Leo Wirth, BSc

Internet based database laboratory

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieurin

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Nikolai Scerbakov

Institute of Interactive Systems and Data Science

Graz, May 2017

AFFIDAVIT

Abstract

Universities that want to offer a high quality of education have to provide e-learning courses nowadays. These workshops can facilitate the participants a lot of information and allow gaining more knowledge as in traditional ways. All this information has to be provided interactively to support the student's needs. To do so, modern e-learning environments have to offer a possibility to work on a system and get the feedback about the results immediately. This is not always possible, but in most technical courses, where participants have to solve tasks to learn something about the course topic, it should be done.

In this work, a system is developed to provide such an interactive role to an existing e-learning environment called WBT-Master. This newly developed internet based database laboratory provides the availability to directly execute database queries from the preferred e-learning environment on a database management system. These queries are then sent over an exchange protocol, called simple object access protocol, to the developed service. From there the query is executed on a database system. After the execution, the result is returned to the calling e-learning system. The participants are able to get the result directly after they executed the query. They do not have to wait until a tutor checks their solutions and they get feedback. With the use of this service, the learning rate is increased, and the managing effort is much less than in traditional database courses. The work splits into three main chapters. In the first one, the theory behind e-learning is explained. This is essential for the planning of the internet based database system, which is done in the second chapter. The third chapter introduces the developed system and describes how it can be adapted and how things are realized. The system itself can be clearly defined as a prototype because it has to be evaluated in real database courses, to guarantee a raising learning rate and to motivate course supervisors to use it for their systems.

Kurzfassung

Universitäten, die eine gute Ausbildung anbieten wollen, müssen in der heutigen Zeit e-Learning-Kurse zur Verfügung stellen. Diese Kurse sollen den Teilnehmern eine Menge an zusätzlichen Informationen zur Verfügung stellen, um mehr Wissen als auf traditionellen Wegen zu generieren. Diese Unterlagen müssen auf interaktive Art und Weise zur Verfügung gestellt werden. Um dies zu tun, sollten moderne e-learning Umgebungen eine Möglichkeit bieten das Feedback sofort zu liefern. Das ist vor allem in technischen Kursen wo die Teilnehmer Aufgaben lösen müssen, um ihre Kenntnisse über das Kursthema zu verbessern, praktisch.

In dieser Arbeit wird die Entwicklung eines Systems beschrieben, welches eine solch interaktive Methode für eine bestehende e-learning Umgebung namens WBT-Master bietet. Dieses neu entwickelte, Internet-basierende Datenbanklabor bietet die Möglichkeit, Datenbankabfragen direkt in der bevorzugten e-learning Umgebung auszuführen. Diese Abfragen werden dann über ein spezielles Protokoll (Simple Object Access Protocol) an das neu entwickelte Service gesendet. Anschließend wird die Abfrage mit Hilfe eines Datenbanksystems ausgeführt. Nach der Ausführung wird das Ergebnis an das e-learning System zurück gesendet und der Teilnehmer kann das Ergebnis direkt abrufen. Die Studenten müssen nicht warten, bis ein Tutor ihre Lösungen überprüft und das daraus resultierende Feedback an sie zurück sendet. Durch die Nutzung des Internet-basierenden Datenbanklabor soll einerseits die Lernrate erhöht werden und andererseits der Organisationsaufwand viel geringer sein als in herkömmlichen Datenbankkursen. Die Arbeit teilt sich in drei Hauptkapitel. Im ersten Abschnitt wird die Theorie hinter e-learning erklärt. Dies ist für die Planung des Internet-basierten Datenbanksystems wichtig, welche im zweiten Kapitel beschrieben wird. Der dritte Abschnitt stellt die entwickelte Applikation vor und beschreibt die Funktionalität und wie Modifikationen realisiert werden können.

Contents

List of Figures

List of Tables

1. Introduction

The project described in this document shows how to combine modern e-learning environments with a web service to generate a whole new course experience. Interactive laboratories help students to understand theoretical things easier. But in a digital economy, as we live, laboratories have to be integrated into e-learning systems to gain more comfort. To do so, the internet-based database laboratory is a virtual laboratory where all participants can easily train their database knowledge. The whole training can be done in this system, as well as the exam and participants get their results shortly after they send the query to the web service.

To gain the maximum benefits out of this web service, knowledge about e-learning systems is as important as database laboratory experience. The database laboratory is developed as web service, to extend currently available online learning systems. The integration will be done over the Simple Object Access Protocol (SOAP) and the database management system for this project will be MySQL. The e-learning environment used to demonstrate this project is the WBT-Master which is also used in database courses at the Technical University of Graz. To enhance the current methodology of the database course, the internet based database laboratory will bring the learning experience to a new level. The lecturing team will benefit from this system because it will be easier to train, support and evaluate course attendees. It will be less time-consuming to check the submitted queries and also the time used to teach students will be shortened.

1.1. Motivation

The motivation of this project is the problem that more and more courses at the university use e-learning systems where students have to submit their assignments, but most of the time there is no real fast response of the result. There are several test cases, and students get a number of test cases they passed. So there is no real feedback of the mistakes. To generate a system which immediately sends back the solution for a given query, the internet-based database laboratory is developed and integrated into an e-learning system. The whole project will be done in context with database courses at the Technical University of Graz to generate an internet-based database laboratory, where students can better train their database skills.

1.2. Thesis structure

The thesis starts with a theoretical part about e-learning, e-learning environments and several definitions. This is necessary because many decisions in the development of the systems were done because of the aspects mentioned in e-learning development. After that, the WBT-Master system, the architecture and the function principle are mentioned. This is important for the integration of the web service into the e-learning environment. In chapter 2 the theoretical part of the web service is described. All the requirements and the primary functionality are described here. The architecture and the design of the system are mentioned here and the database interaction and the communication process are defined there. The practical implementation can be found in chapter 3. Every single component of the system is described here. It starts with the main part, the Java servlet, where the request/response handling is defined. The system configuration and the user interface described there. The next big part of chapter 3 describes the communication process between the WBT-Master and the web service.

1.3. The e-learning idea

e-learning is an essential part of every digital native nowadays. e-learning has evolved over the years from e-learning 1.0 to e-learning 2.0 which we have reached now. Ebner, 2007 described e-learning 1.0 as the act of creating perfect content with interactions, animations, simulations and similar:

> Comparable to the typical classroom there is a teacher and there are students (learners). The teaching person provides the content in high quality. The learning material is accessible via a learning management system. The "new" (in relation to the traditional face-face teaching) components are further tools, like communication tools or interactive exercises.

But this was not enough for most use cases and did not add enough benefits to students and teachers. Something new has to be included to the e-learning 1.0 concept. At the same time, the Web 2.0 came up and people over the world started to use social media applications. Downes, 2005 describes the transformation of the "Read-Web" to the "Read-Write-Web" as a social revolution, not a technical revolution.

Ebner, 2007 mentioned that e-learning 2.0 is not the product of e-learning 1.0 combined with web 2.0. There has to be a third component which he calls "human factor". This factor is a combination of the explanation and the time to realize from people to use such systems. He also points out that e-learning 2.0 is the word of the next generation and it will be a step into modern education. There are other papers which show the importance of e-learning in context with modern education.

Edrees, 2013 reports that e-learning 2.0 is the integration of web 2.0 applications into educational and institutional practice:

> They emphasized that, tools like blogs, wikis, media sharing applications, and social networking sites can support and encourage informal conversation, dialogue, collaborative content generation, and knowledge sharing, giving learners access to a wide range of ideas and representations.

Summarized, it has to be mentioned, that e-learning is the big thing which actually changes education all over the world. Only good e-learning systems will be used by the audience because otherwise there is no overvalue for education.

1.3.1. Definition

In the very widespread field of e-learning, many concepts and definition are mentioned. Ozkan and Koseler, 2009 defined e-learning as follows:

> Electronic learning (hereafter e-learning), referring to the use of electronic devices for learning, including the delivery of content via electronic media such as Internet, audio or video, satellite broadcast, interactive TV, CD-ROM, and so on, has become one of the most significant developments in the information systems (hereafter IS) industry (Wang, Liaw, & Wang, 2003).

This definition shows a very common thinking of e-learning in it is early days. In these days e-learning was just a way of consuming educational content in an electronic way. As already mentioned in 1.3 e-learning 1.0 was just a distribution of content. Another more modern definition of e-learning is described by Pieri and Diamantini, 2014:

> E-learning 2.0, in opposition to e-learning systems not based on CSCL (computer-supported collaborative learning), undertakes that knowledge (as meaning and understanding) is socially constructed.

All these definitions show that e-learning 2.0 can be seen as social media of the learning paradigm. Wever et al., 2007 defines e-learning 2.0 as a ”social software for educational use”, which is in principle the approach, all the other definitions do, but they also mention the difference between the instructors and the participants. They report the following differences between them:

> However, the penultimate question, focusing on the kind of information that is shared, reveals that instructors do share information,

> especially information related to their education, whereas students use social software mainly for sharing multimedia and other leisure related activities, other than educational purposes.

The difference between the two main stakeholders of e-learning is one of the problems, where e-learning can fail.

Summarized it can be reported, that the central argument of e-learning is some kind of electronic distribution of content, which is merged with modern technologies to educate people. To get a better understanding of the content, the social aspect, which is already integrated in Web 2.0 has to be integrated into modern e-learning 2.0 environments, to provide powerful systems, which will support a high learning rate.

1.3.2. Synchronous vs. asynchronous systems

There are several types of e-learning systems. All kinds support other needs, therefore all systems can be categorized into divisions. The biggest distinction of e-learning systems is synchronous and asynchronous systems. Shahabadi and Uplane, 2015 describe synchronous e-learning as a process of live, real-time, facilitated instruction and learning-oriented interactions. It's a kind of virtual classroom where the whole participants learn together via an online environment. In context to this, asynchronous systems are defined by Mayadas, 1997 as an interactive learning community, which is not limited by time or place. It is a virtual course, where everyone can learn whenever and wherever he wants, it is no real virtual live class. The internet based database-laboratory in this course is designed as an asynchronous system, where participants can practice whenever and wherever they want.

1.4. Learning impact

The biggest goal of e-learning is to provide the additional benefit of the participants to gain a faster learning rate and increase the motivation. This learning impact is going to be directly influenced by the use of an e-learning environment. Aparicio, Bacao, and Oliveira, 2016 came to the conclusion, that their study indicates, that students perceive e-learning systems increase their productivity and simplify the given task. This means that the usage of e-learning influences the university success. The performance of the university is increased by the use of such concepts. Aparicio, Bacao, and Oliveira, 2017 mentioned, that usage and satisfaction have a significant impact on individual impacts, perceived different impacts of e-learning systems is noticeable in accomplishing their tasks, in their productivity, and in the usefulness of the system. This is a very interesting fact for the development of e-learning systems. Most of the times, students complain about the usage of e-learning platforms. But the result of the study shows, that in the end, students gain more advantages as disadvantages with the use of such systems. Therefore, if a system provides good quality to participants and instructors, the impact on the learning rate is much higher because both sites will use it. Khatri, Chouskey, and Singh, 2013 mention the biggest aspect in context with successful e-learning systems. They conclude that students have to be ready to use these environments, therefore educators have to take care that they train participants to use these systems, otherwise there will not be any advantage in the teaching process. So there must be a kind of combination of e-learning and traditional learning in the beginning, which will be very time-consuming and expensive, but after that training phase, both parties will be happy with the learning impact on the daily teaching process. Summarized, it has to be mentioned, that the learning impact or the increase of the learning rate is directly connected to the skills of an instructor for motivating students to work with e-learning.

1.5. Online learning environments

The meaning of e-learning is described, but e-learning without a competitive environment is not very useful. To do so an online learning environment is used. Moore, Dickson-Deane, and Galyen, 2011 mentioned that the definition of online learning is pretty hard and there are several definitions. D. G. Oblinger and J. L. Oblinger, 2005 see every learning act which is done online as online learning. Another definition by Lowenthal, Wilson, and Parrish, 2009 is that they define the used medium as the type of learning. But Moore, Dickson-Deane, and Galyen, 2011 finds out that most authors report, that online learning is described as access to learning experiences via the use of some technology. So this is a very widespread and weak definition, but online learning is as simple as described.

1.5.1. Introduction

There are several designations for online learning environments Some authors call them virtual learning environments and describe them as online platforms for e-learning. These virtual learning environments (VLE) are as described by Dillenbourg, Schneider, and Synteta, 2002 as more than just a popular label to describe any educational software. They defined seven aspects, which every VLE can cover:

- A virtual learning environment is a designed information space.
- A virtual learning environment is a social space: educational interactions occur in the environment, turning spaces into places.
- The virtual space is explicitly represented: the representation of this information/social space can vary from text to 3D immersive worlds.
- Students are not only active, but also actors: they co-construct the virtual space.
- Virtual learning environments are not restricted to distance education: they also enrich classroom activities.

- Virtual learning environments integrate heterogeneous technologies and multiple pedagogical approaches.
- Most virtual environments overlap with physical environments.

At least one facet is covered by every environment out there. Otherwise, the system is no real online or virtual learning system. Reese, 2015 points out, that while online education lacks the substance needed for higher education, distance education has the potential to set a high standard for valuable learning experiences in virtual environments. Instructors and students can benefit from an environment that is rich in communication, collaboration, and community. All these benefits have to be covered by an online learning environment, otherwise the benefit of using it is not given.

Another aspect, which should be kept in mind is the fact, that virtual learning or online learning systems are not only used by schools or universities. There are many companies which use such systems to train their employees. It saves money, time and energy. And not only employees benefit from e-learning systems, but also customers get the possibility to train their skills in online academy, where professional trainees interact with them. Especially for schooling for business software, like SAP, online learning institutes are very common. Thus, the learning rate in comparison to the invested energy is much higher than traditional product schoolings.

1.5.2. History

Online learning was on of the first fields which was developed in the beginning of electronic communication and the internet. Harasim, 2000 shows in figure 1.1 the development of online learning in the last century.

It can be seen that the early online learning environments use very few ”online” activities. For instance, the first university courses which were supplemented by e-mail and computer conferences. It can also be seen, that the first online learning courses started already about ten years before the release of the World Wide Web. Harasim, 2000 defined the online learning systems in the following way:

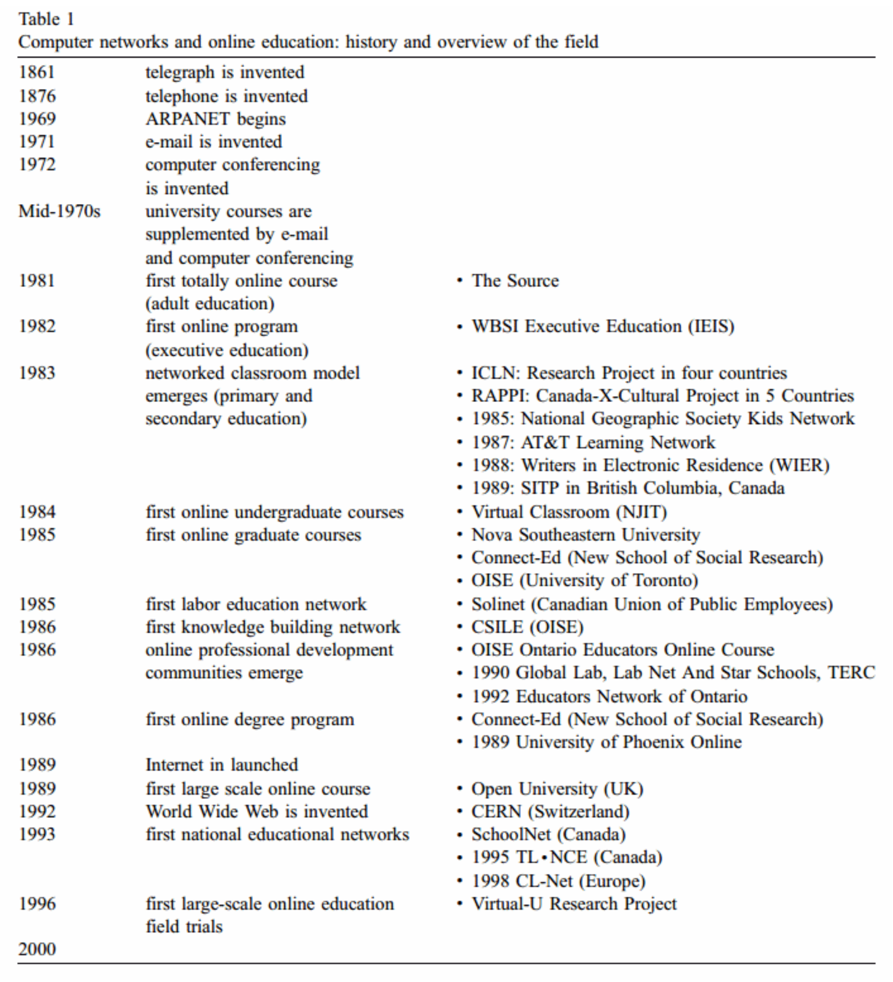


Table 1
Computer networks and online education: history and overview of the field

1861	telegraph is invented	
1876	telephone is invented	
1969	ARPANET begins	
1971	e-mail is invented	
1972	computer conferencing is invented	
Mid-1970s	university courses are supplemented by e-mail and computer conferencing	
1981	first totally online course (adult education)	• The Source
1982	first online program (executive education)	• WBSI Executive Education (IEIS)
1983	networked classroom model emerges (primary and secondary education)	• ICLN: Research Project in four countries • RAPPI: Canada-X-Cultural Project in 5 Countries • 1985: National Geographic Society Kids Network • 1987: AT&T Learning Network • 1988: Writers in Electronic Residence (WIER) • 1989: SITP in British Columbia, Canada
1984	first online undergraduate courses	• Virtual Classroom (NJIT)
1985	first online graduate courses	• Nova Southeastern University • Connect-Ed (New School of Social Research) • OISE (University of Toronto)
1985	first labor education network	• Solinet (Canadian Union of Public Employees)
1986	first knowledge building network	• CSILE (OISE)
1986	online professional development communities emerge	• OISE Ontario Educators Online Course • 1990 Global Lab, Lab Net And Star Schools, TERC • 1992 Educators Network of Ontario
1986	first online degree program	• Connect-Ed (New School of Social Research) • 1989 University of Phoenix Online
1989	Internet in launched	
1989	first large scale online course	• Open University (UK)
1992	World Wide Web is invented	• CERN (Switzerland)
1993	first national educational networks	• SchoolNet (Canada) • 1995 TL•NCE (Canada) • 1998 CL-Net (Europe)
1996	first large-scale online education field trials	• Virtual-U Research Project
2000		

Figure 1.1.: Computer networks and online education: history and overview of the field (Harasim, 2000)

> The computer communications revolution of the mid 20th century affected all social and economic realms. Educators were among the first to embrace the revolution, and the increased educational opportunities and especially the new learning models that have emerged are now influencing education and society as a whole. At the turn of the 21st century, public discourse is beginning to recognize the implications of this educational transformation

Online learning environments changed over the years. In the early years, this platforms were very static and have not changed the way the education was done. There were normal courses, in class, and the online platforms contain additional information about the actual topic which was taught. There was no real virtual class, where many students work on their learning skills. Another aspect is the teaching style, which has not changed with early versions of e-learning. Most of the time, it was a teacher-centred teaching, where students only get to know, what the teachers learned them. This was also displayed in the content of the learning platforms, only the teacher was able to upload his material and the students could download it over their small internet connection. It was also a problem for many people to download the additional material because an internet connection was not available for everyone.

Mason, 2000 reports from e-learning courses in the UK Open University, where students with tutors form small groups which interact with the teachers by conferencing and e-learning systems. This was a pretty good approach because these tutoring sessions were held all over the land and the use of online learning helps them to provide a pretty good course quality. But there were problems because some students were not able to afford online access to their program, so there were still difficulties with the domain of online learning. Nevertheless, he praised online learning but mentioned that the majority of the courses were not delivered over the web and were still optional or additional benefits. He summarized his results that the business of encouraging people to learn is delicate, artful, and evolutionary. It will be ready as soon as all the students are ready, to create an e-university.

1.5.3. At the moment

As described in the last section, the lag of an internet connection, was the biggest point, why online learning in the first years was not as important as now. With the invention of the Web 2.0, which was first mentioned in the year 2003, many social media services came up. As already reffered in 1.3, the social aspect has completely changed e-learning. Thus, online learning environments have changed in the way they have to be used. Not only teachers have to upload content and build a static online course, but also students have to do this. Now, the students is the center of the online course and over the last ten years also the educational style changed in several countries. The teacher-centred learning was repressed more and more and now there is some kind of student-centred learning. O'Neill and Geraldine, 2005 has given a definition for the student-centred learning:

> The interpretation of the term "student–centred learning" appears to vary between authors as some equate it with 'active learning', while others take a more comprehensive definition including: active learning, choice in learning, and the shift of power in the teacher–student relationship.

To do so, it is necessary that students get access to the online environment every time and everywhere. It is not very useful if they only get access to the training site. This problem was solved with the simple internet access. At some point in the 21st century many people all over the world get internet access very simple. The number of internet devices grows rapidly and Statista, 2016a reports a total of 3.47 average connected devices per person. Internet has become a daily thing and is no problem to get connected. At this point, e-learning has grown very fast. Therefore, online learning has become a business. It is not like a simple learning environment, programmed by a group of students or tutors to support a teacher's course, it is much more. There are several companies and scientist to find out how to build a modern e-learning system which supports students and gain an extra benefit to the users. In figure 1.2 it can be seen, that the growth of the market in the last 3 years was great, especially in North America and Asia.

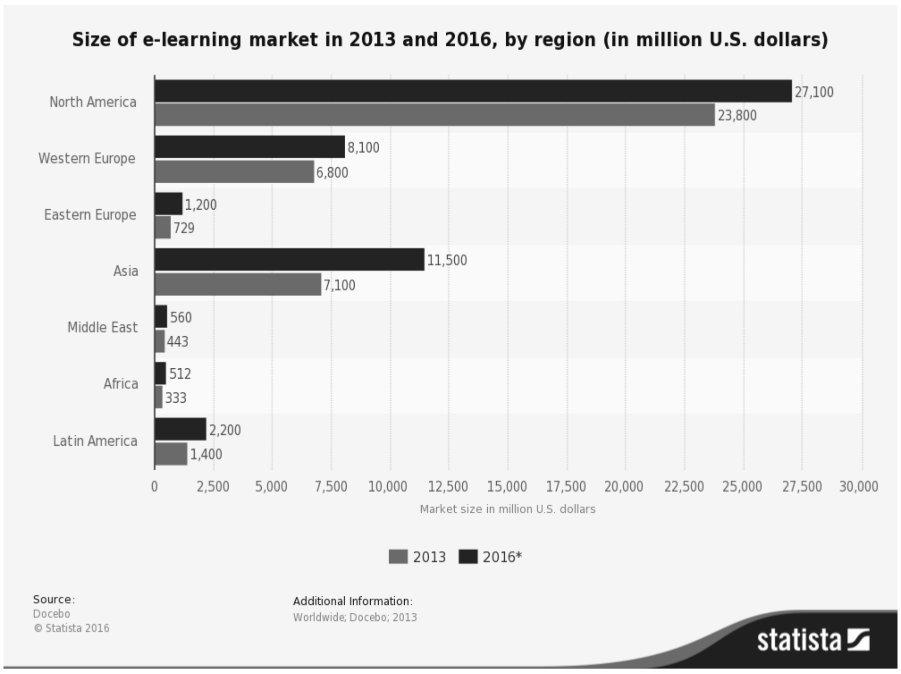


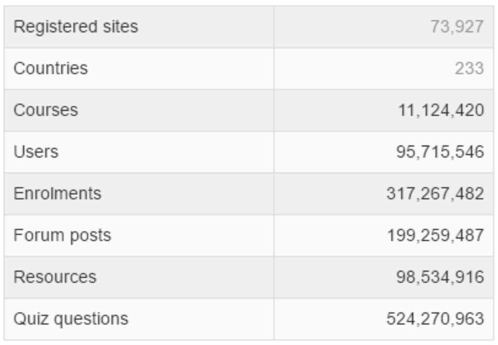
Figure 1.2.: e-learning market worldwide (Statista, 2016b)

1.5.4. Environments

The field of online learning environments is quite big. There are various big players, which provide different services for miscellaneous use cases. For this project, some systems for university usage will be mentioned and their features will be described.

