

Buchungssystem

Dokumentation zur Modulprüfung MP226

Gruppenmitglieder:

Name: Eberhard Linus

Name: Frei Sven

Kurzbeschreibung

In dieser Dokumentation wird das Buchungssystem, welches das Budget und die dazugehörigen Änderungen einer Person aufzeigen. Dazu haben wir mehrere verschiedene Funktionen implementiert. Es werden auch theoretische Schritte erklärt, wie die unterschiedlichen UML Diagramme.

Inhaltsverzeichnis

1	ZIELE UND PLANUNG	2
1.1	ANFORDERUNGSSPEZIFIKATION	2
1.2	ZEITLICHE PLANUNG	3
2	ANALYSE UND DESIGN	5
2.1	UML DIAGRAMME	5
2.1.1	Anwendungsfalldiagramm	5
2.1.2	Aktivitätsdiagramme	18
2.1.3	Klassendiagramme	20
	20
2.2	GUIs	21
2.3	DATENBANK	24
2.3.1	Datenbankmodell	24
2.3.2	Datenbank SQL	24
2.3.3	Allgemeiner Aufbau	25
2.3.4	Tabellen	26
3	SOFTWAREDOKUMENTATION	28
3.1	BESCHREIBUNG	28
3.1.1	Controller	28
3.1.2	Models	29
3.1.3	Views	29
3.1.4	Schnittstellen	30
3.2	PATTERNS, KONZEPTE, MODELLE, ETC.	31
3.2.1	Datenbankzugriffs Framework	31
3.3	SCHWIERIGKEITEN UND KRITISCHE STELLEN	31
3.4	GEWÄHLTE TESTFÄLLE, TESTKONZEPT	32
4	ARBEITSKONZEPT	32
4.1	AUFTEILUNG UND ZUSAMMENARBEIT	32
5	REFLEXION	33
5.1	LINUS EBERHARD	33
5.2	SVEN FREI	33
5.3	SCHLUSSWORT	33
6	BEILAGEN	33

1 Ziele und Planung

1.1 Anforderungsspezifikation

Das Endprodukt sollte vom Benutzer Daten einlesen können und diese in die dazugehörige Datenbank einfügen. Damit in anderen Funktionen dieser als Output benutzt werden kann. Der erste Schritt für den Benutzer ist sich überhaupt einmal registrieren zu können. Dafür sollte er persönliche Daten eingeben, darunter die Email-Adresse. Diese sollte eigentlich verifiziert werden, wofür uns aber die Infrastruktur fehlt. Beim Nachfolgenden Schritt sollte man sich als Benutzer einloggen können, d.h. die Email-Adresse und das dazugehörige Passwort soll mit den Eingaben überprüft werden.

Wenn diese beiden Schritte gemacht wurden, sollte erst die eigentliche Aufgabe der Applikation zur Geltung kommen. Denn ab diesem Punkt werden die personenbezogenen Daten ausgelesen und im Hintergrund ausgerechnet und auf der Oberfläche ausgegeben. Dazu wird eine Datenbank benötigt, welche richtig konstruiert wurde, diese wird später erklärt. Jedoch schon vorab, haben wir uns für einen einfacheren und zentralisierten Datenbankzugriff entschieden. Damit wir nicht in allen Klassen unterschiedlich auf die Datenbank zugreifen. Dafür haben wir ein eigenes Framework für diesen Zugriff erstellt.

Die Applikation sollte Einnahmen und Ausgaben erfassen können, welche erst an einem bestimmten Datum vom aktuellen Budget abgezogen werden, indem man ein Datum mitgibt. Zusätzlich kann man auch einen Dauerauftrag erstellen, welcher an einem Datum anfängt und wieder an einem endet. Damit man auch weiss was man als Benutzer eingegeben hat, kann man einen Verlauf seines Budget ansehen. Dieser wird tabellarisch ausgegeben, mit den Änderungen und an welchem Datum diese erfolgten. Es ist auch möglich den aktuellen Monat mit Hilfe einer Grafik ausgeben zu können. Somit hat man eine einfachere Einsicht in den Zeitraum in dem man sich befindet. Schlussendlich kann man somit Budgets jeglicher Art verwalten und hat immer einen Einblick in die momentane Lage.

Falls man nicht mehr mit seinen Daten, welche man zu Beginn bei der Registrierung eingegeben hat, zufrieden ist. Kann man diese ohne einwirken eines Admins anpassen. Jedoch haben wir uns nicht damit befasst zum Einträge der Budgettabelle anpassen zu können.

Wir konzentrieren uns jedoch nicht so stark auf das Aussehen der Applikation, sondern auf die Funktionalität. Damit wir mehr Zeit in Berechnungen und Möglichkeiten für die Ausgabe haben.

Wir haben uns auch nicht damit befasst, diese Daten direkt exportieren zu können. Da wir eher wollen, dass mit dieser Applikation das Budget verwaltet wird. Und auch hier hätten wir auf die Ausgabeformatierung achten müssen, was wieder Zeit benötigt hätte.

Beim Anmelden oder auch bei der Registrierung achteten wir nicht auf das Verschicken einer Email, denn es ist kein SMTP-Server bereitgestellt. Darum ist es auch schade, falls jemand das Passwort vergessen würde, wäre dieses sozusagen verloren, da man es nicht von ausserhalb zurücksetzen kann. Dann sind auch die Daten, des Budget der Personen, auch verloren gegangen, da kein Zugriff mehr besteht.

1.2 Zeitliche Planung

Bei unserem Projekt gingen wir schon von Anfang an nach dieser Planung vor. Sie bekam einzelne Veränderungen, weil wir nicht immer nach Zeitplan arbeiten konnte. Z.B. brauchten wir bei den Berechnungen im Hintergrund bedeutend mehr Zeit als vorgesehen. Was aber gegen das Ende des Projektes wieder aufging, da wir sonst nicht ganz soviel Zeit benötigten. Jedoch ist diese Planung hier die angepasste Version.

Linus Eberhard	→ blau
Sven Frei	→ grün
Beide	→ rot

Zeitplanung	
Datum	Was
20.05	<p>Bis zur Lektion: Idee des Projektes bis zu diesem Zeitpunkt suchen und Brainstormartig notieren.</p> <p>In der Lektion:</p> <ul style="list-style-type: none"> Ideen fertig selektieren Spezifikationen für die Applikation festlegen Planung machen Programmarchitektur festlegen
27.05	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> Planung fertig machen Programmarchitektur fertig machen Datenbankzugriff überlegt und Notizen dazu gemacht Datenbankdesign fertig machen <p>In der Lektion:</p> <ul style="list-style-type: none"> Programmarchitektur, Datenbankdesign und Datenbankzugriff besprechen Datenbank erstellen Datenbankzugriff erstellen Bisheriger Teil anfangen zu dokumentieren
03.06	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> Bisheriger Teil fertig dokumentieren Datenbank erstellt Datenbankzugriff zu teilen programmiert <p>In der Lektion:</p> <ul style="list-style-type: none"> GUI-Komponenten anfangen zu designen Datenbankzugriff weiter programmieren
10.06	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> GUI Komponenten fertig designed Datenbankzugriff für ersten Testlauf fertig programmiert.(RELEASE) <p>In der Lektion:</p> <ul style="list-style-type: none"> GUI Komponenten anfangen zu programmieren, inkl. dazugehörige Controller Datenbankzugriff testen auf Schwachstellen
17.06	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> GUI Komponenten erster Teil fertig programmiert(anmelden, registrieren und Datenabfüllen fertig → ohne Datenbankzugriff) Schwachstellen des Datenbankzugriff ausprogrammiert <p>In der Lektion:</p> <ul style="list-style-type: none"> GUI Komponenten mit Datenbankzugriff erweitern GUI Komponenten testen(RELEASE) Datenbankzugriff ausprogrammieren
25.06	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> Dokumentation erweitern mit den bis jetzt erstellten Teilen

	<p>In der Lektion:</p> <ul style="list-style-type: none"> - GUI Komponenten auf uns zwei aufteilen - GUI Komponenten weiterentwickeln
08.07	<p>Bis zur Lektion:</p> <ul style="list-style-type: none"> - GUI Komponenten weiterentwickeln - Dokumentation erweitern
	<p>In der Lektion:</p> <ul style="list-style-type: none"> - GUI Komponenten fertig entwickeln und testen(RELEASE) - Dokumentation erweitern
15.07	<p>Abschluss des Projektes</p> <ul style="list-style-type: none"> - Dokumentation fertig für Abgabe - Programm fertig für Abgabe(Abschliessender RELEASE) - Präsentation vorbereiten
	<p>In der Lektion:</p> <ul style="list-style-type: none"> - Präsentationsvorbereitung abschliessen

