Institut für Mathematik
Fakultät II – Mathematik und Naturwissenschaften
Technische Universität Berlin

Praktikumsbericht

Betreuer: Gerald Lach
Unternehmen: MathPlan GmbH

5. Juli 2022

Inhaltsverzeichnis

1 Das Unternehmen

Die MathPlan GmbH mit Sitz in Berlin erstellt und vertreibt seit 2013 Software für verschiedene Bereiche der Hochschulverwaltung [5]. Hauptanwendung ist dabei die automatisierte Stundenplanung von Tutorien, Klausuren und Lehrveranstaltungen. Weitere Aufgaben, die von der Firma abgedeckt werden, sind die Bereitstellung von Vorlesungsverzeichnissen, die Kurseinschreibung von Studierenden oder, seit der Corona-Pandemie, die Anwesenheitserfassung bei Präsenzveranstaltungen. Bereits 2002 entsteht an der Technische Universität Berlin ein Projekt, bei dem mithilfe von mathematischen Optimierungsverfahren die bis zu 2.300 Studierenden einer mathematischen Lehrveranstaltung nach Angabe von Präferenzen in parallel stattfindende Tutorien eingeteilt werden können. An der TU Berlin wird dafür eine Webanwendung unter dem Namen *Moses* (Mathematisch Optimale Stundenplan-Erstellungs-Software) veröffentlicht [4]. Später folgen weitere Anwendungen wie die automatisierte Planung von Klausuren und Lehrveranstaltungen, sowie die Einführung der Software an andere Universitäten, wofür ein eigenes Unternehmen gegründet wird. Heute vertreibt und betreut die MathPlan GmbH Software-Produkte an verschiedenen Universitäten und Hochschulen in ganz Deutschland, darunter an der Charité Berlin, der Rwth Aachen und der Technischen Universität München. Aktuell laufen zudem Einführungsprojekte an sechs weiteren Hochschulen in Deutschland und Österreich. MathPlan hilft damit bisher jedes Semester, Klausuren oder Lehrveranstaltungen für über 130 000 Studierende zu planen, was ca. 4,7% der Studierenden in Deutschland betrifft [2]. Vor allem die Planung in medizinischen Studiengängen ist vergleichsweise aufwändig, da dort die Veranstaltungen für praktische Übungen in Kleingruppen und vierwöchigen Themenblöcken ausgelegt sein müssen, und bei der Planung die Verfügbarkeiten der einzelnen Institute berücksichtigt werden müssen. Dies führt z.B. zu ca. 32 000 zu planenden Einzelterminen jedes Semester an 81 Kliniken und 51 Instituten allein an der Charité Berlin [5].

Mein Praktikum beim oben beschriebenen Unternehmen habe ich vom 16.2.2022 bis zum 14.3.2022 in dessen Bereich für Software-Entwicklung absolviert. Dabei bestand meine Aufgabe darin, im Austausch mit anderen Entwickler:innen der Firma, Komponenten der Moses-Software zu erweitern, wobei ich überwiegend die *Schnittstellen* der Software kennengelernt habe. Neben kleineren Aufgaben zur Tutorienplanung habe ich darüber hinaus zusätzliche Kenntnisse in Java, Html und anderen relevanten Frameworks wie JPA erworben und einen kleinen Einblick in das Optimierungs-Modell zur Stundenplanung erhalten (ausgeführt im nachfolgenden Kapitel). Insgesamt arbeiten an Moses zur Zeit vorwiegend Informatiker:innen, aber auch Studierende und Absolvent:innen aus den Bereichen Mathematik, Ingenieurswissenschaften und Volkswirtschaftslehre.

