TARTU UNIVERSITY

FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE

STUDY PROGRAM “INFORMATION TECHNOLOGY SYSTEMS DEVELOPMENT “

Maksim Naprejev

“DEVELOPMENT OF SERVICE THAT EXECUTES TASKS BY SCHEDULE“

Diploma thesis

Supervisors: Nadežda Furs-Nižnikova

Assistant Andre Sääsk

NARVA 2019

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……

Töö autori allkiri ja kuupäev

Non-exclusive license to reproduce thesis

I, Maksim Naprejev (date of birth: 19.08.1987),

1. Herewith grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright,

"Development of service that executes tasks by schedule" supervised by assistant Andre Sääsk and Nadežda Furs-Nižnikova. Please do not publish my diploma thesis until 1.01.2022.

2. I am aware of the fact that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Narva, 21.05.2019

Contents

Terms and abbreviations

Enum - in C#, is a keyword that represents a value type for declaring a set of named constants.

Platform - is a group of technologies that are used as a base upon which other applications, processes or technologies are developed.

Functional requirement - is a declaration of the intended function of a system and its components.

Framework - software library that provides a fundamental structure to support the development of applications for a specific environment.

API - an application programming interface (API) is a set of protocols, routines, functions and/or commands that programmers use to develop software or facilitate interaction between distinct systems.

GUI - a graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances.

Debugging - the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools.

Logging - the process of collecting and storing data over a period of time in order to analyze specific trends or record the data-based events/actions of a system, network.

NuGet - the package manager for .NET.

Introduction

Eesti Energia is an Estonian state company which operates internationally, including the energy market of the Baltic Sea area and the global oil market. The group is engaged in the mining of oil shale, production of electricity, heat and oil, development of the know-how and technologies required for oil shale processing as well as providing customers with digital solutions and products for services related to network service, electricity, heat and gas.

In year 2015, Eesti Energia launched a group development program named Industry 3.5. The main goal of Industry 3.5 is to integrate the processes of the oil shale chain, then implement the information systems that support the business processes and supply the data to the managers. Industry 3.5 is carried out in cooperation between the business and information technology service, oil shale chain companies, energy trading, management accounting, environmental service, development service and legal service.

The problem

Platform ECOS -is an own development of Eesti Energia, which was created to standardize and accelerate the development process of future company applications. Eesti Energia development department received orders for the development of the following applications:

- Enefit Green – monitoring system of sun panels, wind parks, hydro station.
- Oil Factory – monitoring system of the oil factories.
- B & E – power unit monitoring system
- Calorimeters – monitoring system of conveyor calorimeters
- Emissions – monitoring of emissions from power plants.
- LogisticMes - operational report of shale transportation by railcars

The goals of all these projects is to increase the efficiency of power generation through the process management that based on operational data, to digitalize electrical power generation processes and to complete this task they need to retrieve data from the data source and run various tasks based on this data.

The Solution

The solution of the problem is implementation of a common service for the ECOS platform that will perform scheduled tasks to get data from the data source. and run various tasks based on this data.

The aim

The aim is to provide operational data to applications that are based on the ECOS platform.

Tasks

To achieve the aim of this work the author has to complete the following tasks:

- Development of a common service for applications using the ECOS platform
- Creating a system for monitoring the implementation of tasks
- Base task implementation

Contents of this thesis

Section 1 describes technology stack of the ECOS platform

Section 2 describes functional requirements of ECOS applications

Section 3 describes Task Scheduler and its components

Section 4 describes chosen technologies for implementation

Section 5 describes the development process

Section 6 describes data base

Section 7 describes applications for which the common service was created

1 Overview of the ECOS technology stack

The ECOS platform uses the following technologies:

- C# - object-oriented programming language.
- ASP.NET (Active Server Pages) - a developer platform made up of tools, programming languages, and libraries for building many different types of applications. [1]
- MVC (Model-View-Controller) - a design pattern used to decouple user-interface (view), data (model), and application logic (controller). [2]
- Entity Framework - an object-relational mapper that enables .NET developers to work with a database using .NET objects. [3]
- MS SQL - relational database management system.
- JS - (JavaScript) is a prototype-based, multi-paradigm, dynamic language, supporting object-oriented, imperative, and declarative styles. [4]
- jQuery - fast, small, and feature-rich JavaScript library. [5]
- Bootstrap - is an open source toolkit for developing with HTML, CSS, and JS. [6]
- HTML – (Hyper Text Markup Language) defines the meaning and structure of web content. [7]
- CSS - (Cascading Style Sheets) is a declarative language that controls how webpages look in the browser. [8]
- Windows Server - generally capable of providing server-oriented services, such as the ability to host a website, user management, resource management across users and applications, messaging, security and authorization and many other server-focused services. [9]

1.1 Entity Framework

Entity Framework is an open-source object-relational mapping framework for .NET applications supported by Microsoft. It enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with data and can create and maintain data-oriented applications with less code compared with traditional applications. [10]

