Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Entwicklerdokumentation

Case-Gruppe 04

Modul: Software Engineering II

Studiengang Informatik

Sommersemester 2014

Contents

1 Einleitung

Die vorliegende Dokumentation dient zukünftigen Entwicklern dazu sich in das bestehende System einarbeiten zu können und die Arbeit fortsetzen zu können.

2 Enwurf

2.1 Grobentwurf

2.1.1 Architektur

2.1.2 Paketdiagramm

3 Implementierungsentwurf

Unser Programm ist in insgesamt 4 Pakete gegliedert. Die beiden Pakete DozentBelegverwaltungUI und StudentBelegverwaltungUI enthalten sämtliche Klassen für die Benutzeroberflächen (Login, Bearbeitung, ...) des Dozenten- und des Studentenprogrammes. Sie verwenden die Klassen aus dem Typen-paket und greifen gemeinsam auf die Datenbank des Paketes DB_Services zu.
Im folgenden werden für jedes Paket die Klassen aus dem Grobentwurf mit ihren Methoden definiert und ihre Aufgaben beschrieben.

3.1 Paket Typen

3.1.1 Die Klasse Beleg

- **public Beleg(string kennung, string semester, DateTime startDatum, DateTime endDatum, int minM, int maxM, string passwort)**
  Der Konstruktor initialisiert die Kennung, das Passwort zur Erstanmeldung, das automatisch generierte Semester, den Dozenten, ein Start- und Enddatum, sowie die minimale und maximale Gruppenanzahl.

- **public void AddGruppe(Gruppe gruppe)**
  Mit dieser Methode kann einem Beleg eine Case-Gruppe vom Typ Gruppe hinzugefügt werden.

3.1.2 Die Klasse Thema

- **public Thema(int themenNummer, string aufgabe)**
  Ein Thema besitzt nur eine Nummer und einen Namen. Diese beiden Attribute werden durch den Konstruktor gesetzt.

3.1.3 Die Klasse Gruppe

- **public Gruppe(string password, string belegkennung)**
  Durch diesen Konstruktor wird das Passwort und die Belegkennung gesetzt und eine neue Liste vom Typ Student angelegt.

- **public Gruppe(string kennung, int themennummer, string password)**
  Der zweite Konstruktor initialisiert Gruppenkennung, die Themennummer und das Gruppenpasswort. Außerdem wird eine leere Liste vom Typ Student angelegt.

- **public void addStudent(Student student)**
  Diese Methode dient zum Hinzufügen eines Studenten zu einer Case-Gruppe.

3.1.4 Die Klasse Rolle

- **public Rolle(string rolle)**
  Der Konstruktor initialisiert die Rollen-Bezeichnung.

3.1.5 Die Klasse Student

- **public Student(string name, string vorname, string sNummer, string mail, string rolle)**
  Der Konstruktor setzt die wichtigsten Informationen über einen Studenten, nämlich Vor- und Nachname, die s-Nummer, die EMail-Adresse und seine Verantwortlichkeit.

3.2 Paket DozentBelegverwaltungUI

3.2.1 Die Klasse MainForm

Die MainForm-Klasse im Programm Dozent ist dafür da, um einen schnellen Überblick über alle laufenden Belege, dessen Gruppen und der darin enthaltenden Studenten zu gewährleisten. Dabei wird sehr eng mit der Datenbank gearbeitet, um immer aktuelle und richtige Daten anzuzeigen.

- **public MainForm()**
  Dieser Konstruktor initialisiert die Klasse. Es wird der Titel des Fenster gesetzt und die Startposition auf die Mitte des Bildschirms verlagert, so dass der Nutzer einen direkten Überblick über die Funktionen bekommt. Außerdem wird die Funktion UpdateBelege() aufgerufen, wodurch die aktuell laufenden Belege aus der Datenbank geladen und angezeigt werden. Desweiteren werden die Eventhandler für die Doppelklick-Funktionen der Listboxen für die Belege und Gruppen hinzugefügt, damit man diese intuitiv bearbeiten kann.

- **void mitgliederDataGridView_UserDeletingRow (object sender, DataGridViewRowCancelEventArgs e)**
  Diese Funktion wird aufgerufen, wenn der Nutzer eine Zeile (einen Studenten) aus einer Gruppe löschen möchte. Dies wird gemacht, um bei dem Nutzer nachzufragen, ob er sich über die Konsequenzen im Klaren ist und außerdem, um diese Änderung direkt in der Datenbank einzutragen. Ein Student mit der S-Nummer "na" als Platzhalter kann nicht gelöscht werden, damit die Mindest- und Maximalanzahl der Studenten in der Gruppe intakt bleibt.

