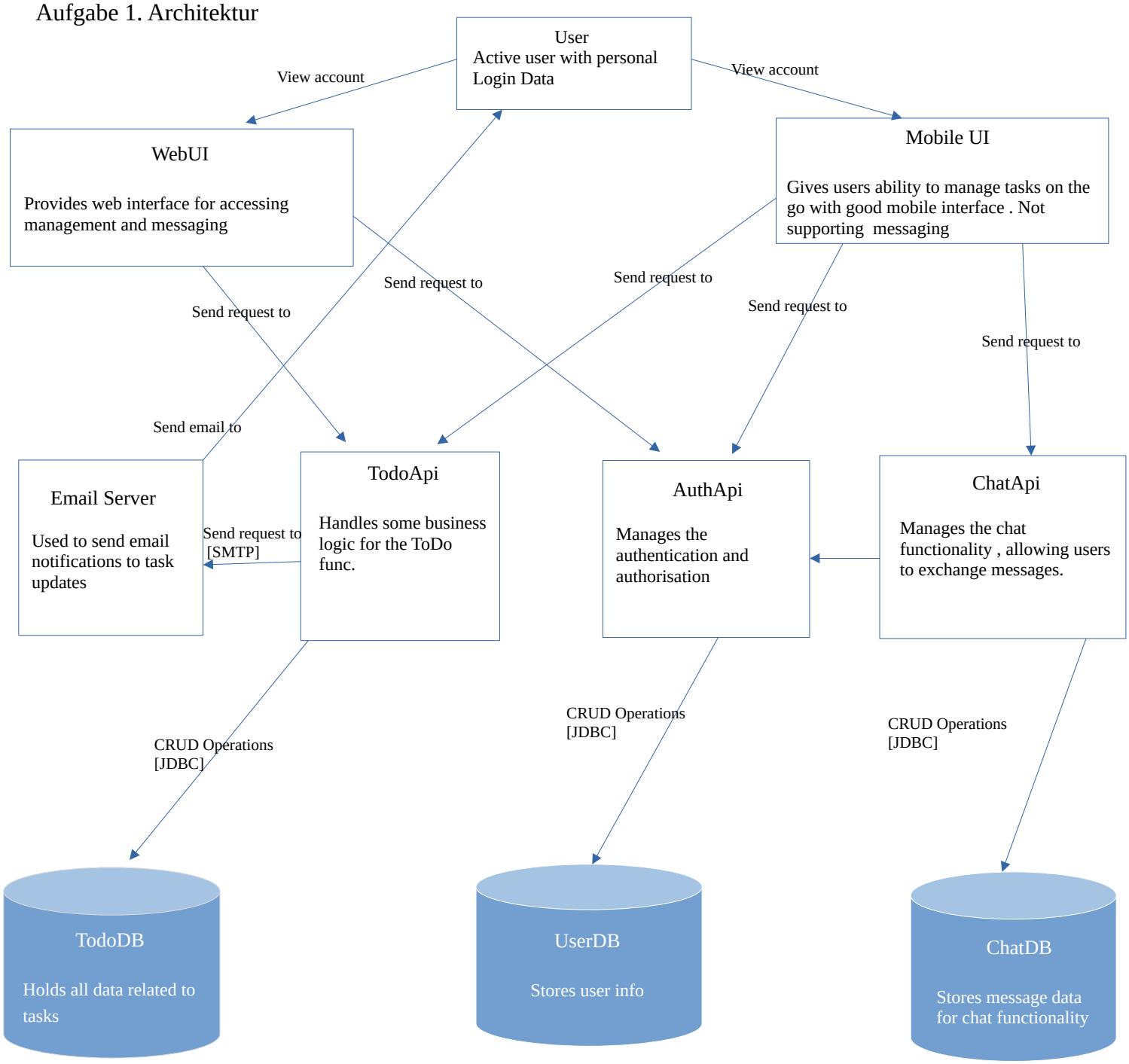
Aufgabenblatt 01 Hristo Baldzhiyski Email: st187553@stud.uni-stuttgart.de

