19/2022 12:40:32	1	20
05/19/2022 12:40:27	1	17
05/19/2022 12:40:22	1	19
05/19/2022 12:40:19	1	18
05/19/2022 12:40:04	1	14
05/19/2022 12:40:02	1	18
05/19/2022 12:39:57	1	115
05/19/2022 12:39:55	1	12
05/19/2022 12:39:51	1	14

Figure 0-12: Data from SQL displayed in Blazor application

The data is added with the storage procedure, and can be seen in Figure 0-12.



```
    Data
    lib
    Pages
      Components
        TaskList.razor
      MyPages
        Components
          AlarmConfig.razor
          Alarms.razor
          CsvAndResetComponent.razor
          FilterButtons.razor
          Tables.razor
          TableSortComp.razor
          TagData.razor
      _Host.cshtml
      Counter.razor
      Error.cshtml
      Index.razor
    Shared
      MainLayout.razor
      NavMenu.razor
      SurveyPrompt.razor
      Table.razor
      Test.razor
    _Imports.razor
    App.razor
    appsettings.json
    Program.cs
    Startup.cs
  DataAccesLib
    Dependencies
    Properties
    Models
      Queries.cs
      QueriesV1.cs
      RecipeModels.cs
    DataFormatter.cs
    DataWrite.cs
    DropContent.cs
    IRecipeData.cs
    ISQLDataAccess.cs
    RecipeData.cs
    SQLDataAccess.cs
    TableFilter.cs
    TableQueryFormat.cs
```

Figure 0-13: Structure of the code used.

```
using System.Diagnostics;

namespace DataAccesLib.Models
{
    public class QueriesV1
    {
        public static int TableLen = 10;
        public static string sql = $"SELECT top ({ TableLen }) * FROM
PIX318_ReseptData ORDER BY BatchNr DESC";
        public string _tableName;


        public void SetTableType(string TableName)
        {
            _tableName = TableName;
        }

        public string TableUpdate()
        {
            string sql = "";
            if (_tableName == "Alarms")
            {
                //sql = $"SELECT top({ TableLen }) * FROM ALARM_DATA ORDER BY
ActivationTimeStamp DESC";
                sql = $"SELECT TOP 20 * FROM ALARM_DATA WHERE Acknowledge = 0 ORDER
BY ActivationTimeStamp DESC";
            }
            else if (_tableName == "Tags")
            {
                sql = $"SELECT TOP 20 * FROM TAG_DATA ORDER BY TimeStamp DESC";
            }
            else if (_tableName == "AlarmConfig")
            {
                sql = $"SELECT * FROM ALARM_CONFIGURATION ORDER BY AlarmId DESC";
            }


            return sql;

        }

        public string AlarmAck(string AlarmId, string timestamp)
        {
            string sql = $"UPDATE ALARM_DATA SET Acknowledge = 1 WHERE AlarmId =
{AlarmId} AND ActivationTimeStamp between '{timestamp}.000' AND '{timestamp}.999'";
            Debug.WriteLine(sql);

            return sql;
        }

        public string AlarmAckQuery()
        {

            string sql = $"SELECT TOP 20 * FROM ALARM_DATA WHERE Acknowledge = 1
ORDER BY ActivationTimeStamp DESC";
            return sql;
        }
    }
}
```

```
using System.Collections.Generic;

namespace DataAccesLib.Models
{
    public class RecipeModels
    {
        public int BatchNr { get; set; }

        public string Dato { get; set; }
        public int SAP { get; set; }
        public string ID { get; set; }
        public int Reaktor { get; set; }

        public double SatsVolum { get; set; }

        public double ForvFe { get; set; }
        public double OnsketFe { get; set; }
        public double OnsketSyre { get; set; }
        public double OnsketFe2 { get; set; }
        public double HCLType { get; set; }
        public double ForvDamp { get; set; }
        public double VannOverordnet { get; set; }
        public double VarmtVann { get; set; }
        public double SpillVann { get; set; }
        public double ScrubberVaeske { get; set; }
        public double HCL { get; set; }
        public double Jernsulfat { get; set; }
        public double Temp { get; set; }
        public double Modningstid { get; set; }
        public double DampVentil { get; set; }
        public bool Etterspyling { get; set; }
        public double O2Trykk { get; set; }
        public double O2Reaksjonstid { get; set; }
        public double DeltaTemp { get; set; }
        public double AnalysertFe3 { get; set; }
        public double AnalysertFeTot { get; set; }
        public double VannSluttJustering { get; set; }
        public double VirkeligMVann { get; set; }
        public double TotTilLager { get; set; }

        //Analysis
        public double TotalFe { get; set; }
        public double Egenvekt { get; set; }
        public double Verdi2Fe { get; set; }
        public double ManuellVerdi2Fe { get; set; }
        public double Verdi3EtterManuell2Fe { get; set; }
