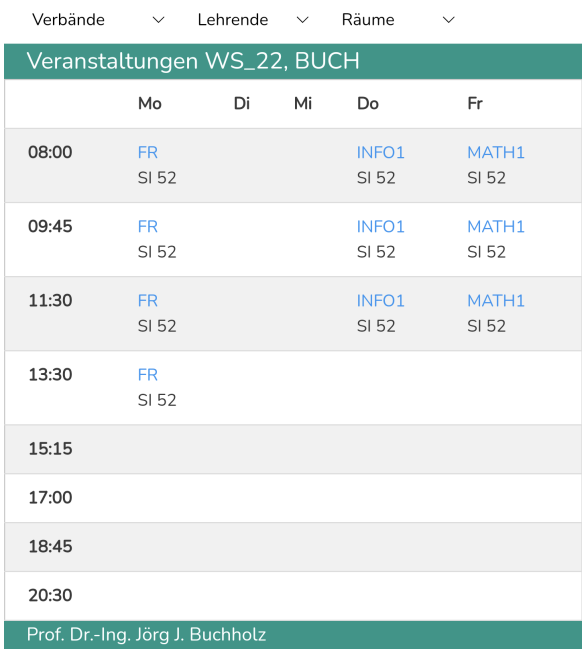


Iplan

Jörg J. Buchholz

16. November 2023

Teil I

Bedienungsanleitung

1 Einführung

Diese Dokumentation beschreibt die aktuelle Darstellung der Lehrveranstaltungs- und Prüfungspläne der Abteilung Maschinenbau der Hochschule Bremen und die dazu nötigen Programme.

Der Name des Projekts lplan lässt sich sowohl englisch "I plan" als auch bayerisch „I plan“ lesen. In beiden Fällen kommt zum Ausdruck, dass die Lehrenden ihre Veranstaltungspläne weitestgehend selbst gestalten können. In der Vergangenheit hatte die Veranstaltungsplanerin[1] vor Beginn der Veranstaltungs- bzw. Prüfungsphase alle Zuordnungen von Dozentinnen, Verbänden, Zeiten, Räumen und weiteren Informationen zu treffen und in den Plan einzutragen:

1. Wer liest bzw. prüft welche Module in welchen Verbänden?
2. Wann findet die Veranstaltung statt?
3. In welchen Räumen findet die Veranstaltung statt?
4. Welche Anmerkungen haben die Dozentinnen zu ihrer Veranstaltung?
5. Wie lautet der Aulis-Link zur Veranstaltung?

Mit lplan reduziert sich der Aufwand der Planerin auf den ersten Punkt, also die Zuordung der Module zu Verbänden und Dozentinnen. Die Auswahl der Veranstaltungszeiten und -räume und das Eintragen weiterer Veranstaltungsinformationen kann die Dozentin jetzt selbstständig durchführen und gegebenenfalls beliebig oft optimieren.

[1] Dieses Dokument verwendet das generische Femininum. Männer sind automatisch mitgemeint.

2 Nutzung des Programms

2.1 Mobile First Design

lplan priorisiert die unter Studentinnen übliche Darstellung auf dem Mobiltelefon („mobile first“). Insbesondere der Stundenplan ist auf jedem aktuellen Telefon ohne Scrollen sichtbar (Abbildung 2.1). Andererseits wird der Plan selbstverständlich auch auf Tablets und Desktoprechnern übersichtlich dargestellt.

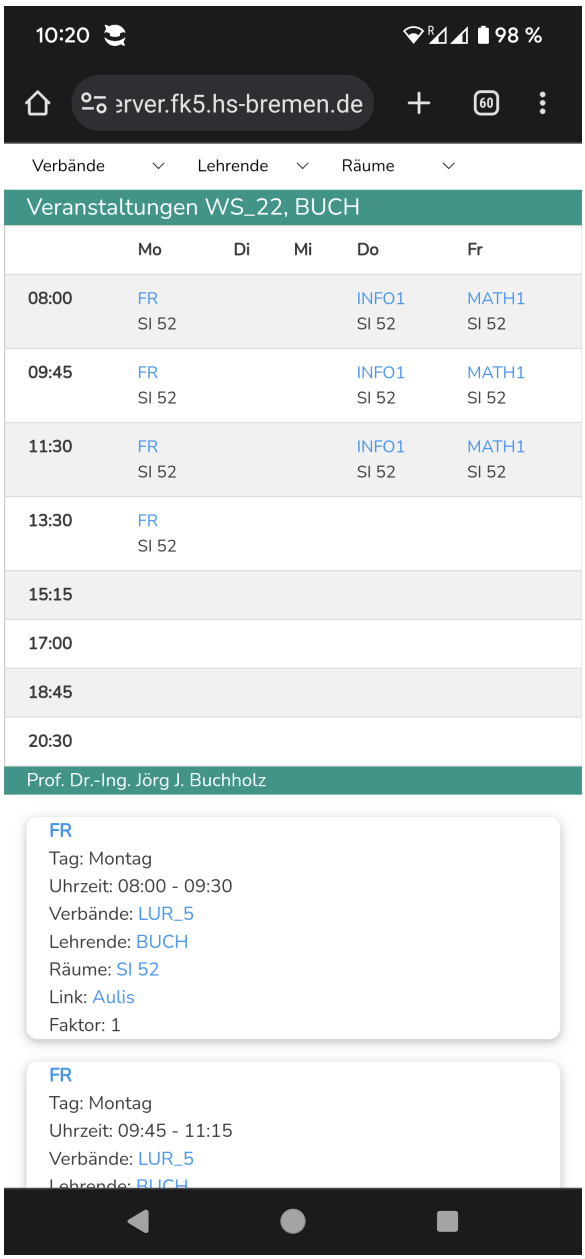


Abbildung 2.1: Screenshot der Veranstaltungsseite auf einem Mobiltelefon (Pixel 7 Pro)

2.2 Auswahl

Nutzerinnen verwenden üblicherweise die Einstiegsseite[1]

[1]Diese Seite ist mit ihrem Query-String [1] (`team=5M`) ein Überbleibsel aus der Zeit, als wir noch die Pläne der gesamten Hochschule Bremen dargestellt hatten [2]. Wir werden sie aktualisieren, wenn auch die letzte noch verbleibende Fakultät ihre eigene Plandarstellung verwendet.

`https://m-server.fk5.hs-bremen.de/plan/semester.aspx?team=5M`

um dort das gewünschte Semester auszuwählen (Abbildung 2.2).

Abbildung 2.2: Auswahl des Veranstaltungs- oder Prüfungsplans

Durch Anklicken eines der aufgelisteten Semester (z. B. Wintersemester 2022/2023) wird die Plandarstellungsseite mit angehängtem Query-String [1] (`semester=ws_22`) aufgerufen

`.../iplan/veranstaltungen.aspx?semester=ws_22`

und die Nutzerin gelangt zu einer „leeren“ Plandarstellungsseite, auf der sie in der oberen Dropdownlisten-Zeile einen Verband, eine Dozentin oder einen Raum auswählen kann (Abbildung 2.3).

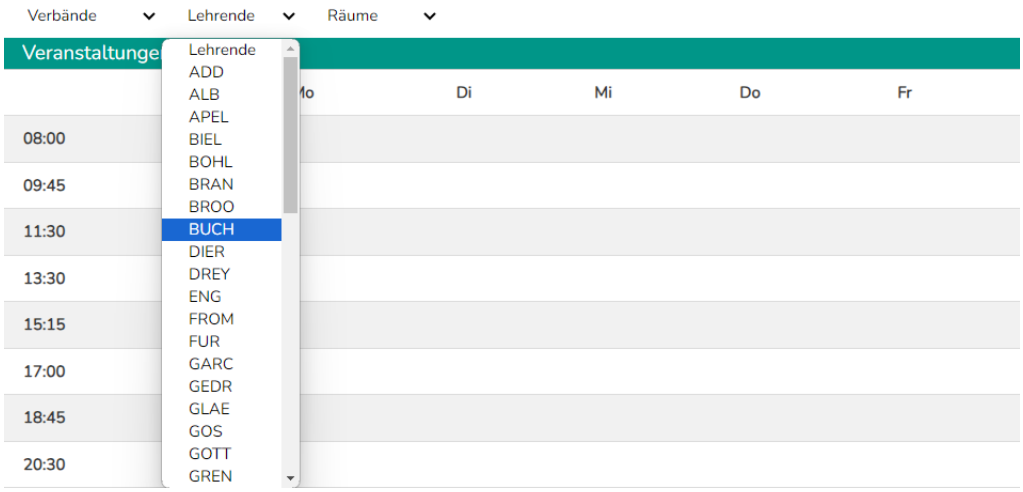


Abbildung 2.3: Auswahl des Verbands, der Dozentin oder des Raumes auf dem Desktoprechner

Nach Auswahl des Verbands, der Dozentin oder des Raumes ruft sich die Seite nochmals selbst mit dem passenden Query-String auf:

```
.../iplan/veranstaltungen.aspx?semester=ws_22&art=dozent&code=BUCH
```

Dieses Vorgehen hat den Vorteil, dass die Nutzerin danach ihren Plan aus der Adresszeile ihres Browsers direkt „bookmarken“ kann; der Query-String[2] enthält dann alle Informationen zum ausgewählten Veranstaltungsplan:

`semester=ws_22`	Auswahl des Semesters, hier des Wintersemesters 2022/2023
`art=dozent`	Auswahl, ob der Plan eines Verbandes, einer Dozentin oder eines Raumes dargestellt werden soll. Hier wird der Plan einer Dozentin dargestellt.
`code=BUCH`	Auswahl der Dozentin, hier Prof. Dr.-Ing. Jörg J. Buchholz

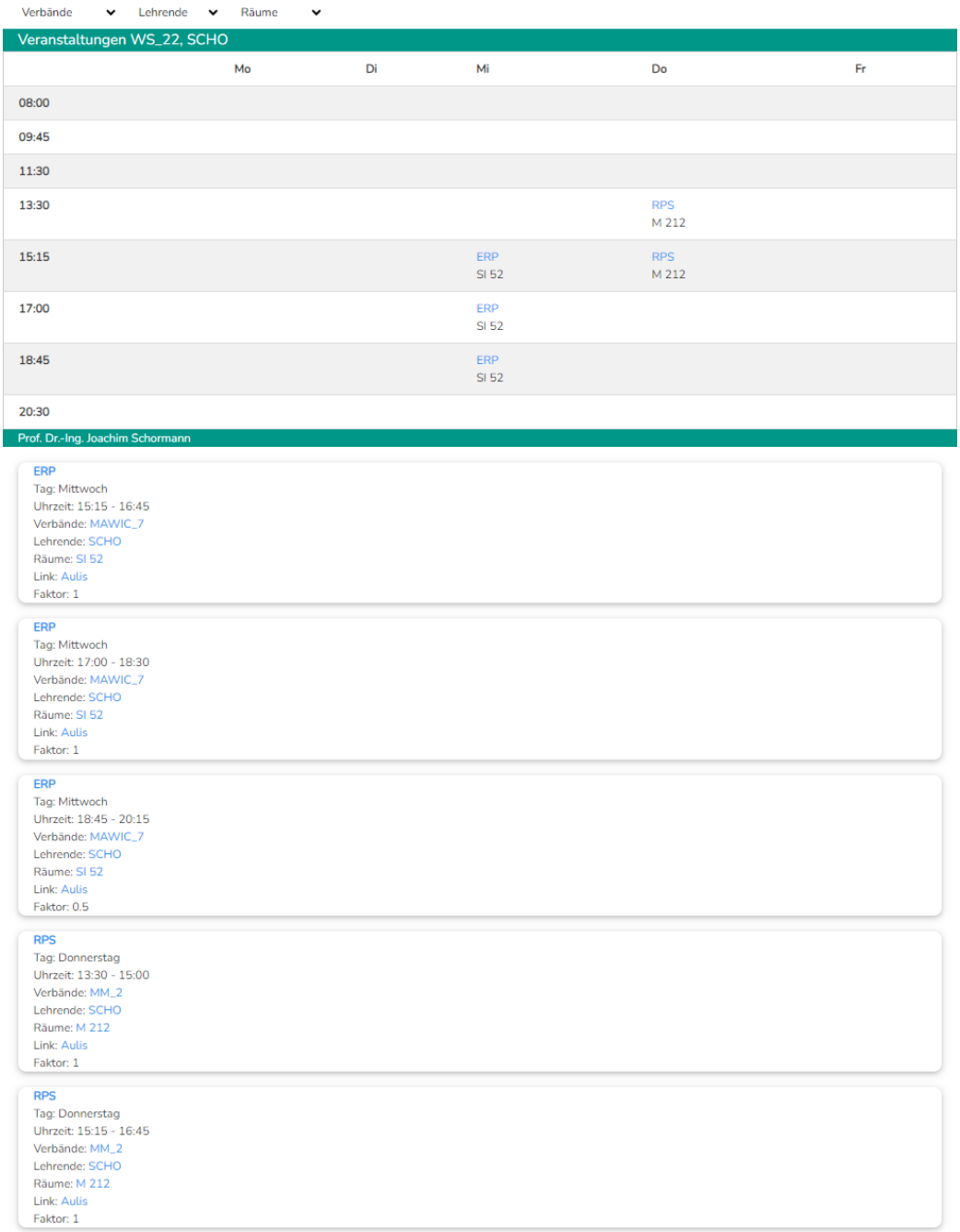
Schließlich gelangt die Nutzerin zum ausgewählten Veranstaltungsplan (Abbildung 2.4). Dieser[3] besteht auf dem Desktoprechner aus drei Teilen:

- Stundenplan
- Details jedes einzelnen Planeintrags
- Tabelle[4] aller Planeinträge

[2] Die Unterscheidung zwischen Veranstaltungsplänen und Prüfungsplänen geschieht nicht über den Query-String, sondern über den Namen der Seite (`.../pruefungen.aspx...`) selbst, da sich die Darstellungen von Veranstaltungsplänen und Prüfungsplänen ein wenig unterscheiden.

[3] In Abbildung 2.4 haben wir bewusst den Veranstaltungsplan eines Kollegen gewählt, der im dargestellten Semester weniger Veranstaltungen gelesen hat, sodass wir den Gesamtplan auf einer DIN-A4-Seite besser lesbar abbilden können.

[4] Auf dem Mobiltelefon zeigen wir die Tabelle aller Planeinträge nicht an, da sie üblicherweise zu breit für ein Telefon im Hochformat ist.



Modul	Dozent	Verband	Tag	Zeit	Raum	Aulis	Bemerkung	Faktor
ERP	SCHO	MAWIC_7	Mittwoch	15:15 - 16:45	SI 52	https://aulis.hs-bremen.de/goto.php?target=frm_1255358&client_id=hsbremen		1
ERP	SCHO	MAWIC_7	Mittwoch	17:00 - 18:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=frm_1255358&client_id=hsbremen		1
ERP	SCHO	MAWIC_7	Mittwoch	18:45 - 20:15	SI 52	https://aulis.hs-bremen.de/goto.php?target=frm_1255358&client_id=hsbremen		0.5
RPS	SCHO	MM_2	Donnerstag	13:30 - 15:00	M 212	https://aulis.hs-bremen.de/goto.php?target=crs_1219771&client_id=hsbremen		1
RPS	SCHO	MM_2	Donnerstag	15:15 - 16:45	M 212	https://aulis.hs-bremen.de/goto.php?target=crs_1219771&client_id=hsbremen		1

Abbildung 2.4: Die Plandarstellung besteht auf dem Desktoprechner aus drei Teilen: Stundenplan (Abschnitt 2.3), Details (Abschnitt 2.4) und Tabelle (Abschnitt 2.5). 8

2.3 Stundenplan

2.3.1 Dozentin

Der Stundenplan wird in der klassischen Ansicht (Wochentage horizontal, Zeiten vertikal) dargestellt (Abbildung 2.5). Wichtigstes Kriterium bei der Darstellung des Stundenplans ist die Möglichkeit, den gesamten Plan auf einem Mobiltelefon im Hochformat übersichtlich darstellen zu können: Wir nutzen daher Abkürzungen für die Wochentage (Mo, Di, ...) und ausschließlich die Anfangszeiten (08:00, 09:45, ...) der Blöcke. Zeitangaben wie 08:00 - 09:30 würden die erste Spalte zu breit werden lassen, sodass der Plan auf einigen aktuellen Telefonen nicht ohne horizontales Scrollen komplett darstellbar wäre. Alternativ könnten wir 08:00 - 09:30 in zwei oder drei Zeilen schreiben; dadurch würde der Plan aber deutlich länger werden und an Kompaktheit und Übersichtlichkeit verlieren. Die Planeinträge einer Dozentin zeigen ihr aus dem gleichen Grund nur die Abkürzung (FR, INFO1, ...) des Moduls, das sie liest und die Abkürzung (SI 52, ...) des Raumes, in dem die Veranstaltung stattfindet.

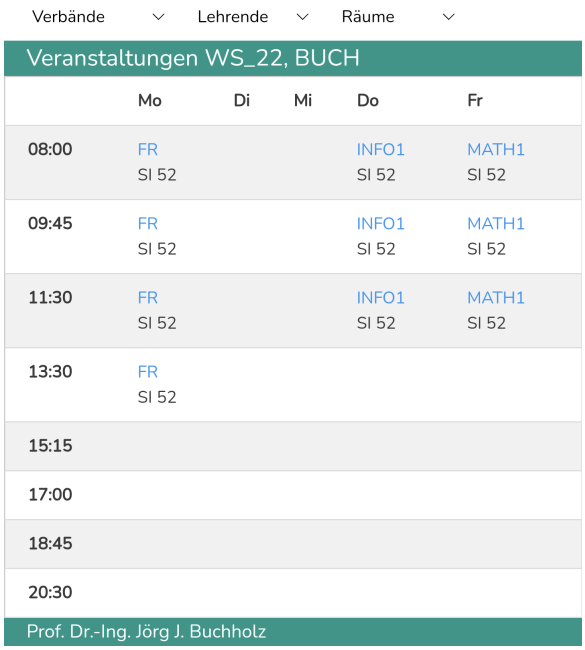


Abbildung 2.5: Stundenplan des Autors im Wintersemester 2022/2023

2.3.2 Studentin

Auch der Stundenplan eines Semesterverbands zeigt der Studentin den Wochentag, die Anfangszeit, das Modul und den Raum einer jeden Veranstaltung (Abbildung 2.6)

Abbildung 2.6: Stundenplan eines Semesterverbands

In seltenen Fällen, beispielsweise bei Wahlpflichtmodulen, können an einem Termin auch mehrere Module gleichzeitig angeboten werden. Diese gleichzeitigen Module haben wir im Stundenplan durch eine gestrichelte Linie (-------) voneinander getrennt (s. Abbildung 2.6).

2.3.3 Raum

Während wir in Dozentin- und Verbandsplänen in jedem Eintrag das Modul und den Raum darstellen, ersetzen wir in Raumplänen die Räume sinnvollerweise durch die jeweiligen Dozentinnen (Abbildung 2.7).

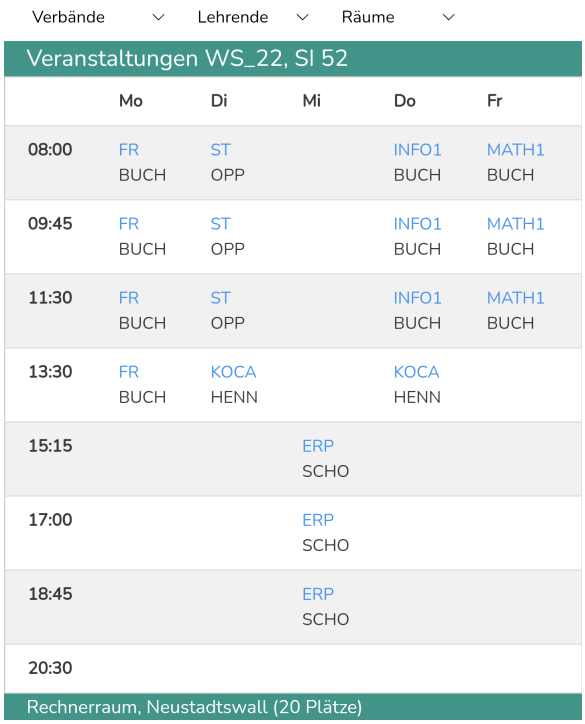


Abbildung 2.7: Stundenplan eines Raumes

2.4 Details

Wenn die Studentin oder die Dozentin weitere Informationen über einen Stundenplaneintrag bekommen möchte, klickt sie in der Stundenplandarstellung einfach auf die Modulabkürzung. Das Programm scrollt dann automatisch zur Detaildarstellung des ausgewählten Eintrags herunter. In Abbildung 2.8 sind exemplarisch die Detailkarten zweier Stundenplaneinträge des Plans aus Abbildung 2.5 dargestellt.

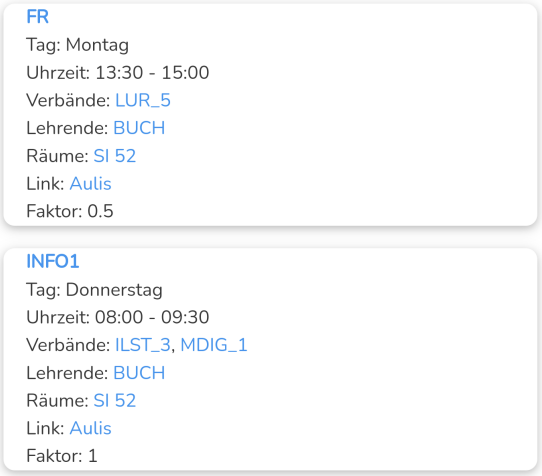


Abbildung 2.8: Detaildarstellung zweier Stundenplaneinträge

Die Karten sind nach Tag und Uhrzeit sortiert und bieten weitere Informationen zum jeweiligen Stundenplaneintrag:

Tag	Der Wochentag, an dem diese Veranstaltung stattfindet
Uhrzeit	Die Uhrzeit (Anfang bis Ende), zu der diese Veranstaltung stattfindet
Verbände	Alle Semesterverbände, die diese Veranstaltung hören
Lehrende	Alle Dozentinnen, die diese Veranstaltung lesen
Räume	Alle Räume, in denen diese Veranstaltung stattfindet
Aulis	Der Link zur Lernplattform der Hochschule Bremen
Bemerkung	Optionale Informationen der Dozentin für diese Veranstaltung
Faktor	Anrechnungsfaktor dieser Veranstaltung für die hauptamtliche Dozentin

2.4.1 Modulbeschreibung

Ein Klick auf die Veranstaltungsabkürzung auf der Detailkarte führt zur Modulbeschreibung der Veranstaltung; so öffnet das Klicken auf FR beispielsweise die Modulbeschreibung der Flugregelungsveranstaltung (Abbildung 2.9).

