MARSKOLONIE LOGISTIK VERWALTUNG

WS 25/26

Agenda

1. Vorstellung der Idee
2. Case Study & Use Cases
3. Umsetzung & Verbesserungen
4. ERD & PDM
5. Demonstration via WebApp & SQL Developer
6. Selbstreflexion zum Projekt

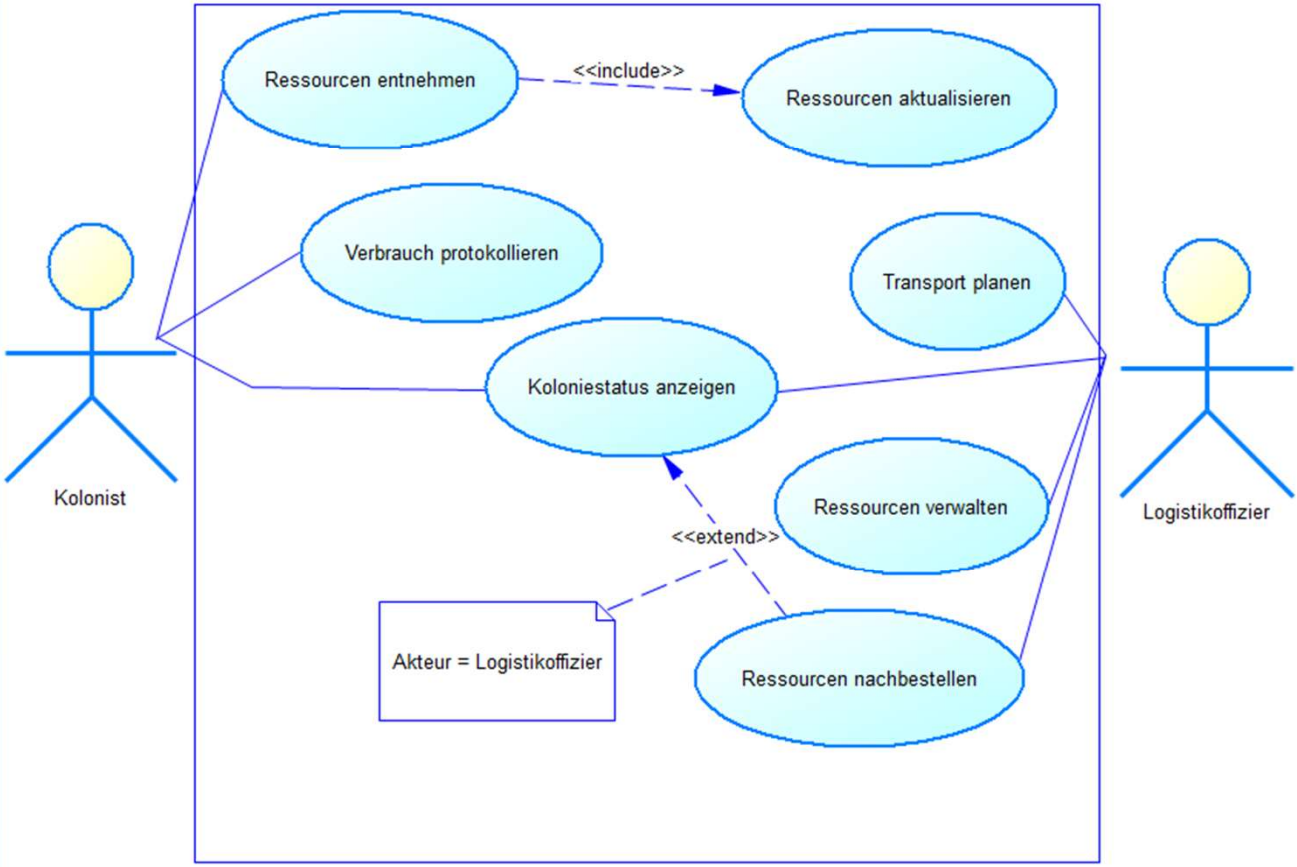
IDEE

IDEE

- Verwaltung von Städten auf dem Mars
    - Transportwege zwischen den Städten und Lager (einzelne Routen)
    - Bewohner der Städte
    - Ressourcengewinnung (z.B. Wasser) für die einzelnen Städte
    - Allgemeine Infos zur Logistikverwaltung
        - Fahrzeuge / Raumschiffe
        - Energiequellen
        - Mitarbeiter
            - Abteilung (auch Ressourcengewinnung)
            - Berufung


Case Study
&
Use Cases

Use Cases


Ressourcen entnehmen
<<include>>
Ressourcen aktualisieren
Verbrauch protokollieren
Transport planen
Koloniestatus anzeigen
Kolonist
Logistikoffizier
Ressourcen verwalten
<<extend>>
Akteur = Logistikoffizier
Ressourcen nachbestellen

1. Ausgangslage & Problemstellung

- 100% angewiesen auf lokale Vorräte (Sauerstoff, Wasser, Erze) und seltene Erd-Lieferungen.
- Fragmentierte Systeme und manuelle Erfassung führen zu Intransparenz.
- Lange Lieferzeiten (Erde-Mars) verhindern kurzfristige Korrekturen bei Fehlplanungen.

2. Die kritischen Risiken

- Unvorhergesehene Knappheit lebensnotwendiger Güter.
- Unnötige Überbestände blockieren begrenzten Lagerraum und binden wertvolle Arbeitskraft.

3. Zielsetzung des neuen Systems

- Datenbankgestützte Überwachung aller Warenbewegungen und Bestände.
- Strukturierte Erfassung von Bewohnern und Tätigkeiten zur optimierten Arbeitszeiteinteilung.
- Datengestützte Entscheidungsbasis zur Vermeidung existenzieller Versorgungsengpässe.