- **private void UpdateBelege(object sender)**
  Die Funktion UpdateBelege zieht die aktuell laufenden Belege aus der Datenbank und zeigt diese direkt in der belegListBox an. Somit bekommt der Nutzer eine direkte Übersicht auf die Daten in der Datenbank.

- **private void belegListBox_SelectedIndexChanged (object sender, EventArgs e)**
  Dies ist ein Eventhandler, der aufgerufen wird, sobald ein neuer Beleg ausgewählt wurde. Er wird dafür genutzt, um die zu dem Beleg gehörigen Gruppen anzuzeigen. Es wird zuerst der ausgewählte Beleg gesucht. Da die Funktion allgemein aufgerufen werden soll, um die Listbox der Gruppen zu aktualisieren, wird zuerst geschaut, ob überhaupt ein Beleg existiert. Wenn keiner exisitiert, wird der Inhalt der gruppenListBox geleert, da es dann keine zugehörigen Gruppen gibt. Wenn ein Beleg exisitiert, der ausgewählt ist, dann wird eine Anfrage an die Datenbank geschickt, die die zugehörigen Gruppen zurückgibt. Diese werden direkt in die gruppenListBox eingetragen.

- **private void UpdateRollen(Beleg beleg)**
  Diese Funktion zieht die verfügbaren Rollen zu dem ausgewählten Beleg aus der Datenbank. Sie ist dafür notwendig, damit man den Studenten einer Gruppe eine neue Rolle zuweisen kann. Der Beleg als Parameter wird benötigt, um die Belegkennung in der Datenbankabfrage zu verwenden.

- **private void belegListBox_DoubleClicked(object sender, EventArgs e)**
  Dies ist der oben angesprochene Eventhandler für das Doppel-Klick-Event in der belegListBox. Es wird eine neue Instanz der Klasse BelegBearbeiten erstellt, wobei direkt der Delegate-Handler festgelegt wird. Dieser ist notwendig, um Änderungen nach beenden dieses Dialoges in den ListBoxen anzuzeigen. Der zweite Parameter in dem Konstruktor ist ein Boolean-Wert, der angibt, ob es sich um einen neu angelegten Beleg handelt. Da hier das Doppel-Klick-Event abgefangen wird, handelt es sich nicht um einen neuen Beleg, sondern um einen existierenden, der bearbeitet werden soll. Danach wird der Dialog mit der Methode show() dem Nutzer angezeigt.

- **private void gruppenListBox_DoubleClicked(object sender, EventArgs e)**
  Dieser Eventhandler behandelt das Doppel-Klick-Event für die gruppenListBox. Wie bei den Belegen auch gibt es hier einen Boolean-Wert, der der Instanz sagt, ob es sich um eine neue oder eine existierende Gruppe handelt. Auch hier handelt es sich um eine existierende Gruppe, weshalb der Wert false übergeben wird.

- **private void gruppenListBox_SelectedIndexChanged (object sender, EventArgs e)**
  Dieser Eventhandler wird aufgerufen, sobald der Nutzer eine neue Gruppe in der gruppenListBox auswählt. An dieser Stelle müssen jetzt die Studenten dieser Gruppe angezeigt werden. Dafür wird zuerst die ausgewählte Gruppen rausgesucht und anhand dieser und der Gruppenkennung (Case-Kennung) eine Datenbankabfrage nach den Studenten gestartet. Mit dem Ergebnis der Abfrage wird das mitglieder-DataGridView gefüllt, sodass der Nutzer alle Studenten dieser Gruppe einsehen kann. Bei dem Füllen werden außerdem Platzhalter für Studenten angezeigt, die repräsentieren, wieviele Studenten noch bis zur festgelegten Minimalanzahl bzw bis zur festgelegten Maximalanzahl fehlen. Noch auszufüllende Studenten sind dabei Gelb markiert.

- **protected override void OnFormClosing(FormClosingEventArgs e)**
  Wie das Stichwort override verrät, wird an dieser Stelle die Closing-Methode der Windows-Form überschrieben. Dies ist notwendig, weil es sein kann, dass diese Form nur von einer anderen präsentiert wird. Wenn das Fenster geschlossen wird, soll auf jeden Fall auch der Prozess beendet werden.

