# LU00f - LB02 Inhalt

Bei der LB2 im Modul 307 wollen wir uns mit den CRUD-Operationen, die Sie im Unterricht kennen gelernt haben, beschäftigen. Dabei steht das Acronym CRUD für die vier im Web gängigen Operationen:

- **C**rud = Create, Neuanlegen
- c**R**ud = Read, Einzelansicht oder Listenansicht
- cr**U**d = Update, Bearbeitung von einzelen Datensätzen
- cru**d** = Delete, Löschen von Datensätzen

Diese Operationen sollen, gemäss Input aus dem Unterricht, via HTML, JavaScript under der Middleware Express (Server in JS) umgesetzt werden. Ihre Lehrperson wird Ihrem Team eine der nachfolgenden LB2-Fokusthemen zuordnen.

## 1. Freunde

### 1.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung einer persönlichen Freundesliste. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (friends.json) gespeichert und von dort gelesen.

### 1.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei friends.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Freunde werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, Alter, Wohnort).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Freund mit Vorname, Nachname, Alter und Wohnort erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 1.3 Technische Anforderungen

- Die Datei friends.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird für die Auflistung, zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

# 2: Musiktitel

## 2.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung einer persönlichen Musiktitel-Sammlung. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (songs.json) gespeichert und von dort gelesen.

## 2.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei songs.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Musiktitel werden beim Laden der Seite als Liste angezeigt (Titel, Interpret, Album, Erscheinungsjahr).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Musiktitel mit Titel, Interpret, Album und Erscheinungsjahr erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

## 2.3 Technische Anforderungen

- Die Datei songs.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Titel und Interpret) sind mit einer einfachen Validierung zu versehen.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

# 3: Subscriptions

## 3.1 Ausgangslage

Streaming-Dienste und digitale Abonnements (z. B. Netflix, Spotify, Disney+) gehören heute zum Alltag. Sie erstellen eine Webanwendung zur übersichtlichen Verwaltung solcher Abonnements. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (subscriptions.json) gespeichert und von dort gelesen.

## 3.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei subscriptions.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Abonnements werden beim Laden der Seite als Liste

angezeigt (Dienst, Kategorie, monatliche Kosten, Startdatum).
2. Hinzufügen (Create): Über ein Formular kann ein neues Abonnement mit Dienstname, Kategorie (z. B. Video, Musik, Software), monatlichen Kosten und Startdatum erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 3.3 Technische Anforderungen

- Die Datei subscriptions.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. Netflix, Spotify, Adobe Creative Cloud).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Dienstname und Kosten) sind mit einer einfachen Validierung zu versehen; die Kosten dürfen keinen negativen Wert annehmen.
- Die Gesamtkosten aller aktiven Abonnements pro Monat werden unterhalb der Liste berechnet und angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

## 4. Fuhrpark

### 4.1 Ausgangslage

Unternehmen verwalten ihre Fahrzeuge häufig in einfachen Listen oder Tabellen. Sie erstellen eine Webanwendung zur digitalen Verwaltung eines Fuhrparks. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Fahrzeugdaten werden in einer lokalen JSON-Datei (vehicles.json) gespeichert und von dort gelesen.

### 4.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei vehicles.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle Fahrzeuge des Fuhrparks werden beim Laden der Seite als Liste angezeigt (Kennzeichen, Marke, Modell, Baujahr, Status).
2. Hinzufügen (Create): Über ein Formular kann ein neues Fahrzeug mit Kennzeichen, Marke, Modell, Baujahr und Status (verfügbar / in Betrieb / in Wartung) erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Fahrzeugeintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Fahrzeugeintrag kann über eine Schaltfläche aus der Liste entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 4.3 Technische Anforderungen

- Die Datei vehicles.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. Lieferwagen, Personenwagen, Transporter).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Kennzeichen und Marke) sind mit einer einfachen Validierung zu versehen; das Kennzeichen darf nicht doppelt vorkommen.
- Die Anzahl der Fahrzeuge je Status (verfügbar / in Betrieb / in Wartung) wird unterhalb der Liste angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

## 5. BZZ-Zimmerverwaltung

### 5.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Schulzimmern. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (rooms.json) gespeichert und von dort gelesen.

### 5.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei rooms.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Schulzimmer werden beim Laden der Seite als Liste angezeigt (Zimmernummer, Stockwerk, Kapazität, Ausstattung).
2. Hinzufügen (Create): Über ein Formular kann ein neues Schulzimmer mit Zimmernummer, Stockwerk, Kapazität und Ausstattung erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 5.3 Technische Anforderungen

- Die Datei vehicles.json enthält zu Beginn mindestens 3 Einträge als JSON-Array (z. B. Lieferwagen, Personenwagen, Transporter).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder (Kennzeichen und Marke) sind mit einer einfachen Validierung zu versehen; das Kennzeichen darf nicht doppelt vorkommen.
- Die Anzahl der Fahrzeuge je Status (verfügbar / in Betrieb / in Wartung) wird unterhalb der Liste angezeigt.
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

# 6. EFZ/BMS/ABU-Zeugnis

## 6.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von EFZ-Notenausweisen für Mediamatiker/innen. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (zeugnisse.json) gespeichert und von dort gelesen.

