**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
Ясногородський Н.В.

**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. Викликаються модератори форуму. Кожен з яких слідкує за певним забороненим словом. Повідомлення 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()
```

**login.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
```

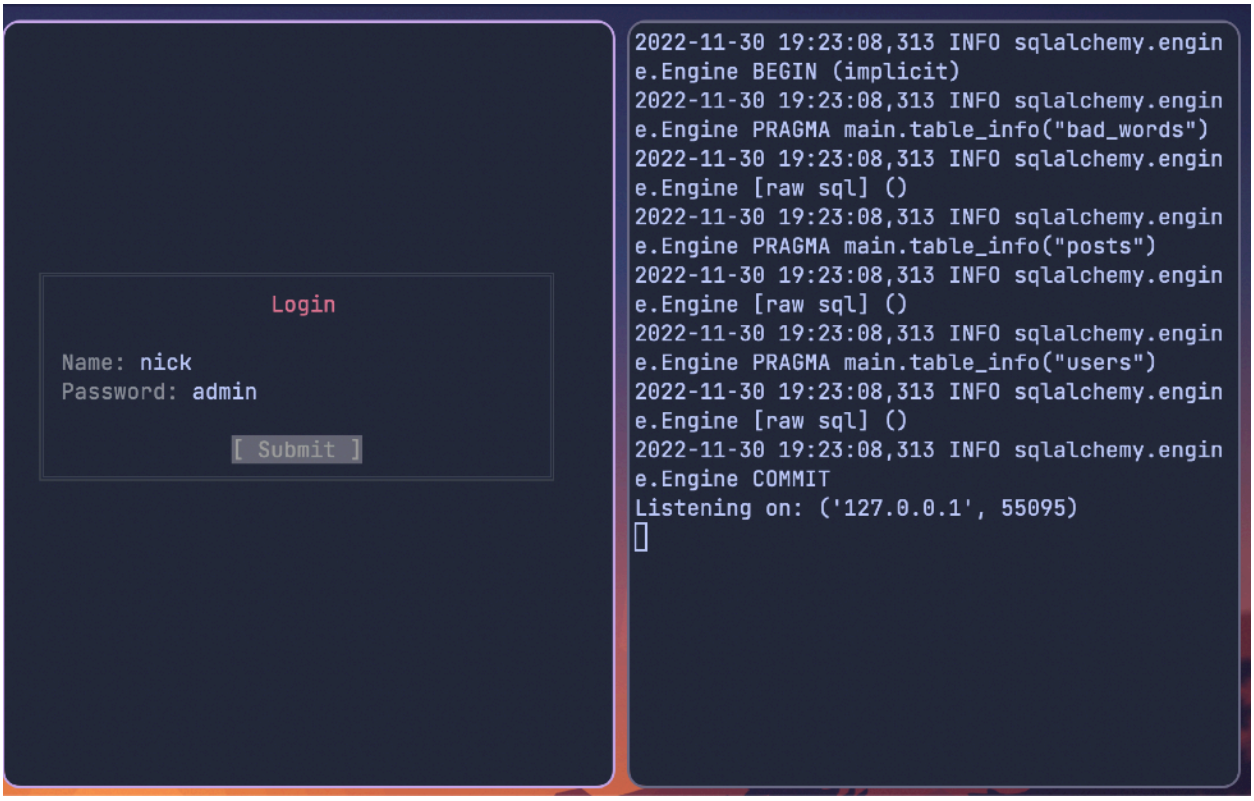
```
    socket_client.set_token(res)

    client_view_posts(ui)
```

**socket_client.py**

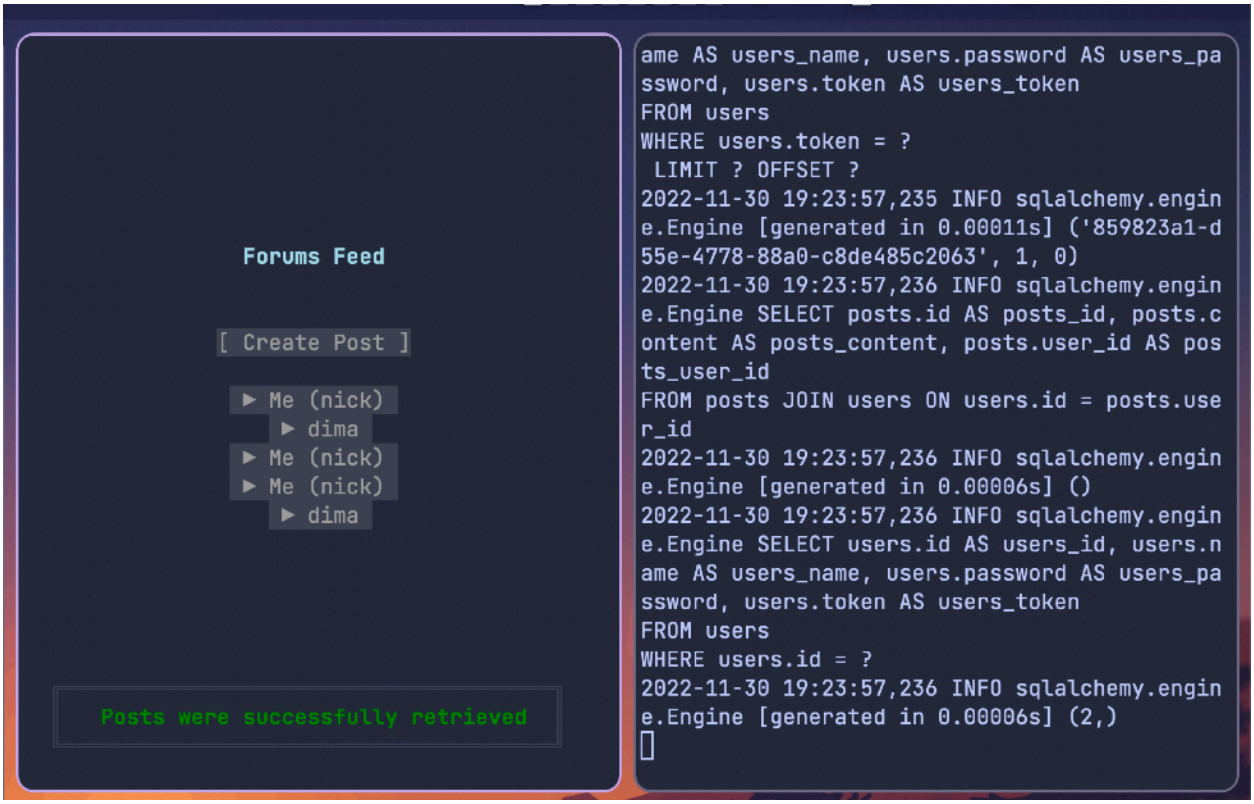
```
import json
import socket

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()
```