Moodle

The Moodle, 2016a system is one of the biggest open source learning management system. The environment is a PHP based web-framework, which supports many e-learning types. It is fully compatible with mobile devices, supports plug-ins, themes and is available in many languages. Several universities and schools use this system because the costs are small. The Moodle system is a long developed framework, which was first released in 2002. The Moodle team provides usage statistics at their site, where they show the actual usage, which is shown in figure 1.3 by Moodle, 2016b. It can be seen that



Registered sites	73,927
Countries	233
Courses	11,124,420
Users	95,715,546
Enrolments	317,267,482
Forum posts	199,259,487
Resources	98,534,916
Quiz questions	524,270,963

Figure 1.3.: Moodle usage statistics(Moodle, 2016b)

there are more than 11 million courses provided by the different Moodle frameworks all over the world. Another statistics shows that most Moodle registration comes from the USA followed by Western Europe.

Dokeos

Dokeos, 2016 is another web-based learning software which is widely used by many institutions. Dokeos is not an open source tool, and therefore the price is calculated by the maximum users, authors and plug-ins. An interesting fact is, that Dokeos supports the integration into industries. It is a fully professional tool, which covers nearly every aspect of e-learning.

ATutor

ATutor, 2016 is another open source environment, which is widely used by many university. The big advantage of this system is, that supports many accessibility features which is important for disabled people. It also supports different e-learning topics like, online courses, research, etc. It is also build very modular and there are plug-ins for many situations

TeachCenter

TeachCenter, 2016 is an e-learning online platform developed by the Technical University of Graz. This learning management software is used since 2007 and is based on the WBT-Master system which is used for this project and described in section 1.5.5. A big opportunity for the system is, that it is not addicted to licensing, updates and the modular concept. All necessary modules are developed at the university and are integrated into the environment. It is very simple to upload, download and manage learning documents. The integration of Web2.0 services (cloud-systems, social media, online editors) can be done unproblematically A benefit is the use of communication tools like communication boards, e-mail and newsgroups. Another big advantage is the interaction with the online administration tool of the university.

1.5.5. WBT-Master

WBT-Master 2016 (WebBasedTraining-Master) is an e-learning environment developed by the Technical University of Graz in cooperation with various partners. The system was developed within the scope of Corporate Software Engineering Knowledge Networks for Improved Training of the Work Force (CORONET) project which was funded by the European Union. It is actually provided in the Version 1.1a, which was built in December 2011.

The WBT-Master software was developed, because students were not motivated to use other successful e-learning environments, because providing eBooks was not the act, students need to finish a course. They requested simple printable materials, in addition to the virtual animations and interactions, but after that, they did not use them anymore. That's why teacher started to reduce the development of eBooks. Therefore, the developers describe the motivation for their work in *WBT-Master white paper* 2016 as follows:

> Analysing the situation, we came to a conclusion, that browsing through a bunch of documents combined into a navigable structure and provided with communicational tools, can be seen as just one (and perhaps, not a best one) training paradigm. We can easily imagine a situation where students are requested to write short essay on a predefined topics, practically implement projects, answer questions and discuss their answers, discuss materials selected by teachers, etc., etc. From a teacher perspective, we can say that teachers should be able to create a training curriculum by combining different training applications into a new entity called a course [as opposed to authoring of an e-Book]. After recognizing such requirements, a new system called WBT-Master was developed on entirely new principles.

The WBT-Master system is used in this project for demonstrating the web service and for explaining the learning scenarios which occur in a database laboratory. It has to be noticed, that the web service is able to work with any kind of e-learning environment, but WBT-Master is already used for database courses at TU Graz.

2. Internet-based database laboratory

Internet based laboratory is a widely used environment in many educational courses. It combines the simplicity and comfortability of e-learning with real laboratory lessons. To combine this two aspects for database courses, several mechanisms are necessary, which will be mentioned in this chapter.

2.1. Introduction

It is every student's dream, to have an online system where you can train your knowledge and submit your work. But most of the time, you have to wait until someone of the lecturing team, either the tutors or the professor, looks through your result and return you some feedback.

Another type are systems, where people can upload their solutions and get the feedback immediately, but most of the time they don't know what was wrong and what was the source of the error. With the internet-based database laboratory, which has many benefits as mentioned in 2.3, the participants can solve their tasks and every solution gets executed on a real database environment. They get the result of the execution and also the lecturing team gets some feedback, about the problems of their course participants. To afford a good learning experience, the participants should think that they work with a real database management system.

In this chapter the functionality and the requirements of the internet-based database laboratory are described. Furthermore, the technical specification of the communication process and the system architecture can be found here.

2.2. Functionality

This internet based laboratory should especially fulfill the needs in context with database courses. In such courses many databases, queries and modifications can be done. Therefore, it would be good, if all these queries can be checked by a system and can be executed in a given environment. After that, the result of the operation could automatically be returned to the participants and the lecturer, so that both know what was wrong. The participants should get a system where they can train their database skills on a real system, but all their training should be supported by lecturing team via the e-learning system.

To integrate such systems in currently used environments a stand-alone web service is the best solution. Because with it, already existing e-learning software can easily be integrated with this database laboratory. To do so, the service offers a SOAP interface for the communication between course environment system and the web service. Another aspect is the integration of a database management system. Therefore, the web service has a connection to the MySQL database management system. The web service itself is responsible for all the communication between MySQL and SOAP and it would be possible to extend the communication interface in the future to use REST for example. The database system can be changed, to use the system in different database courses.

The web service can handle the following operations with the database system:

- Create database
- Define database schemas
- Modify database schemas
- Create data
- Read data
- Update data
- Delete data

To do so, the system gets split into several components. To reach a high compatibility with other existing learning environments and database management systems, the whole internet-based database laboratory is encapsulated in four main parts, where every part of the system will handle a subtask of the preferred functionality.

A big aspect is the reporting of the result, so the participants should get a helpful feedback. The result of the execution on the database management system will be returned.

2.3. Benefits

There are many benefits the system gives the lecturing team as well as the participants. All these benefits are generated from the web service in combination with a good e-learning system like WBT-Master. All the benefits can be assigned either to the laboratory attendees or the course instructor team.

2.3.1. Benefits for instructors

The instructors of a database laboratory get the advantages that they get much more feedback from the learning success of their participants. They can see the results of every single query as well as the errors every participant did. Also, the possibility to add additional feedback to the response of each query execution should help the participants to reach their goals.

Another big aspect is the time-saving factor in exams. Most time of the correction of an exam in the actual e-learning environment, without the internet-based database laboratory web service, is the correction of the query. Sometimes there are several possibilities for solving a given task, and the laboratory team has to correct every single query by hand. With this system, every query gets executed on a live database management system and the instructors will only get the result.

2.3.2. Benefits for participants

Learning database specific things on a real life system with very fast feedback, this is the way of learning what participants want. Every participant does not have to wait for his result of an exam or an assignment. The result should be returned as quick as possible and also the feedback should be returned in a way, that everyone knows what is wrong.

The second big benefit is the learning success. Because of the live execution of the query, the participants get the feedback very fast. So in practice mode, they can try again and execute their query again and again, until they get their preferred result. Also, the additional information in the response helps them to get an easier understanding of what was wrong in their query.

2.4. Learning scenarios

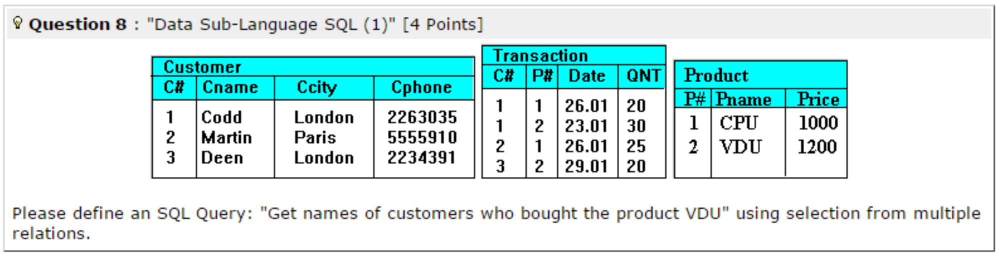
Every learning environment can handle various learning scenarios. These scenarios can be described as different modes to solve tasks to get an educational benefit. The goal of such learning scenarios are quite different and the supervisors of the course have to validate which scenario fits best to the given tasks. The different e-learning environments provides different learning scenarios, which could either be some interactive learning by doing projects where only a little data set is provided and the students have to solve their task from scratch. Or some quizzes where the students have to provide a solution for a task and the result can either be true or false. The scenarios are responsible for the knowledge transfer from the provided information to the students and differ from the domain of the course. The number of possible scenarios can be very high, but everyone has to be evaluated if it is good for the students and the knowledge domain. Courses which support a student to learn technical things have other requirements to learning scenarios then courses for other domains. To choose the correct scenario is important otherwise, the students get bored and the benefit of using an online environment is too small.

The internet based database environments should also be able to handle such scenarios. To explain and describe them in combination with the IBDL, two of them, which are most often used in the actual database course in WBT-Master are described. These two scenarios differ in their whole workflow and also the evaluation of the results is different. The evaluation of the result have to be done in various ways, to gain very less false positive and false negative results, and the participant of a course do not get bad results because of a wrong evaluation of their provided solution. Also, the supervisors benefit form the assessment because they have to check only a few results. In section 2.4.1, the first type "quiz" is described and in section 2.4.2 the scenario type "project schema" is mentioned. By the covering of the two scenarios in this document, the other ones can be derived.

2.4.1. Quiz

The first scenario which is mentioned is the type "quiz". In such quizzes, the participants of a database course have to provide the solution for questions they were asked. A quiz is generated by the instructors of the course and the students get an amount of points for every correct solution they provide. To demonstrate such a quiz scenario, an example of a currently used quiz in the WBT-Master environment is shown below. This mentioned scenario is most often used for exams in the actual database course.

To start a quiz, the participant has to open the quiz and solve the questions, provided by the supervisors of the course. These do not necessary correlate with each other so it can be that, there have to be many database schemas the student have to work with. Each question of the quiz shows the actual database schema or at least the database table which is necessary to get the correct answer. The questions can deal with different topics of databases courses. Most of them have the goal to get the right SQL query for an asked data set. Such a question can be the one, shown in figure 2.1.


Question 8 : "Data Sub-Language SQL (1)" [4 Points]

Customer			
C#	Cname	Ccity	Cphone
1	Codd	London	2263035
2	Martin	Paris	5555910
3	Deen	London	2234391

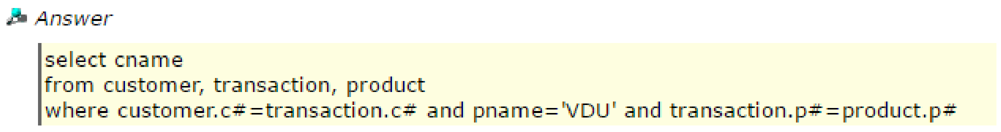
Transaction			
C#	P#	Date	QNT
1	1	26.01	20
1	2	23.01	30
2	1	26.01	25
3	2	29.01	20

Product		
P#	Pname	Price
1	CPU	1000
2	VDU	1200

Please define an SQL Query: "Get names of customers who bought the product VDU" using selection from multiple relations.

Figure 2.1.: Quiz question

With the provided information the participant has to solve the question. For this question, three tables are given and the participant has to create a query which returns all names of customers who bought the product VDU. Therefore, all tables have to be used. This can be done in different ways with several queries. If the student knows the result, it has to be filled in the answer box like shown in figure 2.2. The query which is shown in the yellow box is then used as answer and the supervisors of the course have to evaluate this answer. With the IBDL system the evaluation of the result should be done automatically, which is described in section. 2.4.3.



```
select cname
from customer, transaction, product
where customer.c#=transaction.c# and pname='VDU' and transaction.p#=product.p#
```

Figure 2.2.: Quiz answer

Summarized it could be mentioned that the quiz scenario is a mode for practicing the knowledge of the course topic by the use of questions, where the result can either be true or wrong. Only if the result delivers the preferred data set, it is correct and no other answers are right. The students do not have to build a database from scratch, only simple data querying have to be done. The learning experience for this scenario can be very high because the students have to understand the given database schema without any further informations and they have to know how to extract the information or how to create new pieces of information in the actual schema. To do such quizzes, experience in the domain of databases is required, but at the beginning of a course, the scenario ”project schema” described in the next section is more useful.

2.4.2. Project schema

The second learning scenario which is very common in most technical courses is the ”project schema” or ”project” mode. This mode is very extensive and allows the student to test their whole knowledge of a certain domain. The scenario has to be specified by the supervisors of a course. If the students full fill the required needs and provide the correct solution for a project they get the defined amount of points. To demonstrate the project schema - scenario the

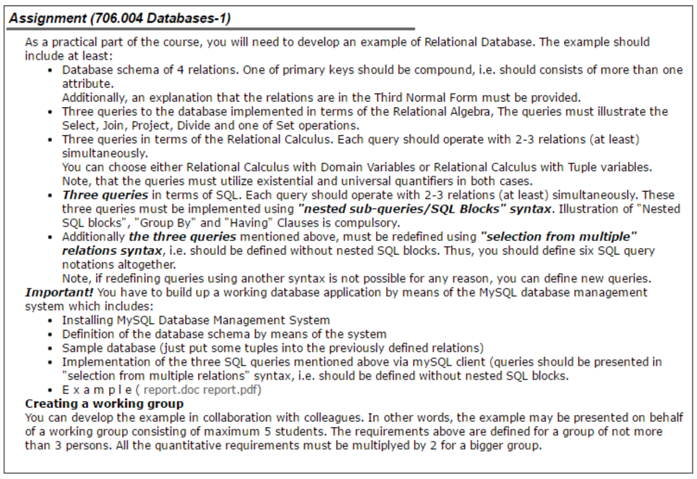
WBT environment is used because this scenario type is also used in database courses supported by WBT-Mater environment.

The first step is to create a personal project space in the e-learning tool. This is called "locker" in the WBT system. After this space is created, every file for the project has to be uploaded there. The supervisors release a detailed description of the SQL project. These descriptions define the entire database schema and the available tables. Mostly only the table and column names are specified and the students have to define the content on their own. It can also be the case, as it is in the final assignment in the current database course, that only the type of the resulting SQL queries is defined, and how the tables must relate to each other. In figure 2.3 such a description of an assignment is shown. With all the give definitions the students have to build the whole database schema as well as find the queries to get a solution for the tasks. After Everything is done, the SQL scripts with the solution have to be uploaded into the locker. The resulting locker can look like the one, shown in figure 2.4. After all files are uploaded the student is able to unlock his locker and the supervisors know, that a solution is provided. Then they have to check the schema and the provided queries to calculate the points for the project and evaluate the student.

Consolidated it has to be mentioned, that this type of e-learning scenario is a good opportunity to evaluate the student's knowledge for big parts of the course. Because of the extensive possibilities of projects or available queries, this scenario can proof if the students understand the domain of the course and know what they were doing. The time-consuming part of this scenario is the evaluation because every schema and query have to be loaded into a database management system and the result has to be compared with the original solution.

2.4.3. Evaluation of results

The evaluation of the result is a part of the system. This feature can save a lot of time for the course supervisors because the service automatically evaluates the results of the given queries. To do so, the evaluation has to support different learning scenarios as already mentioned before. The different scenarios require miscellaneous processes to evaluate the result to get a correct solution and



Assignment (706.004 Databases-1)

As a practical part of the course, you will need to develop an example of Relational Database. The example should include at least:

- Database schema of 4 relations. One of primary keys should be compound, i.e. should consists of more than one attribute.
  Additionally, an explanation that the relations are in the Third Normal Form must be provided.
- Three queries to the database implemented in terms of the Relational Algebra, The queries must illustrate the Select, Join, Project, Divide and one of Set operations.
- Three queries in terms of the Relational Calculus. Each query should operate with 2-3 relations (at least) simultaneously.
  You can choose either Relational Calculus with Domain Variables or Relational Calculus with Tuple variables.
  Note, that the queries must utilize existential and universal quantifiers in both cases.
- ***Three queries*** in terms of SQL. Each query should operate with 2-3 relations (at least) simultaneously. These three queries must be implemented using ***"nested sub-queries/SQL Blocks" syntax***. Illustration of "Nested SQL blocks", "Group By" and "Having" Clauses is compulsory.
- Additionally ***the three queries*** mentioned above, must be redefined using ***"selection from multiple" relations syntax***, i.e. should be defined without nested SQL blocks. Thus, you should define six SQL query notations altogether.
  Note, if redefining queries using another syntax is not possible for any reason, you can define new queries.

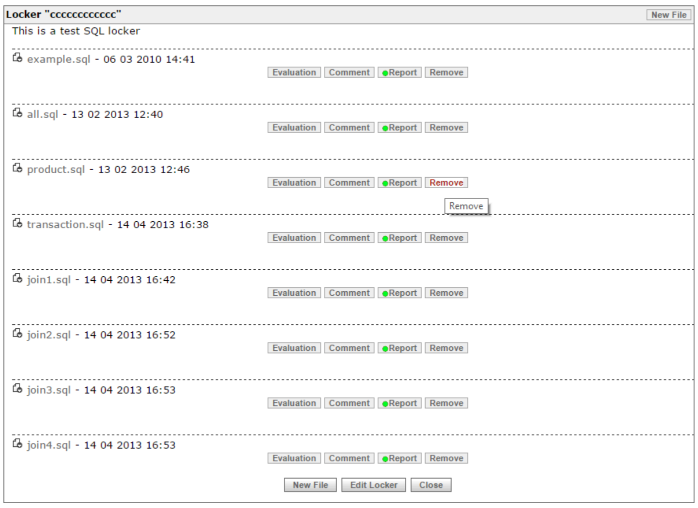
Important! You have to build up a working database application by means of the MySQL database management system which includes:

- Installing MySQL Database Management System
- Definition of the database schema by means of the system
- Sample database (just put some tuples into the previously defined relations)
- Implementation of the three SQL queries mentioned above via mySQL client (queries should be presented in "selection from multiple relations" syntax, i.e. should be defined without nested SQL blocks.
- E x a m p l e (report.doc report.pdf)

Creating a working group

You can develop the example in collaboration with colleagues. In other words, the example may be presented on behalf of a working group consisting of maximum 5 students. The requirements above are defined for a group of not more than 3 persons. All the quantitative requirements must be multiplyed by 2 for a bigger group.

Figure 2.3.: Assignment definition



Locker "cccccccccccc"

This is a test SQL locker

example.sql - 06 03 2010 14:41

all.sql - 13 02 2013 12:40

product.sql - 13 02 2013 12:46

transaction.sql - 14 04 2013 16:38

join1.sql - 14 04 2013 16:42

join2.sql - 14 04 2013 16:52

join3.sql - 14 04 2013 16:53

join4.sql - 14 04 2013 16:53

Figure 2.4.: Assignment solution

reduce the number of errors the system made. In this section, two different type of evaluation will be mentioned. The first type is the "Result evaluation" which is used to compare the result of a query provided by the students in a quiz for instance. The second on is the type "Query evaluation" which is used to check if different query types are provided by the students, which is the case in the "Project schema" learning scenario.

Result evaluation

This mode is easier for the evaluation of the results because the results can be evaluated as true or as false. This means that the result of the query, the participants provide has to be the same which is specified by the supervisors of the course for this question. If it is not the same, the number of points can be decreased. It is not enough to compare the two result-sets one-to-one because the metadata, like table cells or number of rows can be different and the evaluation will fail. To demonstrate this, the example below is provided.

The claimed result-set which is defined by the tutors should look like shown in table 2.1. In this case, all informations of brands should be displayed where land="Deutschland". One participant provides a query which returns the result-set shown in table 2.2.

id_b	**name**	**strasse**	**plz**	**land**
1	Intel	Intelstrasse	9112	Deutschland
2	AMD	Amdstrasse	7474	Deutschland

Table 2.1.: Requested result-set

id_b	**name**	**strasse**	**plz**	**land**
1	Intel	Intelstrasse	9112	Deutschland
2	AMD	Amdstrasse	7474	Deutschland
4	Seagate	Seegasse	2014	Österreich

Table 2.2.: Participants result-set

With this definition, further information can be retrieved. The correct result should have two rows and five columns. For the evaluation of the results, the number of rows is the first aspect which has to be checked. If this matches, the content can be compared with the result-set the participant's query provides. It can be seen that the result-set which is provided by the student's query has three rows. Therefore, the query is partly wrong, and the result can either be false, or the number of points can be decreased.

If the number of rows and columns match with the correct data set, the content of the table has to be compared column by column and row by row. It is important to trim each value and compare take care of upper-case and lower-case comparison. If all data elements match, the two result-sets are similar and the answer is evaluated as correct in the learning scenario and the full points are provided. If the number of rows or columns does not match, the result can be compared and it can be checked if the correct result-sets are in the provided result, if this is the case, no further points will be decreased from the maximum points for this answer.

It can be seen, that the evaluation of the result in the scenario "quiz" is quite simple because the provided result has to be the same, to get all possible points. If there are more or fewer answers, points will be decreased and if the rows or columns are wrong, further points will be decreased. It would also be possible to only allow correct answers, so if any evaluation part fails, the result will automatically be zero points because the answer is wrong. The evaluation of the second learning scenario is much more complex because not the result-set will be evaluated, but rather the query.

Query evaluation

The second type of evaluation is the query evaluation. Especially in the project mode, it is important to check the query and not the result because a student can build their own schemas. The main task of such projects is to create different types of queries. The result of each query is not as important as the correct query and can not be compared with any predefined results. Such a task for a question can be the following:

The project which should be submitted, should contain:

- Database schema of 4 relations (brands,customer,sales,goods)
- Three queries in terms of Relational Calculus. Each query should operate with 2-3 relations
- Three queries to illustrate the Select, Join and set operations. Each should work with at least 2 relations.

As already mentioned in the scenario, the database schema has to be constructed by the participants of the course. The queries are developed by the participants, and the IBDL should evaluate the result for the given data. To handle the different types of queries and to judge if the given query is of the correct type, a mechanism is necessary which can extract the meaning of the query to determine for instance if a JOIN is used with at least two relations. To do so, some kind of metric or distance have to be calculated to get information about the similarity of a given query in comparison to a defined query which only specifies how a JOIN query should look like but does not contain concrete data like table name or column name.

To evaluate the result provided by the student, there are several methods available. The most common would be a metric to determine the similarity of the two strings - the participants-result and the students result. Such an approach is supported by the Levenshtein distance.

This metric, published by Levenshtein, 1965 calculates the similarity of two strings. This is an important measure to evaluate the queries provided as a string to generate the result for the task. The idea described by Levenshtein is to sum up the steps to get from string to the other. The steps can be deletions, insertions and substitutions of every single character of the string and cost 1. The distance between the two words "data" and "base" will be 3. The higher the distance, the lower the similarity of both words. Navarro, 2001 describes an algorithm using the idea of dynamic programming, which can calculate the distance for two given strings x and y. The algorithm builds a matrix $C_{0..length(x),0..length(y)}$, where $C_{i,j}$ is the minimum number of steps to match $x_{1..j}$ to $y_{1..j}$. The computation is done as shown in the formula below.

$$C_{i,0} = i$$

$$C_{j,0} = j$$

$$C_{i,j} = \begin{cases} C_{i-1,j-1}, & \text{if } x_i = y_j \\ 1 + min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}) & \text{otherwise} \end{cases}$$

The mentioned example with the formula above, will result in a Matrix $C_{i,j}$ which is shown in figure 2.5.

$$C_{i,j} = \begin{array}{cccccc} & & \mathbf{d} & \mathbf{a} & \mathbf{t} & \mathbf{a} \\ & 0 & 1 & 2 & 3 & 4 \\ \mathbf{b} & 1 & 1 & 2 & 3 & 4 \\ \mathbf{a} & 2 & 2 & 1 & 2 & 3 \\ \mathbf{s} & 3 & 3 & 2 & 2 & 3 \\ \mathbf{e} & 4 & 4 & 3 & 3 & \boxed{3} \end{array}$$

Figure 2.5.: Levenshtein example matrix

It can be seen that the Levenshtein distance provides a good possibility do determine the similarity of two given strings. If all attributes of the database schema are predefined in the task for a certain project, the Levenshtein distance should be very small, because there is less tolerance. In some projects, the attributes of a database schema can be freely chosen by the participants. Therefore, the query which is defined by the tutors can not contain the correct database, table and column names. The distance of the query has then to be adjusted because the pre-defined query can differ in the name of the attributes which results in a higher value of the distance.