2.1.1.1 Beschreibung der Anwendungsfälle

Beschreibung Anwendungsfall	
Name	Passwort nachfragen
Kurzbeschreibung	Der Benutzer weiss sein eigenes Passwort nicht. Dafür kann er es nachfragen, wodurch das Passwort zurückgesetzt
Akteure	Unauthenticated User
Auslöser	Benutzer weiss sein eigenes Passwort nicht mehr
Ergebnis(se)	Passwort wird zurückgesetzt und dadurch neu vom Benutzer erstellt
Eingehende Daten	Benutzername, danach bei der nächsten Seite das Passwort
Vorbedingungen	Account muss vorhanden sein
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Seite öffnen "Passwort vergessen" • Benutzername oder Email angeben • Seite öffnen, welche durch das Email verschickt wurde • Passwort ändern und bestätigen
Offene Punkte	Email-Service
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Anmelden
Kurzbeschreibung	Der Benutzer meldet sich mit Passwort und Benutzernamen an.
Akteure	Unauthenticated User
Auslöser	Benutzer will auf sein Budget zugreifen.
Ergebnis(se)	Benutzer ist angemeldet
Eingehende Daten	Benutzername und das dazugehörige Passwort
Vorbedingungen	Account muss vorhanden sein und das dazugehörige Passwort ist richtig
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Passwort und Benutzername werden eingegeben • Bestätigung der Eingaben
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Registrierung aufnehmen
Kurzbeschreibung	Ein neuer Benutzer eröffnet ein neues Konto mit den erforderlichen Daten dafür.
Akteure	Unauthenticated User
Auslöser	Benutzer will ein neues Konto eröffnen.
Ergebnis(se)	Benutzer hat ein eigenes Konto
Eingehende Daten	Benutzername, Passwort & Email
Vorbedingungen	Benutzername und Email ist noch nicht vorhanden
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Registrierung Seite wird geöffnet • Gültige Daten werden eingegeben • Daten werden bestätigt
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Abmelden
Kurzbeschreibung	Benutzer beendet die Session
Akteure	Authenticated User
Auslöser	Benutzer will sich abmelden
Ergebnis(se)	Benutzer hat seine Session beendet
Eingehende Daten	-
Vorbedingungen	Benutzer ist angemeldet
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> Abmelde-Button wird gedrückt
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Budget-Grafik erzeugen
Kurzbeschreibung	Benutzer wechselt auf die Seite der Budge-Grafiken und sieht wie sein Budget verteilt ist
Akteure	Authenticated User
Auslöser	Benutzer will sich selbst eine graphische Übersicht über sein Budget verschaffen
Ergebnis(se)	Benutzer hat eine graphische Übersicht über sein Budget
Eingehende Daten	Budgetdaten aus der DB
Vorbedingungen	Benutzer ist angemeldet und hat Budgetdaten bereits eingegeben
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> Benutzer wählt die Seite "Grafiken" aus
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Profil Änderung in DB abspeichern
Kurzbeschreibung	Benutzer ändert seine eigenen Daten wie Passwort oder Email ab.
Akteure	Authenticated User
Auslöser	Benutzer will seine eigenen Daten abändern
Ergebnis(se)	Benutzer wurde in der DB angepasst
Eingehende Daten	Email und/oder Passwort
Vorbedingungen	Benutzer ist eingeloggt
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Benutzer wählt die Seite "Profilinformationen ändern" • Gibt Daten ein, in den Feldern welche geändert werden sollten • Bestätigung der Eingaben • Eingaben werden verarbeitet und Schlussendlich in der DB aktualisiert
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Gesamtbudget erfassen
Kurzbeschreibung	Benutzer erfasst seine Einnahmen in einem Monat.
Akteure	Authenticated User
Auslöser	Benutzer will seine Einnahmen eines Monats erfassen.
Ergebnis(se)	Zum Benutzer wird ein Eintrag in den Einnahmen in der DB gemacht.
Eingehende Daten	Budget, Datum
Vorbedingungen	Benutzer ist eingeloggt
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Benutzer wählt die Seite "Budget erfassen" • Gibt das Budget ein • Bestätigung der Eingaben • Eingaben werden verarbeitet und Schlussendlich in der DB aktualisiert
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Ausgabe erfassen
Kurzbeschreibung	Benutzer erfasst eine Ausgabe.
Akteure	Authenticated User
Auslöser	Benutzer will seine Ausgabe.
Ergebnis(se)	Zum Benutzer wird ein Eintrag in den Ausgabe in der DB gemacht.
Eingehende Daten	Ausgabe Wert, Datum
Vorbedingungen	Benutzer ist eingeloggt
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Benutzer wählt die Seite "Ausgabe erfassen" • Gibt den Ausgabewert ein • Bestätigung der Eingaben • Eingaben werden verarbeitet und Schlussendlich in der DB aktualisiert
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Dauerauftrag erfassen
Kurzbeschreibung	Benutzer erfasst eine Ausgabe / Einnahme als Dauerauftrag.
Akteure	Authenticated User
Auslöser	Benutzer will einen Dauerauftrag(Einnahme/Ausgabe) erfassen.
Ergebnis(se)	Zum Benutzer wird ein Eintrag in der Dauerauftragtabelle in der DB gemacht.
Eingehende Daten	Ausgabe Wert, Startdatum, Enddatum, Intervall
Vorbedingungen	Benutzer ist eingeloggt
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Benutzer wählt die Seite "Dauerauftrag erfassen" • Gibt die Werte ein • Bestätigung der Eingaben • Eingaben werden verarbeitet und Schlussendlich in der DB aktualisiert
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Ausgabe der Daten
Kurzbeschreibung	Benutzer fragt sein eigenes Budget ab.
Akteure	Authenticated User
Auslöser	Benutzer einen Überblick über sein Budget haben.
Ergebnis(se)	Der Benutzer hat einen Überblick über sein Budget
Eingehende Daten	-
Vorbedingungen	Benutzer ist eingeloggt
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> Benutzer wählt die Seite "Budget anzeigen"
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

Beschreibung Anwendungsfall	
Name	Benutzer verwalten
Kurzbeschreibung	Admin bearbeitet/schaltet frei einen Benutzer.
Akteure	Admin
Auslöser	Admin will Benutzer anpassen oder Datenbank reinigen.
Ergebnis(se)	Admin hat die Datenbank abgeändert
Eingehende Daten	Gewünschte Daten, welche abgeändert werden wollten
Vorbedingungen	Person ist berechtigt dafür
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> • Admin wählt die Option "Benutzer verwalten" • Ändert Daten nach belieben • Bestätigt diese
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

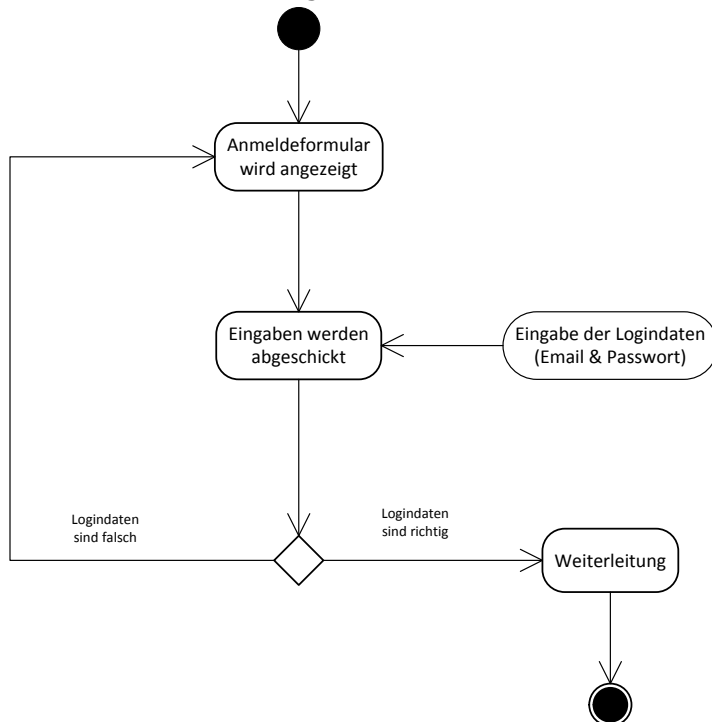
Beschreibung Anwendungsfall	
Name	Datenbank verändern
Kurzbeschreibung	Admin bearbeitet die Datenbank
Akteure	Admin
Auslöser	Admin will an der Datenbank Änderungen vornehmen
Ergebnis(se)	Admin hat die Datenbank abgeändert
Eingehende Daten	Datenbank Abänderungen
Vorbedingungen	Person ist berechtigt dafür
Nachbedingungen	-
Essentielle Schritte	<ul style="list-style-type: none"> Admin ändert die Datenbank von aussen ab
Offene Punkte	-
Änderungshistorie (Log)	-
Sonstiges, Anmerkungen	-

2.1.2 Aktivitätsdiagramme

Für das aufzeigen der Aktivitäten haben wir uns für die vier wichtigsten Anwendungsfälle entschieden. Diese haben wir dann genauer angeschaut und modelliert.

