Software Engineering 1 – Modellierung

System zur Erstellung individueller Stundenpläne

Team-05

Samantha Tolxdorf (41321)
Mert Özdemir (41193)
Celal Akköprü (39652)
Mohammad Rasool Mirzai (41396)
Sascha Janssen (41075)
Justus Schlicht (40795)

Die Einleitung in den Bericht von Team-05

Im Rahmen des Moduls Software Engineering I haben wir als Team die Aufgabe erhalten, einen Prototyp für eine Webanwendung zu entwickeln. Dieses Projekt diente nicht nur dazu, unsere Programmierfähigkeiten zu verbessern, sondern auch die theoretischen Konzepte, die wir im Laufe des Semesters erlernt haben, in die Praxis umzusetzen. Dabei lag ein besonderer Fokus auf der Anwendung der Unified Modeling Language (UML), die uns ermöglichte, das System detailliert zu planen und zu visualisieren. Die UML ist eine Modellierungssprache zur Darstellung von Softwaresystemen. Sie unterstützt die Entwicklung einer Software durch Planung, Visualisierung und Dokumentation (vgl. Visual Paradigm 2022).
Dieser Bericht umfasst die Relevanz verschiedener UML-Modelle, welche die Struktur und die Funktionen unseres Prototyps definieren. Im Verlauf des Berichts werden wir genauer auf die projektspezifischen UML-Diagramme eingehen. Dazu gehört ein Anwendungsfalldiagramm, drei Aktivitätsdiagramme und zwei Klassendiagramme. Ein Zustandsdiagramm kam bei diesem Projekt nicht zur Anwendung, daher wird auf dieses Modell nicht genauer eingegangen.
Zusätzlich zur Modellierung mit UML haben wir einen Wireframe der Anwendung entwickelt (siehe Anhang), um das Layout und die Benutzeroberfläche zu planen. Die Website wurde ausschließlich mit HTML/CSS und Bash-Skripten erstellt und umfasst einen funktionalen User-Interface-Prototypen. Dadurch wird den Studierenden ermöglicht, ihren Stundenplan zu individualisieren, indem sie die zu belegenden Module nach eigenem Ermessen bestimmen.
Das vorliegende Projekt wurde von Mert, Celal, Justus, Samantha, Sascha und Mohammad als Team-05 abgeschlossen. Die Zusammenarbeit war von Beginn an von einer positiven und produktiven Atmosphäre geprägt. Besonderer Wert wurde auf eine klare Aufgabenverteilung und kontinuierliche Kommunikation gelegt. Regelmäßige Meetings und Statusupdates ermöglichten es allen Beteiligten, den Projektfortschritt zu verfolgen und sich gegenseitig zu unterstützen. Die positive Stimmung und der gemeinsame Wille zum Erfolg trugen maßgeblich zur Erreichung der Projektziele bei.
Im Folgenden werden die einzelnen Projektphasen detailliert beschrieben, dabei werden die verwendeten UML-Diagramme umfassend erläutert und die Entwicklung der Webanwendung veranschaulicht.

Veranschaulichung von Anwendungsfällen: Das Use-Case-Diagramm

Ein Anwendungsfalldiagramm bzw. Use-Case-Diagramm ist eine grafische Darstellung eines Systems, welches die Interaktion zwischen einem oder mehreren Akteuren und ebendiesem System beschreibt. Es zeigt die Funktionalität, erfasst die Anforderungen und hilft das Gesamte zu visualisieren (vgl. Creatly 2022).
Das Use-Case-Diagramm zeigt ein System zur Erstellung individueller Stundenpläne (siehe Abb. 1). Die Akteure sind Studierende und andere Benutzer der Webanwendung, welche in dem

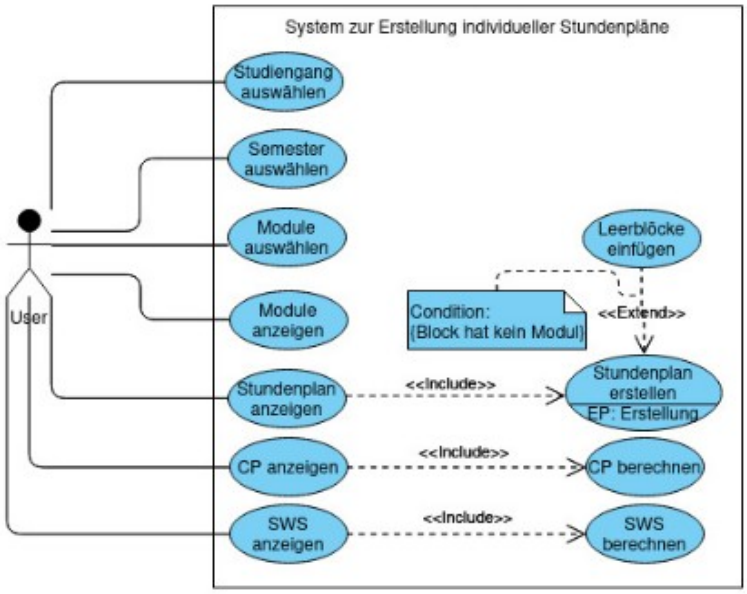


Abbildung 1: Use-Case Diagramm eines Systems zur Erstellung individueller Stundenpläne

