TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Online aplikace pro hodnocení ekologických dopadů

Bakalářská práce

Studijní program:	B2646 – Informační technologie
Studijní obor:	1802R007 – Informační technologie
Autor práce:	**Ion Ciubaciuc**
Vedoucí práce:	Mgr. Kamil Nešetřil, Ph.D.

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Online application for ecological impact assessment

Bachelor thesis

Study programme:	B2646 – Information Technology
Study branch:	1802R007 – Information Technology
Author:	**Ion Ciubaciuc**
Supervisor:	Mgr. Kamil Nešetřil, Ph.D.

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ion Ciubaciuc**
Osobní číslo:	**M14000255**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Online aplikace pro hodnocení ekologických dopadů**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Z á s a d y   p r o   v y p r a c o v á n í :

1. Na základě podkladů a konzultací se zadavatelem navrhněte online aplikaci pro hodnocení ekologických dopadů obsahující formulář pro sběr dat, grafické znázornění materiálových toků, vyhodnocení klíčových indikátorů a reportování.
2. Seznamte se s frameworkem Laravel.
3. Navrženou aplikaci implementujte.
4. Sepište práci v angličtině.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**30–40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] KROENKE, David a David J. AUER, 2015. Databáze. Přel. Jakub GONER. Brno: Computer Press. ISBN 978-80-251-4352-0.

[2] LAVIN, Peter, 2009. PHP – objektově orientované: koncepty, techniky a kód. Přel. Michal POSPÍŠEK. Praha: Grada. ISBN 978-80-247-2137-8.

[3] SINHA, Sanjib, 2016. Beginning Laravel: A beginner's guide to application development with Laravel 5.3. 1st ed. New York, NY: Apress. ISBN 978-1-4842-2537-0.

[4] WIEGERS, Karl Eugene, 2008. Požadavky na software: Od zadání k architektuře aplikace. Přel. Tomáš ZNAMENÁČEK. Brno: Computer Press. ISBN 978-80-251-1877-1.

[5] ŽÁRA, Ondřej, 2015. JavaScript: programátorské techniky a webové technologie. Brno: Computer Press. ISBN 978-80-251-4573-9.

Vedoucí bakalářské práce:	**Mgr. Kamil Nešetřil, Ph.D.**
	Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce:	**10. října 2017**
Termín odevzdání bakalářské práce:	**14. května 2018**

[illegible]

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

Technická univerzita v Liberci
- 14 -

Kolář

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14.5.2018

Podpis: [illegible]

Acknowledgement

Here I would like to thank my supervisor Mgr. Kamil Nešetřil, Ph.D. for the advises and the time he devoted me during the work on my bachelor thesis. Also, I would like to thank my advisor Anna Kunath for consultations and the useful information provided each time it was needed. Last but not least, I would like to thank my family for the offered support, especially to my sister Diana.

Abstrakt

Táto bakalářská práce popisuje návrh a vývoj webové aplikace která má za cíl podnikům poskytnout pomoc při zavádění systematického managementu biodiverzity a zpřístupnit tuto tématiku i menším firmám. Aplikace je napsána s využitím programovacího jazyka PHP a frameworkem Laravel. Výsledná aplikace poskytuje uživatelům formulář pro sběr dat o jejích podniku a následné jím dovoluje zobrazit materiálové toky a sledovat vývoj klíčových indikátorů pomocí reportování.

Klíčová slova:

PHP, Laravel, environmentální management

Abstract

This Bachelor thesis describes the process of designing and developing a web application which aims to introduce small and medium sized companies to the subject of environmental management and helps them implement a systematic management of biodiversity. The application is written in the PHP programming language and is powered by the Laravel framework. The final application provides its future users with a questionnaire for collecting data about the organization they work for while it allows them to visualize the flow of materials and the evolution of key indicators.

Key words:

PHP, Laravel, environmental management

Table of Contents

List of images

Shortcuts list

EM	– environmental management
ISO	– International Standardization Organization
EMAS	– Eco-Management and Audit Scheme
GRI	– Global Reporting Initiative
EHS	–Environmental health and safety
MVC	– model-view-controller
PHP	– server-side scripting language designed for web development

1. Introduction

Today, through their activities, the organizations almost always have an impact on the environment. Because generally this is a negative one, in the corporate sector, there is an increasing interest for the environmental measures. The first reason, why this is happening, is because of the raising awareness among the general public, thus and consumers, about the ways of protecting the environment. This social consciousness is then reflected in the behavior of the customers. They prefer an organic production or boycott the businesses that harm the environment [7]. A second reason can be the increasing prices of energy, raw materials and waste disposal or the fact that larger organizations are exerting pressure on their suppliers to introduce environmental tools.

The main target of this work is the implementation of an application that can be used as a tool for environmental management. Based on the analysis of the documents and consultation with my advisor is required to design the database, the graphical user interface and the components that the application needs to have. Also, another target of this work was to familiarize myself with the Laravel framework and use it for the implementation of the application.

I selected this topic because it was possible to be realized using the Laravel framework and the PHP programming language. I deal with the development of web applications, so this topic is close to me and at the same time it offers me the possibility to gain new experience in this field which will be useful for me in the future. Other than this, the protection of the key elements of our environment is very important for our health. I consider that everyone should have the ability to breathe clean air, drink clean water and be protected from harmful effects of waste. In order for this to be possible, everyone should do everything they can to protect the environment.

2. Environmental Management

The following paragraph will first introduce the term of environmental management (EM), then it will analyze how EM is implemented in the Czech Republic and what tools are available on the market that address this problem.

2.1.Definition of the environmental management

EM is not an easy term to define. As J. Barrow has acknowledged [1], it can refer to a goal or vision, to attempts to steer a process, to the application of a set of tools, to a philosophical exercise seeking to establish new perspectives towards the environment and human societies, and to much more besides that. However, EM is mainly concerned with the ways in which humans relate to their environment, as well with the understanding of the structure and functioning of the Earth system. Environmental management is, therefore, concerned with the description and monitoring of environmental changes, the prediction of future changes and the attempts to maximize human benefit and to minimize environmental degradation due to human activities.

EM is a broad and rapidly expanding field, crucial to the well-being of humans and the maintenance of environmental quality. The human impact on the environment is causing widespread concern, as a consequence the successful management of the Earth's biodiversity is rendered vital. The term biodiversity refers to the variety of plant and animal life in the world or in a particular natural habitat, a high level of which is usually considered to be important and desirable [2].

The biodiversity is the basis for the intact ecosystem services that bring benefits to companies and are often freely used [3]. The ecosystem services are categorized into four different groups. These are the following: supply services which are food, water, raw materials and regulatory services meaning the regulation of climate or flood, cultural services including landscape aesthetics and inspiration, and a number of support services such as nutrient circulation or soil processes [4]. For companies these services are important from two aspects: on one hand, the companies are dependent on them, given that the services provide important factors for production that is raw materials of a certain quantity and quality, while on the other hand, through their activity the companies impact on these services and on biodiversity, and can influence the latter in a negative way due to overuse or pollution [5]. The systematic business management of biodiversity helps companies reduce risks and negative impacts on biodiversity through corporate practices, and at the same time offers them entrepreneurial opportunities [6]. The loss of biodiversity can mean a drop in raw material supply, a reduction in quality and/or quantity, and also might bring on price risks [5]. Companies that are aware of their impact on biodiversity, the need of its existence and which systematically take into account these aspects in their decision-making processes, can only benefit from this approach. Therefore, the best way of becoming aware of how human activity impacts on biodiversity is by implementing an environmental management system.

2.2. Environmental management in the Czech Republic

In the Czech Republic the EM is mainly based upon the ISO 14001 and EMAS (Eco-Management and Audit Scheme) standards. In more detail, ISO 14001 is published by the International Standardization Organization and its current version is that of 2015. This standard has a global character and is applicable on industries and organizations of varying sizes [7]. In contrast, the EMAS is a European Commission regulation and has only a European outreach. Like ISO 14001, it can be applied on all sectors and businesses, regardless of their size [8]. Both standards aim to build a systematic corporate EM, however, the chosen procedure, methodology and terminology are different. In general, the claims of EMAS are higher than the ISO 14001 requirements [9]. Biodiversity, in these standards, plays only a small role and the involvement of this aspect in EM is left to companies [5]. EMAS recommends one key indicator of biodiversity, which is land use (more precisely, use of land, expressed in m^2 of built-up area) [10]. ISO 14001, however, does not specify any mandatory indicators. The topic of incorporating aspects of biodiversity into EM is addressed in several handbooks and recommendations, but there is no standard comparable to ISO 14001 or EMAS, which would deal with biodiversity management in companies.

The situation is similar on the field of applications and corporate software for diversity management. Nowadays more products for EM, such as ecological balance preparation or product life cycle assessment are available on the market. Biodiversity is often part of the tool but rarely forms a separate product. There is also a need to distinguish applications by the target group. Therefore, the subject of research is often the applications and models for biodiversity assessment and planning of various measures for administrative authorities or environmental research. These applications often focus on data evaluation or modeling of the interaction of various factors on biodiversity. The developed application that will be described in the next chapters, serves mainly corporate management hence, the target groups are the companies.

2.3.Current applications on the market

Below is presented a list of applications for corporate EM that nowadays are available on the market.

360report

360report (www.360report.org) is a web-based sustainability software that complies with sustainability standards such as GRI, ISO 26000 and United Nations Global Compact. The entered data are used for automatic generation of a GRI-certifiable Sustainability Report (Word document or PDF file). In addition, a CO_2 balance is created in accordance with Greenhouse Gas Protocol.

Enablon EHS-Management

Enablon EHS (Environmental health and safety) software helps companies to meet their EHS management and compliance challenges, including environmental analysis and reporting, management of air, water, waste and chemicals, regulatory compliance, worker health and safety, incidents prevention etc. [16].

