

TALLINN UNIVERSITY OF TECHNOLOGY  
School of Information Technologies

Artjom Jakovlev 179633IAIB

# **CREATING A SOFTWARE FOR PRESENTING AND ENACTING PROCESSES**

Bachelor's thesis

Supervisor: Erki Eessaar  
PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL  
Infotehnoloogia teaduskond

Artjom Jakovlev 179633IAIB

# **PROTSESSIDE ESITAMISE NING LÄBIMÄNGIMISE TARKVARA LOOMINE**

Bakalaureusetöö

Juhendaja: Erki Eessaar  
PhD

Tallinn 2020

## **Author's declaration of originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Artjom Jakovlev

24.05.2020

## **Abstract**

The aim of this thesis is to design and implement a web-based software that could be used for creating interactive representations of processes that can be accessed by other users who are then able to move through the different steps of a published process. This is supposed to be more interactive than alternative representations of procedural guidelines and could encourage users to experiment with choosing different options and arriving at different outcomes.

There is already a similar system created back in 2008 as a bachelor's thesis for the same goals. The supervisor of both theses is the same. This thesis aims to redesign the system from scratch, choosing better solutions for the design and adding some additional functionality.

Based on the supervisor's wishes the system consists of a PostgreSQL database and a web application built using PHP. The communication between the database and the application happens through the virtual data layer achieved by an extensive usage of functions and views. The integrity of the data in the database is guaranteed by declarative constraints and triggers that are created in the database.

As the result, processes can be successfully represented in the system, but the addition of new processes through a web interface is not finished as of the time of writing this abstract. Nevertheless, a web application that uses the virtual data layer can be created that allows users to manage processes. Compared to the older system, the new system also has a more modern design, and contains new elements of processes that can be displayed.

The working web application is accessible at <http://apex.ttu.ee/processes/> as of May 2020. The software's source code is open source (protected with the MIT license) and can be found at <https://github.com/edelmoedig/ProcessEnacting-IAIB2020>.

In its chapters the thesis describes the idea, related work and system analysis performed for designing this system. It also explains the database design and design of the web application's front-end and back-end.

This thesis is written in English and is 62 pages long, including 7 chapters, 28 figures and 4 tables.

## **Annotatsioon**

# **PROTSESSIDE ESITAMISE NING LÄBIMÄNGIMISE TARKVARA LOOMINE**

Selle lõputöö eesmärk on kavandada ja realiseerida veebipõhine tarkvara, mida saaks kasutada protsesside interaktiivse esituse loomiseks. Loodud protsessidele pääsevad ligi teised kasutajad, kes seejärel võivad protsesse samm-sammult läbides ja valikuid tehes neid läbimängida. Tulemus peaks olema interaktiivsem kui erinevate protseduuriliste juhiste alternatiivsed esitused. See võiks julgustada kasutajat katsetama erinevate võimaluste valimisega erinevate lõpptulemuste saamiseks.

Sama eesmärgi saavutamiseks loodi 2008. aastal bakalaureusetöö tulemusel sarnane süsteem. Mõlemal lõputööl on sama juhendaja. Käesoleva lõputöö eesmärk on vana süsteem nullist ümber kirjutada, valides selleks paremad lahendused ja lisades mõned uued funktsionaalsused.

Vastavalt juhendaja soovile koosneb süsteem PostgreSQL andmebaasist ja PHP abil ehitatud veebirakendusest. Seos andmebaasi ja rakenduse vahel toimub virtuaalse andmekihi kaudu, mis on saavutatud andmebaasi funktsioonide ja vaadete laialdase kasutamisega. Andmebaasi andmete terviklikkus tagatakse andmebaasis loodud deklaratiivsete kitsendustega ja trigeritega.

Töö tulemusena saab protsesse süsteemis edukalt esitada, aga uute protsesside lisamine veebiliidese kaudu ei ole selle annotatsiooni kirjutamise hetkeks veel realiseeritud. Saab luua veebirakenduse, mis virtuaalse andmekihi vahendusel võimaldab protsesse hallata. Võrreldes vanema süsteemiga on uus süsteem ka moodsama kujundusega ja võimaldab kasutada uusi protsessi elemente.

Töötav veebirakendus asub 2020. aasta maikuu seisuga aadressil <http://apex.ttu.ee/processes/>. Tarkvara on avatud lähtekoodiga (kaitstud MIT litsentsiga) ja see asub aadressil <https://github.com/edelmoedig/ProcessEnacting-IAIB2020>

Selle lõputöö peatükkides on kirjeldatud süsteemi idee, seotud tööd, süsteemianalüüs, andmebaas ning veebirakenduse ees- ja tagasüsteem.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 62 leheküljel, 7 peatükki, 28 joonist, 4 tabelit.

## List of abbreviations and terms

ANSI	American National Standards Institute
CASE	Computer Aided Software Engineering A set of labor-saving tools used in software development that allow creating models to support following software development lifecycle [1].
CSS	Cascading Style Sheets “A style sheet language used for adding style (e.g., fonts, colors, spacing) to Web documents” [2].
DBMS	Database Management System A software for implementing, managing, and providing access to databases.
Domain	In SQL reusable specification of column properties where one must specify data type and can specify things like default value, NOT NULL constraint, and CHECK constraints.
HTML	Hypertext Markup Language “A markup language that is used to create documents on the World Wide Web incorporating text, graphics, sound, video, and hyperlinks” [3].
HTTP	Hypertext Transfer Protocol A protocol for transferring data over a network.
HTTPS	Hypertext Transfer Protocol Secure An extension of the HTTP that uses TLS (or SSL) to encrypt HTTP requests and responses [4].
PDO	PHP Data Objects An interface for PHP that allows accessing databases regardless of their DBMSs.
PHP	PHP: Hypertext Preprocessor “A scripting language used to create dynamic and interactive HTML Web pages” [5].
SQL	Structured Query Language A special-purpose programming language designed for managing data, database objects, privileges, transactions, etc. in a relational database that has been built up based on the

underlying data model of SQL. It is used by a huge number of apps and organizations [6].

SSL	Secure Sockets Layer A cryptographic protocol that is now deprecated in favor of TLS.
TLS	Transport Layer Security A cryptographic protocol.
Trigger	In SQL a set of actions that is executed when a certain event occurs, possibly when some additional conditions are met. The event can be a data modification in a base table or a view.
UML	Unified Modeling Language A standardized graphical language for “visualizing, specifying, constructing, and documenting the artifacts” of mostly information and software systems [7].
URL	Uniform Resource Locator A reference to a Web resource, most often a webpage, that “specifies its location on a computer network and a mechanism for retrieving it” [8].
WFMS	Workflow Management System “A software system that provides the infrastructure to arrange, track, control, and coordinate the business processes known as workflows” [9].

## Table of contents

1 Introduction .....	15
1.1 Problem.....	15
1.2 Objective.....	15
1.3 Tools and Processes .....	16
1.4 Structure of the Thesis .....	16
2 Related Work.....	18
2.1 Users Do Not Like Manuals and What to do About This .....	18
2.2 Dmitri Karlõšev’s Thesis.....	18
2.3 Modelling Processes .....	19
2.4 Workflow Management Systems.....	20
2.5 Interactive Fiction.....	21
3 System Analysis .....	22
3.1 Areas of Competence.....	22
3.2 Functional Subsystems .....	22
3.3 Registers .....	22
3.4 User Stories.....	23
3.5 Non-functional Requirements.....	24
3.5.1 Database .....	25
3.5.2 Back-end.....	25
3.5.3 Front-end .....	25
3.5.4 Language .....	25
4 Database Design .....	26
4.1 Tables.....	27
4.2 Views .....	31
4.3 Functions .....	31
4.4 Security .....	32
4.5 Data Integrity .....	33
5 Back-end Design .....	36

5.1 Database Connection .....	36
5.2 Structure.....	36
5.3 Security .....	37
6 Front-end Design .....	39
6.1 Implementation.....	39
6.2 Navigation .....	39
6.3 Main Page .....	40
6.4 Process Enactment .....	41
6.5 Search .....	42
6.6 Work in Progress .....	42
7 Summary.....	43
References .....	45
Appendix 1 – Alternative Applications .....	49
Appendix 2 – Entity-Relationship Diagrams.....	50
Appendix 3 – Physical Database Design Diagrams .....	53
Appendix 4 – Code Examples .....	57
Appendix 5 – Example Diagram .....	62

## List of figures

Figure 1. Trigger preventing activation of processes with invalid parallel activities.....	34
Figure 2. Top navigation menu with three links shown. ....	39
Figure 3. The main page of the web application listing active processes. ....	40
Figure 4. The container with the process's details. ....	40
Figure 5. A step (parallel activity in this case) of a process in the web application. .	41
Figure 6. A decision step with two options in the web application.....	41
Figure 7. The search form for the active processes by name. ....	42
Figure 8. Interface of draw.io. ....	49
Figure 9. Interface of Bizagi Modeler. ....	49
Figure 10. Register of administrators. ....	50
Figure 11. Register of classifiers. ....	50
Figure 12. Register of decision tables. ....	50
Figure 13. Register of process usages. ....	51
Figure 14. Register of processes.....	51
Figure 15. Register of processes (steps). ....	52
Figure 16. Physical design of the register of administrators. ....	53
Figure 17. Physical design of the register of classifiers. ....	53
Figure 18. Physical design of the register of decision tables.....	54
Figure 19. Physical design of the register of process usages.....	55
Figure 20. Physical design of the register of processes.....	56
Figure 21. Physical design of the register of processes (steps). ....	56
Figure 22. An anonymous function that can be used to insert the example process. ....	58
Figure 23. Two main tables – Process and Step – that depend on each other.....	60
Figure 24. One of the two domains in the database, used for registration time. ....	60
Figure 25. An example of a function: function used to add the first step (Action here) to a process. ....	60
Figure 26. An example of a trigger: trigger used to prevent invalid status changes. ....	61
Figure 27. An example of a view: basic information about steps. ....	61

Figure 28. Process of submitting and grading the independent work in the course  
Databases I (in 2020)..... 62

## **List of tables**

Table 1. Number of different types of objects in the database.....	26
Table 2. Explanation of the meaning of tables. ....	27
Table 3. Explanation of the meaning of columns. ....	29
Table 4. Structure of the PHP back-end. ....	37

# **1 Introduction**

A process is “a series of actions that you take in order to achieve a result” [10]. The goal of this thesis is to design and implement a web-based system that allows people to create step-by-step representations of real-life processes that can be accessed and “walked through” by other people.

## **1.1 Problem**

There is already a previously created system for this goal [11] [12]. Because it was designed back in 2008, it is technically outdated by now.

This thesis focuses more on redesigning and reimplementing its technical side from scratch with some added functionality.

## **1.2 Objective**

As with various Workflow Management Systems that share some similarities with this piece of software, users with the administrative role are allowed to specify processes consisting of multiple steps. End users can then walk through the processes by simply accessing the application that is available on the public Web. At each step of the process, the user is presented with information related to the step (possibly including links or decision tables) and is able to make a decision and transition into one of the next possible steps, eventually arriving at the final step.

The system is meant to interactively guide a single user through a process and does not involve other users (i.e., there are no requests for approval to transition between steps/states like in workflow systems [13]) nor does it automate the process itself. The intent of the system is to make procedural guidelines more lively and interactive by allowing users to “walk through” and play with them by selecting different options, and tunnel them through different paths and options by allowing them to concentrate to their

particular needs and interests. For instance, the system could be used to explain processes related to studying a university course.

While this system attempts to fulfill the needs of a specific client (the thesis's supervisor in this case), its usefulness is not limited to only some specific domains of usage. The created software can be used to represent any real-life process regardless of the real-life area of usage it belongs to. The software's source code is publicly available at <https://github.com/edelmoedig/ProcessEnacting-IAIB2020>. The software is open source and has been licensed with the MIT License [14].

### **1.3 Tools and Processes**

Enterprise Architect 12 CASE tool is used for modeling the database of the system and generation of the statements for creating base tables (tables), declarative constraints, and indexes.

From the technical side, the system consists of a database and a website. According to the specific requirements of the client, the system must be able to work in the environment that has PostgreSQL (at least version 10) and PHP (version 5).

The system is successfully tested with the database that uses the PostgreSQL 12 database management system (DBMS). The website part was built by using pure PHP 5.6 without any frameworks. The website is hosted on the Apache web server. The website meets the HTML5 standards and is styled with Fomantic UI as the CSS framework. During the development, the database and the website were hosted locally on a Windows machine similarly running Apache server with PHP and PostgreSQL DBMS.

A process shown in a UML activity diagram for the course “Andmebaasid I (ITI0206)” (Figure 28) is chosen as the example process that is implemented by using the newly created system.

### **1.4 Structure of the Thesis**

In chapter 2 related work of the thesis topic, including existing software, is explained. Chapter 3 presents the requirements (both functional and non-functional) to the new system. Chapter 4 explains the design of the underlying database of the software. Chapters

5 and 6 explain the back-end and front-end of the software, respectively. Finally, the summary is provided that concludes the work and points towards future work with the current topic.

## **2 Related Work**

A justification for the chosen method of representing processes and multiple ways to visually represent real processes are described in this chapter.