Diagramm als User gekennzeichnet werden. Diese werden zusammengefasst als Strichmännchen dargestellt. Das System selbst wird durch das Rechteck gekennzeichnet und die sich darin befindenden Anwendungen werden durch das System selbst durchgeführt. Dadurch grenzt sich das System von den Akteuren ab. Durch das System wird ein Stundenplan erstellt, die CP sowie Semesterwochenstunden berechnet und anschließend dem Nutzer ausgegeben. Um den Stundenplan vollständig darzustellen, werden automatisch Leerblöcke eingesetzt falls dem Zeitblock kein Modul zugewiesen wurde. Damit das System diese Aufgaben erfüllt, muss der Nutzer zunächst einen Studiengang und ein Semester auswählen. Danach besteht die Möglichkeit die zu belegenden Module zu bestimmen.
Ein Use-Case ist eine Funktion innerhalb des Systems. Es wird als Oval gekennzeichnet und beinhaltet die benannte Funktion, welche durch das System ausgeführt werden soll. Eine Beziehung stellt eine Verbindung zwischen Akteuren und Use-Cases oder zwischen Use-Cases untereinander dar. Es gibt verschiedene Arten von Beziehungen, wie beispielsweise Assoziation, Generalisierung, Erweiterung und Einbeziehung. Sie werden als Linien mit unterschiedlichen Pfeilen oder Symbolen gekennzeichnet (vgl. Visual Paradigm 2022). Ein Use-Case-Diagramm stellt eine effektive Methode dar, um die Anforderungen an ein System zu ermitteln. Es ist leicht zu verstehen und bietet eine hervorragende Möglichkeit Ideen zu kommunizieren, da es große visuelle Aussagekraft besitzt. Es hilft, die Komplexität großer Projekte zu vereinfachen, indem es das Problem in Use-Cases unterteilt und die Anwendungen aus der Sicht der Benutzer darstellt (vgl. Creatly 2022).

(Celal Akköprü, 39652)

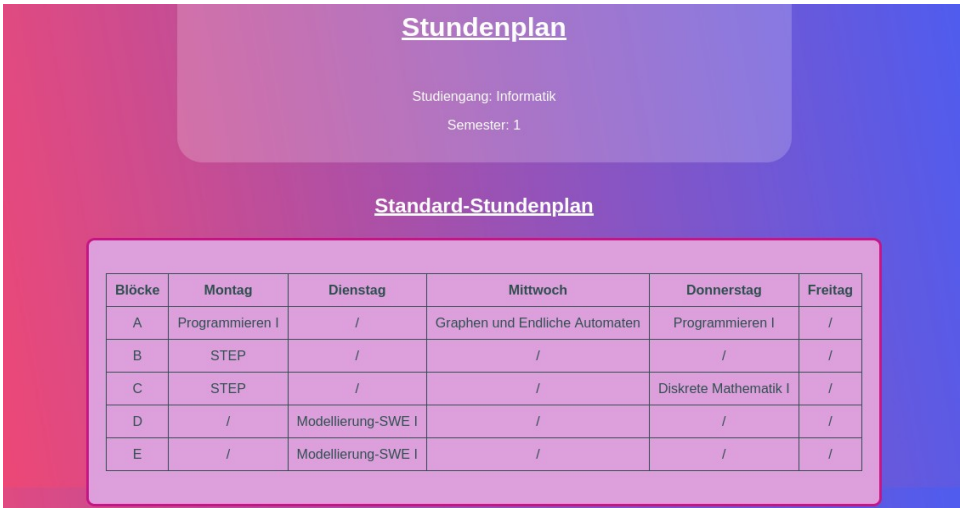
Die Entwicklung des Prototyps

Das Ziel des von uns entwickeltem Prototyps war es, dem User einen individuellen Stundenplan, gemäß seines Inputs bezüglich Semester, Studiengang und gegebenenfalls Modulen ausgeben zu lassen. Durch diese Funktionen soll es dem User erleichtert werden mehr Zeit für das Selbststudium der einzelnen Module zu finden, um so seine Leistungen zu verbessern (vgl. Kaushar 2013: 59).
Dafür wurde zuerst eine Startwebsite entwickelt, auf der die zutreffenden Eingaben getätigt werden können (siehe Abb. 2). Die Auswahl erfolgt über ein Dropdown-Menü, welches per HTML-Code

Abbildung 2: Startseite des SWE-Projekts von Team 05 (Quelle: https://informatik.hs-bremerhaven.de/docker-step2023-team-05-web/auswahl.html)

eingefügt wurde. Da es sich bei diesem Projekt lediglich um einen Prototypen handelt, haben wir uns dazu entschieden die Auswahl auf die Studiengänge Informatik und Wirtschaftsinformatik, sowie auf die Abbildung des ersten Semesters zu beschränken. Nach der Bestätigung der Eingabe wird der User auf eine zweite Website weitergeleitet, indem der Query-String, welcher bei der Auswahl des Users erstellt wird, an die zweite Website übergeben wird.