EPM-KOMPAS

Developed by Technical University of Dresden in cooperation with a circle of industrial partners from Saxony, the software can be used as an entry aid or to develop an EM system in medium-sized companies. The following functions are supported by the software:

- Handling of hazardous substances, waste, emissions
- Creation of material and energy flows (balance sheets)
- Defining environmental goals
- Assessment of environmental measures

• Generation of reports (for authorities)

• Researching conspicuous material and energy flows

• Control of results/successes

The problem is that in these applications biodiversity plays only a minor role, and that generally applications focus mostly on aspects such as waste, energy consumption or reporting.

In the Czech Republic, EM is a voluntary and not a very common business process. Many companies have not yet recognized the benefit of this approach, and only large, multinational corporations with adequate backgrounds often deal with this issue [11]. However, this subject is also important for small and medium-sized enterprises. For them, the topic is often very complex in the sense that they lack information sources and staff. The online application for ecological assessment is designed particularly for them. The software aims to help users implement systematic biodiversity management and make it accessible to smaller businesses. Through the application the users will get necessary information about how they interact with the biodiversity and they will be able to formulate their business strategy with respect to these aspects. Also, the application will allow them, using indicators, to determine and evaluate the impact of business activities and outline possible measures.

3. Work aim

The aim of this work is to prepare the schema of an online application for ecological impact assessment and implement it using the PHP framework Laravel (www.laravel.com). The application has to contain the following components:

The questionnaire

The first step into analyzing a company's business impact on biodiversity, and the most important, without which all the other steps would not work or make sense, is collecting information about the company's strategy and philosophy, processes, and flow of materials.

There are two types of data that the companies will fill in the questionnaire. The first type, such as the management certificates of the organization or the features that characterize the company's business philosophy and products, is required only once and changes only from time to time. As for the second type of data, this should be filled in once per year, and it includes the input materials and the quantity that has been used, electricity consumption and cost or how were the output products distributed last year or even at an earlier time.

The questionnaire should be divided in the next sections:

- General information about the company
- Management
- Input materials
- Output products
- Marketing strategy
- Company headquarters/branches
- Employees
- Finances

Most of the questions offer multiple choice answers, but given the fact that the project is in an early development stage, it can often happen that some answer options will be missing, so the application should offer the user the possibility to fill in manually his answers. This should help avoiding the loss of important information about the organization and will help in the future to extend the list of choices when answering a certain question.

Flow of materials

In first version of this application, the materials flow component should be able to show a basic graphical representation of the company's flow of materials for a selected year. Data about the packaging and auxiliary materials, also about the input materials, their quantity and supplier, the quantity of water and electricity used, by-products, their quantity and how they are handled, the product lines, as well the final products and their quantity, will be presented to the user. Also here should be shown the information about the number of employees, customer program, transport means and traveled distance, the sectors to which the enterprise belongs, their activities and other information.

Key indicators assessment and reporting

This component should show an overview of the key indicators like:

- Electricity consumption
- Water consumption
- Electricity from renewable sources
- Number of trainings on environmental management
- Financial expenses
- Table of emissions

The first 3 indicators should be assessed by absolute quantity, quantity share for each employee and quantity share for the total surface size of the company. The number of training on environmental management should be assessed according to how many took place within the organization, also to the number of employees that took part in them. Financial expenses should be assessed by how much has been spent for each unit of consumed water, energy or produced and transported waste.

All these indicators should help the companies observe their impact on the environment. The data should be displayed in tables, and for an effortless interpretation, there should also be a graphical display on how the data evolved during the years. Given that the target market of this application is Germany, the Czech Republic and Poland, the application should be multilingual.

4. Solution design

Before starting to implement the application, it was required to choose a design pattern, to prepare a concept of how the graphical user interface (GUI) will look like, to analyze all the data I got from my adviser, and based on that, to prepare the database design.

4.1.Design pattern

Laravel is a PHP framework that is based on the model-view-controller (MVC) design pattern, so the choice here is obvious. The MVC is a software architecture that organizes code in an application in order to improve maintainability. It does this by separating the application into three parts: the model, the view, and the controller. In addition to this, it defines the interactions between them [12] (Image 1).

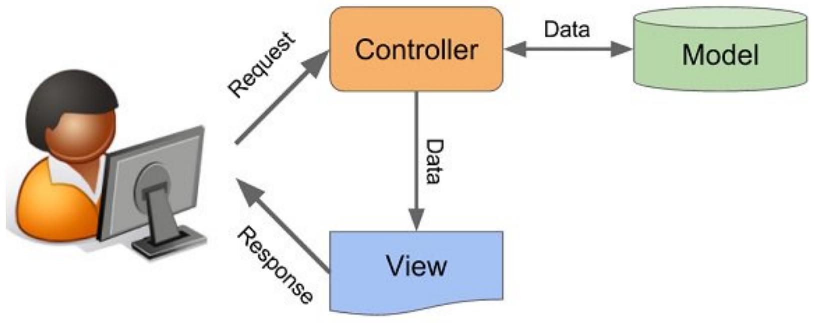


Image 1: MVC interactions (source: www.helloacm.com)

The model manages the data of the application. It receives user input from the controller. The view effectively provides the user interface element of the application. It will render data from the model into a form that is suitable for the user interface. Whereas the controller receives user input and makes calls to model objects and view to perform appropriate actions.

4.2.The user interface

The GUI is the part of a software that helps the user to interact with the rest of the application. GUI has to be simple, efficient and enjoyable to use. I divided the GUI of the application in the following components:

The Dashboard

This is the page where the user will get after signing in. The dashboard will give him the possibility to choose the section of the questionnaire he wants to fill in next, to see the flow of materials, the report of the assessed indicators or the emissions table. Also, here he will find information about what each component does.

The questionnaire

In Chapter 3 it is said that the questionnaire will have more sections and that some of the questions have to be answered once, while others every year. When designing the questionnaire's GUI, my thought was that the user should be able to see what data and for what years he already has in the database, without needing to navigate to the flow of materials or reporting. This is useful especially when a new user of the application wants to provide data for more years of activity, he started to fill in data, but he decided to continue during another day, either he wants to edit data or just wants to see it.

The first concept was that all the questions will be in the same form. This solution was easy to implement, but it had the disadvantage of crowding the form and also, because in the same form could have been more questions where is required the year, therefore this could confuse the user. The second idea was to divide the questions that require an answer every year from the others (see Image 2).

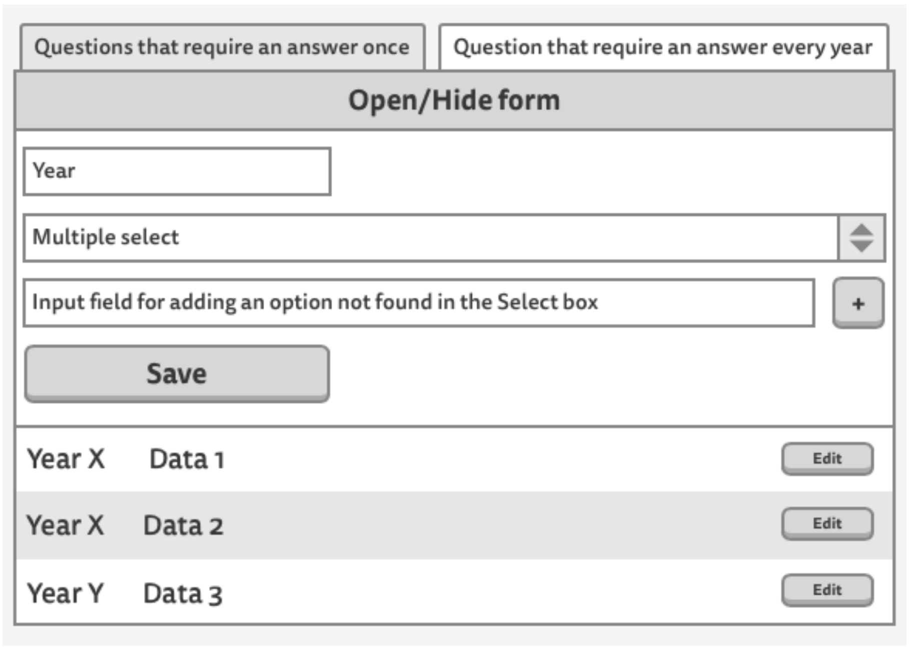

Questions that require an answer once
Question that require an answer every year
Open/Hide form
Year
Multiple select
Input field for adding an option not found in the Select box
+
Save
Year X Data 1 Edit
Year X Data 2 Edit
Year Y Data 3 Edit

Image 2: GUI concept for questionnaire form

"Open/Hide form" button will allow the user to show or hide the form for adding new data. Under the form, there will be the data from other years. The "Edit" button gives to the user the option to add additional information about an indicator if this is required, for this reason, a modal will appear. As for the multiple select, it will allow the user to select his answers and in case he can't find something in the provided options, he can use the input field to add his own option. Also, if he wants to add more options, he can press "+", which will generate a new input filed. The "Save" button will trigger the action that will store the data into the database.

Report

As it was specified in Chapter 3, each indicator will be provided with a table of values of how a given indicator has evolved over the years and also with a graphical overview for an easier interpretation. For this component I decided to have the values in the table displayed from the newest to the oldest, and for the graphical part the opposite (see Image 3).