### **2.1 Users Do Not Like Manuals and What to do About This**

Various authors report, based on scientific research [15] or anecdotal evidence [16], that users do not like reading manuals and long explanations. For instance, Novick and Ward [15] found, based on 25 in-depth interviews, that the subjects avoided using both paper and online help systems because these are hard to navigate, have wrong level of detail, and they thought it was easier to ask from someone else or try to find the answer themselves. User interface designers suggest using onboarding tours in the software [17] [18] that walk the user through the features of an application or highlight the key features, as well as wizards [19] that walk the user through a sequence of steps so that they can achieve a goal. The present system tries to use the same approach in order to explain processes to its users. The system's creators are encouraged and motivated by the following design patterns of persuasion in user interfaces.

- Reduction of complex behavior into a set of sequential tasks [20].
- Allowing simulation so that users can play with a system without being afraid of messing something up [21].
- Sequencing, according to which complex activities should be decomposed into smaller and more manageable tasks in order to make it easier to take action [22].

### **2.2 Dmitri Karlůšev's Thesis**

A system for visualizing workflows was created by Dmitri Karlůšev back in 2008 [11]. The thesis had the same supervisor as the current thesis. It is also a web-based application created using PHP and PostgreSQL, where workflows or processes (used there synonymously) are shown as sequences of steps (activities and decision points) between

which the user can transition by making choices at the decision points. Each step has a mandatory textual description or instruction and optional related links and files.

The process implementations (processes in short) are created by process administrators. The processes are listed on the website [12] and can be accessed by anyone who knows the URL and has web access. The users can go through a process by reading information provided to them at each step and making decisions as to how to proceed at decision points. In this way the users can “play” with different scenarios (sequences of steps through the process) and find out what the implications in case of following the scenario that is described by the process would be in the real life. The goal was to make descriptions more “lively” – users can experiment with different scenarios instead of reading a wall of the text. Different users might have to follow different scenarios that are more applicable to their situations than the others and in this way each user can get information that is the most relevant to him/her.

As mentioned before, the thesis is based on this previous work and aims to improve it.

## **2.3 Modelling Processes**

Models are simplifications of reality. Models can be visualized by using diagrams. Diagrams are like views – each model element can be depicted on zero or more diagrams and each diagram depicts zero or more model elements. Models can describe the static structure of a system as well as its behavior, including the processes that take place in or around it. Thus, processes too can be visualized using diagrams. A possibility is to use flowcharts with symbols assigned by the American National Standards Institute [23]. Over the years many process modeling diagram notations have been proposed like functional flow block diagram (FFBD) [24], control-flow diagrams [25], graphical program evaluation and review technique [26], all of which have some applications. Perhaps nowadays the most well-known process modeling notations are UML (Unified Modelling Language) activity diagrams [27] and BPMN (Business Process Modeling and Notation) [28]. The process that will be implemented in the part of the thesis where we validate the developed program is initially presented as a UML activity diagram (see Figure 28 from Appendix 5).

In order to create the diagrams one can use drawing tools (like Visio) or CASE tools (like Enterprise Architect). These tools could be desktop applications or web-based environments.

Many of the programs [29] [30] for creating flowcharts are available online, like for example diagrams.net, VisualParadigm Online, Microsoft Visio, Lucidchart or Smartdraw; others like Edraw Max must be installed on the computer. The programs are mostly similar to each other in terms of their basic functionality: diagram elements are dragged and dropped into the grid, and connective arrows can be created between them. The designed diagrams can be shared as a direct link or be converted into picture format. Some programs such as VisualParadigm Online and Lucidchart are free of charge and offer additional functionality for subscribing to their service. Some examples of the added functionality are more storage, more templates, collaborative work, and integration with other services.

One example of the web-based programs is diagrams.net (previously known as Draw.io). It is completely free, open source, and does not require the user to register to use it. It offers a selection of templates and elements that comply with the ANSI or UML standards (Figure 8 in Appendix 1).

In case of drawing tools, the result is essentially a static picture with no possibility to simulate the depicted process. In case of models created in CASE tools one could also, for instance, search models from a set of models based on their content, generate documentation, simulate processes, or generate code from the models [31].

Another example is the free version of Bizagi Modeler [32]. Its interface allows constructing processes by dragging the first step into the modeling area and then adding new steps by clicking on the previous steps and dragging connections if needed (Figure 9 in Appendix 1) [33]. Bizagi Modeler is a part of a line of solutions for defining and implementing workflows, i.e., automating processes.

## **2.4 Workflow Management Systems**

A workflow can be defined as “collection of tasks organized to accomplish some business process” [34] with the distinction made between material processes, information processes, and business processes. Material processes are performed by humans,

information processes are automated, whereas business processes are market-centered implementations of material or information processes that describe an organization's activities [34] [35]. Workflow Management Systems (WFMS) systems provide organizations with the means of overseeing their workflows.

While WFMS systems allow visualizing processes, they are intended to be used in a multi-user environment. They also offer some degree of automation (either involving humans or automatically processing and modifying input data according to some pre-existing algorithms), which is out of scope of this work. For instance, in case of the paid versions of Bizagi one can let the system automate the processes by interpreting the model as well as simulate the processes [36] in order to, for instance, optimize the processes. In the workflow systems it may be possible to specify processes in terms of visual models (diagrams) (for instance, the aforementioned Bizagi Modeler).

In comparison, all three types of processes can be represented in the designed system, but there is just a single user that should proceed with completing the tasks according to the provided instructions and move on to the next step without automation of the tasks in question. At least in the current software release it will not be possible to specify processes visually or conduct simulations. Instead, one will have to use a form-based user interface to specify processes.

## **2.5 Interactive Fiction**

The topic of this thesis is also somewhat related to interactive novels or gamebooks. These types of fiction typically have a starting point, the reader is asked to make a choice to move to the next step and so on. In the end the reader arrives to one of the possible endings [37]. The main similarity with the topic of this thesis is that interactive novels and the designed program are both text-based, as opposed to diagrams.

Programs available for creating interactive novels usually output the result as HTML and allow the author to use CSS and JavaScript [38].

## **3 System Analysis**

This chapter reviews the requirements to the system. Firstly, we decompose the system into smaller parts (subsystems) as suggested, for instance, by Eessaar [39]. There are three types of subsystems. Areas of competence correspond to the roles of the users of the system. The representatives of the areas of competence use the services of functional subsystems. Each functional subsystem corresponds to a main function (a chunk of functionality) of the system. Each functional subsystem uses the services (reads data or modifies data) of one or more registers. One should create the user interface, application, and database based on the areas of competence, functional subsystems, and registers, respectively. The subsystems and their interactions constitute the business architecture of the system.

### **3.1 Areas of Competence**

- Administrator's area of competence
- User's area of competence

### **3.2 Functional Subsystems**

- Process management functional subsystem
- Administrator management functional subsystem
- Classifier management functional subsystem

### **3.3 Registers**

- Register of processes
- Register of decision tables
- Register of process usages
- Register of administrators
- Register of classifiers

### 3.4 User Stories

This section presents the basic functionality of the process enactment system as expressed in terms of user stories [40].

The user story that is related to the administrator management functional subsystem:

- As an administrator, I want to authenticate, so that I can receive required permissions for creating processes.

The user stories that are related to the process management functional subsystem:

- As an administrator, I want to create a process, so that I can start editing it.
- As an administrator, I want to add steps to the newly created process (meaning the first step that is connected directly to the process and the following steps that follow the previously added), so that any sequence of real steps can be depicted.
- As an administrator, I want to create processes that can depict parallel steps (all of which must be completed to transition further), so that real-life processes are represented accurately.
- As an administrator, I want to create processes with decision points, so that the users can choose the most suitable course of action out of those presented for them.
- As an administrator, I want to assign weights to some or all options of a decision step that are visible to the users, so that the users can make a more informed decision about the further course of action.
- As an administrator, I want to add a link to a web resource to any step of the process, so that the users can open them to get additional information about the current step, carry out some actions required for proceeding further at the location of that link, or to download required files from the link.
- As an administrator, I want to add a link to a web resource to the process in general and that is visible at every step of the process, so that the users can get additional information about the process or download files that are in some way required for this process.
- As an administrator, I want to add decision tables to any step, so that the users can make a decision based on the table or receive otherwise relevant information about the step.

- As an administrator, I want to have a possibility to assign a password to a process, so that only the users who know the password can walk through the process.
- As an administrator, I want the process to be accessible through a permanent link (URL) that can be shared with other people, so that they can access the process through it.
- As an administrator, I want to search through a list of process to find the process that needs modification or review, or to make sure that there is no such process already.
- As an administrator, I want to be able to change the status of the process, so that it can be temporarily hidden from the users to be edited or for any other possible purpose, or so that it can be permanently hidden from the users.
- As an administrator, I want to edit an already created process: change its name, its password, add, modify, or remove steps and connections between them as well as add, modify, or remove links, so that the process is kept up-to-date and the users access its latest version.
- As an administrator, I want to be sure that the process is correctly implemented technically, so that users can successfully finish it.
- As an administrator, I want to be able to collect statistics about the completion of my process, so that I can use the statistics to modify either the enactment of the process in this application or the real-life process itself.
- As a user, I want to be able to access a list of processes, so that I can choose the one I want to go through.
- As a user, I want to go through a process, so that I can gain the relevant information.
- As a user, I want to access a password-protected process, so that I can go through it.
- As a user, I want to be able to search processes to find the one that is currently the most relevant to me.

### **3.5 Non-functional Requirements**

This subchapter lists non-functional requirements to the software that were presented by the supervisor.

### **3.5.1 Database**

The system should use the PostgreSQL (at least version 10) DBMS.

### **3.5.2 Back-end**

The system should use PHP (version 5) and shouldn't require installation of any extra components/programs in the server. Every piece of supporting software that the system needs must be provided within the folder of the web application or linked from the Internet.

### **3.5.3 Front-end**

The system must have a web interface. The website should be implemented using HTML, CSS and JavaScript.

### **3.5.4 Language**

The website should be available in English and Estonian with the possibility to add more languages in the future.

The language used in the database (in case of the identifiers of database objects and comments) is English.

## 4 Database Design

The following chapter describes the design of the database used in the system.

Appendix 2 shows the entity-relationship diagrams of the conceptual model of the database, while the Appendix 3 shows the physical design.

Table 1 shows the different types of objects in the database. All of them are located in the schema called *processes*.

Table 1. Number of different types of objects in the database.

Database object	Count
Base table (Table)	15
Column of a table	63
View	10
Column of a view	54
Function	87 (including those of pgcrypto extension)
Trigger	38
Domain	2

The exclusive usage of functions and views for operations on the database creates a layer of abstraction (so called virtual data layer) between the application and the database, maximizing the reusability of the data structures [41]. Its other advantages include possibilities to hide certain base table structure changes behind the layer, i.e., these do not require changes in the application source code. Moreover, the layer can be used to implement a layer of security in the multi-layered security system. More precisely, the database user under which the application uses the database can read data through views and modify data through user-defined routines (functions in this case) but it cannot access directly base tables. A technical difficulty in PostgreSQL is that making changes in the structure of base tables that have dependent views required dropping and recreating the views [42].

## 4.1 Tables

There are 15 tables in the database that belong to the registers found during the system analysis. Table 2. Explanation of the meaning of tables. Table 2 explains the meaning of the tables to human users.

Table 2. Explanation of the meaning of tables.

<b>Table</b>	<b>Register</b>	<b>Definition</b>
Action	Register of processes	The most basic type of a step that represents a straightforward action that can either lead to another step or be the final step unless inside a parallel activity.
Action_in_parallel_activity	Register of processes	An action that is contained inside a parallel activity. Cannot lead to the next step.
Administrator	Register of administrators	A registered user that can create, modify, or otherwise manipulate the processes.
Decision	Register of processes	A type of a step that contains multiple options (at least two) inside itself. Cannot lead to the next step.
Decision_table	Register of processes	A table that is connected to an action step and that specifies conditions and their respective actions.
Decision_table_entry	Register of processes	A condition-action pair.
Option	Register of processes	A choice inside the decision step. Must lead to the next step.
Parallel_activity	Register of processes	A type of a step that contains multiple actions (at least two) inside itself. All of the actions must be completed to proceed further, and their order does not matter. Must lead to the next step.
Process	Register of processes	A sequence of steps that helps someone or something to achieve a goal.
Process_link	Register of processes	A link associated with the whole process that is displayed on every step of the process in the web application.

<b>Table</b>	<b>Register</b>	<b>Definition</b>
Process_status_type	Register of classifiers	A status of a process on which the process's visibility, accessibility, and modifiability depend.
Process_usage	Register of process Usages	Statistics collected every time a user starts walking through the process.
Step	Register of processes	A single part in a series of transitions inside the process. A step must belong to exactly one out of three types: action, decision, or parallel activity.
Step_click	Register of process Usages	Statistics collected every time a user arrives on the step during the process usage.
Step_link	Register of processes	A link associated with just one step that is displayed only during that step in the web application.

Table 3 explains the meaning of all the non-foreign key columns to human users. The database design follows some conventions.

- Values of all the surrogate keys are generated automatically by the system. It is achieved by using SERIAL/BIGSERIAL notation.
- None of the textual columns can contain empty strings and strings that consist of only whitespace characters. It is achieved with CHECK constraints (see Figure 23 in Appendix 4).
- Timestamps (for instance, registration time) must be between January 1, 2020 and December 31, 2200 (end points included). It is achieved with a CHECK constraints. Time zone and fractional seconds are not registered and the default value is found by using the expression LOCALTIMESTAMP(0).
- A domain is created if it is worth the effort – the domain is used in case of multiple columns and if it has enough “content”, i.e., specifies a CHECK constraint or a default value. In the present project a domain is used in case of temporal columns (see Figure 24 in Appendix 4) and another domain is used in case of columns that must keep URLs.

