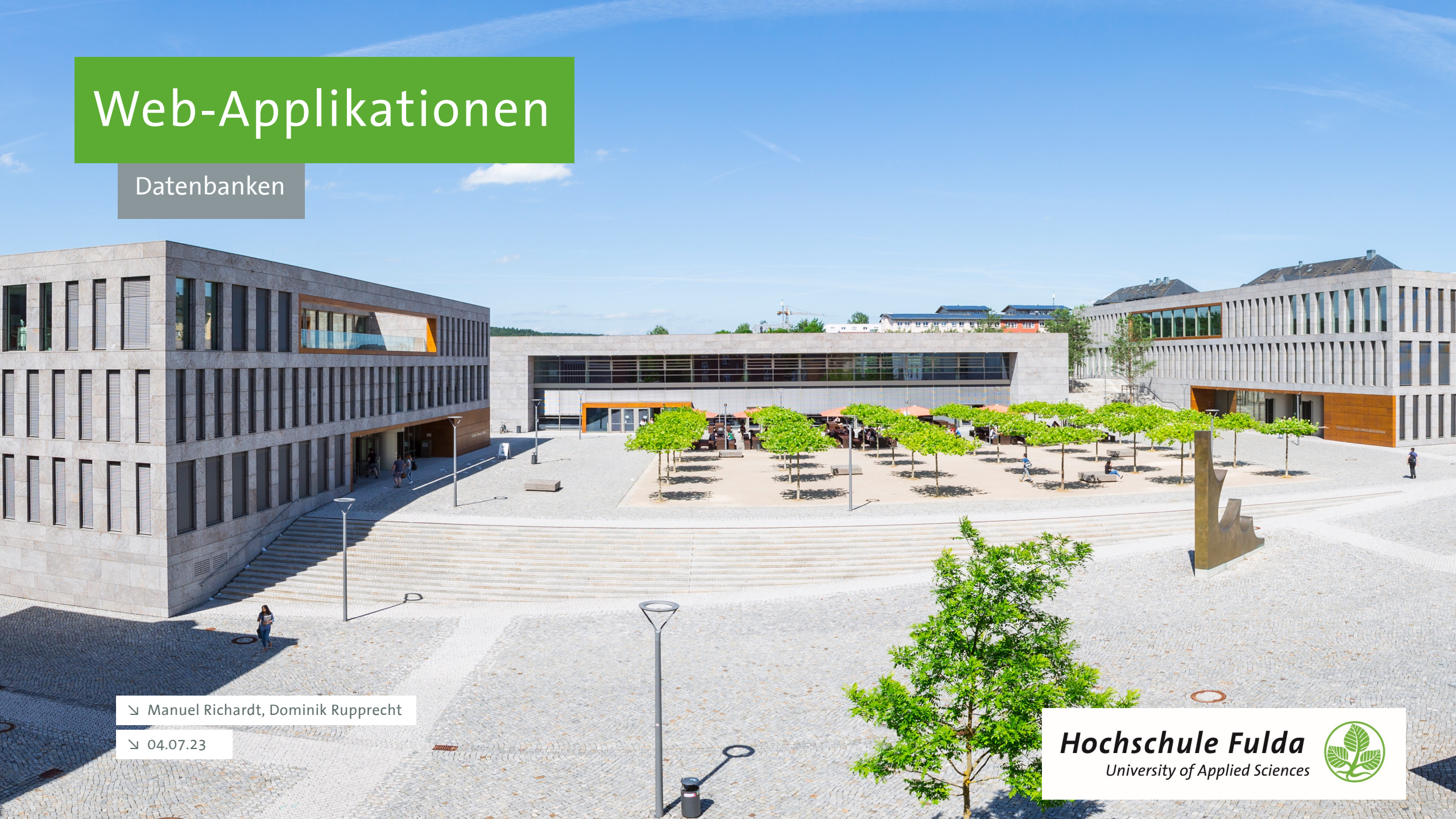
Web-Applikationen

Datenbanken



↘ Manuel Richardt, Dominik Rupprecht

↘ 04.07.23

Hochschule Fulda
University of Applied Sciences

Fragen

AI

↘Rückfragen zur SU/Übung?