Auf der zweiten Website werden zunächst die Angaben des Users im Header der Website wiederholt. Dafür wurden die übermittelten Angaben aus dem Query-String in den Variablen „Studiengang“ und „Semester“ abgespeichert, um diese anschließend auf der Website darzustellen. Nachfolgend wird der, von der Hochschule Bremerhaven vorgesehene Stundenplan in Form einer HTML-Tabelle erstellt (siehe Abb. 3). Um die Tabelle zu erstellen ruft das Skript der Website automatisch ein weiteres Bash-Skript namens „testing.sh“ auf. Das „testing.sh“ Skript greift bei der



Blöcke	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
A	Programmieren I	/	Graphen und Endliche Automaten	Programmieren I	/
B	STEP	/	/	/	/
C	STEP	/	/	Diskrete Mathematik I	/
D	/	Modellierung-SWE I	/	/	/
E	/	Modellierung-SWE I	/	/	/

Abbildung 3: Darstellung des Standard-Stundenplan von Seite 2 des SWE-Projekts von Team 05 (https://informatik.hs-bremerhaven.de/docker-step2023-team-05-web/cgi-bin/skript1.sh?studiengang=Informatik&semester=1)

Generierung der HTML-Tabelle auf die „cp.csv“-Datei zu. Diese Datei beinhaltet Informationen zu den Modulen des Studiengangs, die wir aus dem Modulhandbuch sowie dem Stundenplan der Hochschule zusammengetragen haben (vgl. Hochschule Bremerhaven 2022; vgl. Hochschule Bremerhaven 2023). Das „testing.sh“ Skript durchläuft innerhalb einer For-Schleife die CSV-Datei und gibt für die entsprechenden Zeitblöcke innerhalb des Stundenplans die übereinstimmenden Kurse oder Leerblöcke zurück, cuttet dieses, sodass nur noch der Name des Moduls wiedergegeben wird, und fügt die relevanten HTML-Tags hinzu (siehe Abb. 4).

```
#!/usr/bin/env bash

for i in {A..E}
do
jo=$(cat /home/docker-step2023-team-05/cp.csv| grep
-w "Informatik"| grep -w "$i"|sort -t ';' -k7|sed 's
/^/|/g'|cut -d ';' -f1| tr '\n' ' '| sed -e "s/^|/<t
r><td>$i<\/td><td>/g" -e 's/ |/<\/td><td>/g')
echo "$jo"
done
```

Abbildung 4: testing.sh Skript (Quelle: testing.sh)

Ergänzend zum Stundenplan erhält der User außerdem Informationen darüber, wie viele Stunden des Selbststudiums für diesen Stundenplan vorgesehen sind und die Anzahl der Credit-Points die am Semesterende erreicht werden können.
Der User kann sich nun entscheiden diesen standardisierten Stundenplan zu akzeptieren oder weitere Änderungen daran vorzunehmen. Dafür wurde am Ende der zweiten Website eine Checkbox-Ausgabe mittels HTML-Code generiert. Diese Auswahl ermöglicht dem User die von ihm präferierten Module zu wählen und sich so einen individuellen Stundenplan erstellen zu lassen.

Die ausgewählten Kurse werden mit Hilfe eines Query-Strings an die dritte Website des Projektes übergeben.
Sowohl die Darstellung der Website des standardmäßigen Stundenplans als auch die des optimierten Stundenplans verwenden dasselbe Layout. Allerdings werden bei der Website des optimierten Stundenplans zuallererst die ausgewählten Module wiedergegeben. Dafür wurde hier der Query-String der zweiten Website aufgegriffen, gecuttet und in fünf Variablen abgespeichert.
Um die Darstellung des angepassten Stundenplans zu ermöglichen testet das Skript vorab, welche der Variablen mit Inhalt belegt sind und schreibt diese anschließend, entsprechend der Ergebnisse, in die Datei „kurse.dat“ (siehe Abb. 5). Im nächsten Schritt wird die Datei nochmals aufgerufen und

```
if test -z $Kurs2; then
  echo "$Kurs1" > kurse.dat
elif test -z $Kurs3; then
  echo "$Kurs1;$Kurs2" > kurse.dat
elif test -z $Kurs4; then
  echo "$Kurs1;$Kurs2;$Kurs3" > kurse.dat
elif test -z $Kurs5; then
  echo "$Kurs1;$Kurs2;$Kurs3;$Kurs4" > kurse.dat
else
  echo "$Kurs1;$Kurs2;$Kurs3;$Kurs4;$Kurs5" > kurse.dat
fi
wait

cat kurse.dat | tr ';' '\n' > kurse1.dat
cat kurse1.dat | while read k1 Rest; do
grep "$k1" /home/docker-step2023-team-05/cp.csv >> Module.dat
done
```

Abbildung 5: Codeausschnitt aus skript2.sh. Zu sehen ist der Variablen-Test, die Erstellung und Veränderung der "kurse"-Datein sowie der "Module"-Datei (Quelle: skript2.sh)

umgeschrieben, sodass die Modulnamen untereinander aufgeführt sind. Diese Angaben werden dann in der Datei „kurse1.dat“ abgespeichert. Durch die Umschreibung der Datei war es uns möglich, mithilfe einer While-Schleife, die zusätzlichen Informationen für die ausgewählten Module aus der ursprünglichen CSV-Datei zu entnehmen. Diese Daten speichert das Skript anschließend in der Datei „Module.dat“ ab (siehe Abb. 5). Durch dieses Vorgehen stellen wir sicher, dass in der Datei „Module.dat“ nur die ausgewählten Kurse des Users enthalten sind.
Im Anschluss werden zwei For-Schleifen durchlaufen um den Stundenplan, gemäß der Angaben des Users, zu erstellen (siehe Abb. 6). Die äußere For-Schleife iteriert über die Zeitblöcke von A bis E und erstellt dabei die zu schreibende Zeile innerhalb der HTML-Tabelle. Die innere For-Schleife

```
for Block in {A..E}
do
  echo "<tr><td>$Block</td>"
  for Wochentag in {1..5}
  do
   cat Module.dat | if grep -q "$Block;$Wochentag$"
    then
      Blub=$(grep "$Block;$Wochentag$" Module.dat|
cut -d ';' -f1 | uniq)
      echo "<td>$Blub</td>"
    else
      echo "<td>/</td>"
    fi
  done
  echo "</tr>"
done
```

