Dokumentation Projekt „Onlinebanking“ – Prognose

1. Inhaltsverzeichnis

2. Motivation und Zielsetzung

Jeder Studierende weiß, dass zu Beginn jedes Semesters die Semestergebühren anfallen und diese meist schon vor Vorlesungsbeginn vom eigenen Konto per Lastschriftverfahren abgebucht werden. Zusätzlich wird darauf hingewiesen, sicherzustellen, dass zum Abbuchzeitpunkt genügend Guthaben auf dem Konto verfügbar ist. Wenn aber über den Monat hinweg mehrere Transaktionen (z.B. Beitrag fürs Fitnessstudio oder Kosten eines Abonnements für Streaming-Dienste) über das Konto getätigt werden, kann man schnell den Überblick über sein Guthaben zum Zeitpunkt „X“ verlieren. Darum habe ich mir es zur Aufgabe gemacht in einem Onlinebanking-Portal das Kontoguthaben zum Zeitpunkt „X“ vorherzusagen (zu prognostizieren). Das Ziel der Prognose ist es also zu jedem möglichen Zeitpunkt „X“ in der Zukunft den Kontostand bestimmen zu können. Dies gelingt durch das Erfassen von – in gleichen Intervallen – wiederkehrenden Transaktionen, deren Betrag dann entsprechend den Intervallen dem Guthaben gutgeschrieben oder abgezogen wird.

3. Einrichtung

Die Einrichtung / Installation der Prognosekomponente erweist sich als sehr einfach, da lediglich die Dateien (*.php, *.js) auf den Webserver geladen werden müssen. Dies erreicht man am bequemsten mit der Nutzung eines beliebigen FTP-Clients (Voraussetzung: FTP-Server installiert) oder per Remotedesktopverbindung. Zusätzlich muss eine Tabelle `konto` erweitert und eine Tabelle `umsaetze` in der Datenbank angelegt werden müssen. Die Tabelle `umsaetze` kann man einfach per (*.sql)-File importieren. Diese Option steht bei der Tabelle `konto` ebenfalls zur Verfügung. Wurde die Tabelle allerdings zuvor schon genutzt und enthält bereits Daten muss sie erweitert werden. Erweitern kann man diese durch folgende SQL-Querys:

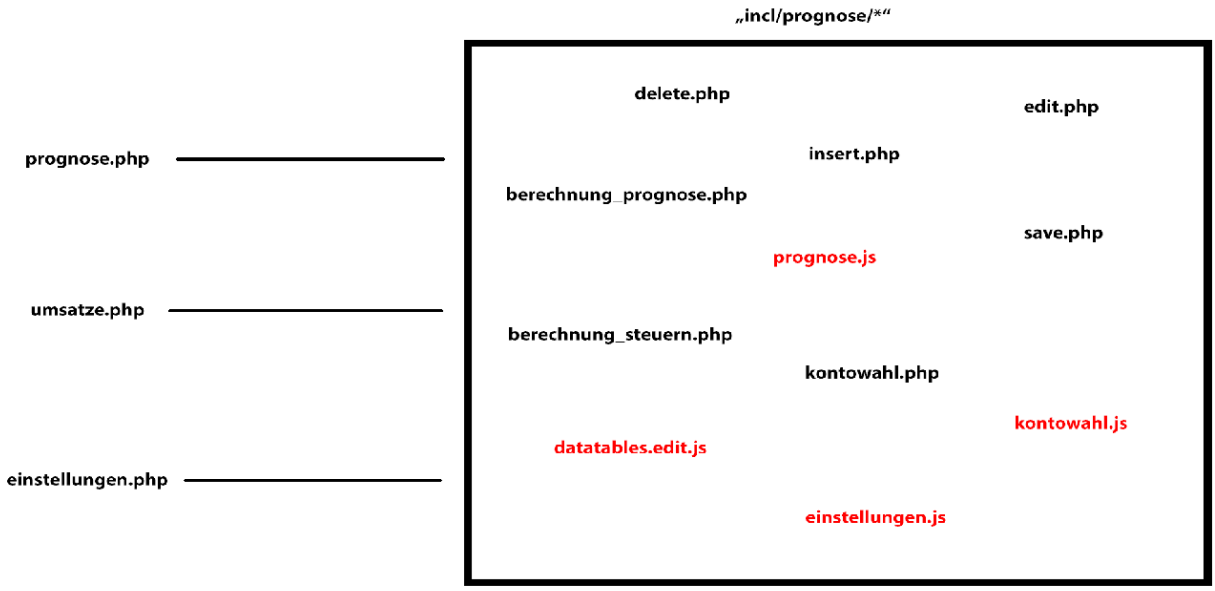
```
1  ALTER TABLE `konto` ADD `BERECHNUNG` BOOLEAN NOT NULL DEFAULT FALSE ;
2  ALTER TABLE `konto` ADD `TAG` TINYINT NOT NULL DEFAULT '1' ;
3  ALTER TABLE `konto` ADD `MONAT` TINYINT NOT NULL DEFAULT '1' ;
```

