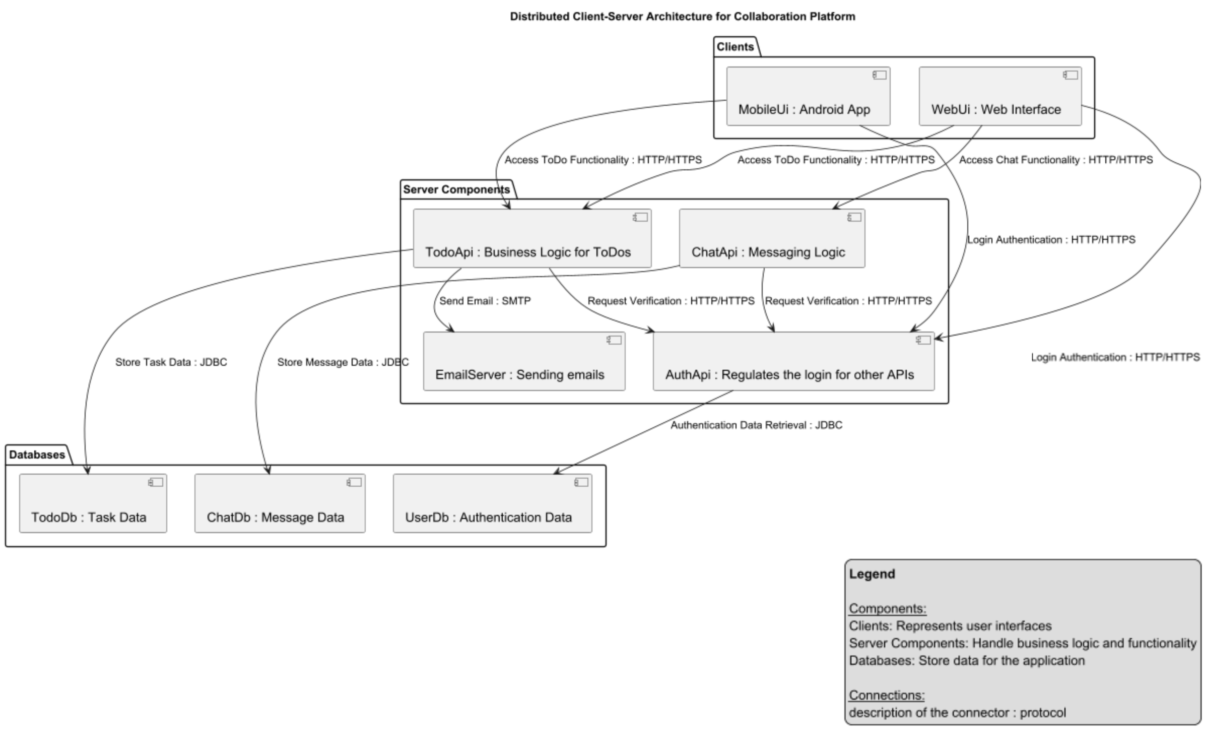
PE2 Aufgabenblatt 01

Von Lennart Eisenmann,

Matrikelnummer: 3686049

Aufgabe 1:


Distributed Client-Server Architecture for Collaboration Platform
Clients
MobileUi : Android App
WebUi : Web Interface
Access ToDo Functionality : HTTP/HTTPS
Access ToDo Functionality : HTTP/HTTPS
Access Chat Functionality : HTTP/HTTPS
Server Components
TodoApi : Business Logic for ToDos
ChatApi : Messaging Logic
Login Authentication : HTTP/HTTPS
Send Email : SMTP
Request Verification : HTTP/HTTPS
Request Verification : HTTP/HTTPS
EmailServer : Sending emails
AuthApi : Regulates the login for other APIs
Login Authentication : HTTP/HTTPS
Store Task Data : JDBC
Store Message Data : JDBC
Authentication Data Retrieval : JDBC
Databases
TodoDb : Task Data
ChatDb : Message Data
UserDb : Authentication Data
Legend
Components:
Clients: Represents user interfaces
Server Components: Handle business logic and functionality
Databases: Store data for the application
Connections:
description of the connector : protocol

Aufgabe 2:

2.1

a) **SQL-Befehl, der die bereits existierende Tabelle todos erstellen würde:**

```sql
USE `pe2-db-a1`;
CREATE TABLE IF NOT EXISTS todos (
    id INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description VARCHAR(500)
);

INSERT IGNORE 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', NULL);
```

b) **SQL-Befehl , der verwendet wurde, um die Zeile mit id = 1 zur Tabelle todos hinzuzufügen:**

```sql
INSERT IGNORE 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-Befehl:**

```sql
USE `pe2-db-a1`;
SELECT description
FROM todos
WHERE description LIKE '%Weihnacht%';
```

2.2

Letter.java Link:

https://github.tik.uni-stuttgart.de/iste-pe2-2024/repo051/blob/main/docs/JavaClasses_exc1/Letter.java

a)

The Solution is:

EntwickLUnGPrOgrAMMII

WordOfArrayIndex.java Link:

https://github.tik.uni-stuttgart.de/iste-pe2-2024/repo051/blob/main/docs/JavaClasses_exc1/WordOfArrayIndex.java

b)

The Solution is:

Ids für V: 52, 78, 52, 78

Ids für b: 9, 32, 58, 9, 32, 58,

Ids für t: 50, 76, 50, 76

IdsForLettersVTB.java Link:

https://github.tik.uni-stuttgart.de/iste-pe2-2024/repo051/blob/main/docs/JavaClasses_exc1/IdsForLettersVTB.java

c)

The Solution is:

Sum of all IDs: 4167

Mean of all IDs: 50

SumAndMeanOfIds.java Link:

https://github.tik.uni-stuttgart.de/iste-pe2-2024/repo051/blob/main/docs/JavaClasses_exc1/SumAndMeanOfIds.java

Aufgabe 3: HTTP und Rest

a) GET-Request senden

Request:

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

Erhaltener Response Body:

```
{
  "categories": [
    "history"
  ],
  "created_at": "2020-01-05 13:42:19.576875",
  "icon_url": "https://api.chucknorris.io/img/avatar/chuck-norris.png",
  "id": "umwzezi7siitlteg3jiyiw",
  "updated_at": "2020-01-05 13:42:19.576875",
  "url": "https://api.chucknorris.io/jokes/umwzezi7siitlteg3jiyiw",
  "value": "Nagasaki never had a bomb dropped on it. Chuck Norris jumped out of a plane and punched
the ground"
}
```

b) POST-Request senden

Response:

```
{
  "args": {},
  "data": {
    "key": "pe2ws23",
    "purpose": "This is a test."
  },
  "files": {},
  "form": {},
  "headers": {
    "host": "postman-echo.com",
    "x-request-start": "t=1730737634.277",
    "connection": "close",
    "content-length": "57",
    "x-forwarded-proto": "https",
    "x-forwarded-port": "443",
    "x-amzn-trace-id": "Root=1-6728f5e2-5a8b91fe66db666467c78a24",
    "content-type": "application/json",
    "user-agent": "PostmanRuntime/7.42.0",
    "accept": "*/*",
    "postman-token": "2d2d5983-a576-4a49-ac7d-aaf57db5809d",
    "accept-encoding": "gzip, deflate, br",
    "cookie": "sails.sid=s%3A6c2T_qKcrEpSkqNGzs_1y-
zE7iVgRyO6.I4mZT02QPjfDWVPt8tBkK%2FDW%2FNgqYBuOi%2F1WjQfS%2F%2Fc"
  },
  "json": {
    "key": "pe2ws23",
    "purpose": "This is a test."
  },
  "url": "https://postman-echo.com/post"
}
```

c) REST API Design

<u>Retrieve all dvds with filter by category, substring
of the title and if its for adults or not:</u>

GET /dvds?category={categoryString}&subTitle={subTitleString}&forAdults={boolean}

<u>Create a single dvd:</u>

POST /dvds

<u>Retrieve a single dvd by its id:</u>

GET /dvds/$id

<u>Update a single dvd by its id:</u>

PUT /dvds/$id

<u>Delete a single dvd by its id and if wanted save a copy in the archive:</u>

DELETE /dvds/$id?keepInArchive={boolean}