# **Interaktive Anwendung und Sichten**

## Inhaltsverzeichnis

Interaktive Anwendung und Sichten	1
1. Nutzung durch Schüler*innen	
1.1. AG-Übersicht	
1.2. Eigene AGs	2
1.3. Besuchte Veranstaltungen	
2. Nutzung durch Lehrer*innen	
2.1. Klassenlisten	
2.2. AG anlegen	
2.3. AG-Verwaltung	
2.3.1. AG-Daten editieren	
2.3.2. AG-Teilnehmer festlegen	6
2.3.3. AG-Termin anlegen und Anwesenheit notieren	
2.3.4. AG-Leiter festlegen	
3. Nutzung durch Jedermann	
3.1. Eigenes Profil editieren	
4. Nutzung durch den Administrator	
4.1. Benutzersuche	
4.2. Benutzerdaten editieren	
4.3. Benutzer hinzufügen	

Bei der zugrundeliegenden Miniwelt handelt es sich um die Verwaltung von Arbeitsgemeinschaften im schulischen Kontext. Die Sichten und die interaktive Anwendung sollen genau dies ermöglichen. Dabei sind verschiedene Nutzungsszenarien relevant, die entsprechend angepasste Sichten erfordern:

### 1. Nutzung durch Schüler\*innen

### 1.1. AG-Übersicht

Schüler\*innen sollen sich über die angebotenen Arbeitsgemeinschaften (wgs) einen Überblick verschaffen können. Daher ist es wichtig, dass AG-Titel, Namen der AG-Leiter, Wochentag, Uhrzeit, Dauer, freie Plätze, Fortsetzbarkeit und eine kurze Beschreibung angezeigt werden.

SELECT \* FROM wgs NATURAL LEFT JOIN

(SELECT count(user\_id),wg\_id FROM participate

WHERE schoolyear = :sy GROUP BY wg\_id) AS one

WHERE schoolyear=:sy

ORDER BY day, time, title ASC

	neinschaften auf ei	Ten Brien				
	AG-Titel öffnet das zugehör ein informativ. Es können kei	=	=			G-Leiter.
Tital	Laibaa	Whh	116	D	6!- DI"	6
	Leiter	Wochentag	Uhrzeit	Dauer	freie Plätze	fortsetzbar
	<b>Leiter</b> Adler, Macker	<b>Wochentag</b> Montag	<b>Uhrzeit</b> 15:00:00	<b>Dauer</b> 90 min	freie Plätze	<b>fortsetzbar</b> nein
Titel Nähen Schach		-				
Nähen	Adler, Macker	Montag	15:00:00	90 min	4	nein
Nähen Schach	Adler, Macker Engel, Usus	Montag Montag	15:00:00 15:00:00	90 min 90 min	4	nein ja

(wg\_list.php)

### 1.2. Eigene AGs

Schüler\*innen sollten sich einen Überblick verschaffen können hinsichtlich der AGs, welche sie selbst besuchen. Dieselben Infos wie bei der Komplettliste sind relevant.

SELECT \* FROM wgs,participate AS p
WHERE p.user\_id = :id AND p.wg\_id = wgs.wg\_id
ORDER BY p.schoolyear DESC, wgs.title ASC

Dabei wird :id durch die aktuelle User-Id ersetzt.

#### Meine Arbeitsgemeinschaften Klicke auf den Titel einer AG, um die AG-Beschreibung zu sehen. Klicke auf das Schuljahresdatum einer AG, um die Liste deiner Teilnahmen zu sehen. Schuljahr Leiter fortsetzbar Titel Wochentag Uhrzeit Dauer Maximum 2020/21 Basketball Brot Mittwoch 16:00:00 90 min 20 ja 2020/21 Nähen Adler, Macker Montag 15:00:00 90 min 10 nein 2020/21 Technik 14:00:00 90 min 6 Jensen Freitag ja 2019/20 Schach Engel, Usus Montag 15:00:00 90 min 10

(wg\_my.php)

### 1.3. Besuchte Veranstaltungen

Für Schüler\*innen sind ihre besuchten AG-Veranstaltungen (events) eventuell von Interesse, wenn Sie z.B. ihren Eltern zeigen möchten, was sie in den letzten Monaten dort getan haben. Datum, Uhrzeit, Dauer und eine kurze inhaltliche Beschreibung reichen hierfür aus.