Sind die Dateien auf dem Server und die Datenbank angepasst, müssen die Dateien 'prognose.php' / 'umsatze.php' / 'einstellungen.php' nur noch in die Navigation eingebaut werden, damit sie auch über die GUI aufrufbar sind. Damit ist die Installation der Prognosekomponente abgeschlossen und die Prognose voll funktionsfähig. Alle Seiten nutzen die Style Sheets (*.css) des Frameworks Bootstrap 4.

4. Aufbau

4.1. Übersicht

Die Prognose besteht letztendlich aus drei Seiten 'prognose.php', 'umsatze.php', 'einstellungen.php', die wiederum auf Dateien in 'incl/prognose/' zugreifen.



Alle Dateien, welche auf die Datenbank zugreifen inkludieren die Datei 'incl/frontend/dbconnect.php', welche eine Verbindung zur Datenbank herstellt. Die Seite 'prognose.php' beinhaltet die eigentliche Prognose, 'umsatze.php' ermöglicht es dem User wiederkehrende Transaktionen einzutragen, zu editieren und zu löschen. Auf der Seite 'einstellungen.php' können Einstellungen zur Steuerberechnung vorgenommen werden. Wichtiger Bestandteil der drei „Hauptseiten“ ist die Kontoauswahl, welche durch das Inkludieren der 'kontoauswahl.php' eingebunden wird. Diese bindet ein Dropdownmenü ein durch das ein Konto – in Form einer IBAN – ausgewählt werden kann. Die Daten dieses Kontos dienen dann als Grundlage für alle Funktionen.

Hierbei wird zu Beginn die in der Session hinterlegte UserID entnommen. Mit Hilfe der UserID wird in der Datenbank nach Konten des Users gesucht und eine Sessionvariable '$_SESSION["iban"]' angelegt und standardmäßig auf das erste Konto des Users gesetzt. Das Dropdownmenü wird mit allen Konten des Users gefüllt. Wird im Dropdown nun ein Konto ausgewählt, wird eine Funktion der eingebundenen JavaScript Datei 'kontowahl.js' aufgerufen. Diese entnimmt dem Dropdownmenü den ausgewählten Wert (IBAN) und sendet ihn (per ‚POST'-Methode) an die jeweilige Datei, in der das Konto gewählt wurde. Hier bekommt nun die Sessionvariable '$_SESSION["iban"]' den Wert der per „POST" gesendeten IBAN. Die Seitenvariable '$iban' bekommt den Wert der Sessionvariable '$_SESSION["iban"]'. Dadurch wird auch beim Wechseln der Seite das ausgewählte Konto beibehalten – zumindest bis zum Erlöschen der Session.

```php
//User (Login)
$userid = $_SESSION["id"];

//Set Default Session IBAN
if (!$_SESSION["iban"])
{
    $sql = "SELECT `IBAN` FROM `konto` WHERE `KUNDE_ID` = '$userid'";
    $result = $db->query($sql);
    $row = $result->fetch_assoc();
    $_SESSION["iban"] = $row["IBAN"];
}

//Set Selected Session IBAN
if(isset($_POST[iban]))
{
    $_SESSION["iban"] = $_POST[iban];
}

//Set Page IBAN
$iban=$_SESSION["iban"];
```