2 Beschreibung meiner Tätigkeiten und Erläuterung der Resultate

MathPlan vertreibt mit Moses kein vollständiges *Campus-Management-System*. Letzteres ist ein Programm, mit dem alle oder zumindest ein Großteil der Hochschul- (wie Raum- oder Stundenplanung) und Studierendenverwaltung (wie Kurseinschreibung oder Prüfungsverwaltung) vorgenommen werden können. Daher liegen die Daten, die z.B. für die Berechnung von Klausurterminen nötig sind, (außer bei der TU Berlin) in separaten Systemen, von wo sie zunächst nach Moses exportiert werden müssen, bevor eine Zuweisung stattfinden kann. Diese Daten umfassen Lage und Größe der verfügbaren Räume, Angaben zu den Dozierenden und vor allem Informationen über Länge, Häufigkeit und Inhalt der Lehrveranstaltungen oder Klausuren. Hinzu kommt, dass an den über 400 Hochschulen in Deutschland (vgl. [3]) unterschiedliche Campus-Management-System verwendet werden, die die oben genannten Daten jeweils unterschiedlich abbilden. Das bedeutet, dass vor der Berechnung von Veranstaltungs-, Klausur- oder Tutorienplänen zunächst die betreffenden Daten einer Hochschule nach Moses importiert und anschließend zurück zur Hochschule exportiert werden müssen (siehe auch Abbildung 1). Dabei ist in der Regel eine Konvertierung der Daten nötig, um deren Verknüpfungen (z.B. eine Lehrveranstaltung mit einem zugehörigen Raum) in einem Format zu erhalten, das Moses oder die betreffende Hochschule weiterverwenden können. In MathPlan wird diese Konvertierung in einem Proxy-Server vorgenommen. Informationen, die von einer externen Hochschule kommen werden dabei von diesem, oft individuell konfigurierten, Proxy entgegengenommen, konvertiert und danach zur automatisierten Planung an der Moses-System weitergeleitet. Vor allem die unterschiedlichen Abbildungen von Studiengängen und Lehrveranstaltungen bei den verschiedenen MathPlan-Kunden erfordern also eine gut funktionierende Schnittstelle, über die Daten im- und exportiert werden können. Einen Großteil meines Praktikums habe ich im Schnittstellen-Team von MathPlan verbracht, das

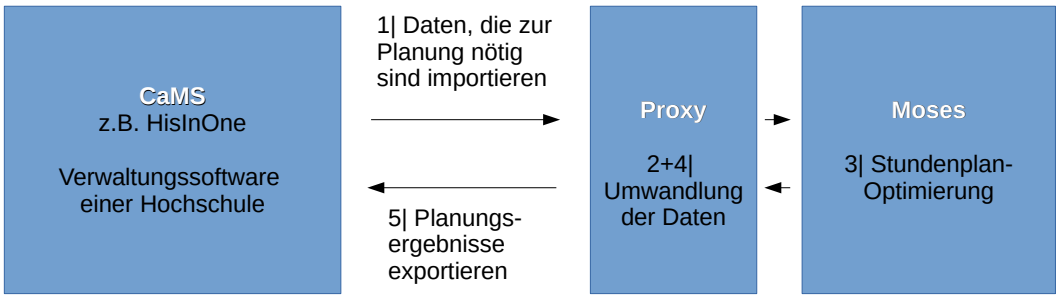


Abbildung 1: Vereinfachte Darstellung des Im- und Export nach Moses

sich damit befasst, die genannte Schnittstelle zu warten, zu erweitern oder für neue Campus-Management-Systeme vorzubereiten (siehe Kapitel 2.2).

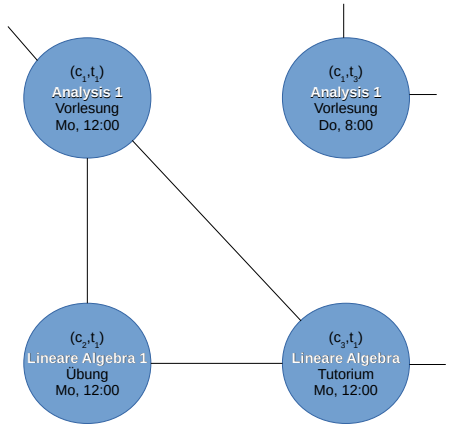
Um die folgenden Erläuterungen zu vereinfachen, bezeichne ich die Software, die in der Mathplan GmbH entwickelt wird, in diesem Praktikumsbericht nur als *Moses*, auch wenn es sich dabei streng genommen um eine Sammlung von Komponenten handelt, die in unterschiedlichen Kombinationen an verschiedenen Hochschulen und unter unterschiedlichen Namen (wie z.B. „carpe diem“ an der Rwth Aachen) eingesetzt werden. Der Großteil der Anwendung bei MathPlan ist in Java programmiert.

2.1 Mathematische Stundenplan-Optimierung

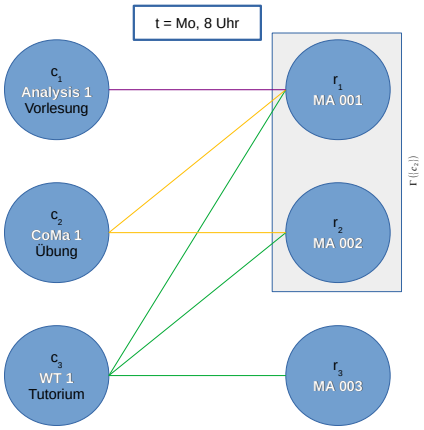
Wie erwähnt ist die wichtigste Anwendung von Moses, automatisiert Räume und Zeiten für Lehrveranstaltungen, Klausurtermine oder Tutorien zu planen. Dazu müssen, wie in Kapitel 1 bereits beschrieben, einer oft fünfstellige Anzahl von Einzelterminen je ein fester Raum und eine feste Zeit zugewiesen werden. Im Folgenden wird kurz erläutert, wie sich dieses NP-vollständige Problem als ganzzahliges (lineares) Programm (IP) formulieren lässt, das später von einem IP-Löser gelöst werden kann. Wir folgen dafür der Beschreibung in [6] und den Erklärungen, die ich im Betrieb erhalten habe.