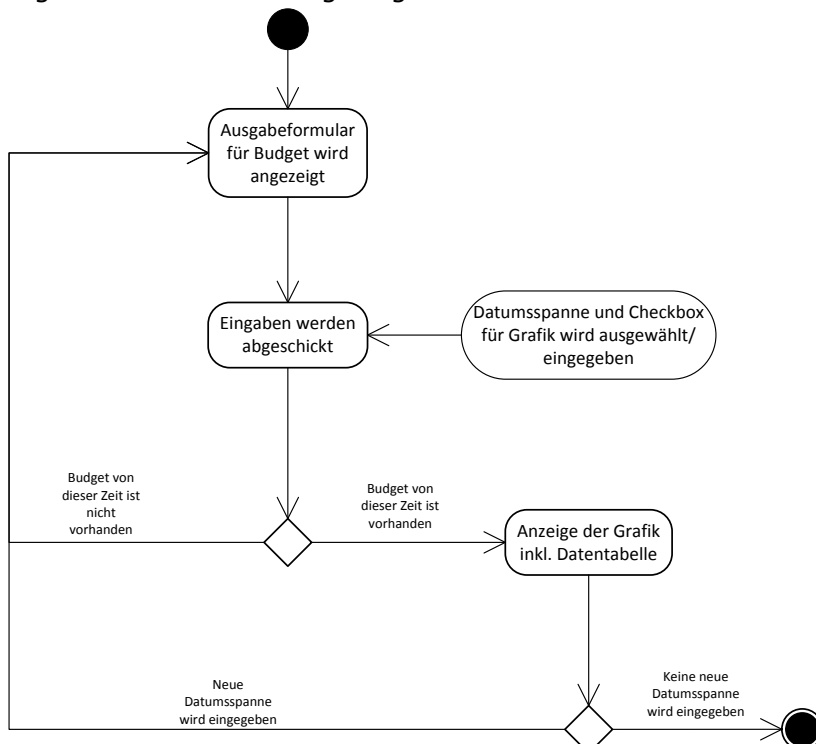
2.1.2.1 Anmelden

Bei diesem Aktivitätsdiagramm wird der Ablauf vom anmelden eines Benutzers aufgezeigt.



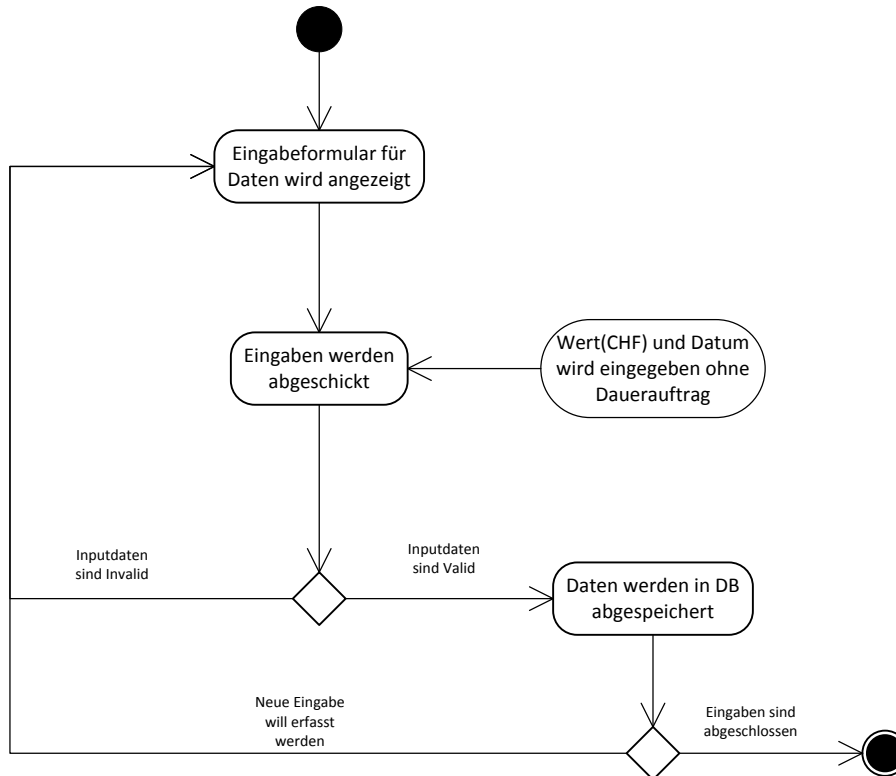
2.1.2.2 Budget-Grafik erzeugen

Bei diesem Diagramm wird der Ablauf aufgezeigt, welcher ein Benutzer macht, wenn er sein Budget als eine Grafik angezeigt haben will.



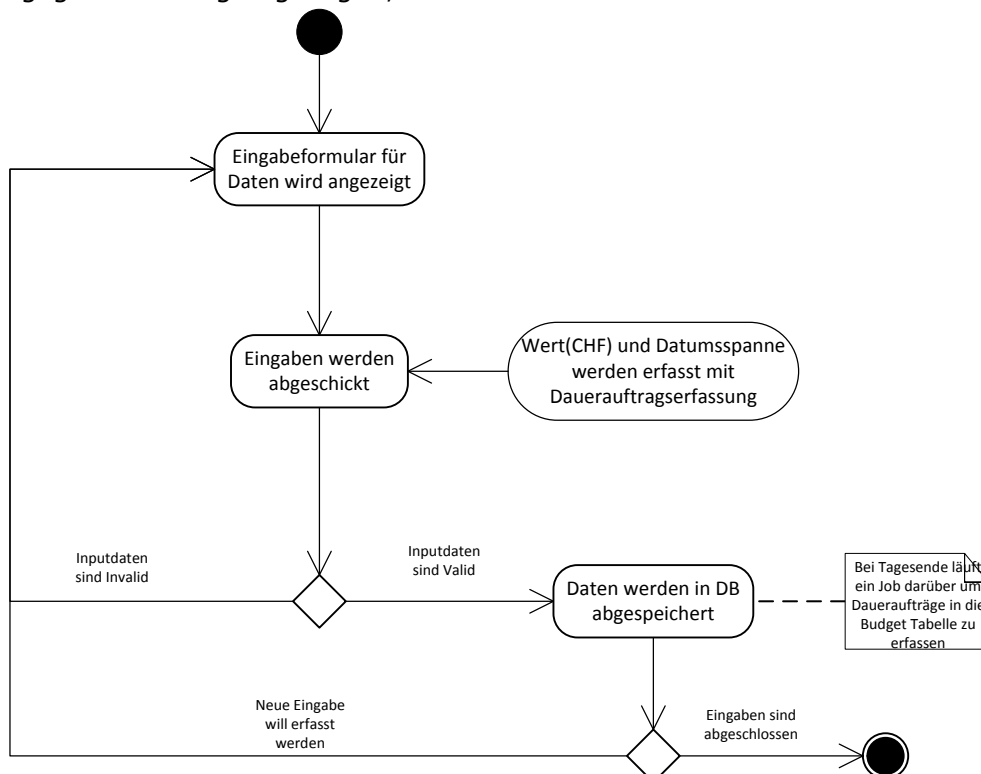
2.1.2.3 Ausgabe erfassen

Bei dieser Aktivität wird vom Benutzer bestimmte Daten eingegeben, welche danach in der Datenbank abgespeichert werden. Und schlussendlich am Budget abgezogen.



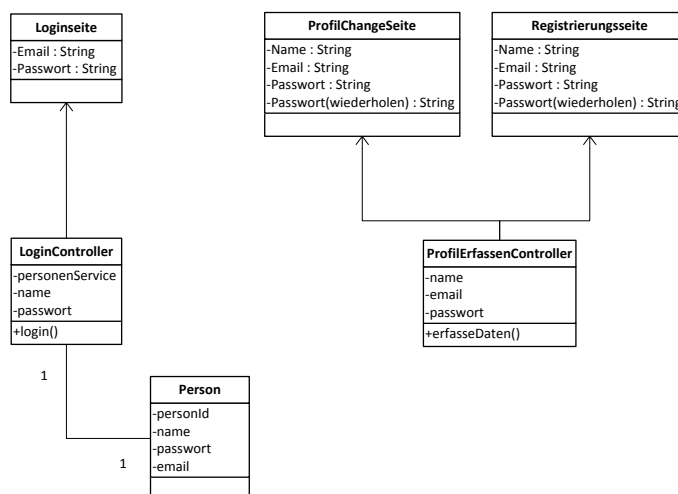
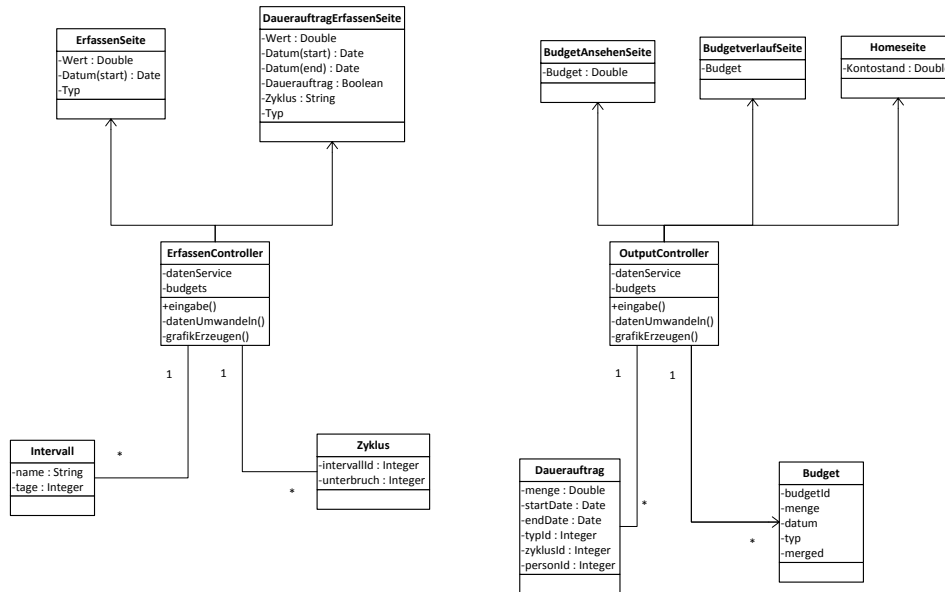
2.1.2.4 Dauerauftrag erfassen

