

Muck Classification Matrix

Research Results



Chair of Subsurface Engineering
University of Leoben

Department of Civil and Environmental Engineering
Massachusetts Institute of Technology



**Massachusetts
Institute of
Technology**

Julian Wiedorn
August 23, 2013

Contents

1	Introduction	2
1.1	Reuse of Tunnel Excavation Material - Current Situation, Application and Reasons	2
1.2	An Automated And Centralized Classification	3
2	Classification Basics	6
2.1	Material Requirements For Different Reuse Scenarios	6
2.1.1	Concrete Production	6
2.1.2	Minerals For Industrial Use	11
3	The Classification System	12
3.1	Input Information	12
3.2	Information Processing and Output	15
3.3	The Platform	17
3.3.1	Classification Interface	17
3.3.2	Sale Interface	22
3.3.3	Administration Interface	23
4	Software Engineering Basics	24
4.1	Programming Principles	26
4.1.1	Model - View - Controller	26
4.1.2	The Template System	27
4.2	Client - Server Interaction	28
4.3	Programming Language, Database and Additions	31
4.3.1	PHP and Alternatives	31
4.3.2	MySQL and Alternatives	36
4.4	The Classification System	43
4.4.1	Client - Server Interaction	43
4.4.2	The Database	44
4.4.3	User Classes and Their Functions	46
4.4.4	The Main Routines	47
5	Software Engineering Details	49
5.1	The Template System	49
5.2	The MySQL Database Structure	54
5.3	The Section Template and PHP File	57
5.4	Pages and their Functions	61
5.4.1	Basis	62

<i>CONTENTS</i>	II
5.4.2 The "Log on" and "Log out" Procedures	62
5.4.3 The "Classification"	68
6 Example Classification	82
7 Summary and Outlook	87
A Annex	A-1
A.1 Tables for Material Requirements for Industrial Use	A-1
A.1.1 End Product	A-1
A.1.2 Intermediate Product	A-8

Abstract

The reuse of tunnel excavation material represents an outstanding possibility to lower the environmental and economical impact of tunneling projects. The contemplation of the material as a valuable asset, used as raw material in different industrial fields, is just one of the many advantages. With a minimization of landfill areas and, therefore, optimization of the used capital, it could become mandatory for both economical and environmental considerations. As future projects all over the world promise a huge accumulation and handling of raw material, the overall benefits of reusing the material are remarkable. To use this potential and find a possible application for excavated material a proper classification has to be evaluated.

The first part of this thesis describes the basics of the classification, the material requirements for different reuse scenarios. With this information the concept of the classification system, which uses the requirements and reorders the input information is stated. The system is divided in three different parts: a classification interface, a sale interface - a sales platform for the classified material - and an administration interface. The main focus of this work is the classification interface, which is then implemented as a web-based software.

Basics of software engineering are discussed to understand the essential structure of the classification system. A detailed explanation of the server-user interaction, the decision for the system structure (programming language, database and programming structure) is made.

On this basis the programming of the web-based platform, which uses the script language PHP and a MySQL database for data management, is explained in more detail. Source code extracts are commented to understand the programmed routines.

In the final part the system is used for a classification of different tunnel excavation material to show the functionality of the development.

1 Introduction

1.1 Reuse of Tunnel Excavation Material - Current Situation, Application and Reasons

Global changes and the resulting demand for transportation routes significantly increase the number of tunneling projects.

Future infrastructure plans of the EU and Asia show more than 2100km of tunnels on each continent to be built, with an increasing average diameter up to 15m and a length of about 100km (EU). These projects consist of road-, railway, water tunnels and big underground galleries. Considering only the length and the diameter of these tunnels a total cubature of at least about 1.5 billion cubic meter of excavation material has to be managed only in Europe.¹

Nowadays most of the tunnel excavation material is hauled and transported to a waste disposal facility. Only a small percentage of the extracted material is supplied to another use. Because of various reasons this only covers the internal reuse, i.e. aggregates for concrete production or bulk material for different foundations. Therefore, cost savings exist but their extent is rather marginal. The major amount of the material is disposed, which often requires the acquisition of estates. This makes up, together with the maintenance of such landfills, a considerable part of the total costs of a project.²

Another interesting legal component shaped this development in Austria. Currently excavated material is not only treated as waste, it is also legally defined as such. A commercial, external use of muck material, i.e. as raw material for the processing industry, would be accompanied by other regulations than raw minerals from the mining industry.³

Beside those peculiarities with the reuse of muck material, other more problematic and crucial regulations could be avoided. Special treatment of material and strict environmental regulations impede or even prohibit the acquisition and use of surface areas. Also long-term costs, such as renaturation of disposal areas, can be prevented.

In Istanbul, Turkey, for example the extension of the existing metro system with

¹Tokgoomez, "Use of TBM excavated materials as rock filling material in an abandoned quarry pit designed for water storage".

²Zarai, Uromeihy, and Sharifzadeh, "A new tunnel inflow classification (TIC) system through sedimentary rock masses".

³Entacher, D. Resch, and R. Galler, "Abfall oder Rohstoff? Rechtsgrundlagen für die Wiederverwertung von Tunnelausbruchmaterial".

about 200km additional length of tunnel and calculated cubature of about 12 million cubic meter would fill nearly all of the legal storage areas around the city. The strict environmental regulations would make the reuse "mandatory" to implement this project.⁴

The most important point, however, is the fact that the raw material can be considered as available and sure capital or reserve. With the start of the project and the excavation of the tunnel the revenue of selling the material lowers the overall project costs and can even be booked as future income. Increasing the value of a company or just lowering overall project costs decreases the economical latency.

Therefore, from an economic point of view, reusing excavated material has many different advantages. Next to the possible cost savings, involved companies could participate in more projects or include other companies if this is already considered during the first negotiations.

Thereby raw material supply shortages can be prevented and price fluctuations balanced, since low prices are guaranteed by unsafe delivery conditions. This would be easy adaptable for different raw materials needed in the processing industry.

The reuse can even signify an economic boost and justify financially problematic projects, as different, also higher value raw materials, are going to be excavated. The creation of new jobs supports this mode of development.

To use all those advantages the material has to perform a new purpose. The basis for such a use enhancement is a sufficient classification of the material.

1.2 An Automated And Centralized Classification

The engineers of the Gotthard⁵ and Lötschberg⁶ base tunnels have done pioneering work concerning their reuse policy. Their studies and consequent planning led to an intensive reuse of tunnel excavation material as raw material for concrete production.

To make the reuse suitable for different projects a general classification matrix can be introduced. With information about research of the past years and raw material properties of several industry fields a proper classification system can be found. A

⁴Tokgoez, "Use of TBM excavated materials as rock filling material in an abandoned quarry pit designed for water storage".

⁵Lieb, "Materials management at the Gotthard Base Tunnel – experience from 15 years of construction".

⁶Teuscher et al., "Alpenquerende Tunnel: Materialbewirtschaftung und Betontechnologie beim Lötschberg-Basistunnel".

collection of reduced material parameters is the basis of the classification.

Therefore, aim of this work is the development of a general classification system for the excavated material in different tunneling projects.

This is done by establishing a web-based platform for muck material management. As a first step basic information, i.e. name, geography, cubature and time, is entered for a first distinction. Then, considering the required material information (properties) for different use criterias a list of chemical (chemical composition), mineralogical (mineralogy), technical properties (compressive strength and elastic modulus) and aggregate distributions are added. Most of these data is obtained by standardized tests, such as uniaxial tensile testing for technical, X-ray fluorescence spectroscopy for chemical and X-ray diffraction for mineralogical properties.⁷

The platform then compares the properties to a list of predefined reuse scenarios and states the possible reuse. These scenarios consist of internal, i.e. concrete production, and external (industrial) use, both evaluated in intensive literature research. The automated process of the classification establishes a simple evaluation, which takes a wide variety of applications into account.

Such a system helps companies to unleash this tremendous potential and may solve associated problems. Gathered uniform information and the permanent development and improvement support a centralized classification.

In addition to an expansion in a variety of industry sectors, the uniform data structure can enable an automation of the classification process. In case of a continuous excavation this development can be implemented more easily.

Possible clients can be found on both sides: as sellers and buyers. Therefore building contractors and sponsors are just as concerned as processing industries, material producers, waste dealers and landfill operators.

Its use is not limited to the construction phase for quality control, even in the planning, it may very well be considered.

The development of a system including all those features will change the common way of reusing excavation material. In the future a classification matrix could be enlisted as standard planning tool for pricing and selling; the integration of a sales platform would support this step. Furthermore, the enhancement to a raw material stock market would allow prospective buyers to send requests for needed raw materi-

⁷Erben and Robert Galler, "Ressourceneffizienz im Tunnelbau – On-site Analysemöglichkeiten für die Weiterverwertung von Tunnelausbruchmaterial".

als and receive information when it is available.⁸ This would expedite the utilization and marketing of the tunnel excavation material and make it available immediately.

⁸Erben and Robert Galler, "Ressourceneffizienz im Tunnelbau – On-site Analysemöglichkeiten für die Weiterverwertung von Tunnelausbruchmaterial".

2 Classification Basics

The following sections show the reuse criteria and the information which is needed to obtain classification results. This part is based on literature⁹ and provides the basis for the classification system, which is described after that.

2.1 Material Requirements For Different Reuse Scenarios

This section covers possibilities of reuse and the needed information to classify the material. Following constraints have to be in mind to economically reuse a material, see Table 1.

Geology	Technology	Guidelines	Demand
Geotechnics	Excavation method	Laws	Construction site
Geochemistry	Processing	Standards	External Use
Petrography	Site Organization		

Table 1: Reuse Constraints

Geotechnics, Geochemistry and Petrography describes the geological constraints, which affect the classification. Grain size distribution and contamination for example is influenced by the excavation method, processing and site organization. The classification itself and the reuse criteria have to be chosen according to laws and technical standards, which also affect the evaluation and testing. Additionally the demand, both on-site and external, should be considered.

A classification system, as described here, only considers few aspects of these constraints, mostly some of the geological information and different industrial standards.¹⁰

2.1.1 Concrete Production

As these classifications consider tunneling excavation material, the main focus is the on-site concrete production for tunneling purposes.¹¹ Aggregates used for concrete construction influence the characteristics of concrete, i.e. elastic modulus, tensile strength, temperature expansion or processability. Specific requirements are written in different standards, i.e. the ÖNORM standard in Austria, which is published

⁹Daniel Resch, “Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen”.

¹⁰Ibid.

¹¹Gertsch et al., “Use of TBM Muck as Construction Material”.

by the Austrian Standards Institute (ASI). The Following ÖNORM standards and guidelines are used to specify a classification¹²:

- ÖNORM EN 12620:2008 - Aggregates for concrete
- ÖNORM B 3131:2010 - Aggregates for concrete - Rules for implementation of ÖNORM EN 12620
- ÖNORM B 4710-1:2007 - Concrete - Part 1: Specification, production, use and verification of conformity (Rules for the implementation of ÖNORM EN 206-1 for normal and heavy concrete)
- ÖBV guideline for inner lining concrete
- ÖBV guideline for shotcrete

Inner lining concrete in Austria is divided in exposition classes which follow a systematic classification according to the environmental conditions on site. The following classes are distinguished:

- XC - carbonation and density of the concrete
- XD - Deicing Salt (chlorides, i.e. road salt)
- XF - frost attack
- XA - chemical attack

Shotcrete production is mainly needed during the process of excavation. The process itself requires a smaller maximum aggregate size compared to inner lining concrete. Shotcrete is categorized in shotcrete classes, which use following properties, next to the already mentioned exposition classes:

- early strength classes - J1-J3
- strength classes

Concrete segments, which are mainly used during machine-assisted excavation, are characterized by statics and safety needs of tunneling constructions and use similar classification criteria.

With consideration of the external influences, some testing has to be done to withstand their effects. Table 2¹³ shows a summary of these requirements.

¹²Ayaydin, "Classification of Excavation with Austrian Code B2203: Main aspects and experiences".

¹³Daniel Resch, "Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen".

Requirements	Symbol
grain density	for concrete composition
aggregate distribution	according to grain size distribution
compressive strength	minimum between 60-70 N/mm ²
elastic modulus	minimum 30.000 N/mm ²
grain shape	SI25, SI40
mica content	<30% or <25% amount of mass
wear resistance	LA<40, BR<75
freeze-thaw resistance	F1, F2 depending on influences
alkali-silicic reactivity	strain < 1%
water-soluble chloride	chloride free (<0.01%)
acid-soluble sulfate	AS0,8 (<0.8%)

Table 2: Aggregate Requirements

Compressive Strength and Elastic Modulus¹⁴ Aggregates strongly influence the compressive strength and elastic modulus of the concrete, but no limits are defined for them. As the values are usually much higher than those of the concrete, a strength of about 100N/m², depending on the concrete, and an elastic modulus of 30.000N/mm² can be adopted. An evaluation of the strength can be done with a point load test. The elastic modulus can be calculated on the basis of the measured modulus of the concrete or by testing the intact rock.

Aggregate Distribution Aggregate distribution is determined by distribution curves for different grain sizes, usually between 0.063 - 2mm and 4 - 62.5mm. This is specified stating the minimum and maximum grain size. The ÖNORM sets the limits for oversize and undersize grains. Therefore the denomination GCXX/XX defines coarse, GFXX fine and GAXX a mixture of grain size distributions, where XX is the range of the grain size in mm. The amount of required water and cement has to be considered, when using different grain distributions.

The grain distribution requirements for inner lining concrete is given by distribution classes, which depend on the area of use i.e. heading, bench, ceiling, ... The range of the maximum grain size is usually between 16, 22 and 32 mm.

Shotcrete uses a grain size distributions between 0/8 and 8/11 mm with the designation mentioned above.

The easiest and normal way to evaluate the grain distribution is by comparing the

¹⁴Daniel Resch, "Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen".

actual distribution curve to the standard curves in the guidelines.

Tables 3 and 4 show the average arithmetic values of the distribution curves for both, the inner lining concrete and shotcrete.

maximum grain size	0/4 mm	4/8 mm	8/16 mm	16/22 mm	16/32 mm
16 mm	54%	16%	30%		
22 mm	47%	8%	26%	19%	
32 mm	47%	7%	25%		21%

Table 3: Arithmetic mean of the mass fractions of the grain size distribution curves sorted by the maximum grain size for inner lining cement

maximum grain size	0/4 mm	4/8 mm	8/11 mm
11 mm	70%	90%	10%

Table 4: Arithmetic mean of the mass fractions of the grain size distribution curves sorted by the maximum grain size for shotcrete

Grain Shape To describe the grain shape both of the following indices can be used:

- SIXX - the maximum amount of non cubic grains with a maximum of XX=40 mass percent for some exposition classes. The SI value describes the ratio of the smallest and largest grain.
- FIXX - the ratio of flat grains sorted by distribution classes. The screening is done by bar screens and the limit XX is chosen individually, according to lab tests and guidelines.

Amount Of Muddy Fines The grain size of washable fines is smaller than 0.063 mm and can be evaluated according to ÖNORM EN 933-1. The limits are stated in the ÖNORM B 4710-1 and their values range from 4% for small (4 mm) down to 2% for large grain sizes (32 mm). These limits can be exceeded by doing different tests concerning given regulations. In addition a minimum amount of all fines <0.125 mm is set in the ÖBV guideline for inner lining concrete. These limits are set by the fX.X value which states a allowed deviation of this minimum (X.X%).

Wear Resistance According to the field of application the following limits of indices may have to be considered:

- Los Angeles Index LA < 40
- Refrangibility index BR < 75
- Point load index index ABR - ranges between >2.5 N/mm² and >3.5N/mm² as minimum

The evaluation is done according to the ÖNORM EN 1097-2 standard (LA Test), the french AFNOR P 18-579 standard (LCPC Test for BR and ABR). As there is a direct correlation between the BR and the LA value and the BR value needs less material, it is preferred if the access to the material is limited.

Freeze Thaw Resistance The freeze-thaw resistance testing has to be done for fine and coarse grain sizes individually. Coarse material (4-63 mm) undergoes several freeze and thaw procedures, according to ÖNORM EN 1367-1. Their mass loss has to be less than 2% (F2). The limits for fine grain sizes are similar, but the testing is done by measuring the surface mass loss on a standardized concrete cube.

Alkali Silicate Reactivity The evaluation of the alkali silicate reactivity is done by measuring the strains, according to ÖNORM B 3100. These effects occur because of a chemical reaction and are not allowed to be higher than 0.1% after a 2-week short test and 0-05% from the 2nd to the 52nd week (long term). For tunneling projects experience showed that the short test is sufficient.¹⁵

Water Soluble Chloride For corrosion protection the amount of the chloride should be less than 0.01%, according to ÖNORM EN 1744-1.

Acid Soluble Sulfate To prevent increases in volume through chemical reactions the amount of acid soluble sulfates has to be less than 0.8% (AS0.8). ÖNORM EN 1744-1 describes the evaluation of the acid soluble sulfate content.

Aggregates With Mica The content of mica in aggregates has an effect on

- strength properties,
- water content
- processability

¹⁵Daniel Resch, "Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen".

of concrete. Therefore it is important to evaluate the content. This can be done with following methods:

- count method on a thin-sections
- shape separation
- specific gravity separation
- X-ray diffractometry

Different tests, which were done in research studies, determined a maximum content of 25%. Because of a mass loss of about 5% caused by wet processing the final value is 30%. The swiss SIA standard defines a maximum mica content of 2%.

2.1.2 Minerals For Industrial Use

This part shows the classification criteria for industrial minerals. Both, so called "intermediate raw material" - raw materials with lower limits, which are then processed - and "end raw materials" - for direct industrial use, therefore higher limits - are considered.

"Intermediate Products" are classified by their mineralogy, i.e. category "Carbonates" - material "Limestone", whereas "End Products" are classified in two ways:

- First by the mineralogy and their use in the industry, i.e. category "Brick-earth" - use "Wall Bricks".
- Second by industrial use and the required minerals, i.e. category "Steel Industry" - use "Bauxite as Slagformer".

As the requirements only consist of tables of chemical composition, the details - limits - of each use scenario can be found in the Annex section.

Table 5 and 6 state some reuse scenarios divided into "Intermediate Products" and "End Products" and the mentioned categories. A complete list of the reuse scenarios and its limits can be found in the Annex section.

category	suggested raw material
Carbonates	Limestone
Carbonates	Dolomite
Carbonates	Spat-Magnesite
Carbonates	Gel-Magnesite
Carbonates	Magnesite (RHI)
Clay	kaoline (crude)
Quartz	rock crystal
Sulfates	Baryt
Vulcanic Stones	Bims
Feldspar	Feldspar
Al-Oxides	Raw Alunite
Mg-Oxides	Raw Talc
Phosphates, Sulfur, Salt	Phosphate Minerals
Mica	Biotite
Heavy Minerals	Ilmenite
Beryllium Minerals, Bromine, Iodine	Raw Graphite

Table 5: Reuse Scenarios for Intermediate Products

3 The Classification System

This section summarizes the use criteria and arranges the information in a way as it can be processed systematically. Properties, which should be evaluated, are collected in the "Input Information", whereas the classification itself is described in "Information Processing and Output". The last part shows an automated platform and the way it will be implemented.

3.1 Input Information

The Input Information contains of variables which describe the properties of the materials needed for classification. To introduce an economical affordable system the input parameters are restricted to evaluations which are done anyway or could be done very easily and cheaply. Parameters needed for special use or handling, i.e. processing of contaminated material, are not considered.