Table 3. Explanation of the meaning of columns.

<b>Table name</b>	<b>Column name</b>	<b>Column description</b>
Action_in_parallel_activity	reg_time	Registration time.
Administrator	administrator_id	Primary key column. Surrogate key.
Administrator	email	E-mail address of the administrator. Must have case insensitive uniqueness and is used as the username in case the administrator has to log in. Must contain at least one @ sign.
Administrator	password	Salted and hashed password of the administrator.
Administrator	given_name	At least one of these names must be registered, i.e., NOT NULL.
Administrator	surname	
Administrator	is_active	Whether the administrator can access the system (TRUE) or not (FALSE).
Administrator	reg_time	Registration time.
Decision_table	decision_table_id	Primary key column. Surrogate key.
Decision_table	name	Name of the decision table.
Decision_table	is_active	Whether the decision table is displayed to an end user who walks through the process (TRUE) or not (FALSE).
Decision_table	reg_time	Registration time.
Decision_table_entry	decision_table_entry_id	Primary key column. Surrogate key.
Decision_table_entry	condition	Free form description of a condition that determines which action to perform. Must be unique within a decision table.
Decision_table_entry	action	Free form description of an action that is performed if the condition is met.
Decision_table_entry	seq_nr	Determines the order of displaying decision table entries. Must be unique within a decision table.
Decision_table_entry	reg_time	Registration time.
Option	option_id	Primary key column. Surrogate key.

<b>Table name</b>	<b>Column name</b>	<b>Column description</b>
Option	weight	Numeric value that determines the relative importance of the option.
Option	reg_time	Registration time.
Option	guard	Condition that must be met to choose the option.
Process	process_id	Primary key column. Surrogate key.
Process	name	Name of the process.
Process	description	Free form explanation of the process.
Process	reg_time	Registration time.
Process	password	Optional salted and hashed password. If the process is password protected, i.e., the password is registered, then the end user who accesses it must provide the password.
Process_link	process_link_id	Primary key column. Surrogate key.
Process_link	url	URL of a web-resource that is relevant in terms of the entire process. Must start with "http://" or https://. Must be unique within a process.
Process_link	name	Name of the process link.
Process_link	priority_nr	Determines the order of displaying links in case a process has multiple links. Must be unique within a process.
Process_link	reg_time	Registration time.
Process_status_type	process_status_type_code	Code of the process status. It is registered by a human user, not generated by the system.
Process_status_type	name	Name of process status.
Process_usage	process_usage_id	Primary key column. Surrogate key.
Step	step_id	Primary key column. Surrogate key.
Step	reg_time	Registration time.
Step	description	Free form explanation of the step.
Step_click	step_click_id	Primary key column. Surrogate key.

Table name	Column name	Column description
Step_click	click_time	Timestamp of a moment when the step was accessed by an end user who walks through the process.
Step_link	step_link_id	Primary key column. Surrogate key.
Step_link	url	URL of a web-resource that is relevant in terms of the step. Must start with "http://" or https://. Must be unique within a step.
Step_link	name	Name of the step link.
Step_link	priority_nr	Determines the order of displaying links in case a step has multiple links. Must be unique within a step.
Step_link	reg_time	Registration time.

## 4.2 Views

There are 10 views that are used for SELECTing data to display in the application:

- all\_processes,
- active\_inactive\_on\_hold\_processes,
- active\_processes,
- process\_steps (see Figure 27 of Appendix 4),
- parallel\_actions,
- decision\_options,
- decision\_tables,
- decision\_table\_entries,
- process\_links,
- step\_links.

## 4.3 Functions

All of the required INSERTs, UPDATEs, and DELETEs can be done in the database using user-defined functions. Most of the functions are used to create and modify processes and other objects that are connected to them. In total at the time of writing the thesis, there are 49 non-trigger user-defined functions that do not belong to any extension.

46 functions are written in SQL and one in PL/pgSQL. Many functions use a feature of PostgreSQL according to which one can change data in multiple tables with one statement and can use the identifier that was generated while inserting a row to a table within the same statement to modify data in other tables (see Figure 25 of Appendix 4).

Using these functions, a process is constructed step-by-step. For example, there are five closely related functions for adding steps.

1. The new step is the first step of the process.
2. The new step is added so that it is connected to the previous step but is not connected to the next step at the time of creation.
3. The new step is added so that it is connected to the previous and next steps at the time of creation.
4. The new step is added to an option but is not connected to the next step at the time of creation.
5. The new step is added to the option and leads to an existing next step.

Together with using triggers, this approach guarantees that a process is constructed correctly and can be successfully finished after its activation.

In order to demonstrate it, the process depicted in Appendix 5, Figure 28 can be created using the code from Appendix 4, Figure 22.

#### **4.4 Security**

The right to connect to the database is taken from PUBLIC (that is, any user who could theoretically connect to the database) and is restricted to two users (corresponding to the application) created specifically for this database: *process\_administrator* and *process\_user*. All the database objects are created in the schema called *processes*. Access to this schema and to the schema *public* is also taken from PUBLIC. Finally, rights to execute routines are also taken from PUBLIC.

The functions are created with SECURITY DEFINER meaning that they have the privileges of the function's creator which allows the unprivileged users that have access to them execute operation with the higher privileges. However, they are restricted to only using these functions and operations inside them, which improves security.

There are views that are created for showing information from the tables required for the web application. As additional security, the views are created with *security\_barrier* property to reliably hide hidden rows and prevent potential malicious functions from extracting more information, and with the CHECK OPTION on those views that refer only to one table to prevent UPDATES and INSERTs that do not comply with the definition of the view.

Access to the schema *processes* along with the right to execute the functions and SELECT from the views is given to the *process\_administrator* and *process\_user* depending on their needs, with the former receiving access to all of the functions and the latter receiving access only to the functions for authorization and logging process usages. The principle of least privilege [43] was used to give privileges to these users, i.e., they got only the privileges that are essential to complete their tasks.

The passwords in the database are salted and hashed using the PostgreSQL extension *pgcrypto* with the Blowfish algorithm.

## **4.5 Data Integrity**

The software uses BEFORE INSERT, BEFORE UPDATE, and BEFORE DELETE triggers to ensure the integrity of the data inside the database in case it is not possible to achieve it declaratively by using database constraints. These triggers fire corresponding trigger functions that prevent the called operation from succeeding and raise exceptions if the constraints are violated. The trigger functions are written in PL/pgSQL language.

The extensive usage of triggers in the database instead of processing the logic in the application has its pluses and minuses. The pluses of the trigger-based approach is the guarantee that data modification is checked against the enforced rule regardless of the way it is entered to the database (through an application or perhaps directly by using an administrative interface), the guarantee of atomicity (either all of the operations occur or none of them), and the ability to change the business logic without changing the application's code. Its minuses are the difficulties in migrating from the DBMS because the triggers must be rewritten, the inner workings of the triggers that are sometimes unclear to the user, and a decrease in the speed of the operations in the database [44].

As an example of a trigger in the database, the trigger that prevents the activation of a process with a parallel activity that has less than two actions in it can be seen in Figure 1. The trigger function uses explicit locking of a table to deal with the situation when at the same time of activation parallel activities are deleted.

```

CREATE OR REPLACE FUNCTION
processes.f_activate_process_parallel_activity_less_than_2_actions() RETURNS
trigger AS
$$
DECLARE
    v_count bigint;
BEGIN
    LOCK TABLE processes.Process, processes.Action_in_parallel_activity,
processes.Parallel_activity, processes.Step IN ACCESS EXCLUSIVE MODE;
    v_count := (SELECT Count(*)
                FROM (SELECT Parallel_activity.parallel_activity_id,
count(action_id)
                    FROM processes.Parallel_activity
                    INNER JOIN processes.Step ON
parallel_activity_id = step_id
                    LEFT JOIN
processes.Action_in_parallel_activity
                    ON
Action_in_parallel_activity.parallel_activity_id =
Parallel_activity.parallel_activity_id
                    WHERE Step.process_id = NEW.process_id
                    GROUP BY Parallel_activity.parallel_activity_id
                    HAVING Count(*) < 2) AS Parallel_action_count);
    IF v_count > 0 THEN
        RAISE EXCEPTION 'There are % parallel activities that have less than
2 parallel actions.', v_count;
    ELSE
        RETURN NEW;
    END IF;
END;

$$ LANGUAGE plpgsql SECURITY DEFINER
    SET search_path = processes, public, pg_temp;
CREATE TRIGGER trig_activate_process_parallel_activity_less_than_2_actions
BEFORE UPDATE OF process_status_type_code
ON processes.Process
FOR EACH ROW
WHEN (OLD.process_status_type_code <> NEW.process_status_type_code AND
NEW.process_status_type_code = 2)
EXECUTE FUNCTION
processes.f_activate_process_parallel_activity_less_than_2_actions();

```

Figure 1. Trigger preventing activation of processes with invalid parallel activities.

The trigger function uses explicit locking of a table to deal with the situation when at the same time of activation of a process a parallel activity of the process is deleted. PostgreSQL uses multiversion concurrency control method [45], according to which reading a data element does not block updating or deleting the element and vice versa. Thus, in case of control logic sometimes explicit locking of specific rows or the entire table is needed. Another example of a trigger is presented in Figure 26 of Appendix 4. It is simpler and it only has to access the row which modification fired it. Thus, its function does not have to conduct explicit locking of rows.

## 5 Back-end Design

The following chapter describes the part of the system's backend that was created using pure PHP without any extensions. The software is not so big and thus the influence on the developer's productivity was not a real problem. Moreover, frameworks that implement the use of object-relational mapping have their own problems (like potential leaky abstractions, weakened data validation, performance, problems, etc.) [46].

As of the time of finishing this thesis, this part along with the corresponding parts of the website, has not been completely finished, but the plan is to complete it before the defense of the thesis.

### 5.1 Database Connection

The connection to the database is established using PDO. The parameters are read in the class `Connection` from the file `config.ini`. The object of this class is in every `.php` file where the connection is required.

There are two more classes: *Administrator* that contains queries for registering a new process administrator and *Process* that contains queries for selecting the data that needs to be displayed during the walkthroughs of the processes along with the logging functions that save statistic about step clicks to the database. As it is not currently possible to create processes from the application, the required PDO functions that call the corresponding functions from the database are missing from here right now.

### 5.2 Structure

Generally, most of the queries to the database and the following logic are located at the top of the `.php` files. PHP scripts are also called inside the HTML code in order to generate those parts of the page that depend on the result of the queries.

There are seven `.php` files in this project not including classes `Administrator`, `Process`, and `Connection`. The explanation of their purpose is shown in Table 4.

Table 4. Structure of the PHP back-end.

File name	Purpose
enact.php	The page that shows all available information about the current step that can be received from the database. process_id and step_id are parameters of the query string (pr={process_id}&step={step_id}) in the URL. The step must belong to the corresponding process and the process must be accessible to the user, otherwise the user is redirected to the main page. Includes navigation.
index.php	The main page that lists all active processes that can be accessed by ordinary users. Includes navigation and search. Search uses the query string “search={value}” in the URL, and when it is set only shows the processes have this substring in their names (using LIKE in the query to the database).
login.php	The page used by administrators to log in. Includes navigation.
logout.php	PHP script that ends the current session.
navigation.php	A navigation menu located at the top of the screen. Always contains the link to the main page, shows additional links to administrators.
register.php	The page used for registration of new administrators. Includes navigation.
search.php	A search form that is used for searching among all active processes by name.

### 5.3 Security

All of the parameterized queries are first prepared and then executed to prevent SQL injection. htmlspecialchars() is used on inputs where possible in order to sanitize them.

Function header() is used to move the user from the location he or she should not access to a different one, and exit() is used to terminate the running script. A session is started and \$\_SESSION variables are used to store information about the current session, like for example, whether the current user is an administrator or not: for an administrator \$\_SESSION['id'] is set to the id received during the login. During the logout the previously set variables are unset and the session is destroyed.

The passwords are sent as plain text to the database and are hashed in the database. Because the connection over at the apex.ttu.ee server happens over the HTTP protocol and not the HTTPS, this is potentially unsafe [47] [48] [4]. The solution would be to place the production version of the software to the server that supports HTTPS.

## 6 Front-end Design

This chapters describes the design of the website.

### 6.1 Implementation

The frontend is designed using the open-source CSS framework Fomantic-UI, which is a community fork of another CSS framework called Semantic-UI. It requires jQuery to fully function. The framework and its dependencies are included in the head of every page.

All of the styling has been done using Fomantic-UI classes and there is no styling outside of the framework as of now.

### 6.2 Navigation

The navigation menu is location in the middle top of the screen (Figure 2). When on the main page or inside the processes, a guest user only sees the link “Processes” that takes the user to the main page. When on the administrative page and not logged in, there are three menu entries that lead to the list of processes, registration page, or login page respectively. When logged it, the latter two buttons are replaced with one “Log out” button. When the currently missing functionality of designing processes inside the application is added, a logged-in administrator will be able to see a link to the list of all modifiable processes.

