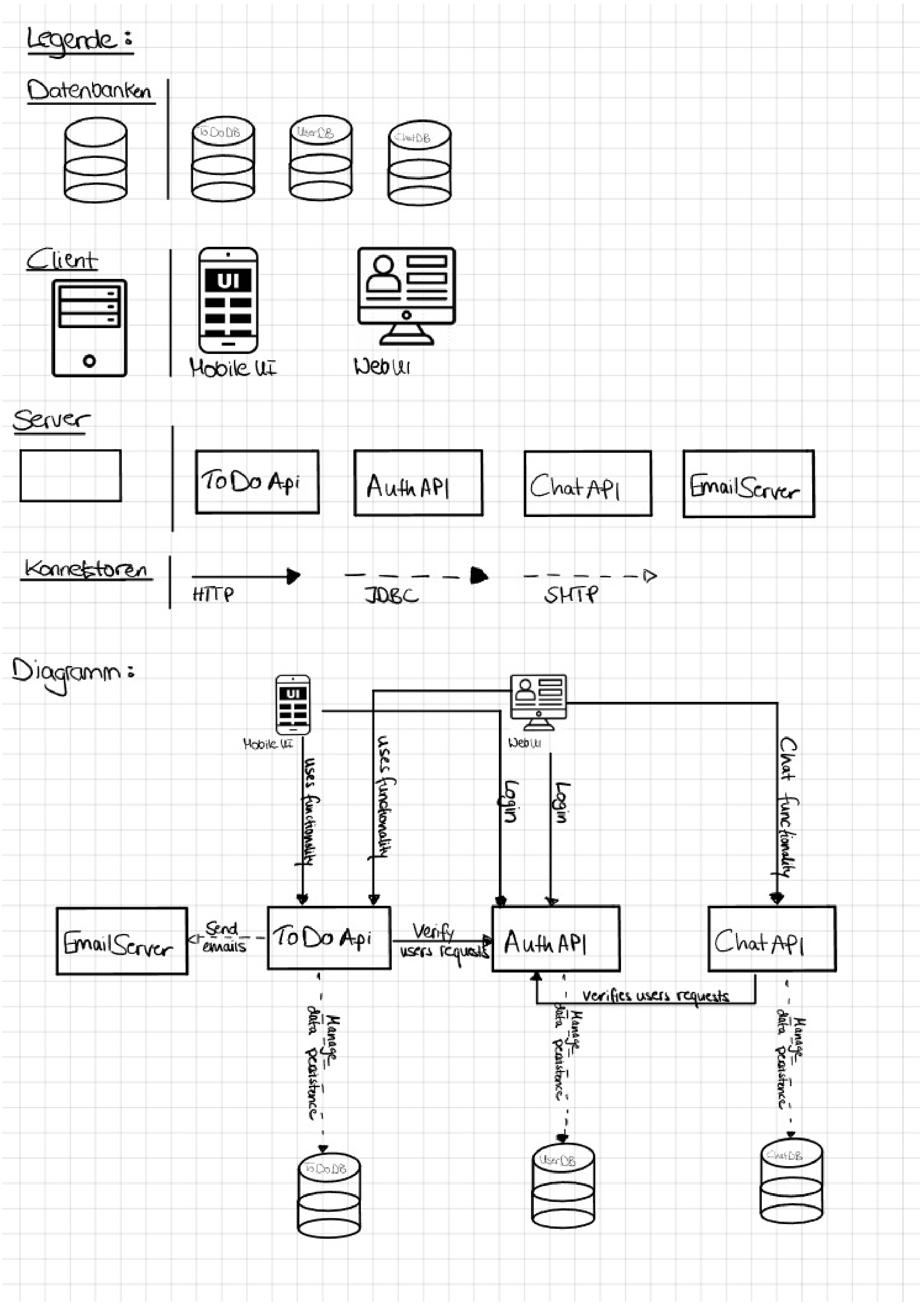
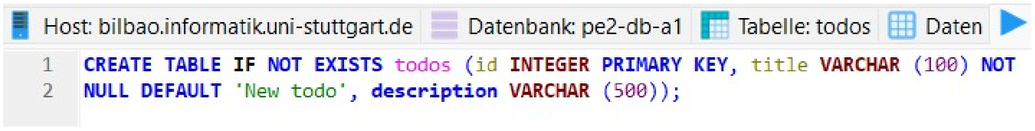
Exercise 1: Architektur



