НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни

**«Технологія розробки Web-додатків»**

Виконав: студент V курсу

ФПМ групи КВ-32мп

Неживий Б. М.

Перевірив: доц. Петрашенко А. В.

 Київ – 2024

**Загальна інформація**

Завдання на лабораторну роботу: розробити функції обміну даними між користувачами Web-додатка, а також адміністрування користувачами у реальному часі.

Посилання на [Github-репозиторій](https://github.com/n0tark/web-app-2);

Посилання студента в [Telegram](https://t.me/unalive1);

**Назва додатку**: ***PhoneBook***

Цей онлайн інструмент призначений для управління списками контактів. Однією з його ключових особливостей є можливість обміну даними в реальному часі. Додаток також вміє відображати списки активних користувачів та їх діяльність.

Опис функцій:

• Управління контактами: додавання, видалення та редагування контактів.

• Обмін даними в реальному часі: можливість миттєвого обміну інформацією без перезавантаження сторінок.

• Функція адміністратора: перегляд списку активних користувачів та їх активності.

**Технології, використані в розробці**: Python 3, Django, Django Channels, redis, Pycharm Community Edition

**Структура бази даних**

**Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание**

**Демонстрація роботи**

Postman показує шаблон, де реалізовано обмін даних

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

У скрипті налаштовується клієнт WebSocket для взаємодії з сервером через адресу (ws://127.0.0.1:8000/ws/phones/), дозволяючи відправляти та отримувати повідомлення. Також встановлюються обробники подій:

1. onopen: ця функція активується при успішному встановленні з'єднання з сервером і реєструє в консолі повідомлення про успішне з'єднання.
2. onmessage: коли сервер надсилає повідомлення, ця подія відпрацьовує, перетворюючи отримані дані в формат JSON і витягуючи з них номер телефону. Однак код не містить обробки самого номера.
3. onclose: ця функція спрацьовує, коли з'єднання закривається.
4. Функція sendPhoneNumber: вона використовується для відправлення номера телефону на сервер. Приймає номер як аргумент, створює JSON-об'єкт і відправляє його як рядок через WebSocket.

HTML-код створює веб-сторінку зі списком контактів, відображених за допомогою Django-шаблонів, та використовує WebSocket для миттєвої взаємодії з сервером.

<!DOCTYPE html>

<html>

<head>

    <title>Contact List</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f4f4f4;

        }

        h1 {

            background-color: #333;

            color: #fff;

            padding: 10px 15px;

            text-align: center;

        }

        .contact {

            background-color: #fff;

            padding: 10px 15px;

            margin-bottom: 5px;

            border-radius: 5px;

            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

        }

    </style>

</head>

<body>

    <h1>Online Contact List</h1>

    <h2>Online Users:</h2>

    <ul id="onlineUsers"></ul>

        <p id="noUsersMessage" style="display: none;">No users are currently online.</p>

    <h2>All Contacts:</h2>

    <ul>

        {% for contact in contacts %}

            <div class="contact">

                {{ contact.phone\_number }}

            </div>

        {% endfor %}

    </ul>

<script>

        const onlineSocket = new WebSocket('ws://http://127.0.0.1:8000//ws/online/');

        onlineSocket.onmessage = function(event) {

            const message = JSON.parse(event.data);

            const onlineUsers = message['online\_users'];

            const onlineUsersList = document.getElementById('onlineUsers');

            const noUsersMessage = document.getElementById('noUsersMessage');

            onlineUsersList.innerHTML = '';

            if (onlineUsers.length === 0) {

                noUsersMessage.style.display = 'block';

            } else {

                noUsersMessage.style.display = 'none';

                onlineUsers.forEach(function(user) {

                    const li = document.createElement('li');

                    li.textContent = user.username;

                    onlineUsersList.appendChild(li);

                });

            }

        };

        onlineSocket.onopen = function() {

            console.log('WebSocket connection established.');

        };

        onlineSocket.onclose = function() {

            console.log('WebSocket connection closed.');

        };

    </script>

</body>

</html>

Отже, опис стилів і функціональності для даного веб-додатку такий:

- Тіло сторінки має шрифт Arial, світло-сірий фон і відсутні відступи.

- Заголовок h1 центрований з верхнім відступом.

- Елементи списку ul не мають маркерів та відступів.

- Стилізація елементів списку li включає фон, закруглені кути, відтінки і тіні.

