



Game

1. Introduction and Goals

The ARZ wants to establish a new way of gamified education for the members. ARZ means "Allgemeines Rechenzentrum" and it is an IT service provider with locations in Innsbruck and Vienna. Their homepage: <https://www.arz.at/>

The workers often have different knowledge about programming, so it is rather time consuming and costly to further educate them.

This is where the idea of "GAME" comes into play. The "GAME" is intended to be a platform on which employees can improve their programming knowledge with little games, quizzes or competitions.

A great focus lies on the terms "self-study" and "Gamification". Gamification means that the information is transmitted through games to ease the learning process.

Furthermore there should be a possibility for trainers to bring in their own ideas, quizzes or modules. One of the goals of this diploma thesis is to motivate the people to continue their education in their free time. There should be a possibility to learn some skills without much effort.

Another important aspect of "GAME" is the scoring system, which will be implemented in the software. This will create transparency for employees and employers. This scoring system will be a great motivation for workers.

The most relevant Stakeholders

- Roland Moder
- Konrad Renner
- Michaela Würzl
- Cornelia Sammer



The expectations of the Stakeholders should be fulfilled.

The expectations of the stakeholders regarding the thesis are:

- sufficient documentation regarding the arc42 template

- Fulfillment of the quality goals
- a clear and well-functioning software
- an easy-to-use user interface

2. Abstract (Deutsche Version)

Der Ziel der folgenden Diplomarbeit war, eine Lernplattform für Mitarbeiter einer Softwarefirma zu entwickeln. Anfangs haben wir die verschiedenen Arten wie Menschen lernen genauer betrachtet, und diese zu Lerntypen zusammengefasst. (Vester, Pask, Dunn,) Darüber sind wir auf das Thema "Wissensmanagement" näher eingegangen, und haben identifiziert, was Menschen mit ihrem Wissen anstellen. Diese Beiden Fachgebiete haben wir in unsere Website einfließen lassen, und diese so aufgebaut, dass sich Mitarbeiter Kurse anschauen können und mit dem daraus gewonnenen Wissen Quizze beantworten sollen. Das besondere hierbei ist, dass jeder "Spieler" Punkte und Embleme für richtig beantwortete Fragen bekommt. Dadurch können sich diese miteinander vergleichen und anspornen die meisten Punkte zu gewinnen. Somit ist ein spielerischer Faktor inkludiert, der dem Lernerfolg hilft.

3. Abstract (Englische Version)

In the following diploma thesis, we created a learning platform for employees of a software firm. In the beginning, we looked at the different ways to learn, which are summarized in learning types. Furthermore we looked into knowledge management and what people do with their knowledge. With these two factors in mind, we built a web-page, where employees are able to learn different programming-skills by taking courses and answering quizzes. The special thing about this tool is, that everyone is rewarded points and icons for correct answered questions. This way, employess can match and compare each other and so a playful factor is included.

4. Requirements and Overview

- create/modify a system to get a simple, easy to use education platform
- trainers should be able to upload their own tasks and games to the platform
- a variety of difficulty levels
- a point system which rewards the user and lets him compare his points with other users
- the points can also be used to rank up

5. Qualitygoals

Table 1. Quality Goals

Name	Description
System ready and usability	The goal of this project is to create or modify a system to get a learning platform for the ARZ members. So the most important quality goal is the completion of the system because there cannot be any other features if the basic system does not exist. Also the system should be easy to use/ understand and should not have flaws in it.
The point system	One of the most important goals of the education platform is to implement a point system for the users. The point system rewards the ARZ members and the users should be able to compare their points with each other. This will boost the motivation of the learners to educate themselves even more, by for example trying to reach certain points.
Trainers can upload games and tasks	Another feature should be the possibility for selected trainers to upload their own tasks and games for the users of the education platform. This should be the foundation to never run out of new games and tasks.
Up to date documentation is released with every update of the system	The documentation of the whole project is another goal which is very important for the successful completion. It should be without

Name	Description
	grammar or writing errors and should have an overall good looking format.

6. Stakeholders

- **Thesis supervisor** (contractor side): Roland Moder roland.moder@aon.at¹
- **Thesis supervisor** (client side): Konrad Renner konrad.renner@arz.at²
- **Human resource development:** Michaela Würzl michaela.wuerzl@arz.at³,
Cornelia Sammer cornelia.sammer@arz.at⁴

Expectations:

Roland Moder:

He expects the diploma thesis to run smoothly and be completed within a reasonable time, this means it has to be completed by the end of April. Additionally, the diploma students should fulfill the requirements for a diploma thesis. Furthermore, there should be a sufficient documentation of the work done.

Konrad Renner:

He expects the software to contain a scoring system that records the learning progress of every member, which can be evaluated and compared. Furthermore, he expects to be able to maintain the tasks independently, this means if all tasks have been fulfilled on the platform, there should be an opportunity for coaches to be able to put in new tasks.

Michaela Würzl and Cornelia Sammer:

They expect a timely completion of the project, so that the concept is completed by 31.12.2019, and the implementation should be completed by the end of April. Moreover, the software should be utilizable and integrable, this means the user interface should be easy to use for employees, trainers and moderators.

¹ <mailto:roland.moder@aon.at>

² <mailto:konrad.renner@arz.at>

³ <mailto:michaela.wuerzl@arz.at>

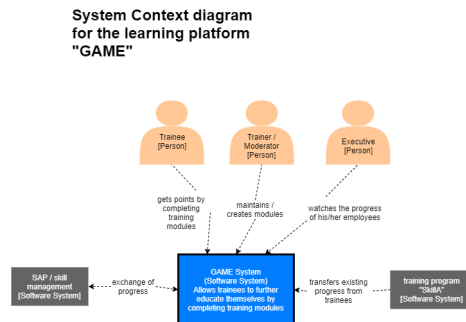
⁴ <mailto:cornelia.sammer@arz.at>

7. System, Scope and Context

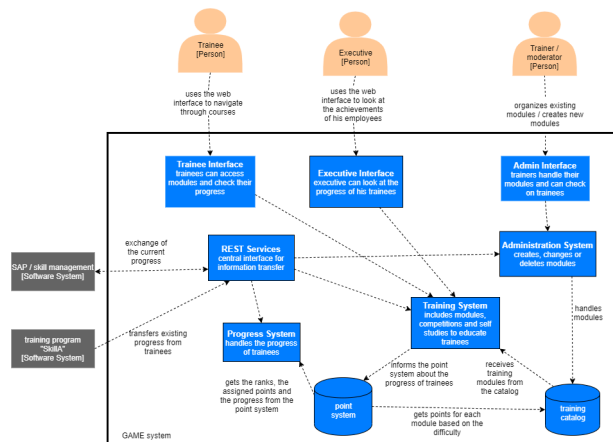
7.1. Technical Context

The technical context describes interfaces linking the system to its environment.

The abstract technical context is shown by the following C4-Context Model:



The GAME system is precisely described through this C4-Container Model:



7.2. Description

The game system communicates with the following systems:

- Skill / SAP employee management system
 - This system transmits the current progress of the employees to the GAME System and vice versa.

- training system "SkillA"
 - This system transfers progress from the "SkillA" training system to the GAME system.

7.3. Considerations

In the point system several difficulty levels are defined which reward the user with previously determined points.

The trainer has to give every module he created a certain difficulty level.

In the beginning it should be possible to manually enter the progresses of trainees from "SkillA" in the progress system.

8. Solution Strategy

8.1. *Building from Scratch*

The key decision of this project is the settlement to program the system from scratch, since there is no suitable, open-source platform which can be used for our purpose. Which systems were examined are written in [Building from Scratch](#)⁵.

8.2. *Programming Language*

The programming language we will use throughout this project is the highly acknowledged Java programming language. For in depth view of technology decisions please have a look in [Design Decisions](#).

⁵ ../decisions/ADR001-BuildingFromScratch.adoc

9. Design Decisions

This chapter includes all decisions, the diploma team had to make. The following decisions are written in the [Lightweight Architecture Decision Records⁶](https://adr.github.io/)-format.

⁶ <https://adr.github.io/>

9.1. ADR001-BuildingFromScratch

Status

accepted

Context

The ARZ wants to create a learning platform for their members. The project team evaluated possibly alternatives to adapt from. An alternative needs to be able to run on-premise, be up to date with today's standards and preferably be opensource. The evaluated systems are the following:

<http://www.javaranch.com>

- is based on "JForum" (a java forum)
- OpenSource
- Code is on SourceForge

-> doesn't meet the expectations

<http://www.skill-guru.com>

- no On-premise possible

-> therefore excluded

<http://www.betterprogrammer.com>

- no further development, outdated (java version 1.5/1.6)

-> therefore not effective

<http://www.javapassion.com>

- no possibility for extensions
- developed by an individual person

-> therefore no viable option

Decision

None of the evaluated systems meet the requirements, so the system will be build up from scratch by the project team.

Consequences

Development of a web application based on Java/Java Enterprise.

9.2. ADR002-Technology stack

Status

accpeted

Context

The GAME application consists of different subsystems. To make it possible, that these subsystems are maintainable in the context of the whole application, it is important that these subsystems are based on the same base technology stack.

Decision

The GAME application and its subsystems are developed with the help of the [Java EE⁷](#) technology stack. The Java EE technology is a proven technology which is standardised and offers therefore highest future-proofness.

Consequences

The GAME application uses Java EE in version 8 and is developed as application server neutral as possible. Version 8 is the latest production ready version of the platform and therefore it is the version of choice for developing Java EE applications.

⁷ https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

9.3. ADR003-Build and dependency managment

Status

accpeted

Context

The GAME application consists of different subsystems. To make it possible, that these subsystems can be automatically built and bundled a build and dependency management tool ist needed.

Decision

[Apache Maven](https://maven.apache.org/)⁸ will be used for build and dependency management. This tool is well-proven and can be extended with many plugins. Its convention-over-configuration paradigm lets a developer focus on creating the software.

⁸ <https://maven.apache.org/>

9.4. ADR004-UI technology stack

Status

accpeted

Context

The primary user interface of the GAME application will be web based. Therefore a web UI Toolkit is needed for an efficient development of the UI and seamless integration with the backend systems.

