ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Створення Web-застосунку для пошуку вакансій та робітників для фізичної праці мовою Python з використанням Django»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ігор КНИШ
(підпис)

Виконав: здобувач(ка) вищої освіти групи ПД-43

Ігор КНИШ

Керівник: Ігор ГАМАНЮК
Ст. викладач

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Книшу Ігору Олександровичу

1. Тема кваліфікаційної роботи: «Створення Web-застосунку для пошуку вакансій та робітників для фізичної праці мовою Python з використанням Django»
керівник кваліфікаційної роботи Ст. викладач кафедри ІПЗ Ігор ГАМАНЮК
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про пошук вакансій та робітників. Закон України Про зайнятість населення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд застосунків для пошуку вакансій та робітників для фізичної праці.
2.Аналіз інструментальних засобів для пошуку вакансій та робітників, визначення потреб і вимог.
2. Проектування web-застосунку для пошуку вакансій та робітників «NUMO»
3. Програмна реалізація та опис функціонування web-застосунку для пошуку вакансій та робітників «NUMO».

4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Структура бази даних.
5. Діаграма використання.
6. Діаграма послідовності створення оголошення.
7. Екранні форми.
8. Файлова структура проєкту.
9. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03.-13.03.2024	
3	Огляд застосунків для пошуку вакансій та робітників для фізичної праці	14.03.-20.03.2024	
4	Проектування web-застосунку для пошуку вакансій та робітників	21.03.-30.03.2024	
5	Програмна реалізація застосунку «NUMO»	01.04.-22.04.2024	
6	Тестування застосунку	23.04.-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04.-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05.-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач(ка) вищої освіти _____ Ігор КНИШ
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 52 стор., 4 табл., 32 рис., 10 джерел.

Мета роботи – спрощення процесу пошуку вакансій та робітників за допомогою web-застосунку NUMO.

Об'єкт дослідження – процес пошуку вакансій та робітників для фізичної праці.

Предмет дослідження – web-застосунок для пошуку вакансій та робітників для фізичної праці.

Короткий зміст роботи: в ході виконання дипломної роботи, було проаналізовано різні застосунки для пошуку фізичної робітників та вакансій для фізичної роботи. Та було створено web-застосунок «NUMO» в ньому було реалізовано основні ключові функціональні можливості, зокрема: створення профілю користувача, його редагування та видалення, створення різних оголошень, можливість редагування оголошень та видалення. Також було створено можливість додавання окремого типу оголошень «Благодійні оголошення». При створенні оголошень є можливість вказати клас небезпеки роботи.

Сферою використання web-застосунку є розміщення та перегляд оголошень про роботу, пропозиції та пошук вакансій, а також публікація благодійних оголошень.

КЛЮЧОВІ СЛОВА: Web-застосунки з пошуку фізичної роботи, Django, розробка веб-застосунку, Python, SQLite, бази даних, PyCharm.

ЗМІСТ

ВСТУП

У сучасному світі, де швидкість і доступність відіграють ключову роль, цей веб-застосунок стане надійним помічником для тих, хто шукає роботу або хоче найняти працівників. Сьогодні ринок праці змінюється з неймовірною швидкістю, і старі методи пошуку вакансій та працівників стають менш ефективними. Сучасні технології пропонують нові інструменти, які значно полегшують цей процес і роблять його більш продуктивним та зручним для всіх учасників.

Застосунок створений з метою об'єднання всіх учасників ринку праці в одному місці, де вони можуть легко і швидко знаходити одне одного. Роботодавці можуть розміщувати оголошення про вакансії, а ті хто шукають роботу можуть переглядати ці оголошення, фільтрувати їх за різними критеріями або просто подати оголошення про пошук роботи. Це значно економить час і зусилля, які зазвичай витрачаються на традиційні методи пошуку роботи або підбору персоналу

Також в застосунку була приділена увага безпеці даних користувача. Всі особисті дані корисувачів захищені надійними технологіями шифрування, що гарантує конфеденційність інформації.