Sei $\mathcal{T}$ die Menge aller Zeit-Slots, $\mathcal{C}$ die Menge aller Kurse und $\mathcal{R}$ die Menge aller Räume, z.B. an einer bestimmten Hochschule. Die Menge der Zeit-Slots $\mathcal{T}(c)$, die zu einem bestimmten Kurs c zur Verfügung stehen ist bekannt, ebenso wie eine Gewichtung $prio(c,t) \geq 0$ für alle t in $\mathcal{T}(c)$, die angibt, wie favorisiert eine Zuordnung (c,t) ist. Letzteres kann beides z.B. von den Dozierenden angegeben werden, wenn sie ihre Lehrveranstaltungen in der Webanwendung eintragen. Je kleiner die angebende Priorität ist, desto mehr ist eine Kombination gewünscht. Außerdem bekannt ist die Anzahl von Veranstaltungen $\ell(c)$ eines Kurses, die auch der Zahl der für ihn benötigten Zeit-Slots entspricht. Neben der beschriebenen Priorisierung existieren weitere Nebenbedingungen. Darunter sind harte Bedingungen, z.B. dass ein Raum nicht mehreren Kursen gleichzeitig zugewiesen sein darf oder Dozierende nicht in mehreren Kursen gleichzeitig unterrichten können, und weiche Bedingungen wie Raumkapazitäten, Ausstattungswünsche oder eine weite Verteilung der Veranstaltungstermine eines Kurses in einer Woche. Potentiellen Zeit-Konflikte lassen sich nun mithilfe eines Graphen $G_{konf} := (V_{konf}, E_{Konf})$ repräsentieren, wie in Abbildung 2a schematisch darstellt ist. Dabei steht jede Ecke (c,t) für eine Kombination aus einem Kurs und einem Zeit-Slot. Zwei Ecken sind durch eine Kante verbunden, wenn die zugehörigen Buchungen nicht miteinander zu vereinbaren sind, z.B. weil zwei Kurse zur gleichen Zeit stattfinden würden, die von einer Studierendengruppe beide besucht werden müssen. Schließlich benutzen wir eine binäre Variable $x_{c,r,t}$ um anzuzeigen, ob eine Kurs-Raum-Zeit Kombination gebucht wird oder nicht.

Die obige Modellierung würde bereits ausreichen, ein ganzzahliges Programm zu formulieren, mit dem die Kurse Räumen und Zeiten zugeordnet werden können. Um die Formulierung noch zu vereinfachen ist es nun möglich, die Raumzuordnung von Kursen implizit anzugeben und die Variable x so von drei Dimension auf zwei zu reduzieren. Dazu betrachten wir die Stundenplanung für jeden verfügbaren Zeit-Slot einzeln. Bezeichne $\mathcal{C}_t$ für alle $t \in \mathcal{T}$ die Menge der Kurse, die im Zeit-Slot t stattfinden können, und $\mathcal{R}_t$ die Menge der Räume, die für mindestens einen der Kurse in



(a) Konfliktgraph G_{konf} für die Stundenplanung, skizziert anhand von Beispielkursen c_i und Zeit-Slots t_i



(b) Bipatiter Buchungsgraph G_t für ein festes t, skizziert anhand von Beispielkursen c_i und Räumen r_i. Die Umgebung von c_2 ist hellgrau hervorgehoben.

Abbildung 2: Schematische Darstellung der beiden für die Optimierung verwendeten Graphen