Installation und Konfiguration über Kommandozeile

AI

- Installieren von Sequelize und SQLite3
  - `npm install sequelize sqlite3`
- Installieren von Sequelize-CLI (Kommandozeilen-Tool)
  - `npm install --save-dev sequelize-cli`
  - `sequelize init`
  - `npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string`
  - `sequelize-cli db:migrate`

Einbinden und nutzen

AI

↘app.js

```
var db = require('./models/index');

db.sequelize.sync({force: true}).then(() => {
  console.log("Database is ready");
});
```

↘userController.js

```
const db = require('../models/index');

async function getAllUsers(req,res,next) {
  await tmp(); //Erstellt tmp-Benutzer
  const users = await db.User.findAll();
  res.status(200).json(users);
}
```

Modell definieren

- Modell als Klasse
- Implementiert Init
- Attribute (Spalten) des Modells
- Attribute können mit Eigenschaften belegt werden

- Anpassungen auch in „Migrations“ übernehmen
- Mit `sequelize db:migrate` Datenbank aktualisieren

```
module.exports = (sequelize, DataTypes) => {
  class Post extends Model {
    static associate({User}) {
      // define association here
      this.belongsTo(User)
    }
  }
  Post.init({
    id: {
      allowNull: false,
      autoIncrement: true,
      primaryKey: true,
      type: DataTypes.INTEGER,
    },
```

Modelle definieren

AI

↘ Methoden in Modellen

↘ Definition von Modellspezifischen Methoden

```
class User extends Model {
  validPassword(password) {
    var hash = crypto.pbkdf2Sync(password,
      this.salt, 1000, 64, `sha512`).toString(`hex`);
    console.log(hash == this.password)
    return this.password === hash;
  }
```

```
if (user.validPassword(password)) {
```

Beziehungen zwischen Modellen

AI

- ↘ Associations
  - ↘ hasOne
    - ↘ Besitzt eins
  - ↘ belongsTo
    - ↘ Gehört einem
  - ↘ hasMany
    - ↘ Besitzt viele
  - ↘ belongsToMany
    - ↘ Gehört zu Vielen
- ↘ Muss wechselseitig gesetzt sein
- ↘ Bezieht sich auf Primary Key

```
module.exports = (sequelize, DataTypes) => {
  class Post extends Model {
    static associate({User}) {
      // define association here
      this.belongsTo(User)
    }
  }
```

```
static associate({Post}) {
  // define association here
  this.hasMany(Post)
}
```

CRUD mit Sequelize

AI

↘ Datensätze lesen

↘
```
let users = await User.findAll({…
```

↘ Liefert Array

↘
```
let user = await User.findByPk(id , {…
```

↘ Finden über Primär Schlüssel

↘ Liefert einen Datensatz

↘
```
let user = await User.findOne({ where: { userName } })
```

↘ Liefert einen Datensatz

↘ „where“ kann Bedingung enthalten

CRUD mit Sequelize

AI

↘ Datensätze erstellen

↘
```
let user = await User.create({ userName, password, email, salt: "" })
```

↘ „create“ erzeugt Datensatz

↘ Werte werden als Objekt übergeben

CRUD mit Sequelize

AI

↘ Datensätze bearbeiten

↘ Update

↘ Werte werden als Objekt übergeben

↘ „where“ muss gesetzt sein

```
user = await User.update({ email, password:pwd, salt }, {
    where: {
        userName: userName
    }
})
```

CRUD mit Sequelize

AI

↘ Datensätze löschen

↘ destroy

↘ „where“ muss gesetzt sein

```
await User.destroy({
    where: {
        id: id
    }
})
```