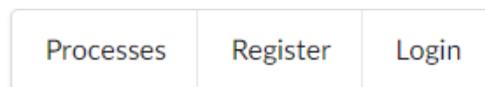


Figure 2. Top navigation menu with three links shown.

## 6.3 Main Page

The main page of the website (Figure 3) lists all active processes that can be accessed by the users.

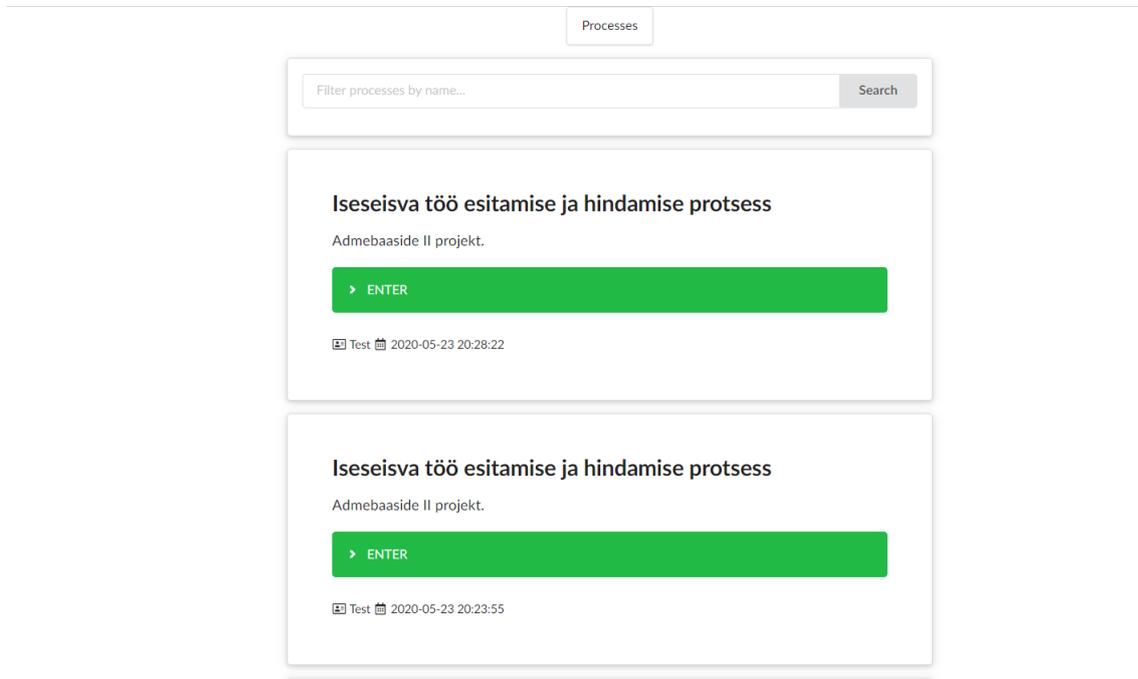


Figure 3. The main page of the web application listing active processes.

Figure 4 shows the part with a process's information shown in the list up close. The name, description, owner, and registration time of the process are show along with the button that allows the user to access the process.

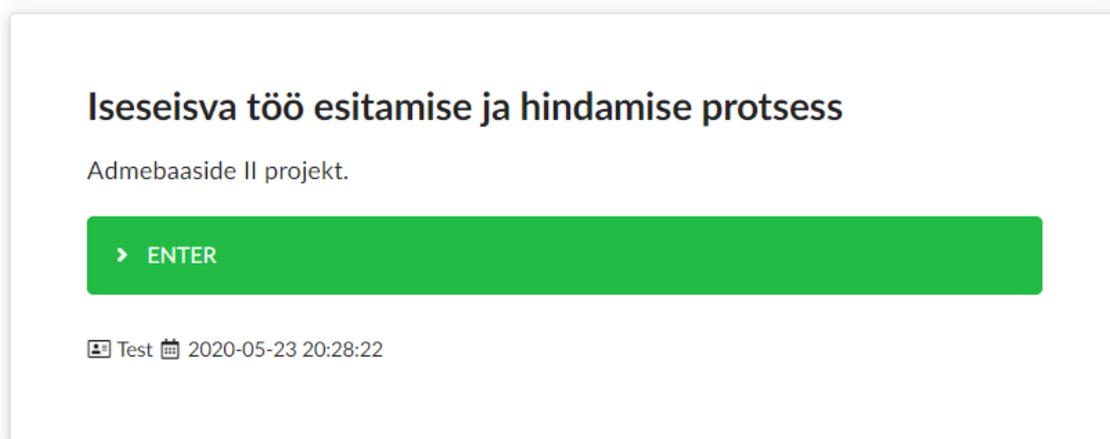


Figure 4. The container with the process's details.

## 6.4 Process Enactment

Figure 5 shows how a step looks during the “walkthrough” of a process. On the left there is the name of a process, clicking on which takes the user back to the first step, in the middle there is a description of the step (with descriptions of multiple parallel actions in this case) with the buttons leading to the next steps to the bottom of it. On the right there are related links, with the process links followed by the step links. In this case, the parallel activity itself has no related links, but every parallel action inside this parallel activity has an associated link.

The screenshot displays a web application interface for a process. At the top center, there is a tab labeled "Processes". On the left side, there is a navigation menu with a back arrow and the text "Iseseisva töö esitamise ja hindamise protsess". The main content area is titled "Parallel activity" and contains the text "Ettevalmistus". Below this, there is a list of two actions: "1. Registreeri ettenäitamisele" and "2. Lae failid Maurusesse". A green button with a right arrow and the text "NEXT" is positioned below the list. On the right side, there are two sections: "Process links" and "Step links". The "Process links" section contains two links: "Ainekava ÕiSis" with URL <https://ois.ttu.ee/aine/ITI0206> and "Õppeaine koduleht" with URL <https://maurus.ttu.ee/378>. The "Step links" section contains two links: "1. Registreeri ettenäitamisele" with URL <https://maurus.ttu.ee/tk.php?aine=378> and "Ettenäitamisele registreerimise tegevusdiagramm" with URL <https://maurus.ttu.ee/download.php?aine=378&document=35412&tyyp=do>. The second link in the "Step links" section is "2. Lae failid Maurusesse" with URL [https://maurus.ttu.ee/t\\_ylesanded.php?aine=378](https://maurus.ttu.ee/t_ylesanded.php?aine=378).

Figure 5. A step (parallel activity in this case) of a process in the web application.

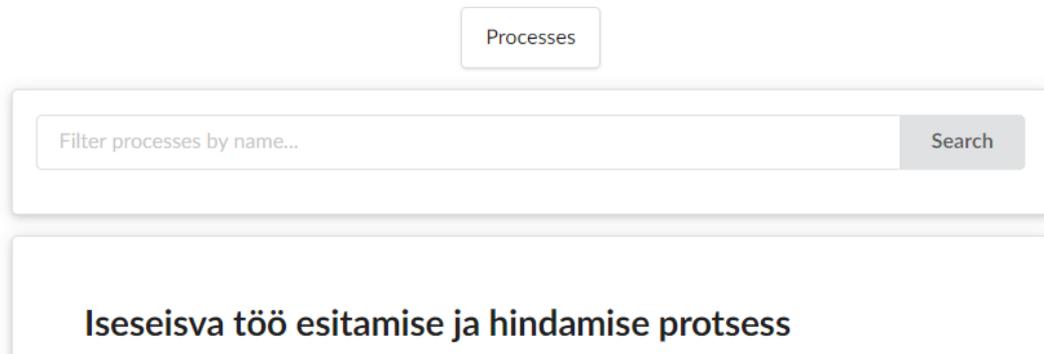
Figure 6 shows a decision step in the same process. By clicking on one of the two buttons, the user can move to one of the possible steps.

The screenshot displays a web application interface for a process. At the top center, there is a tab labeled "Processes". On the left side, there is a navigation menu with a back arrow and the text "Iseseisva töö esitamise ja hindamise protsess". The main content area is titled "Decision" and contains the text "Kas saan kohale tuua?". Below this, there are two buttons with right arrows and text: "Ei JA ettenäitamiseni rohkem kui 24 tundi" and "Jah VÕI (ei ja ettenäitamiseni vähem kui 24 tundi)". On the right side, there are two sections: "Process links" and "Step links". The "Process links" section contains two links: "Ainekava ÕiSis" with URL <https://ois.ttu.ee/aine/ITI0206> and "Õppeaine koduleht" with URL <https://maurus.ttu.ee/378>. The "Step links" section contains the text "There are no associated step links."

Figure 6. A decision step with two options in the web application.

## 6.5 Search

The search function (Figure 7) is located on the main page. It has only one input field that is used for entering the substring that should be in the process's name in order for it to show up in the results.



The image shows a web interface for searching processes. At the top, there is a button labeled "Processes". Below it is a search bar with the placeholder text "Filter processes by name..." and a "Search" button. The search results area below the bar displays the text "Iseseisva töö esitamise ja hindamise protsess".

Figure 7. The search form for the active processes by name.

## 6.6 Work in Progress

At the time of submitting the thesis the following main functionalities were unimplemented or partially implemented due to the mistakes in planning.

- Web-application for the administrator to manage processes.
- Access to password-protected processes by end users.
- Support of multiple languages in the user interface.

The plan now is to complete these before the defense of the thesis in June 2020.

Retrospectively, it would have been important to start the work earlier and employ some method for release planning like, for instance, the one that is proposed by Norman [49].

## 7 Summary

The objective of this thesis was to design and implement a web-based system than could be used for specifying representations of processes. These would be subsequently accessed and walked through (enacted) by other people. The representations of processes are displayed to the users as sequences of steps, among which the users can transition and eventually arrive to the final step, successfully completing the enactment of the process.

The system's functional needs were given by the supervisor and additionally found with the help of the user stories from the perspectives of an administrator and a guest user. In order to design the database that could fulfill these needs, the required areas of competence, functional subsystems, and registers were found during the system analysis. Enterprise Architect 12 was used to create the entity-relationship diagrams and physical database design diagrams. The latter was specifically created for PostgreSQL. The initial version of the SQL code that contained the statements for creating tables, declarative constraints, and indexes was generated with the help of Enterprise Architect.

The SQL code that was used for the creation of the final version of the database was completed by manually adding other types of database objects, such as domains, functions, triggers, and views. All of the interactions with the database from the web application happen through these functions and views. Moreover, the database usage privileges were removed from unauthorized users and were given to two new users corresponding to the roles of a process administrator and a simple user. These users are used by the application to access the database. The privileges to the users were given according to their needs and following the principle of least privilege. The integrity of the data that is kept in the database is guaranteed by using declarative constraints and triggers that stop operations that lead to inconsistencies and mistakes in data. To summarize, the database contains everything that is needed for specifying and accessing process representations.

As for what concerns the web application written in PHP and HTML using the Fomantic-UI CSS framework, currently it is missing the functionality for managing the processes

by the administrators: there is no working PHP and HTML code written for it at the time of submission. The website can be used to display and access the processes that have been manually created by using the database functions. At the time of the submission it shows correctly different steps and links related to these, allows to transition between steps, collects statistics about the accessed steps, and permits registering and logging in. Nevertheless, as such the usefulness of this application is severely limited. The other function that is described in the user stories but is currently missing is accessing a password-protected process. The website is also currently available only in English. All these functionalities will be added by the time of defense the thesis in June 2020.

The best way of designing a new process should be similar to modelling a flowchart where every step of the process and the connections between the steps are seen all the time. The most similar method to this is used in Bizagi Modeler [33]. This method essentially turns a diagram into its step-by-step representation that is then shown to the users. An alternative that is easier to implement is to make the stage of creation like the walkthrough of a process that is currently implemented, however this requires the administrator to constantly go back to modify steps and have less overview over the process overall, which is a major inconvenience.

When compared with the older version of the similar process enactment system, this implementation has a more modern interface, supports parallel activities, option weights, and decision tables, and has an arguably better database design in general.

Both the database and the website are hosted on apex.ttu.ee as of May 2020. The website is available on <http://apex.ttu.ee/processes/>. The software's source code is open source (protected with MIT license) and can be found at <https://github.com/edelmoedig/ProcessEnacting-IAIB2020>.

Future work could include experimentation with user interface to find the best ways for specifying and presenting processes. Another line of work is to implement the system by using a graph-based DBMS (for instance, Neo4j). A goal of this would be to find out as to whether it is better suited for such task than PostgreSQL or not.

