Design-Dokument Studienplaner-SPAsS

Version: 3

Datum: 06.07.2022

Inhaltsverzeichnis

Einführung

Das Programm soll als interaktive Modulwahl fungieren und dabei auf CP-Grenzen, Voraussetzungen und Regelungen (z.B. Fortschrittregelungen, Module, CP) des eigenen Studiums eingehen. Dafür muss ein Dokument mit eben jenen Informationen eingelesen werden. Außerdem soll die Möglichkeit bestehen, die eigenen Noten einzutragen. Daraufhin soll dann der eigene Notenspiegel berechnet werden. Genauere Programmanwendungen werden in den Anforderungsspezifikationen Version 4 erläutert. Das Dokument orientiert sich am ARC42-Format.

Bausteinsicht

Es folgt nun die Bausteinsicht, in welcher wir eine statische Ansicht der zur Umsetzung benötigten Klassen und Komponenten angefertigt haben. Auf die einzelnen Diagramme werden wir zusätzlich im Detail eingehen, um Abhängigkeiten zwischen diesen Komponenten und Klassen zu verdeutlichen und damit die Struktur des Programms genauestens zu erläutern.

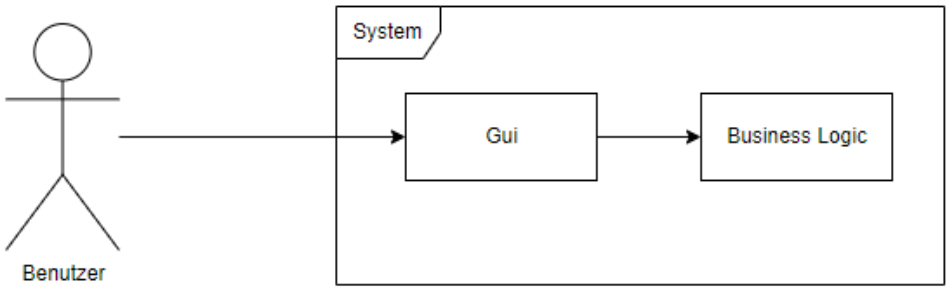

Abb. 1: Nutzerzugriff auf GUI

Alles beginnt damit, dass der/die Benutzer:in das Programm öffnet. Bei der Benutzung wird die GUI ausgegeben, mit welcher interagiert werden kann. Die GUI ist inhaltlich abhängig von der Business Logic, da dort interne Werte geändert werden. Diese Änderungen werden dann von der GUI erfasst und die GUI verändert sich dementsprechend auch.

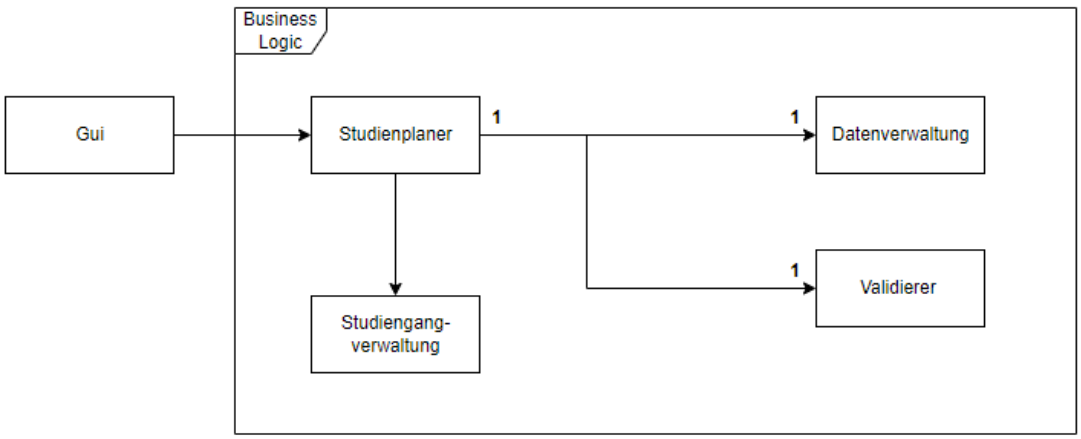

Abb. 2: GUI-zugriff auf Businesslogik

Hier wird der Zugriff der GUI auf die Businesslogik erläutert. Im Detail observiert die GUI die Klasse, welche den gesamten Studienplaner verwaltet. Diese Klasse hat Zugriff auf alle relevanten Daten, wie diese validiert werden und die Verwaltung des Studiengangs. Somit können alle Veränderungen der Business Logic von der GUI erfasst werden, obwohl diese nur auf eine Klasse zugreift.