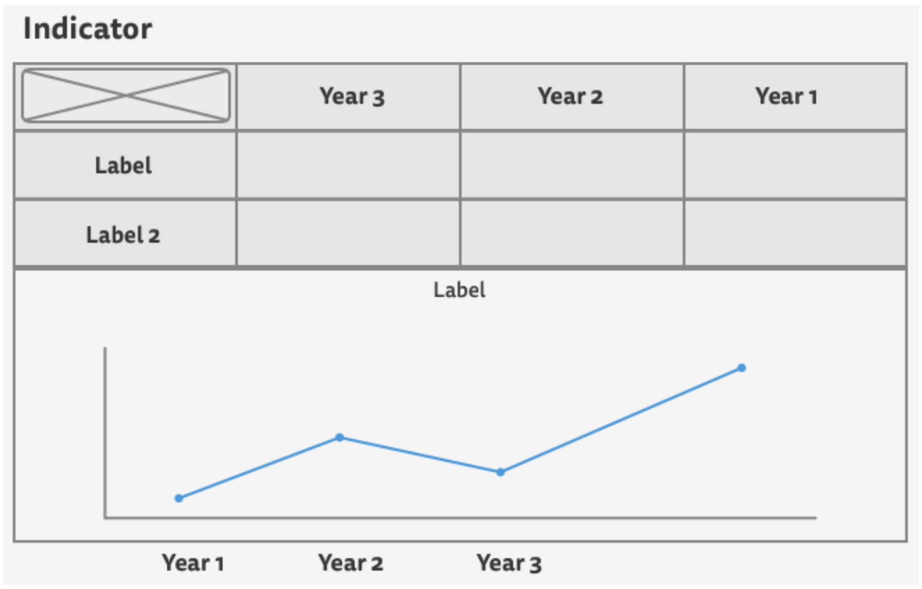


Image 3: GUI concept for indicator report

The labels can be, for example, the absolute quantity of the indicator or the quantity share for each employee. In the empty cells will be the values of a given label in a given year.

The flow of materials

For this component, I decided that each element in the Image 4 will be a table or a list where all the items for a given year (if applicable) will be shown.

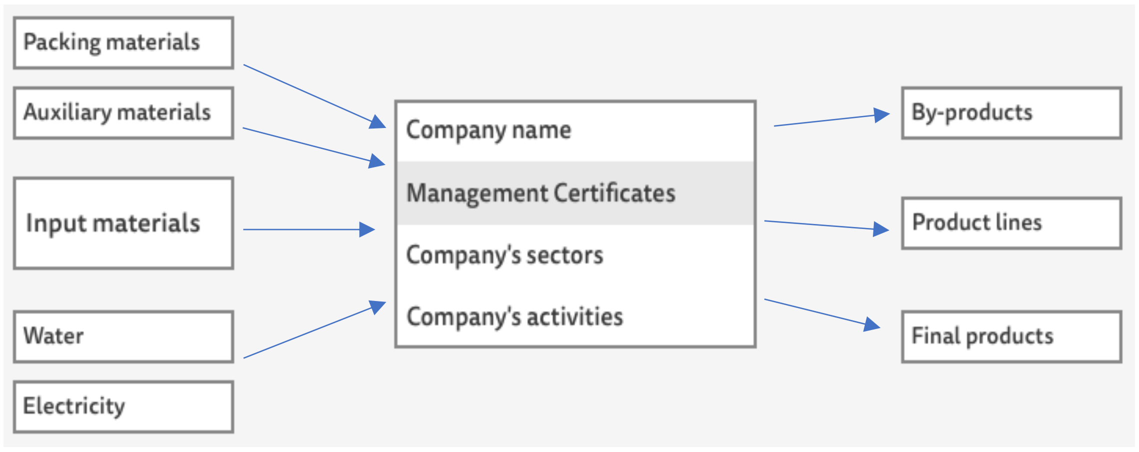


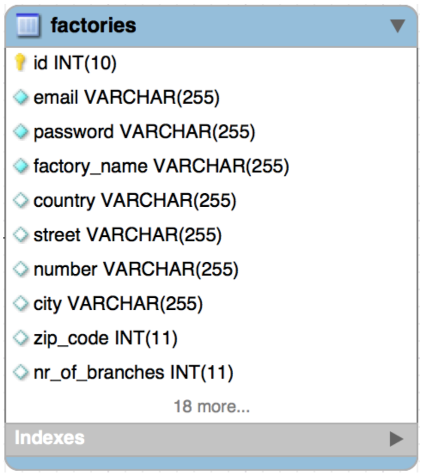
Image 4: GUI concept for the flow of materials

4.3.The database

During the consultations with my advisor, I got more Excel files containing all the questions that the questionnaire had to contain. Based on that, I had to design the database.

The application will allow the user to answer all the questions contained in the questionnaire's forms. When I started to analyze the data from the documents, I went from the idea that the company is a factory and that is why I named the main table in the database "factories".

Other than dividing the type of questions by how often they should be answered, they can be divided by the number of answers, which can be either one or multiple. Each of these, can be with or without an already provided list of possible answers. The questions that require one answer and only once, for example, the number of branches, will have their answer stored in the table "factories" (see Image 5).



factories
id INT(10)
email VARCHAR(255)
password VARCHAR(255)
factory_name VARCHAR(255)
country VARCHAR(255)
street VARCHAR(255)
number VARCHAR(255)
city VARCHAR(255)
zip_code INT(11)
nr_of_branches INT(11)
18 more...
Indexes

Image 5: Factories table design preview

The questions that require multiple answers will have the list of possible answers stored in the database. The company's sectors of activities as they are classified by the Statistical Classification of Economic Activities of the Czech Republic (CZ-NACE) can serve as an example. The table in which they will be stored is named "sectors". The relationship between "factories" and "sectors" will have the cardinality of many to many (M:N) and the data that describes the relationship will be stored in the table "factory_sector" (see Image 6).

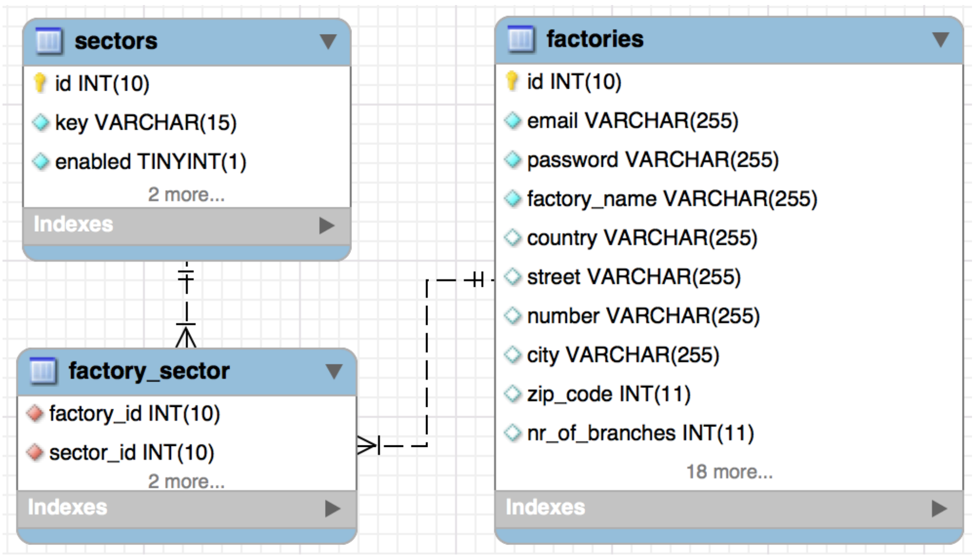


Image 6: Relationship between factories and sectors tables

The "key" column in the table "sectors" it's unique and will be used as a key to retrieve the translation of a given sector from the translations files. Keeping in the database only the keys of translations will allow, if later needed, the generation of multilingual flow of materials. To avoid duplicates, the columns "factory_id" and "sector_id" in table "factory_sector" will create a unique constraint.

One of the questions that requires multiple answers but doesn't provide the user with a list of possible answers is "What are the company's final products?". The relationship between "factories" and "final_products" will have the cardinality of one to many (1:N).

Another question like "What are the input materials that you use in your company?", combines both situations described above. For the already provided list of materials a relationship with the cardinality M:N will be used. However, for the materials that are not illustrated in the list, the user will be able to provide them himself. His input will be stored in the table “other_input_materials”, the relationship between the latter and the table “factories” will have the cardinality 1:N.

5. Application implementation

My task in this work was to implement an online application for ecological impact assessment. The application was written mostly using the PHP programming language, powered by the Laravel framework. Together with PHP was used jQuery programming language. The most actual version of Laravel at the beginning of the development was version 5.5, which requires PHP version 7.0+. The database server used was MySQL, distribution 5.7.21. During the development, the application ran on Apache web server, under the operational system macOS High Sierra. As integrated development environment was used PhpStorm. The design of the user interface uses the front-end framework Materialize (materializecss.com).

5.1. Laravel framework

Laravel is a web application framework with expressive, elegant syntax which attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching. Laravel aims to make the development process a pleasing one for the developer without sacrificing the application's functionality.

5.1.1. System Requirements

When installing Laravel framework, the following system requirements should be met [13]:

- PHP version 7.0.0 or higher
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

5.1.2. Laravel Installation

To install Laravel, it is required to have installed Composer. This is a dependency manager that the framework uses to manage its dependencies. First it is needed to download the Laravel installer using Composer. Running the following command in the terminal did the job:

```
composer global require "laravel/installer"
```

For this to run, it is needed to make sure that composer's system-wide vendor bin directory is placed in `$PATH` [13]. Once installed, I ran in the terminal the next command:

```
laravel new netsci
```

This created a fresh Laravel installation in my home directory, under the folder `netsci` and generated the application's key. The key is used to secure encrypted data and the user sessions.

To serve the application, I configured the Apache web server's document/web root to be `public` directory. The `index.php` in this directory serves as the front controller for all HTTP requests that enter the application.