- **private void belegAnlegenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion kann man einen neuen Beleg anlegen. Sie wird geworfen, wenn der entsprechende Button im Interface angeklickt wird. Dafür wird ebenfalls die Klasse BelegBearbeiten verwendet, allerdings wird diesmal im Konstruktor mit dem Boolean "true" mitgeteilt, dass es sich um eine neue Klasse handelt. Auch hier wird ein Delegate-Handler festgelegt, um die Listboxen nach dem Anlegen zu aktualisieren.

- **private void gruppeAnlegenButton_Click(object sender, EventArgs e)**
  Dies ist der äquivalente Button zu der BelegAnlegen-Funktion. Es wird eine neue Gruppe angelegt, indem eine Instanz der Klasse GruppeBearbeiten angelegt wird. Auch hier wird ein Delegate gesetzt, um nach dem Anlegen die gruppenListBox zu aktualisieren.

- **List<string>getFreieCases(Gruppe gruppe)**
  Diese Funktion repräsentiert eine Datenbankabfrage, mit der freie Case-Kennungen zu einem Beleg aus der Datenbank abgerufen werden können. Diese Funktion

wird in der gruppeAnlegenButton_Click-Funktion verwende, um zu schauen, ob es überhaupt möglich ist, eine weitere Gruppe zu diesem Beleg anzulegen, oder ob nicht alle Case-Kennungen schon vergeben sind. Es wird eine Gruppe und kein Beleg übergeben, weil diese Funktion wie gesagt bei dem Anlegen einer Gruppe aufgerufen wird. Dabei wird vorher eine Platzhalter-Gruppe angelegt, die dem Konstruktor von GruppeBearbeiten übergeben wird. Diese Gruppe kann auch direkt als Anhaltspunkt für diese Funktion gewählt werden.

- **private void dataGridViewFreigebenButton_Click (object sender, EventArgs e)**
  Mit dieser Funktion wird das DataGridView, das die Studenten einer Gruppe repräsentiert für Bearbeitungen freigegeben. Standardmäßig ist dieses gesperrt, damit nicht versehentlich Änderungen gemacht werden.

- **private void saveDataGridViewButton_Click(object sender, EventArgs e)**
  Diese Funktion ruft die Funktion SaveMitglieder() auf und sperrt danach das DataGridView wieder, um den Nutzer vor ungewollten Änderungen zu schützen.

- **private bool SaveMitglieder()**
  Diese Funktion speichert alle Änderungen von Studenten der ausgewählten Gruppe. Dabei wird für jeden Studenten unterschieden, ob es sich um einen neuen oder einen geänderten Studenten handelt. Dies wird davon abhängig gemacht, ob die Zelle, die die S-Nummer beinhaltet, gesperrt ist oder nicht. Wenn sie gesperrt ist, handelt es sich um einen schon existierenden Studenten, der in der Datenbank mit den neuen Werten aktualisiert wird. Ansonsten wird der Student eingefügt. Teil dieses Prozesses ist auch die Überprüfung auf eine korrekte und neue S-Nummer sowie auf einen korrekte E-Mail-Adresse.

- **private bool checkMail(string mail)**
  Dies ist eine einfache und kurze Funktion, die anhand von Regular-Expressions eine eingegebene E-Mail-Adresse auf korrekte Validierung untersucht.

- **private bool checkSNummer(string sNummer)**
  Diese Funktion überprüft, ob es sich bei der eingegebenen S-Nummer wirklich um eine korrekte S-Nummer handelt.

- **private void updateStudent(Student student)**
  Die Funktion bekommt einen Studenten als Parameter, um dessen Werte in der Datenbank zu aktualisieren. Da nicht unterschieden werden kann, ob bei dem Studenten überhaupt etwas geändert wurde, wird diese Funktion beim Speichern für jeden existierenden Studenten durchgeführt.

- **private void insertStudent(Student student, Gruppe gruppe)**
  Diese Funktion trägt einen neuen Studenten in die Datenbank ein. Da an dieser Stelle nicht nur die Tabelle Student wichtig ist, sondern auch die Zuordnungs-Tabelle zwischen Student und Gruppe, wird hier auch die Gruppe mit übergeben.

- **private void cancelDataGridViewButton_Click (object sender, EventArgs e)**
  Mit dieser Funktion kann das Bearbeiten der Studenten einer Gruppe wieder rückgängig gemacht werden. Dabei werden die Elemente zum Bearbeiten wieder gesperrt und

außerdem werden die Studenten der aktuell ausgewählten Gruppe einfach wieder neu aus der Datenbank geladen.

