

WEBTECHNOLOGIEN

05 — NODE UND DATENBANKEN

PROF. DR. MARKUS HECKNER

WARUM DATENBANKEN

- Webanwendungen verarbeiten oft strukturierte Massendaten (Profile, Likes, Posts, Playlisten, Kommentare, etc.)
- Die Daten müssen aus der Datenbank in die Webanwendung und die Webanwendung muss die Daten einfügen, aktualisieren und löschen können (CRUD – Create Read Update Delete)
- Zur Verwaltung dieser Daten stehen zahlreiche Datenbankmanagementsysteme (DBMS) zur Verfügung, z.B.
 - MySQL
 - Oracle 12c
 - **PostgreSQL**
 - SQLite
 - MongoDB (nicht-relational)
 - ...

DATENBANKEN – HEUTE SQL MIT POSTGRES SQL



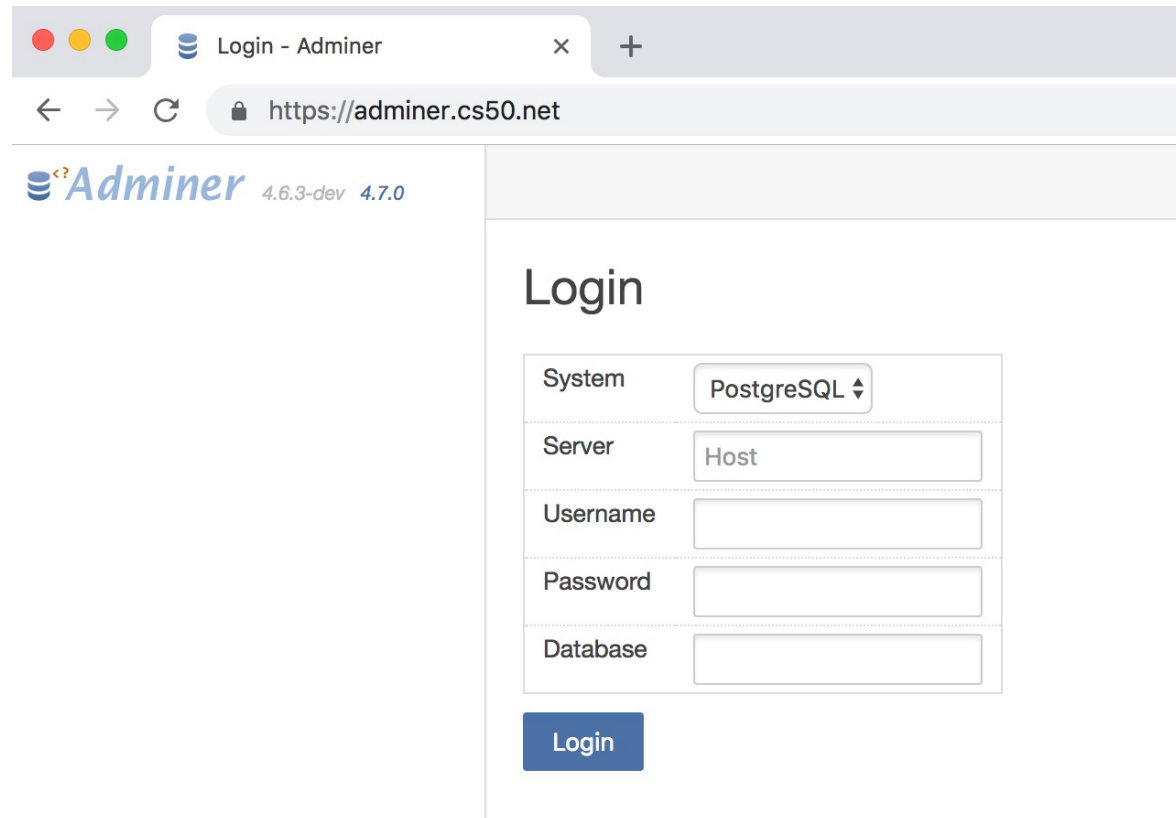
- Relationales DBMS
- Open Source
- Packages für den Zugriff über Node.js verfügbar
- Kostenlose Datenbank bei Heroku verfügbar

IN DIESER SESSION NUR EINE TABELLE

```
CREATE TABLE dbdemo_playlists (  
    ID SERIAL PRIMARY KEY,  
    TITLE VARCHAR  
);
```

```
INSERT INTO dbdemo_playlists (TITLE) VALUES ('Happy Mood');  
INSERT INTO dbdemo_playlists (TITLE) VALUES ('Iconic songs');
```

ZUGRIFF AUF DIE POSTGRESQL DATENBANK MIT ADMINER



The screenshot shows a web browser window with the title "Login - Adminer" and the URL "https://adminer.cs50.net". The Adminer logo and version "4.6.3-dev 4.7.0" are visible in the top left. The main content area is titled "Login" and contains a form with the following fields:

System	PostgreSQL ▾
Server	Host
Username	<input type="text"/>
Password	<input type="password"/>
Database	<input type="text"/>

Below the form is a blue "Login" button.