5.1.3. Laravel Configuration

All of the configuration files for the Laravel framework are stored in the `config` directory. Because it is often useful to have different configuration for local and production environment, Laravel utilizes the `DotEnv` library. In project's directory is a file named `.env` that contains more variables. At the moment when I started do develop the application the following variables were the most important:

- `DB_CONNECTION=mysql`
- `DB_HOST=127.0.0.1`
- `DB_PORT=3306`
- `DB_DATABASE=netsci`
- `DB_USERNAME=netsciapp`
- `DB_PASSWORD=NETSCIapp`

These variables define the database server used by the application (MySQL), the IP address and the port where the database server is listening, the name of the database to use and the access credentials. Other database configurations as the `charset` or the `collation` can be set in `config/database.php`. Also here, we can see the use of the `DotEnv` library. If we take for example the database name:

```
'database' => env('DB_DATABASE', 'forge'),
```

the `env` function will set the value of the `database` parameter to be the variable `DB_DATABASE` from the `.env` file, however if the variable will not be found, the second parameter (`forge`) of the function will be used.

As it is already known from Chapter 3, the application should be multilingual. Laravel's default language and fallback language is English. The language configurations can be found in `config/app.php`, where I set them to be in Czech:

```
'locale' => 'cs',
'fallback_locale' => 'cs'.
```

The fallback language is used to return text translations when a translation for a given current language cannot be found.

5.1.4. Error handling

In Laravel, all exceptions are handled by the `App\Exceptions\Handler` class. The `report` method in this class logs the exception with the whole stack trace in to the `storage/logs/laravel.log` file. This is rather of no use. When you have many lines of stack trace and more exceptions, it is hard to find the actual error message in the log. To make things easier, I modified the method to log only the exception's code, the exception message, the file and the line where this occurred (`$e` is the exception):

```
Log::error('[' . $e->getCode() . '] "' . $e->getMessage() . '"
on line '. $e->getLine() . ' in file ' . $e->getFile());
```

5.1.5. Facades

In this thesis, the "facade" term is often mentioned. In Laravel, facades provide an interface to classes that are registered in the application's service container. Out of the box, Laravel is containing many facades that provide access to almost all of the frameworks features.

5.1.6. Artisan

Artisan is a command-line interface that is included into Laravel. The multiple commands that are provided by it, assisted me while I was building the application. Artisan commands should be run in the project's root folder like this:

```
php artisan "command_name".
```

5.1.1. Users authentication

Out of the box, Laravel comes with pre-built authentication controllers, which are located in `App\Http\Controllers\Auth` namespace. In many cases, there is no need to modify these controllers at all, but this was not the case with this application.

Laravel uses `User` model to retrieve data about the logged in user, which needs a "users" table to exist in the database. In this application, authentication data is stored in table "factories", as follow, the name "Factory" for the model will be more appropriate than "User". Now that a different table and model are used than in the standard Laravel, changes have been made to the `config/auth.php` file to use the table "factories" and the `Factory` model to log in users.

Having the configurations ready, the next step is to prepare the routes and the views for the authentication. Luckily, Laravel provides a quick way to scaffold them using one simple command:

```
php artisan make:auth
```

As agreed with the advisor, during the registration of a new factory, the user will provide the factory's name alongside with the email and the password. To make this possible, to the form contained by the view `register.blade.php` has been added a new input field "factory_name" and the `RegisterController` has been modified to validate and save the new added field.

Accessing the authenticated user can be done using the `Auth` facade. Alternatively, if the user is already authenticated, it can be accessed using an instance of `Illuminate\Http\Request`.

5.2. Building the database

5.2.1. Migrations

To design the database, was used MySQLWorkbench application, but for the actual implementation of the tables was used Laravel's `Migration` class. Migrations allow the developer to easily modify or share the application's database schema. To build the application's schema, `Migration` class uses the `Schema` facade, which provides database support for creating and manipulating tables. In the actual form, the database contains over 80 tables. All created migrations can be found in `database/migrations` folder.

To create a new migration for the "factories" table, I ran in the terminal the following command:

```
php artisan make:migration create_factories_table
```

where the result of this command was the following file:

```
2017_10_05_204736_create_factories_table.php
```

The file name contains the date and time when the migration was created. As a result, when the migrations will run, the tables will be created in chronological order, hence avoiding possible errors.

Migration `create_factories_table` contains the functions `up` and `down`. The `up` function contains the following code:

```
Schema::create('factories',    function    (Blueprint    $table){
  $table->increments('id');
  $table->string('email')->unique();
  $table->integer('year')->unsigned()->nullable();
  $table->text('key_processes')->nullable();
  $table->boolean('client_service')->nullable();
  $table->timestamps();…
});
```

After running the `up` function, the table "factories" will be created in the database. The method `increments` will create the private key `id`, which will auto increment when a new record will be added. The `string` method creates columns of data type `varchar` and if the second parameter of the function is not specified, which is the length, the columns will store strings up to 255 characters long. Chaining the `unique` method to the `email` column will create a unique index for it. As for the `boolean` method, it will create a column of data type `tinyint(1)`. Moreover, the `timestamps` method will create two additional columns of data type `timestamp`: `created_at` and `updated_at`. The first will store information about the date and time when the record was created, and the second – data about when the record was updated. Chaining the method `nullable` to the columns allows them to accept

NULL values. In addition, the integer method will create a column of data type int. Adding the unsigned method makes the column of type int unsigned.

The down function in the migration class contains the code for dropping the table (Schema::dropIfExists('factories');) which, when needed, will reverse the migration.

In many to many relations is required to have an intermediary table that will describe the relationship between two tables. In this case, foreign keys should be added to the table. If we take as example the "factory_sector" table creation, we see how easily this is done in the up function of the migration:

```
Schema::table('factory_sector', function (Blueprint $tb){
  $tb->foreign('factory_id')->references('id')
     ->on('factories');
  $tb->foreign('sector_id')->references('id')
     ->on('sectors');
});
```

5.2.2. Seeding

If you take a look at the Image 7 you can see that an activity can have one or more specifications. In the questionnaire, depending on what sectors the user will choose, to him will be shown a list of activities that are specific only for the selected sectors.

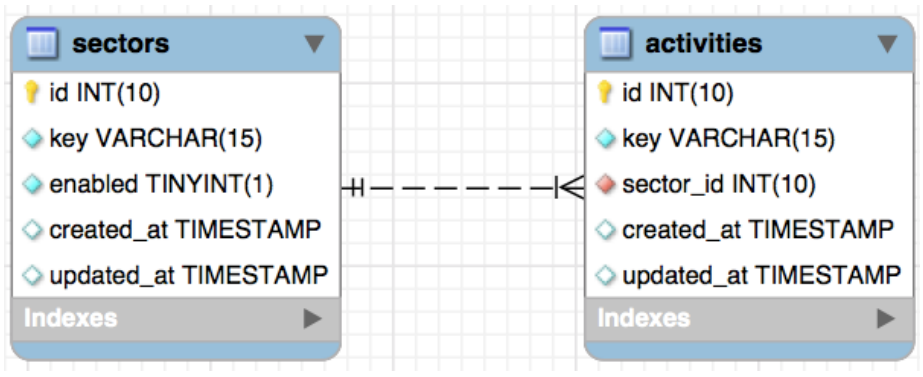


Image 7: Sectors and activities tables diagram

Laravel includes a simple method to seed the database with test data using seed classes. However, these classes can be used not only for test data. As my case can show, I used them to seed the real data that needed to be in the database for the user. Also, the seeder classes came in handy each time it was a big change to the database tables and it was needed to reinsert or change the data.

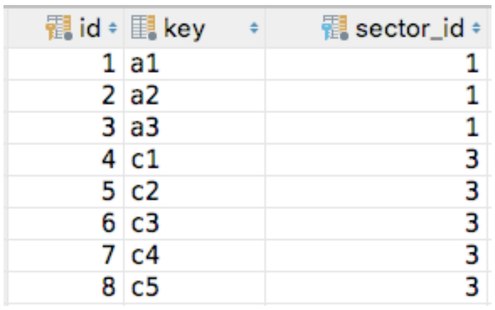
All the seeder classes can be found in `database/seeds` directory. In order to generate the seeder class for the "activities" table, the following command was used:

```
php artisan make:seeder ActivitiesTableSeeder
```

After creating the seeder class, the author added the data that should be saved into the database to the run function:

```
$items = [
['key' => 'a1', 'sector_id' => '1'],
    ['key' => 'a2', 'sector_id' => '1'],
    ['key' => 'c1', 'sector_id' => '3'], …
];
foreach ($items as $item)
    Activity::create($item);
```

When the `run` function of the `ActivitiesTableSeeder` class is later called, the `Activity` model inserts the data into the database (see Image 8).



id	key	sector_id
1	a1	1
2	a2	1
3	a3	1
4	c1	3
5	c2	3
6	c3	3
7	c4	3
8	c5	3

Image 8: Preview of activities table in the database

The seeders created by author have been later added to the `run` function of the `DatabaseSeeder` class using `call` function, which for "activities" table seeder is as follows:

```
$this->call(ActivitiesTableSeeder::class).
```

5.3.The request and response

The questionnaire has 8 sections. Each section has its own controller and view. The views and the controllers are named after the section they represent. The form of the section "General information" has been divided in 3 parts (Image 9). To navigate between different parts a tab like navigation system is used.

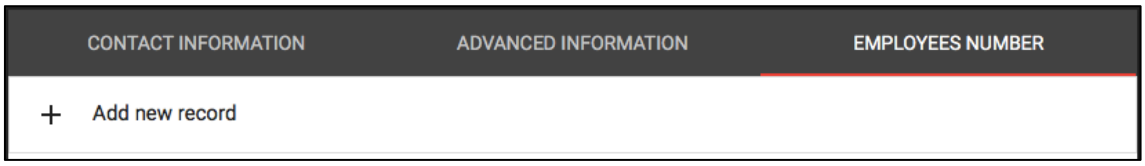


Image 9: General information view tab navigation

When sending pure HTTP requests, if there is an error, the whole page has to be reloaded and the navigation will jump back to the first tab. The user then will be constrained to select the tab that corresponds to the part he was trying to fill in and correct the mistakes he possibly made. Jumping back and forward in the tab navigation can be a bad experience for the user, and in order to make things simpler it has been decided to save the form data using AJAX requests instead.

