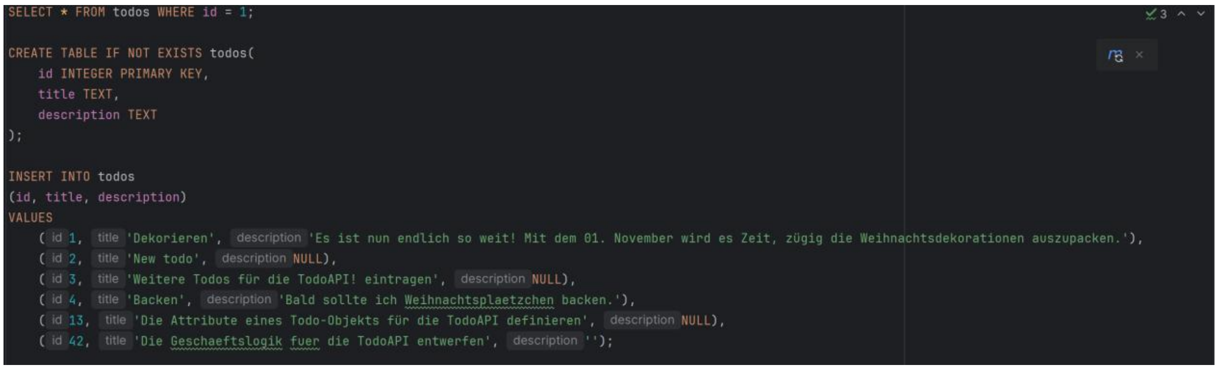
Aran Moradi st192331@stud.uni-stuttgart.de

Programmentwicklung 2
Übungsblatt 1

Aufgabe 1

1.1

a)

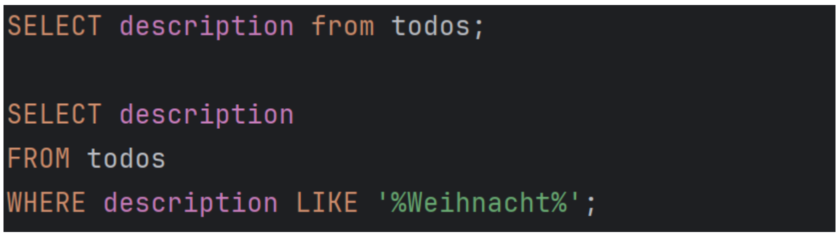


```sql
SELECT * FROM todos WHERE id = 1;

CREATE TABLE IF NOT EXISTS todos(
    id INTEGER PRIMARY KEY,
    title TEXT,
    description TEXT
);

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.'),
    (2, 'New todo', NULL),
    (3, 'Weitere Todos für die TodoAPI! eintragen', NULL),
    (4, 'Backen', 'Bald sollte ich Weihnachtsplaetzchen backen.'),
    (13, 'Die Attribute eines Todo-Objekts für die TodoAPI definieren', NULL),
    (42, 'Die Geschaeftslogik fuer die TodoAPI entwerfen', '');
```

b)

```sql
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.'),
```

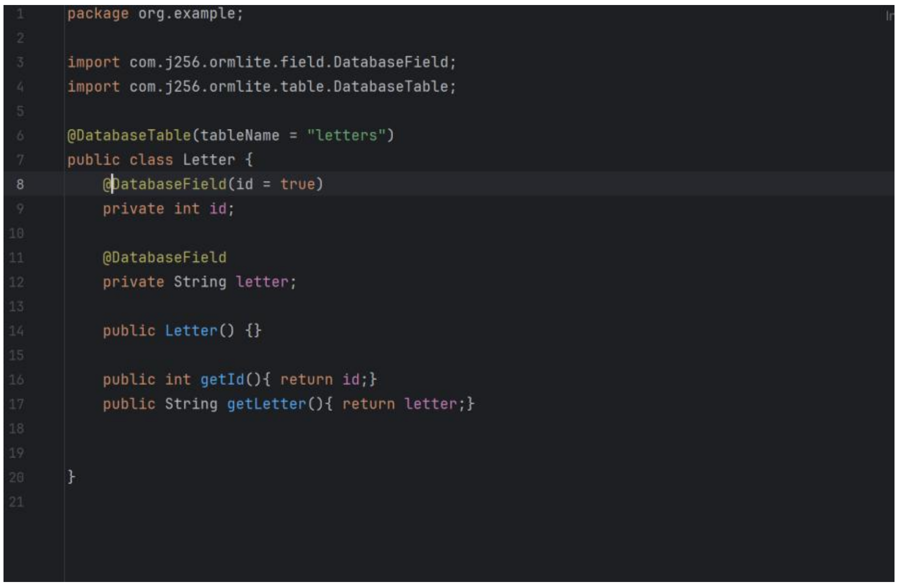
c)