2.5. Unique features

The internet based database laboratory is a unique system which can not be compared with online database management systems as mentioned in section 2.5.1. The entire IBDL system provides a lot of possibilities which made the system a unique e-learning extension for database courses. This system can be categorized as stand-alone web service which supports database lectures with any available e-learning environment. The SOAP interface provides the possibility to access the service from every environment which is used to handle courses. This gave the participants the possibility to work with their familiar systems but use all the comfort features the IBDL system provides. There are several aspects, like the evaluation of the results, the creation of the databases and their concerning dumps, the logging and much more which can not be done with already existing systems.

The system can not be compared to classical online database management systems like phpMyAdmin or Adminer. It is a newly developed database course support system and provides completely different features to manage the databases for a special course then the mentioned software tool. It has to be mentioned that the IBDL can manage the database system online, but this is not the intention of the system. To prevent the system from handling the whole database management system, the IBDL is unable to handle the core table of the database management system, which can be done with other online management system.

In the next section, the difference between the internet based database system and an online database management system are described to show the difference between the two types of environments.

2.5.1. IBDL vs. phpMyAdmin

On of the most used online database managment tools is phpMyAdmin, 2017. The phpMyAdmin team define their system as following:

> phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

The definition shows that phpMyAdmin is a MYSQL management tool, which allows handling different operations on databases. The system supports nearly every MYSQL command and most of them can be executed with a graphical user interface. The system is able to create database schemas, create tables, manage content, manage users, manage privileges, measure performance,

Features	**phpMyAdmin**	**IBDL**
database management	yes	yes
backup & restore	yes	yes
database querying	yes	yes
GUI for all actions	yes	no
web service interface	no	yes
result evaluation	no	yes
result logging	no	yes
automatically create databases from template	yes	yes

Table 2.3.: IBDL vs phpMyAdmin

show configuration variables and handle different languages. The usage of the system is very high because nearly every web server, which is mostly based on an Apache web server can run the system, and the developers are able to handle their database with little afford and no additional tools. The system itself can be compared to the MYSQL Workbench which also allows to managing a database, but it has to be installed on the server to access the database. phpMyAdmin does not support any interface to communicate with it over SOAP or Rest, but it is possible to create different users for it, and manage their responsibilities.

As from the description above, it can be thought that phpMyAdmin is something similar to the IBDL system. But, this is not the case. Of course there are several features where both systems overlap each other and basically both system can do similar things, but they are both made for different use cases. phpMyAdmin is the number one software for handling database related stuff and is mostly used by developers for managing their database. The tool can be used to enhance the IBDL system and manage the created databases, but most actions will be more complicated.

The process of creating databases for course participant is much more complicated then with IDBL, which is especially made for things like that. Also, the act of logging and result evaluation is not available in the phpMyAdmin environment. Table 2.3 shows a brief comparison of both systems. It can be noticed that for the use case "database courses" the IBDL system is essential because of the availability of the provided web service interface. Without this interface, it is not possible to combine the features of a database management interface with an already approved e-learning environment. There is no way, to integrate phpMyAdmin functionality into an existing service because phpMyAdmin is a stand-alone tool to manage MYSQL database. It would be possible to develop an extension which can provide a web service and then execute the queries on the database. But then, two tools have to be maintained. Therefore, the better solution is the IBDL system, which is especially implemented for the topic of database courses in combination with existing e-learning environments.

Summarized it can be said, that the internet based database laboratory is a unique system, which core functionality is especially made for database courses. It can not be compared to online database management tools because IBDL is made especially for handling database courses and to provide a web interface where students are able to train their database knowledge and execute their queries. IBDL is developed to support all needs which raise in the context of database laboratories and is able to provide solutions for every requirement.

2.6. Requirements

The web service should satisfy all requirements which are necessary for relation with internet based database laboratory. All these requirements were defined in addition to the currently active e-learning environment WBT-Master and the database courses at the Technical University Graz. That's why the following requirements were defined:

- Initialising a database environment by the lecturer
- Execution of queries given by the participants
- Definition of the available database operations
- Specification of user permissions by the lecturer
- Possibility to filter queries
- Possibility to evaluate result
- Return a detailed result of the execution
- Reset to initial state of the database

Every mentioned requirement is specified in a more detailed use-case in the following section.

Initialising a database environment by the lecturer

At the beginning of each laboratory, the lecturer should be able to define a default database if necessary. This database can be used by the participants for further use, querying the data for example. After that, the service has the possibility to save a dump of the database for another restore or initialization.

Execution of queries given by the participants

It will be possible to create, modify, databases and schemas as well as put and query data in the database. So all possible database operations should be possible. The system gets the query from the soap interface and will check it. Therefore, a filtering and an execution of the operation via MySQL will be done. The lecturer and the participants of the laboratory will get the result and the errors they made. A detailed description of the error should be returned. Definition of the available operations or user permissions by the lecturer. The lecturer has to set up the available operations. So he can only allow SELECT queries and block UPDATE queries for instance.

Definition of the available operations or user permissions by the lecturer

The lecturer has to set up the available operations. So he is able to only allow SELECT queries and block UPDATE queries for instance. Another advantage will be the fact, which in earlier stages of a database course, the participant cannot ruin their work with commands they did not know. Operations which can destroy the whole service have to be protected. Therefore, a kind of user management has to be implemented, because root privileges for every participant will be a security problem.

Return a detailed result of the execution

Every SOAP request to the web service will return a response. It will contain a detailed result of the execution. If the request only interacts with the web service alone, so that a MySQL call will not be done, because of a bad query for instance, the result will contain only the web service error. If a MySQL call is executed, the web service result and additional the message from the database management system will be sent back in the response. So the feedback will contain the normal MySQL execution status and additional information from the web service itself. This has to be done because there could happen an error in the web service but MySQL query is still fine or vice versa.

Reset to initial state of the database

In case of an error or a broken database, the lecturer can send a message via SOAP to the web service, which will then reinitialize the database schema to the beginning. So this will be needed in case of a wrong permission or queries which will ruin the schema. Normally this function does not have to be used, but in the case of emergency, the system can be rebuild very easy.

This list shows all requirements for the internet based database laboratory system. All other functionalities like course administration, assignment creation and so on, will be done in already implemented systems like WBT-Master which is already used in several courses at TU Graz. This system will then interact with the web service via the mentioned SOAP interface. Therefore, an accurate description of the system has to be defined.

2.7. Architecture

The system itself is a Java web service with SOAP interface and MySQL connection via the database connector to a MySQL server. The main part of the system is the web service. The figure bellow will give an overview of the system architecture.

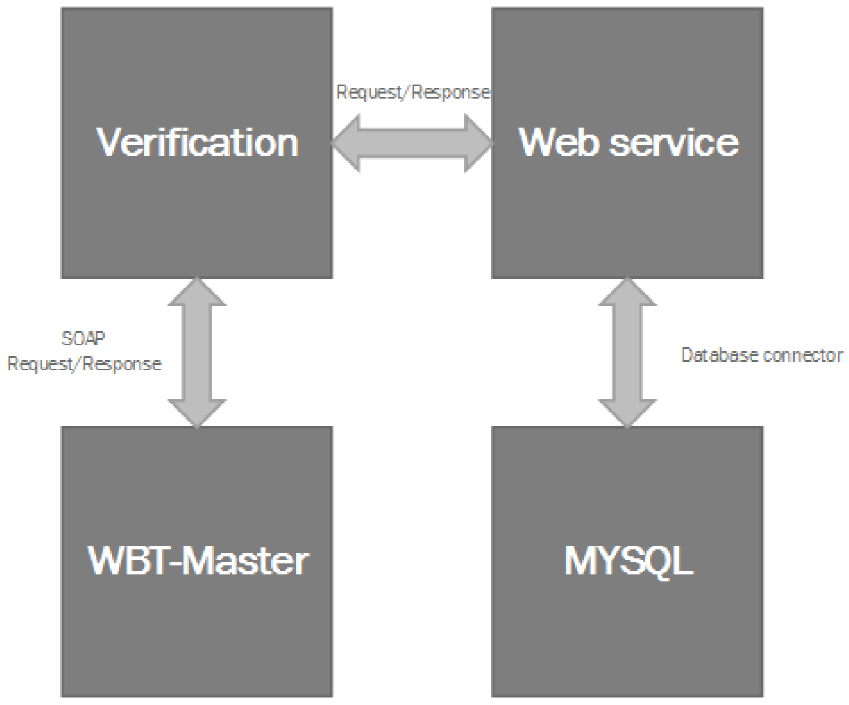


Figure 2.6.: System architecture overview

Figure 2.6 shows an abstract view of the whole system when it will be used in an internet based laboratory in a database course for instance. The course tool will be connected to the web service which is connected to a database. The course tool and the MySQL tool are given environments which must be accessed by the service. To do so, the service has several components which will manage all the different parts. The communication with the web service is done via SOAP because it serves many possibilities for handling the requirements. Another big advantage is the XML communication which can be handled by various other systems and therefore a high compatibility is given.

Web service

There are several web service technologies out there. One of the most often used is the Java Technologies for Web Applications which is fully specified by Oracle, 2006. The web service is a Java Web Application with several components which will be explained in this section. Therefore, the service provides a Java servlet, which manages all the services the system provides. Another advantage of the use of this system is that Java Web also supports the use of MVC pattern in applications. So it is possible to develop an administration interface above the system to manage the configuration by the supervisor team. Of course, it will be possible to manage all configurations via the SOAP interface too.

To deploy such web services there are in principle to a main server system which can handle the Java Technology for Web Applications. The first one is *Apache Tomcat* 2016, which is more like a Java servlet container and does not need a lot of computing power but does not support the whole Java Enterprise Edition. The second one is called Glassfish, which supports the whole environment. Glassfish is also used for the reference implementation of the Java EE. Both can be integrated into a lot of IDEs for development purpose.

To do so, the implementation uses the Java API for XML Web Services (JAX-WS) which is a technology for building web services that communicate over XML. The technology is specified by Oracle, 2017b. For the communication with the database system, a connector is used. Therefore, the figure 2.7 shows the main three parts of the service and the responsibilities of every single section are mentioned.

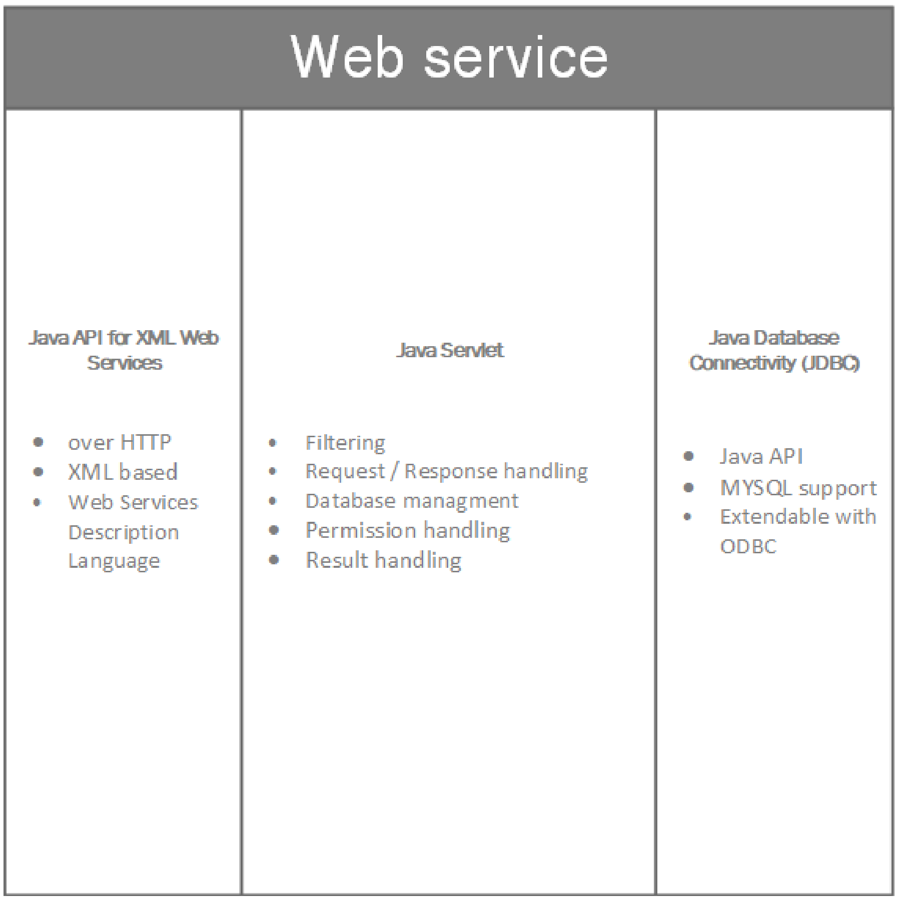


Figure 2.7.: Web service overview

Figure 2.7 shows an overview of all parts of the system. The servlet as part of the Java Web Application is the main part of the web service. The first thing the web service will do with an incoming request, which will be handled by the SOAP part of the service is a filtering of the query. Therefore, a special protection filter will analyze the parameters of it and escape them, so that there is no security vulnerability and the system does not get hacked. In the next step, the system has to check if the incoming call has the correct permissions to execute the query. The web service needs a little redundant storage in the existing database, to save information like this. Information about the servlet admin user is stored in the database. A big aspect of the whole system is, that most of the information for the web service can be retrieved from the database setup. This is a big advantage because MySQL can already handle different permissions and user levels per default and so the administration effort is much less, than a system which has its user and privilege management. The check of the available operations can be done only by the database management system.

But the web service itself needs to have two privilege levels. One is the level supervisor, which allows to change settings, create new labs and has root access to the database. The other level is the user privilege level, where only the common MySQL statements are allowed which are defined on the MySQL system itself. So the user does not have to know anything about the web service because he should feel like he is working directly on the database. This is important because a participant in the lab should train with real systems.

Another aspect of the servlet is the connection to the database connector. All filtered and escaped queries should be transmitted to the database connector which can then do further activities. If there is an error at this early stage, an error is returned and the next execution steps do not work. This could be the case if the database server is not running or there is a problem with its connection. If everything works and the servlet was able to send the query to the database connector, then the system goes into the next phase. The execution phase, where the query will be executed on a working MySQL database management system via the Java Database Connector (JDBC).

Verification

The verification process of the system describes the ability of the system to handle the main three security aspects. This is necessary because only verified clients should be able to use this web service and no-one else can abuse the service for any other purpose. There are three security mechanisms which were used in this project. The first aspect is integrity, which describes the fact, which it should be proved that the message which was sent by the client is the same which was received, or was the message really send by the expected client. There has to be some protection mechanism which signs the requests/responses to guarantee this aspect. The second aspect is authentication, which defines that only web services which can authenticate on the server, are able to use the system. The third aspect is confidentially, which describes the encrypted and decrypted of the XML content which is sent between the client and the server. All three security measurements are solved as follows:

Integrity To get integrity in the system, the SOAP messages for date exchange have to be signed. Therefore, several libraries are available which provide tools for signing the XML request. It would also be necessary to modify the client to be able to use such libraries, hence this have to be integrated into WBT-Master.

Authentication Because the project is developed for *Apache Tomcat* 2016 server systems their preferred authentication method can be used. The web service uses a defined security role, so the Tomcat server knows, that authentication has to be offered. The client itself is able to identify via the provided API key.

Confidentially The *Apache Tomcat* 2016 module which is used to provide authentication also supports the use of HTTPS. This means the transport encryption is solved by the use of the tomcat module. But it is possible to use XML encryption but this is only necessary if more web services are evolved, because otherwise the message have to be encrypted/decrypted at every web service.

2.8. Integration

There are several ways for the communication process between the web service and a client. All exchange protocols have different advantages and disadvantages. There are two common types which are used in the most web services nowadays.

The first one is the Representational State Transfer (REST), which describes a pattern which was first mentioned by Fielding, 2000. It is important to mention that REST is not a standard but only an architecture. It is used by many web service providers and is used to call a resource on an external server. To do so, RESTful web services use HTTP/HTTPS to fulfill all create, read, update, delete operations.

The second big thing is the Simple Object Access Protocol (SOAP) which is a protocol specified by the World Wide Web Consortium (W3C). It uses XML over HTTP or SMTP for the data exchange process. All requests and responses are sent in a valid XML message and the client has to exactly fulfill the specification the server describes via a web service description language document. Table 2.4 shows an overview of the differences of the two exchange formats.

REST	**SOAP**
Architecture	Protocol
Many data formats (JSON, XML,. . .)	XML
HTTP / HTTPS	HTTP / HTTPS, SMTP
High performance	Slower because of overhead
Caching	No Caching
Widely used	Usage in complex web services
Simple communication process	Strictly used Standard for communication process

Table 2.4.: REST/SOAP comparison

The choice for this service is SOAP because the standardization of the protocol by the W3C allows a simpler integration into web clients. A REST interface can be easily added in future versions of this project because the modular system architecture allows the usage of several data exchange types at the same time.

The Java API for XML Web Services (JAX-WS) is the other main part of the web service. This API allows a programming language independent data exchange between different services. So data exchange and remote procedural calls are available to every system over a network transfer. For this project, the SOAP protocol over the HTTP layer, which is defined as SOAP 1.1, will be used to transfer response and request over the internet. To process a request, several steps have to be done.

In the first step, the client has to access the provided Web Services Description Language file. This file is available on the web service and is a standardized way to read the available methods. Such files contain the parameters and calling conventions of every single method the web service provides to the client. The WSDL file contains per definition of W3C, 2001 the following parts:

- Types– a container for data type definitions using some type system (such as XSD).
- Message– an abstract, typed definition of the data being communicated.
- Operation– an abstract description of an action supported by the service.
- Port Type–an abstract set of operations supported by one or more endpoints.
- Binding– a concrete protocol and data format specification for a particular port type.
- Port– a single endpoint defined as a combination of a binding and a network address.
- Service– a collection of related endpoints.

The WSDL file for this project can be found in the appendix A.1. It can be seen that the whole document uses XML to describe the content. The elements of this XML file are specified in the XML Schema Definition (XSD) which is also available in the web service as noticed in section 3.4.4. The whole file is specified in addition to the SOAP protocol, so it is fully compatible with it. After the client got this file, it is able to send requests to the service. Therefore, the SOAP protocol over HTTP has to be used. In that second step, the client generates a request which has to be in a special form.

Simple object access protocol (SOAP)

To use SOAP for the communication between the clients and the web service is a big advantage over other systems because SOAP is a very good specified protocol where the developer only has to fulfill the specification by the World Wide Web Consortium (W3C) and many clients with various programming languages can communicate with the service. For the communication process, several aspects have to be fulfilled by the client in a specified format. Therefore, SOAP consists out of three parts defined by W3C, 2000:

- The SOAP envelope construct defines an overall framework for expressing what is in a message; who should deal with it, and whether it is optional or mandatory.
- The SOAP encoding rules defines a serialization mechanism that can be used to exchange instances of application-defined data types.
- The SOAP remote procedure call representation defines a convention that can be used to represent remote procedure calls and responses.

Every part of a SOAP request is also defined by the W3C, but the main thing is the SOAP Envelop, which is the most important part off the communication process. The envelop is defined by W3C, 2000 as follows:

> A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body. This XML document is referred to as a SOAP message for the rest of this specification. The name space identifier for the elements and attributes defined in this section is "http://schemas.xmlsoap.org/soap/envelope/".
>
> SOAP message contains the following:
>
> - The Envelope is the top element of the XML document representing the message.
> - The Header is a generic mechanism for adding features to a SOAP message in a decentralized manner without prior agreement between the communicating parties. SOAP defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory
> - The Body is a container for compulsory information intended for the ultimate recipient of the message. SOAP defines one element for the body, which is the Fault element used for reporting errors.

As mentioned by the W3C, the SOAP protocol is very widespread and can be used for several use cases. All parts of this Envelope are necessary for a communication process. But it is not compulsory to have any detail information in these parts. Most often it is enough to use the default parameters for the system because they cover most aspects.

In the SOAP Header attribute covers the information of the encoding style, as well as the must understand attribute and the actorAtrribute. The first one describes if the header information is important and the recipient. The second attribute covers the information about the destination of a request. Because the SOAP message can travel around multiple destinations and forwarded to other recipients and this attribute describes, who is responsible for the processing.

The Body of a SOAP document is the main part of it. It covers all exchanging information as well as the RPC calls and the error reporting. All child elements of the SOAP body are identified by their qualified element name. The encoding Style which can be set in the header is used to encode the information. Another element which can be used in SOAP is the FAULT element. It can be used to carry an error or status information in a SOAP message. If used, the FAULT element must be used in the body element of an SOAP envelope. The W3C specifies all primitive data types plus structures like lists and arrays on their SOAP specification. This information is enough for the exchange of objects and RPC.

All the mentioned SOAP aspects are used to define the necessary SOAP Envelopes for this system. Because this specification is responsible for the function of the whole system, this detailed definition is necessary to have a well working system.

2.9. Database interaction

The communication between the web service and the database is done by the Java database connectivity. This Application Programming Interface (API) is supported by MySQL and defines how a Java application can access the database. The JDBC API is part of the Java Standard Edition and provides a call-level API for SQL-based databases.

One big advantage of such an API is the integration of the database management system in the Java application. There are several tiers which allow the developer to change the type of the database for instance from MySQL to Microsoft SQL and the driver can handle this. So the developer has his own interface for communication and does not have to know anything about the database. Every tier can be exchanged with any other compatible tier. The only problem appears, if the developer uses system-specific SQL queries which do not work on other structures, therefore the developer has to use the standard SQL commands. But for this system, all SQL commands can be handled by JDBC, and it would be possible to change MySQL with other database management systems. So the usability of the system is very widespread and it can be used in different labs with different database management systems.

Another possibility for the database interaction would be a self-made database connector. This connector can be placed in the middle of the communication process between the web service and database. But for the interaction in this project, the available JDBC connectors are pretty fine.

2.10. Summary

The service pushes internet based database laboratories to a new level because the participants are now able to directly execute their tasks in a real database management system. They are able to get the result of their work as soon as possible, generated by the system and they also got the advantages, that they receive a detailed result of the error if their queries fail. To do so, this service connects to existing laboratory clients via SOAP over the HTTP layer, so it is possible to integrate the web service in numerous available systems, like the WBT-Master, used in several courses at TU Graz.

The supervisor of the course has the advantages which the results of a query are sent back to the participants, with detailed error information about the failure. The administrative configuration of the service is very simple, to guarantee a pretty fast workflow. The aspect, which the service should feel like a real MySQL system for the users, will always be kept in mind during the development phase because this is very important to train the participants for real world system they will find in their daily business. So there should be no difference, except the detail of the error messages, in comparison to a stand-alone database system. The mentioned SOAP messages, which are described with the web service description language in the WSDL document, are pretty detailed specified because this is important for the further integration of this system in a laboratory client. To use SOAP is a good decision, because it is specified by the W3C, and therefore there are no troubles with 3rd party clients, because they only have to act like specified.

The security aspect of the service is important because there have to be at least two types of users. The supervisor need root privileges to configure everything, whereas the participants only have normal user privileges. The incoming queries can be pre-filtered, to avoid any security related issues by some bad queries.

3. Implementation

In this chapter, the implementation process of the internet based database laboratory (IBDL) is described. In this first section, a general description and an overview diagram to describe the architecture of the implementation will explain the functionality of the system. The following sections describe the important parts of the system.

The internet based database laboratory is fully implemented in Java, 2017. Therefore, Java-web technologies are used. The decision to use Java was made because WBT-master was also implemented in Java and is running on a Apache Tomcat Server as described in 1.5.5 and therefore no extra server is needed. The whole project was build under the aspect of the Model-View-Controller (MVC) pattern and is therefore very adaptable to new requirements. The MVC-pattern is done by pure Java and does not use any external libraries. One big aspect of the development was the fact that only a very few third-party libraries should be used. Because of that fact, only the JSqlParser, 2017, and the MySQL, 2017 have to be used.The first one is used for pre-filtering and sand-boxing described in 3.6. The second library is the standard library for the communication process between MySQL and Java. The system is compatible with every available database management system and can simply be modified to work with on of these, but this basic implementation only supports the MySQL. The process for changing the database management provider can be found in section 3.2.

In 3.1 the important Java components of the project are mentioned in about the package it is located. Because of the MVC-pattern, each package name defines the type of the class, regarding model, view or controller. This diagram will give an overview of the responsibilities of each component of the system. Detailed information can be found in each section of this chapter.