Decision

As described in [ADR002-Technology stack](#)⁹ Java EE is the base technology stack for the GAME application. The Java EE standard defines the JSF substandard as its UI Framework/Toolkit for web based applications. Therefore the web based UI of the GAME application will be developed with the JSF framework.

Consequences

For some parts of a modern web based UI, the JSF framework doesn't have components to develop the UI efficient. For these parts it is allowed to use components of the [Primefaces](#)¹⁰ UI library.

⁹ ADR002-Technologystack.adoc

¹⁰ <https://www.primefaces.org/>

9.5. ADR005-Communication with distributed systems

Status

accpeted

Context

The GAME application must be able to communicate with other distributed systems in both directions (inbound and outbound).

Decision

The GAME application will provide resources for other distributed systems via [RESTful webservices¹¹](#).

By using a stateless protocol ([HTTP¹²](#)) and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running. It is also evolving as the defacto standard for communication between distributed systems in the digital age.

Consequences

JAX-RS is the standard for creating RESTful Webservices in Java EE based applications and so it will be used in the GAME application.

¹¹ https://en.wikipedia.org/wiki/Representational_state_transfer

¹² https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

9.6. ADR006-Database

Status

accepted

Context

For persistent storage of data in the GAME application a database system is needed.

Decision

The [PostgreSQL](https://www.postgresql.org/)¹³ database will be used as the default database system in the GAME application.

PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. It's worth mentioning that PostgreSQL also supports many NoSQL features as well.

Consequences

PostgreSQL is defined as the default database of the application, but database accesses must be as SQL standard compliant as possible, so that the portability of the application is still warranted.

¹³ <https://www.postgresql.org/>

9.7. ADR007-Java EE application server

Status

accepted

Context

Applications which are implemented on top of the Java EE specification, which is defined as the base technology of the GAME application in the link:ADR002-Technology stack.adoc[ADR002-Technology stack], need an application server as runtime environment.

Decision

The [Wildfly application server](https://wildfly.org/)¹⁴ will be used as the default application server in the GAME application.

The Wildfly application is open source and has full support for the Java EE 8 specification. It is also widely used in the ARZ and therefore the medium of choice.

Consequences

Wildfly is defined as the default application server of the application, but the application must be as Java EE standard compliant as possible, so that the portability of the application is still warranted. Further it must be possible to run the application also on application servers which are just Java EE Web Profile compliant.

¹⁴ <https://wildfly.org/>

9.8. ADR008-Standard code formatter

Status

accpeted

Context

For seamless working together on a shared code base it is advantageous to use the same code formatter. So a standard code formatter must be defined.

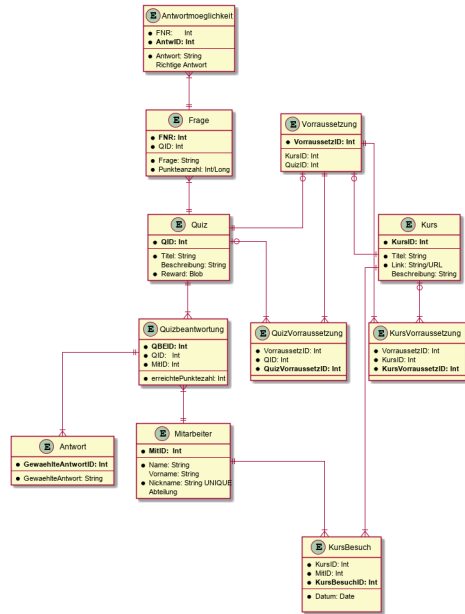
Decision

The standard code format is defined in the file [formatter.xml](#). This XML is created as an [eclipse](#)¹⁵-Formatter file, but can also be importet in other IDEs, such as Apache Netbeans or IntelliJ Idea.

Consequences

Everyone who creates or changes source code for the GAME application in this repository, must use this formatter setup to format the source code.

¹⁵ <https://www.eclipse.org/>



10. Theoretische Aufgabenstellung: Eric Haneder

11. (Lernen,) Lerntypen, Lernstile, Lernpräferenzen und selbstreguliertes Lernen

11.1. Lernen

Everyone has to learn, both actively and passively. Whether at school or in professional life, further education is part of the essence of man. The term "learning" usually means the acquisition of knowledge, the development of skills or the practice of motor processes. However, "learning" also refers to activities that are carried out with the aim of changing internal conditions.

(vgl. Selbstreguliertes Lernen in der dualen Ausbildung - Lerntypen und Bedingungen, Johannes Rosendahl, 2010)

There are many people who have tried to explain the process behind learning, or to classify different learning preferences and learning types. I will discuss these classifications in more detail below.

11.2. Learning type classification according to Vester

In the book "Denken, Lernen, Vergessen. Was geht in unserem Kopf vor, wie lernt das Gehirn und wann lässt es uns im Stich?" (1975, Stuttgart), Frederick Vester defined four types of learning. However, his book was interpreted differently, so that different versions of his classification can be found. At least on the fact that there is an auditory, a visual and a haptic learning type, there is agreement. Though there are some people who think that the fourth learning type after Frederik Vester is the communicative learning type and others think that it is an intellectual learning type.



One also comes across terms like "cognitive learning type" # "intellectual learning type", or the "kinesthetic learning type" # "haptic learning type".

Evidence for a classification according to Vester with a "communicative learning type".

<https://www.teko.ch/die-vier-lerntypen-und-ihre-besonderheiten;>
<https://www.mystipendium.de/studium/lerntypen;>

<https://karrierebibel.de/lerntypentest/#Die-vier-Lerntypen-angelehnt-an-Frederic-Vester;>
[https://learnsolution.de/die-vier-lerntypen-nach-frederic-vester/;](https://learnsolution.de/die-vier-lerntypen-nach-frederic-vester/)

Evidence for a classification to Vester with an "intellectual learning type".

[http://www.rechtschreibwerkstatt-konzept.de/wp-content/uploads/2015/02/Looss_Lerntypen.pdf;](http://www.rechtschreibwerkstatt-konzept.de/wp-content/uploads/2015/02/Looss_Lerntypen.pdf)
[https://wb-web.de/wissen/lehren-lernen/lernstile-und-lerntypen.html;](https://wb-web.de/wissen/lehren-lernen/lernstile-und-lerntypen.html)
[https://smarter-learning.de/lerntypen/klassische-lerntypen-nach-vester/;](https://smarter-learning.de/lerntypen/klassische-lerntypen-nach-vester/)
[https://wissenschafts-thurm.de/lerntypen/;](https://wissenschafts-thurm.de/lerntypen/)
"Erforschung der Lerntypen und -strategien am Beispiel einer Handelsschulklasse", Petra Hochleitner, 2016

I will be analyzing this classification according to Vester:

1. optical/visual → Learning through seeing and observing
2. auditory/acoustically → Learning through listening and speaking
3. haptisch → Learning through action (touching and feeling)
4. cognitive → Learning through recognition/intellect

Auditory learning type

The auditory learning type learns while using his hearing. He/She has no problem listening to someone over long periods of time. The auditive type prefers auditioned learning content and learns better if he/she reads the text aloud himself/herself.

Visual learning type

This learning type learns the best, when he takes up the learning content through his eyes. Reading texts will lead to great learning achievements. This gets a better understanding of facts through looking at pictures.

Haptic learning type

The haptic learning type achieves the best learning succes if he cant feel the information he has to learn. "Learning by Doing" describes this lerning method pretty good. He prefers to be actively integrated into the learning process. Practical demonstrations are helpful too.

Cognitive/Intellectual learning type

This learning type understands and saves information the best, by thinking about and critically examining the information. The perception channel not important for taking up learning content.

(vgl. "<https://smarter-learning.de/lerntypen/klassische-lerntypen-nach-veste/>"; letzter Zugriff: 14.01.2020)

Connections

Most of the time, it is not possible to assign someone a certain learning type, as there mix-types of learning. There are some people who learn faster by reading through text and writing it down afterwards (visual & haptic), and others are better of learning by reading and reciting out loud (auditory & haptic).

So if you want to learn effectively, you should try to appeal to more than one sense, to find a combination that fits your style.

How are different types of learning formed? It would be a lot easier, if everybody could learn the same way. In reality, there are many people with different learning types, this is due to changes in personal characteristics, habits and previous experiences.

So how can you find out which learning type suits you best?

On the Internet you can find many so-called "learning type tests". In these tests, you have to answer a few simple questions, that depict typical life situations. Here are two hyperlinks, which lead to easy tests: * <http://arbeitsblaetter.stangl-taller.at/TEST/HALB/Test.shtml> * <http://www.philognosie.net/index.php/tests/testview/150/>

After completing one of these tests, you will receive a recommendation as to which type of learner is more suitable for you.



It should be noted that different tests can also lead to different recommendations. This is due to the fact that the tests have different focuses and are more or less comprehensive.

If you do not want to take such tests, you can just look up different well-approved learning methods and try them yourself. Check how good you can remember something while using all of the different learning methods either alone or combined.

(vgl. *"Erforschung der Lerntypen und -strategien am Beispiel einer Handelsschulklasse"*, Petra Hochleitner, 2016)

11.3. Learning preferences according to Dunn

Dunn and Price (1989) defined learning style as a typical way of learning that is influenced by different elements of the environment. This regards:

- physical stimuli (light, sound, temperature, design)
- social stimuli (pairs, peers, adults, groups)
- stimuli of learning material (auditiv, visuell, taktil, kinästhetisch)
- emotional stimuli (responsibility, persistence, motivation, disciplin)

These factors are measured by the "Learning Styles Inventory". However, this model takes little account of the actual cognitive processes that play a role in learning.

