IZIS - Ilias Zusatzmodule der Informatik an der Uni Stuttgart

Dokumentation des Plugins TestOverview

TestOverview
Jan Ruthardt, Jonathan Schuder, Martin Dinkel, Benedict Steuerlein

SS16-WS16/17

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument ist im Rahmen des Studienprojektes
IZIS - *ILIAS Zusatzmodule der Informatik an der Universität Stuttgart*
entstanden. Es soll dem Leser einen allgemeinen, wie auch technischen Überblick über das weiterentwickelte ILIAS-Plugin **TestOverview** geben.

1.1 Aufbau des Dokuments

Über die grundlegende Verwendung des Plugins, d.h. Installation und Anwendungsmöglichkeiten wird im **ReadMe** File auf Github referiert. Einen genaueren Einblick in technische Details wird in diesem Dokument in den Kapiteln 2 und 3 gegeben. Dort werden sowohl Funktionsweise, Modellierung der Datenbank und Sequenzdiagramme der wichtigsten Funktionen des Plugins TestOverview gezeigt und erläutert.

1.2 Bedarf

Die gleichzeitige Verwendung mehrerer Lernplattformen führt in der Regel zu Verwirrung und Unübersichtlichkeit.
Aus diesem Grund sollte das eClaus-Lernsystem der Informatik Fakultät der Universität Stuttgart von der Lernplattform ILIAS abgelöst werden. In einer Analyse der bestehenden Funktionen des eClaus-Systems wurde festgestellt, dass es in ILIAS bis jetzt keine Möglichkeit gibt gesamte Kursstatistiken über MC-Tests und Exercises schnell und effizient einzusehen.
Neben der Analyse der Funktionen von eClaus, zeigte die Untersuchung der Funktionen von ILIAS, dass das Plugin TestOverview bereits auf den Servern der Universität Stuttgart lief, jedoch nur eine tabellarische Übersicht über MC-Tests lieferte.
Eine Übersicht über Exercises wurde bis dahin von ILIAS noch nicht bereitgestellt. Das Erstellen einer Scheinliste musste in ILIAS aufwändig von Übungsbetreuern per Hand gemacht werden, bzw. von den Studenten selbst ausgerechnet werden.

1.3 Kunden des Plugins

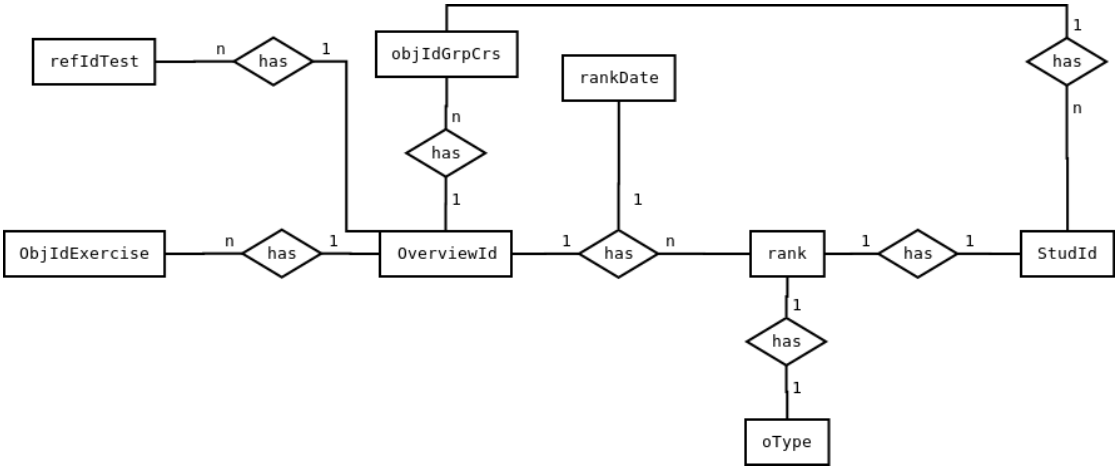
Das Plugin wurde 2013 im Auftrag der Universität Stuttgart von Databay entwickelt und von September 2016 bis April 2017 im Auftrag des Instituts für Formale Methoden der Informatik[1] an der Universität Stuttgart von Studenten weiterentwickelt.

