



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №9

Технології розроблення програмного забезпечення

РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER,
PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE

Варіант 24

Виконала:
студентка групи ІА-14
Шеліхова А.О

Перевірив:
Мягкий М.Ю.

Київ 2023

РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE

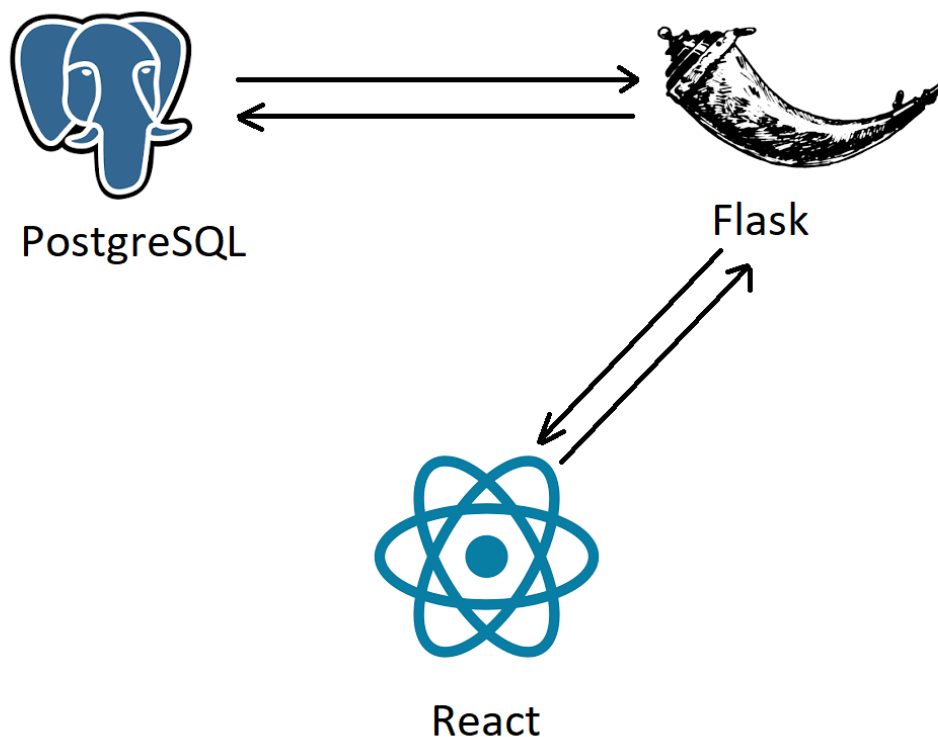
Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.
3. Реалізувати взаємодію програми в одній з архітектур відповідно до обраної теми.

Виконання роботи:

У ході роботи було створено додаток для реєстрації нових користувачів, збереження їх у базі даних, та авторизації користувачів. Додаток побудований відповідно до Service-Oriented Architecture. Frontend було реалізовано на React, backend – на Flask. В якості системи управління базами даних вибрано PostgreSQL. Для авторизації використовується JWT-токен.

Архітектура додатку:



Код програми:

Головний код додатку на Flask:

```
import json

from datetime import datetime as dt, timezone as tz

from flask import Flask, jsonify
from flask_cors import CORS
from flask_jwt_extended import JWTManager, create_access_token, get_jwt, get_jwt_identity

from regauth.regauth import regauth

import utils.application as app_utils

from config.application import UPDATE_JWT_ACCESS_TOKEN_IN

app = Flask(__name__)

CORS(app)
JWTManager(app)

app_utils.init(app)

app_utils.register_blueprints(app, regauth)

@app.route('/')
def index():
    return jsonify({'success': True})

@app.after_request
def refresh_token(response):

    try:

        exp_timestamp = get_jwt()["exp"]

        now = dt.now(tz.utc)

        target_timestamp = dt.timestamp(now + UPDATE_JWT_ACCESS_TOKEN_IN)

        if target_timestamp > exp_timestamp:

            access_token = create_access_token(identity=get_jwt_identity())
            data = response.get_json()

            if type(data) is dict:

                data["access_token"] = access_token
                response.data = json.dumps(data)

        return response

    except Exception:
        return response

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

Головний код додатку на React:

```
import { useState, useEffect } from "react";

import { BrowserRouter } from "react-router-dom";
```

```

import Navbar from "../components/navbar/Navbar";
import AppRouter from "../components/AppRouter";
import Flash from "../components/flash/Flash";

import { AppContext } from "../context";

import './App.css';

function App() {

  const [jwtToken, setJwtToken] = useState(null);
  const [user, setUser] = useState(null);

  const [flashMessage, setFlashMessage] = useState(null);
  const [flashMessageType, setFlashMessageType] = useState(null);

  const [flashRedirect, setFlashRedirect] = useState(false);

  const [isLoading, setIsLoading] = useState(true);

  useEffect(
    () => {

      if (localStorage.getItem('jwtToken') && (localStorage.getItem('user')))) {

        setJwtToken(localStorage.getItem('jwtToken'));

        setUser(JSON.parse(localStorage.getItem('user')));

      }

      setIsLoading(false);

    }, []);

  return (

    <AppContext.Provider value={{
      jwtToken, setJwtToken,
      user, setUser,
      flashMessage, setFlashMessage,
      flashMessageType, setFlashMessageType,
      flashRedirect, setFlashRedirect,
      isLoading
    }}>

      <BrowserRouter>

        <Navbar />

        {flashMessage && <Flash message={flashMessage} type={flashMessageType} />}

        <AppRouter />

      </BrowserRouter>
    </AppContext.Provider>


  );
}

export default App;

```

Результати:

Форма реєстрації:



Sign InSign Up

Sign Up

Username

First Name

Last Name


Email

Password

Confirm Password

Sign Up

Успішна реєстрація:



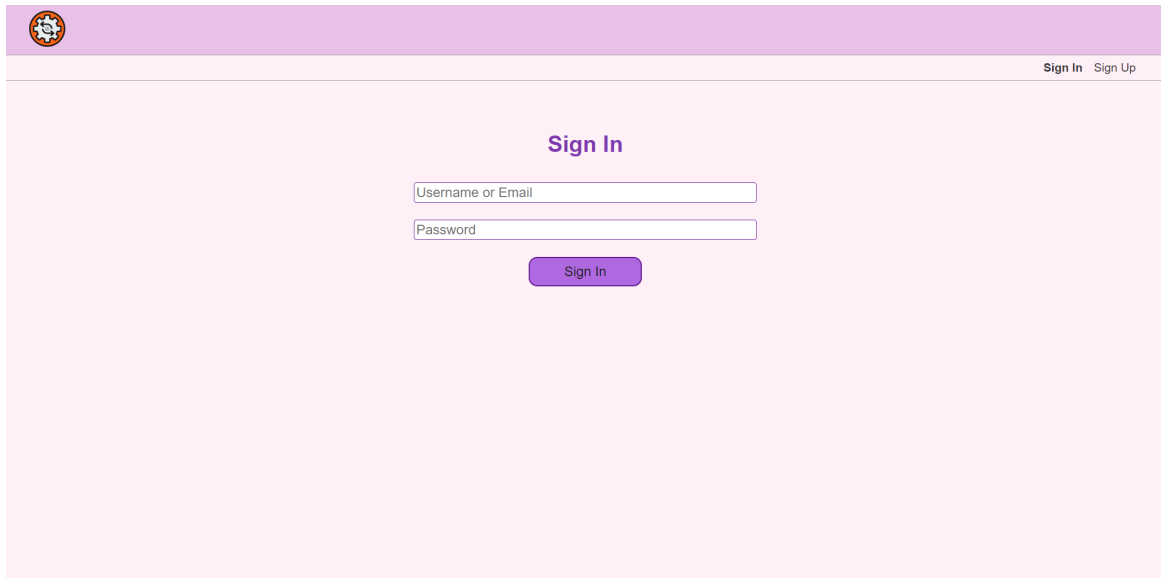
User TestLogout

Welcome, user2023

Дані про користувача успішно збереглися в базі даних:

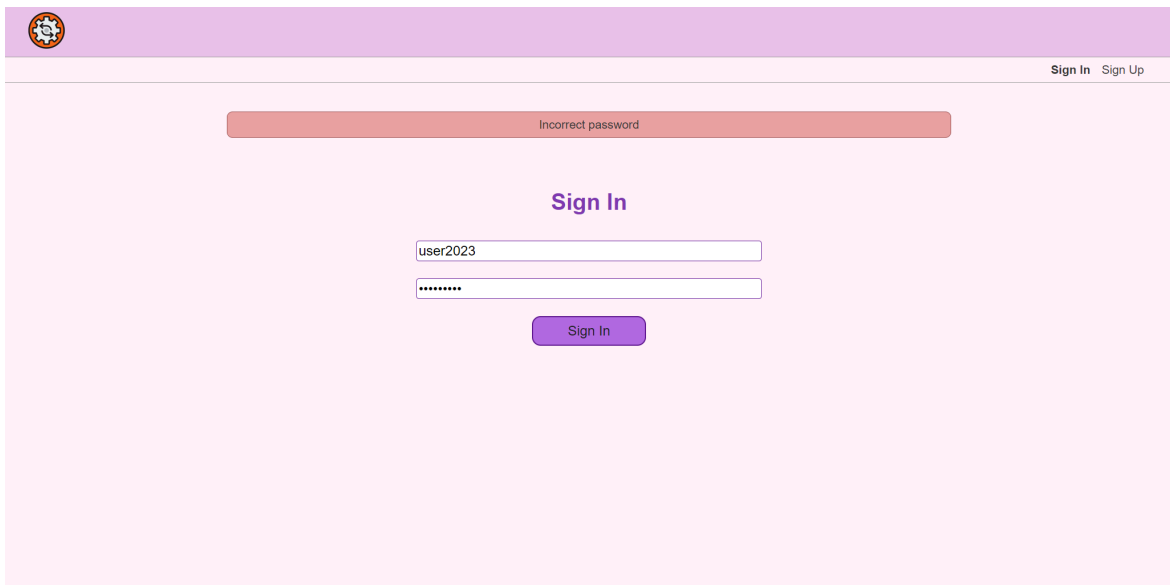
| Data Output | | | | | | | | Messages | Notifications |