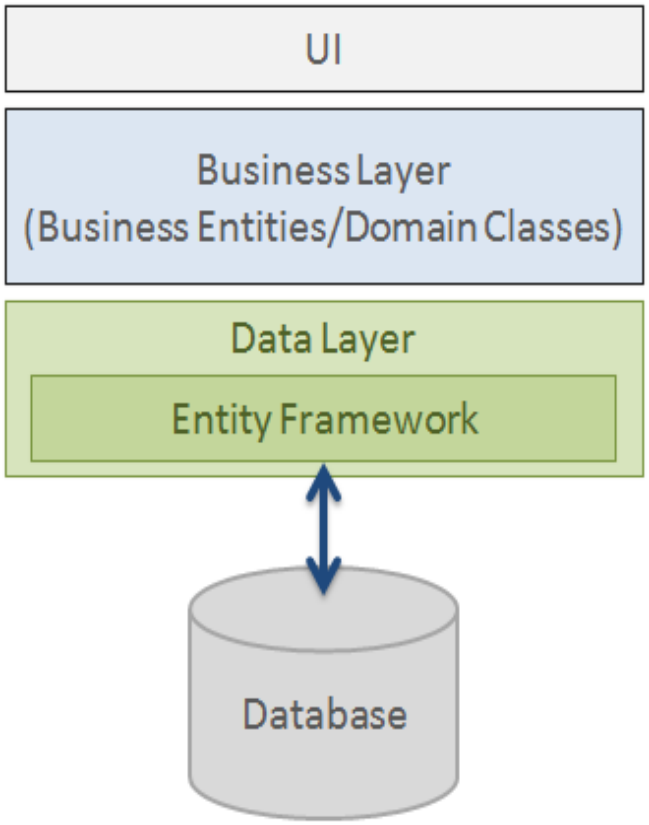

Figure 3. *Entity Framework (Source: enetityframework.net).*

1.2 .NET

.Net is a free, open source developer platform for building many different types of applications. With .NET, you can use multiple languages, editors, and libraries to build for web, mobile, desktop, gaming, and IoT. .NET Standard is a base set of APIs that are common to all .NET implementations. Each implementation can also expose additional APIs that are specific to the operating systems it runs on. For example, .NET Framework is a Windows-only .NET implementation that includes APIs for accessing the Windows Registry. To extend functionality, Microsoft and others maintain a healthy package ecosystem built on .NET Standard. NuGet is a package manager built specifically for .NET that contains over 90,000 packages. It increases developer productivity by reducing the quantity of codes used in large web applications. [11]

1.2.1 Pros of .NET framework

- This platform is multi-language compatible.
- Its base class library supports enables to achieve varied functions like file reading, graphic rendering, etc.

- It possesses all the resources to equip your websites with all the functionalities to manage it smoothly.
- Entrusting your web development project on a dot net development agency will make sure that a highly-functional website is built with exceptional user experience. Your interactive website would attract more visitors and they are more likely to convert.

1.2.2 Cons of .NET framework

- This framework may not be pre-installed in the older versions of Windows. So, you must check it first and if it not there then you should follow the guidelines provided in the user manual.
- None of the versions of Windows have pre-installed versions of frameworks, especially the newer ones.
- In certain cases, the consumption of time may be more because of garbage collection occurring at regular interval in order to reclaim memory.
- Applications that run in managed environments often require more system resources than the ones accessing machine resources directly.
- As reverse engineering of codes is easier, security is rather less. However, many advanced techniques have been developed in order to protect the privacy to the best possible extent. [12]

2 Applications functional requirements

2.1 Enefit Green

1) Based on the minute values obtained from the data source, system should display information of wind farms, sun panels, hydro station in real time:

- generated power
- wind speed
- direction of the wind
- status (working, service, broken)
- efficiency

The frequency of updating information should be one minute.

2) Sending SMS in case of a wind turbine breakdown and coming into operation after breakdown.

2.2 B & E

Based on the minute values obtained from the data source system should calculate efficiency of power unit blocks. The frequency of updating information should be one minute.

2.3 Oil Factory

1) Based on the minute values obtained from the data source, system should display information of oil factories in real time:

- power
- emissions
- oil
- status (working, service, broken)

The frequency of updating information should be one minute.

2) Based on the minute values obtained from the data source system should send SMS about the breakdowns of oil factories and coming into operation after breakdown.

2.4 Emissions

Based on the minute values obtained from the data source system should display emissions data of power unit blocks. The frequency of updating information should be 30 seconds.

2.5 Calorimeters

Based on the minute values obtained from the data source system should display information from calorimeters:

- weight
- wetness
- content of ash
- calorific value.

The frequency of updating information should be one minute.

2.6 LogisticMes

System should import waybills of railcars every day at 17:00.

2.7 Summary of applications functional requirements

Based on the functional requirements of these applications, there is a need to implement a service that will perform tasks on a schedule such as exporting data from sources, sending alerts etc.

3 Task Scheduler and its components

3.1 Task Scheduler

The Task Scheduler enables automatically perform routine tasks on a chosen computer. The Task Scheduler does this by monitoring whatever criteria you choose to initiate the tasks and then executing the tasks when the criteria is met.
Scheduler will perform:

- when a specific system event occurs,
- at a specific time,
- at a specific time on a daily schedule,
- at a specific time on a weekly schedule,
- at a specific time on a monthly schedule,
- at a specific time on a monthly day-of-week schedule,
- when the computer enters an idle state,
- when the task is registered,
- when the system is booted,
- when a user logs on,
- when a Terminal Server session changes state. [13]

3.2 Task

A task is the scheduled work that the Task Scheduler service performs. A task is composed of different components, but a task must contain a trigger that the Task Scheduler uses to start the task and an action that describes what work the Task Scheduler will perform. [14]

3.3 Trigger

A set of criteria that, when met, will cause a task to be executed. Task Scheduler provides time-based and event-based triggers that can specify task start times, repetition criteria, and other parameters. [15]

4 Chosen technologies

4.1 Choose of scheduler

Since ECOS uses the .NET framework then for common Windows Service implementation the choice was from two popular scheduling frameworks Hangfire и Quartz.net, because use of the default built in .Net library scheduler (System.Timers.Timer) was rejected at the initial stage because of two main disadvantages:

- Timers have inflexible scheduling (they can set only the start time and the retry interval, nothing is based on dates, time of day, etc.).
- Timers do not have real control schemes - you need to write your own mechanism to be able to memorize, organize and retrieve your tasks by name. [16]

4.1.1 Quartz.net

Quartz.net is a job scheduling system that can be integrated with or used alongside virtually any other software system. Quartz is distributed as a small dynamically linked library that contains all the core Quartz functionality. The main interface to this functionality is the Scheduler interface. It provides simple operations such as scheduling or unscheduling jobs, starting, stopping, pausing the scheduler. The main Quartz process can be started and ran within your own application, or a stand-alone application (with a remote interface). [17]