[1] `www.fmi.uni-stuttgart.de`

2 Modellierung

2.1 ER - Datenbank

Bei der Installation des Plugins werden zusätzlich 7 Tabellen, in der Datenbank die für die ILIAS Installation genutzt wird, angelegt. Die Zusammenhänge werden durch folgendes ER Diagramm gezeigt.



Die hinzugefügten Tabellen haben folgenden Inhalt:

- **rep_robj_xtov_e2o** : Verknüpft Overview Ids mit obj_id der hinzugefügten Exercises
- **rep_robj_xtov_t2o** : Verknüpft Overview ID's mit obj_ids der hinzugefügten Tests
- **rep_robj_xtov_p2o** : Vernknüpft Overview IDs mit den ausgewählten Gruppen ID's
- **rep_robj_xtov_overview** : Speichert die ID aller Overview Objekte
- **rep_robj_xtov_eorank** : Verknüpft student_id und overview_id mit der Gesamtpunktzahl an erreichten Punkten des Studenten in Exercises
- **rep_robj_xtov_torank** : Verknüpft student_id und overview_id mit dem durchschnittlichen Prozentsatz aller Test die der Student absolviert hat
- **rep_robj_xtov_rankdate** : Speichert das Datum des Ranks

2.2 Sequenzdiagramme

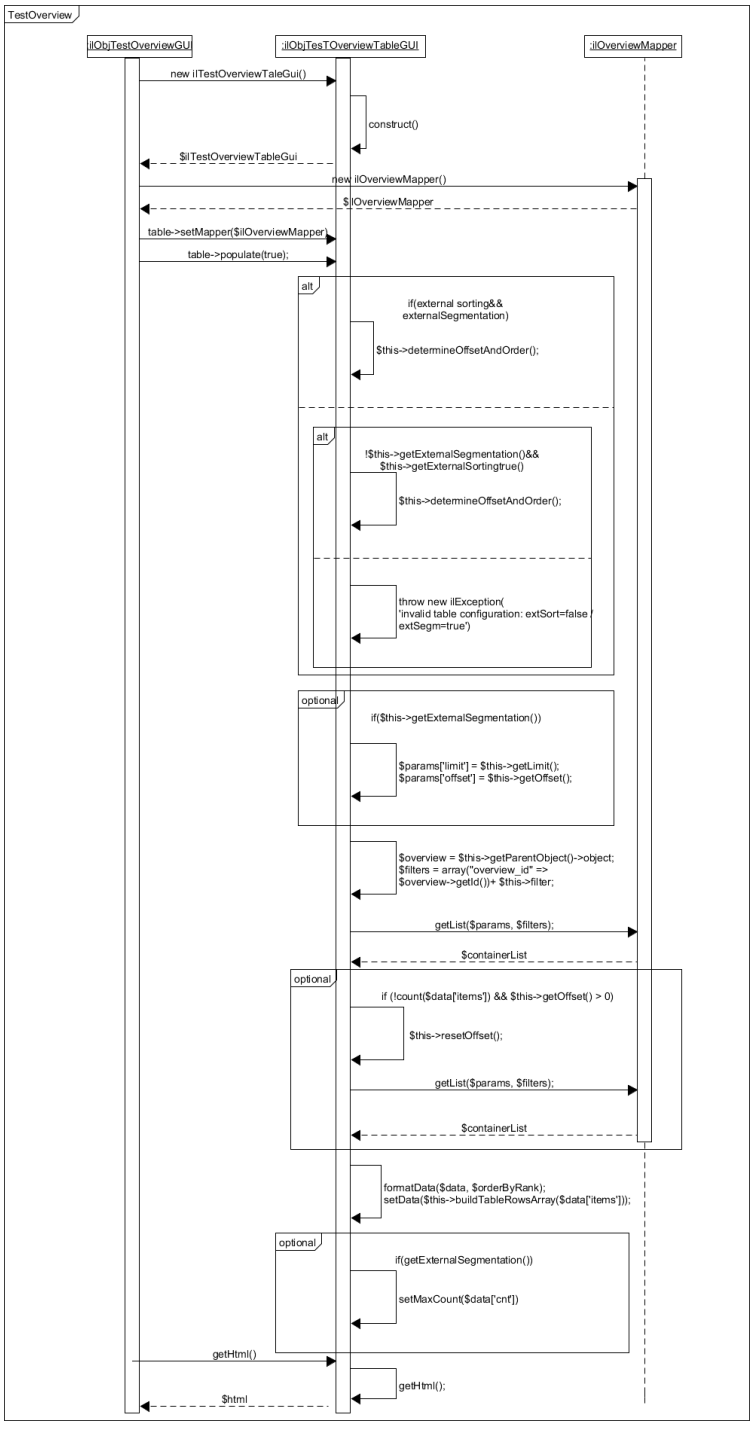


Abbildung 2.0: TestOverview Testübersicht 3.4.1

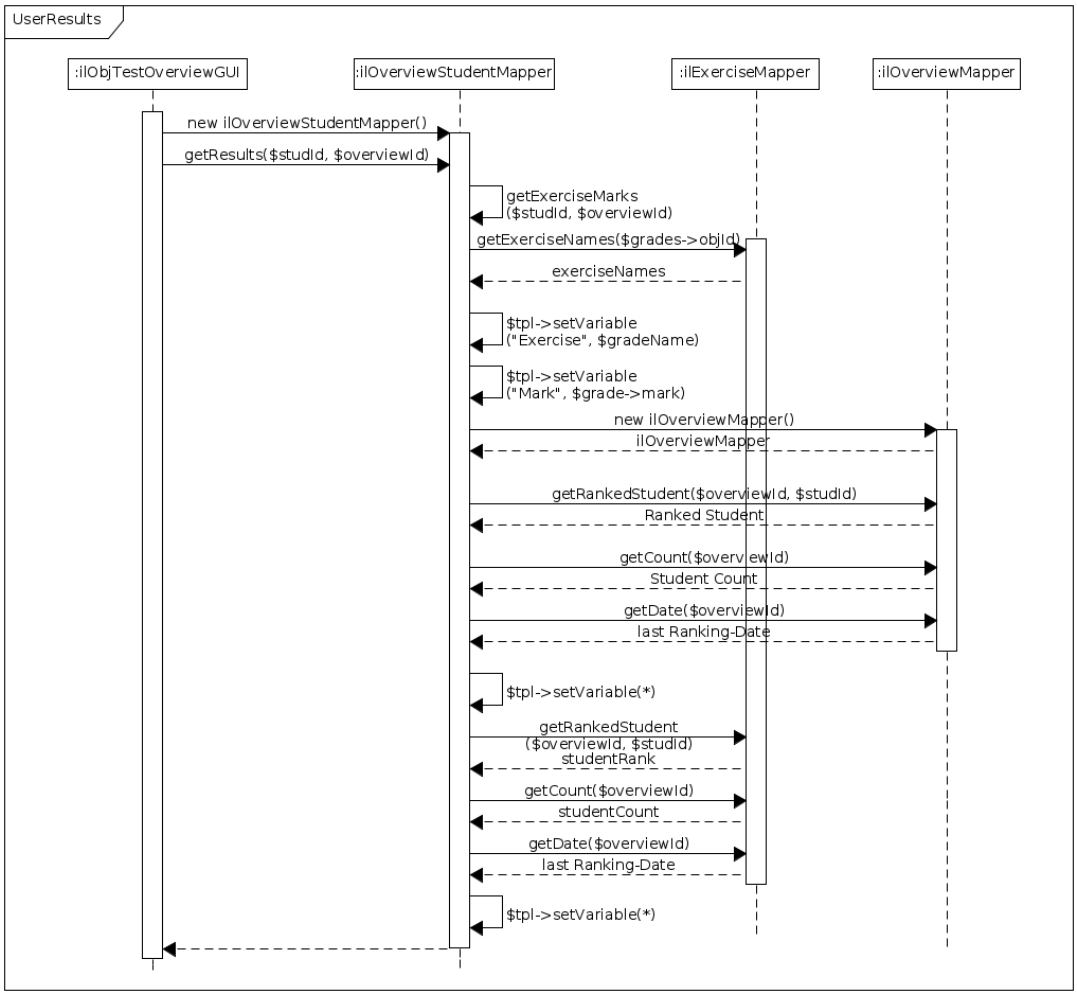


Abbildung 2.1: User Results Sequenzdiagramm, 3.4.2

