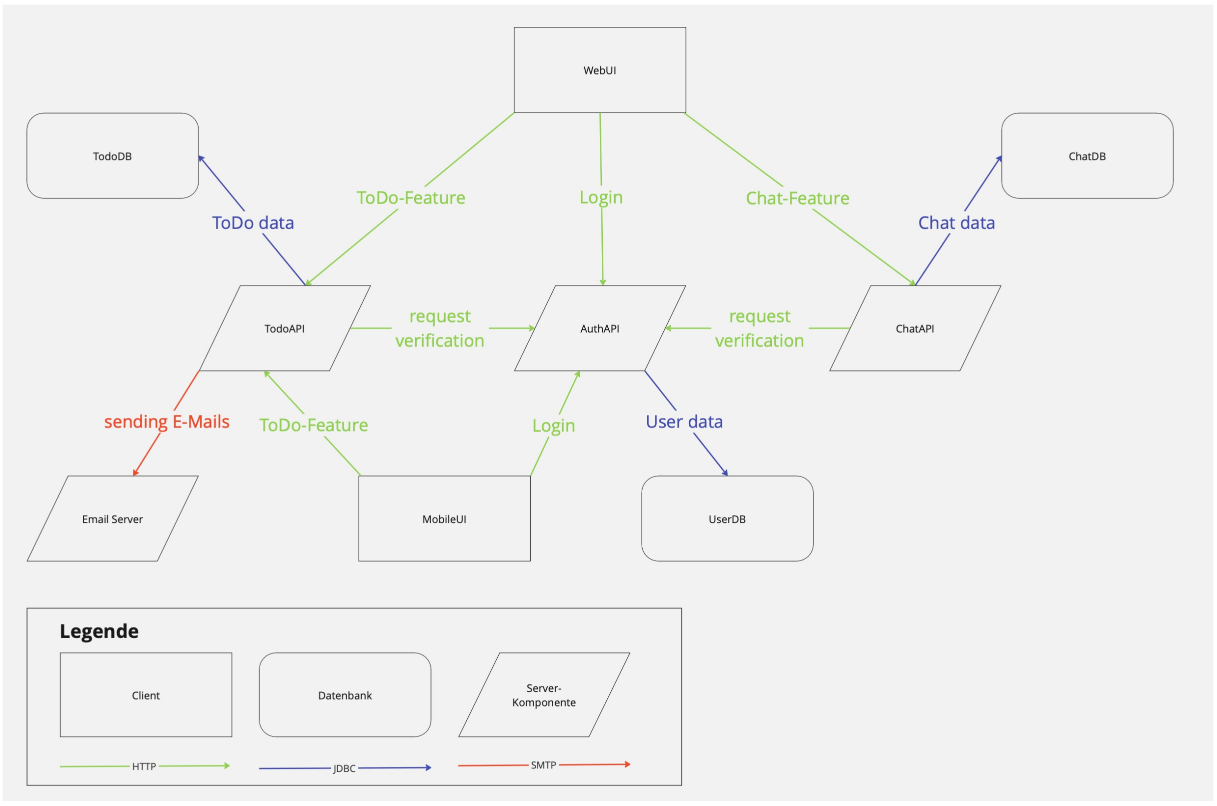
Aufgabe 1


WebUI
TodoDB
ChatDB
ToDo-Feature
Login
Chat-Feature
ToDo data
Chat data
TodoAPI
request verification
AuthAPI
request verification
ChatAPI
sending E-Mails
ToDo-Feature
Login
User data
Email Server
MobileUI
UserDB
Legende
Client
Datenbank
Server-Komponente
HTTP
JDBC
SMTP

Aufgabe 2.1

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

Aufgabe 2.1a)

```
CREATE TABLE IF NOT EXISTS todos (
    id INTEGER PRIMARY KEY ,
    title VARCHAR(255) NOT NULL ,
    description TEXT
);
```

Aufgabe 2.1b)

```
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.');
```

Aufgabe 2.1c)

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

Aufgabe 2.2

Letters-Klasse:

```
package org.example;

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

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

    @DatabaseField
    private String letter;

    public Letters() {
    }

    public Integer getId() {
        return id;
    }

    public String getLetter() {
        return letter;
    }
}
```

Aufgabe 2.2a)

Main-Klasse:

```
package org.example;

import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import java.sql.SQLException;
import java.util.List;

public class Main {

    public static void main(String[] args) throws SQLException {

        ConnectionSource connectionSource = new
JdbcConnectionSource("jdbc:mariadb://bilbao.informatik.uni-
stuttgart.de/pe2-db-a1",
                "pe2-nutzer",
                "esJLtFm6ksCT4mCyOS");

        Dao<Letters, Integer> letterDao =
DaoManager.createDao(connectionSource, 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
        };

        StringBuilder word = new StringBuilder();
        for (int id : arrayIndexes) {
            Letters letter = letterDao.queryForId(id);
            if (letter != null) {
                word.append(letter.getLetter());
            } else {
                System.out.println("Keine Übereinstimmung für ID: "
+ id);
            }
        }

        System.out.println("Das Lösungswort ist: " +
word.toString());
try {
            connectionSource.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

→ Das Lösungswort ist: EntwickLUnGPrOgrAMMII

Aufgabe 2.2b)