The tested parameters range widely, i.e. from the compressive strength to different chemical compounds and elements, according to the use. To ensure a systematic classification all input parameters have to be considered and rearranged into different

category	suggested raw material
Brickearth	Brickearth for Wall bricks
Steel Industry	Bauxite as slagformer
Steel Industry	Olivine for steel industry
Steel Industry	Olivine for foundry industry
Steel Industry	Fluorite for metallurgical grade fluorite
Steel Industry	Limestone for metallurgy
Steel Industry	Dolomite (uncalcined) for pig iron (direct use)
Steel Industry	Dolomite (uncalcined) for steel production
Steel Industry	Dolomite (calcined) for refractory industry
Steel Industry	Dolomite (calcined) for steel production
Steel Industry	Magnesite (calcined) for transformersteel-coating
Steel Industry	Magnesite (calcined) for steel industry
Cement Industry	Clay Cement
Cement Industry	Nepheline Syenite
Cement Industry	Nature Cement
Cement Industry	Portland Cement
Cement Industry	Gypsum Anhydrite
Paper Industry	Limestone
Paper Industry	Magnesite

Table 6: Reuse Scenarios for End Products

categories. Parameters which are needed for different use scenarios are assembled in groups, which should simplify the evaluation process.

The input parameters¹⁶ are mainly parted in following categories:

- Technical Parameters
- Chemical Parameters
- Mineralogical Parameters

Chemical Parameters The chemical parameters cover following elements, compounds and properties.

For the eluate (liquid phase):

1. general: pH value, electrical conductivity, exhaust residues
2. anorganic parameters: Al, Sb, As, Ba, Be, Pb, B, Cd, Cr, Co, Fe, Cu, Mo, Ni, Hg, Se, Ag, Zn, N, Cl, CN, F, Mn, nitrates, nitrites, P, SO₄, Tl, V, DOC, TDS
3. organic parameters: TOC, C-index, EOX, AOX, Tensides, Phenole, PAK, PCP.

For solids:

1. chemical compounds: SiO₂, CaO, MgO, K₂O, Na₂O₃, CaCO₃, Corg, S, loss on ignition, MgCO₃, SO₃, P₂O₅
2. anorganic parameters: As, Pb, Cd, Cr, Co, Cu, Ni, Hg, Zn
3. organic parameters: TOC, C-index, PAK, PCB, BTEX.
4. other: alcali-silicate reactivity, water-soluble chloride, acid-soluble sulfate.

Mineralogical Parameters Those include the following minerals: mica, kaolinite, sericite-illite, smectite, chlorite, quartz, feldspar, calcite, dolomite-ankerite, goethite, hematite, siderite, pyrite, gypsum and hornblende.

Technical Parameters The group of technical parameters mainly contains input information for the concrete reuse scenario.

- compressive strength

¹⁶Daniel Resch, "Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen".

- wear resistance - LCPC Test and LA Test
- freeze-thaw resistance
- grain shape - percentage of the non-cubic grains
- fine grain size distribution - <0.002mm to 0.063mm
- coarse grain size distribution - 0.063mm to 80mm

3.2 Information Processing and Output

The input parameters are the basis of different output scenarios, which were discussed in Section 2.1.

With the help of the input information the output can be generated. Before this result can be obtained, the information has to be processed, which is done by comparing different usage possibilities.

Right now the following main fields of usage (ouput) were considered:

- End Products:
 - Concrete Production
 - Cement Industry
 - Paper Industry
 - Lime as Industrial Mineral
 - Brick Production
- Intermediate Products:
 - Carbonates
 - Clay
 - Sulfate
 - Quartz
 - Vulcanic Stones
 - Feldspar
 - Al-Oxides
 - Mg-Oxides

- Phosphates, Sulfur, Salt
- Mica
- Heavy Minerals
- Beryllium Minerals, Bromine, Iodine

All input parameters have to be in between the limits of one output scenario. The limits can be reviewed in Section 2.1. If a given muck material is in between all of these limits, it can be considered as potential raw material. This procedure has to be repeated for all use scenarios.

Because of the fact that many input data is collected, the output scenarios can be expanded as long as their classification parameters are already evaluated.

3.3 The Platform

With the knowledge of the classification concept, explained in Sections 3.1 and 3.2, one can generate a platform with a systematic sequence and easy management of the muck materials and their usage as raw material. Much effort and time can be saved when using a predefined input system (frontend) with a built-in comparison (processing) and an easily accessible output (backend). Furthermore such a platform can be used as a resource for both the owner of the raw material and the prospective consumer. This is very important, because the material immediately performs its intended purpose (evaluated reuse).

Functions of the Classification platform include

- easy management of different raw materials,
- classification according to predefined criteria,
- sale platform for the owner and
- resource for raw material buyers.

Next to the functional part the following basic prerequisites should be considered¹⁷:

- clarity
- comprehensibility
- customization

As the system should also be easily accessible, but protected from unauthorized access, the optimal implementation of the classification platform is as an admission based system.

The entered input and the output are stored on a specific computer, which makes remote use from every workstation possible and save due the fact that all data are gathered on a single and secure computer. The admission is granted via a user-password system, which prevents unauthorized access.

3.3.1 Classification Interface

A flowchart of the classification system can be seen on Figure 1.

The description of the system is done by a commented and reduced optical representation with the help of an example classification. The single steps are performed

¹⁷Welling and Thomson, *PHP and MySQL*.

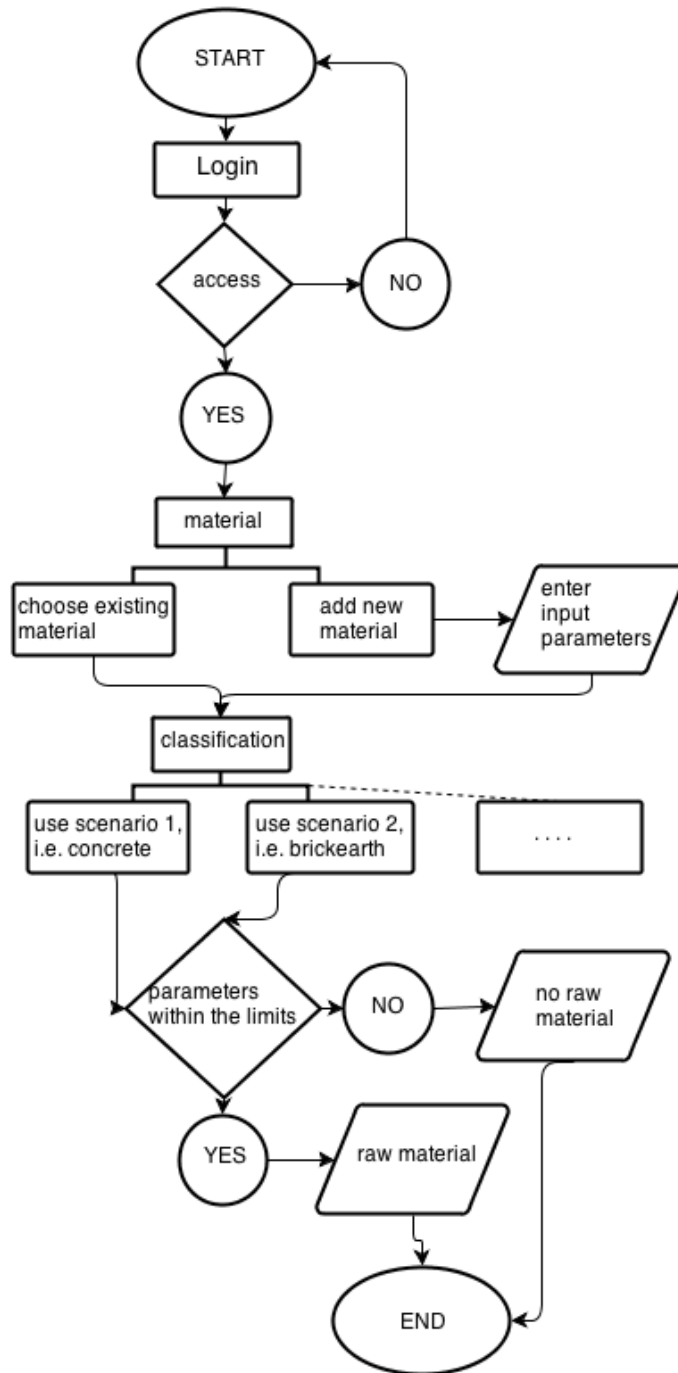


Figure 1: Flowchart - Classification

after the principle in Figure 1.

The following basic steps, which will be discussed in detail, have to be performed to get a classification result:

1. add a new or activate an existing material
2. enter technical, chemical and / or mineralogical input parameters
3. obtain the results, by using the different output functions

The management of the muck materials is done with a simple user interface according to Figure 2.

Main Menu

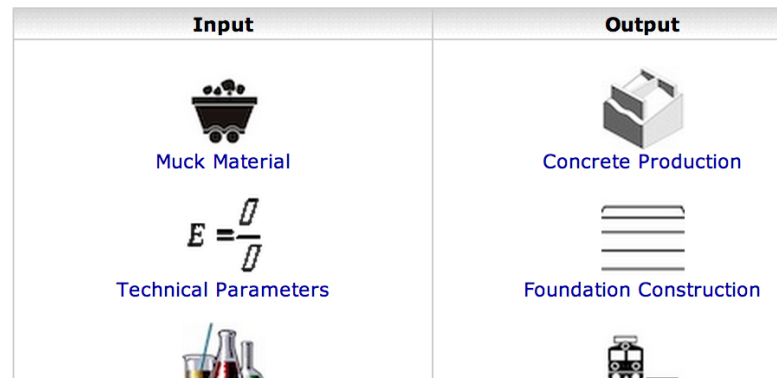


Figure 2: Interface - Main Menu

The control menu is divided into two sections, input and output, which fulfill the purpose of parameter input and usage (output), as discussed in Sections 3.1 and 3.2.

In addition to the technical, chemical and mineralogical parameters the function "Muck Material" was added, see Figure 2. Within this category basic information like name, geographic information, cubature can be saved. Also administrative functions like selecting, editing or deleting muck material can be done there.

Adding New / Choose Existing Material According to Figure 1 the first step of the classification, after login, is adding a new material or choosing an existing material. Figure 3 shows the "Add Material" function in detail.

If one chooses an existing material a list of already saved materials can be seen at the bottom of the same page, see Figure 4.

Enter Technical, Chemical and / or Mineralogical Parameters The input parameters can be entered by going back to the Main Menu and using the other

Material Selection

deactivate material / edit material / add new material	
date (last edited)	
name	<input type="text"/>
description	<div style="border: 1px solid black; height: 100px;"></div>
<input type="button" value="add new material"/> <input type="button" value="edit material"/> <input type="button" value="deactivate material"/>	
delete material	
<input type="checkbox"/>	Shale7

Figure 3: Interface - Add Material

activate material	
<input type="radio"/>	Kaolinite22
<input type="radio"/>	Illite
<input type="radio"/>	Granite
<input type="radio"/>	Kaolinite
<input type="radio"/>	Shale
<input type="radio"/>	Clay
<input type="radio"/>	Shale
<input type="button" value="activate material"/>	

Figure 4: Interface - Activate Material

control sections in the "Input" category, i.e. "Technical Parameters" see Figure 2.

A simple example of the input can be seen in Figure 5. The information, gathered in preceding tests, i.e. X-ray analysis, can be inserted in the predefined forms. In this example values for some chemical parameters - the percentage of SiO_2 , Al_2O_3 , Fe_2O_3 and TiO_2 - are already filled in the form.

chemical compounds		
SiO_2	M-%	50.61
Al_2O_3	M-%	14.49
Fe_2O_3	M-%	1.53
TiO_2	M-%	0.36
CaO	M-%	4.34
MgO	M-%	4.33

Figure 5: Interface - Chemical Parameters

Obtain the Result After entering all the needed input parameters the output can be obtained. One can get detailed information if the entered material can be reused for one of the sections in the "Output" category by selecting the output of interest, i.e. "Brickearth". If the material is not within the limits the exceeding values can be seen easily.

Brickearth

input parameters			
	M-%	measured value	limits
SiO2	M-%	55	50-68
Al2O3	M-%	40	10-20
Fe2O3	M-%	5	3-8
TiO2	M-%	1	0-2
not usable as brickearth			

Figure 6: Interface - Brickearth

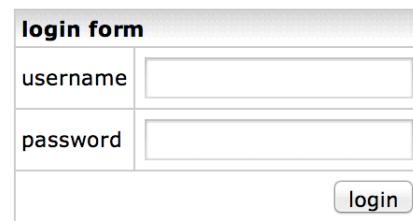
The evaluation shows the measured values, which were entered in the "Input" category, and the limits taken from industry standards. The "processing unit" compares the values, e.g. Al_2O_3 , and returns "not usable" because the input of 40% exceeds the range of 10% - 20%.

This automated process should be easily adaptable to new / changed standards and contains all the information needed for a proper classification.

System Environment, Permission and Rights The entered data is stored in a database - to make the material available at different workspaces and let the user redo the classification as often as needed. The given data is saved for re-evaluation or updating. In this special case a connection to the internet is required during the use of the system.

The system is also designed for simultaneous work of multiple users. For multi-use every user has to be registered in the system, which then grants access if one has the permission. This is done by a simple login query when the software starts according to Figure 7.

Login



The image shows a web-based login form titled "login form". It contains two input fields: "username" and "password". Below these fields is a "login" button. The form is enclosed in a light gray border.

Figure 7: Interface - Login

The system itself is programmed to provide the users functions according to their predefined rights. More importantly, the saved data are strictly separated for each user to avoid overlaps and ensure privacy.

3.3.2 Sale Interface

To gain access to the "Sale Interface" one has to be registered in the system. The structure of this part is very simple and straight-forward. It will consist of a small search engine, which provides the user with information about the available materials, i.e. cubature, geographical area and, of course, parameters of the raw material.

The user is able to change the search settings according to his needs:

- location
- field of use
- cubature

- time frame

With a given location the user is able to delimit his search for materials which are within the delivery area. The area of use defines the industrial use the material is anticipated. The cubature acts as a restriction or a order criteria.

The result of the search should be a list of possible and available raw materials, see Figure 8, which then can be selected for further information.

material selection				
	ID	name	owner	date
<input type="radio"/>	11	Illite	julian	2013-09-20 16:42:41
<input checked="" type="radio"/>	12	Granite	julian	2013-09-20 16:42:19
<input type="radio"/>	14	Kaolinite	julian	2013-09-20 16:42:08
<input type="radio"/>	13	Shale	julian	2013-09-20 16:40:29
<input type="radio"/>	7	Clay	julian	2013-08-08 18:51:44
<input type="radio"/>	10	Shale	julian	2013-08-06 19:04:58
				<input type="button" value="show material"/>

Figure 8: Interface - List Materials

3.3.3 Administration Interface

To ensure a functional and clear environment an administrative part is needed. The main purpose of this interface is

- user management,
- right management (access granting),
- application support
- and quality management of the input data.

All these features can be carried out by using a control menu, see Figure 9. The handling of this menu works just like the control menu of the "Classification Interface" - actions can be executed by using the preexisting functions, e.g. "Add User".

The user management, mentioned above, consists of registering, deleting and editing users, see category "User" on Figure 9.

Main Menu

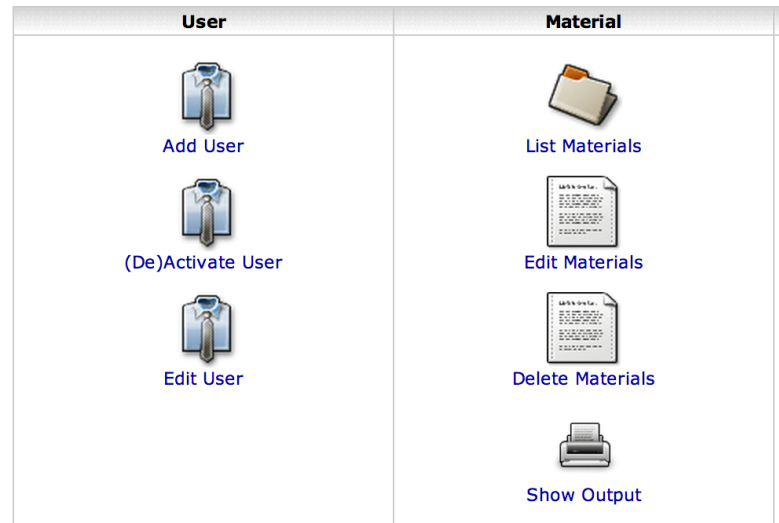


Figure 9: Interface - Main Menu Admin

Figure 10. shows the "Add User" function in detail.

A sample register form is shown on Figure 10.

The difference between the "Add User" and "Edit User" function is a prefilled form and a selection menu for the user, who will be edited.

The application support is done by the "Material" category on Figure 9. There all materials of all registered users can be seen, edited and deleted. Also the possible reuse options can be evaluated, but with less information output data.

4 Software Engineering Basics

This section represents a basic introduction answering the following questions:

- How does a system communicate with the user?
- What is needed to implement it?

The first part, the "Programming Principles", explains how the internal structure of the software looks like, what is needed for programming and how each part is connected. Then "Client - Server Interaction" describes how the user, the client, sees the system and how the communications with the software in the server works. Finally, different programming languages, databases and additional functions are mentioned with their advantages and disadvantages. From this results a decision for

Add User

register form	
username	<input type="text"/>
password #1	<input type="text"/>
password #2	<input type="text"/>
email	<input type="text"/>
description	<input type="text"/>
name of the university	<input type="text"/>
user levels	
standard, if nothing is checked!	0 - inactive / no rights
<input type="checkbox"/>	1 - admin
<input type="checkbox"/>	2 - seller
<input type="checkbox"/>	3 - buyer
<input type="button" value="register"/>	

Figure 10: Interface - Add User Admin

a certain programming language and database.

4.1 Programming Principles

4.1.1 Model - View - Controller

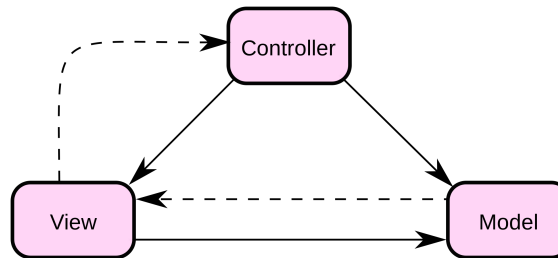


Figure 11: Model - View - Controller

The Model - View - Controller principle, short MVC, is a software-architecture pattern, which explains the basic principle, how the system deals with the users' interaction. It is divided into three units, see Figure 11¹⁸:

1. Model
2. View
3. Controller

The goal of these programming principles is to introduce a flexible program design, which allows subsequent amendments or extensions. To understand the principle behind this partition, the function of each part has to be known.¹⁹

The model contains the data, and optionally the logic of the system. It is manipulated by the "Controller" and informs the "View" part. The "Controller", the controlling unit, sends information to the model, according to a possible user input. "View" is the representation which processes the data from the "Model" part. There can also be a direct connection between the "Controller" and the "View" if no new data are needed and only the "View"'s representation of the "Model"'s data should change.²⁰

Figure 12²¹ shows a graphical representation of this principle with additional consideration of user interaction.

¹⁸Wikipedia, *Model View Controller*.

¹⁹Powers, *PHP Object-Oriented Solutions*.

²⁰Welling and Thomson, *PHP and MySQL*.

²¹Wikipedia, *Model View Controller*.

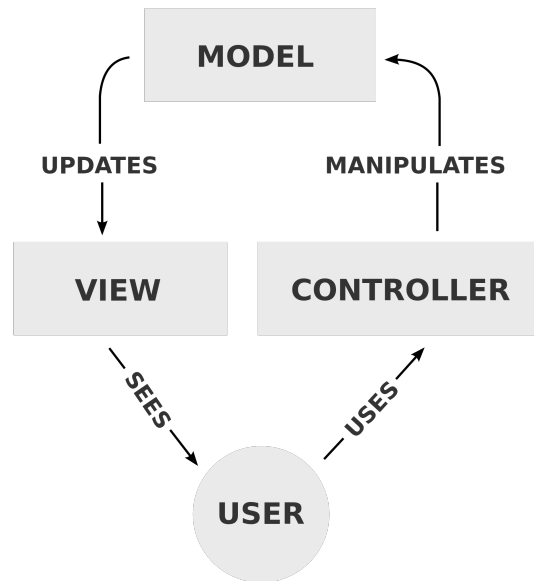


Figure 12: Model - View - Controller with User Interaction

4.1.2 The Template System

The implementation of the MVC principles is done with a simple template engine. A Template Engine is a software, which takes a file and fills its "gaps" with content. A template system is the easiest and most common system for introducing the basic MVC principles into small web-based software solutions.²²

The "View" part is saved in the template files, mostly in its own directory, which only consists of information for the optical presentation. The "Controller" (controlling unit), which is saved in other files, acts according to the user input and fills the "gaps" with the requested information. This is done by requesting the data from the "Model", which acts as a space for these data.²³

The advantage of this system is the independence of code and design. Changes in layout or additions can be arranged very easily, just by manipulating the template files. The programmed routines and, therefore, the software itself, are strictly separated and protected from unintentional changes.

