Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине: “РСиОТ”

По теме: «Написание базового веб-приложения»

**Выполнил**:

студент 4 курса

группы ПО-7

Комиссаров А.Е.

**Проверил:**

Козик И.Д.

Брест 2024

**Цель.** Разработать базовое веб-приложение, предоставляющее API для взаимодействия с базой данных. Программа должна обрабатывать HTTP-запросы и выполнять операции CRUD (создание, чтение, обновление, удаление) над данными в базе данных.

**Ход работы:**

API был создан на базе библиотеки fastapi для python. База данных выбрана MySQL. Вариант задания – баг-трекер.

Код программы:

from fastapi import FastAPI, Form, HTTPException

from fastapi.responses import RedirectResponse

import mysql.connector

print("creating FastAPI app")

app = FastAPI(

title="bug\_tracker",

version="lab1",

description="server\_description"

)

@app.get("/")

async def redirect\_root\_to\_docs():

return RedirectResponse("/docs")

#=====================================

global db

db = mysql.connector.connect(

host="localhost",

user="root",

database="rsiot\_lab"

)

print("connected to MySQL at "+str(db.server\_host)+":"+str(db.server\_port))

@app.get("/bugs/list/all")

async def bugsGetAll():

cursor = db.cursor()

SQL = "SELECT \* FROM `bugs`"

cursor.execute(SQL)

data = cursor.fetchall()

result = str(data)

return {"success":result}

@app.get("/bugs/list/app")

async def bugsGet(appid):

cursor = db.cursor()

SQL = "SELECT \* FROM `bugs` WHERE app="+appid

cursor.execute(SQL)

data = cursor.fetchall()

result = str(data)

return {"success":result}

@app.post("/bugs/add")

async def bugsAdd(appid: int = Form(...), authorid: int = Form(...), title: str = Form(...), status: str = Form(...)):

cursor = db.cursor()

cursor.execute("SELECT \* FROM `users`")

users = cursor.fetchall()

cursor.execute("SELECT id FROM `apps`")

apps = cursor.fetchall()

cursor.execute("SELECT app, author, title, status FROM `bugs`")

bugs = cursor.fetchall()

bug\_added = (appid, authorid, title, status)

app\_found = False

user\_found = False

bug\_already = False

for app in apps:

if app[0] == appid:

app\_found = True

if not app\_found:

return {"status":"app not found"}

for user in users:

if user[0] == authorid:

user\_found = True

if not user\_found:

return {"status":"user not found"}

if bug\_added in bugs:

bug\_already = True

return {"status":"bug already exists"}

sql = "INSERT INTO bugs (app, author, title, status) VALUES (%s, %s, %s, %s)"

cursor.execute(sql, bug\_added)

db.commit()

if cursor.rowcount>0:

return {"status":"success"}

else:

return {"status":"failed to add to DB"}

@app.post("/bugs/delete")

async def bugsDelete(id: int = Form(...)):

cursor = db.cursor()

cursor.execute("SELECT id FROM `bugs`")

bugs = cursor.fetchall()

if (id,) not in bugs:

return {"status":"not found"}

sql = "DELETE FROM bugs WHERE id=%s"

cursor.execute(sql, (id,))

db.commit()

if cursor.rowcount>0:

return {"status":"success"}

else:

return {"status":"failed to delete"}

@app.post("/bugs/edit")

async def bugsEdit(id: int = Form(...), appid: int = Form(...), authorid: int = Form(...), title: str = Form(...), status: str = Form(...)):

cursor = db.cursor()

app\_found = False

user\_found = False

bug\_found = False

cursor.execute("SELECT id FROM `apps`")

apps = cursor.fetchall()

for app in apps:

if app[0] == appid:

app\_found = True

if not app\_found:

return {"status":"app not found"}

cursor.execute("SELECT \* FROM `users`")

users = cursor.fetchall()

for user in users:

if user[0] == authorid:

user\_found = True

if not user\_found:

return {"status":"user not found"}

cursor.execute("SELECT \* FROM `bugs`")

bugs = cursor.fetchall()

for bug in bugs:

if bug[0] == id:

bug\_found = True

if not bug\_found:

return {"status":"not found"}

sql = "UPDATE bugs SET app=%s, author=%s, title=%s, status=%s WHERE id=%s"

cursor.execute(sql, (appid, authorid, title, status, id))

db.commit()

if cursor.rowcount>0:

return {"status":"success"}

else:

return {"status":"failed to update"}

@app.get("/bugs/list/open")

async def bugsGetWorked():

cursor = db.cursor()

cursor.execute("SELECT \* FROM `bugs` WHERE status='open'")