$\mathcal{C}_t$ zur Verfügung stehen. Wir können nun alle infrage kommenden Kurse und Räume zu einem Zeit-Slot t mit einem ungerichteten bipartiten Graphen $G_t := (\mathcal{C}_t \cup \mathcal{R}_t, E_t)$ darstellen. Ein Kurs $c \in \mathcal{C}_t$ und ein Raum $r \in \mathcal{R}_t$ sind in diesem Graphen genau dann verbunden, wenn c in r stattfinden kann. Eine schematische Darstellung dieses Graphen findet sich in Abbildung 2b. Sei nun ferner $\Gamma(U) := \{i \in \mathcal{C} \cup \mathcal{R} | j \in U, (i,j) \in E\}$ die Menge aller benachbarten Knoten von $U \subseteq \mathcal{C}$. Dann gibt es laut dem in [6] verwendeten Satz von Hall (Heiratssatz) genau dann eine eindeutige Zuordnung von Kursen und Räumen, wenn $|\Gamma(U)| \geq |U|$ für alle $U \subset \mathcal{C}$ gilt. Hierbei bezeichnet jeweils $|\cdot|$ die Mächtigkeit der eingesetzten Mengen. Damit lässt sich das Optimierungsproblem zur Stundenplanung formulieren als

$$
\begin{aligned}
\min \sum_{(c,t) \in V_{\text{konf}}} & \mathit{prio}(c,t) \cdot x_{c,t} & & \\
\text{s.t.} \quad \sum_{t \in T(c)} & x_{c,t} = \ell(c) & & \text{für alle } c \in \mathcal{C}, \\
\sum_{c \in U} & x_{c,t} \leq |\Gamma(U)| & & \text{für alle } U \subseteq \mathcal{C}, t \in \mathcal{T}, \\
x_{c_1,t_1} + x_{c_2,t_2} & \leq 1 & & \text{für alle } \big((c_t,t_1),(c_2,t_2)\big) \in E_{konf} \\
x_{c,t} & \in \{0,1\} & & \text{für alle } (c,t) \in V_{konf}.
\end{aligned}
$$

Anders als vorher in $x_{c,r,t}$, taucht hier die Zuordnung von Kursen und Räumen nicht mehr direkt auf. Sie ergibt sich nach der obigen Minimierung jedoch als Lösung eines einfachen Matching-Problems. In der Praxis muss die (mehrstündige) Stundenplan-Berechnung oft einige Male wiederholt werden, um triviale Lösungen auszuschließen oder übersehene Nebenbedingungen nachträglich hinzuzufügen. Diese Versuche werden in der Moses-Oberfläche als verschiedene *Szenarien* angezeigt, aus denen das gewünschte schließlich ausgewählt werden kann.

2.2 Export von Veranstaltungen in ein CaMS von HisInOne

Aktuell plant MathPlan eine neue Universität „anzuschließen“, die ein Campus-Management-System der HIS Hochschul-Informations-System eG (im Folgenden kurz *HIS*) verwendet. Wie oben erwähnt, ist es dafür nötig Planungsdaten, z.B. Orte und Zeiten von Veranstaltungen, aus Moses in das HIS-System zu exportieren. Während meiner Zeit bei MathPlan habe ich geholfen, den Export von Klausur- und Veranstaltungsdaten zu einem HIS-System aufzubauen, wobei die Daten vor dem Export in ein Format gebracht werden, dass vom Zielsystem verarbeitet werden kann.

Angenommen die erforderlichen Daten für die Planung wurde bereits in Moses eingespielt. Dann können mit diesen, falls nötig in mehreren Iterationen, die Raum- und Dozierendenzuteilungen berechnet werden. Tabelle 1 gibt einen

Kurs	**LvVorlage**	**Lehrveranstaltung**	**Veranstaltungsplanung**
Name	Veranstaltungsformat	Dozent:in	Bestätigt
LP	LP	Semester	Bearbeiter:in
SWS	SWS	Teilnahmebedingung	Ansprechpartner:in
Organisationseinheit	Turnus (jedes Sem usw.)	Anmeldezeitraum	
	Sprache	Vorgänger	
		Name	
		Organisationseinheit	

Termingruppe	**Terminplanung**	**Raumkriterium**	
Anzahl Studierende	Name	Min. Sitzplätze	
Name	Dauer	Max. Sitzplätze	
Vorr. Teilnehmendenanzahl	Vorlaufzeit	Spacingtype	
Semesterwochen-Pattern	Nachlaufzeit	kein Raum benötigt	
	Startzeit		
	Alternativer Wochentag		
	Alternative Startzeit		
	Raumkriterium		

Tabelle 1: Entiäten und Felder für die Lehrverantstaltungsplanung (nicht alle Felder sind hier abgebildet)

Überblick über die für die Planung verwendeten Daten. Nachdem das automatische Planungswerkzeug in Moses eine valide Lösung unter Berücksichtigung (manuell) angegebener Constraints gefunden hat, wird diese Lösung in der Datenbank gespeichert, in einem Format, das MathPlan zur Abbildung der Veranstaltungs-, Klausur- oder Tutoriendaten verwendet. Dieses Format ist in Abbildung 3 am Beispiel der Vorlesung „Analysis 1“ dargestellt.

