Implementierungsdokument
Praxis der Softwareentwicklung am KIT
TasKit - Der Übungsblatt Helfer

Jan Hoffmann, Katharina Kaus, Felix Müller,
Markus Sihler und Theo Wieland

02. Februar 2020

Inhaltsverzeichnis

1 Änderungen zum Entwurf

1.1 Model

1.1.1 Ranking

Es wurde die Methode getPlacements() hinzugefügt.

1.1.2 Placement

Es wurde die Methode getUser() hinzugefügt.

1.1.3 Submission

Es wurden die Methoden getScore(), setScore(), getFeedback(), setFeedback(), getUser() und setUser() hinzugefügt.

1.1.4 AdminModule

Es wurde das Attribute adminServer hinzugefügt.

1.1.5 Module

Implementiert nun Comparable<Module>. Die compareTo Methode vergleicht zwei Module erst nach ihrem Semester und anschließend alphabetisch nach ihrem Namen
Es wurde die Methode setModuleID() hinzugefügt. Die Methoden setModuleID() und getModuleID() wurden umbenannt zu setId() und getId().

1.1.6 UserModule

Der Parameter User tutor wurde zur Signatur des Konstruktors hinzugefügt
Die Methode enterScore(double score, ExerciseSheet sheet) wirft nun eine Exception, falls Punkte für ein Blatt, das nicht zum Modul gehört, eingetragen werden sollen

1.1.7 SupportedAdminModule

Es wurde das Attribute supAdminServer hinzugefügt.
Die Parameter User creator, List<User> admins und List<User> tutors wurden zur Signatur des Konstruktors hinzugefügt

1.1.8 TutorModule

Zum Konstruktor wurde ein passwort: String Parameter hinzugefügt, der das Passwort zum beitreten zu diesem Tutorium initialisiert.
In der addTutorialParticipant(User participant) Methode wird für den Fall, dass übergebener Nutzer schon Mitglied in einem anderen Tutorium ist, keine IllegalArgumentException geworfen.
In der importPoints Methode wurde der Typ des points Parameters zu List<Submission> geändert. Außerdem wird eine IllegalArgutment geworfen falls kein Übungsblatt mit der ID des sheetID Parameters existiert.
In der exportPoints Methode wurde die Matrikelnummer Spalte im auszugebenden Dokument entfernt. Außerdem wurde der Rückgabetyp zu List<CommaSeperatedValues> geändert.

1.1.9 TutorServer

Es wurde die Methode checkIfUserIsInOtherTutorial() hinzugefügt, die überprüft, ob der übergebene Benutzer bereits Mitglied in einem anderem Tutorium ist.

1.1.10 PDFFileStorage

Den Methoden savePDF() und fetchPDF() werden nun auch Callbacks übergeben, über die das Resultat der Server Operationen mitgeteilt wird.

1.1.11 FirebasePDFFileStorage

Die StorageReference wird nicht mehr im Konstruktor übergeben, sondern wird direkt von FirebaseStorage angefordert.

1.1.12 FirebaseUserServer, FirebaseAdminServer, FirebaseSupportedAdminServer und FirebaseTutorServer

Es wurden Attribute für FirebaseFirestore und FirebaseUser hinzugefügt

1.1.13 NoReturnServerCallback, SingleReturnServerCallback, und MultiReturnServerCallback

Es wurde eine CountDownLatch hinzugefügt, die per Konstruktor an die Callbacks übergeben wird. Über diese CountDownLatch kann ein Thread, der eine Serverfunktionalität aufgerufen hat, warten, bis diese beendet ist.

1.1.14 Certification, CertificationCriterion, CertificationCriterionMinPercentagePoints, CertificationCriterionMinRequiredPoints, CertificationCriterionSubmittedSheets und CertificationCriterionType

Die jeweiligen Methoden getProgress() wurden in calculateProgress() umbenannt, da sie von Firebase fälschlicherweise für Getter gehalten wurden.
Die Übungsschein- und Scheinkriterien-Klassen überschreiben nun die toString()-Methode.

1.1.15 CertificationCriterionMinPercentagePoints

Die Methode public double getMinPercentage(): void wurde eingefügt, welche den privaten Wert minPercentage zurückgibt, damit dieser auch auf dem Server gespeichert werden kann.

