Klausur
„Entwicklung webbasierter Anwendungen“
Sommersemester 2025

Nachname	**Vorname**	**Matrikelnummer**	**Note**

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**
Punkte	20	15	15	15	15	20
Ihre Punkte						

Hinweise

- **Bearbeitungszeit:** 90 Minuten
- **Erlaubte Hilfsmittel:** Ein beidseitig handbeschriebenes DIN-A4-Blatt

- **Lesen** Sie die gesamte Klausur vollständig, bevor Sie mit der Bearbeitung beginnen.
- Achten Sie auf **Vorder- und Rückseite**!
- **Schreiben** Sie klar und deutlich. Unleserliche Antworten werden nicht bewertet!
- Bitte achten Sie auf präzise und knappe Formulierungen.
- Verwenden Sie für Ihre **Lösungen** die vorgesehenen Antwortfelder. Sollte Ihnen der Platz nicht reichen, nutzen Sie das Zusatzpapier auf der letzten Seite.
- **Codequalität:** Schreiben Sie wartbaren und standardkonformen Code, der den Vorgaben der Veranstaltung entspricht.
- Es werden ausschließlich **Inhalte** als richtig bewertet, die im Rahmen der Veranstaltung behandelt oder besprochen wurden.

- **Abgabehinweis:** Die Klammerung der Aufgabenblätter darf nicht entfernt werden. Es liegt in Ihrer Verantwortung sicherzustellen, dass alle Blätter vollständig abgegeben werden. Im Falle eines versehentlichen Lösens der Klammerung beschriften Sie alle Blätter mit ihrem Namen.

Die Aufgabe allgemein: „H_DA – TODO“
Gegenstand dieser Klausur ist die Umsetzung ausgewählter Teilfunktionen einer To-Do-Webanwendung samt minimalistischem Dashboard.

Für die technische Realisierung kommen PHP und MySQL im Backend sowie HTML, CSS und JavaScript im Frontend zum Einsatz. Es sollen keine Frameworks verwendet werden.

Im Anschluss folgen zusätzlich einige Wissensfragen zu grundlegenden Webtechnologien.

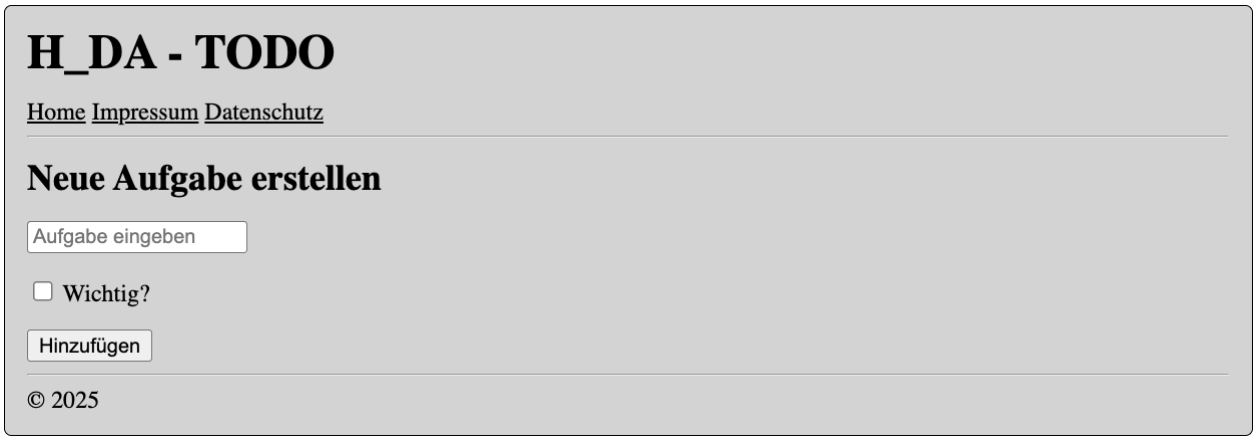

Abbildung 1: Layout

Datenbankstruktur
Die folgende Abbildung zeigt die Struktur unserer Datenbank. Sie bildet die Grundlage für alle nachfolgenden Aufgaben und Abfragen.

- Die Datenbank heißt hda-todos
- Die Tabelle heißt aufgabe.

#	Name	Typ	Kollation	Attribute	Null	Standard
1	**id** 🔑	int(11)			Nein	*kein(e)*
2	**beschreibung**	text	utf8mb4_general_ci		Nein	*kein(e)*
3	**wichtig**	tinyint(1)			Nein	*kein(e)*
4	**status**	text	utf8mb4_general_ci		Nein	*kein(e)*

id	beschreibung	wichtig	status
1	Essen kaufen	1	todo
2	Auto waschen	0	in-progress
3	Für EWA lernen	1	done

Abbildung 2: Datenbankstruktur mit Beispieldaten

Aufgabe 1: HTML (20 Punkte)

Setzen Sie das Layout gemäß der Abbildung 1 als **semantisches HTML** um. Kein PHP!

