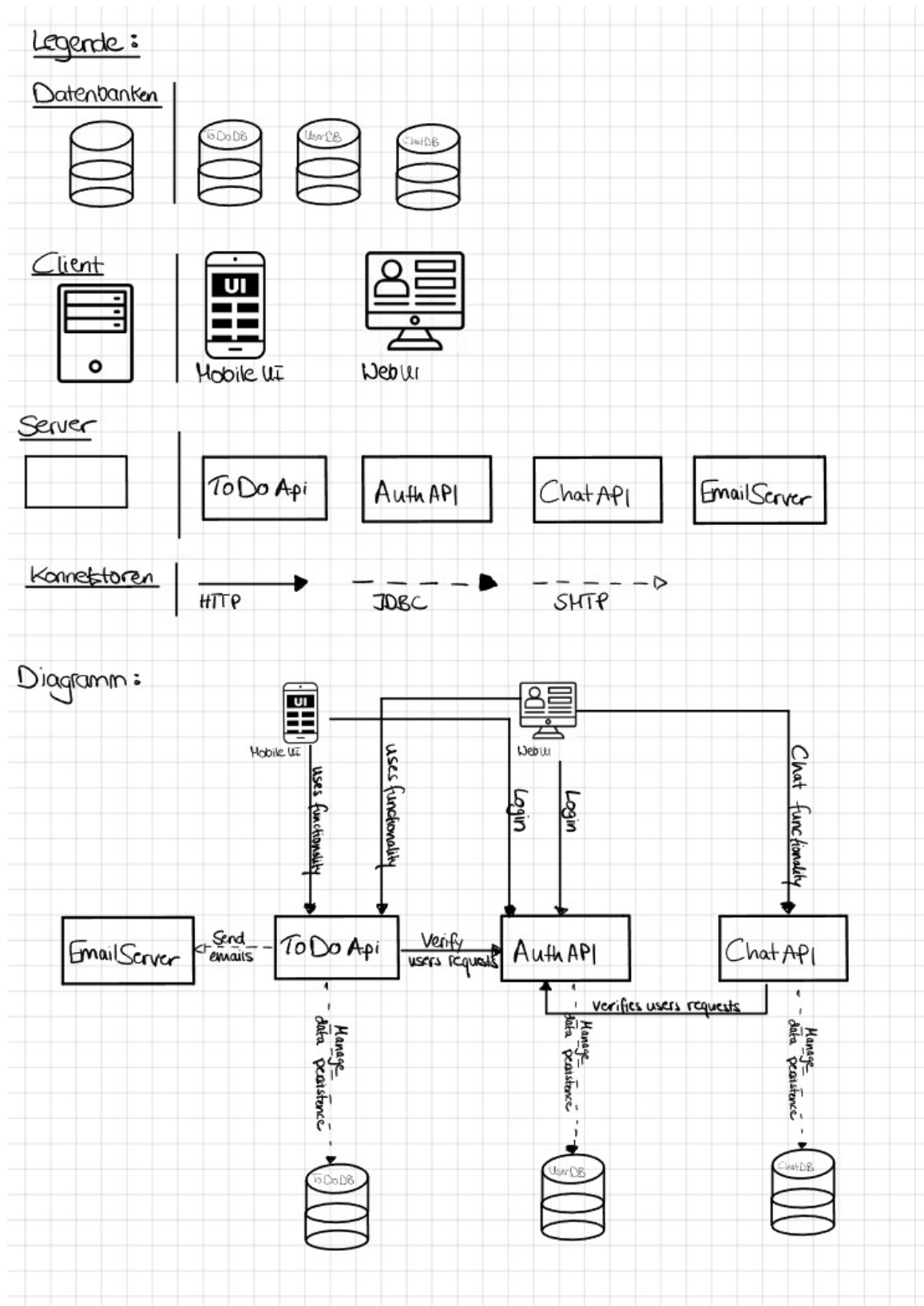


Exercise 1: Architektur



Exercise 2: SQL und JDBC

2.1 Werkzeugunterstützte SQL-Befehle

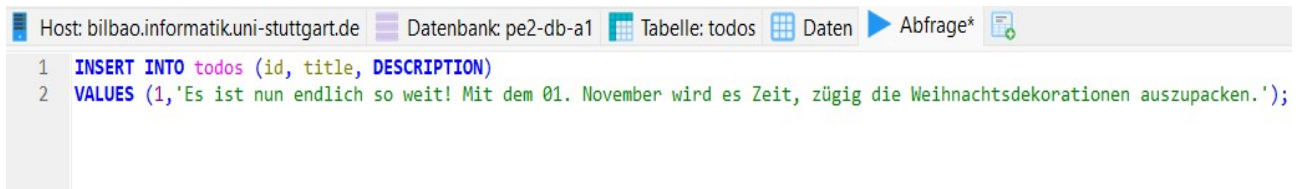
(a) .