1.1.16 CertificationCriterionMinRequiredPoints

Die Methode public double getMinPoints(): void wurde eingefügt, welche den privaten Wert minPoints zurückgibt, damit dieser auch auf dem Server gespeichert werden kann.

1.1.17 CertificationCriterionSubmittedSheets

Die Methode public double getNumberOfSheets(): void wurde eingefügt, welche den privaten Wert numberOfSheets zurückgibt, damit dieser auch auf dem Server gespeichert werden kann.

1.1.18 Semester

Die von Objekt geerbte Methode equals(Object o) wurde überschrieben um zu überprüfen ob zwei Semester identisch sind (Selbes Jahr und Semestertyp).

1.1.19 SemesterType

Die toString() Methode von Semestertype wurde überschrieben und es wurde eine yearToString(int year): String Methode erstellt, die eine Jahreszahl für die jeweilige Semesterart formatiert (SS: 2019 -> "19", WS: 2019 -> "19/20").

1.1.20 UserServer

Es wurde die Methode leaveModule() hinzugefügt, über die ein Benutzer aus einem unterstütztem Modul wieder entfernt werden kann

1.1.21 AdminServer, SupportedAdminServer, FirebaseAdminServer, FirebaseSupportedAdminServer

Diesen Klassen wurde eine Methode hinzugefügt, welche es ermöglicht, die maximal erreichbare Punktzahl auf einem Übungsblatt zu ändern:
public changeMaxPointsOfExerciseSheet(moduleID: String, sheetID: int, newMaxPoints: int, callback: NoReturnServerCallback) Ändert die maximal erreichbare Punktzahl auf dem Angegebenen Übungsblatt auf dem Server und speichert das Ergebnis im Callback.
@param moduleID: Die ID des Moduls, dessen Übungsblatt bearbeitet werden soll
@param sheetID: Die ID des Übungsblattes, dessen Maximale Punktzahl bearbeitet werden soll
@param newMaxPoints: Der neue Wert der maximal erreichbaren Punkte
@param callback: Der Callback, welcher aufgerufen wird, um zu bestätigen, ob das Ändern der Punktzahl des Übungsblattes geklappt hat oder nicht funktioniert hat.

1.1.22 Server

Die Methode getUserByName wurde hinzugefügt, um einen Benutzer über seinen Benutzer-Namen vom Server zu laden

1.1.23 FirebaseSupportedAdminServer

Die Methode fetchAllScore wurde hinzugefügt, um alle die Gesamtpunktzahl aller Benutzer eines Moduls über die Ausführung einer CloudFunction zu erhalten

1.1.24 CSVFileConverter

Die Methode writeScores wurde hinzugefügt, um das Ergebnis der vorher erwähnten fetchAllScore Methode in eine CSV-Datei zu schreiben

1.1.25 CertificationCriterion

Es wurde eine neue Methode „setId(int id)“ hinzugefügt, welche die Id des Scheinkriteriums setzt. Es wurde ein Konstruktor hinzugefügt welcher eine „int id“, „CertificationCriterionType type“ sowie „List<ExerciseSheet> sheets)“ benötigt.

1.1.26 CertificationCriterionMinPercentagePoints

Die Signatur des Konstruktors wurde von „(minPercentage: double)“ zu „(int id, CertificationCriterionType type, List<ExerciseSheet> sheets, double minPercentage)“ geändert.

1.1.27 CertificationCriterionMinRequiredPoints

Die Signatur des Konstruktors wurde von „(minPoints: double)“ zu „(int id, CertificationCriterionType type, List<ExerciseSheet> sheets, double minPoints)“ geändert.

1.1.28 CertificationCriterionSubmittedSheets

Die Signatur des Konstruktors wurde von „(numberOfSheets: int)“ zu „(int id, CertificationCriterionType type, List<ExerciseSheet> sheets, int numberOfSheets)“ geändert.

1.2 View, ViewModel

1.2.1 AddUserScreenViewModel

Das currentUsers-Attribut ist nun ein MutableLiveData<List<User»-Attribut anstatt eines <List<User>-Attributes

1.2.2 AddTuteeScreenViewModel

Die remove(User user)-Methode der Oberklasse wird nun überschrieben, um sich zu merken, welche Benutzer entfernt wurden.