DB DEMO – LISTE ALLER PLAYLISTEN

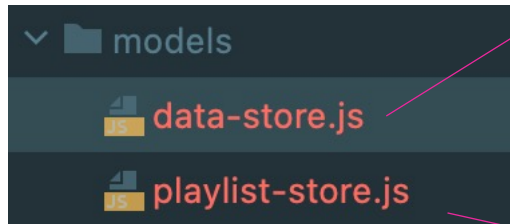
[Playlist 1](#) [Dashboard](#) [About](#)

Dashboard

Happy Mood

Iconic songs

ZUSAMMENSPIEL ZWISCHEN DATA-STORE.JS UND DEM MODEL PLAYLIST-STORE.JS



Model benötigt Zugriff auf die Datenquelle
(hier: PostgreSQL) – data-store.js stellt diesen Zugriff her
– Alle Models nutzen data-store.js (mehr später)

Model zur Abfrage und Ablegen von Daten zur playList

DAS MODEL STEHT DEN CONTROLLERN ZUR VERFÜGUNG UM DATEN ZU SPEICHERN UND ZU HOLEN

data-store.js

```
const datastore = require("../data-store.js");
const datastoreClient = datastore.getDataStore();
const logger = require("../utils/logger.js");

const playlistStore = {
  async getAllPlaylists() {
    const query = 'SELECT * FROM playlist2_playlists';
    try {
      let result = await datastoreClient.query(query);
      return result.rows;
    } catch (e) {
      logger.error("Error fetching all playlists", e);
    }
  },
};

module.exports = playlistStore;
```

Model benötigt Zugriff auf die Datenquelle
(hier: PostgreSQL) – data-store.js stellt diesen Zugriff her
– Alle Models nutzen data-store.js (mehr später)

Objekt playListStore ist das Model und bietet
Seine Methoden Zugriff auf die Daten

SQL-Query-String

Absenden des Queries mithilfe
des data-stores

Rückgabe des Ergebnisses an
den Controller

Error Handling – Logging des
DB Fehlers

DATA-STORE.JS STELLT VERBINDUNG ZUR DB HER

- Stellt Verbindung zur DB her
- Hier postgres, beliebige andere relationale und nicht-relationale DBMS integrierbar

```
let pg = require("pg");
const logger = require("../utils/logger.js");
const conString = process.env.DB_CON_STRING;

const dbConfig = {
  connectionString: conString,
  ssl: { rejectUnauthorized: false }
}

if (conString == undefined) {
  logger.error("ERROR: environment variable DB_CON_STRING not set.");
  process.exit( code: 1);
}

let dbClient = null;

const datastore = {
  getDataStore() {
    if (dbClient !== null) {
      return dbClient;
    } else {
      dbClient = new pg.Client(dbConfig);
      dbClient.connect();
      return dbClient;
    }
  },
  async endConnection() {
    await dbClient.end();
  }
}
```

DATA-STORE.JS (1/2)

Liest String für Zugriff auf die Datenbank aus der Datei .env

```
let pg = require("pg");
const logger = require("../utils/logger.js");
const conString = process.env.DB_CON_STRING;

const dbConfig = {
  connectionString: conString,
  ssl: { rejectUnauthorized: false }
}

if (conString == undefined) {
  logger.error("ERROR: environment variable DB_CON_STRING not set.");
  process.exit( code: 1);
}
```

Config

Error-Handling

DATA-STORE.JS (2/2)

Gibt dbClient für das Absenden von Anfragen
an die Models zurück

```
let dbClient = null;

const datastore = {
  getDataStore() {
    if (dbClient !== null) {
      return dbClient;
    } else {
      dbClient = new pg.Client(dbConfig);
      dbClient.connect();
      return dbClient;
    }
  },
  async endConnection() {
    await dbClient.end();
  }
}
```

Mehr ist erstmal nicht wichtig!