(vgl. *"Lernorientierungen, Lernstile, Lerntypen und kognitive Stile"*, Ulrike Creß, in *"Handbuch Lernstrategien"* von Heinz Mandl & Helmut Felix Friedrich, S.373)

11.4. Learning styles according to Pask

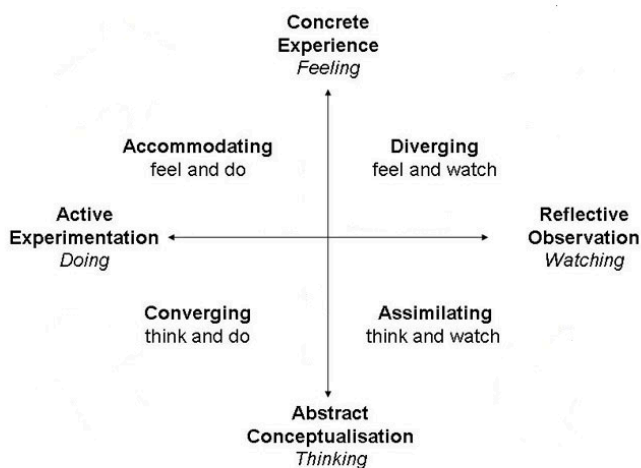
Around 1972, Pask and Scott identified two opposing learning strategies used in problem-solving tasks where people had to search for information independently. They described the consistent usage of these strategies as a learning style. The holistic strategy means that learners always keep the big picture in mind and only turn to detailed questions in a second step. If this strategy is applied consistently, Pask speaks of the learning style of comprehension learning. On the other hand, learners with a serial strategy work their way step by step through the learning material and primarily turn to individual questions. If this strategy is used consistently, Pask speaks of operation learning. Both strategies can lead to the same success. In their extreme form, however, both have a negative effect on performance, which is why Pask assigns both

learning styles to corresponding learning pathologies. *Globetrotting* refers to the learning pathology of extreme comprehension learning, in which learners make inadmissible generalizations without the corresponding individual analysis. *Improvidence* describes the extreme form of operation learning, in which people lose themselves in details without being able to connect them to a big picture. Since the differences between holistic and serial approaches affect not only learning behaviour but the entire way in which information is sought and processed, they are often interpreted as cognitive styles.

(vgl. "Lernorientierungen, Lernstile, Lerntypen und kognitive Stile", Ulrike Creß, in "Handbuch Lernstrategien" von Heinz Mandl & Helmut Felix Friedrich, S.369)

11.5. Learning styles according to Kolb

In 1984 David Kolb took a completely different approach to classifying learning types. According to Kolb, the learning process is based on two orthogonal bipolar dimensions. The first dimension depicts how people perceive and collect information. Persons can perceive via the senses through practical experience or through abstract comprehension. The second dimension represents the way information is processed. It ranges from active trying to mental observation. (orthogonal → two straight lines are called orthogonal if they enclose a 90 degree angle) The following figure shows the dimensions:



Quelle (<https://selfdirectedlearning.webnode.es/learning-styles-by-kolb/>; letzter Zugriff 28.01.2020)

Kolb presents four learning styles defined by the four quadrants that result from these orthogonal dimensions.

Convergers explore their environment through active probing and process information in an abstract way. They are therefore interested in testing their theories and solving problems deductively.

Divergers combine mental observation with practical experience. This often leads them to creative solutions.

Assimilators connect abstract comprehension with mental observation. They are therefore mainly interested in developing abstract theories and defining problems, less in solving concrete problems.

Accommodators combine active experimentation with concrete experience. They prefer casual learning directly from the situation. The learning style of a person is measured by Kolbs' Learning Style Inventory (KLSI).

Kolb's approach is by far the most frequently cited of the approaches for recording learning styles.

(vgl. "Lernorientierungen, Lernstile, Lerntypen und kognitive Stile", Ulrike Creß, in "Handbuch Lernstrategien" von Heinz Mandl & Helmut Felix Friedrich, S.371-372)

11.6. Selbstreguliertes Lernen

The concept of self-regulated learning is neither a precisely scientifically defined term nor a uniformly used term in everyday language. Furthermore, the terms self-regulated learning, self-directed learning, learner control can hardly be defined clearly.

Niegemann and Hofer (1997) or Büser (2003) define that in self-directed learning, in contrast to self-directed or self-regulated learning, the learning goal is determined by the person himself. Other authors, on the other hand, see the decision on learning goals explicitly as a component of self-directed or self-regulated learning (Arnold & Gomez-Tutor 2006; Dehnbostel 2003; Lang & Pätzold 2006; Neber 1978; Schreiber 1998, S. 45).

(vgl. Selbstreguliertes Lernen in der dualen Ausbildung - Lerntypen und Bedingungen, Johannes Rosendahl, 2010)

12. Praktische Aufgabenstellung: Eric Haneder

13. Front End: Design und Graphic User Interface

13.1. Beginning

Nachdem die Technologie festgelegt war, mit der die Website programmiert werden sollte, konnten wir anfangen. Zuerst muss aber mit dem Auftraggeber ein Design abgeklärt werden, welches ihm zusagt, und für uns umsetzbar ist.

Hier ist ein abstraktes Diagramm der User-Interfaces: <Bild einfügen>

Nachdem auch das Design abgeklärt war, konnte nun die Erstellung der Interfaces beginnen.

13.2. Grundlagen

Um eine Website zu programmieren, benötigt man eine Menge an umfangreichen Wissen. Man muss html, javascript und css beherrschen. Darüber hinaus sollte man auch wissen, wie aus dem Code eine fertige Seite erzeugt wird, und wie Requests vom User ausgewertet und darauf geantwortet werden kann.

HTML

Hypertext Markup Language (HTML) is the common used language for building web pages. A HTML page is a text document (with a .html or .htm extension) used by browsers to present text and graphics. A web page is made of content, tags to change some aspects of the content, and external objects such as images, videos, JavaScript, or CSS files.

Beispielcode für eine html page.

```
<html>
  <head>
    <title>My webpage</title>
  </head>
  <body>
    <h1>This is my webpage</h1>
    <p>
      I hope you like it.
```

```

    </p>
    <a href="www.google.at">Link to Google</a>
  </body>
</html>

```



U can notice several tags in this code (such as <body> or <p>). Every tag has its on purpose, attributes and must be closed. Href is an attribute of the a-tag.

XHTML is just a validated version of html. This means, that there are certain rules, a html-page has to follow, to be valid. XHTML pages have a .xhtml extension.

Some of the rules are the following:

- All tags must be closed. (so no
, <hr>, ...)
- All tags are lowercase.
- Attributes appear appear between single or double quotes (<table border="0"> instead of <table border = 0>)
- There must be a strict structure with <html>, <head> and <body> tags.

CSS

Cascading Style Sheets (CSS) is a styling language used to describe the presentation of a document written in html or xhtml. CSS is used to define colors, fonts layouts, and other aspects of document presentation. It allows separation of a document's content (written in XHTML) from its presentation (written in CSS). To embedd a .css file in your html or xhtml page, use the <link> tag. (e.g. <link rel="stylesheet" type="text/css" ?

Your css file could look like this:


```

p {
    font-size: 10px;
}
h1 {
    color: red;
    font-style: italic;
}

```

13.3. Getting Started

Projektstruktur erstellen

Um mit dem Programmieren zu beginnen, muss erst eine stabile Projektstruktur gegeben sein. Dafür haben wir in NetBeans auf "Projekt erstellen" geklickt, alle benötigten Ressourcen vereinbart, und mit Maven wurde dann automatisch eine passende Projektstruktur erstellt. Nun muss man auch darauf achten, dass diese Struktur befolgt. Java-Dateien sollten unter `src\main\java` abgelegt werden. Everything regarding JSF should lie in `src\main\webapp`. That includes the `html`-files, the `CSS`, `JavaScript` and `Images` should lie in a untergeordnetem `resources` folder. This is needed, for JSF to locate the resources faster.  welches die Projektstruktur zeigt

Layout erstellen

Um die Erstellung mehrerer UIs zu vereinfachen, wird ein Layout erstellt, welches als ein Template für die Pages dient. Dieses Layout habe ich in der `mainlayout.xhtml` realisiert.

mainlyout.xhtml.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

    <h:head>
        <link rel="shortcut icon" type="image/x-icon" href="faces/
favicon.ico" />
    </h:head>

    <h:body>
        <div id="header">

            <h:graphicImage library="img" name="header.png" style="width: 100%"></
h:graphicImage>
        </div>
```



```

<ui:insert name="menubar">
    <ui:include src="./faces/menubar.xhtml"></ui:include>
</ui:insert>

<ui:insert name="header">
    <h:outputStylesheet library="resources" name="css/
stylesheet.css"></h:outputStylesheet>
</ui:insert>
<br />

<ui:insert name="content">
    Content
</ui:insert>
<br />

<h:graphicImage library="img" name="footer1.jpg" style="top:100%;
margin-bottom: 0px; padding-bottom: 0px;"></h:graphicImage>
</h:body>
</html>

```

Diese Seite wird dem Benutzer nicht direkt angezeigt, sondern sie stellt eine Vorlage für alle User-Interfaces dar. Andere Seiten können diese Seite als template definieren, und übernehmen dann diesen Inhalt. Indem man in dem Template-file `<ui:insert>` benutzt, kann man den Template-Clients die Möglichkeit geben, den Inhalt dieses Tags mit `<ui:define>` selbst zu bestimmen.

<Bild einfügen>

Erklärung.

Anfangs wird dem Browser die Information weitergegeben, dass es sich um eine xhtml Datei handelt. Weiters werden die Buchstaben h, ui, f und p definiert, die alle auf eine jsf-library verweisen (Diese Verweise sind nötig, damit der Browser die JSF-Tags lesen kann). Im head wird hier keine Information definiert (!). Im body wird eine Logo angezeigt, mit einer Überschrift daneben (`<h1>`). Der div-tag ist nur dazu da, um einen Bereich abzugrenzen, und diesen dann mit CSS zu positionieren/formatieren. Mit `<ui:insert>` gibt man im mainlayout Informationen an, die die Clients des Layouts individuell einbauen, oder überschreiben wollen. `<h:outputStylesheet>` wird zum Verknüpfen der CSS-Datei verwendet. Mit `<ui:include>` können wir Informationen einer anderen Page, in diese Page einbauen.

Im footer-Bereich werden noch Kontaktdaten angezeigt.

menubar.xhtml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

  <body>
    <h:form id="menubar">
      <div id="menu">

        <p:tabMenu activeIndex="#{param.i}" style="width:133.2%">

          <p:menuitem value="Home" outcome="index.xhtml" style="width:5em" icon="fa
fa-home">
            <f:param name="i" value="0" />
          </p:menuitem>

          <p:menuitem value="Courses" outcome="courses.xhtml" style="width:7em" icon="fa
fa-book">
            <f:param name="i" value="1" />
          </p:menuitem>

          <p:menuitem value="Quizzes" outcome="quizzes.xhtml" style="width:7em" icon="fa
fa-question">
            <f:param name="i" value="2" />
          </p:menuitem>

          <p:menuitem value="Trainers" outcome="trainers.xhtml" style="width:5em" icon="fa
fa-users">
            <f:param name="i" value="3" />
          </p:menuitem>

          <p:menuitem value="Emblemtafel" outcome="leaderboard.xhtml" style="width:8em" icon="fa
fa-eye">
            <f:param name="i" value="4" />
          </p:menuitem>
        </p:tabMenu>
      </div>
    </h:form>
  </body>
</html>
```

```

<p:menuitem value="Help" outcome="help.xhtml" icon="fa fa-
question" style="width:5em">
    <f:param name="i" value="5" />
</p:menuitem>
</p:tabMenu>
</div>
<br />

<div id="logout">

<p:splitButton id="basic" value="Account" action="index.html"
icon="pi pi-save">

<p:menuitem value="Points" action="index.html" icon="pi pi-refresh" />

<p:menuitem value="Courses" action="courses.xhtml" ajax="false" icon="pi
pi-times" />
    <p:separator />
    <p:menuitem value="Logout" url="http://
www.google.com" icon="pi pi-home" />
</p:splitButton>
</div>
</h:form>
</body>
</html>

```

This page is unique, because its only purpose is to build the menubar, which is displayed on every Page. I had to split the menubar from the mainlyout, because of interferences with the formulars.