Aufgabe 1. Architektur



Aufgabe 2. SQL and JDBC

2.1

a)
```
CREATE TABLE IF NOT EXISTS todos (
  id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  title VARCHAR(100) NOT NULL,
  description VARCHAR(500)
);
```

b)
```
INSERT INTO todos (id, title, description)
VALUES (1, 'Some random title', 'Some random description');
```

c) *SQL- Befehl* :
```
SELECT description
FROM todos
WHERE description LIKE '%Weihnacht%';
```

Ergebnisse :

Es ist nun endlich so weit! Mit dem 01. November wird es Zeit, zügig die Weihnachtsdekorationen auszupacken.

Bald sollte ich Weihnachtsplaetzchen backen.

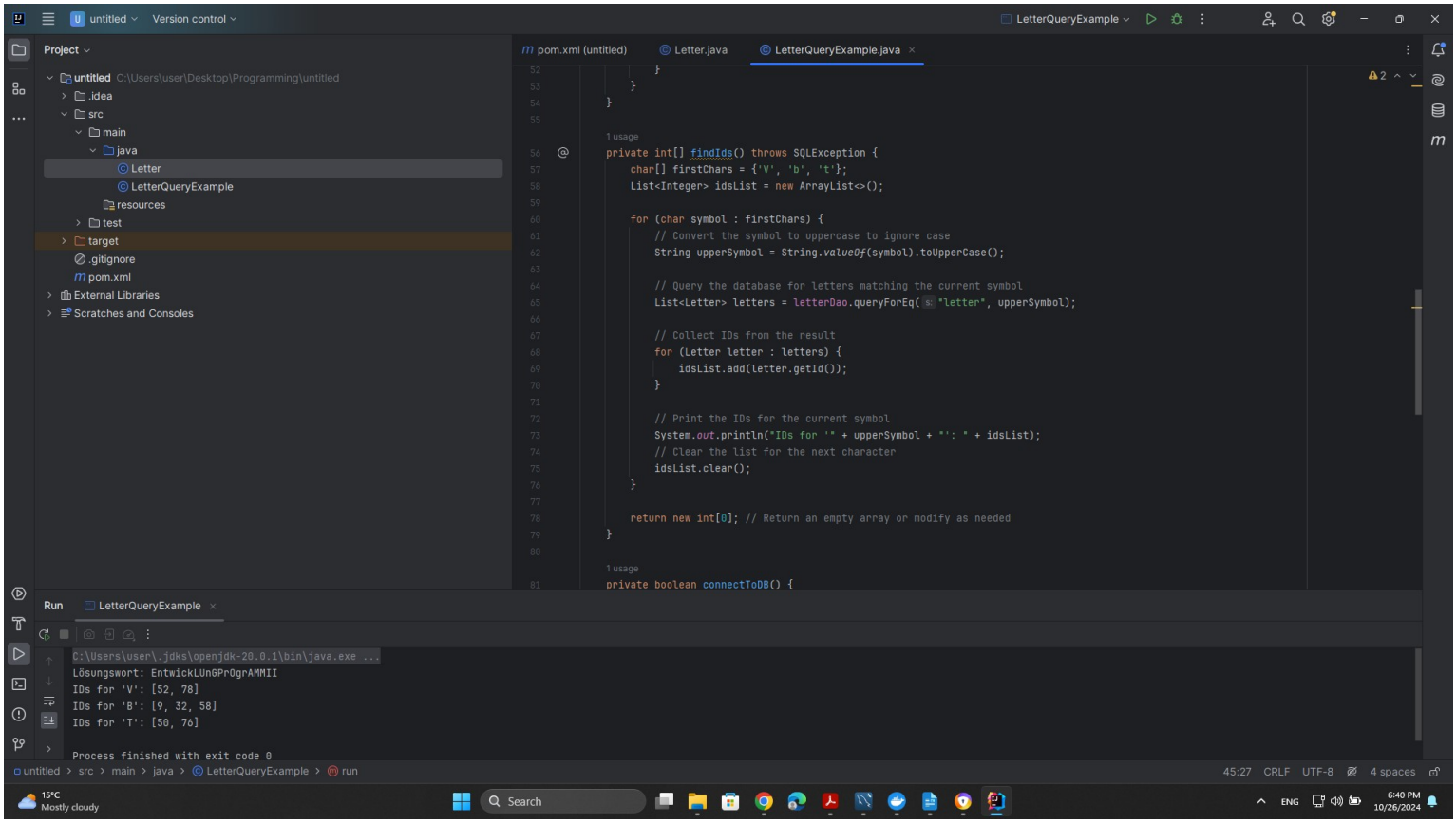
2.2

a) Lösungswort: EntwickLUnGPrOgrAMMII

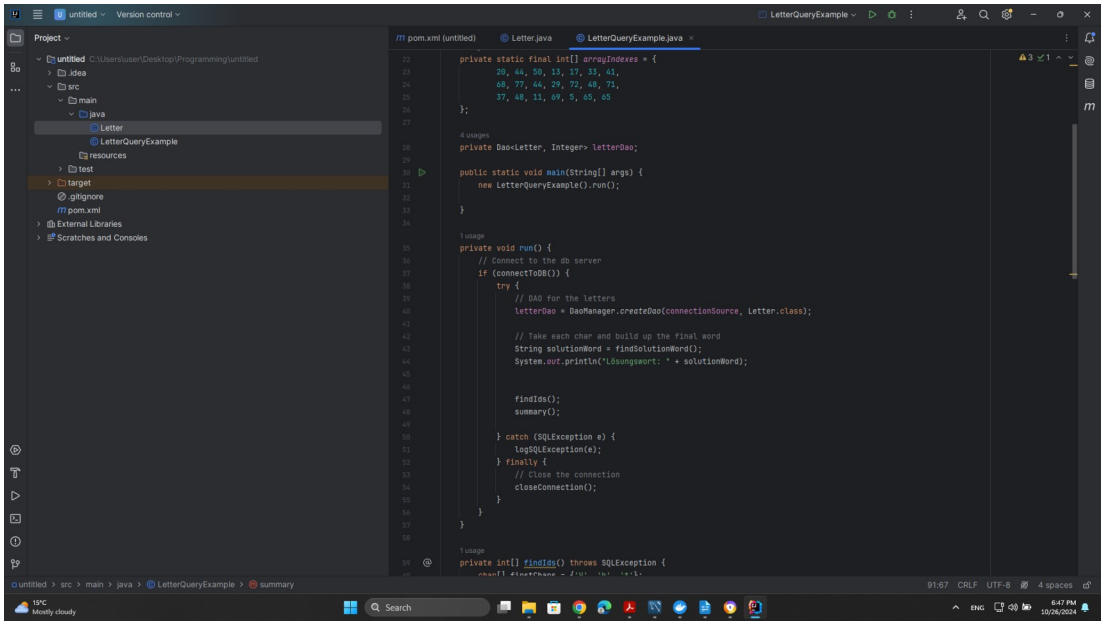
File - C:\Users\user\Desktop\Programming\untitled\src\main\java\LetterQueryExample.java