A practical example of such a system is an online telephone directory. The user requests information with the help of the "Controller", for example a search engine. This engine connects to the telephone database and gives the requested telephone number to the "View" part, which is the optical presentation of the requested num-

²²smarty.net, *Smarty Template Engine*.

²³Quakenet/#php.de Staff, *Quakenet/#php Tutorial*.

ber to the user.²⁴

Based on these principles a more detailed explanation of the user - server interaction is provided in Section 4.2.

4.2 Client - Server Interaction

The most important part of using a webbased database system is to understand how the connection between the actual user and the system itself works. The foundation is a server-client interaction where the client sends requests to the server and gets specific responses, according to the given action.²⁵ Figure 13²⁶ shows an example connection of the client-server situation.

The client, in this case is a browser, which runs on a computer and is controlled by the user. The browser, e.g. Internet Explorer, sends requests to another computer, called the server.

In this case the server is a HTTP web daemon, running on an existing physical computer. HTTP is the transfer protocol standard, similar to a postal carrier, like UPS. Then the HTTP web daemon would be a physical place, where the request, the "package" can be delivered. In this analogy a postal office would act as the browser system.

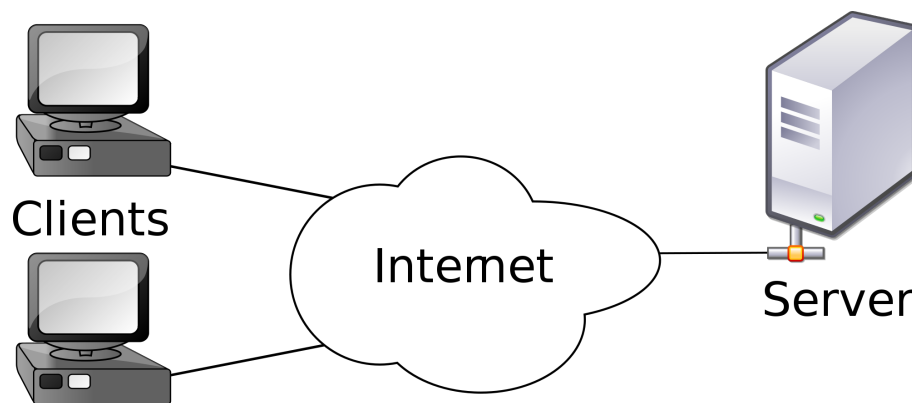


Figure 13: Client-Server Model

A dynamic webpage has some more extras which are necessary to fulfill the requested options. To handle specific requests and get an adequate answer a programmed course has to exist, which only uses the given functions and excludes all

²⁴Wikipedia, *Model View Controller*.

²⁵tutorialspoint.com, *Client Server Model - Architecture*.

²⁶Wikipedia, *Client-Server Model*.

other possible inputs. This "code" is saved on the server.²⁷

This programmed course is described with a script language, which is interpreted by the server, with the help of an so-called interpreter. The most common script language in case of client-server platforms with databases is PHP, which has a C / Pearl-like syntax, ASP or Java.²⁸

The actual code to handle the different input cases is, as mentioned, on the server and the user can only communicate with it through HTML-requests sent by the browser. HTML is the markup language for creating webpages, which can be read by the browser. So the user sends a request by using the predefined options on a HTML page, which was sent from the server to the client's browser, see Figure 14.

Script languages such as PHP have no possibility to alter the system, the computer of the client directly, it can only control what will be sent to the client's browser. So PHP only changes what the user sees, that means it uses a process which is saved on the HTTP server and the user will get, i.e.: a different HTML-page with the help of this process. There is no direct access to the user's computer.²⁹

The user can not see the programmed routines and how the server will react (the "answers" (HTML pages) the user receives). The PHP program acts as a compiled program, which makes it executable immediately.³⁰

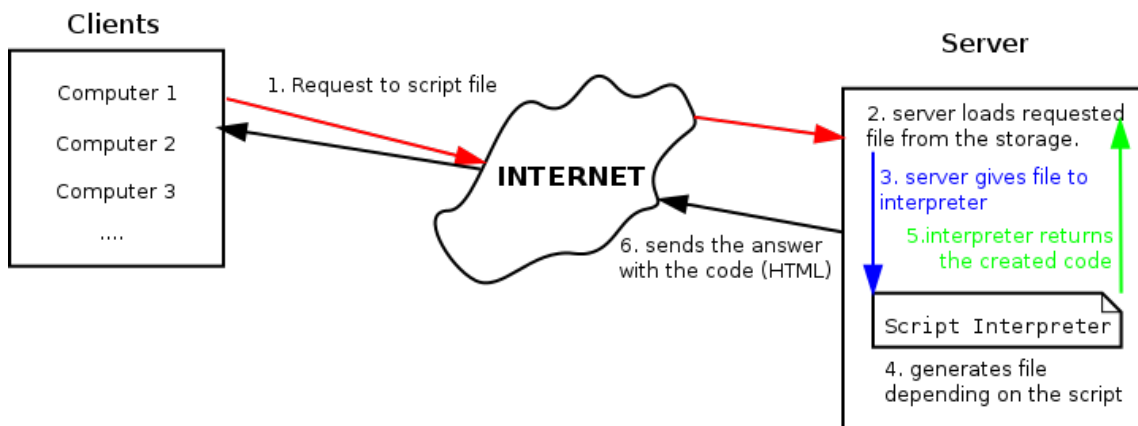


Figure 14: Server-Client-Script Interpreter Model

Requests always contain data from the user whether already sent as an input by a specific form or "hidden" by using different menu options.

Usually a structured database is used to save data. Therefore input, which is given

²⁷Wikipedia, *Client-Server Model*.

²⁸The PHP Group, *PHP.NET Manual*.

²⁹MacIntyre, Danchilla, and Gogola, *Pro PHP Programming*.

³⁰The PHP Group, *PHP.NET Manual*.

by a form, can be stored in an accurate structure and recalled later.³¹

To make use of the advantages of this structured space a database server is required. The database server is, similar to HTTP servers, a part of the physical computer i.e. with the help of an input language data can be stored.³²

Finally the script program processes the user requests and therefore "talks" to the database and saves or requests data from it. This is done by a database handler, which is provided by the script language. A graphical interpretation of the interaction can be seen on Figure 15.

So following things are needed to provide the basis of the classification software and to use it³³:

- A physical computer with a client software, a browser, i.e. Internet Explorer, which is used by the user and can send requests.
- Another computer which acts as server and receives requests and sends information back. It is subdivided into a webserver, which stores all the programming files and a database server, which stores the data of the user.
- A HTTP handler to send data from the server to the client and vice versa and Interpreter to run the programmed routines and "talk" with the database, both on the server.

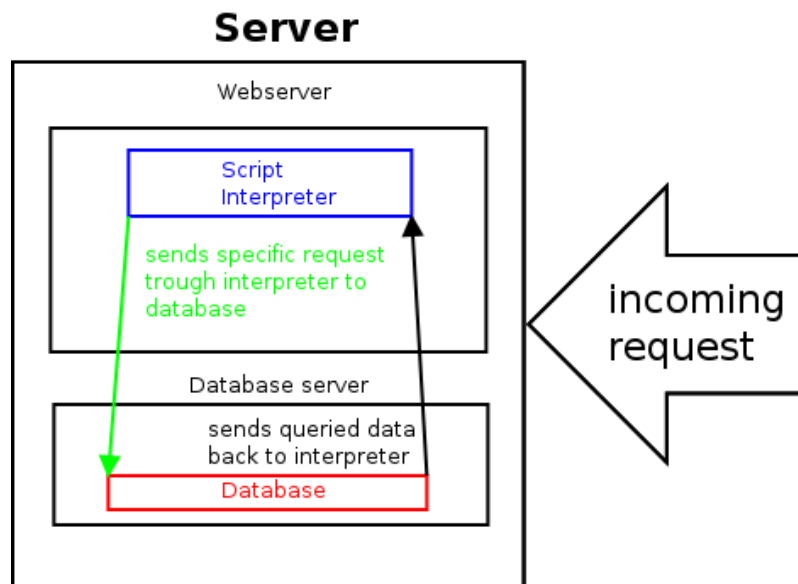


Figure 15: Script Language - Database Interaction

³¹MacIntyre, Danchilla, and Gogola, *Pro PHP Programming*.

³²Welling and Thomson, *PHP and MySQL*.

³³Rauch and Beer, *Netzwerke - Grundlagen*.

4.3 Programming Language, Database and Additions

This section states the differences of available script languages with a general overview of common databases and how the script language interacts with them. All of the discussed script languages are able to create dynamic webpages, because the objective is to react to the user input, i.e. data input and classification. As grain size distributions have to be displayed, the needed methods to implement this requirement in a web platform are mentioned briefly.

4.3.1 PHP and Alternatives

PHP³⁴ PHP, a recursive acronym for PHP: Hypertext Preprocessor, is an open-source server-side script language invented by Rasmus Lerdorf in 1995 with a syntax similar to C and Perl. It is the most used language for web programming in about 79% of all dynamic websites and has a wide range of supported databases³⁵ and numerous predefined function libraries. When used as foundation for web appearances, the script is often embedded in the HTML code and saved on the server, although PHP can be used as a general-purpose programming language to create any kind of software. In contrast to JavaScript³⁶ the PHP code is interpreted on and by the server and sends the generated HTML result to the client. Because of this reason PHP can be used with all major operating systems, i.e. Linux and derivatives, Microsoft Windows, Apple MacOs, and also all mobile operating systems.

Because of its wide range of usage nearly all servers have the ability to execute PHP codes. There are different ways how to interpret the code but the most common, which is also mentioned in a previous chapter, is with a webserver module - available for all important webservers like Apache and Microsofts IIS (Internet Information Services).

In version 5 of PHP, Objects were introduced, which filled the gap between the "modern" programming language and old procedural style of programming. This produces better performance and brought many new functions and possibilities, which are now state of art in object oriented programming, i.e. abstract, final classes and methods, and also interfaces.

³⁴The PHP Group, *PHP.NET Manual*.

³⁵Oracle Corporation, *MySQL Manual*.

³⁶Sun Microsystems, *JSP Manual*.

ASP.NET³⁷ With about 20% of market share Microsoft introduced the second most important technology to create dynamic websites - ASP.NET, the replacement of the original Active Server Pages (ASP). ASP.NET is based on the .NET framework which is a foundation to display .NET programs on Microsofts IIS or at least ASP.NET compatible servers. Writing programs for ASP.NET means that no specific programming language skill has to be acquired. All .NET supported programming languages like C# and VB.NET can be used to finish projects.³⁸ Therefore ASP.NET is not a programming language, it is more a framework of different techniques.

In contrast to the free software PHP, ASP.NET is proprietary, which means - in this special case - that the code is closed for inspection (closed-source vs. open-source). ASP.NET is accompanied by a wide range of possible functions, like image processing, without the need of installing extra libraries.³⁹

To use all the features of ASP.NET, one is bound to the technologies of Microsoft and its server software. There exist some free alternatives but, because of the closed-source policy, they are not 100% compatible and never up-to-date. So ASP.NET is free available, but the used software is not, which means royalties have to be paid to get full support and functions.

With ASP.NET the concept of the Code-Behind-Model (CBM) emerges. Similar to other .NET applications this means a strict distinction of the code for presentation and the content itself. The Code-Behind-File is interpreted before the site is requested, which reduces the errors which take place during runtime. In this file the developer has the possibility to set all the possible actions in a lifecycle of a ASP form including events like user interactions. Figure 16⁴⁰ shows the "page lifecycle" of an ASP.NET framework.

This Code-Behind-Model in ASP.NET is one of the most important differences to the old version, ASP. The CBM strictly follows the principles of Model-View-Controller(MVC) separation and therefore makes division of labour in development possible and enhances the work flow significantly. An simple representation of MVC in the ASP.NET framework can be seen on Figure 17⁴¹.

³⁷Microsoft, *ASP.NET Manual*.

³⁸Wikipedia, *ASP.NET*.

³⁹Microsoft, *ASP.NET Manual*.

⁴⁰Ibid.

⁴¹Ibid.

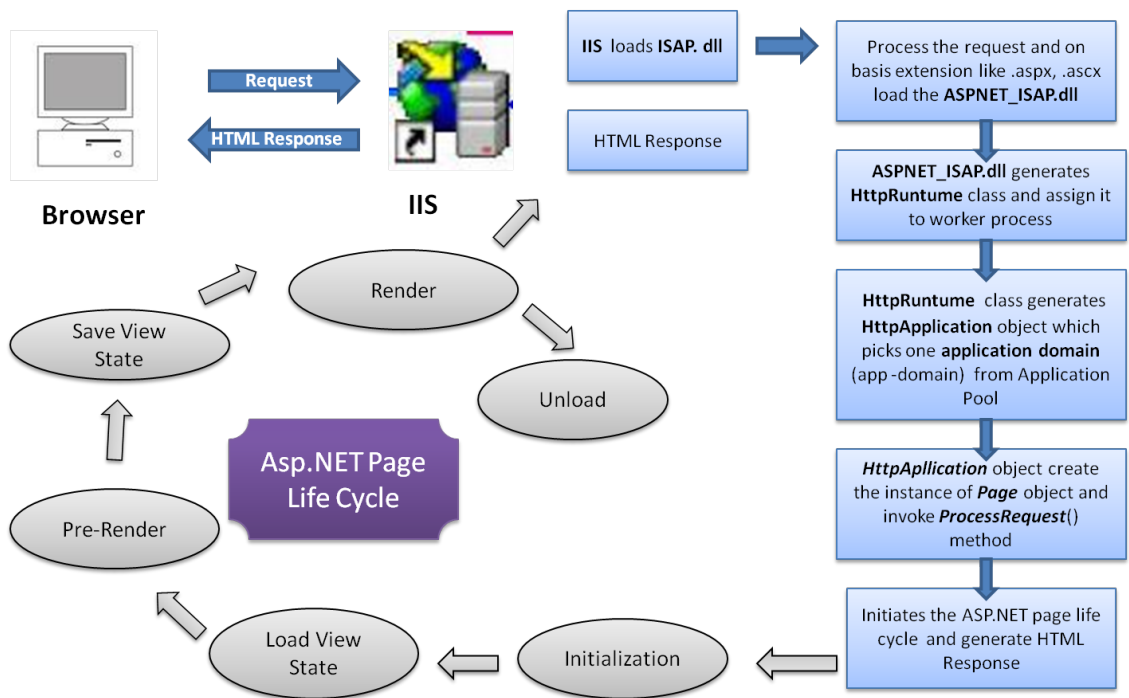


Figure 16: ASP.NET Page "Lifecycle"

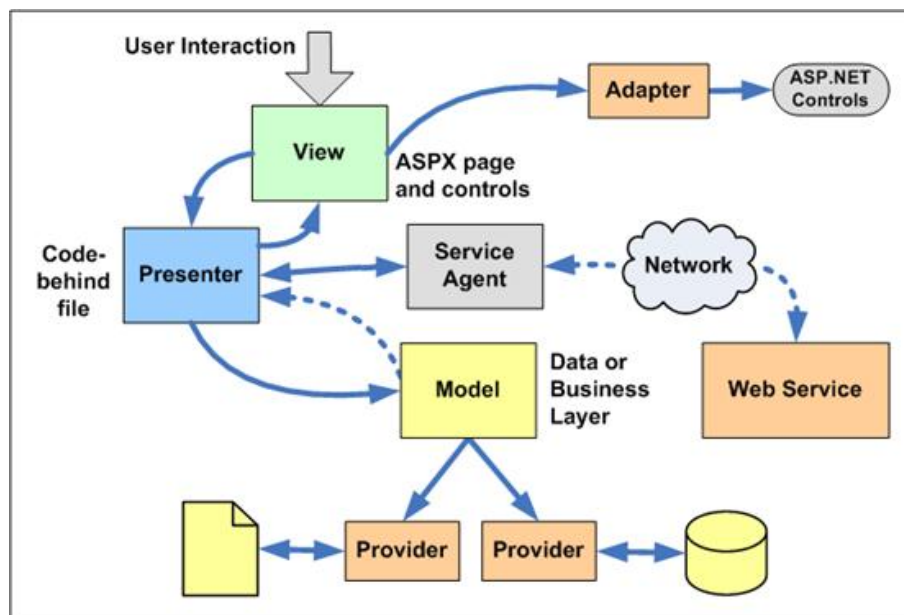


Figure 17: ASP.NET MVC

JSP⁴² Other alternatives to PHP include Java, Perl and CFM (ColdFusion Markup language), where Sun's JSP (JavaServer Pages) has the highest share with about 4%.

Like PHP and ASP, JSP is a web programming language for dynamic generation of HTML and XML responses. In 1995 Sun Microsystems released as a high-level abstraction of Java servlets. JSP and therefore the "translated" servlets, which are classes to extend the applications of a webserver, are programmed in Java. JavaServer pages need a translator, the JSP compiler, that performs a conversion of the JSP page into a servlet and then to bytecode. These generated programs can then be executed with the JRE and the results are presented in HTML. Figure 18⁴³ shows a simple understandable "lifecycle" of a JS Page.

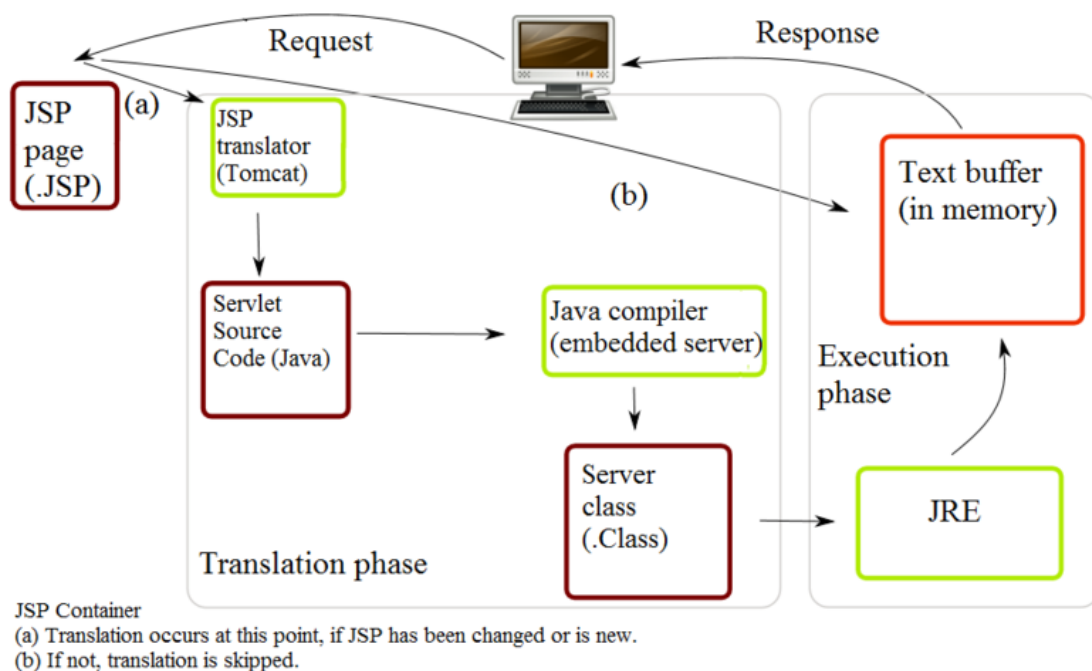


Figure 18: JSP Lifecycle

A main advantage of JSP is that it is not based on an own programming language: it uses Java.⁴⁴ Therefore a massive library is available for programming with JSP, although the libraries of ASP and PHP are also quite big and sufficient to implement "normal" web projects. Considering this fact and the free available support of JSP concerning webserver it can be seen as a mixture of both, PHP and ASP.NET. An important decision factor, nearly a knock out criterion, is the speed of JSP. As described above there are many steps which have to be processed, next to installing an own engine needed to execute the scripts, which makes the whole system very

⁴²Sun Microsystems, *JSP Manual*.