13.4. User-Interfaces

Die UIs liegen hier unter src/main/webapp/faces. "ev. noch was zu UIs sagen"

The index-page is the standard page the browser will run, if you enter a website. On this page, the user should get an overview about his statistics, and he should be able to navigate to other pages.

index.xhtml.

```

<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```

```

<ui:composition template="../../mainlayout.xhtml" ❶
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.org/ui">

❷
    <ui:define name="header"> ❸
        <title>Activities</title>
        <h:outputStylesheet library="css" name="stylesheet.css" />
    </ui:define>

    <ui:define name="content">
        <h1>Activities</h1>
        <p>
            <strong>User:</strong>
            #{traineeController.trainees.get(0).name} <br />
            <strong>Nickname:</strong>
            #{traineeController.trainees.get(0).nickname} <br/>
            Progress: 50% out of 100% <br />

            Badges: <h:graphicImage library="img" name="#{traineeController.trainees.get(0).emblem}" />
            h:graphicImage>

            </p>
            <p>
                Your score (Quiz):
                <h:outputText value="#{testQuizController.score}"></
h:outputText> <br/>
                Your progress (Overall):
                <h:outputText value="#{traineeGenerator.scores.get(0)}"></
h:outputText>
            </p>
            <p>
                Hier sollte spezifisch zum User, dessen Aktivitäten und
                Fortschritt dargestellt werden.
                weiters können hier Kurse vorgeschlagen werden.
            </p>
            <br />
        </ui:define>
    </ui:composition>

```

- ❶ As you can see, the index-page is using the mainlayout with the "template" attribute.

- ② Everything above this callout will be same for every template client. I will therefore only include it here, to shorten the other code-snippets.
- ③ I defined an individual title with <ui:define>. The content of this page is enclose in <ui:define> too.

We dont need to set a footer here, because in the template page, there is already an universal footer defined.

Courses-page

The Courses-page should display a list of courses the trainee can go through. These courses can be mandatory to complete Quizzes.

courses.xhtml.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<ui:composition template="../../mainlayout.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.org/ui">

    <ui:define name="header">
        <title>Kurse</title>
        <h:outputStylesheet library="css" name="stylesheet.css" />
    </ui:define>

    <ui:define name="content">
        <h1 align="center">Courses</h1>
        <p>
            <strong>Java SE Learning Beginner Course</strong> <br/>
            This course will give you some basic knowledge of Java SE.
            <h:form>
                <h:commandButton action="#{kursController.takeKurs}"
value="Take Course!" ></h:commandButton>
            </h:form>
        </p>
        <p>
            <strong>Creating a GUI with Swing</strong> <br/>
            A comprehensive introduction to GUI creation on the Java
platform.
```

```

        <p:button href="https://docs.oracle.com/javase/tutorial/
uiswing/index.html" value="Take Course!"></p:button>
    </p>
    <p>
        <strong>A Guide for JavaBeans</strong> <br/>
        The Java platform's component technology.
        <a href="https://docs.oracle.com/javase/tutorial/javabeans/
index.html">Take Course!</a>
    </p>
    <p>
        <strong>JDBC Database Access Course</strong> <br/>
        Introduces an API for connectivity between the Java
        applications and a wide range of databases and data sources.
        <a href="https://docs.oracle.com/javase/tutorial/jdbc/
index.html">Take Course!</a>
    </p>
</ui:define>
</ui:composition>

```

At the the end there is a `<h:link>` tag. This is a JSF tag that resolves in to an `<a>` tag, leading to another page. There is a link that leads to a page where you can take a quiz, and there is a link where you can create a quiz. More on that later.

Quiz-pages

quizzes.xhtml.

```

<!-- <ui:composition ... -->

<ui:define name="header">
    <title>Quizzes</title>
    <h:outputStylesheet library="css" name="stylesheet.css" />
</ui:define>

<ui:define name="content">
    <h1 align="center">Quizzes</h1>
    <table id="quizzes">
        <tr>
            <th>
                Title
            </th>
            <th>
                Description
            </th>
        </tr>
    </table>

```

```

        <tr>
            <td>
                Java Learning Quiz - Basics (Single Choice)
            </td>
            <td>
                This quiz will check your knowledge on basic java
                skills.

                <h:outputText value="#{testQuizController.quizBezeichnung}"></h:outputText>
            </td>
            <td>
                <h:form>

                <p:commandButton action="#{testQuizController.quizuebergabe()}" value="Take Quiz!">

                    <f:param name="QID" value="1"></f:param>

                    <!-- value sollte dann mit ui:repeat raufgezogen werden und alle Quizzes sollen so nacheinander angezeigt werden -->
                </p:commandButton>

                <!--value="{QuizEJB.getQID}"-->
                </h:form>
            </td>
        </tr>
        <tr>
            <td>
                Java Learning Quiz - Advanced (Multiple Choice)
            </td>
            <td>
                This quiz will check, if you have enough in depth
                knowledge with java.
            </td>
            <td>

                <p:button outcome="takeQuizMultipleChoice.xhtml" value="Take Quiz!"></p:button>
            </td>
        </tr>
    </table>
    <hr/>
    <p>
        If you want to create a quiz,
        <p:button outcome="createquiz.xhtml" value="click here!"></p:button>
    </p>
    <hr/>

```

```

    <br/><br/>
</ui:define>

<!-- </ui:composition -->

```

On the quizzes-site, all the different quizzes are displayed in a table format. By clicking on the buttons, the user can take the quiz.

takequiz.xhtml.

```

<!-- <ui:composition ... -->

<ui:define name="header">
    <title>Test-JavaQuiz</title>
    <h:outputStylesheet library="css" name="stylesheet.css"></
h:outputStylesheet>
</ui:define>

<ui:define name="content">
    <h1 align="center">Java Learning Quiz!</h1>
    <p style="margin-left: 20%; font-family: 'Arial', sans-serif;">
        Für jede richtig beantwortete Frage bekommst du 10 Punkte! <br/>
        Es ist immer nur eine Antwort richtig! <br/>
        Wenn du alle Fragen richtig beantwortest, bekommst du dieses
        Emblem:

    <h:graphicImage library="img" name="#{testQuizController.emblem}"></
h:graphicImage>
    </p>
    <div id="questions">
        <h:form>
            <ui:repeat var="quiz" value="#{testQuizController.quiz}">
                <div class="question">
                    <p:outputLabel for="radio" value="#{quiz.frage}"></
p:outputLabel>

                    <p:selectOneRadio id="radio" value="#{quiz.selectedAnswer}" unselectable="true" layout="table">

                        <f:selectItem itemValue="#{quiz.antworten[0]}" itemLabel="#{quiz.antworten[0]}"></
f:selectItem>

                        <f:selectItem itemValue="#{quiz.antworten[1]}" itemLabel="#{quiz.antworten[1]}"></
f:selectItem>

```



```

    <f:selectItem itemValue="#{quiz.antworten[2]}" itemLabel="#{quiz.antworten[2]}">
f:selectItem>

    <f:selectItem itemValue="#{quiz.antworten[3]}" itemLabel="#{quiz.antworten[3]}">
f:selectItem>
        </p:selectOneRadio>
        <br/>
    </div>
    <br/><br/>
</ui:repeat>
    <p:commandButton value="Check Answers" style="margin-
left: 20%" action="#{testQuizController.checkAnswersSingleChoice}"></
p:commandButton>
    </h:form>
</div>
</ui:define>

<!-- </ui:composition> -->

```

This is the page for taking singlechoice-quizzes. The questions are repeatedly display by the `<ui:repeat>` tag. This tag runs through a given list, and displays the wanted data.

takeQuizMultipleChoice.xhtml.

```

<!-- <ui:composition ... -->

<ui:define name="header">
    <title>Test - JavaQuiz</title>
    <h:outputStylesheet library="css" name="stylesheet.css"></
h:outputStylesheet>
</ui:define>

<ui:define name="content">
    <h1 align="center">Java Learning Quiz!</h1>
    <p style="margin-left: 20%; font-family: 'Arial', sans-serif;">
        Für jede richtig beantwortete Frage bekommst du 20 Punkte! <br/>
        Es können mehrere Antworten richtig sein! <br/>
        wenn du alle Fragen richtig beantwortest, bekommst du dieses
        Emblem:

    <h:graphicImage library="img" name="#{testQuizController.emblem}"></
h:graphicImage>
    </p>