Sind die Veranstaltungen geplant, können Sie in das His-System exportiert werden. Dazu müssen zunächst die erforderlichen *Buchungsgruppen* aus der Datenbank geladen werden. Sie repräsentieren jeweils regelmäßige Termine (vgl. Abbildung 3) und beinhalten die wesentlichen Ergebnisdaten aus der automatischen Stundenplanung. Mehrere Buchungsgruppe zu einer Veranstaltung werden beispielsweise auch bei den Tutorien nötig, wo mehrere Termine (mit dem gleichen Inhalt) parallel, aber für immer die gleiche Untergruppe von Studierenden stattfinden. Das Auslesen der Buchungsgruppen aus der Datenbank geschieht in dem von MathPlan verwendeten Modell-View-Controller-Konzept mit einer Java-Klasse, die eigens zu diesem Zweck erstellt wurde (den Großteil ihrer Methoden jedoch von einer übergeordneten erbt). Mithilfe des Frameworks *JPA* (Jakarta Persistence API) kann ohne viel Aufwand ein Java-seitiges Abbild jeder Tabelle der relationalen Datenbank, die an Moses angeschlossen ist, erstellt werden. So ist es leicht möglich, die gewünschten Buchungsgruppen, z.B. aus einem festen Semester, performant aus der Datenbank zu lesen und als Liste von Java-Objekten an die sogenannte *View-Klasse* zu übergeben, von wo aus sie weiterverwendet werden können. In der View können allgemein Daten verarbeitet werden, um sie anschließend auf einer Html-Seite anzuzeigen. In unserem Fall, werden die gesammelten Buchungsgruppe von der View direkt an eine Provider-Klasse weitergereicht, die Teil der bereits bestehenden Export-Infrastruktur von Moses ist. Jede Entität, d.h. jede Datenart, wie z.B. „Buchungsgruppe“, „Raum“, „Lehrveranstaltung“, „Semester“ usw., bekommt eine eigene Provider-Klasse, in der geregelt ist, welche Felder der Entität (oder verknüpfter Entitäten) in welchem Format später an das Zielsystem geschickt werden sollen. Anschließend werden die so bereitgestellten Objekte in ein JSON oder XML geschrieben und mit einem Post-Request an den oben erwähnten Proxy geschickt.

Dem Proxy werden neben den Veranstaltungs- oder Klausurdaten im Header des Requests auch weitere Informationen übergeben, wie der Art der gesendeten Entitäten (z.B. „Buchungsgruppe“) und der Adresse des Zielsystems, an das die Daten nach der Konvertierung gesendet werden sollen. Im vorliegenden Fall entspricht dieses Zielsystem dem an einer Hochschule eingesetzte Campus-Management-System von HisInOne. Das gesendete JSON oder XML wird im Proxy zunächst deserialisiert, d.h. wieder auf ein Java-Objekt abgebildet (analog zu dem, das in der Providerklasse der WebApp erstellt wurde), das im Folgenden leichter ausgelesen werden kann. Eine eigene Konverter-Klasse für jeden Datentyp nimmt diese neuen Java-Objekte entgegen und befüllt damit wiederum Objekte, die der Kurs-Abbildung durch das HIS-System entsprechen. Die Implementation dieser Umwandlung benötigt hier die meiste Zeit, obwohl die Konverter-Klasse verschiedene Funktionalitäten wie Fehlerbehandlung, zusätzliche Kommunikation mit dem Ziel-