        public double Verdi3Fe { get; set; }
        public double FriSyre { get; set; }



        public bool filter { get; set; }
        public bool edit { get; set; }


    }

    public class SqlData
    {
        public string Name1 { get; set; }
        public string Name2 { get; set; }
        public string Name3 { get; set; }
        public string Name4 { get; set; }
        public string Name5 { get; set; }
        public string Name6 { get; set; }
        public string Name7 { get; set; }
        public string Name8 { get; set; }
        public string Name9 { get; set; }
        public string Name10 { get; set; }
    }

    public class AlarmConfig
    {
        public string AlarmId { get; set; }
        public string SeverityName { get; set; }
        public string Tag { get; set; }
        public string AlarmType { get; set; }
        public string AlarmDescription { get; set; }
        public bool Disable { get; set; }



    }
```

```
public class AlarmConf
{
    public string AlarmId { get; set; }

    public string TagId { get; set; }
    public string AlarmType { get; set; }

    public string AlarmDescription { get; set; }
    public string SeverityName { get; set; }
    public bool Disable { get; set; }
}

public class AlarmType
{
    public string Alarm { get; set; }

}

public class SeverityConfiguration
{
    public string SeverityName { get; set; }
    public int SeverityLevel { get; set; }

}

public class TagLog
{
    public string TimeStamp { get; set; }
    public string TagId { get; set; }
    public string Value { get; set; }
    //public string Quality { get; set; }
    //public string Status { get; set; }
}


public class AlarmData
{
    public string AlarmId { get; set; }
    public string ActivationTimeStamp { get; set; }
    public string AcknowledgeTimeStamp { get; set; }
    public string AcknowledgeOperator { get; set; }
    public string Silence { get; set; }
    public string SilenceDuration { get; set; }
    public string SeverityName { get; set; }
    public bool Acknowledge { get; set; }
}

public class TagData
{
    public string TimeStamp { get; set; }
    public string TagId { get; set; }
    public string Value { get; set; }
    public string Quality { get; set; }
    public string Status { get; set; }

}

public class TagConfig
{
    public int TagId { get; set; }
    public string TagName { get; set; }
```

```
        public string ItemId { get; set; }
        public string Desc { get; set; }
    }



    public static class Extensions
    {
        public static string[] AlarmConfigExtensions = new string[]
        {
            "AlarnId",
            "Severityame",
            "TagId",
            "AlarmType",
            "AlarmDescription",
            "Disable",
            "Value"
        };

        public static string[] AlarmDataExtensions = new string[]
        {
            "AlarmId",
            "ActivationTimeStamp",
            "Ack TimeStamp",
            "Ack Operator",
            "Silence",
            //"Silence Duration",
            "Severity Name",
            "Acknowledge"


        };

        public static string[] TagDataExtensions = new string[]
        {
            "TimeStamp",
            "TagId",
            "Value",
            "Quality",
            "Status"
        };


    }


    //public static class FilterModel
    //{
    //    //make this a list
    //    public static bool[] filter { get; set; } = new bool[26];
    //    public static string[] StringFilter = new string[]
    //    {

    //    "BatchNr",

    //     "Dato",
    //     "SAP",
    //     "ID",
    //     "Reaktor",
    //     "Satsvolum",
    //     "ForvFe",
    //     "OnsketFe",
    //      "OnsketSyre",
    //      "OnsketFe2",
    //      "HCLType",
    //      "ForvDamp",
    //      "VannOverordnet",
    //      "VarmtVann",
    //      "SpillVann",
    //      "ScrubberVaeske",
    //      "HCL",
    //      "Jernsulfat",
    //      "Temp",
    //      "Modningstid",
    //      "Dampventil",
    //      "Etterspyling",
    //      "O2Trykk",
    //      "O2Reaksjonstid",
    //     "DeltaTemp",
    //     //"AnalysertFe3",
    //     //"AnalysertFeTot",

    //     "VannSluttjustering",
    //     "VirkeligMVann",
    //     "TotTilLager",
    //     "TotalFe",
    //     "Egenvekt",
    //     "Verdi2Fe",
    //     "ManuellVerdi2Fe",
    //     "Verdi3EtterManuell2Fe",
    //     "Verdi3Fe",
    //     "FriSyre",


    //};


    //}

    public class filtering
    {
        public bool sort { get; set; }
        public string variable { get; set; }
    }

    public class EditRow
    {