Exercise 2: SQL und JDBC

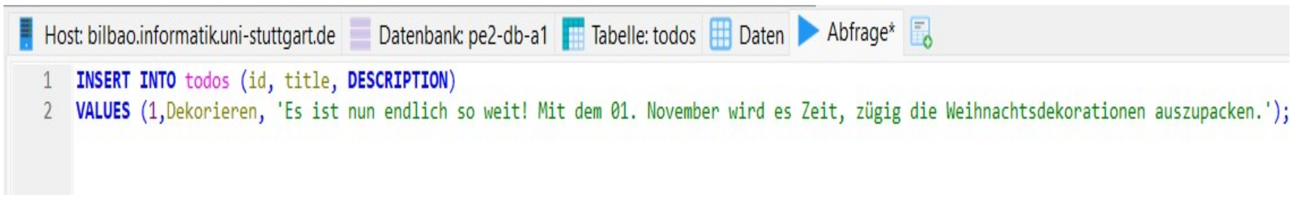
2.1 Werkzeugunterstützte SQL-Befehle

(a) .



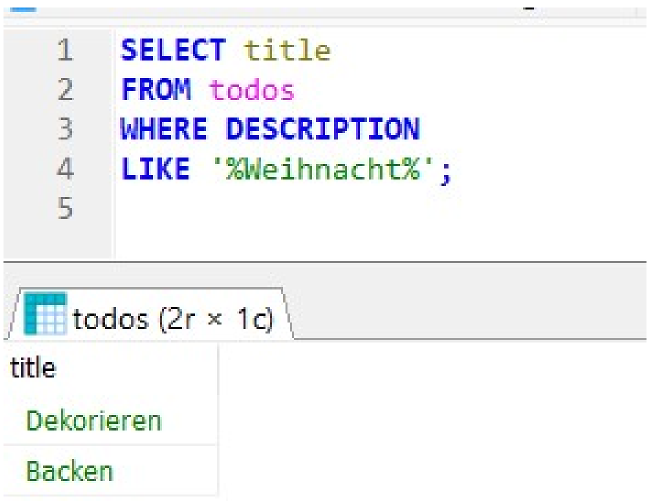
```
CREATE TABLE IF NOT EXISTS todos (id INTEGER PRIMARY KEY, title VARCHAR (100) NOT
NULL DEFAULT 'New todo', description VARCHAR (500));
```

(b) .



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

(c) .

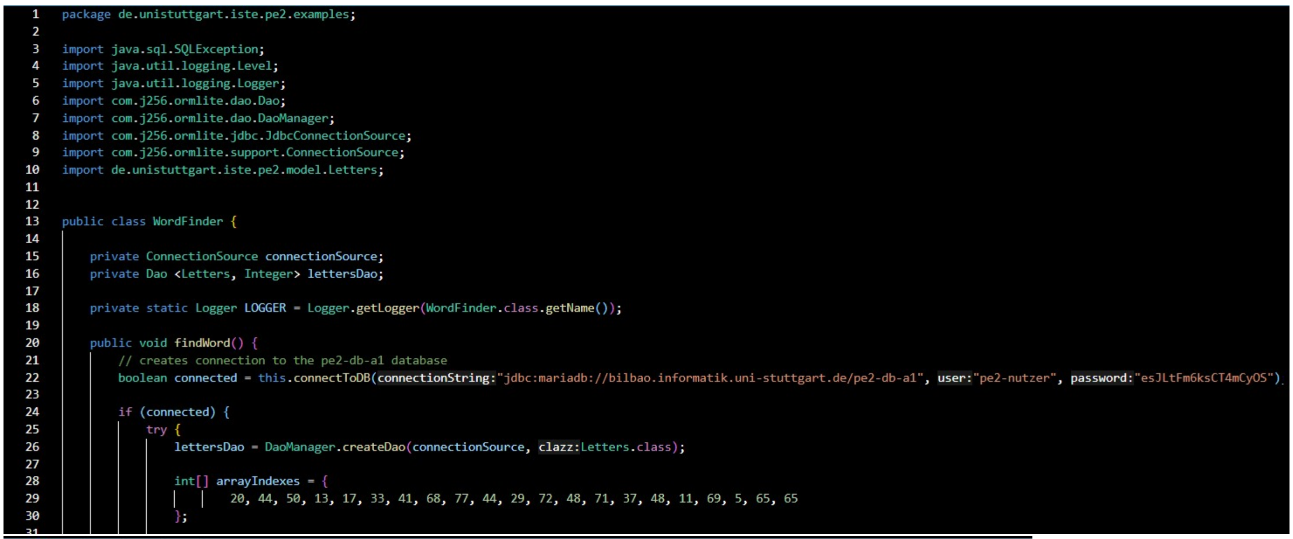


```
SELECT title
FROM todos
WHERE DESCRIPTION
LIKE '%Weihnacht%';
```