- Der Titel der Seite lautet: H_DA – Todo. Die JS und CSS-Dateien heißen: Exam.js und Exam.css
- Der <head> soll so ergänzt werden, dass die Seite auf mobilen Geräten korrekt dargestellt wird.
- Mögliche Formulare sollen an Exam.php geschickt werden.
- Das Formular darf nur abgeschickt werden, wenn das Feld „Aufgabe eingeben“ ausgefüllt ist.
- Beim Absenden des Formulars soll der Status der Aufgabe den Wert „todo“ haben – dieser Wert soll automatisch mitgeschickt werden, ohne dass der Benutzer ihn sieht oder eingeben muss.
- **Hinweis:** Die Formulardaten werden in Aufgabe 2 verarbeitet.

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>H_DA – Todo</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="Exam.css">
    <script src="Exam.js"></script>
</head>
<body>
    <div class="container">
        <header>
            <h1>H_DA – TODO</h1>
            <nav>
                <a href="#">Home</a>
                <a href="#">Impressum</a>
                <a href="#">Datenschutz</a>
            </nav>
        </header>
        <hr>
        <section>
            <h2>Neue Aufgabe erstellen</h2>
            <form action="Exam.php" method="post">
                <p>
                    <input type="text" name="todo" placeholder="Aufgabe eingeben" required>
                </p>
                <p>
                    <input type="checkbox" id="important" name="wichtig">
                    <label for="important">Wichtig?</label>
                </p>
                <p>
                    <input type="hidden" name="status" value="todo">
                </p>
                <button type="submit">Hinzufügen</button>
            </form>
        </section>
        <hr>
        <footer>
            &#xA9; 2025
        </footer>
    </div>
</body>
</html>
```

Aufgabe 2: PHP (15 Punkte)

Sie möchten eine neue Aufgabe hinzufügen. Schreiben Sie den **PHP-Code**, der die übermittelten Formulardaten aus Aufgabe 1 mithilfe eines vorhandenen Datenbankobjekts (Siehe Anhang: Page.php) in die Datenbank speichert. Achten Sie dabei auf eine sinnvolle Fehlerbehandlung, Prüfung eingehender Formulardaten und unsere Seitenklassenstruktur. **Hinweise:** Eine Checkbox übermittelt ihren Wert nur, wenn sie aktiviert wurde. Es reicht, wenn Sie hier nur die benötigte Methode unserer Seitenklassen niederschreiben der Rest wird als gegeben betrachtet.

```
protected function processReceivedData():void {
    if (isset($_POST['beschreibung']) && isset($_POST['status'])) {
        $beschreibung = $this->_database->real_escape_string($_POST['beschreibung']);
        $status = $this->_database->real_escape_string($_POST['status']);

        if (isset($_POST['wichtig'])) {
            $wichtig = 1;
        } else {
            $wichtig = 0;
        }

        $SQLabfrage = "INSERT INTO aufgabe (beschreibung, wichtig, status)
        VALUES ('$beschreibung', '$wichtig', '$status')";
        $this->_database->query($SQLabfrage);

        if ($this->_database->query($SQLabfrage) === true) {
            header('Location: Exam.php');
            exit();
        } else {
            echo "Datenbankfehler: " . $this->_database->error;
            // hier reicht einfach nur irgendein try and catch oder error ausgabe!
        }
    }
}
```

ACHTUNG: Alternativ mit Prepared Statments auch ok, dann gibt's die Punkte von real_escape_string dafür!

Aufgabe 3: PHP und JavaScript (15 Punkte)

Angenommen, es gibt ein Dashboard, das alle Aufgaben als eine unsortierte Liste anzeigt und sich alle drei Sekunden per clientseitigem Polling aktualisiert.
Das Backend liefert hierfür alle Aufgaben als JSON-String, den das Frontend per XMLHttpRequest anfragt und anschließend darstellt. Das Backend erwartet den Status der Aufgabe als Parameter.
Vervollständigen Sie folgenden PHP- und JavaScript-Code. Vorder- und Rückseite beachten!