```

```

<div id="questions">
  <h:form>
    <ui:repeat var="quiz" value="#{testQuizController.quiz}">
      <div class="question">
        <h:outputText value="#{quiz.frage}">/
h:outputText> <br />
        <table>
          <tr>
            <td>

            <h:outputText value="#{quiz.antworten[0]}">/h:outputText>
            </td>
            <td>

            <p:<selectBooleanCheckbox value="#{quiz.buttons[0]}">/>
p:selectBooleanCheckbox>
            </td>
          </tr>
          <tr>
            <td>

            <h:outputText value="#{quiz.antworten[1]}">/h:outputText>
            </td>
            <td>

            <p:<selectBooleanCheckbox value="#{quiz.buttons[1]}">/>
p:selectBooleanCheckbox>
            </td>
          </tr>
          <tr>
            <td>

            <h:outputText value="#{quiz.antworten[2]}">/h:outputText>
            </td>
            <td>

            <p:<selectBooleanCheckbox value="#{quiz.buttons[2]}">/>
p:selectBooleanCheckbox>
            </td>
          </tr>
          <tr>
            <td>

            <h:outputText value="#{quiz.antworten[3]}">/h:outputText>
            </td>
            <td>

```

```

    <p:selectBooleanCheckbox value="#{quiz.buttons[3]}"></
p:selectBooleanCheckbox>
        </td>
    </tr>
</table>
<br/><br/>
</div>
</ui:repeat>
<p:commandButton value="Check Answers" style="margin-left:
20%" action="#{testQuizController.checkAnswersMultipleChoice()}"></
p:commandButton>
    </h:form>
</div>
</ui:define>

<!-- </ui:composition> -->

```

On this page, multiplechoice quizzes are displayed. The questions are displayed in the same way, as mentioned above. The difference here is, that I had to create every button separately (<p:selectBooleanCheckBox>). Each of these buttons has to be bound to a *Boolean*-variable.

results.xhtml.

```

<!-- <ui:composition ... -->

<ui:define name="header">
    <title>Results</title>
    <h:outputStylesheet library="css" name="stylesheet.css"></
h:outputStylesheet>
</ui:define>

<ui:define name="content">
    <h1>Results</h1>
    <p style="font-size: 1.2em; margin-left:25%;">
        Du hast #{testQuizController.ricounter}/4 Fragen richtig
        beantwortet! <br/>
        Damit bekommst du #{testQuizController.ricounter*10}
        Punkte! <br/>
        Die von dir richtig/falsch beantworteten Fragen siehst du hier
        mit den richtigen Antworten:
    </p>
    <p>
        <ui:repeat var="result" value="#{testQuizController.results}">

```

```

<div id="results">

  <h:outputText value="#{result.frage}" style="#{result.istfalsch?
'color:red; font-weight: bold;' : 'color:green; font-weight:
bold;'}"></h:outputText>
    <i class="#{result.istfalsch? 'fa fa-fw fa-close' : 'fa
fa-fw fa-check'}"></i><br/>

  <h:outputText value="#{result.antworten[0]}" style="#{result.richtigeAntworten.get(0)
== 1? 'color:green' : 'color:red'}"></h:outputText> <br/>

  <h:outputText value="#{result.antworten[1]}" style="#{result.richtigeAntworten.get(1)
== 1? 'color:green' : 'color:red'}"></h:outputText> <br/>

  <h:outputText value="#{result.antworten[2]}" style="#{result.richtigeAntworten.get(2)
== 1? 'color:green' : 'color:red'}"></h:outputText> <br/>

  <h:outputText value="#{result.antworten[3]}" style="#{result.richtigeAntworten.get(3)
== 1? 'color:green' : 'color:red'}"></h:outputText> <br/>
</div>
</ui:repeat>
</p>
<h:outputText value="Diesen Score hast du jetzt:
#{testQuizController.score} Punkte"></h:outputText>
</ui:define>

<!-- </ui:composition -->

```

The Trainers page should display all the trainers associated with the GAME platform. The trainees can contact these trainers if they need help.

trainers.xhtml.

```

<!-- <ui:composition ...> -->

<ui:define name="header">
  <title>Trainers</title>
  <h:outputStylesheet library="css" name="stylesheet.css" />
</ui:define>

<ui:define name="content">
  <h1 align="center">Trainers</h1>
  <h:form id="trainerform">

```

```

    <p:carousel value="#{trainerController.trainers}" headerText="Trainers" var="tra
align:center" responsive="true"> ❶
        <p:panelGrid columns="2" style="width:100%;margin:10px
0px" columnClasses="label,value" layout="grid" styleClass="ui-
panelgrid-blank">
            <f:facet name="header">

                <p:graphicImage library="img" name="trainer.jpg"/>
                </f:facet>

                <h:outputText value="Name:" />
                <h:outputText value="#{trainer.name}" />

                <h:outputText value="Role:" />
                <h:outputText value="#{trainer.role}" />

                <h:outputText value="Branch:" />
                <h:outputText value="#{trainer.branch}" />
            </p:panelGrid>
        </p:carousel>
    </h:form>
</ui:define>

<!-- </ui:composition> -->

```

- ❶ Here we used a primefaces tag called `<p:carousel>`. This tag is used to create a carousel. The `<panelGrid>` tag is used to display data in a grid. The `<p:graphicImage>` is just like the JSF tag `<h:graphicImage>`. `<h:outputText>` is used to display text, with the function to call a Backing Bean. For easier explanation here is a picture: Bild einfügen

The leaderboard page is used to display all the trainees with their names, nicknames, branches and icons they got. You should be able to sort them by their names.

leaderboard.xhtml.

```

<!-- <ui composition ...> -->

<ui:define name="header">
    <h:outputStylesheet library="css" name="stylesheet.css" />
    <title>Emblemtafel</title>
</ui:define>

```

```

<ui:define name="content">
    <h1 align="center">Emblemtafel</h1>
    <p>
        Hier siehst du eine Auflistung aller Trainees, mit ihren
        Nicknames und ihren Fortschritt.
    </p>
    <p:dataTable var="trainee" value="#{traineeController.trainees}"> ❶
        <p:column headerText="Name">
            <h:outputText value="#{trainee.name}"></h:outputText>
        </p:column>

        <p:column headerText="Nickname">
            <h:outputText value="#{trainee.nickname}"></h:outputText>
        </p:column>

        <p:column headerText="Abteilung">
            <h:outputText value="#{trainee.abteilung}"></h:outputText>
        </p:column>

        <p:column headerText="Emblems">

            <h:graphicImage library="img" name="#{trainee.embleme.get(0)}"></
h:graphicImage>
            </p:column>
        </p:dataTable>
    </ui:define>

<!-- </ui:composition> -->

```

- ❶ Here, the `<p:dataTable>` tag is used, which is rendered as a `<table>` tag, with some style-modifications. If you give it a list, it will know to display all elements of the list.

13.5. Java Classes

Trainer Classes

TrainerController.java.

```

package org.game.trainee.trainerview;

import java.io.Serializable;
import java.util.List;
import javax.annotation.PostConstruct;

```

```
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

/**
 *
 * @author Eric
 */
@Named
@ViewScoped
public class TrainerController implements Serializable {

    private List<Trainer> trainers;

    private Trainer selectedTrainer;

    @Inject
    private TrainersService service;

    @PostConstruct
    public void init() {
        trainers = service.createTrainers(6);
    }

    public List<Trainer> getTrainers() {
        return trainers;
    }

    public void setService(TrainersService service) {
        this.service = service;
    }

    public Trainer getSelectedTrainer() {
        return selectedTrainer;
    }

    public void setSelectedTrainer(Trainer selectedTrainer) {
        this.selectedTrainer = selectedTrainer;
    }
}
```

The TrainerView-Class is used to display all the trainers on the trainers.xhtml page.

Trainee Classes

TraineeController.java.

```
package org.game.trainee.traineeview;

import java.io.Serializable;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

/**
 *
 * @author Eric
 */
@Named
@ViewScoped
public class TraineeController implements Serializable {
    private List<Trainee> trainees;
    private Trainee selectedTrainee;

    @Inject
    private TraineeGenerator generator;

    @PostConstruct
    public void init() {
        trainees=generator.createTrainees(4);
    }

    public List<Trainee> getTrainees() {
        return trainees;
    }

    public void setSelectedTrainee(Trainee selectedTrainee) {
        this.selectedTrainee = selectedTrainee;
    }

    public void setGenerator(TraineeGenerator generator) {
        this.generator = generator;
    }

    public Trainee getSelectedTrainee() {
        return selectedTrainee;
    }
}
```



```

    }

}

```

The TraineeView-Class is used to display all trainees on the leaderboard-page.

Quiz Classes

QuizController.

```

package org.game.trainee.testquiz;

import java.io.Serializable;
import java.util.ArrayList;
import javax.inject.Inject;
import javax.inject.Named;
import java.util.List;
import java.util.Map;
import javax.annotation.PostConstruct;
import javax.enterprise.context.RequestScoped;
import javax.faces.context.FacesContext;
import org.game.trainee.traineeview.TraineeGenerator;

/**
 *
 * @author Eric
 */
@Named
@RequestScoped
public class TestQuizController implements Serializable {

    private List<TestQuiz> quiz;
    private List<Results> results;
    private int score;
    private int ricounter;
    private String emblem = "javapro.png";
    private int qid;

    @Inject
    private TraineeGenerator trainee;

    @Inject
    private TestQuizSpeicher speicher;

```

```

@Inject
private QuizEJB quizbean; ❶

@PostConstruct ❷
public void init() {
    quiz = speicher.createQuiz(4, false);
    score = (int)trainee.getProgressFromIndex(0);
    results = new ArrayList<>();
    ricounter = 0;
}

public String checkAnswersSingleChoice() {
    evaluateScoreRadio();
    return "result.xhtml";
}

public String checkAnswersMultipleChoice() {
    evaluateScoreMultiple();
    return "result.xhtml";
}

public void evaluateScoreMultiple() { ❸
    List<Integer> falsche = new ArrayList<>();
    int richtige=0;
    for(int i=0; i<quiz.size(); i++ ) {
        List<Integer> indexrichtig =
umwandler(quiz.get(i).indexrichtig);
        for(int z=0; z<4; z++) {
            if(indexrichtig.get(z) == 1 && !quiz.get(i).buttons[z]
|| indexrichtig.get(z) == 0 && quiz.get(i).buttons[z]) {
                falsche.add(i);
                z=999;
            } else {
                richtige++;
            }
        }
        if(richtige==4) {
            score+=10;
            ricounter++;
            falsche.add(9999);
        }
        richtige = 0;
    }
    checkResults(falsche);
}