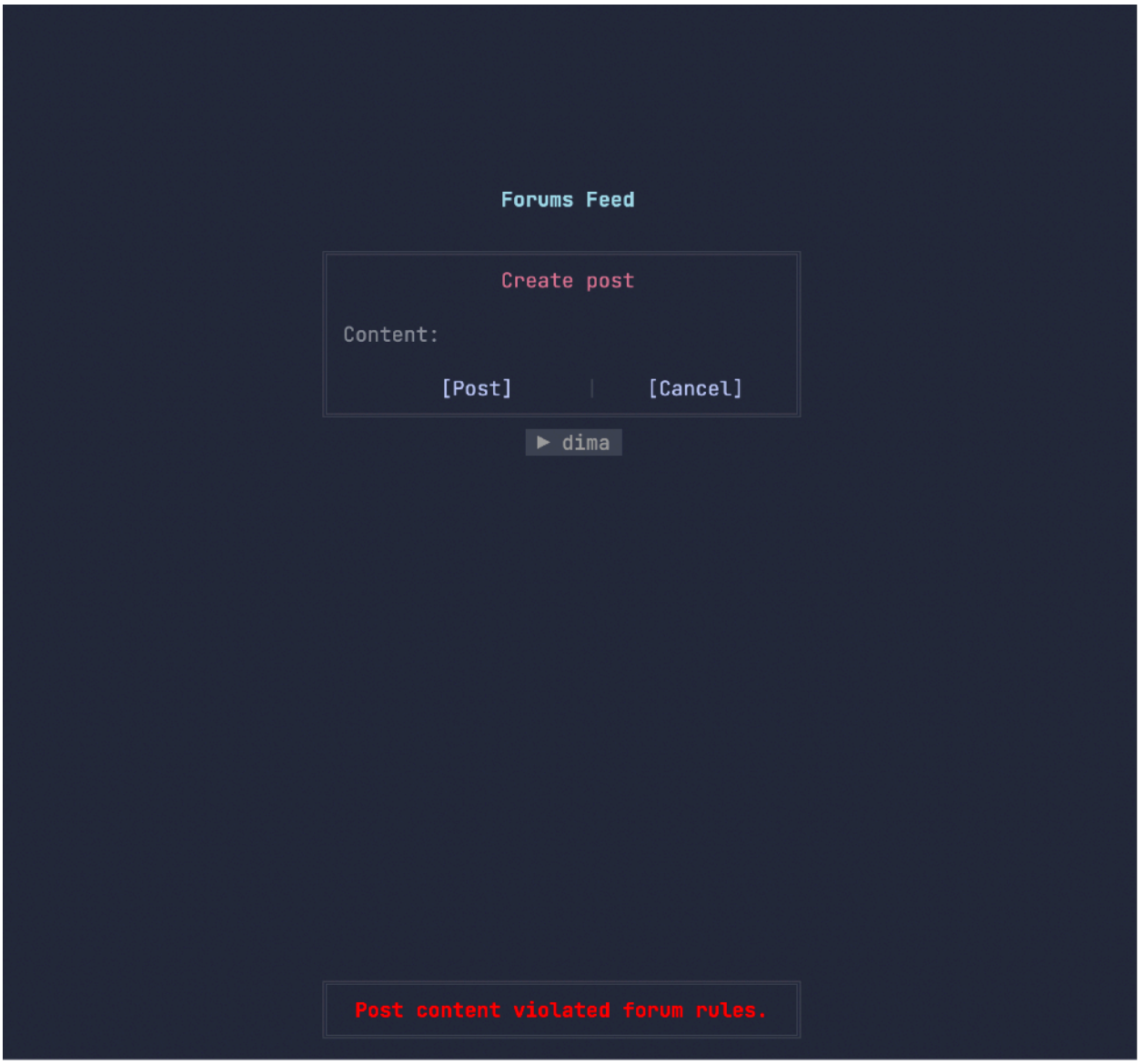
## Приклад використання




*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