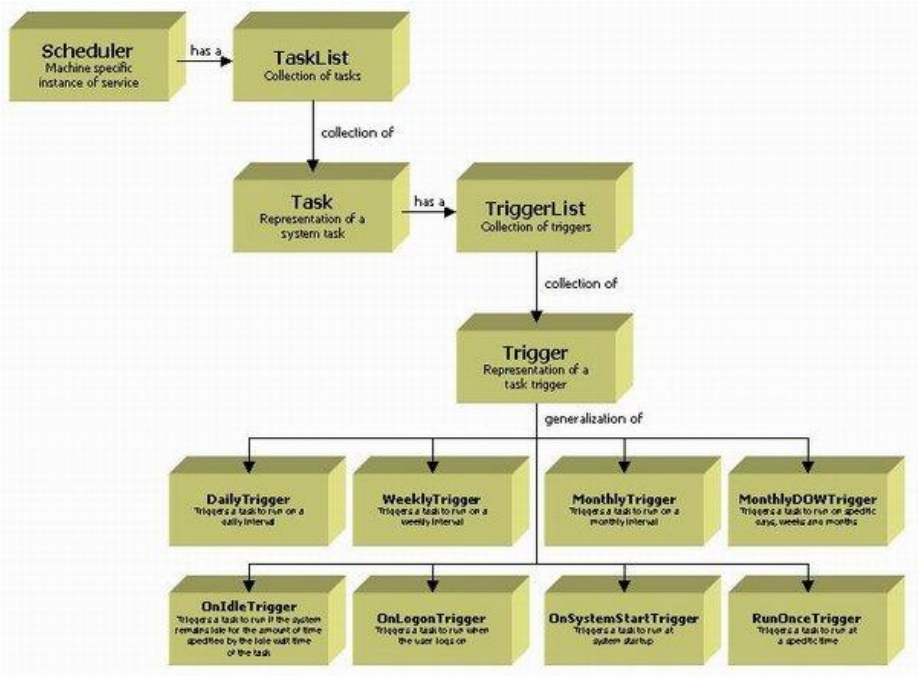

Figure 1. *Scheme of work of the Quartz.Net (Source: codeproject.com).*

4.1.2 Hangfire

Hangfire allows you to kick off method calls outside of the request processing pipeline in a very easy, but reliable way. These method invocations are performed in a background thread and called background jobs. You can create any kind of background jobs using Hangfire: fire-and-forget (to offload the method invocation), delayed (to perform the call after some time) and recurring (to perform methods hourly, daily and so on). Hangfire does not require you to create special classes. Background jobs are based on regular static or instance methods invocation. [18]

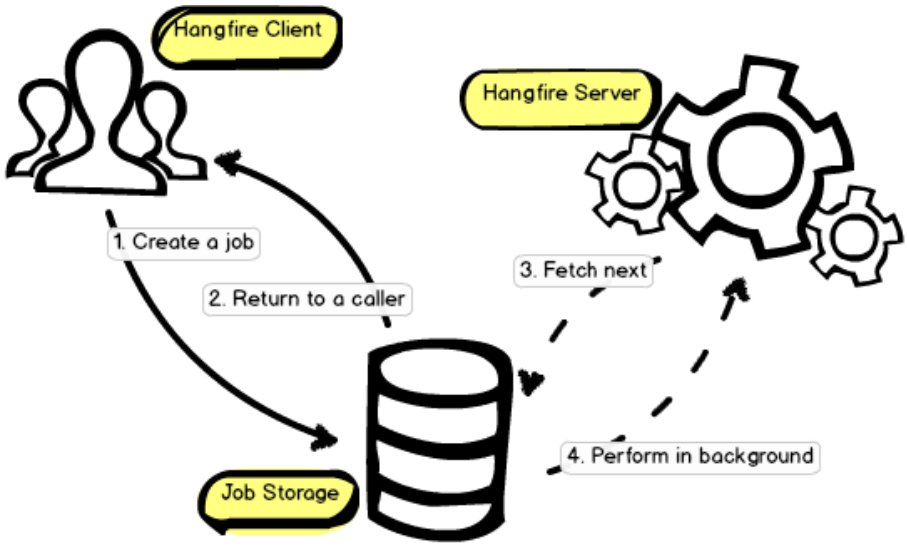

Figure 2. *The main processes in Hangfire (Source: hangfire.io).*

4.1.3 Hangfire and Quartz.Net comparison

***Table 2**. Hangfire and Quartz.Net comparison table*

Feature	Hangfire	Quartz.NET
Number of clients and servers	Unlimited	Unlimeted
Hosting	Windows Server, Azure	Windows Server, Azure
Delayed jobs	Yes	Yes
Minimum interval of jobs	1 minute	1 millisecond
Price	Paid subscriptions start from 500 $ per year	Free

Data: created by the author

According to the results of comparison between the two frameworks choice was made in favor of Quartz.NET:

- Quartz.NET is free,
- Recurring job in Quartz.NET is more flexible (minimum interval is 1 millisecond).

4.2 Log4Net

The logger concept lies at the heart of log4net's configuration. Loggers are organized into a hierarchy and give the programmer run-time control on which logging statements are printed or not. A log statement is routed through to the appender depending on its level and its logger.

