Dossier

<!--

- Praktikum DBWT. Autoren:

- Alexander, Matthew, 3532885

- Bryan, Nathanael Joestin, 3517701

-->

Meilenstein	Aufgabe	Geschätzte Zeit (Minuten)	Tatsächliche Zeit (Minuten)
1-W1	1	1	1
	2	1	1
	3	2	2
	4a	10	15
	4b	10	10
	4c	5	5
	4d	5	5
	4e	2	2
	5	10	15
	6.1	7	10
	6.2	10	7
	7	5	5
1-W2	8a	15	10
	8b	5	8
	9.1	15	20
	9.2	5	5
	10	45	60

Grund für die Abweichung:

Aufgabe 4a - Überlegen des passenden Inhalts

Aufgabe 5 - eine kleine Verwirrung sowie zu den name- & id-Attributen für die radio-Buttons, als auch das Label dafür.

Aufgabe 6.1 - Beschreibung der Attributen im HTTP Header

Aufgabe 9.1 - Auswählen der zweite Zeile der Tabelle und der Layout von Footer Elemente

Aufgabe 10 - Layout der Zahlen und der Formular. Setzen eine linksbündige List in der Mitte der Seite

Ressourcen: MDN Docs, w3schools

Aufgabe 9.2:

- font-family
- font-weight
- text-decoration
- color
- max-width
- margin
- Border
- padding
- padding-bottom
- border-bottom
- border-left
- text-align
- display
- grid-auto-flow
- grid-auto-columns
- justify-content
- height
- list-style

Meilenstein	Aufgabe	Geschätzte Zeit (Minuten)	Tatsächliche Zeit (Minuten)
2-W1	1	5	5
	2	30	23
	3.1	7	5
	3.2	10	10
	3.3	5	5
	3.4	120	200
	4	1	1
	5	75	90
	6	60	65
2-W2	7	15	25
	8	40	45

Ressourcen: MDN Docs, w3schools, php.net

Grund für die Abweichung:

Aufgabe 3.4 - Bei e) hängen geblieben, weil es zunächst schwer zu verstehen war, wie die GET-Methode & -Parameter mit php interagieren.

Aufgabe 5 - War bei d) ein bisschen langsam, da wir herausfinden müssen, wie man zwischen dem array-value und dem int-value unterscheidet.

Aufgabe 7 - Bei b) verzögert, weil wir finden müssen, wie man die Wörter trennt.

Aufgabe 3.1:

- 1. Fehler: Linie 12. Es fehlt ein Komma nach 'Fisch'
- 2. Fehler: Linie 22. Es fehlt ein eckige Klammer (array)
- 3. Fehler: Linie 41. Es fehlt ein Klammer am Ende
- 4. Fehler: Linie 59. Es soll eine funktion sein aber fehlt "Function" Statement am Anfang

Aufgabe 3.2:

Die Benutzer:innen können in einem Suchfeld sowie ein String, der in jeder Textbewertung gesucht wird, als auch ein Integer, der als Rating-Sterne gesucht wird, eintragen. Danach werden die Ergebnisse dieser Suche der Benutzer:innen angezeigt. Der Webbrowser (Client) sendet das eingetragene Suchwort durch ein Formular mit der GET-Methode (bei URL beigefügt). Diese Daten werden dann als GET-Parameter zu der \$_GET Superglobal-Variable gesammelt und damit ein Filter auf alle Bewertungen anwenden. Letztlich werden dann die gefilterten Ergebnisse mit einer Schleife angezeigt.

Aufgabe 3.3:

• : float = Typ-Deklaration für den return-value (hier float) einer Funktion spezifizieren.