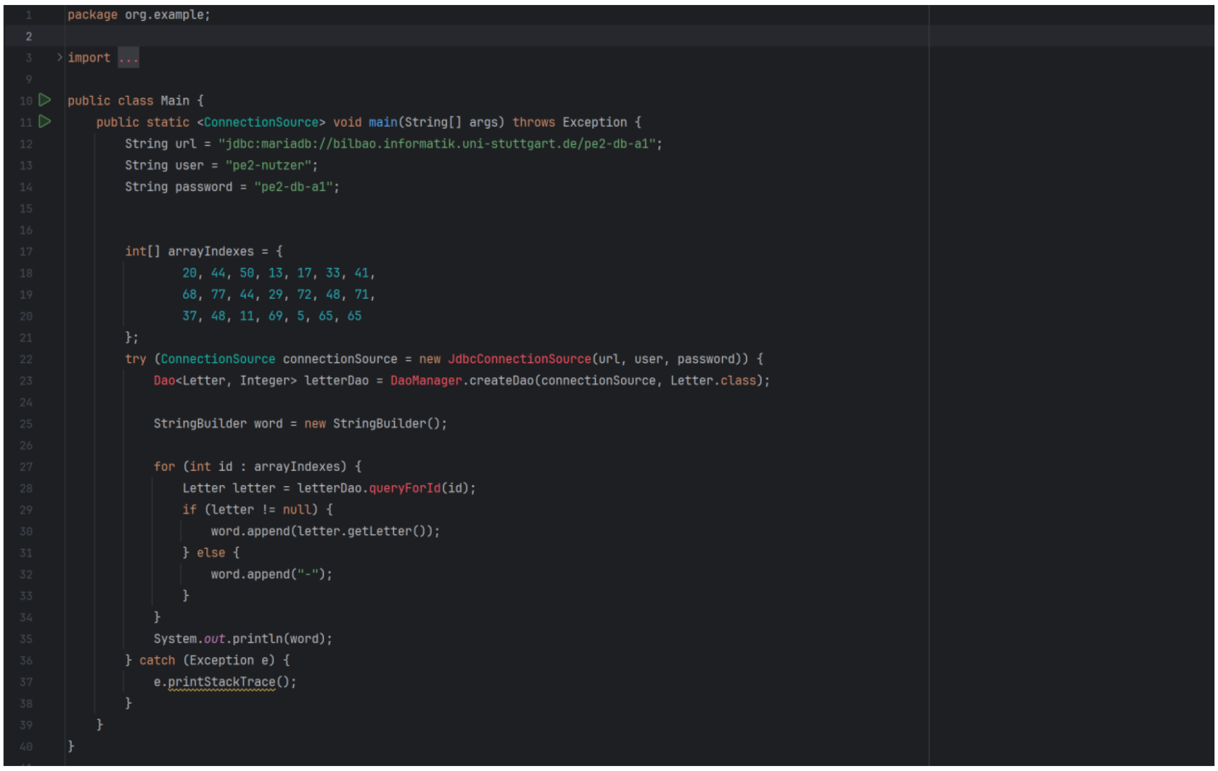
```sql
SELECT description from todos;

SELECT description
FROM todos
WHERE description LIKE '%Weihnacht%';
```

1.2a)

Lösungswort: EntwickLUnGPrOgrAMMII



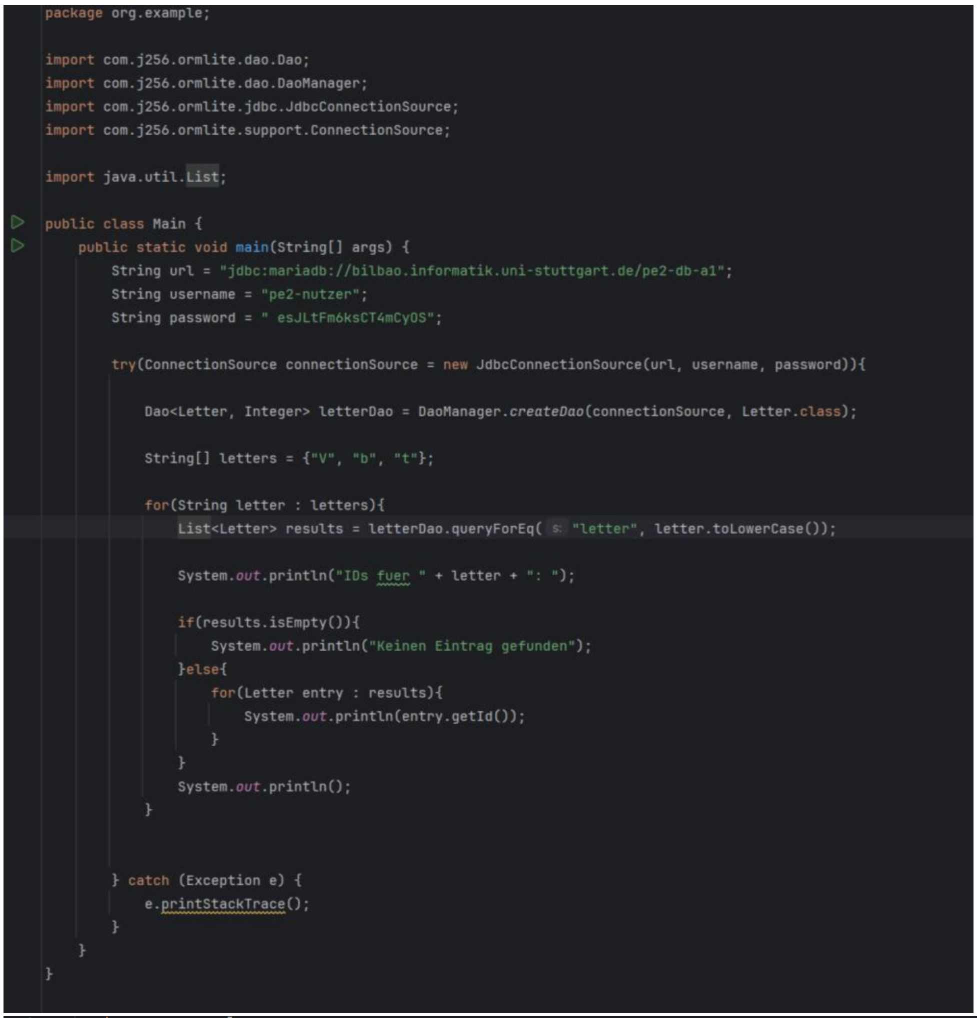
```java
package org.example;

import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.table.DatabaseTable;

@DatabaseTable(tableName = "letters")
public class Letter {
    @DatabaseField(id = true)
    private int id;

    @DatabaseField
    private String letter;

    public Letter() {}

    public int getId(){ return id;}
    public String getLetter(){ return letter;}


}
```