- При наведенні на елемент списку li:hover змінюється фон.

- За допомогою li::before додається символ точки перед кожним елементом списку.

- Класи .contact та .phone визначають стилі для імені та номера телефону відповідно.

- Заголовок h1 відображається як "Contact List".

- Елемент ul містить шаблонний код Django, який генерує елементи списку для кожного контакту з колекції contacts.

- Для кожного контакту відображаються ім'я та номер телефону з відповідними стилями.

- Встановлюється нове WebSocket-з'єднання з сервером за адресою ws://127.0.0.1:8000/ws/phones/.

- Обробники подій onmessage, onopen та onclose відповідають за обробку повідомлень від сервера, реєстрацію успішного відкриття та закриття з'єднання відповідно.

- Функція sendPhoneNumber відправляє номер телефону на сервер у форматі JSON.

Для перевірки в розширенні для dev в браузері відображається

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Відслідковування користувачів в онлайні (відсутні такі)

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Код шаблону

Цей HTML-код створює веб-сторінку зі списком контактів, включаючи можливість відображення онлайн користувачів у реальному часі за допомогою WebSocket.

<!DOCTYPE html>

<html>

<head>

    <title>Contact List</title>

     <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            margin: 0;

            padding: 0;

        }

        h1 {

            color: #333;

            text-align: center;

            margin-top: 20px;

        }

        ul {

            list-style-type: none;

            padding: 0;

            margin: 0;

        }

        li {

            background-color: #fff;

            border-radius: 5px;

            margin: 5px 0;

            padding: 10px;

            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

        }

        li:hover {

            background-color: #f9f9f9;

        }

        li::before {

            content: "\2022";

            color: #999;

            display: inline-block;

            width: 1em;

            margin-left: -1em;

        }

        .contact {

            font-weight: bold;

        }

        .phone {

            color: #333;

        }

    </style>

</head>

<body>

     <h1>Contact List</h1>

    <ul>

        {% for contact in contacts %}

            <li>

                <span class="contact">{{ contact.first\_name }} {{ contact.last\_name }}</span> -

                <span class="phone">{{ contact.phone\_number }}</span>

            </li>

        {% endfor %}

    </ul>

  <script>

        const phoneSocket = new WebSocket('ws://127.0.0.1:8000/ws/phones/');

        phoneSocket.onmessage = function(event) {

            const data = JSON.parse(event.data);

            const phoneNumber = data['phone\_number'];

        };

        phoneSocket.onopen = function() {

            console.log('WebSocket connection established.');

        };

        phoneSocket.onclose = function() {

            console.log('WebSocket connection closed.');

        };

        function sendPhoneNumber(phoneNumber) {

            const data = {

                'phone\_number': phoneNumber

            };

            phoneSocket.send(JSON.stringify(data));

        }

    </script>

</body>

</html>

- Тіло сторінки має шрифт Arial, обнулені відступи і світло-сірий фон.

- Заголовок h1 визначає стиль для заголовка сторінки, включаючи фон, колір тексту і відступи.

- Клас .contact визначає стиль для кожного контакту, включаючи фон, відступи, нижній відступ, закруглені кути і тінь.

- Заголовок h1 відображається як "Online Contact List".

- Підзаголовок h2 відображає підзаголовок для списку онлайн користувачів.

- Елемент ul з id="onlineUsers" містить список онлайн користувачів, який буде заповнюватися динамічно через WebSocket.

- Елемент p з id="noUsersMessage" показує повідомлення, коли жоден користувач не онлайн.

- Ще один підзаголовок h2 відображається для всіх контактів.

- Елемент ul містить шаблонний код Django, який генерує елементи списку для кожного контакту в колекції contacts.

- Встановлюється нове WebSocket-з'єднання з сервером за адресою ws://127.0.0.1:8000/ws/online/.

- Обробники подій onmessage, onopen і onclose відповідають за обробку повідомлень від сервера, реєстрацію відкриття і закриття з'єднання відповідно. При отриманні повідомлення вони парсять дані у форматі JSON, очищають список онлайн користувачів і заповнюють його новими даними. Якщо онлайн користувачів немає, виводиться повідомлення "No users are currently online".