Abbildung 6: Codeausschnitt aus skript2.sh. Zu sehen sind die zwei ineinander verschachtelten For-Schleifen zur Generierung des Stundenplans (Quelle: skript2.sh)

iteriert über die Wochentage. Im inneren der zweiten For-Schleife wird zunächst die zuvor erstellte

„Module.dat“-Datei aufgerufen und eine bedingte Abfrage gestartet. Wenn die aktuell durchlaufene Kombination aus Zeitblock und Wochentag in der „Module.dat“-Datei vorhanden ist, wird diese aus der Datei entnommen und in einer Variable abgespeichert. Anschließend wird diese Variable mit den korrekten HTML-Tags wiederholt, um sie in der Tabelle darzustellen. Sollte die Kombination aus Zeitblock und Wochentag nicht in der „Module.dat“-Datei vorhanden sein, wird ein Leerblock erstellt. Nachdem beide For-Scheifen durchlaufen sind wird der optimierte Stundenplan auf der Website ausgegeben (siehe Abb. 7) und die Datei „Module.dat“ wieder gelöscht.

Blöcke	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
A	/	/	Graphen und Endliche Automaten	/	/
B	/	/	/	/	/
C	/	/	/	Diskrete Mathematik I	/
D	/	Modellierung-SWE I	/	/	/
E	/	Modellierung-SWE I	/	/	/

Abbildung 7: Darstellung des optimierten Stundenplans von Seite 3 des SWE-Projekts von Team 05 (Quelle: https://informatik.hs-bremerhaven.de/docker-step2023-team-05-web/cgi-bin/skript2.sh?1=Mathematik_I&4=Graphen&5=SWE_I)

Zuletzt wird ebenfalls auf dieser Website der Selbstlernaufwand sowie die Credit-Point Anzahl für die ausgewählten Module dargestellt. Um diese Informationen angeben zu können, wird im Skript erneut der Query-String aufgegriffen, umgeschrieben und in die Datei „tmpLine.dat“ gespeichert. Diese Datei wird als Counter für die darauf folgende For-Schleife genutzt. Dafür werden die Zeilen innerhalb der Datei gezählt und in der Variable „counter“ gespeichert. Innerhalb der For-Schleife wird die Variable „Kurs“ deklariert und mithilfe einer Kommandosubstitution mit dem entsprechendem Kursnamen initiiert (siehe Abb. 8). Anschließend wird eine weitere Variable

```
for i in $(seq $counter); do

  Kurs=$(cat tmpLine.dat | cut -d '=' -f2|
nl -s ';' | grep "$i" | sed 's/_/ /g' | cut
-d ';' -f2)
  cp=$(cat /home/docker-step2023-team-05/cp
.csv | grep -w "Informatik" | grep -w "$Kurs
" | sed -e 's/;[A-E];//g' -e 's/[1-5]$//g' |
 uniq | cut -d ';' -f2 | tr '\n' ' ')

  cpG=$((cpG+cp))
```

Abbildung 8: Codeausschnitt aus skript2.sh. Zu sehen ist Berechnung der Gesamtanzahl an Credit-Points (Quelle: skript2.sh)

namens „cp“ per Kommandosubstitution initiiert. Die Variable „cp“ beinhaltet die Credit-Points des vorher von User ausgewählten Moduls. Um die Gesamtanzahl an Credit-Points ausweisen zu können wird die Variable „cpG“ benötigt. In dieser Variable werden nach jedem Durchlauf der For-Schleife die Credit-Points der ausgewählten Module addiert und anschließend ausgegeben. Für die Ausgabe des Selbstlern-Aufwands wurde ein analoges Vorgehen verwendet. Damit ist die Ausgabe der Website beendet und der User kann alle vorgesehenen Informationen auf der Website ablesen.

(Samantha Tolxdorf, 41321; Mert Özdemir, 41193; Sascha Janssen, 41075)

Visualisierung von Abläufen anhand von Aktivitätsdiagrammen

Das Aktivitätsdiagramm ist die detaillierteste UML-Darstellung eines Vorgangs. In diesem werden alle ablaufenden Prozesse hierarchisch und chronologisch, mit einem Start- und Endknoten dargestellt. Aufgrund dessen eignet es sich, ähnlich wie das Use-Case-Diagramm, die Funktionsweisen eines Systems zu beschreiben, wobei ein Aktivitätsdiagramm sogar als Bedienungsanleitung dienen kann (vgl. Seemann und Wolff von Gudenberg 2006: 2-3). Ein entscheidender Unterschied der beiden Modelle ist der Detaillierungsgrad. Da Aktivitätsdiagramme nicht nur Anwendungsfälle darstellen, sondern auch über welche Methoden diese erreicht werden, welche Bedingungen eventuell zuvor erfüllt werden müssen und in welcher Reihenfolge die verschiedenen Prozesse stattfinden. Des Weiteren eignen sich Aktivitätsdiagramme als optimale Darstellung für dynamische Prozesse und angewendete Schleifen in der Informatik, da diese mit logischen Operatoren innerhalb der Abbildung dargestellt werden können.