Зараз, коли технології стрімко розвиваються, а ринок праці стає все більш конкурентним, важливо мати надійний інструмент, який допоможе швидко і легко знайти роботу або працівників. Цей веб-застосунок є саме таким інструментом. Він не тільки спрощує процес пошуку, але й робить його більш зручним і ефективним.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Постановка задачі щодо реалізації web-застосунку для пошуку вакансій та робітників «NUMO»

Веб-застосунок "NUMO" створюється для забезпечення зручного та ефективного майданчика для пошуку вакансій та кандидатів на роботу. Застосунок розробляється на мові програмування Python з використанням фреймворку Django, що надає швидку та масштабовану розробку веб-додатків.

Функціональні вимоги:

1. Реєстрація нових користувачів використовуючи нік користувача, ім’я та фамілію, електронну пошту та пароль. Для підвищення безпеки, в системі реалізована перевірка пароля на міцність, а також перевірка унікальності електронної пошти.
2. Вхід в профіль за допомогою ніку користувача та паролю.
3. Перегляд особистого профілю користувача, де користувач може змінити особисті данні та переглянути всі оголошення які він додав.
4. Видалення профілю користувача. При видаленні профілю користувача, видаляються всі оголошення які він додав на сайт «NUMO».
5. Можливість додавання оголошень в різні категорії, а саме:

- Категорія пошук роботи: користувач може додати що він зараз знаходиться в пошуку роботи, йому потрібно вказати вік, місто та номер телефону щоб з ним могли зв’язатись роботодавці, також він може вказати в описі його можливості, та володіння спецнавичками.
- Категорія пошук працівників: користувач може створити оголошення що він шукає працівників, вказавши вік який йому потрібен для цієї роботи, місто, короткий опис кандидатів які йому потрібні, клас небезпеки та зарплатню.

- Категорія благодійні оголошення: в цій категорії додаються оголошення безоплатні(розібрати завали після прильоту ракети, допомогти пристарілим людям, зібрати fpv дрон, прибирання парків та інше). При створенні потрібно вказати вік робітників, які підходять для цієї роботи, клас небезпеки та короткий опис цієї роботи.

6. Користувач може редагувати оголошення(вік, опис, клас небезпеки).

7. Користувачі можуть видаляти свої оголошення про пошук роботи, пошук працівників та благодійні оголошення з системи.

8. Користувачі можуть переглядати всі існуючі оголошення про пошук роботи, оголошення про пошук робітників та благодійні оголошення.

9. Переглядати деталі всіх оголошень: Користувачі можуть переглядати детальну інформацію про кожне оголошення, включаючи інформацію про автора, умови, та контактну інформацію користувача.

10. Користувачі можуть фільтрувати оголошення по місцю знаходження роботи або робітників, можливість відфільтрувати по віку та класу небезпеки.

Технічні деталі

Веб-застосунок "NUMO" реалізований за допомогою фреймворку Django, що забезпечує швидку розробку та масштабованість. Для зберігання даних використовується реляційна база даних, така як SQLite. Фронтенд частину створено з використанням HTML, CSS.

2. ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ЗАСТОСУНКІВ ДЛЯ ПОШУКУ ВАКАНСІЙ ТА РОБІТНИКІВ

Для аналізу існуючих аналогів будуть взяті такі програми: OLX робота, work.ua. Це два найпопулярніші застосунки які люди використовують в Україні.

2.1. OLX робота

OLX робота - це онлайн-платформа, де користувачі можуть шукати роботу у різних сферах та галузях, розміщувати оголошення про вакансії безкоштовно, використовувати фільтри для точного пошуку, співпрацювати з рекрутинговими агентствами, забезпечити безпеку особистої інформації, залишати відгуки про роботодавців та використовувати мобільний додаток для ще більшого зручності.

