

# LB3 - Projektbeschreibungen

## Allgemeine Anforderungen an alle Projekte

Jedes Team wählt genau **einen** Use Case. Pro Klasse darf jeder Use Case **nur einmal** vergeben werden.

### Technische Minimalanforderungen

- Mindestens **2 Tabellen** mit einer **1:n-Beziehung**
- Mindestens **1 JOIN-Abfrage**
- Mindestens **1 Aggregatfunktion** (z.B. COUNT, AVG, MIN, MAX)
- Vollständige **CRUD-Operationen** auf der Haupttabelle:
  1. Create (POST)
  2. Read (GET – Liste + Detail)
  3. Update (PUT)
  4. Delete (DELETE)
- Umsetzung als **REST-API** mit Express und Zugriff auf MySQL via AppUser (nicht Root)
- Datenbank-Dump inkl. Struktur und Daten
- Test und Demonstration über **Postman**



## Projekt A - Reisedatenbank «Wo war ich schon?»

### Ausgangslage

Sie reisen gerne oder möchten Ihre vergangenen Reisen übersichtlich dokumentieren. Bisher liegen Informationen wie Zielort, Jahr, Kosten und Erinnerungen verstreut in Fotos, Notizen oder im Kopf. Eine kleine Reise-Datenbank soll helfen, alle besuchten Orte und Bewertungen zentral zu verwalten.

## Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
    - **Reise** (z.B. Stadt, Jahr, Kommentar, Sterne-Bewertung, Kosten)
    - **Land** oder **Kontinent** (z.B. Name, Kontinent, Kürzel)
  - Verknüpfen Sie Reise mit Land/Kontinent über einen **Fremdschlüssel** (1:n: ein Land – viele Reisen).
  - Legen Sie Startdaten an (z.B. mindestens 10 Reisen in verschiedene Länder/Kontinente).
  - Implementieren Sie CRUD-Routen für Reisen (z.B. /api/reisen).
  - Implementieren Sie mindestens:
    - eine Route mit **JOIN** (z.B. „Alle Reisen inkl. Ländernamen und Kontinent anzeigen“)
    - eine Route mit **Aggregatfunktion** (z.B. „Anzahl Reisen pro Kontinent“, „Durchschnittliche Bewertung pro Land“)
  - Optional erstellen Sie eine dritte Tabelle mit Angaben zu den Reisenden (Namen, Email-Adresse, ect.) und stellen Sie über einen Fremdschlüssel dar, wer von Ihnen auf welcher Reise war (darf fiktiv sein).



# **Projekt B - Lieblingsfilm-Datenbank**

## Ausgangslage

Sie schauen gerne Filme und möchten Ihre Lieblingsfilme inkl. Bewertungen systematisch erfassen. Film-Infos stehen zwar auf Plattformen wie IMDb oder Letterboxd, aber Ihre persönliche Sicht fehlt dort. Eine eigene Film-Datenbank soll diese Lücke schliessen.

## Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Film** (Titel, Erscheinungsjahr, Synopsis, Genre, IMDb/Letterboxd-Rating, eigene Sterne-Bewertung, Kommentar)
  - **Person** oder **Cast** (Name, Rolle-Typ: Schauspieler:in, Regie)
- Zwischen Film und Person besteht eine **n:m-Beziehung** (ein Film – mehrere Personen, eine Person – mehrere Filme). Minimal reicht eine 1:n-Beziehung (z.B. Film – Regisseur:in), wenn n:m zu komplex wird – alternativ arbeiten Sie mit einer Zwischentabelle.
- Legen Sie Daten für mehrere Filme mit unterschiedlichen Genres an.
- Implementieren Sie CRUD-Routen für Filme (z.B. /api/film).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Filme mit Regisseur:in anzeigen“)
  - eine Aggregat-Route (z.B. „Durchschnittliche eigene Bewertung pro Genre“, „Anzahl Filme pro Genre“)



## Projekt C - Lieblingsbuch-Datenbank

### Ausgangslage

Sie lesen gerne und möchten festhalten, welche Bücher Ihnen gefallen haben, welche Verlage sie herausgeben und wie Ihre persönliche Meinung dazu ist. Die Bewertungen auf Online-Shops reichen Ihnen dafür nicht.

## Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Buch** (Titel, Erscheinungsjahr, Kurzbeschreibung, Kategorie/Genre, eigene Sterne-Bewertung, Kommentar)
  - **Autor:in** (Name, Herkunft, ggf. Geburtsjahr)
- Beziehung: 1:n (eine Autor:in – viele Bücher).
- Fügen Sie optional eine Entität **Verlag** hinzu (Buch – Verlag als 1:n).

- Legen Sie Daten für mehrere Bücher und Autor:innen an.
- Implementieren Sie CRUD-Routen für Bücher (z.B. /api/buecher).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Bücher inkl. Autor:in und Verlag anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Bücher pro Kategorie“, „Durchschnittliche Bewertung pro Autor:in“)



## Projekt D - Lieblings-Kaffeehäuser im Kanton

### Ausgangslage

Sie sind gerne in Cafés unterwegs (z.B. im Kanton Zürich oder Ihrem Heimatkanton) und möchten Ihre Lieblingsorte mit Notizen und Bewertungen speichern. Online-Karten zeigen zwar Standorte, aber nicht Ihre persönliche Meinung.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Kaffeehaus** (Name, Adresse, Gemeinde, Lieblingsgetränk, eigene Sterne-Bewertung, Kommentar)
  - **Ort/Gemeinde/Stadtkreis** oder **Kanton** (Name, ggf. PLZ-Bereich)
- Beziehung: 1:n (eine Gemeinde/ein Kanton/ein Stadtkreis – viele Kaffeehäuser).
- Legen Sie Daten mit mehreren Kaffeehäusern in verschiedenen Gemeinden/Kantonen/Stadtkreisen an.
- Implementieren Sie CRUD-Routen für Kaffeehäuser (z.B. /api/kaffeehaeuser).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Kaffeehäuser inkl. Gemeindenname und Kanton anzeigen“)
  - eine Aggregat-Route (z.B. „Durchschnittliche Bewertung pro Gemeinde“, „Anzahl Kaffeehäuser pro Kanton“)



## Projekt E - Geburtstagsdatenbank für Freunde & Familie

### Ausgangslage

Sie möchten Geburtstage von Freund:innen und Familie im Griff haben und z.B. sehen, wer im gleichen Monat oder am gleichen Tag Geburtstag hat. Kalender-Apps können das teilweise, aber ohne zusätzliche Informationen zu Beziehungen oder Sternzeichen.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Person** (Name, Geburtsdatum, Beziehungstyp zu Ihnen, Kommentar)
  - **Sternzeichen** (Name, Zeitraum)
- Verbindung: 1:n oder 1:1 (jedem Geburtstag wird ein Sternzeichen zugeordnet).
- Legen Sie Daten mit verschiedenen Personen an (unterschiedliche Monate/Sternzeichen).
- Implementieren Sie CRUD-Routen für Personen (z.B. /api/personen).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Personen inkl. Sternzeichen anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Geburtstage pro Monat“, „Anzahl Personen pro Sternzeichen“)

Hinweise: Da es sich in diesem Case unter Umständen um personenbezogene Daten handelt und diese entsprechenden Datenschutz geniessen, können Sie hier auch mit fiktiven Namen und Daten arbeiten.



## Projekt F - (Fan-)Sport-Team-Datenbank

### Ausgangslage

Sie verfolgen ein oder mehrere Sportteams (Fussball, Eishockey, Volleyball etc.) und möchten Kader, Liga-Zugehörigkeit und Trainer:innen strukturiert erfassen. Online-Statistiken sind umfangreich, aber oft überladen für Ihre Zwecke.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Team** (Name, Sportart, Liga, Heimatort)
  - **Spieler:in** (Name, Position, Rückennummer)
- Beziehung: 1:n (ein Team – viele Spieler:innen).
- Optional: Entität **Trainer:in** oder **Saison**.
- Legen Sie Daten für mindestens 2-3 Teams mit mehreren Spieler:innen an.
- Implementieren Sie CRUD-Routen für Teams oder Spieler:innen (z.B. /api/teams).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Spieler:innen inkl. Team anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Spieler:innen pro Team“, „Anzahl Teams pro Liga“)