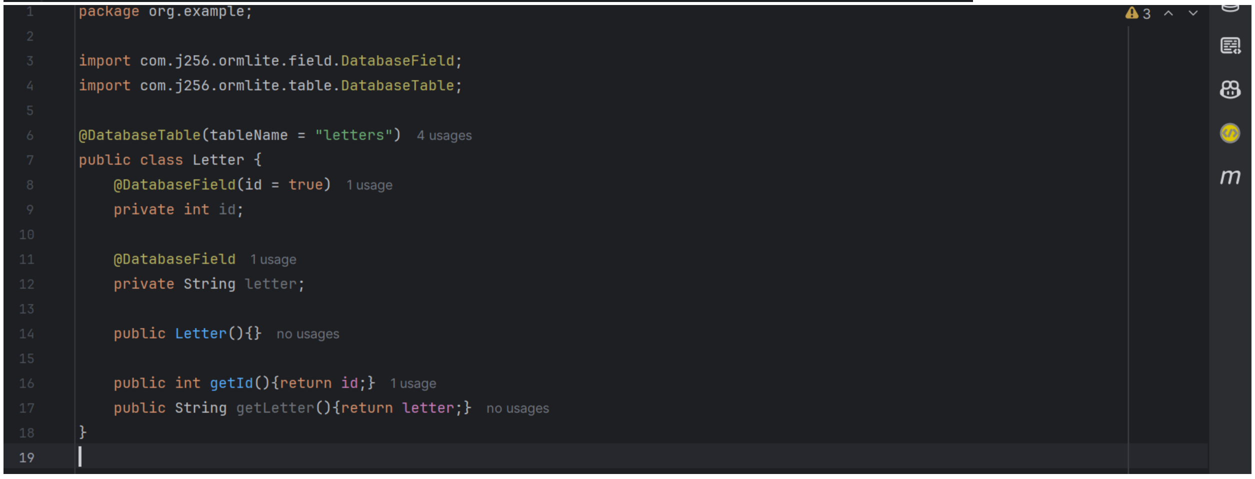
```java
package org.example;

import ...

public class Main {
    public static <ConnectionSource> void main(String[] args) throws Exception {
        String url = "jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1";
        String user = "pe2-nutzer";
        String password = "pe2-db-a1";


        int[] arrayIndexes = {
                20, 44, 50, 13, 17, 33, 41,
                68, 77, 44, 29, 72, 48, 71,
                37, 48, 11, 69, 5, 65, 65
        };
        try (ConnectionSource connectionSource = new JdbcConnectionSource(url, user, password)) {
            Dao<Letter, Integer> letterDao = DaoManager.createDao(connectionSource, Letter.class);

            StringBuilder word = new StringBuilder();

            for (int id : arrayIndexes) {
                Letter letter = letterDao.queryForId(id);
                if (letter != null) {
                    word.append(letter.getLetter());
                } else {
                    word.append("-");
                }
            }
            System.out.println(word);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

(Problem mit den Imports gehabt deswegen rote Tokens)

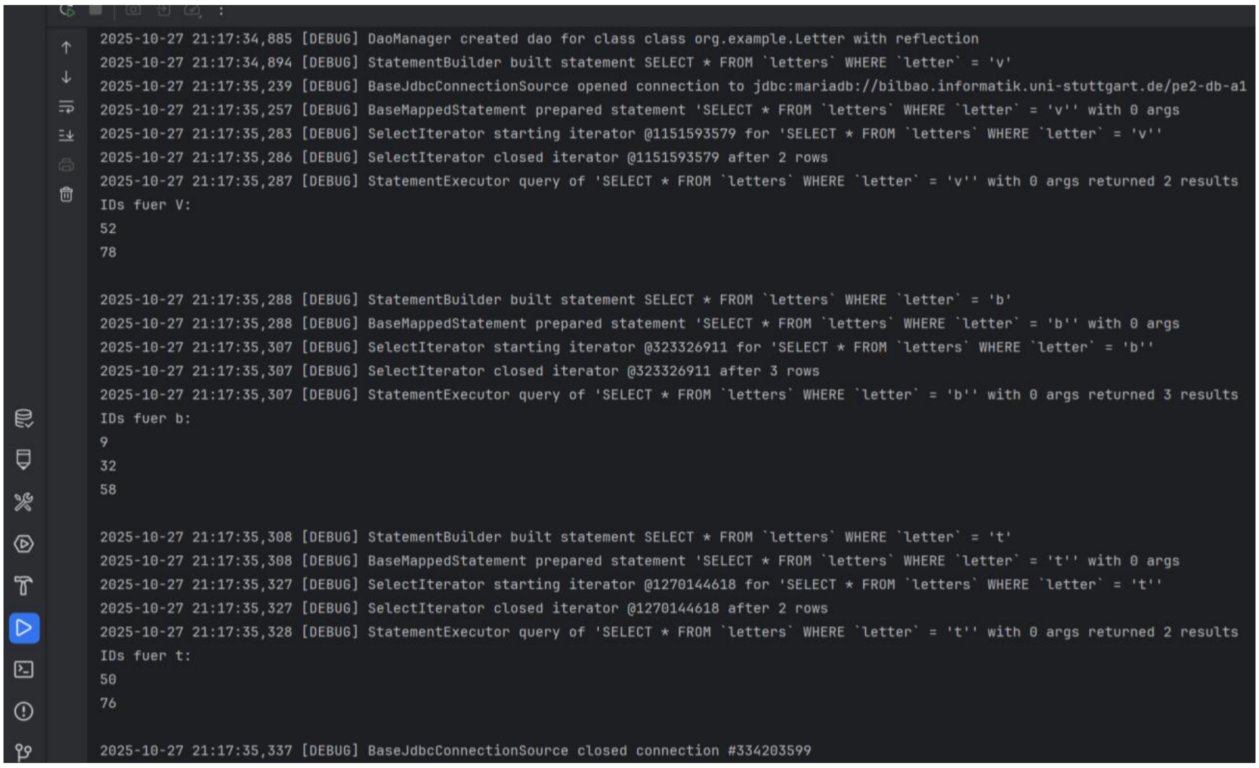
b)