Pезультат прикладу, де один користувач є в онлайн

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

У разі відсутності відповідних прав

Изображение выглядит как снимок экрана, программное обеспечение, Мультимедийное программное обеспечение, текст

Автоматически созданное описание

Код програми

**views.py**

Визначає кілька Django views для обробки запитів, що стосуються контактів та онлайн користувачів, а також для автентифікації користувачів за допомогою токенів.

from django.shortcuts import render

from phonebook.models import Contact

from channels.layers import get\_channel\_layer

from asgiref.sync import async\_to\_sync

from django.http import JsonResponse

from rest\_framework.response import Response

from rest\_framework.decorators import api\_view, permission\_classes

from rest\_framework.permissions import IsAuthenticated

from rest\_framework.authtoken.models import Token

def phones(request):

    contacts = Contact.objects.all()

    return render(request, "phones.html", {"contacts": contacts})

def contact\_list(request):

    contacts = Contact.objects.all()

    return render(request, 'contactList.html', {'contacts': contacts})

def user\_online(user\_id):

    channel\_layer = get\_channel\_layer()

    async\_to\_sync(channel\_layer.group\_add)('online\_users', str(user\_id))

def user\_offline(user\_id):

    channel\_layer = get\_channel\_layer()

    async\_to\_sync(channel\_layer.group\_discard)('online\_users', str(user\_id))

def online\_users(request):

    if request.user.is\_superuser:

        channel\_layer = get\_channel\_layer()

        online\_users = async\_to\_sync(channel\_layer.group\_channels)('online\_users')

        return JsonResponse({'online\_users\_count': len(online\_users)})

    else:

        return JsonResponse({'error': 'Only superuser can access this endpoint'}, status=403)

Перевіряємо, чи запитувач є суперкористувачем. Якщо так, він отримує кількість каналів у групі "online\_users" і повертає цю інформацію у форматі JSON. У випадку, якщо запитувач не є суперкористувачем, повертаємо повідомлення про помилку зі статусом 403.

@api\_view(['GET'])

@permission\_classes([IsAuthenticated])

def get\_auth\_token(request):

    user = request.user

    token, created = Token.objects.get\_or\_create(user=user)

    return Response({'token': token.key})

Використовує декоратор @api\_view для визначення дозволених методів HTTP (в даному випадку лише GET). Декоратор @permission\_classes([IsAuthenticated]) забезпечує, що доступ до view мають лише автентифіковані користувачі. Отримує користувача з запиту, створює або отримує токен для користувача і повертає токен у форматі JSON.

**Веб-сокети:** функції user\_online та user\_offline дозволяють додавати/видаляти користувачів з групи онлайн користувачів, використовуючи Django Channels.

**Автентифікація:** get\_auth\_token забезпечує API для отримання токену автентифікації, необхідного для доступу до захищених ресурсів.

**Інтерфейс:** views phones та contact\_list забезпечують рендеринг HTML-шаблонів з даними про контакти.

Цей набір views можна використовувати для створення функціоналу онлайн телефонної книги з можливістю автентифікації користувачів і відстеження онлайн статусу.

**urls.py**

Визначає маршрути для різних представлень (views) Django-додатку.

from django.urls import path

from . import views

urlpatterns = [

    path("", views.phones, name="phones"),

    path('online\_users/', views.online\_users, name='online\_users'),

    path('contacts/', views.contact\_list, name='contact\_list'),

]

**path("", views.phones, name="phones"),**

Це маршрут за замовчуванням (кореневий URL, наприклад, http://localhost:8000/).

Викликає функцію views.phones, яка рендерить шаблон phones.html з усіма контактами.

Назва маршруту (name="phones") дозволяє посилатися на цей маршрут у шаблонах або іншому коді.

**path('online\_users/', views.online\_users, name='online\_users'),**

Викликає функцію views.online\_users.

Цей маршрут повертає кількість онлайн користувачів у форматі JSON, якщо запит виконує суперкористувач.

Назва маршруту (name='online\_users') дозволяє посилатися на цей маршрут у шаблонах або іншому коді.

**path('contacts/', views.contact\_list, name='contact\_list'),**

Викликає функцію views.contact\_list, яка рендерить шаблон contactList.html з усіма контактами.

Назва маршруту (name='contact\_list') дозволяє посилатися на цей маршрут у шаблонах або іншому коді.

