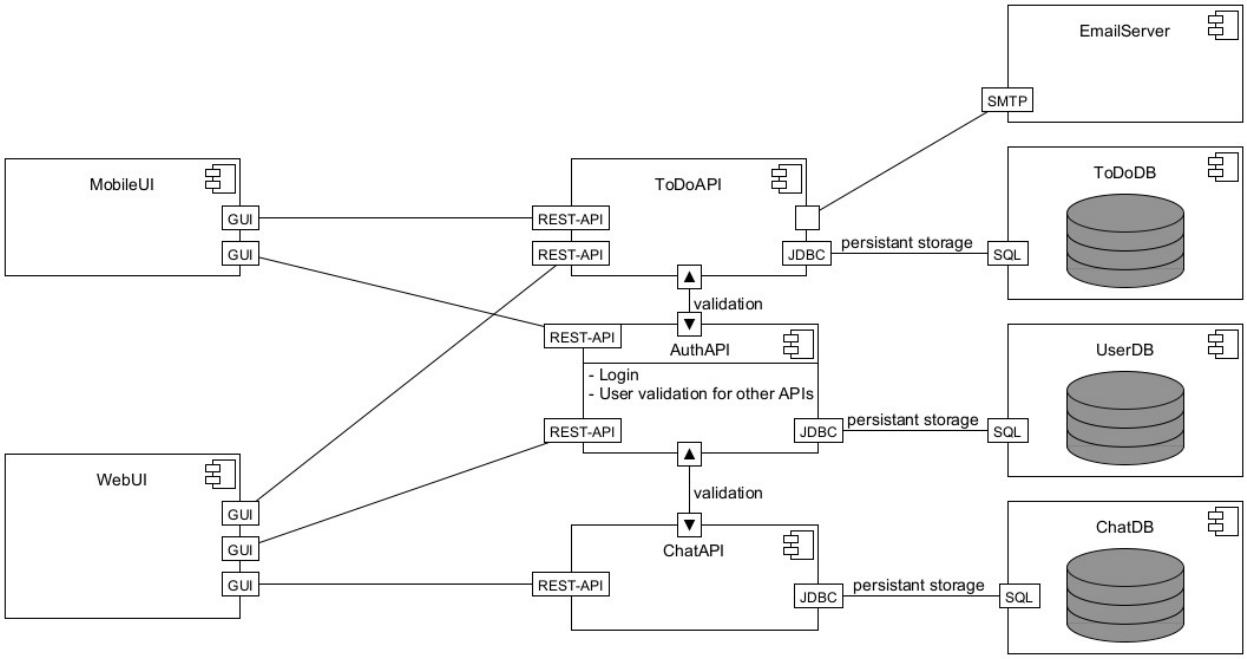
Nr 1)


EmailServer
SMTP
MobileUI
GUI
GUI
ToDoAPI
REST-API
REST-API
JDBC
persistant storage
SQL
ToDoDB
validation
REST-API
AuthAPI
- Login
- User validation for other APIs
REST-API
JDBC
persistant storage
SQL
UserDB
WebUI
GUI
GUI
GUI
validation
ChatAPI
REST-API
JDBC
persistant storage
SQL
ChatDB

Nr 2.1

a)

```
CREATE TABLE IF NOT EXISTS todos (id int PRIMARY KEY, title varchar(100) NOT
NULL DEFAULT 'New todo', description varchar(500), CONSTRAINT CHECK (title !=
''))COLLATE='utf8mb4_general_ci';
```

Laut Tabelleneintrag muss „`CONSTRAINT CHECK (title != '')`“ vorhanden sein, jedoch ist die Datenbank so nicht konfiguriert (Erkennbar daran, das in MariaDB keine Prüfbedingung angezeigt werden). Je nachdem, was benötigt ist kann dieser Teil weggelassen werden.