Umsetzung & Verbesserungen

Umsetzung & Verbesserungen

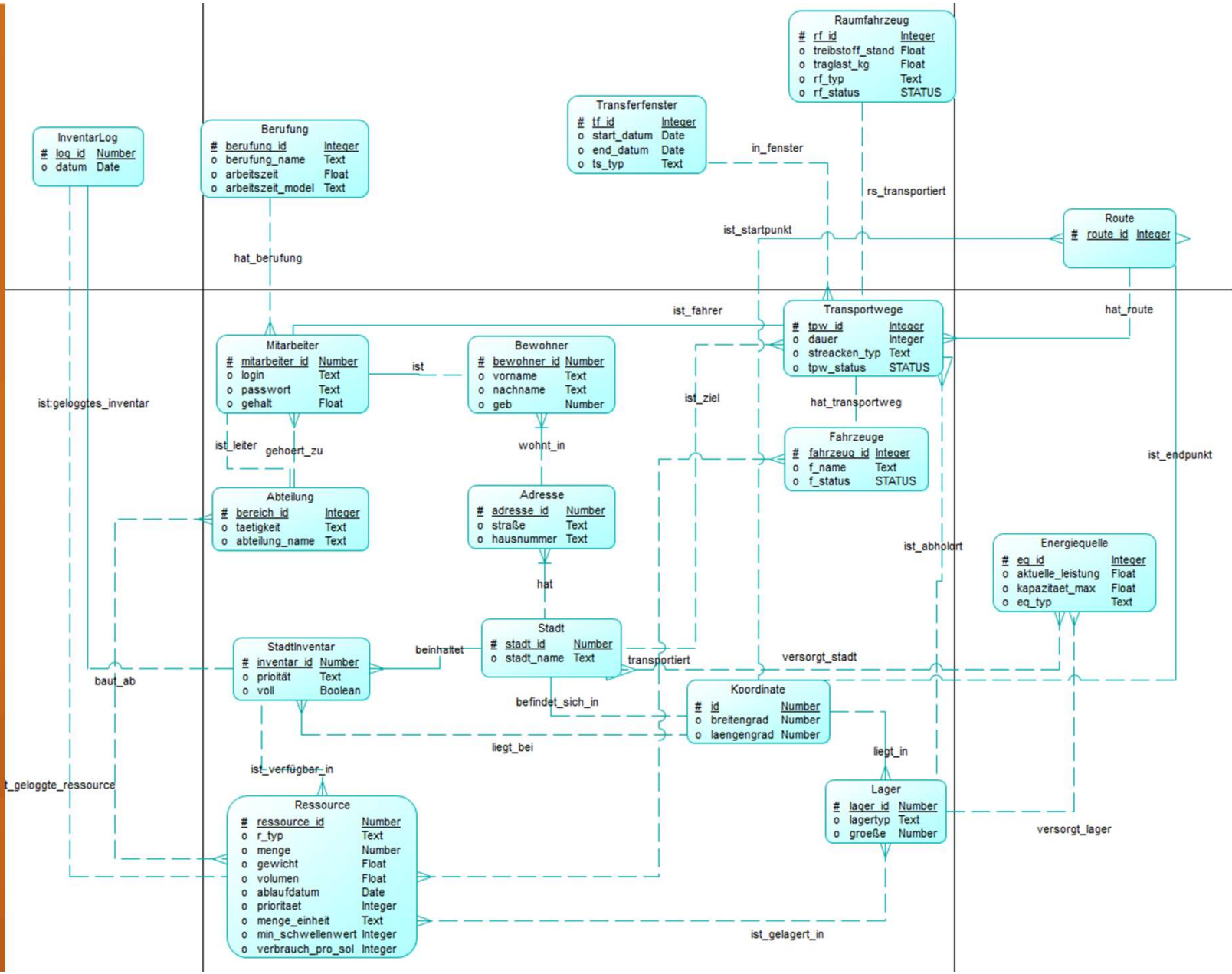
- Allgemein:
  - Vom Allgemeinen zum Speziellen: Begonnen als universelles Verwaltungstool, später gezielte Neuausrichtung auf Logistik.
  - Daten-Evolution: Erstellung einer ersten Grundstruktur (ERD), die nach dem Fokuswechsel komplett auf logistische Abläufe optimiert wurde.
  - Starke Vernetzung: Besonderes Augenmerk auf die Transport-Logik – alle Daten (Fahrzeuge, Routen, Waren) sind sauber und logisch miteinander verknüpft.