- **private void themenVerwaltenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion, durch den Button "Themen verwalten" ausgelöst, wird eine neue Instanz der Klasse ThemenVerwalten erstellt und diese angezeigt.

- **private void rolleTextBox_Click(object sender, EventArgs e)**
  Mit dieser Funktion wird eine Instanz der Klasse RolleVerwalten geladen und angezeigt, damit man den Pool an Rollen neu aufstellen kann.

- **private void buttonArchivieren_Click(object sender, EventArgs e)**
  Diese Funktion erlaubt das Archivieren von allen aktuellen Belegen. Dabei wird eine Instanz der Klasse PdfArchivierung erstellt. Die Klasse hat ebenfalls einen Delegate-Handler, damit die gelöschten Datensätze nach der Archivierung nicht mehr in dem Programm sichtbar sind.

- **private void button1_Click(object sender, EventArgs e)**
  Diese etwas unglücklich benannte Funktion erstellt eine neue Instanz der Klasse kontaktForm, wodurch der Nutzer bestimmte Gruppen oder Studenten der einzelnen Belege per Mail kontaktieren kann.

- **private void belegLoeschenButton_Click(object sender, EventArgs e)**
  Mit dieser Funktion kann ein ausgewählter Beleg gelöscht werden. Dabei wird zuerst nachgefragt, ob der Nutzer sich der Konsequenzen bewusst ist und ob er sich sicher ist. Danach wird geschaut, ob der zu löschende Beleg noch Gruppen enthält. Wenn dem so ist, wird der Nutzer darüber informiert und ein zweites Mal gefrag, ob er sich dessen sicher ist. Wenn die Funktion bis hier noch nicht beendet wurde, wird der Beleg inklusive der existierenden Gruppen und dessen Studenten aus der Datenbank gelöscht. Danach werden automatisch die Listboxen und das Datagridview aktualisiert, sodass der Nutzer den gelöschten Beleg nicht mehr im Programm sieht.

- **private void gruppeLoeschenButton_Click(object sender, EventArgs e)**
  Die Funktion löscht die ausgewählte Gruppe. Äquivalent zum Beleg wird auch hier Nachgefragt und im Anschluss geschaut, ob die Gruppe Studenten beinhaltet. Wenn dem so ist, wird auch hier darüber informiert und ein zweites Mal nachgefragt. Erst danach werden die Studenten und die Gruppe selbst aus allen betreffenden Tabellen in der Datenbank gelöscht. Danach werden die Listboxen wieder aktualisiert, um keine alten, falschen Daten anzuzeigen.

3.2.2 Die Klasse Eingabe

Mit dieser Klasse wird eine allgemein gültige Form bereitgestellt, über die in Form eines Dialoges Daten eingegeben werden können. Dafür wird ein Textfeld bereit gestellt. Über einen Delegate erfährt die präsentierende Form, dass die Eingabe abgeschlossen ist und bekommt die Daten in Form des Textfeldes zugeteilt.

- **public delegate void textEingabeHandler(object sender);**
  public textEingabeHandler textEingabe;
  Dieser Delegate-Handler muss beim Erstellen einer Instanz dieser Klasse gesetzt werden, um die erfolgreiche Eingabe abzufangen.

- **public Eingabe()**
  Der Konstruktor stellt den Titel und die Position des Fensters ein.

- **private void eingabeButton_Click(object sender, EventArgs e)**
  Dieser Handler wird durch den "Bestätigen"-Button ausgelöst und gibt den eingegebenen Text mittels Delegate-Handler an seinen Delegate zurück.

3.2.3 Die Klasse BelegBearbeiten

Diese Klasse stellt ein User Interface zur Verfügung, mit dem man einen Beleg anlegen oder einen existierenden bearbeiten kann.
Properties:

- isNeuerBeleg
  Dieser Boolean-Wert gibt an, ob es sich bei dem zu bearbeitenden Beleg um einen neuen Beleg oder einen exisitierenden Beleg handelt. Dies ist wichtig, um verschiedene UI-Elemente zu sperren und die richtige Speicher-Methode zu wählen.

- Listen
  Diese Listen repräsentieren die jeweiligen Pools für auswählbare Themen, Rollen und Case-Kennungen. Listen mit dem Präfix "Alle" repräsentieren alle in der Datenbank verfügbaren Daten zu demjeweiligen Bereich. Listen mit dem Präfix "Verf" beinhalten alle Daten, die durch die Studenten dieses Beleges auswählbar sind.