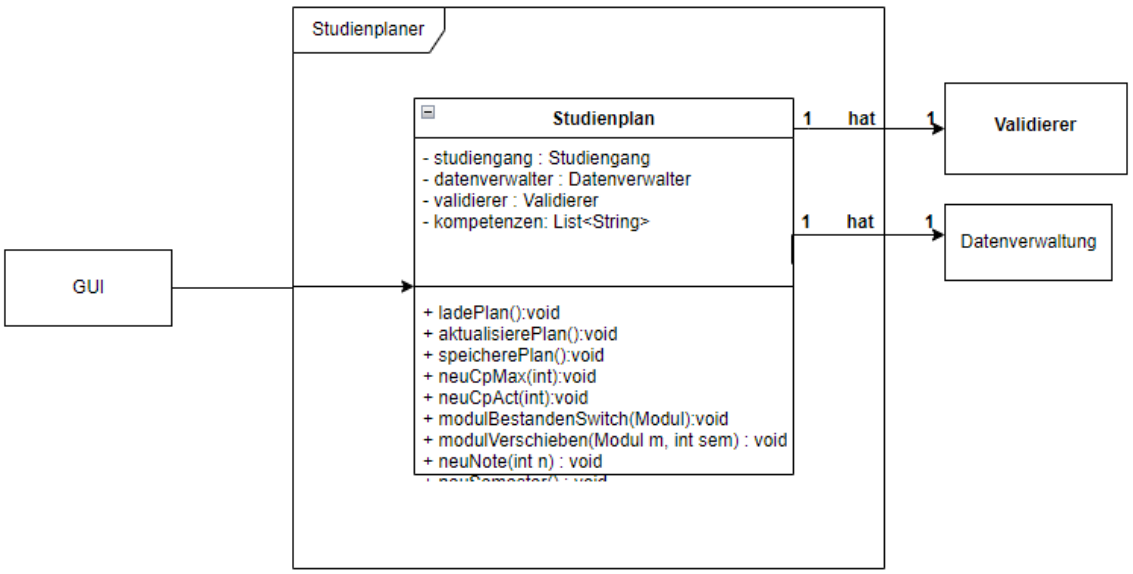

Abb. 3: Studienplaner

Die Studienplaner-Klasse verwaltet die gesamte Anwendung. Sie hat durch ihre Variablen Zugriff auf ihre Unterklassen, wodurch die Änderungen wie zuvor erwähnt an die GUI weitergegeben werden können. Auch wird hier der Plan geladen und gespeichert, als auch eine Aktualisierung angestoßen. Die angestoßenen Veränderungen im Plan, den CP oder dem Bestehen eines Moduls können auch wieder direkt von der GUI wahrgenommen und ausgegeben werden.

Im folgenden wird kurz auf die einzelnen Funktionen eingegangen.
Die Funktion ladePlan() stößt in der Datenverwaltung eine Methode an, um sich den Plan auslesen und erstellen zu lassen, um diesen ansehen zu können.
Mit speicherePlan() sorgt wiederum dafür, dass die Datenverwaltung alle Änderungen am Plan, also alle neuen Werte, in die Datei gespeichert werden.
Um neue Werte aus der Datei zu erhalten und den Plan aktuell anzuzeigen, wird aktualisierePlan() aufgerufen, der dies der Datenverwaltung mitteilt.
Die Funktion neuCpMax(int) bekommt die neue Cp Grenze als Integer mit und übergibt dies dem Validierer, der prüft, ob es eine gültige Eingabe ist.
Beim Füllen der Semester wird neuCpAct(int) dem Validierer übermittelt, dieser überprüft, ob die gesetzte Cp Grenze mit der aktuellen Menge an Cp in den Semestern übereinstimmt.
Zuletzt wird mit modulBestandenSwitch(Modul) im Validierer geschaut, ob ein Modul bestanden ist oder nicht und damit die Fortschrittsregel überprüft.