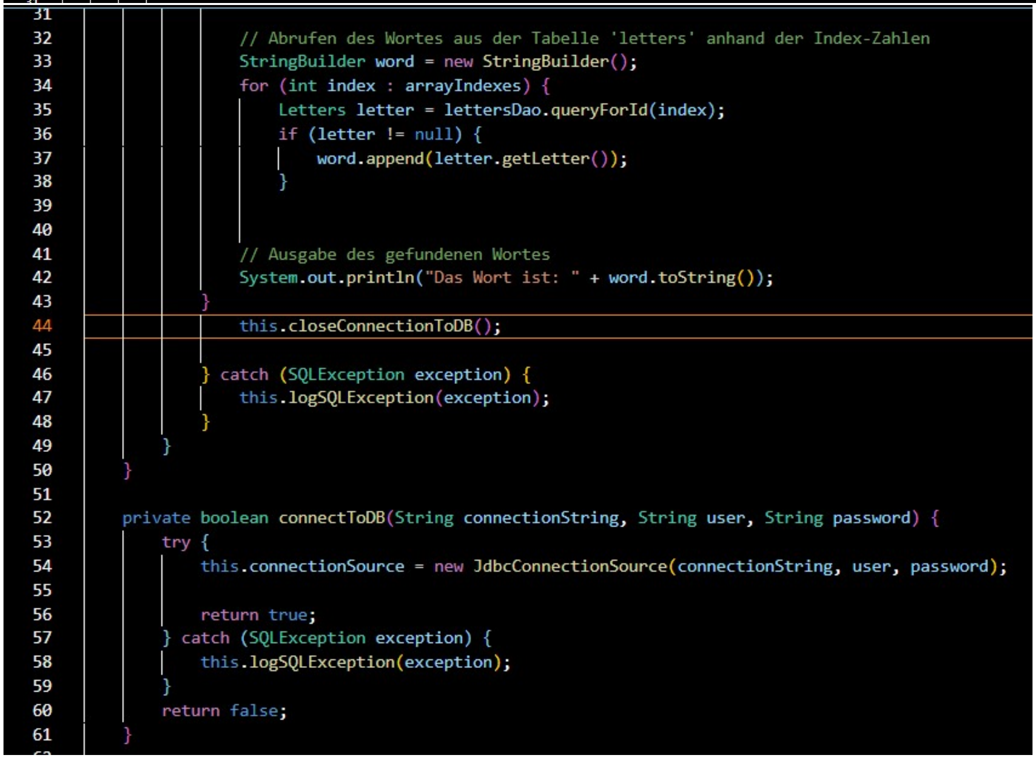
title
Dekorieren
Backen

2.2 Programmatische Datenbankabfrage

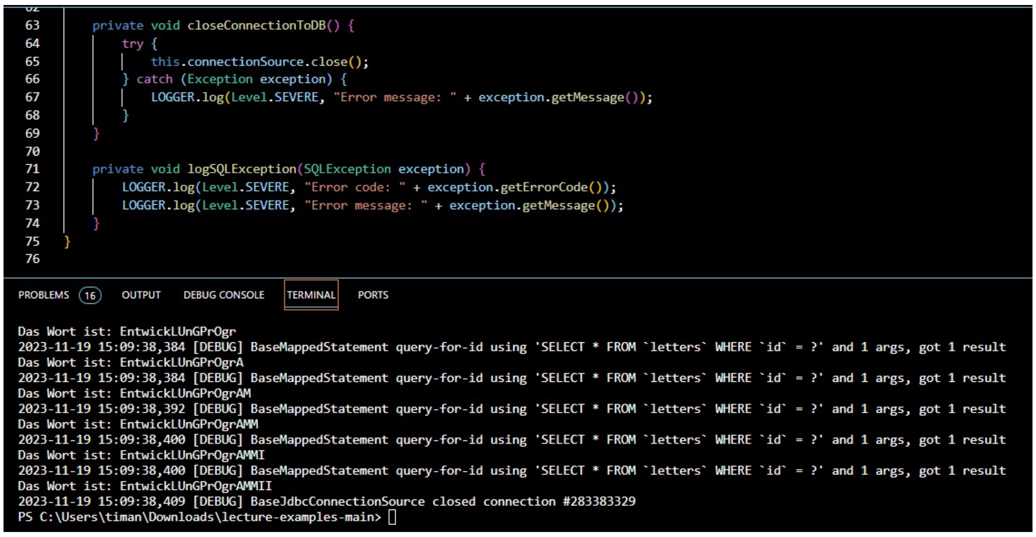
(a) .



```java
package de.unistuttgart.iste.pe2.examples;

import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import de.unistuttgart.iste.pe2.model.Letters;


public class WordFinder {

    private ConnectionSource connectionSource;
    private Dao <Letters, Integer> lettersDao;

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

    public void findWord() {
        // 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);

                int[] arrayIndexes = {
                        20, 44, 50, 13, 17, 33, 41, 68, 77, 44, 29, 72, 48, 71, 37, 48, 11, 69, 5, 65, 65
                };
```



```java

                // Abrufen des Wortes aus der Tabelle 'letters' anhand der Index-Zahlen
                StringBuilder word = new StringBuilder();
                for (int index : arrayIndexes) {
                    Letters letter = lettersDao.queryForId(index);
                    if (letter != null) {
                        word.append(letter.getLetter());
                    }


                // Ausgabe des gefundenen Wortes
                System.out.println("Das Wort ist: " + word.toString());
            }
                this.closeConnectionToDB();

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

    private boolean connectToDB(String connectionString, String user, String password) {
        try {
            this.connectionSource = new JdbcConnectionSource(connectionString, user, password);

            return true;
        } catch (SQLException exception) {
            this.logSQLException(exception);
        }
        return false;
    }
```