Methoden:

- **public BelegBearbeiten(string belegKennung, bool neu)**
  Dieser Konstruktor bekommt die Belegkennung, um den jeweiligen Beleg aus der Datenbank zu ziehen und den Boolean-Wert, ob es sich um einen neuen Beleg handelt. Danach werden Titel des Fensters, die Fenster-Position und die jeweiligen Daten aus der Datenbank in die UI-Elemente geladen. Handelt es sich um einen

existierenden Beleg, der bearbeitet werden soll, wird das Textfeld der Belegkennung gesperrt, da die Belegkennung ein Primärschlüssel in der Datenbank ist und demzufolge nicht geändert werden darf.

- **private void cancelButton_Click(object sender, EventArgs e)**
  Diese Funktion gibt dem "Abbrechen"-Button die Funktion, dass die Form geschlossen wird.

- **private void speichernbutton_Click(object sender, EventArgs e)**
  Diese Funktion speichert den eingegebenen Beleg in der Datenbank. Dabei wird unterschieden zwischen einem komplett neuen Beleg und einen bearbeiteten Beleg. Ist der Beleg neu, wird außerdem überprüft, ob die Belegkennung schon in der Datenbank vorhanden ist. Dies ist wichtig, da die Kennung ein Primärschlüssel in der Datenbank darstellt.

- **bool checkTextFields()**
  Diese Funktion überprüft, ob die Belegkennung eingegeben wurde und ob die zulässige Größe nicht überschritten wurde.

- **void UpdateBeleg()**
  Diese Methode wird augerufen, wenn ein existierender Beleg bearbeitet wurde und dann gespeichert werden soll. Es werden alle eingegebenen Daten in die Datenbank übernommen. Eventuelle Fehleingaben werden mit den entsprechenden Eventhandlern schon vorher abgefangen.

- **void InsertBeleg()**
  Die Methode InsertBeleg() ist äquivalent zu der Methode UpdateBeleg() mit dem Unterschied, dass es sich hier um einen neuen Beleg handelt. Dabei werden in der Datenbank nicht die Werte geändert, sondern neue Werte eingeschrieben.

- **private void addButtonThema_Click(object sender, EventArgs e)**
  Diese Methode und auch die äquivalenten Methoden ..Rolle_Click(), und ..Case_Click() fügen einen Datensatz aus einer Alle ..Liste in die Verf..-Liste ein und aktualisieren die angezeigten Daten.

- **private void remButtonThema_Click(object sender, EventArgs e)**
  Diese Funktion und auch die äquivalenten Methoden ..Rolle_Click(), und ..Case_Click() fügen einen Datensatz aus einer Verf ..-Liste in die Alle..-Liste ein und aktualisieren die angezeigten Daten.

- **private Beleg GetBelegFromKennung(string belegKennung)**
  Diese Methode wird am Anfang aufgerufen, um den zu ändernden Beleg aus der Datenbank zu ziehen und die dazugehörigen Daten in der Form anzuzeigen.

- **private void startDateTimePicker_ValueChanged (object sender, EventArgs e)**
  Dieser Eventhandler überprüft, dass das Start-Datum vor dem End-Datum liegt. Wenn dem nicht so ist, werden die Werte so angepasst, dass diese Bedingung stimmt. Damit werden Fehleingaben in der Datenbank ausgeglichen, bevor diese überhaupt in die Datenbank geschrieben werden.

- **private void minGR_ValueChanged(object sender, EventArgs e)**
  Mit diesem Eventhandler wird sicher gegangen, dass die Mindestanzahl von Studenten dieses Beleges immer kleiner ist, als die Maximalanzahl. Außerdem wird sicher gegangen, dass die Anzahl nicht kleiner als 0 sein kann.

3.2.4 Die Klasse GruppeBearbeiten

Diese Klasse stellt eine Windows-Form bereit, mit der eine Gruppe bearbeitet werden oder eine neue Gruppe angelegt werden kann.

- **public delegate void GIsSavedHandler(object sender, EventArgs e); public GIsSavedHandler SavedG;**
  Dieser Delegate-Handler teilt später dem Delegate mit, dass die Bearbeitung abgeschlossen ist, sodass dieser seine Ausgabe aktualisieren kann.

- **public gruppeBearbeiten(Gruppe gruppe, bool neu)** Der Konstruktor nimmt die zu bearbeitende Gruppe und einen Boolean-Wert entgegen. Der Boolean-Wert gibt an, ob es sich um eine neue oder eine existierende Gruppe handelt. Außerdem werden die Daten in die Textfelder und Comboboxen geladen. Handelt es sich um eine existierende Gruppe, wird die Combobox für die Kennung gesperrt, da die Casekennung einen Primärschlüssel in der Datenbank darstellt und deswegen nicht mehr geändert werden kann.

