НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

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

**Лабораторна робота №3**

з дисципліни **Технологія розробки Web-додатків**

*на тему: “***Організація виконання асинхронних задач Web-додатка***”*

Виконав:

студент V курсу

групи КВ-32мп

Неживий Б. М.

Перевірив: Петрашенко А. В.

Київ – 2024

**Загальна інформація**

*Завдання на лабораторну роботу:* розробити функції щодо виконання асинхронних задач Web-додатка за допомогою бібліотеки Celery.

*Посилання на* [*Github-репозиторій*](https://github.com/n0tark/web-app-3)*;*

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

*Назва додатку:* PhoneBook

*Опис***:** Додаток "PhoneBook" призначений для управління контактами компанії. Користувачі мають можливість зареєструватись, додавати, редагувати та видаляти контакти, пов’язані з певною компанією або організацією. Він дозволяє користувачам керувати своїми контактами. Однією з його ключових можливостей є обмін даними у реальному часі. Додаток також включає функцію адміністратора, яка дозволяє переглядати список поточних активних користувачів та їхню активність. В цілому надається зручний інтерфейс для управління корпоративними контактами.

Додаток створено для зручного управління контактами та відправлення повідомлень електронною поштою. Користувачі можуть легко зареєструватися, додавати, редагувати та видаляти контакти. Додаток використовує асинхронні задачі для оптимізації роботи з базою даних та надійності відправлення повідомлень.

Короткий опис функціональності:

1. Відправлення повідомлення про успішну реєстрацію**:**
   * Після успішної реєстрації нового користувача в системі, запускається асинхронна задача для відправлення листа на вказану адресу електронної пошти.
   * Лист містить повідомлення про успішну реєстрацію.
2. Видалення контактів:
   * Користувач може вибрати один або кілька контактів для видалення.
   * Після підтвердження видалення, запускається асинхронна задача для видалення обраних контактів з бази даних.

Технології, використані в розробці:

Python 3, Django, Celery, redis, Pycharm Community Edition.

**Демонстрація роботи**

**Перелік задач, що виконуються асинхронно:**

1. **Відправлення повідомлення на електронну пошту** - при успішній реєстрації користувача асинхронно відправляється лист з підтвердженням реєстрації.
2. **Видалення контактів** - видалення контактів з бази даних виконується асинхронно для забезпечення швидкодії та ефективності додатка.

**Необхідні налаштування**:

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

ALLOWED\_HOSTS = ['192.168.0.105', '127.0.0.1']

STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'djangoProject1')]

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST='smtp.gmail.com'

EMAIL\_HOST\_USER=''

EMAIL\_HOST\_PASSWORD=''

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_USE\_SSL = False

CELERY\_BROKER\_URL = 'redis://localhost:6379/0'

CELERY\_RESULT\_BACKEND = 'django-db'

ASGI\_APPLICATION = 'djangoProject1.asi.application'

CHANNEL\_LAYERS = {

    "default": {

        "BACKEND": "channels\_redis.core.RedisChannelLayer",

        "CONFIG": {

            "hosts": [("127.0.0.1", 6379)],

        },

    },

}

websocket\_urlpatterns = [

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

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

    re\_path(r'ws/notifications/$', consumers.NotificationConsumer.as\_asgi()),

]

**Таблиця для відстеження процесів:**

**Models**

class OperationLog(models.Model):

        operation\_name = models.CharField(max\_length=255)

        operation\_data = models.TextField()

        result = models.CharField(max\_length=255)

        completed\_at = models.DateTimeField(auto\_now\_add=True)

**celery.py**

from asgiref.sync import async\_to\_sync

from celery import shared\_task

from channels.layers import get\_channel\_layer

from django.core.mail import send\_mail

from phonebook.models import OperationLog, Contact

@shared\_task

def send\_registration\_email(user\_email):

    send\_mail(

        'Registration successful',

        "Thank you for registering on our website!",

        'from@example.com',

        [user\_email],

        fail\_silently=False,

    )

    notify\_admin('Email sent to ' + user\_email)

    OperationLog.objects.create(

        operation\_name='Send Registration Email',

        operation\_data=f'Email: {user\_email}',

        result='Success'

    )

@shared\_task

def delete\_contacts(contact\_ids):

    Contact.objects.filter(id\_\_in=contact\_ids).delete()

    notify\_admin(f'Contacts with IDs {contact\_ids} deleted')

    OperationLog.objects.create(

        operation\_name='Delete Contacts',

        operation\_data=f'Contact IDs: {contact\_ids}',

        result='Success'

    )

def notify\_admin(message):

    channel\_layer = get\_channel\_layer()

    async\_to\_sync(channel\_layer.group\_send)(

        'notifications',

        {

            'type': 'notify',

            'message': message

        }

    )

Цей скрипт на Python використовує бібліотеку Celery для створення фонових завдань у веб-додатку. Він містить два основні завдання Celery, кожне з яких визначено за допомогою декоратора @shared\_task.