        public List<EditInstance> Edits { get; set; }

    }

    public class EditInstance
    {
        public string Name { get; set; }
        public bool filter { get; set; }
    }

   public static class TableColumnLength
    {

    }
}
```

```
using DataAccesLib.Models;
using System.Collections.Generic;
using System.Linq;

namespace DataAccesLib
{
    public class DataFormatter
    {
        public int nColumns;


        public List<List<string>> DataFormat(string TableType, List<RecipeModels>
recipes)
        {
            List<List<string>> list = new List<List<string>>();
            if (TableType == "PIX318") { return Pix318Model(recipes); }
            else if (TableType == "PIX318Analyse") { return
Pix318ModelAnalyse(recipes); }
            else return list;
        }


        public List<List<string>> Pix318Model(List<RecipeModels> recipes)
        {
            List<List<string>> FormattedList = new List<List<string>>();



            foreach (RecipeModels recipe in recipes)
            {
                List<string> TempList = new List<string>();

                TempList.Add(recipe.BatchNr.ToString());
                TempList.Add(recipe.Dato);
                TempList.Add(recipe.SAP.ToString());
                TempList.Add(recipe.ID);
                TempList.Add(recipe.Reaktor.ToString());
                TempList.Add(recipe.SatsVolum.ToString());
                TempList.Add(recipe.ForvFe.ToString());
                TempList.Add(recipe.OnsketFe.ToString());
                TempList.Add(recipe.OnsketSyre.ToString());
                TempList.Add(recipe.OnsketFe2.ToString());
                TempList.Add(recipe.HCLType.ToString());
                TempList.Add(recipe.ForvDamp.ToString());
                TempList.Add(recipe.VannOverordnet.ToString());
                TempList.Add(recipe.VarmtVann.ToString());
                TempList.Add(recipe.SpillVann.ToString());
                TempList.Add(recipe.ScrubberVaeske.ToString());
                TempList.Add(recipe.HCL.ToString());
                TempList.Add(recipe.Jernsulfat.ToString());
                TempList.Add(recipe.Temp.ToString());
                TempList.Add(recipe.Modningstid.ToString());
                TempList.Add(recipe.DampVentil.ToString());
                TempList.Add(recipe.Etterspyling.ToString());
                TempList.Add(recipe.O2Trykk.ToString());
                TempList.Add(recipe.O2Reaksjonstid.ToString());
                TempList.Add(recipe.DeltaTemp.ToString());
                //Are Present in Analysis
                //TempList.Add(recipe.AnalysertFe3.ToString());
                //TempList.Add(recipe.AnalysertFeTot.ToString());
                TempList.Add(recipe.VannSluttJustering.ToString());
                TempList.Add(recipe.VirkeligMVann.ToString());
                TempList.Add(recipe.TotTilLager.ToString());
                //Not part of columns, used in editing and filtering
                TempList.Add(recipe.filter.ToString());
                TempList.Add(recipe.edit.ToString());

                FormattedList.Add(TempList);
                nColumns = TempList.Count() - 2;

            }

            return FormattedList;
        }

        public List<List<string>> Pix318ModelAnalyse(List<RecipeModels> recipes)
        {
            List<List<string>> FormattedList = new List<List<string>>();

            foreach (RecipeModels recipe in recipes)
            {
                List<string> TempList = new List<string>();

                TempList.Add(recipe.BatchNr.ToString());
                TempList.Add(recipe.Dato);
                TempList.Add(recipe.ID);
                TempList.Add(recipe.TotalFe.ToString());
                TempList.Add(recipe.Egenvekt.ToString());
                TempList.Add(recipe.Verdi2Fe.ToString());
                TempList.Add(recipe.ManuellVerdi2Fe.ToString());
                TempList.Add(recipe.Verdi3EtterManuell2Fe.ToString());
                TempList.Add(recipe.Verdi3Fe.ToString());
                TempList.Add(recipe.FriSyre.ToString());

                //Not part of columns, used in editing and filtering
                TempList.Add(recipe.filter.ToString());
                TempList.Add(recipe.edit.ToString());

                FormattedList.Add(TempList);
                nColumns = TempList.Count() - 2;

            }

            return FormattedList;
        }

        public List<List<string>> Alarms(List<AlarmData> alarms)
        {
            List<List<string>> FormattedList = new List<List<string>>();

            foreach (AlarmData entry in alarms)
            {