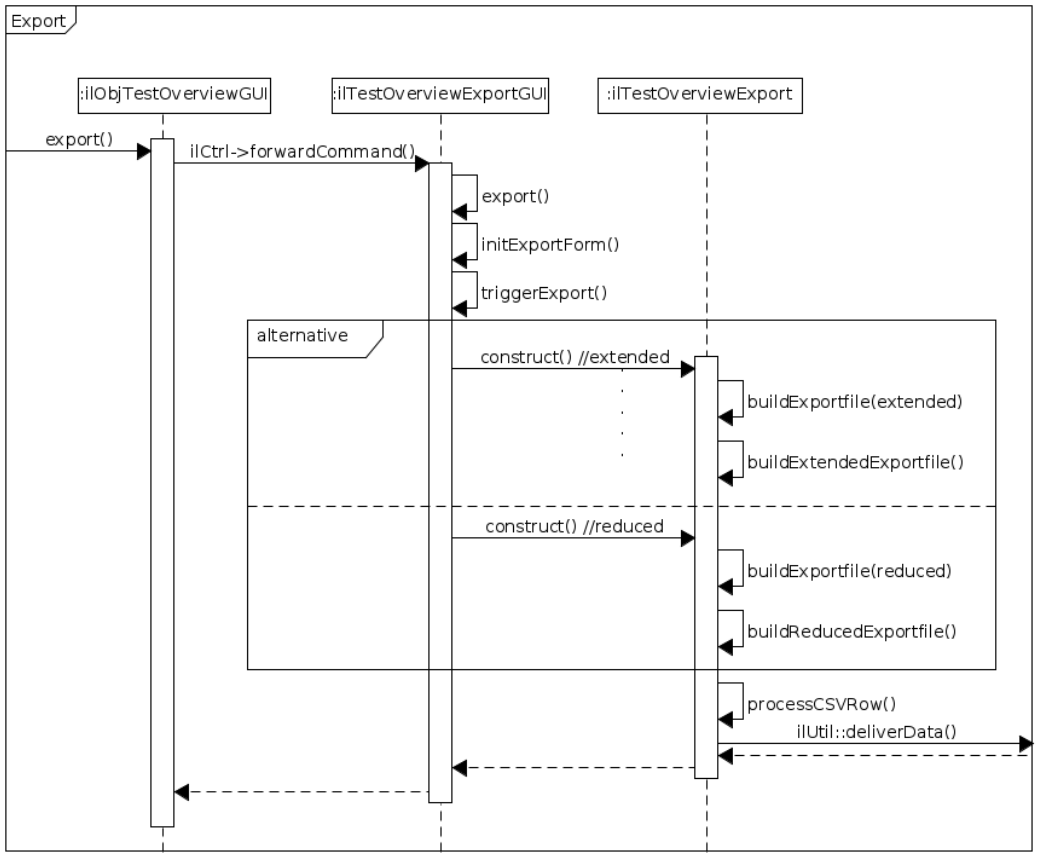


Abbildung 2.2: Export Sequenzdiagramm, 3.4.3

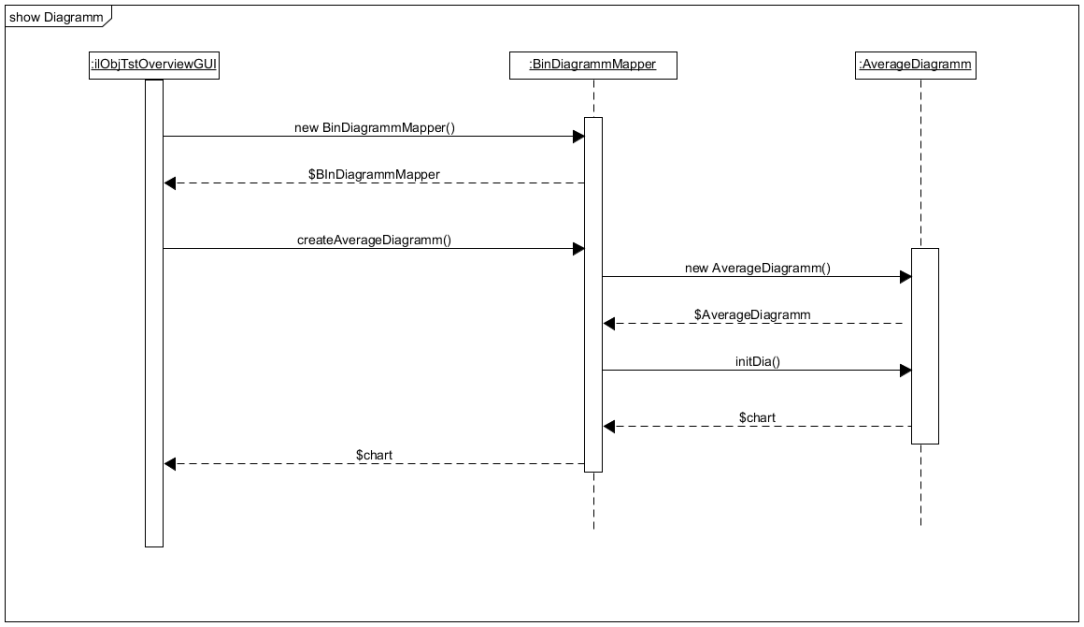


Abbildung 2.3: Anzeige des Diagramms, 3.4.4

3 Funktionsweise

3.1 Empfehlungen für die Benutzung

Um den Übungsbetrieb und die Kursstruktur in ILIAS besser zu organisieren und gestalten, empfehlen wir die Verwendung des parallel entwickelten Plugins *Admin Container Object*. Der Quellcode ist hier auf GitHub zu finden.
Das Plugin erlaubt es, die Kursstruktur automatisiert zu erstellen. Dazu zählt das:

- gleichzeitige Anlegen von mehreren Gruppen und Verwalten der Einstellungen
- Verschieben von Studenten zwischen Gruppen
- Erstellen von mehreren Ordnern gleichzeitig
- Verlinken von Tests und Aufgaben in die erstellten Gruppen
- Hinzufügen eines Tabs „Gruppenfilter“ mit dem es möglich ist, in der Bewertungsansicht für Übungseinheiten von Aufgaben nach Gruppen zu filtern

Die Verwendung des Plugins *TestOverview* setzt keineswegs die Benutzung des Plugins *ACO* vorraus. Die simultane Verwendung ist jedoch empfehlenswert.

3.2 Installation

Die Installation auf Unix-Systemen wird im ReadMe-File in GitHub beschrieben.