Modulbeschreibung

HSB Hochschule Bremen City University of Applied Sciences

Titel *Title*	Flugregelung	
Modulcode *Module Code*	FR	
Modulverantwortliche *Responsible Members of Staff*	Prof. Dr.-Ing. Jörg J. Buchholz	
Kompetenzziele des Moduls *Module Competence Goals*	Die Studierenden werden in die Lage versetzt, die grundlegenden praxisbezogenen Methoden und Verfahren der angewandten Flugregelung zur Erstellung mathematischer Modelle der Regelstrecke Flugzeug und zur Synthese einfacher Autopilotenfunktionen zu beherrschen.	*Knowledge and methods of flight control used to create mathematical models of the aircraft control for the synthesis of simple autopilot functions.*
Lehrinhalte *Content*	1. Die oben aufgeführten Kompetenzen werden durch einen seminaristischen Unterricht vorbereitet und dann in Form von angeleiteten Übungsaufgaben auch mit Laborbeispielen im betreuten Selbststudium, durch Hausaufgaben und durch eigenständige Literaturstudien ausgebaut. Hierzu werden jeweils Literaturempfehlungen ausgegeben. Um die angestrebten Lernziele zu erreichen, werden in der Lehre folgende spezifische Kompetenzschwerpunkte gesetzt: 2. Grundlagen der Regelungstechnik mit Matlab und Simulink 3. Bezeichnungen der Luftfahrt 4. Koordinatentransformation 5. Aerodynamik 6. Triebwerk	*1.* *2. Fundamentals of automatic control with Matlab and Simulink* *3. Aviation denominations* *4. Coordinate transformation* *5. Aerodynamics* *6. Engine* *7. Actuator dynamics*

Abbildung 2.9: Durch Anklicken der Modulabkürzung FR auf der Detailkarte gelangt die Nutzerin zur Beschreibung des Moduls Flugregelung. In dieser Abbildung ist nur ein kleiner Teil der Modulbeschreibung dargestellt.

2.4.2 Verbände, Dozentinnen und Räume

Anklicken der auf der Detailkarte eingetragenen Verbände, Dozentinnen und Räume führt zu deren eigenen Veranstaltungsplänen.

2.4.3 Aulis

Der anklickbare Aulis-Link, beispielsweise

`https://aulis.hs-bremen.de/goto.php?target=crs_1153066`

führt zu der entsprechenden Veranstaltungsseite auf der Lernplattform der Hochschule Bremen, auf der die Dozentin ausführliche Informationen (Skripte, Aufgaben, ...) zur Veranstaltung bereitstellen kann (Abbildung 2.10).

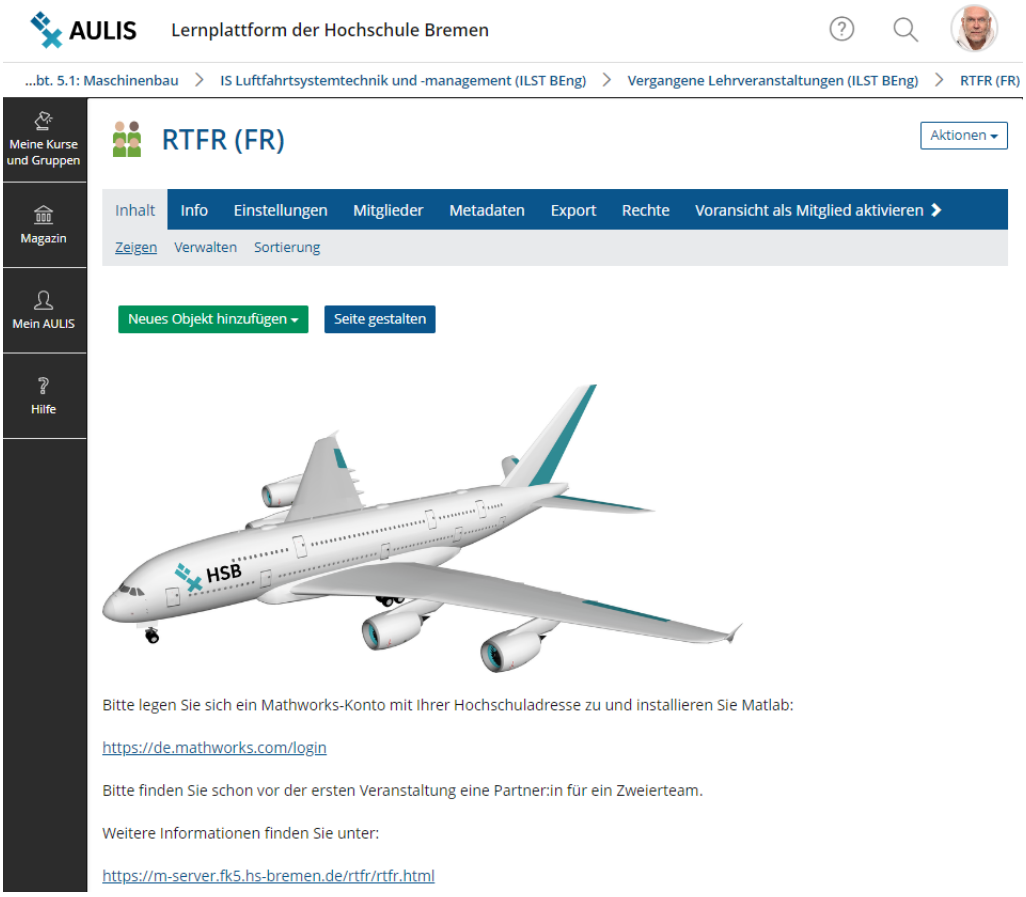


Abbildung 2.10: Aulis-Seite des Moduls Flugregelung in der Ansicht der Dozentin

2.4.4 Bemerkung

Kurze Informationen (Ausfall, Verschiebung, ...) kann die Dozentin direkt als Bemerkung auf der Detailkarte eintragen.

2.4.5 Faktor

Hauptamtlich Lehrende müssen am Ende eines akademischen Jahres eine Erklärung über Art und Umfang der Lehrtätigkeit abgeben, auf der ihr Lehrdeputat den von ihnen tatsächlich gelesenen Veranstaltungen gegenübergestellt wird. Da hierzu der veröffentlichte Veranstaltungsplan verwendet wird, kann die Veranstaltungsplanerin unter Faktor eintragen, ob eine Veranstaltung beispielsweise nur halb (modulbezogene Übung) oder gar nicht (zusätzliche Labortermine, ...) angerechnet wird. Üblicherweise geschieht dieser Eintrag, nachdem sich die Studentinnen für ihre Prüfungen angemeldet habe, sodass berechnet werden kann, wie viele Semesterwochenstunden der Dozentin angerechnet

werden [3]. Für Lehrbeauftragte wird der Faktor zwar angezeigt, hat dort aber keine Bedeutung.

2.4.6 Mehrere Verbände, Dozentinnen und Räume

Insbesondere bei Prüfungsplänen kann es durchaus vorkommen, dass mehrere Dozentinnen gemeinsam eine Veranstaltung in mehreren Verbänden und mehreren Räumen durchführen. In diesem Fall kann die Detailkarte schon mal etwas voller werden (Abbildung 2.11).

Abbildung 2.11: Mehrere Dozentinnen führen eine Thermodynamikprüfung gemeinsam in mehreren Verbänden und mehreren Räumen durch.

2.5 Tabelle

Auf Wunsch einzelner Kolleginnen stellen wir auf dem Desktoprechner am Ende der Veranstaltungsseite alle Informationen noch einmal in tabellarischer Form dar (Abbildung 2.12). Auf dem Mobiltelefon zeigen wir diese Tabelle nicht an, da sie die Breite eines Telefons im Hochformat sprengen würde.

Modul	Dozent	Verband	Tag	Zeit	Raum	Aulis	Bemerkung	Faktor
FR	BUCH	LUR_5	Montag	08:00 - 09:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen		1
FR	BUCH	LUR_5	Montag	09:45 - 11:15	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen		1
FR	BUCH	LUR_5	Montag	11:30 - 13:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen		1
FR	BUCH	LUR_5	Montag	13:30 - 15:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen		0.5
INFO1	BUCH	ILST_3, MDIG_1	Donnerstag	08:00 - 09:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153182&client_id=hsbremen		1

Abbildung 2.12: Tabellarische Auflistung aller Informationen. In dieser Abbildung wird nur ein Teil der Veranstaltungen dargestellt.

2.6 Prüfungspläne

Prüfungspläne werden ähnlich wie die Veranstaltungspläne dargestellt (Abbildung 2.13), weisen allerdings ein paar Unterschiede auf: Da sich der Prüfungszeitraum über zwei bis drei Wochen erstreckt, verzichten wir auf den ja für jede Woche identischen Stundenplan (einschließlich der Zeile unter dem Stundenplan). Außerdem verzichten wir auch auf einem Desktoprechner auf eine tabellarische Prüfungsdarstellung. Es werden also lediglich die Detailkarten der Prüfungen dargestellt (Abschnitt 2.6.1). Zusätzlich gibt es sowohl für Verbände als auch für Dozentinnen eine Möglichkeit, die Termine der eigenen Prüfungen in Form einer ICS-Datei [4] herunterzuladen, um sie in den persönlichen Kalender zu importieren (Abschnitt 2.6.2).

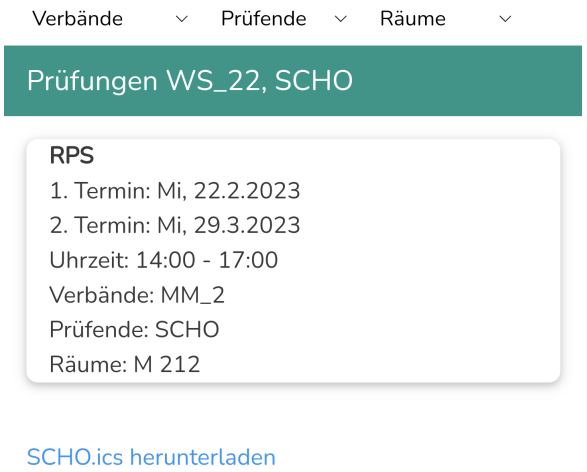


Abbildung 2.13: Prüfungsdarstellung auf dem Mobiltelefon

2.6.1 Prüfungsdetails

Bei Prüfungen (Abbildung 2.11 und Abbildung 2.13) werden die Termine beider Prüfungen (Erst- und Zweitprüfung) auf den Detailkarten angezeigt. Die Zeitspanne zwischen beiden Prüfungsterminen wird von der Planerin a priori in Abhängigkeit von der Länge der vorlesungsfreien Zeit und etwaigen Feiertagen festgelegt und ist für **alle** Prüfungen gleich, sodass die Terminplanung von der Prüferin nur einmalig für beide Prüfungen gemeinsam durchgeführt werden muss. Gleichzeitig bedeutet das für die Studentin, dass sie immer die gleiche Zeit zwischen beiden Prüfungen zur Vorbereitung hat.

Bei Verbänden gibt es wegen der üblicherweise fünf Module pro Semester häufig fünf Prüfungsdetailkarten. Da einige Module keine singuläre Prüfungen im Prüfungszeitraum durchführen oder es mehrere Wahlpflichtmodule im Semester gibt, kann es möglicherweise aber auch weniger oder mehr als fünf Verbandsprüfungskarten geben.

2.6.2 ICS-Datei

Wenn die Dozentin in Abbildung 2.13 auf den Link SCHO.ics herunterladen klickt, wird die folgende ICS-Datei [4] automatisch auf ihren eigenen Rechner heruntergeladen:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:iplan
METHOD:PUBLISH
BEGIN:VEVENT
UID:21cfc657-bc69-4fda-99cb-a6b47168e2d4
SUMMARY:RPS
LOCATION:M 212
DESCRIPTION:Verbände: MM_2\nPrüfende: SCHO
DTSTART:20230222T140000
DTEND:20230222T170000
DTSTAMP:20231101T205534
END:VEVENT
BEGIN:VEVENT
UID:0fab2d1b-04ba-4bd0-baec-1a473b6392a8
SUMMARY:RPS
LOCATION:M 212
DESCRIPTION:Verbände: MM_2\nPrüfende: SCHO
DTSTART:20230329T140000
DTEND:20230329T170000
DTSTAMP:20231101T205534
END:VEVENT
END:VCALENDAR
```

Die ICS-Datei kann die Nutzerin dann in die Kalenderanwendungen aller[5] aktuellen Betriebssysteme importieren, um die eigenen Prüfungen im persönlichen Kalender darzustellen. Abbildung 2.14 zeigt, wie die Prüfung beispielsweise im Google-Kalender dargestellt wird.

[5] Ja, selbst die Kalender unter macOS, IOS und Linux lesen ICS-Dateien.

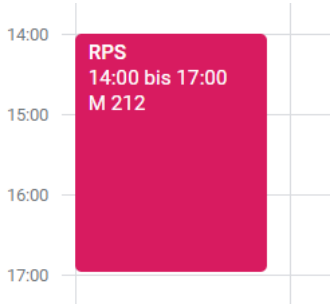


Abbildung 2.14: Prüfungseintrag im Google-Kalender

Abbildung 2.15 offenbart, dass auch die in die Kalenderdatei eingetragenen Details (Verbände und Prüferinnen) mit in den Eintrag übernommen werden.

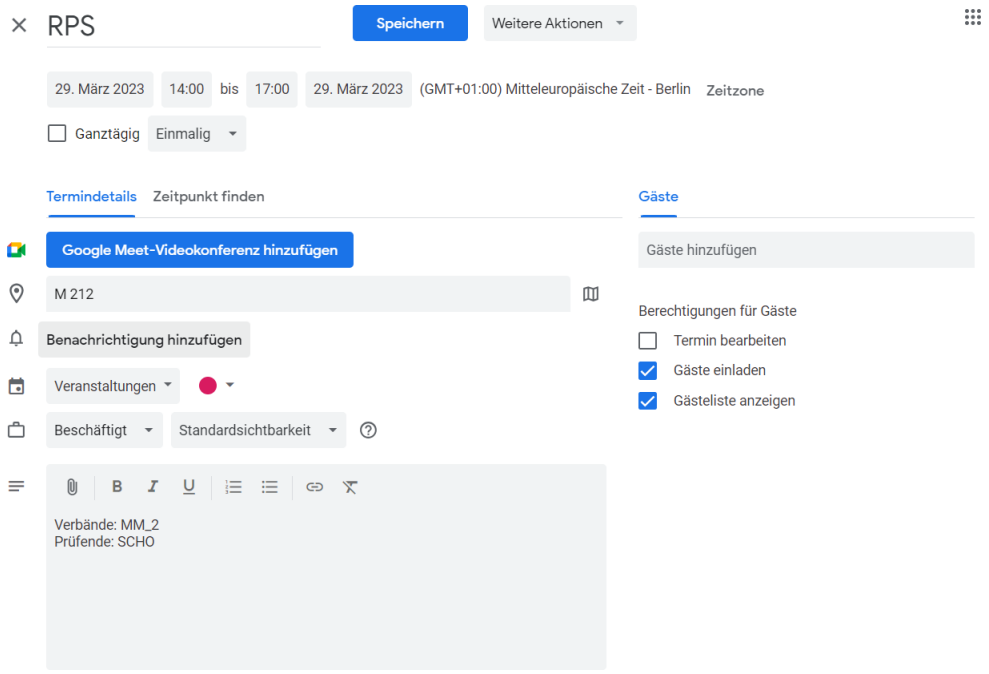


Abbildung 2.15: Prüfungsdetails im Google-Kalender

2.7 Veranstaltungsplanung

Nachdem die Planerin alle Dozentinnen den Semesterverbänden und Modulen zugeordnet hat, bittet sie alle Dozentinnen mit einer E-Mail, ihre Veranstaltungspläne selbst zu gestalten:

Liebe Kolleginnen und Kollegen,
bitte nutzen Sie einen Desktoprechner, um Ihre Veranstaltungswünsche (Tage, Zeiten, Räume und Aulislinks) einzutragen unter:

https://m-server.fk5.hs-bremen.de/iplan/planung/veranstaltungsplanung.aspx?semester=ws_22
Die mobiltelefongerechte Darstellung Ihrer Veranstaltungen erfolgt dann unmittelbar unter:
https://m-server.fk5.hs-bremen.de/iplan/veranstaltungen.aspx?semester=ws_22
Beachten Sie bitte folgende Randbedingungen:
* Wenn Ihr Wunschraum zum ausgewählten Termin nicht verfügbar ist, probieren Sie bitte einen anderen Termin (Tag, Zeit).
* Versuchen Sie bitte - aus Rücksicht auf Ihre Kolleginnen und Kollegen - keine zu großen Räume auszuwählen.
* Viele Kolleginnen und Kollegen möchten ein Modul gerne an einem Vormittag abarbeiten. Vermeiden Sie daher bitte Einzelblöcke an Vormittagen.
* Bei Online-Veranstaltungen teilen Sie mir Ihre Terminwünsche bitte unter joerg.buchholz@hs-bremen.de mit.
* Die Planung der ISWI-Verbände wird teilweise von der Fakultät 1 durchgeführt. Stimmen Sie Ihren ISWI-Wunschtermin daher bitte zusätzlich mit claudia.humbert@hs-bremen.de ab.
Mit freundlichen Grüßen
Prof. Dr.-Ing. Jörg J. Buchholz
https://jjbuchholz.de

Die Dozentin ruft dann die Veranstaltungsplanungsseite mit dem aktuellen Semester als Query-String auf