Umsetzung & Verbesserungen

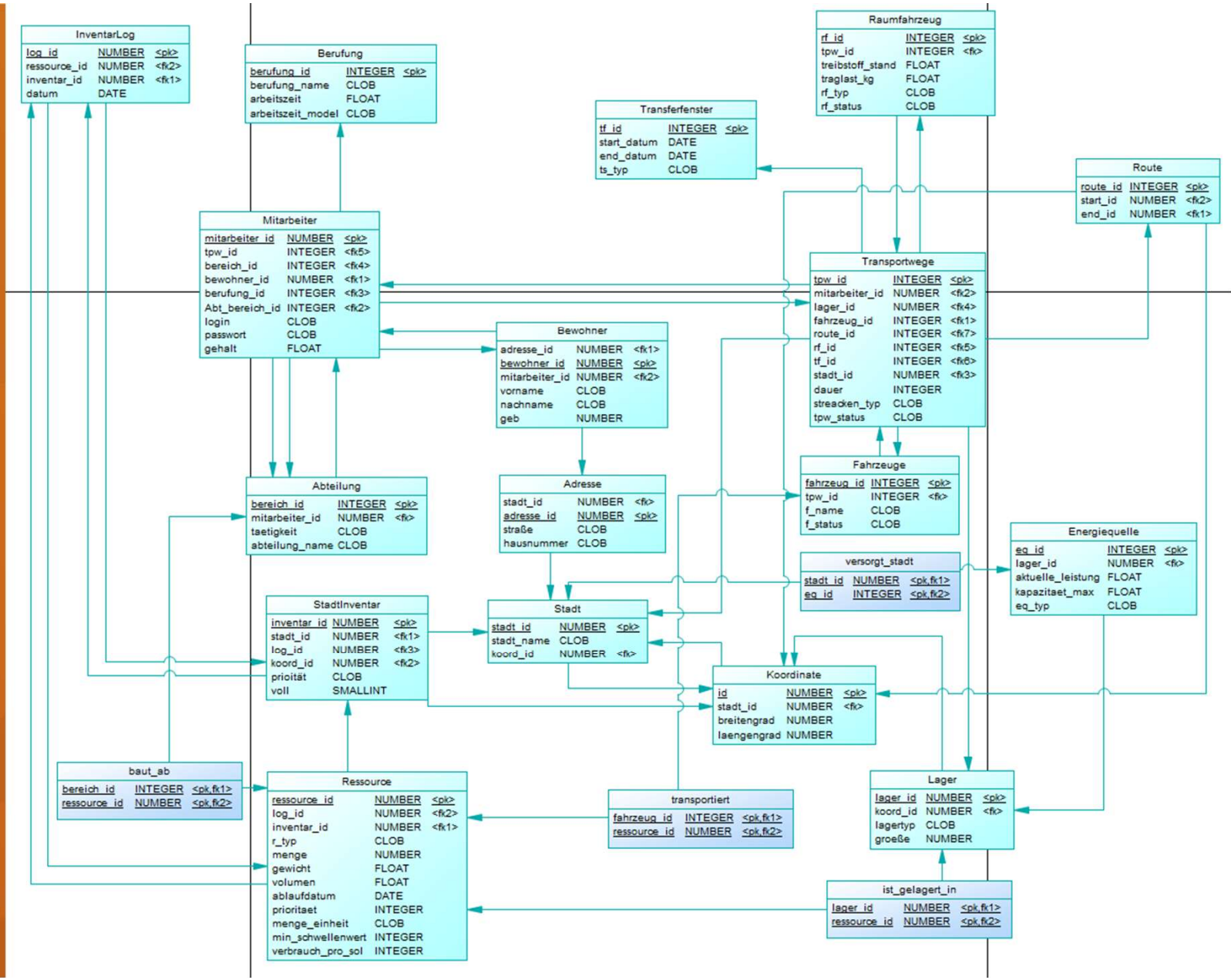
- WebApp mit TypeScript im frontend & php im Backend:
  - Abfragen für die Datenbank erfolgen per RestAPI
  - RestAPI liefert nur JSON zurück
  - Die RestAPI kann vorgefertigte Aktionen oder .sql Dateien ausführen und dessen Resultate per JSON zurückgeben