1.2.3 ModuleDataHolder

Der Datentyp des Attributs „semester“ wurde von „MutableLiveData<String>“ zu „MutableLiveData<Semster>“ geändert.
Es wurden die Attribute „MutableLiveData<List<User» administrators“ und „MutableLiveData<List<User» tutors“ hinzugefügt.
Es wurde eine neue Methode „void parseModule(UserModule selectedModule)“ hinzugefügt, wMelche das übergebene Modul in den View Elementen der GUI konvertiert und anzeigt.

1.2.4 ExerciseSheetDataHolder

Es wurde das Attribut „MutableLiveData<Boolean> editable“ hinzugefügt, welches beschreibt ob ein Modul erstellt wird oder lediglich bearbeitet wird.
Es wurde ein Konstruktor „ExerciseSheetDataHolder(int id)“ hinzugefügt, welcher den

ExerciseSheetDataHolder mit der übergebenen Id initialisiert.

1.2.5 CertificationDataHolder

Es wurde das Attribut „MutableLiveData<Boolean> examBonusPointsEnabled“ hinzugefügt, welches den Status speichert ob die Klausur Bonuspunkte gibt oder nicht.
Es wurde ein Konstruktor „CertificationDataHolder(int id)“ hinzugefügt, welcher den CertificationDataHolder mit der übergebenen Id initialisiert.
Der Datentyp des Attributs „exerciseSheets“ wurde von „MutableLiveData<List<ExerciseSheetDataHolder»
zu „MutableLiveData<HashSet<ExerciseSheetDataHolder»“ geändert.

1.2.6 CertifictaionCriterionDataHolder

Der Datentyp des Attributs „type“ wurde von „MutableLiveData<String>“ zu „MutableLiveData<CertificationCriterionType>“ geändert.

1.2.7 InputVerification

Die InputVerification Klasse wurde komplett eingefügt und besitzt eine statische Methode „assertInput(ModuleDataHolder moduleDataHolder)“ welche die Benutzereingaben in die Modul Erstellungs/Bearbeitungs-ansicht auf Syntaxfehler überprüft.

1.2.8 CreateModuleFragmentViewModel

Es wurde die Methode „uploadExerciseSheetPdfFiles(HashMap<ExerciseSheet, File> filePairs“ hinzugefügt, welche die Übergebenen Dateien als Übungsblätter auf den Server hochlädt.

1.2.9 EditModuleFragmentViewModel

Es wurden die Attribute „ArrayList<Integer> deletedExerciseSheetIds“ und „ArrayList<Integer> newlyAddedExerciseSheetIds“ hinzugefügt, welche Änderungen des Benutzers an den

Übungsblättern eines zu bearbeitenden Moduls speichern.
Es wurde das Attribut „AdminModule moduleToParse“ hinzugefügt, welches zur Kommunikation zwischen mehreren Fragments dient und somit das Modul beinhaltet welches als nächstes bearbeitet werden soll.

1.2.10 ModuleFormFragmentViewModel

Es wurde das Attribut „MutableLiveData<Integer> viewOutput“ hinzugefügt, welches gegebenenfalls eine Nachricht enthält welche als nächstes an den Benutzer ausgegeben werden soll.
Es wurde die Methode „ArrayList<ExerciseSheet> parseExerciseSheets()“ welche die in der GUI angelegten Übungsblätter in ExerciseSheet Objekte konvertiert.
Es wurde die Methode „List<Certification> parseCertifications(ArrayList<ExerciseSheet> exerciseSheets)“ hinzugefügt, welche die in der GUI angelegten Übungsscheine in Certifiation Objekte konvertiert.
Die Signatur der Methode „addCertificationCriterion()“ wurde von „addCertificationCriterion()“ zu „addCertificationCriterion(CertificationDataHolder certification)“ geändert.

1.2.11 ModuleFormFragment

Es wurde die Methode „bindLayout(FragmentModuleFormBinding binding, ModuleFormFragmentViewModel viewModel)“ hinzugefügt, welche zur Vererbung der Layouts dient und somit all diejenigen GUI Element steuert und initialisiert welche von allen Unterklassen der ModulFormFragment Klasse benötigt werden.