```
.../iplan/planung/veranstaltungsplanung.aspx?semester=ws_22
```

und gelangt auf die zentrale Authentifizierungsseite der Hochschule Bremen (Abbildung 2.16)

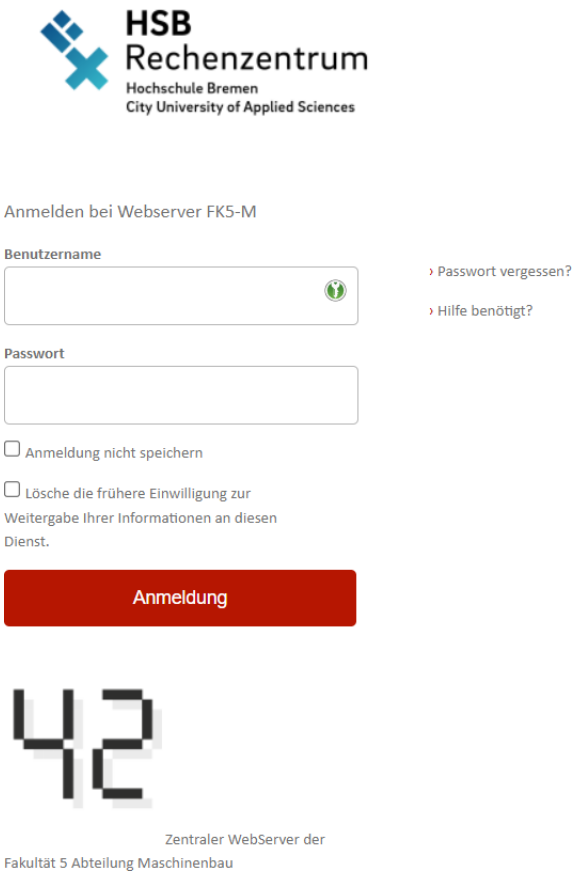


Abbildung 2.16: Zentrale Authentifizierungsseite der Hochschule Bremen

wo sie sich mit ihrem Hochschulkonto anmeldet und im nächsten Schritt die Auswahlseite[6] mit genau nur den Modulen erhält, die sie selbst lesen wird (Abbildung 2.17).

[6]Zu Beginn der Planung sind Tag, Zeit, Raum, Aulis und Bemerkung natürlich noch nicht ausgefüllt.

	Modul	Verband	Tag	Zeit	Raum	Aulis	Bemerkung
Auswählen	FR	LUR_5	Montag	08:00 - 09:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen	
Auswählen	FR	LUR_5	Montag	09:45 - 11:15	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen	
Auswählen	FR	LUR_5	Montag	11:30 - 13:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen	
Auswählen	FR	LUR_5	Montag	13:30 - 15:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153066&client_id=hsbremen	
Auswählen	INFO1	ILST_3, MDIG_1	Donnerstag	08:00 - 09:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153182&client_id=hsbremen	
Auswählen	INFO1	ILST_3, MDIG_1	Donnerstag	09:45 - 11:15	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153182&client_id=hsbremen	
Auswählen	INFO1	ILST_3, MDIG_1	Donnerstag	11:30 - 13:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153182&client_id=hsbremen	
Auswählen	MATH1	ILST_1, MDIG_1	Freitag	08:00 - 09:30	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153133&client_id=hsbremen	
Auswählen	MATH1	ILST_1, MDIG_1	Freitag	09:45 - 11:15	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153133&client_id=hsbremen	
Auswählen	MATH1	ILST_1, MDIG_1	Freitag	11:30 - 13:00	SI 52	https://aulis.hs-bremen.de/goto.php?target=crs_1153133&client_id=hsbremen	

Abbildung 2.17: Module des Autors im Wintersemester 2022

Wenn die Dozentin die Auswählen-Schaltfläche einer Veranstaltung anklickt, wird oberhalb der Tabelle eine Detaildarstellung der Veranstaltung eingeblendet (Abbildung 2.18)

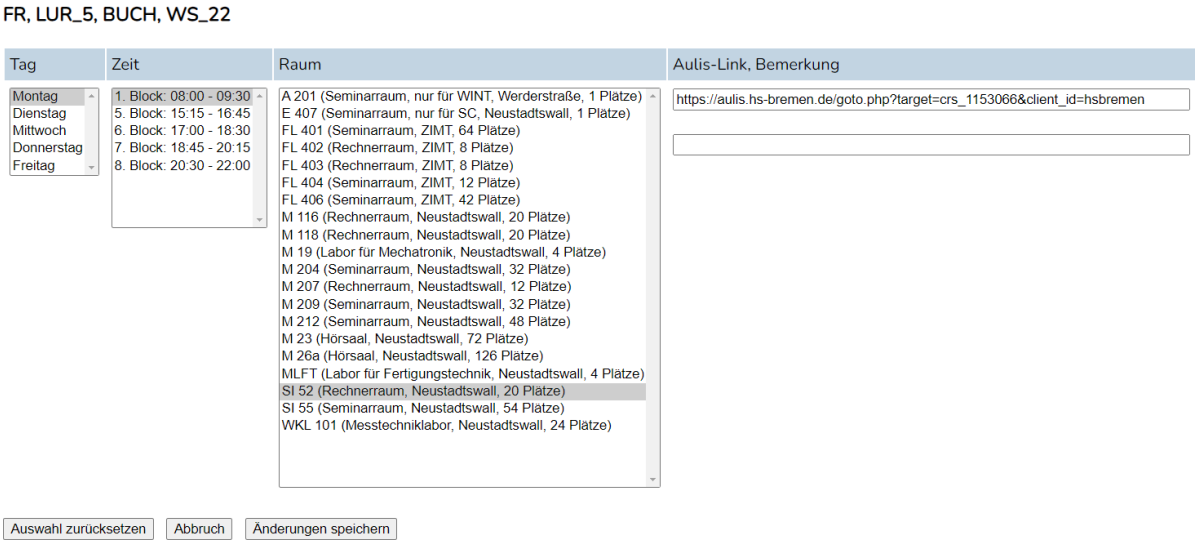


Abbildung 2.18: Details einer Veranstaltung

auf der sie den gewünschten Tag, den gewünschten Block und den gewünschten Raum

auswählen kann. Außerdem kann sie dort den Aulis-Link und eine kurze Bemerkung zur Veranstaltung eintragen.

Das Programm überprüft bei jeder Wahl im Hintergrund, welche Ressourcen unter Berücksichtigung der bisherigen Wahl noch verfügbar sind. Wenn die Dozentin also wie in Abbildung 2.18 geschehen, den Montag als Wochentag für ihre Veranstaltung ausgewählt hat, entfernt das Programm automatisch den zweiten bis vierten Block aus der Zeit-Liste, da diese Blöcke für den Semesterverband schon belegt sind. Wenn die Dozentin dann wie in Abbildung 2.18 geschehen, den ersten Block auswählt, entfernt das Programm automatisch alle Räume, die am Montag im ersten Block schon belegt sind, sodass die Dozentin nur noch unter den noch freien Räumen auswählen kann. Wenn sie dann die Änderungen speichern-Schaltfläche anklickt, um ihre Auswahl in der Datenbank abzuspeichern, überprüft das Programm nochmals, ob die Auswahl tatsächlich immer noch möglich ist und informiert die Dozentin gegebenenfalls darüber, dass eine andere Dozentin schneller[7] war.

Die Auswahl zurücksetzen-Schaltfläche entfernt alle ausgewählten Zeiten und Räume und lässt die Dozentin wieder bei der Auswahl des Wochentags beginnen. Der Aulis-Link und etwaige Bemerkungen werden dabei nicht entfernt.

Die Abbruch-Schaltfläche schließt die Detaildarstellung, ohne aktuelle Änderungen abzuspeichern.

2.8 Prüfungsplanung

Die Prüfungsplanung läuft sehr ähnlich ab wie die Veranstaltungsplanung. Die Planerin erzeugt aus dem aktuellen Veranstaltungsplan eine Zuordnung von Prüferinnen zu Verbänden und Modulen und bittet dann ihre Kolleginnen, Termine und Räume für ihre Prüfungen auszuwählen:

Liebe Kolleg:innen,
bitte nutzen Sie einen Desktoprechner, um Ihre Prüfungswünsche (Tage, Zeiten und Räume) einzutragen unter:
https://m-server.fk5.hs-bremen.de/iplan/planung/pruefungsplanung.aspx?semester=ss_23
Die mobiltelefongerechte Darstellung Ihrer Prüfung erfolgt dann unmittelbar unter:
https://m-server.fk5.hs-bremen.de/iplan/pruefungen.aspx?semester=ss_23
Beachten Sie bitte folgende Randbedingungen:
* Die Prüfungsplanung für die ISWI-Verbände wird von der Fakultät 1 durchgeführt.
* Die Prüfungen finden jeweils montags, mittwochs und freitags statt, um den in der Prüfungsordnung vorgesehenen freien Tag zwischen zwei Prüfungen zu gewährleisten.
* Zwischen Ihrem ersten und Ihrem zweiten Prüfungstermin liegen zehn Wochen.
* Durch Verwenden der Strg-Taste können Sie mehrere Räume für Ihre Prüfung auswählen.

[7] Es kann ja schließlich passieren, dass in der Zeit zwischen der Wahl von Dozentin_1 und dem Abspeichern ihrer Auswahl Dozentin_2 ihr eine Ressource vor der Nase weggeschnappt hat. Dieses Problem lässt sich bei großen Datenbankanwendungen natürlich mit Transaktionsklammern oder ähnlichen Konzepten lösen; hier wird Dozentin_1 einfach informiert und sie muss sich einen anderen Termin oder Raum suchen.

* Wenn Ihr Wunschraum zum ausgewählten Termin nicht verfügbar ist, probieren Sie bitte einen anderen Termin (Tag, Zeit).
* Abhängig von Ihrem Sicherheitskonzept möchten Sie eventuell nicht die gesamte angegebene Anzahl der Plätze nutzen.
* Versuchen Sie bitte - aus Rücksicht auf Ihre Kolleg:innen - trotzdem nicht zu viele und zu große Räume auszuwählen.
* Wenn Sie für ein Modul keine (singuläre) Prüfung durchführen möchten, lassen Sie bitte Tag, Zeit und Raum frei.
* Bei Online-Prüfungen lassen Sie bitte den Raum frei und tragen "Online" im Bemerkungsfeld ein.
* Sie können die Daten Ihrer Prüfung bis drei Wochen vor Beginn der Prüfungsphase selbst ändern.
Mit freundlichen Grüßen
Prof. Dr.-Ing. Jörg J. Buchholz
https://jjbuchholz.de

Nachdem sich die Prüferin mit ihrem Hochschulkonto authentifiziert hat, gelangt sie zu der in Abbildung 2.19 exemplarisch dargestellten Seite, auf der ihre Prüfungen aufgelistet sind.

	Modul	Verband	Tag	Zeit	Raum	Aulis	Bemerkung
Auswählen	FR	LUR_5				https://aulis.hs-bremen.de	
Auswählen	INFO1	ILST_3, MDIG_1				https://aulis.hs-bremen.de/	
Auswählen	MATH1	ILST_1, MDIG_1				https://aulis.hs-bremen.de/	

Abbildung 2.19: Prüfungsplanung des Autors im Wintersemester 2022

Nachdem sie dann beispielsweise FR ausgewählt hat, wird ihr über der Liste die Detailplanung (immer mit beiden Prüfungsterminen gemeinsam) dieser Prüfung angeboten (Abbildung 2.20).

Abbildung 2.20: Detailplanung

Sie hat nun die Möglichkeit, einen der vier[8] hier noch freien Termine und anschließend die Zeit und den Raum für die Prüfung auszuwählen. Üblicherweise finden die Prüfungen pro Prüfungstermin drei Wochen lang nur Montags, Mittwochs und Freitags statt,

[8]Die Auswahlmöglichkeit der Termine ist hier so klein, da vorher schon alle anderen Prüfungen des

um den Studentinnen den in der Prüfungsordnung vorgesehenen freien Tag zwischen zwei Prüfungen zu gewähren. Das Programm lässt für jeden Semesterverband nur eine Prüfung pro Tag zu, bietet wie bei der Veranstaltungsplanung während der Auswahl nur freie Ressourcen an und überprüft nochmals vor dem Abspeichern, ob tatsächlich noch alle ausgewählten Ressourcen verfügbar sind.

Verbandes LUR_5 eingetragen wurden, da der Autor üblicherweise keine singulären Prüfungen am Ende des Semesters durchführt.

Generell gilt bei der Prüfungsplanung genau wie bei der Veranstaltungsplanung das First-come-first-served-Prinzip: Wer sich bei der Planung ein paar Tage Zeit lässt, riskiert, dass die „besten“ Termine schon weg sind ...

Teil II

Unter der Haube

Wir erläutern in den folgenden Kapiteln den Aufbau der vier Webseiten

- Veranstaltungsdarstellung
- Prüfungsdarstellung
- Veranstaltungsplanung
- Prüfungsplanung

Die ersten beiden Seiten stellen die Veranstaltungs- bzw. Prüfungspläne für die Nutzerin (Dozentin oder Studentin) dar, während die letzten beiden Seiten von der Dozentin genutzt werden, um ihre Veranstaltungen bzw. Prüfungen zu planen.

3 Veranstaltungsdarstellung

3.1 Blockschaltbild

Abbildung 3.1 zeigt das Blockschaltbild der Veranstaltungsdarstellung.

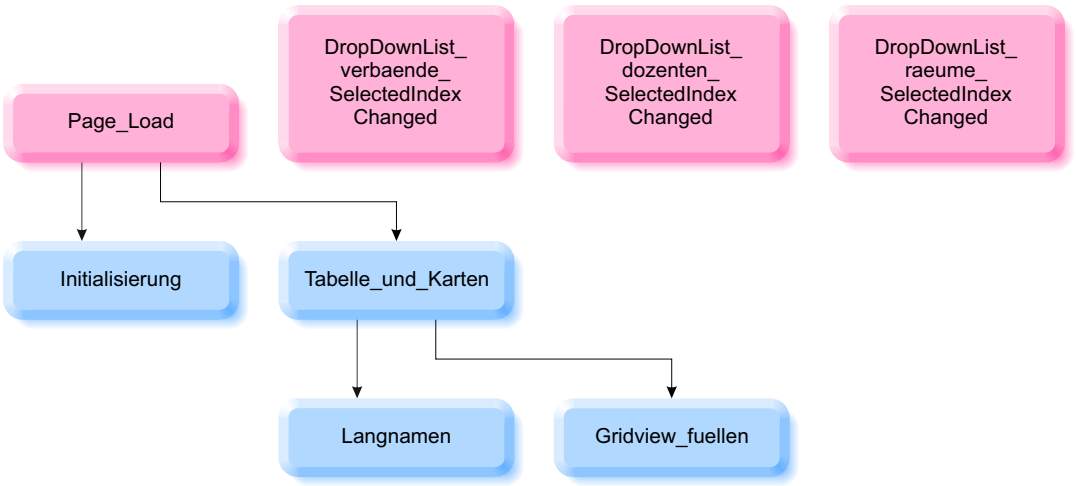


Abbildung 3.1: Blockschaltbild der Veranstaltungsdarstellung

wie es für die Veranstaltungsdarstellungsseite `veranstaltungen.aspx` verwendet wird. Die Unterschiede zur Prüfungsdarstellungsseite beschreiben wir unter Prüfungsdarstellung.

Die roten Blöcke beinhalten Unterprogramme, die vom Laufzeitsystem aufgerufen werden, wenn die Webseite geladen wird oder wenn Schaltflächen gedrückt werden. Die roten Blöcke rufen dann gegebenenfalls die Unterprogramme in den blauen Blöcken auf.

Wenn die Webseite zum ersten Mal geöffnet wird, ruft `Page_Load` das Unterprogramm `Initialisierung` auf, in dem wir die Dropdownlisten aus den XML-Dateien befüllen. Außerdem ruft `Page_Load` das Unterprogramm `Tabelle_und_Karten` auf, in dem wir den Stundenplan mit Leben füllen, die Detailkarten erzeugen und die Veranstaltungen tabellarisch darstellen.

`Tabelle_und_Karten` verwendet dabei das Unterprogramme `Langnamen`, in dem wir weitere Informationen zu Dozentinnen, Semesterverbände und Räume ausgeben und

das Unterprogramm Gridview_fuellen, in dem wir – auf einem Desktoprechner – die Veranstaltungstabelle anzeigen.

Die Unterprogramme DropDownList_verbaende_SelectedIndexChanged, DropDownList_dozenten_SelectedIndexChanged und DropDownList_raeume_SelectedIndexChanged werden vom Laufzeitsystem immer dann aufgerufen, wenn die Nutzerin einen neuen Eintrag in einer der drei Dropdownlisten ausgewählt hat.

3.2 veranstaltungen.aspx

Wir nutzen für alle aktiven Seiten das ASP.NET Framework [5]. Dazu gehört jeweils sowohl die eigentliche ASPX-Seite, die wir in diesem Kapitel beschreiben als auch die den aktiven Code beinhaltende Seite veranstaltungen.aspx.vb.

Eine ASPX-Seite beginnt daher üblicherweise mit einer (hier umbrochenen) Direktive, die die Sprache festlegt und erklärt, wo der Code der Seite zu finden ist:

```
<%@ Page
  Language="VB"
  AutoEventWireup="false"
  CodeFile="veranstaltungen.aspx.vb"
  Inherits="veranstaltungen" %>
```

Danach folgt wie gewohnt die Deklaration, dass es sich um eine HTML5-Seite handelt:

```
<!DOCTYPE html>
```

Da wir das W3CSS-Framework [6] verwenden (s. u.), können wir in der Direktive gleich festlegen, dass der gesamte Text der Seite eine dunkelgraue Farbe haben soll:

```
<html class="w3-text-dark-gray">
```

Im Kopf der Seite mit dem Server-Attribut

```
<head runat="server">
```

definieren wir als erstes, dass die Seite auf Mobiltelefonen und Tablets genauso aussehen soll wie auf Desktoprechnern („Responsive Web Design“, „Mobile First“)

```
  <meta
    name="viewport"
    content="width=device-width, initial-scale=1" />
```

und dass wir als Zeichenkodierung UTF-8 verwenden:

```
  <meta charset="utf-8" >
```

Es folgt der Titel der Seite

```
  <title>Veranstaltungen</title>
```

der Link zum W3CSS-Framework [6]

```
  <link
    rel="stylesheet"
    href="https://www.w3schools.com/w3css/4/w3.css" />
```

und der Link zu unseren eigenen Stilvorlagen (iplan.css)

```
  <link
    rel="stylesheet"
    href="iplan.css" />
</head>
```

Im Körper der Seite legen wir unter Verwendung der W3CSS-Klasse w3-content als erstes fest, dass die Seite zentriert mit einer Defaultbreite von 980 Pixeln dargestellt werden soll:

```
<body class="w3-content">
```

Die Eingaben der Nutzerin werden in ASP.NET üblicherweise in einem Formular gesammelt und auf dem Server verarbeitet:

```
  <form
    id="form1"
    runat="server">
```

Um in Abbildung 2.1 die drei Dropdownlisten für die Verbände, Dozentinnen und Räume in einer Zeile nebeneinander anzuordnen, verwenden wir die dafür vorgesehene W3CSS-Klasse w3-bar:

```
    <div class="w3-bar">
```

Jede der drei folgenden Dropdownlisten ist ein Element (w3-bar-item) der oben definierten w3-bar-Zeile, verwendet kleine Schrift (w3-small) auf weißem Hintergrund (w3-white) und löst ein automatisches Senden an den Server aus, wenn die Nutzerin einen neuen Eintrag auswählt (AutoPostBack="True"),

```
      <asp:DropDownList
        ID="DropDownList_verbaende"
        runat="server"
        AutoPostBack="True"
        class="w3-bar-item w3-white w3-small">
      </asp:DropDownList>
      <asp:DropDownList
        ID="DropDownList_dozenten"
        runat="server"
        AutoPostBack="True"
        class="w3-bar-item w3-white w3-small">
      </asp:DropDownList>
      <asp:DropDownList
```

```
        ID="DropDownList_raeume"
        runat="server"
        AutoPostBack="True"
        class="w3-bar-item w3-white w3-small">
      </asp:DropDownList>
    </div>
```

Die nächste Zeile in Abbildung 2.1 ist die „Überschrift“ des Stundenplans, die wir später dynamisch mit dem ausgewählten Semester und dem Verband, der Dozentin oder dem Raum füllen. Diese Überschrift setzen wir petrolfarben (`w3-teal`) mit passenden Polsterungen über die gesamte Seite (`w3-container`).

```
    <div class="w3-container w3-teal">
      <asp:Label
        ID="Label_ueberschrift"
        runat="server">
      </asp:Label>
    </div>
```

Als nächstes definieren wir das Gerüst des Stundenplans, den wir später ebenfalls dynamisch in `veranstaltungen.aspx.vb` füllen. Der Plan soll sich automatisch auf jedem Gerät an die Seitenbreite anpassen (`w3-responsive`)

```
    <div class="w3-responsive">
```

und besteht aus einer ASP.NET-Tabelle (`asp:Table`), die wir mit kleiner Schrift (damit die Tabelle auch von allen Telefonen im Hochformat dargestellt werden kann) von W3CSS hübsch (`w3-table-all`), mit alternierenden Zeilenhintergründen, ... formatieren lassen :

```
      <asp:Table
        ID="Table_veranstaltungen"
        runat="server"
        class="w3-table-all w3-small">
```

Die Tabellenkopfzeile setzen wir fett

```
        <asp:TableHeaderRow
          ID="Ueberschrift"
          runat="server"
          Font-Bold="True">
```

und füllen sie mit einer Leerzelle

```
          <asp:TableHeaderCell
            runat="server">
          </asp:TableHeaderCell>
```

und den Abkürzungen der Wochentage:

```
          <asp:TableHeaderCell
            runat="server">
          Mo
          </asp:TableHeaderCell>
          <asp:TableHeaderCell
            runat="server">
          Di
          </asp:TableHeaderCell>
          <asp:TableHeaderCell
            runat="server">
          Mi
          </asp:TableHeaderCell>
          <asp:TableHeaderCell
            runat="server">
          Do
          </asp:TableHeaderCell>
          <asp:TableHeaderCell
            runat="server">
          Fr
          </asp:TableHeaderCell>
        </asp:TableHeaderRow>
```

Die restlichen Tabellenzeilen sind identisch aufgebaut. In die erste Zelle einer Zeile formatieren wir die Anfangszeit des Blockes fett:

```
        <asp:TableRow
          ID="Block1"
          runat="server">
          <asp:TableCell
            runat="server"
            Font-Bold="True">
          08:00
          </asp:TableCell>
```

Die übrigen Zellen (eine pro Wochentag) sind aktuell noch leer und werden ebenfalls später in veranstaltungen.aspx.vb gefüllt:

```
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
        </asp:TableRow>
```

Das gleiche Spiel wiederholt sich für jede Anfangszeit bis zum achten Block um 20:30 Uhr:

```
        <asp:TableRow
```

```
          ID="Block8"
          runat="server">
          <asp:TableCell
            runat="server"
            Font-Bold="True">
          20:30
          </asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
          <asp:TableCell runat="server"></asp:TableCell>
        </asp:TableRow>
      </asp:Table>
    </div>
```

Als nächstes erzeugen wir die petrolfarbene Zeile direkt unter dem Stundenplan, die zusätzliche Informationen (beispielsweise den vollständigen Name der Dozentin) über den Plan enthält (Abbildung 2.1):

```
    <div class="w3-container w3-teal w3-small">
      <asp:Label
        ID="Label_langname"
        runat="server">
      </asp:Label>
    </div>
```

Das nächste Element ist ein Platzhalter, in dem wir später in `veranstaltungen.aspx.vb` die Detailkarten jeder Veranstaltung erzeugen:

```
    <div class="w3-container w3-small">
      <asp:PlaceHolder
        ID="PlaceHolder_veranstaltungen"
        runat="server"></asp:PlaceHolder>
    </div>
```

Abschließend schaffen wir noch Platz für das Gridview, in dem wir später alle Veranstaltungen tabellarisch ausgeben:

```
    <p>
      <asp:GridView
        ID="GridView_veranstaltungen"
        runat="server"
        CssClass="Grid">
      </asp:GridView>
    </p>
  </form>
</body>
</html>
```

3.3 veranstaltungen.aspx.vb

Den aktiven, dynamischen Code („Code-Behind“) der Veranstaltungsseite, den wir im Folgenden beschreiben werden, haben wir in Visual Basic geschrieben und in der Datei `veranstaltungen.aspx.vb` abgelegt.

3.3.1 Globale Variablen

Ja, globale Variablen sind schlechter Programmierstil, aber sie machen das Leben so viel einfacher, wenn Informationen zwischen mehreren Unterprogrammen ausgetauscht werden sollen. Wir leisten uns daher den dekadenten Luxus, hier ein paar überall verfügbare Variablen zu deklarieren:

```
Public aktuelles_semester As String
Public aktuelle_art As String
Public aktueller_code As String
```

3.3.2 Page_Load

Das als „Hauptprogramm” fungierende

```
Protected Sub Page_Load(
  sender As Object,
  e As EventArgs) _
  Handles Me.Load
```

wird aufgerufen, wenn die Nutzerin die Webseite `veranstaltungen.aspx` anfordert und liest als erstes den Query-String aus:

```
aktuelles_semester = Request.QueryString("semester")
aktuelle_art = Request.QueryString("art")
aktueller_code = Request.QueryString("code")
```

Wenn die Seite zum ersten Mal aufgerufen wird

```
If Not IsPostBack Then
```

rufen wir das Initialisierungsunterprogramm auf:

```
  Initialisierung()
End If
```

Das Unterprogramm `Tabelle_und_Karten` wird schließlich bei jedem Aufruf der Seite ausgeführt:

```
Tabelle_und_Karten()
End Sub
```

3.3.3 Initialisierung

Im Unterprogramm

```
Private Sub initialisierung()
```

füllen wir die Dropdownlisten mit den Verbänden, Dozentinnen und Räumen. Dazu lesen wir die Datei `verbaende.xml`, in der die Planerin alle im aktuellen Plan auftretenden Semesterverbände definiert hat

```
Dim verbaende_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/verbaende.xml"))
```

und verwenden eine linq2xml-Abfrage, um eine Liste der Abkürzungen aller Semesterverbände zu erhalten:

```
Dim alle_verbaende =
  From verband In verbaende_xml.<verband>
  Order By verband.<abk>.Value
  Select verband.<abk>.Value
```

Diese Liste verwenden wir dann als Datenquelle für die in `veranstaltungen.aspx` definierte Dropdownliste der Verbände

```
DropDownList_verbaende.DataSource = alle_verbaende
```

Mit

```
DropDownList_verbaende.DataBind()
```

stellen wir die Verbände in der Dropdownliste schließlich dar.

Das gleiche Vorgehen führen wir für die Dropdownlisten der Dozentinnen