bugs = cursor.fetchall()

if len(bugs)>0:

return {"success":str(bugs)}

else:

return {"status":"no open bugs"}

@app.get("/bugs/list/fixed")

async def bugsGetClosed():

cursor = db.cursor()

cursor.execute("SELECT \* FROM `bugs` WHERE status='fixed'")

bugs = cursor.fetchall()

if len(bugs)>0:

return {"success":str(bugs)}

else:

return {"status":"no fixed bugs"}

@app.post("/users/register")

async def usersRegister(name: str = Form(...), password: str = Form(...), access: str = Form(...)):

cursor = db.cursor()

user\_found = False

cursor.execute("SELECT name FROM `users`")

users = cursor.fetchall()

if (name,) in users:

return {"status":"already exists"}

sql = "INSERT INTO users (name, password, access) VALUES (%s, %s, %s)"

cursor.execute(sql, (name, password, access))

db.commit()

if cursor.rowcount>0:

return {"status":"success"}

else:

return {"status":"failed to add to DB"}

@app.post("/users/login")

async def usersLogin(name: str = Form(...), password: str = Form(...)):

cursor = db.cursor()

user\_found = False

cursor.execute("SELECT \* FROM `users`")

users = cursor.fetchall()

for user in users:

checking = (user[1], user[2])

if checking == (name, password):

id = user[0]

access = user[3]

return {"status":(id, access)}

return {"status":"failed"}

#=====================================

@app.get("/connection/ping")

def pong():

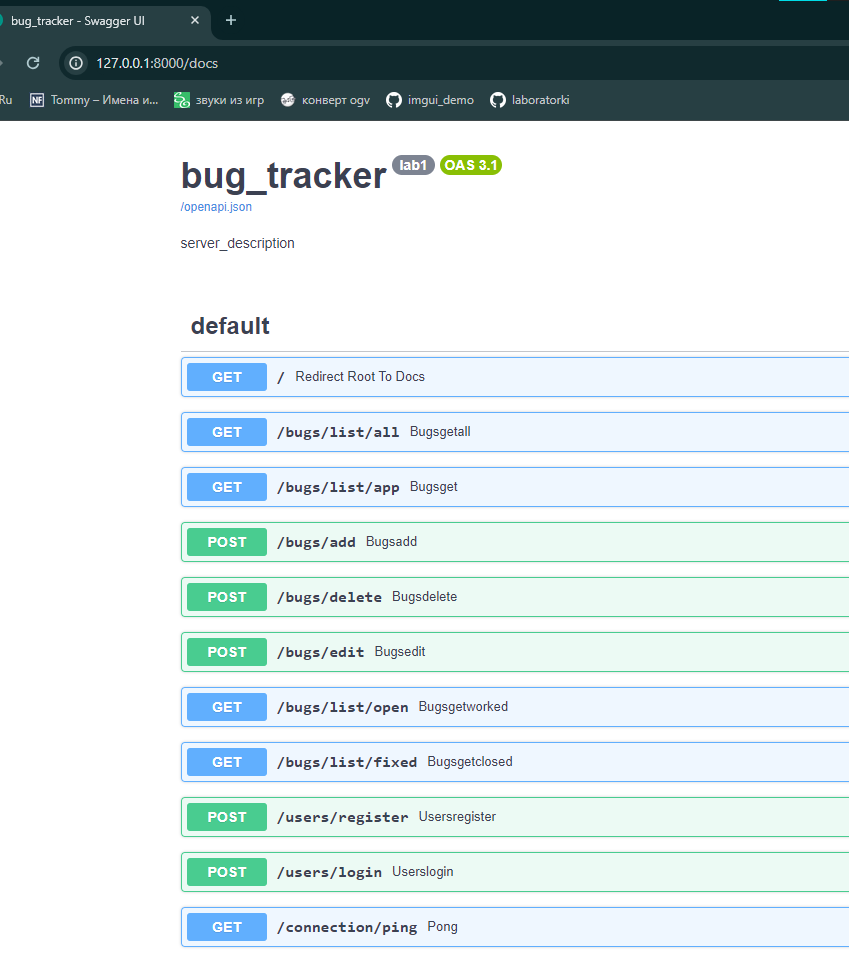
return {"message":"pong"}

if \_\_name\_\_ == "\_\_main\_\_":

import uvicorn

uvicorn.run(app, host="0.0.0.0", port=8000)

Доступные эндпоинты можно увидеть в веб-браузере:



**Вывод.** Научился создавать базовое веб-приложение, предоставляющее API для взаимодействия с базой данных.