b)

```
INSERT INTO todos (id) VALUES (1);
UPDATE todos SET title = 'Einen SQL-Befehl zur Erstellung einer Datenbanktabelle
entwerfen' WHERE id=1;
UPDATE todos SET description = '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 einen Standardwert.'
WHERE id=1;
```

c)

Folgender Befehl wurde verwended:

```
SELECT title FROM todos WHERE title LIKE '%ToDoAPI%';
```

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

2.2)

a)

Lösung : `PrOgrAMMEntwickLUnGII`

Code:

```
private void retrieveTableContent() {
        try {
            // get entries that match id
            List<Letters> searchedLetters =new ArrayList<Letters>();
            for (int number : arrayIndexes) {
             //System.out.println(number);
              searchedLetters.add(lettersDao.queryForId(number));
            }

            for (Letters letters : searchedLetters) {
                System.out.print(letters.getLetter());
            }

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

```
    }
```

b)

Folgender Code wurde verwendet:

```
String[] was = {"W","a","s"};
            for (String letter : was) {
             List<Letters> searchedLetters = lettersDao.queryForEq("letter", letter);
                for (Letters letterr : searchedLetters) {
                    Integer letterID = letterr.getId();
                    LOGGER.log(Level.INFO, "Letter ID: " + letterID);
                }
            }
```

```
Folgende Ausgabe wurde erzeugt (vereinfacht):
StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 'W'' with 0 args
returned 3 results
INFO: Letter ID: 13
INFO: Letter ID: 53
INFO: Letter ID: 79
StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 'a'' with 0 args
returned 5 results
Letter ID: 3
Letter ID: 11
Letter ID: 18
Letter ID: 31
Letter ID: 57
StatementExecutor query of 'SELECT * FROM `letters` WHERE `letter` = 's'' with 0 args
returned 2 results
Letter ID: 49
Letter ID: 75
```

c)

Verwendete Code:

```
             int sum=0;
             double avg =0.0;
            List<Letters> searchedLettersByID =lettersDao.queryForAll();
            for (Letters letter : searchedLettersByID) {
             sum=sum+letter.getId();
            }
            avg=sum/searchedLettersByID.size();
            LOGGER.log(Level.INFO, "Summe aller IDs " + sum);
            LOGGER.log(Level.INFO, "Durchschnitt aller IDs " + avg);
```

Ausgabe:

```
Summe aller IDs 3766
Durchschnitt aller IDs 46.0
```

Nr 3)

a)


```
{
  "categories": [
    "science"
  ],
  "created_at": "2020-01-05 13:42:18.823766",
  "icon_url": "https://assets.chucknorris.host/img/avatar/chuck-norris.png",
  "id": "sbdpxom2tm2-h_wdxncvta",
  "updated_at": "2020-01-05 13:42:18.823766",
  "url": "https://api.chucknorris.io/jokes/sbdpxom2tm2-h_wdxncvta",
  "value": "Chuck Norris eats lightning and shits out thunder."
}
```


b)


```
{
  "args": {},
  "data": {
    "key": "pe2ws21",
    "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=1-61842eb8-2119ca821a6fd2bf60b07bae",
    "content-length": "64",
    "content-type": "application/json",
    "user-agent": "PostmanRuntime/7.28.4",
    "accept": "*/*",
    "postman-token": "33920e07-580a-4ffe-89bd-6c16341fc193",
    "accept-encoding": "gzip, deflate, br"
  },
  "json": {
    "key": "pe2ws21",
    "purpose": "This is a test."
  },
  "url": "https://postman-echo.com/post"
}
```

c)

POST	/api/games	Erstellt neues Spiel, oder verändert Inforamtion, falls Spiel schon existiert
GET	/api/games	Liefert alle Spiele
GET	/api/games/{id}	Liefer Spiel mit ID
PUT	/api/games	Erstellt neues Spiel, oder verändert Inforamtion, falls Spiel schon existiert
DELETE	/api/games/{id}	Löscht Spiel mit ID