prognose.php, umsatze.php, einstellungen.php

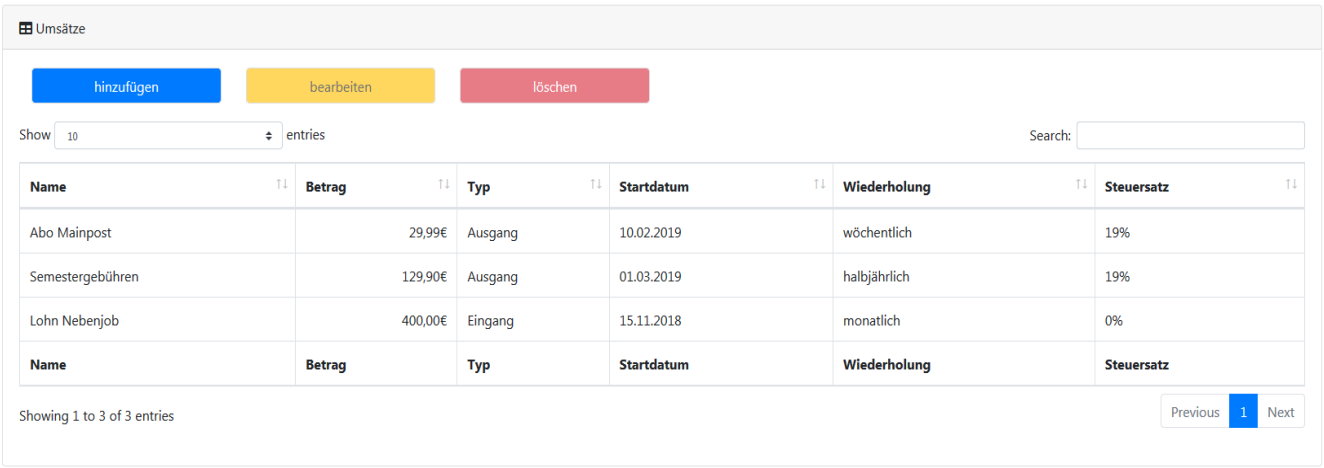
4.2. Einstellungen

Auf der Seite 'einstellungen.php' können Einstellungen zur Steuerberechnung vorgenommen werden. Dabei werden immer die Einstellungen des bei der Kontoauswahl angegebenen Kontos (IBAN) bearbeitet. Die Einstellungen umfassen zum einen, ob überhaupt eine Berechnung der Steuern durchgeführt werden soll und zum anderen an welchem Datum => „Bilanzstichtag" bei Unternehmen. Dazu verwende ich hier ein Formular („form"), dessen Eingabe beim Abschicken (betätigen des 'speichern' Buttons) von der eingebundenen 'einstellungen.js' entgegengenommen wird.

In der Datei 'einstellungen.js' werden die Daten (per ‚POST'-Methode) – inklusive IBAN des ausgewählten Kontos – an die Datei 'save.php' gesendet, welche die Daten per SQL-Query in die Datenbank einpflegt.

4.3. Umsätze

Auf der Seite 'umsatze.php' können wiederkehrende Transaktionen (Umsätze) angelegt, bearbeitet und gelöscht werden. Dafür wird ein „DataTable" mit allen Transaktionen aus der Tabelle 'umsaetze' der derzeit hinterlegten IBAN angezeigt. Jede angelegte Transaktion beinhaltet {(ID,) Name, Betrag, Typ, Startdatum, Wiederholung, Steuersatz}. Für die Funktionen 'hinzufügen', 'bearbeiten', 'löschen' gibt es jeweils einen gleichnamigen Button. Die Spalte 'ID' wird benötigt um beim Bearbeiten und Löschen die entsprechenden Einträge in der Datenbank zu finden. Da der User die 'ID' allerdings nicht benötigt, wird sie für den User ausgeblendet.



Die beiden Buttons 'bearbeiten' und 'löschen' sind standardmäßig deaktiviert, denn um einen Eintrag zu löschen, muss in dem „DataTable" erst ein Eintrag selektiert sein. Beim Hinzufügen und Bearbeiten wird ein Modal geöffnet, welches dem User die nötigen Input Felder vorgibt. Bei Bearbeiten sind diese Felder mit den aktuellen Werten vordefiniert. Wenn man einen Wert ändern möchte, muss man also die unveränderten nicht erneut eingeben. Das selektieren von Einträgen sowie die Button-Funktionen werden in der Datei 'datatable.edit.js' gemanaged.