```java
package org.example;

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 java.util.List;

public class Main {
    public static void main(String[] args) {
        String url = "jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1";
        String username = "pe2-nutzer";
        String password = " esJLtFm6ksCT4mCyOS";

        try(ConnectionSource connectionSource = new JdbcConnectionSource(url, username, password)){

            Dao<Letter, Integer> letterDao = DaoManager.createDao(connectionSource, Letter.class);

            String[] letters = {"V", "b", "t"};

            for(String letter : letters){
                List<Letter> results = letterDao.queryForEq( s: "letter", letter.toLowerCase());

                System.out.println("IDs fuer " + letter + ": ");

                if(results.isEmpty()){
                    System.out.println("Keinen Eintrag gefunden");
                }else{
                    for(Letter entry : results){
                        System.out.println(entry.getId());
                    }
                }
                System.out.println();
            }


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



```java
package org.example;

import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.table.DatabaseTable;

@DatabaseTable(tableName = "letters")  4 usages
public class Letter {
    @DatabaseField(id = true)  1 usage
    private int id;

    @DatabaseField  1 usage
    private String letter;

    public Letter(){}  no usages

    public int getId(){return id;}  1 usage
    public String getLetter(){return letter;}  no usages
}

```



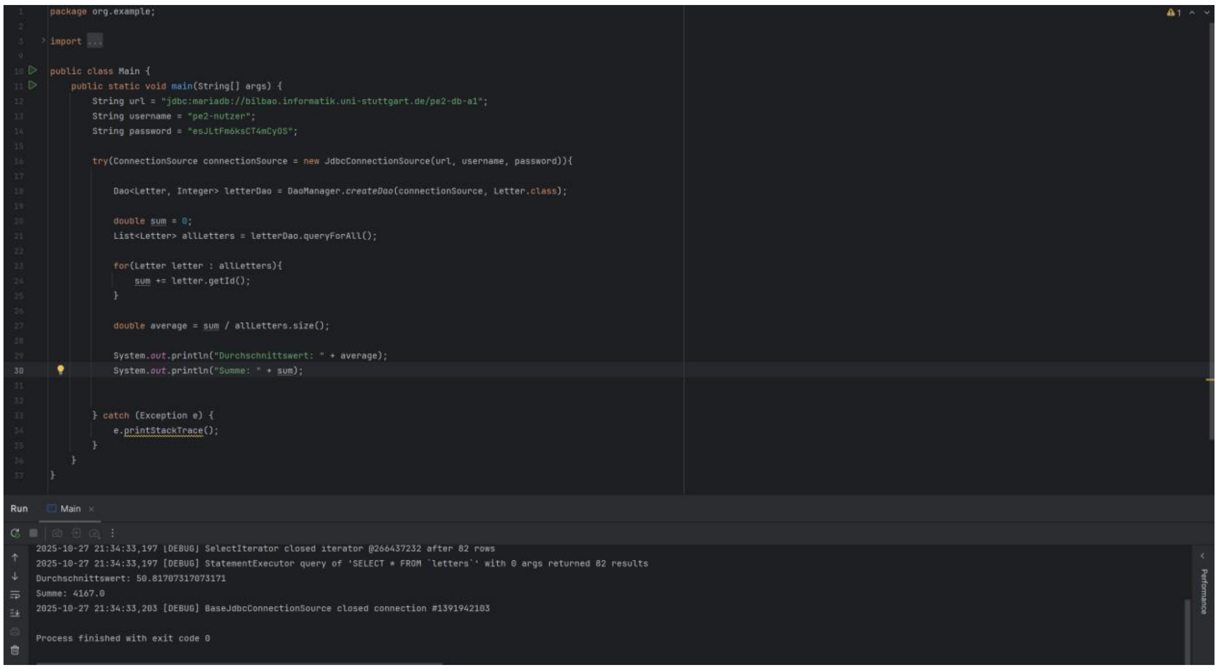
```
2025-10-27 21:17:34,885 [DEBUG] DaoManager created dao for class class org.example.Letter with reflection
2025-10-27 21:17:34,894 [DEBUG] StatementBuilder built statement SELECT * FROM `letters` WHERE `letter` = 'v'
2025-10-27 21:17:35,239 [DEBUG] BaseJdbcConnectionSource opened connection to jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1
2025-10-27 21:17:35,257 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM `letters` WHERE `letter` = 'v'' with 0 args
2025-10-27 21:17:35,283 [DEBUG] SelectIterator starting iterator @1151593579 for 'SELECT * FROM `letters` WHERE `letter` = 'v''
2025-10-27 21:17:35,286 [DEBUG] SelectIterator closed iterator @1151593579 after 2 rows
2025-10-27 21:17:35,287 [DEBUG] StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 'v'' with 0 args returned 2 results
IDs fuer V:
52
78

2025-10-27 21:17:35,288 [DEBUG] StatementBuilder built statement SELECT * FROM `letters` WHERE `letter` = 'b'
2025-10-27 21:17:35,288 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM `letters` WHERE `letter` = 'b'' with 0 args
2025-10-27 21:17:35,307 [DEBUG] SelectIterator starting iterator @323326911 for 'SELECT * FROM `letters` WHERE `letter` = 'b''
2025-10-27 21:17:35,307 [DEBUG] SelectIterator closed iterator @323326911 after 3 rows
2025-10-27 21:17:35,307 [DEBUG] StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 'b'' with 0 args returned 3 results
IDs fuer b:
9
32
58

2025-10-27 21:17:35,308 [DEBUG] StatementBuilder built statement SELECT * FROM `letters` WHERE `letter` = 't'
2025-10-27 21:17:35,308 [DEBUG] BaseMappedStatement prepared statement 'SELECT * FROM `letters` WHERE `letter` = 't'' with 0 args
2025-10-27 21:17:35,327 [DEBUG] SelectIterator starting iterator @1270144618 for 'SELECT * FROM `letters` WHERE `letter` = 't''
2025-10-27 21:17:35,327 [DEBUG] SelectIterator closed iterator @1270144618 after 2 rows
2025-10-27 21:17:35,328 [DEBUG] StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 't'' with 0 args returned 2 results
IDs fuer t:
50
76

2025-10-27 21:17:35,337 [DEBUG] BaseJdbcConnectionSource closed connection #334203599
```