Meilenstein	Aufgabe	Geschätzte Zeit (Minuten)	Tatsächliche Zeit (Minuten)
3-W1	1	10	60
	2	30	16
	3	15	23
	4	10	11
	5	45	79
3-W2	6	45	36
	7	10	10
	9	30	56
	10	5	12
	11		
Total:			303

Aufgabe 2.4:

- SELECT COUNT(*) FROM allergen;
- SELECT COUNT(*) FROM gericht;
- SELECT COUNT(*) FROM gericht_hat_allergen;
- SELECT COUNT(*) FROM gericht_hat_kategorie;
- SELECT COUNT(*) FROM kategorie;

Aufgabe 3:

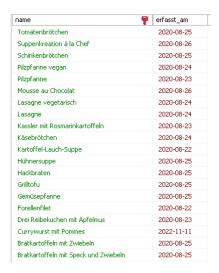
SELECT * FROM gericht; (auch für 11))



SELECT erfasst_am FROM gericht;



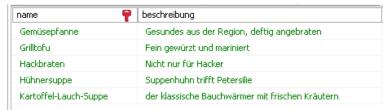
SELECT name, erfasst_am FROM gericht ORDER BY name DESC;



SELECT name, beschreibung FROM gericht ORDER BY name ASC LIMIT
 5;



 SELECT name, beschreibung FROM gericht ORDER BY name ASC LIMIT 5 OFFSET 5;



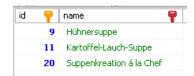
SELECT DISTINCT typ FROM allergen;



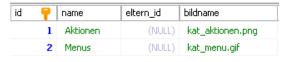
SELECT name FROM gericht WHERE name LIKE 'K%';



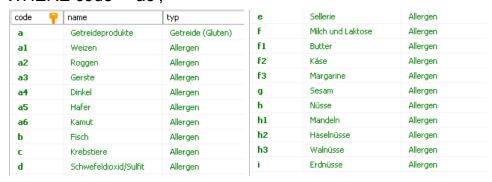
SELECT id, name FROM gericht WHERE name LIKE '%suppe%';



SELECT * FROM kategorie WHERE eltern_id IS NULL;

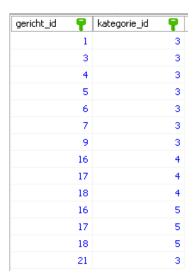


UPDATE allergen
 SET name = 'Kamut'
 WHERE code = 'a6';



 INSERT INTO gericht
 VALUES (21, 'Currywurst mit Pommes', 'Bratwurst mit Currysauce und gefrittierte Pommes', '2022-11-11', 0, 0, 2.8, 5);

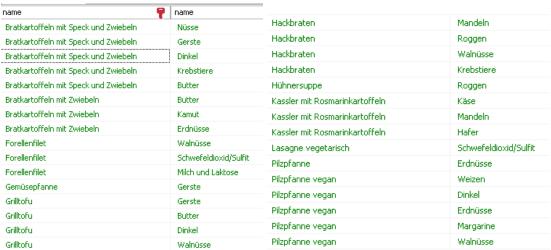
INSERT INTO gericht_hat_kategorie
VALUES (21, 3);



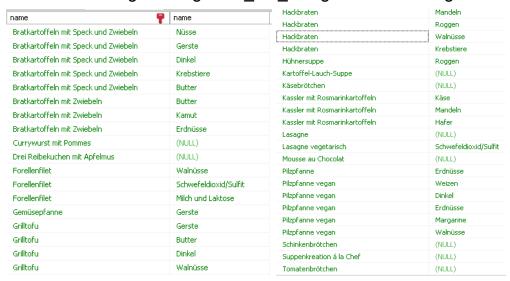
Aufgabe 6:

SELECT gericht.name, allergen.name
 FROM gericht

JOIN gericht_hat_allergen ON gericht.id = gericht_hat_allergen.gericht_id JOIN allergen ON gericht_hat_allergen.code = allergen.code;



SELECT gericht.name, allergen.name
 FROM gericht
 LEFT JOIN gericht_hat_allergen ON gericht.id =
 gericht_hat_allergen.gericht_id
 LEFT JOIN allergen ON gericht hat allergen.code = allergen.code;



