Lehrplanung

Die Anwendung „Lehrplanung“ ist im Rahmen der Lehrveranstaltung „Enterprise Knowledge Graph Implementation" entwickelt worden.

Ziel des Projektes ist die Entwicklung einer Anwendung, welche die Dozenten bei der Modulplanung an der Technischen Hochschule Brandenburg unterstützt.

Die Anwendung wurde von Grund auf neu entwickelt.
Anforderungen die Rahmen dieses Projektes umgesetzt worden sind:

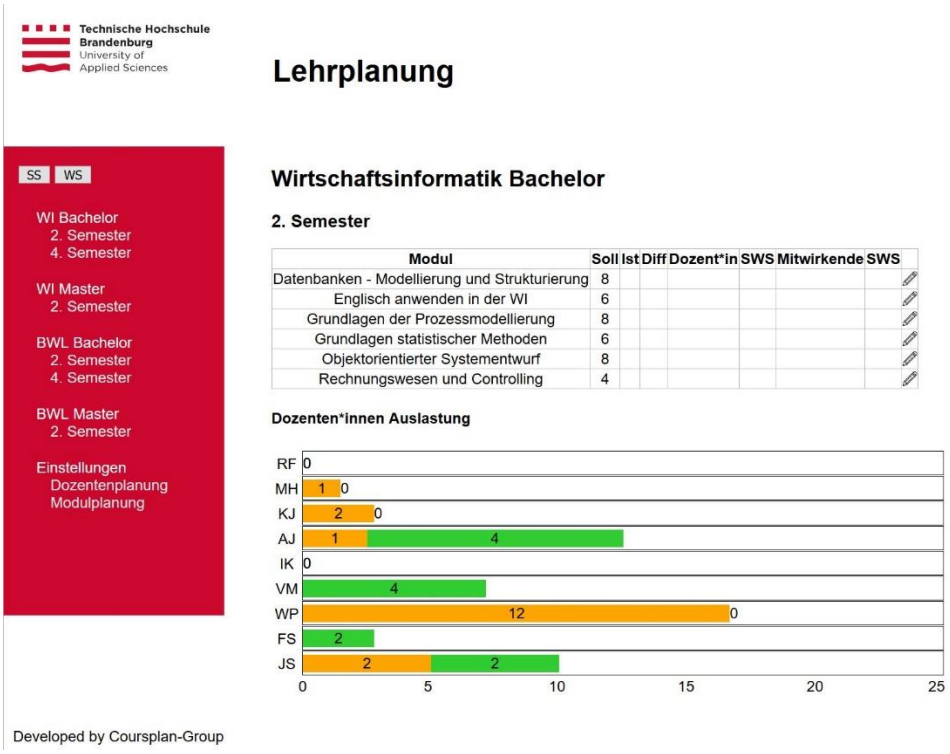
- Entwicklung eines Wissensschema
- Umsetzung eines RDF-Schemas mit Beispieldaten
- Kompaktes klares Design das alle notwendigen Informationen übersichtlich darstellt
- Auswahl der System-Architektur und Einrichtung der Server
- Entwicklung der Anwendung mit PHP und SPARQL
- Implementierung der Anwendung Lehrplanung
- Interaktive Zuordnung von Lehrveranstaltungen zu Dozenten mit paralleler Statusvisualisierung
- Ist innerhalb des Universitätsnetzes erreichbar, läuft auf einem internen Server
- Visualisiert den Planungsstatus von Studiensemestern, Modulen und Dozenten

Design

Der erste Design Prototyp wurde mit Hilfe der Design Thinking Methode umgesetzt und wurden in einen iterativen Prozess stetig weiterentwickelt.

Durch Anwendung der Scrum Methode und die damit verbundenen regelmäßige Meetings, konnten die einzelnen Arbeitspakete schnell umgesetzt werden. Das Hauptziel war es einfache übersichtliche Oberfläche zu erstellen, damit alle relevanten Informationen schnell erreicht und kompakt dargestellt werden. Während des Entwicklungsprozesses wurde das Design stetig weiterentwickelt. Die Anforderungen und neue Funktionen konnten durch die agile Arbeitsweise und den regelmäßigen

Austausch mit unserer Auftraggeberin (Fr. Prof. Dr. Vera G. Meister) schnell umgesetzt werden.