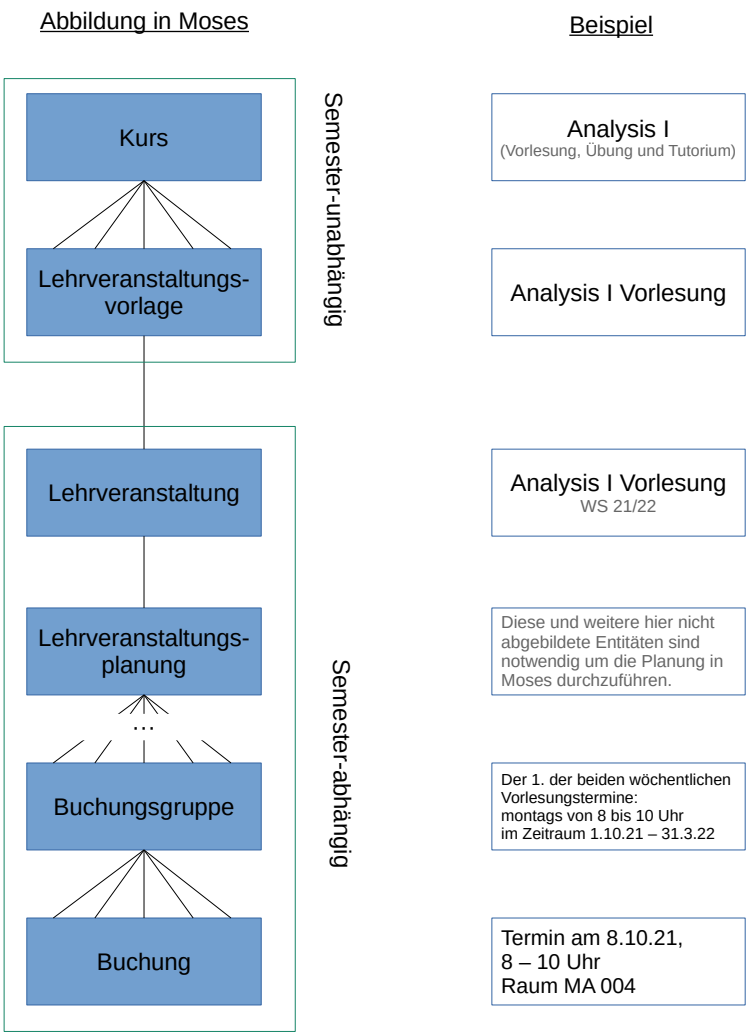


Abbildung 3: Vereinfachte Darstellung der Lehrveranstaltungsabbildung in Moses

system (die in manchen Fällen notwendig ist) oder andere mehrmals verwendete Methoden von einer allgemeineren Klasse erben kann. Denn bis heute gibt es (trotz des Bologna-Prozesses) weder europa- noch deutschlandweit eine einheitliche Abbildung von Hochschulkursen und Studiengängen (obgleich z.B. in [1] entsprechende Vorschläge gemacht wurden).

Die Lehrveranstaltungsabbildung bei Moses und His haben zunächst gemeinsam, dass alle Veranstaltungen als Semester-abhängige „Lehrveranstaltungsvorlagen“ gespeichert werden, die je auf eine allgemeinere, Semester-unabhängige Entität verweisen. So findet beispielsweise die Veranstaltung „Biologische Chemie I“ jedes Wintersemester statt und behält dabei meistens wesentliche Eckpunkte wie Titel, zuständigen Lehrstuhl, ECTS und Lerninhalte bei. Wechselnde Informationen wie Dozent:in, Raum oder maximale Anzahl der Teilnehmenden liegen in der Semester-abhängigen Entität „Lehrveranstaltung“ und entsprechenden untergeordneten Entitäten in der Moses-Datenbank vor (siehe Abbildung 3). Diese untergeordneten und übergeordneten Entäten sind es, bei denen sich His und Moses unterscheiden. So werden bei Ersterem die Semester-unabhängigen Veranstaltungen zu unterschiedlichen Einheiten wie *Veranstaltungsgruppen*, *Modulen* oder *Teilmodulen* zusammengefasst, während bei Moses zusätzliche strukturierende Elemente zwischen den Semester-abhängigen Veranstaltungen und den Buchungsgruppen existieren. Auch wird hier z.B. ein Raum einer einzelne Buchungen zugeordnet, während dies bei His für ganze Terminserien geschieht. Der letztgenannte Unterschied sorgt dafür, dass beim Export von Buchungsgruppen von Moses nach His im Proxy nicht für jede Moses-Buchungsgruppe eine äquivalente His-Terminserie angelegt werden kann, sondern stattdessen pro Buchungsgruppe und Raum eine eigene Terminserie an das HisInOne-System gesendet wird. Desweiteren unterscheiden sich die Abbildungen beispielsweise darin, dass die His-Terminserie das zugehörige Semester direkt referenziert, während in Moses diese Information erst in der Lehrveranstaltung steht. Daher ist es beim Senden der Buchungsgruppen an den Proxy untere anderem notwendig, zusätzlich Felder aus der übergeordneten Lehrveranstaltung mitzugeben, um im Proxy eine His-Terminserie zu erstellen und deren Felder setzen zu können.

Um die Planungsergebnisse vollständig zu übermitteln, müssen zudem auch die einzelnen Termine, die Moses-*Buchungen*, umgewandelt und versendet werden. Dies geschieht analog zum oben beschriebenen Export von Buchungsgruppen. Nicht in Abbildung 3 enthalten ist die Moses-Entität *Klausurplanung*, die wie die Lehrveranstaltungsplanung auf eine Lehrveranstaltung verweist, und mit der vor allem Zeit und Ort von Klausuren geplant und gespeichert werden können. Wie oben liegen die Ergebnisse der Klausurplanung wieder als Buchungsgruppen und Buchungen vor, wobei sich die Felder mancher Entitäten, auf die diese verweisen, sowie die Ziel-Entitäten auf His-Seite im Vergleich zum obigen Fall unterscheiden. Der Export von Klausurplanungsergebnissen funktioniert daher prinzipiell analog zum oben beschriebenen *Lehrveranstaltungsplanungsergebnisexport*. Die Provider- und Konverter-Klassen der beiden Fälle unterscheiden sich jedoch teilweise erheblich.