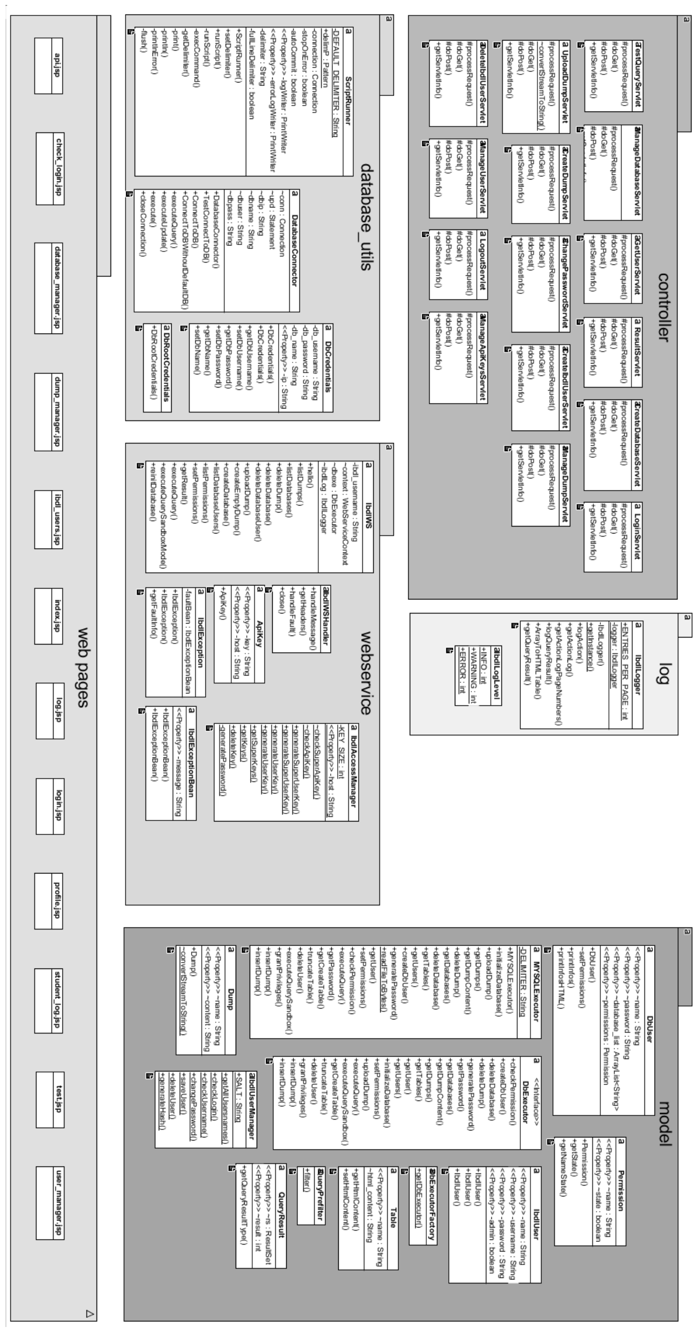


Figure 3.1.: Class diagram

3.1. Definitions

To understand the following chapters, and to get an idea of the work flow of this web service some definitions have to be made. This definition will allow the user to simplify the usage of the system.

Dumps The first thing which has to be created in the system is a dump. A dump can either be uploaded, so the course supervisor creates a database schema and upload it with a specific name. Another possibility is to create an empty dump with a specific name. The dumps are in principle the basic database for everyone.

Database To provide a database schema to every single WBT user, a database has to be generated. This means every single user has his own environment. The naming schema for the database is given by the dumpname and the username and results in {dumpname}_{username}. Therefore, it is pretty simple to identify each database.

Database user Whenever a dump gets uploaded, a MySQL user with the dumpname is generated. This database user is then responsible for all created databases starting with {dumpname}. All privileges set for this user, affect all the privileges for the WBT users using a databases starting with {dumpname}.

All mentioned definitions give a basic understanding of the system but will be explained in more detail in the next chapters. Every single functionality will be explained and also the code for important parts of every class or module will be shown.

3.2. Database executor

One of the most important parts of the system is the DatabaseExecutor object. It is responsible for all database process in the system and also handles every action initialized either by the web service or the GUI. To use the system with some different database management systems, the DatabaseExecutor defines an interface which methods have to be provided by every single executor for the different management systems. To choose the needed executor, there is an DBExecutorFactory which returns an instance of the needed DatabaseExecutor.

Therefore, the static Method getDbExecutor have to be called with a string as a parameter which defines the needed DBexecutor object as mentioned in Listing 3.1.

```
public class DbExecutorFactory {

        public static DbExecutor getDbExecutor(String dbType)
           {
                if (dbType == null) {
                   return null;
                }
                if (dbType.equalsIgnoreCase("MySQL")) {
                   return new MySQLExecutor();
                }
                return null;
        }
}
```

Listing 3.1: DBExecutorFactory

To handle all different database management systems, the DatabaseExecutor defines an interface which defines all needed methods. All these required methods have to be implemented, to guarantee the same functionalities by the system as are available through MySQL. These methods, which are defined in this interface only execute several queries on the database to maintain this web service. The implementation for a MySQL database management system can be found in the MySQLExecutor which implements DBExecutor. A short explanation can be found in the following descriptions.

checkPermission

Checks if permission is in the array permissions given in the parameters.

createDbUser

Creates a new database user with the name of the provided username. Therefore, username and a generated password are saved in the table 'ibdl'.'db_user'.

After that, a MySQL user is created with the CREATE USER statement. In case of success true is returned, otherwise false.

deleteDatabase

Drops the database with the name provided in the parameter via the MySQL DROP DATABASE command.

deleteDump

Deletes dump stored in 'ibdl'.'dump' given the name of the dump from the method parameter selected_dump.

deleteUser

Drop MySQL user identified by the name of the user. The MySQL statement DROP User is used.

executeQuery

This method is one of the most important methods used in this project. The execution of the user provided query is done here. The method requires the WBT-Master username, the database name, and the query. All these parameters are checked. After that, the MySQL user for this database has to be retrieved from the database ibdl.db_users. If the user for the MySQL connection is available, new DbCredentials are generated and a database connection with the given credentials is made and the query is executed. After that, the QueryResult is checked and the output gets generated. This is done like described in 3.2.

```
DbCredentials dbc = new DbCredentials(db.getName(), db.
    getPassword(), database + "_" + web service_user, "
    localhost");
DatabaseConnector dc = new DatabaseConnector(dbc);
dc.ConnectToDB();
```

```
qr = dc.execute(QueryPrefilter.filter(query));
Object[][] result;
if (qr.getQueryResultType()) {
    qr.getRs().last();
    int rows = qr.getRs().getRow() + 1;
    qr.getRs().beforeFirst();
    ResultSetMetaData rsmd = qr.getRs().getMetaData();
    int columns = rsmd.getColumnCount();
    result = new Object[rows][columns];
    for (int i = 0; i < columns; i++) {
        result[0][i] = rsmd.getColumnLabel(i + 1);
    }
    int j = 1;
    while (qr.getRs().next()) {
        for (int i = 0; i < columns; i++) {
            result[j][i] = qr.getRs().getObject(i + 1);
        }
        j++;
    }
} else {
    result = new Object[1][1];
    result[0][0] = qr.getResult();
}
dc.closeConnection();
return result;
```

Listing 3.2: execute query

The returned result contains either the number of effected rows or the retrieved rows where the name of the rows can be found at index zero of the returned array.

executeQuerySandbox

This method is for executing a query without an effect on the database. But the query should be executed on the real table. After the execution the table looks like before. Therefore, it is essential to get the type of the query because only DELETE, INSERT, UPDATE queries are supported. The first step makes a copy of the tables which should be modified. After that, the query is executed and the result stored. In the next step, the tables get truncated and the backup is restored.

generatePassword

Used to generate a save password, therefore a SecureRandom object is used. The specified length in the parameters is responsible for the length of the password.

getCreateTable

Retrieve the "SHOW CREATE TABLE" command for the table identified by the name available in the parameters.

getDatabases

Retrieve a list of strings containing the names of all databases on the MySQL server. Tables ibdl, information_schema, mysql, performance_schema and sys are hidden because of security measures.

getDumpContent

Get a dump object of the dump specified by the name, containing the name of the dump and the content.

getDumps

Get a list of all availble dumps as Dump objects stored in table ibdl.dump.

getPassword

Returns password of an database user stored in table ibdl.db_user for user with username specified in the parameter of the method.

getTables

Returns list with all tables existing in a database identified by the name of the table.

getUser

Returns a user as DbUser object for user available in mysql.user identified by the username given in the parameters.

getUsers

Returns a list of all available users stored in mysql.user.

grantPrivileges

Grant all privileges for a user to the database specified in the parameters. Also, the username is given in the parameters.

initializeDatabase

This method is used for initialising a new database. Therefore, the dump, the dump name and the username of the WBT-user have to be provided. The method creates a new database with the name {dump name}_{username} and the dump content is executed with ScriptRunner and can be found here The method have to return true if successful and false if an error occurred.

insertDump

Insert given dump content and dump name into table ibdl.dump.

setPermissions

Set all permissions given by the grants array for a user. If the permission is part of the array, then it has to be set with the statement GRANT, if not it has to be unset by calling the MySQL statement REVOKE.

truncateTable

This method is used for truncating the table specified in the parameters. To do so the TRUNCATE statement is used.

uploadDump

This method saves a dump to the database. Therefore, the method insertDump is called, which saves the dump as blob to the table 'ibdl'.'dump'.

All mentioned methods in these class are implemented and generate the expected results. To do so, the class needs to implement a database connection. This connection is defined in the next section. This database connection is essential for all further connections needed in this project.

3.2.1. Database Connection

The database connection is an important component of this system. Without a reliable and manageable database connection, most of the operations of this system would not be possible. Therefore, this service uses the official and most common way for database communication between MySQL and Java. The Java database connector (JDBC) driver for MySQL, 2017 is provided by Oracle and supports all needs for handling the database communication process.

Because the web service either needs a MySQL account for handling all web service related stuff, for instance for creating new databases it also needs the MySQL account which runs in the context of the used database. Therefore, a new class DbCredentials is introduced which is used for providing the IP, database name, username and password. The credentials were simply stored in this object mentioned in listing 3.3

```
public DbCredentials(String db_username, String db_password,
    String db_name, String ip) {
        this.db_username = db_username;
        this.db_password = db_password;
        this.db_name = db_name;
        this.ip = ip;
}
```

Listing 3.3: DbCredentials

To work with root credentials, the class DbRootCredentials is available. It extends the DbCredentials class and contains the MySQL authentication data which has access to the web service back end. The credentials are described in 3.4. If the password of the MySQL user 'ibdl' has changed, it has to be corrected in this file. These credentials are only defined here.

```
public class DbRootCredentials extends DbCredentials {
        public DbRootCredentials() {
                super();
                this.setDbUsername("ibdl");
                this.setDbPassword("password");
                this.setDbName("ibdl");
                this.setIp("localhost");
        }
}
```

Listing 3.4: DbRootCredentials

The root credentials will be used in all methods in the MySQLExecutor except the executeQuery() and the executeQuerySandboxMode(). These two methods use the credentials of the MySQL user which is created for a specific database.

The class DatabaseConnector is responsible for the database communication process. This class awaits DBCredentials in the initialisation and provides a couple of methods. The method which is used most is shown in listing 3.5. This method connects to a database and returns an open connection. If the connection failed, the method returns null and the thrown exception can be found in the output. There is no logging of such events in the IBDL log, because if there is no database connection, no logging can be done. All further methods will fail if this method is not called at the beginning of a database operation.

```
public Connection ConnectToDB() {
    Driver mySqlDriver;
    try {
        mySqlDriver = (Driver) Class.forName("com.mysql.jdbc.
            Driver").newInstance();
        DriverManager.registerDriver(mySqlDriver);
        conn = (Connection) DriverManager.getConnection("jdbc
            :mysql://" + dbip + ":3306/" + dbname + "",
            dbuser, dbpass);
        upd = (Statement) conn.createStatement();
        System.out.println("Connection␣established!");
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(DatabaseConnector.class.getName())
        .log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
       Logger.getLogger(DatabaseConnector.class.getName())
        .log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        Logger.getLogger(DatabaseConnector.class.getName())
        .log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(DatabaseConnector.class.getName())
        .log(Level.SEVERE, null, ex);
    }

    return conn;
}
```

Listing 3.5: ConnectToDb

This method has to be called every time a query on the database should be executed. Otherwise, there is no open connection. It has to be mentioned that this connection uses the database specified in the credentials as default table.

If a connection should be made to an undefined table, there is a method called ConnectToDBWithoutDefaultDB().

To execute a query, three different methods are available. The first one is executeQuery() which is used whenever some response is available. Typical here will be a SELECT query. The second one is the executeUpdate(). This method is used whenever something in the database gets modified, but the response is only the number of effected rows. Typical statements here would be a UPDATE query. The third method for execution is the method execute(), which returns the custom object QueryResult. This method should be used whenever it is not known what query should be executed. The QueryResult which is returned contains the result.

If a ResultSet is available, then it can be retrieved out of this object. The ResultSet is only disposable if there are datasets returned by the query. If no ResultSet is available, the number of effected rows is set in the QueryResult object. This is down as described in listing 3.14.

```
public QueryResult execute(String query) throws SQLException {
    QueryResult qr = new QueryResult();
    if (upd.execute(query)) {
        qr.setRs(upd.getResultSet());
    } else {
        qr.setResult(upd.getUpdateCount());
    }
    return qr;
}
```

Listing 3.6: execute

As already mentioned, this method should only be used, if the type of the query is unknown. This is the case when the web service client sends a query which should be executed.

One important thing is to close the connection after all execution processes are done. To close a connection, the method closeConnection() have to be used. It is important to close the connection because if the service has to many open connections, the MySQL performance slows down. This class is the main interface for communication process between MySQL and Java. Because of the JDBC driver by MySQL, 2017 used in this web service it is very simple to use other database management systems. The Java Database Connectivity

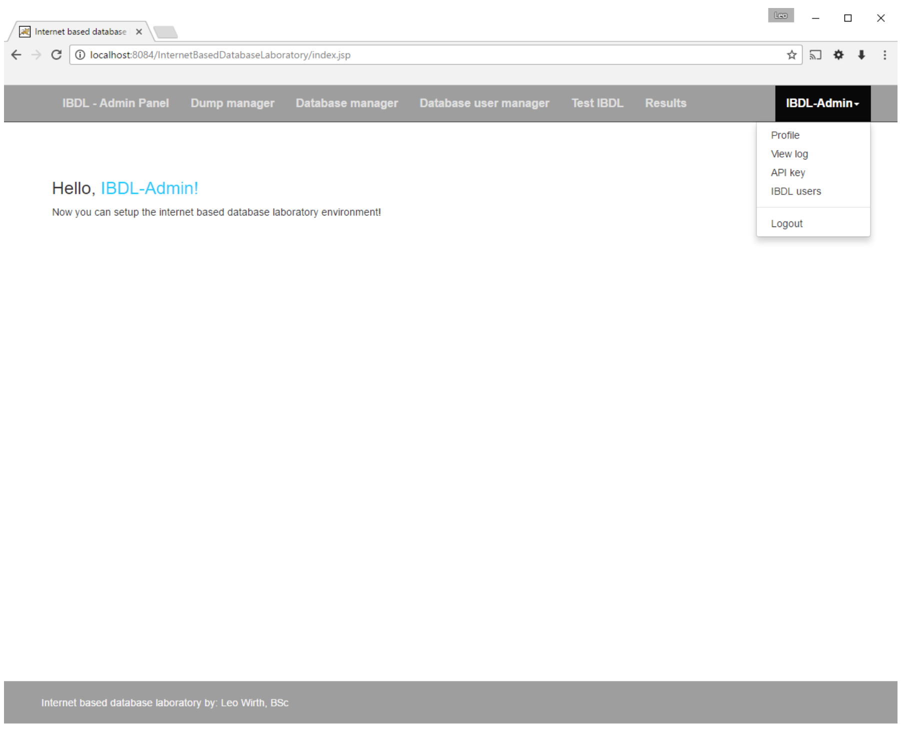
API specified by Oracle, 2017c describes the process of changing the database management system as simple, because only a few lines have to be changed. In this web service, it is only necessary to create a new database connector and load the available JDBC driver for the used database system.

All necessary parts concerning the connection section of the web service are explained. The next chapter will describe the interaction with this mentioned logic. There are in principle two channels to communicate with the web service. On the one hand the graphical user interface, which describes a web application running in a browser and on the other hand the SOAP service which also allows calling every mentioned method.

3.3. Graphical configuration interface

The first method to handle the service is the graphical user interface (GUI). This interface allows the supervisors of the database course to administrate the course environment. It gives the opportunity to handle all settings from one interface. But it has to be mentioned, that all functionalities apart from creating API keys and creating IBDL users are available through SOAP. So it is not necessary to use the GUI but it simplifies the system. The implementation of the GUI is done in HTML with the support of the Framework Bootstrap, 2017. This framework gives the opportunity to create beautiful responsive Websites with little afford. Another aspect is, that no Javascript was used for the realization of the interface.

The interface requires a logon. A default user admin is available, which should be used by the head supervisor of the course. The admin can then create new users, called IBDL-users. It can be chosen if these new users are admins as well. Only admins can create new users. This IBDL-users should be all tutors who were responsible for database courses. After the successful login, the supervisors get the following interface to work with. On the top right corner, the administration options are displayed. The admin can create API keys, which are explained in section 3.4.5 and create new IBDL-users here. It's like a configuration menu for the web service itself. The IBDL-users also have the possibility to check the logs. Every single servlet- and web service- operation is logged in a database table. This is important because of that, every error can



Hello, IBDL-Admin!

Now you can setup the internet based database laboratory environment!

Internet based database laboratory by: Leo Wirth, BSc

Figure 3.2.: Graphical user interface

be reproduced. As already mentioned, the web service uses the model-view-controller pattern and this means that all files for the visualization of the GUI, which are realized with Java Servlet Pages by Oracle, 2017e can be assigned to the view part of the MVC architecture. The graphical user interface consists out of one index.jsp which includes all other views if required. The other files are protected against unauthorized visiting by a header redirect, which checks if there is a valid login session is available.

The decision for using only one page for displaying the content, and including only the views was done because of the simplification of the systems. This means that every page equals another because only the content of the main body of a page gets changed in dependence of the selected menu. Therefore, the index.jsp contains some logic, which can parse the parameters in the URL to include the necessary layout.

There are mainly five sections in the GUI which are necessary for working with the system:

Dump manager

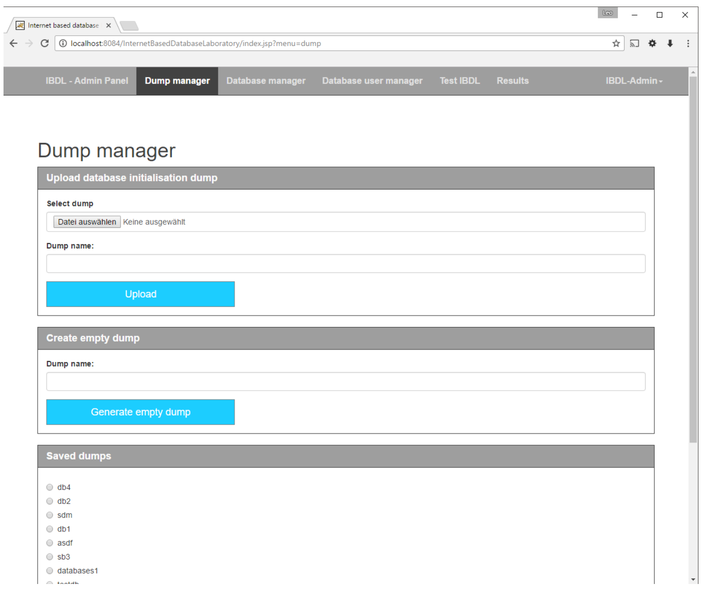
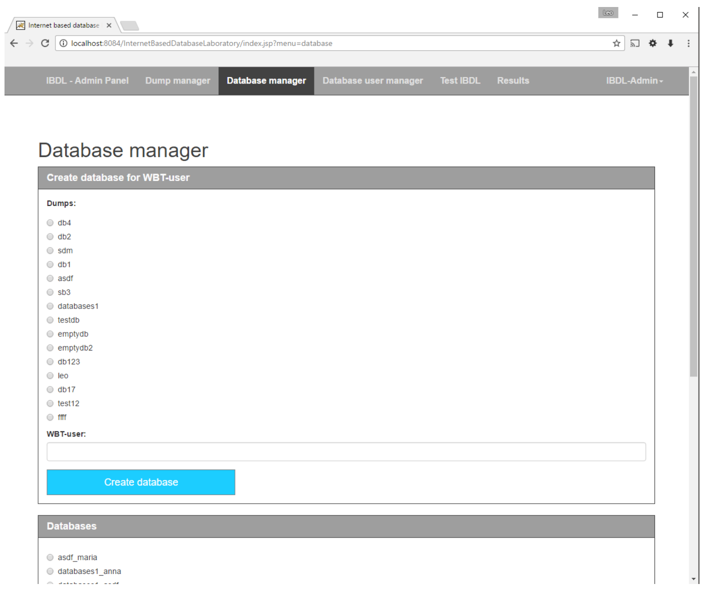


Figure 3.3.: Dump manager interface

The dump manager in figure 3.3 allows the user to upload exported SQL dumps from predefined database schemas. The tutors can create databases in their preferred environment, and upload this framework to the system. Another possibility is to create an empty dump, which can then be used as an empty framework for the students. The dumps can also be viewed and deleted. The code is located in the file dump_manager.jsp.

Database manager

The database manager, which can be found in database_manager.jsp, displayed in figure 3.4, creates the real database schema for a given web service



Database manager

Create database for WBT-user

Dumps:

- db4
- db2
- sdm
- db1
- asdf
- sb3
- databases1
- testdb
- emptydb
- emptydb2
- db123
- leo
- db17
- test12
- ffff

WBT-user:

Databases

- asdf_maria
- databases1_anna

Figure 3.4.: Database manager interface

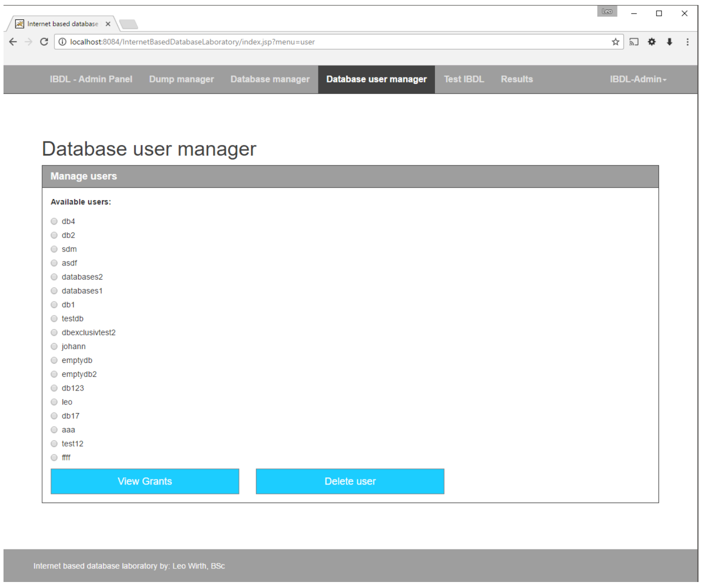
username. Therefore, a dump and an username have to be selected, and the schema will be created in the MySQL environment. It is also possible to view the tables of the schema and to delete databases.