## References

- [1] M. Rouse, "Computer-aided software engineering (CASE)," October 2018. [Online]. Available: <https://searcherp.techtarget.com/definition/CASE-computer-aided-software-engineering>. [Accessed 23 May 2020].
- [2] "Cascading Style Sheets," The World Wide Web Consortium, [Online]. Available: <https://www.w3.org/Style/CSS/Overview.en.html>. [Accessed 23 May 2020].
- [3] "HTML Definition," Merriam-Webster, [Online]. Available: <https://www.merriam-webster.com/dictionary/HTML>. [Accessed 23 May 2020].
- [4] "Why is HTTP not secure? | HTTP vs. HTTPS," Cloudflare, [Online]. Available: <https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/>. [Accessed 25 May 2020].
- [5] "PHP: Hypertext Preprocessor (PHP)," Techopedia , 5 November 2011. [Online]. Available: <https://www.techopedia.com/definition/24406/php-hypertext-preprocessor-php>. [Accessed 23 May 2020].
- [6] "Intro to SQL: Querying and managing data," Khan Academy, [Online]. Available: <https://www.khanacademy.org/computing/computer-programming/sql>. [Accessed 23 May 2020].
- [7] "Unified Modeling Language," Object Management Group, [Online]. Available: <https://www.omg.org/spec/UML/>. [Accessed 23 May 2020].
- [8] "URL," [Online]. Available: <https://en.wikipedia.org/wiki/URL>. [Accessed 25 May 2020].
- [9] "Save Time by Taking the Time: Creating Workflows," Smartsheet, [Online]. Available: <https://www.smartsheet.com/save-time-taking-time-creating-workflows>. [Accessed 23 May 2020].
- [10] "Meaning of process in English," Cambridge Dictionary, [Online]. Available: <https://dictionary.cambridge.org/dictionary/english/process>. [Accessed 11 April 2020].
- [11] D. Karlõsev, "Töövoo süsteemi projekteerimine ja realiseerimine," Bakalaureusetöö. TTÜ Informaatikainstituut, 2008.
- [12] D. Karlõsev, "Töövoo süsteemi projekteerimine ja realiseerimine," [Online]. Available: <http://apex.ttu.ee/protsessid/>. [Accessed 21 February 2020].
- [13] M. Jones, "Designing a Workflow Engine Database," [Online]. Available: <https://exceptionnotfound.net/designing-a-workflow-engine-database-part-1-introduction-and-purpose/>. [Accessed 21 February 2020].
- [14] "The MIT License," Open Source Initiative, [Online]. Available: <https://opensource.org/licenses/MIT>. [Accessed 25 May 2020].
- [15] D. G. Novick and K. Ward, "Why don't people read the manual?," in *Proceedings of the 24th Annual ACM International Conference on Design of Communication*, Myrtle Beach, 2006.

- [16] J. Spolsky, "Designing for People Who Have Better Things To Do With Their Lives," 26 April 2000. [Online]. Available: <https://www.joelonsoftware.com/2000/04/26/designing-for-people-who-have-better-things-to-do-with-their-lives/>. [Accessed 24 May 2020].
- [17] "Real Users Don't Read Manuals," Open Social, 20 June 2017. [Online]. Available: Real Users Don't Read Manuals. [Accessed 24 May 2020].
- [18] M. Cook, "UX Flows: How to Turn Onboarding into an Amazing First Date with Your User," Telepathy, [Online]. Available: UX Flows: How to Turn Onboarding into an Amazing First Date with Your User. [Accessed 24 May 2020].
- [19] "Wizard," UI Patterns, [Online]. Available: <https://ui-patterns.com/patterns/Wizard>. [Accessed 25 May 2020].
- [20] "Reduction," UI Patterns, [Online]. Available: <https://ui-patterns.com/patterns/Reduction>. [Accessed 25 May 2020].
- [21] "Simulation," UI Patterns, [Online]. Available: <https://ui-patterns.com/patterns/Simulation>. [Accessed 25 May 2020].
- [22] "Sequencing," UI Patterns, [Online]. Available: <https://ui-patterns.com/patterns/Sequencing>. [Accessed 25 May 2020].
- [23] "ISO 5807:1985 [ISO 5807:1985] Information processing — Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts," 1985.
- [24] "Systems Engineering Fundamentals," Defense Acquisition University Press, 2001.
- [25] R. M. Smelik, "Specification and Construction of Control Flow Semantics a generic approach using graph transformations".
- [26] A Guide to the Project Management Body of Knowledge, Project Management Institute, 2013.
- [27] "Activity Diagrams," uml-diagrams.org, [Online]. Available: <https://www.uml-diagrams.org/activity-diagrams.html>. [Accessed 25 May 2020].
- [28] "Business Process Model and Notation," December 2013. [Online]. Available: <https://www.omg.org/spec/BPMN/2.0.2/PDF>. [Accessed 1 April 2020].
- [29] B. Aston, "The Best Flowchart Software Of 2020," The Digital Project Manager, 1 January 2020. [Online]. Available: <https://thedigitalprojectmanager.com/flowchart-software/>. [Accessed 1 April 2020].
- [30] K. Franks, "What's the best workflow diagram software?," 2020. [Online]. Available: <https://www.jotform.com/blog/workflow-diagram-software/>. [Accessed 23 February 2020].
- [31] S. Backhaus, "Code Generation for UML Activity Diagrams in Real-Time Systems," Hamburg University of Technology, 2016.
- [32] "Bizagi Modeler," Bizagi, [Online]. Available: <https://www.bizagi.com/en/platform/modeler>. [Accessed 22 May 2020].
- [33] Bizagi, "Bizagi Modeler Tutorial: How to Model Your First Business Process using BPMN," [Online]. Available: <https://www.youtube.com/watch?v=GpXYgNVcdMU>. [Accessed 23 May 2020].

- [34] D. Georgakopoulos, M. Hornick and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distrib Parallel Databases*, no. 3, p. 119–153, 1995.
- [35] R. Medina-Mora, T. Winograd and R. Flores, "ActionWorkflow as the Enterprise Integration Technology," *Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society*, vol. 16, no. 2, 1993.
- [36] "Simulation in Bizagi," Bizagi, [Online]. Available: [http://help.bizagi.com/bpm-suite/en/index.html?simulation\\_in\\_bizagi.htm](http://help.bizagi.com/bpm-suite/en/index.html?simulation_in_bizagi.htm). [Accessed 25 May 2020].
- [37] N. Montfort, *Twisty Little Passages: An Approach to Interactive Fiction*, The MIT Press, 2005.
- [38] J. Paul, "Best open source tools to create Interactive Fiction," 9 November 2019. [Online]. Available: <https://itsfoss.com/create-interactive-fiction/>. [Accessed 8 March 2020].
- [39] E. Eessaar, "A Set of Practices for the Development of Data-Centric Information Systems," in *22nd International Conference on Information Systems Development (ISD2013)*, Seville, 2013.
- [40] "User Stories," [Online]. Available: <https://www.mountangoatsoftware.com/agile/user-stories>. [Accessed 28 February 2020].
- [41] L. Burns, *Management, Building the Agile Database: How to Build a Successful Application Using Agile Without Sacrificing Data*, Technics Publications, 2011.
- [42] "PostgreSQL: Drop column and recreate dependent views," Stack Overflow, [Online]. Available: <https://stackoverflow.com/questions/50455507/postgresql-drop-column-and-recreate-dependent-views>. [Accessed 25 May 2020].
- [43] N. Lord, "What is the Principle of Least Privilege (POLP)? A Best Practice for Information Security and Compliance," 12 September 2018. [Online]. Available: <https://digitalguardian.com/blog/what-principle-least-privilege-polp-best-practice-information-security-and-compliance>. [Accessed 25 May 2020].
- [44] A. Põlluste, "Veebi- ja andmebaasipõhise metamodelleerimise vahendi üleviimine andmebaasi triggeritel põhinevaks süsteemiks," *Magistritöö. TTÜ Informaatikainstituut*, 2016.
- [45] "PostgreSQL Concurrency with MVCC," Heroku, 20 February 2019. [Online]. Available: <https://devcenter.heroku.com/articles/postgresql-concurrency>. [Accessed 25 May 2020].
- [46] A. Korban, "The case against ORMs," 2 November 2017. [Online]. Available: <https://korban.net/posts/postgres/2017-11-02-the-case-against-orms/>. [Accessed 25 May 2020].
- [47] "Passwords in the Clear," World Wide Web Consortium, 8 October 2008. [Online]. Available: <https://www.w3.org/2001/tag/doc/passwordsInTheClear-52>. [Accessed 25 May 2020].
- [48] "How to secure passwords over HTTP?," Stack Overflow, [Online]. Available: <https://security.stackexchange.com/questions/197330/how-to-secure-passwords-over-http>. [Accessed 25 May 2020].
- [49] T. Norman, "Agile Release Planning 101," 6 September 2012. [Online]. Available: <http://tommynorman.blogspot.com/2012/09/agile-release-planning-101.html>. [Accessed 25 May 2020].

[50] "Validating process errors," Bizagi, [Online]. Available:  
[http://help.bizagi.com/process-modeler/en/index.html?validating\\_process\\_errors.htm](http://help.bizagi.com/process-modeler/en/index.html?validating_process_errors.htm). [Accessed 24 May 2020].

## Appendix 1 – Alternative Applications

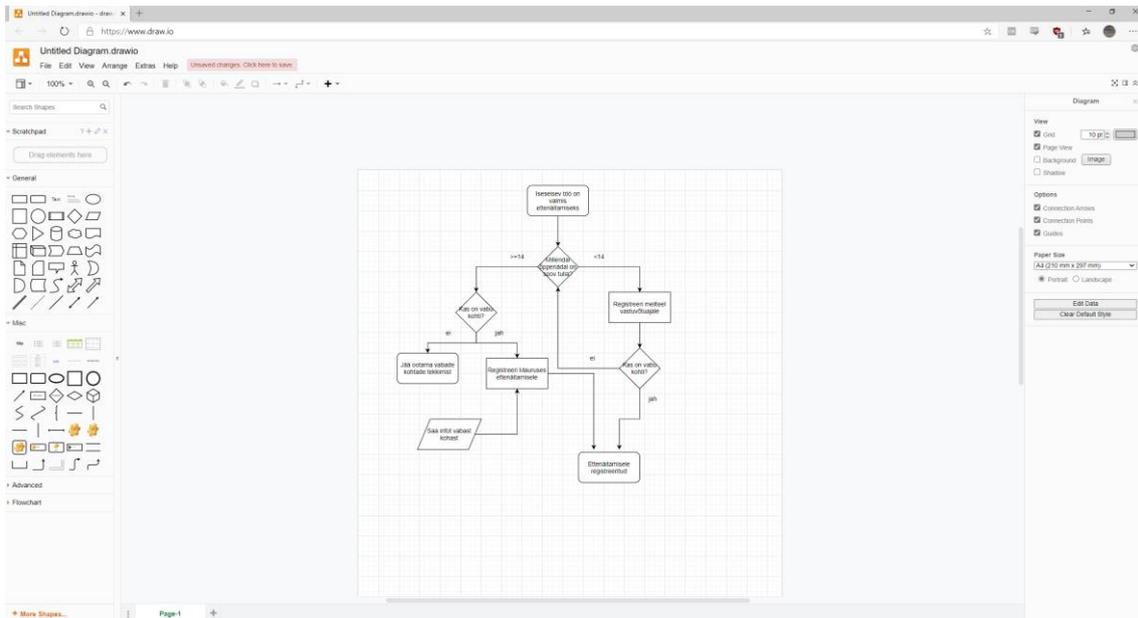


Figure 8. Interface of draw.io.

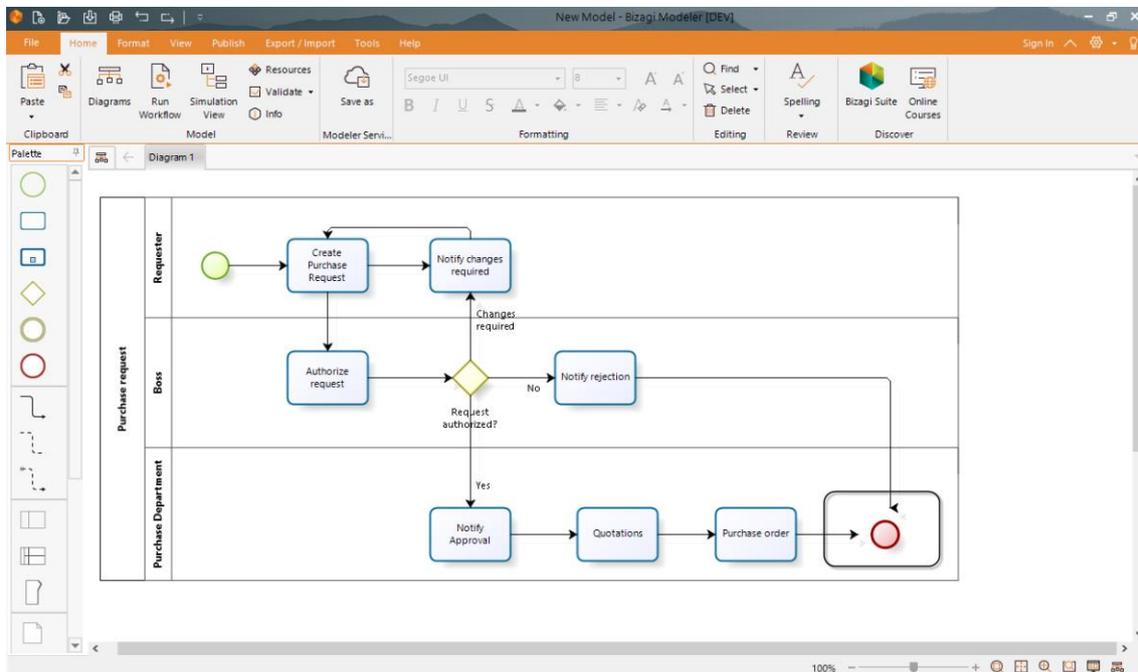


Figure 9. Interface of Bizagi Modeler [50].