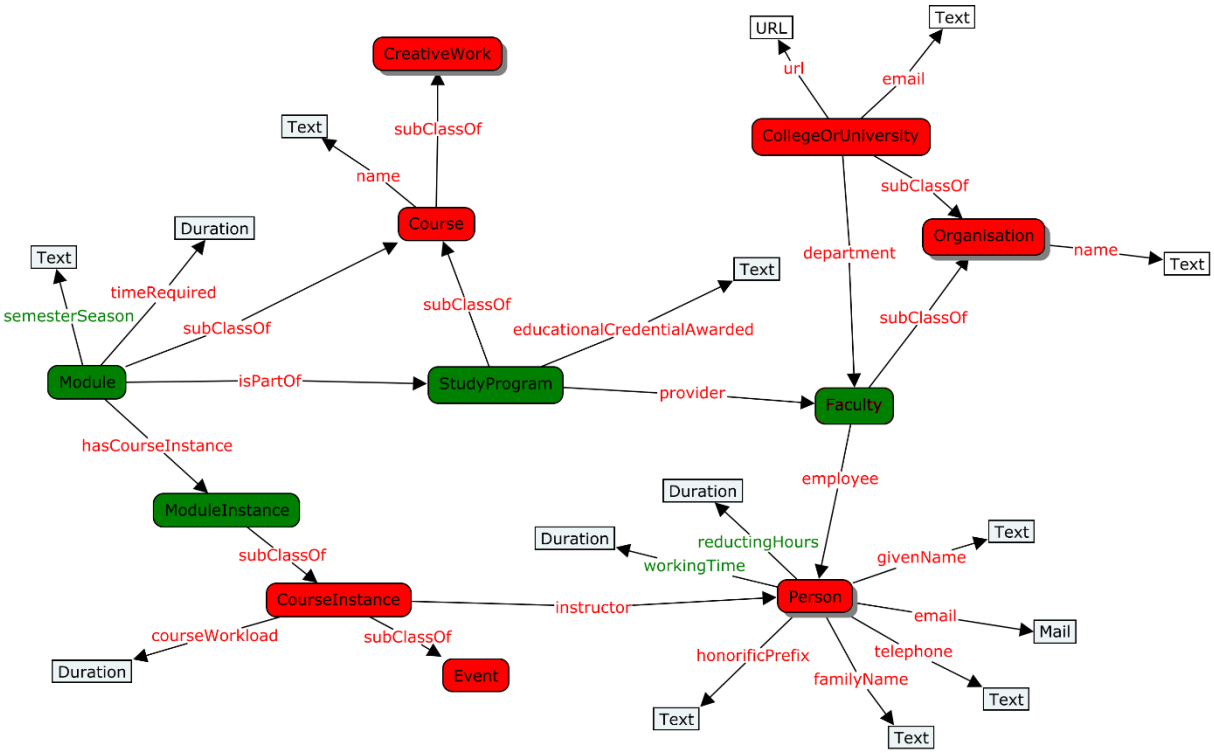
Wissensschema Lehrplanung

Die Grundlage zur Entwicklung unserer Anwendung bildet das Wissensschema, mit dem Ziel, die Dozenten bei der Lehrplanung innerhalb der einzelnen Fakultäten der Technischen Hochschule Brandenburg zu unterstützen. Es wird die Struktur, sowie die Relationen zu den Studiengängen mit den einzelnen Modulen und Dozenten der Technischen Hochschule Brandenburg im Gesamtüberblick vermittelt und dient als Basis zur Entwicklung der Semantischen Anwendung.

Die aktuelle Version des Wissen Schema umfasst neben den Modulen und Dozenten sowohl die Studiengänge als auch die Fachbereiche der Technischen Hochschule Brandenburg.

In der aktuellen Entwicklung wird ausschließlich der Fachbereich Wirtschaft betrachtet. Sollte das Tool in der Zukunft, auch in den anderen Fachbereichen genutzt werden, ist mit dem Wissensschema bereits die Grundlage geschaffen und kann bei Bedarf erweitert werden.