                List<string> TempList = new List<string>();


                TempList.Add(entry.AlarmId.ToString());
                TempList.Add(entry.ActivationTimeStamp.ToString());
                if(entry.ActivationTimeStamp == entry.ActivationTimeStamp)
                {
                    TempList.Add("--");
                }
                else
                {
                    TempList.Add(entry.AcknowledgeTimeStamp.ToString());
                }



                TempList.Add(entry.AcknowledgeOperator.ToString());
                TempList.Add(entry.Silence.ToString());
                //TempList.Add(entry.SilenceDuration.ToString());
                TempList.Add(entry.SeverityName.ToString());
                TempList.Add(entry.Acknowledge.ToString());

                FormattedList.Add(TempList);
                nColumns = TempList.Count();

            }

            return FormattedList;
        }

        public List<List<string>> Tags(List<TagLog> Tags)
        {
            List<List<string>> FormattedList = new List<List<string>>();

            foreach (TagLog entry in Tags)
            {
                List<string> TempList = new List<string>();


                TempList.Add(entry.TimeStamp.ToString());
                TempList.Add(entry.TagId.ToString());
                TempList.Add(entry.Value.ToString());
                //TempList.Add(entry.Quality.ToString());
                //TempList.Add(entry.Status.ToString());

                FormattedList.Add(TempList);
                nColumns = TempList.Count();

            }

            return FormattedList;
        }

        public List<List<string>> AlarmConfig(List<AlarmConf> Tags)
        {
            List<List<string>> FormattedList = new List<List<string>>();

            foreach (AlarmConf entry in Tags)
            {
                List<string> TempList = new List<string>();


                TempList.Add(entry.AlarmId.ToString());

                TempList.Add(entry.TagId.ToString());
                TempList.Add(entry.AlarmType.ToString());
                TempList.Add(entry.AlarmDescription.ToString());
                TempList.Add(entry.SeverityName.ToString());
                TempList.Add(entry.Disable.ToString());



                FormattedList.Add(TempList);
                nColumns = TempList.Count();
            }

            return FormattedList;
        }


        public List<List<string>> DataFormat(List<SqlData> Table)
        {
            List<List<string>> FormattedList = new List<List<string>>();

            foreach (SqlData entry in Table)
            {
                List<string> TempList = new List<string>();


                TempList.Add(entry.Name1.ToString());
                TempList.Add(entry.Name2.ToString());
                TempList.Add(entry.Name3.ToString());
                TempList.Add(entry.Name4.ToString());
                TempList.Add(entry.Name5.ToString());
                TempList.Add(entry.Name6.ToString());
                TempList.Add(entry.Name7.ToString());
                TempList.Add(entry.Name8.ToString());
                TempList.Add(entry.Name9.ToString());
                TempList.Add(entry.Name10.ToString());


                FormattedList.Add(TempList);
                nColumns = TempList.Count() - 2;

            }

            return FormattedList;
        }
    }
}
```

```
@using DataAccesLib.Models
@using DataAccesLib
@using BlazorDateRangePicker;
@using System.Diagnostics
@using System.Threading



@inject IRecipeData _db


@if (TableList is null)

{
    <p><em>Loading....</em></p>
}
else
{

  <div>
      <table class="table table-striped">
        <thead>
            <tr>
```

```
                @foreach(var Var in filter)
                    {
                       @if(!Var.sort)
                       {<th>@Var.variable</th>}

                    }


            </tr>
        </thead>
        <tbody>

          @for(int n = 0; n < nRows; n++)
                {

                    @if (true)
                    {
                        <tr @key="RecipeRead.TableList[n]">
                            @for (int i = 0; i < format.nColumns; i++)
                            {
                                @*@if (!filter[n].sort)
                                {

                                    <td>@RecipeRead.TableList[n][i]</td>

                                }*@
                                <td>@RecipeRead.TableList[n][i]</td>
                            }
                            @if (TableType == "Alarms")
                            {


                                @if (!AlarmData[n].Acknowledge)