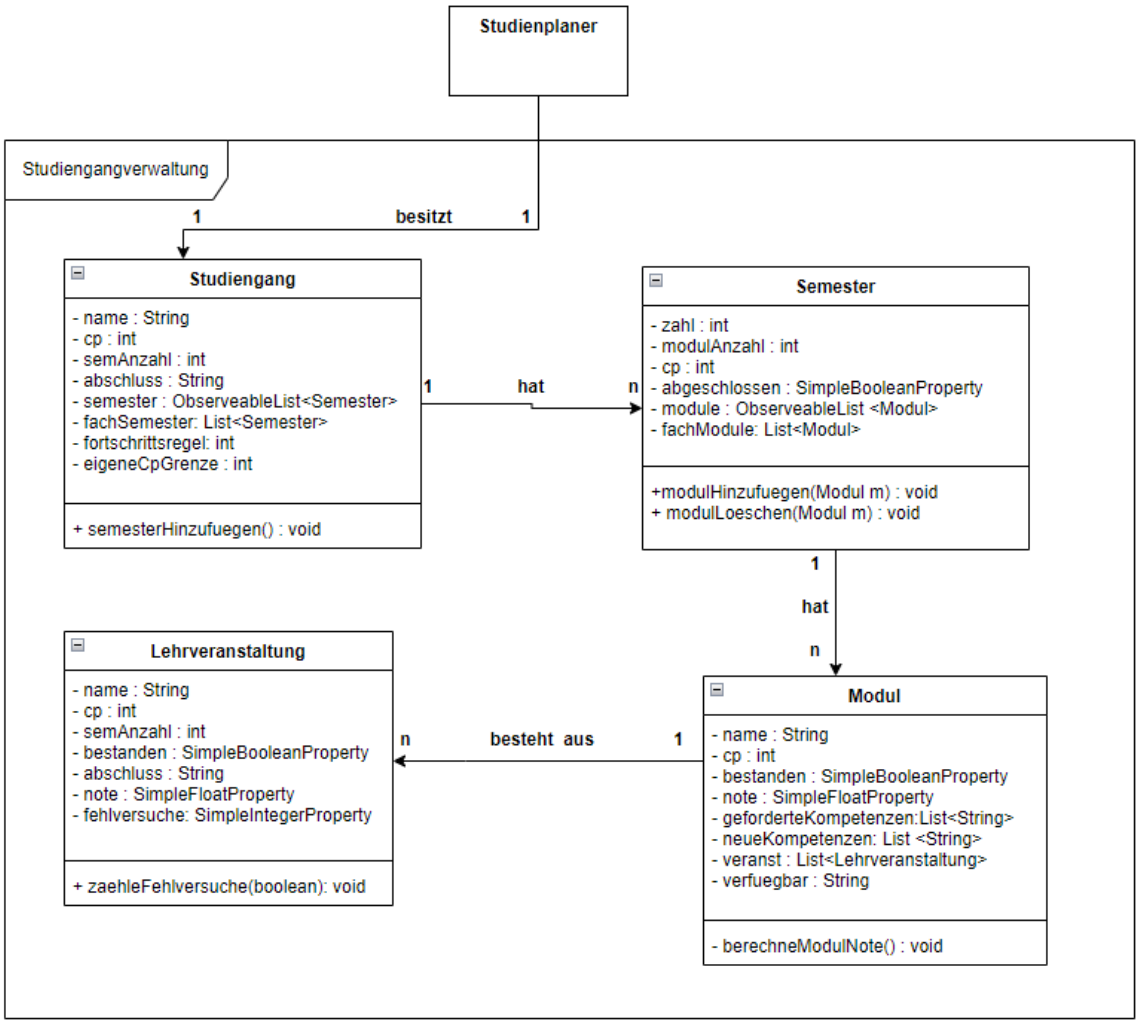

Abb. 4: Studiengangverwaltung

Unsere Studiengangsverwaltung besteht aus den Bestandteilen eines Studiums, beginnend mit einem Studiengang. Dieser enthält die Daten des jeweiligen Studiengangs in Form von dem Namen, CP, der Anzahl an Semestern, sowie Listen zu den zugeordneten Semester Objekten. Auch eine Variable zum Abschluss des Studiums ist vorhanden. Der Studiengang überprüft auch, ob die Fortschrittsregelung erfüllt ist. Um Semester hinzuzufügen, gibt es eine semesterHinzufuegen() Methode, die ein Semester an die bestehende Semesterliste hängt mit einer erhöhten Semesterzahl und einer leeren Modulklasse.
Da man eine Liste an Semester Objekten benötigt, gibt es eine Semester Klasse, die die Anzahl an Modulen, Listen zu den zugehörigen Modulen und auch eine CP Anzahl beinhaltet. Auch wird im Semester überprüft, ob es abgeschlossen wurde. Hier werden Module, die durch das DragAndDrop verschoben wurden den Funktionen mitgegeben und durch modulHinzufuegen(Modul m) bzw. moduLoeschen (Modul m) zur Liste hinzugefügt oder gelöscht.
Um das Semester mit den Modulen versorgen zu können, gibt es eine Modul Klasse. Sie beinhaltet den Namen, CP, neue und geforderte Kompetenzen und eine Liste von Lernveranstaltungen, da ein Modul aus mehreren Lehrveranstaltungen bestehen kann. Dazu benötigt das Modul noch einen Wert zum überprüfen, in welchen Semestern es verfügbar

ist. Das Modul kann die Gesamtnote eines Moduls berechen und bezieht dabei die Gewichtungen aus den Lehrveranstaltungen und die jeweiligen Noten ein.
Die Lehrveranstaltungen, die das Modul benötigt, sind gegliedert in einen Namen, CP, das Semester, in dem die Veranstaltung vorgesehen ist und, ob es bestanden wurde. Dazu benötigt es eine Fehlversuch Variable, um zu überprüfen, ob man bereits einen 3. Fehlversuche hat und somit exmatrikuliert wird.

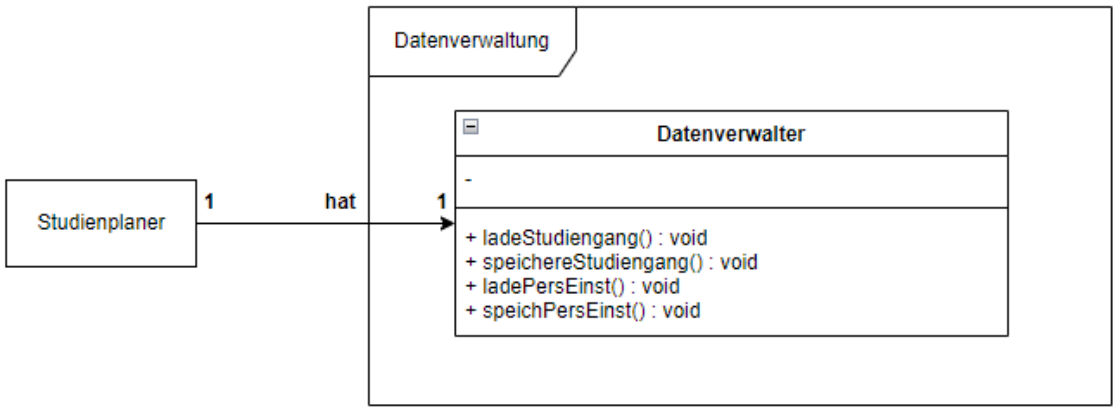

Abb. 5: Datenverwaltung

Die Datenverwaltung ist zur Verwaltung aller Daten und besteht aus einem Datenverwalter, der einen Studienplaner besitzt. Dadurch kann der Studiengang mit ladeStudiengang() geladen und mit speichereStudiengang() gespeichert werden. Gleiches gilt auch für die Einstellungen. Der Datenverwalter sorgt dadurch dafür, dass eine Datei eingelesen werden kann und Änderungen darin gespeichert werden.