Die rot gekennzeichneten Klassen, Relationen und Attributen sind durch das Vokabular von schema.org definiert. Das grün gekennzeichnete Vokabular basiert auf die Sprache, die im Hochschulkontext verwendet wird und dem Vokabular, das in der der Technischen Hochschule Brandenburg verwendet wird. Die Modellierung dieses Wissensgraphen erfolgt mit Hilfe des Tools Cmap.



Entwicklung des RDF-Dokuments

Die Grundlage zur Entwicklung des RDF-Schemas bildet das Wissensschema, dieser wurde mit Hilfe des Tools RDF-Editor in das Schema überführt.

Das aktuelle RDF Dokument umfasst nur einen Teil der Objekte und Attribute, die im Wissensschema abgebildet sind, da in der aktuellen Version wie bereits erwähnt nur der Wirtschaftsbereich der TH Brandenburg berücksichtigt wurde. Zu Beginn wurden die Objekte und dazugehörigen Attribute definiert.

Im nächsten Schritt wurde das RDF Schema mit Daten befüllt, die notwendigen Informationen stammen aus dem Modulkatalog des Fachbereich Wirtschaft und der Webseite der TH Brandenburg. Mit den dazugehörigen Datenbasis konnten die ersten SPARQL Abfragen erstellt und getestet werden.

Architektur
Der Zugriff auf die Anwendung erfolgt ausschließlich über den internen Bereich der TH Brandenburg und ist nur innerhalb des internen Hochschulnetzes oder mit Hilfe eines VPN abrufbar.

Im Zuge der Entwicklung wurde ein eigener Fuseki Server im internen Hochschulnetz eingerichtet, auf dem die RDF-Daten gespeichert sind.

Zusätzlich wurde ein Webserver eingerichtet, auf dem sich eigentliche Anwendung mit den dazugehörigen PHP-Dateien befindet.

http://fbw-sgmwi.th-brandenburg.de/lehrplanung/ befindet sich eine Webseite der Lehrplanung.
http://fbw-sgmwi.th-brandenburg.de:3030/index.html befindet sich der verwendete Fuseki Server.
https://github.com/dittmanm/EKGE-WS2019-CoursePlan/tree/master/Prototyp
hier befindet sich der Prototyp auf github.

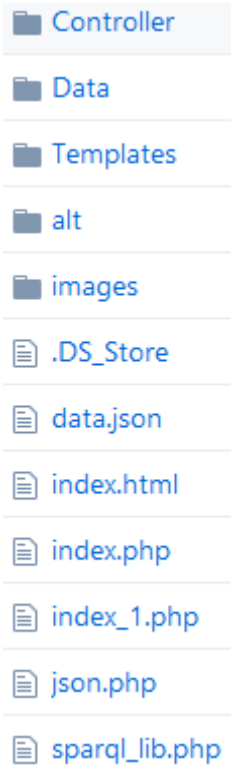
Umsetzung der Website „Lehrplanung“

Die Webseite Lehrplanung wurde nach dem MVC Prinzip entwickelt, wobei in der Entwicklung nur die View und der Controller betrachtet wurde und die Daten im RDF Schema abgerufen und gespeichert werden. Ziel ist es eine übersichtliche Anwendung zu schaffen, die gut wartbar ist und die Weiterentwicklung zusätzlichen Funktionen auch in Zukunft erleichtert.

- **Model:** Das Model hat die Aufgabe, dass die Webanwendung mit Daten aus der Datenbank (oder von wo auch immer) zu versorgen und die Daten, wenn gewünscht, zu speichern. In der Anwendung Kursplanung werden die Daten im RDF Schema gespeichert und abgerufen und werden nicht im Model abgebildet.
- **View:** Die View beinhaltet die Verwaltung der Templates, und generiert die HTML-Ausgabe.
- **Controller:** Im Controller wird entschieden wie und wohin die Parameter übergeben werden und steuert die eigentliche Anwendung.

Struktur des Projektes

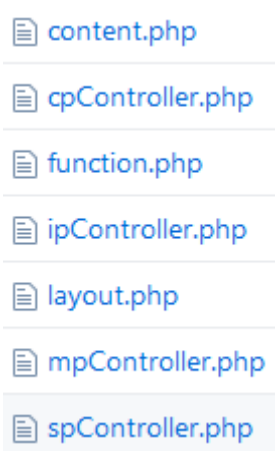
Die Struktur des Projektes sieht wie folgt aus:



Im weiteren Teil dieses Abschnitts werden einzelne Teile der Webseite und Struktur näher erklärt wie z.B. die Controller und die View und deren einzelnen Aufgaben und Funktionen.