1.2.12 CreateModuleFragment

Es wurde die Methode „bindLayout(FragmentModuleFormBinding binding, ModuleFormFragmentViewModel viewModel)“ hinzugefügt, welche von der Oberklasse geerbt wird und zur Vererbungsbeziehung der Layouts dient.

1.2.13 ModuleFormFragmentCertificationCriterionListViewAdapter

Es wurde ein Konstruktor eingefügt welcher eine Referenz auf einen „LifecycleOwner“, „ModuleFormFragmentViewModel“ sowie „List<CertificationCriterionDataHolder>“ be-

nötigt.
Es wurde die Methode „setCertificationCriteria(List<CertificationCriterionDataHolder> criteria)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(CertificationCriterionDataHolder data)“ Methode welche das angegebenen Übungsscheinkriterium an der entsprechenden Position in der Liste anzeigt.

1.2.14 ModuleFormFragmentCertificationListViewAdapter

Es wurde ein Konstruktor eingefügt welcher eine Referenz auf einen „LifecycleOwner“, „ModuleFormFragmentViewModel viewModel“ sowie „List<CertificationDataHolder> certifications“ benötigt.
Es wurde die Methode „setCertifications(List<CertificationDataHolder> certifications)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(CertificationDataHolder data)“ Methode welche den angegebenen Übungsschein an der entsprechenden Position in der Liste anzeigt.

1.2.15 ModuleFormFragmentCriterionSelectedSheetsListViewAdapter

Es wurde ein Konstruktor eingefügt welcher eine Referenz auf einen „LifecycleOwner“, „ModuleFormFragmentViewModel viewModel“ sowie „List<ExerciseSheetDataHolder> sheets“ benötigt.
Es wurde die Methode „setSheets(CertificationCriterionDataHolder criterion, List<ExerciseSheetDataHolder sheets)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelement weiterreicht. Der ViewHolder verfügt über eine „bind(ExerciseSheetDataHolder data)“ Methode welche das übergebene Übungsblatt an der entsprechenden Position in der Liste anzeigt.

1.2.16 ModuleFormFragmentSheetsListViewAdapter

Es wurde ein Konstruktor eingefügt welcher eine Referenz auf einen „LifecycleOwner“, „ModuleFormFragmentViewModel viewModel“ sowie „List<ExerciseSheetDataHolder>

sheets“ benötigt.
Es wurde die Methode „setSheets(List<ExerciseSheetDataHolder> sheets)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(ExerciseSheetDataHolder data)“ Methode welche das übergebene Übungsblatt an der entsprechenden Position in der Liste anzeigt.

1.2.17 JoinSupportedModuleViewModel

Der Datentyp des Attributs „modules“ wurde von „List<SupportedAdminModule> modules“ zu „MutableLiveData<List<UserModule» modules“ geändert.
Die Methode „joinModuleButtonClicked(UserModule module)“ wurde um den Parameter „NoReturnServerCallback callback“ erweitert, welcher benötigt wird um Auskunft über den Erfolg es Beitritts weiter zu geben.

1.2.18 JoinSupportedModuleFragmentModuleListViewAdapter

Es wurde ein Konstruktor hinzugefügt welcher eine Referenz auf das JoinSupportedModuleViewModel benötigt.
Es wurde die Methode „setModules(List<UserModule> updatedModules)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(UserModule data, int position)“ Methode welche das gegebene Modul an der entsprechenden Position in der Liste anzeigt.

1.2.19 RecyclerViewItemClickListener

Dieses Interface wurde neu hinzugefügt.

1.2.20 NetworkErrorListener

Dieses Interface wurde neu hinzugefügt.

1.2.21 FrontPageFragment

Diese Klasse implementiert nun das NetworkErrorListener Interface.

1.2.22 FrontPageFragmentModuleListViewAdapter

Es wurden Getter und Setter für das modules Attribut hinzugefügt.

1.2.23 FrontPageFragmentModuleListViewAdapter.ModuleViewHolder

Diese Klasse wurde neu hinzugefügt. Sie hält die View Elemente einer Modul Kachel auf der Startseite.