Diese Aktivität ist fast genau gleich wie die vorherige, ausser es wird jeweils nicht nur einmal der eingegeben Betrag abgezogen, sondern soviel mal wie es der Benutzer eingibt.


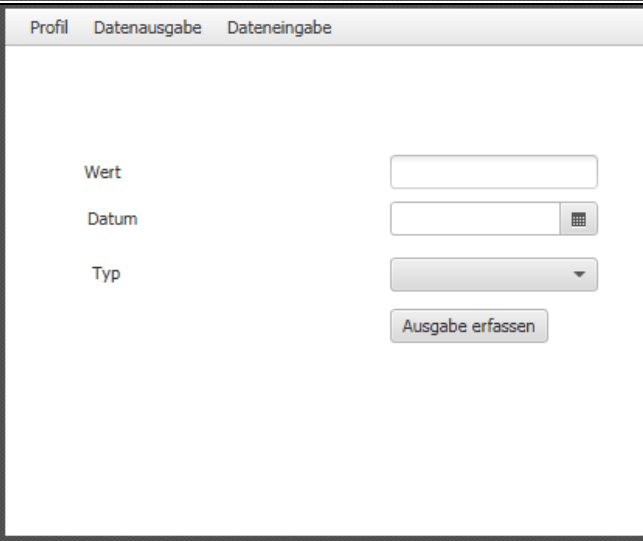

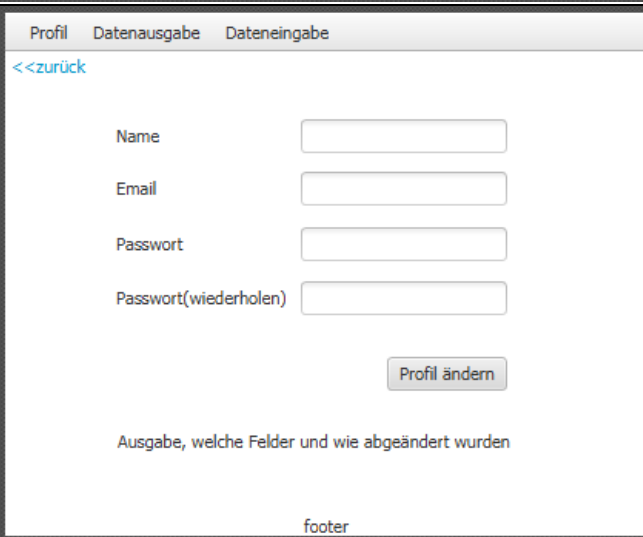


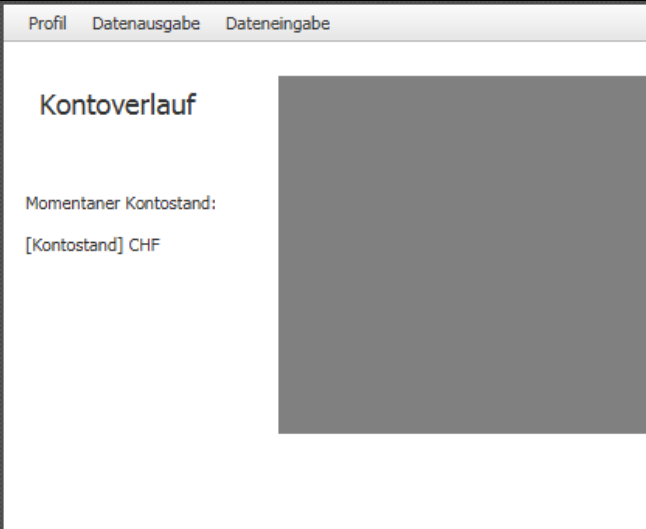
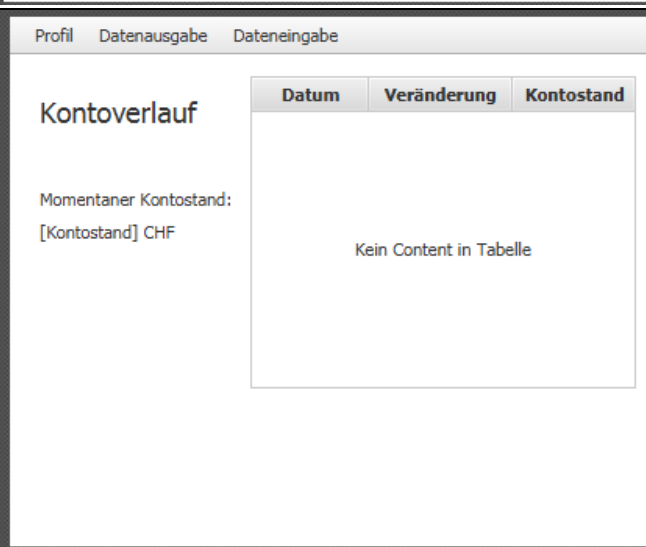

2.1.3 Klassendiagramme

Beim Klassendiagramm haben wir den Aufbau einfach und übersichtlich gehalten. D.h. wir haben möglichst wenige Methoden von Klassen aufgelistet, jedoch haben wir dafür auf die Beziehungen genauer geschaut, und welche Daten jeweils hin- und herfließen. Und wie der allgemeine Zusammenhang zwischen den Klassen ist.



2.2 GUIs

GUI	Überlegung
 <p>Geben Sie ihre Logindaten ein</p> <p>Email <input type="text"/></p> <p>Passwort <input type="password"/></p> <p>Passwort vergessen? <input type="button" value="login"/></p> <p><input type="button" value="registrieren"/></p> <p>footer</p>	<p>Dieses GUI dient dazu, dass sich ein neuer Benutzer einloggen kann. Diese Seite ist einfach gestaltet und falls eine Person noch keinen Account besitzt kann er diesen beim „registrieren“ Button erstellen. Oder falls er das Passwort vergessen hat kann der den Hyperlink „Passwort vergessen?“ klicken(Dieser geht jedoch nicht wegen eines fehlenden SMTP-Server).</p> <p>Wir fanden dass dies alle erfordereten Kriterien erfüllt, d.h. man kann sich anmelden, registrieren oder Passwort erinnern.</p>
 <p>Profil Datenausgabe Dateneingabe</p> <p>Wert <input type="text"/></p> <p>Datum <input type="text"/> </p> <p>Typ <input type="text"/></p> <p><input type="button" value="Ausgabe erfassen"/></p>	<p>Dieses GUI dient dazu eine neue Ausgabe zu erfassen, das Gegenstück „Einnahme erfassen“ sieht vom Aufbau genau gleich aus. Und von der Verarbeitung ist es ganz ähnlich. Hier wird der Wert also Betrag und das Datum, an welchem es abgezogen wird abgefüllt. Dazu kann man noch einen Dauerauftrag erfassen, d.h. die drei weiteren Felder müssen auch abgefüllt werden.</p> <p>Schlussendlich werden die abgefüllten Werte in der Datenbank abgefüllt zum dazugehörigen Benutzer. Wir dachten, dass wir Einnahmen und ausgaben geteilt erfassen und nicht auf der gleichen Seite, da es so auch für den Benutzer in der Menüleiste übersichtlicher ist.</p>
 <p>Profil Datenausgabe Dateneingabe</p> <p><<zurück</p> <p>Name <input type="text"/></p> <p>Email <input type="text"/></p> <p>Passwort <input type="password"/></p> <p>Passwort(wiederholen) <input type="password"/></p> <p><input type="button" value="Profil ändern"/></p> <p>Ausgabe, welche Felder und wie abgeändert wurden</p> <p>footer</p>	<p>In diesem GUI erfasst man Profiländerungen. Hierbei kann man beliebig viele Felder abfüllen, denn es wird jeweils einfach das angepasst, was nicht leer ist. Denn wenn man zuerst sagen müsste welche Felder man anpassen will, würde es aus unserer Sicht recht unübersichtlich werden. Und so kann der Benutzer noch vorzu entscheiden ob er jetzt das Feld abändern will und muss nicht wieder zurück gehen und seine Wünsche abzuändern.</p>