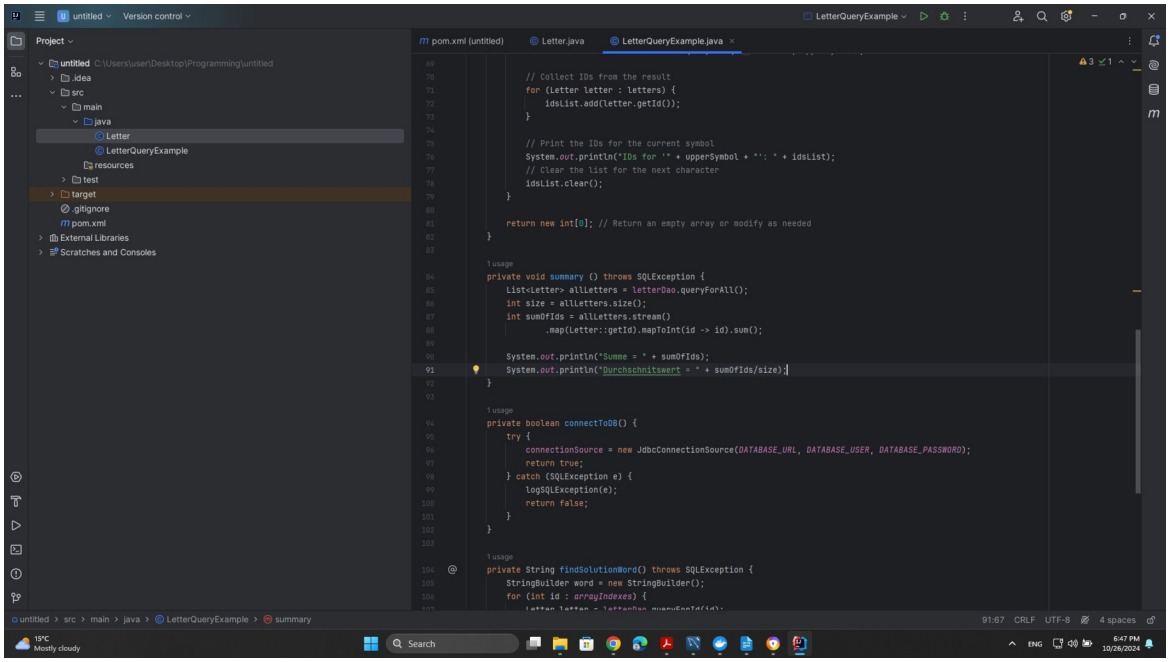
```java
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.support.ConnectionSource;
import com.j256.ormlite.table.TableUtils;


public class LetterQueryExample {
    private static final Logger LOGGER = Logger.getLogger(LetterQueryExample.class.getName());
    private static final String DATABASE_URL = "jdbc:mariadb://bilbao.informatik.uni-stuttgart.de/pe2-db-a1";
    private static final String DATABASE_USER = "pe2-nutzer";
    private static final String DATABASE_PASSWORD = "esJLtFm6ksCT4mCyOS";

    private ConnectionSource connectionSource;
    private static final int[] arrayIndexes = {
            20, 44, 50, 13, 17, 33, 41,
            68, 77, 44, 29, 72, 48, 71,
            37, 48, 11, 69, 5, 65, 65
    };

    private Dao<Letter, Integer> letterDao;

    public static void main(String[] args) {
        new LetterQueryExample().run();

    }

    private void run() {
        // Connect to the db server
        if (connectToDB()) {
            try {
                // DAO for the letters
                letterDao = DaoManager.createDao(connectionSource, Letter.class);

                // Take each char and build up the final word
                String solutionWord = findSolutionWord();
                System.out.println("Lösungswort: " + solutionWord);

            } catch (SQLException e) {
                logSQLException(e);
            } finally {
                // Close the connection
                closeConnection();
            }
        }
    }

    private boolean connectToDB() {
        try {
            connectionSource = new JdbcConnectionSource(DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD);
            return true;
        } catch (SQLException e) {
            logSQLException(e);
            return false;
        }
    }

    private String findSolutionWord() throws SQLException {
        StringBuilder word = new StringBuilder();
        for (int id : arrayIndexes) {
            Letter letter = letterDao.queryForId(id);
            if (letter != null) {
                word.append(letter.getLetter());
            }
        }
        return word.toString();
    }

    private void closeConnection() {
        try {
            if (connectionSource != null) {
                connectionSource.close();
            }
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Fehler beim Schließen der Verbindung: " + e.getMessage());
        }
    }

    private void logSQLException(SQLException e) {
        LOGGER.log(Level.SEVERE, "SQL Fehlercode: " + e.getErrorCode());
        LOGGER.log(Level.SEVERE, "SQL Fehlermeldung: " + e.getMessage());
    }
}
```

b)
IDs for 'V': [52, 78]
IDs for 'B': [9, 32, 58]
IDs for 'T': [50, 76]



c)