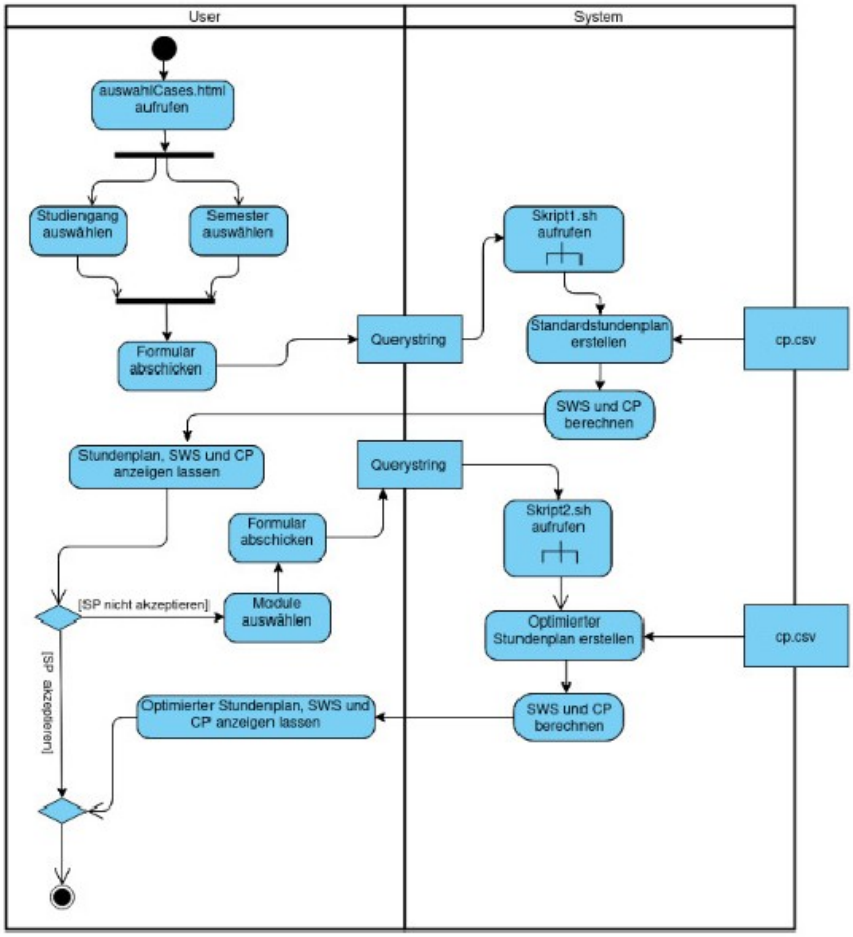


Abbildung 9: Schwimmbahn-Diagramm eines Systems zur Erstellung individueller Stundenpläne (Quelle: Eigene Darstellung)

Das Schwimmbahn-Diagramm, bei dem Prozesse zwischen dem Benutzer und dem entwickelten System aufgeteilt wurden, diente als anschauliche Darstellung der interaktiven Prozesse unseres Prototyps (siehe Abb. 9). Dies war vor allem von Bedeutung, da unser Projekt als dynamische Website fungieren sollte. Die Visualisierung des User-Inputs spielte eine entscheidende Rolle bei der Erstellung des Aktivitätsdiagramms. Mittels eines „logischen Und“-Operators für die Auswahl in den Formularen und eines „logischen Oder“-Operators für die Entscheidung zwischen den Stundenplänen, wird der User-Input effektiv dargestellt. Auch verschachtelte Aktivitäten lassen sich im Falle von Kompartimentierung visualisieren, was sich am Beispiel der Aktivität „Skript2.sh aufrufen“ innerhalb unseres Schwimmbahn-Diagramms erkennen lässt. Dieser Prozess dient als Aktivitätsaufruf eines weiteren Aktivitätsdiagramms (siehe Abb. 10).

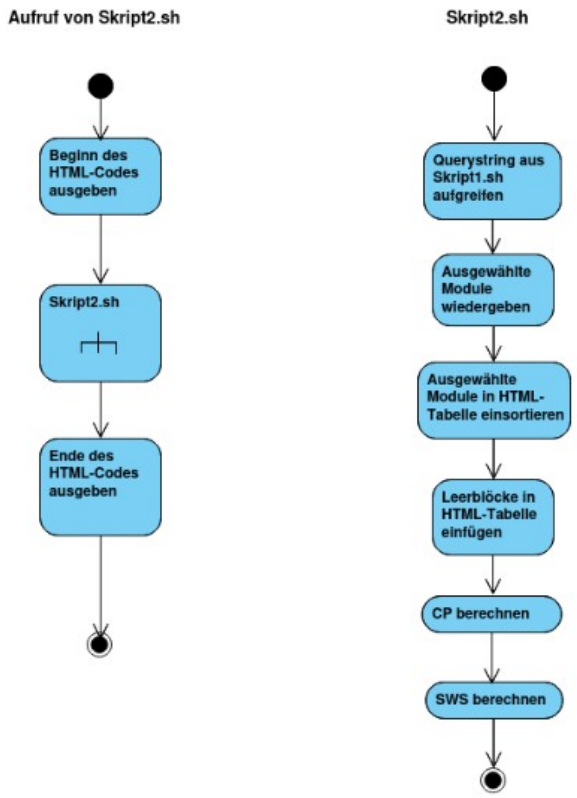


Abbildung 10: Anwendungsfalldiagramme zum Aktivitätsaufruf des Skript "skript2.sh" (Quelle: Eigene Darstellung)