When the user clicks the save button of the form the jQuery function, `makeAjaxRequest`, the data is sent to the controller. Here is where the data is validated, and then passed to the model where it is saved. If the action is completed successful, a toast message containing the text "Success" will be displayed, otherwise – "Something went wrong". If the errors are caused by the data not passing the validation of the controller, the function `makeAjaxRequest` will pass the response to the jQuery function `printErrors` which will display an alert box at the top of the form containing the errors (Image 10).

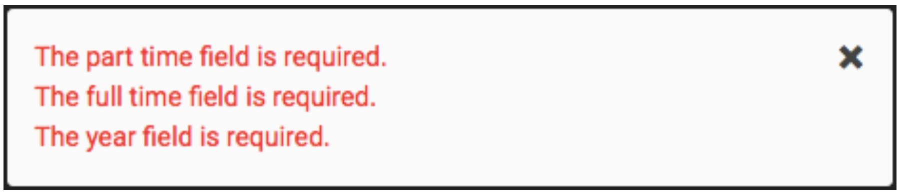

The part time field is required.
The full time field is required.
The year field is required.

Image 10: Error when adding a new record about the number of employees

5.4.Routes validation

All of the application's routes are defined in the `routes/web.php` file. If, for example, the routes for the "Output" section of the questionnaire are taken:

```
Route::group(['prefix' => 'output', 'as' => 'output.'],
function() {
  Route::get('/', ['as' => 'product',
    'uses' => 'Forms\OutputController@getProducts']);
  Route::post('/product_lines', ['as' => 'product_lines',
    'uses' => 'Forms\OutputController@postProductLines']); …});,
```

results the following routes that will be available to the user:

- `/output/`
- `/output/product_lines`

This first route displays the form and allows only requests that use the `GET` method. As for the second route, it is used to store information about the product lines and allows the method `POST`. If any other method will be used for this routes, the validation of the request methods will fail and a response with status code 405 (Method Not Allowed) will be returned. If the user requests a page that does not have the route defined in `routes/web.php` file, then a response with status code 404 (Page not found) is returned.

5.5.Input validation

User's input validation is taking place in the controller using the function `validate`. The function requires two parameters. The first one is an object of `Request` type, whereas the second is an array of rules. Out of the box, Laravel provides a lot of rules to validate the application's incoming data and when this is not enough, the framework allows to use custom created rules.

When the user fills in information about the factory in the section "General information", he needs to provide the year when his organization was founded. The year field is mandatory, it should be a whole number and it is expected to be somewhere between 1900 and the current year. The key in the array containing the rules must be named after the input fields that are contained in the form. The rule for the year of foundation is:

```
$this->validate($request, [
       'year'=>'required|integer|between:1900,'.date('Y'),
        …
   ]);
```

After knowing the year of foundation, later on, when the year is required in the other forms, its value is then expected to be between the year of foundation and the current year. When a field is not mandatory, the rule `required` is replaced with `nullable`.

When the user, for example, provides information about the product lines of his organization, there are more input fields available to him. Sometimes it can happen that the user will fill in the same text in two fields. To constrain him to fill in only different product lines, the rule `distinct` (work only with arrays) is used:

```
'product_lines.*' => 'distinct'
```

Checking the uniqueness of data is a more complicated situation. Every table in the database has unique defined constraints but when the application tries to add a new record that already exists an error will occur. To avoid this, the `unique` rule is used. Checking if an email already exists in the database it is an easy task, and it is done when a new user wants to register. The email is mandatory, it should be a string that is formatted as an email, is no longer than 255 characters and has to be unique in table “factories”:

```
'email'=>'required|string|email|max:255|unique:factories'
```

However, things get more complicated when more fields together have to form a unique constraint. In the section “Input” of the questionnaire, the user is asked to provide 5–10 most used input materials. When saving a new record about what materials the organization uses, it should be considered the fact that a company can have in the database the same material that appears only once for one year, but multiple times for different years. Also, the application does not limit the number of how many times the user can add new materials for a chosen year. To check if an organization already has a record about a given material for a given year in the database the next rule is used:

```
'input_materials.*'=>'unique:factory_input_material,
input_material_id,null,null,factory_id,' . $factory->id .
',year,' . $request['year']
```

The rule will generate the following SQL query:

```
select count(*) as aggregate from `factory_input_material` where
`input_material_id`=? and `factory_id`=? and `year`=?
```

After the query is executed, if the number of records contained in the table is bigger than zero, the validation will fail and an error message will be shown to the user.

The two `null` are used when is needed to skip a value for a specific column, the first one being the value and the second one the name of the column.

Because the number of materials used can be bigger than 10, there is a question that asks the user to provide their total number. Here, the user can change the number of materials, so the `unique` rule will not be effective any longer, and more records for the same year will be added in the database. To avoid this, a custom rule `ValidUMaterials` was created (Image 11). The custom rules are located in `app/Rules/` directory.

The constructer of the class takes up to 3 parameters (`$request`, `$function`, `$message`). The `$function` parameter contains the name of the function to be called in the function `passes`. The called function executes a query and returns true when no records are found, otherwise returns false. The `$message` parameter is by default empty and when is not provided, the constructor will set the class variable `$message` to be the value of `$function`. When the called function returns false, the `message` function is executed and that returns the error text that will be displayed to the user. In the function `validate` the custom rule then looks like:

```
'nr_of_materials'=>['nullable','integer', new ValidUMaterials
($request, 'uniqueCount')]
```

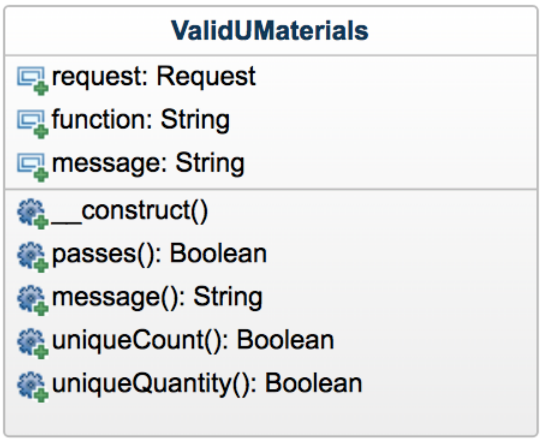


Image 11: UML diagram for "ValidUMaterials" rule

5.6.Saving and retrieving data

Laravel uses Eloquent ORM (Object-relational mapping) to work with database. Each database table has a corresponding “Model” which is used to interact with that table. Models allow to query for data in the tables, as well as insert new records into the table [14].

All models of the application are located in `app/Models/` directory. Most of them are in directories that are named after the questionnaire's sections where they are used.

The models by default work with the table after they are named. For example the model `Factory` will automatically search for the table that is the plural of its name in the database ("factories"). To use another table, this has to be explicitly specified in the class using the variable `$table`. The model `Specification16` can be used as an example. By default, Laravel will search for the table "specification16s" which does not exist in the database but there is a "specifications16" instead.

The variable `$fillable`, can also be found in the models, which is an array containing the columns names that can be mass assignable. When the user sends a new request to save data, if there is an unexpected field, then the model will ignore it and will save only the data for the allowed columns.

Eloquent models allow to define relationships between the tables. In this application are used one to many and many to many relationships. The relationship from Image 6 is described in `Factory` model as:

```
public function sectors() {
  return $this->belongsToMany(
    'App\Models\GeneralInfo\Sector', 'factory_sector'
  );
}
```

As it can be seen from the code, the first parameter of the function `belongsToMany` is the model of the table that is related to the current model, and the second parameter, which is optional, is the name of the intermediary table that these two form. Eloquent is smart enough to understand from the name of the classes what is the name of the intermediary table. It can be specified when the table is named otherwise than the convention or just to make things easier to understand. The function can also take more parameters, where the foreign key or the local key can be specified if they are not named after the convention. Because the application does not require to know to which factories a sector belongs, there is no need to specify the inverse of the relationship in `Sector` model.

For the relationship one to many from the Image 7 in the model `Sector` will be used the `hasMany` function:

```
public function activities() {
   return $this->hasMany('App\Models\Activity');
}
```

Once the models and their tables are created, the application can save and retrieve data from the database. The application uses 3 different ways to save new data.

sync

This method accepts as parameter an array of IDs to place on the intermediate table [15]. All the IDs that are not contained in the array are then removed. In the application, this method is used for saving multiple answers for one time questions. When the current logged in user calls the method `sync` on `sectors`:

```
$this->sectors()->sync($request[sectors]);
```

the sync method will automatically assign the sectors to the user.

attach

This method is used to save data for many times questions. The `attach` allows an additional array of data to be saved in the intermediary table when attaching a new relationship. When attaching a new input material to an organization, the next three parameters have to be provided to the database table: `factory_id`, `year`, `input_material_id`. The controller passes the data to the model as a `Request` object containing:

- logged in user
- array containing the materials IDs
- year

To attach the IDs to the database, first is needed to synchronize each one of them with the year. To do this, the function `syncYearAndItems` from the model `Factory` is used:

```
public function syncYearAndItems($items, $year){
  $pivotData = array_fill(0, count($items),
               ['year'=>$year]);
  return array_combine($items, $pivotData);
}
```

where the `items` parameter is the array containing the IDs. After this the attach method is called:

```
$this->inputMaterials()->attach($syncedItems);
```

which attaches the `$syncedItems` to the logged user.

firstOrCreate

This method is used to save user input which is not validated by `unique` rule. The method itself protects against creating duplicates in the database. First, it tries to find a record that matches the provided columns and their values. If nothing is found, then a new record is inserted in the database, else the instance of the model is returned:

```
OtherInputMaterial::firstOrCreate([
    'factory_id' => $this->id,
    'name' => $item,
    'year' => $request['year'],
]);
```

To retrieve data from the database, it is called the function that defines the relationship between the models. For example, to see what activities an organization has, it is used:

```
$factory->activities()->get()
```

where this for example is used to retrieve all certificates from database:

```
Certificate::all()
```

To retrieve 5 records from the table “water_costs”, that belongs to the logged-in user, where the column “year” is smaller than a given value, next function is used:

```
$factory->waterCost()
    ->where('year', '<=', $year)
    ->take(5)->get();
```

In some cases, when a table from the database does not have an Eloquent model defined, the `DB facade is used to retrieve data.`

To generate a new model, the next command was used:

```
php artisan make:model NewModel
```

5.7.Updating the data

The update of the data in the database is also made in a few different ways. One way of doing this is to retrieve a instance of the model, set the parameters values and then call the `save` method on that model.