⁴³Wikipedia, *Java Servlet*.

⁴⁴Leitenmüller, *JSP - Java Server Pages*.

slow (in comparison to other available options). That is also one of the main reasons the programming language Java is not used when it comes to performance.

Comparison Tables 7, 8 and 9 show the summarized advantages and disadvantages of each scripting language.

PHP	
<i>+ Advantage</i>	<i>- Disadvantage</i>
speed good support open source free server support easy programming	content / code management wide range of function with extra packages

Table 7: Advantages and Disadvantages PHP

Usually the missing MVC principle can be solved easily: an advanced programmer is able to do the content / code management separation and can even implement a MVC framework. Usually it is easy to find free packages for needed extraordinary functions. In case of the muck classification matrix, a package to display grain size distribution has to be loaded, all other functions are available without loading additional packages.

ASP.NET	
<i>+ Advantage</i>	<i>- Disadvantage</i>
lots of good implemented functions "out-of-box" MVC framework possibility of using different programming languages	not as fast as PHP, but faster than JSP proprietary license royalties for webserver limited webserver support special webserver needed for full support smaller community

Table 8: Advantages and Disadvantages ASP.NET

All discussed systems to implement the muck classification matrix are object-oriented, or at least would be able to use it. The list of supported databases is large but all of them vary concerning the working procedure. The speed of the database depends on the amount of data and the used database. As we save same data sets, only the second point has to be considered.

JavaServer Pages	
<i>+ Advantage</i>	<i>- Disadvantage</i>
huge range of functions (Java) "out-of-box" MVC good support free server support easy programming	content / code management really slow own engine is needed to run a lot of intern steps

Table 9: Advantages and Disadvantages JSP

In addition to these objective advantages and disadvantages there is the subjective opinion of the programmer. Although PHP is not meant to be "Object-Oriented", one can use it, if needed. Another important fact is the personal preference, which supports a positive and fast handling while programming. Considering all these facts (objective and subjective) PHP is the language of choice.

In the case of the muck classification matrix only one part of the code is object-oriented - the interaction with the SQL database. The rest is straight-forward and can be programmed procedurally. With an implemented MVC, an easy template engine, the objective to be expandable can be fulfilled easily.

4.3.2 MySQL and Alternatives

According to the previous chapters a system for organized data collection, a database, is needed. The term "database" is often used for both the real database - the data and its structure - and the management system (DBMS). Different of these software management applications exist to create, define, query, update and administrate databases.⁴⁵ The most important are:

- MySQL^{46,47}
- Microsoft SQL⁴⁸
- Oracle⁴⁹

⁴⁵Rauch and Beer, *Netzwerke - Grundlagen*.

⁴⁶Oracle Corporation, *MySQL Manual*.

⁴⁷DuBois, *MySQL*.

⁴⁸Microsoft, *SQL Server Manual*.

⁴⁹Wikipedia, *Oracle Database*.

All of these databases use the Structured Query Language (SQL) or a similar language, which is based on SQL, as database language to process and request data.⁵⁰

Database Models⁵¹ The most important difference between these systems is the restructuring of the given data. The following models are distinguished:

- Hierarchical database model (Figure 19)
- Network model (Figure 20)
- Relational model (Figure 21)
- Object model

There exist some more, but their usage is very limited and they are not noteworthy for the webbased usage. The hierarchical model has a tree-like structure and only has one-to-many relationships. It uses simple parent/child relationships, which means that every "parent" data or category can have more children "data", but only one parent is connected to each child, see Figure 19.

Hierarchical Model

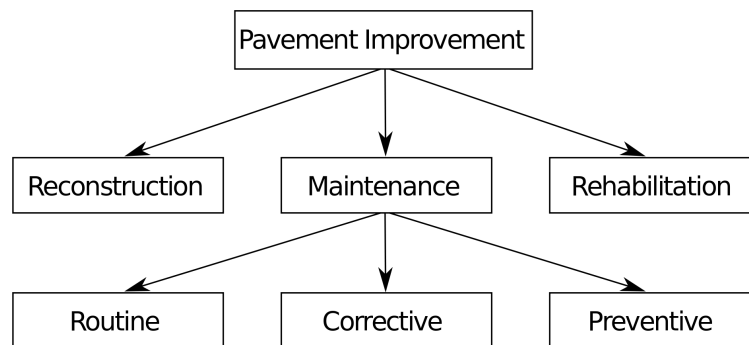


Figure 19: Hierarchical Model

This structure limits the use of the database to simple models. The network model extends this concept and permits multiple parents and therefore is able to display a generalized graph model. As its structure is still hierarchical some different data relationships are not possible, it prohibits cycles.

The relational model replaced the network model as it is able to manage data in a more declarative way with the help of first order predicate logic. The data is grouped in tuples and connected with relations. Object databases enhance this concept and

⁵⁰Date, *An Introduction to Database Systems*.

⁵¹Wikipedia, *Database model*.

Network Model

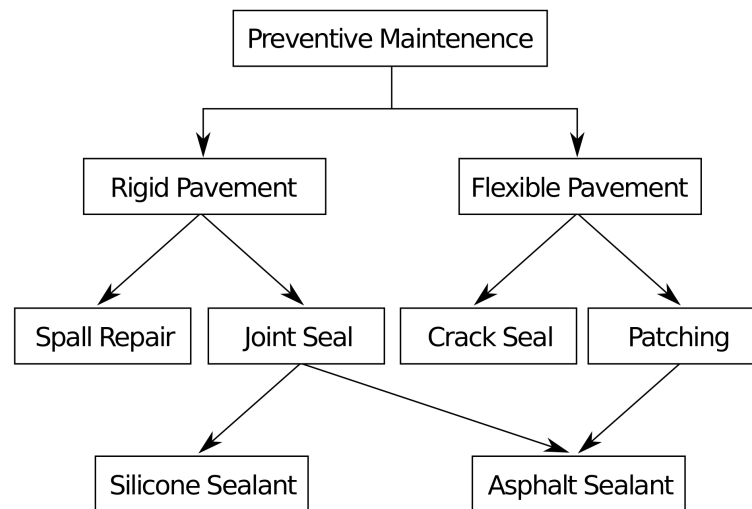


Figure 20: Network Model

introduce the object as datatype. Therefore hard-to-structure data can be saved and recalled as objects easily, it differs in the table structure of the relational model, see Figure 21. It shows two different tables, which set up a simple user database:

- The first table consists of different names and their user alias.
- The second table is an extension of the first one. It consists the user alias (login) as "key" for the telephone number.

The point of relation is the "login" of each user. Therefore the phone number is assigned to one user (relation).

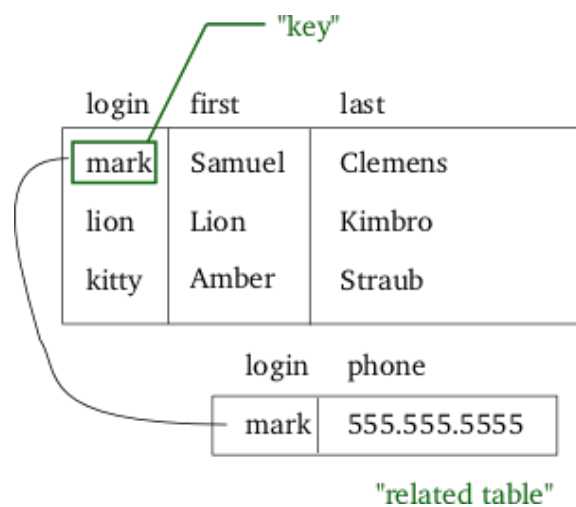


Figure 21: Relational Model

Database Orientation⁵² Another distinctive criterion is the "orientation" of the system. This includes:

- OLTP - Online Transaction Processing
- OLAP - Online Analytical Processing

OLTP, also called real-time transaction processing, is built for many small requests - direct and fast without time delay. This system is mainly used for operational business processes or, more importantly, webshops, knowledges bases and content management.

OLAP is an analytical information system, which is mainly used for prolonged evaluations, e.g. data mining, which is a statistical process of data pattern evaluation. In contrast to OLTP the main purpose is to perform complex analysis projects, which cause a very high volume of data.

So in consideration of the data in the classification matrix, which means "small" data sets and direct requests for further information processing, all of the mentioned DBMS use OLTP for information processing.

Comparing the database management systems the differences concerning performances are relatively small. Like PHP MySQL is open-source and free, which means a strong community and high availability of add-ons. Oracle and Microsoft SQL are proprietary, which may become very important for companies, which need special features and add-ons as it has a better customer support. But using normal features this feature can be ignored and implies additional costs.

In contrast to Microsoft SQL and Oracle, MySQL has no database administration tool included. This can be an advantage because there is a wide range of different tools available and one can decide for according to what is needed. But, because MySQL is the most common database type, all servers already have an administration tool preinstalled, phpMyAdmin.

A preferred field of MySQL is the data storage for web services. It is frequently used in conjunction with the Apache web server and the PHP scripting language. Famous web applications and web pages like Flickr, Nokia.com, YouTube, Google, Facebook and Twitter use this combination.

The fact that MySQL is free, has a good support, and the field of use or handling of the database is very similar to other existing projects, it is the tool of choice.

⁵²Gabriel, Gluchowski, and Pastwa, *Datawarehouse und Data Mining*.

The work experience with MySQL databases and its administration software, php-MyAdmin, supports this decision.

PHP and MySQL - Interaction⁵³ There are several possibilities for "speaking" with a MySQL database in PHP. An Application Programming Interface (API), which is a class and/or collection of methods or functions is needed. These interaction tools are called "connectors" . The most common tools are:

- MySQL - the original MySQL API
- MySQLi - the MySQL Improved Extension
- PDO - PHP Data Objects

All of the APIs above are included in PHP and available for the immediate use. The original MySQL API is the oldest and first connector of the three. Because of its procedural programming interface and the lack of security the PHP development team stopped working on it and switched the development status to "Maintenance only". Its functionality is also limited to MySQL versions below 5.1, which is why PHP does not recommend using it for creating new projects. MySQLi is an improved version of the original MySQL. It supports both procedural and object-oriented programming and its development status is still "Active". Like MySQL it is limited to the use with a MySQL database, but has improved security options.⁵⁴

The most important difference concerning MySQL and alternatives like MySQLi and PDO is the possibility of "Prepared Statements" and "Stored Procedures". As variables of a user interaction often define the data which has to be requested from the MySQL database, there was the possibility of a system harm with a "SQL injection". There the user manipulates the SQL-command with the help of this missing variable. This is done by sending not the predefined input for the variable but an intentional extension, e.g.

- "GET INFORMATION1 FROM \$VARIABLE", where \$VARIABLE can be a predefined database. Usually the system sends, according to the actions of the user, the name of a database.
- If the user would define the \$VARIABLE as "DATABASE1 AND DELETE EVERYTHING", the command would be: "GET INFORMATION1 FROM DATABASE1 AND DELETE EVERYTHING".

⁵³The PHP Group, *PHP.NET Manual*.

⁵⁴Welling and Thomson, *PHP and MySQL*.

- This injection extended the command and changed its underlying purpose - instead of only requesting data, everything would be deleted.

"Prepared Statements" bypass this possibility by preparing the exact SQL command before processing. The missing variable is then, defined by exact syntax, inserted later. This prevents the unauthorized altering of a given SQL command by stating restrictions.

"Stored procedures" go a step further by saving the chain of SQL commands in the database as a function and therefore strictly separate it from the code in the PHP files on the web server. This may be important for big projects where different people are programming the database and the actual PHP code - the PHP programmer can work with predefined functions without knowing the structure of the database.

The PHP Data Objects have another special feature: they are not limited to MySQL databases. If a change of the database is considered or planned one does not have to change the commands written in the PHP code. Only the driver has to be changed according to the preferred database. When using multiple databases or executing frequent changes PDO would be the ideal and most comfortable choice.⁵⁵

Because of the lack of security the selection options are reduced to PDO and MySQLi. When it comes to performance benchmarks showed that MySQL is slightly faster (2.5 - 6.5 %) - if it deals with large amounts of data.⁵⁶

Table 10⁵⁷ outlines the main differences of the APIs.

	original MySQL	MySQLi	PDO
PHP version introduced	2.0	5.0	5.1
Included with PHP 5.x	Yes	Yes	Yes
Development status	Maintenance only	Active	Active
Lifecycle	Deprecated	Active	Active
OOP Interface	No	Yes	Yes
Procedural Interface	Yes	Yes	No
Prepared Statements	No	Yes	Yes
Stored Procedures	No	Yes	Yes
MySQL 5.1+ functionality	No	Yes	Most
Performance	1+	1	1-

Table 10: MySQL, MySQLi and PDO - Comparison

⁵⁵The PHP Group, *PHP.NET Manual*.

⁵⁶tutspius.com, *PDO vs. MySQLi: Which Should You Use?*

⁵⁷The PHP Group, *PHP.NET Manual*.

Considering the fact that the classification system only uses MySQL databases with no intended change in future, MySQLi is the connector of choice. It is faster than PDO, is able to perform all security statements and has the best functionality. Developer recommendations and ongoing development underline the selection.

4.4 The Classification System

This section presents the application to the classification system. It shows a detailed explanation of how the user connects with the system and how the internal processes work.

4.4.1 Client - Server Interaction

To apply the interaction to the system of the muck classification matrix, the individual steps have to be explained in more detail. On the client end of the system there are three different classes of users - the seller, the buyer and the admin interface. Every class consists of several users, depending on the number of people using the platform.

Every user class acts as a system of its own, which means that there is no interaction in the script language, but all of them use the same database. Figure 22 below shows the architecture of this interaction.

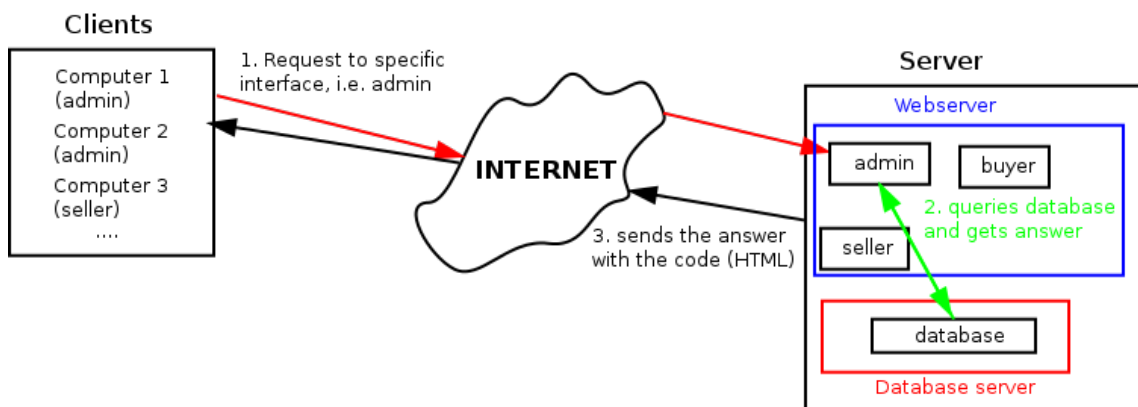


Figure 22: Different User - Database Interaction

On the other end there is the database and the HTTP-server which has the extra feature of interpreting script language codes. The muck classification itself is stored in script files (shown as "admin", "buyer" and "seller" in Figure 22) on the HTTP server and gets called whenever the user on the other end sends a request.

To understand the programming routines, the verbal running simulation, which are a verbal explanation of the connection concepts above, is used. A standardized call can be seen below. Two "interactions" have to be distinguished: requesting and sending data, now called "REQUEST" and "SEND".

1. user REQUESTS/SENDS information

2. server PROCESSES the request
3. not necessarily: server REQUESTS/SENDS data FROM/TO database
4. server SENDS information/confirmation to user
5. user GETS information/confirmation

There are several small steps in between, but to keep the "command" small and clear - using the KISS principle - they are not cited. The excluded steps contain for example:

1. The actual request through the browser. The user is not directly in touch with the information.
2. The HTTP-server gets the request and sends it to the script language handler.
3. If information from a database is needed, the script has to send a request with a database handler to the database.
4. The script language builds the HTML command for the HTTP-server.
5. The browser finally gets the information from the server and displays it.

Comment: Figure will be added!

4.4.2 The Database

As users want to receive and/or save, e.g. their classification or profile data, a special structured database is needed for this purpose. MySQL is a relational database, which means that it includes different tables with a predefined syntax - number of columns and possible datatypes - and a variable number of rows depending on the entered data. Table 11 shows the "User" table, where information about different users is saved.⁵⁸

ID	Username	Password	email	description
<i>integer</i>	<i>character</i>	<i>character</i>	<i>character</i>	<i>text</i>
1	admin	*encrypted*	admin@admin.at	This is the admin account
...

Table 11: Example User table

In the case of the classification matrix tables with the following functions were created:

⁵⁸Oracle Corporation, *MySQL Manual*.

- User. Contains the information about the users.
- Input Parameters. Mostly divided into more tables, like technical, chemical and mineralogical parameters.
- Materials. An easy forward description of the material can be found here, i.e. Name, Cubature, Time, ...
- Management. Comprises all data to ensure the interconnection of the databases and to keep the web page running.
- Logbox. Protocols with information about "who, when and what" is saved here.

The structure of the Materials and Input Parameters table is based on the classification in Section 3. Figure 23 shows the main structure of the database and its interconnections. Rows with attribute "Table" are supposed to be own tables, but are written as normal datatypes to ensure the simplicity of the system.

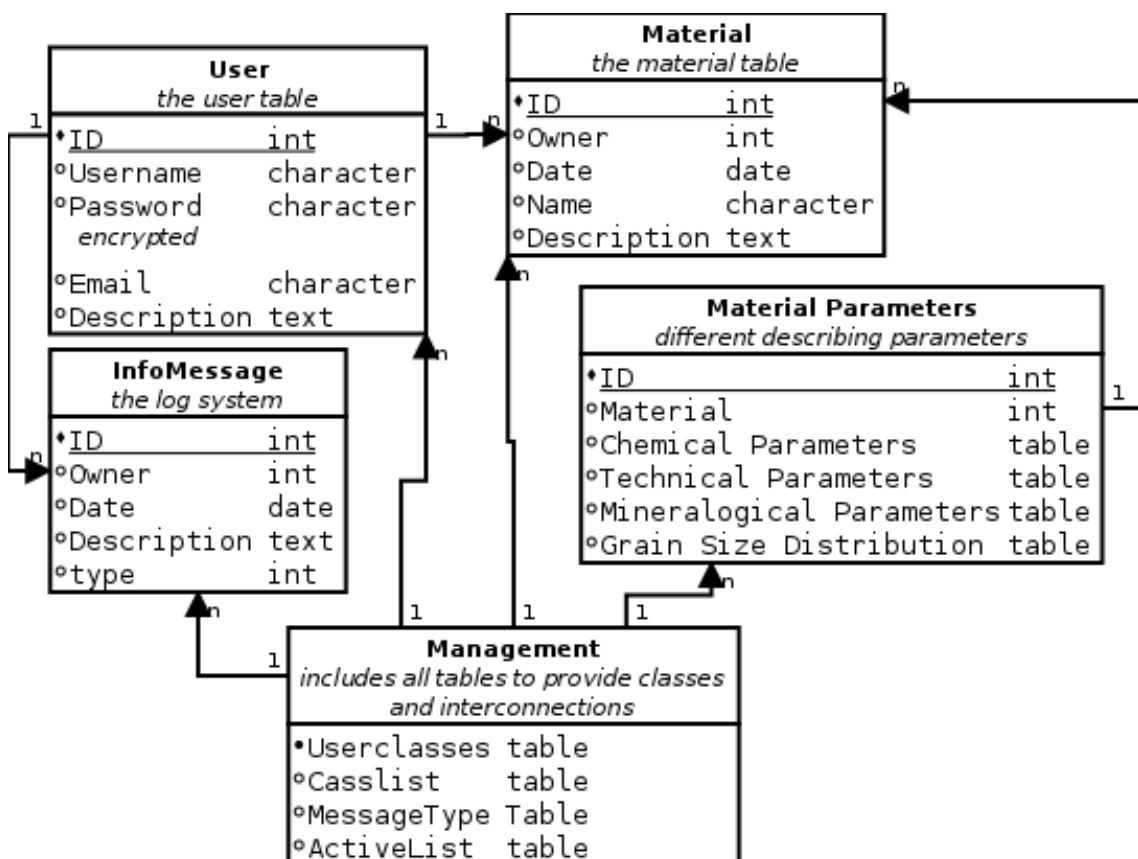


Figure 23: Database structure

The relation between the tables (User, Material, Material Parameters, ...) are meant as an exact assignment to an "upper" level. As the user is able to add a material, the material has to have an "Owner" to know who is allowed to edit a specific material

- a specific user. This fact is described by the "Owner" row of the material - the same applies for the InfoMessage. The MaterialParameters table is connected to the Material itself (the Material table is the "owner").