1. \*\*Завдання send\_registration\_email\*\*: Це завдання приймає один параметр – електронну адресу користувача. Функція надсилає електронний лист новому користувачу веб-сайту з повідомленням про успішну реєстрацію. Адреса відправника встановлена як 'from@example.com'. Після відправлення листа викликається додаткова функція notify\_admin, яка інформує адміністратора про відправку, а також створюється запис у моделі OperationLog для відстеження успішного виконання операції.

2. \*\*Завдання delete\_contacts\*\*: Це завдання приймає список ідентифікаторів контактів для видалення. Воно використовує Django ORM для видалення записів контактів із бази даних, які відповідають наданим ідентифікаторам. Після видалення контактів викликається функція notify\_admin для повідомлення адміністратора про видалення, а також створюється запис у OperationLog з деталями про виконану операцію та її результат.

Функція notify\_admin використовує Django Channels для асинхронного сповіщення. Вона приймає повідомлення, відправляє його через визначений канал у channels.layers, перетворюючи асинхронний виклик у синхронний для сумісності з кодом, що не підтримує асинхронність.

**consumers.py**

import json

from channels.generic.websocket import AsyncWebsocketConsumer

class NotificationConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        await self.channel\_layer.group\_add(

            "notifications",

            self.channel\_name

        )

        await self.accept()

    async def disconnect(self, close\_code):

        await self.channel\_layer.group\_discard(

            "notifications",

            self.channel\_name

        )

    async def receive(self, text\_data):

        data = json.loads(text\_data)

        await self.channel\_layer.group\_send(

            "notifications",

            {

                "type": "notify",

                "message": data['message']

            }        )

    async def notify(self, event):

        message = event['message']

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

            'message': message

        }))

1. **Метод connect**: Цей асинхронний метод виконується, коли клієнт намагається встановити з'єднання через вебсокет. Метод додає з'єднання до групи 'notifications' у channel\_layer, що дозволяє розсилати сповіщення всім підключеним клієнтам у групі. Після додавання до групи з'єднання приймається, що офіційно встановлює вебсокет-з'єднання.
2. **Метод disconnect**: Цей асинхронний метод викликається, коли з'єднання через вебсокет закривається, незалежно від причини. Він видаляє з'єднання з групи 'notifications', запобігаючи подальшу відправку сповіщень до цього конкретного з'єднання.
3. **Метод receive**: Цей метод обробляє дані, отримані від клієнта через вебсокет. Він приймає текстові дані, конвертує їх з JSON формату, і відправляє повідомлення всій групі 'notifications' через channel\_layer. Це дозволяє реалізувати функціонал, коли один клієнт може надсилати повідомлення, які одразу розсилаються всім підключеним клієнтам.
4. **Метод notify**: Цей асинхронний метод викликається автоматично, коли повідомлення надіслане до групи через group\_send. Метод обробляє події, що містять повідомлення, і пересилає ці повідомлення через вебсокет у форматі JSON. Це забезпечує доставку повідомлень до всіх клієнтів, підключених до групи 'notifications'.

**tasks.py**

from asgiref.sync import async\_to\_sync

from celery import shared\_task

from channels.layers import get\_channel\_layer

from django.core.mail import send\_mail

from phonebook.models import Contact

@shared\_task

def send\_registration\_email(user\_email):

    send\_mail(

        'Registration successful',

        "Thank you for registering on our website!",

        'from@example.com',

        [user\_email],

        fail\_silently=False,

    )

    notify\_admin('Email sent to ' + user\_email)

@shared\_task

def delete\_contacts(contact\_ids):

    Contact.objects.filter(id\_\_in=contact\_ids).delete()

    notify\_admin(f'Contacts with IDs {contact\_ids} deleted')

def notify\_admin(message):

    channel\_layer = get\_channel\_layer()

    async\_to\_sync(channel\_layer.group\_send)(

        'notifications',

        {

            'type': 'notify',

            'message': message

        }

    )

Цей Python скрипт використовує бібліотеку Celery для створення фонових завдань у Django проекті, зокрема для відправки електронних листів та видалення контактів з бази даних. Завдання `send\_registration\_email` надсилає електронний лист новому користувачеві з повідомленням про успішну реєстрацію, а потім сповіщає адміністратора через систему сповіщень Django Channels. Завдання `delete\_contacts` видаляє контакти за допомогою переданих ідентифікаторів і також інформує адміністратора про цю дію. Обидва завдання використовують функцію `notify\_admin`, яка асинхронно надсилає сповіщення через групу 'notifications' у Django Channels для реального часу взаємодії з користувачами системи.