```
Dim dozenten_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/dozenten.xml"))
Dim alle_dozenten =
  From dozent In dozenten_xml.<dozent>
  Order By dozent.<abk>.Value
  Select dozent.<abk>.Value
DropDownList_dozenten.DataSource = alle_dozenten
DropDownList_dozenten.DataBind()
```

und der Räume durch:

```
Dim raeume_xml = XElement.Load(
  MapPath("xml/" &
```

```
  aktuelles_semester &
  "/veranstaltung/raeume.xml"))
Dim alle_raeume =
  From raum In raeume_xml.<raum>
  Order By raum.<abk>.Value
  Select raum.<abk>.Value
DropDownList_raeume.DataSource = alle_raeume
DropDownList_raeume.DataBind()
```

Abschließend führen wir in jeder Dropdownliste einen ersten Eintrag als „Überschrift“ ein, der angezeigt wird, solange die Nutzerin noch keinen anderen Eintrag ausgewählt hat (Abbildung 2.1):

```
DropDownList_verbaende.Items.Insert(0, "Verbände")
DropDownList_dozenten.Items.Insert(0, "Lehrende")
DropDownList_raeume.Items.Insert(0, "Räume")
End Sub
```

3.3.4 DropDownList_verbaende_SelectedIndexChanged

Das Unterprogramm

```
Private Sub DropDownList_verbaende_SelectedIndexChanged(
  sender As Object,
  e As EventArgs) _
  Handles DropDownList_verbaende.SelectedIndexChanged
```

wird aufgerufen, wenn die Nutzerin einen neuen Semesterverband auswählt. Wir verwenden dann den ausgewählten Verband (`DropDownList_verbaende.SelectedValue`), um einen Link zur Seite aufzubauen, in dem der Query-String den ausgewählten Verband beschreibt

```
Dim ziel As String =
  "https://m-server.fk5.hs-bremen.de/iplan/veranstaltungen.aspx"
      &
  "?semester=" & aktuelles_semester &
  "&art=verband" &
  "&code=" & DropDownList_verbaende.SelectedValue
```

und rufen die Seite nochmals mit dem aktuellen Query-String auf:

```
Response.Redirect(ziel)
End Sub
```

Auf diese Weise kann die Nutzerin den ausgewählten[1] Plan direkt „bookmarken“, indem sie die gesamte angezeigte URL aus der Adresszeile des Browsers kopiert.

[1] Wenn die Nutzerin versteht, wie der Query-String aufgebaut ist, kann sie natürlich auch leicht den Link zu jedem anderen Plan konstruieren.

3.3.5 DropDownList_dozenten_SelectedIndexChanged

Das Prozedere bei der Auswahl einer neuen Dozentin ist analog zur Auswahl eines neuen Verbandes (DropDownList_verbaende_SelectedIndexChanged):

```
Private Sub DropDownList_dozenten_SelectedIndexChanged(
  sender As Object,
  e As EventArgs) _
  Handles DropDownList_dozenten.SelectedIndexChanged
Dim ziel As String =
  "https://m-server.fk5.hs-bremen.de/iplan/veranstaltungen.aspx"
      &
  "?semester=" & aktuelles_semester &
  "&art=dozent" &
  "&code=" & DropDownList_dozenten.SelectedValue
Response.Redirect(ziel)
End Sub
```

3.3.6 DropDownList_raeume_SelectedIndexChanged

Auch das Prozedere bei der Auswahl eines neuen Raumes ist natürlich analog zur Auswahl eines neuen Verbandes (DropDownList_verbaende_SelectedIndexChanged):

```
Private Sub DropDownList_raeume_SelectedIndexChanged(
  sender As Object,
  e As EventArgs) _
  Handles DropDownList_raeume.SelectedIndexChanged
Dim ziel As String =
  "https://m-server.fk5.hs-bremen.de/iplan/veranstaltungen.aspx"
      &
  "?semester=" & aktuelles_semester &
  "&art=raum" &
  "&code=" & DropDownList_raeume.SelectedValue
Response.Redirect(ziel)
End Sub
```

3.3.7 Tabelle_und_Karten

Das Unterprogramm

```
Private Sub Tabelle_und_Karten()
```

wird bei jedem Seitenaufruf von Page_Load aufgerufen. Als erstes lesen wir „für später“ die Moduldatenbank der Abteilung Maschinenbau (modul.xml), in der wir im Rahmen eines anderen Projektes die Beschreibungen aller Module abgelegt haben:

```
Dim module_xml = XElement.Load(MapPath("../modul/xml/modul.xml"))
```

Als nächstes löschen wir alle aus vorherigen Aufrufen möglicherweise vorhandenen Stundenplaneinträge

```
For i_zeit = 1 To 8
  For i_tag = 1 To 5
    Table_veranstaltungen.Rows(i_zeit).Cells(i_tag).Text = ""
  Next
Next
```

und erzeugen die petrolfarbene Überschriftzeile der Seite (Abbildung 2.1) in zwei Schritten. Im ersten Schritt geben wir das aktuelle Semester aus, das wir während der Initialisierung aus dem Query-String erhalten haben:

```
Label_ueberschrift.Text =
  "Veranstaltungen " &
  aktuelles_semester.ToUpper
```

Im zweiten Schritt schreiben wir die Abkürzung des Verbandes, der Dozentin oder des Raumes, wenn er aus dem Query-String gelesen werden konnte:

```
If Not aktueller_code = "" Then
  Label_ueberschrift.Text &=
  ", " &
  aktueller_code
  Langnamen()
End If
```

Innerhalb der If-Bedingung verwenden wir außerdem das Unterprogramm `Langnamen`, um die petrolfarbene Zeile unter dem Stundenplan auszugeben (Abbildung 2.1).

In den nächsten Schritten lesen wir die XML-Dateien aus, die die Wochentage, die Anfangszeiten und die aktuellen Veranstaltungen beinhalten. Dazu lesen wir als erstes die XML-Datei der Tage

```
Dim tage_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/tage.xml"))
```

und verwenden dann eine linq2xml-Abfrage, um eine Liste aller Wochentage zu erhalten:

```
Dim alle_tage =
  From tag In tage_xml.<tage>.<tag>
  Select tag.Value
```

Das gleiche wiederholen wir für die Anfangszeiten

```
Dim zeiten_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/zeiten.xml"))
Dim alle_anfangszeiten =
  From zeit In zeiten_xml.<zeit>.<beginn>
  Select zeit.Value
```

Auch für die Veranstaltungen lesen wir zuerst die entsprechende XML-Datei:

```
Dim veranstaltungen_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/veranstaltungen.xml"))
```

Wir suchen jetzt aber nur die schon verplanten Veranstaltungen des ausgewählten Verbandes, der ausgewählten Dozentin oder des ausgewählten Raumes. Abbildung 3.2 zeigt das Datenmodell einer Veranstaltung aus der XML-Datei.

```
<veranstaltung>
  <id>8ea90ff5-fd9f-43bb-abf1-7e9d5fae6080</id>
  <modul>MATH1</modul>
  <tag>Freitag</tag>
  <zeit>11:30 - 13:00</zeit>
  <aulis>https://aulis.hs-bremen.de/goto.php?
  target=crs_1153133&amp;client_id=hsbremen</aulis>
  <bemerkung></bemerkung>
  <faktor>0.5</faktor>
  <dozent>BUCH</dozent>
  <verband>ILST_1</verband>
  <verband>MDIG_1</verband>
  <raum>SI 52</raum>
</veranstaltung>
```

Abbildung 3.2: Eine einzelne Veranstaltung in der Datei veranstaltungen.xml

Wenn die Nutzerin also jetzt beispielsweise über den Query-String `art=dozent` und `code=BUCH` ausgewählt hat, suchen wir in der XML-Datei nach allen Elementen der Form `<dozent>BUCH</dozent>` und lesen dann mittels `Select code.Parent` deren Elternelement, also die Veranstaltung selbst:

```
Dim alle_verplanten_veranstaltungen =
  From code In veranstaltungen_xml.<veranstaltung>.Elements(
     aktuelle_art)
  Where code.Value = aktueller_code _
  And Not code.Parent.<tag>.Value = "" _
  And Not code.Parent.<zeit>.Value = ""
```

```
  Order By
    alle_tage.ToList.IndexOf(code.Parent.<tag>.Value),
    code.Parent.<zeit>.Value,
    code.Parent.<modul>.Value,
    code.Parent.<dozent>.Value
  Select code.Parent
```

Dabei beschränken wir uns mit Hilfe von

```
And Not code.Parent.<tag>.Value = "" _
And Not code.Parent.<zeit>.Value = ""
```

auf verplante Elemente, bei denen also schon ein Wochentag und eine Blockanfangszeit eingetragen ist. Mit

```
alle_tage.ToList.IndexOf(code.Parent.<tag>.Value),
```

sortieren wir als erstes nach Wochentagen, was aber – da ja „Montag“ alphabetisch erst nach „Dienstag“ kommt – diese etwas umständliche Konstruktion bedingt.

Nachdem wir den Detailkartenzähler initialisiert haben

```
Dim div_card_zaehler = 0
```

beginnen wir die große Schleife über alle (schon verplanten) Veranstaltungen

```
For Each veranstaltung In alle_verplanten_veranstaltungen
```

Wir besorgen uns den (nullbasierten) Index des Wochentages der aktuellen Veranstaltung

```
  Dim i_tag As Integer =
    alle_tage.ToList.IndexOf(
    veranstaltung.<tag>.Value)
```

und den Index ihrer Blockanfangszeit. Wir verwenden dazu die ersten fünf Zeichen des `<zeit>`-Elements in Abbildung 3.2:

```
  Dim i_zeit As Integer =
    alle_anfangszeiten.ToList.IndexOf(
    veranstaltung.<zeit>.Value.Substring(0, 5))
```

Mit diesen Zeilen- und Spaltenindizes können wir die aktuelle Zelle im Stundenplan ansprechen:

```
  Dim aktuelle_zelle =
    Table_veranstaltungen.
    Rows(i_zeit + 1).
    Cells(i_tag + 1)
```

Wir müssen jetzt unterscheiden, ob in der aktuellen Zelle des Stundenplanes nur eine Veranstaltung oder mehrere parallele Veranstaltungen eingetragen werden sollen. Wenn die Zelle momentan noch leer ist

```
  If aktuelle_zelle.Text = "" Then
```

dann tragen wir die Abkürzung der aktuellen Veranstaltung in den Plan ein und geben ihr gleich einen Hyperlink zur passenden Detailkarte mit auf den Weg:

```
    aktuelle_zelle.Text =
    "<a href='#div_card_id_" &
    div_card_zaehler &
    "'>" &
    veranstaltung.<modul>.Value &
    "</a>"
```

Wenn in der Zelle aber schon etwas eingetragen ist, wenn also mehrere Veranstaltungen zum gleichen Zeitpunkt stattfinden

```
  Else
```

dann spendieren wir über dem neuen Eintrag noch eine Trennungszeile (-------) (Abbildung 2.6):

```
    aktuelle_zelle.Text &=
    "</br> ------- </br>" &
    "<a href='#div_card_id_" &
    div_card_zaehler &
    "'>" &
    veranstaltung.<modul>.Value &
    "</a>"
  End If
```

Jetzt schreiben wir noch den Raum oder die Dozentin. Wenn es sich um einen Raumplan handelt

```
  If aktuelle_art = "raum" Then
```

tragen wir die Dozentin in der nächsten Zeile der Zelle ein:

```
    aktuelle_zelle.Text &=
    "</br>" &
    veranstaltung.<dozent>.Value
```

Wenn es sich um den Plan eines Verbandes oder einer Dozentin handelt, tragen wir den Raum ein:

```
  Else
    aktuelle_zelle.Text &=
    "</br>" &
    veranstaltung.<raum>.Value
  End If
```

Da an einer Veranstaltung mehrere Verbände, Dozentinnen oder Räume beteiligt sein können, lesen wir die beteiligten Verbände in Form eines String-Arrays und verpassen

jedem Verband auch gleich einen passenden Hyperlink, sodass die Nutzerin auf den noch zu erstellenden Detailkarten den entsprechenden Verbandsplan direkt aufrufen kann:

```
Dim verband_array As String() =
  (From verband In veranstaltung.<verband>
  Select
    "<a href='veranstaltungen.aspx?semester=" &
    aktuelles_semester &
    "&art=verband&code=" &
    verband.Value &
    "'>" &
    verband.Value &
    "</a>"
    ).ToArray
```

Das gleiche machen wir mit den an der Veranstaltung beteiligten Dozentinnen

```
Dim dozent_array As String() =
  (From dozent In veranstaltung.<dozent>
  Select
    "<a href='veranstaltungen.aspx?semester=" &
    aktuelles_semester &
    "&art=dozent&code=" &
    dozent.Value &
    "'>" &
    dozent.Value &
    "</a>"
    ).ToArray
```

und den beteiligten Räumen:

```
Dim raum_array As String() =
  (From raum In veranstaltung.<raum>
  Select
    "<a href='veranstaltungen.aspx?semester=" &
    aktuelles_semester &
    "&art=raum&code=" &
    raum.Value &
    "'>" &
    raum.Value &
    "</a>"
    ).ToArray
```

Jetzt erzeugen wir die Detailkarte der aktuellen Veranstaltung und füllen sie mit ihren Veranstaltungsdaten. Dazu erzeugen wir eine die Karte als einfaches HTML- -Element

```
Dim div_card As New HtmlGenericControl("div")
```

dessen Klassenattribut wir gemäß W3CSS als hübsche Karteikarte mit runden Ecken definieren:

```
div_card.Attributes.Add(
  "class", "w3-panel w3-card w3-round-large")
```

Der Karte geben wir eine eindeutige ID unter Verwendung des Zählers

```
div_card.ID = "div_card_id_" & div_card_zaehler
```

den wir oben initialisiert haben und hier inkrementieren:

```
div_card_zaehler += 1
```

Da wir beim Abruf eines Moduls aus der Moduldatenbank der Abteilung Maschinenbau im Query-String nicht die Abkürzung des Moduls (beispielsweise `MATH1`), sondern eine GUID (beispielsweise `8a6da1b9-3e77-40a7-a166-e031ed86d9fc`) verwenden, müssen wir in der Moduldatenbank erst die GUID des aktuellen Moduls mit Hilfe seiner Abkürzung herausfinden, bevor wir den Hyperlink zusammenbasteln können, der die Modulbeschreibung des Moduls auf der Detailkarte erklickbar macht:

```
Dim modul_mit_link =
  From modul In module_xml.<modul>
  Where modul.<modul_abk>.Value = veranstaltung.<modul>.Value
  Select
  "<a href='https://m-server.fk5.hs-bremen.de/modul/
     beschreibung.aspx?modul_id=" &
  modul.<modul_id>.Value &
  "'>" &
  veranstaltung.<modul>.Value &
  "</a>"
```

Wir tragen dann die ersten Zeilen (Modul, Tag, Uhrzeit, Verbände, Dozentinnen) auf der Detailkarte ein:

```
div_card.InnerHtml =
  "<b>" &
  modul_mit_link.FirstOrDefault &
  "</b></br>" &
  "Tag: " &
  veranstaltung.<tag>.Value &
  "</br>" &
  "Uhrzeit: " &
  veranstaltung.<zeit>.Value &
  "</br>" &
  "Verbände: " &
  Join(verband_array, ", ") &
  "</br>" &
  "Lehrende: " &
  Join(dozent_array, ", ")
```

Da die Modulabkürzung in der Moduldatenbank kein „Primärschlüssel“ ist, wäre es theoretisch denkbar, dass die Abfrage mehrere Module mit der gleichen Abkürzung zurückliefert. Wir verwenden hier mit `FirstOrDefault` das erste zurückgelieferte Modul, was ein Fehler[2] sein könnte.

Wenn mehrere Verbände oder Dozentinnen an einer Veranstaltung beteiligt sind, möchten wir die einzelnen Teilnehmer gerne – mit Komma getrennt – auflisten. Wir haben oben schon die entsprechenden String-Arrays erzeugt und können jetzt den freundlichen Befehl `Join` auf das Array anwenden, um automatisch die komma-separierte Liste zu erzeugen (Abbildung 2.11).

Da wir nach wie vor auch Online-Veranstaltungen durchführen, fügen wir „Räume: ...“ nur dann auf der Detailkarte ein, wenn die Dozentin auch tatsächlich Räume ausgewählt hat:

```
If raum_array.Count > 0 Then
  div_card.InnerHtml &=
  "</br>" &
  "Räume: " &
  Join(raum_array, ", ")
End If
```

Wenn die Dozentin den Defaulteintrag `https://aulis.hs-bremen.de/` konkretisiert hat

```
If Not veranstaltung.<aulis>.Value =
  "https://aulis.hs-bremen.de/" Then
```

dann fügen wir ihn auf der Karte ein:

```
  div_card.InnerHtml &=
  "</br>" &
  "Link: " &
  "<a href=" &
  veranstaltung.<aulis>.Value &
  ">Aulis</a>"
End If
```

Wenn die Dozentin eine Bemerkung für diese Veranstaltung eingetragen hat, geben wir sie auf der Karte aus:

```
If Not veranstaltung.<bemerkung>.Value = "" Then
  div_card.InnerHtml &=
  "</br>" &
  "Bemerkung: " &
  veranstaltung.<bemerkung>.Value
End If
```

[2] Der Autor, der auch gleichzeitig der Administrator der Moduldatenbank ist, hat sich in den vergangenen Jahren bemüht, keine Modulabkürzungsdubletten in der Datenbank zu erzeugen, sodass die Abkürzung de facto „eigentlich“ eindeutig sein sollte; aber nach Murphys Gesetz ...

Hauptamtliche Dozentinnen müssen Ihr Lehrdeputat alljährlich abrechnen (Abschnitt 2.4.5). Dazu tragen wir bei Dozentinnen gegebenenfalls einen von der Planerin gesetzten Anrechnungsfaktor ein (Abbildung 2.8):

```
  If aktuelle_art = "dozent" And
    Not veranstaltung.<faktor>.Value = "" Then
    div_card.InnerHtml &=
    "</br>" &
    "Faktor: " &
    veranstaltung.<faktor>.Value
  End If
```

Schließlich können wir die fertige Detailkarte dem Platzhalter auf der Webseite hinzufügen:

```
  PlaceHolder_veranstaltungen.Controls.Add(div_card)
Next
```

Auf Desktoprechnern geben wir jetzt noch alle Veranstaltungen in tabellarischer Form aus (Gridview_fuellen):

```
If Not Request.Browser.IsMobileDevice Then
  Gridview_fuellen()
End If
End Sub
```

3.3.8 Langnamen

Das Unterprogramm

```
Private Sub Langnamen()
```

wird von Tabelle_und_Karten aufgerufen, um die petrolfarbene Zeile unter dem Stundenplan auszugeben. Dazu unterscheiden wir, ob es sich um einen Verbands-, Dozenten- oder Raumplan handelt. Wenn es um einen Verbandsplan geht

```
If aktuelle_art = "verband" Then
```

lesen wir die aktuelle Verbandsdatenbank

```
  Dim verbaende_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/verbaende.xml"))
```

extrahieren den aktuellen Verband

```
  Dim aktueller_verband =
    From verband In verbaende_xml.<verband>
    Where verband.<abk>.Value = aktueller_code
```

und schreiben den Langnamen des Verbands in die Zeile unter dem Stundenplan:

```
  Label_langname.Text =
    aktueller_verband.<name>.Value
```

Auch beim Plan einer Dozentin lesen wir die entsprechende Datenbank und die aktuelle Dozentin

```
ElseIf aktuelle_art = "dozent" Then
  Dim dozenten_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/dozenten.xml"))
  Dim aktueller_dozent =
    From dozent In dozenten_xml.<dozent>
    Where dozent.<abk>.Value = aktueller_code
```

Hier geben wir allerdings den Titel, den Vornamen und den Nachnamen der Dozentin aus:

```
  Label_langname.Text =
    aktueller_dozent.<titel>.Value &
    " " &
    aktueller_dozent.<vorname>.Value &
    " " &
    aktueller_dozent.<name>.Value
```

Bei einem Raumplan

```
ElseIf aktuelle_art = "raum" Then
  Dim raeume_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/raeume.xml"))
  Dim aktueller_raum =
    From raum In raeume_xml.<raum>
    Where raum.<abk>.Value = aktueller_code
```

geben wir den Raumnamen, den Standort und die Anzahl der Sitzplätze aus:

```
  Label_langname.Text =
    aktueller_raum.<name>.Value &
    ", " &
    aktueller_raum.<ort>.Value &
    " (" &
    aktueller_raum.<anzahl>.Value &
    " Plätze)"
End If
End Sub
```

3.3.9 Gridview_fuellen

Das Unterprogramm

```
Sub Gridview_fuellen()
```

wird von Tabelle_und_Karten aufgerufen, wenn die Seite auf einem Desktoprechner dargestellt wird (Abschnitt 2.5), um alle Informationen nochmals tabellarisch aufzulisten. Dazu lesen wir die Veranstaltungsdatenbank

```
Dim veranstaltungen_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/veranstaltung/veranstaltungen.xml"))
```

deklarieren ein Datatable, das wir später als Quelle für das Gridview verwenden werden

```
Dim DataTable_veranstaltungen As New Data.DataTable
```

und lesen alle ausgewählten Veranstaltungen, verplante und unverplante (3.3.7 auf Seite 38):

```
Dim alle_veranstaltungen =
  From code In veranstaltungen_xml.<veranstaltung>.Elements(
     aktuelle_art)
  Where code.Value = aktueller_code
  Order By
    code.Parent.<modul>.Value,
    code.Parent.<dozent>.Value,
    code.Parent.<verband>.Value,
    code.Parent.<tag>.Value,
    code.Parent.<zeit>.Value,
    code.Parent.<raum>.Value
  Select code.Parent
```

Wir erzeugen Tabellenspalten mit sinnvollen Überschriften

```
DataTable_veranstaltungen.Columns.Add("Modul")
DataTable_veranstaltungen.Columns.Add("Dozent")
DataTable_veranstaltungen.Columns.Add("Verband")
DataTable_veranstaltungen.Columns.Add("Tag")
DataTable_veranstaltungen.Columns.Add("Zeit")
DataTable_veranstaltungen.Columns.Add("Raum")
DataTable_veranstaltungen.Columns.Add("Aulis")
DataTable_veranstaltungen.Columns.Add("Bemerkung")
```

und fügen für den Fall, dass es sich um einen Dozentinplan handelt, noch die Anrechnungsfaktor-Spalte hinzu:

```
If aktuelle_art = "dozent" Then
  DataTable_veranstaltungen.Columns.Add("Faktor")
End If
```

Jetzt beginnt die Schleife über alle Veranstaltungen:

```
For Each veranstaltung In alle_veranstaltungen
```

Da an einer Veranstaltung auch mehrere Verbände beteiligt sein können, lesen wir alle Verbände der aktuellen Veranstaltung in ein String-Array:

```
  Dim verband_array As String() =
    (From verband In veranstaltung.<verband>
    Select verband.Value).ToArray
```

Das gleiche machen wir mit allen an der Veranstaltung beteiligten Räumen

```
  Dim raum_array As String() =
    (From raum In veranstaltung.<raum>
    Select raum.Value).ToArray
```

und Dozentinnen:

```
  Dim dozent_array As String() =
    (From dozent In veranstaltung.<dozent>
    Select dozent.Value).ToArray
```

Wir erzeugen eine neue Zeile im Datatable

```
  Dim DataRow_veranstaltung As Data.DataRow =
     DataTable_veranstaltungen.NewRow()
```

und füllen die Zeile mit den Daten der aktuellen Veranstaltung:

```
  DataRow_veranstaltung("Modul") = veranstaltung.<modul>.Value
  DataRow_veranstaltung("Dozent") = Join(dozent_array, ", ")
  DataRow_veranstaltung("Verband") = Join(verband_array, ", ")
  DataRow_veranstaltung("Tag") = veranstaltung.<tag>.Value
  DataRow_veranstaltung("Zeit") = veranstaltung.<zeit>.Value
  DataRow_veranstaltung("Raum") = Join(raum_array, ", ")
  DataRow_veranstaltung("Aulis") = veranstaltung.<aulis>.Value
  DataRow_veranstaltung("Bemerkung") = veranstaltung.<bemerkung
     >.Value
```

Dabei verwenden wir den `Join`-Befehl, um die Dozentin-, Verbands- und Raum-String-Arrays in eine komma-separierte Liste umzuwandeln.

Wenn es sich um einen Dozentinplan handelt, schreiben wir zusätzlich noch den Anrechnungsfaktor in die aktuelle Zeile:

```
  If aktuelle_art = "dozent" Then
    DataRow_veranstaltung("Faktor") = veranstaltung.<faktor>.
       Value
  End If
```

Schließlich fügen wir die gefüllte Zeile an das Datatable an

```
  DataTable_veranstaltungen.Rows.Add(DataRow_veranstaltung)
Next
```

verwenden das Datatable als Quelle für das Gridview

```
GridView_veranstaltungen.DataSource = DataTable_veranstaltungen
```

und stellen das Gridview dar:

```
GridView_veranstaltungen.DataBind()
End Sub
```

4 Prüfungsdarstellung

4.1 Blockschaltbild

Abbildung 4.1 zeigt das Blockschaltbild der Prüfungsdarstellung.

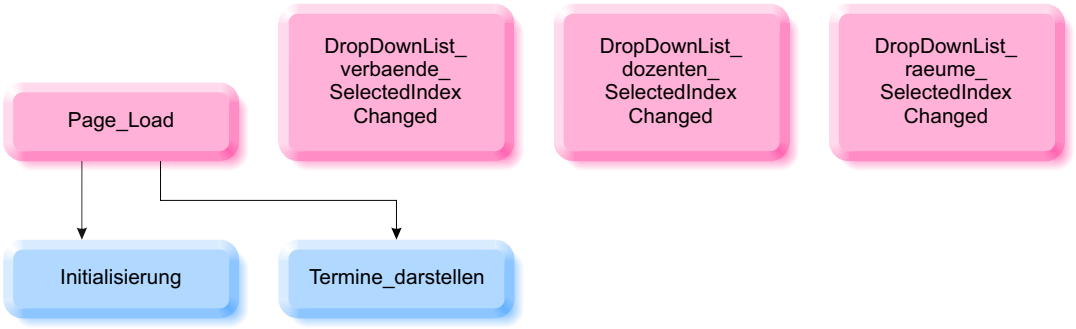


Abbildung 4.1: Blockschaltbild der Prüfungsdarstellung

Es sieht dem Blockschaltbild der Veranstaltungsdarstellung (Abschnitt 3.1) sehr ähnlich; lediglich das Unterprogramm `Termine_und_Karten` wurde durch `Termine_darstellen` ersetzt, das wiederum weder `Langnamen` noch `Gridview_fuellen` aufruft.

4.2 `pruefungen.aspx`

Entsprechend ist die ASPX-Datei der Prüfungsdarstellung mit ihren Dropdownlisten und dem Platzhalter für die Detailkarten analog zu `veranstaltungen.aspx` aufgebaut. Es fehlen natürlich der hier nicht dargestellte Stundenplan samt Langname und das Gridview. Hinzu kommt am Ende der Form ein Platzhalter für den Link zur ICS-Datei:

```
:
:
    <br />
    <div class="w3-container">
      <asp:HyperLink
        ID="HyperLink_ics"
        runat="server"
        Visible="False">
```

```
      </asp:HyperLink>
    </div>
    <br />
  </form>
</body>
</html>
```

4.3 pruefungen.aspx.vb

4.3.1 Page_Load

Das Unterprogramm `Page_Load` der Prüfungsdarstellung ist fast identisch mit dem gleichnamigen Unterprogramm der Veranstaltungsdarstellung (`Page_Load`). Lediglich der Aufruf von `Tabelle_und_Karten` wird gemäß Abbildung 4.1 ersetzt durch den Aufruf von `Termine_darstellen`:

```
:
:
 Termine_darstellen()
End Sub
```

Das Unterprogramm `Initialisierung` und die bei der Auswahl eines neuen Dropdownlisteneintrags aufgerufenen Unterprogramme entsprechen denen der Veranstaltungsdarstellung.

4.3.2 Termine_darstellen

Das Unterprogramm

```
Sub Termine_darstellen()
```

wird bei jedem Aufruf der Seite ausgeführt. Es erzeugt die Detailkarten und die ICS-Kalenderdatei der Prüfungen. Dazu lesen wir als erstes die Datei `tage.xml` im aktuellen Prüfungsordner (Abbildung 4.2):

```
<root>
  <wochendifferenz>5</wochendifferenz>
  <tage>
    <tag>2023-02-06</tag>
    <tag>2023-02-08</tag>
    <tag>2023-02-10</tag>
    <tag>2023-02-13</tag>
    <tag>2023-02-15</tag>
    <tag>2023-02-17</tag>
    <tag>2023-02-20</tag>
    <tag>2023-02-22</tag>
    <tag>2023-02-24</tag>
  </tage>
  <reserven></reserven>
</root>
```

Abbildung 4.2: `tage.xml` im aktuellen Prüfungsordner

```
Dim tage_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/pruefung/tage.xml"))
```

und extrahieren daraus die für alle Prüfungen geltende Wochendifferenz zwischen dem ersten und dem zweiten Prüfungstermin, die die Planerin in Abhängigkeit von der Länge der vorlesungsfreien Zeit und etwaigen Feiertagen a priori festgelegt hat (Abbildung 4.2):

```
Dim wochendifferenz As Integer = tage_xml.<wochendifferenz>.
   Value
```

Für die Überschrift verwenden wir das im Query-String ausgewählte Semester

```
Label_ueberschrift.Text =
  "Prüfungen " &
  aktuelles_semester.ToUpper
```

und – wenn im Query-String angegeben – die Bezeichnung des Verbands, der Dozentin oder des Raumes:

```
If Not aktueller_code = "" Then
  Label_ueberschrift.Text &=
  ", " &
  aktueller_code
End If
```

Als nächstes beginnen wir mit dem Schreiben der Kalenderdatei. Dazu verwenden wir einen Streamwriter

```
Dim ics_file As System.IO.StreamWriter
```

und verbinden ihn mit einer gegebenenfalls neu zu erzeugenden Datei[1] im Verzeichnis `ics`:

```
ics_file = My.Computer.FileSystem.OpenTextFileWriter(
    MapPath("ics/" & aktueller_code & ".ics"), False)
```

In die Kalenderdatei schreiben wir einen passenden „Header“, der von den prüfungsspezifischen Daten unabhängig ist:

```
ics_file.WriteLine("BEGIN:VCALENDAR")
ics_file.WriteLine("VERSION:2.0")
ics_file.WriteLine("PRODID:iplan")
ics_file.WriteLine("METHOD:PUBLISH")
```

Nachdem wir die Datei gelesen haben, die alle Prüfungen beinhaltet (Abbildung 4.3)

```
Dim veranstaltungen_xml = XElement.Load(MapPath(
  "xml/" &
  aktuelles_semester &
  "/pruefung/veranstaltungen.xml"))
```

```
<veranstaltung>
  <id>74ae9388-eac4-4fd5-9373-416fa5dc657c</id>
  <modul>RPS</modul>
  <tag>22.02.2023</tag>
  <zeit>14:00 - 17:00</zeit>
  <aulis>https://aulis.hs-bremen.de/</aulis>
  <bemerkung></bemerkung>
  <faktor></faktor>
  <dozent>SCHO</dozent>
  <verband>MM_2</verband>
  <raum>M 212</raum>
</veranstaltung>
```

Abbildung 4.3: Ausschnitt aus `veranstaltungen.xml` im Prüfungsordner

extrahieren wir die verplanten (`Not code.Parent.<tag>.Value = ""`) Prüfungen des ausgewählten Verbandes, der ausgewählten Dozentin oder des ausgewählten Raumes:

```
Dim alle_veranstaltungen =
  From code In veranstaltungen_xml.<veranstaltung>.Elements(
     aktuelle_art)
  Where code.Value = aktueller_code _
  And Not code.Parent.<tag>.Value = ""
  Order By
```

[1] Wir verwenden hier nur die Abkürzung des Verbands, der Dozentin oder des Raumes im Dateinamen. Die Wahrscheinlichkeit, dass beispielsweise eine Dozentin und ein Verband oder Raum die gleiche Abkürzung haben, ist verschwindend gering, da Verbände und Räume üblicherweise Zahlen beinhalten. Ja, in einer „professionellen“ Datenbankanwendung würden wir hier beispielsweise echte Primärschlüssel im Dateinamen verwenden, semesterweise sortieren, alte Einträge löschen, ...

```
    Date.Parse(code.Parent.<tag>.Value),
    code.Parent.<zeit>.Value,
    code.Parent.<modul>.Value
  Select code.Parent
```

Dabei sortieren wir in der Reihenfolge Datum, Uhrzeit und Modulname. Die abgespeicherte Datumszeichenkette müssen wir natürlich erst noch mit `Date.Parse` als Datum interpretieren, damit „22.02.2023“ auch wirklich vor 21.03.2023 einsortiert wird.

Wir beginnen dann die Schleife über alle ausgewählten Veranstaltungen

```
For Each veranstaltung In alle_veranstaltungen
```

und berechnen in der Schleife aus dem abgespeicherten ersten Prüfungstermin unter Verwendung der festen Wochendifferenz den zweiten Prüfungstermin:

```
  Dim tag_1 As Date = veranstaltung.<tag>.Value
  Dim tag_2 As Date = tag_1.AddDays(wochendifferenz * 7)
```

Die an der Prüfung beteiligten Verbände

```
  Dim verband_array As String() =
    (From verband In veranstaltung.<verband>
    Select verband.Value).ToArray
```

Dozentinnen

```
  Dim dozent_array As String() =
    (From dozent In veranstaltung.<dozent>
    Select dozent.Value).ToArray
```

und Räume schreiben wir jeweils in ein String-Array:

```
  Dim raum_array As String() =
    (From raum In veranstaltung.<raum>
    Select raum.Value).ToArray
```

Für jede Prüfung erzeugen wir eine Detailkarte als HTML-`<div>`-Element

```
  Dim div_card As New HtmlGenericControl("div")
```

der wir passende W3CSS-Eigenschaften geben

```
  div_card.Attributes.Add("class", "w3-panel w3-card w3-round-
     large")
```

und die wir dann mit den Daten der Prüfung befüllen:

```
  div_card.InnerHtml =
    "<b>" &
    veranstaltung.<modul>.Value &
    "</b></br>" &
```

```
    "1. Termin: " &
    tag_1.ToString("ddd, d.M.yyyy") &
    "</br>" &
    "2. Termin: " &
    tag_2.ToString("ddd, d.M.yyyy") &
    "</br>" &
    "Uhrzeit: " &
    veranstaltung.<zeit>.Value &
    "</br>" &
    "Verbände: " &
    Join(verband_array, ", ") &
    "</br>" &
    "Prüfende: " &
    Join(dozent_array, ", ")
```

Dabei wandeln wir die Datumsangaben mit `"ddd, d.M.yyyy"` wieder in eine lesbare Form um und nutzen den `Join`-Befehl, um die Verbände und Prüferinnen in kommaseparierter Form auszugeben.

Da es auch Online-Prüfungen ohne Raumanforderungen gibt, tragen wir die Raum-Zeile nur dann ein, wenn die Prüferin tatsächlich Räume ausgewählt hat:

```
  If raum_array.Count > 0 Then
    div_card.InnerHtml &=
    "</br>" &
    "Räume: " &
    Join(raum_array, ", ")
  End If
```

Den Aulis-Link tragen wir nur dann ein, wenn er sich vom vorgegebenen Standard (`https://aulis.hs-bremen.de/`) unterscheidet:

```
  If Not veranstaltung.<aulis>.Value = "https://aulis.hs-bremen.
     de/" Then
    div_card.InnerHtml &=
    "</br>" &
    "Link: " &
    "<a href=" & veranstaltung.<aulis>.Value & ">Aulis</a>"
  End If
```

Auch die Bemerkungszeile wird nur dann übertragen, wenn die Dozentin dort wirklich etwas eingetragen hat:

```
  If Not veranstaltung.<bemerkung>.Value = "" Then
    div_card.InnerHtml &=
    "</br>" &
    "Bemerkung: " &
    veranstaltung.<bemerkung>.Value
  End If
```

Jetzt können wir die komplette Prüfungsdetailkarte ausgeben

```
PlaceHolder_pruefungen.Controls.Add(div_card)
```

und uns als nächstes wieder um die Kalenderdatei kümmern. Nachdem wir den festen Kopf der Kalenderdatei oben schon definiert haben, tragen wir jetzt in zwei fast identischen Schritten die Daten des ersten und des zweiten Prüfungstermins ein. Termine werden in einer ICS-Datei von `BEGIN:VEVENT` und `END:VEVENT` eingerahmt. Wir verwenden `SUMMARY` für die Modulbezeichnung, `LOCATION` für die Räume, `DESCRIPTION` für die an der Prüfung beteiligten Verbände und Dozentinnen und `DTSTART` und `DTEND` für die Anfangs- bzw. Endzeit der Prüfung. Außerdem nutzen wir den aktuellen Zeitstempel `now` in `DTSTAMP`. Für den ersten Termin schreiben wir also:

```
ics_file.WriteLine("BEGIN:VEVENT")
ics_file.WriteLine("UID:" &
  Guid.NewGuid.ToString())
ics_file.WriteLine("SUMMARY:" &
  veranstaltung.<modul>.Value)
ics_file.WriteLine("LOCATION:" &
  Join(raum_array, ", "))
ics_file.WriteLine("DESCRIPTION:" &
  "Verbände: " &
  Join(verband_array, ", ") &
  "\n" &
  "Prüfende: " &
  Join(dozent_array, ", "))
ics_file.WriteLine("DTSTART:" &
  tag_1.ToString("yyyyMMdd") &
  "T" &
  veranstaltung.<zeit>.Value.Substring(0, 2) &
  veranstaltung.<zeit>.Value.Substring(3, 2) &
  "00")
ics_file.WriteLine("DTEND:" &
  tag_1.ToString("yyyyMMdd") &
  "T" &
  veranstaltung.<zeit>.Value.Substring(8, 2) &
  veranstaltung.<zeit>.Value.Substring(11, 2) &
  "00")
ics_file.WriteLine("DTSTAMP:" &
  Now.ToString("yyyyMMddTHHmmss"))
ics_file.WriteLine("END:VEVENT")
```

Dabei verlassen wir uns – was sicherlich nicht ganz ungefährlich ist – darauf, dass die Zeit einer Prüfung in `veranstaltungen.xml` exakt in der Form `14:00 - 17:00` eingetragen ist (Abbildung 4.3).

Den zweiten Prüfungstermin tragen wir analog ein (`tag_2`):

```
  ics_file.WriteLine("BEGIN:VEVENT")
  ics_file.WriteLine("UID:" &
    Guid.NewGuid.ToString())
  ics_file.WriteLine("SUMMARY:" &
    veranstaltung.<modul>.Value)
  ics_file.WriteLine("LOCATION:" &
    Join(raum_array, ", "))
  ics_file.WriteLine("DESCRIPTION:" &
    "Verbände: " &
    Join(verband_array, ", ") &
    "\n" &
    "Prüfende: " &
    Join(dozent_array, ", "))
  ics_file.WriteLine("DTSTART:" &
    tag_2.ToString("yyyyMMdd") &
    "T" &
    veranstaltung.<zeit>.Value.Substring(0, 2) &
    veranstaltung.<zeit>.Value.Substring(3, 2) &
    "00")
  ics_file.WriteLine("DTEND:" &
    tag_2.ToString("yyyyMMdd") &
    "T" &
    veranstaltung.<zeit>.Value.Substring(8, 2) &
    veranstaltung.<zeit>.Value.Substring(11, 2) &
    "00")
  ics_file.WriteLine("DTSTAMP:" &
    Now.ToString("yyyyMMddTHHmmss"))
  ics_file.WriteLine("END:VEVENT")
Next
```

Nachdem wir alle verplanten Prüfungen in die Kalenderdatei eingetragen haben, beenden wir den Kalendereintrag

```
ics_file.WriteLine("END:VCALENDAR")
```

und schließen die Datei:

```
ics_file.Close()
```

Wenn die Nutzerin im Query-String ein Verband, eine Dozentin oder ein Raum angefordert hat, machen wir den Hyperlink auf der Seite sichtbar

```
If Not aktueller_code = "" Then
  HyperLink_ics.Visible = True
End If
```

und tragen den passenden Link ein:

```
HyperLink_ics.Text = aktueller_code & ".ics herunterladen"
```

```
HyperLink_ics.NavigateUrl = "ics/" & aktueller_code & ".ics"
End Sub
```

5 Veranstaltungsplanung

In Abschnitt 2.7 beschreiben wir, wie die Prüferin ihre Veranstaltungen plant. In den folgenden beiden Kapiteln (`veranstaltungsplanung.aspx` und `veranstaltungsplanung.aspx.vb`) erläutern wir den dazu verwendeten Code.

5.1 `veranstaltungsplanung.aspx`

Nach dem schon in `veranstaltungen.aspx` beschriebenen ASPX-Vorgeplänkel

```
<%@ Page
  Language="VB"
  AutoEventWireup="false"
  CodeFile="veranstaltungsplanung.aspx.vb"
  Inherits="veranstaltungsplanung" %>
<!DOCTYPE html>
<html>
<head
  runat="server">
  <meta charset="utf-8" />
  <link
    href="../iplan.css"
    rel="stylesheet" />
  <title>Veranstaltungsplanung
  </title>
</head>
```

beginnen wir die Seite mit einer Fehlermeldung, die wir natürlich erst dann sichtbar machen, wenn wir vor dem Abspeichern feststellen, dass von der Dozentin ausgewählte Ressourcen nicht mehr verfügbar sind:

```
<body>
  <form
    id="form1"
    runat="server">
    <h3>
      <asp:Label
        ID="Label_wunsch_nicht_erfuellbar"
        runat="server"
        ForeColor="Red"
```

```
        Text="Ihr Wunsch kann leider nicht erfüllt werden, da
           jemand anderes schneller war."
        Visible="False">
      </asp:Label>
    </h3>
```

Eine weitere Fehlermeldung aktivieren wir, wenn die Nutzerin nicht in der Liste der von der Planerin definierten Dozentinnen auftaucht:

```
    <h3>
      <asp:Label
        ID="Label_kein_dozent_gefunden"
        runat="server"
        ForeColor="Red"
        Text="Kein Dozent gefunden"
        Visible="False">
      </asp:Label>
    </h3>
```

Auch das folgende Panel ist anfangs unsichtbar und wird erst dann sichtbar, wenn die Dozentin eine ihrer Veranstaltungen zur Bearbeitung ausgewählt hat:

```
    <asp:Panel
      ID="Panel_details"
      runat="server"
      Visible="False">
```

Nach einer dynamisch in `veranstaltungsplanung.aspx.vb` noch zu definierenden Überschrift

```
      <h3>
        <asp:Label
          ID="Label_modul_verband"
          runat="server">
        </asp:Label>
      </h3>
```

beginnen wir eine Tabelle, in der wir die Details der aktuellen Veranstaltung darstellen:

```
      <asp:Table
      ID="Table_details"
      runat="server"
      CssClass="Grid">
```

Wir schreiben die passenden Tabellenüberschriften

```
        <asp:TableHeaderRow>
          <asp:TableHeaderCell>
            Tag
          </asp:TableHeaderCell>
```

```
          <asp:TableHeaderCell>
            Zeit
          </asp:TableHeaderCell>
          <asp:TableHeaderCell>
            Raum
          </asp:TableHeaderCell>
          <asp:TableHeaderCell>
            Aulis-Link, Bemerkung
          </asp:TableHeaderCell>
        </asp:TableHeaderRow>
```

und beginnen dann die erste (und einzige) Tabellenzeile:

```
        <asp:TableRow
          runat="server">
```

In der ersten Spalte der Tabellenzeile definieren wir eine Listbox mit 5 Einträgen, aus der die Dozentin später den Wochentag der Veranstaltung auswählen kann:

```
          <asp:TableCell
            runat="server"
            VerticalAlign="Top">
            <asp:ListBox
              ID="ListBox_tag"
              runat="server"
              Rows="5"
              AutoPostBack="True">
            </asp:ListBox>
          </asp:TableCell>
```

In der zweiten Spalte wählt die Dozentin später die Uhrzeit (den Block) der Veranstaltung:

```
          <asp:TableCell
            runat="server"
            VerticalAlign="Top">
            <asp:ListBox
              ID="ListBox_zeit"
              runat="server"
              Rows="8"
              AutoPostBack="True">
            </asp:ListBox>
          </asp:TableCell>
```

Die dritte Spalte ist für die Auswahl des Raumes bzw. der Räume der Veranstaltung vorgesehen. Da die Dozentin die Veranstaltungsplanung üblicherweise auf einem Desktoprechner[1] durchführt, können wir die Dozentin mit einem Tooltipp darüber informieren,

[1] Theoretisch kann die Planung auch auf einem Tablet oder sogar einem Mobiltelefon durchgeführt werden. Das horizontale Hin- und Herscrollen bei breiten Tabellen ist dort allerdings etwas lästig.

dass sie auch mehrere Räum für ihre Veranstaltung auswählen kann:

```
<asp:TableCell
  runat="server"
  VerticalAlign="Top">
  <asp:ListBox
    ID="ListBox_raum"
    runat="server"
    Rows="23"
    SelectionMode="Multiple"
    ToolTip="Mehrfachauswahl mit der Strg-Taste"
    AutoPostBack="True">
  </asp:ListBox>
</asp:TableCell>
```

In der letzten Spalte informieren wir die Dozentin gegebenenfalls mit einer Fehlermeldung darüber, dass sie „aus Versehen“ versucht hat

```
<asp:TableCell
  runat="server"
  VerticalAlign="Top">
  <asp:Label
    ID="Label_aulis_fehler"
    runat="server"
    Text="Bitte verwenden Sie nur Aulis-Links. <br />"
    Visible="False"
    ForeColor="Red">
  </asp:Label>
```

einen Nicht-Aulis-Link in die entsprechende Textbox einzutragen:

```
  <asp:TextBox
    ID="TextBox_aulis"
    runat="server"
    Width="500"
    ToolTip="Aulis-Link (kann leer bleiben)">
  </asp:TextBox>
  <br />
```

Zusätzlich überprüfen wir auf dem Server mit einem regulären Ausdruck, ob die Dozentin „aus Versehen böse Zeichen“[2] in der Textbox verwendet:

```
  <asp:RegularExpressionValidator
    ID="RegularExpressionValidator_aulis"
    runat="server"
```

[2]Eine in einer XML-Datei mittendrin abgespeicherte Datei-Ende-Markierung (0x04) führt beispielsweise dazu, dass die Datei an dieser Stelle einfach „aufhört“ und der darauffolgende Rest der Datei dem Betriebssystem zur freien Verfügung gestellt wird und damit in den meisten Fällen unwiederbringlich verloren ist.