Controller

Der kontrolliert die Anwendung und bestimmt was angezeigt wird. Welche Daten übergeben werden sollen oder empfangen werden soll. Die Daten werden an die View übergeben und an den Besucher der Seite übermittelt.



Im Folgenden wird kurz dargestellt, für welche Aufgaben die einzelnen Controller zuständig sind.

layout.php

- Aufbau der Menüs
- Header logo
- Header Name/Hauptname

Function.php

- in zuständig für die Verbindung mit dem Fuseki Server
- organisiert die Datenabfragen vom Fuseki Server (SPARQL)
- Überprüfung der Session zur Auswahl von SoSe oder WiSe

content.php

- Organisiert die Dateiabfrage zur Darstellung des Inhalts im Template

cpController.php

- organisiert die Datenabfrage, -update und -löschen der Kursplanung

ipController.php

- organisiert die Datenabfrage, -update und -löschen der Dozenten

mpController.php

- organisiert die Datenabfrage, -update und -löschen der Modulpläne

spController.php

- organisiert die Datenabfrage, -update und -löschen der Studiengänge

Exemplarisch wird kurz auf den Code vom ipController näher eingegangen von der Struktur sind die Controller (mp, ip, sp, cp) fast identisch aufgebaut und beinhalten ähnliche Funktionen.

Github codeblock einfügen ipcontroller

```php
<?php
class InstructorPerson {
  protected $result;
    public function getFielddata () {
      global $request;
      $fielddata = array(
        "id" => $request["id"],
        "givenName" => $request["givenName"],
        "familyName" => $request["familyName"],
        "honorificPrefix" => $request["honorificPrefix"],
        "email" => $request["email"],
        "contructualHours" => $request["contructualHours"],
        "reductingHours" => $request["reductingHours"]);
    return $fielddata;
  }
```

Im ipController befindet sich die Klasse InstructorPerson die jeweils eine Vielzahl von Funktionen beinhalten, um die Informationen vom Fuseki Server abzurufen, speichern, löschen oder auch ändern.

Eine Funktion im ipController ist die Funktion getFielddate in dem die einzelnen Informationen wie Vorname, Nachname, Titel, E-Mail, Deputatsstunden und Minderungsstunden als Array in der Variable fielddata gespeichert werden.

Weitere Funktionen sind

- getInstructorHours
  Stunden der Dozenten als Instructor

- getContributorHours
  Stunden der Dozenten als Mitarbeiter

- function listAction
  komplette Auflistung von der jeweiligen Klasse im Controller

- function filterAction
  mit Hilfe der Funktionen können Informationen gefiltert werden, bzw. abfragen gezielt ausgeführt werden z.B. zeige nur die Module des Studienganges Bachelors Wirtschaftsinformatik an

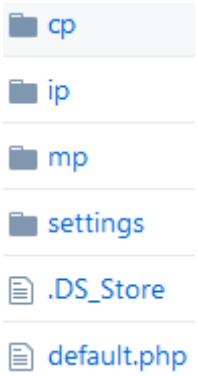
- public function insertAction
  neue Daten dem RDF Schema hinzufügen

- function updateAction
  Daten im RDF Schema aktualisieren

- function deleteAction
  Daten aus dem RDF Schema löschen

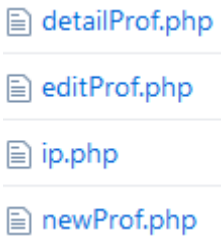
View

Die Templates sind zuständig für die Darstellung der Daten mithilfe des Controllers und der Funktionen, die Anzeige des Inhalts erfolgt im Hauptbereich der Webseite.



Im Template Ordner befinden sich ich die Dateien zu Kursplanung, Modulplanung und den Dozenten. Im jedem diesem Ordner befindet sich jeweils die Templates für die Übersicht, Löschen, Editieren und Neu Erstellen einer Instanz.