Hier ein kleiner Ausschnitt, in dem die Funktion des 'bearbeiten'-Buttons definiert ist.

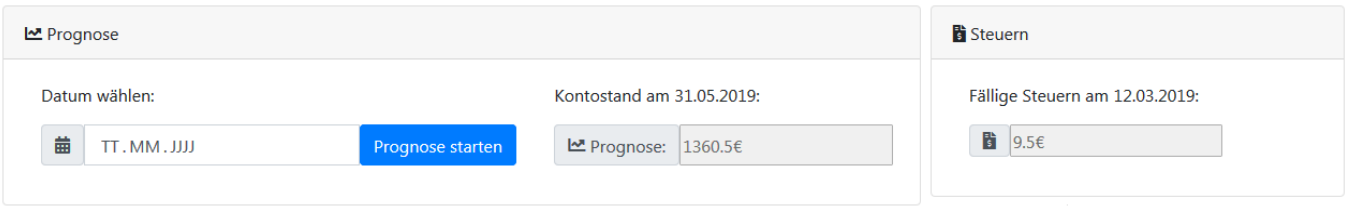
```
// BUTTON FUNCTION
$('#edit').click( function () {
    var bezeichnung = table.$('tr.table-active').find("td").eq(1).html();
    var betrag = table.$('tr.table-active').find("td").eq(2).html();
    var typ = table.$('tr.table-active').find("td").eq(3).html();
    var wiederholung = table.$('tr.table-active').find("td").eq(5).html();
    var steuer = table.$('tr.table-active').find("td").eq(6).html();

    // Datum umformatieren
    var datumNative = table.$('tr.table-active').find("td").eq(4).html();
    var datumAr = datumNative.split('.');
    var datum = datumAr[2] + '-' + datumAr[1] + '-' + datumAr[0];

    // Formular ausfüllen
    $("#bezeichnung").val(bezeichnung);
    $("#betrag").val(betrag.replace(/\./g,'').replace(/\,/g,'.').replace('€', ''));
    $('#typ').find(':radio[name=typ][value='+typ+']').prop('checked', true);
    $("#datum").val(datum);
    $("#wiederholung").val(wiederholung);
    $('#steuer').find(':radio[name=steuer][value='+steuer.slice(0,-1)+']').prop('checked', true);


});

// EDIT
$('#edit_form').on('submit', function(event){
    event.preventDefault();
    var edit_data = $('#edit_form').serialize();

    // TransaktionsID
    var id = table.$('tr.table-active').find("td").eq(0).html();
    edit_data += "&id=" + id;

    if (confirm('Sind sie sicher, dass sie diesen Eintrag bearbeiten möchten?')) {
        $.ajax({
            url: 'incl/prognose/edit.php',
            type: 'POST',
            data: edit_data,
            success: function(data) {
                $('#edit_form')[0].reset();
                $('#edit_data_modal').modal('hide');
                location.reload();
            }
        });
    };
});
```

datatable.edit.js

In der Datei 'datatable.edit.js' werden für die jeweiligen Funktionen {hinzufügen, bearbeiten, löschen} „POST“s an die Dateien {'insert.php', 'edit.php', 'delete.php'}, welche dann die jeweiligen Funktionen – durch SQL-Querys - auf der Datenbank ausführen.

4.4. Prognose

Auf der Seite 'prognose.php' findet letztendlich die eigentliche Prognose statt – zumindest die Darstellung des Ergebnisses. Dafür wird der User aufgefordert, links ein Datum anzugeben für das der Kontostand vorhergesagt werden soll und rechts daneben, wird ihm das Ergebnis angezeigt. Am rechten Rand, wird dem User die prognostizierte fällige Steuer am Bilanzstichtag angegeben – falls Steuerberechnung aktiviert.