```

```

    }

    public void evaluateScoreRadio() { ④
        List<Integer> falsche = new ArrayList<>();
        for(int i=0; i<quiz.size(); i++) {

            if(quiz.get(i).selectedAnswer.equals(quiz.get(i).antworten[indexrichtig])
            {
                score+=10;
                ricounter++;
                falsche.add(9999);
            } else {
                falsche.add(i);
            }
        }
        checkResults(falsche);
    }

    public String quizUebergabe() {
        FacesContext fc = FacesContext.getCurrentInstance();
        Map<String, String> params =
        fc.getExternalContext().getRequestParameterMap();
        this.qid = Integer.parseInt(params.get("QID"));
        return "takequiz.xhtml";
    }

    public List<Integer> umwandler(int indexrichtig) {
        List<Integer> liste = new ArrayList<>();
        for(int i=0; i<4; i++)
            if(indexrichtig==i) {
                liste.add(1);
            } else {
                liste.add(0);
            }
        return liste;
    }

    public void checkResults(List<Integer> falsche) { ⑤
        for(int i=0; i<quiz.size(); i++) {
            List<Integer> indexrichtig =
            umwandler(quiz.get(i).indexrichtig);
            if(falsche.get(i) == i) {
                results.add(new Results(quiz.get(i).frage,
                quiz.get(i).antworten, indexrichtig, true));
            } else {

```

```

        results.add(new Results(quiz.get(i).frage,
quiz.get(i).antworten, indexrichtig, false));
    }
}

public String getQuizBezeichnung() {
    return quizbean.find(1).getBeschreibung();
}

//Getters and Setters
}

```

- ❶ Here, the QuizEJB is injected via the Java EE Dependency Injection System. This way, we can use everything from the injected Class, without the need of calling a constructor.
- ❷ The annotation *PostConstruct* functions as a note to the container, that this method must be run, before the construction of the Class.
- ❸ This method is used to evaluate the results of a MultipleChoice-Quiz. Each button is bound to a boolean-wert of the buttons[]. Every Question is checked, if every button matches the right answers. The user only gets points, if he answers the question correctly.
- ❹ Here, the singlechoice-quizzes get evaluated. This is much easier, because the radiobuttons function differently than the normal buttons. Every set of radiobuttons is bound to one value (selectedAnswer). We only need to check if the selected Answer matches the correct Answer. <5> In *checkResults*, a *Results-List* is generated. This list is used to display the results on the results.xhtml page.

The QuizController is used to handle everything surrounding the action of taking a quiz. It is responsible for displaying the content on the quizzes-, takequiz- and results-page. It will forward the user from the quizzes page to the takequizpage, where he/she can take the quiz. By clicking on the Submit button, the user is forwarded to the results-site, where their results are shown.

14. Theoretische Aufgabenstellung: Jan Binder

15. Knowledge management

15.1. Knowledge

16. Praktische Aufgabenstellung: Jan Binder

17. Back End: Datenbank und Datenbankverbindung

17.1. *Begin*

Zu aller erst wurden die allgemeinen Grundlagen, wie die Technologie, die verwendet wird, und das Design festgelegt. Auch die Datenbank, in der die entsprechenden Tabellen und Datensätze stehen soll, wurde vereinbart.

First of all the basics, like the technology we are using and the design are set. The database, in which all the tables and data sets are, is also appointed.

17.2. *Grundlagen*

Man braucht, um eine Datenbank und eine Datenbankverbindung zu erzeugen, ein gewisses Vorwissen. Hier werden SQL, JPA und Java von Nöten sein. Also wie das Programm mit der Datenbank kommuniziert und wie sogenannte Queries erstellt werden.

You need some foreknowledge to build a database and a database connectivity. Knowledge about SQL, JPA and Java is required. You need these languages to create queries or to understand how the program communicates with the database.

SQL

SQL is short for "Structured Query Language". It is used to insert, change and delete data sets or create or delete tables.

Beispielcode von SQL.

```
CREATE TABLE mitarbeiter (  
    id integer,  
    nachname text,  
    vorname text,  
    gehalt integer  
);  
  
INSERT INTO mitarbeiter (id, nachname, vorname, gehalt)
```

```
VALUES (1, Mustermann, Max, 2000);
```

```
SELECT * FROM mitarbeiter;
```

All this code does is it creates a new table called "mitarbeiter" which contains the columns "id", "nachname", "vorname" and "gehalt". Every column has its datatype next to it. Optionally there are constraints to the columns, such as "NOT NULL", which tells the database that this column cannot be empty or "UNIQUE", which means that a data set must be unique in this column. The next command is INSERT INTO. This code tells the database to write data into the table. First you need to set the table with all its columns and then the values like shown. Last there is the "SELECT * FROM mitarbeiter;" command which picks out every data set from the table "mitarbeiter". Further there is a possibility to only select a few specific data sets.

JPA

17.3. Getting Started

Entities

Kurs Entities

Kurs.java.

```
/**
 *
 * @author Jan
 */
@Entity
@Table(name = "kurs", schema = "game")
public class kurs implements Serializable {
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="KURSID")
    private int KursID;
    @NotNull
    @Column(name="TITEL")
    private String titel;
    @NotNull
```

```

@Column(name="LINK")
private URL link;
@Column(name="BESCHREIBUNG")
private String beschreibung;

//Getter, Setter

```

Here is the Kurs-Entity, it has the primary key "KursID", which helps to find a specific course from the table. The table also has a column "titel" which is basically the title of this course. It also has the constraint NOT NULL so if there is a new course there must be a title as well. The next column is the "link" it contains the links of the courses so the visitors can be redirected to the page with the explanation of the topic. This cannot be empty too. The last column is "beschreibung", which is the description of the course. This column can be empty, but it is not recommended since it helps the user specify which course contains what information.

KursBesuch.java.

```

/**
 *
 * @author Jan
 */

@Entity
@Table(name = "kursbesuch", schema = "game")
public class KursBesuch implements Serializable {
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="KURSBESUCHID")
    private int KursBesuchID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="MitID")
    @NotNull
    private Trainee trainee;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="KursID")
    @NotNull
    private Kurs kurs;
    @NotNull
    @Column(name="DATUM")
    private Date datum;

    //Getter, Setter

```


The entity "KursBesuch" is a table, which splits the many-to-many relationship between "trainee" and "kurs". That means it has the PK of both tables as a foreign key and it also provides a column named "datum" which indicates the date the user has visited the course.

Voraussetzung Entities

Voraussetzung.java.

```
/**
 *
 * @author Jan
 */

@Entity
@Table(name="voraussetzung", schema = "game")
public class Voraussetzung implements Serializable {
    @Id @GeneratedValue(strategy =
    GenerationType.SEQUENCE) @Column(name="VORAUSSETZID")
    private int VoraussetzID;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="KursID")
    private Kurs kurs;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="QID")
    private Quiz quiz;

    //Getter, Setter
}
```

KursVoraussetzung.java.

```
/**
 *
 * @author Jan
 */

@Entity
@Table(name = "kursvoraussetzung", schema = "game")
public class KursVoraussetzung implements Serializable {
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="KURSVORAUSSETZID")
    private int KursVoraussetzID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="KursID")
    private Kurs kurs;
}
```

```

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name="VoraussetzID")
private Voraussetzung voraussetzung;

//Getter, Setter

```

The function of "KursVoraussetzung" is that a course can have a course or a quiz as a requirement to take this course. This table contains courses and their required quizzes or courses.

"KursVoraussetzung" is another entity which splits up a many-to-many relationship. This time it is between "kurs" and "voraussetzung". Since there is no column other than the two foreign keys and the primary key "KursVoraussetzID" this table has no function except as a middle table for "kurs" and "voraussetzung".

QuizVoraussetzung.java.

```

/**
 *
 * @author Jan
 */
@Entity
@Table(name = "quizvoraussetzung", schema = "game")
public class QuizVoraussetzung implements Serializable{
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="QUIZVORAUSETZID")
    private int QuizVoraussetzID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="QID")
    private Quiz quiz;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="VoraussetzID")
    private Voraussetzung voraussetzung;

    //Getter, Setter

```

This table has the same function as "KursVoraussetzung", but instead of courses with quizzes. So this table contains quizzes and their required courses or quizzes.

Quiz Entities

Quiz.java.

```

/**
 *
 * @author Jan
 */
@NamedQuery(name = Quiz.QUERY_FINDBY_BESCHREIBUNG, query = "SELECT quiz
    FROM Quiz quiz WHERE quiz.beschreibung like :beschreibung")
@Entity
@Table(name = "quiz", schema = "game")
public class Quiz implements Serializable {
    public static final String
    QUERY_FINDBY_BESCHREIBUNG="Quiz.findByBeschreibung";
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int QID;
    @NotNull @Column(name = "TITEL")
    private String titel;
    @Column(name = "BESCHREIBUNG")
    private String beschreibung;
    @NotNull @Column(name = "REWARD")
    private Blob reward;

    //Getter, Setter

```

This is the table for all quizzes. It contains the "QID", which is the primary key, a title, a description and a reward. The title has a constraint called NOT NULL because every quiz must have a specification which topic it has, since the description is optional the title is the only differentiation between the quizzes.

Quizbeantwortung.java.

```

/**
 *
 * @author Jan
 */
@Entity
@Table(name = "quizbeantwortung", schema = "game")
public class Quizbeantwortung implements Serializable {
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int QBEID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="QID")
    private Quiz quiz;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="MitID")

```

```

private Trainee trainee;
@NotNull
@Column(name="ERREICHTEPUNKTE")
private int erreichtePunkte;

//Getter, Setter

```

Frage.java.

```

/**
 *
 * @author Jan
 */
@Entity
@Table(name = "frage", schema = "game")
public class Frage implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private int FID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="QID")
    private Quiz quiz;
    @NotNull
    @Column(name="FRAGE")
    private String Frage;
    @NotNull
    @Column(name="PUNKTEZAHL")
    private int punktezahl;

    //Getter, Setter

```

Every Quiz contains of many questions. These questions are stored in the table "Frage". Each question has a "FID", a "QID", so it can be ... to a quiz. A question also has the question itself, set points you get for each question.