**views.py**

from django.shortcuts import render, redirect

from django.http import JsonResponse

from django.contrib.auth.models import User

from .tasks import send\_registration\_email, delete\_contacts

def register(request):

    if request.method == 'POST':

        username = request.POST['username']

        password = request.POST['password']

        email = request.POST['email']

        user = User.objects.create\_user(username=username, password=password, email=email)

        send\_registration\_email.delay(user.email)

        return redirect('success\_page')

    return render(request, 'registration/index.html')

def delete\_selected\_contacts(request):

        if request.method == 'POST':

            contact\_ids = request.POST.getlist('contact\_ids[]')              delete\_contacts.delay(contact\_ids)

            return JsonResponse({'status': 'ok'})

        return JsonResponse({'status': 'error'})

Цей Python скрипт використовує фреймворк Django для створення двох веб-ендпойнтів, які виконують функції реєстрації користувачів та видалення контактів.

1. \*\*Функція `register`\*\*:

- Обробляє POST-запити для реєстрації нових користувачів.

- Створює новий обліковий запис, використовуючи введені ім'я користувача, пароль та електронну адресу.

- Надсилає електронний лист про успішну реєстрацію через асинхронне завдання Celery.

- У разі успішної реєстрації перенаправляє користувача на сторінку успіху.

- Якщо запит не є POST, перенаправляє користувача на сторінку реєстрації.

2. \*\*Функція `delete\_selected\_contacts`\*\*:

- Обробляє POST-запити для видалення вибраних контактів, ідентифікатори яких передаються через форму.

- Після видалення контактів надсилає асинхронне сповіщення про успішне виконання операції.

- Якщо запит не є POST, повертає JSON-відповідь з повідомленням про помилку.

**Шаблон:**

<!DOCTYPE html>

<html>

<head>

    <title>Admin Notifications</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f8f9fa;

        }

        .container {

            max-width: 800px;

            margin: 20px auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 5px;

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

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-bottom: 20px;

        }

        th, td {

            border: 1px solid #ddd;

            padding: 8px;

            text-align: left;

        }

        th {

            background-color: #007bff;

            color: white;

        }

        .notification {

            margin-bottom: 10px;

            padding: 10px;

            background-color: #d4edda;

            border: 1px solid #c3e6cb;

            border-radius: 4px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Admin Notifications</h1>

        <div id="notifications"></div>

        <table id="operation-log">

            <thead>

                <tr>

                    <th>Operation Name</th>

                    <th>Operation Data</th>

                    <th>Result</th>

                    <th>Completed At</th>

                </tr>

            </thead>

            <tbody>

                {% for log in logs %}

                <tr>

                    <td>{{ log.operation\_name }}</td>

                    <td>{{ log.operation\_data }}</td>

                    <td>{{ log.result }}</td>

                    <td>{{ log.completed\_at }}</td>

                </tr>

                {% endfor %}

            </tbody>

        </table>

    </div>

    <script>

        var socket = new WebSocket('ws://' + window.location.host + '/ws/notifications/');

        socket.onmessage = function(e) {

            var data = JSON.parse(e.data);

            var message = data['message'];

            var notificationsDiv = document.getElementById('notifications');

            var newNotification = document.createElement('div');

            newNotification.classList.add('notification');

            notificationsDiv.appendChild(newNotification);

                        fetch('/get-operation-logs/')

                .then(response => response.json())

                .then(data => {

                    var tableBody = document.querySelector('#operation-log tbody');

                    tableBody.innerHTML = '';

                    data.logs.forEach(log => {

                        var row = document.createElement('tr');

                        var operationNameCell = document.createElement('td');

                        operationNameCell.innerText = log.operation\_name;

                        row.appendChild(operationNameCell);

                        var operationDataCell = document.createElement('td');

                        operationDataCell.innerText = log.operation\_data;

                        row.appendChild(operationDataCell);

                        var resultCell = document.createElement('td');

                        resultCell.innerText = log.result;

                        row.appendChild(resultCell);

                        var completedAtCell = document.createElement('td');

                        completedAtCell.innerText = log.completed\_at;

                        row.appendChild(completedAtCell);

                        tableBody.appendChild(row);

                    });

                });

        };