```
// Backend ExamAPI.php
<?php declare(strict_types=1);

require_once './Page.php';

class ExamAPI extends Page {
    protected string $status = '';

    protected function getViewData():array {
        if($this->status) {
            $status = $this->_database->real_escape_string($this->status);

            $sql = "SELECT * FROM aufgabe WHERE status='" . $status . "'";
        }

        $recordset = $this->_database->query($sql);
        if (!$recordset) {
            throw new Exception("Abfrage fehlgeschlagen: " . $this->_database->error);
        }

        $result = array();
        $record = $recordset->fetch_assoc();
        while ($record) {
            $result[] = $record;
            $record = $recordset->fetch_assoc();
        }

        $recordset->free();
        return $result;
    }

    protected function generateView():void {
        $data = $this->getViewData();

        header("Content-Type: application/json; charset=UTF-8");
        $serializedData = json_encode($data);
        echo $serializedData;
    }

    protected function processReceivedData():void {
        if(isset($_GET['status'])) {

            $this->status = $_GET['status'];
        }
    }
    public static function main():void {
        try {
            $page = new ExamAPI();
            $page->processReceivedData();
            $page->generateView();
        } catch (Exception $e) {
            header("Content-type: text/html; charset=UTF-8");
            echo $e->getMessage();
        }
    }
}

ExamAPI::main();
```

```
// Frontend Exam.js

let request = new XMLHttpRequest();
function requestData() {
        request.open("GET", "ExamAPI.php?status=done");
        request.onreadystatechange = processData;
        request.send(null);
}
function processData() {
    if (request.readyState !== 4) return;
    if (request.status === 200 && request.responseText) {
        process(request.responseText);
    } else {
        console.error("Fehler beim Laden der Daten");
    }
}
function process(data) {
    // Dieses <ul>-Element wird als gegeben getrachtet
    let ul = document.getElementById("todo-list");
    let todos = JSON.parse(data);
    ul.innerHTML = "";
    for (let i = 0; i < todos.length; i++) {
        let todo = todos[i];
        // Erstellt ein neues <li>- Element
        let li = document.createElement("li");
        // textContent mögliche Beispiel-Ausgabe “Essen kaufen: todo”
        li.textContent = todo.beschreibung + " " + todo.status.
        ul.appendChild(li);
    }
}
document.addEventListener("DOMContentLoaded", function() {
    window.setInterval(requestData, 3000);
});
```

Aufgabe 4: CSS (15 Punkte)

Setzen Sie das folgende Layout aus Abbildung 3 mit **HTML, CSS und Flexbox** um.
Das vollständige HTML-Grundgerüst ist nicht erforderlich.

- Die Boxen sind 200px breit und 150px hoch.
- Die Root-Element Schriftgröße beträgt 30px.
- Die Schriftgröße in den Boxen beträgt das 1,5-Fache der Schriftgröße des Root-Elements.
- Die Hintergrundfarbe ist gray und die der Boxen lightgray.
- Der freie Platz wird gleichmäßig zwischen den Boxen 1, 2 und 3 verteilt.
- Der Text in den Boxen ist horizontal und vertikal zentriert.
- Bis zu einer Bildschirmbreite von 314px sind die Boxen untereinander, ansonsten nebeneinander.
- Berücksichtigen Sie auch die „ungefähren“ Abstände gemäß dem CSS-Boxmodell.

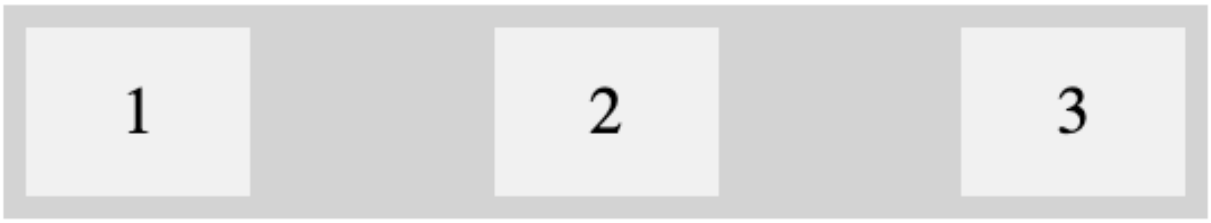

Abbildung 3: Zufälliges CSS-Layout

```
<style>
    html {
        font-size: 30px; // hier auch *, body oder :root ok!
    }

    .flex-container {
        display: flex;
        justify-content: space-between; // hier auch around, evenly etc. ok
        font-size: 1.5rem;
        background-color: gray;
    }

    .flex-item {
        display: flex;
        justify-content: center;
        align-items: center;
        background-color: lightgrey;
        width: 200px;
        height: 150px;
        margin: 10px; // oder padding auf container
    }

    @media (max-width: 314px) {
        .flex-container {
            flex-direction: column; // hier auch andersrum ok mit min-width etc.
        }
    }
</style>

<div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
</div>
```

Aufgabe 5: Alles zusammen (15 Punkte)

A) Der folgende HTML-Code soll beim Klicken auf den Button die ID einer Aufgabe mit einem Formular an den Endpunkt update.php senden, sodass dieses dort in der Datenbank als erledigt aktualisiert werden kann. **Korrigieren Sie alle Fehler. (5 Punkte)**