```
            ControlToValidate="TextBox_aulis"
            ForeColor="Red"
            ValidationExpression="[\x21-\x3B  \x3D-\x7E \n \r
               \t ä ö ü Ä Ö Ü ß € ° ’ §]*"
            Width="500"
            Display="Dynamic">
            Bitte verwenden Sie nur Zeichen einer deutschen
               Standardtastatur ohne das Kleinerzeichen.
          </asp:RegularExpressionValidator>
          <br />
```

Auch die Bemerkungstextbox

```
          <asp:TextBox
            ID="TextBox_bemerkung"
            runat="server"
            Width="500"
            ToolTip="Bemerkung (kann leer bleiben)">
          </asp:TextBox>
          <br />
```

schützen wir mit einem „Überprüfer“:

```
          <asp:RegularExpressionValidator
            ID="RegularExpressionValidator_bemerkung"
            runat="server"
            ControlToValidate="TextBox_bemerkung"
            ForeColor="Red"
            ValidationExpression="[\x21-\x3B  \x3D-\x7E \n \r
               \t ä ö ü Ä Ö Ü ß € ° ’ §]*"
            Width="500"
            Display="Dynamic">
            Bitte verwenden Sie nur Zeichen einer deutschen
               Standardtastatur ohne das Kleinerzeichen.
          </asp:RegularExpressionValidator>
        </asp:TableCell>
      </asp:TableRow>
    </asp:Table>
    <br />
```

Unter die Tabelle setzen wir jetzt noch drei Schaltflächen, um die Auswahl wieder auf den Ursprungszustand zurückzusetzen

```
    <asp:Button
      ID="Button_reset"
      runat="server"
      Text="Auswahl zurücksetzen" />
     
```

sämtliche Änderungen zu verwerfen

```
      <asp:Button
        ID="Button_abbruch"
      runat="server"
      Text="Abbruch" />
     
```

oder alle Änderungen abzuspeichern[3]:

```
    <asp:Button
      ID="Button_speichern"
      runat="server"
      Text="Änderungen speichern"
      Visible="true" />
    </asp:Panel>
    <br />
```

Während das soeben beschriebene Panel anfangs unsichtbar bleibt, ist das Gridview, in dem wir in `veranstaltungsplanung.aspx.vb` die zu verplanenden Veranstaltungen auflisten, ständig sichtbar:

```
    <asp:GridView
      ID="GridView_veranstaltungen"
      runat="server"
      CssClass="Grid"
      AutoGenerateSelectButton="True"
      SelectedRowStyle-ForeColor="#2196F3">
    </asp:GridView>
  </form>
</body>
</html>
```

Im Gridview verwenden wir das Attribut `AutoGenerateSelectButton="True"`, das freundlicherweise automatisch in jeder Veranstaltungszeile ganz links eine Auswählen-Schaltfläche einfügt (Abbildung 2.17).

5.2 Blockschaltbild

Abbildung 5.1 zeigt das Blockschaltbild der Veranstaltungsplanung. Die roten Blöcke beinhalten Unterprogramme, die vom Laufzeitsystem aufgerufen werden, wenn die Web-

[3] Bei der Speichern-Schaltfläche haben wir schon vorgesehen, dass wir – wenn wir zu einem bestimmten Zeitpunkt den Plan „einfrieren“ möchten – einfach die Sichtbarkeit der Schaltfläche auf `false` setzen können, sodass die Dozentin keine weiteren Änderungen in ihrem Plan durchführen kann. Insbesondere bei Prüfungsplänen kann dies – beispielsweise drei Wochen vor Beginn des Prüfungszeitraums – aus Planungssicherheitsgründen für die Studentinnen sinnvoll sein. Ein spätes Vorverlegen einer Prüfung könnte beispielsweise katastrophale rechtliche Konsequenzen nach sich ziehen, wenn dadurch eine Studentin nicht rechtzeitig zur Prüfung erscheint ...

seite geladen wurde, die Dozentin eine Schaltfläche drückt, ... Die roten Blöcke rufen dann die Unterprogramme in den blauen Blöcke auf, die wiederum die Hilfsfunktionen in den grünen Blöcken benutzen.

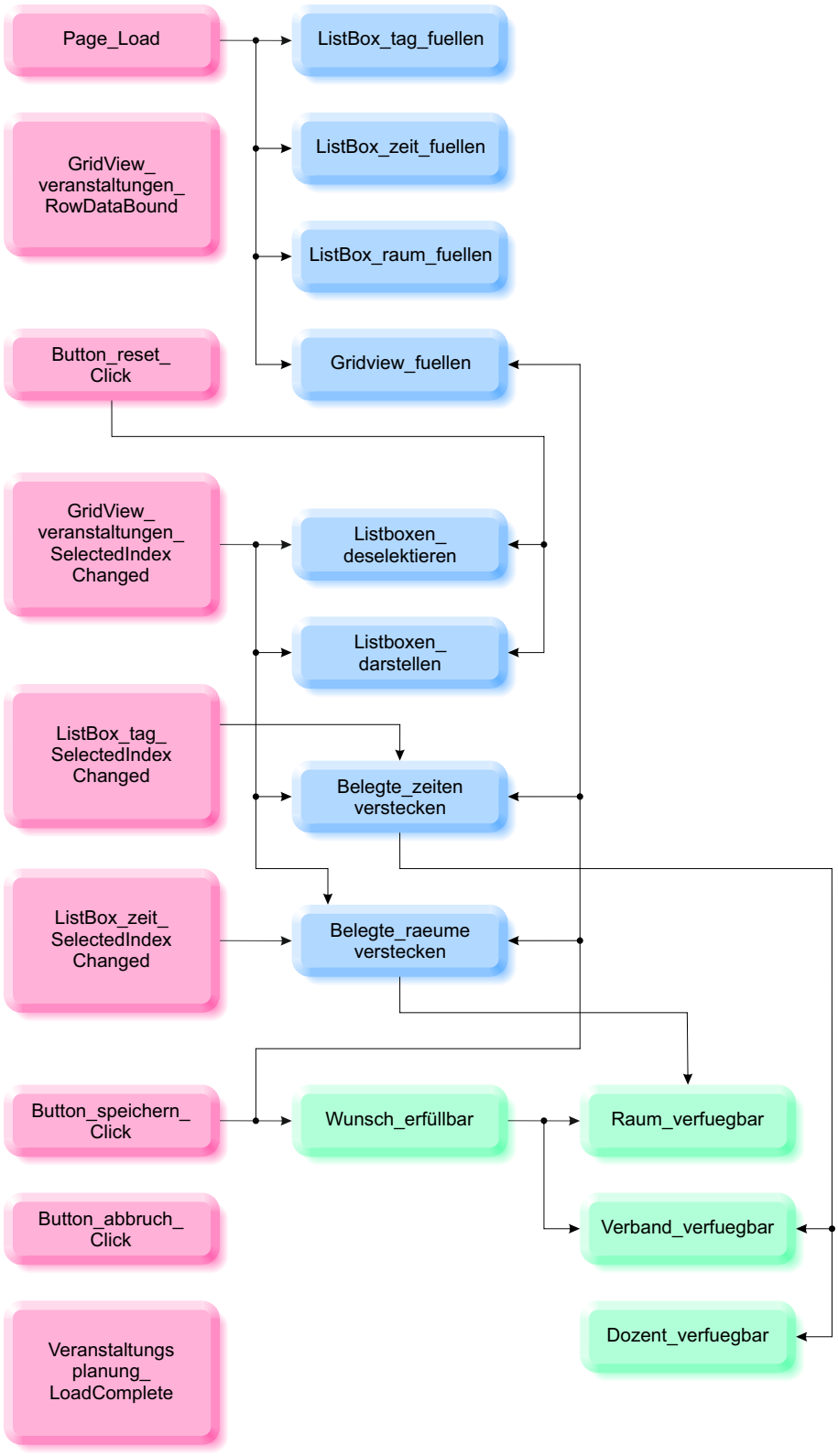


Abbildung 5.1: Blockschaltbild der Veranstaltungsplanung

Wenn die Webseite geöffnet wird, liest und verarbeitet Page_Load die Variablen des Query-Strings und ruft beim ersten Durchlauf die Unterprogramme ListBox_tag_fuellen, ListBox_zeit_fuellen und Listbox_raum_fuellen auf, die die Detail-Listboxen mit Leben füllen. Außerdem ruft Page_Load das Unterprogramm Gridview_fuellen auf, das die Liste der Veranstaltungen der Dozentin erzeugt.

Wenn die Dozentin eine Veranstaltung aus der Liste ausgewählt hat, ruft das Laufzeitsystem das Unterprogramm GridView_veranstaltung_SelectedIndexChanged auf, das wiederum die Listboxen zurücksetzt und sichtbar macht (Listboxen_deselektieren und Listboxen_darstellen). Außerdem versteckt GridView_veranstaltung_SelectedIndexChanged die schon belegten Zeiten und Räume in den Listboxen (Belegte_zeiten_verstecken und Belegte_raeume_verstecken).

Das Unterprogramm Button_speichern_Click wird aufgerufen, wenn die Dozentin ihre Plananpassungen abspeichern möchte und auf die entsprechende Schaltfläche klickt. Das Unterprogramm verwendet die Funktion Wunsch_erfuellbar, um zu überprüfen, ob alle gewünschten Ressourcen noch verfügbar sind, speichert – wenn dies der Fall ist – die Veranstaltungsänderungen und aktualisiert die Liste der Veranstaltungen durch Aufruf von Gridview_fuellen. Andernfalls gibt es eine Fehlermeldung aus und zeigt den Istzustand der belegten Zeiten und Räume an (Belegte_zeiten_verstecken und Belegte_raeume_verstecken).

Die Funktion Wunsch_erfuellbar verwendet ihrerseits die Funktionen Verband_verfuegbar und Raum_verfuegbar, um festzustellen, ob die gewünschten Ressourcen verfügbar sind.

Das durch Klick auf die entsprechende Schaltfläche aufgerufene Unterprogramm Button_reset_Click stellt den Ausgangszustand wieder her, indem es Listboxen_deselektieren und Listboxen_darstellen aufruft.

Wenn die Dozentin einen neuen Tag ausgewählt hat (ListBox_tag_SelectedIndexChanged), machen wir die Raumliste unsichtbar und zeigen in der Zeitliste nur die noch verfügbaren Zeiten an (Belegte_zeiten_verstecken).

Belegte_zeiten_verstecken selbst verwendet die Funktionen Verband_verfuegbar und Dozent_verfuegbar, um nur die Zeiten anzuzeigen, zu denen sowohl der Verband als auch die Dozentin der aktuellen Veranstaltung verfügbar ist.

Wenn die Dozentin dann im zweiten Schritt auch eine (neue) Zeit ausgewählt hat (ListBox_zeit_SelectedIndexChanged), machen wir die Raumbox wieder sichtbar und verstecken die schon belegten Räume (Belegte_raeume_verstecken).

Dabei untersucht Belegte_raeume_verstecken mit Hilfe der Funktion Raum_verfuegbar, welche Räume am gewählten Tag zur gewählten Zeit noch verfügbar sind und versteckt die nicht mehr verfügbaren Räume.

Einige Unterprogramme rufen keine weiteren Unterprogramme oder Funktionen auf:

GridView_veranstaltungen_RowDataBound dient nur dazu, die ID-Spalte in der Veranstaltungsliste unsichtbar zu machen.

`Button_abbruch_Click` schließt die Detaildarstellung und deselektiert die vorher ausgewählte Veranstaltung in der Veranstaltungsliste.

`Veranstaltungsplanung_LoadComplete` wird vom Laufzeitsystem am Ende aufgerufen, wenn die Seite komplett aufgebaut ist. Hier entscheiden wir in Abhängigkeit davon, was die Dozentin ausgewählt hat, ob wir die Speichern-Schaltfläche freigeben.

5.3 veranstaltungsplanung.aspx.vb

Genau wie bei der Veranstaltungsdarstellung (Abschnitt 3.3.1) verwenden wir auch in Abschnitt 5.1 ein paar globale Variablen zum vereinfachten Datenaustausch zwischen den Unterprogrammen:

```
Partial Class veranstaltungsplanung
Inherits System.Web.UI.Page
Public veranstaltungen_xml_datei As String
Public veranstaltungen_xml As XElement
ReadOnly DataTable_veranstaltungen As New Data.DataTable
Public Shared aktueller_dozent As IEnumerable(Of XElement)
Public aktuelles_semester As String
```

5.3.1 Page_Load

In dem bei jedem Aufruf des „Haupt“-Programms

```
Protected Sub Page_Load(sender As Object, e As EventArgs)
   Handles Me.Load
```

lesen wir aus dem Query-String das angegebene Semester

```
aktuelles_semester = Request.QueryString("semester")
```

und die E-Mail-Adresse der Dozentin, nachdem sie sich im Authentifizierungssystem der Hochschule angemeldet hat:

```
Dim dozent_mail = Request.ServerVariables("mail").Split(";")(0).
   ToLower
```

Dabei beschränken wir uns mit `.Split(";")(0)` auf den vor dem Semikolon liegenden Teil, da der Shibboleth-Identity-Provider der Hochschule alle seine Informationen aus ungeklärten Gründen in doppelter Form ausspuckt, beispielsweise:

`Joerg.Buchholz@hs-bremen.de;Joerg.Buchholz@hs-bremen.de`

Wir lesen die Dozenten-XML-Datei des aktuellen Semesters

```
Dim dozenten_xml As XElement = XElement.Load(MapPath("../xml/" &
    aktuelles_semester & "/veranstaltung/dozenten.xml"))
```

und untersuchen, ob die Planerin darin eine Dozentin mit der E-Mail-Adresse der aktuellen Nutzerin eingetragen hat:

```
aktueller_dozent =
  From dozent In dozenten_xml.<dozent>
  Where dozent.<email>.Value.ToLower = dozent_mail
```

Wenn dies der Fall ist, geht es im Programm ohne Fehlermeldung weiter; wenn nicht, beenden wir das Programm an dieser Stelle mit `Return` und einer Fehlermeldung:

```
If aktueller_dozent.Any Then
  Label_kein_dozent_gefunden.Visible = False
Else
  Label_kein_dozent_gefunden.Visible = True
  Return
End If
```

Mit

```
Page.Title =
  "Veranstaltungsplanung " &
  aktueller_dozent.<vorname>.Value &
  " " &
  aktueller_dozent.<name>.Value &
  "  "
```

schreiben wir den Namen der Dozentin in den Tab-Titel der Seite, was allerdings – wie man in Abbildung 5.2 sieht – in den meisten Browsern noch etwas optimierungswürdig[4] ist.

Abbildung 5.2: Page.Title schreibt den Tab-Titel der Seite.

Wir definieren den Pfad zur aktuellen Veranstaltungsdatei

```
veranstaltungen_xml_datei = MapPath("../xml/" &
   aktuelles_semester & "/veranstaltung/veranstaltungen.xml")
```

und lesen sie:

```
veranstaltungen_xml = XElement.Load(veranstaltungen_xml_datei)
```

[4]Durch das lange Wort „Veranstaltungsplanung“ werden nur die ersten paar Buchstaben des Namens der Dozentin angezeigt. Ja, man könnte mit dem Namen beginnen oder den Titel auf die Seite selbst schreiben. Andererseits bekommt ja immer nur die gerade angemeldete Dozentin selbst ihre eigenen zu verplanenden Veranstaltungen zu sehen, sodass weitere Informationen eigentlich überflüssig sind. Auf dem Mobiltelefon wird der Seitentitel überhaupt nicht angezeigt.

Schließlich rufen wir beim ersten Aufruf der Seite die Unterprogramm ListBox_tag_fuellen, ListBox_zeit_fuellen, Listbox_raum_fuellen und Gridview_fuellen auf, die die statischen Informationen (Wochentage, Blockzeiten, Räume und Veranstaltungen der Dozentin) in die Listboxen und das Gridview schreiben (Abbildung 5.1):

```
If Not Page.IsPostBack Then
  ListBox_tag_fuellen()
  ListBox_zeit_fuellen()
  ListBox_raum_fuellen()
  Gridview_fuellen()
End If
End Sub
```

5.3.2 ListBox_tag_fuellen

Das Unterprogramm

```
Sub ListBox_tag_fuellen()
```

wird einmalig beim Start der Seite aufgerufen, um die Wochentage in die entsprechende Listbox zu schreiben. Vor dem Befüllen entfernen wir alle vorherigen[5] Einträge aus der Listbox:

```
ListBox_tag.Items.Clear()
```

Wir lesen die in Abbildung 5.3 dargestellte XML-Datei der zu verplanenden Wochentage[6] aus dem aktuellen Veranstaltungsordner

```
Dim tage_xml =
  XElement.Load(
  MapPath(
  "../xml/" &
  aktuelles_semester &
  "/veranstaltung/tage.xml"))
```

[5] Um ganz ehrlich zu sein, ist sich der Autor nicht mehr wirklich sicher, warum wir hier „alte“ Einträge löschen müssen, obwohl das Unterprogramm doch nur einmalig beim ersten Aufruf der Seite durchlaufen wird. Aber es gab bestimmt einen besonderen Fall, ...

[6] Eigentlich könnten wir die Wochentage (Montag bis Freitag) auch einfach hart verdrahten. In den letzten 25 Jahren hat es jedenfalls keine offiziellen Wochenendveranstaltungen im Maschinenbau gegeben ...

```
<root>
  <tage>
    <tag>Montag</tag>
    <tag>Dienstag</tag>
    <tag>Mittwoch</tag>
    <tag>Donnerstag</tag>
    <tag>Freitag</tag>
  </tage>
  <reserven>
    <reserve>Samstag</reserve>
    <reserve>Sonntag</reserve>
  </reserven>
</root>
```

Abbildung 5.3: `tage.xml` im aktuellen Veranstaltungsordner

und extrahieren daraus eine Liste aller[7] Wochentage:

```
Dim alle_tage =
  From tag In tage_xml.<tage>.<tag>
  Select tag.Value
```

Für jeden Tag erzeugen wir einen neuen (potenziellen) Listeneintrag

```
For Each tag In alle_tage
  Dim neuer_tag As New ListItem With {
    .Value = tag,
    .Text = tag
  }
```

und fügen ihn ans Ende der Listbox an:

```
  ListBox_tag.Items.Add(neuer_tag)
Next
End Sub
```

Das Ergebnis können wir unter Abbildung 2.18 bewundern.

5.3.3 `ListBox_zeit_fuellen`

Das Unterprogramm zum Befüllen der Blockzeiten sieht sehr ähnlich wie `ListBox_tag_fuellen` aus:

```
Sub ListBox_zeit_fuellen()
ListBox_zeit.Items.Clear()
Dim zeiten_xml As XElement = XElement.Load(MapPath("../xml/" &
   aktuelles_semester & "/veranstaltung/zeiten.xml"))
```

[7] Wir verlassen uns dabei darauf, dass die Wochentage schon in der richtigen Reihenfolge in der XML-Datei stehen und von linq2xml auch genau in dieser Reihenfolge gelesen werden.

Wie wir in Abbildung 5.4 sehen, haben wir aber – etwas inkonsequenterweise – in der XML-Datei der Blockzeiten etwas mehr prophylaktischen Aufwand getrieben.