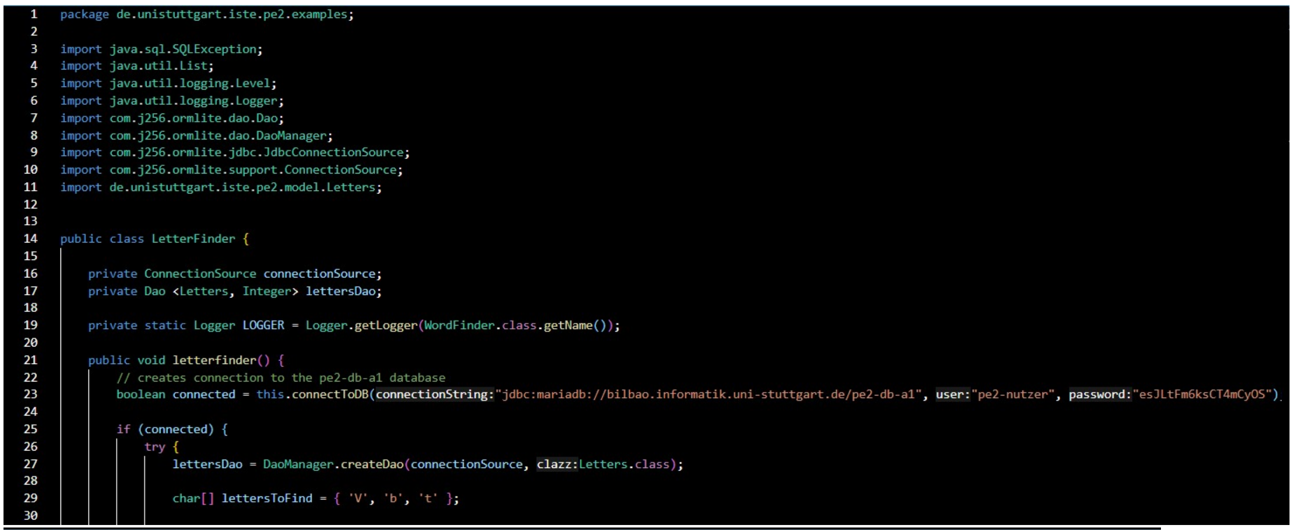


```
    private void closeConnectionToDB() {
        try {
            this.connectionSource.close();
        } catch (Exception exception) {
            LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
        }
    }

    private void logSQLException(SQLException exception) {
        LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
        LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
    }
}