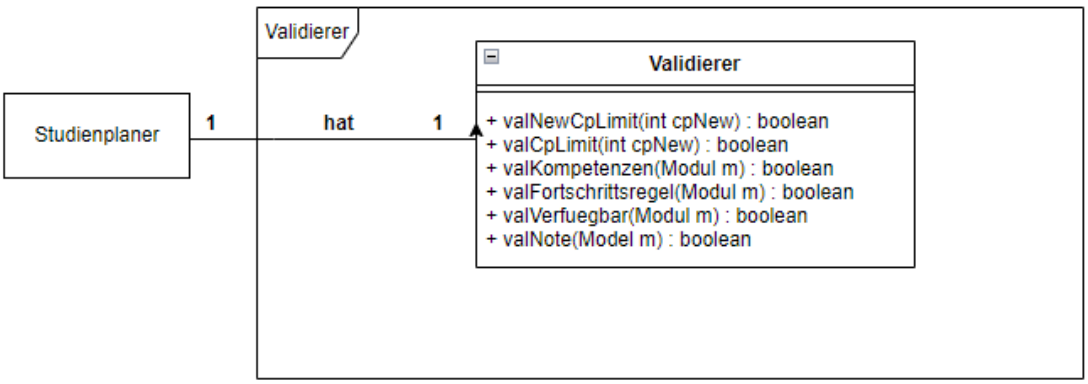

Abb. 6: Validierer

Der Validierer hat einen Studienplaner und überprüft an diesem, ob Vorgaben wie ein selbst gesetztes CP Limit eingehalten wurden. Dabei wird die derzeit erreicht CP Menge in dem Semester zusammen addiert und der Methode valCpLimit(int cpNew) mitgegeben. Alle Methoden geben true zurück, wenn die Bedingung erfüllt ist bzw. false, wenn nicht. Zudem überprüft er, ob das neu gesetzte CP Limit in Ordnung ist oder nicht, da es z.B. nicht negativ sein kann. Hier würde die neue Cp Grenze mitgegeben werden. Auch überprüft er, ob ein Modul im aktuellen Semester verfügbar ist mithilfe der valVerfuegbar(Modul m) Methode, die ein Modul mitbekommt und mithilfe valKompetenzen(Modul m), welche Kompetenzen im jeweiligen Modul vorliegen sollten. Die Methode valNote(Modul m) überprüft, ob die eingetragene Note im mitgegebenem Modul gültig ist.

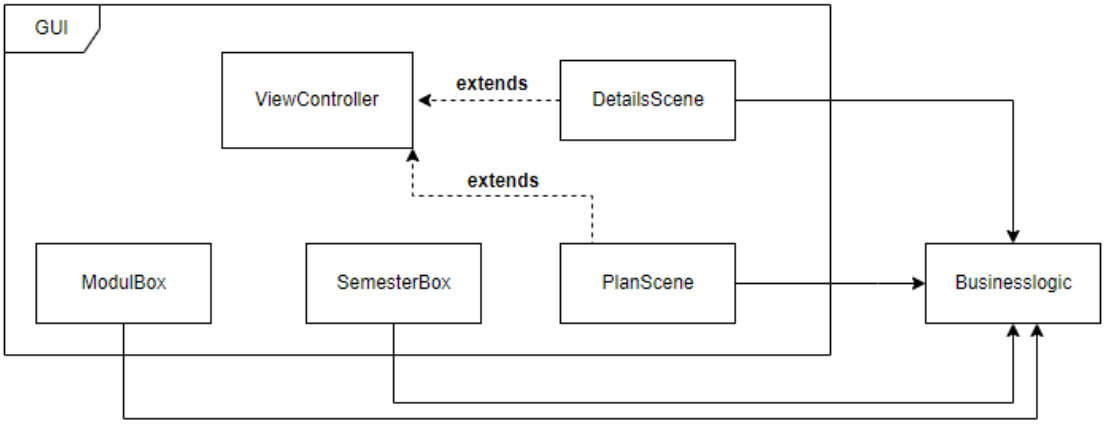

Abb. 7: GUI

Nun gehen wir näher auf die GUI ein. Wie vorher schon sichtbar, laden sich unsere GUI-Elemente alle benötigten Informationen aus der Businesslogik. Die GUI besteht grundlegend aus einer PlanScene, welche unsere gesamte Studienplan-Anwendung darstellt und einer DetailsScene, die alle wichtigen Informationen über ein Modul präsentiert.
Die PlanScene besteht dabei aus SemesterBoxen, die wiederum aus ModulBoxen aufgebaut sind.
Alle diese Szenen und Boxen erben die Oberklasse ViewController.