3.3 Module

TestOverview besteht aus drei Modulen. Die GUI-Klassen, die Mapper-Klassen und einem losen Verband von Klassen die hauptsächlich Berechnungen dienen. Der Klasse ilObjTestOverviewGUI kommt hier die Rolle des Controllers zu.
Sie überwacht GUI-Elemente und leitet den Kontrollfluss an zuständige Klassen weiter. GUI-Elemente sind in unserem Fall Buttons, Tabs und Subtabs. Führt der Nutzer eine Aktion aus, wird am Controller das Signal empfangen und die entsprechende Methode wird von der zuständigen Klasse aufgerufen.
Mapper-Klassen sind die Schnittstellen zwischen Plugin und Datenbank. In der Regel initialisiert der Controller einen Mapper, welcher Daten bereitstellt, welche eine GUI Klasse für Berechnungen und Darstellung nutzt.

3.4 Dokumentation der Funktionen

3.4.1 Tabellen hinzufügen

Zu den Tabellen die erstellt werden können, zählen:

- Testübersichten
- Aufgabenübersicht
- Gruppenübersicht
- Übersicht über hinzugefügte Tests und Aufgaben

Um die Tabellen zu erstellen, werden verschiedene Methoden aufgerufen.
Die Methoden *showRanking()*, *getTestList()*, *getMembershipList()*, *subTabEo()* und *subTabEoRanking()* funktionieren in identischer Weise. Exemplarisch wird das hier an der *showContent()*-Methode aus der Klasse ilObjTestOverviewGUI gezeigt.
Sobald die Übersicht aufgerufen wird, wird die *showContent()* Methode aufgerufen. Innerhalb der *showContent()* Methode wird ein ilOverviewMapper und eine ilTestOverviewTableGUI erstellt. Es ist zu beachten, dass ilTestOverviewTableGUI eine Kindklasse von MappedTableGUI ist, welche wiederum eine Kindklasse von Table2GUI ist.
Durch den Befehl *setMapper()*, wird dem GUI-Objekt ein Mapper zugewiesen. Dadurch wird festgelegt welche Daten dem GUI-Objekt übergeben werden. Falls ein **ilOverviewMapper** hinzugefügt wird, werden Daten für eine Testübersicht bereitgestellt Für eine Aufgabenübersicht verwendet man einen **ilExerciseMapper**.

Die *populate()* Methode, die in MappedTableGUI deklariert ist, liefert alle Nutzer, welche in der Übersicht anzuzeigen sind. Hierfür wird festgestellt, ob eine externe Sortierung und Segmentierung stattfindet. In unseren Fall trifft das nicht zu, da die Sortierung durch das Plugin übernommen wird.
Die *getList()* Methode gibt alle Gruppen zurück, welche in der Verwaltung der Übersicht hinzugefügt wurden. Sollte ein Gruppenfilter aktiv sein wird dies von der *getList()* Methode berücksichtigt. Die ID's der Teilnehmer werden in *formatData()* ermittelt.
Sollten keine Gruppen ausgewählt worden sein, werden in *formatData* sämtliche Nutzer ermittelt, welche an den zum Overview-Objekt hinzugefügten Tests teilgenommen haben. In der in *formatData* aufgerufenen Methode *fetchUserInformation()* werden den Nutzern persönlichen Daten zugeordnet, wie Name und Geschlecht. Sollte der Namens- bzw. Geschlechtsfilter aktiv sein werden die Nutzer danach gefiltert. Die so ermittelten Daten werden durch *setData()* an ilTestOverviewTableGUI übergeben.

3.4.2 Eigene Leistungen anzeigen

Um die eigenen Leistungen anzuzeigen, wird die Methode *UserResults()* in ilObjTestOverviewGUI aufgerufen. Dort wird ein **ilOverviewStudentMapper** initialisiert, der GUI Funktionen und gleichzeitig Mapperfunktionen bereitstellt.
Die Methode *getResults()* generiert das Template. Hierzu werden in einer Datenbankabfrage alle ausgewählten Tests und Aufgaben in TestOverview ausgegeben. Durch mehrere Methoden werden alle Werte im Template eingefügt.
Die Nutzerdaten werden durch die Methoden *getTestData()* und *getExerciseMarks()* ausgegeben und in das Template geschrieben.
Um die Daten für das Ranking abzufragen, wird sowohl ein ilExerciseMapper, wie auch ein ilOverviewMapper erstellt. Die Methoden *getRankedStudent()*, *getCount()* und *getDate()* liefern die Daten.