IDs für V:
52, 78

IDs für b:
9, 32, 58

IDs für t:
50, 76

c)



```java
package org.example;

import ...

public class Main {
    public static void main(String[] args) {
        String url = "jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1";
        String username = "pe2-nutzer";
        String password = "esJLtFm6ksCT4mCy0S";

        try(ConnectionSource connectionSource = new JdbcConnectionSource(url, username, password)){

            Dao<Letter, Integer> letterDao = DaoManager.createDao(connectionSource, Letter.class);

            double sum = 0;
            List<Letter> allLetters = letterDao.queryForAll();

            for(Letter letter : allLetters){
                sum += letter.getId();
            }

            double average = sum / allLetters.size();

            System.out.println("Durchschnittswert: " + average);
            System.out.println("Summe: " + sum);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

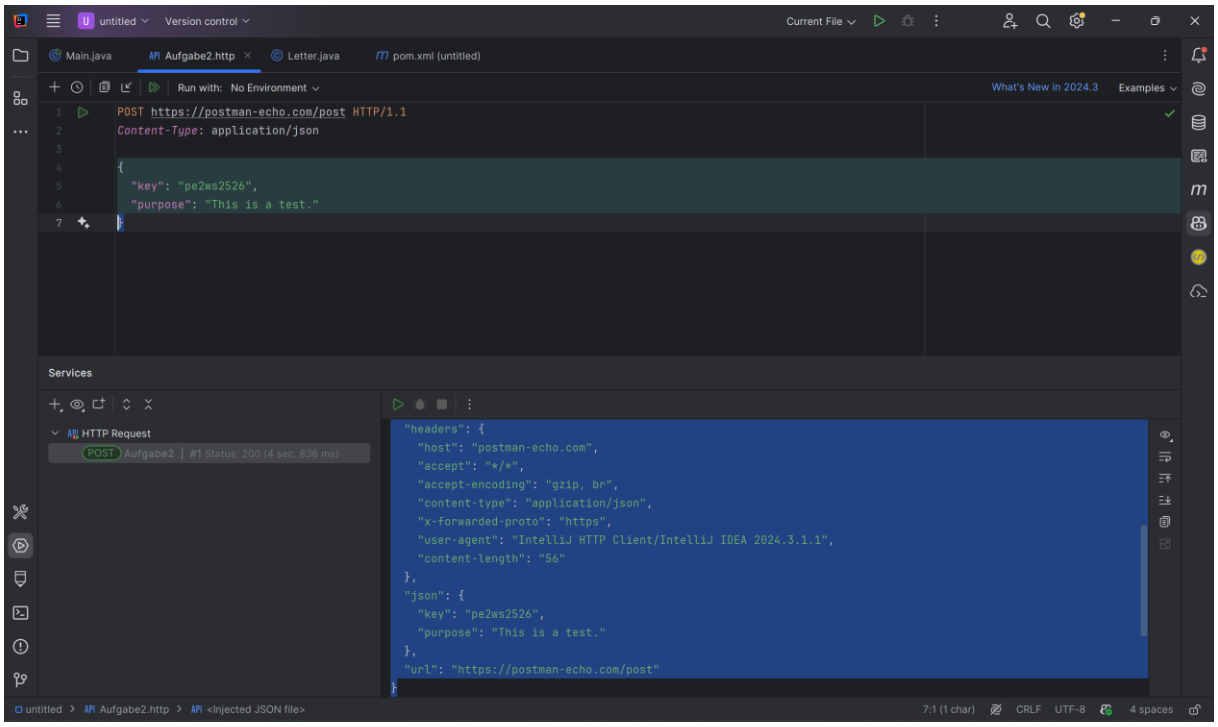
```
2025-10-27 21:34:33,197 [DEBUG] SelectIterator closed iterator @266437232 after 82 rows
2025-10-27 21:34:33,197 [DEBUG] StatementExecutor query of 'SELECT * FROM `letters`' with 0 args returned 82 results
Durchschnittswert: 50.81707317073171
Summe: 4167.0
2025-10-27 21:34:33,203 [DEBUG] BaseJdbcConnectionSource closed connection #1391942103