                                {
                                    <th>
                                        <button type="button" class="btn btn-link">
                                            <span class="oi oi-task" aria-
hidden="true" @onclick="() => AlarmAck(n)"></span>
                                        </button>
                                    </th>
                                }
                            }


                        </tr>

                    }
                    else
                    {
                        @foreach(var Variable in TableEdit.Edits)
                            {

                            @if(!Variable.filter)
                            {

                                <td>
                                <input type="text" class="form-control" @bind-
value="@Variable.Name"/>

                                </td>

                            }
```

```
                             }


                     }
                 }



        </tbody>
    </table>
  </div>

}

@code {
    public List<string> RecipeEdit;
    private List<AlarmData> Alarms;
    private int nRows = 0;
    private bool tableLoaded;
    private System.Threading.Timer timer1;

    public List<List<string>> TableList = new List<List<string>>();
    public List<AlarmData> AlarmData = new List<AlarmData>();

    QueriesV1 query = new QueriesV1();
    EditRow TableEdit = new EditRow { Edits = new List<EditInstance>() };


    [Parameter]
    public filtering[] filter {get;set;}
    [Parameter]
    public string TableType { get; set; }




    public DataFormatter format = new DataFormatter();


    protected override async Task OnInitializedAsync()
    {
        Debug.WriteLine(TableType);
        base.OnInitialized();

        query.SetTableType(TableType);
        AlarmData.Clear();



        while(true)
        {
            TableUpdate();
            await Task.Delay(1000);

        }
    }



    private void EnableEditing(bool flag, List<string> batch)
    {

        if(flag)
        {

            batch[format.nColumns+1] = "True";
            RecipeEdit = batch;
            flag = !flag;

            List<EditInstance> _EditInstance = new List<EditInstance>();

            for(int n = 0; n < format.nColumns; n++)
            {
                EditInstance _Entry = new EditInstance(){Name = batch[n], filter = 
filter[n].sort};

                TableEdit.Edits.Add(_Entry);
            }

        }
        else
        {
            batch[format.nColumns + 1] = "False";
            TableEdit.Edits.Clear();
        }

        StateHasChanged();
    }









    public async void TableUpdate()
    {





        if (TableType == "Alarms")
        {

            //List<AlarmData> AlarmData = new List<AlarmData>();

            AlarmData = await _db.GetAlarms(query.TableUpdate());

            TableList = format.Alarms(AlarmData);

            RecipeRead.TableList = TableList;

            Debug.WriteLine(RecipeRead.TableList.Count + "Debug");
            nRows = RecipeRead.TableList.Count;
            List<bool>  AlarmAck = new List<bool>();
            for(int n = 0; n < RecipeRead.TableList.Count()+1; n++)
            {
                AlarmAck.Add(true);
                //Debug.WriteLine(AlarmAck.Count);
            }
            RecipeRead.AlarmAck = AlarmAck;



        }
        else if (TableType == "Tags")
        {


            List<TagLog> TagLog = new List<TagLog>();

            TagLog = await _db.GetTags(query.TableUpdate());

            TableList = format.Tags(TagLog);

            RecipeRead.TableList = TableList;
            nRows = RecipeRead.TableList.Count - 1;
            //Debug.WriteLine("EnteredTagPage" + RecipeRead.TableList[0][0]);
        }
        else if (TableType == "AlarmConfig")
        {
            Debug.WriteLine("DebugConfig");
            List<AlarmConf> AlarmConfig = new List<AlarmConf>();

            AlarmConfig = await _db.GetAlarmConfigs(query.TableUpdate());

            TableList = format.AlarmConfig(AlarmConfig);

            RecipeRead.TableList = TableList;
            Debug.WriteLine(TableList[0][1] + "ConfigTest");
            nRows = RecipeRead.TableList.Count;

        }

        try
        {
            StateHasChanged();
        }
        catch
        {

        }StateHasChanged();
    }



    //public void FilterUpdate()
    //{
    //    for(int n = 0; n < format.nColumns - 1; n++)
    //    {
    //        FilterModel.filter[n] = filter[n].sort;
    //    }
    //}


    public async void AlarmAck(int n)
    {
        RecipeRead.AlarmAck[n] = !RecipeRead.AlarmAck[n];