Antwortmoeglichkeiten.java.

```

/**
 *
 * @author Jan
 */

```

```

@Entity
@Table(name = "antwortmoeglichkeiten", schema = "game")
public class Antwortmoeglichkeiten implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="ANTWID")
    private int AntwID;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="FID")
    private Frage frage;
    @NotNull
    @Column(name="ANTWORT")
    private String antwort;
    @Column(name="RICHTIGEANTWORT")
    private boolean richtigeAntwort;

    //Getter, Setter

```

Trainee Entity

Trainee.java.

```

/**
 *
 * @author Jan
 */

@Entity
@Table(name = "trainee", schema = "game")
public class Trainee implements Serializable {
    @Id @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name="MITID")
    private int MitID;
    @NotNull @Column(name="VORNAME")
    private String vorname;
    @NotNull @Column(name="NACHNAME")
    private String nachname;
    @NotNull @Column(unique = true, name="NICKNAME")
    private String nickname;
    @Column(name="ABTEILUNG")
    private String abteilung;
    private double progress;
    public List<String> embleme;

```

```
public String name;  
  
//Getter, Setter
```

EJBs

KursEJB.java.

```
/**  
 *  
 * @author Jan  
 */  
@Stateless  
public class KursEJB {  
    @PersistenceContext(unitName = "Diplomarbeit")  
    private EntityManager em;  
  
    public Kurs find(int KursID) {  
        return em.find(Kurs.class, KursID);  
    }  
  
    public void update(Kurs kurs) {  
        em.merge(kurs);  
    }  
  
    public void delete(int KursID) {  
        em.getTransaction().begin();  
        Kurs k = em.getReference(Kurs.class, KursID);  
        em.remove(k);  
        em.getTransaction().commit();  
    }  
}
```

FrageEJB.java.

```
/**  
 *  
 * @author Jan  
 */  
@Stateless  
public class FrageEJB {  
    @PersistenceContext(unitName = "Diplomarbeit")  
    private EntityManager em;
```

```

    public Frage find(int FID) {
        return em.find(Frage.class, FID);
    }

    public void update(Frage f) {
        em.merge(f);
    }

    public void delete(int FID) {
        em.getTransaction().begin();
        Frage f = em.getReference(Frage.class, FID);
        em.remove(f);
        em.getTransaction().commit();
    }
    /*
    public List<Frage> findQuizFragen(int index) {
        Query query = em.createQuery("SELECT * FROM frage f WHERE QID
        =" + index + ";");
        Collection<Frage> collection;
        collection = (Collection<Frage>) query.getResultList();
        ArrayList<Frage> newList =
        collection.stream().collect(toCollection(ArrayList::new));
        return newList;
    }
    */
}

```

QuizEJB.java.

```

/**
 *
 * @author Jan
 */

@Stateless
public class QuizEJB {
    @PersistenceContext(unitName = "Diplomarbeit")
    private EntityManager em;

    public Quiz find(int QID) {
        return em.find(Quiz.class, QID);
    }

    public void update(Quiz q) {

```

```

        em.merge(q);
    }

    public void delete(int QID) {
        em.getTransaction().begin();
        Quiz q = em.getReference(Quiz.class, QID);
        em.remove(q);
        em.getTransaction().commit();
    }

    public List<Quiz> findByBeschreibung(String beschreibung) {
        return em.createNamedQuery(Quiz.QUERY_FINDBY_BESCHREIBUNG,
Quiz.class)
                .setParameter("beschreibung", "%" + beschreibung + "%")
                .getResultList();
    }
}

```

TraineeEJB.java.

```

/**
 *
 * @author Jan
 */
@Stateless
public class TraineeEJB {
    @PersistenceContext(unitName = "Diplomarbeit")
    private EntityManager em;

    public Trainee find(int MitID) {
        return em.find(Trainee.class, MitID);
    }

    public void update(Trainee t) {
        em.merge(t);
    }

    public void delete(int MitID) {
        em.getTransaction().begin();
        Trainee t = em.getReference(Trainee.class, MitID);
        em.remove(t);
        em.getTransaction().commit();
    }
}

```


Persistence.xml

persistence.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2" xmlns="http://java.sun.com/xml/ns/
persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://
xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
  <!-- Define Persistence Unit -->
  <persistence-unit name="Diplomarbeit" transaction-type="JTA">
    <jta-data-source>jdbc/postgrespool</jta-data-source>
    <class>org.game.trainee.testquiz.Frage</class>
    <class>org.game.trainee.testquiz.Antwortmoeglichkeiten</class>
    <class>org.game.trainee.kurs.Kurs</class>
    <class>org.game.trainee.kurs.KursBesuch</class>
    <class>org.game.trainee.testquiz.Quizbeantwortung</class>
    <class>org.game.trainee.testquiz.Quiz</class>
    <class>org.game.trainee.traineeview.Trainee</class>
    <class>org.game.trainee.testquiz.Antwort</class>
    <class>org.game.trainee.testquiz.QuizVoraussetzung</class>
    <class>org.game.trainee.kurs.Voraussetzung</class>
    <class>org.game.trainee.kurs.KursVoraussetzung</class>
    <properties>
      <property name="javax.persistence.schema-
generation.database.action" value="drop-and-create"/>
      <property name="javax.persistence.schema-generation.create-source"
value="script"/>
      <property name="javax.persistence.schema-generation.drop-source"
value="script"/>
      <property name="javax.persistence.schema-generation.drop-script-
source" value="META-INF/dropSeqSQL.sql"/>
      <property name="javax.persistence.schema-generation.create-script-
source" value="META-INF/createSeqSQL.sql" />
      <property name="javax.persistence.schema-generation.drop-script-
source" value="META-INF/dropSQL.sql"/>
      <property name="javax.persistence.schema-generation.create-script-
source" value="META-INF/createSQL.sql"/>
      <property name="javax.persistence.sql-load-script-source"
value="META-INF/alterTableSQL.sql"/>
      <property name="javax.persistence.sql-load-script-source"
value="META-INF/insertSQL.sql"/>

    </properties>
```

```
</persistence-unit>  
</persistence>
```

The "persistence.xml" is used to configure many things, such as the source of the script used for dropping, if the database should always drop and then create all tables, and much more.

SQL-Files

In this chapter I will introduce all SQL-files used in this project in the order they are when the program is started.

dropSeqSQL.sql.

```
DROP SEQUENCE game."antwortmoeglichkeiten_ANTWID_seq" CASCADE;  
  
DROP SEQUENCE game."frage_FNR_seq" CASCADE;  
  
DROP SEQUENCE game."kurs_KURSID_seq" CASCADE;  
  
DROP SEQUENCE game."kursbesuch_KURSBESUCHID_seq" CASCADE;  
  
DROP SEQUENCE game."kursvoraussetzung_KURSVORAUSETZID_seq" CASCADE;  
  
DROP SEQUENCE game."quiz_QID_seq" CASCADE;  
  
DROP SEQUENCE game."quizbeantwortung_QBEID_seq" CASCADE;  
  
DROP SEQUENCE game."quizvoraussetzung_QUIZVORAUSETZID_seq" CASCADE;  
  
DROP SEQUENCE game."trainee_MITID_seq" CASCADE;
```

```
DROP SEQUENCE game."voraussetzung_VORAUSETZID_seq" CASCADE;
```

Before everything is created, you need to make sure that there is no existing sequence, because otherwise there would be errors when creating the sequences. This script drops every sequence at the start.

dropSQL.sql.

```
DROP TABLE game.antwortmoeglichkeiten;  
  
DROP TABLE game.frage;  
  
DROP TABLE game.kurs CASCADE;  
  
DROP TABLE game.kursbesuch;  
  
DROP TABLE game.kursvoraussetzung;  
  
DROP TABLE game.quiz CASCADE;  
  
DROP TABLE game.quizbeantwortung;  
  
DROP TABLE game.quizvoraussetzung;  
  
DROP TABLE game.trainee;  
  
DROP TABLE game.voraussetzung;
```

This script has the function of dropping every table before creating so there are no errors.

createSeqSQL.sql.

```
CREATE SCHEMA IF NOT EXISTS "game" AUTHORIZATION darbeit;  
  
CREATE SEQUENCE game."antwortmoeglichkeiten_ANTWID_seq"  
    INCREMENT 1  
    START 1  
    MINVALUE 1  
    MAXVALUE 9223372036854775807
```

```
CACHE 1;
```

This SQL-file is used to create all the sequences for the primary keys of the tables, so it automatically goes to the next number whenever a new data row is created. I only included one sequence because it repeats for every table.

createSQL.sql.

```
CREATE SCHEMA IF NOT EXISTS game AUTHORIZATION arbeit;

CREATE TABLE game.antwortmoeglichkeiten
(
    "ANTWID" bigint NOT NULL DEFAULT
    nextval('game."antwortmoeglichkeiten_ANTWID_seq"::regclass),
    "FNR" bigint NOT NULL,
    "ANTWORT" text COLLATE pg_catalog."default" NOT NULL,
    "RICHTIGEANTWORT" boolean
);

CREATE TABLE game.frage
(
    "FNR" bigint NOT NULL DEFAULT
    nextval('game."frage_FNR_seq"::regclass),
    "QID" bigint NOT NULL,
    "FRAGE" text COLLATE pg_catalog."default" NOT NULL,
    "PUNKTEZAHL" bigint NOT NULL
);

CREATE TABLE game.kurs
(
    "KURSID" bigint NOT NULL DEFAULT
    nextval('game."kurs_KURSID_seq"::regclass),
    "TITEL" text COLLATE pg_catalog."default" NOT NULL,
    "LINK" text COLLATE pg_catalog."default" NOT NULL,
    "BESCHREIBUNG" text COLLATE pg_catalog."default"
);

CREATE TABLE game.kursbesuch
(
    "KURSBESUCHID" bigint NOT NULL DEFAULT
    nextval('game."kursbesuch_KURSBESUCHID_seq"::regclass),
    "KURSID" bigint NOT NULL,
    "MITID" bigint NOT NULL,
    "DATUM" date NOT NULL
);
```

```
CREATE TABLE game.kursvoraussetzung
(
    "KURSVORAUSETZID" bigint NOT NULL DEFAULT
    nextval('game."kursvoraussetzung_KURSVORAUSETZID_seq"::regclass),
    "VORAUSETZID" bigint NOT NULL,
    "KURSID" bigint NOT NULL
);