SELECT date, time, duration, annotation FROM events AS e, present AS p WHERE e.wg\_id = :wid AND p.user\_id = :uid AND e.event\_id = p.event\_id ORDER BY e.date DESC, e.time DESC

Dabei werden :uid durch die aktuelle User-Id und :wid durch die AG-Id ersetzt.



(wg\_my.php)

### 2. Nutzung durch Lehrer\*innen

#### 2.1. Klassenlisten

Klassenlehrer\*innen können sich durch Auswahl ihrer Klasse eine Sicht aller AG-Teilnahmen ihrer Schüler\*innen ausgeben lassen. Dies ist praktisch für das Ausfüllen der Zeugnisvermerke hinsichtlich der AG-Teilnahmen. Ausgegeben werden die Namen der Schüler\*innen, die AG-Titel, die Namen der AG-Leiter und die Anzahl der Teilnahmen.

SELECT u.firstname, u.lastname, w.title, w.wg\_id, COUNT(e.event\_id) AS eventcount FROM students AS s, users AS u, wgs AS w, present AS p, events AS e WHERE s.class=:cl AND e.date>=:da AND u.user\_id=s.user\_id AND s.user\_id=p.user\_id AND p.event\_id=e.event\_id AND e.wg\_id=w.wg\_id GROUP BY u.user\_id, w.wg\_id ORDER BY lastname ASC, firstname ASC, title ASC

Dabei werden :cl durch die gewählte Klasse und :da durch das Datum des ersten Schultages ersetzt.



(classlist.php)

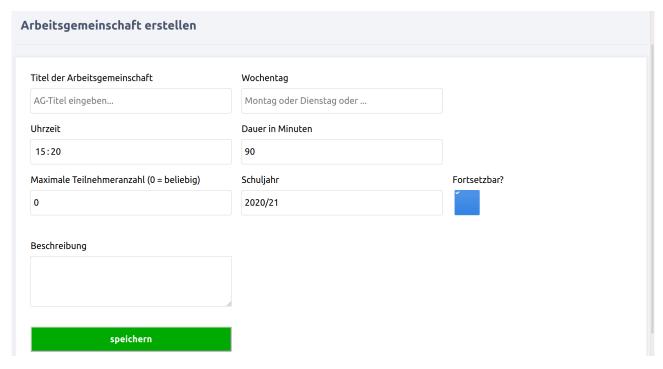
### 2.2. AG anlegen

Die Erfassung neuer AGs muss durch Lehrer selbst möglich sein. Dafür muss eine Sicht zur Verfügung gestellt werden, die es ermöglicht, sämtliche AG-Daten einzugeben, welche anschließend in die Tabelle wgs eingefügt werden. Gleichzeitig muss der/die Lehrer\*in zur AG-Leitungstabelle lead hinzugefügt werden. Beide Insert-Befehle werden in Form einer PDO-Transaktion ausgeführt. Gibt es irgendeinen Fehler, wird ein Rollback durchgeführt.

INSERT INTO wgs(title,day,time,duration,max\_num,multiple,schoolyear,description) VALUES (:title,:day,:time,:dur,:max,:mul,:year,:des)

INSERT INTO lead (user\_id, wg\_id) VALUES (:uid,(SELECT currval('wgs\_wg\_id\_seq')))

Da die neue wg\_id vom Datenbanksystem automatisch vergeben wird und somit vorher noch unbekannt ist, muss sie nach dem ersten Insert abgefragt werden. Dies geschieht mittels SELECT currval('wgs\_wg\_id\_seq').



(wg\_add.php)

### 2.3. AG-Verwaltung

Den AG-Leiter\*innen muss es möglich sein, Änderungen an den AG-Daten vorzunehmen (Tabelle: wgs), Teilnehmer\*innen hinzuzufügen oder zu entfernen (Tabelle: participate), die Anwesenheit bei AG-Terminen zu dokumentieren (Tabellen: events und present) und weitere AG-Leiter festzulegen. Hierfür wird zunächst eine Sicht erzeugt, die alle AGs anzeigt, welche der/die Lehrer\*in leitet. Titel, Wochentag, Uhrzeit und Schuljahr reichen hierfür aus.

SELECT title, day, time, schoolyear,wgs.wg\_id FROM wgs, lead WHERE lead.user\_id = :uid AND wgs.wg\_id = lead.wg\_id ORDER BY schoolyear DESC, title ASC, day ASC, time ASC



(wg\_edit.php)

Einzelne Symbole zum Anklicken ermöglichen die oben geschilderten Funktionalitäten.

