**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 9

**На тему:** “Організація взаємодії між процесами”

**З дисципліни:** *“Операційні системи”*

**Лектор:**

Старший викладач ПЗ

Грицай О.Д.

**Виконав:**

ст. гр. ПЗ-22

Солтисюк Д.А.

**Прийняв:**

Старший викладач ПЗ

Грицай О.Д.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_

Львів – 2022

**Тема роботи:** Організація взаємодії між процесами

**Мета роботи:**  Ознайомитися зі способами міжпроцесної взаємодії. Ознайомитися з класичним прикладом взааємодії між процесами на прикладі задачі «виробник – споживач». Навчитися працювати із процесами з використанням способів міжпроцесної взаємодії, синхронізувати їхню роботу. .

**Індивідуальне завдання**

Завдання.

1. Реалізувати алгоритм моделювання заданої задачі за допомогою окремих процесів згідно індивідуального завдання.

2. Реалізувати синхронізацію роботи процесів.

 3. Забезпечити зберігання результатів виконання завдання.

4. Результати виконання роботи відобразити у звіті.

Варіант 4. Створити програму, що моделює наступну ситуацію: Модератори форуму. Користувач реєструється на форумі і його ім’я записується у базу-даних (файл). Після того він може написати [повідомлення](https://vns.lpnu.ua/mod/page/view.php?id=74782). Викликаються модератори форуму. Кожен з яких слідкує за певним забороненим словом. [Повідомлення](https://vns.lpnu.ua/mod/page/view.php?id=74782) може бути виведене на екран, якщо загальна сума заборонених слів не перевищує певне число. Кількість заборонених слів заноситься в базу даних. На форумі можуть працювати кілька користувачів.

**Хід роботи**

**Код серверної частини**

**main.py**

import json

import socket

from threading import Thread

from sqlalchemy import create\_engine, delete

from sqlalchemy.orm import Session

from constants import (DEFAULT\_BAD\_WORDS, LEADER\_HOST, LEADER\_PORT,

MAX\_PAYLOAD\_SIZE)

from leader\_server.database import Base

from leader\_server.handlers.create\_post import create\_post

from leader\_server.handlers.login import login

from leader\_server.handlers.view\_posts import view\_posts

from leader\_server.models.bad\_word import BadWord

from leader\_server.models.post import Post

from leader\_server.models.user import User

HANDLERS = {

"login": login,

"create\_post": create\_post,

"view\_posts": view\_posts,

}

def init\_db\_engine():

engine = create\_engine("sqlite:///db.sqlite", echo=True)

engine.connect()

Base.metadata.create\_all(engine)

return engine

def seed\_data(db):

db.execute(delete(BadWord).where())

db.execute(delete(User).where())

db.execute(delete(Post).where())

db.add\_all([BadWord(value=bad\_word) for bad\_word in DEFAULT\_BAD\_WORDS])

db.commit()

def on\_new\_client(engine, conn, address):

with Session(engine) as db:

# seed\_data(db)

with conn:

print("Listening on: " + str(address))

try:

while True:

data = conn.recv(MAX\_PAYLOAD\_SIZE)

if data:

msg = json.loads(data.decode())

res = HANDLERS[msg["method"]](db, msg)

conn.send(json.dumps(res).encode())

except Exception as e:

print("Error: ", address, e.with\_traceback(None))

def main():

engine = init\_db\_engine()

with socket.socket() as server:

server.bind((LEADER\_HOST, LEADER\_PORT))

# configure how many client the server can listen simultaneously

server.listen(100)

while True:

sock, address = server.accept() # accept new connection

Thread(target=on\_new\_client, args=(engine, sock, address)).start()

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

main()

**Код клієнтської частини**

**main.py**

import pytermgui as ptg

from client.views.login import client\_login\_form, login

def main():

with ptg.WindowManager() as ui:

client\_login\_form(

ui,

login,

)

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

main()

l**ogin.py**

import pytermgui as ptg

from client.socket\_client import socket\_client

from client.views.view\_posts import client\_view\_posts

PROMT\_TO\_KEY = {"Name: ": "name", "Password: ": "password"}

def client\_login\_form(ui, callback):

def submit(window: ptg.Window) -> None:

output = {}

for widget in window:

if isinstance(widget, ptg.InputField):

key = PROMT\_TO\_KEY[widget.prompt]

output[key] = widget.value

continue

callback(ui, output)

ui.remove(window)

inputs = [ptg.InputField(prompt=prompt) for prompt in PROMT\_TO\_KEY.keys()]

window = ptg.Window(

"[secondary]Login",

"",

\*inputs,

"",

["Submit", lambda \*\_: submit(window)],

).center()

ui.add(window)

def login(ui, creds):

res = socket\_client.send\_request(

ui=ui,

method="login",

payload=creds,

)

if res is None:

# retry login

client\_login\_form(ui, login)

return

socket\_client.set\_token(res)

client\_view\_posts(ui)

**socket\_client.py**

import json

import socket

from constants import LEADER\_HOST, LEADER\_PORT, MAX\_PAYLOAD\_SIZE

class SocketClient:

def \_\_init\_\_(self):

self.client\_token = None

self.client = socket.socket()

self.client.connect((LEADER\_HOST, LEADER\_PORT))

def set\_token(self, token):

self.client\_token = token

def get\_token(self):

return self.client\_token

def send\_request(self, method, payload, ui):

self.client.send(

json.dumps(

{

"auth": {"token": self.client\_token},

"method": method,

"payload": payload,

}

).encode()

)

raw\_res = self.client.recv(MAX\_PAYLOAD\_SIZE)

res = json.loads(raw\_res.decode())

message = res["message"]

match res["type"]:

case "error":

ui.toast(

f"[bold red]{message}",

delay=6 \* 10\*\*3,

slot="Alert",

animate=False,

)

return None

case "ok":

ui.toast(

f"[bold green]{message}",

delay=6 \* 10\*\*3,

slot="Alert",

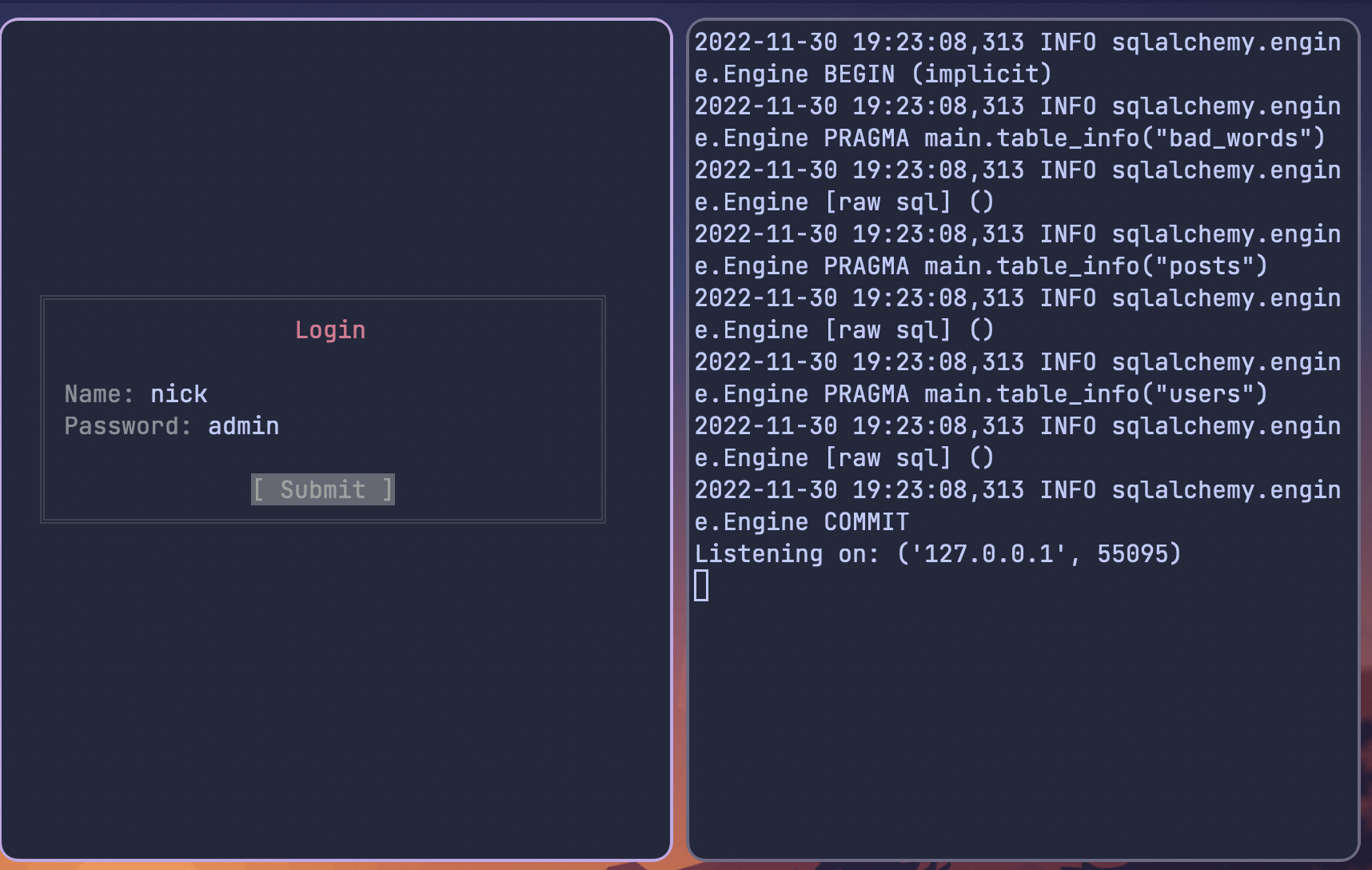
animate=False,

)

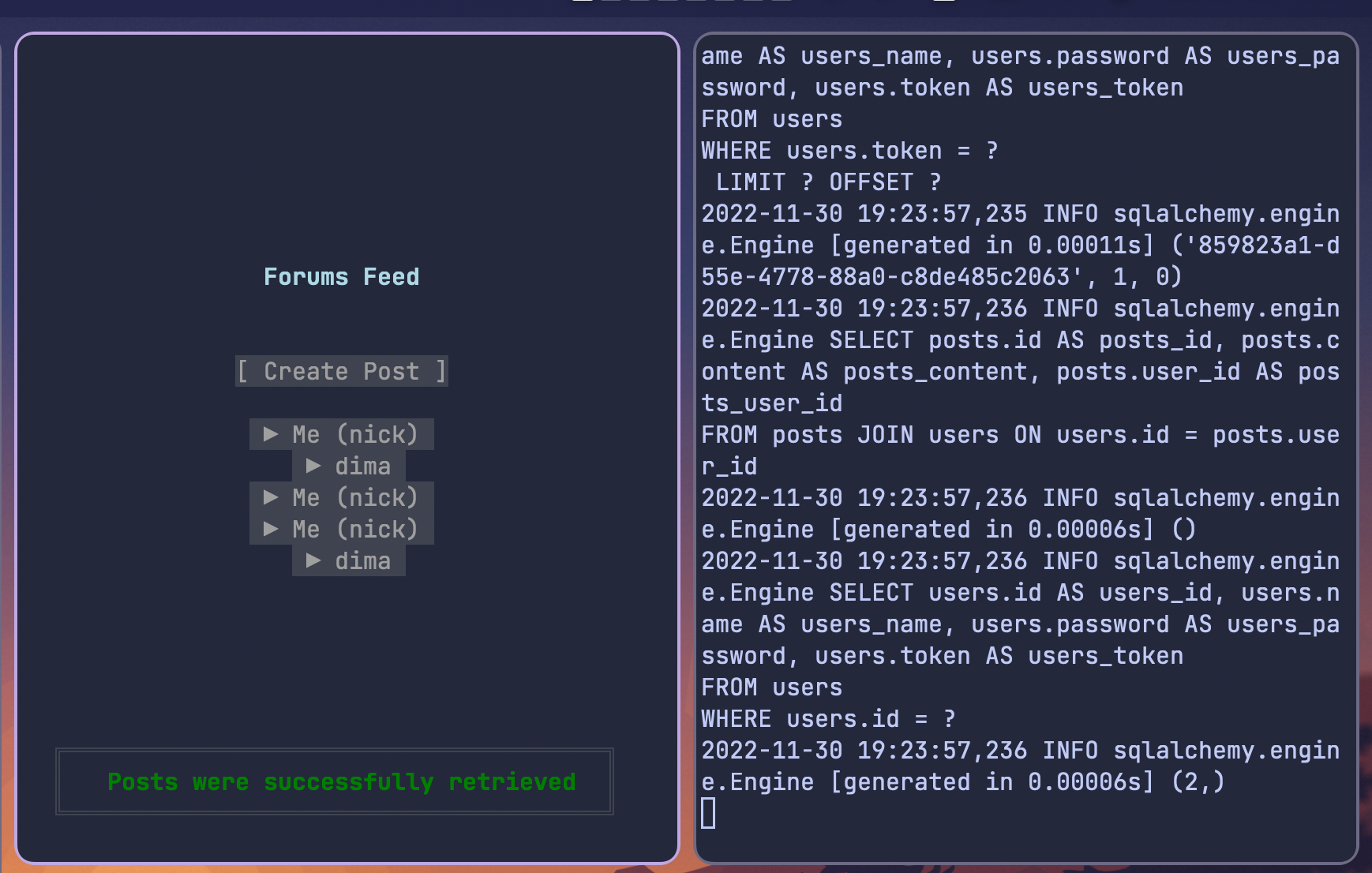
return res["payload"]

socket\_client = SocketClient()