4.4.3 User Classes and Their Functions

With the help of the verbal running system and the given structure of the database the different functions of the user classes regarding the programming can be discussed.

Because of the fact that the classification is part of the userclass "seller", it is considered to be the most important. The following functions are used in this interface:⁵⁹

- Add / Edit / Activate / Deactivate / Delete Material
- List / Edit Material Properties
- Show output / classification of the material
- Show / Edit profile
- Show Log

Similar to the "seller" interface, but with less functions, the "buyer" can be defined as followed:

- Multi-option search for different materials
- Show properties of the material of interest
- Show / Edit profile

The userclass "admin" has the most rights to interact with the system, since it has the control function of the classification matrix. To help whenever problems for other classes - seller and buyer - appear, the admin interface should be able to see / do what users can do, which means that most functions are combined in this interface.

The following tasks have to be solvable as an admin:

- Admin level: Add / Edit / Activate / Inactivate Users; Show log of all users
- Seller level: Edit / Delete Material; List Material Properties; Show output result / classification of the material

⁵⁹Quakenet/#php.de Staff, *Quakenet/#php Tutorial*.

- User level: Show / Edit profile

The exact interaction of the user and system, according to the mentioned functions above, are discussed in the next section..

4.4.4 The Main Routines

This section explains the different functions of the users in more detail. This is done with the help of the verbal running system. Each function would require many single steps, but, because of their similarity, they are grouped to few "Main Routines":

- The Get Query
- The Add/Update and Delete Query
- The Cookie Activate / Deactivate Query

As a result, one can understand how each function is built according to the running system and therefore the summarized routines. This is shown by an example at the end of this section.

The Get Query To enter a section, i.e.: "Adding Material Section" the user has to request it:

1. seller REQUESTS "Page:Adding Material"
2. server PROCESSES the request
3. server SENDS "Page:Adding Material" to seller
4. seller GETS "Page:Adding Material"

As this is always done when requesting new pages (e.g.: Adding Material, Apply Classification,..), it will not be mentioned in the other routines! If the get-query concerns information, which is stored in a database, the following line has to be added after the PROCESS-command:

server SENDS get "Material:existing materials" TO database: material

The Add/Update and Delete Query Another step - for example after requesting the material page - is sending a complete form and add some information, in this case a material:

1. seller SENDS "Material:form data"

2. server PROCESSES the request
3. server SENDS insert "Material:form data" TO database: material
4. server SENDS "Page:Material Added" to seller
5. seller GETS "Page:Material Added"

The same steps, but "update" and "delete" instead of the "insert" are used when editing or deleting a material.

The Cookie Activate / Deactivate Query For example, to activate a material one has to go to the "Material Section" and choose an already existing material. The properties can be altered, only when it is activated! Therefore two actions have to be processed:

- Requesting the existing materials with the database-GET-query (see above)
- Choosing and saving active material
 1. seller SENDS "Activate Material"
 2. server PROCESSES the request
 3. server SENDS "Cookie: Active Material" to seller
 4. server SENDS "Page:Material Activated" to seller
 5. seller GETS "Page:Material Activated"

When deactivating a material the second part is slightly modified, Line 3 becomes:

- server SENDS "Cookie: No Active Material" to seller

Assembling the functions With a combination of these six queries all tasks of the different user classes can be done. For example, the log-in:

1. A normal Get Query for requesting the page
2. The next step is a database Get Query, where the system controls the login information with the actual written.
3. If the information is ok a COOKIE Activate query is send to save the ID of the User

The actual code behind is often a combination of different sub-procedures, but the idea is the same.

5 Software Engineering Details

After a short introduction of the basics in software engineering and how the interaction between the client and server works, a deeper insight into the actual programming of the classification platform can be given. With the result of the last chapter - the decision for the programming language PHP, the database MySQL and its handler MySQLi - the focus switches then from the interaction of the client-server, with its functions and routines, to a slightly different field. The implementation of the MVC principle with PHP is explained and the functions are reordered and restructured so that the programmed routine can be understood easily. Therefore the "classification" and "login" function is explained in more detail. The structure will be application-oriented; compare Section 3.3.

5.1 The Template System

This section elaborates the Model-View-Controller principle extensively for the Classification matrix system. This is done by an easy implementable template system, compare Section 4.1.1 and 4.1.2.⁶⁰

The server-side file structure strictly separates the template files, which are needed for the optical presentation, and the code files, which contain the programmed routines and therefore the "brain" of the program.

Figure 24 shows the basic folder-structure of the system. The "template" and "img" folder consist of the HTML code and the needed graphics, whereas the code files are represented by the "inc", "includes" and "scripts" folders.

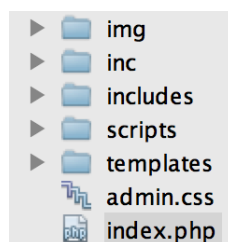


Figure 24: Folder structure

This subdivision is done for organizational purposes as the "inc" folder comprises important constants, variables, functions and information for the output and input parameters of the classification (Figure 25). The "config.php" file loads "constants.php", "functions.php" and "variables.php". This is done for simplification

⁶⁰Quakenet/#php.de Staff, *Quakenet/#php Tutorial*.

as all of them are always needed and loaded together. "constants.php" has exact information about the database (username, password, ..) and defines important constants for program version, which represents the development status, and error messages. The "variables.php" file limits the possible user webpages by defining an exact array of possibilities, e.g. menu, register, profile, parameters, ...

This organizational structure is a standard procedure used in different projects, books⁶¹ and tutorials⁶².

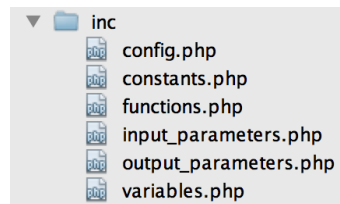


Figure 25: "inc" folder

"input_paramters.php" includes the description of the technical, chemical and mineralogical input parameters with information about their title in the MySQL database.

The different output possibilities (brickearth, cement, ...) are defined in the "output_paramters.php" with specifications of their limits and their title in the MySQL database. This is needed for the comparison of the input and output parameters during the classification.

The "index.php" file is the most important file of the webbased system. It "builds" the webpage by loading the content, which is needed by the user. Below is an excerpt of the "index.php" file including the pertinent command lines.

```
<?php
    // P A R T 1 - BEGIN

    error_reporting(E_ALL);
    ini_set('display_errors', 1);

    include('inc/config.php'); // loading the configuration files

    if(get_magic_quotes_gpc()) {
        array stripslashes($_GET);
        array stripslashes($_POST);
        array stripslashes($_COOKIE);
    }
```

⁶¹Gunnar Thies, *PHP 5.4 und MySQL 5.5*.

⁶²Quakenet/#php.de Staff, *Quakenet/#php Tutorial*.

```
// connection to database
$db = @new MySQLi(MYSQL_HOST, MYSQL_USER, MYSQL_PASS,
    MYSQL_DATABASE);

// P A R T 1 - END

// P A R T 2 - BEGIN

$ret = 1;
if (mysqli_connect_errno()) {
    $ret = 'no connection to database possible, MySQL error: '.
        mysqli_connect_error();
} else if (is_string($error = getUserID($db))) { // String ->
    error!
    $ret = $error; // save error message in $ret to display it
        later.
} else {
    // creation of the include command
    if (!getUserID($db)) {
        $ret = include 'includes/'. $content['login'];
    } else {
        if (isset($_GET['section'], $content[$_GET['section']])) {
            if (file_exists('includes/'. $content[$_GET['section']])) {
                $ret = include 'includes/'. $content[$_GET['section']];
            } else {
                $ret = "could not load the include file: 'includes/" .
                    $content[$_GET['section']] . "'";
            }
        }
    }
} else {
    // loading default area
    $ret = include 'includes/'. $content['menu'];
}
}
}

// P A R T 2 - END

// P A R T 3 - BEGIN

// loading of the html header
include 'templates/html_header.tpl'; // Doctype, <html>, <
    head>, <meta>
include 'templates/html_body_tag.tpl'; // <body> and some
    definitions;
// include 'templates/menu.tpl';
```

```
// loading of the template file
if (is_array($ret) and isset($ret['filename'], $ret['data'])
and
is_string($ret['filename']) and
is_array($ret['data'])) {
// valid template file
if (file_exists($file = 'templates/' . $ret['filename'])) {
    $data = $ret['data']; // save the array data in
        variable $data
                                // which can be used in the template.
    include $file;
} else {
    $data['msg'] = 'the template file "' . $file . '" does not
        exist.';
    include 'templates/error.tpl';
}
} else if (is_string($ret)) {
// got error message
$data['msg'] = $ret;
if(msgToLog($db, 2, $data['msg']))
    $data['msg'] = $data['msg'] . ' the error was added
        to the logsystem!';
else
    $data['msg'] = $data['msg'] . ' the error was not
        added to the logsystem!';
include 'templates/error.tpl';
} else if (1 === $ret) {
// return value forgotten
$data['msg'] = 'the return is missing in the include file.';
if(msgToLog($db, 2, $data['msg']))
    $data['msg'] = $data['msg'] . ' the error was added
        to the logsystem!';
else
    $data['msg'] = $data['msg'] . ' the error was not
        added to the logsystem!';
include 'templates/error.tpl';
} else {
// wrong return value
$data['msg'] = 'the include file processed an invalid value
    .';
if(msgToLog($db, 2, $data['msg']))
    $data['msg'] = $data['msg'] . ' the error was added
        to the logsystem!';
else
    $data['msg'] = $data['msg'] . ' the error was not
```

```
                added to the logsystem!';
            include 'templates/error.tpl';
        }

// load HTML footer
include 'templates/html_footer.tpl'; // includes </body> and
</html> ...

// P A R T 3 - END

?>
```

The first part of the "index.php" file sets the error settings (which errors are shown), loads the config file and removes invalid characters of the GET, COOKIE and POST variable. These variables contain the user input, e.g. given through a form or just by using different menu options. At the end the connection to the MySQL database is built by using the MySQLi handler.

The second part focuses on possible errors and eventually - if no error occurs - builds the command for the needed template file, which should be shown. The considered errors include:

- database connection errors
- errors loading the template file
- all other errors, which are saved as error messages

As an output \$res variable can either be the array that includes the information for loading the needed section, or a string with an error message. If there is no error and no section selected, e.g. "chemical parameters" or "brickearth", then the "menu" section, which represents the main menu, will be loaded. This means if the user requests the section "brickearth" the corresponding php file "brickearth.php" is executed and data, which are needed in the routine, are saved in the \$res variable. So \$res contains:

- the name of the template file - is saved in \$ret['filename']
- data, which were processed in the corresponding php file and are needed in the template file later - is saved in \$ret['data']

The third and last part of the "input.php" file actually builds the webpage. At the beginning the header and the first body are loaded, see "header" and "body" on Figure 26. The header includes the graphics and information on top of the webpage and is always loaded, whether a user is logged in, a section is selected or an error

occurs, see Figure 26. The body part includes the message (log) box, the "active" material and user information - it changes if nobody is logged in or no material is selected.

The figure shows a web page template structure with the following sections:

- header** (red border): "Muck Classification - Sellers Interface" and "Main Menu".
- body** (green border):
 - Logged in as 'Julian' with a Logout link.
 - Version 0.1 © wiedorn.at, 2013.
 - information box** (grey background) containing log entries:

info	2013-11-01 17:00:45	material with name: "Mica" is active!
info	2013-11-01 16:59:39	material is deactivated! a new material can be choosen.
info	2013-10-24 20:56:07	material with name: "Granite" is active!
 - active material** (white background) containing a "deactivate" link and the name "Mica".
 - Info Message** (white background) containing the text "material with name: "Mica" is active! the message was added to the logsystem!" and a "template file include" label.
- footer** (purple border): "Powered by wiedorn.at 0.1, © Julian Wiedorn, 2013, All Rights Reserved".

Figure 26: Template Structure

After that, the \$res variable mentioned before is examined in more detail. The routine checks if all the needed variables are set (\$ret['filename'] and \$ret['data']) and if set - checks the existence of the requested template file. Then - if everything is ok - the content of the section is included (the template file, which shows the needed information), see "template file include" on Figure 26.

The following extensive if-then commands handle the different errors and include, if an error occurred, the error section, which shows the error message, instead of the requested content. All errors are saved into a log system, which should help the user to understand what was done.

That means the content of the \$res variable is examined and therefore can contain:

- the valid definitions of template files and data
- or an error message which was returned by one of the section pages

Figure 27 shows a simplified flow chart of the template system process.

The last command of the "index.php" file includes the footer (see "footer" on Figure 26), which is - similar to the header file - always loaded, but represents the bottom part of the page.

5.2 The MySQL Database Structure

Before the different user functions are discussed, the database structure should be shown in detail. As already mentioned, all the input data, necessary to ensure the

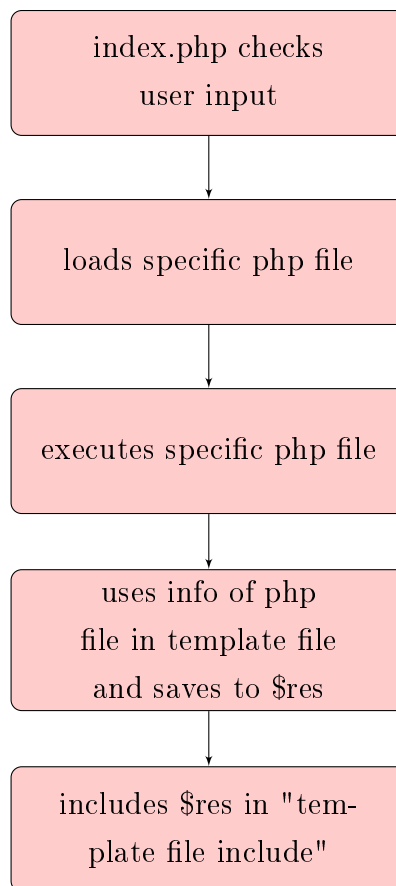


Figure 27: index.php - Flow Chart

functionality of the classification system, are saved here. Section 5.4 (reference!) gives a first introduction about the the database and the structure and should be the basis of the exact explanation in this section.

A total of about 15 tables are needed to enable the classification:

- four tables for the user description
- nine tables for the material description (technical, mineral and chemical parameters)
- two tables for the log system

The tables and their relation can be seen on Figure 28.

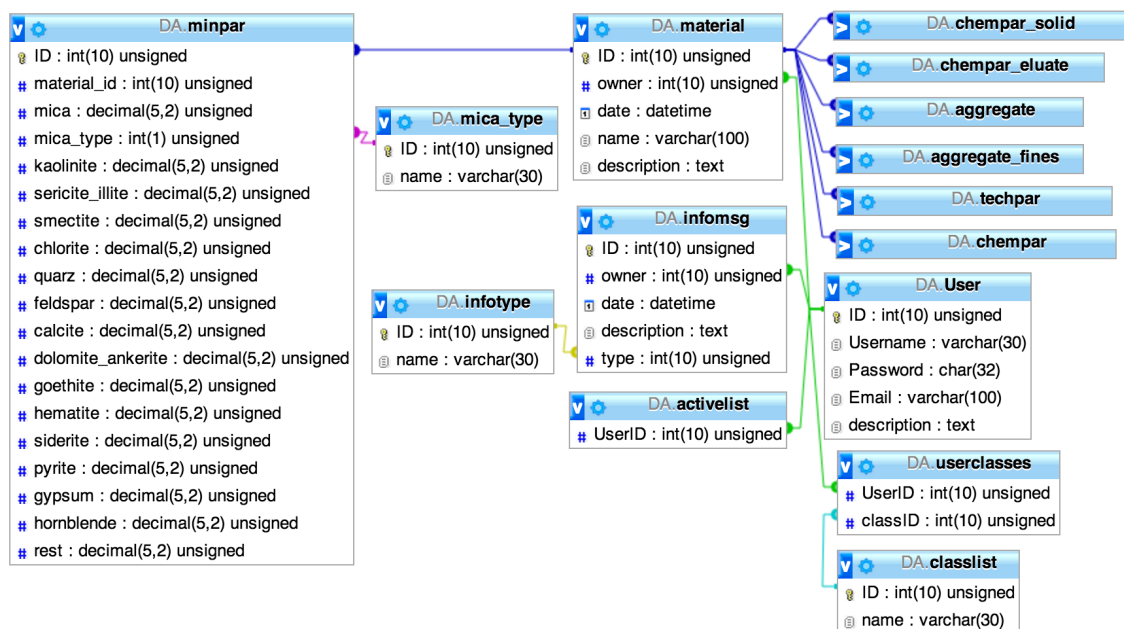


Figure 28: SQL Database Structure

Each blue column and the information listed below it, stands for one table. The written information under the blue line is the exact structure of each table. It states the prerequisites for a data set, which will be saved through user input.

The following information is saved in each table:

- user - contains information about the user, e.g. username, password, email, description
- infomsg - contains the log message with information about the user, who performed the action, the date and the type (normal or error)
- infotype - has information about the different types of log messages; is related to the "infomsg" table

- `activelist` - saves the users, who are allowed to use the system. Therefore The User ID is needed
- `userclasses` - states the "rights" of a user: if the user is admin, seller or/and buyer
- `classlist` - includes the different classes (admin/seller/buyer); is related to user-classes with the ID
- `material` - has the basic information about the material, e.g. owner of the material, date of change, name and short description
- `minpar`, `mica_type`, `chempar_solid`, `chempar_eluate`, `chempar`, `techpar`, `aggregate`, `aggregate_fines` - these table include all the input parameter informations which are needed: technical, chemical and mineralogical parameters (see Section ..)

All tables have at least one additional line: "ID", which is needed to differentiate data sets and also works as an index. The dark blue, light blue ,green, purple and yellow lines in between the tables state the relations (different color - different relation).

The "minpar" table for example, which represents the mineralogical parameters, saves different minerals, e.g. mica and kaolinite, but also has, next to the "ID" line: "material_id". This relations are needed to distinguish every single dataset. The "material_id" relates the mineral properties to a special material (with its material ID), compare dark blue lines on Figure 28. The other input parameter tables (`chempar`, `chempar_solid`, `chempar_eluate`, `techpar`, `aggregate`, `aggregate_fines`) have a similar structure - "ID" and "material_id" but the remaining lines differ because of the input.

The "user" table is the "starting point" of all the relations. It connects the "material", "userclasses", "activelist" and "infomsg" table through the User ID, called "ID" in the "user" table and "owner" or "UserID" in all others (green lines on Figure 28).

5.3 The Section Template and PHP File

Section .. introduces the principle of the template system and mentions the template and PHP files briefly. A detailed explanation of the two file types and their structure are stated here.