Database user manager

The database user manager in figure 3.5 allows setting privileges for every database user. Every dump gets an MySQL user which is then responsible for all databases created out of this dump. The available privileges are the same as specified by Oracle, 2017h. The source of this manager is implemented in user_manager.jsp

Test environment

The test environment, found in file test.jsp, in figure 3.6 give the possibilities to the tutors, to execute a query on a specific database. This can be useful for



Database user manager

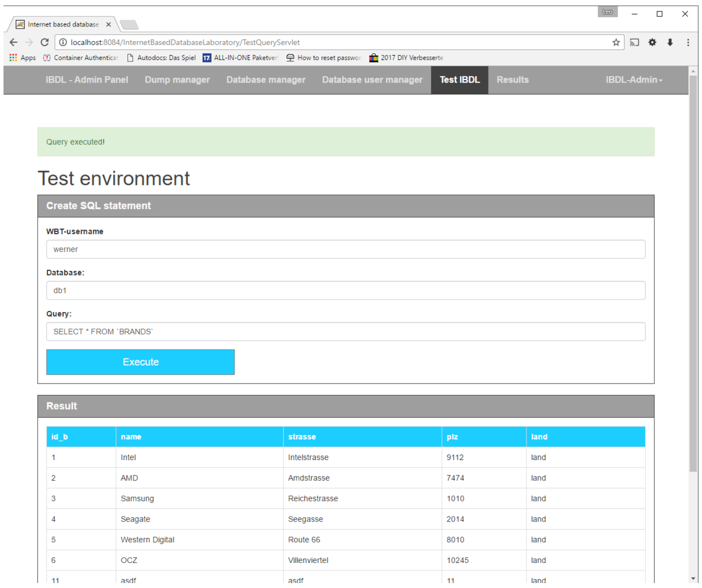
Manage users

Available users:

- db4
- db2
- sdm
- asdf
- databases2
- databases1
- db1
- testdb
- dbexclusivtest2
- johann
- emptydb
- emptydb2
- db123
- leo
- db17
- aaa
- test12
- ffff

Internet based database laboratory by: Leo Wirth, BSc

Figure 3.5.: Database user manager interface



Query executed!

Test environment

Create SQL statement

WBT-username: werner

Database: db1

Query: SELECT * FROM `BRANDS`

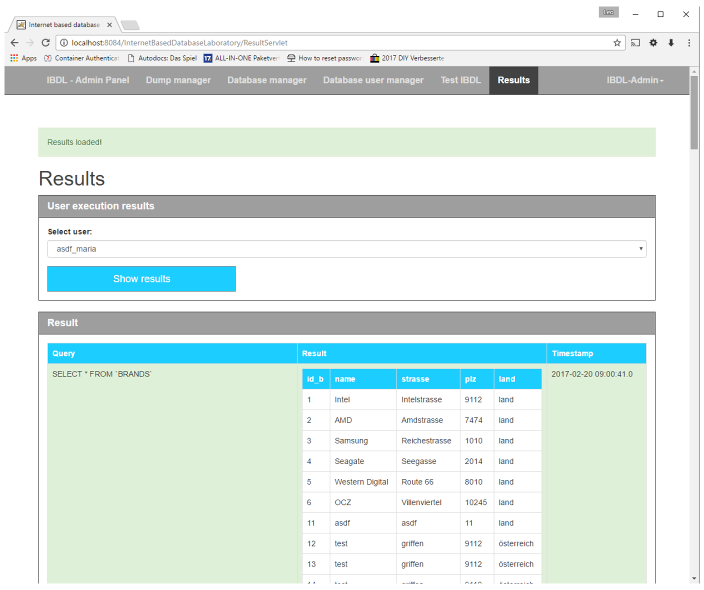
Result

id_b	name	strasse	plz	land
1	Intel	Intelstrasse	9112	land
2	AMD	Amdstrasse	7474	land
3	Samsung	Reichestrasse	1010	land
4	Seagate	Seegasse	2014	land
5	Western Digital	Route 66	8010	land
6	OCZ	Villenviertel	10245	land
11	asdf	asdf	11	land

Figure 3.6.: Test environment

testing a specific problem a user had.

Results



Results loaded!

Results

User execution results

Select user: asdf_maria

Result

Query	Result	Timestamp
SELECT * FROM `BRANDS`	(see below)	2017-02-20 09:00:41.0

id_b	name	strasse	plz	land
1	Intel	Intelstrasse	9112	land
2	AMD	Amdstrasse	7474	land
3	Samsung	Reichestrasse	1010	land
4	Seagate	Seegasse	2014	land
5	Western Digital	Route 66	8010	land
6	OCZ	Villenviertel	10245	land
11	asdf	asdf	11	land
12	test	griffen	9112	österreich
13	test	griffen	9112	österreich

Figure 3.7.: Results interface

The result section in figure 3.7 shows which queries were executed for a specific user and what was the result of the execution. Successful results are marked green, and failed results are red colored. The source of this JSP can be found in the file result.jsp. This basic implementation of the result page can be adapted to the required result evaluation mode.

It can be seen that the GUI provides all features, which were required. Nevertheless, it was important in the development that the GUI is a matter of simplicity, and it has to be mentioned that the development of the GUI was done because the development of the system can be done much simpler.

3.3.1. Java Servlets

Every functionality provided by the graphical user interface is realized via the Java servlets described by Oracle, 2017d. These servlets allocate an elegant way for working with requests and responses. It is possible to pass nearly everything directly from the jsp files, which took over the view part of the MVC-architecture, to the servlet and generate a response. Therefore, the most servlets do not create HTML output but generate a response which is den send back to the calling view and shown. To generate such responses, it is necessary to do it like explained in listing 3.7. It can be seen, that error and success is handled separately in this interface because there are pre-defined error and success fields in the index.jsp.

```
if(success){
    request.setAttribute("success", "<SUCCESS_MESSAGE>");
}else{
    request.setAttribute("error", "<ERROR_MESSAGE>");
}
    request.getRequestDispatcher("<PAGE>")
      .include(request, response);
```

Listing 3.7: response handling

The next section gave an overview of all servlets. Important code segments will be shown in listings, but it has to be noticed, that the code of the listings is shrunk to get an easier understanding of what is happening. The whole graphical user interface will use the following servlets.

ChangePasswordServlet

This servlet has the responsibility to change the password of the logged in IBDL user. Therefore, the request contains the old password, the new one and the reply of the new password. The username can be retrieved out of the user object in the session. After that, all three parameters get checked. In case of an error, like a wrong password or different passwords, an error is returned in the response. Otherwise, the IbdlUserManager is called to change the password. If the manager returns true, the password is changed and a success message is returned.

The mentioned IbdlUserManager is responsible for user related stuff, like login checking, password changing and for creating new users.

CreateDatabaseServlet

The CreateDatabaseServlet contains all the logic for creating a database. Therefore, the username which should be the same as the user got in WBT-Master, and a dump name are necessary Both are available in the request. The parameters are checked if they are not empty and it is also checked if the the dump is available After that a new database is initialised as shown in listing 3.8.

```
if (dbexe.initializeDatabase(dump.getContent(), dump.getName()
    , username)) {
    if (dbexe.grantPrivileges(dump.getName() + "_"+ username,
        dump.getName())) {
        request.setAttribute("success", "Database for dump 
            created!");
    }
}
```

Listing 3.8: creating a new database

It can be seen that in the first step, the database gets initialized and if that operation was successful, all privileges are activated on the database. This means that the MySQL user who has the same name as the dumb, is now responsible for all permissions.

CreateDumpServlet

This servlet handles the empty dump creation process initialized by the dump_manager.jsp. Therefore, only the dump name is needed and it should not be empty. If all aspects are fulfilled, the dump is created. The normal uploadDump() function from DbExecutor is called, with a null pointer to the dump content. After the creation of the dump, a MySQL user with the same name as the dump is created. In case of an error, a message is returned.

CreateIbdlUserServlet

The servlet is responsible for the creation process of a new IbdlUser. Therefore, the user name, the real name, the password and the reply of the password are needed. Also, the type of the user should be available in the request. At first, all parameters get checked if they available and not empty. After that, a IbdlUser object gets created and checked if user name is available. If this is true a new user is created and stored.

DeleteIbdlUserServlet

As the CreateIbdlUserServlet, this servlet does the opposite. It is responsible for deleting the user. The request has to contain a user name which should be deleted. If the delete-operation was successful, the response is sent back otherwise an error is returned.

GetUserServlet

This servlet manages the database users. Therefore, a user name and an action are required the action is either the view grants action or the delete-user action. In the servlet both actions are handled differently as shown in listing 3.9

```
if (action.equals("View Grants")) {
    request.setAttribute("success", "User " + selected_user +
        " loaded!");
    request.setAttribute("user", user);
} else if (action.equals("Delete user")) {
    if (dbexe.deleteUser(selected_user)) {
        request.setAttribute("success", "User " +
            selected_user + " deleted!");
    }
}
```

Listing 3.9: handle database user

If view grants is defined as action, than the whole user object is refreshed and stored in the session, if the delete option is selected, then the user is deleted.

LoginServlet

The login into the IBDL admin panel is handled by the LoginServlet. This servlet is the only servlet which returns a response wich is directly shown in the browser. This is pretty helpful because no extra loading page has to be generated while waiting for the login. To do so, the servlet gets the username and the password in the request and tries to log in the user. To log in, the username and password get checked with the help of the IbdlUserManager. If the login process was successful the user object is written to the session as well as authentication is set to true. In the index.jsp file it is checked if authenticated is set to true and if the user object is available. To notify the user an HTML response is generated, which redirects back to index.jsp after 5 seconds as seen in listing 3.10;

```
if (login) {
    ibdlLog.logAction(IbdlLogLevel.INFO, "logged in",request.
        getParameter("username"));
    out.println("Welcome, " + user.getName() + " !</br>");
    out.println("You will be redirected automatically or "+"
        click <a href='index.jsp'>here</a>");
    out.print("<meta http-equiv=\"Refresh\""+"content=\"2;URL=
        index.jsp\">");
    HttpSession session = request.getSession();
    session.setAttribute("user", user);
    session.setAttribute("authenticated", true);
}
```

Listing 3.10: login routine

All other Java servlet pages can check the session if the user is logged in. If a JSP file is accessed without a valid authentication, the user gets redirected to the login page.

LogoutServlet

The LogoutServlet is responsible for logging out a logged in user. This is necessary if a tutor wants to exit the system. To do so, the servlet has to invalidate the actual session by doing the following shown in listing 3.11.

```
HttpSession session = request.getSession();
session.invalidate();
String redirect = response.encodeRedirectURL(request.
    getContextPath() + "/index.jsp");
response.sendRedirect(redirect);
```

Listing 3.11: logout routine

ManageApiKeysServlet

The management of the API keys is very important because without them the web service is not working. The functionality of these keys can be found in 3.4.5. The servlet is responsible for generating and deleting API keys and super user API keys. Therefore, the servlet distinguishes between super user and user API key, and also between creating and deleting. For every case, there is an extra operation. To generate and delete the keys the IbdlAccessManager is called. It is an object which is similar to the IbdlUserManager but is responsible for the API key management. If everything was successful, a message is returned otherwise an error is going to be displayed.

ManageDatabaseServlet

The ManageDatabaseServlet is responsible for managing databases. It allows viewing the tables of the database, as mentioned in listing 3.12. Another possibility is to reinitialize databases. This means it is possible to set a specific database back to the content the dump contains and of course the deletion is also possible. To do this, the request has to contain the name of the database and the action.

```
    ArrayList<Table> tables = null;
    try {
        tables = dbexe.getTables(selected_db);
    } catch (SQLException ex) {
            out.println("<h2>" + ex.getLocalizedMessage() + "
               </h2>");
    }
```

```
for (int i = 0; i < tables.size(); i++) {
    out.println("<h2>" + tables.get(i).getName() + "</h2>"
        );
    out.println(tables.get(i).getHtmlContent());
    out.println("<hr>");
}
```

Listing 3.12: view database

ManageDumpServlet

This servlet manages the dumps. The servlet gives the opportunity to view and delete dumps. Therefore, the request has to contain the name of the dump which should be viewed or deleted. The action has to be available in the request. In the view action, the content of the dump is displayed, in the delete-action, the DbExecutor is called to delete the dump.

ResultServlet

The ResultServlet is responsible for displaying the retrieve the results for a special WBT-username. Therefore, the request has to contain the whole user name and returns an object to the view which contains the whole result. This object is can then be used to create the HTML output. To generate the result object the servlets calls the IbdlLog and saves the result to an object[][]. This is done like shown in listing 3.13.

```
result = ibdlLog.getQueryResult(username);
if (result.length > 0) {
    request.setAttribute("success", "Results␣loaded!");
    request.setAttribute("result", result);
} else {
    request.setAttribute("error", "No␣results␣available!");
}
```

Listing 3.13: retreive result

TestQueryServlet

The TestQueryServlet provides the possibility to execute queries on a database for a special WBT-user. Therefore, the request has to contain the username, the database and the query. All these parameters get checked, if they are not empty and after that the execute() method form the DbExecuter gets executed, with the required parameters. The return of the execute() method is an Object[][] which contains either the returned table cells or the at [0][0] the number of influenced rows by the query, as described in **??**. The execution of the query is done like shown in listing 3.14

```
result = dbexe.executeQuery(username, database, query);
request.setAttribute("success", "Query␣executed!");
request.setAttribute("result", result);
```

Listing 3.14: execute query

The result will be returned an can be retrieved as attribute wit the key "result". The view in test.jsp is responsible for the correct displaying of the result.

UploadDumpServlet

This servlet is responsible for uploading a predefined SQL dump. This dump can be exported from other database systems. But it is important that it is fully compatible to the used database management system. The servlets need to have the dump and the dump name in its request. With all this information the uploadDump() function from the DBexecutor is called. If everything was fine in the upload process, a success message will be returned in the response.

3.4. Web-Service

The web service is the main part of the system. It is responsible for the interaction between the existing WBT-Master system and the internet based database system. There are many ways to connect these two components. The most famous types are REST and SOAP. This system uses SOAP as mentioned in 2.8. The SOAP implementation was done in Netbeans 8.2 which is provided by Oracle, 2017g. Netbeans give the developer the opportunity to test and generate parts of the system out of existing code. The integration of the web service components is supported by Netbeans, so the process of integrating the service is very comfortable.

The web service part is fully encapsulated from the graphical user interface part and implements all functions the GUI supports. Therefore, the web service uses the function provided by the DbExecutor as the servlets in the interface do it. Only with the strict separation of GUI and web service, the simplicity of the service can be done. Also, code segment which should only be available in the GUI or the web service can only be realized with this architecture. To integrate the web service architecture defined in the SOAP specification in W3C, 2000, Java provides the Java API for XML Web-Services. With this implementation, it is possible to provide a SOAP interface in the used application server. To do this in Netbeans, the following steps have to be made.

Add file At first a new web service component have to be added to the existing project. Therefore, it is necessary to add a new file to the project by right-clicking the project node and choose "New" and then "Other...".

Choose web service In the next step the category "Web services" has to be selected. On the right side "Web service" has to be chosen as file type.

Define web service In the next step the web service has to be defined. Therefore, a name and a new package name have to be provided, because all web service stuff is than available in this package After everything is filled the assistant can be finished.

After this procedure, a new package with the defined package name is created. In this package all web service files are available. Also, a new folder Web Services is now disposable in the working tree. The main file is the web service_name.java. It contains the proper web service and the developer is able to create the provided methods here. To setup, the name of the entire web service, under which the client is able to connect to it, the first line of the web service are crucial, because here the main settings for the whole SOAP we service can be made. The settings are annotated at the beginning of the class starting with an @. These annotations are specified by Oracle, 2017f. The most important annotations used in this project are the following.

@WebService(serviceName = "IbdlWS", targetNamespace ="") The annotations marks the service name. It is also possible to define the name and the namespace here, in the project the targetNamespace is definded as ibdl.tugraz.at. But service name is the minimum for every web service which has to be defined.

@HandlerChain(file = "IbdlWS_handler.xml") The handler chain is used to redirect the requests, which come in to a chain element before. This will be necessary for the pre filtering a soap request. This is mentioned in section **??**.

@SOAPBinding(style = SOAPBinding.Style.RPC) The SOAPBinding annotation describes the mapping of the web service to the SOAP protocol. Therefore, the style was changed to RPC, because it does not allow the return of empty responses.

@WebMethod(operationName = "hello") To define a method which is accessible through the web service, this annotation is used. The parameter in this annotation have to be at least the operationName, which describes the name of the method in the WSDL.

@WebParam(name = "name") The last annotation which is important for this project is the WebParam, this is for marking the parameters a function requires. Also, the parameter name have to be specified.

All defined annotations are often used in the web service class. As already mentioned a response can not be empty. JAX-WS implements a way, to return errors to the client. Every WebMethod defined in the web service has the possibility to return a fault. A fault is a kind of exception. To standardize this exception a custom IbdlException is introduced, to handle all possible errors in the same way. This IbdlException works as seen in listing 3.15.

```
@WebFault(name="IbdlException")
public class IbdlException extends Exception{
    private IbdlExceptionBean faultBean;

    public IbdlException(String message,IbdlExceptionBean
        faultInfo){
         super(message);
         faultBean = faultInfo;
    }
    public IbdlException(String message,  IbdlExceptionBean
       faultInfo, Throwable cause) {
        super(message, cause);
        faultBean = faultInfo;
    }

    public IbdlExceptionBean getFaultInfo(){
        return faultBean;
    }
}
```

Listing 3.15: IbdlException

This exception extends the Java standard exception and is marked as WebFault with the annotation @WebFault(name="IbdlException"). Hence, this exception can be used in methods provided by the web service. The exception needs an object called IbdlExceptionBean, which is just a wrapper to a string object because more objects and variables can be added in the bean and used if mandatory.

3.4.1. Java API for XML Web-Services

The whole pre-settings are defined, therefore the web service itself can be explained. The following part describes all available methods in the web service and how they are realised.

createDatabase

The createDatabase method requires an dump name, an username and the super-user-api-key. If the API key is valid and a dump with the specified name is available, then the database gets initialized by the use uf the method initializeDatabase from the DbExecutor. A string with the success message is returned. In case of an error the mentioned IbdlException is sent back as a response. In every possible error case the exception is sent back, to provide the client with the information what was wrong with the execution. The return string of the function is only returned in success state because only then, the function executed successfully.

deleteDatabase

The method requires a database name and the super-user-api key. All parameters get checked and in case of an error, an IbdlException is thrown. If all parameters are okay, the deleteDatabase() form DbExecutor is called. If the process was successful, the string ″success″ is returned to the client in the response.

deleteDatabaseUser

The provided method deleteDatabaseUser is responsible for deleting a database user. To do so, a username and a super-user-api-key is required. After the key was checked, the username is also audited. If all checks passed, the method calls the deleteUser() from DbExecutor. If the process was effective, ″success″ is returned in the response to the client. If something is wrong, an IbdlException is thrown immediately and the execution gets aborted.

deleteDump

The deleteDump method is used for deleting a dump in the internet based database laboratory. The method needs the name of the dump and the super-user-api-key. The parameters get checked and if all parameters are legal, the method deleteDump form the DbExecutor object is called. If the operation was executed successful, the String ″success is returned to the calling client. Otherwise, an IbdlException is returned.

executeQuery

This method is important and is one of the most used methods in the entire web service. Every execution of a query by a WBT-Master user is done via this query. Therefore, an username, database name, query and user-api-key are required. After the API key is checked, the system executes the query on the given database. If the username or the database not exist, a IbdlException is returned. If that is not the case, an Object[][] is returned, where the first[] are the columns and the second[] are the rows. The method is shown in listing 3.16.

```
try {
  if (IbdlAccessManager.checkApiKey(user_api_key)) {
    Object[][] result = dbexe.executeQuery(username, database,
        QueryPrefilter.filter(query));
    if (result.length == 1 && result[0].length == 1) {
      ibdlLog.logQueryResult(database + "_" + username, query,
          "" + result[0][0], false);
    } else {
      ibdlLog.logQueryResult(database + "_" + username, query,
          ibdlLog.ArrayToHTMLTable(result), false);
    }
    return result;
  } else {
    throw new IbdlException("Wrong␣user␣api␣key!", new
        IbdlExceptionBean());
  }
} catch (SQLException ex) {
  throw new IbdlException("SQL-Error:␣" + ex.
      getLocalizedMessage(), new IbdlExceptionBean());
}
```

Listing 3.16: execute query

The usage of the Java class Object, is pretty useful here, because soap can auto-detect the type of the content in the object and mark the type in the XML response as shown in listing 3.17.

```
<item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">id_b</item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">name</item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">strasse</item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">plz</item>
</item>
<item>
    <item xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">1</item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">Intel</item>
    <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">Intelstrasse</item>
    <item xsi:type="xs:int" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
        /2001/XMLSchema-instance">9112</item>
</item>
```

Listing 3.17: response

In listing 3.17 it can be seen, that every returned value gets an xsi:type which defines the type of the returned value where xsi is the used namespace definition. With this description, it is possible to convert the data back into a datatype the client can work with.

executeQuerySandboxMode

To execute queries in sandbox mode, which is explained in 3.6.2, a separated method is provided to avoid misunderstanding. This method needs the same parameters as the normal executeQuery function, namely username, database name, query and user-api-key. At first, the API key is checked, and after that, the query gets executed with the executeQuerySandbox() method from the DbExecutor. For the result, an Object[][] is generated and if everything goes fine, the 2-dimensional array gets returned to the client. As already mentioned, the JAX-WS libraries specify the objects in the soap response, so the client can retrieve the information which data type the result has.

getResult

The getResult method available trough the web service gives the client the possibility to retrieve the execution results for a special user. The method needs the super-user-api-key and the name of the WBT-Master user, for whom the result should be retrieved and the database name. After the API key is checked the result is loaded, with the help of the getQueryResult method from the IbdlLogger which is responsible for the logging stuff. This is done as shown in listing 3.18.

```
if (IbdlAccessManager.checkSuperApiKey(super_user_api_key)) {
  return ibdlLog.getQueryResult(db_name+"_"+username);
} else {
  ibdlLog.logAction(IbdlLogLevel.ERROR, "Invalid API key! ("+
      super_user_api_key + ")", "web service");
  throw new IbdlException("Invalid API key!", new
      IbdlExceptionBean());
}
```

Listing 3.18: get execution results

The returned result is a two-dimensional object, which contains all entries for the username and the name of the database provided in the request. The information about id, database, query, result, state and timestamp are available in the XML response.

hello

This method is just a "HelloWorld" method for the web service. It is possible to test the basic functionality and check if the web service is up and running. It is also possible to provide some text in the request to get back a "Hello" concatenated with the provided text.

listDatabaseUsers

The listDatabaseUsers method is responsible for displaying all available database users stored in the ibdl.db_users table. The request has to contain the super-user-api-key. If the key is okay, the users are gathered via the getUsers method of DbExecutor. To generate the output, the names are stored in the list which is returned like described in listing 3.19.

```
users = dbexe.getUsers();
String[] user_names = new String[users.size()];
for (int i = 0; i < users.size(); i++) {
  user_names[i] = (users.get(i).getName());
}
ibdlLog.logAction(IbdlLogLevel.INFO, "listUsers successfull",
    "web service");
return user_names;
```

Listing 3.19: get database users

In case of an error the IbdlException is sent back to the client, otherwise the list is sent back.

listDatabases

This method can be used to call a list of all available databases. Therefore, it is necessary that the request contains the super-user-api-key. If the API key is valid, the list of databases is generated by the use of the method getDatabases() from the DbExecutor object. The list is sent back to the client in case of success, otherwise, an error is sent via the IbdlException.

listDumps

To get all allocatable dumps stored in the internet database laboratory, this method has to be used. The super-user-api-key has to be provided by the client requesting the information. To build up the output, the method getDumps from DbExecuter is used. Because only the names are required in the output, the list has to be converted by abstracting the name out of the dump list.

listPermissions

To get the actual MySQL permissions for a MySQL user this method has to be used. Thus, the request has to contain the super-user-api-key and the name of the MySQL user which is the same as the dump name. If all these parameters are good, a DbUser object is retrieved. After that, the list of permissions is excluded and send back to the client.

reinitDatabase

In case something is wrong with the database for a special user, this method can help by resetting the database back to it initial state. The method needs to have the super-user-api-key and the database name in the request. First of all the API key is checked, if it is valid, the database gets deleted After that a new database form the original dump is generated. This means a completely new database is initialized in this step and the old one gets removed.

setPermissions

The setPermissions method is available for setting the permission for a user. To specify the permissions, an username, the super-user-api-key and of course the permission have to be provided in the request. This is done by a list, where every ¡item¿ represents a new permission. The permissions available are the one defined by MySQL earlier explained in this document. All permissions provided in the list are then set to GRANT, and permissions which are not in the list are REVOKED. This is done by calling the DbExecutor and running the setPermissions method.

uploadDump

The possibility to upload dumps is also available trough the web service. The request has to contain the content of the dump, the name of the dump and the super-user-admin-key. If all tree parameters are available and valid, the dump is uploaded. Therefore, the dump has to be available in a Base64 encoded string, because otherwise there are troubles with special things in the entire SQL statement. The dump gets then encoded in the upload process like mentioned in listing 3.20

```
byte[] content = Base64.decodeBase64(dump);
InputStream stream = new ByteArrayInputStream(content);
try {
  dbexe.uploadDump(stream, dumpname);
  return "success";
} catch (SQLException ex) {
  throw new IbdlException("SQL-Error:␣" + ex.
      getLocalizedMessage(), new IbdlExceptionBean());
}
```

Listing 3.20: upload dump

After the content is decoded, the uploadDump method from the DbExecutor is called. If everything works like expected, the string "success" is returned other wards an IbdlException is returned.

createEmptyDump

This method creates a new empty dump and stores this information in the ibdl.dumps table. To do so the method requires the super-user-api-key and a name. This name is used as dumpname, as well ass the name for the new MySQL user created for this dump. If all parameters are valid, the uploadDump and the createDbUser of the DbExecutor is called. If both operations were successful, the string "success" is returned to the client. If there is an error, an IbdlException is sent back.