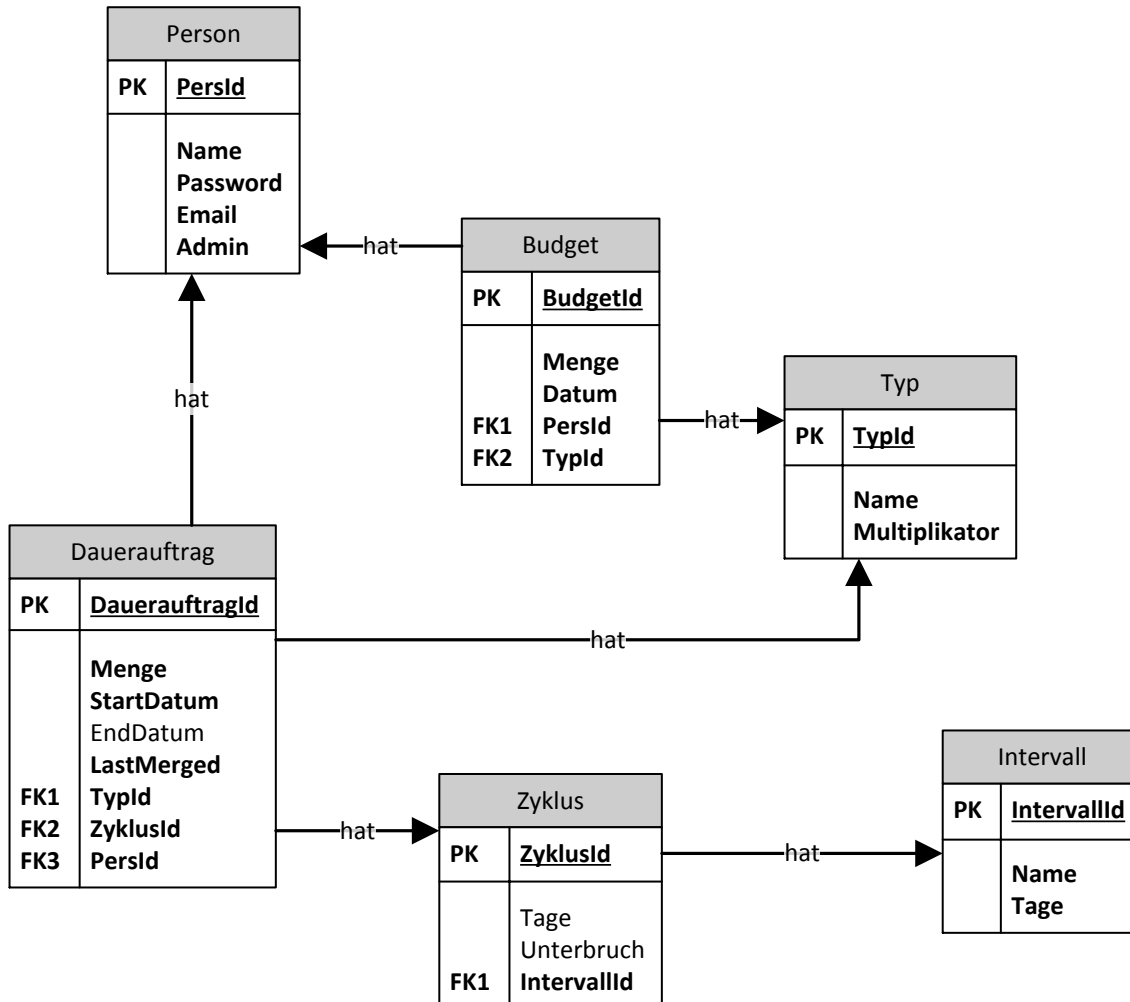
	<p>Bei dieser Oberfläche wird der Kontoverlauf mithilfe eines Charts angegeben, also das Budget des Benutzers.</p>
	<p>Hier wird der Kontoverlauf der Person mithilfe einer Tableview ausgegeben.</p>
	<p>Dies ist die Seite, welche angezeigt wird wenn sich ein Benutzer erfolgreich angemeldet hat. Hier wird auch gleich der aktuelle Kontostand der Person angezeigt.</p>

<p>Geben Sie Ihre persönlichen Daten ein</p> <p>Name* <input type="text"/></p> <p>Email* <input type="text"/></p> <p>Passwort* <input type="password"/></p> <p>Passwort(wiederholen)* <input type="password"/></p> <p><input type="button" value="registrieren"/></p>	<p>Mithilfe dieser Seite kann sich eine Person überhaupt für diese Applikation registrieren. Dazu werden die gezeigten Felder benötigt. Wenn sich eine Person erfolgreich registriert hat, kann sie sich als Benutzer im Anschluss anmelden.</p>
---	--

2.3 Datenbank

Wir haben uns für eine „XAMPP mySQL“ Datenbank entschieden, da diese gerade von der Schule zur Verfügung gestellt wurde und wir diese auch einfach zuhause benutzen können.

2.3.1 Datenbankmodell



2.3.2 Datenbank SQL

```
create database BUDGET_DB;
use BUDGET_DB;
```

```
create table PERSON (
    PERSON_ID Integer AUTO_INCREMENT PRIMARY KEY,
    NAME VARCHAR(100) NOT NULL,
    PASSWORT VARCHAR(100) NOT NULL,
    EMAIL VARCHAR(100) NOT NULL,
    IS_ADMIN BIT NOT NULL
);
```

```
create table TYP (
    TYP_ID Integer AUTO_INCREMENT PRIMARY KEY,
    NAME VARCHAR(100) NOT NULL,
    MULTIPLIKATOR Integer NOT NULL
);
```

```
create table INTERVALL (
    DOKUMENTATION.doc / 15.07.2016
    bbw Linus Eberhard & Sven Frei
```



```
INTERVALL_ID Integer AUTO_INCREMENT PRIMARY KEY,  
NAME VARCHAR(100) NOT NULL,  
TAGE Integer NOT NULL  
);  
  
create table ZYKLUS (  
    ZYKLUS_ID Integer AUTO_INCREMENT PRIMARY KEY,  
    TAGE Bit(7),  
    UNTERBRUCH Integer,  
    INTERVALL_ID Integer NOT NULL,  
    FOREIGN KEY (INTERVALL_ID) REFERENCES INTERVALL(INTERVALL_ID)  
);  
  
create table DAUERAUFTRAG (  
    DAUERAUFTRAG_ID Integer AUTO_INCREMENT PRIMARY KEY,  
    MENGE DOUBLE NOT NULL,  
    START_DATUM DATE NOT NULL,  
    END_DATUM DATE,  
    TYP_ID Integer NOT NULL,  
    ZYKLUS_ID Integer NOT NULL,  
    PERSON_ID Integer NOT NULL,  
    FOREIGN KEY (TYP_ID) REFERENCES TYP(TYP_ID),  
    FOREIGN KEY (ZYKLUS_ID) REFERENCES ZYKLUS(ZYKLUS_ID),  
    FOREIGN KEY (PERSON_ID) REFERENCES PERSON(PERSON_ID)  
);  
  
create table BUDGET (  
    BUDGET_ID Integer AUTO_INCREMENT PRIMARY KEY,  
    MENGE DOUBLE NOT NULL,  
    DATUM DATE NOT NULL,  
    TYP_ID Integer NOT NULL,  
    PERSON_ID Integer NOT NULL,  
    FOREIGN KEY (TYP_ID) REFERENCES TYP(TYP_ID),  
    FOREIGN KEY (PERSON_ID) REFERENCES PERSON(PERSON_ID)  
);
```

2.3.3 Allgemeiner Aufbau

Die Tabelle wurde nach einem logisch relationalen Modell erstellt. Sie dient dazu die registrierten Benutzer, deren Budget und Ein- bzw. Ausgaben geordnet abzulegen. Diese Daten werden zum Grossteil bei den implementierten Funktionen benötigt, um den gewünschten Output zu kriegen. Da die Funktionen oft sehr spezifische Werte benötigen, wie z.B. den Abstand zwischen den Daueraufträgen, haben wir die einzelnen Tabellen stark auf mehrere aufgesplittet. Somit haben wir garantiert, dass keine integritäts Fehler auftreten. Insgesamt haben wir 6 Tabellen, wobei alle von der Tabelle "Person" abhängt. Was Sinn macht, weil ohne Person wird es kein Budget etc. geben. Und von dieser Tabelle gehen dann die weiteren Tabellen aus und so werden dann auch Einträge aus den Funktionen mit den Beziehungen zwischen einander abgefüllt.