## Appendix 2 – Entity-Relationship Diagrams

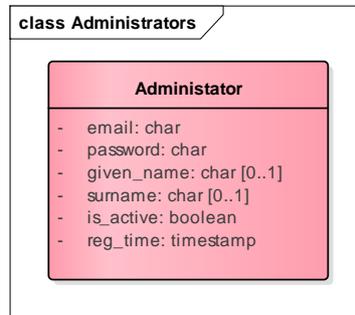


Figure 10. Register of administrators.

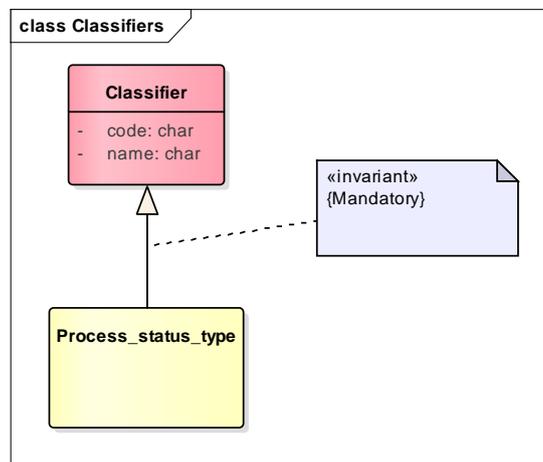


Figure 11. Register of classifiers.

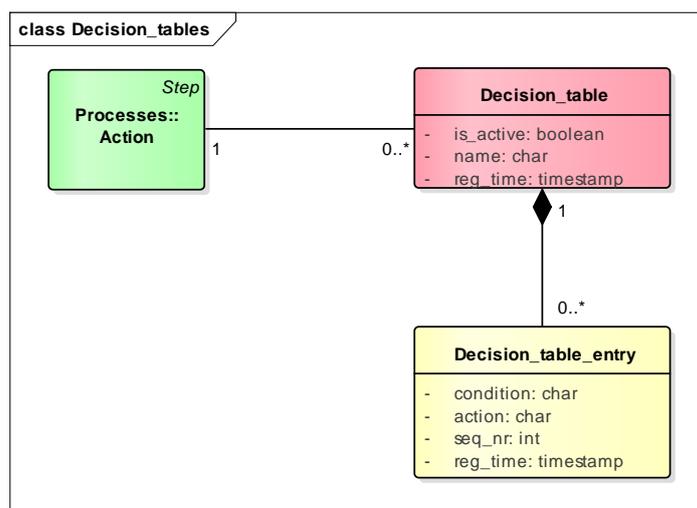


Figure 12. Register of decision tables.

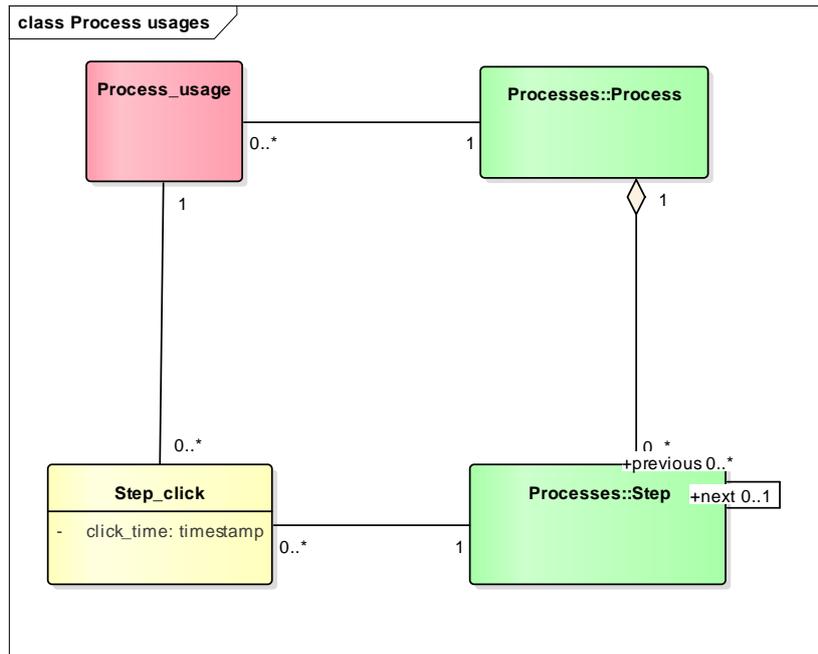


Figure 13. Register of process usages.

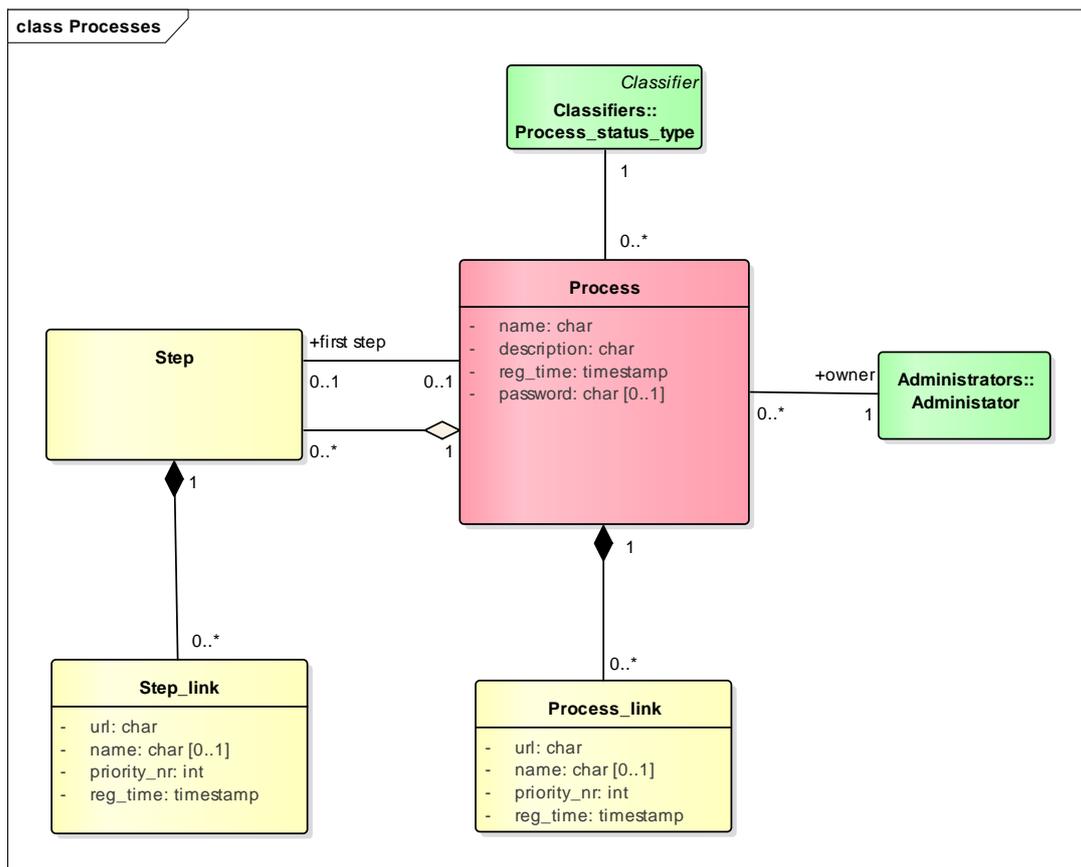


Figure 14. Register of processes.

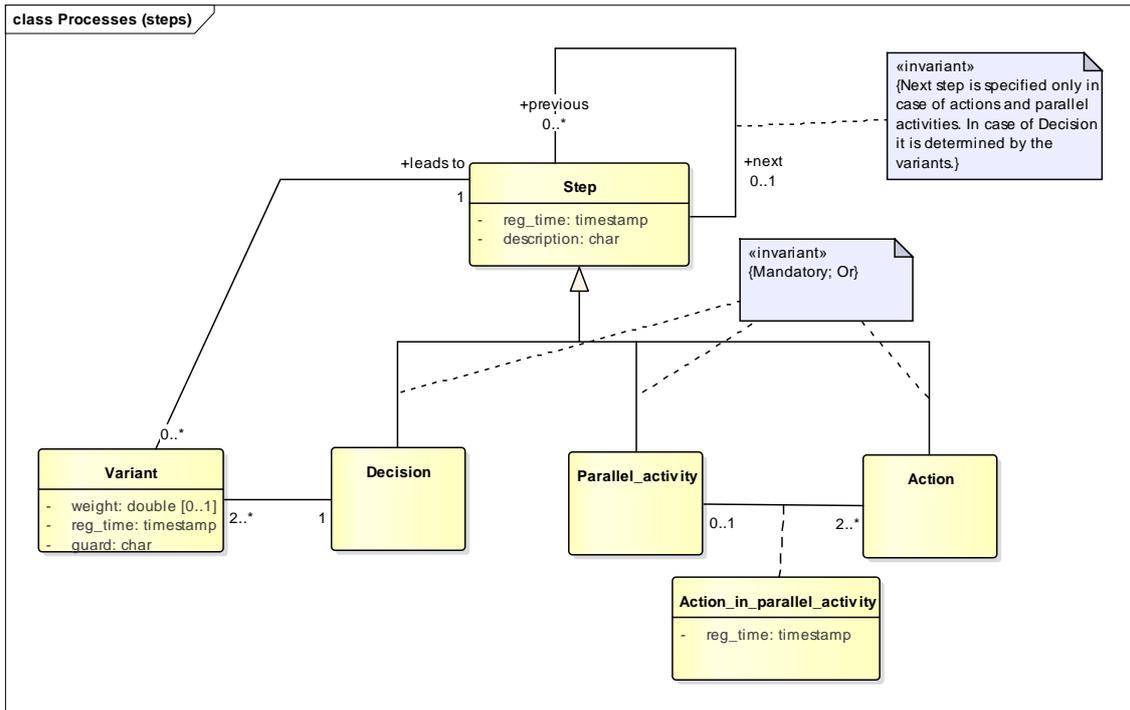


Figure 15. Register of processes (steps).

## Appendix 3 – Physical Database Design Diagrams

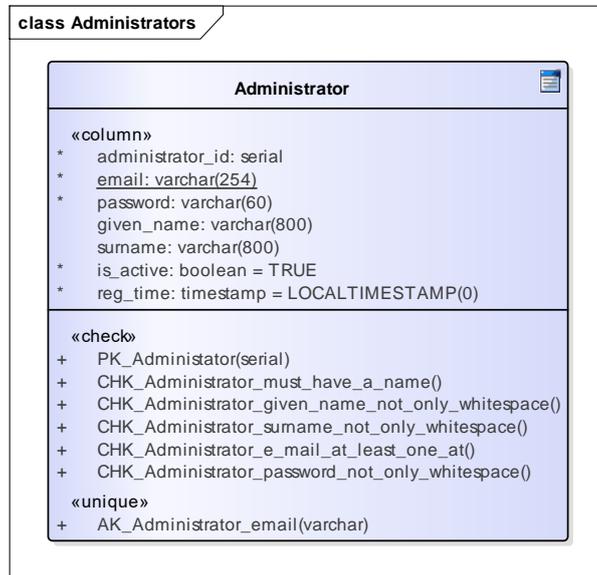


Figure 16. Physical design of the register of administrators.

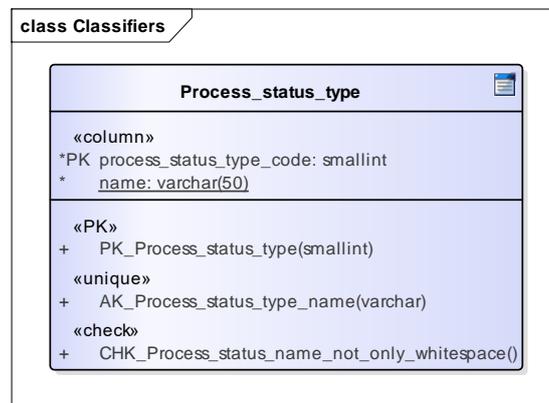


Figure 17. Physical design of the register of classifiers.