        string sql = query.AlarmAck(AlarmData[n-1].AlarmId, AlarmData[n-
1].ActivationTimeStamp);
        Debug.WriteLine(n+": ack: " + sql);
        await _db.EditRecipe(sql);

        StateHasChanged();
    }




    }
```

```
@page "/TagData"

@using DataAccesLib
@using DataAccesLib.Models
@using BlazorDateRangePicker;
@using System.Diagnostics

@inject IRecipeData _db
@inject IConfiguration _config


<h1>TagData</h1>


<h5 class ="mx-5">&nbsp;&nbsp;&nbsp; Dato
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp; Antall Rader</h5>



 <TableSortComp rangeQuery="@UpdateDateRange"></TableSortComp>

<Tables filter="@filter" TableType="@TableType"></Tables>

@*<CsvAndResetComponent TableRestart="@UpdateResetTable"></CsvAndResetComponent>

<FilterButtons FilterSort="@filter"
FilterSortUpdate="@UpdateFilter"></FilterButtons>*@

}



@code {

    private int TableLen = Queries.TableLen;

    private int nColumns = 3;
    private string TableType = "Tags";

    private List<RecipeModels> recipes;
    private RecipeModels RecipeEdit = new RecipeModels();

    public filtering[] filter = new filtering[3];
    public DataFormatter format = new DataFormatter();
    public List<List<string>> TableList = new List<List<string>>();
    private string PIX318Table;

    QueriesV1 query = new QueriesV1();



    protected override async Task OnInitializedAsync()
    {
        PIX318Table = _config.GetSection("Tables").GetSection("TagData").Value;
        query.SetTableType(TableType);
        for(int i = 0; i < nColumns; i++)
        {
            filter[i] = new filtering();
            filter[i].variable = Extensions.TagDataExtensions[i];
        }




    }

    public async void TableUpdate()
    {
        List<TagLog> TagData = new List<TagLog>();
            TagData = await _db.GetTags(query.TableUpdate());
            TableList = format.Tags(TagData);
            Debug.WriteLine(TableList[0][0]);
            RecipeRead.TableList = TableList;

        try
        {
            StateHasChanged();
        }
        catch
        {

        }


    }

    private void UpdateFilter(filtering[] _filter)
    {
        filter = _filter;
    }


    private void UpdateDateRange(string sql)
    {
        TableUpdate();
    }

    private void UpdateResetTable(string sql)
    {
        TableUpdate();
    }
}
```

```
using Microsoft.Extensions.Configuration;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dapper;
using System.Data.SqlClient;
//using Microsoft.Extensions.Configuration;

namespace DataAccesLib
{
    public class SQLDataAccess : ISQLDataAccess
    {
        private readonly IConfiguration _config;

        public string ConnectionStringName { get; set; } = "Default";
        public SQLDataAccess(IConfiguration config)
        {
            _config = config;
        }

        public async Task<List<T>> LoadData<T, U>(string sql, U parameters)
        {
            string connectionString =
_config.GetConnectionString(ConnectionStringName);

            using (IDbConnection connection = new SqlConnection(connectionString))
            {
                var data = await connection.QueryAsync<T>(sql, parameters);

                return data.ToList();
            }
        }

        public async Task SaveData<T>(string sql, T parameters)
        {
            string connectionString =
_config.GetConnectionString(ConnectionStringName);

            using (IDbConnection connection = new SqlConnection(connectionString))
            {
                await connection.ExecuteAsync(sql, parameters);
            }
        }

    }
}
```

```
using DataAccesLib.Models;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace DataAccesLib
{
    public interface IRecipeData
    {
        Task<List<RecipeModels>> GetRecipes(string sql);
        Task<List<AlarmData>> GetAlarms(string sql);
        Task<List<TagLog>> GetTags(string sql);
        Task<List<SqlData>> GetData(string sql);
        Task<List<AlarmConf>> GetAlarmConfigs(string sql);
        Task InsertRecipe(RecipeModels recipe);

        Task EditRecipe(string sql);
    }
}
```

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace DataAccesLib
{
    public interface ISQLDataAccess
    {
        string ConnectionStringName { get; set; }

        Task<List<T>> LoadData<T, U>(string sql, U parameters);
        Task SaveData<T>(string sql, T parameters);

    }
}
```

References:

N.-O. Skeie, ”Course IIA2017 – Industrial Information Technology”, in Lecture Notes with sections for system engineering, process……, 2022

*Work for scada tasks cannot be referenced, as it gives away name of student doing the exam.