#### 2.3.1.AG-Daten editieren

Zunächst werden die Daten zu der AG mit der Id :wid abgefragt und angezeigt. Durch einen Klick auf den Speichern-Button werden sie anschließend in der Datenbank gesichert.

SELECT \* FROM wgs WHERE wg\_id=:wid

UPDATE wgs SET title=:title, day=:day, time=:time, duration=:dur, max\_num=:max, multiple=:mul, schoolyear=:year, description=:des
WHERE wg\_id=:wid

gene AGs verwalten		
Titel	Wochentag	
Nähen	Montag	
Uhrzeit	Dauer in Minuten	
15:00:00	90	
Maximale Teilnehmeranzahl (0 = beliebig)	Schuljahr	Fortsetzbar?
10	2020/21	
Beschreibung		
Wir nähen alle einen Teddybären.		
	<u>h</u>	
		_
speichern	abbrechen	

(wg\_edit.php & wg\_edit\_ajax.php & wg\_edit\_save.php)

### 2.3.2.AG-Teilnehmer festlegen

Damit eine Person der AG hinzugefügt werden kann, muss zunächst einmal erkennbar sein, welche Personen bereits in der AG sind. Zur besseren Unterscheidung von Lehrern und Schülern werden die Infos aus den entsprechenden Tabellen (students / teachers) ebenfalls abgefragt und mittels NATURAL LEFT JOIN angefügt.

```
SELECT * FROM (

SELECT user_id, firstname, lastname, email

FROM users NATURAL JOIN participate p

WHERE p.wg_id = :wid AND p.schoolyear='$schoolyear') AS one

NATURAL LEFT JOIN students

NATURAL LEFT JOIN teachers

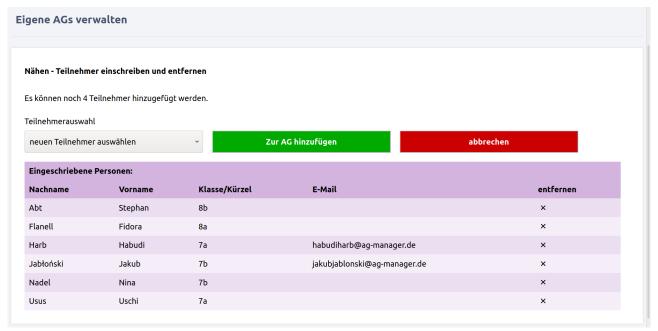
ORDER BY lastname, firstname, class, shorthand
```

Wie viele Plätze noch frei sind, ist ebenfalls wichtig:

SELECT count(user\_id) AS num

FROM participate

WHERE wg\_id = :wid AND schoolyear = :sy



(wg\_edit.php & wg\_participate.php)

Durch ein einfaches Klicken au ein x ist es möglich, Schüler\*innen leicht aus der AG zu entfernen:

DELETE FROM participate WHERE user\_id=:uid AND wg\_id=:wid AND schoolyear=:sy

Für ein einfaches Hinzufügen von Personen steht eine Auswahlliste bereit, welche alle Personen enthält, die nicht bereits in die AG eingeschrieben wurden:

SELECT \* FROM (

SELECT user\_id, firstname, lastname

FROM users

EXCEPT (

SELECT u.user\_id, firstname, lastname

FROM users u, participate p

WHERE p.wg\_id = :wid AND u.user\_id=p.user\_id)) AS noparticipator

NATURAL LEFT JOIN students

NATURAL LEFT JOIN teachers

ORDER BY lastname, firstname, class, shorthand

Wurde eine Person ausgewählt, kann sie nun zur AG hinzugefügt werden:

INSERT INTO participate (user\_id, wg\_id, schoolyear) VALUES (:uid, :wid, :sy)

#### 2.3.3.AG-Termin anlegen und Anwesenheit notieren

Für eine einfache Dokumentation der AG-Inhalte und der Anwesenheit existiert ein View, der Datum, Zeit, Dauer sowie ein Anmerkungsfeld zur Verfügung stellt. Beim Speichern werden diese Daten in die Tabelle events eingetragen.

INSERT INTO events (time, date, duration, annotation, wg\_id) VALUES (:ti, :da, :du, :an, :id)

Ebenso kann man durch das Setzen von Häkchen auf einfache Weise die Anwesenheit der Schüler\*innen dokumentieren.