Host: bilbao.informatik.uni-stuttgart.de Datenbank: pe2-db-a1 Tabelle: todos Daten

```
1 CREATE TABLE IF NOT EXISTS todos (id INTEGER PRIMARY KEY, title VARCHAR (100) NOT
2 NULL DEFAULT 'New todo', description VARCHAR (500));
```

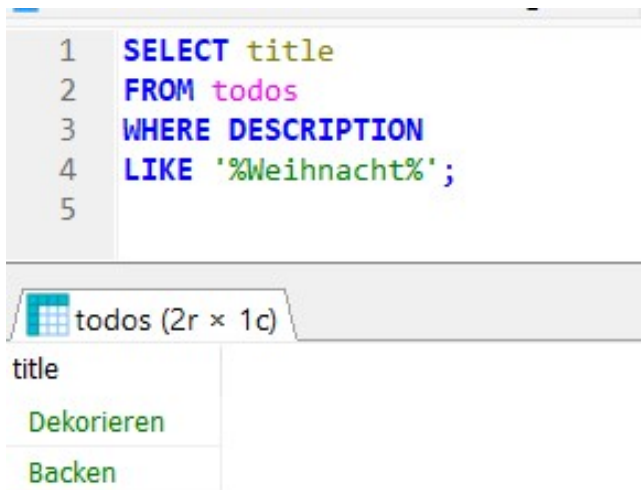
(b) .



Host: bilbao.informatik.uni-stuttgart.de Datenbank: pe2-db-a1 Tabelle: todos Daten Abfrage*

```
1 INSERT INTO todos (id, title, DESCRIPTION)
2 VALUES (1,'Es ist nun endlich so weit! Mit dem 01. November wird es Zeit, zügig die Weihnachtsdekorationen auspacken.');
```

(c) .



```
1 SELECT title
2 FROM todos
3 WHERE DESCRIPTION
4 LIKE '%Weihnacht%';
5
```

todos (2r × 1c)

title
Dekorieren
Backen

2.2 Programmatische Datenbankabfrage

(a) .

```
1 package de.unistuttgart.iste.pe2.examples;
2
3 import java.sql.SQLException;
4 import java.util.logging.Level;
5 import java.util.logging.Logger;
6 import com.j256.ormlite.dao.Dao;
7 import com.j256.ormlite.dao.DaoManager;
8 import com.j256.ormlite.jdbc.JdbcConnectionSource;
9 import com.j256.ormlite.support.ConnectionSource;
10 import de.unistuttgart.iste.pe2.model.Letters;
11
12
13 public class WordFinder {
14
15     private ConnectionSource connectionSource;
16     private Dao <Letters, Integer> lettersDao;
17
18     private static Logger LOGGER = Logger.getLogger(WordFinder.class.getName());
19
20     public void findWord() {
21         // creates connection to the pe2-db-a1 database
22         boolean connected = this.connectToDB(connectionString:"jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1", user:"pe2-nutzer", password:"esJLfM6ksCT4mCy05");
23
24         if (connected) {
25             try {
26                 lettersDao = DaoManager.createDao(connectionSource, clazz:Letters.class);
27
28                 int[] arrayIndexes = {
29                     | 20, 44, 50, 13, 17, 33, 41, 68, 77, 44, 29, 72, 48, 71, 37, 48, 11, 69, 5, 65, 65
30                 };
31
32                 // Abrufen des Wortes aus der Tabelle 'letters' anhand der Index-Zahlen
33                 StringBuilder word = new StringBuilder();
34                 for (int index : arrayIndexes) {
35                     Letters letter = lettersDao.queryForId(index);
36                     if (letter != null) {
37                         word.append(letter.getLetter());
38                     }
39                 }
40
41                 // Ausgabe des gefundenen Wortes
42                 System.out.println("Das Wort ist: " + word.toString());
43             }
44             this.closeConnectionToDB();
45
46             } catch (SQLException exception) {
47                 this.logSQLException(exception);
48             }
49         }
50     }
51
52     private boolean connectToDB(String connectionString, String user, String password) {
53         try {
54             this.connectionSource = new JdbcConnectionSource(connectionString, user, password);
55
56             return true;
57         } catch (SQLException exception) {
58             this.logSQLException(exception);
59         }
60         return false;
61     }
62 }
```

```
63 private void closeConnectionToDB() {
64     try {
65         this.connectionSource.close();
66     } catch (Exception exception) {
67         LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
68     }
69 }
70
71 private void logSQLException(SQLException exception) {
72     LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
73     LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
74 }
75 }
76
```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Das Wort ist: EntwicklUnGPrOgr
2023-11-19 15:09:38,384 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwicklUnGPrOgrA
2023-11-19 15:09:38,384 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwicklUnGPrOgrAM
2023-11-19 15:09:38,392 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwicklUnGPrOgrAMM
2023-11-19 15:09:38,400 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwicklUnGPrOgrAMMI
2023-11-19 15:09:38,400 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwicklUnGPrOgrAMMII
2023-11-19 15:09:38,409 [DEBUG] BaseJdbcConnectionSource closed connection #283383329
PS C:\Users\timan\Downloads\lecture-examples-main> []