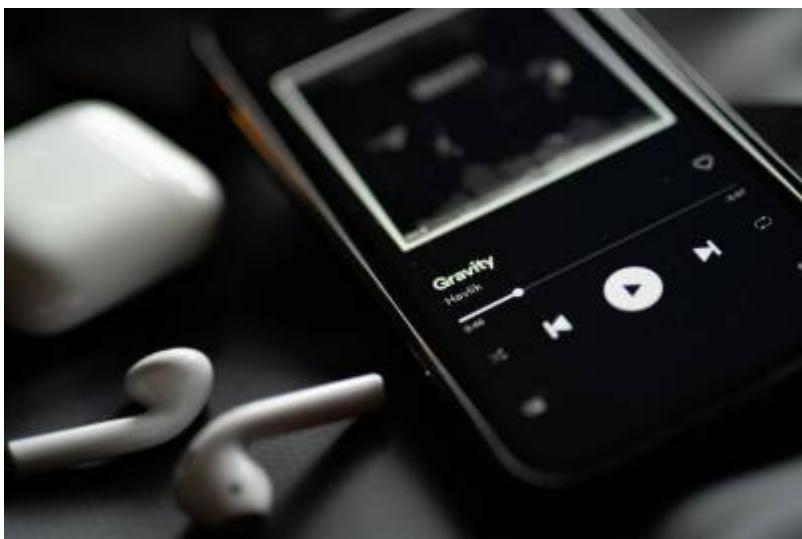
## Projekt G - Lieblingsmuseen in der Schweiz

### Ausgangslage

Sie besuchen gerne Museen und möchten eine Übersicht über Ihre Lieblingsmuseen in der Schweiz haben, inkl. Kategorie, Besucherzahlen und besonderen Ausstellungen. Tourismus-Webseiten liefern viele Infos, aber keine persönliche Filterung.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Museum** (Name, Ort, Kanton, Kategorie, Besucherzahl pro Jahr, berühmtestes Ausstellungsstück)
  - **Ausstellung** (Titel, Künstler:in, Start- und Enddatum)
- Beziehung: 1:n (ein Museum – viele Ausstellungen).
- Legen Sie Daten mit mehreren Museen und zugehörigen Ausstellungen an.
- Implementieren Sie CRUD-Routen für Museen oder Ausstellungen (z.B. /api/museen).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Ausstellungen inkl. Museumsname anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Ausstellungen pro Museum“, „Durchschnittliche Besucherzahl pro Kategorie“)



## Projekt H - Lieblingssong-Playlist

### Ausgangslage

Sie hören viel Musik und möchten Ihre Lieblingssongs mit Album, Artist und Genre verwalten. Streamingdienste haben zwar Playlists, aber keine eigene kleine Statistik über Ihre Lieblingsstücke.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Song** (Titel, Dauer, eigene Bewertung, Kommentar)
  - **Artist** (Name, Herkunft)
  - Optional: **Album** (Titel, Erscheinungsjahr)
- Beziehung: 1:n (ein Artist – viele Songs), ggf. 1:n (ein Album – viele Songs).
- Legen Sie Daten mit Songs verschiedener Artists und Genres an.
- Implementieren Sie CRUD-Routen für Songs (z.B. /api/songs).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Songs inkl. Artist- und Albumname anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Songs pro Genre“, „Durchschnittliche Bewertung pro Artist“)



## Projekt I - Kamera-Datenbank

### Ausgangslage

Sie interessieren sich für Fotografie oder Video und haben den Überblick über verschiedene Kamera-Modelle, Hersteller und Preisklassen verloren. Eine kleine Datenbank soll dabei helfen, Modelle zu vergleichen.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Kamera** (Modellname, Typ: Systemkamera/Video/Foto, Line: Consumer/Prosumer/Professional, Preis)
  - **Hersteller** (Name, Land)
- Beziehung: 1:n (ein Hersteller – viele Kameras).
- Legen Sie Daten mit mehreren Herstellern und Kamera-Modellen an.
- Implementieren Sie CRUD-Routen für Kameras (z.B. /api/kameras).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Kameras inkl. Herstellernamen anzeigen“)
  - eine Aggregat-Route (z.B. „Durchschnittlicher Preis pro Line“, „Anzahl Kameras pro Hersteller“)