Die fertigen His-Terminserien werden schließlich mittels *Soap* (Simple Object Access Protocol) an den passenden Endpunkt von HisInOne gesendet, der mit *Wsdl* angesprochen werden kann. Dabei handelt es sich um ein standardisiertes Protokoll, um Daten zwischen zwei Endpunkte auszutauschen. MathPlan verwendet bei den meisten seiner Kunden *REST*, eine verbreitete Alternative zur Kommunikation mit Soap. Wie oben kurz erwähnt, muss bei allen Schritten zwischen dem Auslösen des Exports in der Weboberflächen von Moses bis zur Meldung (in selbiger), dass der Export erfolgreich verlaufen ist, dafür gesorgt werde, dass unerwartet auftretenden Fehler abgefangen werden. Dabei existieren bereits Mechanismen, die dafür sorgen, dass die Webanwendung und der Proxy bei kleineren Fehlern nicht vollständig abstürzen und nutzer:innenfreundliche Dialoge anzeigen. Doch gerade bei einem Export von mehreren tausend Dateneinträgen kann es leicht passieren, dass Felder oder Referenzen bei manchen Einträgen fehlen. In solchen Fällen ist es am sinnvollsten, den Vorgang nicht bei jedem Fehler abzubrechen, sondern so viele Einträge wie möglich zu senden und die auftretenden Fehler am Ende gesammelt in Moses anzuzeigen. Daher ist zu beachten, dass Fehler in den verwendeten (Sub-) Methoden nicht einfach in die übergeordnete Methode weitergeleitet, sondern z.B. pro Klasse in einer Liste gesammelt und schließlich in der Antwort vom Proxy an die Moses-Anwendung mitgeliefert werden. Die Infrastruktur für Letzteres gibt es, wie angedeutet, ebenfalls schon.

Insgesamt bestand meine Aufgabe also darin, die beiden gegebene Datenmodelle zur Prüfungs- und Lehrveranstaltungsabbildung nachzuvollziehen und anschließend Klassen zum Versenden und zur Umwandlung zu implementieren, in denen die Daten aus Moses den richtigen Feldern und Entitäten von His zugeordnet werden.

3 Fazit und Ausblick

Während meiner Zeit bei MathPlan konnte ich viele Eindrücke verschiedener Art gewinnen. Zum einen habe ich viel über die Arbeitsinhalte vor allem von Informatiker:innen erfahren, und inhaltlich etwas über ganzzahlige Optimierung und einen kleinen Teil der Universitätsverwaltung gelernt. Bei Letzterer hätte ich am wenigsten erwartet, dass sie so kompliziert ist. Das betrifft vor allem, das Kursangebot einer Uni digital abzubilden und dafür eine Struktur zu finden, in der die Daten sinnvoll abgespeichert und verwendet werden können. Vielleicht deswegen, oder weil ich selber einen Teil zu einer Software beigetragen habe, finde ich die manchmal auftretenden Fehler im CaMS der TU Berlin und in Moses, das ich als Student selber benutzt habe, seit meinem Praktikum (etwas) verständlicher. Inhaltlich habe natürlich viel über Java gelernt. Dazu gehört vor allem was es zu beachten gibt, um Code zu schreiben, der robuster gegenüber auftretenden Fehlern oder nicht-abgedeckten Randfällen ist, der performanter oder für andere Entwickler:innen besser lesbar ist. In der vergleichsweise kurzen Zeit konnte ich hier nur einen Einblick bekommen und ein Ahnung, wie viel mehr es dazu noch zu lernen gibt. Ein wenig vertrauter erschien mir objektorientiertes Entwickeln, bei dem Systeme abstrakter Objekte und Klassen, die einander referenzieren oder von einander erben können, gebraucht wurden. Ähnlich, wenn auch vielleicht weniger komplex, war es bei der relationalen Datenbank von Moses. Allgemeiner habe ich erfahren wie es ist, hauptberuflich Software zu entwickeln, sich eingehend und detailliert über einen längeren Zeitraum mit einer Aufgabe zu beschäftigen, daran zu knobeln und vor allem am Ende den Frust beim Fehler-finden so lange auszuhalten, bis mein geschriebener Code schließlich funktioniert (was ich je in Teilen schon aus dem Mathestudium kannte). Einmal durfte ich an einem digitalen Treffen mit einem Vertreter von HisInOne teilnehmen und habe zumindest kurz gesehen was es bedeuten kann, beruflich mit Kunden (bzw. in diesem Fall Vertretern andere Firmen) zu tun zu haben. Obwohl ich mir bereits vor diesem Treffen eher vorstellen konnte mehr „inhaltlich“ zu arbeiten und weniger mit „Externen“ zu kommunizieren, war dieses Treffen eine interessante Abwechslung. Ingesamt habe ich auf jeden Fall einen Einblick in viele Frameworks und Sprachen erhalten, wie Java EE, JPA, PosgreSQL, JSF, Html und JBoss Wildfy.