|-------------|--------------|------------------------|------------------------|------------------------|------------------------|------------------------|---|----------|---------------|
| | id | username | avatar | first_name | last_name | email | password | | |
| | [PK] integer | character varying (30) | character varying (50) | character varying (50) | character varying (50) | character varying (50) | character varying (118) | | |
| 1 | 5 | user2023 | [null] | User | Test | test@example.com | pbkdf2:sha256:260000\$9NJoZGuqPBHD62avRmf1zjncPo70fyGS269cd7ae5579a0c0300ded7201fe28ae44bed5a3e0cd5545782a61dd02b5... | | |

Форма авторизації:



The screenshot shows a web application interface with a purple header bar. On the left of the header is a circular logo with a gear and a person. On the right are links for "Sign In" and "Sign Up". The main content area has a light pink background. In the center, the text "Sign In" is displayed in purple. Below it are two input fields: "Username or Email" and "Password". A purple "Sign In" button is positioned below the password field.

Помилка авторизації:



The screenshot shows the same web application interface as before, but with an error message. A red horizontal bar at the top of the main content area contains the text "Incorrect password". The "Sign In" text and input fields are still present. The "Username or Email" field now contains the text "user2023", and the "Password" field contains a series of asterisks. The "Sign In" button remains below the password field.

Висновки:

При виконанні даної лабораторної роботи було вивчено Service-Oriented архітектуру. У ході роботи було створено додаток для реєстрації та авторизації користувачів. Додаток побудований відповідно до Service-Oriented Architecture.