```
<form action="update.php" method="post">
    <p>Todo: Auto waschen (in-progress)</p>
    <input type="hidden" id="myid" value="2" name=”myid”> </input>
    <button type="submit">Als erledigt markieren</button>
</form>
```

B) Das folgende CSS beschreibt das grobe Styling des Elements mit der CSS-Klasse container, in dem sich der Seiteninhalt befindet (siehe Abbildung 1). Außerdem soll sichergestellt werden, dass ein p-Tag, der das letzte Kindelement eines Containers ist, keinen unteren Außenabstand mehr besitzt. **Korrigieren Sie alle Fehler. (5Punkte)**

```
PunktStatt#container {
    color background: lightgray;
    border styling: 1px solid black;
    border-radius: 5px;
    padding: 1rem; // 1 x root font-size
    margin-left: auto;
    margin-right: auto;
    max-width: 800px;
}

/* Jeder p-Tag, der das letzte Kindelement ist, soll keinen Abstand nach unten haben */
PunktStatt#container > p:last-child {
    margin-bottom: 0;
}
```

C) Der folgende PHP-Code soll die Beschreibung einer Aufgabe in der Session speichern. **Korrigieren Sie alle Fehler. (5 Punkte)**

```
<?php
session_start();
$_SESSION['todo-description'] => 'Für EWA lernen';
```

Aufgabe 6: Wissensfragen (20 Punkte)

Frage 1: Welchen HTTP-Request sollte man verwenden, um Daten von einer API abzurufen, und warum? **(5 Punkte)**

GET 2

Idempotent: Mehrfache Anfragen haben denselben Effekt - sie verändern nichts. 3

oder

Query-Parameter erwähnt für paar extra Punkte.

Frage 2: Welche Rolle spielen Cookies im Zusammenhang mit serverseitigen Sessions in PHP, und wie wird die Session-ID hier verwendet? **(5 Punkte)**

Session-ID wird im Cookie gespeichert. 5

Nur Session/Cookie erklärt? 1 – 3

Frage 3: Welche Aussagen zu Cross-Site-Scripting (XSS) und SQL-Injektion sowie deren Schutzmaßnahmen in PHP sind korrekt? **Markieren Sie alle korrekten Aussagen. Punkte gibt es nur bei vollständiger und fehlerfreier Auswahl. (5 Punkte)**

A) Cross-Site-Scripting (XSS) ermöglicht es Angreifern, schädlichen JavaScript-Code in Webseiten einzuschleusen, der im Browser anderer Nutzer ausgeführt wird.

B) Die Funktion htmlspecialchars() schützt vor SQL-Injektionen, indem sie spezielle SQL-Zeichen entschärft.

C) SQL-Injektionen können durch real_escape_string() oder durch Prepared Statements verhindert werden.

D) Die Funktion htmlspecialchars() verhindert, dass Benutzereingaben als HTML oder JavaScript interpretiert werden, und schützt somit vor XSS.

E) XSS kann vollständig verhindert werden, indem man real_escape_string() auf Benutzereingaben anwendet.

5

Frage 4: Welche Werkzeuge bietet uns CSS und HTML zur Umsetzung von Responsive Design? **Markieren Sie alle korrekten Aussagen. Punkte gibt es nur bei vollständiger und fehlerfreier Auswahl. (5 Punkte)**

A) Media Queries zur Anpassung von Styles abhängig von Bildschirmgröße oder Gerätetyp

B) Flexbox zur flexiblen Anordnung und Verteilung von Elementen im Layout

C) Das <picture>-Element für die Auswahl unterschiedlicher Bilder je nach Bildschirmgröße

D) Die Verwendung von relativen CSS-Einheiten.

E) Die CSS-Regel responsive: true; zur Aktivierung von Responsive Design

5

Anhang: Basisklasse - Page.php

```php
<?php declare(strict_types=1);
abstract class Page
{
    protected MySQLi $_database;

    protected function __construct()
    {
        error_reporting(E_ALL);

        $host = "localhost";

        if (gethostbyname('mariadb') != "mariadb") {
            $host = "mariadb";
        }

        $this->_database = new MySQLi($host, "public", "public", "hda-todos");

        if (mysqli_connect_errno()) {
            throw new Exception("Connect failed: " . mysqli_connect_error());
        }

        if (!$this->_database->set_charset("utf8")) {
            throw new Exception($this->_database->error);
        }
    }

    public function __destruct()
    {

    }

    protected function generatePageHeader():void
    {

    }

    protected function generatePageFooter():void
    {

    }

    protected function processReceivedData():void
    {

    }
}
```

Zusatzpapier (Falls Sie zu einer Aufgabe etwas ergänzend erwähnen wollen)