(Justus Schlicht, 40795)

Die Bausteine des Systems: Klassendiagramme im Fokus

Ein Klassendiagramm visualisiert die Beziehungen zwischen verschiedenen Klassen innerhalb eines Systems. Dabei fasst eine Klasse Objekte zusammen, die gemeinsame Eigenschaften besitzen (vgl. Chornaya 2022). Das Klassendiagramm wurde erstellt, um die Struktur und Beziehungen der relevanten Klassen unserer Website darzustellen.
Die Klasse „Stundenplan“ setzt sich aus verschiedenen Methoden, Attributen und elementaren Klassen, mit ihren jeweiligen Attributen, zusammen (siehe Abb. 11). Die elementaren Klassen dieses Diagramms umfassen StudiengangT (Studiengang), SemesterT (Semester), ArbeitspensumT (Semesterwochenstunden und CP), ModulT (Modul) und BlockT (Block). Innerhalb der Klasse StudiengangT werden die Attribute „Winf“ und „Inf“ definiert, welche die verschiedenen

Studiengänge repräsentieren. Die Klasse SemesterT enthält die verschiedenen Semesterstufen. ArbeitspensumT besitzt die Attribute Arbeitsaufwand, Credit-Points (CP) und Semesterwochenstunden (SWS), die die Arbeitsbelastung für ein bestimmtes Modul repräsentieren. Anschließend wurden die elementaren Klassen ModulT, welche die verschiedenen Module der Studiengänge beinhaltet, und BlockT, die sich aus den Attributen Uhrzeit und Wochentag zusammensetzt, definiert.
Die Methoden der Klasse „Stundenplan“ ermöglichen es die Informationen der diversen elementaren Klassen abzurufen, als Beispiel können hier die Methoden „SWSberechnen()“ und „CPberechnen()“ herangezogen werden, welche dazu dienen den Arbeitsaufwand zu berechnen. Wenn für eine Variable ein bestimmter Wert angegeben werden soll, wie beispielsweise „Inf“ und „Winf“ bei der elementaren Klasse „StudiengangT“, sollte, statt des Typen Strings, ein eigener Aufzählungstyp (<<enumeration>>) definiert werden. (vgl. Randen et al. 2016: 107).

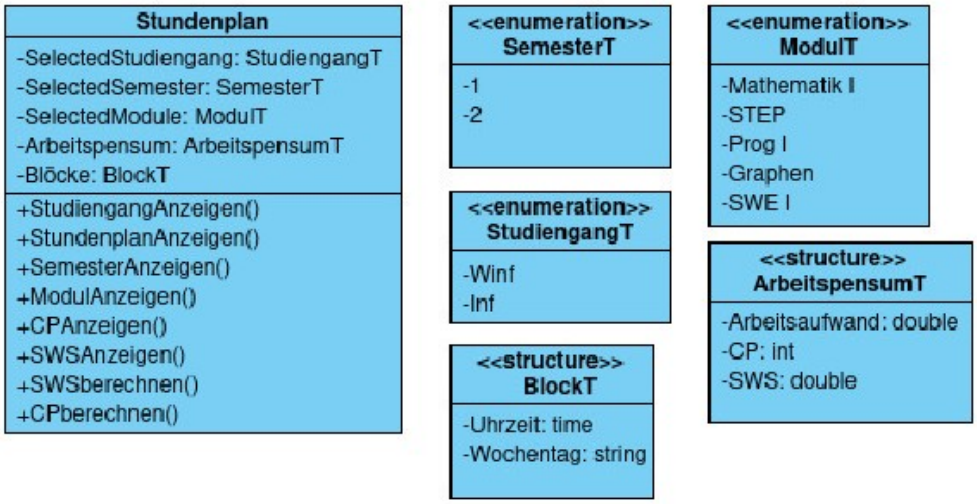


Abbildung 11: Klassendiagramm der Klasse „Stundenplan“ (Quelle: Eigene Darstellung)

Das zweite Klassendiagramm beschreibt die Beziehung zwischen der Klasse „Modul“ und der elementaren Klasse „BlockT“ (siehe Abb. 12). Die Klasse "Modul" verfügt über die Attribute Block (BlockT), CP (CPT) und SWS (SWST). Die Klasse „BlockT“ speichert die Uhrzeit und den Wochentag eines bestimmten Moduls, während die Klasse „Modul“ die Credit-Points und die Semesterwochenstunden verwaltet.