ERD & PDM

ERD


InventarLog
log_id Number
o datum Date
Berufung
berufung_id Integer
o berufung_name Text
o arbeitszeit Float
o arbeitszeit_model Text
Transferfenster
tf_id Integer
o start_datum Date
o end_datum Date
o ts_typ Text
Raumfahrzeug
rf_id Integer
o treibstoff_stand Float
o traglast_kg Float
o rf_typ Text
o rf_status STATUS
Route
route_id Integer
in_fenster
rs_transportiert
ist_startpunkt
hat_berufung
hat_route
ist_fahrer
Transportwege
tpw_id Integer
o dauer Integer
o streacken_typ Text
o tpw_status STATUS
Mitarbeiter
mitarbeiter_id Number
o login Text
o passwort Text
o gehalt Float
ist
Bewohner
bewohner_id Number
o vorname Text
o nachname Text
o geb Number
ist:geloggtes_inventar
ist_ziel
hat_transportweg
Fahrzeuge
fahrzeug_id Integer
o f_name Text
o f_status STATUS
ist_leiter
gehoert_zu
wohnt_in
ist_endpunkt
Abteilung
bereich_id Integer
o taetigkeit Text
o abteilung_name Text
Adresse
adresse_id Number
o straße Text
o hausnummer Text
ist_abholort
Energiequelle
eq_id Integer
o aktuelle_leistung Float
o kapazitaet_max Float
o eq_typ Text
hat
Stadt
stadt_id Number
o stadt_name Text
StadtInventar
inventar_id Number
o prioität Text
o voll Boolean
beinhaltet
transportiert
versorgt_stadt
baut_ab
befindet_sich_in
Koordinate
id Number
o breitengrad Number
o laengengrad Number
liegt_bei
liegt_in
ist_verfügbar_in
t_geloggte_ressource
Lager
lager_id Number
o lagertyp Text
o groeße Number
versorgt_lager
Ressource
ressource_id Number
o r_typ Text
o menge Number
o gewicht Float
o volumen Float
o ablaufdatum Date
o prioritaet Integer
o menge_einheit Text
o min_schwellenwert Integer
o verbrauch_pro_sol Integer
ist_gelagert_in

PDM


InventarLog
log_id NUMBER <pk>
ressource_id NUMBER <fk2>
inventar_id NUMBER <fk1>
datum DATE
Berufung
berufung_id INTEGER <pk>
berufung_name CLOB
arbeitszeit FLOAT
arbeitszeit_model CLOB
Transferfenster
tf_id INTEGER <pk>
start_datum DATE
end_datum DATE
ts_typ CLOB
Raumfahrzeug
rf_id INTEGER <pk>