```
<zeiten>
  <zeit>
    <id>41fe51b9-0b84-4204-9f38-278c57b825e5</id>
    <abk>1</abk>
    <beginn>08:00</beginn>
    <ende>09:30</ende>
    <bemerkung></bemerkung>
  </zeit>
  <zeit>
    <id>42c6c839-6c35-4166-ba04-adb5bb7f9f76</id>
    <abk>2</abk>
    <beginn>09:45</beginn>
    <ende>11:15</ende>
    <bemerkung></bemerkung>
  </zeit>
:
:
```

Abbildung 5.4: `zeiten.xml` im aktuellen Veranstaltungsordner

So haben wir jedem Blockeintrag eine eindeutige ID spendiert – die wir aber im Programm bislang noch nicht nutzen – und sortieren nach der unter `<abk>` eingeführten Blocknummer, statt uns auf die Reihenfolge der Einträge in der XML-Datei zu verlassen:

```
Dim alle_zeiten =
From zeit In zeiten_xml.<zeit>
Order By zeit.<abk>.Value
```

Wir können dann abschließend für jeden Block einen Listeneintrag mit Anfangs- und Endzeit und der passenden Blocknummer erzeugen und an die Liste anhängen (Abbildung 2.18):

```
For Each zeit In alle_zeiten
  Dim neue_zeit As New ListItem With {
    .Value =
    zeit.<beginn>.Value &
    " - " &
    zeit.<ende>.Value,
    .Text =
    zeit.<abk>.Value &
    ". Block: " &
    zeit.<beginn>.Value &
    " - " &
    zeit.<ende>.Value
  }
  ListBox_zeit.Items.Add(neue_zeit)
```

```
Next
End Sub
```

5.3.4 Listbox_raum_fuellen

Im Unterprogramm zum Eintragen der Räume in ihre Listbox lesen wir die entsprechende XML-Datei (Abbildung 5.5)

```
<raeume>
  <raum>
    <id>5a3ec66d-e746-4cf1-b973-56618ebe3264</id>
    <abk>FL 403</abk>
    <name>Rechnerraum</name>
    <anzahl>8</anzahl>
    <ort>ZIMT</ort>
    <bemerkung></bemerkung>
  </raum>
  <raum>
    <id>bffc78ab-4eaf-4ba4-ac14-187b9d25997a</id>
    <abk>FL 404</abk>
    <name>Seminarraum</name>
    <anzahl>12</anzahl>
    <ort>ZIMT</ort>
    <bemerkung></bemerkung>
  </raum>
:
:
```

Abbildung 5.5: raeume.xml im aktuellen Veranstaltungsordner

und extrahieren alle Räume sortiert nach ihrer Abkürzung:

```
Sub ListBox_raum_fuellen()
ListBox_raum.Items.Clear()
Dim raeume_xml As XElement = XElement.Load(MapPath("../xml/" &
   aktuelles_semester & "/veranstaltung/raeume.xml"))
Dim alle_raeume =
From raum In raeume_xml.<raum>
Order By raum.<abk>.Value
```

In der Liste stellen wir jeden Raum mit seinem Namen, seiner Abkürzung und der Anzahl seiner Plätze dar:

```
For Each raum In alle_raeume
  If raum.<anzahl>.Value > 0 Then
    Dim neuer_raum As New ListItem With {
      .Text =
      raum.<abk>.Value &
      " (" &
```

```
      raum.<name>.Value &
      ", " &
      raum.<ort>.Value &
      ", " & raum.<anzahl>.Value &
      " Plätze)",
      .Value = raum.<abk>.Value
    }
    ListBox_raum.Items.Add(neuer_raum)
  End If
Next
End Sub
```

Da wir Räume mit einer Platzanzahl von 0 nicht darstellen, kann die Planerin dies nutzen, um aktuell in diesem Semester nicht verfügbare Räume nicht anzuzeigen. Außerdem kann sie für einen Raum, der nur für eine bestimmte Veranstaltung genutzt werden soll, eine Platzanzahl von 1 eintragen, um den Dozentinnen zu signalisieren, diesen Raum nicht für eigene Veranstaltungen auszuwählen (Abbildung 2.18).

5.3.5 Gridview_fuellen

Das Unterprogramm

```
Sub Gridview_fuellen()
```

liest alle Veranstaltungen der aktuellen Dozentin und macht sie in einer detaillierten Liste auswählbar.

Dazu lesen und extrahieren wir nochmals[8] die Wochentage

```
Dim tage_xml = XElement.Load(MapPath(
  "../xml/" &
  aktuelles_semester &
  "/veranstaltung/tage.xml"))
Dim alle_tage =
  From tag In tage_xml.<tage>.<tag>
  Select tag.Value
```

Wir extrahieren alle Veranstaltungen der aktuellen Dozentin[9] aus der aktuellen Veran-

[8] Wir müssen die Liste der Tage hier nochmals erstellen, damit wir in der Liste der Veranstaltungen mit `IndexOf` vernünftig (nicht alphabetisch) danach sortieren können: Montag, Dienstag, ... Leider können wir die Liste nicht über eine Session Variable – die nur einfache Datentypen erlaubt – transportieren. Über eine globale Variable geht es auch nicht, da `Gridview_fuellen` mehrmals, `ListBox_tag_fuellen` aber nur einmal aufgerufen wird und die darin definierten Variablen daher beim Postback nicht mehr definiert sind.

[9] Wenn es mehrere Dozentinnen in der Veranstaltung gibt, wird die Veranstaltung nur für die erste Dozentin angezeigt und kann daher auch nur von ihr geplant werden. Ob Nutzerinnen diese Eindeutigkeit der Bearbeiterin nun als „bug“ oder als „feature" empfinden, muss sich erst noch herausstellen ...

staltungsdatei (Abbildung 3.2) und sortieren sie nach Modulabkürzung, Verbandsabkürzung, Wochentag und Uhrzeit:

```
Dim alle_veranstaltungen =
From veranstaltung In veranstaltungen_xml.<veranstaltung>
Where veranstaltung.<dozent>.Value = aktueller_dozent.<abk>.
   Value
Order By
veranstaltung.<modul>.Value,
veranstaltung.<verband>.Value,
alle_tage.ToList.IndexOf(veranstaltung.<tag>.Value),
veranstaltung.<zeit>.Value
```

Alsdann verwenden wir das als globale Variable deklarierte Datatable (Abschnitt 5.3) und definieren seine[10] Spalten(überschriften):

```
DataTable_veranstaltungen.Columns.Add("ID")
DataTable_veranstaltungen.Columns.Add("Modul")
DataTable_veranstaltungen.Columns.Add("Verband")
DataTable_veranstaltungen.Columns.Add("Tag")
DataTable_veranstaltungen.Columns.Add("Zeit")
DataTable_veranstaltungen.Columns.Add("Raum")
DataTable_veranstaltungen.Columns.Add("Aulis")
DataTable_veranstaltungen.Columns.Add("Bemerkung")
```

In der Schleife über alle Veranstaltungen der aktuellen Dozentin

```
For Each veranstaltung In alle_veranstaltungen
```

lesen wir alle an der aktuellen Veranstaltung beteiligten Semesterverbände

```
  Dim verband_array As String() =
    (From verband In veranstaltung.<verband>
    Select verband.Value).ToArray
```

und Räume jeweils in ein String-Array:

```
  Dim raum_array As String() =
    (From raum In veranstaltung.<raum>
    Select raum.Value).ToArray
```

Wir erzeugen eine neue Zeile für das Datatable

```
  Dim DataRow_veranstaltung As Data.DataRow =
     DataTable_veranstaltungen.NewRow()
```

und füllen ihre Spalten mit den Daten der aktuellen Veranstaltung:

[10] Die ID-Spalte machen wir in `GridView_veranstaltungen_RowDataBound` später unsichtbar. Auf diese Weise stört die Spalte nicht; wir können aber trotzdem auf die ID zugreifen.

```
  DataRow_veranstaltung("ID") = veranstaltung.<id>.Value
  DataRow_veranstaltung("Modul") = veranstaltung.<modul>.Value
  DataRow_veranstaltung("Verband") = Join(verband_array, ", ")
  DataRow_veranstaltung("Tag") = veranstaltung.<tag>.Value
  DataRow_veranstaltung("Zeit") = veranstaltung.<zeit>.Value
  DataRow_veranstaltung("Raum") = Join(raum_array, ", ")
  DataRow_veranstaltung("Aulis") = veranstaltung.<aulis>.Value
  DataRow_veranstaltung("Bemerkung") = veranstaltung.<bemerkung>
     .Value
```

Nachdem wir die Zeile an das Datatable angehängt haben

```
  DataTable_veranstaltungen.Rows.Add(DataRow_veranstaltung)
Next
```

verwenden wir das Datatable als Datenquelle des Gridviews und stellen dieses dar:

```
GridView_veranstaltungen.DataSource = DataTable_veranstaltungen
GridView_veranstaltungen.DataBind()
End Sub
```

5.3.6 GridView_veranstaltung_SelectedIndexChanged

Das Unterprogramm

```
Private Sub GridView_veranstaltungen_SelectedIndexChanged(sender
    As Object, e As EventArgs) Handles GridView_veranstaltungen.
   SelectedIndexChanged
```

wird vom Laufzeitsystem aufgerufen, wenn die Dozentin eine Veranstaltung in der Veranstaltungsliste auswählt. Da der Dozentin dann nur die noch freien Ressourcen (Tage, Zeiten, Räume) angeboten werden, gehen wir üblicherweise davon aus, dass ihre Auswahlwünsche erfüllbar[11] sind:

```
Label_wunsch_nicht_erfuellbar.Visible = False
```

In der ersten[12] Spalte der Veranstaltungsliste hatten wir in `Gridview_fuellen` die ID der Veranstaltung (unsichtbar) eingetragen. Diesen ID-Eintrag (`SelectedRow.Cells(1)`) verwenden wir jetzt, um auf die gerade ausgewählte Veranstaltung zuzugreifen:

```
Dim aktuelle_veranstaltung =
From veranstaltung In veranstaltungen_xml.<veranstaltung>
Where veranstaltung.<id>.Value = GridView_veranstaltungen.
   SelectedRow.Cells(1).Text
```

[11] Vor dem Abspeichern der Auswahl überprüfen wir in `Button_speichern_Click` nochmals, ob die gewünschten Ressourcen noch immer verfügbar sind; andernfalls machen wir `Label_wunsch_nicht_erfuellbar` sichtbar.

[12] Die nullte Spalte beinhaltet die Auswählen-Schaltfläche.

Mit den nächsten Befehlen werden wir die eventuell schon vorhandenen Daten der aktuellen Veranstaltung in die Listboxen eintragen. Dazu deselektieren wir alle Einträge in allen Listboxen durch Aufruf von `Listboxen_deselektieren`

```
Listboxen_deselektieren()
```

deklarieren ein allgemeines Listen-Element

```
Dim gefunden As ListItem
```

und verwenden es, um den möglicherweise schon gesetzten Wochentag der aktuellen Veranstaltung in der Listbox der Wochentage zu finden:

```
gefunden = ListBox_tag.Items.FindByValue(aktuelle_veranstaltung.
   <tag>.Value)
```

Wenn wir den Wochentag in der Listbox gefunden haben, selektieren wir ihn dort:

```
If gefunden IsNot Nothing Then
  gefunden.Selected = True
End If
```

Das gleiche Spiel wiederholen wir für die Listbox der Blockzeiten:

```
gefunden = ListBox_zeit.Items.FindByValue(aktuelle_veranstaltung
   .<zeit>.Value)
If gefunden IsNot Nothing Then
  gefunden.Selected = True
End If
```

Während es für eine Veranstaltung sinnvollerweise nur einen Wochentag und eine Blockzeit geben kann und der Verband und die Dozentin ja schon von der Planerin festgelegt wurde, kann die Dozentin mehrere Räume für eine Veranstaltung buchen; beispielsweise, wenn sie abwechselnd einen Seminarraum und einen Rechnerraum nutzen möchte. Wir müssen daher die Raumselektion in einer Schleife über alle Räume der aktuellen Veranstaltung durchlaufen:

```
For Each raum In aktuelle_veranstaltung.<raum>
  gefunden = ListBox_raum.Items.FindByValue(raum)
  If gefunden IsNot Nothing Then
    gefunden.Selected = True
  End If
Next
```

Jetzt können wir durch Aufruf von `Listboxen_darstellen` die Listboxen mit selektierten Einträgen (und die jeweils „nächste“ Listbox) darstellen

```
Listboxen_darstellen()
```

und die schon von anderen Dozentinnen belegten Blockzeiten und Räume verstecken (`Belegte_zeiten_verstecken`, `Belegte_raeume_verstecken`), sodass die Dozentin nur aus noch verfügbaren Einträgen auswählen kann:

```
Belegte_zeiten_verstecken()
Belegte_raeume_verstecken()
```

Wir tragen einen eventuell schon vorhandenen Aulis-Link und eine eventuell schon vorhandene Bemerkung in die entsprechenden Textboxen ein

```
TextBox_aulis.Text = aktuelle_veranstaltung.<aulis>.Value
TextBox_bemerkung.Text = aktuelle_veranstaltung.<bemerkung>.
   Value
```

und schreiben den Modulnamen, die Verbände, die Dozentin und das Semester der Veranstaltung in die Überschrift der Detailansicht:

```
Label_modul_verband.Text =
  GridView_veranstaltungen.SelectedRow.Cells(2).Text &
  ", " &
  GridView_veranstaltungen.SelectedRow.Cells(3).Text &
  ", " &
  aktueller_dozent.<abk>.Value &
  ", " &
  aktuelles_semester.ToUpper
```

Dabei können wir den Modulnamen (`Cells(2)`) und die Verbände (`Cells(3)`) direkt aus der Veranstaltungsliste entnehmen, was den Vorteil hat, dass die Verbände gleich als komma-separierte Liste erscheinen. Die Dozentin und das Semester treten in der Veranstaltungsliste nicht auf, finden sich aber in den entsprechenden (globalen) Variablen.

Abschließend machen wir das fertig gefüllte Detail-Panel sichtbar:

```
Panel_details.Visible = True
End Sub
```

5.3.7 `Listboxen_deselektieren`

Das Unterprogramm

```
Sub Listboxen_deselektieren()
```

wird von `GridView_veranstaltung_SelectedIndexChanged` aufgerufen, bevor es selbst die schon ausgewählten Elemente in den Listboxen selektiert:

```
ListBox_tag.ClearSelection()
ListBox_zeit.ClearSelection()
ListBox_raum.ClearSelection()
End Sub
```

5.3.8 Listboxen_darstellen

Im Unterprogramm

```
Sub Listboxen_darstellen()
```

entscheiden wir, welche Listboxen wir darstellen wollen. Da die Dozentin üblicherweise zuerst den Tag, dann die Zeit und dann den Raum einer Veranstaltung auswählt, stellen wir die Listbox der Tage immer dar und die Listbox der Zeiten nur dann, wenn die Dozentin schon einen Tag ausgewählt hat:

```
If ListBox_tag.GetSelectedIndices.Count > 0 Then
  ListBox_zeit.Visible = True
Else
  ListBox_zeit.Visible = False
End If
```

Die gleiche Argumentation gilt für die Raum-Listbox, die wir nur dann darstellen, wenn die Dozentin schon eine Zeit ausgewählt hat:

```
If ListBox_zeit.GetSelectedIndices.Count > 0 Then
  ListBox_raum.Visible = True
Else
  ListBox_raum.Visible = False
End If
End Sub
```

Bei beiden Bedingungen brauchen wir auch den `Else`-Teil, da Auswahlen – beispielsweise durch die Auswahl zurücksetzen-Schaltfläche – auch wieder zurückgenommen werden können.

5.3.9 Button_speichern_Click

Wenn die Dozentin die Änderungen speichern-Schaltfläche anklickt, ruft das Laufzeitsystem das Unterprogramm

```
Protected Sub Button_speichern_Click(sender As Object, e As
   EventArgs) Handles Button_speichern.Click
```

auf. Um zumindest zu verhindern, dass die Dozentin „aus Versehen“ einen malignen Link in das Aulis-Feld einträgt, untersuchen wir als erstes, ob der eingetragene Link mit `https://aulis.hs-bremen.de` beginnt oder komplett leer ist[13]:

[13] Dass diese „Sicherheitsmaßnahme“ bei weitem nicht wasserdicht ist, sollte jederfrau mit auch nur minimaler krimineller Energie schnell klar werden; andererseits hat sich die Dozentin ja gerade mit ihrem Hochschulkonto angemeldet und kann für etwaige Konsequenzen ihrer Einträge jederzeit leicht persönlich verantwortlich gemacht werden ...

```
If TextBox_aulis.Text.StartsWith("https://aulis.hs-bremen.de")
   Or
  TextBox_aulis.Text = "" Then
  Label_aulis_fehler.Visible = False
```

Wenn dies nicht der Fall ist, informieren wir die Dozentin darüber, setzen das Aulis-Feld zurück und brechen den Speichervorgang ab:

```
Else
  Label_aulis_fehler.Visible = True
  TextBox_aulis.Text = "https://aulis.hs-bremen.de"
  Return
End If
```

Das Ändern eines Elements in einer XML-Datei ist mit linq2xml sehr komfortabel: Wir extrahieren dazu das Element aus der Datei, modifizieren es und speichern es wieder in die Datei. Wir lesen also die Datei aller Veranstaltungen

```
veranstaltungen_xml = XElement.Load(veranstaltungen_xml_datei)
```

und extrahieren die aktuelle Veranstaltung über ihre ID:

```
Dim aktuelle_veranstaltung =
From veranstaltung In veranstaltungen_xml.<veranstaltung>
Where veranstaltung.<id>.Value = GridView_veranstaltungen.
   SelectedRow.Cells(1).Text
```

Damit wir – quasi zum Löschen – auch eine Veranstaltung ohne Tag und ohne Zeit abspeichern können, deklarieren wir diesen Fall als Default:

```
Dim tag As String = ""
Dim zeit As String = ""
```

Wenn die Dozentin aber einen Tag

```
If ListBox_tag.SelectedIndex >= 0 Then
  tag = ListBox_tag.SelectedItem.Value
End If
```

und/oder eine Zeit ausgewählt hat, verwenden wir natürlich ihre Auswahl:

```
If ListBox_zeit.SelectedIndex >= 0 Then
  zeit = ListBox_zeit.SelectedItem.Value
End If
```

Wir können jetzt die Auswahl (Wochentag, Blockzeit, Aulis-Link und Bemerkung) in die aktuelle Veranstaltung eintragen und dabei möglicherweise schon vorhandene Einträge automatisch überschreiben:

```
aktuelle_veranstaltung.<tag>.Value = tag
aktuelle_veranstaltung.<zeit>.Value = zeit
aktuelle_veranstaltung.<aulis>.Value = TextBox_aulis.Text
aktuelle_veranstaltung.<bemerkung>.Value = TextBox_bemerkung.
   Text
```

Das Eintragen der ausgewählten Räume ist etwas umständlicher, da in der XML-Datei einer Veranstaltung auch mehrere Räume zugeordnet sein können (Abbildung 5.6).

```
<veranstaltung>
  :
  :
  <raum>M 111</raum>
  <raum>M 116</raum>
  <raum>M 118</raum>
</veranstaltung>
```

Abbildung 5.6: Mehrere Räume in einer Veranstaltung

Wir entfernen daher alle Räume aus der aktuellen Veranstaltung

```
aktuelle_veranstaltung.<raum>.Remove
```

und fügen die ausgewählten Räume wieder hinzu. Dazu deklarieren wir einen neuen Raum, den wir im Folgenden mehrfach verwenden

```
Dim neuer_raum As XElement
```

fragen noch ab, ob wir überhaupt neue Räume eintragen müssen

```
If ListBox_raum.SelectedIndex >= 0 Then
```

und starten dann eine Schleife über alle ausgewählten Räume:

```
  For Each index In ListBox_raum.GetSelectedIndices()
```

In der Schleife erzeugen wir einen neuen XML-Eintrag[14] für jeden Raum

```
    neuer_raum = <raum><%= ListBox_raum.Items(index).Value %></
       raum>
```

und hängen ihn an die aktuelle Veranstaltung[15] an:

[14] Es ist schon sehr hilfreich, dass wir in Visual Basic einer Variable direkt XML-Literale zuweisen (`neuer_raum = <raum> ...`) und darin mit `<%= ... %>` auf Variablenwerte des Programms zugreifen können.

[15] Da die `aktuelle_veranstaltung` formal ein `IEnumerable(Of XElement)` ist, da sie theoretisch ja auch mehrere Veranstaltungen enthalten könnte (wenn die ID in der XML-Datei mehrfach vorkommen würde, was aber in unserem Fall natürlich nicht passieren sollte) und einem `IEnumerable` mit `Add` kein `XElement` hinzugefügt werden kann, müssen wir mit `aktuelle_veranstaltung.First` auf die erste (und einzige) Veranstaltung zugreifen, die wiederum ein `XElement` ist, dem wir ein weiteres `XElement`, nämlich den neuen Raum, hinzufügen können. Kompliziert, aber so geht's ...