3.4.2. SOAPhandler

To access SOAP messages before the actual processing of the requests, the JAX-WS standard provides the possibility to use handlers. The implementation of such handlers is mentioned by Oracle, 2017a. This SOAPhandler is able to access the message before the execution of the requested web service method and can either be implemented on the server-side as well as on the client-side. It is possible to have multiple handlers attached before and after the SOAP processing web service as shown in figure 3.8 provided by Oracle, 2017a.

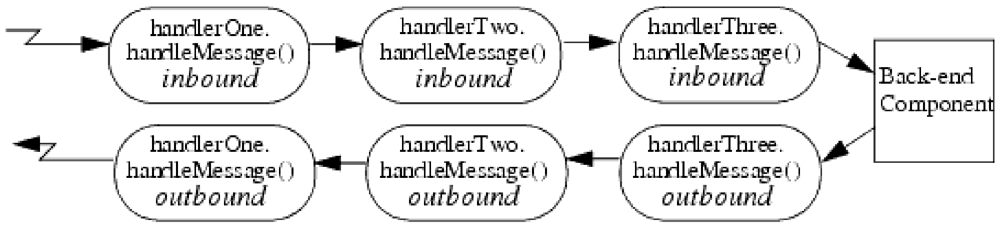


Figure 3.8.: SOAP handler chain

This is very useful to process SOAP requests before they are actually executed in the web service. There are two types of handlers which can be used for the implementation. The SOAPHandlers allow to access the full SOAP message whereas the LogicalHandlers allow the access of the payload of the message, so they will not allow changing any protocol specific information. For this web service, a SOAPHandler is used to extract the clients host address out of the system. To implement such a SOAPHandler it is necessary to create a new handler, in this case the IbdlWShandler which implements the provided interface SOAPHandler.

```
public class IbdlWSHandler implements  SOAPHandler {

  public boolean handleMessage(SOAPMessageContext msgCtxt) {
    SOAPMessage msg = msgCtxt.getMessage();
    HttpServletRequest request = (HttpServletRequest)
    msgCtxt.get(MessageContext.SERVLET_REQUEST);
    IbdlAccessManager.setHost(request.getRemoteHost());
    return true;
  }

  public Set<QName> getHeaders() {
    return Collections.EMPTY_SET;
  }

  public boolean handleFault(SOAPMessageContext msgContext) {
    return true;
  }

  public void close(MessageContext context) {
  }

}
```

Listing 3.21: SOAPHandler

In listing 3.21 can be seen that the method handleMessage is very important. This method got the message context as a parameter. With this parameter, it is possible to retrieve every available method out of the message. It is feasible to get the header, the body and all attached metadata out of a SOAP request. In the shown handler, the remote host is read out of the message and stored in the IbdlAccessManager.

To link the SoapHandler to the web service which should use this handler, it is essential to generate an XML file where the handler chain is specified. This file should look like explained in listing 3.22.

```
<?xml version="1.0" encoding="UTF-8"?>
<handler-chains xmlns="http://java.sun.com/xml/ns/javaee">
  <handler-chain>
    <handler>
      <handler-name>web service.IbdlWSHandler</handler-name>
      <handler-class>web service.IbdlWSHandler</handler-class>
    </handler>
  </handler-chain>
</handler-chains>
```

Listing 3.22: handler chain

This handler chain contains all classes which should be used in the chain. To concatenate the handler chain to the entire web service the annotation @HandlerChain(file = "IbdlWS_handler.xml") is used. After this steps were done, every incoming request is routed through the handler chain before or after reaching the entire web service definition.

3.4.3. SOAP specification

The specification of the web service is one of the most import things in this project. In SOAP the description of such web services is done via the web service description language (WSDL). The WSDL is already explained in section 2.8. To generate such file, Netbeans and several other IDE's are able to auto create the WSDL file by using the annotation in the web service. When running this project on the localhost, the WSDL file is available at `http://localhost:8084/InternetBasedDatabaseLaboratory/IbdlWS?wsdl`. By opening the provided URL, the description of the web service and its provided methods can be found.

The WSDL file of this project can also be found in the appendix A.1. The description is XML encoded because it is mainly for machine code to auto-create the requests out of it. The file shows every single method, the required parameters and the return-types. To integrate a web service, a developer can parse the information available in the WSDL file to generate a request which is then send to the web service. To do so, an example with the getResult method provided by this service is demonstrated in the following description.

In the first step, the WSDL file defines the message types and the message response types. For the method getResult this looks like shown in listing 3.23.

```
<message name="getResult">
  <part name="superuser-api-key" type="xsd:string"/>
  <part name="wbt-username" type="xsd:string"/>
  <part name="database-name" type="xsd:string"/>
</message>
<message name="getResultResponse">
  <part name="return" type="ns1:stringArrayArray"/>
</message>
<message name="IbdlException">
  <part name="fault" element="tns:IbdlException"/>
</message>
```

Listing 3.23: message description

In the listing above, it can be seen, that every parameter is described by its name as well as by its type. Also, the possible IbdlException is professed. In case of the return value, the type is defined as "ns1:stringArrayArray" because this is a two-dimensional array from the namespace class ns1 which is defined on top of the WSDL file. By now the types of all parameters and return value is defined. In the next section, the method itself have to be defined.

```
<operation name="getResult" parameterOrder="superuser-api-key␣
    wbt-username␣database-name">
  <input wsam:Action="http://web␣service/IbdlWS/
      getResultRequest" message="tns:getResult"/>
  <output wsam:Action="http://web␣service/IbdlWS/
      getResultResponse" message="tns:getResultResponse"/>
  <fault message="tns:IbdlException" name="IbdlException"
      wsam:Action="http://web␣service/IbdlWS/getResult/Fault/
      IbdlException"/>
</operation>
```

Listing 3.24: operation description

In listing 3.24 the description of the operation "getResult" is shown. The order of the parameters is given, as well as the defined messages explained before. It can also be seen, that the fault message is thrown if something is wrong. In the WSDL document, also the SOAP binding is defined, but this only shows the binding of SOAP action to the actual method, which is equal for all provided operations. Thus, the SOAP client can now build the request for calling an operation on the web service. For the explained operation, the minimal request is shown in listing 3.25.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
    soap/envelope/" xmlns:ibdl="http://ibdl.tugraz.at">
  <soapenv:Header/>
  <soapenv:Body>
    <ibdl:getResult>
      <superuser-api-key>Iai4doeYVIfj6kt7JbV1SNHaz</superuser-
          api-key>
      <wbt-username>werner</wbt-username>
      <database-name>db1</database-name>
    </ibdl:getResult>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 3.25: getResult request

In the listing above it can be seen that the client creates a new SOAP envelope, with an empty header, because no header information is needed. In the body of the envelope, the two parameters superuser-api-key and the wbt-username have to be specified, as well as the name of the requested web service method. If this valid request is sent to the web service, a valid response is returned as shown in listing 3.26.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/
    ">
  <S:Body>
   <ns2:getResultResponse xmlns:ns2="http://ibdl.tugraz.at">
     <return>
       <item>
         <item>49</item>
         <item>db1_werner</item>
         <item>SELECT * from 'BRANDS'</item>
         <item>...</item>
         <item>success</item>
         <item>2017-02-23 12:06:07.0</item>
       </item>
           .
           .
```

```
            .
      </return>
    </ns2:getResultResponse>
  </S:Body>
</S:Envelope>
```

Listing 3.26: getResult response

The response in the listing above shows that the returned result contains the needed information, encoded as XML in a format, that displays the mentioned two-dimensional array. On the client this response can be transferred back into the array. As already noticed, every web service call need to have an API key, which is described in the next section. To form a valid request to the web interface the XML schema as described in the next section can be used. If this schema is used, the request is valid and a valid response is returned.

3.4.4. Interface as XML Schema

XML schema is a standardized way to define the structure of a XML document. The XML Schema Definition (XSD) will be specified to guarantee that an XML file is valid and can be processed by the machine. The schema defines all possible nodes and is able to cover the data type of every element. The XSD file is different for every single request, because the number of parameters can differ. It is possible to use a schema which does not specify the parameters, but this will affect the granularity of the information to validate the XML request. Therefore, a sample XSD file is shown below, which demonstrates the schema for the executeQuery operation.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      attributeFormDefault="unqualified">
  <xs:element name="soapenv:Envelope">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="soapenv:Header"></xs:element>
        <xs:element name="soapenv:Body">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ibdl:executeQuery">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="webservice-username" type="xs:string"></xs:element>
                    <xs:element name="database" type="xs:string"></xs:element>
                    <xs:element name="query" type="xs:string"></xs:element>
                    <xs:element name="user-api-key" type="xs:string"></xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="_xmlns:soapenv" type="xs:string"></xs:attribute>
      <xs:attribute name="_xmlns:ibdl" type="xs:string"></xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 3.27: executeQuery request XSD

This schema demonstrates an exact specification for request of the executeQuery method. Every single request has in principle the same schema. Only the deepest nodes change because of the different parameters. In the current example, there are the four parameters web service-username, database, query, user-api-key. In other requests, there are more or fewer parameters. Therefore, the schema has to be adapted to provide a strict valid schema.

Not only the request is processed in the XML format, but also the response is an XML message which contains the data. This response contains pretty different result types. The are strings, arrays and objects which are returned by the web service. The different data-types of the return values are specified in the attributes of each node. An example of the response of the executeQuery method is shown in listing 3.28.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:executeQueryResponse xmlns:ns2="http://ibdl.tugraz.at">
      <return>
        <item>
          <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
                w3.org/2001/XMLSchema-instance">id_b</item>
          <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
                w3.org/2001/XMLSchema-instance">name</item>
          <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
                w3.org/2001/XMLSchema-instance">strasse</item>
          <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
                w3.org/2001/XMLSchema-instance">plz</item>
          <item xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.
                w3.org/2001/XMLSchema-instance">land</item>
        </item>
      </return>
    </ns2:executeQueryResponse>
  </S:Body>
</S:Envelope>
```

Listing 3.28: executeQuery response XSD

It can be seen, that all types are specified inline and no extra XML schema exists. It is possible to create them and use it for validation. The last part which can be defined via XML schema is the returned fault. This fault is returned if something failed. Because the fault is especially implemented for this project, no pre-defined standard fault can be used. The IbdlException fault, which is explained in listing 3.15 is validated by the XSD file shown in listing 3.29.

```
<xs:schema version="1.0" targetNamespace="http://ibdl.tugraz.at" xmlns:tns="http://ibdl.tugraz.at"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="IbdlException" nillable="true" type="tns:ibdlExceptionBean"/>
    <xs:complexType name="ibdlExceptionBean">
      <xs:sequence/>
    </xs:complexType>
  </xs:schema>
```

Listing 3.29: ibdlExceptionBeany XSD

It can be seen that the schema for this exception is weak because only the container of the exception is specified and the main content can be specified. This is useful for the exception because different errors are returned, and therefore this specification can cover all of them.

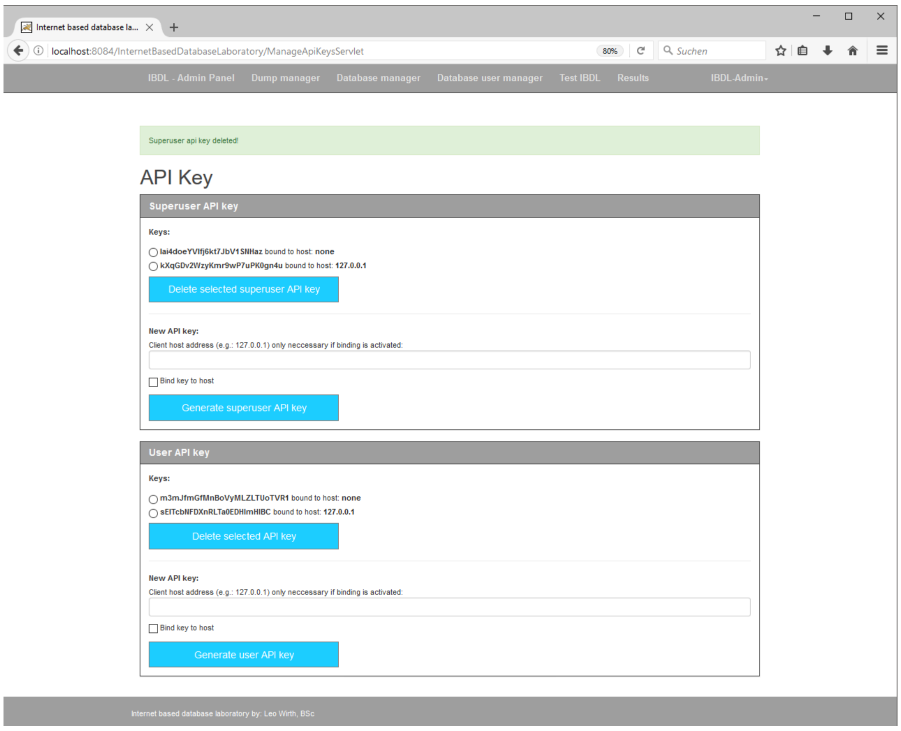
3.4.5. API Keys

To provide a high level of security, and to prevent unauthorized usage of the web service, the system has an API access manager implemented. This access manager is responsible for handling the API keys. There are two different types of keys for accessing the web service. These keys are called "user API key" and "superuser API key". The difference between the two types can be described as follows.

superuser API key This type of API key is responsible for accessing methods provided by the system which is only allowed to access by the teaching team of the course. Such methods are for instance the creation of new dumps, the creation of new databases, etc. In principle, all methods except the executeQuery() and the executeQuerySandboxMode() method, need the superuser API key.

user API key The user API key is needed for normal operations, which are executed by the participants of the course. In this actual implementation, this would be the case for the executeQuery() and the executeQuerySandboxMode() method, which is usually run in the user context of WBT-master.

The usage of the keys is explained, but the creation of the keys is not mentioned yet. To create an API key, the admin has to log into the graphical user interface. On the right, in the user menu the section "API key" can be found. The interface looks like figure 3.9. In the interface, two keys can be


Internet based database la...
localhost:8084/InternetBasedDatabaseLaboratory/ManageApiKeysServlet
80%
Suchen
IBDL - Admin Panel
Dump manager
Database manager
Database user manager
Test IBDL
Results
IBDL-Admin
Delete selected superuser API key
Generate superuser API key
Delete selected API key
Generate user API key

Superuser api key deleted!

API Key

Superuser API key

Keys:

○ **lai4doeYVlfj6kt7JbV1SNHaz** bound to host: **none**
○ **kXqGDv2WzyKmr9wP7uPK0gn4u** bound to host: **127.0.0.1**

New API key:
Client host address (e.g.: 127.0.0.1) only neccessary if binding is activated:

☐ Bind key to host

User API key

Keys:

○ **m3mJfmGfMnBoVyMLZLTUoTVR1** bound to host: **none**
○ **sEITcbNFDXnRLTa0EDHImHIBC** bound to host: **127.0.0.1**

New API key:
Client host address (e.g.: 127.0.0.1) only neccessary if binding is activated:

☐ Bind key to host

Internet based database laboratory by: Leo Wirth, BSc

Figure 3.9.: API key managment interface

generated. For a higher level of security, the keys can be bound to a given host. This means only a special client host address, e.g. localhost, can access the methods provided by the web service with the provided API key. If the binding option is not selected, the key is able to run on every client. The check of the clients host address and the api key are made in the IbdlAccessManager. This class can be called as a static object and is responsible for all API key management. To create new API keys, two methods are available. The generateSuperUserKey() and the generateUserKey() method generate an API key, the length of the key is defined by the parameter KEY_SIZE which is 25 in the actual implementation, and store the generated key in the database table ibdl.api_keys. These methods are also overloaded for the use with a provided client host address. The mentioned table ibdl.api_keys contains all the information about the available API keys and their type and preferences. To check if a key is valid for the actual request, the method checkSuperApiKey() and checkApiKey() are callable. Both methods work in the same way, as described in listing 3.30 by the use of the checkApiKey method.

```
static boolean checkApiKey(String user_api_key)
                                            throws SQLException {
  DatabaseConnector dc =
          new DatabaseConnector(new DbRootCredentials());
  dc.ConnectToDB();
  ResultSet rs;
  rs = dc.executeQuery(
        "SELECT * FROM ibdl.api_keys where is_super='0'"+
                "and 'key' = '" + user_api_key + "';");

  if (rs.next()) {
    if (rs.getInt("check_host") == 1) {
      if (rs.getString("host").equals(getHost())) {
        dc.closeConnection();
        return true;
      }else{
        dc.closeConnection();
        return false;
      }
    } else {
      dc.closeConnection();
      return true;
    }
  }
  dc.closeConnection();
  return false;
}
```

Listing 3.30: check API key

As mentioned in the listing, the method returns true if the API key is valid, this is important in the web service where these methods are used to check whether a key is valid or not. Only if a valid key is served the operation can be executed, otherwise an IbdlException should be thrown to inform the client about the wrong key. But also the developer has to take care, which API key should be used, either the user API key or the superuser one. With all the mentioned methods the IbdlAccessManager provides a lot of measurements to protect the web service from unauthorized usage. To do so, also the methods described in section 3.4.6 are responsible for protecting unauthorized access.

3.4.6. Verification

To verify the incoming request, it is essential to provide some features to get integrity, confidentiality and authenticity. These three aspects are important to support, otherwise, there can be troubles with the system, if someone tries to attack it. This is very unlikely, but it can happen. Therefore, the possibility have to be supported to gain all three points.

Authenticity

The web service uses application level authentication. This means that the web service is responsible for determining if a request by the client can be executed because the authentication data provided is valid. As already mentioned, the system implements an API manager mentioned in section 3.4.5, which handles both types of user levels. The configuration of the authentication data is all done by the application itself. Therefore, it is regardless of which application server is used. Another possibility to gain authenticity is the use of container authentication. For this type, the authentication process is fully managed by the application server. Therefore, it is necessary to adapt the service every time the type of the server changes. Also, the server has to be configured, to work with the web service, because the authentication data is checked on it.

Confidentiality

To offer confidentiality, the entire system and the Tomcat server are able to use SSL. The graphical user interface and the SOAP web service can use a Java keystore to generate confidentiality. To do so, the following steps have to be ensured. At first, the keystore have to be generated with the keytool provided by the Java Development Kit and located in the /bin folder. The tool have to be called with the requested parameters. For this environment, the parameters are shown in listing 3.31 were used.

```
keytool -genkey -alias tomcat -keyalg RSA -keysize 2048
```

Listing 3.31: Java keytool

The provided command creates a key with the alias tomcat, the algorithm RSA, and a keysize of 2048 bytes. These parameters can be modified to support the requested needs. The next step is to modify the server with the that it is able to provide HTTPS by modified the server.xml file of the tomcat instance like shown in listing 3.33.

```
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
disableUploadTimeout="true" enableLookups="false" maxThreads="25"
port="8443" keystoreFile="PATH_TO_KEYSTORE/.keystore" keystorePass="PASSWORD"
protocol="org.apache.coyote.http11.Http11NioProtocol" scheme="https"
secure="true" sslProtocol="TLS" />
```

Listing 3.32: server.xml

The attached configuration includes the keystore file with the keystore password and sets the protocol. After a reboot of the server HTTPS is available. To run the internet based database application only in HTTPS mode, the web.xml file has to be changed like shown in listing 3.33.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>ibdlmanager</web-resource-name>
    <URL-pattern>/*</URL-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Listing 3.33: server.xml

The important keyword in the added constraint is "CONFIDENTIAL", which forces the server to deploy this application only with HTTPS. Also, the web service is now sent over HTTPS, which raises the security level for the entire system. But there is the third point, integrity, which have to be provided to offer a safe system.

Integrity

Integrity should also be supported by the web service, to do so, JAX-WS and Metro support a lot of mechanisms to do so. Some of them, also provide authenticity and confidentiality. In the actual implementation, the integrity feature is disabled by default, but this can be changed to the needs of the client by setting another mechanism. The project supports the most important mechanisms described by Oracle, 2017j.

Username Authentication with Symmetric Keys This security setting gains integrity and confidentiality by symmetric keys. The key is used to sign and encrypt the messages which are sent between the client and the server. Both share the same key but the client has to send the username and the password to the server, and must identify the alias.

Mutual Certificates Security The mutual certificates security provides authenticity, integrity and confidentiality. At both actors, the client and the server, the keystore and the truststore have to be specified.

Transport Security (SSL) This mechanism is able to serve authenticity, integrity and confidentiality. To do so, the transport security provides a secure channel between the client and the sender, but it is not protected at the destinations. But this problem is not very important because both destinations in the usage of this project can be trusted.

Endorsing Certificate This setting is able to gain integrity and confidentiality. Therefore the mechanism uses a symmetric key and the client knows the servers certificate. The client requests the authorization by a special identity and after that, the client is able to communicate with the server and provide the mentioned security features.

With all three security features, the system provides all possible mechanism of protections. Because the system is mostly run in the same trusted network, it is possible to activate and deactivate the mechanism by the needs of the runtime environment.

3.4.7. IBDL database schema

Because of the complexity of the system, the internet based database laboratory needs to have a database schema, for supporting all required information. The usage of the different tables is already mentioned in the different sections of chapter 3. In the development process of the system, one goal was to minimize the number of tables required for the entire system. But all in all the system uses six tables which are essential for running it. The format of the table, extracted with the MySQL command DESCRIBE can be found in the tables 3.1 to 3.6.

Field	Type	Null	Key	Default	Extra
id_a	int(11)	NO	PRI	NULL	auto_increment
loglevel	int(11)	NO		NULL	
message	text	NO		NULL	
user	text	NO		NULL	
caller	text	YES		NULL	
timestamp	timestamp	NO	PRI	CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

Table 3.1.: action_log table

Field	Type	Null	Key	Default	Extra
id_a	int(11)	NO	PRI	NULL	auto_increment
key	text	YES		NULL	
is_super	tinyint(4)	YES		NULL	
check_host	tinyint(4)	YES		NULL	
host	text	YES		NULL	

Table 3.2.: api_keys table

Field	Type	Null	Key	Default	Extra
id_u	int(11)	NO	PRI	NULL	auto_increment
name	text	YES		NULL	
username	text	YES		NULL	
password	text	YES		NULL	

Table 3.3.: db_user table

Field	Type	Null	Key	Default	Extra
id_d	int(11)	NO	PRI	NULL	auto_increment
name	text	YES		NULL	
file	blob	YES		NULL	

Table 3.4.: dump table

Field	Type	Null	Key	Default	Extra
id_u	int(11)	NO	PRI	NULL	auto_increment
name	text	YES		NULL	
username	text	YES		NULL	
password	text	YES		NULL	
admin	tinyint(4)	YES		0	

Table 3.5.: user table

Field	Type	Null	Key	Default	Extra
id_l	int(11)	NO	PRI	NULL	auto_increment
user	text	YES		NULL	
query	text	YES		NULL	
result	text	YES		NULL	
state	varchar(45)	YES		NULL	
timestamp	timestamp	YES		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

Table 3.6.: user_log table

3.5. Permission handling

The handling of permissions is split into two parts. The first part is the execution of the provided methods in the web service. As already mentioned in section 3.4.5, there is a user level and a superuser level for executing a requested method. For all administration stuff, the superuser key is required. For the user stuff, like executing the method executeQuery the user API key is used. But there have to be an action to manage the database permissions.