SELECT gericht.name, allergen.name
 FROM gericht
 RIGHT JOIN gericht_hat_allergen ON gericht.id =
 gericht_hat_allergen.gericht_id
 RIGHT JOIN allergen ON gericht hat allergen.code = allergen.code;

name	P name	(NULL)	Sellerie
(NULL)	Getreideprodukte	Forellenfilet	Milch und Laktose
Pilzpfanne vegan	Weizen	Bratkartoffeln mit Zwiebeln	Butter
Hackbraten	Roggen	Grilltofu	Butter
Hühnersuppe	Roggen	Bratkartoffeln mit Speck und Zwiebeln	Butter
Bratkartoffeln mit Speck und Zwiebeln	Gerste	Kassler mit Rosmarinkartoffeln	Käse
Grilltofu	Gerste	Pilzpfanne vegan	Margarine
Gemüsepfanne	Gerste	(NULL)	Sesam
Bratkartoffeln mit Speck und Zwiebeln	Dinkel	Bratkartoffeln mit Speck und Zwiebeln	Nüsse
Grilltofu	Dinkel	Hackbraten	Mandeln
Pilzpfanne vegan	Dinkel	Kassler mit Rosmarinkartoffeln	Mandeln
Kassler mit Rosmarinkartoffeln	Hafer	(NULL)	Haselnüsse
Bratkartoffeln mit Zwiebeln	Kamut	Grilltofu	Walnüsse
(NULL)	Fisch	Hackbraten	Walnüsse
Hackbraten	Krebstiere	Forellenfilet	Walnüsse
Bratkartoffeln mit Speck und Zwiebeln	Krebstiere	Pilzpfanne vegan	Walnüsse
Lasagne vegetarisch	Schwefeldioxid/Sulfit	Bratkartoffeln mit Zwiebeln	Erdnüsse
Enrellenfilet	·	Pilzpfanne	Erdnüsse
Forelienniet	Schwefeldioxid/Sulfit	Pilzpfanne vegan	Erdnüsse

 SELECT kategorie.name, COUNT(gericht.name) AS anzahl FROM gericht JOIN gericht_hat_kategorie ON gericht.id = gericht_hat_kategorie.gericht_id

JOIN kategorie ON gericht_hat_kategorie.kategorie_id = kategorie.id

GROUP BY kategorie.name

ORDER BY anzahl ASC;

name	anzahl
Vorspeisen	3
Desserts	3
Hauptspeisen	8

SELECT kategorie.name, COUNT(gericht.name) AS anzahl
FROM gericht
JOIN gericht_hat_kategorie ON gericht.id =
gericht_hat_kategorie.gericht_id
JOIN kategorie ON gericht_hat_kategorie.kategorie_id = kategorie.id
GROUP BY kategorie.name
HAVING anzahl > 2
ORDER BY anzahl ASC;



Aufgabe 7:

ALTER TABLE gericht
 ADD PRIMARY KEY (id)

ADD UNIQUE (name)
ADD CHECK (preis_intern <= preis_extern);

ALTER TABLE allergen

ADD PRIMARY KEY (code);

ALTER TABLE kategorie

ADD PRIMARY KEY (id);

ALTER TABLE gericht_hat_kategorie

ADD FOREIGN KEY (gericht_id) REFERENCES gericht(id)

ADD FOREIGN KEY (kategorie_id) REFERENCES kategorie(id);

ALTER TABLE gericht_hat_allergen

ADD FOREIGN KEY (gericht_id) REFERENCES gericht(id)

ADD FOREIGN KEY (code) REFERENCES allergen(code);

Aufgabe 11:

Warum das Speichern in einfachen Dateien (ohne Einsatz von Datenbanken) direkt aus der Anwendung heraus Probleme bereiten kann?