Zum anderen, und das fand ich vermutlich am interessantesten, habe ich eine Idee davon bekommen was es bedeutet, als Informatiker mit diesen Frameworks zu arbeiten und auf dem Laufenden zu bleiben welche Neuerungen es gibt. Dieser Teil des Berufs ist wohl auch nicht unähnlich zu dem, was ich während meiner Zeit bei MathPlan mitbekommen habe. Denn dabei muss auch mit vorhandener Berufserfahrung immer wieder Neues gelernt werden, sei es, um Sicherheitslücken zu schließen oder bei neuen Technikstandards mithalten zu können. Ich habe gelernt mit einer IDE (einem Code-Editor) und *Git* (einer Software zur Versionskontrolle, mit der auch Projekte auch mit externen Quellen ausgetauscht werden können) mit anderen Entwickler:innen am gleichen Projekt zu arbeiten, ohne sich dabei in die Quere zu kommen. Ähnlich dem Bearbeiten von Mathe-Übungsblättern war es bei dem Projekt an dem ich teilgenommen habe wichtig, zusammenzuarbeiten um auftretende Probleme zu lösen. Dies hat, trotz Corona und einem Großteil Home-Office, gut geklappt. Es war schön und interessant, in einem sehr netten Team gemeinsam an einer Aufgabe zu arbeiten und zu sehen, wie dabei sichtbare Ergebnisse entstanden, die andere Menschen in Zukunft verwenden werden.

In der nächsten Zeit wird sich am mathematischen Teil der in Kapitel 2.1 beschriebenen Optimierung vermutlich nicht viel ändern. Die von mir bearbeitete Export-Schnittstelle zum HisInOne-System wird voraussichtlich Anfang des nächsten Jahres in Betrieb genommen. Falls dafür noch Änderungen nötig werden oder beim Life-Betrieb unerwartet Fehler auftreten, wird die Arbeit daran von anderen Mitarbeiter:innen aus dem Schnittstellen-Team von MathPlan fortgeführt werden. Einen Datenaustausch mit weiteren Hochschulen einzurichten ist bei jedem neuen Kunden von MathPlan eine zeitaufwändige Aufgabe, wie hier für den Fall in Kapitel 2.2 zum Teil schon illustriert wurde. Die Firma plant daher langfristig ihre Webanwendung so zu erweitern, dass dies in Zukunft leichter und mit weniger individuellen Anpassungen möglich sein wird. Ich bedanke mich recht herzlich für die gute Zeit bei MathPlan und die viele interessanten Eindrücke, die ich dort sammeln durfte.

Literatur

[1] M. Carolla, *Ein Referenz-Datenmodell für Campus-Management-Systeme in deutschsprachigen Hochschulen*, Advances in Information Systems and Business Engineering, Springer, Berlin Heidelberg (2015).

[2] *Anzahl an Studierenden in Deutschland 2022 vom Statistischen Bundesamt*, zuletzt besucht am 13. Juni 2022, `https://www.destatis.de/DE/Presse/Pressemitteilungen/2021/11/PD21_538_21.html`.

[3] *Hochschul-Kompass mit einem Verzeichnis aller deutschen Hochschulen*, zuletzt besucht am 10. Juni 2022, `https://www.hochschulkompass.de/home.html`.

[4] *Informationen über das MOSES-Projekt an der TU Berlin*, zuletzt besucht am 7. Juni 2022, unter `https://www.innocampus.tu-berlin.de/projekte/moses`.

[5] *Aktuelle und zukünftige Projekte der MathPlan GmbH*, zuletzt besucht am 7. Juni 2022, unter `https://www.mathplan.de/projekte.html`.

[6] G. Lach, M. E. Lübbecke *Optimal University Course Timetables and the Partial Transversal Polytope*, In: McGeoch, C.C. (ed.) WEA 2008. LNCS 5038, 235–248. Springer, Heidelberg (2008).