The second part of the permission handling. To handle the permissions for a database, a MySQL user is created whenever a new dump (eg.: db1) is created. This user (db1) is now responsible for all permissions for every database retrieved from this dump db1. This means every database which starts with db1_* is now managed by the MySQL user db1. This is pretty useful, because every student in a database course should get the same permissions to work on a system, and therefore the supervisor only need to set the permissions once. Another advantage is that the server has a much lower numbers of users to manage and the administration is not as hard as if every generated database has its own user in the database management system. The permissions can either be set via the provided setPermission() method over SOAP, or in the graphical user interface. For sure the permissions can also be set in the database management system administration tool. The following permissions, which where mentioned by Oracle, 2017i are able to manage the actual implementation of the system.

SELECT Enables the user to retrieve data from a selected table.
INSERT Allows the insertion of data into a table of the database.
UPDATE Gives the possibility to update a dataset with the provided value.
DELETE Enables the deletion of a dataset in the database table.
EXECUTE Offers the possibility to execute stored procedures.

SHOW VIEW Enables to use the SHOW CREATE VIEW.
CREATE Gives the possibility to create new databases and tables.
ALTER Allows to change the structure of a table.
REFERENCES To create the reference for the parent table.
INDEX Allows to create or drop indexes.
CREATE VIEW Possibility to create views.
CREATE ROUTINE Possibility to create routines.
ALTER ROUTINE Allows to drop or alter a stored routine.
EVENT Enables the privilege to create, view, drop events.
DROP Allows to delete databases, tables and views.
TRIGGER Enables trigger operations.
GRANT OPTION Allows to change permissions for a user.
CREATE TEMPORARY TABLES Allows to create temporary tables.
LOCK TABLES Gives the opportunity to lock tables.

It is essential that the supervisor only allows the necessarily needed permissions, otherwise, there is the possibility that the students, which are not familiar with MySQL and database can cause damage on their databases, because they copy some queries from their preferred internet portal and execute them, without knowing what is happening. In the beginning, all mentioned privileges are enabled for a user created when a dump is uploaded, this means that the privileges have to be checked before the specific table is used by the course participants, otherwise there will be troubles. It would be possible to disallow all privileges on the creation of a user, but in the actual implementation, all privileges are granted.

3.6. Protection mechanisms

The system provides some features to protect it from damage. To do so, two measurements are implemented which are able to protect against the main causes which can provoke harm. There is a pre-filtering option available as described in section 3.6.1 as well as and sandbox mode, described in section 3.6.2 to limit the possibility to destroy something. Both methods can be used optionally, but they are a good opportunity to protect against human errors caused by the participants. There are other measurements mentioned in the verification section in 3.4.6. The alluded mechanisms are described in the following section.

3.6.1. Pre-filtering

The pre-filtering of queries is an aspect of the project which should be available This filtering can either be done, to filter possible broken SQL queries where the syntax is obviously wrong because some essential parts of the query are missing and therefore the query should not be executed to reduce the load on the MySQL server, or to detect harmful queries which should not be executed. The actual project only provides an interface where the actual filtering can be done. This is because only the possibility to filter queries should be allocatable. To filter the queries, an adequate SQL parser would be necessary like the JSqlParser, 2017 used for filtering queries in the sandbox mode. These parsers provide the chance to parse the provided query in the filter to the requested needs.

The filter has to be built in the class QueryFilter where a static method filter(String query) is available The method takes the unfiltered query and returns the filtered query in the return value of the method. The method already provides the chance to invoke the mentioned parser wit the statement object as shown in listing 3.34.

```
public static String filter(String query) {
  String filtered_query = query;
  IbdlLogger ibdlLog = IbdlLogger.getInstance();
  Statements stmt = CCJSqlParserUtil.parseStatements(query);

  //---------------------------------
  // Queries can be filtered here!
  //---------------------------------

  ibdlLog.logAction(IbdlLogLevel.INFO, "Query filtered!","
      prefilter" );

  return filtered_query;
}
```

Listing 3.34: filterQuery

By now the method returns the ingoing unfiltered query. The method is called, whenever a query is going to be executed either in the graphical user interface or via the web service.

3.6.2. Sandbox

Sandboxing is another feature the system provides. This peculiarity gives the user the possibility to execute queries and nothing should change. It is like an isolated playground, where no havoc can be wreaked. This is especially expedient if a user does not know what he is doing in a query, he can run it via the sandbox feature, and his database schema is not influenced. Therefore, it is necessary to determine which queries are able to modify the content of a database. In database courses, the MySQL standard CRUD operations are used most, this means that only UPDATE, INSERT, DELETE and SELECT queries are in the pool of possible sandbox operations. Because a SELECT query does not influence or modify a table in the database schema, the sandbox mode is able to handle UPDATE, INSERT and DELETE queries. To determine the type of given query, a parser is needful. The parser used for this operation is the JSQLParser by JSqlParser, 2017, which is able to determine the influenced tables, as shown in listing 3.35.

```
Statement statement = CCJSqlParserUtil.parse(query);
Insert insertStatement = (Insert) statement;
TablesNamesFinder tablesNamesFinder = new TablesNamesFinder();
tableList = tablesNamesFinder.getTableList(insertStatement);
```

Listing 3.35: query detection

To get the tables out of the query, it is necessary to know the type of the query, this can be determined with a split operation. After this information is gained, the type of the statement can be used to parse the query. After the information about all tables is on hand, every table is copied via a SELECT * query into a two-dimensional object.

After that step is done, the provided query is going to be executed in the system. This query modifies the table and returns data. After the execution is finished, the tables have to be restored with the information stored in the list tableContent. To restore a table it is necessary to reset the table so that all auto-increments are set back, and the table is empty. This is realized via the MySQL TRUNCATE TABLE operation as mentioned by Oracle, 2017k. After this passage is executed, the table is empty and can now be refilled with the data stored in the backup process. Therefore, an INSERT query is build up from every single row in the backup array and inserted in the table as it can be seen in listing 3.36.

```
for (int i = 0; i < tableList.size(); i++) {
  if (this.truncateTable(database + "_" + username + "." +
      tableList.get(i)) == false) {
    throw new IbdlException("Failed␣to␣truncate␣table", new
        IbdlExceptionBean());
  }
  for (int k = 1; k < tableContent.get(i).length; k++) {
    String restore = "INSERT␣INTO␣" + database + "_" +
        username + "." + tableList.get(i) + "(";
    for (int j = 0; j < tableContent.get(i)[0].length - 1; j
        ++) {
      restore += tableContent.get(i)[0][j] + ",␣";
    }
    restore += tableContent.get(i)[0][tableContent.get(i)[0].
        length - 1] + ")␣VALUES␣(";
    for (int j = 0; j < tableContent.get(i)[k].length - 1; j
        ++) {
      restore += "'" + tableContent.get(i)[k][j] + "',␣";
    }
    restore += "'" + tableContent.get(i)[0][tableContent.get(i
        )[k].length - 1] + "');";
    this.executeQuery(username, database, restore);
  }
}
```

Listing 3.36: restore table

To be on the safe side, the tables can be locked for write operations before the backup process and unlocked after the restoring procedure, to prevent from unexpected access. A call of the MySQL provided backup function would be more elegant but is not available via the JDBC driver. One possibility would be, that the MySQL command line interface (CLI) is called via a process in Java, but there is the big disadvantage, that this would by very system specific and has to be adjusted for the used operation system.

3.7. Logging

To log all events, a special IbdlLogger was introduced which can be called directly in every class and is responsible for logging events as well as logging results from executed queries by the users. It is the core element for handling all logging stuff for superusers and users. To do so, the IbdlLogger object provides several methods and is implemented as a singleton. To get the instance of this object, it is necessary to call IbdlLogger.getInstance(). The log messages which are generated if an action is called in the entire internet based database laboratory are stored in the table ibdl.action_log. The results of the execution, generated if a query is executed, are stored in the table ibdl.user_log.

The logging of actions is the first part which has to be noticed. The IbdlLogger provides the method logAction(int level, String message, String username), which should be called if an event has to be logged. The method needs an level, which is the log level and can either be IbdlLogLevel.INFO, IbdlLogLevel.WARNING, IbdlLogLevel.ERROR. If required, new log levels can be created in the file IbdlLogLevel in the same way the current levels are created. This is for filtering the type of log message, to get a better overview of all logged messages. The next parameter is the message itself, and the third parameter is the username who triggered the action. The logging of all action is pretty useful in this project because it is a completely new developed software, where errors can occur under special circumstances. The logAction method also extracts the caller of the method out of the stack-trace, to find out which class called the logAction method. To view the logged messages, the graphical user interface provides the action "View log" in the user menu. In this view it is possible to view all generated log messages, sorted by the time stamp To extract this information out of the database table the IbdlLogger provides the method getActionLog() which needs the actual page number as a parameter, because the view supports paging with 20 entries/page. This means that page one contains the entries 1-20, page two entries 21-40 and so on. With all the mentioned methods, it is possible to provide a high level of logging, with a little afford and a high degree of logged information.

The second task of the IbdlLogger class is the logging of the executed queries a user send to the system. The logging of this information is important for the learning success of the students, as well as for the result of possible examinations. These log messages contain all the information which are needed to retrace the solutions a student executes on the system. The class provides the method logQueryResult(String username, String query, String result, boolean error). The method needs the username, which is the WBT-username, the query, the result and if the query was successful or not as a parameter. To handle all the provided information, the method works as shown in listing 3.37.

```
public void logQueryResult(String username, String query,
    String result, boolean error) {
  try {
    DatabaseConnector dc = new DatabaseConnector(new
        DbRootCredentials());
    Connection con = dc.ConnectToDB();
    PreparedStatement statement;
    String state = "";
    if (error == true) {
      state = "error";
    } else {
      state = "success";
    }
    statement = con.prepareStatement("INSERT INTO 'ibdl'.'
        user_log' ('user', 'query', 'result','state') VALUES
        (?,?,?,?);");
    statement.setString(1, username);
    statement.setString(2, query);
    statement.setString(3, result);
    statement.setString(4, state);
    statement.execute();
    dc.closeConnection();
  } catch (SQLException ex) {
    Logger.getLogger(IbdlLogger.class.getName()).log(Level.
        SEVERE, null, ex);
  }
}
```

Listing 3.37: logQueryResult

It can be seen, that the provided information is stored in the ibdl.user_log table in the database. As in all methods in the IbdlLogger, the possible thrown exceptions are not logged, because if something is wrong, the logging can not work either. Therefore, the exception is only shown in the system internal log output. After the results are logged, the system provides the possibility to view the results either in the graphical user interface as well as in the web service via the getResult method. Both ways call the method getQueryResult(String username) in the IbdlLogger, which needs a username and the database name to return a two-dimensional string array, where the first dimension contains the columns and the second dimension contains the rows. The returned result, contains the name, the query, the result and the time stamp. The result either shows if a query failed, and the source of the fail, or the extracted or modified information in the table if the query was successful. With this information, the supervisors are able to get an insight into the students proceedings in the course.

3.8. Setup

The system in the actual implementation, working as described in previous sections, have to be installed. It is inevitable to set up the necessary applications. This environment has to be used with an application server. It is designed to work with *Apache Tomcat* 2016, but it is also able to run with other application servers like Glassfish, because no system independent properties were used. The second type of software which is important to run the system is the database server. The type of the database server ca be changed to the customers need, but therefore a mutation of the DbExecuter has to be implemented to allow the internet based database laboratory to communicate with this type of database management system. As already mentioned, this default implementation uses the *MySQL Server* 2016, which have to be installed.

As already mentioned the internet based database laboratory is available as Netbeans project. It can be imported and the source can be changed to fit the customer's requirements. It is also possible to import the project in other IDEs but therefore the deployment setting has to be specified in the used development environment. Of course, it is possible to run the application directly from the IDE on the used application server. Because in productive environments this is very unlikely to do, the build which is generated by the IDE or the command line, has the specified format, belonging to the defined application server system. The default implementation, which uses Tomcat generates a web application archive (war) file. which can then be deployed on the application server. The process of deploying and configuration is described in section 3.8.1. The MySQL server also has to be configured to work with the actual implementation. To do so the steps mentioned in section 3.8.2 have to be maintained, to guarantee the described functionality and have a solid functioning system. All necessary setup facets are mentioned in the following chapters, as well as a description of the implementation. Following all mentioned aspects is important.

3.8.1. Tomcat

The Tomcat application server is the most important part of the system. Without it, the provided web application archive can not be deployed. The server is developed at *Apache Tomcat* 2016 and can be downloaded for the most common operation systems. The preferred version is 8.0, but it is also possible to run the system on the actual version 9.0. After the setup procedure of the provides install executable is done, the server.xml file has to be changed to configure HTTPS. This file is located in the installation path of the server. To configure a keystore which is mentioned in section 3.4.6, the following lines shown in listing 3.38 have to be added to the server.xml file.

```
<Connector SSLEnabled="true" acceptCount="100"
  clientAuth="false" disableUploadTimeout="true"
  enableLookups="false" maxThreads="25" port="8443"
  keystoreFile="PATH_TO_KEYSTORE/.keystore"
  keystorePass="PASSWORD"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  scheme="https"
  secure="true" sslProtocol="TLS" />
```

Listing 3.38: server.xml

If the server is already running, a restart has to be introduced, to load the new settings and provide HTTPS. To test the configuration the URL `http://<YOURHOST>:8443` can be opened in a browser. If it is working, the browser recognizes the provided certificate. To deploy the application, the simplest way is to open the server manager at `http://<YOURHOST>:8443/manager`.

To log in a user with the role ”manager-gui” have to be specified first in the file tomcat-users.xml. In the manager, it is possible to deploy a new application by selecting the web archive (war) file provided for this project. After the deployment, the application is available at `http://<YOURHOST>:8443/InternetBasedDatabaseLaboratory/` and can be used. But before the internet based database laboratory is available the database system have to be configured.

3.8.2. MySQL

The second service with has to be available for the system is the database system. In this implementation, MySQL is featured. Therefore, it is essential to install a server. The installation files can be retrieved on the website provided by *MySQL Server* 2016. After the execution of the database server and the execution of the initialization assistant, where the root name and the password are specified, the database can be accessed by the command line interface. To initialise the ibdl - database schema, it is necessary to import the provided .sql file. The init_ibdl.sql file contains all data the system needs to start up successfully. To import the script the following command has to be executed in the CLI.

```
mysql -u root -p < init_ibdl.sql
```

Listing 3.39: Load init script

The password of the root user has to be specified when asked for it. After the execution finished, a database schema ibdl should be available. This can be verified by executing the following commands in the CLI.

```
SHOW DATABASES;
USE ibdl;
SHOW TABLES;
```

Listing 3.40: Load init script

If the schema ’ibdl’ is disposable, and all six tables, which are mentioned in 3.4.7 are showed, the root credentials have to be changed in the IBDL-source, and after that, the system can be accessed with default username ”admin” and password ”password”. If the login process was successful the entire installation process was successful.

As both systems are installed and configured successfully, the next step is the integration of the system in the client. Therefore, the next section will show how to add the IBDL web service into an existing Java application.

3.8.3. Sample Java Client

The implementation shows an example for the localhost as host and the executeQuery method. An example is also available in the provided files called IBDLClientExample, and demonstrates how to integrate the service with its provided information. To implement the web service into the client, the following steps have to be done.

At first, a SOAP connection has to be initialised. The URL for the connection can be specified later because the connection object does not need it in the idle state. To create a new connection the objects in shown in listing 3.41 have to be created.

```
SOAPConnectionFactory soapConnectionFactory =
    SOAPConnectionFactory.newInstance();
SOAPConnection soapConnection = soapConnectionFactory.
    createConnection();
```

Listing 3.41: Create SOAP connection

After that, the request can be generated. This step is important because only valid requests are accepted by the server and if it is wrong, a fault will be returned. To generate the request the Java SOAPMessage object is used, and the request is build up step by step, starting with the envelop, adding a header and a body. Then add the necessary information like username, database, query and API key to the body. It is important to specify the correct namespace, because otherwise, the server can not handle the message. The building of the request is shown in listing 3.42;

```
MessageFactory messageFactory =MessageFactory.newInstance();
SOAPMessage soapMessage = messageFactory.createMessage();
SOAPPart soapPart = soapMessage.getSOAPPart();
```

```
// SOAP Envelope
SOAPEnvelope envelope = soapPart.getEnvelope();
envelope.addNamespaceDeclaration("ibdl", "http://ibdl.tugraz.
    at");

// SOAP Body
SOAPBody soapBody = envelope.getBody();

// SOAP Body elements
SOAPElement soapBodyElem = soapBody.addChildElement("
    executeQuery", "ibdl");
SOAPElement soapBodyUsername = soapBodyElem.addChildElement("
    webservice-username");
soapBodyUsername.addTextNode(username);
SOAPElement soapBodyDatabase = soapBodyElem.addChildElement("
    database");
soapBodyDatabase.addTextNode(database);
SOAPElement soapBodyApi = soapBodyElem.addChildElement("query"
    );
soapBodyApi.addTextNode(query);
SOAPElement soapBodyQuery = soapBodyElem.addChildElement("user
    -api-key");
soapBodyQuery.addTextNode(api_key);

MimeHeaders headers = soapMessage.getMimeHeaders();
headers.addHeader("SOAPAction", "https://localhost:8443/
    InternetBasedDatabaseLaboratory/IbdlWS/" + "executeQuery")
    ;

soapMessage.saveChanges();
```

Listing 3.42: Create request

After the response is generated, it can be sent to the web service by calling the call(request, URL) method of the SOAPConnection object, with the request and the URL of the web service as parameters. The method returns the response which can then be handled, by parsing it. For the demonstrated method this can be done in the way shown in listing 3.43.

```
OAPPart soapPart = response.getSOAPPart();
SOAPEnvelope envelope = soapPart.getEnvelope();
SOAPBody soapBody = envelope.getBody();
```

```
NodeList list = soapBody.getElementsByTagName("item");
String row = "";
for (int i = 0; i < list.getLength(); i++) {
    if (list.item(i).getChildNodes().getLength() == 1) {
        row += "|" + cellString(list.item(i).getTextContent())
            ;
    } else {
        if (!row.isEmpty()) {
            row += "|";
            System.out.println(row);
            row = "";
        }
    }
}
```

Listing 3.43: Parse response

The parsing method transforms the response into a table looking console output to display the table in the demonstration method. This parsing method should only demonstrate how the nodes of the XML response can be accessed.

3.9. Summary

The implementation of the internet based database laboratory shows, that in the development phase the aspect was always kept on the fact, that the system has to feel like a real database system. This is essential to guarantee a high learning rate, and motivate the supervisors and students to work with that system. To use the system not only in the current implementation and be able to modify it to the changing requirements, every important part of the system is described in this section.

The graphical user interface, as well as the web service, provide the aspects mentioned in chapter 2. The GUI is meant to be the number one configuration tool for this system because it provides all functionalities in a way, the supervisors should be able to create courses with little afford. The web service provides most methods, but the user has to take more care of how information is gained and provided as mentioned before. Another aspect is the extensibility of the system. It is able to handle MySQL database courses in the actual implementation, but with extensions for other database management systems, which have to implement the interface DBExecutor, the system can handle other database systems. This is important because, in the domain of information technology, the life cycle of systems can be quite short because systems can be outstripped by newer technologies.

Another important fact is the SOAP interface and the concerning web service description file. This is essential for the implementation of the client because the developer needs these informations to build it. The specification of the SOAP messages can also be found in the appendix of this thesis. The project also implements verification measurements to provide confidentiality, and authenticity. If needed, the system can also provide integrity. To provide authenticity, an API manager is implemented and responsible for the API key handling. To gain confidentiality, the whole system, the web service as well as the graphical user interface provide transport over the HTTPS protocol. But there are also the system internal protection mechanisms which are responsible for sandboxing and pre-filtering. These systems gain the advantage that the system can be adapted to the needs of the database course. Summarized it has to be noticed, that the system is constructed to be fully extensible, maintainable and reliable. All important aspects are mentioned in this section.

4. Further development

The IBDL is ready to be applied in database courses at TU Graz. Because of the completely new development of such a system, there can occur troubles. The actual implementation should show, that it is possible to execute SQL queries from a learning management system on a real MySQL environment so that it could be called as proof of concept. To improve the system and to gain more advantage out of it, several aspects can be enhanced.

Also, the functionality of the system can be extended, but it has to be clear what the service should provide. Should it be a stand-alone e-learning tool, or shall the system assist and existing tool to provide this functionality, as it is done by now. It is a fine line to determine the role of the system, but the actual implementation sets the tone to find an answer for this question.

4.1. Outlook

The internet based database laboratory provides the required functionalities to support an existing e-learning environment to organize and execute database courses. The defined requirements clearly define the service as an assistance system for e-learning environments. As already mentioned, there are some aspects which should be added in future implementations to gain more advantages.

The first extension can be the modification of provided graphical user interface. The usability has to be checked and in case of any new insights, the interface should be adapted. An integration of a database creation assistant, which combines all steps and create new courses is also an aspect of the implementation. But here the question raises if it is meaningful to implement such an assistant on the internet based database laboratory, or would it be better to construct it on the web service. However, the methods for both species mentioned are already present and therefore such a modification can be realized with reasonable effort.

The second aspect which can be revised is the result evaluation. At the moment the results of every single execution by a user are logged and the supervisors can extract them to evaluate the exam result or to help the students to finish their tasks. As already described in section 2.4.3 an automated evaluation of the result would be a good feature. With this new module, the supervisor is able to automatically check if the result is the same as the expected one. Such a module with the described functionality can replace the current result page in the GUI and in the web service, a new service method has to be provided. The third measure which should be enlarged is the sandbox mode. As already mentioned in the protection system section, the system provides sand-boxing for INSERT, DELETE and UPDATE queries. The backup process is a little complicated and can be done in a more elegant way. But therefore the operating system have to be known because a third-party process has to be started because the JDBC driver in its actual version is not able to provide access to the mysqldump executable. With access to this process it is possible to back up database schemas by only calling the command mysqldump -u root -p DATABASENAME >backup.sql. The generated SQL script can then be used to reload the database after the execution of the sandbox query. Therefore, it would also be possible to execute more than one query and start and stop sandbox mode on request.

The mentioned features clear aspects of the category optional equipment. For all implementations, the principle of keeping the system easy to use should not be forgotten because the best features are useless if nobody sees the advantage for using it. More features, which should be developed will evolve from the operation of the system in database courses. Maybe there are also parts of the actual system which are useless and should be removed from the actual implementation.

5. Conclusion

This thesis represents the planning and the implementation work for the internet based database laboratory. Also, the domain of e-learning is described because, without deeper knowledge, the development of a web service which should support e-learning environment can not be done successfully. To develop this system, it was necessary to get a deeper knowledge of database systems, Java Web, SOAP and JDBC. The knowledge was gained from documentations provided by the developer of the different components. The decision to use the mentioned systems was made because of the requirements. Another reason for the decisions made in this work was the e-learning aspect mentioned in the first chapter. The system was designed in the second chapter, were all requirements are defined. The system architecture is described. Another aspect mentioned here is the fact that the system should use SOAP for data exchange and different database has to be available.

The technical implementation was done by the use of Java Web. To develop the system, a MVC pattern was used to guarantee extensibility. This is important because the system should be able to use different database management systems in the future. Also, the possibility to develop new components or extensions is provided. To implement SOAP, the JAX-WS framework was used because it is a standardized way to provide web services to the client. The generation of the web service description file can be done. The integration of the MYSQL database management system is realized with the JDBC driver provided by Oracle. Therefore many database operations can be called via the Java code, which is extremely important for this system. The system itself provides two methods to execute queries directly on the database, and a user would not recognize that he is not directly working on a database management system. This aspect is important to gain a high learning rate, and the course participants are able to work on real database management systems. As with any new system, there is also room for improvement, but the core functionality is given. The improvements are not entirely necessary but will provide more comfort. It was a great experience to develop the internet based database laboratory, and also many lessons were learned from it.

5.1. Lessons learned

The author of this thesis was able to gain knowledge in the domain of e-learning systems and Java Web. The challenge was the combination of the different components. The JDBC driver was already used for several projects, but it was not clear that nearly every MySQL operation is possible with it. Another challenge was the creation of the web project itself and the resulting web service as well as the graphical user interface. The usage of the MVC pattern was not very complex to implement because Java Web already provides a good possibility to do so. The creation of the web service was not trivial because there are many annotations and properties to define to get a reliable system. The communication over SOAP and the decoding of the different responses was an interesting fact that could be solved satisfyingly.

The other aspect which was learned was the importance of e-learning in the actual century. For the author, e-learning was that thing what everybody had to use in the university in some courses. A few courses provide e-learning with extra information which gained extra knowledge, but it was not clear what was the benefit. In the research for this project mentioned in chapter one, the topic of e-learning became clear and it was tried to avoid a problem of e-learning in the system.