3.4.3 Export

Bei der Erstellung der Tabs durch die Methode *setTabs()*, welche in der Klasse ilObjTestOverviewGUI deklariert ist, wird für den Exporttab mit der Methode *addTarget()* der Befehl *export* übergeben. Dies führt dazu, dass der erste Befehl der ausgeführt wird *export()* ist, sobald der ExportTab aufgerufen wird.

Nach dem Aufruf des Exporttabs, wird in der *performCommand()* Methode die nächste zuständige Klasse ermittelt. Der Switch-Case Fall *iltestoverviewexportgui* wird aufgerufen. Dieser erstellt eine ilTestOverivewExportGUI und leitet durch die Methode *forwardCommand()* den Kontrollfluss an die ilTestOverviewExportGUI weiter. Dadurch wird nun anstatt der *performCommand()* Methode der Klasse ilObjTestOverviewGUI, die *performCommand()*-Methode der Klasse ilTestOverviewExportGUI genutzt.
Es wird wie bereits beschrieben nun die Methode *export()* ausgeführt. Diese ruft die Methode *initExportForm()* auf. Hierbei wird eine ilPropertyFormGUI erstellt, welche die Exporteinstellungen anzeigt. Schließlich wird ein Export-Button hinzugefügt, der bei Aktivierung die *triggerExport()* Methode auslöst.
Die *triggerExport()* Methode prüft welches Ausgabeformat genutzt werden soll und erstellt ein neues ilTestOverviewExport-Objekt. Dieses ist für den eigentlichen Export verantwortlich und erstellt mit der *buildExportFile()* Methode schließlich ein CSV-Dokument. Die ILIAS-interne Methode *ilUtil::deliverData()* ist schlussendlich für den Output-Stream der Datei verantwortlich.

3.4.4 Ein Diagramm anzeigen

Um ein Diagramm anzuzeigen, wird die Methode *SubTabTo()* aufgerufen. Dort wird ein **BinDiagrammMapper** erstellt. Mit diesem Mapper wird durch die Methode *createAverageDia()* ein neues Diagramm-Objekt erstellt. Über die Methode *initDia()* wird das Chart erstellt und zurückgegeben.

4 Tests

Im folgenden Abschnitt werden alle Testfälle beschrieben, die auf dem Plugin getestet wurden. Sie teilen sich in folgende Kategorien auf :
Allgemein | TestOverview | ExerciseOverview | Export

4.1 Testbedingungen

Die Tests wurden auf einem Unix System mit Apache2 und PHP5.6/PHP7.0 durchgeführt. Hierbei wurden die ILIAS Versionen 5.1 und 5.2 getestet.

4.2 Allgemein

Testbeschreibung	**Soll Ergebnis**	**Resultat**
Exercises und Test wurden hinzugefügt und anschließend gelöscht	In der Useransicht befinden sich nur nicht gelöschte Objekte	✓
Ein Test wurde erstellt und bearbeitet. Das End-Datum des Tests ist allerdings noch nicht erreicht (Datum wird beim Erstellen des Tests festgelegt)	Der Test sollte in der Useransicht nicht sichtbar sein; noch sollten seine Ergebnisse die Statistiken oder den Rang in der Ansicht beeinflussen	✓
Das Testdatum eines Tests wird erst später festgelegt (liegt in der Zukunft)	Ab dem Zeitpunkt ab dem das Datum feststeht sollte der Test nicht mehr in der Useransicht angezeigt werden	✓