The logged in user is an instance of `Factory`. All of the contact information of the organization are contained in the table "factories". When the request is passed to the model, its parameters have their values set to the ones from the request:

```
$this->contact_person = $request['contact_person'];
$this->phone_nr = $request['phone_nr'];
$this->city = $request['city'];
```

After this, the `save` method is called:

```
$this->save()
```

which updates the data in the database.

When the user has already selected and saved the input materials, he has to add additional information like the used quantity or the supplier. This is done by clicking the "Edit" button (Image 12).

Postřiky na ochranu rostlin

Rok: 2011 Množství: ?

Sídlo pěstitele/výrobce: ?

Image 12: Edit button for an input material

The button uses the HTML `data` attribute to store information like the ID of the material and the year when it was used. When the button is clicked, the data is passed to a modal that contains the form for the additional information. After the form is filled in, and the user clicks the "Save" button, an AJAX request is made. Following the validation, the data is passed to the `Factory` model where it is saved using the function `updateInputMaterial`. Here the request parameters are added into an associative array (`$attributes`) where the keys are named after the columns than will be updated in the intermediary table (many to many relationship):

```
$attributes['quantity'] = $request['quantity']; …
```

```
$this->inputMaterials()
    ->wherePivot('year', $request['year'])
    ->updateExistingPivot($request['input_material_id'],
      $attributes);
```

When the user adds the quantity of a final product, the algorithm of how the data is updated is very similar. The only difference is that each record in the table "final_products" has its own ID, so there is no need to use the year in a `where` condition:

```
$this->finalProducts()
    ->where('id', $request['id'])
    ->update($attributes);
```

5.8. Flow of materials

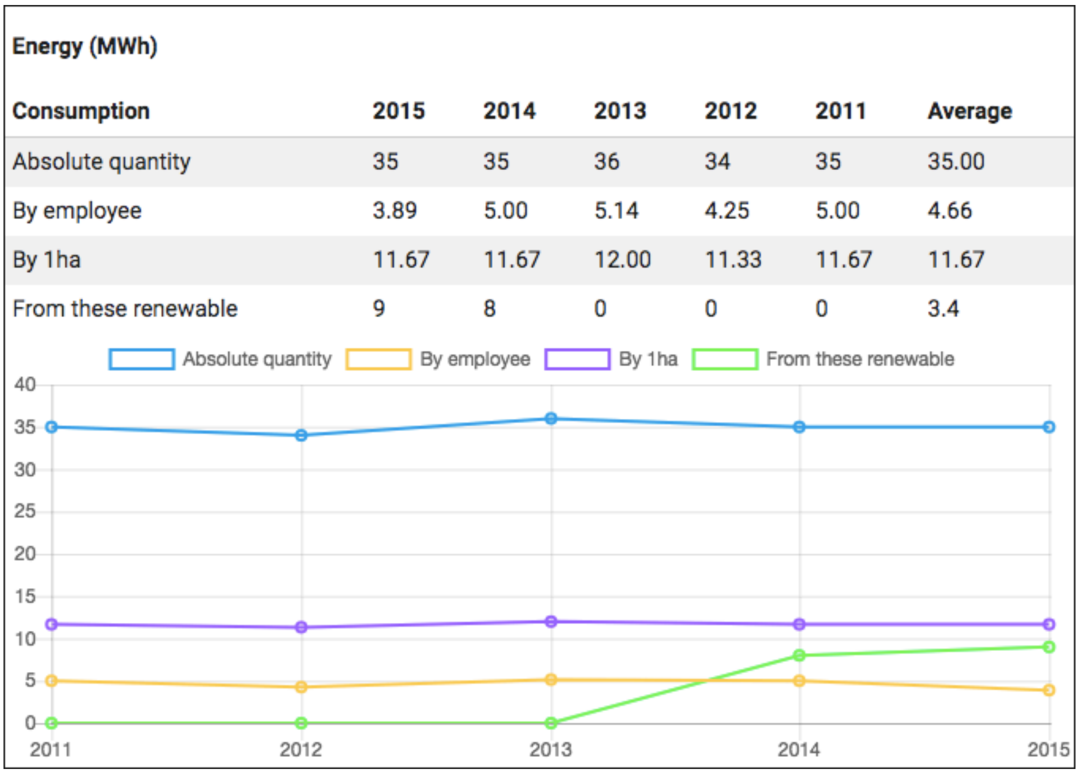
The materials' flow component retrieves the data from the database and uses the HTML tables to organize it in the page. The arrows between tables showing the direction of the flow are arranged in the page using CSS (cascade style sheet). The same arrow image is used, only it is rotated at different angles (Image 16). The HTML block containing the flow of materials is very wide, so in order to see all the information, the user needs to use the scroll bar located at the bottom of this block, to move the flow from the left to the right.

At the top of the page the user has a box, where he can insert the year for which he wants to see the materials' flow. When some of the information that is displayed in the flow is missing, the user is informed about that.

5.9. Reporting

When the user goes to the report page, first thing, he sees a comparison of the evolution of energy and water consumption in the last two years, and secondly, he sees a detailed evolution of the same indicators, together with others, for the last 5 years. When in the database are less than 5 records for an indicator, or the user requires a year when data has not yet been provided, then the evolution of available data is shown.

From chapter 3 is known that the water and energy consumption are assessed by the absolute quantity, quantity share by employees and by organization's surface. The data about the elements mentioned above is stored in different tables. Because the questionnaire has many sections and the application does not constrain the user to answer to the questions contained in them in a specific order, the database can end having, for example, more records about the number of employees than about the company's surface, because both questions require an yearly answer. The application also does not limit the access of the user to the reporting page in case that some data is missing, and as result, the report of indicators can be inaccurate. To avoid this, from the database are first retrieved the absolute quantity, and then accordingly to the years contained by these records, the number of employees and the size of organization's surface.



Energy (MWh)

Consumption	2015	2014	2013	2012	2011	Average
Absolute quantity	35	35	36	34	35	35.00
By employee	3.89	5.00	5.14	4.25	5.00	4.66
By 1ha	11.67	11.67	12.00	11.33	11.67	11.67
From these renewable	9	8	0	0	0	3.4

Image 13: Energy report preview

To draw the graphical part (Image 13) of the report the javascript Chart.js library is used. In the `reporting.blade.php` view are set all the variables needed to be displayed in the chart, some of these are the explanatory labels and the indicators data. After the page is loaded, the script contained in `report.js` file gets the context of the canvas specific for a given indicator, draws and displays the chart.

When the user is asked to indicate the quantity of energy consumed by his organization (table "electricity_consumptions"), in his answer, he should not include the amount of energy coming from own sources. For this, there is an additional part in the section "Input" of the questionnaire. Together with indicating the mix of own energy sources, there should be specified the quantity of energy provided by these sources (table "factory_own_energy"). To get the absolute quantity of energy used, data coming from both named tables would have to be summed. An even more complicated situation is when trying to sum the energy used from renewable sources. The organization can use renewable energy that is not coming from own sources. Of course, the mix can be also indicated together with the quantity used from specific sources (table "factory_energy_mix"). In order to get the quantity of renewable energy used, again, data from the latter two tables has to be summed. The database relation can be seen in Image 14.

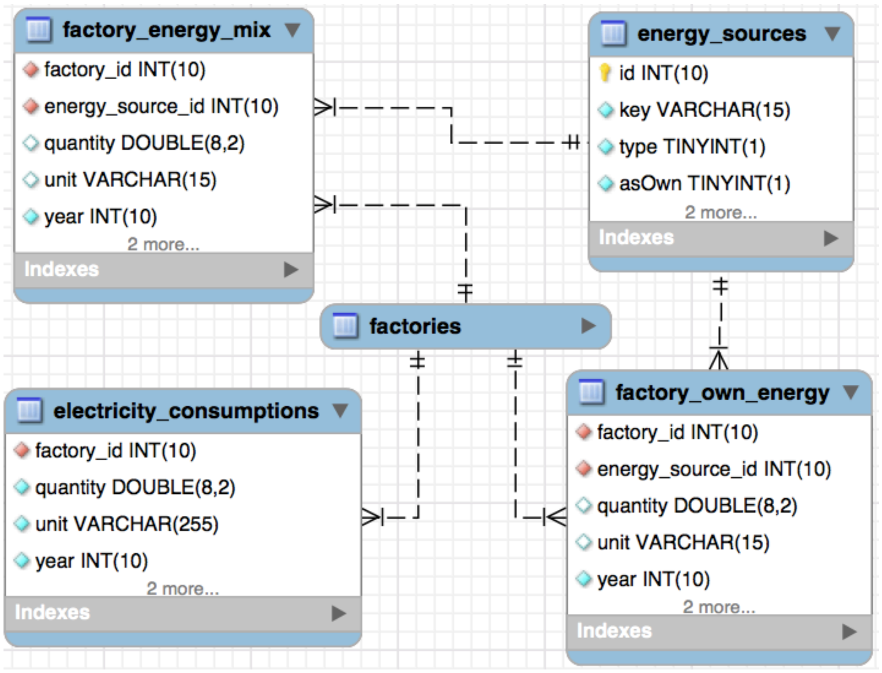


Image 14: Energy consumption and mix – database relationship

Trying to retrieve data and sum the quantities from these tables for 5 distinct years means a lot of operations with the database. To improve this, two additional tables have been added to the database:

- electricity_caches
- renewable_caches

The first one contains the sum for the absolute quantity of energy used, and the second one, the sum of quantities of energy provided by renewable sources (Image 15). The data in these tables is updated by the application whenever the user modifies the quantity of an energy source contained by "factory_energy_mix" or "factory_own_energy". The latter named tables are used when calculating the emissions of dust, SO_2, CO_2, CH_4 and others gases that pollute the air (Image 17) in a given year.