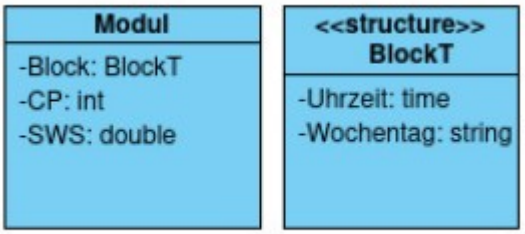


Abbildung 12: Klassendiagramm der Klasse "Modul" (Quelle: Eigene Darstellung)

(Mohammad Mirzai, 41396)

Das Fazit

Das Projekt wurde mit dem Ziel begonnen, einen individualisierten Stundenplan zu entwickeln, der sowohl die Arbeitsstunden als auch die Credit-Points für das dargestellte Semester berücksichtigt. Dabei sollte den Nutzern die Möglichkeit gegeben werden, Module nach ihren Präferenzen auszuwählen, woraufhin der Stundenplan automatisch angepasst wird. Dieser innovative Ansatz erleichtert es den Studierenden erheblich, ihr Studium und eventuelle Erwerbstätigkeiten sowie ihre Freizeitaktivitäten effizient zu organisieren und zu optimieren. Trotz dem zielführenden Abschluss des Projektes und des bereits funktionierenden Prototyps, besteht noch Raum für Weiterentwicklungen und Verbesserungen.
Insbesondere sollte die Anwendung um weitere Fachrichtungen und Semester erweitert werden, um eine größere Vielfalt an Studierenden anzusprechen. Zusätzliche Funktionen wie ein Reset-Button und weitere Elemente des User-Interfaces könnten die Benutzerfreundlichkeit erhöhen und das Gesamterlebnis verbessern. Des Weiteren wäre es vorteilhafter gewesen, anstelle eines umfangreichen Skripts, mehrere kleinere Skripte zu verwenden, um eine bessere Übersichtlichkeit bei der Arbeit am Prototyp zu gewährleisten.
"Klassendiagramme sind ein zentrales Werkzeug der objektorientierten Modellierung." (Booch et al. 1999: 243), dieser Aussage können wir nur zustimmen. Während des gesamten Entwicklungsprozesses haben UML-Diagramme eine wichtige Rolle gespielt, um das Team auf dem selbem Wissensstand zu halten und die Kommunikation zu erleichtern. Die Zusammenarbeit innerhalb des Teams war äußerst ergiebig, was sich in einem funktionsfähigen und ästhetisch ansprechenden Prototyp widergespiegelt hat.
Die Rückmeldungen bezüglich diesem Projekts legen nahe, dass es immenses Potenzial hat, weiter zu wachsen und sich zu einem unverzichtbaren Werkzeug für Studierende verschiedener Fachrichtungen zu entwickeln. Mit einem klaren Fokus auf kontinuierliche Verbesserungen und Benutzerfreundlichkeit kann dieses Projekt in Zukunft einen positiven Einfluss auf die Studienerfahrung vieler Menschen haben.

Anhang:

Wireframe des SWE-Projekts von Team-05 : Ein System zur Erstellung individualisierter Stundenpläne

Auswahlseite:

Abbildung 13: Wireframe der Auswahlseite des SWE-Projekts vom Team 05 (Quelle: Eigene Darstellung)

Seite 2 des Projektes:

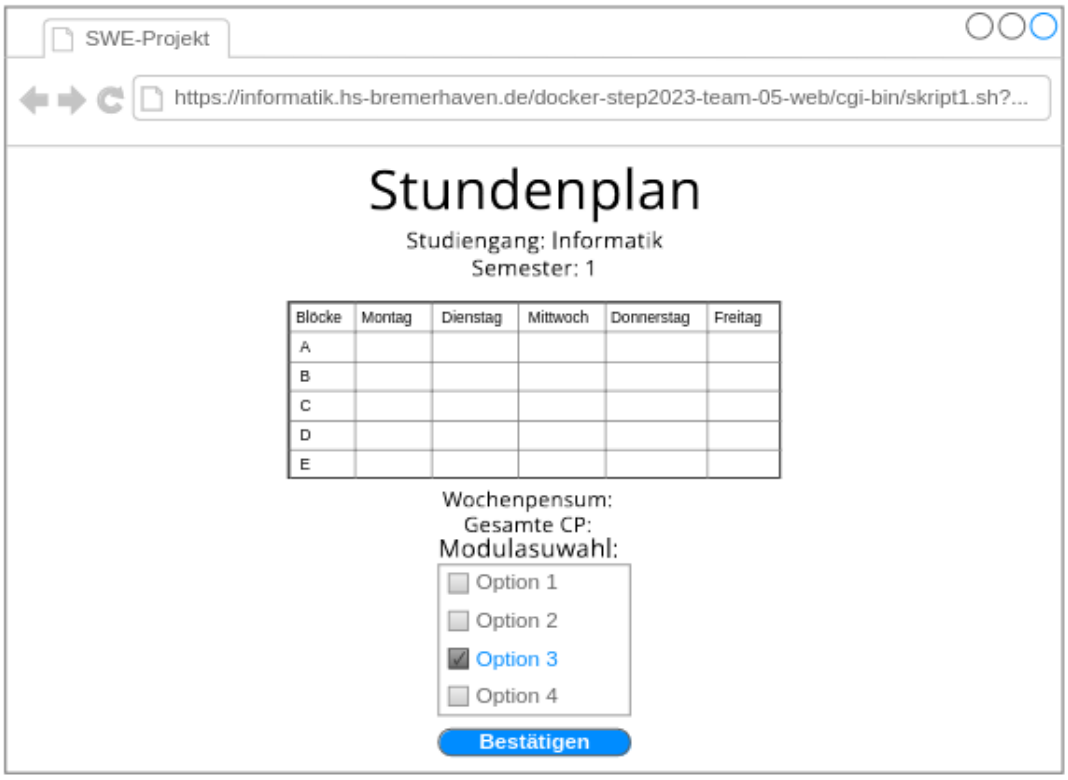


Abbildung 14: Wireframe der zweiten Seite des SWE-Projekts vom Team 05 (Quelle: Eigene Darstellung)