```
package org.example;

import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import java.sql.SQLException;
import java.util.List;

public class Main {

    public static void main(String[] args) throws SQLException {

        ConnectionSource connectionSource = new
JdbcConnectionSource("jdbc:mariadb://bilbao.informatik.uni-
stuttgart.de/pe2-db-a1",
                "pe2-nutzer",
                "esJLtFm6ksCT4mCyOS");

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

        List<Letters> lettersV = letterDao.queryForEq("letter",
"V");
        System.out.println("IDs für 'V': ");
        for (Letters letter : lettersV) {
            System.out.println(letter.getId());
        }

        List<Letters> lettersB = letterDao.queryForEq("letter",
"b");
        System.out.println("IDs für 'b': ");
        for (Letters letter : lettersB) {
            System.out.println(letter.getId());
        }

        List<Letters> lettersT = letterDao.queryForEq("letter",
"t");
        System.out.println("IDs für 't': ");
        for (Letters letter : lettersT) {
            System.out.println(letter.getId());
        }

        try {
            connectionSource.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

Aufgabe 2.2c)

```
package org.example;

import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import java.sql.SQLException;
import java.util.List;

public class Main {

   public static void main(String[] args) throws SQLException {

      ConnectionSource connectionSource = new
JdbcConnectionSource("jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1",
            "pe2-nutzer",
            "esJLtFm6ksCT4mCyOS");

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

List<Letters> allLetters = letterDao.queryForAll();

int sumIDs = 0;
int countIDs = allLetters.size();

for (Letters letter : allLetters) {
   sumIDs += letter.getId();
}

double averageValue = (countIDs > 0) ? (double) sumIDs / countIDs : 0;

System.out.println("Summe der IDs: " + sumIDs);
System.out.println("Durchschnittswert der IDs: " + averageValue);

try {
        connectionSource.close();
} catch (Exception e) {
        e.printStackTrace();
 }
 }
}
```

→ Summe der IDs: 4167
→ Durchschnittswert der IDs: 50.81707317073171

Aufgabe 3a)

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

Ergebnis → Beispiel:

```
{
      "categories": [
      "history"
      ],
      "created_at": "2020-01-05 13:42:19.576875",
      "icon_url": "https://api.chucknorris.io/img/avatar/chuck-
      norris.png",
      "id": "rqcvwdgqq6amwony3nngba",
      "updated_at": "2020-01-05 13:42:19.576875",
      "url":
      "https://api.chucknorris.io/jokes/rqcvwdgqq6amwony3nngba",
      "value": "In the Words of Julius Caesar, \"Veni, Vidi, Vici,
      Chuck Norris\". Translation: I came, I saw, and I was
      roundhouse-kicked inthe face by Chuck Norris."
}
```

Aufgabe 3b)

Man erhält folgenden Response-Body:

```
{
    "args": {},
    "data": {
        "key": "pe2ws23",
        "purpose": "This is a test."
    },
    "files": {},
    "form": {},
    "headers": {
        "host": "postman-echo.com",
        "x-request-start": "t=1730200611.930",
        "connection": "close",
        "content-length": "58",
        "x-forwarded-proto": "https",
        "x-forwarded-port": "443",
        "x-amzn-trace-id": "Root=1-6720c423-404a259d0f670d0071f9d783",
        "content-type": "application/json",
        "user-agent": "PostmanRuntime/7.42.0",
        "accept": "*/*",
        "postman-token": "a1e4d791-4df5-4863-874e-91ac9efeb2c1",
        "accept-encoding": "gzip, deflate, br"
    },
    "json": {
        "key": "pe2ws23",
        "purpose": "This is a test."
  },
  "url": "https://postman-echo.com/post"
}
```

Aufgabe 3c)

Eine DVD erstellen
→ eine neue DVD wird im System erstellt:
- POST /dvds

Eine DVD aktualisieren
→ einzelne DVD wird anhand von ID vollständig aktualisiert:
- PUT /dvds/{id}

Alle DVDs anzeigen
→ es werden alle DVDs, die im System registriert sind, ausgegeben (dabei kann man nach Kategorie, Titel und/oder Altersbeschränkung filtern):
- GET /dvds?category={category}&title={title}&ageRestricted={boolean}

Einzelne DVD anzeigen
→ einzelne DVD wird anhand von ID abgerufen:
- GET /dvds/{id}

Eine DVD löschen
→ einzelne DVD wird anhand von ID gelöscht:
- DELETE /dvds/{id}