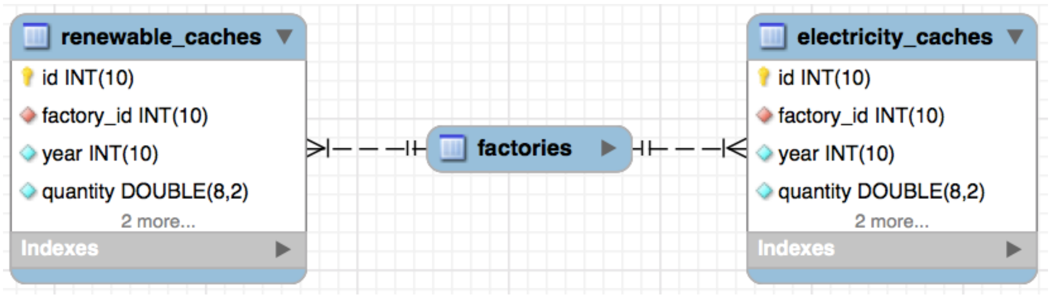


Image 15: Energy cache tables relationship

5.10. Application localization

When the user first opens the welcome page of the application, he can select the language in which he wants the application to be displayed in. The application uses the same routes to display a page in different languages. In order to keep the language selected by the user, the locale is saved in the application's session. When the user selects the desired language, the application navigates to the route for the given locale, sets the session and redirects the user back on the page where he was:

```
Route::get('/lang/{locale}', function ($locale) {
    Lang::setLocale(Session::put('locale', $locale));
    return redirect()->back();
});
```

After the locale being set in the application's session, next time when the user navigates to a different page, the default language of the application is changed to the one selected by the user using the middleware `LanguageSwitcher`. This middleware is executed by the framework's kernel.

5.11. Production mode

One last step, that should be made after uploading the application to a production server, is to change the application's environment from "develop" to "production" and set the debug option to "false". The changes have to be done in the `.env` file. The variables for this are `APP_ENV` and `APP_DEBUG`. Once these changes are made and in case of an error, the user will be seeing a nice page instead of the exception's stack trace. In this way, the exposure of any sensitive data to the user will be avoided.

6. Solution evaluation

The part that took a lot of time during this work was the analysis of the documents I got from my advisor, understanding the relationship between the questions and designing the database to cover every one of them, and store all the possible answers. From time to time, new questions have been added, so the database often suffered modifications.

There are more pairs of tables in the database that could be united in one. However, because the questions that have their answers stored in those tables are optional, it was chosen to use different and smaller tables in order to avoid NULL values in the database.

In some cases, like the reporting of financial cost per unit of waste, the task was confusing. Different types of waste can have their quantity expressed in different types of measuring units (for example kilograms, liters). According to the law of physics, the summing of different units cannot be done, and even if absurdly they will be summed the report will be inaccurate. After consultation with my advisor, it was decided to skip this until further instructions.

The application saves all the user data, but does not always display it. That is the case only with the users input (other input), when an answer was not found in the list provided by the application. The support to edit other input, with the exception of other input materials, is not yet added to the application.

After making an AJAX request to save the questionnaire's data, in case of questions with an yearly answer, the data is not automatically appended and displayed to the user. For this, a message appears to inform the user that in order to see new data the page has to be reloaded. If the user decides to reload the page, then the tab navigation system will jump back to the first option, and the user will need to click the tab where he was before that. Because this can be considered a bad user experience, an option about how to fix this is to use AJAX requests in order to retrieve the new added data, or another option is to send the current tab ID as a parameter in the HTTP request for the view to let it know where the user was before he reloaded the page and then, using jQuery, tell to Materialize.css framework what tab to show. When the user edits some data, like quantity of solar energy used, the new data will be automatically appended to the page.

A drawback of the application is that in order to see the report of some indicators and have accurate information about them, or the emissions table, the user needs to provide answers to many optional questions. For example, knowing the mix of energy sources used by the company is a very specific question and such information may not be always available to the user.

Something that can be confusing, especially for new users, is that the questionnaire has the section “Employees”, however, they are asked about the number of employees the organization has in the section “General information”. A more logical approach, in my opinion, will be to keep all the questions about employees in the same section.

A component that will come in the next version of the application, together with others, is one that will tell the user what his organization has to do in order to improve its impact on biodiversity. This will be done by analyzing the answers that are already stored in the database.

The main advantage of the developed application is that it is focused primarily on biodiversity and the interrelationship between firms and biodiversity, and partly on ecosystem services. The application also deals with aspects like waste, energy or raw materials, but it puts them much more in context with biodiversity. At the same time, it also allows the “classical” evaluation of indicators in terms of consumption or cost.

7. Conclusion

The main task of this work was to prepare the schema of an online application for ecological impact assessment that will have a questionnaire used to collect data. This application will allow the user to see the flow of materials and will report the key indicators, for example the electricity or water consumption. Another task was to study the Laravel framework and use it to implement the application. All the tasks were fulfilled.

In the theoretical part of this work the reader is introduced with the term of environmental management (EM), it is also analyzed why are the ecosystem services provided by biodiversity important for companies and how is EM implemented in the Czech Republic. Together with this, there are also enumerated some benefits that this corporate practice brings to the organizations and are analyzed some of the current applications, available on the market, that are addressing this topic. Also, in this Bachelor thesis are explained some of the Laravel's components used to implement the application.

Next, is described the approach in designing the database schema and the graphical user interface of the components that the application has and are explained the decisions made during its implementation.

In the practical part of this Bachelor thesis is described how the tables of the database were created and what components of the Laravel framework were used to insert the data needed by the application. Next, there are described the modifications that have been made to the framework, also how the user input is validated and the rules used for validation. As well, here is described how the data is saved, updated and retrieved from the database. Last but not least, it is described how do some of the components work and the optimizations made to reduce the number of queries made to the database and the time needed to access it.

The application offers to the user a questionnaire containing more sections to collect data about the organization where he works. The collected data then can be displayed and analyzed using the flow of materials or the report provided by the application.

In the future more components should be added to the application, one of them will tell the user what is that his organization has to do in order to improve its impact on biodiversity, also there will be made improvements on how the flow of materials is displayed.

The application will be offered as service and its target markets at the moment are the Czech Republic, Germany and Poland.

Bibliography

[1] Barrow, C., 2004. *Environment Management and Development*. Taylor & Francis, Inc. ISBN: 0–415–28034–4.

[2] Oxford Dictionaries | English. 2018. *biodiversity | Definition of biodiversity in English by Oxford Dictionaries*. [ONLINE] Available at: https://en.oxforddictionaries.com/definition/biodiversity. [Accessed 09 April 2018].

[3] PricewaterhouseCoopers, 2013. *Naturkapital Deutschland – TEEB DE: Die Unternehmensperspektive – Auf neue Herausforderungen vorbereitet sein,* Berlin. ISBN: 978-3-944280-05-9

[4] Hassan, R., 2005. *Ecosystems and Human Well-Being : Current State and Trends:Findings of the Condition and Trends Working Group, Vol. 1*. Island Pr. p. 25. ISBN 1-55963-228-3.

[5] Kramer a kol., 2017. *Praktická příručka pro management biodiverzity a ekosystémových služeb v regionálních malých a středních podnicích a hodnotových řetězcích*, Zittau

[6] Heitepriem, Nico. (2012). *Unternehmen & Biodiversität – Biodiversität im unternehmerischen Immobilien-und Liegenschaftsmanagement*. p. 20.

[7] Štěpánková, E., 2013. *Environmental Management and its Impact on the Corporate Competitiveness*. p. 20–21 [ONLINE] Availible at: https://is.muni.cz/th/pub0b/Stepankova_DizP.pdf [Accessed 10 April 2018].

[8] EMAS – Environment - European Commission. 2018. *EMAS – Environment - European Commission*. [ONLINE] Available at: http://ec.europa.eu/environment/emas/index_en.htm [Accessed 11 April 2018].

[9] TÜV Rheinland, *Managementsystem – die Unterschied zwischen EMAS und ISO 1400*. [ONLINE] Available at: https://www.tuv.com/media/germany/60_systeme/energie_umwelt/iso1

4001/Umweltmanagement_-
_Unterschied_zwischen_EMAS_und_ISO_14001.pdf

[10] 2013/131/EU: Commission Decision of 4 March 2013 establishing the user's guide setting out the steps needed to participate in EMAS, under Regulation (EC) No 1221/2009 of the European Parliament and of the Council on the voluntary participation by organizations in a Community eco-management and audit scheme (EMAS) (notified under document C (2013) 1114) Text with EEA relevance, p.25.