- **List<string>getFreieCases()**
  Diese Funktion gibt eine Liste aller noch in diesem Beleg verfügbaren Case-Kennung zurück. Die Liste dient als DataSource für die Combobox, aus der man sich eine Case-Kennung für die Gruppe aussuchen kann. Somit wird direkt umgangen, dass eine Fehleingabe stattfindet.

- **void getThemen()**
  Diese Funktion lädt die für diesen Beleg verfügbaren Themen aus der Datenbank in eine Liste. Außerdem dient die Liste gleich als DataSource für die Combobox, aus der der Nutzer sich ein Thema zur Bearbeitung aussuchen kann.

- **private void cancelButton_Click(object sender, EventArgs e)**
  Dieser Eventhandler wird nach einem Klick auf den Button "Abbrechen" ausgelöst und schließt die Form ohne Speicherung der Daten.

- **private void speichernbutton_Click(object sender, EventArgs e)**
  Diese Methode speichert die Gruppe in der Datenbank und schließt anschließen die Form. Dabei wird unterscheiden, ob es sich um eine neue Gruppe oder einen existierende Gruppe handelt. Je nachdem wird die Gruppe entweder mittels insert in der die Datenbank neu hinzugefügt oder mittels update eine bestehende Gruppe aktualisiert.

- **void updateGruppe(bool mitPasswort)**
  Diese Funktion aktualisiert die Daten einer bestehenden Gruppe mit den neu eingegebenen. Als Parameter wird dabei ein Boolean-Wert mitgegeben, der sagt, ob das Passwort auch geändert werden soll. Da das Passwort nur als Hash-Wert in der Datenbank steht, kann es in der Form nicht angezeigt. Geändert werden muss es daher, wenn in dem Textfeld etwas drinsteht, ansonsten wird das alte Kennwort in der Datenbank belassen.

- **void insertGruppe()**
  Diese Methode speichert eine neue Gruppe in der Datenbank mittels insert. Hierbei wird das Passwort über die Sybase-interne Funktion "internal_encrypt()" verschlüsselt, sodass es nicht als Klartext in der Datenkbank steht.

3.2.5 Die Klasse RolleVerwalten

Diese Klasse beschreibt eine Windows-Form, mit der der gesamte Pool von verfügbaren Rollen in der Datenbank verwaltet werden kann. Es können bestehende Rollen gelöscht und neue Rollen hinzugefügt werden.

- **public RolleVerwalten()**
  Der Konstruktor setzt den Titel der Form und die Startposition des Fensters fest. Außerdem werden aus der Datenbank alle Rollen geladen.

- **private void deleteRolleButton_Click(object sender, EventArgs e)**
  Mittels dieses Eventhandlers wird der Klick auf den "Löschen"-Button abgefangen und die ausgewählte Rolle in der Listbox wird aus der Anzeige und aus der Datenbank herausgelöscht. Dabei wird zuerst geschaut, ob Studenten diese Rolle besetzen. Wenn dem so ist, kann die Rolle nicht gelöscht werden, um inkonsistente Daten in der Datenbank zu vermeiden.

- **private void RefreshRollen()**
  Diese Funktion lädt die Rollen aus der Datenbank und zeigt sie in der Listbox an.

- **private void newRolleButton_Click(object sender, EventArgs e)**
  Mit dieser Methode soll eine neue Rolle hinzugefügt werden. Dafür wird eine Instanz der Klasse Eingabe.cs erstellt. Dabei wird auch gleich der Delegate-Handler festgelegt, damit die Rolle nach erfolgreicher Eingabe in die Datenbank eingetragen werden kann. Danach wird die Instanz mittels der Methode show() angezeigt.

- **public void EingabeF(object sender)**
  Dies ist der Delegate-Handler, um die erfolgreiche Eingabe einer neuen Rolle abzufangen und diese in die Datenbank zu speichern. Davor wird überprüft, ob die eingegebene Rolle zu lang oder leer ist.

3.2.6 Die Klasse ThemenVerwalten

Diese Klasse stellt eine Windows-Form bereit, um die bestehenden Themen in der Datenbank zu verwalten. Themen können gelöscht oder neu hinzugefügt werden.