Lösungswort: EntwicklUnGPrOgrAMMII

(b) .

```
1 package de.unistuttgart.iste.pe2.examples;
2
3 import java.sql.SQLException;
4 import java.util.List;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7 import com.j256.ormlite.dao.Dao;
8 import com.j256.ormlite.dao.DaoManager;
9 import com.j256.ormlite.jdbc.JdbcConnectionSource;
10 import com.j256.ormlite.support.ConnectionSource;
11 import de.unistuttgart.iste.pe2.model.Letters;
12
13
14 public class LetterFinder {
15
16     private ConnectionSource connectionSource;
17     private Dao<Letters, Integer> lettersDao;
18
19     private static Logger LOGGER = Logger.getLogger(LetterFinder.class.getName());
20
21     public void letterFinder() {
22         // creates connection to the pe2-db-a1 database
23         boolean connected = this.connectToDB(connectionString:"jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1", user:"pe2-nutzer", password:"esJLfM6ksCT4mCyOS");
24
25         if (connected) {
26             try {
27                 lettersDao = DaoManager.createDao(connectionSource, clazz:Letters.class);
28
29                 char[] lettersToFind = { 'V', 'b', 't' };
30
31                 // IDs für jeden Buchstaben abrufen und ausgeben
32                 for (char letter : lettersToFind) {
33                     List<Letters> result = lettersDao.queryForEq(fieldName:"letter", String.valueOf(letter));
34                     System.out.println("IDs für '" + letter + "' = " + extractIds(result));
35                 }
36
37                 this.closeConnectionToDB();
38             } catch (SQLException exception) {
39                 this.logSQLException(exception);
40             }
41         }
42     }
43
44     private String extractIds(List<Letters> letters) {
45         StringBuilder ids = new StringBuilder();
46         for (Letters letter : letters) {
47             ids.append(letter.getId()).append(", ");
48         }
49         return ids.length() > 0 ? ids.substring(0, ids.length() - 2) : "Keine Einträge gefunden";
50     }
51 }
```

```
55 private boolean connectToDB(String connectionString, String user, String password) {
56     try {
57         this.connectionSource = new JdbcConnectionSource(connectionString, user, password);
58         return true;
59     } catch (SQLException exception) {
60         this.logSQLException(exception);
61     }
62     return false;
63 }
64
65 private void closeConnectionToDB() {
66     try {
67         this.connectionSource.close();
68     } catch (Exception exception) {
69         LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
70     }
71 }
72
73 private void logSQLException(SQLException exception) {
74     LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
75     LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
76 }
77 }
78 }
```