Die Prognoseberechnung ist eine Funktion „prognose($iban, $datum)“ in der Datei 'berechnung_prognose.php'. Der Funktion werden eine IBAN und ein Datum mitgegeben, für das man den Kontostand prognostizieren möchte. Dabei werden zunächst alle zur derzeit aktiven IBAN zugehörigen Einträge aus der Tabelle 'umsaetze' abgerufen und für jeden Eintrag vorerst der 'Betrag', das 'Startdatum', sowie der 'Typ' verarbeitet und als nutzbare Daten in Variable hinterlegt.

```php
$sql = "SELECT * FROM `umsaetze` WHERE `IBAN` = '$iban'";
$result = $db->query($sql);
if ($result->num_rows > 0)
{
    while($row = $result->fetch_assoc())
    {
        //Betrag
        $betrag = $row["BETRAG"];

        //Startdatum
        $start = $row["START"];

        //Eingang oder Ausgang
        $modifier = 1;
        if($row["TYP"] == 'Ausgang')
        {
            $modifier = -1;
        }
```

Es werden der Kontostand, sowie ob eine Steuerberechnung durchgeführt werden soll, aus der Datenbank abgerufen.

```php
$sql = "SELECT * FROM `konto` WHERE `IBAN` = '$iban'";
$result = $db->query($sql);
$row = $result->fetch_assoc();
$kontostand = $row["BETRAG"];
$berechnung = $row["BERECHNUNG"];

// Get Bilanzstichtag
$day = sprintf("%02d", $row["TAG"]);
$month = sprintf("%02d", $row["MONAT"]);
$year = date("Y");

$stichtag = date($year."-".$month."-".$day);
```

Anschließend wird für jeden Eintrag überprüft, in welchem Intervall sich die Transaktion wiederholt und dementsprechend die Berechnung durchgeführt. Hierfür habe ich eine switch-Anweisung verwendet. Folgende Ausschnitte zeigen die Berechnung bei einmaligen, täglichen und vierteljährlichen Intervallen.

```php
switch ($row["WIEDERHOLUNG"]) {
    ////////////////////////////////////////////////////
    //EINMALIG
    case "einmalig":
        if($start <= $datum && $start > $today)
        {
            $kontostand = $kontostand+($betrag*$modifier);
        }
        break;

    ////////////////////////////////////////////////////
    //TÄGLICH
    case "täglich":
        if($start <= $datum)
        {
            if($start < $today) $start = date('Y-m-d', strtotime($today. ' + 1 days'));
            while($start <= $datum)
            {
                $kontostand = $kontostand+($betrag*$modifier);
                $start = date('Y-m-d', strtotime($start. ' + 1 days'));
            }
        }
        break;
```

```php
    ////////////////////////////////////////////////////
    //VIERTELJÄHRLICH
    case "vierteljährlich":
        if($start <= $datum)
        {
            $mt = date('d', strtotime($start));
            while($start <= $datum)
            {
                if($start > $today)
                {
                    $kontostand = $kontostand+($betrag*$modifier);
                }

                $zmonate = 3;

                //Datum auf nächsten Monat setzen
                $monat = date("m",strtotime($start));
                $jahr = date('Y', strtotime($start));
                $zielWert = date('Y-m-d', strtotime($jahr.'-'.($monat + $zmonate).'-'.$mt));
                $zielMonat = date("m", strtotime($zielWert));

                if($monat==$zielMonat-$zmonate)
                {
                    $start = date('Y-m-d', strtotime($jahr.'-'.($monat + $zmonate).'-'.$mt));
                }else
                {
                    $start = date('Y-m-d', strtotime($jahr.'-'.($monat + $zmonate-1).'-'.$mt));
                    $start = date('Y-m-d', strtotime("last day of next month",strtotime($start)));
                }

            }
        }
        break;

    ////////////////////////////////////////////////////
```

Nachdem alle Einträge abgearbeitet sind, wird – falls aktiviert - vom prognostizierten Kontostand noch die berechnete Steuer abgezogen. Die Steuerberechnung ist eine Funktion „steuer($iban)“ in der Datei 'berechnung_steuer.php'. Die Steuerberechnung verläuft nach dem gleichen Prinzip wie die Prognose, nur dass hier kein Datum mitgegeben wird, sondern das in den Einstellungen hinterlegte Datum (Bilanzstichtag) verwendet wird. Zudem wird…

```php
$kontostand = $kontostand+($betrag*$modifier);
```

…durch…

```php
$steuern = $steuern+($betrag*$modifier*($steuersatz/100));
```

…ersetzt.