- **public ThemenVerwalten()**
  Der Konstruktor setzt den Titel des Fensters, sowie dessen Startposition fest. Außerdem werden die Themen aus der Datenbank geladen, damit sie angezeigt werden können.

- **private void deleteThemaButton_Click(object sender, EventArgs e)**
  Dieser Eventhandler fängt das Klicken auf den "Löschen"-Button ab und löscht das ausgewählte Thema aus der Datenbank und der Listbox. Zuvor wird allerdings überprüft, ob eine bestehende Gruppe dieses Thema aktuell ausgewählt hat. Wenn dies der Fall ist, wird der Lösch-Vorgang mit einer Fehlermeldung abgebrochen.

- **private void RefreshThemen()**
  Mittels dieser Methode werden die Themen aus der Datenbank gezogen und direkt in der Listbox angezeigt.

- **private void addThemaButton_Click_1(object sender, EventArgs e)**
  Mit dieser Methode soll ein neues Thema hinzugefügt werden. Dafür wird eine Instanz der Klasse Eingabe.cs erstellt. Dabei wird auch gleich der Delegate-Handler festgelegt, damit das Thema nach erfolgreicher Eingabe in die Datenbank eingetragen werden kann. Danach wird die Instanz mittels der Methode show() angezeigt.

- **public void EingabeF(object sender)**
  Dies ist der Delegate-Handler, um die erfolgreiche Eingabe eines neuen Themas abzufangen und dieses in die Datenbank zu speichern. Davor wird überprüft, ob das eingegebene Thema zu lang oder leer ist.

3.2.7 Die Klasse PdfArchivierung

Die Klasse PdfArchivierung stellt eine Windows-Form bereit, mit der sämtliche Belege des aktuellen Semesters archiviert werden können. Dabei werden alle relevanten Daten bezüglich der Belege mit ihren Case-Gruppen und den jeweiligen Studenten in ein PDF-Dokument gespeichert. Danach wird die Datenbank wieder auf die Werkseinstellungen zurückgesetzt (alle Themen und Rollen werden freigegebe und alle Case-Gruppen werden geleert und freigegeben) sodass die Datenbank wieder für das kommende Semesters genutzt werden kann.

- **public delegate void GIsSavedHandler(object sender);**
  public GIsSavedHandler SavedG;
  Dieser Delegate-Handler teilt später dem Delegate mit, dass die Archivierung abgeschlossen ist, sodass die Listboxen und der DataGridView im Mainform neu geladen werden können.

- **public PdfArchivierung(Database database)**
  Der Konstruktor setzt den Titel des Fensters, sowie dessen Startposition. Aus der Datenbank wird das aktuelle Semester geladen um später in der Überschrift der PDF und im Namen der PDF verwendet werden zu können. Falls der Beleg in einem Semester bereits archiviert wurde, darf der Button natürlich nicht noch einmal betätigt werden.

- **private void buttonArchivieren_Click(object sender, EventArgs e)**
  Nach Betätigen des Buttons wird ein Dialog geöffnet, in welchem der Nutzer den Ort aus wählen kann wo die Archivierungs-PDF gespeichert werden soll. Dieser Vorgang kann auch mittels Abbrechen-Button abgebrochen werden.
  Zunächst wird ein PDF-Dokument zum schreiben geöffnet. Es bekommt eine Überschrift mit dem bereits aus der Datenbank ermittelten Semester. Pro Beleg wird dann in einer Titelseite ein Absatz mit den für den Beleg wichtigsten Informationen generiert (Belegkennung, Semester, Zeitraum, Gruppengröße) und darunter eine Tabelle mit den auswählbaren Themen und den verfügbaren Rollen. Auf einer neuen Seite wird dann pro Case-Gruppe eine Tabelle angelegt, welche das Thema der Gruppe (äußere foreach-Schleife) und die wichtigsten Informationen aller Studenten dieser Gruppe (innere foreach-Schleife aufzeigt.
  Nachdem die PDF mit dem Inhalt gefüllt wurde und schließlich erzeugt wurde, wird sie unter dem vom Nutzer festgelegten Verzeichnis mit dem namen der das Semester enthält gespeichert und dem Nutzer sofort angezeigt. Durch die Delete-Statements wird die Datenbank wieder in ihren Ausgangszustand zurückgesetzt. Der Delegate-Aufruf bewirkt, dass die Listboxen und der DataGridView wieder neugeladen wird, damit die alten, bereits gelöschten Daten nicht mehr angezeigt werden.