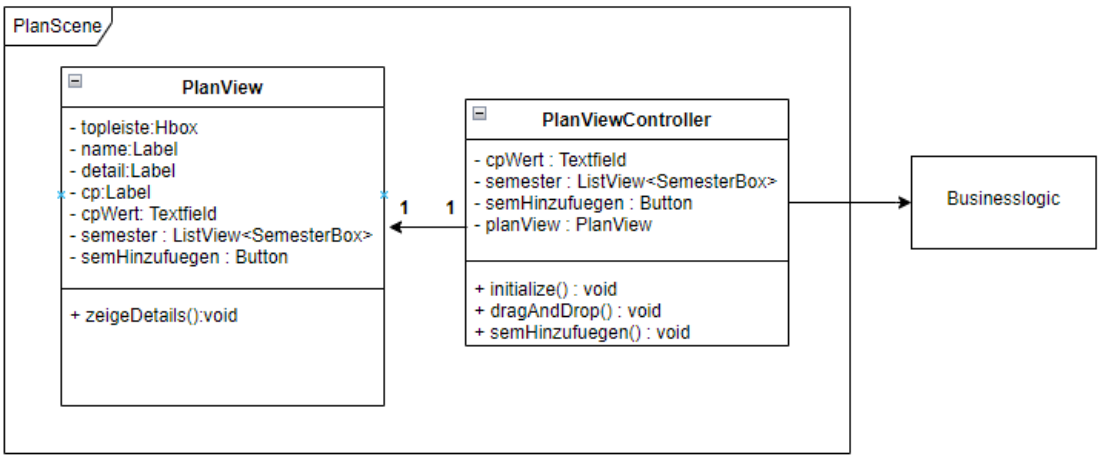

Abb. 8: PlanScene

Die PlanScene besteht aus einer PlanView, die unsere Darstellung übernimmt, sowie einem zugehörigen PlanViewController, der alle nötigen Informationen überwacht und an die Szene weitergibt.
Die PlanView stellt in der oberen Leiste die Elemente Name, Detail, CP und das CP Wertfeld nebeneinander dar. Darunter werden anfangs alle Semester aufgelistet, wie sie standardmäßig stattfinden. Bei Klick auf ein Modul, wird zeigeDetails() ausgelöst und die DetailScene aufgerufen.

Der PlanViewController beobachtet alle änderbaren Werte wie die CP Grenze und die SemesterBoxen. Zudem werden hier die dragAndDrop() Events auf Module verarbeitet. Mit der Funktion fuegeSemHinzu() werden mithilfe des Buttons semHinzufuegen Semester ergänzt.

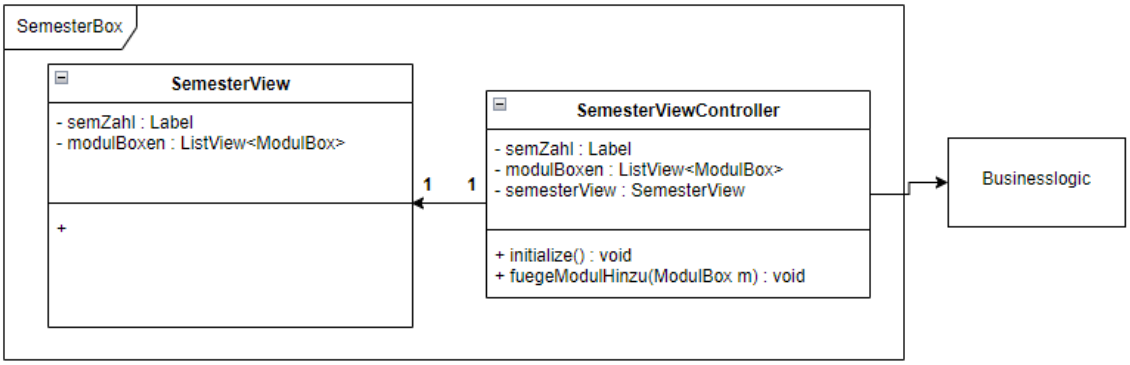

Abb. 9: SemesterBox

Die in der PlanScene enthaltene SemesterBox besteht aus einer SemesterView, die ein Semester darstellt und einem SemesterViewController, der alle Veränderungen beobachtet und passend an die SemesterView weitergibt.
Ein Semester setzt sich durch die Semesterzahl und durch nebeneinander aufgelistete ModulBoxen zusammen.

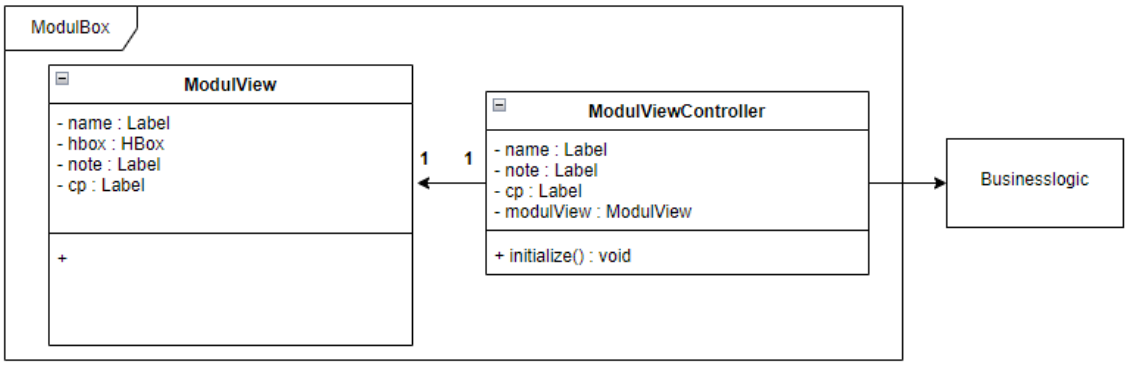

Abb. 10: Modulbox

Die in der SemesterBox befindlichen ModulBoxen bestehen aus einer ModulView, die ein Modul darstellt und einem ModulViewController, der die passenden Informationen an die ModulView mitteilt.
Das Modul wird durch den Namen und darunter in einer Zeile mit Note und CP zusammengestellt.