The focus is on Section Template files, which exclude the "normal" files, like "header.tpl",

"footer.tpl" - they almost only consist of regular HTML code. The Section Template files, now "template files", and their corresponding PHP code are loaded according to the user actions and are represented in the "template file include" on Figure 26.

The PHP File As the PHP files are executed beforehand - because the evaluated data are needed in the template files - the structure of the standard PHP file is discussed first:

```
<?php
// P A R T 1 - BEGIN

//Security Check - User Login

if (!$UserID = getUserID($db)) {
    return NOT_LOGGED_IN;
}

if(!isset($_COOKIE['MaterialID'])) {
    return NO_MATERIAL;
}

//Definition of the $ret array
$ret = array();
$ret['filename'] = 'material.tpl';
$ret['data'] = array();

// P A R T 1 - END

// P A R T 2 - BEGIN

$sql = 'SELECT
        material.ID,
        material.name,
        User.Username AS Owner,
        material.date,
        material.description
FROM
    material
JOIN
    User
ON
    material.owner = User.ID
WHERE
    material.owner = ?
ORDER BY
```

```

                material.date DESC';
if (!$stmt = $db->prepare($sql)) {
    return $db->error;
}
$stmt->bind_param('i', $UserID);
if (!$stmt->execute()) {
    return $stmt->error;
}
$stmt->bind_result($ID, $name, $owner, $date, $description);
$neu = array();
while ($stmt->fetch()) {
    $materials[] = array('ID' => $ID,
                        'name' => $name,
                        'owner' => $owner,
                        'date' => $date,
                        'description' => $description);
}
$stmt->close();

// P A R T 2 - END

// P A R T 3 - BEGIN

$ret['data']['materials'] = $materials;

return $ret;

// P A R T 3 - BEGIN

?>

```

The PHP file above shows the standard structure and contains an example SQL query. In this case the different materials of the user are evaluated and then saved into an array. At the beginning of the first part if-clauses check if the user is logged in and if a material is selected. The material selection check is not included in every single file as it is only required in further work with a specific material. If one of these errors occurs, an error message will be returned (and saved in the \$ret variable of the "index.php" file).

After positive passing of these checks the \$ret array is initialized by including a data array (\$ret['data'] = array()) and information about the needed template file (\$ret['filename'] = 'material.tpl')

The second part consist of the different routines, which are needed in each section. The \$sql is a string with the material query, in SQL language. By looking at the

query the concept of the above mentioned "Prepared Statements" can be understood easily.

```
$sql = 'SELECT
        material.ID,
        material.name,
        User.Username AS Owner,
        material.date,
                material.description
FROM
        material
JOIN
        User
ON
        material.owner = User.ID
WHERE
        material.owner = ?
ORDER BY
        material.date DESC';
```

The "?" is a space holder for a variable, in this case the information about the material owner. The insertion is done by:

```
$stmt->bind_param('i', $UserID);
```

This gives exact information about where it is inserted and what - i for an integer and \$UserID for the user information. So SQL injections, where "hackers" try to harm the system by changing the SQL command, are not possible.

After that the result is prepared - first the error evaluation - and then executed. A while-loop saves the information from the SQL database to the \$materials array.

The third and last part of the php file copies the \$materials array to the \$ret array, which will be "returned" to the "index.php" file and there "forwarded" to the corresponding template file.

The Template File After evaluating the needed information the template file is loaded to represent the information. Following commands are taken from the "index.php" file:

```
if (file_exists($file = 'templates/'. $ret['filename'])) {
    $data = $ret['data']; // save the array data in variable
    $data
                                // which can be used in the template.
    include $file;
```

After several error checks the template is loaded into the \$file variable and the data of the \$ret array are saved into \$data. The \$data variable then can be used in the template file. Below a standard template file can be seen. After that the template file is included with the \$file variable.

```
<div class="title">Material Information</div><br>
<table align="center" border="0" cellspacing="1" cellpadding="3"
  bgcolor="#cccccc">
  <tbody>
  <tr>
    <td bgcolor="#cccccc" class="normal" background="img/bg.gif"
      "><b>activate material</b></td>
  </tr>
<?php
    foreach ($data['materials'] as $material) {
      echo "<tr>";
      echo '    <td bgcolor="#ffffff" class="
        normal">';
      echo $material['name'];
      echo '</td>';
      echo "</tr>";
    }
  ?>
  </tbody>
</table>
```

The template file mainly consists of HTML code for the optical representation. In this case the evaluated materials from the previous PHP file are shown in a table. A foreach loop loads the data which is saved in the \$data['materials'] array.

INFORMATION PROCESSING: data saved in \$ret['data'] in **php file** → \$data in **template file**

5.4 Pages and their Functions

This section provides information about the different pages and their purpose (functions). As mentioned, because of their importance the "Login"- and "Classification" functions will be described in detail.

5.4.1 Basis

The different functions can be loaded through the menu page (Figure 29) and the menu bar (Figure 30). The menu template is, as explained in Section .., loaded if no other template is requested by the user or can be reached by clicking on the "Main Menu" button on Figure 29 and Figure 30. In contrast to the menu page, the menu bar is part of the "body.tpl" template file, which means it is always loaded and available. By using the links of these menu options the requested section will be loaded. If no user is logged in, the "index.php" file will automatically load the "Login" page.

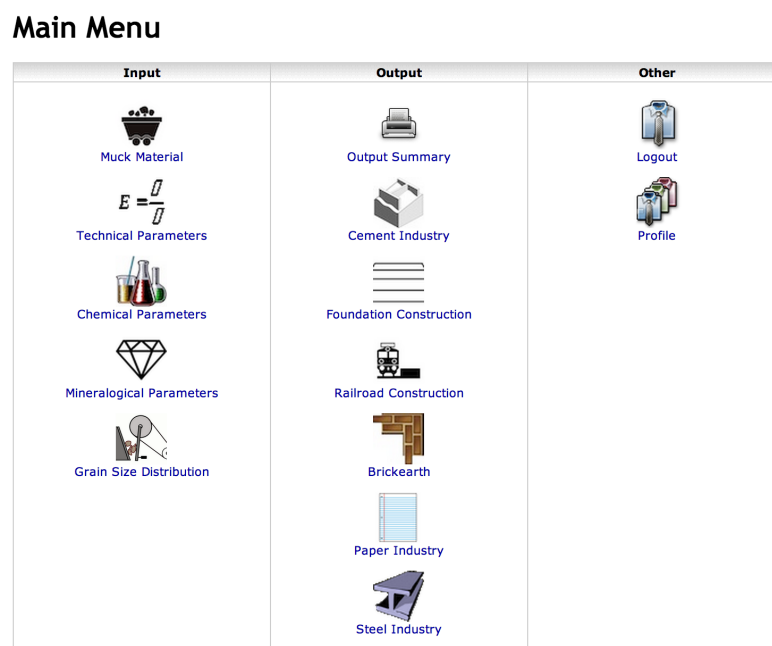


Figure 29: The "Menu" Page

Both, the menu and sidebar, change according to the user interface (admin, seller, buyer) and can be extended easily.

5.4.2 The "Log on" and "Log out" Procedures

The "Login" Check Before any function of the system can be used, the user has to be logged in. This is done by a check in the "index.php" file, compare PART 2 in Section 5.1:

```
if (!getUserID($db)) {
    $ret = include 'includes/'. $content['login'];
}
```

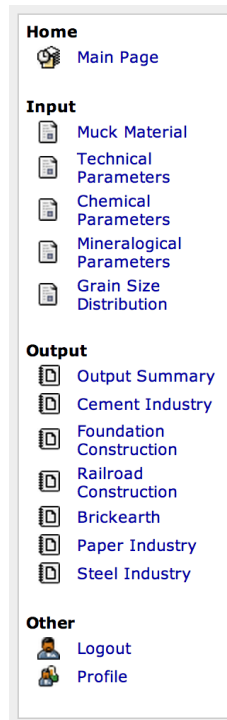


Figure 30: "inc" folder

The `getUserID()` function checks if a user exists and returns the ID of the User when the query was positive and boolean "FALSE" when no user is logged in.

```
function getUserID($db) {

// P A R T 1 - BEGIN

    if (!is_object($db)) {
        return false;
    }
    if (!$db instanceof MySQLi) {
        return false;
    }
    if (!isset($_COOKIE['UserID'], $_COOKIE['Password'])) {
        return false;
    }

// P A R T 1 - END

// P A R T 2 - BEGIN

    $sql = 'SELECT
            ID
            FROM
            User
```

```

        WHERE
            ID = ? AND
            Password = ?';
$stmt = $db->prepare($sql);
if (!$stmt) {
    return $db->error;
}
$stmt->bind_param('is', $_COOKIE['UserID'], $_COOKIE['Password',
    ]);
if (!$stmt->execute()) {
    $str = $stmt->error;
    $stmt->close();
    return $str;
}
$stmt->bind_result($UserID);
if (!$stmt->fetch()) {
    $stmt->close();
    return false;
}
$stmt->close();
return $UserID;

// P A R T 2 - END
}

```

The first part checks the database connection and if the user is already logged in. The COOKIE['UserID'] and COOKIE['Password'] variable should contain the ID and password of the user, if he logged in before - it is set in the "Login" function.

The second part requests the user data from the database and therefore checks, if the set user really exists. If everything is OK the return value will be \$UserID variable otherwise boolean FALSE statement.

The "Login" Page If the result is "FALSE" the "Login" page will be loaded. The php file of the "Login" page is listed below.

```

// P A R T 1 - BEGIN

if (getUserID($db)) {
    return 'You are already logged in.';
}
$ret = array();
$ret['filename'] = 'login.tpl';
$ret['data'] = array();

```

```
// P A R T 1 - END

// P A R T 2 - BEGIN

if ('POST' == $_SERVER['REQUEST_METHOD']) {
    if (!isset($_POST['username'], $_POST['password'], $_POST['formaction'])) {
        return INVALID_FORM;
    }
    if (('' == $Username = trim($_POST['username'])) OR
        ('' == $Password = trim($_POST['password']))) {
        return EMPTY_FORM;
    }
    $sql = 'SELECT
            ID
        FROM
            User
        WHERE
            Username = ?';
    $stmt = $db->prepare($sql);
    if (!$stmt) {
        return $db->error;
    }
    $stmt->bind_param('s', $Username);
    if (!$stmt->execute()) {
        return $stmt->error;
    }
    $stmt->bind_result($UserID);
    if (!$stmt->fetch()) {
        return 'no user with this name found!';
    }
    $stmt->close();
    $sql = 'SELECT
            Password
        FROM
            User
        WHERE
            ID = ? AND
            Password = ?';
    $stmt = $db->prepare($sql);
    if (!$stmt) {
        return $db->error;
    }
    $Hash = md5(md5($UserID).$Password);
    $stmt->bind_param('is', $UserID, $Hash);
    if (!$stmt->execute()) {
```

```

        return $stmt->error;
    }
    $stmt->bind_result($Hash);
    if (!$stmt->fetch()) {
        return 'password is not ok!';
    }
    $stmt->close();

    // P A R T 2 - END

    // P A R T 3 - BEGIN

    setcookie('UserID', $UserID, strtotime("+1 month"));
    setcookie('Password', $Hash, strtotime("+1 month"));
    setcookie('Username', $Username, strtotime("+1 month"));
    $_COOKIE['UserID'] = $UserID; // fake-cookie
    $_COOKIE['Password'] = $Hash; // fake-cookie
    $_COOKIE['Username'] = $Username; // fake-cookie
    return showInfo('you are logged in!', $db);

}
return $ret;

// P A R T 3 - END

```

The first part consists of a login check - in this case an error occurs if the user is already logged in; loading the "Login" page would not make sense - and of the standard array definitions, which can be found in every php file, see Section 5.3.

In the second part, several if-clauses check if

- the user filled the forms of the "Login" template page completely (username and password).
- a user with that name exists.
- the written password is correct.

If some of those checks are not ok, an error occurs and the procedure ends by returning the error message in the \$ret variable.

In the last part the before-mentioned cookies are set (\$_COOKIE['UserID'], \$_COOKIE['Password'] and \$_COOKIE['Username']) with a time validity of 1 month. Because of the fact that a cookie is saved on the clients computer, which means that it has to be at first sent and then can be recalled, a fake-cookie is set. Therefore the cookies can be used immediately without waiting for the save and

request procedure, which usually waits until the user requests another page.

The optical representation of the template file of the "Login" section consists of an input form, see Figure 31.

Login

login form	
username	<input type="text"/>
password	<input type="password"/>
login	

Figure 31: The "Login" Template

The code of the template file contains exclusively HTML commands to represent the table "input form" on Figure 31.

The "Logout" Procedure To end the connection with the classification platform the user has to log out, which means that the system itself somehow has to recognize if the user is still working or not. This is done by the cookies, which were set during the login procedure. So, to perform the logout, the cookies have to be unset.

```
// P A R T 1 - BEGIN

if (!getUserID($db)) {
    return NOT_LOGGED_IN;
}
$ret = array();
$ret['filename'] = 'logout.tpl';
$ret['data'] = array();

// P A R T 1 - END

// P A R T 2 - BEGIN

if ('POST' == $_SERVER['REQUEST_METHOD']) {
    if (!isset($_POST['formaction'])) {
        return INVALID_FORM;
    }
    setcookie('UserID', '', strtotime('-1 day'));
    setcookie('Password', '', strtotime('-1 day'));
    setcookie('Username', '', strtotime('-1 day'));
    setcookie('MaterialID', '', strtotime('-1 day'));
    unset($_COOKIE['UserID']);
```

```

unset($_COOKIE['Password']);
unset($_COOKIE['Username']);
unset($_COOKIE['MaterialID']);
return showInfo('you are logged out!', $db);
}
return $ret;

// P A R T 2 - E N D

```

Part one of the PHP file contains a login check and the standard \$ret definitions. After that, an if-clause ensures that the form of the corresponding template file has been used. Finally the cookies are unset by setting a negative time period and unset the current, by the system used, variables.

The "logout.tpl" file only consists of a simple button to initiate the unset procedure of the PHP file.

These registration conditions ("Log on" and "Log out") act as the basic security functions and are part of all interfaces.

5.4.3 The "Classification"

This section provides a detailed description of the classification system and how it works. The basic procedure can be reviewed in Section 3. The classification is divided into an "Input" and "Output" part; see Figure 32. All the selection options are different pages and therefore consist of the template and PHP files.




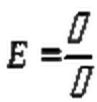






Input	Output - Intermediate	Output - End
 Muck Material	 Output Summary	 Output Summary
 Technical Parameters	 Carbonates	 Cement Industry
 Chemical Parameters		 Foundation Construction
		

Figure 32: Main Menu - Input / Output Categories

To describe the programmed routine only one specific input and output option is chosen, as the principle itself stays unchanged.

The input part is divided into following pages, which are needed to specify all input parameters:

- Muck Material - definition of material name and description
- Technical Parameters - input for technical parameters, e.g. elastic modulus, wear resistance, compressive strength ...
- Chemical Parameters - chemical composition of the eluate and solid part of the muck material
- Mineralogical Parameters - mineralogical composition

The exact input, which can be found on each page, can be taken from Section ...

The reuse is divided into end products and intermediate products (Figure 32). End products are those raw materials with high amounts of special components - the last part of the raw material value chain. These products are usually already processed and therefore adjusted to the requirements of the industry. Intermediate products, however, are those products that can be found in the middle part of the value chain and have a wide range of use. In contrast to end products their classification criteria are not industry specific, they are named after the main component of the raw material, e.g. a limestone, which can be used in different industrial fields.

Each output part mainly distinguishes between single output options and the summary output ("Output Summary"). The latter part represents an abridged version of the other output options with less information provided. It sends a list with the name of the output criteria and a "usable"/"not usable" statement (Figure 33).

Output Summary

Classification		
category	suggested raw material	result
<i>Brickearth</i>	Wall bricks	useable
<i>Steel Industry</i>	Bauxite as Slagformer	not useable
<i>Steel Industry</i>	Olivine for Steel Industry	not useable
<i>Steel Industry</i>	Olivine for Foundry Industry	not useable
<i>Steel Industry</i>	Fluorite for metallurgical grade fluorite	not useable
<i>Steel Industry</i>	Limestone for metallurgy	not useable
<i>Steel Industry</i>	Dolomite (uncalcined) for pig iron (direct use)	not useable
<i>Steel Industry</i>	Dolomite (uncalcined) for Steelproduction	not useable
<i>Steel Industry</i>	Dolomite (calcined) for refractory industry	not useable
<i>Steel Industry</i>	Dolomite (calcined) for Steelproduction	not useable
<i>Steel Industry</i>	Magnesite (calcined) for Transformersteelcoating	not useable
<i>Steel Industry</i>	Magnesite (calcined) for Steelindustry	not useable
<i>Cement Industry</i>	Clay Cement	not useable

Figure 33: Summary of Output Criteria

By using a specific output criteria the actual values of the input parameters and their output limits can be evaluated (Figure 34).

Cement Industry

Clay Cement

Classification			
parameter	unit	measured value	limits
SiO ₂	M-%	55.00	2 - 6
Al ₂ O ₃	M-%	15.00	45 - 55
Fe ₂ O ₃	M-%	5.00	10 - 30
TiO ₂	M-%	1.00	2.5 - 3
CaO	M-%	2.00	0.5 - 3
not useable as Clay Cement			

Figure 34: Output Cement Industry

Figure 32 shows "Cement Industry" as specific output with detailed information on the raw material, whereas Figure 33 shows the output summary.

The Input As an example for the input of the classification, the "Mineralogical Parameters" section was chosen, see Figure ...

First already saved parameters have to be requested from the SQL database and then returned with the \$ret variable to use it in the template file for visual presentation:

```

// P A R T 1 - BEGIN

if (!$UserID = getUserID($db)) {
    return NOT_LOGGED_IN;
}
if(!isset($_COOKIE['MaterialID'])){
    return NO_MATERIAL;
}
$ret = array();
$ret['filename'] = 'mineralogical_parameters.tpl';
$ret['data'] = array();

// P A R T 1 - END

// P A R T 2 - BEGIN

include('inc/input_parameters.php');
$ret['data']['mp']=$mineralogical_parameters;

```

```

// P A R T 2 - END

// P A R T 3 - BEGIN

//get data from mysql database
foreach ($ret['data']['mp'] as $key => $mp) {
$sql = 'SELECT
        minpar.'. $mp['sql']. '
      FROM
        minpar
      WHERE
        material_id = ?';

$stmt = $db->prepare($sql);
if (!$stmt) {
    return $db->error;
}
$stmt->bind_param('i', $_COOKIE['MaterialID']);
if (!$stmt->execute()) {
    return $stmt->error;
}
$stmt->bind_result($erg);
$stmt->fetch();
$stmt->close();
$ret['data']['mp'][$key]['value'] = $erg;
}

// P A R T 3 - END

```

Part 1 contains the usual checks and array initialization with specification of the template file. In Part 2, the input parameters are loaded with the `$input_parameters` array. They are loaded from "input_parameters.php", which is saved in the "inc" folder; see Section 5.1. To get an idea about the array, which is saved in `$ret['data']['mp']`, following code shows an excerpt of `$input_parameters`:

```

$mineralogical_parameters = array(
    'mica' => array(
        'name' => 'Mica',
        'unit' => 'M-%',
        'sql' => 'mica'
    ),
    'mica_type' => array(
        'name' => 'type of mica',
        'unit' => 'M-%',
        'sql' => 'mica_type'
    )
);

```

```

    ),

    'kaolinite' => array(
        'name' => 'Kaolinite',
        'unit' => 'M-%',
        'sql' => 'kaolinite'
    ),

    'sericite_illite' => array(
        'name' => 'Sericite Illite',
        'unit' => 'M-%',
        'sql' => 'sericite_illite'
    ),

    ...

    'other' => array(
        'name' => 'Other',
        'unit' => 'M-%',
        'sql' => 'rest'
    )
);

```

Each element of the `$input_parameters` array is an array itself and consists of the name of the input parameters (which are shown in the template), its unit (also shown in the template) and its name in the MySQL database (index "sql" in the array), which is needed for saving the user input.