PROBLEMS (16) OUTPUT DEBUG CONSOLE TERMINAL PORTS

2023-11-19 15:25:44,423 [DEBUG] BaseJdbcConnectionSource opened connection to jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1 got #1209702763
2023-11-19 15:25:44,456 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM 'letters' WHERE 'letter' = 'V' with 0 args
2023-11-19 15:25:44,488 [DEBUG] SelectIterator starting iterator @22600334 for 'SELECT * FROM 'letters' WHERE 'letter' = 'V'
2023-11-19 15:25:44,488 [DEBUG] StatementExecutor query of 'SELECT * FROM 'letters' WHERE 'letter' = 'V' with 0 args returned 2 results
IDs für 'V' = 52, 78
2023-11-19 15:25:44,498 [DEBUG] StatementBuilder built statement SELECT * FROM 'letters' WHERE 'letter' = 'b'
2023-11-19 15:25:44,498 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM 'letters' WHERE 'letter' = 'b' with 0 args
2023-11-19 15:25:44,504 [DEBUG] SelectIterator starting iterator @173099767 for 'SELECT * FROM 'letters' WHERE 'letter' = 'b'
2023-11-19 15:25:44,504 [DEBUG] SelectIterator closed iterator @173099767 after 3 rows
2023-11-19 15:25:44,504 [DEBUG] StatementExecutor query of 'SELECT * FROM 'letters' WHERE 'letter' = 'b' with 0 args returned 3 results
IDs für 'b' = 9, 32, 58
2023-11-19 15:25:44,504 [DEBUG] StatementBuilder built statement SELECT * FROM 'letters' WHERE 'letter' = 't'
2023-11-19 15:25:44,504 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM 'letters' WHERE 'letter' = 't' with 0 args
2023-11-19 15:25:44,504 [DEBUG] SelectIterator starting iterator @112797691 for 'SELECT * FROM 'letters' WHERE 'letter' = 't'
2023-11-19 15:25:44,504 [DEBUG] SelectIterator closed iterator @112797691 after 2 rows
2023-11-19 15:25:44,504 [DEBUG] StatementExecutor query of 'SELECT * FROM 'letters' WHERE 'letter' = 't' with 0 args returned 2 results
IDs für 't' = 50, 76
2023-11-19 15:25:44,521 [DEBUG] BaseJdbcConnectionSource closed connection #1209702763
PS C:\Users\timan\Downloads\lecture-examples-main

IDs für 'V' = 52, 78
IDs für 'b' = 9, 32, 58
IDs für 't' = 50, 76

(c) .

```
private static Logger LOGGER = Logger.getLogger(LetterSumAvg.class.getName());

public void sumavg() {
    // creates connection to the pe2-db-a1 database
    boolean connected = this.connectToDB(connectionString:"jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1", user:"pe2-nutzer", password:"esJltFm6ksCT4mCyOS");

    if (connected) {
        try {
            lettersDao = DaoManager.createDao(connectionSource, clazz:letters.class);

            // Alle Einträge aus der Tabelle 'letters' abrufen
            List<Letters> allLetters = lettersDao.queryForAll();

            int sum = 0;
            for (Letters letter : allLetters) {
                sum += letter.getId();
            }

            double average = (double) sum / allLetters.size();

            LOGGER.info("Summe = " + sum);
            LOGGER.info("Durchschnittswert = " + average);

            this.closeConnectionToDB();
        } catch (SQLException exception) {
            this.logSQLException(exception);
        }
    }
}
```

```
49 }
50 private boolean connectToDB(String connectionString, String user, String password) {
51     try {
52         this.connectionSource = new JdbcConnectionSource(connectionString, user, password);
53     }
54     return true;
55     } catch (SQLException exception) {
56         this.logSQLException(exception);
57     }
58     return false;
59 }
60
61 private void closeConnectionToDB() {
62     try {
63         this.connectionSource.close();
64     } catch (Exception exception) {
65         LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
66     }
67 }
68
69 private void logSQLException(SQLException exception) {
70     LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
71     LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
72 }
73 }
74 }
```

PROBLEMS (16) OUTPUT DEBUG CONSOLE TERMINAL PORTS

INFORMATION: Summe = 4167
Nov. 19, 2023 3:32:11 PM de.unistuttgart.iste.pe2.examples.LetterSumAvg sumavg
INFORMATION: Durchschnittswert = 50.81707317073171
2023-11-19 15:32:11,202 [DEBUG] BaseJdbcConnectionSource closed connection #1354003114
PS C:\Users\timan\Downloads\lecture-examples-main> []