1.2.24 FrontPageFragmentModuleListViewAdapter.SemesterViewHolder

Diese Klasse wurde neu hinzugefügt. Sie hält die View Elemente eines Semester-Dividers auf der Startseite.

1.2.25 FrontPageFragmentViewModel

Der Typ des modules Attributs wurde zu MutableLiveData<List<UserModule» geändert und es wurde eine Getter-Methode für das modules Attribut erstellt.
Der loadAllUserModules Methode wurde ein NetErrorListener als Parameter hinzugefügt. Dieser wird bei einem Fehler bei der Serververbindung benachrichtigt. Diese Methode sortiert nun außerdem die Module nach ihrem Semester und fügt null Placeholder dort ein, wo später auf der Startseite die Semester-Divider angezeigt werden sollen.
Es wurde eine getFABStatus Methode hinzugefügt, die den aktuellen Wert des fabStatus Attribut zurückliefert. Falls dieser null ist wird false zurückgeliefert.

1.2.26 RankingFragment

Diese Klasse ist nun abstract. Es wurden die abstrakten Methoden refreshRanking und getRanking eingefügt, die die anzuzeigenden Ranking Daten entweder zurückliefern oder aktualisieren sollen.

1.2.27 ExternalRankingFragment

Diese Klasse implementiert nun das NetworkErrorListener Interface.

1.2.28 InternalRankingFragment

Diese Klasse implementiert nun das NetworkErrorListener Interface.

1.2.29 RankingFragmentListViewAdapter

Es wurde eine setPlacements Methode hinzugefügt, die die in der Liste anzuzeigenden Daten aktualisiert und die GUI Elemente bei Änderungen benachrichtigt.

1.2.30 RankingFragmentListViewAdapter.RankingViewHolder

Diese Klasse wurde neu hinzugefügt. Sie hält die View Elemente eines Ranking Listeneintrags.

1.2.31 ExternalRankingFragmentViewModel/InternalRankingFragmentViewModel

Die loadRanking Methode wurde entfernt und durch die neuen Methoden getRanking und refreshRanking ersetzt.

1.2.32 JoinRankingFragment

Diese Klasse wurde entfernt.

1.2.33 JoinExternalRankingFragment/JoinInternalRankingFragment

Beide Klassen sind nun Unterklassen von DialogFragment und nicht mehr von JoinRankingFragment.
Die Sichtbarkeit der joinButtonPressed Methode wurde von public zu private geändert.

1.2.34 WebViewFragment

Diese Klasse wurde entfernt. Die Anmeldung mit OpenID wird stattdessen über den Standardbrowser des Nutzers durchgeführt.

1.2.35 ViewUtils

Diese Klasse wurde neu hinzugefügt. Sie enthält, die statische makeNetworkErrorToast Hilfsmethode.

1.2.36 LoginFragment

Es wurden die Methode „signInAsUser(FragmentLoginBinding binding)“ und „signInAsSystemAdministrator(FragmentLoginBinding binding)“ hinzugefügt, welche den Benutzer zu Testzwecken als Benutzer bzw. Administrator anmelden ohne das hierfür ein KIT-Account benötigt wird. Der Benutzer muss sich hierbei nicht selbes authentifizieren.

1.2.37 LoginFragment.LoginProcess

Dieses Interface wurde neu hinzugefügt.

1.2.38 ActivityWithToolbar

Dieses Interface wurde neu hinzugefügt. Repräsentiert eine Android Activity, die eine Toolbar als AppBar besitzt.

1.2.39 ActivityWithDrawerLayout

Dieses Interface wurde neu hinzugefügt. Repräsentiert eine Android Activity, die einen Navigation Drawer besitzt.

1.2.40 MainActivity

Implementiert nun die LoginFragment.LoginProcess, NavigationView.OnNavigationItemSelectedListener, ActivityWithToolbar und ActivityWithDrawerLayout Interfaces.

