# LU06.L07 - Multiuser Todo-Liste

## TodoItem

`todoItem` angepasst um die `user_id` zu speichern.

todoItem.py

```python
from dataclasses import dataclass

@dataclass
class TodoItem:
    item_id: int
    user_id: int
    title: str
    is_completed: bool
```

## TodoDao

`todoDao` angepasst um die `user_id` zu berücksichtigen.

todoDao.py

```python
import sqlite3
from todoItem import TodoItem

class TodoDao:
    def __init__(self, db_file):
        self.conn = sqlite3.connect(db_file, check_same_thread=False)
        self.cursor = self.conn.cursor()

    def create_table(self):
        self.cursor.execute('''DROP TABLE IF EXISTS todo_items''')
        self.cursor.execute(
            '''CREATE TABLE IF NOT EXISTS todo_items (
                item_id INTEGER PRIMARY KEY,
                title TEXT,
                is_completed BOOLEAN,
                user_id INTEGER,
                FOREIGN KEY(user_id) REFERENCES users(id))'''
        )
        self.conn.commit()

    def add_item(self, todo_item):
        self.cursor.execute("INSERT INTO todo_items (user_id, title,
```

```
is_completed) VALUES (?, ?, ?)",
                              (todo_item.user_id, todo_item.title,
todo_item.is_completed))
        self.conn.commit()

    def get_item(self, item_id, user_id):
        self.cursor.execute("SELECT * FROM todo_items WHERE item_id = ?
AND user_id = ?", (item_id, user_id))
        row = self.cursor.fetchone()
        if row:
            return TodoItem(row[0], row[3], row[1], row[2])
        return None

    def get_all_items(self, user_id):
        self.cursor.execute("SELECT * FROM todo_items WHERE user_id =
?", (user_id,))
        rows = self.cursor.fetchall()
        todo_items = [TodoItem(row[0], row[3], row[1], row[2]) for row
in rows]
        return todo_items

    def update_item(self, todo_item):
        self.cursor.execute("UPDATE todo_items SET title = ?,
is_completed = ? WHERE item_id = ?",
                              (todo_item.title, todo_item.is_completed,
todo_item.item_id))
        if self.cursor.rowcount > 0:
            self.conn.commit()
            return True
        return False

    def delete_item(self, item_id, user_id):
        self.cursor.execute("DELETE FROM todo_items WHERE item_id = ?
AND user_id = ?", (item_id, user_id))
        if self.cursor.rowcount > 0:
            self.conn.commit()
            return True
        return False

    def close(self):
        self.conn.close()
```

## todoBlueprint

`todoBlueprint` angepasst um jeweils die `user_id` zu übergeben und alle funktionen mit `@login_required` zu ergänzen.

todoBlueprint.py

```
from flask import Blueprint, jsonify, request
from flask_login import login_required, current_user
from todoDao import TodoDao
from todoItem import TodoItem

todo_blueprint = Blueprint('todo_blueprint', __name__)
todo_dao = TodoDao('todo_example.db')

@todo_blueprint.route('/todos', methods=['GET'])
@login_required
def get_all_todos():
    items = todo_dao.get_all_items(current_user.id)
    return jsonify([item.__dict__ for item in items]), 200

@todo_blueprint.route('/todos/<int:item_id>', methods=['GET'])
@login_required
def get_todo(item_id):
    item = todo_dao.get_item(item_id, current_user.id)
    if item:
        return jsonify(item.__dict__), 200
    else:
        return jsonify({"message": "Item not found"}), 404

@todo_blueprint.route('/todos', methods=['POST'])
@login_required
def add_todo():
    data = request.get_json()
    new_item = TodoItem(None, current_user.id, data['title'],
data['is_completed'])
    todo_dao.add_item(new_item)
    return jsonify({"message": "Todo item created"}), 201

@todo_blueprint.route('/todos/<int:item_id>', methods=['PUT'])
@login_required
def update_todo(item_id):
    data = request.get_json()
    updated_item = TodoItem(item_id, data['title'],
data['is_completed'], current_user.id)
    if todo_dao.update_item(updated_item):
        return jsonify({"message": "Item updated"}), 200
    else:
        return jsonify({"message": "Item not found or not updated"}),
404

@todo_blueprint.route('/todos/<int:item_id>', methods=['DELETE'])
@login_required
def delete_todo(item_id):
    if todo_dao.delete_item(item_id, current_user.id):
        return jsonify({"message": "Item deleted"}), 200
```

```
    else:
        return jsonify({"message": "Item not found or not deleted"}),
404
```

# main

`main` angepasst die generierung der demo-daten korrekt zu machen.

main.py

```
from flask import Flask
from flask_login import LoginManager

from todoBlueprint import todo_blueprint
from userBlueprint import user_blueprint

app = Flask(__name__)
app.secret_key = 'supersecretkey'

login_manager = LoginManager()
login_manager.init_app(app)

@login_manager.user_loader
def load_user(user_id):
    from userDao import UserDao
    user_dao = UserDao('todo_example.db')
    return user_dao.get_user_by_id(int(user_id))


app.register_blueprint(todo_blueprint)
app.register_blueprint(user_blueprint)


def generate_testdata():

    from todoItem import TodoItem
    from user import User
    from todoDao import TodoDao
    from userDao import UserDao

    todo_dao = TodoDao('todo_example.db')
    user_dao = UserDao('todo_example.db')

    # Generate user
    user_dao.create_user_table()
    user_dao.add_user(User(1, 'admin', 'admin@example', 'admin'))
    user_dao.add_user(User(2, 'user', 'user@example', 'user'))
```

```
    # Generate todo items
    todo_dao.create_table()
    todo_dao.add_item(TodoItem(1,1, 'Buy milk', False))
    todo_dao.add_item(TodoItem(2,1, 'Buy eggs', False))
    todo_dao.add_item(TodoItem(3,2, 'Buy bread', False))
    todo_dao.add_item(TodoItem(4,2, 'Buy butter', False))

    todo_dao.close()
    user_dao.close()


if __name__ == '__main__':
    generate_testdata()
    app.run(debug=True)
```