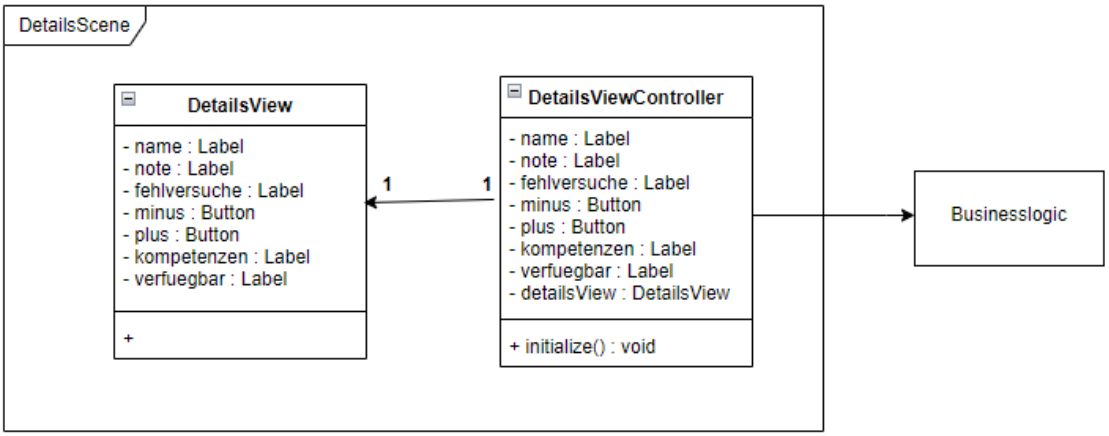

Abb. 11: DetailScene

Die DetailsScene besteht aus einer DetailsView, die alle wichtigen Informationen über ein Modul untereinander darstellt, sowie einem zugehörigem DetailsViewController, der alle nötigen Informationen überwacht und an die Szene weitergibt.

Laufzeitsicht

In der Laufzeitsicht soll dargestellt werden, wie die vorhin beschriebenen Bausteine zur tatsächlichen Laufzeit miteinander arbeiten. Aus den folgenden Diagrammen soll hervorgehen, wann und wie diese Bausteine erzeugt, die enthaltenen Methoden verwendet und die Zugriffe daraufhin beendet werden.