Exemplarisch wird der Template Ordner der Dozenten vorgestellt



In der ip Template Datei werden die Dozenten aufgelistet mit Titel, Namen Deputatsstunden und Minderungsstunden in einer Tabelle übersichtlich dargestellt mit der Option die einzelnen Einträge zu editieren (editProf) oder zu löschen (detailProf). Zusätzlich gibt es die Möglichkeit neue Dozenten (newProf) anzulegen.

Github codeblock einfügen

```php
<?php
  $person = new InstructorPerson();
  $main = new Main();
  global $request;
  if($request['action'] === 'create') {
    $namArr = (str_word_count($request["givenName"].' '.$request["familyName"], 1));
    foreach ($namArr as $value) { $word = $word . substr($value, 0, 1); }
    $request['id'] = $word;
    $res = $main->updateAction($person->insertAction($request));
  } elseif($request['action'] === 'update') {
    $delDat = $main->queryAction($person->filterAction('cp:'.$request['id']));
    foreach($delDat as $datArr) {
      $datArr['id'] = str_replace('https://bmake.th-brandenburg.de/cp/', '', $datArr['id']);
      $res = $main->updateAction($person->deleteAction($datArr));
    }
    $res[] = $main->updateAction($person->updateAction($request));
  } elseif($request['action'] === 'delete') {
    $delDat = $main->queryAction($person->filterAction('cp:'.$request['id']));
    foreach($delDat as $datArr) {
      $datArr['id'] = str_replace('https://bmake.th-brandenburg.de/cp/', '', $datArr['id']);
      $res = $main->updateAction($person->deleteAction($datArr));
    }
  }
  //sleep(15);
  $list = $main->queryAction($person->listAction());
?>
```

Die Ausgabe des ip Template (ip.php) der Dozent*innen-Planung:

Dozent*innen-Planung

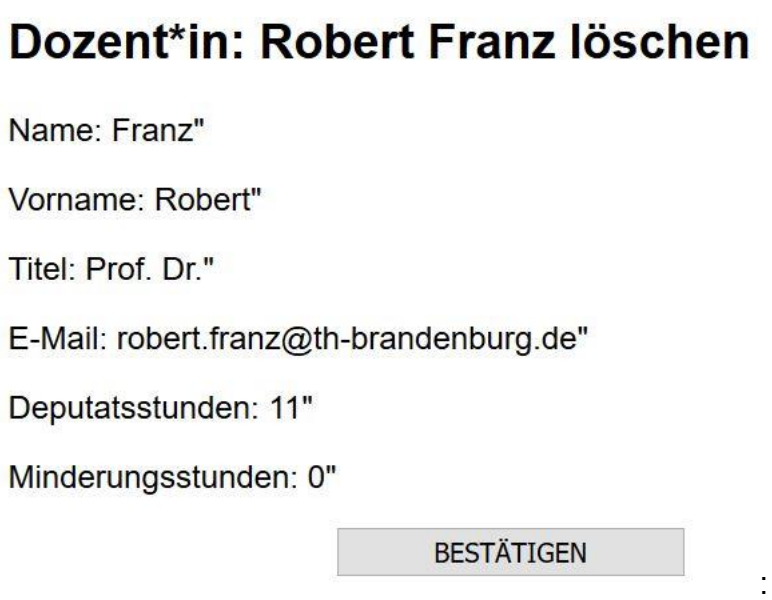
Titel	Name	Dep.	Mind.		
Prof. Dr.	Robert Franz	11	0		
Prof. Dr.-Ing.	Michael Höding	17	1		
Prof. Dr.	Kai Jander	18	2		
Prof. Dr.	Andreas Johannsen	10	1		
Prof. Dr.	Ivo Keller	8	0		
Prof. Dr.	Vera Meister	14	0		
Prof. Dr.	Winfried Pfister	18	12		
Prof. Dr.	Felix Sasaki	18	0		
Prof. Dr.	Jochen Scheeg	10	2		

Neuer Dozent*in

Ausgabe des edit Template (editProf.php) Dozent*innen Einträge zu bearbeiten:

Ausgabe des new Template (newProf.php) Dozent*innen Einträge neu anzulegen:

Ausgabe des delete Template (detailProf.php) Dozent*innen Einträge zu löschen



:

Erklärung einzelner Codeauschnitte im Detail

Menüsichten WS und SS (Session)
Die Darstellung der unterschiedlichen Sichten für das Winter- und Sommersemester, wird mit Hilfe einer Variablen in einer Session verwaltet.