## 6.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei zeugnisse.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Notenausweise werden beim Laden der Seite als Liste angezeigt (Name, Vorname, Lehrjahr, Gesamtnote, Bestanden/Nicht bestanden).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Notenausweis mit Name, Vorname, Lehrjahr sowie den vier Qualifikationsbereichsnoten (VPA, Mediamatikkompetenzen, Allgemeinbildung, Betrieb) erfasst werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus**: Gesamtnote berechnen: Die Gesamtnote wird automatisch aus den vier Qualifikationsbereichsnoten gemäss der offiziellen Gewichtung berechnet (VPA 30%, Mediamatikkompetenzen 30%, Allgemeinbildung 10%, Betrieb 20%, Erweiterte Grundkompetenzen 10%) und in der Liste angezeigt.
6. Filtern: Es soll nach allen vier Kriterien gefiltert werden können.

## 6.3 Technischen Anforderungen

- Die Datei zeugnisse.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: name, vorname, lehrjahr, noteVPA, noteMediamatik, noteAGB, noteBetrieb (alle als Zahl, halbe Noten erlaubt).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, Noten zwischen 1.0 und 6.0).
- **Bonus:** Ein Eintrag wird in der Liste farblich hervorgehoben, wenn er nicht bestanden ist (Gesamtnote < 4.0).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

# 7. Kundenverwaltung

## 7.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Kunden. Die Anwendung besteht aus

einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (kunden.json) gespeichert und von dort gelesen.

### 7.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei kunden.json mit folgenden Funktionen:

- Anzeigen (Read): Alle gespeicherten Kunden werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, E-Mail, Telefon, Ort).
- Hinzufügen (Create): Über ein Formular kann ein neuer Kunde mit Vorname, Nachname, E-Mail, Telefon und Ort erfasst und der Liste hinzugefügt werden.
- Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
- Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
- Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 7.3 Technischen Anforderungen

Technische Anforderungen

- Die Datei kunden.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, gültige E-Mail-Adresse).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

## 8. Lehrbetriebe

### 8.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Lehrbetrieben. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (lehrbetriebe.json) gespeichert und von dort gelesen.

### 8.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei lehrbetriebe.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Lehrbetriebe werden beim Laden der Seite als Liste angezeigt (Firmenname, Branche, Kontaktperson, Telefon, Ort).
2. Hinzufügen (Create): Über ein Formular kann ein neuer Lehrbetrieb mit Firmenname, Branche, Kontaktperson, Telefon und Ort erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und

gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 8.3 Technischen Anforderungen

- Die Datei lehrbetriebe.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Firmenname, kein leere Kontaktperson).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).

## 9. Staatsoberhäupter & Prime-Minister

### 9.1 Ausgangslage

Sie erstellen eine einfache Webanwendung zur Verwaltung von Staatsoberhäuptern und Präsidenten. Die Anwendung besteht aus einer einzigen HTML-Seite mit eingebettetem JavaScript. Alle Daten werden in einer lokalen JSON-Datei (staatsoberhaupter.json) gespeichert und von dort gelesen.

### 9.2 Aufgabenstellung

Erstellen Sie eine HTML-Seite index.html sowie eine Datendatei staatsoberhaupter.json mit folgenden Funktionen:

1. Anzeigen (Read): Alle gespeicherten Staatsoberhäupter werden beim Laden der Seite als Liste angezeigt (Vorname, Nachname, Land, Titel, Amtsantritt).
2. Hinzufügen (Create): Über ein Formular kann ein neues Staatsoberhaupt mit Vorname, Nachname, Land, Titel (z. B. Präsident, König, Bundeskanzler) und Amtsantritt (Jahr) erfasst und der Liste hinzugefügt werden.
3. Bearbeiten (Update): Ein bestehender Eintrag kann ausgewählt, im Formular bearbeitet und gespeichert werden.
4. Löschen (Delete): Jeder Listeneintrag kann über eine Schaltfläche entfernt werden.
5. **Bonus**Amtsdauer berechnen: Die bisherige Amtsdauer in Jahren wird automatisch aus dem Amtsantritt berechnet und in der Liste angezeigt.
6. Filtern: Es soll nach allen Kriterien gefiltert werden können.

### 9.3 Technische Anforderungen

- Die Datei staatsoberhaupter.json enthält zu Beginn mindestens 3 Einträge als JSON-Array.
- Jeder Eintrag enthält mindestens: vorname, nachname, land, titel, amtsantritt (als vierstellige Jahreszahl).
- Das Formular wird zur Eingabe und zur Bearbeitung verwendet (kein separates Formular).
- Pflichtfelder sind mit einer einfachen Validierung zu versehen (kein leerer Name, kein leeres Land, Amtsantritt muss eine gültige vierstellige Jahreszahl sein).
- Die Anwendung soll ohne externe Frameworks auskommen (kein jQuery, kein Bootstrap).