tpw_id INTEGER <fk>
treibstoff_stand FLOAT
traglast_kg FLOAT
rf_typ CLOB
rf_status CLOB
Route
route_id INTEGER <pk>
start_id NUMBER <fk2>
end_id NUMBER <fk1>
Mitarbeiter
mitarbeiter_id NUMBER <pk>
tpw_id INTEGER <fk5>
bereich_id INTEGER <fk4>
bewohner_id NUMBER <fk1>
berufung_id INTEGER <fk3>
Abt_bereich_id INTEGER <fk2>
login CLOB
passwort CLOB
gehalt FLOAT
Transportwege
tpw_id INTEGER <pk>
mitarbeiter_id NUMBER <fk2>
lager_id NUMBER <fk4>
fahrzeug_id INTEGER <fk1>
route_id INTEGER <fk7>
rf_id INTEGER <fk5>
tf_id INTEGER <fk6>
stadt_id NUMBER <fk3>
dauer INTEGER
streacken_typ CLOB
tpw_status CLOB
Bewohner
adresse_id NUMBER <fk1>
bewohner_id NUMBER <pk>
mitarbeiter_id NUMBER <fk2>
vorname CLOB
nachname CLOB
geb NUMBER
Abteilung
bereich_id INTEGER <pk>
mitarbeiter_id NUMBER <fk>
taetigkeit CLOB
abteilung_name CLOB
Adresse
stadt_id NUMBER <fk>
adresse_id NUMBER <pk>
straße CLOB
hausnummer CLOB
Fahrzeuge
fahrzeug_id INTEGER <pk>
tpw_id INTEGER <fk>
f_name CLOB
f_status CLOB
Energiequelle
eq_id INTEGER <pk>
lager_id NUMBER <fk>
aktuelle_leistung FLOAT
kapazitaet_max FLOAT
eq_typ CLOB
versorgt_stadt
stadt_id NUMBER <pk,fk1>
eq_id INTEGER <pk,fk2>
StadtInventar
inventar_id NUMBER <pk>
stadt_id NUMBER <fk1>
log_id NUMBER <fk3>
koord_id NUMBER <fk2>
prioität CLOB
voll SMALLINT
Stadt
stadt_id NUMBER <pk>
stadt_name CLOB
koord_id NUMBER <fk>
Koordinate
id NUMBER <pk>
stadt_id NUMBER <fk>
breitengrad NUMBER
laengengrad NUMBER
baut_ab
bereich_id INTEGER <pk,fk1>
ressource_id NUMBER <pk,fk2>
Ressource
ressource_id NUMBER <pk>
log_id NUMBER <fk2>
inventar_id NUMBER <fk1>
r_typ CLOB
menge NUMBER
gewicht FLOAT
volumen FLOAT
ablaufdatum DATE
prioritaet INTEGER
menge_einheit CLOB
min_schwellenwert INTEGER
verbrauch_pro_sol INTEGER
transportiert
fahrzeug_id INTEGER <pk,fk1>
ressource_id NUMBER <pk,fk2>
Lager
lager_id NUMBER <pk>
koord_id NUMBER <fk>
lagertyp CLOB
groeße NUMBER
ist_gelagert_in
lager_id NUMBER <pk,fk1>
ressource_id NUMBER <pk,fk2>