Sequenz 1: Setzen einer eigenen Cp Grenze

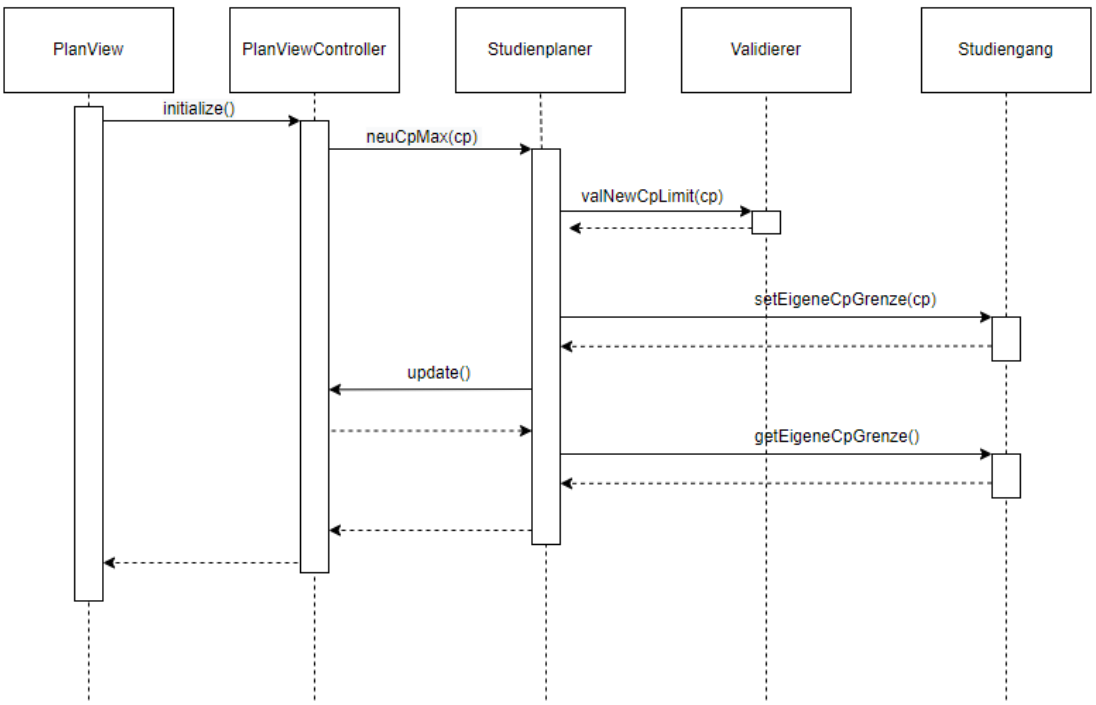

Abb. 12: Sequenz 1

Ein Listener reagiert auf die Änderung im Textfeld der maximalen CP-Angabe und stößt die Methode “neueCpMax()” der Studienplanerinstanz an. Dieser nutzt seinen Validierer und überprüft zunächst durch “valNewCpLimit()”, ob die Werteänderung nicht irgendwelche Kriterien verletzt, da beispielsweise negative Werte nicht zulässig sein sollen.
Wenn dieser sein OK gibt, wird “setEigeneCpGrenze()” der Studienganginstanz angestoßen. Die Methode verändert eine observierbare Property, auf die ein Listener im Controller reagiert und den geänderten Wert für die View übernimmt.

Sequenz 2: Hinzufügen eines Moduls

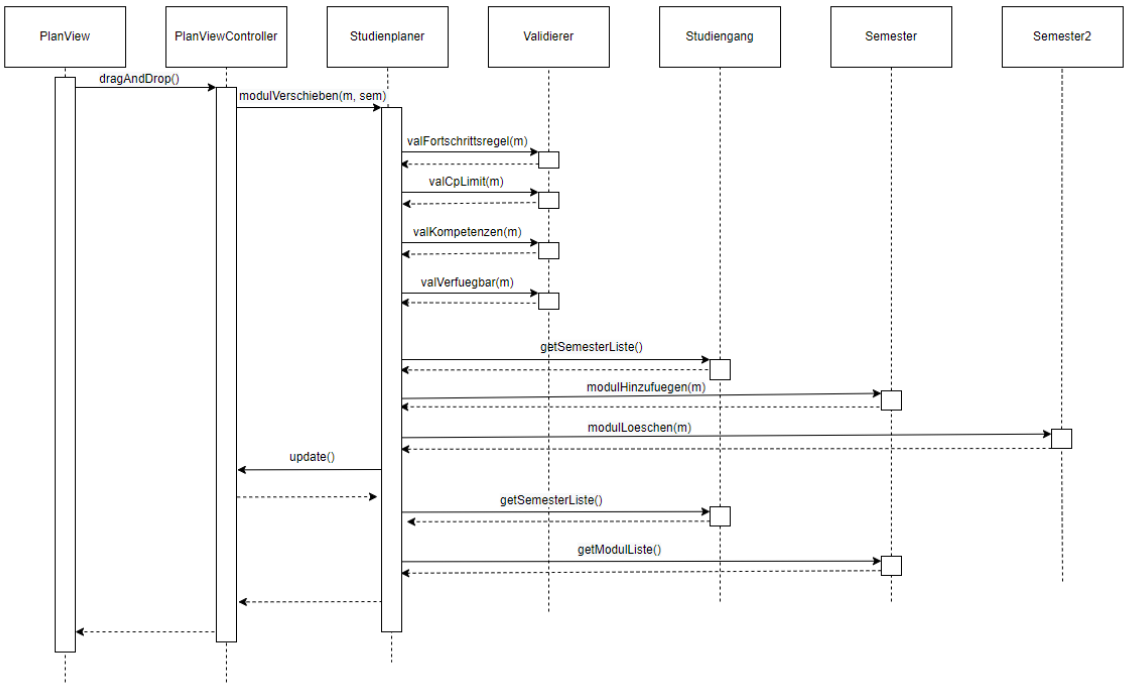

Abb. 13: Sequenz 2

Im obigen Diagramm soll ein Modul unter Berücksichtigung der Fortschrittsregel in ein anderes Semester verschoben werden. Auf der PlanView (GUI) wird von dem/der User:in via Drag and Drop der Vorgang angestoßen. Der Controller dieser View vermittelt dann der Business Logic, dass das Modul “m” in das Semester “sem” verschoben werden soll. Der Studienplaner, welcher Zugriff auf die gesamte Business Logic hat, fragt zunächst beim Validierer nach, ob diese Aktion unter Berücksichtigung der Fortschrittsregel und ander Parameter erlaubt ist. Ist dies der Fall, wird im alten Semester das Modul aus der Liste entfernt und in der Neuen hinzugefügt. Die GUI reagiert auf diese Änderung und zeigt den neuen Plan an.