CREATE TABLE game.quiz
(
    "QID" bigint NOT NULL DEFAULT
    nextval('game."quiz_QID_seq"::regclass),
    "TITEL" text COLLATE pg_catalog."default" NOT NULL,
    "BESCHREIBUNG" text COLLATE pg_catalog."default",
    "REWARD" text COLLATE pg_catalog."default" NOT NULL
);

CREATE TABLE game.quizbeantwortung
(
    "QBEID" bigint NOT NULL DEFAULT
    nextval('game."quizbeantwortung_QBEID_seq"::regclass),
    "QID" bigint NOT NULL,
    "MITID" bigint NOT NULL,
    "ERREICHTEPUNKTEZAHL" bigint NOT NULL
);

CREATE TABLE game.quizvoraussetzung
(
    "QUIZVORAUSETZID" bigint NOT NULL DEFAULT
    nextval('game."quizvoraussetzung QUIZVORAUSETZID_seq"::regclass),
    "VORAUSETZID" bigint NOT NULL,
    "QID" bigint NOT NULL
);

CREATE TABLE game.trainee
(
    "MITID" bigint NOT NULL DEFAULT
    nextval('game."trainee_MITID_seq"::regclass),
    "NAME" text COLLATE pg_catalog."default" NOT NULL,
    "VORNAME" text COLLATE pg_catalog."default",
    "NICKNAME" text COLLATE pg_catalog."default" NOT NULL,
    "ABTEILUNG" text COLLATE pg_catalog."default"
);

CREATE TABLE game.voraussetzung
```

```
(
    "VORAUSSETZID" bigint NOT NULL DEFAULT
    nextval('game."voraussetzung_VORAUSSETZID_seq"::regclass),
    "KURSID" bigint,
    "QID" bigint
);
```

This file is used to create all the tables we are using. It produces every table with its associated columns.

alterTableSQL.sql.

```
CREATE SCHEMA IF NOT EXISTS game AUTHORIZATION arbeit;

ALTER TABLE game.antwortmoeglichkeiten ADD CONSTRAINT
    "ANTWORTMOEGLICHKEITEN_pkey" PRIMARY KEY ("ANTWID");

ALTER TABLE game.frage ADD CONSTRAINT "FRAGE_pkey" PRIMARY KEY ("FNR");

ALTER TABLE game.kurs ADD CONSTRAINT "KURS_pkey" PRIMARY KEY ("KURSID");

ALTER TABLE game.kursbesuch ADD CONSTRAINT "KURSBESUCH_pkey" PRIMARY KEY
    ("KURSBESUCHID");

ALTER TABLE game.kursvoraussetzung ADD CONSTRAINT
    "KURSVORAUSSETZUNG_pkey" PRIMARY KEY ("KURSVORAUSSETZID");

ALTER TABLE game.quiz ADD CONSTRAINT "QUIZ_pkey" PRIMARY KEY ("QID");

ALTER TABLE game.quizbeantwortung ADD CONSTRAINT "QUIZBEANTWORTUNG_pkey"
    PRIMARY KEY ("QBEID");

ALTER TABLE game.quizvoraussetzung ADD CONSTRAINT
    "QUIZVORAUSSETZUNG_pkey" PRIMARY KEY ("QUIZVORAUSSETZID");

ALTER TABLE game.trainee ADD CONSTRAINT "TRAINEE_pkey" PRIMARY KEY
    ("MITID");

ALTER TABLE game.trainee ADD CONSTRAINT "NICKNAME" UNIQUE ("NICKNAME");

ALTER TABLE game.voraussetzung ADD CONSTRAINT "VORAUSSETZUNG_pkey"
    PRIMARY KEY ("VORAUSSETZID");
```

```
ALTER TABLE game.antwortmoeglichkeiten ADD CONSTRAINT "Frage" FOREIGN  
KEY ("FNR")  
    REFERENCES game.frage ("FNR") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.frage ADD CONSTRAINT "Quiz" FOREIGN KEY ("QID")  
    REFERENCES game.quiz ("QID") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.kursbesuch ADD CONSTRAINT "kurs" FOREIGN KEY ("KURSID")  
    REFERENCES game.kurs ("KURSID") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.kursbesuch ADD CONSTRAINT "Trainee" FOREIGN KEY  
("MITID")  
    REFERENCES game.trainee ("MITID") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.kursvoraussetzung ADD CONSTRAINT "kurs" FOREIGN KEY  
("KURSID")  
    REFERENCES game.kurs ("KURSID") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.kursvoraussetzung ADD CONSTRAINT "Voraussetzung"  
FOREIGN KEY ("VORAUSSETZID")  
    REFERENCES game.voraussetzung ("VORAUSSETZID") MATCH SIMPLE  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
    NOT VALID;
```

```
ALTER TABLE game.quizbeantwortung ADD CONSTRAINT "Quiz" FOREIGN KEY
("QID")
    REFERENCES game.quiz ("QID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;
```

```
ALTER TABLE game.quizbeantwortung ADD CONSTRAINT "Trainee" FOREIGN KEY
("MITID")
    REFERENCES game.trainee ("MITID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;
```

```
ALTER TABLE game.quizvoraussetzung ADD CONSTRAINT "Quiz" FOREIGN KEY
("QID")
    REFERENCES game.quiz ("QID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;
```

```
ALTER TABLE game.quizvoraussetzung ADD CONSTRAINT "Voraussetzung"
FOREIGN KEY ("VORAUSSETZID")
    REFERENCES game.voraussetzung ("VORAUSSETZID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;
```

```
ALTER TABLE game.voraussetzung ADD CONSTRAINT "Kurs" FOREIGN KEY
("KURSID")
    REFERENCES game.kurs ("KURSID") MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
    NOT VALID;
```

```
ALTER TABLE game.voraussetzung ADD CONSTRAINT "Quiz" FOREIGN KEY ("QID")
```



```
REFERENCES game.quiz ("QID") MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE CASCADE
NOT VALID;
```

This file is used to give all the tables its associated primary keys and foreign keys. This is not done in the "createSQL" file, because it would generate errors.

insertSQL.sql.

```
INSERT INTO game.trainee ("MITID", "NAME", "VORNAME", "NICKNAME",
    "ABTEILUNG") VALUES (2, 'Haneder', 'Eric', 'ericbensi', NULL);
INSERT INTO game.trainee ("MITID", "NAME", "VORNAME", "NICKNAME",
    "ABTEILUNG") VALUES (1, 'Binder', 'Jan', 'Syreax', NULL);
INSERT INTO game.trainee ("MITID", "NAME", "VORNAME", "NICKNAME",
    "ABTEILUNG") VALUES (3, 'Saliger', 'Alex', 'SaAlex_1010', NULL);
INSERT INTO game.trainee ("MITID", "NAME", "VORNAME", "NICKNAME",
    "ABTEILUNG") VALUES (4, 'wurst', 'Alexander', 'wursti', NULL);

INSERT INTO game.quiz ("QID", "TITEL", "BESCHREIBUNG", "REWARD") VALUES
    (1, 'Start', 'Quiz about the general knowledge of Java', 'test');
INSERT INTO game.quiz ("QID", "TITEL", "BESCHREIBUNG", "REWARD") VALUES
    (2, 'Object, Classes & Constructors', 'Quiz about Objects, Classes and
    Constructors in Java', 'test');

INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (1,
    1, 'what are advantages of Java?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (2,
    1, 'what is written after a line of code?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (9,
    1, 'what does Object Oriented mean?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (10,
    1, 'which Java Version exist', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (11,
    2, 'what concept of the following is Java supporting?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (12,
    2, 'which things do objects include?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (13,
    2, 'what is not a variabletype which classes can contain?', 10);
INSERT INTO game.frage ("FNR", "QID", "FRAGE", "PUNKTEZAHL") VALUES (14,
    2, 'what is the main rule of a constructor?', 10);
```

```

INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (1, 1, 'Flawless', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (2, 1, 'Platform Independent', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (3, 1, 'Only compatible with windows',
false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (4, 1, 'only compatible with Linux', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (5, 2, '"."', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (6, 2, '"{ or }"', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (7, 2, '";"', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (8, 2, 'Java RE', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (9, 9, 'Object is a addon for Java', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (10, 9, 'You have to use the program Object
to run Java.', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (11, 9, 'Java can be extended with Objects.',
true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (12, 9, 'Java has no Objects', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (13, 10, 'Java S', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (14, 10, 'Java E', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (15, 10, 'Java SS', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (16, 10, 'Java SE', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (17, 11, 'Multiple Inheritance', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (18, 11, 'Classes&Methods', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (19, 11, 'Functions', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (20, 11, 'None of the above', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
"RICHTIGEANTWORT") VALUES (21, 12, 'Classes', false);

```

```

INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (22, 12, 'only state', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (23, 12, 'only behaviour', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (24, 12, 'state and behaviour', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (25, 13, 'Global', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (26, 13, 'Local', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (27, 13, 'Class', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (28, 13, 'Instance', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (29, 14, 'Different name as Class', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (30, 14, 'Same name as attribute', false);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (31, 14, 'Same name as Class', true);
INSERT INTO game.antwortmoeglichkeiten ("ANTWID", "FNR", "ANTWORT",
    "RICHTIGEANTWORT") VALUES (32, 14, 'Must contain attributes', false);

INSERT INTO game.kurs ("KURSID", "TITEL", "LINK", "BESCHREIBUNG") VALUES
    (1, 'Start-Kurs', 'https://www.tutorialspoint.com/java/index.htm',
    NULL);
INSERT INTO game.kurs ("KURSID", "TITEL", "LINK", "BESCHREIBUNG")
    VALUES (2, 'Object & Classes', 'https://www.tutorialspoint.com/java/
    java_object_classes.htm', NULL);
INSERT INTO game.kurs ("KURSID", "TITEL", "LINK", "BESCHREIBUNG")
    VALUES (3, 'Constructors', 'https://www.tutorialspoint.com/java/
    java_constructors.htm', NULL);

```

At last the "insertSQL" script is used to insert all the data we want into the belonging table. the order of the commands is very important because, if you insert "antwortmoeglichkeiten" at first there would be no existing matching "FNR" or with "frage " no fitting "QID".