- **private void abbrechenButton_Click(object sender, EventArgs e)**
  Falls der Nutzer den Archivierungs-Vorgang nicht ausführen möchte, kann das Fenster mittels Abbrechen-Button geschlossen werden.

3.2.8 Die Klasse KontaktForm

Diese Klasse stellt eine Windows-Form bereit, mit der bestimmte Studenten oder ein Kreis von Studenten per Mail kontaktiert werden können. Dabei wird das Standard-Mail-Programm des Betriebssystems benutzt und die einzutragenden Mail-Adressen automatisch übernommen.

- **public kontaktForm()**
  Der Konstruktor lädt die benötigten Daten aus der Datenbank, sodass die Filter auf sie angewandt werden können.

- **private void updateBelegData()**
  Äquivalent zu ..Themen..(), ..Group..(), ..Rollen..() und ..Filter..() werden hierbei die Daten des jeweiligen Bereichs aus der Datenbank geladen und als DataSource für die jeweilige Combobox festgelegt.

- **private void comboBoxBeleg_SelectedIndexChanged (object sender, EventArgs e)**
  Äquivalent zu ..Belegthema_..(), ..Rolle_..() und ..Gruppe_..() werden hier die jeweiligen Daten nochmals neu geladen und den jeweils ausgewählten Daten entsprechend gefiltert.

- **private void btnFilter_Click(object sender, EventArgs e)**
  Diese Methode sucht alle gewünschten Mail-Adressen aus den Comboboxen und trägt sie automatisch als Empfänger in das Mail-Programm. Außerdem wird als Betreff der Mail automatisch die Belegkennung gesetzt.

3.3 Paket StudentBelegverwaltungUI

3.3.1 Die Klasse LoginForm

- **public LoginForm()**

- **private void LoginButton_Click(object sender, EventArgs e)**

- **private bool checkBelegLogin(string login, string password)**

- **private bool checkGruppeLogin(string login, string password)**
- **private bool freieGruppen(string BelegKennung)**
- **private string getBelegKennungFromGruppenKennung(string kennung)**
- **private bool isInBelegZeitraum(string belegkennung)**
- **private void cancelButton_Click(object sender, EventArgs e)**

3.3.2 Die Klasse MainForm

- **public MainForm(string gruppenKennung, string belegkennung)**
- **void mitgliederDataGridView_UserDeletingRow (object sender, DataGridViewRowCancelEventArgs e)**
- **private Gruppe GetGruppeFromKennungS (string kennung, string belegkennung)**
- **private bool isInBelegZeitraum(string belegkennung)**
- **private void updateMitgliederData(List<Student>errorStudenten)**
- **private void UpdateThemen()**
- **private void UpdateRollen()**
- **private int getMinAnzahlMitglieder(string beKennung)**
- **private int getMaxAnzahlMitglieder(string beKennung)**

- **protected override void OnClosing(CancelEventArgs e)**
- **private int getAnzahlLeiter()**
- **private void saveButton_Click(object sender, EventArgs e)**
- **private void updateStudent(Student student)**
- **private void insertStudent(Student student, Gruppe gruppe)**
- **private bool checkSNummer(string sNummer)**
- **private bool checkMail(string mail)**
- **private void getZeitraum (string belegkennung)**

3.3.3 Die Klasse FormErstanmeldung

- **public FormErstanmeldung(string belegKennung)**
- **private void button1_Click(object sender, EventArgs e)**
- **private void cancelButton_Click(object sender, EventArgs e)**
- **private bool checkSNummer(string sNummer)**
- **private bool checkMail(string mail)**
- **protected override void OnClosing(CancelEventArgs e)**

3.3.4 Die Klasse FormMitgliederNeuEingeben

- **public FormMitgliederNeuEingeben(Student leiter, string belegKennung)**
- **private void commitButton_Click(object sender, EventArgs e)**
- **private void updateRollen()**
- **private void RefreshDatagrid(Gruppe gruppe)**
- **private void cancelButton_Click(object sender, EventArgs e)**
- **private string getGruppenKennung()**
- **private int getThemenNummer()**
- **private int getMaxAnzahlMitglieder(string beKennung)**
- **private void saveGruppeInDatabase()**
- **private void insertStudent(Student student, Gruppe gruppe)**
- **private bool checkSNummer(string sNummer)**
- **private bool checkMail(string mail)**
- **protected override void OnClosing(CancelEventArgs e)**

3.4 Paket DBService

3.4.1 Die Klasse Database

- **public Database()**

- **public void Connect()**

- **public List<string[]>ExecuteQuery(string query)**