Process finished with exit code 0
```

Summe = 4167
Durchschnittswert = 50,81707317073171

Aufgabe 2
a)
GET https://api.chucknorris.io/jokes/random?category=sport

```
{
  "categories": [
    "sport"
  ],
  "created_at": "2020-01-05 13:42:19.576875",
  "icon_url": "https://api.chucknorris.io/img/avatar/chuck-norris.png",
  "id": "2o6183z1rmkus1imghxsug",
  "updated_at": "2020-01-05 13:42:19.576875",
  "url": "https://api.chucknorris.io/jokes/2o6183z1rmkus1imghxsug",
  "value": "Chuck Norris won super bowls VII and VIII singlehandedly before unexpectedly retiring to pursue a career in ass-kicking."
}
```

b)



```
POST https://postman-echo.com/post HTTP/1.1
Content-Type: application/json

{
  "key": "pe2ws2526",
  "purpose": "This is a test."
}
```

```
  "headers": {
    "host": "postman-echo.com",
    "accept": "*/*",
    "accept-encoding": "gzip, br",
    "content-type": "application/json",
    "x-forwarded-proto": "https",
    "user-agent": "IntelliJ HTTP Client/IntelliJ IDEA 2024.3.1.1",
    "content-length": "56"
  },
  "json": {
    "key": "pe2ws2526",
    "purpose": "This is a test."
  },
  "url": "https://postman-echo.com/post"
}
```

c)

Create
HTTP Methode: POST
Beschreibung: Diese Methode erstellt eine neue Ressource, also in diesem Fall ein neuen DVD Eintrag in der Datenbank.
Pfad: /dvds

Read ½ (getAll)
HTTP Methode:
GET / <collection-name>
Beschreibung: Diese Methode liefert eine Repräsentation der DVDs. Dabei kann ein Format bzw. Filter festgelegt werden durch einen Query Parameter wie z.B. Titel, Kategorie und Alterseinschraenkung.

GET /dvds?category={category}&title={String}&restricted={boolean}

Read 2/2 (getById)
HTTP Methode:
GET /<collection-name>/$id
Beschreibung: Diese Methode ruft eine bestimmte Ressource einer Sammlung ab, also hier eine spezifische DVD anhand ihrer ID.

GET/dvds/$id

Update
HTTP Methode:
PUT /dvds/{id}
Beschreibung: Überschreibt eine DVD bzw. die ganzen Informationen.

Delete
HTTP Methode:
Beschreibung: Diese Methode entfernt eine DVD aus der Datenbank, entsprechend ihrer ID.

Aufgabe 3
a)
Der Statuscode steht dafür, dass eine Methode nicht erlaubt ist. In diesem Fall bedeutet es, dass die Methode PUT nicht auf cats angewendet werden kann. Für das Aufrufen der aller Katzen muss stattdessen eine andere Methode und zwar die GET Methode verwendet werden.

GET /api/v1/cats

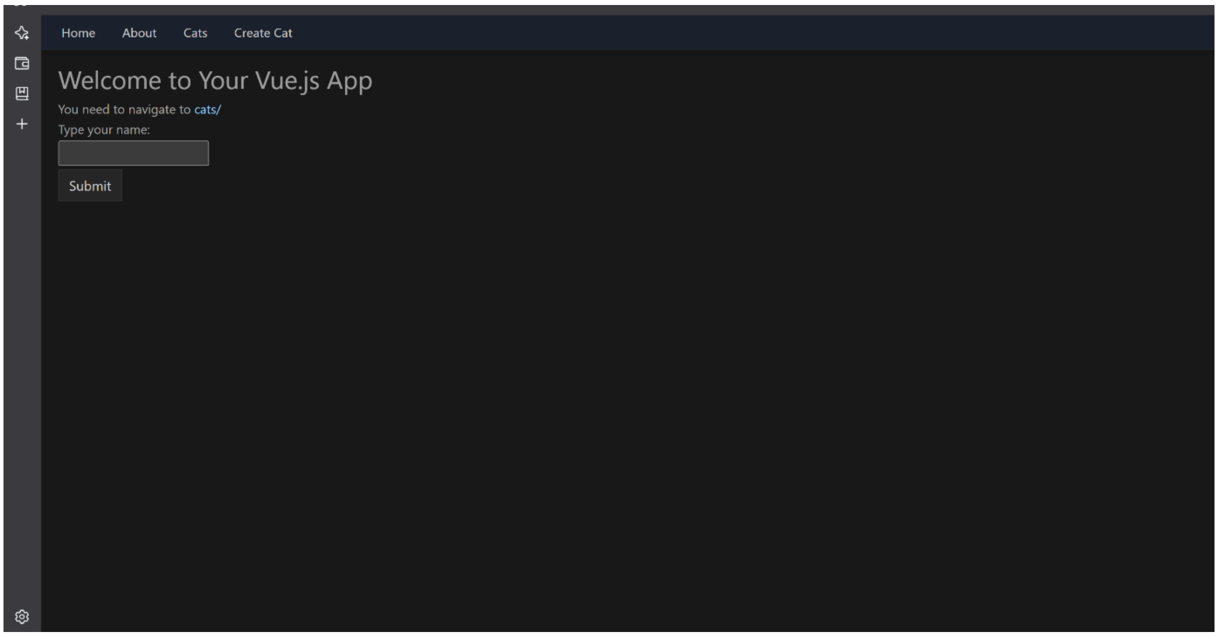
b)

Der Statuscode 400 (Bad Request) deutet darauf hin, dass der Server die Anfrage nicht verarbeiten konnte, wobei der Fehler beim Client liegt, da hier die Anfrage falsch formuliert ist. In diesem Fall wurde der Request Body nicht im JSON Format geschrieben. Der Request Body müsste also folgendermaßen aussehen:

```
{

“name”: “Garfield”
“ageInYears” : 40
“picUrl” : "https://upload.wikimedia.org/wikipedia/en/e/eb/Garfield_ver6.jpg"

}
```

Aufgabe 4

Startseite

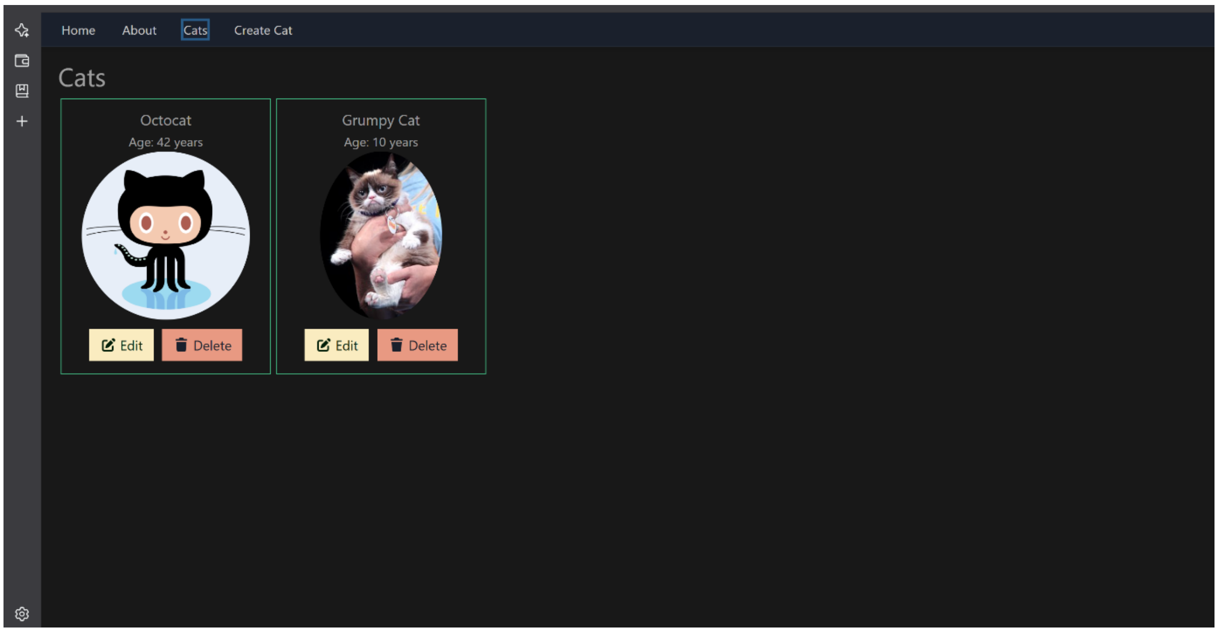


Welcome to Your Vue.js App

You need to navigate to cats/

Type your name:

Cats Seite



Cats

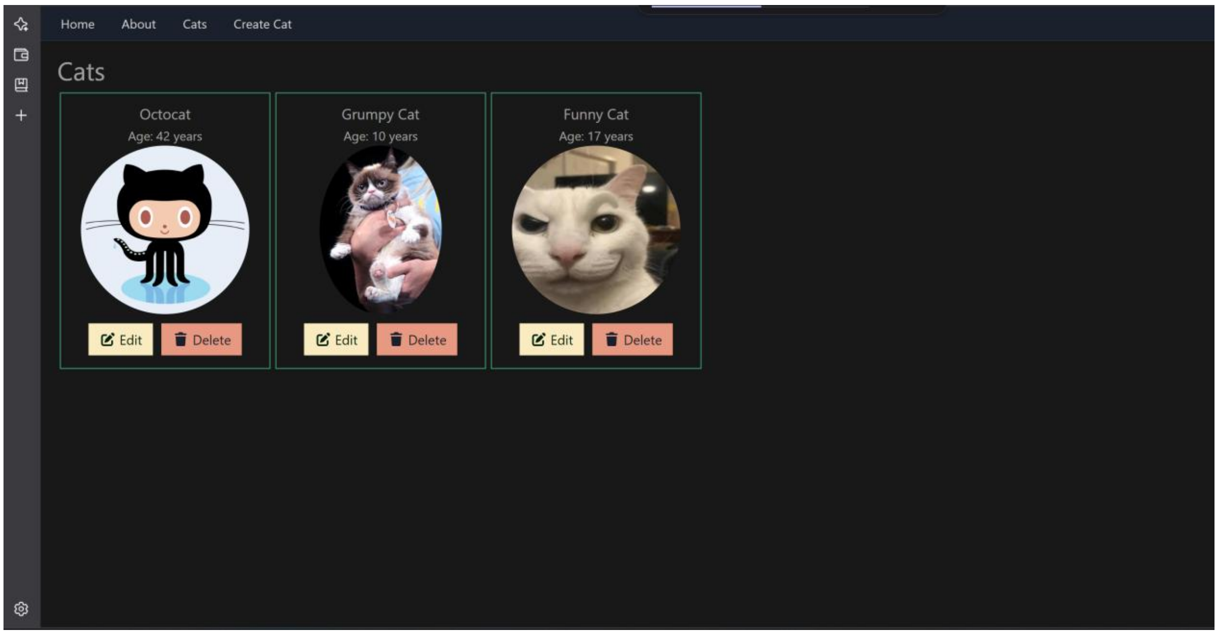
Octocat

Age: 42 years