Summe = 4167

Durchschnittswert = 50.81707317073171

Exercise 3: HTTP und REST

(a) GET-Request senden

GET <https://api.chucknorris.io/jokes/random?category=history>

Params • Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

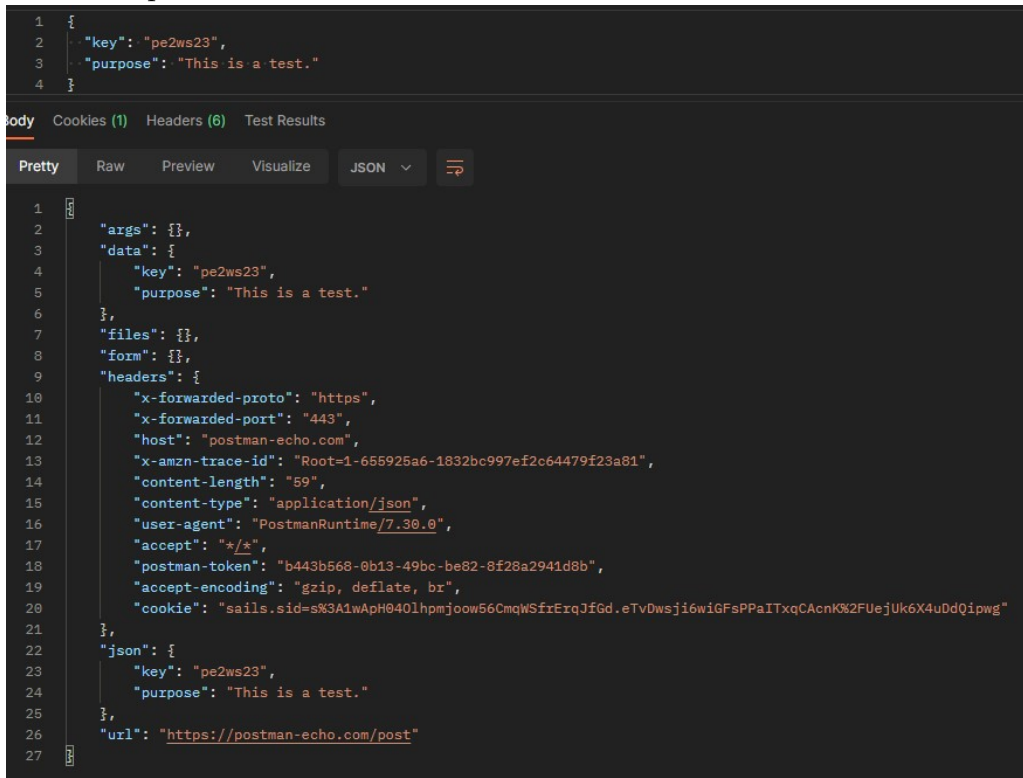
KEY	VALUE
<input checked="" type="checkbox"/> category	history
Key	Value

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2    "categories": [
3      "history"
4    ],
5    "created_at": "2020-01-05 13:42:19.576875",
6    "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
7    "id": "kfjreqvxs464s6rspcj-qq",
8    "updated_at": "2020-01-05 13:42:19.576875",
9    "url": "https://api.chucknorris.io/jokes/kfjreqvxs464s6rspcj-qq",
10   "value": "Chuck Norris once shot down a German fighter plane with his finger. By yelling \"Bang!\""
11 }
```


(b) POST-Request senden



```
1 {
2   "key": "pe2ws23",
3   "purpose": "This is a test."
4 }
```

body Cookies (1) Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "args": {},
3   "data": {
4     "key": "pe2ws23",
5     "purpose": "This is a test."
6   },
7   "files": {},
8   "form": {},
9   "headers": {
10    "x-forwarded-proto": "https",
11    "x-forwarded-port": "443",
12    "host": "postman-echo.com",
13    "x-amzn-trace-id": "Root=1-655925a6-1832bc997ef2c64479f23a81",
14    "content-length": "59",
15    "content-type": "application/json",
16    "user-agent": "PostmanRuntime/7.30.0",
17    "accept": "*/*",
18    "postman-token": "b443b568-0b13-49bc-be82-8f28a2941d8b",
19    "accept-encoding": "gzip, deflate, br",
20    "cookie": "sails.sid=s%3A1wApH040lhpmjoow56CmqWSfrErqJfGd.eTvDwsji6wiGFsPPaITxqCAcnK%2FUejUk6X4uDdQipwg"
21  },
22  "json": {
23    "key": "pe2ws23",
24    "purpose": "This is a test."
25  },
26  "url": "https://postman-echo.com/post"
27 }
```

(c) Eine einzelne DVD über Id identifizieren:

GET /dvds/\$id

Alle DVDs zurückgeben oder suchen (z.B. mit Altersbeschränkung)

GET /dvds/\$id?titleContains=string&category=string&ageRestricted=true

Eine DVD löschen:

DELETE /dvds/\$id

Aktualisiert die Altersbeschränkung einer bestimmten DVD anhand ihrer ID:

PUT /dvds/\$id/ageRestricted=false

Fügt eine neue DVD hinzu:

POST /dvds