```

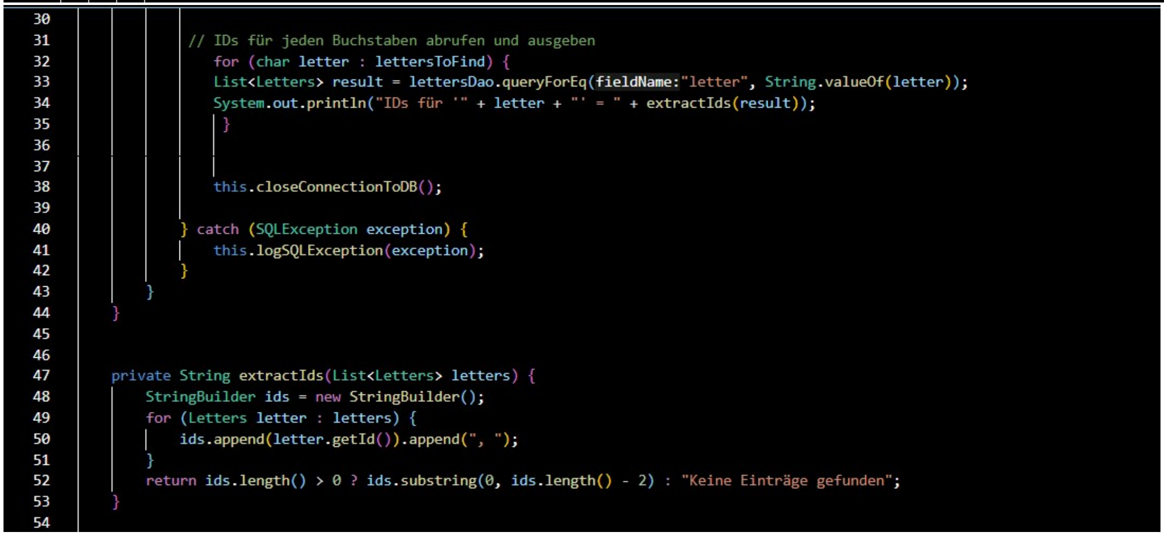
```
Das Wort ist: EntwickLUnGPrOgr
2023-11-19 15:09:38,384 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `id` = ?' and 1 args, got 1 result
Das Wort ist: EntwickLUnGPrOgrA
2023-11-19 15:09:38,384 [DEBUG] BaseMappedStatement query-for-id using 'SELECT * FROM `letters` WHERE `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

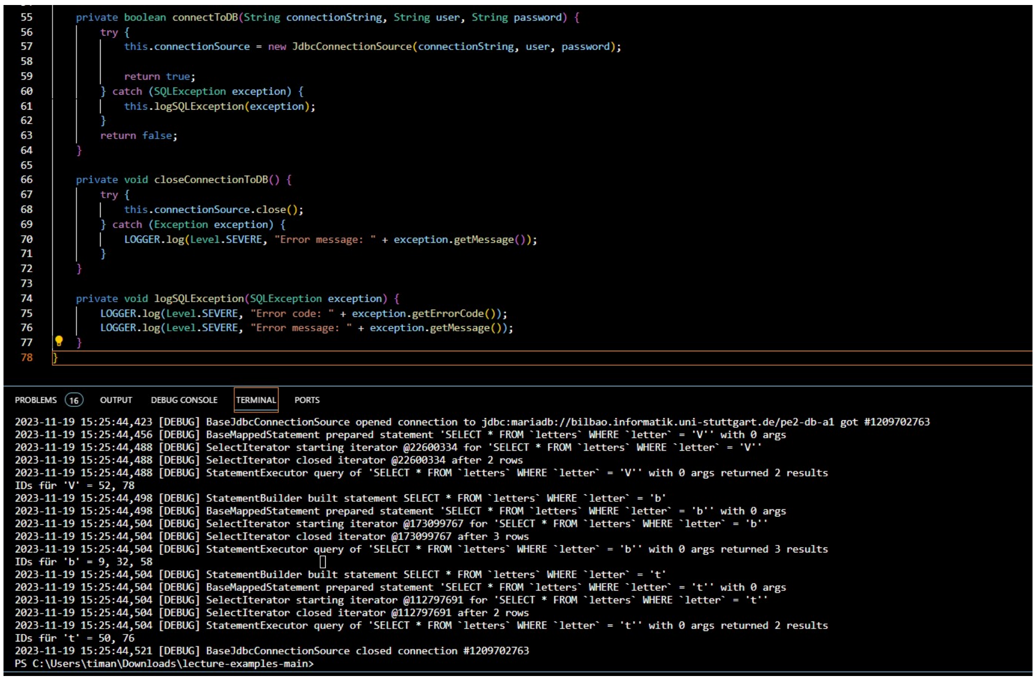
(b) .