Abfrage der AG-Teilnehmer für das Bereitstellen der Häkchen:

SELECT \* FROM

(SELECT user\_id, firstname, lastname FROM participate p NATURAL JOIN users WHERE p.wg\_id=:wid AND p.schoolyear=:sy) AS one

NATURAL LEFT JOIN (SELECT students.user\_id, class FROM students) AS s

NATURAL LEFT JOIN teachers

ORDER BY lastname, firstname, class

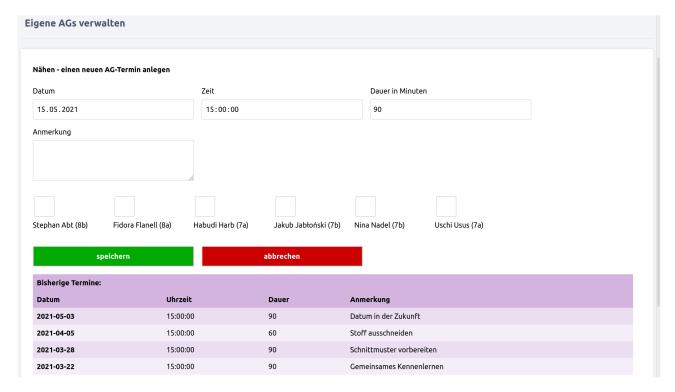
Speichern der Anwesenheit:

INSERT INTO present (user\_id, event\_id) VALUES (u1,e1), (u2,e2), ...

Die einzufügenden Wertepaare erzeugt eine PHP-Schleife durch das systematische Abfragen der Kästchen, ob diese abgehakt wurden oder nicht. Da das Einfügen des Termins in die Tabelle events und das Eintragen der Teilnehmer in die Tabelle present voneinander abhängig sind, werden diese in Form einer Transaktion realisiert. Auf diese Weise bleibt die Datenbank konsistent, falls ein Fehler in einem der beiden INSERT-Befehle auftritt.

Um vorausgegangene Termine im Blick zu haben und diese ggf. noch einmal zu ändern oder zu ergänzen, wird eine entsprechende Tabelle angezeigt.

SELECT \* FROM events WHERE wg\_id = :wid ORDER BY date DESC, time ASC



(wg\_edit.php & wg\_presence\_new.php)

Ein Klick auf einen alten Termin fragt dessen Daten ab.

SELECT \* FROM events WHERE event id = :eid

Außerdem werden die damals anwesenden Teilnehmer als Checkboxen angezeigt. Da zwischenzeiltich ein Teilnehmer aus der AG ausgetreten sein kann, gestaltet sich die Abfrage etwas schwieriger:

#### SELECT \* FROM (

SELECT user\_id, firstname, lastname FROM present pr NATURAL JOIN users

WHERE pr.event\_id=:eid

UNION

SELECT user\_id, firstname, lastname FROM participate pa NATURAL JOIN users

WHERE pa.wg\_id=:wid AND pa.schoolyear=:sy) as one

#### NATURAL LEFT JOIN (

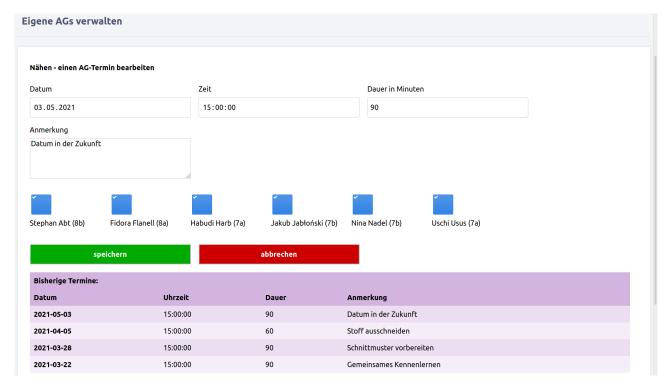
SELECT user\_id, 'y' AS waspresent FROM present WHERE event\_id=:eid) AS two

#### NATURAL LEFT JOIN (

SELECT students.user\_id, class FROM students) AS three

NATURAL LEFT JOIN teachers

ORDER BY lastname, firstname, class



(wg\_edit.php & wg\_presence\_edit.php):

### 2.3.4.AG-Leiter festlegen

Da es durchaus AGs gibt, welche von mehreren Lehrer\*innen oder auch von Schülern selbst geleitet werden, ist es sinnvoll, weitere AG-Leiter festlegen zu können. Ein AG-Leiter, der selbst kein Schüler ist, kann diese Sicht aufrufen. Da Schüler\*innen noch minderjährig sind, wird ihnen diese Funktion verwehrt.