1.2.41 SelectModuleFragment / AdminSelectModuleFragment / TutorSelectModuleFragment

Die Klasse hat nun die int-Konstanten „ModuleViewType“ = 0 und „SemesterDividerViewType“ = 1 als Attribute sowie Getter für die Konstanten.
Aufgrund von Änderungen in den zugehörigen ViewModel-Klassen überdecken beide Unterklassen von SelectModuleFragment das viewModel-Attribut

1.2.42 AdminSelectModuleFragmentListViewAdapter / TutorSelectModuleListViewAdapter

Es wurde ein Konstruktor eingefügt, welcher eine Referenz auf eine List anzuzeigender Module (SupportedAdminModule / TutorModule), das ViewModel des zugehörigen Fragments (AdminSelectModuleViewModel / TutorSelectModuleViewModel) und das ViewModel der ansteuerbaren Fragments (EditSupportedModuleFragment / TutorModuleFragment) benötigt.
Es wurde die Methode „setModules(List<SupportedAdminModule> modules)“ / „setModules(List<TutorModule> modules)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(SupportedAdminModule data)“- / „bind(TutorModule data)“- Methode welche das angegebene Modul an der entsprechenden Position in der Liste anzeigt.

1.2.43 SelectModuleViewModel / AdminSelectModuleViewModel / TutorSelectModuleViewModel

Da den Unterklassen von SelectModuleViewModel und ihren zugehörigen Fragments explizit die Klasse der gespeicherten Module bekannt sein muss, wurde das currentModules-Attribut aus SelectModuleViewModel entfernt und stattdessen ein MutableLiveData<List<SupportedAdmin Attribut zum AdminSelectViewModel und ein MutableLiveData<List<TutorModule»-Attribut zum TutorSelectViewModel hinzugefügt

Die exportAllPoints-Methode des SelectModuleViewModels erhält nun den Parameter Module module anstatt String moduleID

1.2.44 TutorModuleFragment

Es wurde ein privates ActivityWithToolbar-Attribut und eine zugehörige öffentliche onAttach(@NonNull Context context)-Methode hinzugefügt.
Dem Layout wurde ein Eingabefeld für das Tutoriums-Passwort, ein Button zum Speichern des Passwortes und ein Textfeld zum Anzeigen des Passwortes hinzugefügt.

1.2.45 TutorModuleFragmentTutandListViewAdapter

Das Attribut tutees ist nun vom Typ List<SubmissionDataHolder> anstatt vom Typ List<User>
Es wurde ein TutorModuleFragmentViewModel-Attribut viewModel hinzugefügt
Es wurden die int-Konstanten „ViewTypeExpanded“ und „ViewTypeNotExpanded“ hinzugefügt.
Es wurde ein Konstruktor eingefügt, welcher die List<SubmissionDataHolder> tutees und das TutorModuleFragmentViewModel viewModel initialisiert.
Es wurde die Methode „setTutees(List<SubmissionDataHolder> tutees)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt, welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(SubmissionDataHolder data)“- Methode welche das angegebene Modul an der entsprechenden Position in der Liste anzeigt.

1.2.46 TutorModuleFragmentViewModel

Die removeTutee(User user)-Methode wurde aus dem TutorModuleFragmentViewModel entfernt
Es wurde ein MutableLiveData<String>-Attribut password und eine savePasswordButtonPressed()-Methode hinzugefügt
Es wurde die Methode setFromServer() hinzugefügt, welche das Modul mit Daten vom Server aktualisiert
Es wurde eine isExpanded(User user)-Methode hinzugefügt, die überprüft, ob ein Benutzer momentan im currentlyExpandedSheets-Attribut enthalten ist

Es wurde ein List<List<SubmissionDataHolder»-Attribut dataHolder hinzugefügt, welches die anzuzeigenden Punktzahlen und Feedbacks speichert

1.2.47 SubmissionDataHolder

Das Attribut score ist nun vom Typ MutableLiveData<String> anstatt MutableLiveData<Double>

1.2.48 UserViewModel

Das user-Attribut ist nun ein Objekt der Klasse User anstatt MutableLiveData<User>
Die Signatur der displayNecessaryPointsToNextLevel()-Methode ist nun public double displayNecessaryPointsToNextLevel() anstatt private void displayNecessaryPointsToNextLevel()

