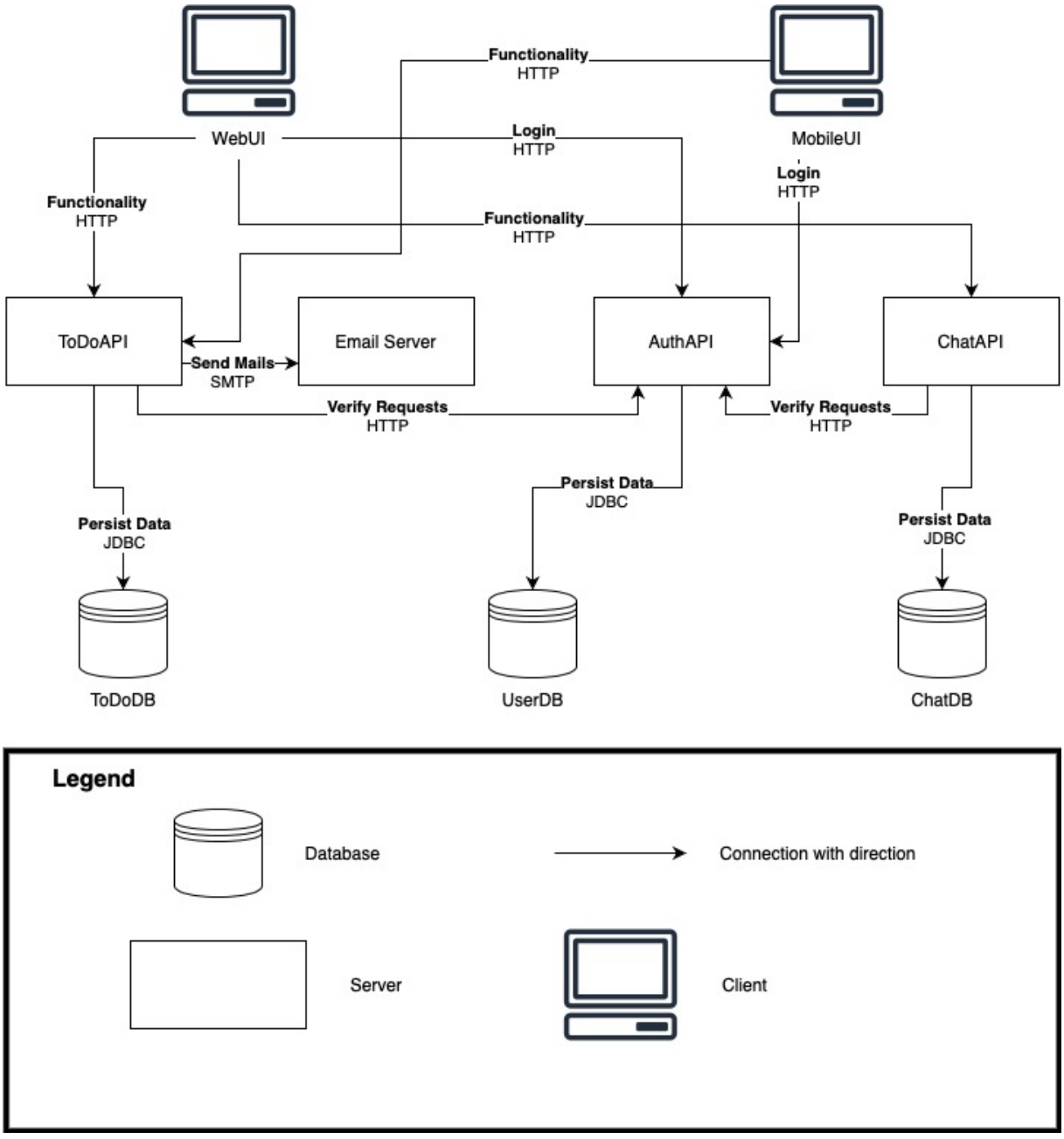
PEII Blatt 1

Aufgabe 1



Aufgabe 2

2.1

a) CREATE TABLE IF NOT EXISTS todos(
id INTEGER PRIMARY KEY,
title VARCHAR(100) NOT NULL CHECK(title <> “”) DEFAULT ‘New todo’,
description VARCHAR(500)
);

b) INSERT INTO todos (id, title, description)

VALUES (1, ‘Einen SQL-Befehl zur Erstellung einer Datenbanktabelle entwerfen‘, ‘Die Tabelle hat drei Spalten (id, title, description); id ist eine Zahl und ist der Primaerschluessel; title und description sind Zeichenfolgen mit maximaler Laenge von 100 und 500; title darf nicht null oder leer sein; Wenn der Tabelle eine neue Zeile zugefügt wird und für title kein Wert angegeben wurde, erhält diese Zeile "New todo" als Standardwert.‘);

c) SELECT title
FROM todos
WHERE title LIKE “%TodoAPI%“
OR description LIKE “%TodoAPI%“;