Part 3 requests the data from the "minpar" table of the MySQL database by using the 'sql' index with `$mp['sql']`. This is done for each mineral parameter in the array and the requested value (the saved mineral parameter) is then added to the `$ret` array with `$ret['data']['mp'][$key]['value'] = $erg` as the last command of part 3. `$erg` contains the value of the mineral parameters and `$key` is the mineral parameter itself, e.g. "kaolinite".

Because of the fact that the `$ret` array is returned and then "forwarded" to the template file (standard procedure for php-template files as explained in Section ..) with `$data=$ret['data']`, all the requested values can easily be recalled with `$data['mp']['MINERAL']['value']` in the template file (MINERAL is a space holder for one of the mineral nominations of the `$input_parameters` array, e.g. "kaolinite").

Mineralogical Parameters

parameter list		
Mica	M-%	<input type="text" value="10.00"/>
type of mica	M-%	<input type="text" value="2"/>
Kaolinite	M-%	<input type="text" value="20.00"/>
Sericite Illite	M-%	<input type="text" value="15.00"/>

Figure 35: Template - Mineralogical Parameters

Figure 35 shows the upper part of the representation of the mineralogical parameters template file.

The actual code behind is listed below:

```

<div class="title">Mineralogical Parameters</div><br>
<form method="POST" action="index.php?section=
  mineralogical_parameters">
<table align="center" border="0" cellspacing="1" cellpadding="3"
  bgcolor="#cccccc">
  <tbody>
  <tr>
    <td colspan="3" bgcolor="#cccccc" class="normal" background
      = "img/bg.gif"><b>parameter list</b></td>
  </tr>
  <?php

  // IMPORTANT - B E G I N

  foreach ($data['mp'] as $mp) {
    echo '<tr>';
    echo '  <td bgcolor="#ffffff" class="normal">'. $mp['name'].
      '</td>';
    echo '  <td bgcolor="#ffffff" class="normal">'. $mp['unit'].
      '</td>';
    echo '  <td bgcolor="#ffffff"><input type="text" name="mp['
      . $mp['sql'] . ']" class="text" value="'. htmlspecialchars(
        $mp['value'] ). '></td>';
    echo '</tr>';
  }

  // IMPORTANT - E N D

  ?>

  <tr>
    <td colspan="3" bgcolor="#ffffff" align="center"><input

```

```

        type="submit" name="formaction" value="save
        mineralogical parameters" class="button"></td>
    </tr>
</tbody></table>
</form>

```

As usual, most of the template file consist of standard HTML code, except the marked part, which "builds" the table of mineralogical input parameters (Figure ..) with a foreach loop in PHP. The \$data array, as already mentioned, consist of the information of the PHP file (\$ret['data']).

The

```
$data['mp'] as $mp
```

part of the foreach loop takes all mineralogical parameters, e.g. "kaolinite", successively, which means that \$mp now represents every single mineral (instead of \$data['mp']['MINERAL'], with MINERAL as space holder).

The values are plotted as input fields to make user input possible. For each mineral parameter the input is then saved as \$mp[\$mp['sql']], where \$mp['sql'] represents the MySQL name of the mineral, again e.g. "kaolinite". By pressing the "save mineralogical parameters" button (Figure 36) the PHP file is again executed, but with additional information because of the button-action.

Gypsum	M-%	<input type="text" value="0.00"/>
Hornblende	M-%	<input type="text" value="0.00"/>
Other	M-%	<input type="text" value="0.00"/>
<input type="button" value="save mineralogical parameters"/>		

Figure 36: Template - Save Mineralogical Parameters

Input given through a form is always saved under the given name but in another array, the \$_POST array. This means, the given input can be recalled in the php file with \$_POST['mp'] - the index 'mp' because of the \$mp declaration before. \$mp[\$mp['sql']] then becomes \$_POST['mp'][\$mp['sql']], which can be seen in the second part of the php file:

```

foreach ($ret['data']['mp'] as $mp) {
    $sql = 'UPDATE
        minpar
        SET

```

```

        minpar.'. $mp['sql']. ' = ?
    WHERE
        material_id = ?';

$stmt = $db->prepare($sql);
if (!$stmt) {
    return $db->error;
}
$stmt->bind_param('di', $_POST['mp'][$mp['sql']], $_COOKIE['
    MaterialID']);
if (!$stmt->execute()) {
    return $stmt->error;
}
}
return showInfo('the material was saved.', $db);

```

The foreach loop does exactly what was described before: it uses the mineral MySQL nomination to save the given user input (`$_POST['mp'][$mp['sql']]`) successively - each pass of the loop updates one parameter with a SQL query saved in `$sql`. After the execute command the loop starts again until the last parameter is updated.

The principle of this input applies for all user inputs, whether other parameters, the user profiles or similar objects are changed/updated.

The Output - Classification As mentioned before the output mainly differentiates between the "Output Summary" and the regular specific output, i.e. "Carbonates" or "Cement Industry" (Figure ..).

To perform the classification two functions were written, which are saved in the "inc" folder within the "functions.php" file. The `get_classification()` function is needed for the specific output:

```

function get_classification($db, $output_parameters, $name) {

// PART 1 - B E G I N

    include('inc/input_parameters.php');
    $data['name']=$name;
    $data['output']=$output_parameters;

    $data['useable']=TRUE;
    foreach ($data['output'] as $key => $op) {
        if(searchForId($op['sql'], $chemical_parameters_compound))
            $data['output'][$key]['database']='chempar';
    }
}

```

```

else if(searchForId($op['sql'], $chemical_parameters_other)
)
    $data['output'][$key]['database']='chempar';
else if(searchForId($op['sql'],
    $chemical_parameters_solids_anorganic))
    $data['output'][$key]['database']='chempar_solid';
else if(searchForId($op['sql'],
    $chemical_parameters_solids_organic))
    $data['output'][$key]['database']='chempar_solid';

// PART 1 - E N D

// PART 2 - B E G I N

$sql = 'SELECT
        '.$data['output'][$key]['database'].'.'.$op['sql'].'
        ,
        FROM
        '.$data['output'][$key]['database'].'
        WHERE
        material_id = ?';

$stmt = $db->prepare($sql);
if (!$stmt) {
    return $db->error;
}
$stmt->bind_param('i', $_COOKIE['MaterialID']);
if (!$stmt->execute()) {
    return $stmt->error;
}
$stmt->bind_result($erg);
$stmt->fetch();
$stmt->close();

$data['output'][$key]['value'] = $erg;
if(($data['output'][$key]['value'] <= $data['output'][$key]
    ['ul']) AND
    ($data['output'][$key]['value'] >= $data['output'][$key]
    ['ll']))
    $data['output'][$key]['classification']=TRUE;
else {
    $data['output'][$key]['classification']=FALSE;
    $data['useable']=FALSE;
}
}
}

```

```

    return $data;
}

// PART 2 - E N D

```

The function needs three parameters to be executed:

- \$db - the database connection for the SQL query
- \$output_parameters - an array with all the limits for a specific output
- \$name - the name of the output parameter which has to be tested

Similar to the input parameters the specific output parameters can be found in the output_parameters.php file. A typical array of this file can be seen below.

```

$clay_cement = array(
    'sio2' => array(
        'name' => 'Si2',
        'sql' => 'sio2',
        'ul' => 6,
        'll' => 2
    ),

    'al2o3' => array(
        'name' => 'Al2O3',
        'sql' => 'al2o3',
        'ul' => 55,
        'll' => 45
    ),

    ...

    'cao' => array(
        'name' => 'CaO',
        'sql' => 'cao',
        'ul' => 3,
        'll' => 0.5
    )
);

```

Again, it is an array consisting of an array with the input parameters, e.g. 'sio2', their limits and their database name. This information is then used in the classification function.

Part 1 of the get_classification() function loads all the prerequisites, that are needed for the classification. It mainly consists of a "search" part, where the routine finds

the SQL table in which the specific parameters can be found - chempar, minpar and so on - this is why the `input_parameters.php` file is included. The `searchForId()` function returns the boolean value `TRUE` or `FALSE`, whether the parameter was found in an array or not. The table information is then saved to each parameter next to their limits - the array now consists of the name, limits, sql nomination and table information and is called `$data['output']`.

The `foreach` loop of the second part takes each output parameter (called `$op`) and loads the homonymous user input parameters to `$data['output'][$key]['value']`, where `$key` is the name of input / output parameters.

After that an `if`-clause controls the limits and saves `TRUE` or `FALSE` to `$data['output'][$key]['classification']`, whether it is in between the limits or not - for each output parameter. So the following information is obtained for every single parameters (and saved in the array):

- name of the output parameters
- the limits
- the SQL name
- the name of the table
- if it is in between the limits or not

As addition `$data['useable']` is initiated, which is needed for the "Output Summary". It is `FALSE` if ONE of the parameters is not in the limits!

The information in the `$data` array then will be returned to the `php` file, where it was executed.

As an example for a specific output the `PHP` file for "Cement Industry" (`cement.php`) is chosen, which contains, next to the standard definitions:

```
...  
  
include('inc/output_parameters.php');  
  
//Clay Cement  
$classification=get_classification($db, $clay_cement, 'Clay  
Cement');  
$ret['data']['cc']=$classification;  
  
...
```

```

//Gypsum Anhydrite
$classification=get_classification($db, $gypsum_anhydrite, '
    Gypsum Anhydrite');
$ret['data']['ga']=$classification;

return $ret;

```

The result of the `get_classification()` function is saved to `$classification` and then in `$ret['data']`. In this file more special output criteria are listed, e.g. "Clay Cement" and "Gypsum Anhydrite", both saved in the same array, which is then returned and used in the template file for optical representation.

INFORMATION PROCESSING: `$data` in the `get_classification()` function → `$classification` and then `$ret['data']` in the **cement php file** → `$data` in the **cement template file**

Figure 34 shows the template file. As one can see, the information, which was evaluated in the `get_classification()`, is listed in table form.

The "Output Summary" page used the another function to obtain a less detailed list of all possible options; see Figure 33.

```

function fast_classification($db, $output_parameters, $product) {

// PART 1 - B E G I N

    if ($product == 'end') include('inc/output_parameters.php');
    else if($product == 'intermediate') include('inc/
        output_parameters_intermediate.php');

    $data['output']=$output_parameters;

// PART 1 - E N D

// PART 2 - B E G I N

    foreach ($data['output'] as $key => $output) {
        //eval('return '. $output['array'] . ';'');
        $data['output'][$key]['useable']=FALSE;
        $result=get_classification($db, eval('return '. $output['
            array'] . ';''), $output['name']);
        if($result['useable']) $data['output'][$key]['useable']=
            TRUE;
    }
}

```

```

    }
    return $data;
}

// PART 2 - E N D

```

Similar to the `get_classification()` function `fast_classification()` uses three arguments:

- `$db` - the database connection for the SQL query
- `$output_parameters` - an array - different to `get_classification()` - with information about all the specific output criteria
- `$product` - for distinction between an output summary for "Intermediate" or "End Products"

The first part loads the special output parameter for either intermediate or end products, depending on how the function was executed - with `$product`. Then the basic `$output_parameters` array, which is equivalent to `$fast_classification` in the `output_parameters.php` file, is saved into `$data['output']`:

```

$fast_classification= array(
...

    'clay_cement' => array(
        'name' => 'Clay Cement',
        'category' => 'Cement Industry',
        'array' => '$clay_cement'
    ),
...

    'magnesite' => array(
        'name' => 'Magnesite',
        'category' => 'Paper Industry',
        'array' => '$magnesite'
    )
);

```

`$fast_classification` is an array including information about each special output criterion (name, category and how it is called in the `output_parameters.php` file).

In the second part a `foreach` loop uses this information and "feeds" the `get_classification()`

function to evaluate if each special output, e.g. "Clay Cement" is usable or not. But - instead of saving all detailed information in the \$data array, which will be then returned - it only saves TRUE or FALSE in a new array field. After that the \$data array includes following information of each special output (e.g. "Clay Cement"):

- name of the output parameters, e.g. "Clay Cement"
- the category, e.g. "Cement Industry"
- the name of the array in the output_parameters.php file
- if it is usable or not

The php file of the "Output Summary" page then just uses few commands:

```
...  
  
include('inc/output_parameters.php');  
  
//Fast Classification  
$classification=fast_classification($db, $fast_classification,  
    'end');  
$ret['data']=$classification;  
  
return $ret;
```

The fast_classification() function is called by "giving" the \$fast_classification array from the output_parameters.php file. The result is then saved to \$ret['data'], which then can be used in the template file.

The optical representation of the template file (Figure 33) uses the information saved in the given array and presents the information in a table.

6 Example Classification

Different raw material based on real excavation material of a current tunneling project, the "Bossler Filder" tunnel in Baden-Württemberg, Germany, is taken. Its chemical properties are analyzed and entered in the muck classification system. Three different groups of raw material are distinguished by a geologist before further tests are done:

- limestone
- sandstone
- claystone/claymarl

In respect of this partition five different raw material analyzes were carried out: two limestone, three sandstone and one claystone/claymarl test objects. Tables 12, 13 and 14 show their chemical composition.

The materials are referenced in the muck classification system as:

- Limestone_1 and Limestone_2
- Sandstone_1 to Sandstone_3
- Claymarl_1

Their description includes the given short reference, i.e. "7058-1 SST-F" for Sandstone_1, and their analysis data; compare Table 12 to 14. In this example, the "Chemical Parameters" section of the system is edited, because the material was only subjected to a chemical analysis.

The classification considers all use criteria, which are stated in Section 2.1.2. One of the stated tunnel excavation material immediately fits to one reuse scenarios. The raw material "Limestone_1" is useable as Limestone regarding the stated limits; compare the Output Summary on Figure 37.

To see the details of the evaluation and the exact limits of the material, the "Carbonates" category is used; compare Figure 38.

Therefore, a simple analysis for the other tunnel excavation materials, why it is not considered for a special use criteria, can be accomplished using the detailed output. With information about the problematic values (comparing limits to the actual values) various processing methods can be considered.

Chemical Composition	Limestone 1 7056-1 KST-H	Limestone 2 7057-1 KST-D
	g/100g	g/100g
Na ₂ O	<0.01	<0.01
MgO	1,71	0,79
Al ₂ O ₃	0,84	0,71
SiO ₂	4,25	3,27
P ₂ O ₅	0,04	0,03
SO ₃	0,08	0,16
K ₂ O	0,33	0,28
CaO	50,88	51,25
TiO ₂	0,04	0,03
Cr ₂ O ₃	<0.01	<0.01
MnO	0,04	0,03
Fe ₂ O ₃	0,34	0,27
LOI	41,56	41,85
sum	99,13	98,67
	mg/kg	mg/kg
Sc	<10	<10
V	13	<10
Cr	11	<10
Co	<10	<10
Ni	<10	<10
Cu	20	29
Zn	<10	<10
Ga	<10	<10
Rb	12	15
Sr	117	129
Y	<10	<10
Zr	13	<10
Nb	<10	<10
Ba	29	16
La	11	10
Ce	11	<10
Pb	<10	<10
Th	<10	10

Table 12: Chemical Composition of tested Limestone

Chemical Composition	Sandstone 1 7058-1 SST-F	Sandstone 2 7061-1 SST-B	Sandstone 3 7060-1 ALPHA-H
	g/100g	g/100g	g/100g
Na ₂ O	0,6	1,39	<0.01
MgO	4,33	8,11	0,1
Al ₂ O ₃	14,49	12,82	2,71
SiO ₂	50,61	46,75	88,98
P ₂ O ₅	0,059	0,063	0,023
SO ₃	0,12	0,08	0,02
K ₂ O	3,55	2,92	0,21
CaO	4,34	11,78	0,08
TiO ₂	0,36	0,17	0,41
Cr ₂ O ₃	<0.01	<0.01	<0.01
MnO	0,07	0,13	0,03
Fe ₂ O ₃	1,53	1,03	3,3
LOI	19,78	14,88	3,63
sum	99,84	100,12	99,49
	mg/kg	mg/kg	mg/kg
Sc	<10	<10	<10
V	64	49	78
Cr	40	20	43
Co	<10	<10	<10
Ni	13	<10	<10
Cu	44	102	32
Zn	57	54	23
Ga	13	<10	<10
Rb	122	98	<10
Sr	223	212	83
Y	16	18	20
Zr	256	174	337
Nb	<10	<10	<10
Ba	2040	967	517
La	34	24	18
Ce	27	54	62
Pb	22	21	<10
Th	<10	<10	<10

Table 13: Chemical Composition of tested Sandstone

Chemical Composition	Sandstone 1 7058-1 SST-F
	g/100g
Na ₂ O	0,06
MgO	0,87
Al ₂ O ₃	20,03
SiO ₂	60,95
P ₂ O ₅	0,076
SO ₃	0,29
K ₂ O	2,54
CaO	0,69
TiO ₂	1,57
Cr ₂ O ₃	0,026
MnO	0,01
Fe ₂ O ₃	2,79
LOI	9,82
sum	99,72
	mg/kg
Sc	11
V	307
Cr	177
Co	14
Ni	29
Cu	53
Zn	32
Ga	24
Rb	111
Sr	212
Y	37
Zr	695
Nb	31
Ba	275
La	63
Ce	146
Pb	19
Th	12

Table 14: Chemical Composition of tested Clay/Marl

Output Summary - Intermediate Product

Classification		
category	suggested raw material	result
Carbonates	Limestone	useable
Carbonates	Dolomite	not useable
Carbonates	Spat-Magnesite	not useable
Carbonates	Gel-Magnesite	not useable
Carbonates	Magnesite (RHI)	not useable

Figure 37: Interface - Output Summary Limestone_1

Carbonates

Limestone

Classification			
parameter	unit	measured value	limits
SiO ₂	M-%	4.25	1.3 - 11.4
Al ₂ O ₃	M-%	0.84	0.7 - 3.1
Fe ₂ O ₃	M-%	0.34	0.2 - 1.3
CaO	M-%	50.88	43.4 - 96.5
MgO	M-%	1.71	1.7 - 3.3
K ₂ O	M-%	0.33	0 - 0.8
CaCO ₃	M-%	86.00	85 - 98.5
SO ₃	M-%	0.08	0 - 0.5
Loss on Ignition	M-%	41.56	40 - 45
useable as Limestone			

Figure 38: Interface - Detail Limestone_1

7 Summary and Outlook

The implementation of a classification system as a platform independent solution for reuse of tunnel excavation material satisfies stated requirements: the simple access provided by a regular browser, the programmed PHP routines and the MySQL data storage give the user a straightforward and simple system to find different fields of reuse, without thinking about all the possible background process.

In addition, the system provides the user with a material management system, which can be saved and edited all the time. Following basic routines can be performed by the user:

- basic material management (add/edit/delete raw materials)
- advanced material management (add/edit/delete chemical/mineralogical/technical properties of materials)
- logging of actions (interactions with the system)
- user profile management

The administration of the registered clients (users) is done by an own interface, which consists of following tasks:

- user management (add/edit/delete raw materials)
- right management (specify user level for registered users - deny/grant access)
- user support with a log system
- restricted material management

An useful extension to "finish" and round off the system would be a third interface - a possibility for clients to seek and find raw materials. A simple search engine, which then establishes a contact between "sellers" and "buyers", would fill this gap. The user could sent material requests to the system and receive information as soon as it is available according to his own conditions (distance, content, ...).

The currently available functions show that an extension to an online analysis of tunnel excavation material is useful and easy to implement. An on-site analysis would make the material immediately available, without time delays caused by manual input. As a result reuse scenarios could be evaluated and followed as soon as the material is excavated and analyzed by measuring devices. The real challenge in this case is the development of such a measuring system, as it is not available yet.

Another important point of discussion is the accuracy of such a classification system.⁶³ The simple handling prevents the consideration of unpredictable fluctuations concerning the properties of the material. All the entered properties are absolute values which result in a deterministic evaluation. Interesting improvements would include the implementation of probabilities of different descriptive characteristics. The system then shows the user the possible reuse together with a simple distribution curve.