Summarized it has to be mentioned, that the development of the system was not as simply as thought before the start of the project. There have always been minor inconsistencies that had to be solved first to develop the system further, but it has to be noticed that the development gained a lot of new knowledge.

Acronyms

API	Application Programming Interface
CLI	Command Line Interface
DBMS	Database Management System
GUI	Graphical User Interface
IBDL	Internet Based Database Laboratory
JAX-WS	Java API for XML Web Services
JDBC	Java Database Connector
JDK	Java Development Kit
MVC	Model-View-Controller
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
VDE	Virtual Learning Environment
W3C	World Wide Web Consortium
WBT	Web Based Training
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language

Appendix

Appendix A.

Appendix

A.1. WSDL file

```
<?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's␣
      version␣is␣JAX-WS␣RI␣2.2-hudson-740-.␣--><!--␣Generated␣by␣JAX-WS␣RI␣at␣http://jax-ws.dev.java.net
      .␣RI's version is JAX-WS RI 2.2-hudson-740-. --><definitions xmlns:wsu="http://docs.oasis-open.org
      /wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-
      policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org
      /2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http:
      //ibdl.tugraz.at" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/
      wsdl/" targetNamespace="http://ibdl.tugraz.at" name="IbdlWS">
   <types>
         <xsd:schema>
               <xsd:import namespace="http://jaxb.dev.java.net/array" schemaLocation="https://
                     localhost:8443/InternetBasedDatabaseLaboratory/IbdlWS?xsd=1" />
         </xsd:schema>
         <xsd:schema>
               <xsd:import namespace="http://ibdl.tugraz.at" schemaLocation="https://
                     localhost:8443/InternetBasedDatabaseLaboratory/IbdlWS?xsd=2" />
         </xsd:schema>
   </types>
   <message name="getResult">
         <part name="superuser-api-key" type="xsd:string" />
         <part name="wbt-username" type="xsd:string" />
         <part name="database-name" type="xsd:string" />
   </message>
   <message name="getResultResponse">
         <part xmlns:ns1="http://jaxb.dev.java.net/array" name="return" type="
               ns1:stringArrayArray" />
   </message>
   <message name="IbdlException">
         <part name="fault" element="tns:IbdlException" />
   </message>
   <message name="setPermissions">
         <part name="superuser-api-key" type="xsd:string" />
         <part name="db-username" type="xsd:string" />
         <part xmlns:ns2="http://jaxb.dev.java.net/array" name="permissions" type="
               ns2:stringArray" />
   </message>
   <message name="setPermissionsResponse" />
   <message name="deleteDatabase">
         <part name="database_name" type="xsd:string" />
         <part name="superuser-api-key" type="xsd:string" />
   </message>
   <message name="deleteDatabaseResponse">
         <part name="return" type="xsd:string" />
   </message>
   <message name="uploadDump">
```

```
		<part name="dump" type="xsd:string" />
		<part name="dumpname" type="xsd:string" />
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="uploadDumpResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="listDatabases">
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="listDatabasesResponse">
		<part xmlns:ns3="http://jaxb.dev.java.net/array" name="return" type="ns3:stringArray" /
			>
	</message>
	<message name="createEmptyDump">
		<part name="dumpname" type="xsd:string" />
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="createEmptyDumpResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="reinitDatabase">
		<part name="database" type="xsd:string" />
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="reinitDatabaseResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="hello">
		<part name="name" type="xsd:string" />
	</message>
	<message name="helloResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="listPermissions">
		<part name="superuser-api-key" type="xsd:string" />
		<part name="db-username" type="xsd:string" />
	</message>
	<message name="listPermissionsResponse">
		<part xmlns:ns4="http://jaxb.dev.java.net/array" name="return" type="ns4:stringArray" /
			>
	</message>
	<message name="createDatabase">
		<part name="dump" type="xsd:string" />
		<part name="wbt-username" type="xsd:string" />
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="createDatabaseResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="listDumps">
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="listDumpsResponse">
		<part xmlns:ns5="http://jaxb.dev.java.net/array" name="return" type="ns5:stringArray" /
			>
	</message>
	<message name="deleteDatabaseUser">
		<part name="db-username" type="xsd:string" />
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="deleteDatabaseUserResponse">
		<part name="return" type="xsd:string" />
	</message>
	<message name="listUsers">
		<part name="superuser-api-key" type="xsd:string" />
	</message>
	<message name="listUsersResponse">
		<part xmlns:ns6="http://jaxb.dev.java.net/array" name="return" type="ns6:stringArray" /
			>
```

```
</message>
<message name="executeQuerySandboxMode">
        <part name="wbt-username" type="xsd:string" />
        <part name="database" type="xsd:string" />
        <part name="query" type="xsd:string" />
        <part name="user-api-key" type="xsd:string" />
</message>
<message name="executeQuerySandboxModeResponse">
        <part xmlns:ns7="http://jaxb.dev.java.net/array" name="return" type="
              ns7:anyTypeArrayArray" />
</message>
<message name="deleteDump">
        <part name="dump_name" type="xsd:string" />
        <part name="superuser-api-key" type="xsd:string" />
</message>
<message name="deleteDumpResponse">
        <part name="return" type="xsd:string" />
</message>
<message name="executeQuery">
        <part name="webservice-username" type="xsd:string" />
        <part name="database" type="xsd:string" />
        <part name="query" type="xsd:string" />
        <part name="user-api-key" type="xsd:string" />
</message>
<message name="executeQueryResponse">
        <part xmlns:ns8="http://jaxb.dev.java.net/array" name="return" type="
              ns8:anyTypeArrayArray" />
</message>
<portType name="IbdlWS">
        <operation name="getResult" parameterOrder="superuser-api-key_wbt-username_database-
              name">
                <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/getResultRequest" message="
                      tns:getResult" />
                <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/getResultResponse" message="
                      tns:getResultResponse" />
                <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                      ibdl.tugraz.at/IbdlWS/getResult/Fault/IbdlException" />
        </operation>
        <operation name="setPermissions" parameterOrder="superuser-api-key_db-username_
              permissions">
                <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/setPermissionsRequest" message
                      ="tns:setPermissions" />
                <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/setPermissionsResponse"
                      message="tns:setPermissionsResponse" />
                <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                      ibdl.tugraz.at/IbdlWS/setPermissions/Fault/IbdlException" />
        </operation>
        <operation name="deleteDatabase" parameterOrder="database_name_superuser-api-key">
                <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDatabaseRequest" message
                      ="tns:deleteDatabase" />
                <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDatabaseResponse"
                      message="tns:deleteDatabaseResponse" />
                <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                      ibdl.tugraz.at/IbdlWS/deleteDatabase/Fault/IbdlException" />
        </operation>
        <operation name="uploadDump" parameterOrder="dump_dumpname_superuser-api-key">
                <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/uploadDumpRequest" message="
                      tns:uploadDump" />
                <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/uploadDumpResponse" message="
                      tns:uploadDumpResponse" />
                <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                      ibdl.tugraz.at/IbdlWS/uploadDump/Fault/IbdlException" />
        </operation>
        <operation name="listDatabases">
                <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/listDatabasesRequest" message=
                      "tns:listDatabases" />
                <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/listDatabasesResponse"
                      message="tns:listDatabasesResponse" />
                <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                      ibdl.tugraz.at/IbdlWS/listDatabases/Fault/IbdlException" />
```

```
</operation>
<operation name="createEmptyDump" parameterOrder="dumpname_superuser-api-key">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/createEmptyDumpRequest"
              message="tns:createEmptyDump" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/createEmptyDumpResponse"
              message="tns:createEmptyDumpResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/createEmptyDump/Fault/IbdlException" />
</operation>
<operation name="reinitDatabase" parameterOrder="database_superuser-api-key">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/reinitDatabaseRequest" message
              ="tns:reinitDatabase" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/reinitDatabaseResponse"
              message="tns:reinitDatabaseResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/reinitDatabase/Fault/IbdlException" />
</operation>
<operation name="hello">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/helloRequest" message="
              tns:hello" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/helloResponse" message="
              tns:helloResponse" />
</operation>
<operation name="listPermissions" parameterOrder="superuser-api-key_db-username">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/listPermissionsRequest"
              message="tns:listPermissions" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/listPermissionsResponse"
              message="tns:listPermissionsResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/listPermissions/Fault/IbdlException" />
</operation>
<operation name="createDatabase" parameterOrder="dump_wbt-username_superuser-api-key">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/createDatabaseRequest" message
              ="tns:createDatabase" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/createDatabaseResponse"
              message="tns:createDatabaseResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/createDatabase/Fault/IbdlException" />
</operation>
<operation name="listDumps">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/listDumpsRequest" message="
              tns:listDumps" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/listDumpsResponse" message="
              tns:listDumpsResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/listDumps/Fault/IbdlException" />
</operation>
<operation name="deleteDatabaseUser" parameterOrder="db-username_superuser-api-key">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDatabaseUserRequest"
              message="tns:deleteDatabaseUser" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDatabaseUserResponse"
              message="tns:deleteDatabaseUserResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/deleteDatabaseUser/Fault/IbdlException" />
</operation>
<operation name="listUsers">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/listUsersRequest" message="
              tns:listUsers" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/listUsersResponse" message="
              tns:listUsersResponse" />
        <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
              ibdl.tugraz.at/IbdlWS/listUsers/Fault/IbdlException" />
</operation>
<operation name="executeQuerySandboxMode" parameterOrder="wbt-username_database_query_
    user-api-key">
        <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/executeQuerySandboxModeRequest
              " message="tns:executeQuerySandboxMode" />
        <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/
              executeQuerySandboxModeResponse" message="
              tns:executeQuerySandboxModeResponse" />
```

```
            <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                ibdl.tugraz.at/IbdlWS/executeQuerySandboxMode/Fault/IbdlException" />
        </operation>
        <operation name="deleteDump" parameterOrder="dump_name_superuser-api-key">
            <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDumpRequest" message="
                tns:deleteDump" />
            <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/deleteDumpResponse" message="
                tns:deleteDumpResponse" />
            <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                ibdl.tugraz.at/IbdlWS/deleteDump/Fault/IbdlException" />
        </operation>
        <operation name="executeQuery" parameterOrder="webservice-username_database_query_user-
            api-key">
            <input wsam:Action="http://ibdl.tugraz.at/IbdlWS/executeQueryRequest" message="
                tns:executeQuery" />
            <output wsam:Action="http://ibdl.tugraz.at/IbdlWS/executeQueryResponse" message
                ="tns:executeQueryResponse" />
            <fault message="tns:IbdlException" name="IbdlException" wsam:Action="http://
                ibdl.tugraz.at/IbdlWS/executeQuery/Fault/IbdlException" />
        </operation>
</portType>
<binding name="IbdlWSPortBinding" type="tns:IbdlWS">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
        <operation name="getResult">
            <soap:operation soapAction="" />
            <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </input>
            <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </output>
            <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
            </fault>
        </operation>
        <operation name="setPermissions">
            <soap:operation soapAction="" />
            <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </input>
            <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </output>
            <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
            </fault>
        </operation>
        <operation name="deleteDatabase">
            <soap:operation soapAction="" />
            <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </input>
            <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </output>
            <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
            </fault>
        </operation>
        <operation name="uploadDump">
            <soap:operation soapAction="" />
            <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </input>
            <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
            </output>
            <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
            </fault>
```

```
</operation>
<operation name="listDatabases">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="createEmptyDump">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="reinitDatabase">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="hello">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
</operation>
<operation name="listPermissions">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="createDatabase">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
```

```
<operation name="listDumps">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="deleteDatabaseUser">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="listUsers">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="executeQuerySandboxMode">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="deleteDump">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
        </fault>
</operation>
<operation name="executeQuery">
        <soap:operation soapAction="" />
        <input>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </input>
        <output>
                <soap:body use="literal" namespace="http://ibdl.tugraz.at" />
        </output>
        <fault name="IbdlException">
                <soap:fault name="IbdlException" use="literal" />
```

```
                        </fault>
                </operation>
        </binding>
        <service name="IbdlWS">
                <port name="IbdlWSPort" binding="tns:IbdlWSPortBinding">
                        <soap:address location="https://localhost:8443/InternetBasedDatabaseLaboratory/
                              IbdlWS" />
                </port>
        </service>
</definitions>
```

Bibliography

Apache Tomcat (2016). URL: `http://tomcat.apache.org/` (visited on 12/03/2016) (cit. on pp. 35, 38, 100, 101).

Aparicio, Manuela, Fernando Bacao, and Tiago Oliveira (2016). "Cultural impacts on e-learning systems' success". In: *The Internet and Higher Education* 31, pp. 58–70. ISSN: 1096-7516. DOI: `http://dx.doi.org/10.1016/j.iheduc.2016.06.003`. URL: `http://www.sciencedirect.com/science/article/pii/S1096751616300367` (cit. on p. 6).

Aparicio, Manuela, Fernando Bacao, and Tiago Oliveira (2017). "Grit in the path to e-learning success". In: *Computers in Human Behavior* 66, pp. 388–399. ISSN: 0747-5632. DOI: `http://dx.doi.org/10.1016/j.chb.2016.10.009`. URL: `http://www.sciencedirect.com/science/article/pii/S0747563216307075` (cit. on p. 6).

ATutor (2016). URL: `http://www.atutor.ca/` (visited on 11/30/2016) (cit. on p. 14).

Bootstrap (2017). *Bootstrap 3*. URL: `http://getbootstrap.com/` (visited on 02/16/2017) (cit. on p. 57).

Dillenbourg, Pierre, Daniel Schneider, and Paraskevi Synteta (2002). "Virtual Learning Environments". In: *3rd Hellenic Conference "Information & Communication Technologies in Education"*. Ed. by A. Dimitracopoulou. Rhodes, Greece: Kastaniotis Editions, Greece, pp. 3–18. URL: `https://telearn.archives-ouvertes.fr/hal-00190701` (cit. on p. 7).

Dokeos (2016). URL: `http://www.dokeos.com/` (visited on 11/30/2016) (cit. on p. 14).

Downes, Stephen (2005). "E-learning 2.0". In: *eLearn* 2005.10, pp. 1–. ISSN: 1535-394X. DOI: `10.1145/1104966.1104968`. URL: `http://doi.acm.org/10.1145/1104966.1104968` (cit. on p. 3).

Ebner, M. (2007). "E-Learning 2.0 = e-Learning 1.0 + Web 2.0?" In: *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pp. 1235–1239. DOI: `10.1109/ARES.2007.74` (cit. on p. 3).

Edrees, M. E. (2013). “eLearning 2.0: Learning Management Systems Readiness”. In: *e-Learning "Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity" , 2013 Fourth International Conference on*, pp. 90–96. DOI: 10.1109/ECONF.2013.57 (cit. on p. 3).

Fielding, Roy Thomas (2000). “Architectural Styles and the Design of Network-based Software Architecture”. dissertation. University of California (cit. on p. 38).

Harasim, Linda (2000). “Shift happens: online education as a new paradigm in learning”. In: *The Internet and Higher Education* 3.1–2, pp. 41–61. ISSN: 1096-7516. DOI: http://dx.doi.org/10.1016/S1096-7516(00)00032-4. URL: http://www.sciencedirect.com/science/article/pii/S1096751600000324 (cit. on pp. 8, 9).

Java (2017). *Java Technologies*. URL: https://java.com (visited on 02/15/2017) (cit. on p. 45).

JSqlParser (2017). *JSqlParser*. URL: https://github.com/JSQLParser/JSqlParser (visited on 02/16/2017) (cit. on pp. 45, 95, 96).

Khatri, B., P. Chouskey, and M. Singh (2013). “Comparative Analysis Study of E-learning and Traditional Learning in Technical Institution”. In: *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*, pp. 770–773. DOI: 10.1109/CSNT.2013.165 (cit. on p. 6).

Levenshtein, Vladimir Iosifovich (1965). “Binary codes capable of correcting deletions, insertions, and reversals.” In: *Soviet Physics - Doklady* 10.8, pp. 707–710 (cit. on p. 27).

Lowenthal, Patrick R., Brent G. Wilson, and Patrick Parrish (2009). “Context matters: A description and typology of the online learning”. In: *Paper presented at the 2009 AECT International Convention, Louisville, KY.* DOI: http://dx.doi.org/10.1016/j.iheduc.2010.10.001. URL: http://www.sciencedirect.com/science/article/pii/S1096751610000886 (cit. on p. 7).

Mason, Robin (2000). “From distance education to online education”. In: *The Internet and Higher Education* 3.1–2, pp. 63–74. ISSN: 1096-7516. DOI: http://dx.doi.org/10.1016/S1096-7516(00)00033-6. URL: http://www.sciencedirect.com/science/article/pii/S1096751600000336 (cit. on p. 10).

Mayadas, Frank (1997). “Asynchronous Learning Networks: A Sloan Foundation Perspective”. In: *Journal of Asynchronous Learning Networks, 1(1)*, pp. 1–16 (cit. on p. 5).

Moodle (2016a). URL: https://moodle.org/ (visited on 11/29/2016) (cit. on p. 13).

Moodle (2016b). *Moodle Statistics*. URL: https://moodle.net/stats/?lang=de (visited on 12/16/2016) (cit. on p. 13).

Moore, Joi L., Camille Dickson-Deane, and Krista Galyen (2011). "e-Learning, online learning, and distance learning environments: Are they the same?" In: *The Internet and Higher Education* 14.2. Web mining and higher education: Introduction to the special issue, pp. 129–135. ISSN: 1096-7516. DOI: http://dx.doi.org/10.1016/j.iheduc.2010.10.001. URL: http://www.sciencedirect.com/science/article/pii/S1096751610000886 (cit. on p. 7).

MySQL (2017). *Connector/J 5.1.40*. URL: https://dev.mysql.com/downloads/connector/j/ (visited on 02/16/2017) (cit. on pp. 45, 53, 56).

MySQL Server (2016). URL: https://dev.mysql.com/downloads/mysql/ (visited on 12/03/2016) (cit. on pp. 100, 102).

Navarro, Gonzalo (2001). "A Guided Tour to Approximate String Matching". In: *ACM Comput. Surv.* 33.1, pp. 31–88. ISSN: 0360-0300. DOI: 10.1145/375360.375365. URL: http://doi.acm.org/10.1145/375360.375365 (cit. on p. 27).

Oblinger, Diana G. and James L. Oblinger (2005). *Educating the net generation*. URL: http://net.educause.edu/ir/library/pdf/pub7101.pdf (cit. on p. 7).

O'Neill, McMahon and Tim Geraldine (2005). *Student-centred learning: what does it mean for students and lecturers?n*. URL: http://www.aishe.org/readings/2005-1/oneill-mcmahon-Tues_19th_Oct_SCL.pdf (cit. on p. 11).

Oracle (2006). *Java Technologies for Web Applications*. URL: http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html (visited on 10/05/2016) (cit. on p. 35).

Oracle (2017a). *Creating and Using SOAP Message Handlers)*. URL: https://docs.oracle.com/cd/E13222_01/wls/docs103/webserv_adv/handlers.html (visited on 02/10/2017) (cit. on p. 79).

Oracle (2017b). *Java^{TM} API for XML Web Services (JAX-WS)*. URL: http://docs.oracle.com/javase/7/docs/technotes/guides/xml/jax-ws/ (visited on 01/10/2017) (cit. on p. 35).

Oracle (2017c). *Java JDBC API*. URL: http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/ (visited on 02/16/2017) (cit. on p. 57).

Oracle (2017d). *Java Servlet Technology*. URL: http://www.oracle.com/technetwork/java/index-jsp-135475.html (visited on 02/20/2017) (cit. on p. 63).

Oracle (2017e). *JavaServer Pages Technology*. URL: http://www.oracle.com/technetwork/java/javaee/jsp/index.html (visited on 02/14/2017) (cit. on p. 58).

Oracle (2017f). *JWS Annotation Reference*. URL: https://docs.oracle.com/cd/E13222_01/wls/docs103/webserv_ref/annotations.html (visited on 01/10/2017) (cit. on p. 70).

Oracle (2017g). *Netbeans IDE*. URL: https://netbeans.org/ (visited on 02/22/2017) (cit. on p. 70).

Oracle (2017h). *Privileges Provided by MySQL*. URL: https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.htmly (visited on 02/20/2017) (cit. on p. 60).

Oracle (2017i). *Privileges Provided by MySQL*. URL: https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html (visited on 02/22/2017) (cit. on p. 93).

Oracle (2017j). *Security Mechanisms*. URL: https://docs.oracle.com/cd/E19159-01/820-1072/6ncp48v3q/index.html (visited on 02/25/2017) (cit. on p. 91).

Oracle (2017k). *TRUNCATE TABLE Syntax*. URL: https://dev.mysql.com/doc/refman/5.7/en/truncate-table.html (visited on 02/20/2017) (cit. on p. 97).

Ozkan, Sevgi and Refika Koseler (2009). "Multi-dimensional students' evaluation of e-learning systems in the higher education context: An empirical investigation". In: *Computers & Education* 53.4. Learning with ICT: New perspectives on help seeking and information searching, pp. 1285–1296. ISSN: 0360-1315. DOI: http://dx.doi.org/10.1016/j.compedu.2009.06.011. URL: http://www.sciencedirect.com/science/article/pii/S0360131509001584 (cit. on p. 4).

phpMyAdmin (2017). *phpMyAdmin*. URL: https://www.phpmyadmin.net/team/ (visited on 01/20/2017) (cit. on p. 29).

Pieri, Michelle and Davide Diamantini (2014). "An E-learning Web 2.0 Experience". In: *Procedia - Social and Behavioral Sciences* 116, pp. 1217–1221. ISSN: 1877-0428. DOI: http://dx.doi.org/10.1016/j.sbspro.2014.01.371. URL: http://www.sciencedirect.com/science/article/pii/S1877042814003887 (cit. on p. 4).

Reese, Sasha A. (2015). “Online learning environments in higher education: Connectivism vs. dissociation”. In: *Education and Information Technologies* 20.3, pp. 579–588. ISSN: 1573-7608. DOI: 10.1007/s10639-013-9303-7. URL: http://dx.doi.org/10.1007/s10639-013-9303-7 (cit. on p. 8).

Shahabadi, Mehdi Mehri and Megha Uplane (2015). “Synchronous and Asynchronous e-learning Styles and Academic Performance of e-learners”. In: *Procedia - Social and Behavioral Sciences* 176, pp. 129–138. ISSN: 1877-0428. DOI: http://dx.doi.org/10.1016/j.sbspro.2015.01.453. URL: http://www.sciencedirect.com/science/article/pii/S1877042815004905 (cit. on p. 5).

Statista (2016a). *Prognose zur Anzahl vernetzter Geräte weltweit in den Jahren 2003 bis 2020*. URL: https://de.statista.com/statistik/daten/studie/479023/umfrage/prognose-zur-anzahl-der-vernetzten-geraete-weltweit/ (visited on 11/23/2016) (cit. on p. 11).

Statista (2016b). *Size of e-learning market in 2013 and 2016, by region (in million U.S. dollars)*. URL: https://www.statista.com/statistics/501144/worldwide-elearning-market-size-by-region/ (visited on 11/23/2016) (cit. on p. 12).

TeachCenter (2016). URL: https://tugraz.at/oe/lehr-und-lerntechnologien/%20lehrtechnologien-und-services/tu-graz-teachcenter/ (visited on 11/30/2016) (cit. on p. 14).

W3C, World Wide Web Consortium (2000). *Simple Object Access Protocol (SOAP) 1.1*. URL: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486 (visited on 10/11/2016) (cit. on pp. 41, 70).

W3C, World Wide Web Consortium (2001). *Web Services Description Language (WSDL) 1.1*. URL: https://www.w3.org/TR/wsdl (visited on 10/07/2016) (cit. on p. 40).

WBT-Master (2016). URL: http://coronet.iicm.tugraz.at/wbtmaster/ (visited on 12/02/2016) (cit. on p. 15).

WBT-Master white paper (2016). URL: http://coronet.iicm.tugraz.at/demo/whitep.htm (visited on 12/02/2016) (cit. on p. 15).

Wever, B. D. et al. (2007). “E-Learning 2.0: Social Software for Educational Use”. In: *Multimedia Workshops, 2007. ISMW '07. Ninth IEEE International Symposium on*, pp. 511–516. DOI: 10.1109/ISM.Workshops.2007.91 (cit. on p. 4).