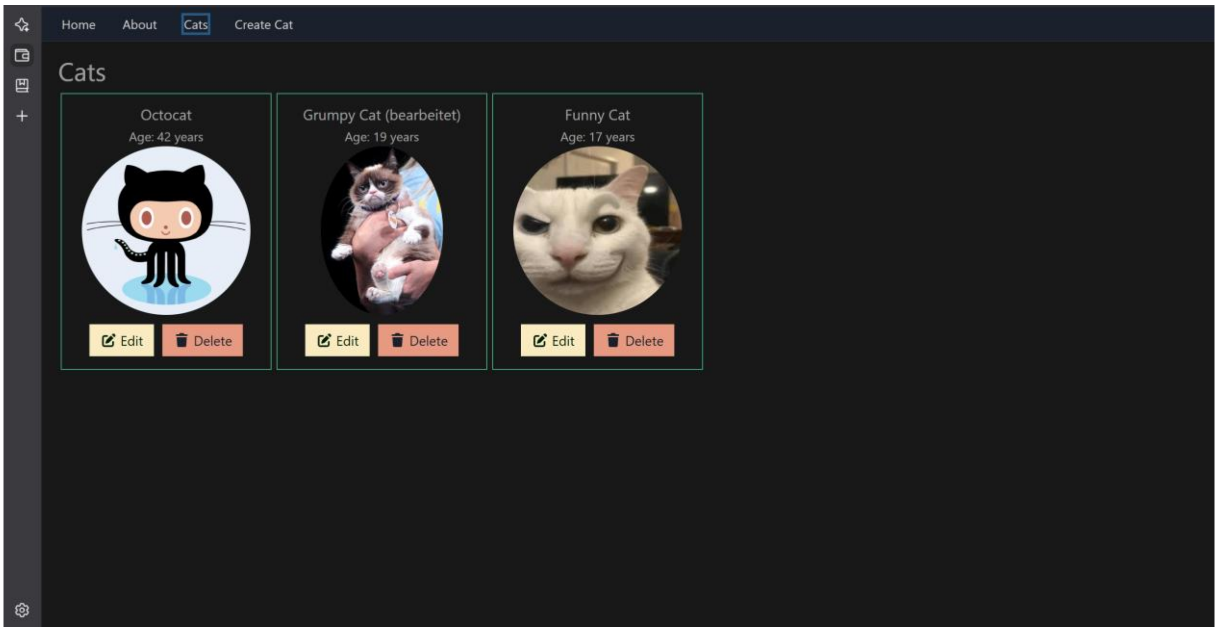
Grumpy Cat

Age: 10 years

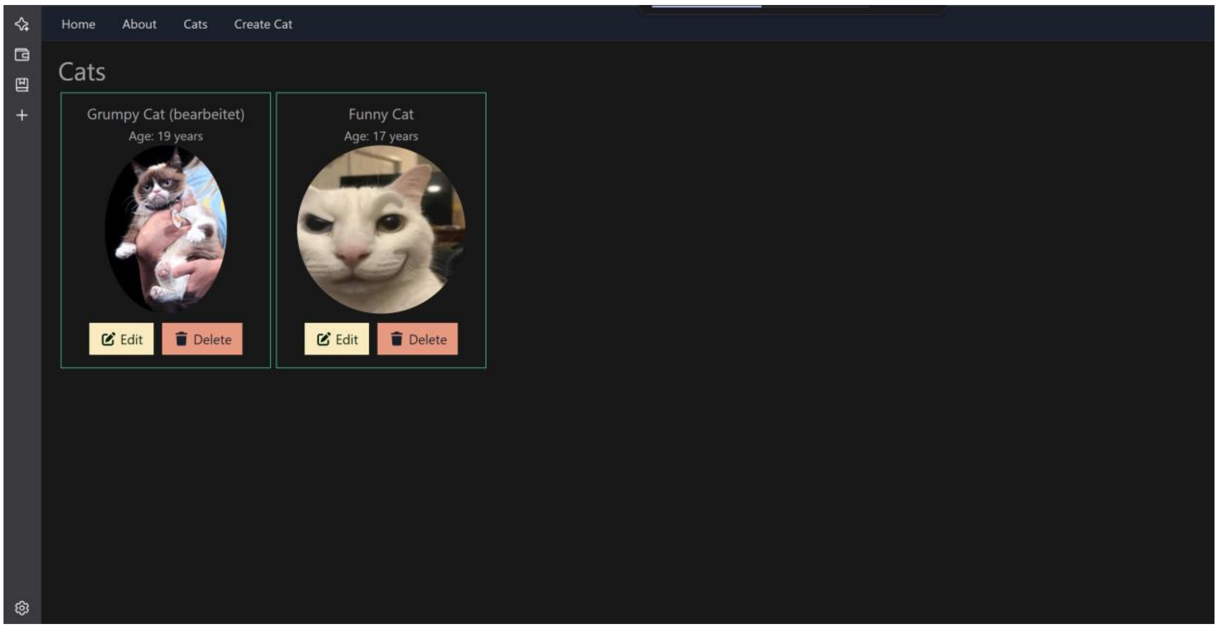
Cats Seite, nachdem eine neue Katze angelegt wurde.



Cats Seite, nachdem eine Katze bearbeitet wurde.



Cats Seite, nachdem eine Katze gelöscht wurde.


Home
About
Cats
Create Cat
Cats
Grumpy Cat (bearbeitet)
Age: 19 years
Edit
Delete
Funny Cat
Age: 17 years
Edit
Delete