Zunächst einmal wird eine Liste aller bereits existierenden AG-Leiter angezeigt:

```
SELECT * FROM (
```

SELECT user\_id, firstname, lastname FROM users NATURAL JOIN lead l

WHERE l.wg\_id = :wid

**EXCEPT** 

SELECT user\_id, firstname, lastname FROM users WHERE user\_id = :uid) AS one

NATURAL LEFT JOIN students

NATURAL LEFT JOIN teachers

ORDER BY lastname, firstname, class, shorthand

Durch einen Klick auf ein x hinter dem Namen und dem Kürzel, kann ein AG-Leiter euch leicht wieder entfernt werden:

DELETE FROM lead WHERE user\_id=:uid AND wg\_id=:wid

Damit man einen neuen AG-Leiter leicht festlegen kann, werden alle Personen abgefragt, die noch kein Leiter der AG sind und in eine Auswahlliste eingefügt:

#### SELECT \* FROM

(SELECT user\_id, firstname, lastname FROM users

**EXCEPT** 

SELECT u.user\_id, firstname, lastname FROM users u, lead l

WHERE l.wg\_id = :wid AND u.user\_id=l.user\_id) AS noleader

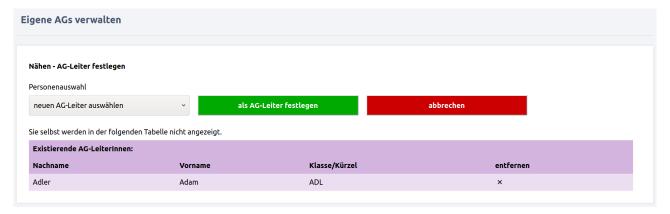
NATURAL LEFT JOIN (SELECT user\_id, class FROM students) AS s

NATURAL LEFT JOIN (SELECT user\_id, shorthand FROM teachers) AS t

ORDER BY lastname, firstname, class, shorthand

Hat man einen neuen AG-Leiter ausgewählt, wird dieser durch Klick auf den grünen Button in der Datenbank gesichert:

INSERT INTO lead (user\_id, wg\_id) VALUES (:uid, :wid)



(wg\_edit.php & wg\_leaders.php)

### 3. Nutzung durch Jedermann

### 3.1. Eigenes Profil editieren

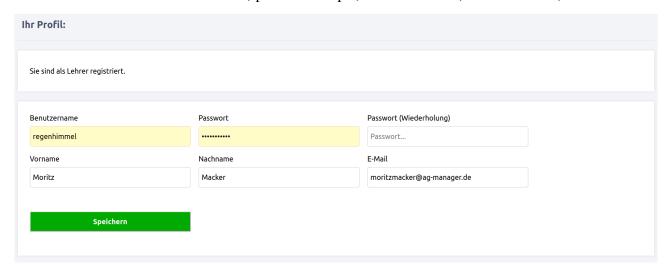
Damit sich die Benutzer leicht zu merkende Benutzernamen aussuchen, Schreibfehler korrigieren, ein neues Passwort aussuchen oder bei Bedarf ihre E-Mailadresse hinterlegen können, ist es sinnvoll, eine Möglichkeit zu schaffen, die eigenen Daten editieren zu können.

Daten aus der Datenbank auslesen:

SELECT \* FROM users

Daten in der Datenbank aktualisieren:

UPDATE users SET username = :un, password = :pw, firstname = :fn, lastname = :ln, email = :em



(profile.php)

### 4. Nutzung durch den Administrator

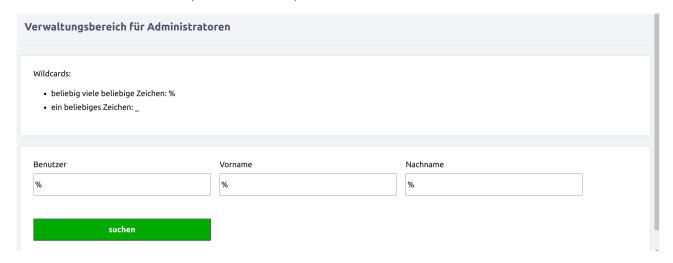
Eine Software, welche von einer Vielzahl von Benutzern eingesetzt wird und deren Daten zentral verwaltet benötigt einen Administrator. Schnell ist einmal ein Passwort vergessen, so dass ein Nutzer keinen Zugriff mehr auf seine Daten hat. Oder ein neuer Kollege muss in das System eingepflegt werden. Hierfür sind mehrere Funktionalitäten vonnöten.