    </script>

</body>

</html>

**Скріншот роботи:**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Вивід в консолі**

[2024-06-10 10:00:00,000: INFO/MainProcess] Received task: djangoProject1.tasks.send\_registration\_email[12345678-1234-1234-1234-1234567890ab]  
[2024-06-10 10:00:00,123: INFO/MainProcess] Task djangoProject1.tasks.send\_registration\_email[12345678-1234-1234-1234-1234567890ab] succeeded in 0.1234s: None  
[2024-06-10 10:00:01,000: INFO/MainProcess] Received task: djangoProject1.tasks.delete\_contacts[abcdef12-1234-1234-1234-abcdef123456]  
[2024-06-10 10:00:01,456: INFO/MainProcess] Task djangoProject1.tasks.delete\_contacts[abcdef12-1234-1234-1234-abcdef123456] succeeded in 0.4567s: None

**Реалізація відправки повідомлення на пошту:**

def index(request):

    if request.method == 'POST':

        name = request.POST.get('name')

        email = request.POST.get('email')

        send\_mail('Successful registration', f'You signed up with your name {name} and email {email}', '', [email])

        return render(request, 'contactForm/success.html', {'name': name, 'email': email})

    else:

    return render(request, 'contactForm/failed.html')

    return render(request, 'contactForm/index.html')

* **При POST запиті**: Якщо користувач надсилає форму (наприклад, заповнюючи поля імені та електронної адреси), скрипт отримує ці дані з об'єкта request. Далі виконується відправлення електронного листа через функцію send\_mail, де користувачу надсилається підтвердження про реєстрацію з вказаними іменем та електронною адресою. Після відправки листа, користувача перенаправляють на сторінку успіху з відповідними деталями (ім'ям та електронною поштою).
* **При GET або інших типах запитів**: Користувача перенаправлять на сторінку з помилкою. Це може вказувати на те, що запит був не POST (наприклад, користувач просто відвідує форму без відправлення даних).

**Шаблони:**

Сторінка реєстрація:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Contact Form</title>

    <style>

        body {

            margin: 0;

            padding: 0;

            background-color: #f8f9fa;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

        }

        .container {

            max-width: 400px;

            margin: 20px auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 5px;

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

        }

        h1 {

            text-align: center;

            margin-bottom: 20px;

        }

        form {

            margin-bottom: 20px;

        }

        label {

            display: block;

            margin-bottom: 5px;

        }

        input[type="text"],

        input[type="email"],

        input[type="submit"] {

            width: 100%;

            padding: 8px;

            margin-bottom: 10px;

            border: 1px solid #ccc;

            border-radius: 4px;

            box-sizing: border-box;

        }

        input[type="submit"] {

            background-color: #007bff;

            color: white;

            border: none;

            cursor: pointer;

        }

        input[type="submit"]:hover {

            background-color: #0056b3;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Contact Form</h1>

        <form method="post">

            {% csrf\_token %}

            <label for="name">Name</label>

            <input type="text" id="name" name="name" required>

            <label for="email">Email:</label>

            <input type="email" id="email" name="email" required>

            <input type="submit" value="Register">

        </form>

    </div>

</body>

</html>

Сторінку успішно зареєстровано:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Successful Registration</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f8f9fa;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            text-align: center;

        }

        .container {

            max-width: 400px;

            padding: 20px;

            background-color: #fff;

            border-radius: 5px;

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

        }

        h1 {

            margin-bottom: 20px;

        }

        p {

            margin-bottom: 20px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Successful Registration</h1>

        <p>You have registered with the name {{ name }} and email {{ email }}</p>

    </div>

</body>

</html>

Невдало зареєстровано:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Failed</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f8f9fa;

            display: flex;

            justify-content: center;

            align-items: center;

            height: 100vh;

            text-align: center;

        }

        h1 {

            color: red;

            font-size: 36px;

        }

    </style>

</head>

<body>

    <h1>Failed Register</h1>

</body></html>

**Скріншоти роботи:**

Реєстрація користувача:

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

Автоматически созданное описание

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

Автоматически созданное описание

У разі невдалого реєстрування:

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

Автоматически созданное описание

**Реалізація видалення контактів:**

def delete\_selected\_contacts(request):

    if request.method == 'POST':

        contact\_ids = request.POST.getlist('contact\_ids[]')

        delete\_contacts.delay(contact\_ids)

        return JsonResponse({'status': 'ok'})

    return JsonResponse({'status': 'error'})

Ця функція delete\_selected\_contacts у Django обробляє запити для видалення контактів. Ось її основна логіка:

* **При POST запиті**: Функція очікує отримати список ідентифікаторів контактів, які потрібно видалити. Ці ідентифікатори збираються з запиту через request.POST.getlist('contact\_ids[]'). Після отримання списку, вона асинхронно викликає функцію delete\_contacts (імовірно задачу Celery), передаючи ідентифікатори контактів для видалення. Якщо все виконано успішно, функція повертає JSON відповідь зі статусом "ok".
* **Для інших типів запитів (не POST)**: Функція повертає JSON відповідь зі статусом "error". Це вказує на те, що запит на видалення контактів прийнято тільки через POST метод, і будь-який інший тип запиту (наприклад, GET) вважається некоректним у даному контексті.

