12 Juni 2025

Aktueller Stand & Arbeitsweise für die Datenbank

Dieses Dokument beschreibt unseren aktuellen Entwicklungsstand und die geplante Vorgehensweise für die Datenbanknutzung. Bitte lest es sorgfältig durch, damit wir alle auf dem gleichen Stand sind.

1. Aktueller Stand: Die lokale Docker-Sandbox

Momentan verwenden wir für die Entwicklung und zum Testen ausschließlich einen lokalen "Sandbox"-Ansatz mit Docker.

Was bedeutet das?

- Jedes Teammitglied kann mit Docker eine private, lokale Postgres-Datenbank auf seinem eigenen Rechner starten
- Diese Datenbank wird aus den SQL-Skripten in unserem Projekt erstellt, sodass jeder die exakt gleiche Tabellenstruktur hat

Warum ist das ein guter Start?

- **Konsistenz:** Jeder hat eine identische Umgebung, was "Bei mir geht's aber nicht"-Fehler vermeidet
- **Isolation:** Wir können unabhängig voneinander arbeiten, ohne die Arbeit der anderen zu stören
- **Sicherheit:** Jeder kann seine lokale Datenbank nach Belieben testen, füllen und zurücksetzen (docker compose down -v), ohne Konsequenzen für das Team

2. Anleitung: Setup & Arbeiten mit Docker & Postgres

Voraussetzungen:

Docker Desktop installieren: Ladet euch Docker Desktop von der offiziellen Docker-Webseite herunter und installiert es. Es enthält sowohl Docker Engine als auch Docker Compose. Startet es nach der Installation.

Schritte:

1. **Projektordner erstellen:** Legt euch einen Ordner für die DB an die das docker-compose.yml File und den Ordner init-db enthalten

2. **Docker starten:**

   Öffnet ein Terminal (wie PowerShell, CMD, Bash, etc.)
   -> Navigiert mit cd in euren Projektordner
   -> Führt den Befehl aus: docker compose up -d

   docker compose up: Startet die Dienste, die in der docker-compose.yml definiert sind.
   -d (detached mode): Lässt die Container im Hintergrund laufen, sodass euer Terminal frei bleibt.

   Beim ersten Mal wird Docker die Images (Postgres, pgAdmin) herunterladen. Das kann ein paar Minuten dauern. Danach werden die Container erstellt und gestartet. Postgres wird euer 01-schema.sql-Skript ausführen.

3. **Überprüfen, ob es läuft:**

   Gebt ***docker ps*** im Terminal ein. Ihr solltet zwei laufende Container sehen: postgres_db und postgres_admin.

4. **Mit pgAdmin verbinden:**

   - Öffnet euren Webbrowser und geht zu http://localhost:8080.
   - Loggt euch mit der E-Mail (admin@example.com) und dem Passwort (admin123) ein, die im docker-compose.yml definiert sind
   - Klickt auf "Add New Server".
   - Im Tab **"General"**: Gebt einen Namen ein, z.B. "SmartCity DB".
   - Im Tab **"Connection"**: **Host name/address:** *postgres_db* (Wichtig: *Nicht* localhost, da pgAdmin innerhalb des Docker-Netzwerks läuft und den Postgres-Container über seinen Namen anspricht)
     - **Port:** 5432
     - **Maintenance database:** webportal
     - **Username:** admin
     - **Password:** Admin!1234
   - Klickt auf "Save".
- Wenn alles geklappt hat, seht ihr links euren Server. Navigiert zu Databases -> webportal -> Schemas -> public -> Tables.
  Dort sind die erstellten Tabellen erscheinen

5. **Stoppen:**
   - Wenn die Arbeit beendet ist, könnt ihr die Container stoppen mit: **docker compose down**. Dies stoppt *und entfernt* die Container, aber **nicht** die Volumes (postgres_data, pgadmin_data). Eure Daten bleiben also erhalten.

   - Beim nächsten **docker compose up -d** werden die Container neu erstellt, aber sie verwenden die existierenden Daten, und euer init-db-Skript wird **nicht** erneut ausgeführt (weil das Datenverzeichnis nicht mehr leer ist).