```
    aktuelle_veranstaltung.First.Add(neuer_raum)
  Next
End If
```

Bevor wir die Veranstaltung jetzt abspeichern, untersuchen wir durch Aufruf der Funktion `Wunsch_erfuellbar`, ob tatsächlich alle gewünschten Ressourcen noch verfügbar sind. Dabei müssen wir außerdem den Fall abdecken, dass die Dozentin den Wochentag und die Blockzeit zurückgesetzt und damit nicht ausgewählt hat, dass sie also vorhandene Einträge in der Datenbank „löschen“ möchte:

```
If Wunsch_erfuellbar(aktuelle_veranstaltung.First) Or
  tag = "" And
  zeit = "" Then
```

Wenn dies der Fall ist, speichern wir die (gegebenenfalls veränderte) Veranstaltung ohne Fehlermeldung ab:

```
  veranstaltungen_xml.Save(veranstaltungen_xml_datei)
  Label_wunsch_nicht_erfuellbar.Visible = False
```

Wenn allerdings in der Zwischenzeit eine andere Dozentin gewünschte Ressourcen schneller beansprucht hat, geben wir die entsprechende Fehlermeldung aus

```
Else
  Label_wunsch_nicht_erfuellbar.Visible = True
```

und spiegeln die nicht mehr auswählbaren Zeiten und Räume in den Listen wider und beenden das Unterprogramm ohne abzuspeichern:

```
  Belegte_zeiten_verstecken()
  Belegte_raeume_verstecken()
  Return
End If
```

Wenn alles geklappt hat, übertragen wir die neuen Informationen in die Veranstaltungsliste

```
Gridview_fuellen()
```

deselektieren die vorher ausgewählte Veranstaltung

```
GridView_veranstaltungen.SelectedIndex = -1
```

und verstecken das Detaildarstellung der vorherigen Veranstaltung:

```
Panel_details.Visible = False
End Sub
```

5.3.10 Wunsch_erfuellbar

In der Funktion

```
Function Wunsch_erfuellbar(wunsch As XElement) As Boolean
```

untersuchen wir, ob – während die Dozentin noch ihre Auswahl trifft – zufällig eine andere Dozentin die gleichen Ressourcen beansprucht und schon reserviert hat, indem sie schneller („reservation race“) ihre eigene Änderungen speichern-Schaltfläche angeklickt hat (Abschnitt 2.7).

Dazu übergeben wir der Funktion die Veranstaltung mit dem Wochentag, der Blockzeit und den Räumen, die sich die Dozentin gerade ausgesucht hat. In der Funktion untersuchen wir dann, ob alle Verbände und alle Räume zum Wunschtermin noch frei sind. Als erstes starten wir eine Schleife über alle zur Veranstaltung gehörenden Semesterverbände:

```
For Each verband In wunsch.<verband>
```

In der Schleife rufen wir für jeden Verband die Funktion `Verband_verfuegbar` mit der Verbandsabkürzung, dem Wochentag und der Blockzeit auf, um dort zu überprüfen, ob dieser Verband zum Wunschtermin noch frei ist. Wenn dies nicht der Fall ist, geben wir zurück, dass der Wunsch nicht erfüllbar ist:

```
  If Not Verband_verfuegbar(
    verband.Value,
    wunsch.<tag>.Value,
    wunsch.<zeit>.Value) Then
    Return False
  End If
Next
```

Analog untersuchen wir mit Hilfe von `Raum_verfuegbar`, ob alle Räume zum Wunschtermin noch verfügbar sind.

Wenn keine der If-Bedingungen greift, signalisieren wir mit `Return True`, dass die Wunschveranstaltung im aufrufenden Unterprogramm `Button_speichern_Click` abgespeichert werden kann:

```
For Each raum In wunsch.<raum>
  If Not Raum_verfuegbar(
    raum.Value,
    wunsch.<tag>.Value,
    wunsch.<zeit>.Value) Then
    Return False
  End If
Next
Return True
End Function
```

5.3.11 GridView_veranstaltungen_RowDataBound

Wir verwenden das vom Laufzeitsystem[16] aufgerufene Unterprogramm

```
Private Sub GridView_veranstaltungen_RowDataBound(sender As
   Object, e As GridViewRowEventArgs) Handles
   GridView_veranstaltungen.RowDataBound
```

um die erste Spalte – die die ID der Veranstaltung enthält – im Gridview auszublenden.

```
e.Row.Cells(1).Visible = False
End Sub
```

Obwohl die ID nun unsichtbar ist, können wir trotzdem beispielsweise in `GridView_veranstaltung_SelectedIndexChanged` auf die ID zugreifen, um die gewünschte Veranstaltung im Detail darzustellen.

5.3.12 Button_reset_Click

Wenn die Dozentin die Auswahl zurücksetzen-Schaltfläche anklickt, ruft das Laufzeitsystem das Unterprogramm

```
Private Sub Button_reset_Click(sender As Object, e As EventArgs)
    Handles Button_reset.Click
```

auf, in dem wir alle Auswahlen der Dozentin rückgängig machen

```
Listboxen_deselektieren()
```

die korrekten Listboxen darstellen (in diesem Fall nur die Listbox der Wochentage)

```
Listboxen_darstellen()
```

und der Dozentin eine weitere Chance geben, einen korrekten Aulis-Link einzutragen, indem wir die Fehlermeldung verstecken:

```
Label_aulis_fehler.Visible = False
End Sub
```

5.3.13 ListBox_tag_SelectedIndexChanged

Das Laufzeitsystem ruft das Unterprogramm

```
Private Sub ListBox_tag_SelectedIndexChanged(sender As Object, e
    As EventArgs) Handles ListBox_tag.SelectedIndexChanged
```

[16] `GridView_veranstaltungen_RowDataBound` wird erst aufgerufen, wenn alle Daten bereits an ihre Zellen gebunden sind. In `Page_Load` ist dies noch nicht der Fall.

auf, wenn die Dozentin einen neuen Wochentag ausgewählt hat. Wir blenden dann die Liste der Blockzeiten ein

```
ListBox_zeit.Visible = True
```

und die der Räume aus, damit die Dozentin erst eine Blockzeit auswählt:

```
ListBox_raum.Visible = False
```

Abschließend deselektieren wir alle eventuell vorher schon ausgewählten Blockzeiten

```
ListBox_zeit.ClearSelection()
```

und blenden die an dem Tag schon von anderen Dozentinnen belegten Blockzeiten durch Aufruf von `Belegte_zeiten_verstecken` aus:

```
Belegte_zeiten_verstecken()
End Sub
```

5.3.14 Belegte_zeiten_verstecken

Das Unterprogramm

```
Sub Belegte_zeiten_verstecken()
```

wird von mehreren anderen Unterprogrammen aufgerufen (Abbildung 5.1), um die Blockzeiten auszublenden, zu denen ein Verband oder ein Dozent nicht verfügbar ist.

Im ersten Schritt blenden wir dazu alle Zeiten in einer Schleife über alle Einträge der Blockzeiten-Listbox ein:

```
For Each item As ListItem In ListBox_zeit.Items
  item.Enabled = True
Next
```

Wir beschaffen uns die aktuelle Veranstaltung, indem wir auf die „unsichtbare“ ID in der ausgewählten Gridview-Zeile zugreifen:

```
Dim aktuelle_veranstaltung =
From veranstaltung In veranstaltungen_xml.<veranstaltung>
Where veranstaltung.<id>.Value = GridView_veranstaltungen.
   SelectedRow.Cells(1).Text
```

Im nächsten Schritt blenden wir alle Blockzeiten aus, zu denen einer der Verbände der aktuellen Veranstaltung nicht verfügbar ist. Dazu beginnen wir eine Schleife über alle Verbände der aktuellen Veranstaltung:

```
For Each verband In aktuelle_veranstaltung.<verband>
```

Innerhalb der Verbandsschleife starten wir eine Schleife über alle Einträge der Blockzeiten-Listbox:

```
  For zeit_index = 0 To ListBox_zeit.Items.Count - 1
```

Innerhalb der Zeitschleife können wir jetzt der Funktion `Verband_verfuegbar` den aktuellen Verband, den gewählten Wochentag und die fragliche Blockzeit übergeben, um dort überprüfen zu lassen, ob dieses Tripel noch verfügbar ist:

```
    If Not Verband_verfuegbar(
      verband.Value,
      ListBox_tag.SelectedValue,
      ListBox_zeit.Items(zeit_index).Value) Then
```

Wenn dies nicht der Fall ist, blenden wir die nicht mehr verfügbare Blockzeit in ihrer Liste aus:

```
      ListBox_zeit.Items(zeit_index).Enabled = False
    End If
  Next
Next
```

Auf die gleiche Weise blenden wir alle Blockzeiten aus, zu denen eine der Dozentinnen der aktuellen Veranstaltung nicht verfügbar ist:

```
For Each dozent In aktuelle_veranstaltung.<dozent>
  For zeit_index = 0 To ListBox_zeit.Items.Count - 1
    If Not Dozent_verfuegbar(
      dozent.Value,
      ListBox_tag.SelectedValue,
      ListBox_zeit.Items(zeit_index).Value) Then
      ListBox_zeit.Items(zeit_index).Enabled = False
    End If
  Next
Next
End Sub
```

5.3.15 ListBox_zeit_SelectedIndexChanged

Das Laufzeitsystem ruft das Unterprogramm

```
Private Sub ListBox_zeit_SelectedIndexChanged(sender As Object,
   e As EventArgs) Handles ListBox_zeit.SelectedIndexChanged
```

auf, wenn die Dozentin eine neue Blockzeit ausgewählt hat. Wir blenden dann die Liste der Räume ein

```
ListBox_raum.Visible = True
```

deselektieren alle eventuell vorher schon ausgewählten Räume

```
ListBox_raum.ClearSelection()
```

und blenden die an dem gewünschten Termin schon von anderen Dozentinnen belegten Räume durch Aufruf von `Belegte_raeume_verstecken` aus:

```
Belegte_raeume_verstecken()
End Sub
```

5.3.16 Belegte_raeume_verstecken

Das Unterprogramm

```
Sub Belegte_raeume_verstecken()
```

wird von mehreren anderen Unterprogrammen aufgerufen (Abbildung 5.1), um die Räume auszublenden, die am gewünschten Wochentag zur gewünschten Blockzeit nicht verfügbar ist.

Im ersten Schritt blenden wir dazu alle Räume in einer Schleife über alle Einträge der Raum-Listbox ein:

```
For Each item As ListItem In ListBox_raum.Items
  item.Enabled = True
Next
```

Wir beschaffen uns die aktuelle Veranstaltung, indem wir auf die „unsichtbare“ ID in der ausgewählten Gridview-Zeile zugreifen:

```
Dim aktuelle_veranstaltung =
From veranstaltung In veranstaltungen_xml.<veranstaltung>
Where veranstaltung.<id>.Value = GridView_veranstaltungen.
   SelectedRow.Cells(1).Text
```

Im nächsten Schritt beginnen wir eine Schleife über alle Einträge der Raum-Listbox:

```
For raum_index = 0 To ListBox_raum.Items.Count - 1
```

Innerhalb der Schleife übergeben der Funktion `Raum_verfuegbar` den fraglichen Raum, den gewählten Wochentag und die gewählte Blockzeit, um dort überprüfen zu lassen, ob dieses Tripel noch verfügbar ist:

```
  If Not Raum_verfuegbar(
    ListBox_raum.Items(raum_index).Value,
    ListBox_tag.SelectedValue,
    ListBox_zeit.SelectedValue) Then
```

Wenn dies nicht der Fall ist, blenden wir den nicht mehr verfügbaren Raum in seiner Liste aus:

```
    ListBox_raum.Items(raum_index).Enabled = False
  End If
Next
End Sub
```

5.3.17 Button_abbruch_Click

Wenn die Dozentin die Abbruch-Schaltfläche anklickt, ruft das Laufzeitsystem das Unterprogramm

```
Protected Sub Button_abbruch_Click(sender As Object, e As
   EventArgs) Handles Button_abbruch.Click
```

auf, das die ausgewählte Veranstaltung in der Veranstaltungsliste deselektiert

```
GridView_veranstaltungen.SelectedIndex = -1
```

die Detaildarstellung unsichtbar macht

```
Panel_details.Visible = False
```

und eine eventuelle Fehlermeldung wegen eines falschen Aulis-Links ausblendet:

```
Label_aulis_fehler.Visible = False
End Sub
```

5.3.18 Verband_verfuegbar

Das Unterprogramm

```
Protected Function Verband_verfuegbar(verband As String, tag As
   String, zeit As String) As Boolean
```

gibt `true` zurück, wenn der über die Parameterliste angefragte Verband am ebenfalls als Parameter übergebenen Wochentag zur angefragten Blockzeit verfügbar ist.

Dazu untersuchen wir in der Liste der Veranstaltungen ob der fragliche Verband am fraglichen Wochentag zur fraglichen Blockzeit auftaucht:

```
Dim kritischer_verband =
From fraglicher_verband In veranstaltungen_xml.<veranstaltung>.<
   verband>
Where fraglicher_verband.Value = verband _
AndAlso fraglicher_verband.Parent.<tag>.Value = tag _
AndAlso fraglicher_verband.Parent.<zeit>.Value = zeit _
AndAlso Not fraglicher_verband.Parent.<id>.Value =
   GridView_veranstaltungen.SelectedRow.Cells(1).Text
```

Dabei müssen wir natürlich den gerade ausgewählten Verband mit `Not` aus der Abfrage ausschließen, da er ja schließlich durchaus schon vorher mit den gewünschten Daten in der Liste der Veranstaltungen eingetragen sein könnte.

Wenn wir einen anderen Eintrag mit den angefragten Daten gefunden haben, geben wir `False` zurück; andernfalls `True`:

```
Return Not kritischer_verband.Any
End Function
```

5.3.19 Raum_verfuegbar

Auf die gleiche Weise wie bei der Verbandsverfügbarkeitsprüfung in Verband_verfuegbar untersuchen wir, ob der gewünschte Raum am gewünschten Wochentag zur gewünschten Blockzeit verfügbar ist:

```
Protected Function Raum_verfuegbar(raum As String, tag As String
   , zeit As String) As Boolean
Dim kritischer_raum =
From fraglicher_raum In veranstaltungen_xml.<veranstaltung>.<
   raum>
Where fraglicher_raum.Value = raum _
AndAlso fraglicher_raum.Parent.<tag>.Value = tag _
AndAlso fraglicher_raum.Parent.<zeit>.Value = zeit _
AndAlso Not fraglicher_raum.Parent.<id>.Value =
   GridView_veranstaltungen.SelectedRow.Cells(1).Text
Return Not kritischer_raum.Any
End Function
```

5.3.20 Dozent_verfuegbar

Auf die gleiche Weise wie bei der Verbandsverfügbarkeitsprüfung in Verband_verfuegbar untersuchen wir, ob die gewünschte Dozentin am gewünschten Wochentag zur gewünschten Blockzeit verfügbar ist:

```
Protected Function Dozent_verfuegbar(dozent As String, tag As
   String, zeit As String) As Boolean
Dim kritischer_dozent =
From fraglicher_dozent In veranstaltungen_xml.<veranstaltung>.<
   dozent>
Where fraglicher_dozent.Value = dozent _
AndAlso fraglicher_dozent.Parent.<tag>.Value = tag _
AndAlso fraglicher_dozent.Parent.<zeit>.Value = zeit _
AndAlso Not fraglicher_dozent.Parent.<id>.Value =
   GridView_veranstaltungen.SelectedRow.Cells(1).Text
Return Not kritischer_dozent.Any
End Function
```

5.3.21 Veranstaltungsplanung_LoadComplete

Das Unterprogramm

```
Private Sub Veranstaltungsplanung_LoadComplete(sender As Object,
    e As EventArgs) Handles Me.LoadComplete
```

wird vom Laufzeitsystem aufgerufen, wenn die Daten der Seite und all ihrer Steuerelemente komplett geladen sind. GPT-4 schreibt dazu:

> *_LoadComplete kann verwendet werden, um Aktionen durchzuführen, die Zugriff auf alle geladenen Daten benötigen, einschließlich der Daten, die in Steuerelementen während ihrer Load-Ereignisse gesetzt wurden. Es ist ein guter Zeitpunkt, um Verarbeitungen durchzuführen, die von den vollständig geladenen Zuständen der Steuerelemente abhängen.*

Zu diesem Zeitpunkt können wir entscheiden, ob wir die Änderungen speichern-Schaltfläche freigeben möchten. Wir geben die Schaltfläche in zwei Fällen frei:

1. Wenn weder ein Wochentag, noch eine Blockzeit, noch ein Raum ausgewählt sind, wenn die Dozentin also alle Einträge dieser Veranstaltung löschen will oder aber
2. Wenn die Dozentin alle drei Ressourcen (Wochentag, Blockzeit, Raum) ausgewählt hat:

```
If _
  ListBox_tag.SelectedValue = "" AndAlso
  ListBox_zeit.SelectedValue = "" AndAlso
  ListBox_raum.SelectedValue = "" Or
  Not ListBox_tag.SelectedValue = "" AndAlso
  Not ListBox_zeit.SelectedValue = "" AndAlso
  Not ListBox_raum.SelectedValue = "" Then
  Button_speichern.Enabled = True
```

Wenn keiner der beiden Fälle vorliegt, kann die Dozentin ihren Teilwunsch nicht[17] abspeichern:

```
Else
  Button_speichern.Enabled = False
End If
End Sub
End Class
```

[17] Wir verhindern auf diese Weise, dass eine Dozentin „aus Versehen“ beispielsweise „prophylaktisch“ einen ganzen Wochentag für ihre Veranstaltung reserviert.

6 Prüfungsplanung

Das Prüfungsplanungsprogramm ist fast identisch mit dem Veranstaltungsplanungsprogramm. Wir stellen in `pruefungsplanung.aspx.vb` daher nur die Unterschiede dar.

6.1 `pruefungsplanung.aspx.vb`

Das Prüfungsplanungsprogramm unterscheidet sich vom Veranstaltungsplanungsprogramm im Wesentlichen in folgenden Punkten:

- Statt der Wochentage einer Veranstaltung bieten wir der Dozentin die einzelnen Termine (jeweils erster und zweiter Termin parallel) der Prüfungen an (Abschnitt 6.1.1).
- Statt üblicherweise acht Veranstaltungsblöcken gibt es üblicherweise vier potenzielle Prüfungstermine pro Tag:
  1. 08:00 - 11:00 Uhr
  2. 11:00 - 14:00 Uhr
  3. 14:00 - 17:00 Uhr
  4. 17:00 - 20:00 Uhr
- Wir müssen sicherstellen, dass jeder Semesterverband nur eine Prüfung am Tag hat. Es gibt daher bei der Prüfungsplanung ein Unterprogramm `Belegte_tage_verstecken`, das wir hier nicht im Detail beschreiben, da es analog zu `Belegte_zeiten_verstecken` aufgebaut ist.

6.1.1 `ListBox_tag_fuellen`

Wir zeigen hier exemplarisch am Unterprogramm

```
Sub ListBox_tag_fuellen()
```

worin sich die Prüfungsplanung von der Veranstaltungsplanung unterscheidet: Das Unterprogramm der Prüfungsplanung wird einmalig beim Start der Seite aufgerufen, um die möglichen Prüfungstermine in die entsprechende Listbox zu schreiben. Vor dem Befüllen entfernen wir alle vorherigen Einträge aus der Listbox:

```
ListBox_tag.Items.Clear()
```

Wir lesen wie in Termine_darstellen die in Abbildung 4.2 dargestellte XML-Datei der zu verplanenden Prüfungstermine aus dem aktuellen Veranstaltungsordner:

```
Dim tage_xml =
  XElement.Load(
  MapPath(
  "../xml/" &
  aktuelles_semester &
  "/pruefung/tage.xml"))
```

und extrahieren daraus die für alle Prüfungen geltende Wochendifferenz zwischen dem ersten und dem zweiten Prüfungstermin, die die Planerin in Abhängigkeit von der Länge der vorlesungsfreien Zeit und etwaigen Feiertagen a priori festgelegt hat:

```
Dim wochendifferenz As Integer = tage_xml.<wochendifferenz>.
   Value
```

Außerdem lesen wir die Datumsangaben der potentiellen Prüfungstermine:

```
Dim alle_tage =
  From tag As Date In tage_xml.<tage>.<tag>
  Select tag
```

In einer Schleife über alle Prüfungstermine

```
For Each tag In alle_tage
```

erzeugen wir für jeden Tag einen neuen Listeneintrag, der beide Prüfungstermine enthält

```
  Dim neuer_tag As New ListItem With {
    .Value = tag,
    .Text = tag.ToString("D") & " UND " & tag.AddDays(
       wochendifferenz * 7).ToString("D")
  }
```

und fügen ihn ans Ende der Listbox an:

```
  ListBox_tag.Items.Add(neuer_tag)
Next
End Sub
```

7 iplan.css

In einer CSS-Datei beschreiben wir die Formatierung von HTML-Elementen; wir bestimmen hier also, wie unsere Seite aussieht.

Neben ein paar unwesentlichen Anpassungen für die Gridviews und die Tabellen gibt es eine Besonderheit, auf die wir hier hinweisen möchten. In der letzten Zeit hat es vermehrt Abmahnungen gegeben, wenn auf Webseiten eigentlich frei verfügbare Google-Fonts verwendet wurden. GPT-4 meint dazu

> ***Datenschutz und DSGVO**: Der Hauptgrund für Abmahnungen im Zusammenhang mit Google Fonts ist oft der Datenschutz. Wenn eine Website Google Fonts verwendet, können Benutzerdaten (wie IP-Adressen) an Google-Server, typischerweise in den USA, übermittelt werden. Dies kann ein Problem im Hinblick auf die Datenschutz-Grundverordnung (DSGVO) darstellen, da die Übermittlung personenbezogener Daten in Länder außerhalb der EU strengen Regeln unterliegt.*

und schlägt als Lösung vor:

> ***Lokales Hosting der Schriftarten**: Eine Möglichkeit, dieses Problem zu umgehen, besteht darin, die Google-Schriftarten lokal auf dem eigenen Server zu hosten, anstatt sie direkt von Google-Servern zu beziehen. Dadurch wird verhindert, dass Benutzerdaten bei jedem Seitenaufruf an Google gesendet werden.*

Da wir für Iplan den Google-Font Nunito verwenden

```
html, body, h1, h2, h3, h4, h5, h6 {
  font-family: 'Nunito', sans-serif;
}
```

stellen wir die Schriftarten in einem gesonderten Ordner auf dem Server zur Verfügung, und referenzieren sie in der CSS-Datei:

```
@font-face {
  font-family: 'Nunito';
  font-style: normal;
  font-weight: 400;
  src: url('../fonts/nunito-v25-latin-regular.eot'); /* IE9
     Compat Modes */
  src: local(''), url('../fonts/nunito-v25-latin-regular.eot?#
     iefix') format('embedded-opentype'), /* IE6-IE8 */
```

```
  url('../fonts/nunito-v25-latin/nunito-v25-latin-regular.woff2
     ') format('woff2'), /* Super Modern Browsers */
  url('../fonts/nunito-v25-latin/nunito-v25-latin-regular.woff')
      format('woff'), /* Modern Browsers */
  url('../fonts/nunito-v25-latin/nunito-v25-latin-regular.ttf')
     format('truetype'), /* Safari, Android, iOS */
  url('../fonts/nunito-v25-latin/nunito-v25-latin-regular.svg#
     Nunito') format('svg'); /* Legacy iOS */
}
```

Literaturverzeichnis

[1] Wikipedia. (2023) Query-string. [Online]. Available: https://de.wikipedia.org/wiki/Query-String

[2] J. J. Buchholz. (2023) plan. Hochschule Bremen. [Online]. Available: https://m-server.fk5.hs-bremen.de/plan/plan.html

[3] ——. (2023) Semesterwochenstundenberechnung. Hochschule Bremen. [Online]. Available: https://m-server.fk5.hs-bremen.de/sws/

[4] Wikipedia. (2023) Icalendar. [Online]. Available: https://de.wikipedia.org/wiki/ICalendar

[5] ——. (2023) Asp.net. [Online]. Available: https://de.wikipedia.org/wiki/ASP.NET

[6] W3Schools. (2023) W3.css. [Online]. Available: https://www.w3schools.com/w3css