**consumers.py**

Цей код визначає два WebSocket консюмера для обробки повідомлень веб-сокетів: один для оновлення онлайн статусу користувачів, а інший для обробки повідомлень про телефонні номери.

import json

from channels.generic.websocket import AsyncWebsocketConsumer

class OnlineStatusConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        await self.accept()

    async def disconnect(self, close\_code):

        pass

    async def receive(self, text\_data):

        data = json.loads(text\_data)

        user\_id = data['user\_id']

        is\_online = data['is\_online']

        await self.channel\_layer.group\_send(

            'online\_users\_group',

            {

                'type': 'user\_status',

                'user\_id': user\_id,

                'is\_online': is\_online

            }

        )

    async def user\_status(self, event):

        user\_id = event['user\_id']

        is\_online = event['is\_online']

        await self.send(text\_data=json.dumps({

            'user\_id': user\_id,

            'is\_online': is\_online

        }))

class PhoneConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        await self.accept()

    async def disconnect(self, close\_code):

        pass

    async def receive(self, text\_data):

        data = json.loads(text\_data)

        phone\_number = data['phone\_number']

        await self.channel\_layer.group\_send(

            'phones\_group',

            {

                'type': 'phone\_message',

                'phone\_number': phone\_number

            }

        )

    async def phone\_message(self, event):

        phone\_number = event['phone\_number']

        await self.send(text\_data=json.dumps({

            'phone\_number': phone\_number

        }))

1. **OnlineStatusConsumer:**

Консюмер для обробки оновлення статусу онлайн користувачів.

connect(): Метод, що викликається при підключенні нового клієнта. Він приймає з'єднання WebSocket.

disconnect(close\_code): Метод, який викликається при відключенні клієнта. В даному випадку нічого не робиться при відключенні.

receive(text\_data): Метод, який викликається при отриманні повідомлення від клієнта. Парсить отриманий JSON, оновлює статус користувача в базі даних (або виконує інші дії, які можна додати в майбутньому) і надсилає повідомлення про статус користувача усім підключеним клієнтам.

user\_status(event): Метод, який викликається для надсилання повідомлення про статус користувача клієнту.

1. **PhoneConsumer:**

Консюмер для обробки повідомлень про телефонні номери.

Має схожу структуру з OnlineStatusConsumer, але обробляє і надсилає повідомлення про телефонні номери.

Ці консюмери можуть використовуватися для обробки повідомлень WebSocket, що надходять з клієнта. Наприклад, вони можуть бути використані для надсилання повідомлень про статус онлайн користувачів або про вхідні дзвінки на певний телефонний номер.

**routing.py**

Цей код налаштовує шляхи WebSocket для вашого додатку Django.

from django.urls import re\_path

from .consumers import OnlineStatusConsumer, PhoneConsumer

websocket\_urlpatterns = [

    re\_path(r'ws/online/$', OnlineStatusConsumer.as\_asgi()),

    re\_path(r'ws/phones/$', PhoneConsumer.as\_asgi()),

]

Відповідно прописано всі налаштування в settings.py

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'phonebook',

    'chat',

    'drf\_yasg',

    'rest\_framework.authtoken',

    'channels'

]

ASGI\_APPLICATION = 'djangoProject1.routing.application'

1. **from django.urls import re\_path**: Цей рядок імпортує функцію re\_path з модуля django.urls. Функція re\_path дозволяє використовувати регулярні вирази для визначення шляхів URL.
2. **websocket\_urlpatterns**: Це список шляхів WebSocket для вашого додатку.
3. **re\_path(r'ws/online/$', OnlineStatusConsumer.as\_asgi()),**: Цей рядок визначає шлях WebSocket для обробки оновлень статусу онлайн користувачів. Він використовує функцію re\_path для визначення шляху ws/online/, який буде асоційований з консюмером OnlineStatusConsumer. Функція as\_asgi() використовується для перетворення консюмера на ASGI (Asynchronous Server Gateway Interface) об'єкт.
4. **re\_path(r'ws/phones/$', PhoneConsumer.as\_asgi()),**: Цей рядок визначає шлях WebSocket для обробки повідомлень про телефонні номери. Він аналогічний до попереднього рядка, але використовує консюмер PhoneConsumer.
5. **INSTALLED\_APPS**: Цей параметр в файлі settings.py містить список ідентифікаторів всіх встановлених додатків у вашому проекті Django. Додаток channels включено в цей список, що позначає наявність підтримки каналів у вашому додатку.
6. **ASGI\_APPLICATION**: Цей параметр вказує Django, яке додаток використовувати для обробки асинхронних запитів. В даному випадку, використовується додаток djangoProject1.routing.application.