WebApp Demo

1. Frontend (TypeScript)

1. **Frontend (TypeScript)**
    a. **Typisierte API-Kommunikation:** Nutzung von Interfaces, damit das Frontend genau weiß, wie die JSON-Daten vom Backend aussehen (verhindert Fehler beim Zugriff auf Datenfelder).
    b. **Zentraler API-Service:** Ein Modul, das alle Anfragen (GET, POST) bündelt und sich automatisch um die Header und die Fehlerbehandlung kümmert.
    c. **Token-Handling:** Das CSRF-Token wird sicher ausgelesen und bei jeder schreibenden Anfrage automatisch im HTTP-Header mitgeschickt.

2. Backend (PHP)

1. **Backend (PHP)**
    a. **JSON-only API:** Das Backend dient rein als Datenlieferant. Es gibt kein HTML aus, sondern setzt den Content-Type fest auf application/json.
    b. **Sicherer SQL-Executor:** Ein Mechanismus, der vordefinierte SQL-Dateien einliest. Dabei werden Benutzereingaben strikt über **Prepared Statements** getrennt, um SQL-Injection zu verhindern.
    c. **Zentrale Routen-Steuerung:** Ein einfacher Controller entscheidet anhand der Anfrage, welche „Aktion“ oder welche .sql-Datei geladen und ausgeführt werden soll.

3. API & Sicherheit

1. **API & Sicherheit**
    a. **CSRF-Schutz:** Eine Überprüfung bei jedem API-Aufruf, die sicherstellt, dass die Anfrage wirklich von deiner WebApp kommt und nicht von einer fremden Seite.
    b. **Standardisierte Antwort-Struktur:** Jede Antwort sieht gleich aus (immer ein Feld für result und eines für status), was die Verarbeitung im Frontend extrem vereinfacht.
    c. **Trennung von Logik und Daten:** SQL-Befehle liegen nicht direkt im PHP-Code, sondern in eigenen Dateien, was die Wartung und Übersicht verbessert.

Bsp. SQL-Datein

RessroucenAtRisk.sql

```sql
SELECT r.ressource_id,
    r.name,
    r.ablaufdatum,
    ig.lager_id
FROM 'Ressource' r
    JOIN 'ist_gelagert_in' ig ON r.ressource_id = ig.ressource_id
WHERE r.ablaufdatum BETWEEN CURRENT_DATE AND DATE_ADD(CURRENT_DATE, INTERVAL 30 DAY)
ORDER BY r.ablaufdatum ASC;
```

AnzahlDerBewohner.sql

```sql
SELECT
    COUNT(*) AS 'citizens_count',
    COUNT(
        CASE
            WHEN 'GEB' > CURRENT_DATE - INTERVAL '18' YEAR THEN 1
            ELSE NULL
        END
    ) AS 'minors_count'
FROM 'BEWOHNER';
```

Selbstreflexion