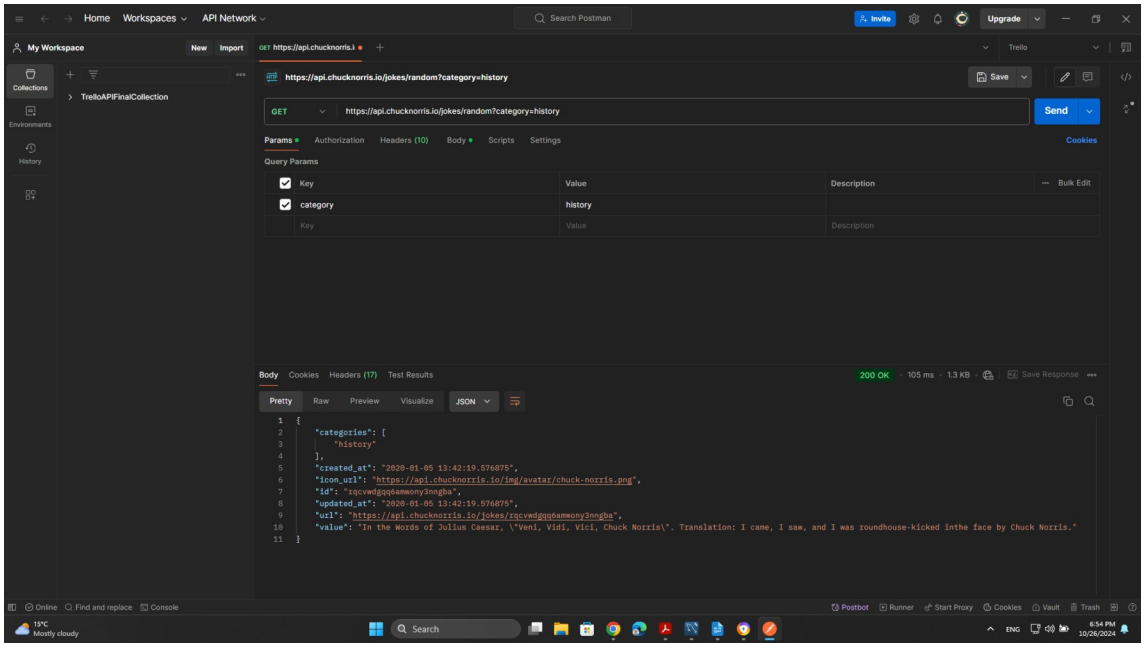




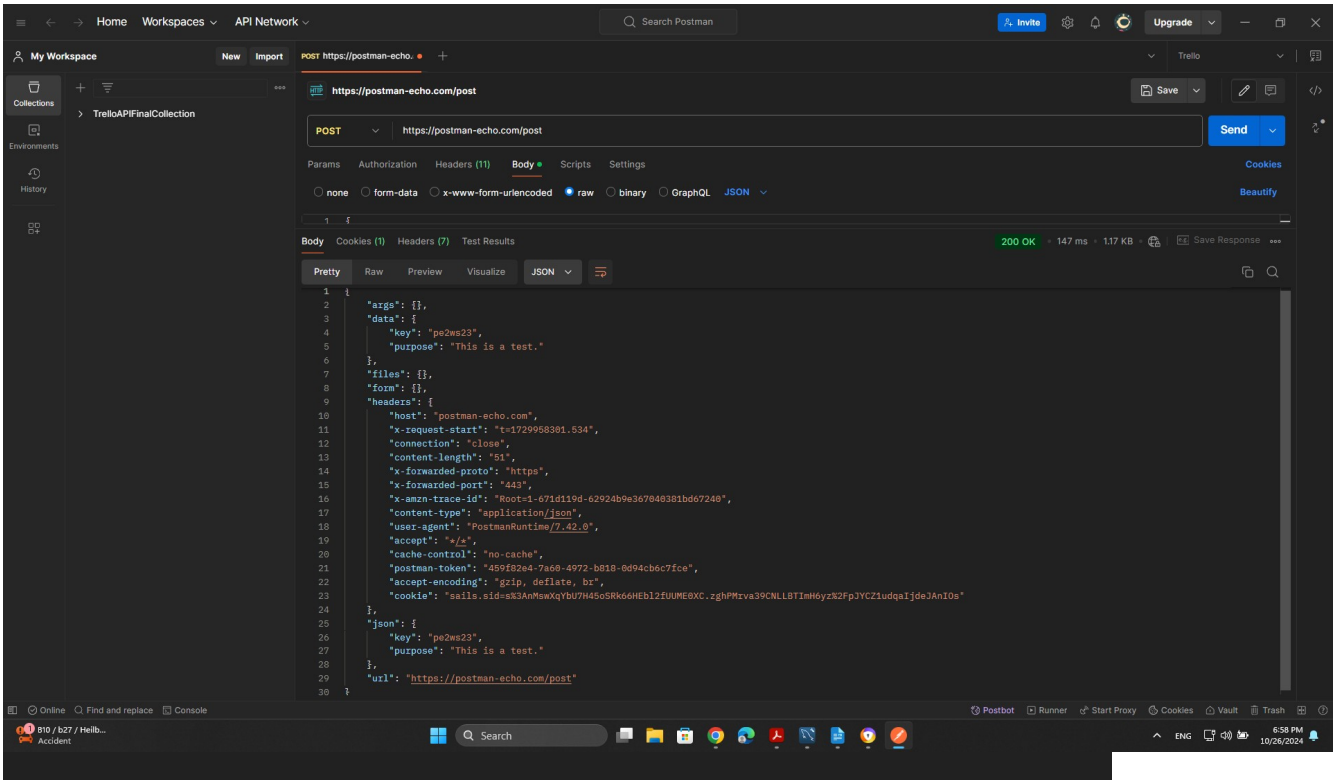
Output : **Summe** = 4167 , **Durchschnitswert** = 50

Aufgabe 3. HTTP und REST

a) Wenn wir den GET-Request erfolgreich ausgeführt haben, erhalten wir ein JSON-Objekt mit verschiedenen Feldern, einschließlich des Witzes selbst.



b) Wir erhalten folgendes body :



c)

1. Abruf aller DVDs
   - HTTP- Method : GET
   - Pfad : /dvds/all
   - Beschreibung : Diese Ressource ermöglicht das Abrufen aller verfügbaren DVDs. Optional können Filterparameter angesetzt werden (Kategorie-String , Titel, Restriciton). Dann sehen z.B die Queries so aus : *GET /dvds/all?category=action* , *GET /dvds/all?title=Adventure* , *GET /dvds?isAgeRestricted=true*

2. Abrufen einer DVD nach ID
   - HTTP- Method : GET
   - Pfad : /dvds/{id}
   - Beschreibung: Diese Ressource ermöglicht das Abrufen einer spezifischen DVD anhand ihrer ID.
3. Erstellen einer neuen DVD via JSON body
   - HTTP-Method : POST
   - Pfad : /dvds/post
   - Beschreibung : Diese Ressource ermöglicht das Erstellen einer neuen DVD. Der Request-Body enthält die Details der DVD (Titel, Kategorie, Altersfreigabe).

4. Löschen einer DVD
   - HTTP- Method : DELETE
   - Pfad : /dvds/{id}
   - Beschreibung : das Löschen einer DVD anhand ihrer ID . Man kann auch ein Query-Parameter *keepInArchive=true* hinzugefügt werden, um eine Archivkopie der DVD zu behalten.

5. Aktualisieren einer DVD
   - HTTP-Method : PUT
   - Pfad: /dvds/id
   - Beschreibung: Dadurch kann man die Info des existierenden Ressource aktualisieren anhang der ID.
   - Der Request-Body enthält die aktualisierten DVD-Daten.