Seite 3 des Projektes:

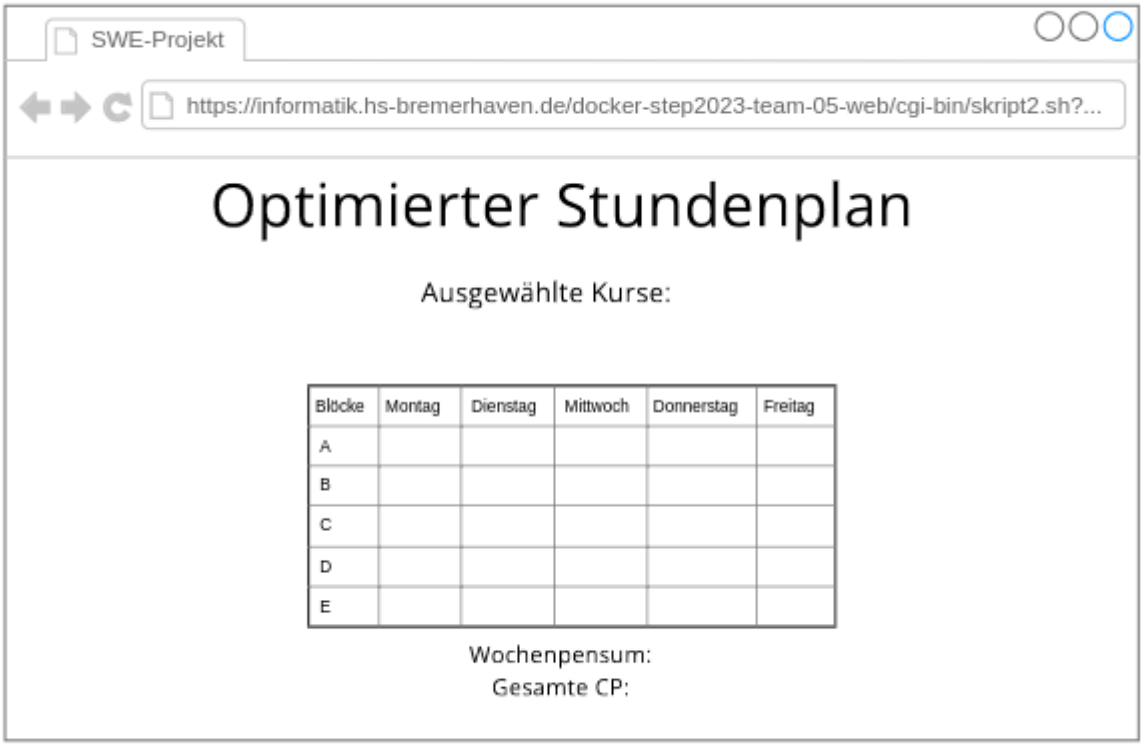


Abbildung 15: Wireframe der dritten Seite des SWE-Projekts vom Team 05 (Quelle: Eigene Darstellung)

Abbildungsverzeichnis:

Literaturverzeichnis

Booch Grady, Ivar Jacobson und James Rumbaugh (1999): *Objektorientierte Analyse und Design mit UML*, 2. Aufl., Boston: Addison-Wesley.

Chornaya, Jenia (2022): Klassendiagramm: Definition, Erstellung und Beispiel[online] https://blog.hubspot.de/website/klassendiagramm [28.02.2024].

Creatly (2022): Use Case Diagram Tutorial (Guide with Examples) [online] https://creately.com/guides/use-case-diagram-tutorial/ [28.02.2024].

Hochschule Bremerhaven (2022): Modulhandbuch Informatik Wirtschaftsinformatik [online] https://www.hs-bremerhaven.de/inf [24.02.2024].

Hochschule Bremerhaven (2023): Semesterplan Informatik Semester 1[online] https://www4.hs-bremerhaven.de/fb2/ws2324.php?action=showplan&weeks=46&fb=%23SPLUS938DBF&idtype=&listtype=&template=Set&objectclass=Studenten-Sets&identifier=%23SPLUS037FE4&days=1;2;3;4;5;6&tabstart=46 [24.02.2024].

Kaushar, Mehnaz (2013): *Study of Impact of Time Management on Academic Performance of College Students*, in: Journal of Business and Management, Vol. 9, Issue 6, S. 59-60.

Randen, Hendrik Jan van, Christian Bercker und Julian Fieml (2016): *Einführung in UML - Analyse und Entwurf von Software*, Springer Vieweg (Hrsg), Wiesbanden, Springer Fachmedien.

Seemann, Jochen und Jürgen Wolff von Gudenberg, (2006): *Das Aktivitätsdiagramm. Software-Entwurf mit UML 2: Objektorientierte Modellierung mit Beispielen in Java*, Springer Verlag, S. 27-41.

Visual Paradigm (2022): What is Unified Modeling Language (UML)? [online] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/ [28.02.2024].

Visual Paradigm (2022): Use Case Diagram Tutorial [online] https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/ [28.02.2024].

Visual Paradigm (2022): Die Vier Arten Von Beziehungen Im Anwendungsfalldiagramm [online] https://blog.visual-paradigm.com/de/the-four-types-of-relationship-in-use-case-diagram/ [28.02.2024].