```
package de.unistuttgart.iste.pe2.examples;

import java.sql.SQLException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import de.unistuttgart.iste.pe2.model.Letters;


public class LetterFinder {

    private ConnectionSource connectionSource;
    private Dao <Letters, Integer> lettersDao;

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

    public void letterfinder() {
        // 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);

                char[] lettersToFind = { 'V', 'b', 't' };

```



```

                // IDs für jeden Buchstaben abrufen und ausgeben
                for (char letter : lettersToFind) {
                List<Letters> result = lettersDao.queryForEq(fieldName:"letter", String.valueOf(letter));
                System.out.println("IDs für '" + letter + "' = " + extractIds(result));
                 }


                this.closeConnectionToDB();

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


    private String extractIds(List<Letters> letters) {
        StringBuilder ids = new StringBuilder();
        for (Letters letter : letters) {
            ids.append(letter.getId()).append(", ");
        }
        return ids.length() > 0 ? ids.substring(0, ids.length() - 2) : "Keine Einträge gefunden";
    }

```



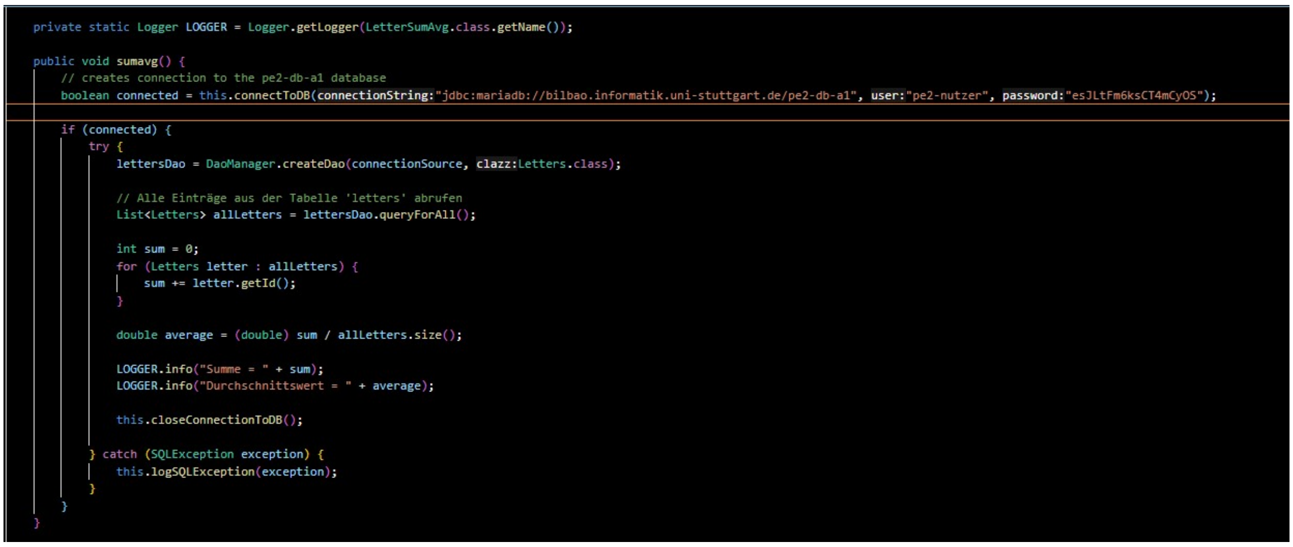
```java
    private boolean connectToDB(String connectionString, String user, String password) {
        try {
            this.connectionSource = new JdbcConnectionSource(connectionString, user, password);

            return true;
        } catch (SQLException exception) {
            this.logSQLException(exception);
        }
        return false;
    }

    private void closeConnectionToDB() {
        try {
            this.connectionSource.close();
        } catch (Exception exception) {
            LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
        }
    }