WS => 1
SS => 2
0 => Automatische Auswahl durch Monat

```
public function getSession() {
  if(isset($_SESSION['name'])) {
    if($_SESSION['name'] === 'WS') { $result = 1; }
    elseif($_SESSION['name'] === 'SS') { $result = 2; }
    else { $result = 0; }
  } else { $result = 0; }
            return $result;
}
```

Die Auswahl welche Sicht aktuell auf der Webseite dargestellt erfolgt über die Buttons SS oder WS im Menü. Diese legen eine Variable [$menu = $layout->getMenu($session); echo $menu;] fest und ändern somit das Aussehen des Menüs.

Index.php

```
<div id="menu">
  <form class="switchbtn">
    <button formmethod="post" name="session" value="SS">SS</button>
    <button formmethod="post" name="session" value="WS">WS</button>
  </form>
  <div class="first-menu"><?php $menu = $layout->getMenu($session); echo $menu; ?></div>
</div>
```

Funtions.php

```
public function checkSession() {
  global $request;
  if(isset($request['session'])) {
    if($request['session'] === 'WS') {
      $_SESSION['name'] = 'WS';
    } elseif($request['session'] === 'SS') {
      $_SESSION['name'] = 'SS';
    } else { session_unset(); }
  }
}
```

Sollte keine Auswahl über die Buttons stattgefunden haben findet die automatische Auswahl über den aktuellen Monat statt.

Layout.php

```
public function getMenu($session) {
              if($session === 1) {
                      $menu = $this->getWsMenu();
              } elseif($session === 2) {
                      $menu = $this->getSsMenu();
              } else {
    //$datum = date("n");
    if((date("n") > 3) AND (date("n") < 10)) { $menu = $this->getWsMenu(); }
    else { $menu = $this->getSsMenu(); }
              }
```

Die weiteren Sichten wie die Auswahl des Studiengangs und das Semester erfolgt über die Weitergabe der Variablen im Link.

layout.php

```
class Layout {

        public function getWsMenu() {
        $wi_ba = array(
                        "1. Semester" => "?model=cp&controller=cp&sp=wi_ba&season=1S",
      "3. Semester" => "?model=cp&controller=cp&sp=wi_ba&season=3S",
      "5. Semester" => "?model=cp&controller=cp&sp=wi_ba&season=5S"
                );
        $wi_ma = array(
                        "1. Semester" => "?model=cp&controller=cp&sp=wi_ma&season=1S",
      "3. Semester" => "?model=cp&controller=cp&sp=wi_ma&season=3S"
                );
    $bwl_ba = array(
                        "1. Semester" => "?model=cp&controller=cp&sp=bwl_ba&season=1S",
      "3. Semester" => "?model=cp&controller=cp&sp=bwl_ba&season=3S",
      "5. Semester" => "?model=cp&controller=cp&sp=bwl_ba&season=5S"
    );
    $bwl_ma = array(
                        "1. Semester" => "?model=cp&controller=cp&sp=bwl_ma&season=1S",
      "3. Semester" => "?model=cp&controller=cp&sp=bwl_ma&season=3S"
    );
    $dp = array(
      "Dozentenplanung" => "?model=ip&controller=ip",
      "Modulplanung" => "?model=mp&controller=mp&sp=wi_ba"
    );

    $menu = array(
                        "WI Bachelor" => $wi_ba,
                        "WI Master" => $wi_ma,
      "BWL Bachelor" => $bwl_ba,
      "BWL Master" => $bwl_ma,
      "Einstellungen" => $dp
                );
    return $menu;
  }
```

SPARQL Abfragen

Im Grunde laufen die einzelnen SPARQL Abfragen nach demselben Schema ab, die View kommuniziert mit dem Controller, welches Request bearbeitet wird (anzeigen aller Einträge, bestimmter Einträge (Filter), Speichern, Anlegen und Löschen von Einträgen). Der Controller gibt den SPARQL Code an die function.php weiter und die Abfrage wird über einen HTTP Request an den Fuseki Server gesendet.