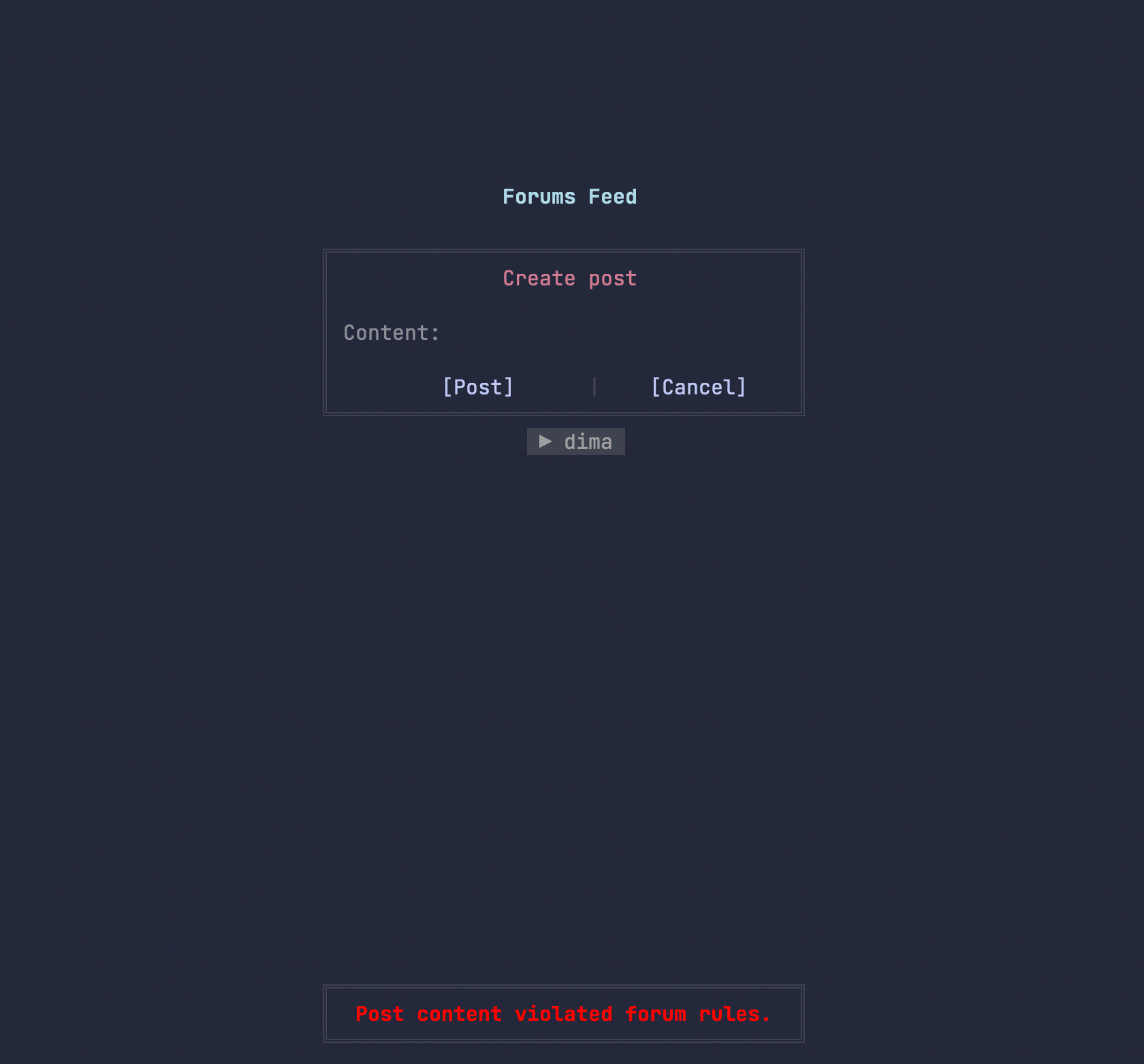
**Приклад використання**

****

*Логін форма*



*Перегляд постів на форумі*



*Створення посту який порушує правила форуму*

**Висновок**

Під час виконання лабораторної роботи я ознайомився зі способами міжпроцесної взаємодії. Та створив програмний комплекс з використанням бази данних