    private void logSQLException(SQLException exception) {
        LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
        LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
    }
}
```

PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL PORTS

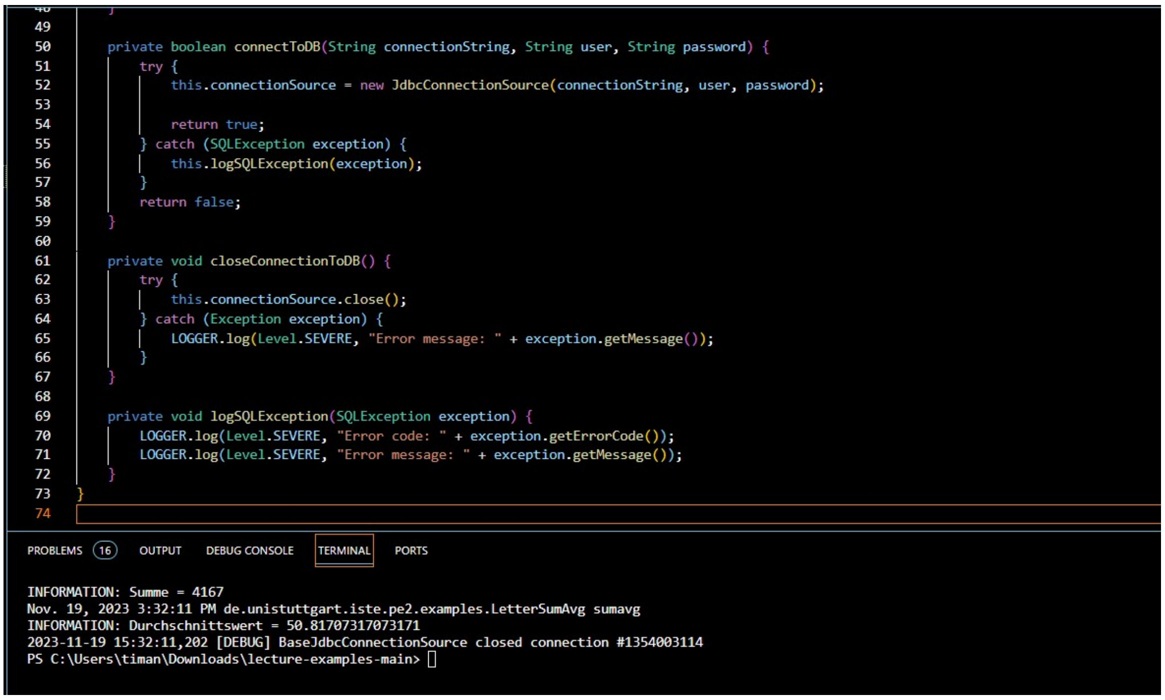
```
2023-11-19 15:25:44,423 [DEBUG] BaseJdbcConnectionSource opened connection to jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1 got #1209702763
2023-11-19 15:25:44,456 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM `letters` WHERE `letter` = 'V'' with 0 args
2023-11-19 15:25:44,488 [DEBUG] SelectIterator starting iterator @22600334 for 'SELECT * FROM `letters` WHERE `letter` = 'V''
2023-11-19 15:25:44,488 [DEBUG] SelectIterator closed iterator @22600334 after 2 rows
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) .



```java
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);
        }
    }
}
```