Log4Net is a tool to help the programmer output log statements to a variety of output targets. In case of problems with an application, it is helpful to enable logging so that the problem can be located. With log4net it is possible to enable logging at runtime without modifying the application binary. The log4net package is designed so that log

statements can remain in shipped code without incurring a high-performance cost. It follows that the speed of logging (or rather not logging) is crucial. At the same time, log output can be so voluminous that it quickly becomes overwhelming. One of the distinctive features of log4net is the notion of hierarchical loggers. Using these loggers, it is possible to selectively control which log statements are output at arbitrary granularity. log4net is designed with two distinct goals in mind: speed and flexibility. Log4Net features:

- Log4net is optimized for speed.
- Log4net is based on a named logger hierarchy.
- Log4net is fail-stop but not reliable.
- Log4net is thread-safe.
- Log4net is not restricted to a predefined set of facilities.
- Logging behavior can be set at runtime using a configuration file. log4net is designed to handle exceptions from the start.
- Log4net can direct its output to many sinks including: a file, the console or even e-mail.
- Log4net categorizes logging into levels: DEBUG, INFO, WARN, ERROR and FATAL.
- The format of the log output can be easily changed by implementing a new layout class.
- The target of the log output as well as the writing strategy can be altered by writing a new appender class.
- Log4net supports multiple output appenders per logger. [19]

5 Development

5.1 Enums of trigger types and names

Created 2 enums TriggerType and TaskNames for task scheduler. TriggerType enum - contains a list of trigger types – cron and interval

- With cron trigger allows to specify firing-schedules such as “every Friday at noon”, or “every weekday and 9:30”, or even “every 5 minutes between 9:00 and 10:00 on every Monday, Wednesday and Friday”. This type of trigger will be used for those tasks that need to be performed at a certain time, for example, import of locomotive way tickets should be done every day at 16:30.
- Interval – the trigger is designed to run after a certain time. For example, every 60 seconds application Enefit Green needs to export data (the power of the generated electricity, wind speed and direction) from the data source.
  Enum TaskName contains a list of tasks names.

```
namespace TaskScheduler
{
    public enum TriggerType
    {
        Interval = 1,

        Cron = 2
    }


    public enum TaskNames
    {
        [TaskType(typeof(ImportFromAroundTask))]
        ImportFromAround = 1,

        [TaskType(typeof(AxaptaOutcomeTask))]
        AxaptaOutcome = 2,

        [TaskType(typeof(BeltScalesDataImportTask))]
        BeltScalesData = 3
    }
}
```

Figure 4. *TaskNames and TriggerType enums (Source: author).*

5.2 Running the service

If the project is compiled in release mode, it must be installed as the Windows Service, if in debug mode, then as a console application, which creates convenience for debugging. Implementation of the task manager is a Singleton Pattern.

```
1  using log4net;
2  using System;
3  using System.Diagnostics;
4  using System.ServiceProcess;
5
6  namespace TaskScheduler
7  {
8      public static class Program
9      {
10         public static readonly ILog Logger = LogManager.GetLogger(
11
12         static void Main()
13         {
14 #if (!DEBUG)
15
16             var ServicesToRun = new ServiceBase[]
17             {
18                 new TaskSchedulerService()
19             };
20             ServiceBase.Run(ServicesToRun);
21
22 #else
23
24             TaskManager.Instance.Start();
25
26 #endif
27         }
28     }
29 }
30
```

Figure 5. *Class Program (Source: author).*

5.3 Singleton Pattern

- Ensures a class has only one instance and provides a global point of access to it.
- A singleton is a class that only allows a single instance of itself to be created, and usually gives simple access to that instance.
- Most commonly, singletons don't allow any parameters to be specified when creating the instance, since a second request of an instance with a different parameter could be problematic! (If the same instance should be accessed for all requests with the same parameter then the factory pattern is more appropriate.)

There are various ways to implement the Singleton Pattern in C#. The following are the common characteristics of a Singleton Pattern.

- A single constructor, that is private and parameter less.
- The class is sealed.
- A static variable that holds a reference to the single created instance, if any.
- A public static means of getting the reference to the single created instance, creating one if necessary.

5.4 Advantages of Singleton Pattern

The advantages of a Singleton Pattern are:

- Singleton pattern can be implemented interfaces.
- It can be also inheriting from other classes.
- It can be lazy loaded.
- It has Static Initialization.
- It can be extended into a factory pattern.
- It helps to hide dependencies.
- It provides a single point of access to a particular instance, so it is easy to maintain

5.5 Disadvantages of Singleton Pattern

The disadvantages of a Singleton Pattern are:

- Unit testing is more difficult (because it introduces a global state into an application).
- This pattern reduces the potential for parallelism within a program, because to access the singleton in a multi-threaded system, an object must be serialized (by locking). [20]

5.6 Task scheduler implementation

The Task Scheduler was implemented with using of the Quartz.net library.

Quartz.net configuration:

- quartz.serializer.type, binary - is use when you want to pass objects between client and server.
- quartz.threadPool.threadCount – counting how many threads in the thread pool, which means that a maximum of counted jobs can be run simultaneously.

```
private void CreateScheduler()
{
    try
    {
        var props = new NameValueCollection
        {
            { "quartz.serializer.type", "binary" },
            { "quartz.threadPool.threadCount", ServiceThreadCount }
        };
        var factory = new StdSchedulerFactory(props);
        Scheduler = factory.GetScheduler().Result;
    }
    catch (Exception ex)
    {
        // log exception and throw it farther
        var exception = new Exception("Unable to create scheduler!", ex);
        Program.Logger.Fatal(exception.Message, exception.InnerException);
        throw exception;
    }
}
```

Figure 6. *Implementation of the Task Scheduler (Source: author).*

5.7 Tasks loading method

LoadTask method gets tasks setups from the database table:

- task names
- total number of tasks
- number of enabled tasks.