#### 4.1. Benutzersuche

Müssen Änderungen an den Daten eines Benutzers vorgenommen werden, so müssen diese erst einmal gefunden werden. Hierfür ist eine Suchfunktion unerlässlich. Eine Suche nach dem Benutzernamen, dem Vornamen oder dem Nachnamen ist ausreichend.

#### SELECT \* FROM users

WHERE username LIKE :user AND firstname LIKE :fn AND lastname LIKE :ln ORDER BY lastname ASC, firstname ASC, username ASC



(user\_search.php)

Die Ausgabe erfolgt in Form einer einfachen Tabelle, welche bis auf den Passworthash alle Daten aus der Tabelle "users" enthält.



(user\_search.php)

Durch einen Klick auf die ID ist es möglich, die Daten des Nutzers zu editieren.

### 4.2. Benutzerdaten editieren

Neben den Daten in der Tabelle "users" sind bei Lehrern auch das Kürzel und bei Schülern die Schülernummer und die Klasse relevant. Daher werden diese in diesen View mit einbezogen.

Abruf der Daten:

```
SELECT * FROM users WHERE (user_id = :id)
```

SELECT \* FROM students WHERE (user\_id = :id)

SELECT \* FROM teachers WHERE (user\_id = :id)

Änderung der Daten bei leerem Passwortfeld:

UPDATE users SET username = :na, enabled = :en, firstname = :fn, lastname = :ln, email = :email, roll = :roll WHERE user\_id = :id

Änderung der Daten bei ausgefülltem Passwortfeld:

UPDATE users SET username = :na, password = :pwd, enabled = :en, firstname = :fn, lastname = :ln, email = :email, roll = :roll WHERE user\_id = :id

Änderung der Daten in den Tabellen "teachers" und "students":

UPDATE teachers SET shorthand = :sh WHERE user\_id = :uid

oder

INSERT INTO teachers (user\_id, shorthand) VALUES (:id, :sh)

DELETE FROM students WHERE user\_id = :uid

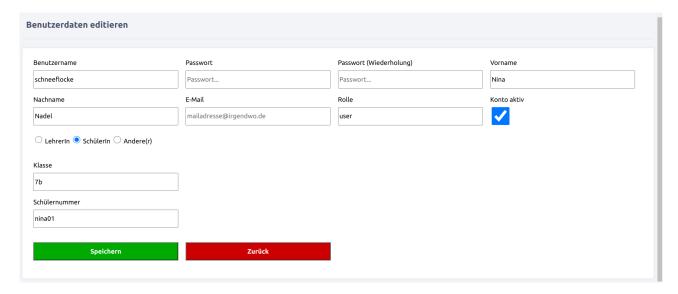
UPDATE students SET class = :cl, studentnumber = :sn WHERE user\_id = :uid

oder

INSERT INTO students (user id, class, studentnumber) VALUES (:id, :cl, :sn)

DELETE FROM teachers WHERE user\_id = :uid

Für den Fall, dass beispielsweise ein Lehrer versehentlich als Schüler eingetragen wurde, und dies nun korrigiert wird, ist kein "UPDATE" auszuführen, sondern die obige Kombination aus "INSERT" und "DELETE".



(user\_edit.php nutzt account\_class.php)

### 4.3. Benutzer hinzufügen

Für das Hinzufügen eines neuen Benutzers wird eine leere Eingabemaske bereitgestellt, die eine Eingabe aller Benutzerinfos ermöglicht.

INSERT INTO users (username, password, firstname, lastname, email, roll, enabled) VALUES (:na, :pw, :fn, :ln, :em, :ro, :en)

ggf.

INSERT INTO teachers (user\_id, shorthand) VALUES (:id, :sh)

oder

INSERT INTO students (user\_id, class, studentnumber) VALUES (:id, :cl, :sn)



(user\_add.php nutzt account\_class.php)