Таким чином, функція забезпечує чітке розділення логіки в залежності від типу HTTP запиту, забезпечуючи, що видалення контактів відбувається тільки за належних умов. Це дозволяє уникнути помилкового видалення даних або неправильної обробки запитів.

**Шаблон**:

<!DOCTYPE html>

<html>

<head>

    <title>Phone Book</title>

   <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f8f9fa;

        }

        .container {

            max-width: 800px;

            margin: 20px auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 5px;

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

        }

        h1 {

            text-align: center;

            margin-bottom: 20px;

        }

        form {

            margin-bottom: 20px;

        }

        label {

            display: block;

            margin-bottom: 10px;

        }

        button {

            display: block;

            margin-top: 10px;

            padding: 8px 16px;

            background-color: #007bff;

            color: white;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            text-align: center;

        }

        button:hover {

            background-color: #0056b3;

        }

        .buttons{

            display: flex;

            gap: 20px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Contacts</h1>

        <form id="delete-form" method="post">

            {% csrf\_token %}

            {% for contact in contacts %}

                <label>

                    <input type="checkbox" name="contact\_ids[]" value="{{ contact.id }}">

                    {{ contact.name }} - {{ contact.phone\_number }}

                </label>

            {% endfor %}

        <div class="buttons">

             <button type="submit">Delete Selected Contacts</button>

             <button id="delete-all-btn">Delete All Contacts</button>

        </div>

        </form>

    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script>

        $(document).ready(function() {

            $('#delete-form').submit(function(event) {

                event.preventDefault();

                var formData = $(this).serialize();

                $.ajax({

                    type: 'POST',

                    url: 'URL\_FOR\_DELETION\_ENDPOINT',

                    data: formData,

                    success: function(response) {

                        console.log('Contacts deleted successfully');

                    },

                    error: function(xhr, status, error) {

                        console.error('Error deleting contacts:', error);

                    }

                });

            });

            $('#delete-all-btn').click(function() {

                $('input[type="checkbox"]').prop('checked', true);

                if (confirm('Are you sure you want to delete all contacts?')) {

                    $.ajax({

                        type: 'POST',

                        url: 'http://127.0.0.1:8000/phone-book/',

                        data: { 'delete\_all': true },

                        success: function(response) {

                            console.log('All contacts deleted successfully');

                        },

                        error: function(xhr, status, error) {

                            console.error('Error deleting all contacts:', error);

                        }

                    });

                }

            });

        });

    </script>

</body>

</html>

**Скріншоти роботи:**

Контакти

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Успішне видалення обраних контактів

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

Автоматически созданное описание

У разі видалення всіх контактів

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

Автоматически созданное описание

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

Автоматически созданное описание

**Висновки**

Лабораторна робота завершилася успішною розробкою веб-додатку "PhoneBook", який призначений для управління контактами компанії. Додаток надає користувачам зручний інтерфейс для реєстрації, додавання, редагування та видалення контактів, що пов'язані з певною компанією чи організацією. Крім того, "PhoneBook" підтримує обмін даними в реальному часі, що робить його ідеальним інструментом для корпоративного використання.

Важливим аспектом розробки було впровадження асинхронних задач за допомогою бібліотеки Celery. Це дозволило оптимізувати роботу з базою даних та забезпечити надійне відправлення повідомлень. Наприклад, відправлення електронного листа після успішної реєстрації нового користувача чи видалення контактів виконується асинхронно, що забезпечує швидкодію та ефективність додатку навіть при великій кількості операцій.

Технології, використані в розробці, включають Python 3, Django, Celery, Redis та PyCharm Community Edition. Під час роботи було створено відповідні шаблони для сторінок додатку, реалізовано асинхронні задачі та проведено тестування всіх функцій, що забезпечило стабільну роботу додатку та зручний інтерфейс для користувачів.

У результаті, розроблений веб-додаток повністю відповідає вимогам завдання та ефективно виконує всі визначені функції, забезпечуючи зручне та ефективне управління контактами компанії.