2.3.4 Tabellen

2.3.4.1 Person

Attribute:

- PersId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Name → In diesem Feld wird der Name des Benutzers stehen, dies in einem String.
- Email → Die Email-Adresse von der Person wird in diesem Feld als String abgespeichert.
- Admin → In diesem Boolean wird abgespeichert ob die Person ein Admin ist, also '1' oder ist ein normaler Benutzer '0'.

Beziehungen:

Die Person steht in beziehung mit dem Budget, von welchem es beliebig viele haben kann, wobei ein Budget immer einer Person zugeordnet sein muss($P-(1)----- (0...n)-B$).

Und eine weitere Beziehung wird von der Person aus beeinflusst, denn eine Person kann beliebig viele Daueraufträge anlegen, jedoch muss ein Dauerauftrag immer einer Person zugeordnet sein($P-(1)----- (0...n)-D$).

2.3.4.2 Budget

Attribute:

- BudgetId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Menge → Der Wert welcher hinzugefügt/abgezogen wird, ist in diesem Attribut abgelegt, als Double.
- Datum → In diesem Feld wird das Datum eingetragen, an welchem der Betrag abgezogen/hinzugefügt wurde, dies ist natürlich als Datum abgelegt.
- PersId → Diesem Feld wird die PersId von der dazugehörigen Person, dies wird in einer Ganzzahl abgespeichert.
- TypId → Diesem Feld wird die TypId von dem dazugehörigen Typen, welcher dieser Wert sein sollte, dies wird in einer Ganzzahl abgespeichert.

Beziehungen:

Das Budget muss einer Person zugeordnet sein, wobei eine Person beliebig viele Budget anlegen kann($P-(1)----- (0...n)-B$).

Ein Budget muss immer einen Typ haben, also Addition oder Subtraktion, und der Typ kann beliebig vielen Budget zugeordnet sein($B-(0...n)----- (1)-T$).

2.3.4.3 Typ

Attribute:

- TypId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Name → Dieses Attribut ist dafür da, dass man von aussen auch identifizieren kann was damit gemeint ist, dieser Wert ist als String konstant abgelegt.
- Multiplikator → Durch dieses Feld findet man in den Funktionen im Programm heraus ob der Wert vom Budget addiert oder subtrahiert wird, dies ist als Char abgespeichert.

Beziehungen:

Ein Typ kann beliebig vielen Budget zugeordnet sein, aber ein Budget muss immer auf einen Typ referenzieren($B-(0...n)----- (1)-T$).

Genauso kann ein Typ beliebig vielen Daueraufträgen zugewiesen sein, jedoch muss ein Dauerauftrag immer einem Typ zugeordnet sein($D-(0...n)----- (1)-T$).

2.3.4.4 Dauerauftrag

Attribute:

- DauerauftragId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Menge → Der Wert welcher hinzugefügt/abgezogen wird, ist in diesem Attribut abgelegt, als Double.
- StartDatum → In diesem Element wird abgelegt zu welchem Zeitpunkt das erste Mal der Wert abgezogen/hinzugefügt wird, dies wird als Date abgelegt.
- EndDatum → Dieses Feld ist als Gegenstück zum StartDatum abgelegt, auch als Date.
- LastMerged → In diesem Attribut wird das Datum gespeichert, wann das letzte Mal der Datensatz abgeändert wurde, dies wird als Date abgespeichert.
- TypId → Diesem Feld wird die TypId von dem dazugehörigen Typen, welcher dieser Wert sein sollte, dies wird in einer Ganzzahl abgespeichert.
- ZyklusId → Diesem Feld wird die ZyklusId von dem dazugehörigen Zyklus mit welchem der Dauerauftrag abgezogen/hinzugefügt wird, dies wird in einer Ganzzahl abgespeichert.
- PersId → Diesem Feld wird die PersId von der dazugehörigen Person, dies wird in einer Ganzzahl abgespeichert.

Beziehungen:

Ein Dauerauftrag muss immer einer Person zugewiesen sein und eine Person kann zu beliebig vielen Daueraufträgen referenzieren($P-(1)----- (0...n)-D$).

Der Dauerauftrag muss auch immer einem Typ zugeordnet sein, aber ein Typ kann beliebig vielen Daueraufträgen zugewiesen sein($D-(0...n)----- (1)-T$).

Ein Dauerauftrag muss sicherlich zu einem Zyklus referenzieren, jedoch kann ein Zyklus beliebig viele Assoziationen zur Dauerauftragstabelle haben($D-(1)----- (0...n)-Z$).

2.3.4.5 Zyklus

Attribute:

- ZyklusId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Tage → Dieses Feld beinhaltet die Anzahl der Tage, welche zum nächsten Abzug/Hinzufügen benötigt werden, dies ist als Ganzzahl abgespeichert.
- Unterbruch → Dieser Wert ist fakultativ, er bedeutet ob es nach einer gewissen Anzahl von Tagen einen Unterbruch geben sollte, dies ist auch als Ganzzahl abgelegt.
- IntervallId → Diesem Feld wird die IntervallId von dem dazugehörigen Intervall, von welchem dieser Wert sein sollte, dies wird in einer Ganzzahl abgespeichert.

Beziehungen:

Ein Zyklus kann an beliebig vielen Daueraufträgen zugeordnet sein, jedoch muss ein Dauerauftrag immer einem Zyklus zugewiesen sein($D-(1)----- (0...n)-Z$).

Ein Zyklus ist immer ist immer genau einem Intervall zugewiesen, ähnlich umgekehrt nur ein Intervall kann einem Zyklus zugewiesen sein muss aber nicht($Z-(1)----- (0...1)-I$).

2.3.4.6 Intervall

Attribute:

- IntervallId → Dieses Attribut dient als Primärschlüssel und ist als Ganzzahl konfiguriert.
- Name → Hier wird der Name abgelegt, welcher in den Funktionen als Referenz dient, dieser wird als String abgelegt.
- Tage → Dieser Wert wird speziell abgelegt, nämlich als Bit(7), woraus man die direkte Anzahl der Tage heraus lesen kann.

Beziehungen:

Ein Intervall kann einem Zyklus zugewiesen sein, wobei ein Zyklus immer zu einem Intervall referenzieren muss ($Z - (1) \text{-----} (0 \dots 1) - I$).

3 Softwaredokumentation

Bei unserer Applikation gab es viele verschiedene Abläufe, welche sehr wichtig für das Endprodukt waren. Diese werden in den folgenden Abschnitten erläutert. Und es werden besondere Teile hervorgehoben. Es werden aber auch allgemeine Programmstellen beschrieben und wie der ganze Zusammenhang zwischen den einzelnen Komponenten ist.

3.1 Beschreibung

Wir haben ein Programm realisiert, welches als Buchungssystem dient. Dieses nimmt Einnahmen oder Ausgaben entgegen und berechnet somit das momentane Budget. Das Programm macht nicht den Anschein, dass es sehr komplex ist, jedoch wird im Hintergrund vor allem viel berechnet was viel von uns abverlangt.

3.1.1 Controller

In den Controllern wurde die ganze Arbeit, welches unsere Applikation macht implementiert. Sie berechnen im Endeffekt das Budget und speichern neue Daten in der Datenbank ab.

3.1.1.1 ErfassenController

Dieser Controller wird dafür benötigt, eine Ausgaber resp. eine Einnahme von dem dafür vorgesehenen View einzulesen und zu bearbeiten. Für diesen Schritt wird vom View einen Double Wert ausgelesen, welcher vom Budget abgezogen/hinzugefügt wird. Dann das Datum, an welchem der Betrag bezogen wird, und noch versteckt ob der Wert addiert oder subtrahiert wird. Dies passiert bei einer einfachen Erfassung.

Bei einer Erfassung eines Dauerauftrages werden zusätzliche Werte benötigt, nämlich noch das Datum, an welchem der Dauerauftrag wieder beendet wird. Und in welchem Zyklus sich diese Manipulation des Budget wiederholt. Dies wird mit einer Checkbox übergeben ob es ein Dauerauftrag ist. Wenn diese Option ausgewählt ist müssen auch noch die Objekte Intervall und Zyklus erzeugt werden, welche für die Implementierung in die Datenbank benutzt werden. Da genau diese beiden Objekte als Tabellen darin vorkommen.

3.1.1.2 OutputController

Der OutputController versorgt die drei Output Views, welche Daten eines Benutzers übersichtlich darstellen. Dieser Controller bezieht seine Daten aus der Datenbank, aus den Tabellen Dauerauftrag und Budget, genau diese Objekte werden auch erzeugt in ArrayList. Da höchstwahrscheinlich mehrere Einnahmen und oder Ausgaben getätigt wurden von einem Benutzer. Bei der Ausgabe der Daten wird noch darauf geachtet, welcher View beliefert wird. Dies mithilfe der Hauptklasse worin steht, auf welchem View man sich im Moment gerade bewegt. Wenn dieser Schritt gemacht wurde müssen die Daten nur noch mit den richtigen Methoden umgewandelt werden, damit sie bereit für die Ausgabe auf dem vorgegebenen View sind.