In cases if the method cannot load tasks, system will try to load tasks again until it succeeds.

```
private void LoadTasks()
{
    var uncompleted = true;
    while (uncompleted)
    {
        try
        {
            Tasks = ServiceTaskSetup.GetAll().AsReadOnly();
            if (Tasks == null || Tasks.Count == 0)
                throw new Exception("No tasks loaded from database!");

            uncompleted = false;

            Program.Logger.Info("Tasks loaded: " + string.Join(", ", Tasks.Select(x => x.TaskName)));
            Program.Logger.Info("Total tasks: " + Tasks.Count);
            Program.Logger.Info("Enabled tasks: " + Tasks.Count(x => x.Enabled));
        }
        catch (Exception ex)
        {
            // log exception (instead of throwing it farther - retry after delay)
            var exception = new Exception("Unable to load tasks!", ex);
            Program.Logger.Error(exception.Message, exception.InnerException);
            Task.Delay(TimeSpan.FromSeconds(Instance.ServiceDutyCheckTime));
        }
    }
}
```

Figure 7. *Implementation of the Task Loading method (Source: author).*

5.8 Service Task Setup model

In the ServiceTaskSetup class declared properties: name, type and values getting and setting. For properties TaskNames and TriggerType used NotMappedAttribute. NotMappedAttribute denotes that a property or class should be excluded from database mapping. [19]

```
public class ServiceTaskSetup
{
    #region Fields

    public int ID { get; set; }

    [StringLength(100)]
    public string TaskName { get; set; }

    public bool Enabled { get; set; }

    [StringLength(50)]
    public string TriggerName { get; set; }

    [StringLength(100)]
    public string StartExpression { get; set; }

    public int? IntervalInSeconds { get; set; }

    public string Description { get; set; }

    [NotMapped]
    public TaskNames TaskType { get; set; }

    [NotMapped]
    public TriggerType Trigger { get; set; }
```

Figure 8. *Implementation of the ServiceTaskSetup class (Source: author).*

5.9 Base task implementation

An abstract attribute DisallowConcurrentExecution is used which is responsible for ensuring that an already running task is not restarted.

```
namespace TaskScheduler
{
    [DisallowConcurrentExecution]
    public abstract class BaseTask : IJob
    {
        public abstract TaskNames TaskName { get; }

        public bool IsEnabled => ServiceTaskSetup.IsEnabled(TaskName);

        public async Task Execute(IJobExecutionContext context)
        {
            try
            {
                if (!IsEnabled)
                    return;

                ServiceTaskLog.Insert(TaskManager.Instance.ServiceName, TaskName.ToString(), SchedulerTaskStates.Started);

                ExecuteTask();

                ServiceTaskLog.Insert(TaskManager.Instance.ServiceName, TaskName.ToString(), SchedulerTaskStates.Completed);
            }
            catch (Exception ex)
            {
                Log(ex);
            }
        }
```

Figure 9. *Implementation of the ServiceTaskSetup class part 1 (Source: author).*

The method ExecuteTask checks whether the tasks are active or not.

```
        public abstract void ExecuteTask();

        public void Log(SchedulerTaskStates state, string comment, string extField = null)
        {
            ServiceTaskLog.Insert(TaskManager.Instance.ServiceName, TaskName.ToString(), state, comment, extField);
        }
        public void Log(Exception ex)
        {
            var comments = $"{ex.Message} {ex.StackTrace}";
            ServiceTaskLog.Insert(TaskManager.Instance.ServiceName, TaskName.ToString(), SchedulerTaskStates.Failed, comments);
        }

        public List<ServiceTaskLog> GetLogs(bool sortDescending = true, int skip = 0, int take = 1, Expression<Func<ServiceTaskLog, bool>> predicate = null)
        {
            return ServiceTaskLog.GetList(TaskName.ToString(), sortDescending, skip, take, predicate);
        }
    }
}
```

Figure 10. *Implementation of the ServiceTaskSetup class part 2 (Source: author)*

5.10 Wind generators getting statuses and SMS sending task

In the implementation of the task for the Enefit Green application to obtain the status of wind generator from the data source and sending SMS in case of breakdown methods and settings of the BaseTask are inherited.

```
namespace TaskScheduler.Tasks.EnergiaTootmine
{
    public enum WorkType
    {
        Stopped = 0,
        Work    = 1,
        NoData  = 2
    };

    public class ETPowerUnitStatusAndSMSSendingTask : BaseTask
    {
        public const string SMS_WAS_SENT = "[Event: {0}:{1}] SMS was sent successfully.";
        public const string SMS_NOT_NEEDED = "[Event: {0}:{1}] SMS not needed.";
        public const int MAX_SQL_REQUEST_COUNT = 3;

        public override TaskNames TaskName => TaskNames.ETPowerUnitStatusAndSMSSendingTask;


        public List<JsonConf> JsonConfs { get; set; }
        public string LogicName { get; set; }

        public DateTime dtRequestFrom { get; set; }
        public DateTime dtRequestTo { get; set; }

        public static void TestExec()
        {
            ETPowerUnitStatusAndSMSSendingTask test = new ETPowerUnitStatusAndSMSSendingTask();
            test.ExecuteTask();
        }
```

Figure 11. *Wind generator status getting and SMS sending task (Source: author)*

6 Database level implementation

At the database 3 tables are implemented:

- ServiceTaskSetup
- ServiceTasksLogs
- ServiceDutyLogs

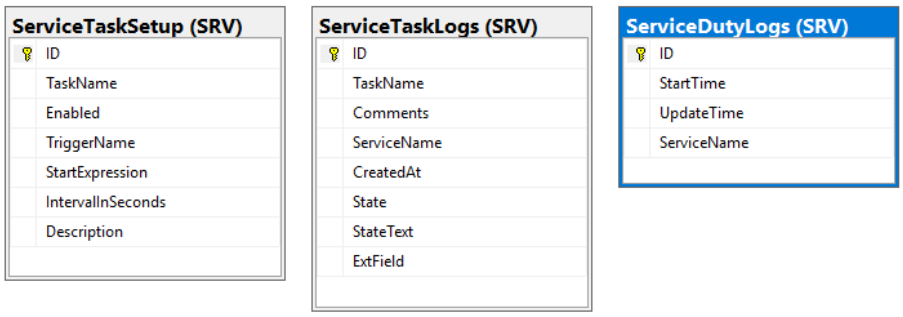

Figure 12. *Tables ServiceTaskSetup, ServiceTaskLogs, ServiceDutyLogs (Source: author).*

6.1 Table ServiceTaskSetup

Table ServiceTaskSetup stores tasks and their settings:

- ID – task id
- TaskName – task name
- TriggerName – trigger name (cron or interval)
- StartExpression – executing expression for cron trigger
- IntervalInSecond -interval of execution in seconds for interval rigger
- Description – optional field for task description

6.2 Table ServiceTaskLogs

Table ServiceTaskLogs stores logs of startup, execution, and errors of tasks

- ID – id of log

- TaskName – task name
- Comments - used for error logging,
- ServiceName – name of server
- CreatedAt – log datetime
- State – id of state
- StateText – state name (started, completed, info, error)
- ExtField – optional field

ID	TaskName	Comments	ServiceName	CreatedAt	State	StateText	ExtField
3644630	PowerUnitSetStatusAndSMSSending	NULL	Serv2	2019-05-22 09:12:11.4025896	5	Completed	NULL
3644629	BjTagRecalculate	NULL	Serv2	2019-05-22 09:12:08.2150471	5	Completed	NULL
3644628	PowerUnitSetStatusAndSMSSending	NULL	Serv2	2019-05-22 09:12:06.9337774	1	Started	NULL
3644627	AnalysisDataFilling	NULL	Serv2	2019-05-22 09:11:56.5273875	5	Completed	NULL
3644626	ETPowerUnitStatusAndSMSSendingTask	NULL	Serv2	2019-05-22 09:11:54.7305613	5	Completed	NULL
3644625	ETPowerUnitStatusAndSMSSendingTask	[Event: Block11:Work] SMS not needed.	Serv2	2019-05-22 09:11:54.6836212	2	Info	ETBlock11
3644624	ETPowerUnitStatusAndSMSSendingTask	[Event: Block8:Work] SMS not needed.	Serv2	2019-05-22 09:11:54.1367351	2	Info	ETBlock8
3644623	BjTagRecalculate	NULL	Serv2	2019-05-22 09:11:53.5429761	1	Started	NULL

Figure 13. *ServiceTasksLogs table (Source: author).*

6.3 ServiceDutyLogs

Table ServiceDutyLogs keeps a list of services and logs of which of them is working now:

- ID – service id
- StartTime – date and time when service starts.
- UpdateTime – every minute the system checks if the service is working now and if the service is in operation, then the last check date and time is recorded
- ServiceName – service name

ID	StartTime	UpdateTime	ServiceName
21	2019-04-22 13:12:15.0000000	2019-05-22 09:04:08.9370000	Serv2
20	2019-04-17 12:51:07.0000000	2019-04-22 13:11:14.6600000	Serv1
19	2019-04-13 05:17:22.0000000	2019-04-17 12:50:06.8200000	Serv2
18	2019-03-20 12:59:36.0000000	2019-04-13 05:16:21.9930000	Serv1

Figure 14. *ServiceDutyLogs table (Source: author).*

7 Examples of applications

7.1 Enefit Green application

Based on the received minute data from the source using the created service, task manager and tasks, the Enefit Green application displays information on the wind farms to the user.

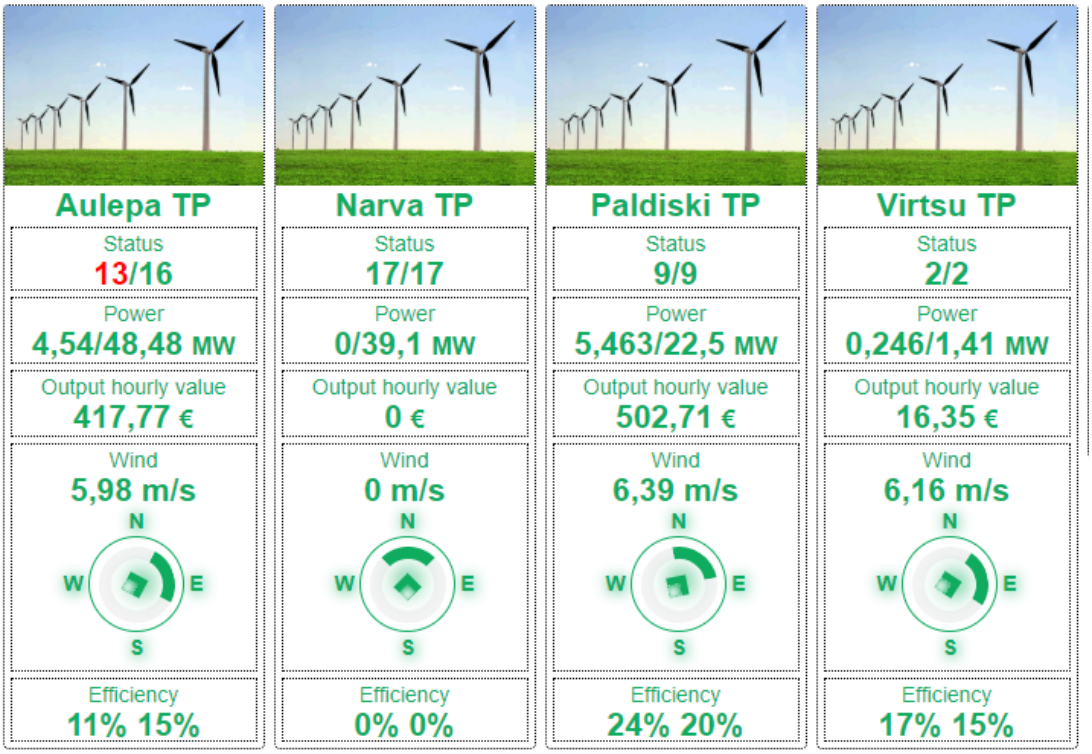

Figure 15. *Enefit Green user interface (Source: author).*

7.2 B&E

Based on the received minute data from the source using the created service, tasks and task manager the B&E application processes the data, calculates the parameters that participate in calculating the efficiency of the power unit and displays information to the user.