[11] Litea Solution s.r.o. 2018. *Dobrovolné nástroje v České republice « Enviros*. [ONLINE] Available at: https://www.enviros.cz/2011/03/16/dobrovolne-nastroje-v-ceske-republice/. [Accessed 11 April 2018].

[12] Buschmann, Frank (1996) *Pattern-Oriented Software Architecture*. ISBN: 978-0-471-95869-7

[13] Taylor Otwell. 2018. *Installation - Laravel - The PHP Framework For Web Artisans.* [ONLINE] Available at: https://laravel.com/docs/5.5. [Accessed 25 April 2018].

[14] Taylor Otwell. 2018. *Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans.* [ONLINE] Available at: https://laravel.com/docs/5.5/eloquent. [Accessed 01 May 2018].

[15] Taylor Otwell. 2018. *Eloquent: Relationships - Laravel - The PHP Framework For Web Artisans.* [ONLINE] Available at: https://laravel.com/docs/5.6/eloquent-relationships. [Accessed 01 May 2018].

[16] Enablon. 2018. *EHS Management Software Solutions | Enablon.* [ONLINE] Available at: https://enablon.com/solutions/ehs-management-software. [Accessed 11 April 2018].

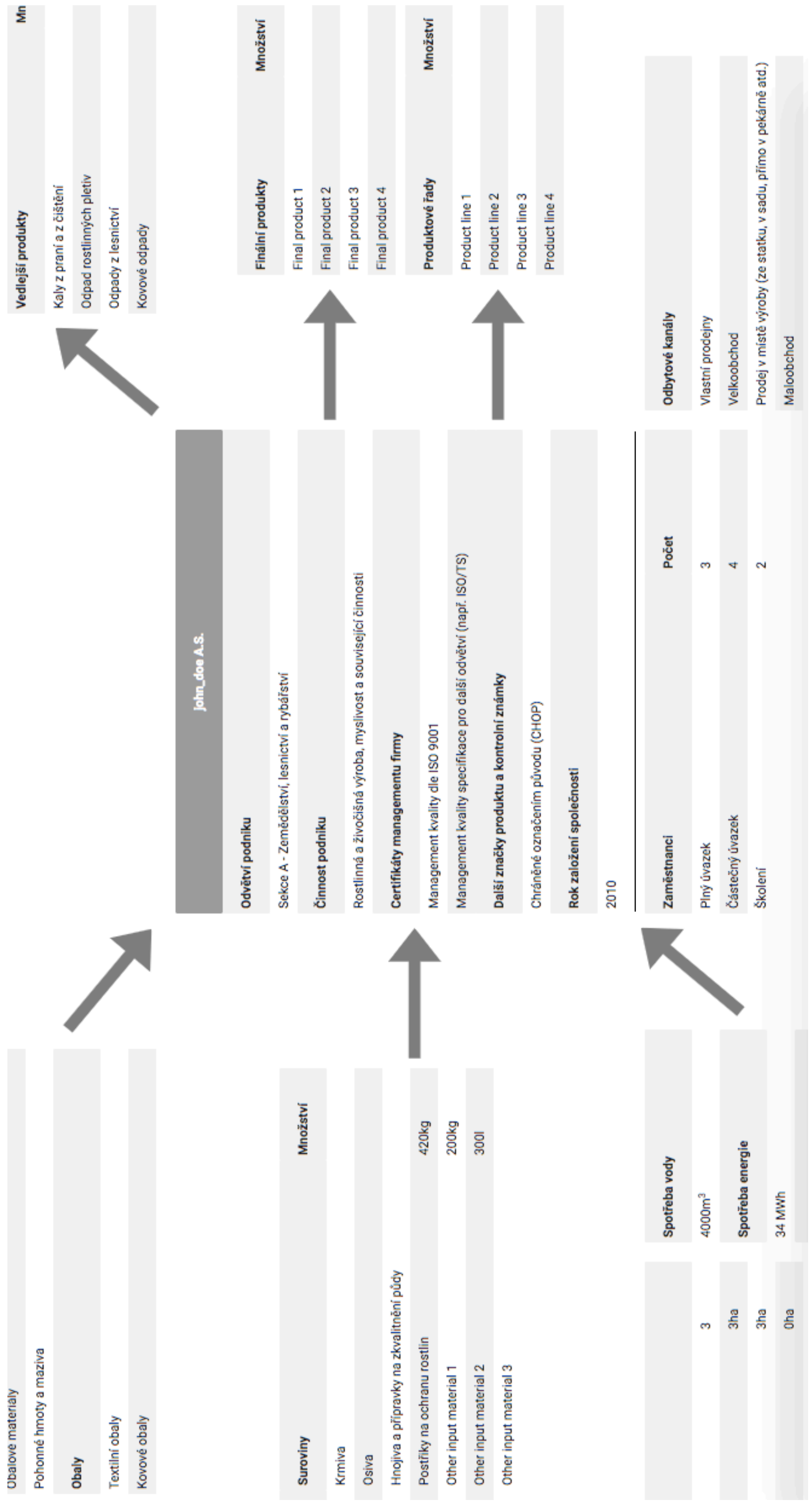
Annex A – Attached CD

On the attached CD can be found the text of this work, together with the source code of this application and the SQL file containing the database structure.

Description of the items:

- *BP_Ion_Ciubaciuc.pdf* – the text of this work in electronic form
- Folder *netsci* – all source codes of the application
- *netsci_tables.sql* – the application's database tables
- *netsci_inserts.sql* – data required by application

Annex B – Flow of materials



Obalové materiály	
Pohonné hmoty a maziva	
Obaly	
Textilní obaly	
Kovové obaly	

Suroviny	**Množství**
Krmiva	
Osiva	
Hnojiva a přípravky na zkvalitnění půdy	
Postřiky na ochranu rostlin	420kg
Other input material 1	200kg
Other input material 2	300l
Other input material 3	

3
3ha
3ha
0ha

Spotřeba vody
4000m^3
Spotřeba energie
34 MWh

john_doe A.S.

Odvětví podniku

Sekce A - Zemědělství, lesnictví a rybářství

Činnost podniku

Rostlinná a živočišná výroba, myslivost a související činnosti

Certifikáty managementu firmy

Management kvality dle ISO 9001

Management kvality specifikace pro další odvětví (např. ISO/TS)

Další značky produktu a kontrolní známky

Chráněné označením původu (CHOP)

Rok založení společnosti

2010

Zaměstnanci	**Počet**
Plný úvazek	3
Částečný úvazek	4
Školení	2

Vedlejší produkty	**Mn**
Kaly z praní a z čištění	
Odpad rostlinných pletiv	
Odpady z lesnictví	
Kovové odpady	

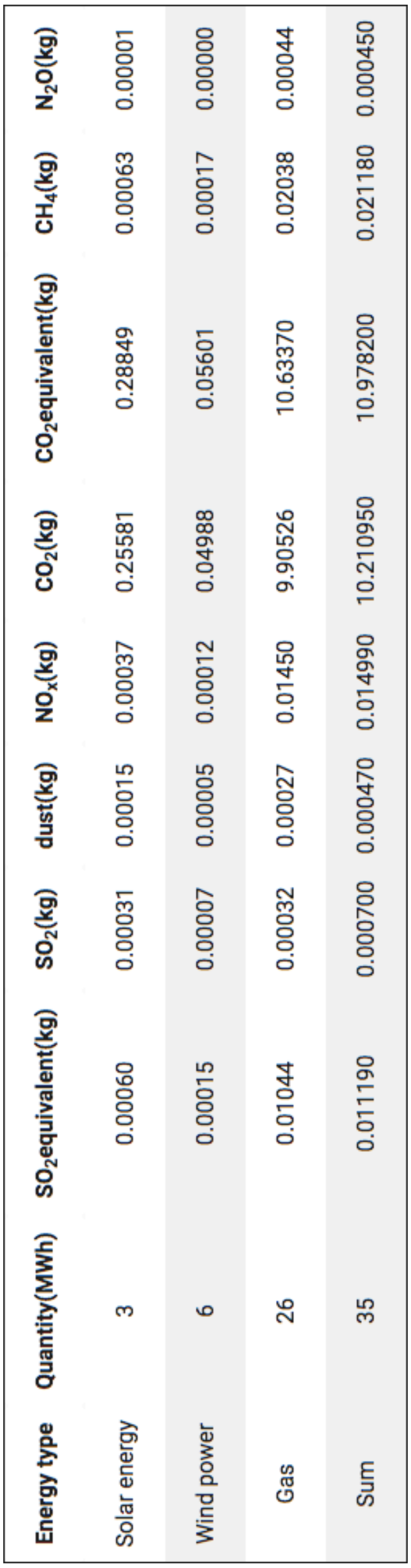
Finální produkty	**Množství**
Final product 1	
Final product 2	
Final product 3	
Final product 4	

Produktové řady	**Množství**
Product line 1	
Product line 2	
Product line 3	
Product line 4	

Odbytové kanály
Vlastní prodejny
Velkoobchod
Prodej v místě výroby (ze statku, v sadu, přímo v pekárně atd.)
Maloobchod

Image 16: Flow of materials

Annex C – Emissions table



Energy type	Quantity(MWh)	SO_2equivalent(kg)	SO_2(kg)	dust(kg)	NO_x(kg)	CO_2(kg)	CO_2equivalent(kg)	CH_4(kg)	N_2O(kg)
Solar energy	3	0.00060	0.00031	0.00015	0.00037	0.25581	0.28849	0.00063	0.00001
Wind power	6	0.00015	0.00007	0.00005	0.00012	0.04988	0.05601	0.00017	0.00000
Gas	26	0.01044	0.00032	0.00027	0.01450	9.90526	10.63370	0.02038	0.00044
Sum	35	0.011190	0.000700	0.000470	0.014990	10.210950	10.978200	0.021180	0.000450

Image 17: Emissions table