4.3 TestOverview

Testbeschreibung	**Soll Ergebnis**	**Resultat**
Tests hinzufügen	Der Test wird in der Overview Matrix angezeigt (+ Ergebnisse aller Studenten)	✓
Test Ranking mit 3 Personen (unterschiedliche Punktzahl)	Studenten werden passend nach ihrem Gesamtergebnis geranked	✓
Ranking Button ohne Personen benutzen	Ranking wird Upgedatet es kommt zu keiner Fehlermeldung	✓
Ranking 2 Personen mit demselben Punktestand	Die Personen erhalten aufeinander folgende Positionen	✓
Test hinzufügen an dem noch kein Student teilgenommen hat	Test wird in der Matrix anzeigt (falls Studenten in der Matrix wird nicht teilgenommen eingeblendet)	✓
Gruppen Filter	Es werden nur Mitglieder der ausgewählten Gruppen angezeigt	✓
Test ohne Gruppen	Es werden alle Mitglieder mit deren Resultaten angezeigt	✓
Link: Testname	Der Link führt zu dem Info Tab des Tests	✓
Link: Statistik	Der Link unter den Testnamen führ zu dem Statistik Tab der Test	✓
Link: Testfeld	Der Link führt zu dem Resultat des einzelnen Studenten	✓

4.4 ExerciseOverview

Testbeschreibung	Soll Ergebnis	Resultat
Exercise hinzufügen	Der Exercise wird in der Overview Matrix angezeigt (+ Ergebnisse aller Studenten)	✓
Exercises wurden mit String bewertet (ohne Zahlen)	Die Summe des Studenten wird auf „Nicht Berechenbar“ gesetzt|Einzelergebnis wird als String angegeben	✓
Exercises wird mit String bewertet (mit Zahlen)	Die Summe des Studenten wird auf „Nicht Berechenbar“ gesetzt|Einzelergebnis wird als String angegeben	✓
Exercise wird nach dem Import gelöscht	Exercise wird nicht mehr angezeigt	✓
Es wird eine Diagrammgranularität gewählt, bei der mehr als 100 Buckets berechnet werden müssten	Die Bucketgröße sollte fix auf 100 Buckets gesetzt werden	✓
Es wird versucht ein Diagramm ohne Studentenresultate zu erstellen	Es wird eine leeres Diagramm angezeigt	✓
Es wir eine Diagramm erstellt, bei dem alle Summen nicht Berechenbar sind	Ein leeres Diagramm + Fehlermeldung wird angezeigt	✓
Es wird als Bucketgröße bei Diagrammen 0 angegeben	Diagramm wird nicht berechnet (Es wird eine Meldung ausgegeben, dass der Wert nicht 0 sein darf)	✓
Es wird als Bucketgröße bei Diagrammen eine negative Zahl angegeben	Diagramm wird nicht berechnet	✓
Link: Exercisename	Link führt zum Assignment Tab der Exercise	✓

4.5 Export

Testbeschreibung	**Soll Ergebnis**	**Resultat**
Export ohne Teilnehmer	Es wird keine Datei Exportiert	✓
Export von zwei gleich benannten Blättern/Tests	Beide Blätter/Tests werden mit ihrem Namen angezeigt	✓
Export von Namen mit Semikolon	Namen werden mit Semikolon angezeigt	✓
Nachträgliche Namensänderung von Tests und Exercises	Der Export soll die Tests und Exercises mit aktuellem Namen listen	✓
Export von Exercises und Test mit gleichen Namen	Test und Exercise werden mit ihrem richtigen Namen im Export angezeigt	✓
Export von Namen mit Anführungszeichen	Anführungszeichen werden ebenfalls exportiert (Es kommt zu keinen Stringfehlern)	✓
Export von Umlauten	Umlaute werden korrekt exportiert	✓
Nachträgliches Löschen von Ergebnissen	Ergebnisse werden nicht mehr im Export angezeigt	✓