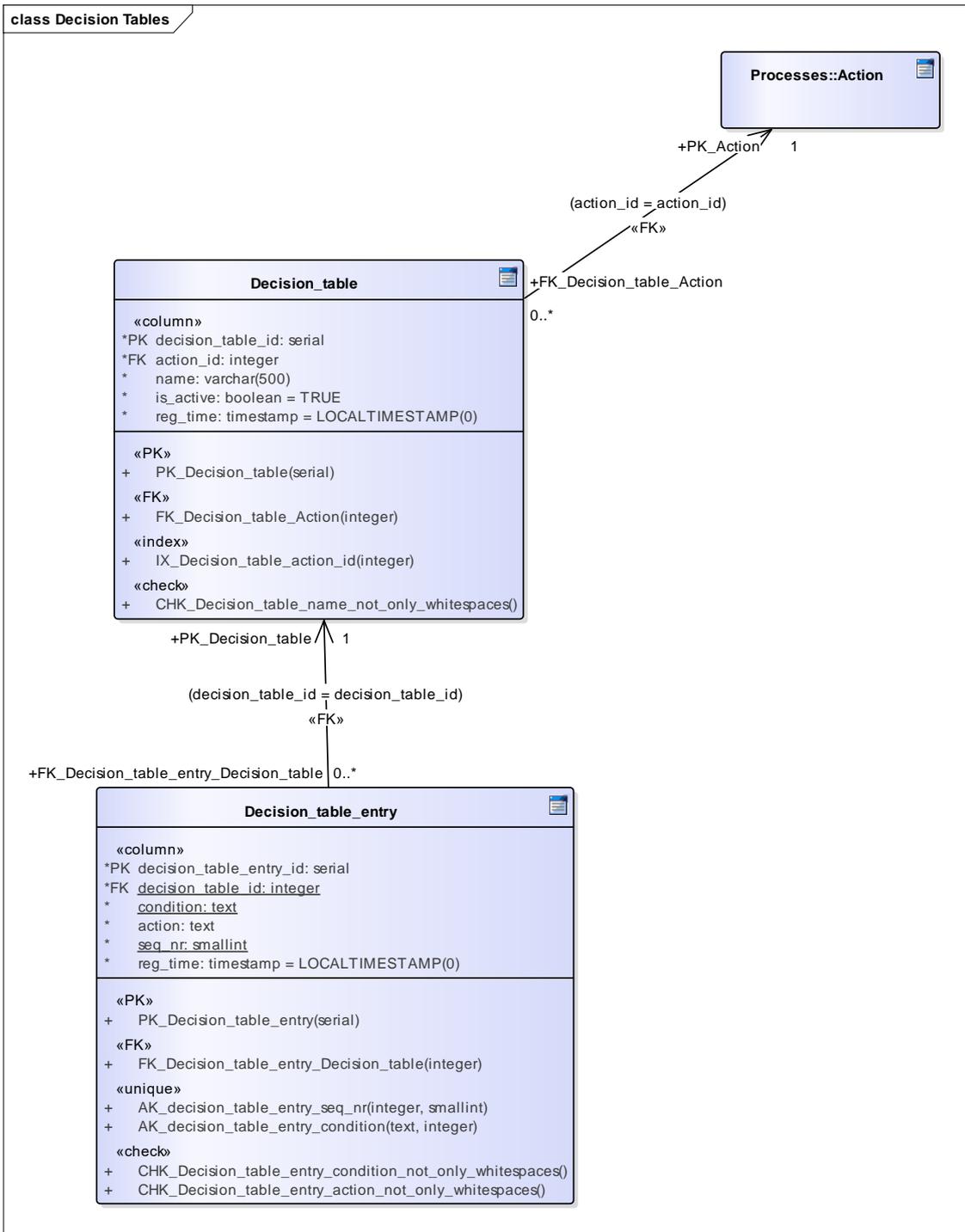


Figure 18. Physical design of the register of decision tables.

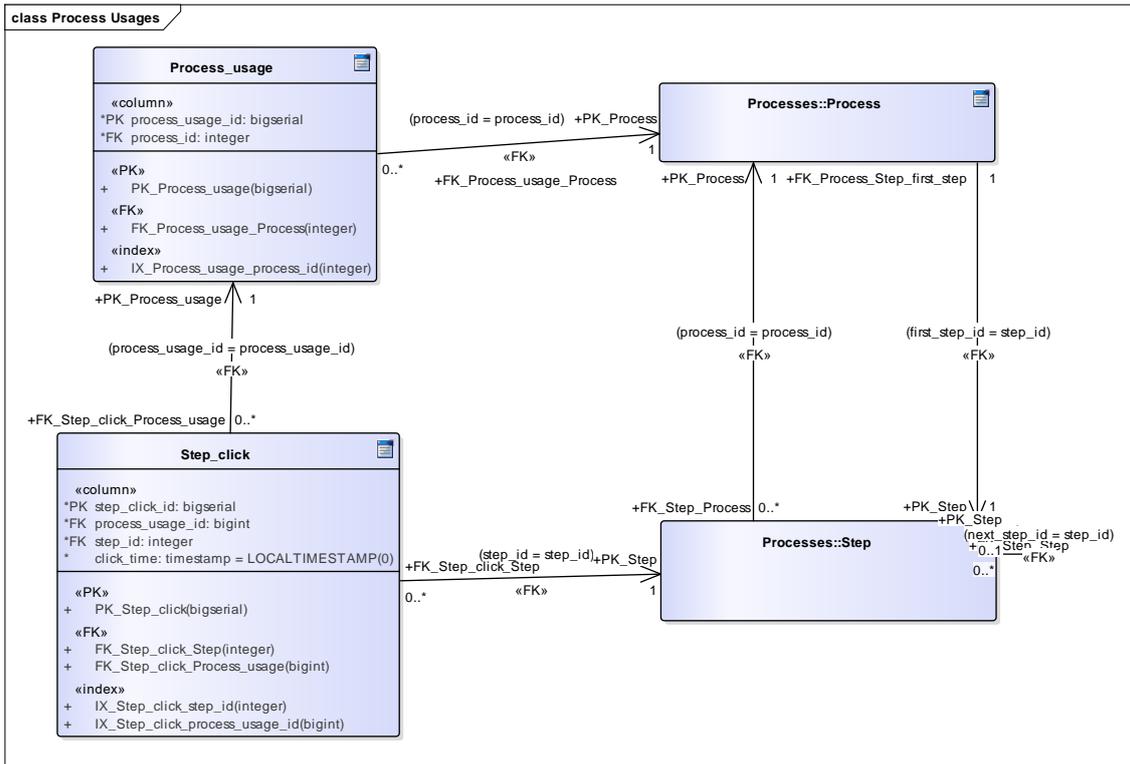


Figure 19. Physical design of the register of process usages.

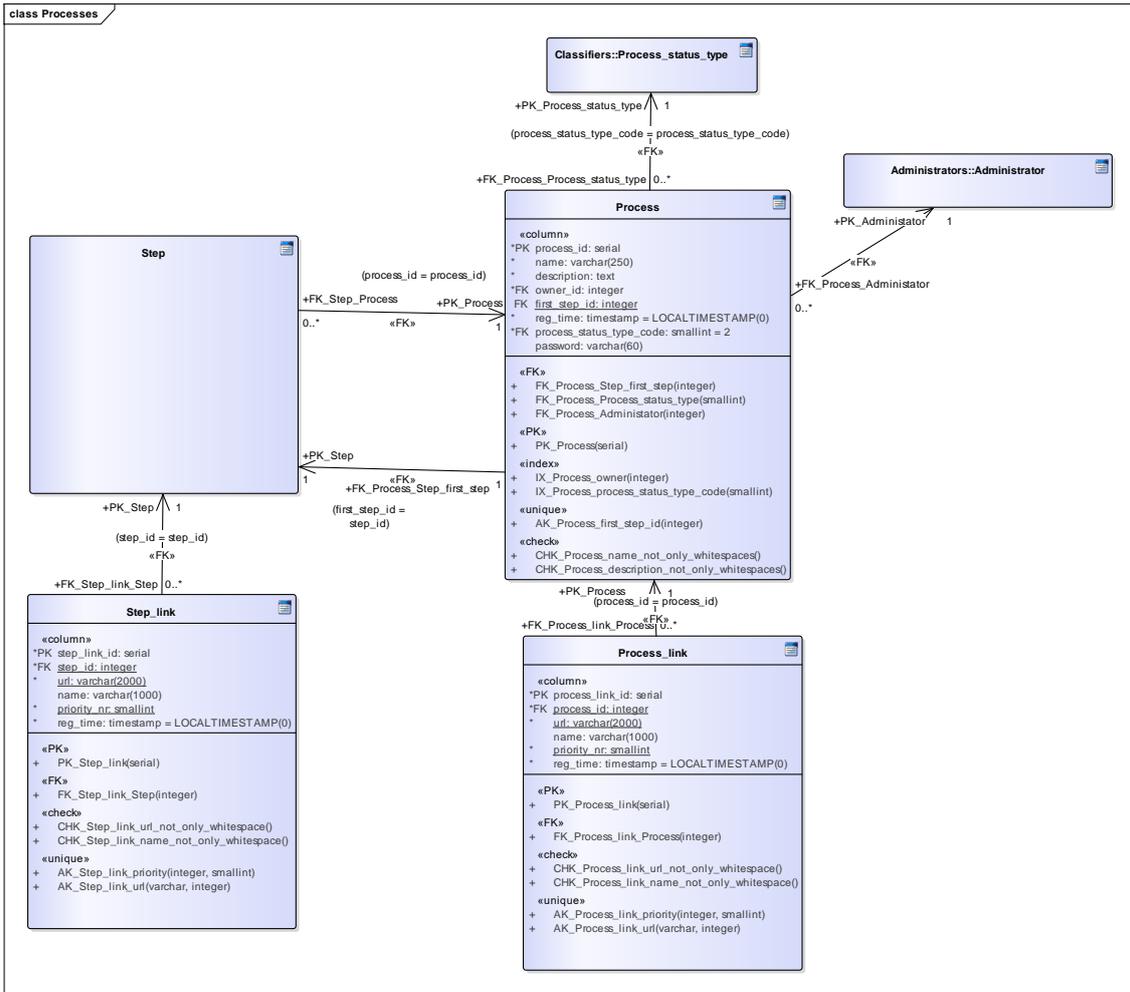


Figure 20. Physical design of the register of processes.

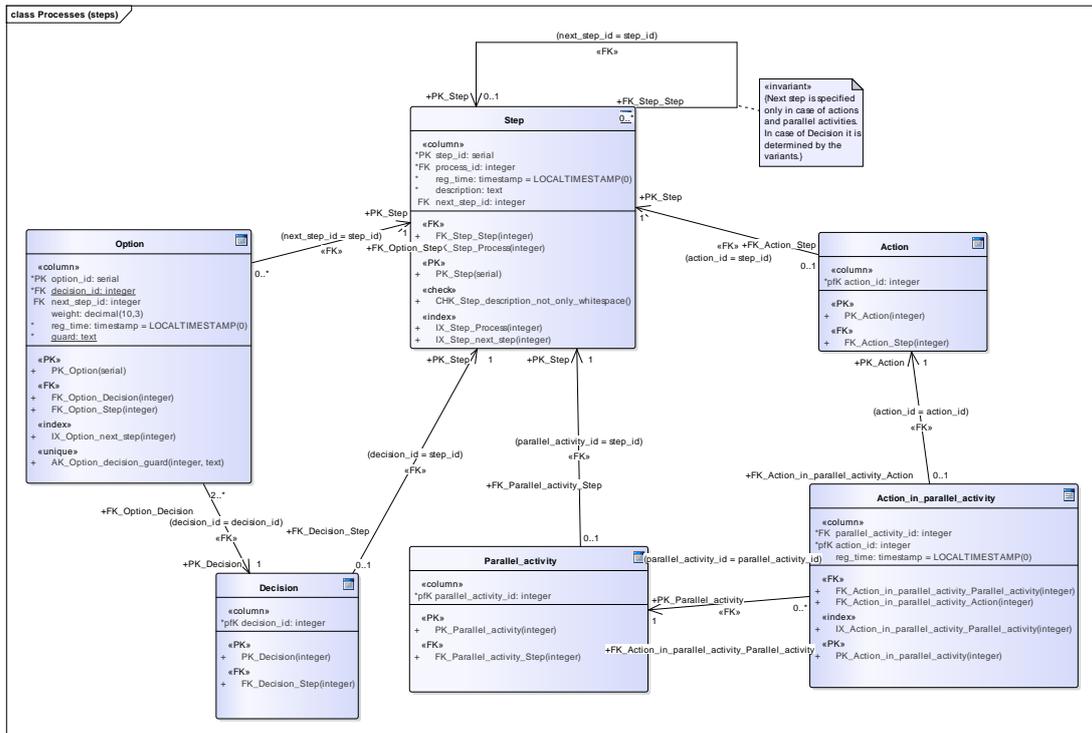


Figure 21. Physical design of the register of processes (steps).