- Datenbanken können große Dateien behandeln
- Datenbanken ist durchsuchbar und sortierbar, daher kann man die Daten einfacher finden
- Man kann Daten gleichzeitig lesen und modifizieren
- Datenbanken sind skalierbar und effizient, da es an einem Ort gespeichert ist

Meilenstein	Aufgabe	Geschätzte Zeit (Minuten)	Tatsächliche Zeit (Minuten)
4-W1	1	30	37
	2	50	45
	4	30	49
	5	30	21
4-W2	6	30	16
	7	90	74
	8	60	51
Total:			293

Aufgabe 1.6):

SELECT * FROM wunschgericht ORDER BY nummer DESC LIMIT 5;

Aufgabe 2):

```
$name = str_replace('\'',"",trim($_POST['Name']??NULL));
$email = str_replace('\'',"",trim($_POST['E-Mail']??NULL));
```

str_replace() ist eine eingebaute Funktion in PHP und wird verwendet, um alle Vorkommen des Suchstrings oder Arrays von Suchstrings durch Ersatz-Strings oder Arrays von Ersatz-Strings in dem angegebenen String bzw. Array zu ersetzen.

Aufgabe 4: (*CONSTRAINTS waren erstmal gelöscht)

- ALTER TABLE gericht_hat_kategorie
 ADD CONSTRAINT uq_gericht_hat_kategorie
 UNIQUE(gericht_id, kategorie_id);
- 2) ALTER TABLE gericht ADD INDEX name_asc (name ASC); => (alphabetisch den Name suchen)
- 3) ALTER TABLE gericht_hat_kategorie ADD FOREIGN KEY (gericht_id) REFERENCES gericht(id) ON DELETE CASCADE:

ALTER TABLE gericht_hat_allergen

ADD FOREIGN KEY (gericht_id) REFERENCES gericht(id) ON DELETE

CASCADE; // RESTRICT erlaubt nicht

*Beim Löschen des Parents (gericht(id)) wird das Child (gericht_id) automatisch gelöscht.

4) ALTER TABLE gericht_hat_kategorie ADD FOREIGN KEY (gericht_id) REFERENCES gericht(id) ON DELETE SET NULL;

ALTER TABLE kategorie
ADD FOREIGN KEY (id) REFERENCES
gericht hat kategorie(kategorie id) ON DELETE SET NULL;

ALTER TABLE kategorie

ADD FOREIGN KEY (eltern_id) REFERENCES kategorie(id) ON DELETE

SET NULL;

DELETE FROM gericht;

DELETE FROM gericht_hat_kategorie WHERE (COUNT(gericht_id IS NULL) = COUNT(kategorie_id = x)) AND kategorie(eltern_id) IS NOT NULL;

- 5) ALTER TABLE gericht_hat_allergen ADD FOREIGN KEY (code) REFERENCES allergen(code) ON UPDATE CASCADE;
 - *Bei Änderung des Parents (gericht(id)) wird das Child (gericht_id) automatisch geändert.
- 6) ALTER TABLE gericht_hat_kategorieADD PRIMARY KEY (gericht_id, kategorie_id);

Meilenstein	Aufgabe	Geschätzte Zeit (Minuten)	Tatsächliche Zeit (Minuten)
5-W1	1	120	183
	2	30	17
5-W2	3	45	31
	4	45	35
	5	50	45
-	Gesamt	-	311 (5,18)

Aufgabe 5c)

```
DELIMITER//
CREATE PROCEDURE LoginTime (IN login_id INTEGER, IN error BOOLEAN)
BEGIN

DECLARE datum DATE DEFAULT now()
UPDATE benutzer
IF !error THEN
SET letzteanmeldung = datum;
ELSE
SET letzerfehler = datum;
END IF;
WHERE id = login_id;
END//
DELIMITER;
```

Eine Prozedur ist ein vorbereiteter SQL-Code, den wir speichern können, sodass der Code immer wieder verwendet werden kann. In diesem Fall machen wir eine

LoginTime Prozedur, sodass bei jeder Anmeldung die Zeit der letzten Anmeldung oder des letzten Fehlers aktualisiert wird.

```
DELIMITER//
CREATE PROCEDURE KategorieNameAsc ()
BEGIN
SELECT name
FROM kategorie
ORDER BY name ASC;
END//
DELIMITER;
```