In folgenden Abschnitt wird der Code einer Abfrage (Dozent*in anlegen) im Detail erläutert.

Beim Drücken des „SPEICHERN“ Buttons im Formular „Dozent*in anlegen“ wird eine create-Action ausgelöst und auf die Seite mit allen Dozenten*innen weitergeleitet. Die create-Action beinhaltet die Generierung der ID durch den ersten Buchstaben des Vornamens und den ersten Buchstaben des Nachnamens.

```php
public function generateKey($name, $length = 10) {
  $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
  $namArr = (str_word_count($name, 1));
  $key = '';
  foreach ($namArr as $value) {
    $word = $word . substr($value, 0, 1);
  }
  if (strlen($word) < $length) {
    $length = $length - strlen($word);
    $key = $key . substr(str_shuffle(str_repeat($alphabet, $length)), 0, $length);
  }
  return $key;
```

Bei einer Weiterentwicklung der Webanwendung ist geplant das diese ID auch selbst eingegeben werden kann. Es ist jedoch wichtig zu prüfen ob die ID schon vorhanden ist. Nach der Generierung der ID erfolgt das Anlegen über den Aufruf „$main->updateAction($person->insertAction($request));“ Das „$main“ steht dabei für die Klasse Main aus der function.php und das $person für die Klasse InstructorPerson. In

der Klasse InstructorPerson wird bei der insertAction das SPARQL Script erstellt und dann an die updateAction der Klasse Main weitergegeben.

SPARQL Script:

```
$data = 'PREFIX schema: <https://schema.org/>
  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  PREFIX cp: <https://bmake.th-brandenburg.de/cp/>

  INSERT DATA {
    cp:'.$datArr["id"].' a schema:Person;
    schema:givenName "'.$datArr["givenName"].'" ;
    schema:familyName "'.$datArr["familyName"].'" ;
    schema:honorificPrefix "'.$datArr["honorificPrefix"].'" ;
    schema:email "'.$datArr["email"].'" ;
    cp:contructualHours "'.$datArr["contructualHours"].'" ;
    cp:reductingHours "'.$datArr["reductingHours"].'" .
  }';
```

Die Werte zum Dozenten werden vom Formular als POST Variablen übergeben und der insertAction als Array ($datArr) weitergeleitet.

```
public function updateAction ($data) {
  $url = $this->prevUrl.'update';
  $options = array('http' => array(
    'header'  => ['Content-type: application/sparql-update'],['Accept: application/json'],
    'method'  => 'POST',
    'content' => $data
  ));
  $context  = stream_context_create($options);
  $result = file_get_contents($url, false, $context);
  return $result;
}
```

Das SPARQL Script wird der updateAction in der Variablen $data übergeben und für den HTTP POST Request mit eingefügt. Mit den PHP Standard Funktionen „stream_context_create“ und „file_get_contents“ können HTTP Requests am einfachsten gelöst werden. Mit der Funktion „stream_context_create“ werden die Daten als Header-Informationen vorbereitet und dann an die zweite Funktion „file_get_contents“ übergeben und die Anfrage an den Server gestellt.
Durch diese beiden abschließenden Funktionen wurde der Dozenten auf den Fuseki Server als Datensatz angelegt.

Ergebnis

Am Ende nochmal eine kleine Übersicht mit den Funktionen die im Rahmen des Projektes umgesetzt wurden.

- Darstellung der unterschiedlichen Sichten des Menüs je nach Sommersemester und Wintersemester
- Lehrplanung Übersicht alles Dozenten mit Dashboard
- Interaktive Zuordnung von Lehrveranstaltungen zu Dozenten mit paralleler Statusvisualisierung
- Ist innerhalb des Universitätsnetzes erreichbar, läuft auf einem internen Server
- Visualisiert den Planungsstatus von Studiensemestern, Modulen und Dozenten

Einstellungen

- Die Möglichkeit neue Professoren anzulegen bearbeiten oder zu löschen
- Die Möglichkeit neue Module anzulegen editieren und löschen
- Sicherheitsabfrage ob der Datensatz wirklich gelöscht werden soll