3.1.1.3 LoginController

Der Name erklärt eigentlich schon die Funktion des Controllers. Denn er überprüft, ob die Eingaben zum Anmelden genauso in der Datenbank vorkommen, wenn nicht wird dies als solches mit einer Meldung gekennzeichnet. Die Daten werden wie vorhin schon genannt aus der Datenbank geholt, genauer aus der Tabelle Person. Darin befinden sich unter anderem die Logindaten und auch der Name, welcher beim OutputController benötigt wird für die Homeseite. Für diese Abfrage wird das Objekt Person benötigt, um gerade die bezogenen Daten aus der Datenbank darin abzufüllen.

3.1.1.4 ProfilErfassenController

Dieser Controller wird einerseits zum erfassen eines neuen Benutzers benötigt, aber auch um einen bereits vorhandenen Benutzer abändern zu können. Änderungen darf der Benutzer selbständig tätigen. Bei beiden Aktionen wird in der Tabelle Person in der Datenbank ein Datensatz abgeändert/hinzugefügt. Beim abändern kann man in der View so viele Felder abfüllen wie man will, halt diese welche man abändern will, denn danach werden auch nur diese angepasst.

Falls man sich neu registriert muss man sich nach einer erfolgreichen Registrierung noch anmelden und wird nicht direkt auf die Homeseite weitergeleitet. Und beim anpassen wird jeweils nur das aktuell benutzte Profil abgeändert und dies kann man nicht manuell abändern.

3.1.2 Models

Die Models sind automatisiert aus der Datenbankdefinitionen heraus generiert worden. Benötigt werden jedoch nicht ganz alle Models, sondern alle ausser das Typ Objekt. Bei jedem Model ist jeweils ein Default Konstruktor, einen Definitions Konstruktor, die Getter und die Setter generiert worden. Somit kann die Applikation auf die benötigten Feldern in den dafür vorgesehenen Situationen zugreifen.

Der Generator für diese Models wurde von unserer Gruppe her entwickelt und liegt somit auch dem restlichen Projekt bei.

3.1.3 Views

Die Models wurden aus dem Klassendiagramm heraus mit FXML implementiert.

3.1.3.1 AusgabeErfassenSeite & EinnahmeErfassenSeite

Diese zwei Views sind mit dem ErfassenController verknüpft. Man gibt in einer der beiden die gewünschten Werte ein und bestätigt diese Eingabe, danach werden sie mit dem Controller in der Datenbank abgelegt.

Felder:

- Wert → In diesem Feld wird der Wert eingegeben, welcher abgezogen wird resp. hinzugefügt.
- Datum(start) → Das Datum an welchem der Wert übertragen wird, oder das Startdatum wenn es ein Dauerauftrag ist.
- Datum(end) → Das Gegenstück zum Startdatum, jedoch wird das Enddatum nur bei einem Dauerauftrag benötigt.
- Dauerauftrag → Hier wird entschieden ob es ein Dauerauftrag ist oder doch nicht mit einer Checkbox.
- Zyklus → Dieser Wert wird via Dropdown ausgewählt mit Daten aus der Datenbank aus der Tabelle Zyklus.

3.1.3.2 BudgetAnsehenSeite, BudgetverlaufSeite & Homeseite

Bei diesen drei Views ist der OutputController zugewiesen. Denn hier werden die Daten bloss auf die Views übergeben und werden dann dort angezeigt. Beim BudgetAnsehenSeite werden diese via Chart angezeigt für den aktuellen Monat. Bei der BudgetverlaufSeite wird ein Tableview angezeigt, in welchem sich alle bisherigen Bewegungen befinden. Und auf der Homeseite wird einen kleinen Begrüssungstext und der aktuelle Kontostand angezeigt.

Felder(BudgetAnsehenSeite):

- Budget → Das Budget wird einerseits als Zahl angezeigt.
- Chart → Hier wird noch als Chart angezeigt, wieviel das Budget beträgt.

Felder(BudgetverlaufSeite):

- Tableview → In diesem Table werden alle bisherigen Bewegungen des Konto des Benutzers aufgelistet.

Felder(Homeseite):

- Budget → Das Budget wird als Zahl angezeigt.
- Willkommenstext → Ein kurzer Begrüssungstext mit dem Namen des Benutzers.

3.1.3.3 Loginseite

Dieser View wird für die Anmeldung benützt. D.h. es werden die nötigen Daten für das Login abgefüllt und danach mit dem LoginController überprüft.

Felder:

- Email → Die Emailadresse, welche dem Benutzer persönlich gehört.
- Passwort → Das dazugehörige Passwort, welches zur Emailadresse des Benutzers gehört.

3.1.3.4 ProfilChangeSeite & Registrierungsseite

Diesen beiden Views ist der ProfilErfassenController zugeordnet. Es werden jeweils alle in der Personentabelle vorhandenen Daten abgefüllt im Registrierungsseite-View im ProfilChangeSeite-View können beliebig viele dieser Felder abgefüllt werden. Wenn die eingegebenen Daten erfolgreich bestätigt wurden, werden sie via Controller in der Datenbank hinzugefügt.

Felder:

- Name → Der Name des Benutzers wird hier abgefüllt.
- Email → In diesem Feld wird die Emailadresse eingefügt, welche auch zum anmeldne benötigt wird.
- Passwort → Dieses Feld beinhaltet schlussendlich das gewünschte Passwort, dieses Feld wird zwei mal abgefragt, um zu überprüfen, dass sich der Benutzer nicht vertippt hat.

3.1.4 Schnittstellen

Die Schnittstelle, welche wir für den Datenaustausch zwischen der Verarbeitung und des Benutzers einsetzten war FXML mit seiner GUI Oberfläche. Und der Verarbeitung mit Views – Controller – Models. Welche uns zusprach. Denn so konnten wir auf der vorgegebenen Javabasis unser Projekt nach unseren Vorstellungen recht einfach implementieren.

Die Schnittstelle zwischen der Verarbeitung und den Daten, also der Datenbank war etwas anders als der bereits kennengelernte Teil. Denn für diesen Teil haben wir uns für einen selbst entwickelten Zugriff entschieden, welcher aber bei Punkt 2.6.1 beschrieben wird.

3.2 Patterns, Konzepte, Modelle, etc.

Beim Aufbau der Applikation mussten wir uns zuerst auf das Umfeld, welches wir benutzen wollten, festlegen. Dabei haben wir uns verschiedene Konzepte überlegt. In manchen kamen Webserver und SMTP Server vor, welche wir jedoch nicht mit der nötigen Infrastruktur und dem Vorwissen realisieren konnten. Darum blieben wir bei einem eher einfacheren Konzept, welches lediglich die Datenbank von XAMPP beinhaltete und die Java-Umgebung. Danach legten wir fest, welche Art von Buchungssystem wir entwerfen wollen, dies war schlussendlich ein Budget-Buchungssystem für Personen. Und dass diese Daten der Personen in einer Datenbank zentral abgelegt sind.

Als wir diese Sachen festlegten widmeten wir uns den verschiedenen Modellen, welche immer spezifischer auf das endgültige Programm eingingen. Diese Modelle werden weiter oben beschrieben. Jedoch gab es da auch wieder kleinere Veränderungen des gesamten Konzept, da wir selbst wieder neue Möglichkeiten oder Einschränkungen bemerkten. Doch schlussendlich haben wir die Situation bestmöglich genutzt und dies auch so mit dem Programm umgesetzt.

3.2.1 Datenbankzugriffs Framework

Input: Mapping, ResultSet

Output: Objekte(Liste)

Der von uns selbst entworfene Datenbankzugriff ist nicht wirklich ein Zugriff auf die Datenbank, sondern ein Zugriff auf die Daten, welche die Datenbank via ResultSet zurückliefert. D.h. man gibt dem Framework das bekommene Resultset mit und, welche Daten man daraus lesen will also das gewünschte Mapping. Danach bekommt man als Output die Objekte oder nur ein Projekt, welches/r von einer Tabelle in der Datenbank kommt. Beim einloggen, will man z.B. alle Benutzer. Bei diesem Aufruf wird eine `List<Person>` zurückgegeben. Es kommt immer darauf an, welche Tabelle man auslesen will. Der weitere Schritt, die Daten auslesen ist ganz normal, man wird die Daten in der Liste finden zur Tabelle. Dabei kommt das mitgegebene Mapping zum Zug. Denn mit diesem wird gesagt, welche Daten schlussendlich in der Liste stehen. Dieses Mapping beinhaltet spezielle Tags, welche die Namen der Attribute der Tabelle haben und so wird dann auch erkannt, welche Daten abgefüllt werden müssen.