1.2.49 UserModuleFragment / SupportedUserModuleFragment / UnsupportedUserModuleFragment

Es wurde ein privates ActivityWithToolbar-Attribut und eine zugehörige öffentliche onAttach(@NonNull Context context)-Methode zum UserModuleFragment hinzugefügt.
Aufgrund von Änderungen in den zugehörigen ViewModel-Klassen überdecken beide Unterklassen von UserModuleFragment das viewModel-Attribut

1.2.50 UserModuleFragmentCertificationListViewAdapter

Es wurden die int-Konstanten „ViewTypeCertification“ und „ViewTypeCriterion“ hinzugefügt.
Es wurde ein Konstruktor hinzugefügt, der das certifications-Attribut initialisiert.
Es wurde die Methode „setCertifications(List<Certification> certifications)“ hinzugefügt, welche die übergebenen Daten in der GUI anzeigt und aktualisiert.
Es wurde ein eigener ViewHolder hinzugefügt, welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(Certification data)“- und „bind(CertificationCriterion data)“ - Methode welche den angegebenen Schein / das angegebene Kriterium an der entsprechenden Position in der Liste anzeigt.

1.2.51 UserModuleFragmentSheetListViewAdapter

Es wurden die int-Konstanten „ViewTypeCollapsed“, „ViewTypeExpanded“ und „ViewTypeUnsupported“ hinzugefügt.
Es wurden die Attribute List<ScoreDataHolder> scores und SupportedAdminViewModel viewModel hinzugefügt.
Es wurde ein Konstruktor hinzugefügt, der de Attribute sheets, scores und viewModel initialisiert.
Es wurde ein eigener ViewHolder hinzugefügt, welcher Daten an die anzuzeigenden Listenelemente weiterreicht. Der ViewHolder verfügt über eine „bind(ExerciseSheet data, in position)“-Methode welche das angegebene Blatt an der entsprechenden Position in der Liste anzeigt.

1.2.52 DataHolder

Das Attribut score ist nun vom Typ MutableLiveData<String> anstatt MutableLiveData<Double>

1.2.53 UserModuleFragmentViewModel

Die loadModule()-Methode wurde aus dieser Klasse und ihren Unterklassen entfernt
Es wurden Getter- und Setter-Methoden für das module Attribut hinzugefügt
Die Sichtbarkeit des module Attributs wurde außerdem von private zu protected geändert.

1.2.54 NotSupportedUserModuleViewModel

Die Klasse heißt nun UnsupportedUserModuleViewModel anstatt NotSupportedUserModuleViewModel

1.2.55 SupportedUserModuleViewModel

Die Signatur der „downloadSheetsButtonPressed(ExerciseSheet sheet)“ Methode wurde zu „downloadSheetButtonPressed(ExerciseSheet sheet, FileDownloadCallback fileDownloadCallback)“ geändert. Der hier übergebene Callback wird benötigt um Auskunft über

den Erfolg das Herunterladen der Datei an das aufrufende Fragment zurückzugeben. Der hierfür benötigte Callback „FileDownloadCallback“ mit den Methoden „finishedDownload(File file)“ und „void errorCreatingFile()“ wurde ebenfalls neu hinzugefügt.

1.3 CloudFunctions

Einige Cloudfunctions lassen sich optimalerweise durch eine Änderung eines Dokuments im Firebase Firestore aufrufen, weshalb diese auch so implementiert wurden.

1.3.1 calculatePlacementWhenLeavingInternalRanking / calculatePlacementWhenLeavingExternalRanking

Es wurden zwei neue Cloudfunctions hinzugefügt, die beim Löschen eines Benutzers aus der Rangliste das Placement aller Benutzer unter dem Gelöschten dekrementieren.

1.3.2 SupportedUserModuleViewModel

Die Signatur der „downloadSheetsButtonPressed(ExerciseSheet sheet)“ Methode wurde zu „downloadSheetButtonPressed(ExerciseSheet sheet, FileDownloadCallback fileDownloadCallback)“ geändert. Der hier übergebene Callback wird benötigt um Auskunft über den Erfolg das Herunterladen der Datei an das aufrufende Fragment zurückzugeben. Der hierfür benötigte Callback „FileDownloadCallback“ mit den Methoden „finishedDownload(File file)“ und „void errorCreatingFile()“ wurde ebenfalls neu hinzugefügt.