## Projekt J - Lieblingsmodeschöpfer:innen

### Ausgangslage

Sie interessieren sich für Mode und möchten Designer:innen, deren Stil und Marken erfassen. Ziel ist eine kleine Übersicht über Lieblingsdesigner:innen und ihre bekanntesten Stücke.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Designer:in** (Name, Geburtsdatum, Stil-Beschreibung, Herkunftsland)
  - **Marke/Label** (Name, Sitz)
- Optional: Entität **Modestück** (Bezeichnung, Kategorie, Erscheinungsjahr, Beschreibung).
- Legen Sie Daten mit mehreren Designer:innen, Marken und ggf. Modestücken an.
- Implementieren Sie CRUD-Routen (z.B. für Designer:innen: /api/designer).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Designer:innen inkl. Marke anzeigen“)
  - eine Aggregat-Route (z.B. „Anzahl Designer:innen pro Land“, „Anzahl Modestücke pro Kategorie“)



## Projekt K - Lieblingsplätze in der Natur in der Schweiz

### Ausgangslage

Sie verbringen gerne Zeit in der Natur und möchten Ihre Lieblingsorte (Seen, Berge, Wälder etc.) dokumentieren. Neben der Position interessieren Sie auch Erreichbarkeit, Menschenmenge und persönliche Bewertungen.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Ort** (Name, Beschreibung, Kanton, Längen-/Breitengrad, Umgebungstyp: Berge/See/Wald etc., eigene Sterne-Bewertung, Kommentar)
  - **Anreise** (Verkehrsmittel: ÖV/Fuss/Fahrrad, Dauer, Kurzbeschreibung)
- Beziehung: 1:n (ein Ort - mehrere Anreisevarianten) oder 1:1 (ein Ort - eine typische Anreise).
- Legen Sie Daten mit mehreren Naturorten in verschiedenen Kantonen an.
- Implementieren Sie CRUD-Routen für Orte (z.B. /api/orte).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Ort inkl. typischer Anreise anzeigen“)
  - eine Aggregat-Route (z.B. „Durchschnittliche Bewertung pro Kanton“, „Anzahl Orte pro Umgebungstyp“)



## Projekt L - Lieblings-TV-Serien

### Ausgangslage

Sie schauen gerne Serien und möchten festhalten, welche davon Sie gesehen haben, wie viele Staffeln/Episoden es gibt und wo die Serie gestreamt werden kann. Große Datenbanken liefern viele Infos, aber nicht Ihre persönliche Bewertung.

### Vorgehen

- Modellieren Sie mindestens folgende Entitäten:
  - **Serie** (Titel, Synopsis, Genre, Studio, Streamingdienst/Sender, Anzahl Staffeln, Anzahl Episoden, eigene Bewertung, Kommentar)
  - **Schauspieler:in** (Name, ggf. Rolle in der Serie)
- Beziehung: 1:n oder n:m (eine Serie – viele Schauspieler:innen).
- Legen Sie Daten mit mehreren Serien unterschiedlicher Genres an.
- Implementieren Sie CRUD-Routen für Serien (z.B. /api/serien).
- Implementieren Sie:
  - eine JOIN-Route (z.B. „Alle Serien inkl. Schauspieler:innen anzeigen“ oder umgekehrt)
  - eine Aggregat-Route (z.B. „Durchschnittliche Bewertung pro Genre“, „Gesamtanzahl Episoden pro Streamingdienst“)

From:  
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:  
[https://wiki.bzz.ch/modul/m290\\_guko/leistungsbeurteilungen/03\\_lb/c\\_usecase?rev=1764532619](https://wiki.bzz.ch/modul/m290_guko/leistungsbeurteilungen/03_lb/c_usecase?rev=1764532619)

Last update: **2025/11/30 20:56**