## Appendix 4 – Code Examples

```
BEGIN;
DO $$
    DECLARE
        administrator integer;
        process integer;
        ettevalmistus integer;
        kas_saan_kohale_tulla integer;
        saan_tulla integer;
        ei_saa_tulla integer;
        tyhista integer;
        ole_kohal integer;
        kohtumine integer;
        kuidas_lood_on integer;
        on_pisivead integer;
        on_korras integer;
        on_suured_puudused integer;
        maara_kordaja integer;
        kordaja_tabel integer;
        parandamine integer;
    BEGIN
        SELECT processes.f_register_administrator('admin@test.ee',
'12345678', 'Admin', NULL) INTO administrator;
        SELECT processes.f_register_process('Iseseisva töö esitamise
ja hindamise protsess', 'Admebaaside II projekt.', administrator,
NULL) INTO process;
        SELECT processes.f_add_first_parallel_activity(process,
'Ettevalmistus') INTO ettevalmistus;
        PERFORM processes.f_add_action_in_parallel_activity(process,
ettevalmistus, 'Registreeri ettenäitamisele');
        PERFORM processes.f_add_action_in_parallel_activity(process,
ettevalmistus, 'Lae failid Maurusesse');
        SELECT processes.f_add_decision_to_step(process,
ettevalmistus, 'Kas saan kohale tuua?') INTO kas_saan_kohale_tulla;
        SELECT
processes.f_add_option_to_decision(kas_saan_kohale_tulla, NULL, 'Ei JA
ettenäitamiseni rohkem kui 24 tundi') INTO ei_saa_tulla;
        SELECT
processes.f_add_option_to_decision(kas_saan_kohale_tulla, NULL, 'Jah
VÕI (ei ja ettenäitamiseni vähem kui 24 tundi)') INTO saan_tulla;
        SELECT processes.f_add_action_to_option(process, ei_saa_tulla,
'Tühista registreerimine') INTO tyhista;
        PERFORM processes.f_add_action_to_step_existing_next(process,
tyhista, kas_saan_kohale_tulla, 'Registreeri ettenäitamisele');
        SELECT processes.f_add_action_to_option(process, saan_tulla,
'Ole kohal (vähemalt üks autor)') INTO ole_kohal;
```

```

        SELECT processes.f_add_parallel_activity_to_step(process,
ole_kohal, 'Kohtumine') INTO kohtumine;
        PERFORM processes.f_add_action_in_parallel_activity(process,
kohtumine, 'Vaata iseseisev töö koos õppejõuga üle');
        PERFORM processes.f_add_action_in_parallel_activity(process,
kohtumine, 'Tee märkmeid');
        SELECT processes.f_add_decision_to_step(process, kohtumine,
'Kuidas iseseisva tööga lood on?') INTO kuidas_lood_on;
        SELECT processes.f_add_option_to_decision(kuidas_lood_on,
NULL, 'Iseseisvas töös on pisivead JA õppejõud annab loa kohapeal
parandamiseks') INTO on_pisivead;
        SELECT processes.f_add_option_to_decision(kuidas_lood_on,
NULL, 'Iseseisev töö on korras') INTO on_korras;
        PERFORM
processes.f_add_action_to_option_existing_next(process, on_pisivead,
ole_kohal, 'Paranda iseseisvat tööd kohapeal');
        SELECT processes.f_add_action_to_option(process, on_korras,
'Õppejõud: määra projekti kordaja, mis sõltub arvestuse saamise
ajast') INTO maara_kordaja;
        SELECT processes.f_add_decision_table(maara_kordaja,
'Kordajad') INTO kordaja_tabel;
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'Õppenädal, mil arvestati', 'Projekti kordaja', 1::smallint);
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'1-13', '1.3', 2::smallint);
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'14-15', '1.2', 3::smallint);
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'16', '1.1', 4::smallint);
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'17-18', '1.0', 5::smallint);
        PERFORM processes.f_add_decision_table_entry(kordaja_tabel,
'19-20', '0.9', 6::smallint);
        PERFORM processes.f_add_action_to_step(process, maara_kordaja,
'Iseseisev töö on arvestatud');
        SELECT processes.f_add_option_to_decision(kuidas_lood_on,
NULL, 'Iseseisvas töös on suured puudused VÕI (iseseisvas töös on
pisivead ja õppejõud ei anna kohapeal parandamiseks luba)') INTO
on_suured_puudused;
        SELECT
processes.f_add_parallel_activity_to_option_existing_next(process,
on_suured_puudused, kas_saan_kohale_tulla, 'Parandamine') INTO
parandamine;
        PERFORM processes.f_add_action_in_parallel_activity(process,
parandamine, 'Registreeri ettenäitamisele');
        PERFORM processes.f_add_action_in_parallel_activity(process,
parandamine, 'Paranda iseseisvat tööd');
        PERFORM processes.f_add_action_in_parallel_activity(process,
parandamine, 'Lae failid Maurusesse');
        PERFORM processes.f_activate_process(process);
    END
$$;
COMMIT;

```

Figure 22. An anonymous function that can be used to insert the example process.

```

CREATE TABLE processes.Process
(
    process_id          serial          NOT NULL,
    name                varchar(250) NOT NULL,
    description         text           NOT NULL,
    owner_id           integer         NOT NULL,
    first_step_id      integer         NULL,
    reg_time            processes.d_time,
    process_status_type_code smallint   NOT NULL DEFAULT 1,
    password            varchar(60)    NULL,
    CONSTRAINT PK_Process PRIMARY KEY (process_id),
    CONSTRAINT AK_Process_first_step_id UNIQUE (first_step_id),
    CONSTRAINT CHK_Process_name_not_only_whitespace CHECK (name !~
'^[[:space:]]*$'),
    CONSTRAINT CHK_Process_description_not_only_whitespace CHECK
(description !~ '^[[:space:]]*$'),
    CONSTRAINT CHK_Process_password_not_only_whitespace CHECK
(password !~ '^[[:space:]]*$'),
    CONSTRAINT FK_Process_Process_status_type FOREIGN KEY
(process_status_type_code) REFERENCES processes.Process_status_type
(process_status_type_code) ON DELETE NO ACTION ON UPDATE CASCADE,
    CONSTRAINT FK_Process_Administrator FOREIGN KEY (owner_id)
REFERENCES processes.Administrator (administrator_id) ON DELETE NO
ACTION ON UPDATE NO ACTION
) WITH (FILLFACTOR = 90);
CREATE INDEX IX_Process_owner ON processes.Process (owner_id ASC);
CREATE INDEX IX_Process_process_status_type_code ON processes.Process
(process_status_type_code ASC);

CREATE TABLE processes.Step
(
    step_id          serial NOT NULL,
    process_id      integer NOT NULL,
    reg_time        processes.d_time,
    description     text NOT NULL,
    next_step_id   integer NULL,
    CONSTRAINT PK_Step PRIMARY KEY (step_id),
    CONSTRAINT CHK_Step_description_not_only_whitespace CHECK
(description !~ '^[[:space:]]*$'),
    CONSTRAINT CHK_Step_next_step_not_itself CHECK (next_step_id <>
step_id),
    CONSTRAINT FK_Step_Step FOREIGN KEY (next_step_id) REFERENCES
processes.Step (step_id) ON DELETE SET NULL ON UPDATE NO ACTION,
    CONSTRAINT FK_Step_Process FOREIGN KEY (process_id) REFERENCES
processes.Process (process_id) ON DELETE NO ACTION ON UPDATE NO ACTION
) WITH (FILLFACTOR = 90);
CREATE INDEX IX_Step_Process ON processes.Step (process_id ASC);
CREATE INDEX IX_Step_next_step ON processes.Step (next_step_id ASC);

/* Add First_step FK to Process */
ALTER TABLE processes.Process

```

```

ADD CONSTRAINT FK_Process_Step_first_step FOREIGN KEY
(first_step_id) REFERENCES processes.Step (step_id) ON DELETE SET NULL
ON UPDATE NO ACTION;

```

Figure 23. Two main tables – Process and Step – that depend on each other.

```

CREATE DOMAIN processes.d_time AS
timestamp NOT NULL DEFAULT LOCALTIMESTAMP(0) CONSTRAINT
CHK_d_time_from_2020_to_2200 CHECK (VALUE >= '2020-01-01' AND VALUE <
'2201-01-01');

```

Figure 24. One of the two domains in the database, used for registration time.

```

CREATE OR REPLACE FUNCTION processes.f_add_first_action(
p_process_id processes.Step.process_id%TYPE,
p_description processes.Step.description%TYPE)
RETURNS processes.Step.step_id%TYPE AS $$
WITH add_step AS (INSERT INTO processes.Step (process_id, description)
VALUES (p_process_id, p_description) RETURNING step_id),
add_action AS (INSERT INTO processes.Action (action_id) SELECT
step_id FROM add_step),
add_first_step
AS (UPDATE processes.Process SET first_step_id = (SELECT
step_id FROM add_step) WHERE process_id = p_process_id)
SELECT step_id
FROM add_step;
$$ LANGUAGE sql SECURITY DEFINER
SET search_path = processes, public, pg_temp;

```

Figure 25. An example of a function: function used to add the first step (Action here) to a process.

```

CREATE OR REPLACE FUNCTION processes.f_change_process_status() RETURNS
trigger AS
$$
BEGIN
    RAISE EXCEPTION 'Allowed status transitions are: "On hold" =>
"Active", "Active" => "Inactive", "Inactive" => "Active", "Active" =>
"Ended", "Inactive" => "Ended"';
END;
$$ LANGUAGE plpgsql SECURITY DEFINER
    SET search_path = processes, public, pg_temp;

CREATE TRIGGER trig_change_process_status
    BEFORE UPDATE OF process_status_type_code
    ON processes.Process
    FOR EACH ROW
    WHEN (NOT ((OLD.process_status_type_code =
NEW.process_status_type_code) OR
              (OLD.process_status_type_code = 1 AND
NEW.process_status_type_code = 2) OR
              (OLD.process_status_type_code = 2 AND
NEW.process_status_type_code = 3) OR
              (OLD.process_status_type_code = 3 AND
NEW.process_status_type_code = 2) OR
              (OLD.process_status_type_code IN (2, 3) AND
NEW.process_status_type_code = 4)))
    EXECUTE FUNCTION processes.f_change_process_status();

```

Figure 26. An example of a trigger: trigger used to prevent invalid status changes.

```

CREATE OR REPLACE VIEW processes.process_steps WITH (security_barrier) AS
SELECT step_id,
       CASE WHEN decision_id IS NULL THEN FALSE ELSE TRUE END      AS
is_decision,
       CASE WHEN parallel_activity_id IS NULL THEN FALSE ELSE TRUE END AS
is_parallel_activity,
       process_id,
       description                                                AS
step_description,
       next_step_id
FROM processes.Step
     LEFT JOIN processes.Decision ON step_id = decision_id
     LEFT JOIN processes.Parallel_activity ON step_id =
parallel_activity_id;

```

Figure 27. An example of a view: basic information about steps.

## Appendix 5 – Example Diagram

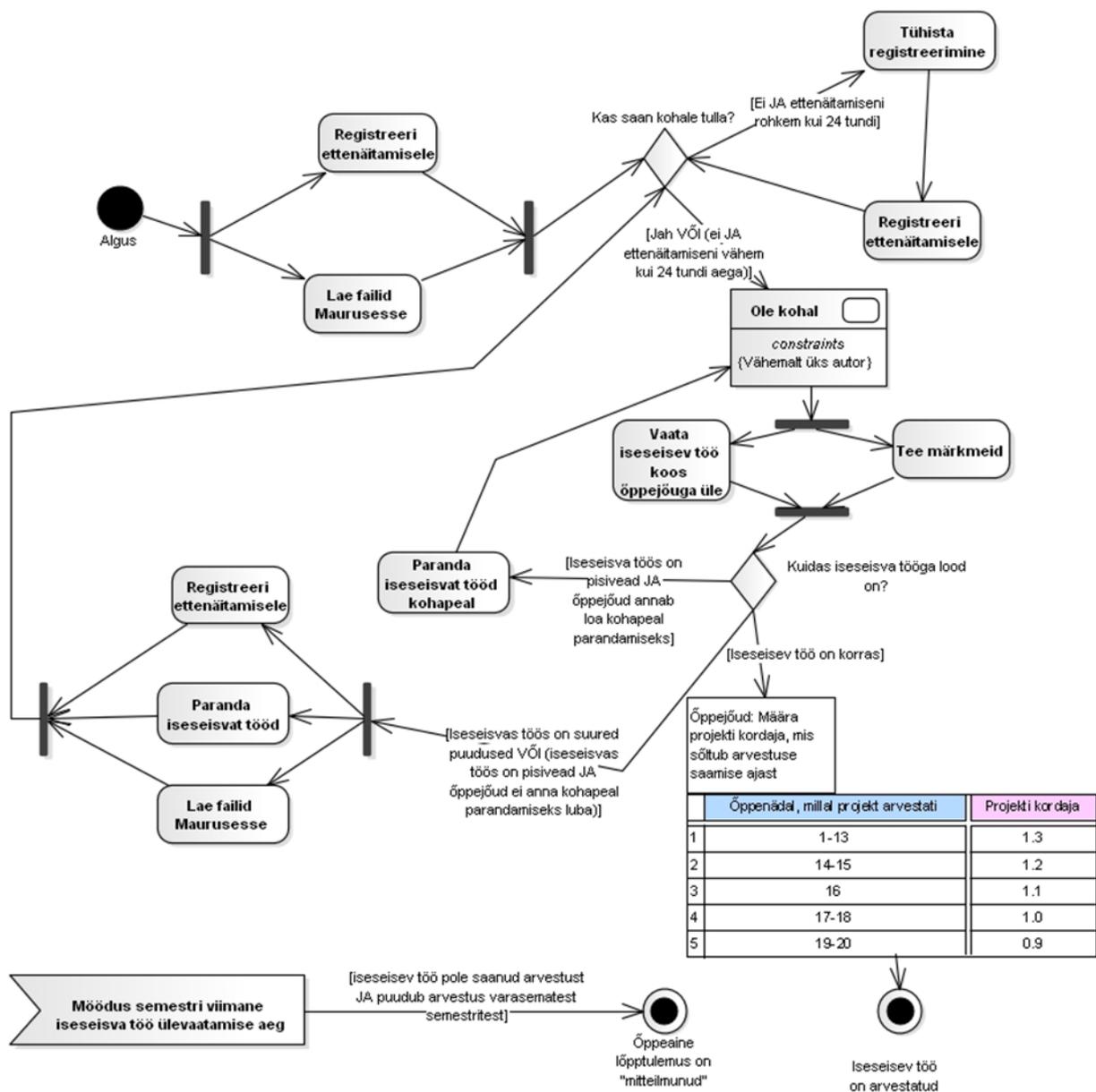


Figure 28. Process of submitting and grading the independent work in the course Databases I (in 2020).