1.4 Weitere Änderungen

1.4.1 Matrikelnummer

Da auf Grund von Datenschutzbedenken die Matrikelnummer vom Open ID Connect Provider nicht ausgeliefert wird, haben wir, wie im Entwurf eigentlich vorgesehen, keinen Zugriff darauf in der App. Da die Benutzer-ID allerdings auch eindeutig ist, wird diese nun in der App zur Identifikation von Benutzern verwendet.

2 Verwendete Libraries

2.1 Storage Chooser

Die Storage Chooser Library `https://github.com/codekidX/storage-chooser` ermöglicht es uns mit sehr wenig Aufwand einen FileChooser zu öffnen über welchen der Benutzer Dateien auf seinem Smartphone auswählen kann. Diese Option wird aktuell nicht direkt von Android angeboten und würde deshalb eine eigene Implementierung erfordern.

3 Nicht implementierte Funktionalität

Folgende Funktionalitäten wurden nicht implementiert. Alle anderen Muss- und Wunschkriterien aus dem Pflichtheft (außer die bereits im Entwurfsdokument gestrichenen) sind allerdings implementiert.

3.1 Login

Im Entwurfsdokument war vorgesehen, dass ein Benutzer sich über seinen KIT-Account beim Open ID Connect Provider für die App anmeldet. Dieser sollte dann einen JSON Web Token generieren, der von Firebase validiert wird und dann zum Login verwendet wird. Die Anmeldung mit Open ID Connect und der Erhalt des Token konnte auch implementiert werden, die Verwendung des Token von Firebase setzt allerdings voraus, dass der Token über einen Google Service Account signiert wird. Aktuell ist der Login also so implementiert, dass die Email-Adresse des Benutzers in der App direkt aus dem Token ausgelesen wird und dann ohne Verifikation der Echtheit des Token zum Login mit Firebase verwendet wird. Dies hat zur Folge, dass ein bösartiger Benutzer sich durch einen modifizierten Client Zugriff zu anderen Accounts beschaffen könnte. Eine mögliche Lösung für dieses Problem wäre es, einen eigenen Server (z.B. in der bwCloud) zu hosten, an den der Token, der vom Open ID Connect Provider empfangen wird, weitergeleitet wird und dort erst auf seine Echtheit überprüft wird und dann mit einem Google Service Account neu signiert wrid. Dieser neu signierte Token könnte dann an die App zurückgeschickt werden und damit der Login mit Firebase fortgesetzt werden. Für die Lösung hatten wir allerdings nicht mehr genug Zeit, um sie bis zur Abgabefrist umzusetzen.

3.2 Beitreten von Modulen nach dem ersten Login

Als Wunschkriterium war vorgesehen, einem Benutzer, der sich das erste Mal anmeldet, eine Liste mit allen Modulen anzuzeigen, damit dieser direkt mehreren Modulen beitreten kann. Da dies allerdings auch über das Menü „Öffentlichem Modul beitreten“ möglich ist, hätte diese Funktionalität keinen großen Nutzen erfüllt und wurde somit nicht implementiert.

4 Bekannte Probleme

4.1 Aktualisieren der Rangliste

Aktuell wird, wenn ein Tutor Punkte für seine Tutanten einträgt, die Submissions für jedes Blatt und jeden Tutanden einzeln aktualisiert. Wenn sich dabei die Punkte des Tutanten geändert haben, wird dadurch auch immer die CloudFunction aufgerufen, um die Rangliste zu aktualisieren. Da in der kostenlose Variante von Firebase nur 50 CloudFunctions alle 100 Sekunden ausgeführt werden dürfen, kann es bei der Aktualisierung der Punkte vorkommen, dass dieses Limit erreicht wird. Dadurch kann es zu inkonsistent Daten kommen.

4.2 Entfernen von Tutoren beim Bearbeiten von Modulen

Beim Entfernen von Tutoren beim Bearbeiten von Modulen stürtzt die App ab

4.3 Entfernen und Hinzufügen von Tutanden

Beim Entfernen und Hinzufügen von Tutanden bleibt die App hängen, die Änderungen werden allerdings korrekt gespeichert