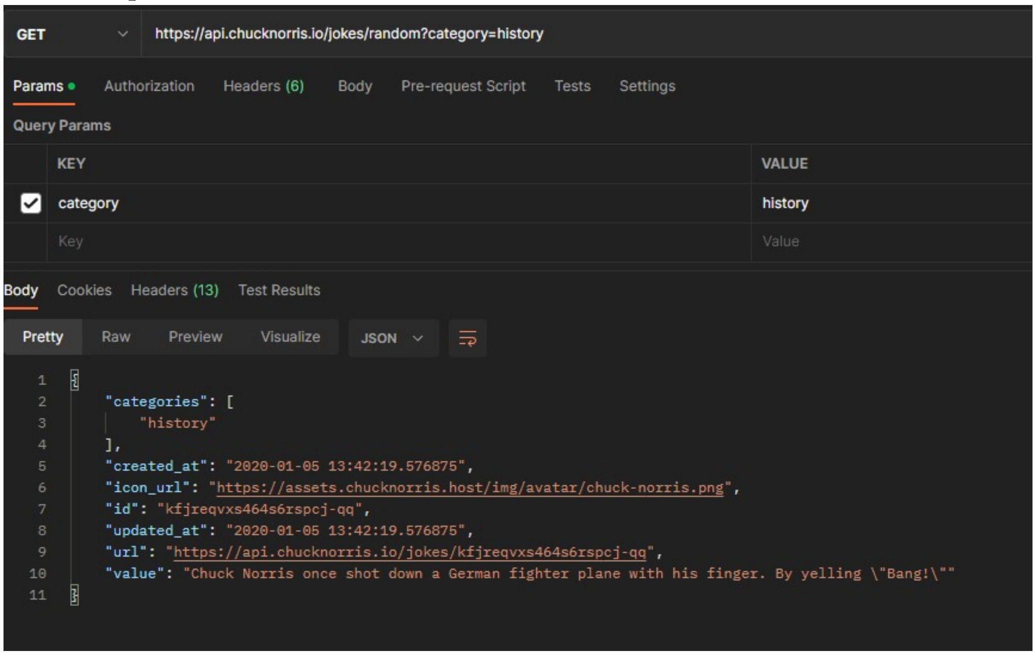
```
    private boolean connectToDB(String connectionString, String user, String password) {
        try {
            this.connectionSource = new JdbcConnectionSource(connectionString, user, password);

            return true;
        } catch (SQLException exception) {
            this.logSQLException(exception);
        }
        return false;
    }

    private void closeConnectionToDB() {
        try {
            this.connectionSource.close();
        } catch (Exception exception) {
            LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
        }
    }

    private void logSQLException(SQLException exception) {
        LOGGER.log(Level.SEVERE, "Error code: " + exception.getErrorCode());
        LOGGER.log(Level.SEVERE, "Error message: " + exception.getMessage());
    }
}
```

```
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

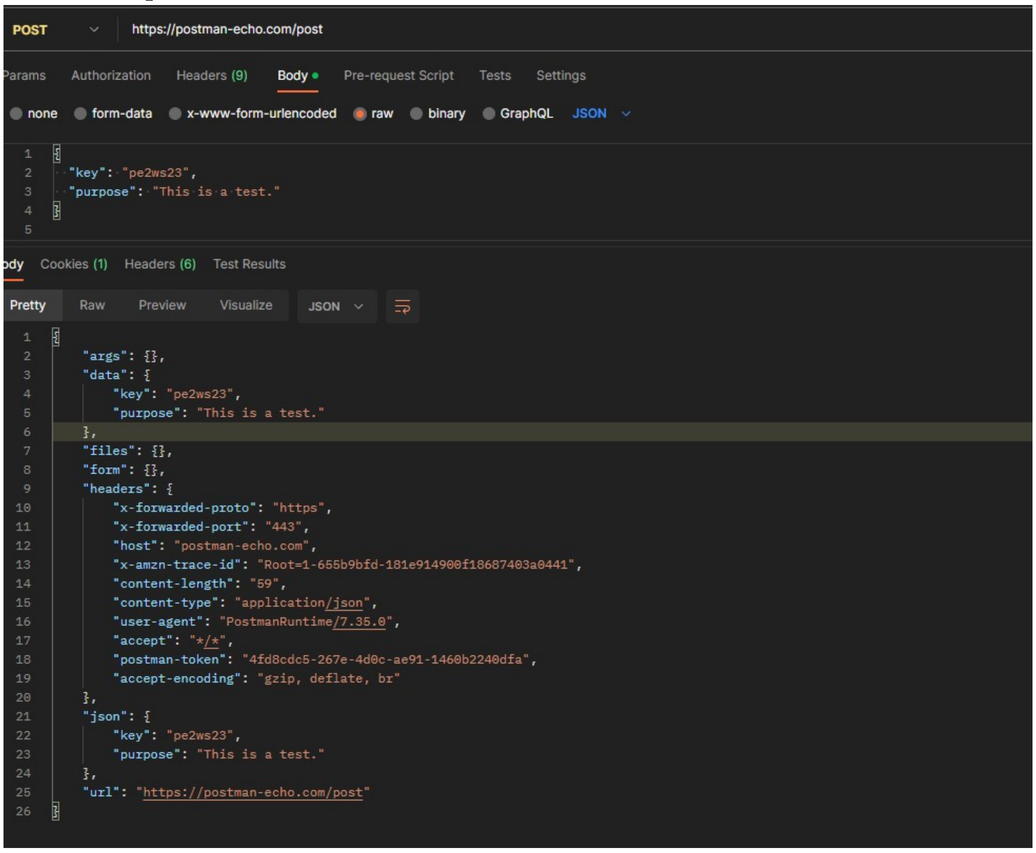


GET https://api.chucknorris.io/jokes/random?category=history

KEY	VALUE
✓ category	history
Key	Value

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

(b) POST-Request senden



```
{
  "key": "pe2ws23",
  "purpose": "This is a test."
}
```

```
{
    "args": {},
    "data": {
        "key": "pe2ws23",
        "purpose": "This is a test."
    },
    "files": {},
    "form": {},
    "headers": {
        "x-forwarded-proto": "https",
        "x-forwarded-port": "443",
        "host": "postman-echo.com",
        "x-amzn-trace-id": "Root=1-655b9bfd-181e914900f18687403a0441",
        "content-length": "59",
        "content-type": "application/json",
        "user-agent": "PostmanRuntime/7.35.0",
        "accept": "*/*",
        "postman-token": "4fd8cdc5-267e-4d0c-ae91-1460b2240dfa",
        "accept-encoding": "gzip, deflate, br"
    },
    "json": {
        "key": "pe2ws23",
        "purpose": "This is a test."
    },
    "url": "https://postman-echo.com/post"
}
```

(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