Sequenz 3: Setzen der Note einer Lehrveranstaltung

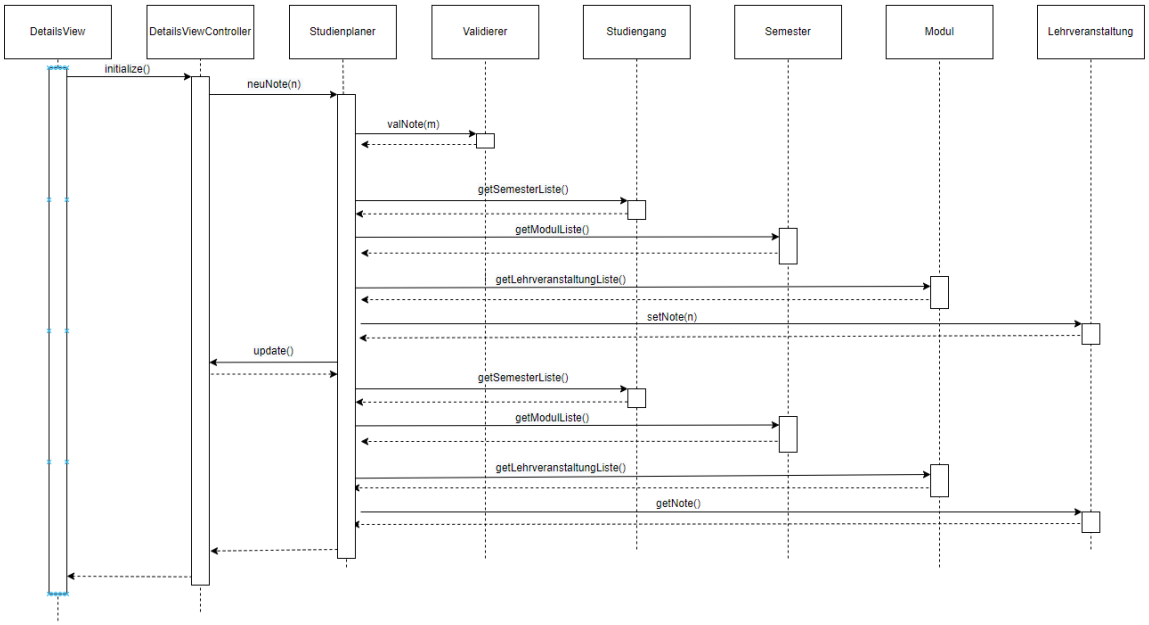

Abb. 14: Sequenz 3

Um die Note im Studienplaner einzutragen, muss man sich auf der Detail-Ansicht eines Moduls befinden. Hier kann dann in der passenden Lehrveranstaltung in einem Textfeld die Note eingegeben werden.
Durch die Eingabe der Note reagiert die GUI auf die Änderung und stößt die initialize-Methode vom DetailsViewController an. Dieser überprüft nun erst den Wert im Validierer über die Methode valNote. Danach wird über Getter die neue Note über Studiengang, Semester, Modul an Lehrveranstaltung, wo der neue Wert gespeichert wird. Zuletzt aktualisiert sich die View wieder.

Sequenz 4: Hinzufügen eines Semesters

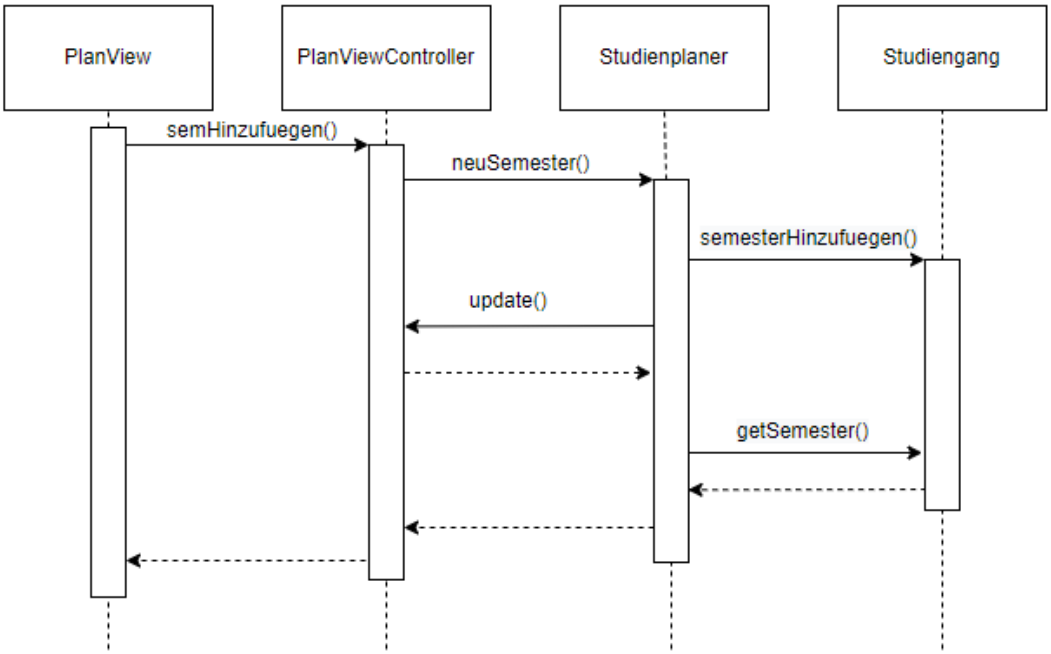

Abb. 15: Sequenz 4

Wenn der Benutzer ein neues Semester anlegen möchte, um den Studienplan zu entzerren, muss er auf einen Button klicken. Das löst die Methode semHinzufuegen im PlanViewController aus, die wiederum über den Studienplaner ein neues Semester in der Semesterliste des Studiengangs hinzufügt.

Versionsverzeichnis

Version	Autor	Änderung	Datum
1	Kevin Emunds	Erstellung des Dokuments	16.06.2022
2	Astrid Klemmer	Struktur verbessert und ergänzt	24.06.2022
3			06.07.2022