Figure 16. *B&E the calculated value of the efficiency of the unit (Source: author).*

Parameter	Equipment	UM	Type	Actual
T primary steam	6KA	C	Automat	508
P primary steam	6KA	Mpa	Automat	12.48
HP steam flow - t/h	6KA	t/h	Automat	306.5
T Hot reheat	6KA	C	Calculate	517
P Hot reheat	6KA	Mpa	Automat	1.94
T primary steam	6KB	C	Automat	505
P primary steam	6KB	Mpa	Automat	12.5
HP steam flow - t/h	6KB	t/h	Automat	272.5
T Hot reheat	6KB	C	Calculate	498
P Hot reheat	6KB	Mpa	Automat	1.98
T feed water	6	C	Automat	229

Figure 17. *B&E parameters that involved in the calculation of efficiency (Source: author).*

7.3 Oil Factory

Based on the received minute data from the data source using the created service, tasks and task manager the Oil Factory application send SMS to the uses about starts and stops.

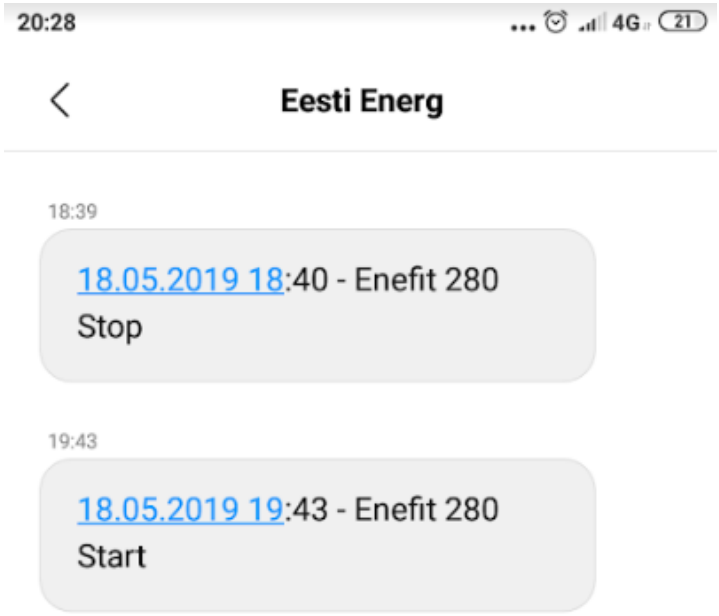

Figure 18. *SMS´s of starts and stops of the Enefit 280 oil factory (Source: author).*

Conclusion

The ECOS platform and related applications received a tool that provides real-time data, which helps to make decisions based on real-time information, which is the goal of not only my work but also the Eesti Energia Industry 3.5 program. As a result, the author may consider that the tasks of this work to be accomplished. Features of the common service:

- Tasks runs on schedule - applications receive operational data.
- Created task logging system - tracks the start time of the task, the time when it was executed and errors in cases of their occurrence.
- Development of new tasks for the ECOS platform will be faster, because there is no need to create new services – one common service for all applications.
- Due to base task the development of new tasks for the ECOS service is standardized and unified.

The solution meets all applications functional requirements.

Production itself becomes more intellectual and being able to benefit from the colossal amounts of data received from working devices becomes a critical success factor. It is important to carefully analyze and understand how to maintain one or another component of production, that it will continue to work efficiently for many years and how the company can use the information obtained to further improve production. Digitalization is a strong trend associated with the integrated introduction of digital technologies at all stages of the creation and operation of a product. Digitalization has a very large impact not only on processes, but also on business models.

Resümee

Töö pealkiri on „ Teenuse arendamine, mis käivitab ülesanded ajakava järgi “. Töö eesmärk oli arendada teenust, mis on võimeline andma operatiivseid andmeid Eesti Energia erinevatele rakendustele ECOS platvormi baasil. ECOS platvorm on Eesti Energia areng, mis loodi tulevaste ettevõtete rakenduste arendamise protsessi ühtlustamiseks ja kiirendamiseks.

Töö koosneb seitsmest peatükist. Peatükk 1 kirjeldab kirjeldab ECOS platvormi tehnoloogilist virna. Peatükk 2 kirjeldab ECOS rakenduste funktsionaalseid nõudeid Peatükis 3 kirjeldatakse „Task Scheduler“ ja selle komponente. Peatükis 4 kirjeldatakse rakendamiseks valitud tehnoloogiaid. Peatükk kirjeldab arenguprotsessi. Peatükk kirjeldab andmebaasi. Peatükk 7 kirjeldab rakendusi, mis kasutavad arendatud teenust.

Arendamise käigus oli tehtud teenus mis käivitab ülesanded ajakava järgi mis tõttu rakendused saavad operatiivandmeid. Oli rakendatud ülesande logimise süsteem, mis jälgib ülesande käivitamise protsessi.

References

1) Microsoft., 2019, .Net. Available at https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet, accessed April 8, 2019.
2) Microsoft, 2019. MVC. Available at https://dotnet.microsoft.com/apps/aspnet/mvc, accessed April 8, 2019
3) Microsoft, 2019.Entity Framework Documentation. Available at https://docs.microsoft.com/en-us/ef/#pivot=entityfmwk&panel=entityfmwk1, accessed April 8, 2019
4) Mills, Chris 2019. JavaScript. Available at https://developer.mozilla.org/en-US/docs/Web/JavaScript ,accessed April 8, 2019.
5) jQuery Available at https://jquery.com/ , accessed April 8, 2019.
6) Bootstrap, 2019. Documentation Bootstrap. Available at https://getbootstrap.com/ , accessed April 8, 2019.
7) .MDM web docs, 2019. HTML Available at https://developer.mozilla.org/en-US/docs/Web/HTML , accessed April 8, 2019.
8) MDM web docs, 2019. CSS Available at https://developer.mozilla.org/en-US/docs/Web/CSS, accessed April 8, 2019.
9) Technopedia, 2019. Windows Server Available at https://www.techopedia.com/definition/359/windows-server, accessed April 8, 2019.
10) Microsoft, 2019. Entity Framework Documentation. Available at https://docs.microsoft.com/en-us/ef/#pivot=entityfmwk&panel=entityfmwk1 , accessed April 8, 2019.
11) Microsoft, 2019. What is .Net? Available at https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet , accessed April 8, 2019.
12) Hellos Solutions 2019.The Pros and Cons of Microsoft .NET Framework. Available at https://heliossolutionsblog.blogspot.com/2017/01/the-pros-and-cons-of-microsoft-net.html, accessed April 8, 2019
13) Microsoft, 2019. Task Scheduler Glossary. Available at https://docs.microsoft.com/ru-ru/windows/desktop/TaskSchd/task-scheduler-start-page, accessed April 8, 2019.
14) Microsoft, 2019. Task Scheduler Glossary. Available at https://docs.microsoft.com/en-us/windows/desktop/taskschd/tasks, accessed April 8, 2019.