Such an extension reduces its main advantage - the simple usability - and increases the costs for the evaluation of material properties significantly. Instead of distribution functions for every parameter the selection should be limited to the most important deciding factors and its extreme deviations.

In what extent an increase of accuracy justifies the reduction of operability should be matched to the demand and wishes of the user.

Future developments will show the acceptance of such a classification system. Changes concerning the law of “waste material” as main topic could be game-changing. However, a wide range of possibilities and its convincing advantages promise an interesting development in this area.

⁶³Zarai, Uromeihy, and Sharifzadeh, “A new tunnel inflow classification (TIC) system through sedimentary rock masses”.

References

- Ayaydin, Nejad. “Classification of Excavation with Austrian Code B2203: Main aspects and experiences”. In: *Elsevier: Tunneling and Underground Space Technology* (2012).
- Date, C.J. *An Introduction to Database Systems*. Pearson, 2003.
- DuBois, Paul. *MySQL*. Developers Library, 2013.
- Entacher, M., D. Resch, and R. Galler. “Abfall oder Rohstoff? Rechtsgrundlagen für die Wiederverwertung von Tunnelausbruchmaterial”. In: *Elsevier* (2012).
- “Recycling of tunnel spoil – laws affecting waste from mining and tunnelling”. In: *Wiley Geomechanik und Tunnelbau* (2011).
- Erben, Hartmut and Robert Galler. “Ressourceneffizienz im Tunnelbau – On-site Analysemöglichkeiten für die Weiterverwertung von Tunnelausbruchmaterial”. In: *BHM: Berg- und Hüttenmännische Monatshefte* (2013).
- Freeman, Adam. *Pro ASP.NET MVC 4*. Apress, 2012.
- Gabriel, Roland, Peter Gluchowski, and Alexander Pastwa. *Datawarehouse und Data Mining*. W3l, 2009.
- Gertsch, L. et al. “Use of TBM Muck as Construction Material”. In: *Elsevier: Tunneling and Underground space Technology* (2000).
- Gunnar Thies, Stefan Reimers und. *PHP 5.4 und MySQL 5.5*. Galileo Computing, 2012.
- Koller, Wolfgang. *Die volkswirtschaftliche Bedeutung mineralischer Rohstoffe in Österreich*. Tech. rep. Industrierwissenschaftliches Institut, 2007.
- Leitenmüller, Horst. *JSP - Java Server Pages*. Website. Available online at <http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2000/Leitenmueller/>; visited on December 9th 2013. 2000.
- Lieb, Rupert. “Materials management at the Gotthard Base Tunnel – experience from 15 years of construction”. In: *Wiley Geomechanics and Tunneling* (2009).
- MacDonald, Matthew. *Beginning ASP.NET 4.5 in C#*. Apress, 2012.
- MacIntyre, Peter, Brian Danchilla, and Mladen Gogola. *Pro PHP Programming*. Apress, 2011.
- McArthur, Kevin. *Pro PHP: Patterns, Frameworks, Testing and More*. Apress, 2008.
- Microsoft. *ASP.NET Manual*. Available online at <http://asp.net/>; visited on December 9th 2013. 2013.
- *SQL Server Manual*. Website. Available online at <http://msdn.microsoft.com/en-us/library/bb545450.aspx/>; visited on December 9th 2013.
- Oracle Corporation. *MySQL Manual*. Available online at <http://mysql.com/>; visited on December 9th 2013. 2013.

- Powers, David. *PHP Object-Oriented Solutions*. Springer, 2008.
- Quakenet/#php.de Staff. *Quakenet/#php Tutorial*. Website. Available online at <http://tut.php-quake.net/>; visited on December 9th 2013.
- Rauch, Roland and Thomas Beer. *Netzwerke - Grundlagen*. HERDT-Verlag für Bildungsmedien, 2004.
- Resch, Daniel. “Verwendung von Tunnelausbruchmaterial – Entscheidungsgrundlagen”. In: (2012).
- Seiden, Alan. “10 Cool PHP Things You Can run on Your System”. In: *System iNEWS* (2008).
- smarty.net. *Smarty Template Engine*. Website. Available online at <http://smarty.net/>; visited on December 9th 2013.
- Sun Microsystems. *JSP Manual*. Available online at <http://www.oracle.com/technetwork/java/javase/jsp/>; visited on December 9th 2013. 2013.
- Teuscher, Peter et al. “Alpenquerende Tunnel: Materialbewirtschaftung und Bionotechnologie beim Lötschberg-Basistunnel”. In: *Wiley Beton- und Stahlbetonbau* (2006).
- The PHP Group. *PHP.NET Manual*. Available online at <http://php.net/>; visited on December 9th 2013. 2013.
- The phpMyAdmin Project. *phpMyAdmin Manual*. Available online at <http://phpmyadmin.net/>; visited on December 9th 2013. 2013.
- Tokgoeoz, Nuray. “Use of TBM excavated materials as rock filling material in an abandoned quarry pit designed for water storage”. In: *Elsevier: Engineering Geology* (2011).
- tutorialspoint.com. *Client Server Model - Architecture*. Website. Available online at http://www.tutorialspoint.com/unix_sockets/client_server_model.htm; visited on December 9th 2013.
- tutsplus.com. *PDO vs. MySQLi: Which Should You Use?* Website. Available online at <http://net.tutsplus.com/tutorials/php/pdo-vs-mysqli-which-should-you-use/>; visited on December 9th 2013.
- Wasserbacher, R. “Die volkswirtschaftliche Bedeutung mineralischer Rohstoffe in Österreich”. In: *Springer: BHM* (2007).
- Welling, Luke and Laura Thomson. *PHP and MySQL*. Developers Library, 2009.
- Wikipedia. *ASP.NET*. Website. Available online at <http://en.wikipedia.org/wiki/Asp.net/>; visited on December 9th 2013.
- *Client-Server Model*. Website. Available online at http://en.wikipedia.org/wiki/Client_server_model/; visited on December 9th 2013.
 - *Database model*. Website. Available online at http://en.wikipedia.org/wiki/Database_models/; visited on December 9th 2013.

- Wikipedia. *Java Servlet*. Website. Available online at http://en.wikipedia.org/wiki/Java_Servlet/; visited on December 9th 2013.
- *Model View Controller*. Website. Available online at http://de.wikipedia.org/wiki/Model_View_Controller/; visited on December 9th 2013.
 - *Model View Controller*. Website. Available online at http://en.wikipedia.org/wiki/Model_View_Controller/; visited on December 9th 2013.
 - *Oracle Database*. Website. Available online at http://en.wikipedia.org/wiki/Oracle_Database/; visited on December 9th 2013.
- Zandstra, Matt. *PHP Objects, Patterns and Practise*. Apress, 2008.
- Zarai, H.R., A. Uromeihy, and M. Sharifzadeh. “A new tunnel inflow classification (TIC) system through sedimentary rock masses”. In: *Elsevier* (2012).

List of Figures

1	Flowchart - Classification	18
2	Interface - Main Menu	19
3	Interface - Add Material	20
4	Interface - Activate Material	20
5	Interface - Chemical Parameters	21
6	Interface - Brickearth	21
7	Interface - Login	22
8	Interface - List Materials	23
9	Interface - Main Menu Admin	24
10	Interface - Add User Admin	25
11	Model - View - Controller	26
12	Model - View - Controller with User Interaction	27
13	Client-Server Model	28
14	Server-Client-Script Interpreter Model	29
15	Script Language - Database Interaction	30
16	ASP.NET Page "Lifecycle"	33
17	ASP.NET MVC	33
18	JSP Lifecycle	34
19	Hierarchical Model	37
20	Network Model	38
21	Relational Model	38
22	Different User - Database Interaction	43
23	Database structure	45
24	Folder structure	49
25	"inc" folder	50
26	Template Structure	54
27	index.php - Flow Chart	55
28	SQL Database Structure	56
29	The "Menu" Page	62
30	"inc" folder	63
31	The "Login" Template	67
32	Main Menu - Input / Output Categories	68
33	Summary of Output Criteria	69
34	Output Cement Industry	70
35	Template - Mineralogical Parameters	73
36	Template - Save Mineralogical Parameters	74
37	Interface - Output Summary Limestone_1	86

38	Interface - Detail Limestone_1	86
----	--	----

List of Tables

1	Reuse Constraints	6
2	Aggregate Requirements	8
3	Arithmetic mean of the mass fractions of the grain size distribution curves sorted by the maximum grain size for inner lining cement	9
4	Arithmetic mean of the mass fractions of the grain size distribution curves sorted by the maximum grain size for shotcrete	9
5	Reuse Scenarios for Intermediate Products	12
6	Reuse Scenarios for End Products	13
7	Advantages and Disadvantages PHP	35
8	Advantages and Disadvantages ASP.NET	35
9	Advantages and Disadvantages JSP	36
10	MySQL, MySQLi and PDO - Comparison	41
11	Example User table	44
12	Chemical Composition of tested Limestone	83
13	Chemical Composition of tested Sandstone	84
14	Chemical Composition of tested Clay/Marl	85
15	Requirements For Lime; Portland Cement Limits according to German Standards	A-1
16	Chemical Requirements for Wall Bricks	A-2
17	Mineralogical Requirements for Wall Bricks	A-3
18	Chemical Requirements for Clay Cement	A-3
19	Chemical Requirements for Nepheline-Syenite	A-3
20	Chemical Requirements for Nature Cement	A-4
21	Chemical Requirements for Portland Cement (Austria)	A-4
22	Chemical Requirements for Gypsum and Anhydrite	A-4
23	Chemical Requirements for Limestone in the Paper Industry	A-5
24	Chemical Requirements for Magnesite in the Paper Industry	A-5
25	Chemical Requirements for Bauxite as Slagformer	A-6
26	Chemical Requirements for Olivine in the Steel Industry	A-6
27	Chemical Requirements for Olivine in the Foundry Industry	A-6
28	Chemical Requirements for Fluorite for metallurgical Grade Fluorite	A-7
29	Chemical Requirements for Limestone for metallurgy	A-7
30	Chemical Requirements for Dolomite (uncalcined) for pig iron (direct use)	A-7

31	Chemical Requirements for Dolomite (calcined) for refractory industry	A-7
32	Chemical Requirements for Dolomite (calcined) for Steelproduction	. A-8
33	Chemical Requirements for Magnesite (calcined) for Transformer- steelcoating	A-8
34	Chemical Requirements for Magnesite (calcined) for Steelindustry	. . A-9

A Annex

A.1 Tables for Material Requirements for Industrial Use

A.1.1 End Product

Lime Table 15 shows the requirements for lime related to different applications (steel industry, quicklime, portland cement and agriculture).

Chemical Composition	Steel Industry	Quicklime	Portland Cement	Agriculture
CaCO ₃	>95%	>97%	>75%	>90%
CaO	>95.2%	>53.2%	>42%	>50.4%
MgCO ₃	<10%	<3%	<6%	advantage
MgO	<5%	<2%	<3%	advantage
SiO ₂	<1.5%	-	<15%	-
Al ₂ O ₃	<1%	<0.9%	<5%	<1%
Fe ₂ O ₃	<2%	<0.9%	<4%	<1%
Na ₂ O	<0.5%	-	<1%	<0.05%
K ₂ O	<0.5%	-	<1%	<0.05%
SO ₃	<0.05%	-	<0.5%	-
P ₂ O ₅	<0.01%	-	<0.5%	-
grain size	0-3, 0-8, 8-40, 20-63mm	10-160mm	different	97% < 3mm 70% < 1mm

Table 15: Requirements For Lime; Portland Cement Limits according to German Standards

Brickearth The requirements for brick clay are set in dependence on the use:

- block brick (facing blocks)
- clay block
- paving bricks
- roofing tiles
- clay pipes

Table 16 and 17 show the chemical composition and the mineralogical composition with their average values. This is only done for wall bricks, as they have the strictest

requirements and are most important for the building industry.

Chemical Composition	Average Limits
SiO ₂	49.2 - 68%
Al ₂ O ₃	10.2 - 19.4%
Fe ₂ O ₃	2.7 - 8.0%
TiO ₂	0.3 - 1.7%
CaO	0.3 - 9.4%
MgO	0.5 - 2.9%
K ₂ O	1.3 - 4.0%
Na ₂ O	0.3 - 1.2%
CaCO ₃	0 - 18%
Corg	0.04 - 1%
total sulfur	0.04 - 0.56%
loss on ignition	4.2 - 9.1%

Table 16: Chemical Requirements for Wall Bricks

Because of the large amount of different raw materials and the lower use the grouping changes, as mentioned, from the mineral itself to the industrial use, i.e. cement industry, glass industry or abrasives ...

Cement Industry The following basic raw materials are needed for cement production:

- bauxite for clay cement
- nepheline-syenite for cement production
- limestone for "Natural" cement
- limestone for "Portland" cement
- magnesite (calcined) for floor cement
- gypsum and anhydrite for set retarder in cement

Table 18 to 22 show the chemical requirements for the different cement types.

Magnesite for floor cements, only, need an amount of MgO between 75 and 87%.

Paper Industry The main raw materials used in the paper industry are talc, limestone, dolomite, magnesite and baryte (as a filler). No Limits were found for dolomite; baryte, however, should have an amount of BaSO₄ between 97 and 100%

Chemical Composition	Average Limits
kaolinite	0 - 15%
sericite+illite	10 - 20%
smektite	0 - 5%
chlorite	0 - 5%
quartzite	30 - 55%
feldspar	0 - 13%
calcite	0 - 10%
dolomite ankerite	<1%
goethite	<1%
hematite	<1%
siderite	<1%
pyrite	<1%
gypsum	<1%
hornblende	<1%
other	1-10%

Table 17: Mineralogical Requirements for Wall Bricks

Chemical Composition	Average Limits
SiO ₂	2 - 6%
Al ₂ O ₃	45 - 55%
Fe ₂ O ₃	10 - 30%
TiO ₂	2.5 - 3%
CaO	0.5 - 3%

Table 18: Chemical Requirements for Clay Cement

Chemical Composition	Average Limits
Al ₂ O ₃	45 - 55%
MgO	0 - 1.5%
FeO	0.5 - 3%

Table 19: Chemical Requirements for Nepheline-Syenite

Chemical Composition	Average Limits
SiO ₂	16 - 35%
Al ₂ O ₃	2 - 20%
Fe ₂ O ₃	1 - 8%
CaO	28 - 55%
MgO	3 - 32%
K ₂ O	1 - 7%
Na ₂ O	1 - 7%
SO ₃	0.5 - 3%

Table 20: Chemical Requirements for Nature Cement

Chemical Composition	Average Limits
SiO ₂	0 - 15%
Al ₂ O ₃	0 - 5%
Fe ₂ O ₃	0 - 4%
CaO	42 - 100%
MgO	0 - 3%
K ₂ O	0 - 1%
Na ₂ O	0 - 1%
CaCO ₃	75 - 100%
MgCO ₃	0 - 6%
SO ₃	0 - 0.5%
P ₂ O ₅	0 - 0.5%
Cl	0 - 0.2%

Table 21: Chemical Requirements for Portland Cement (Austria)

Chemical Composition	Average Limits
MgO	0 - 3%
CaSO ₄ .2H ₂ O	70 - 100%
Cl	0 - 0.5%

Table 22: Chemical Requirements for Gypsum and Anhydrite

and no contamination of Fe_2O_3 . The requirements for talc, limestone and magnesite can be seen in Table 23, 23 and 24.

Chemical Composition	Average Limits
SiO_2	0 - 2%
Al_2O_3	0 - 2%
Fe_2O_3	0%
CaO	52.1 - 100%
MgO	0 - 2%
CaCO_3	93 - 100%
MgCO_3	0 - 4%

Table 23: Chemical Requirements for Limestone in the Paper Industry

Chemical Composition	Average Limits
SiO_2	0 - 3%
Fe_2O_3	0 - 0.5%
MgO	70 - 90%

Table 24: Chemical Requirements for Magnesite in the Paper Industry

Steel Industry A wide range of raw materials, mainly Dolomite and Magnesite, is considered in the Steel Industry section. They are classified as:

- Bauxite as Slagformer
- Olivine for Steel Industry
- Olivine for Foundry Industry
- Fluorite for metallurgical grade fluorite
- Limestone for metallurgy
- Dolomite (uncalcined) for pig iron (direct use)
- Dolomite (uncalcined) for Steelproduction
- Dolomite (calcined) for refractory industry
- Dolomite (calcined) for Steelproduction
- Magnesite (calcined) for Transformersteelcoating
- Magnesite (calcined) for Steelindustry

The requirements can be seen in Table 25 to 34.

Chemical Composition	Average Limits
SiO ₂	2 - 3%
Al ₂ O ₃	55 - 57%
Fe ₂ O ₃	22 - 23%
K ₂ O	0 - 0.1%
TiO ₂	0 - 6%
MgO	0 - 0.1%
S	0 - 0.1%
LOI	11 - 14%

Table 25: Chemical Requirements for Bauxite as Slagformer

Chemical Composition	Average Limits
SiO ₂	42 - 43%
Al ₂ O ₃	0 - 2.5%
Fe ₂ O ₃	6.8 - 7.3%
K ₂ O	0 - 0.2%
CaO	0.1%
MgO	48 - 50%
Na ₂ O	0 - 0.2%
LOI	0 - 1.5%

Table 26: Chemical Requirements for Olivine in the Steel Industry

Chemical Composition	Average Limits
SiO ₂	41.5 - 42.5%
Al ₂ O ₃	0.4 - 0.5%
Fe ₂ O ₃	0 - 7%
CaO	0.1%
MgO	49 - 51%
LOI	0 - 1.3%

Table 27: Chemical Requirements for Olivine in the Foundry Industry

Chemical Composition	Average Limits
SiO ₂	0 - 15%
CaCO ₃	0 - 3%
MgCO ₃	0 - 1%
S	0 - 0.3%
CaF ₂	60 - 100%
Pb	0 - 0.5%

Table 28: Chemical Requirements for Fluorite for metallurgical Grade Fluorite

Chemical Composition	Average Limits
SiO ₂	0 - 1.5%
Al ₂ O ₃	0 - 1%
Na ₂ O	0 - 0.5%
K ₂ O	0 - 0.5%
CaCO ₃	95 - 100%
CaO	95.2 - 100%
MgO	0 - 0.5%
MgCO ₃	0 - 10%
P ₂ O ₅	0%
SO ₃	0 - 0.1%

Table 29: Chemical Requirements for Limestone for metallurgy

Chemical Composition	Average Limits
SiO ₂	0 - 3%
Fe ₂ O ₃	0 - 1.5%
CaO	31 - 35%
MgO	16 - 20%
SO ₃	0 - 0.1%

Table 30: Chemical Requirements for Dolomite (uncalcined) for pig iron (direct use)

Chemical Composition	Average Limits
SiO ₂	0 - 3%
Al ₂ O ₃	0.2 - 0.3%
Fe ₂ O ₃	0.2 - 0.5%
CaO	0 - 35%
MgO	19 - 100%

Table 31: Chemical Requirements for Dolomite (calcined) for refractory industry

Chemical Composition	Average Limits
SiO ₂	0 - 0.8%
Al ₂ O ₃	0 - 0.4%
Fe ₂ O ₃	0 - 1.3%
CaO	0 - 57%
MgO	39 - 100%

Table 32: Chemical Requirements for Dolomite (calcined) for Steelproduction

Chemical Composition	Average Limits
CaO	0 - 0.2%
MgO	99.4 - 100%

Table 33: Chemical Requirements for Magnesite (calcined) for Transformersteel-coating

A.1.2 Intermediate Product

Main groups:

1. Clay
2. a. Carbonates / b. Sulfates
3. Quartz
4. Vulcanic Stones
5. Feldspar
6. Al-Oxides
7. Mg-Oxides
8. Phosphates, Sulfur, Salt
9. Mica
10. Heavy Minerals
11. Beryllium Minerals, Bromine, Iodine

Chemical Composition	Average Limits
MgO	70 - 90%

Table 34: Chemical Requirements for Magnesite (calcined) for Steelindustry