Ausgabe:
'Weitere Todos für die TodoAPI! eintragen'
'Die Attribute eines Todo-Objekts für die TodoAPI definieren'
'Die Geschaeftslogik fuer die TodoAPI entwerfen'

2.2

a) PrOgrAMMEntwickLUnGII

```
private String retrieveLetters() {
    String queryWord = "";
    try {
        for(int id : ARRAYINDEXES){
            Letter letter = letterDao.queryForId(id);
            if(letter != null){
                String nextChar = letter.getValue();
                queryWord += nextChar;
            }
        }
    } catch (SQLException exception) {
        System.out.println("Interaction with database failed");
    }
    return queryWord;
}
```

b) IDs für w: 13, 53, 79
IDs für a: 3, 11, 18, 31, 57
IDs für s: 49, 75

```
private List<Integer> getIDsForLetter(String letter){
    List<Integer> ids = new LinkedList<>();
    try {
        List<Letter> letters = letterDao.queryForEq( fieldName: "letter", letter);
        ids = letters.stream().filter(a -> a!=null).map(x -> x.getId()).collect(Collectors.toList());
    } catch (SQLException exception){
        System.out.println("Data base query for Was failed");
    }
    return ids;
}
```

c) Summe = 3766
Durchschnittswert = 46.49 (gerundet auf zwei Nachkommastellen)

```java
private int sumIDs(){
    int result = 0;
    try {
        List<Letter> letters = letterDao.queryForAll();
        result = letters.stream().mapToInt(Letter::getId).sum();
    } catch (SQLException exception){
        System.out.println("Retrieving all entrys failed");
    }
    return result;
}

1 usage
private double avgIDs(){
    double result = 0;
    try {
        List<Letter> letters = letterDao.queryForAll();
        double count = letters.size();
        System.out.println("COUNT: "+count);
        int sum = letters.stream().mapToInt(Letter::getId).sum();
        result = sum/count;
    } catch (SQLException exception){
        System.out.println("Retrieving all entrys failed");
    }
    return result;
}
```

Verbindung mit der Datenbank und Aufruf der einzelnen Methoden:

```java
1 usage
public void run(){
    boolean connected = this.connectToDB( connectionString: "jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1",

    if(connected){
        try {
            this.letterDao = DaoManager.createDao(connectionSource, Letter.class);

            String solutionA = this.retrieveLetters();
            System.out.println("The missing word is: "+solutionA);

            List<Integer> solutionBw = this.getIDsForLetter("w");
            List<Integer> solutionBa = this.getIDsForLetter("a");
            List<Integer> solutionBs = this.getIDsForLetter("s");
            System.out.println("All IDs of entrys containing w: "+solutionBw+", a: "+solutionBa+", s: "+solutionBs);

            int solutionC1 = this.sumIDs();
            System.out.println("Sum of all ids: "+solutionC1);

            double solutionC2 = this.avgIDs();
            System.out.println("Average of all ids: "+solutionC2);

            this.closeConnectionToDB();

        } catch (SQLException exception){
            System.out.println("Creating DAO failed");
        }
    }
}
```

Aufgabe 3

a) https://api.chucknorris.io/jokes/random?category=science

Antwort: {
"categories": [
"science"
],
"created_at": "2020-01-05 13:42:19.576875",
"icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
"id": "izjeqnjzteeqms8l8xgdhw",
"updated_at": "2020-01-05 13:42:19.576875",
"url": "https://api.chucknorris.io/jokes/izjeqnjzteeqms8l8xgdhw",
"value": "Chuck Norris knows the last digit of pi."
}

b) Response Body:

{
"args": {},
"data": {
"key": "pe2ws22",
"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=1-636d2f2b-0eb165d3595f7b753f5f8519",
"content-length": "58",
"content-type": "application/json",
"user-agent": "PostmanRuntime/7.29.2",
"accept": "*/*",
"postman-token": "2fd32388-e59a-442a-9b89-086cffa6bff5",
"accept-encoding": "gzip, deflate, br"
},
"json": {
"key": "pe2ws22",
"purpose": "This is a test."
},
"url": "https://postman-echo.com/post"
}

c) Abrufen eines Spiels mit über seine ID
GET /games/$id

Abrufen aller Spiele mit optionalen Filtern nach Kategorie, Spieltitel oder Altersbeschränkung
GET /games?category={categoryString}&title={titleString}&ageRestrict={boolean}

Erstellen eines neuen Spiels
POST /games

Aktualisierung eines Spiels über seine ID
PUT /games/$id

Löschen eines Spiels über seine ID
DELETE /games/$id