Als Voraussetzung, dass man dieses Framework benutzen kann ist natürlich eine bestehende Verbindung zu einer Datenbank. Oder zumindest ein ResultSet, welcher die nötigen Daten enthält. Also die Attribute, welche im Framework definiert sind. Ansonsten können keine Werte aus dem ResultSet ausgelesen werden und es entstehen keine gültigen Daten, welche man für die Verarbeitung benutzen kann.

3.3 Schwierigkeiten und kritische Stellen

Für uns war vorallem der Anfang schwierig um dort ein Projekt zu finden, welches uns beide interessiert und in welches wir gerne Zeit investieren würden. Doch nach einer Zeit konnten wir uns auf dieses Projekt als Kompromiss einigen und wir finden auch, dass wir eine gute Wahl machten. Denn wir konnten viel dazu lernen und unsere Aufgabe auf unsere Stärken verteilen.

Eine nächste anspruchsvolle Stelle war, ob wir einen eigenen Datenbankzugriff kreieren oder ob wir den bereits vorgegebenen nicht so schönen Zugriff nehmen. Dabei waren wir uns uneinig, jedoch merkten wir am Schluss dass es eigentlich eine bessere Entscheidung ist, wenn wir einen eigenen kreieren. Denn diesen können wir auch unserem Projekt zulegen.

Wir hatten in der Mitte des Projektes noch oft Probleme beim berechnen der Daten in den Controllern, was uns viel Zeit kostete und unseren Zeitplan etwas durcheinander brachte. Darum mussten wir mehr als gedacht zuhause machen, was eigentlich nach unserer Planung nicht so vorgesehen war. Aber schlussendlich hat sich der Aufwand sicherlich gelohnt und wir haben ein anständiges Produkt als Projekt.

Eigentlich das grösste Problem bei der ganzen Sache war, dass wir nicht so einfach zusammenarbeiten konnten, ausser wir würden jeweils die einzelnen Dokumente oder Teil des Projektes per Email oder Dropbox am anderen übergeben. An dieser Stelle mussten wir uns eine nützlich alternative suchen, welche wir beide benutzen können. Und diese haben wir aus unserer Sicht auch gefunden mit dem GitHub, aber zu dem noch bei einem späteren Punkt.

Ansonsten hatten wir so gut wie keine Probleme nur ein paar Unstimmigkeiten verteilt über das Projekt, welche aber nach Diskussionen bereinigt werden konnten.

3.4 Gewählte Testfälle, Testkonzept

Bei den Testfällen haben wir uns dafür entschieden, dass wir nicht für alle Controller Tests vorbereiten, jedoch für diese, welche Input erhalten haben wir die nötigen Test durchgeführt. Damit haben wir den ErfassenController, LoginController, ProfilErfassenController getestet.

Testfälle ErfassenController:

Testfall	Input	Erwarteter Output	Funktion
A	Einfache Eingabe korrekt	Daten werden abgefüllt	eingabe()
B	Dauerauftrag Eingabe korrekt	Daten werden abgefüllt	eingabe()
C	Wert ist nicht abgefüllt	Fehlermeldung	eingabe()
D	Datum(start) nicht abgefüllt	Fehlermeldung	eingabe()
E	Datum(end) nicht abgefüllt	Fehlermeldung	eingabe()
F	Zyklus nicht abgefüllt	Fehlermeldung	eingabe()

Testfälle LoginController:

Testfall	Input	Erwarteter Output	Funktion
A	Logindaten korrekt	Wird eingeloggt	login()
B	Falsche Email	Fehlermeldung	login()
C	Falsches Passwort	Fehlermeldung	login()

Testfälle ProfilErfassenController:

Testfall	Input	Erwarteter Output	Funktion
A	Profildaten sind korrekt	Daten werden abgefüllt	erfasseDaten()
B	Name nicht abgefüllt	Fehlermeldung	erfasseDaten()
C	Email nicht abgefüllt	Fehlermeldung	erfasseDaten()
D	Passwort(wiederholen) ist unterschiedlich	Fehlermeldung	erfasseDaten()

4 Arbeitskonzept

4.1 Aufteilung und Zusammenarbeit

Die Aufteilung der Aufgaben verlief grösstenteils reibungslos, da wir uns bereits bei der Planung Gedanken über die Stärken von uns machten. Daher wussten wir auch, welche Person etwa auf welchem Gebiet entwickelt. Sven konzentrierte sich mehr auf den theoretischen Teil und Linus auf die Implementation, aber es war für beide Parteien eine ausgeglichene Arbeit, d.h. es war nicht so, dass jemand nur Modelle zeichnete und dokumentierte während der andere programmierte. Sondern es kam jeweils Abwechslung ins Spiel und es war auch beiden jeweils nach einer nicht eintönigen Arbeit.

Wie schon erwähnt hatten wir zu Beginn Probleme mit dem Austausch der Daten. Dies konnten wir mit GitHub lösen, welches bereits kompatibel für Eclipse ist. Was uns sehr entgegenkam. Da wir keine andere so produktive Lösung in Sicht hatten. Denn so konnten wir uns einfacher auf die Arbeit konzentrieren, als auf den Datenaustausch untereinander.

Auch die zwischenmenschliche Zusammenarbeit empfanden wir beide als angenehm, und wir konnten gegenseitig vom Wissen des anderen profitieren. Es war auch jeweils einfach bei Fragen zu Problemen oder zur Weiterarbeit sich mit der anderen Person zu verständigen, da diese mit dem gleichen Engagement beim Projekt dabei war.

5 Reflexion

5.1 Linus Eberhard

Ich hatte grossen Spass bei diesem Projekt, es gefiel mir auch eine Arbeit zu zweit zu machen. Weil man sich so gegenseitig motivieren kann oder Besprechungen über wichtige Programmstellen zu machen. So kommt immer etwas Schwung in die ganze Arbeit und so wird es erst richtig Spannend zusammen zu arbeiten. Ich konnte auch selbst noch Sachen dazu lernen, dies vorallem auf der Theoretischen Seite.

Alles in allem bin ich zufrieden mit dem Projekt sowie mit meiner Partnerwahl, welche für die Zusammenarbeit angenehm war.

5.2 Sven Frei

Auch ich empfand das Projekt als gelungen und auch die Zusammenarbeit mit Linus war wirklich angenehm, auch wenn manchmal grössere Debatten entstanden zu Problemen. Diese wollte jeder auf seine Art lösen, was oftmals in einem Kompromiss endete. Dies fand ich aber nicht schlimm, im Gegenteil ich fand es aufregen die Meinung einer anderen Person zu einem Projekt, in welchem nur wir zwei Ahnung hatten was am Ende heraus kommt, zu erfahren.

Aber vor allem gefiel mir auch, dass es so gut klappte, dass wir uns grösstenteils an den Zeitplan halten konnten und so auch nicht in einen zeitlichen Stress kamen. Auch wenn ich selten genervt war, warum dieser Teil des Programms schon wieder nicht funktionierte, obwohl der Endtermin immer näher rückte. Ich denke wir hätten uns auch in einigen Punkten etwas besser verhalten können, denn wir nahmen aus meiner Sicht das ganze Projekt anfangs eher locker. Da wir es nicht genau einschätzen konnten. Was manchmal dazu führte, dass wir glaubten weiter zu sein als wir eigentlich wirklich waren. Jedoch hat es gegen Ende geklappt und das ist neben der Erfahrung, welche wir gesammelt haben die Hauptsache.

5.3 Schlusswort

Wir beide kamen zum Entschluss, dass wir grossen Spass bei dieser Arbeit hatten und wir konnten jeweils vom Wissen des anderen profitieren. Was für uns sehr wichtig war, obwohl wir schon einiges wussten auf vielen Gebieten gab es immer etwas was neu war und wir nicht so genau einschätzen konnten. Wie zum Beispiel den selber entwickelten Zugriff auf die Datenbank worauf wir besonders stolz sind. Es war auch einmal etwas Anderes als die Projekte, welche wir bisher gemacht hatten, denn diese waren alle eher einzelarbeit und man konnte so nicht so profitieren von dem Wissen einer anderen Person. Oder sich die Arbeit so aufteilen, dass man genau seine Stärken einsetzen kann.

Im Allgemeinen sind wir mit unserem Endprodukt und mit der Erfahrung, welche wir in verschiedenen Gebieten sammeln konnten sehr zufrieden und wir würden gerne ein weiteres Mal zusammen an einem Projekt arbeiten.

6 Beilagen

- 01_Anwendungsfälle.vsd
- 02_BeschreibungAnwendungsfall.docx
- 03_Aktivitätsdiagramm.vsd
- 04_Klassendiagramm.vsd
- 05_budget.sql
- 06_Datenbankmodell.vsd
- 07_Spezifikationen.docx
- 08_Zeitplan.doc
- 09_Präsentation.ppt