15) Microsoft, 2019. Task Scheduler Glossary. Available at https://docs.microsoft.com/en-us/windows/desktop/taskschd/t, accessed April 8, 2019.

16) Quartz.Net, 2019. Frequently Asked Questions. Available at https://www.quartz-scheduler.net/documentation/faq.html, accessed April 8, 2019.

17) Hangfire, 2019. Documentation. Available at https://docs.hangfire.io/en/latest/, accessed April 8, 2019.

18) Apache, 2019. Log4Net. Available at thttps://logging.apache.org/log4net/release/faq.htm , accessed April 8, 2019.

19) Microsoft, 2019. NotMappedAttribute. Available at https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations.schema.notmappedattribute?view=netframework-4.8, accessed April 8, 2019.

20) Refactoring Guru, 2019. Singleton. Available at https://refactoring.guru/design-patterns/singleton, accessed April 8, 2019.

Appendices

Appendix 1

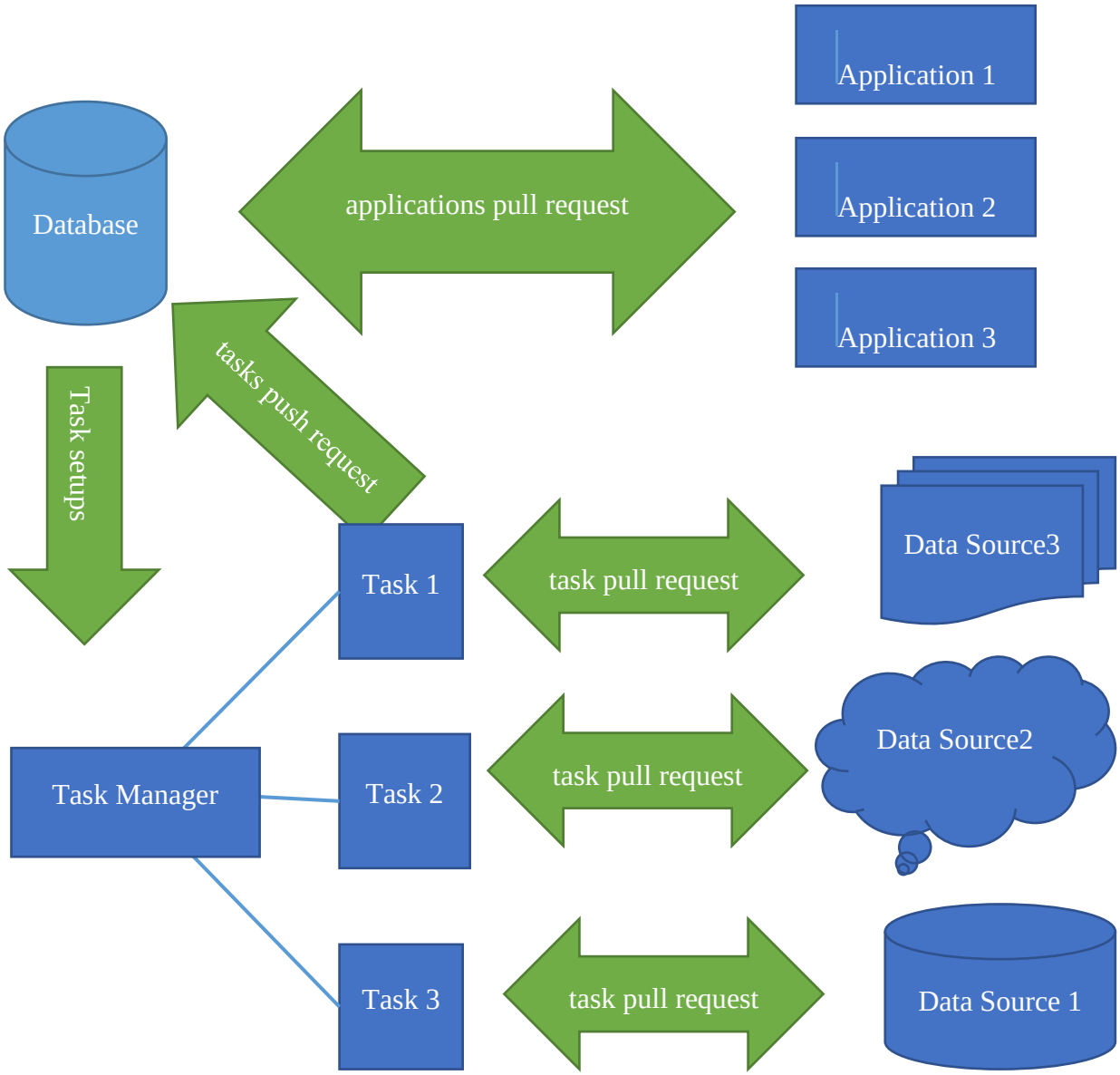

Figure 19. *Solution architecture (Source: author).*

-

Appendix 2

Flash memory stick with source code.