Функції та можливості:

- Роботодавці можуть створювати оголошення про вакансії, детально описуючи умови праці та вимоги до кандидатів.
- Користувачі можуть шукати роботу в різних категоріях та галузях за допомогою зручного пошуку та фільтрів.
- OLX Робота дозволяє користувачам зв'язуватися з роботодавцями безпосередньо через платформу для подальшої взаємодії.

Переваги OLX робота:

- Різноманітність вакансій: OLX.ua має широкий вибір вакансій у різних галузях та регіонах, що дозволяє кандидатам знайти роботу, яка відповідає їхнім потребам та навичкам.
- Легкість пошуку: Сайт має зручний інтерфейс та фільтри, які дозволяють швидко знаходити вакансії за ключовими словами, регіонами та іншими параметрами.

- Можливість безкоштовно розміщувати резюме: Користувачі можуть безкоштовно створювати та розміщувати свої резюме на сайті, що дозволяє роботодавцям знайти їхні профілі та запропонувати вакансії.
- Відгуки та рейтинги: OLX.ua надає можливість користувачам залишати відгуки та оцінки про роботодавців, що допомагає кандидатам зробити кращий вибір під час пошуку роботи.

Недоліки OLX робота:

- Недостатня модерація: Іноді на сайті можуть з'являтися шахраї та недостовірні вакансії, оскільки OLX.ua не завжди може ефективно модерувати всі оголошення.
- Обмежені можливості фільтрації: Деякі користувачі вказують на обмежені можливості фільтрації вакансій на OLX.ua порівняно з іншими платформами.
- Необхідність уважності під час вибору вакансії: Користувачам слід бути обережними та уважними при виборі вакансії на OLX.ua, оскільки існує ризик потрапити на недостовірні пропозиції.

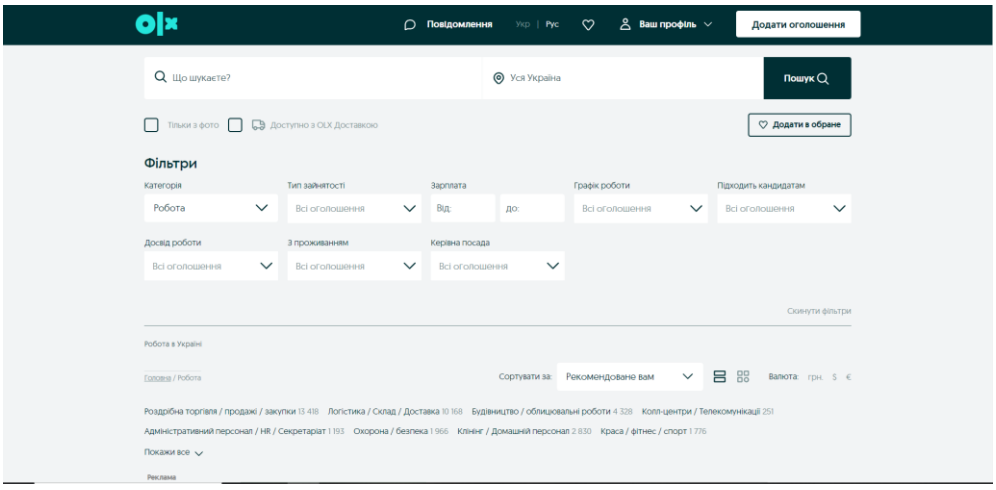


Рис. 2.1. Головний екран OLX робота

2.2. Work.ua

Work.ua – це один з найпопулярніших в Україні онлайн-порталів з пошуку роботи, де шукачі роботи можуть знайти тисячі вакансій від прямого роботодавця в різних галузях та регіонах країни.

Функції та можливості:

- Work.ua пропонує різноманітні вакансії у всіх сферах економіки, від адміністративних та клерикальних робіт до високотехнологічних та інженерних посад.
- Користувачі можуть використовувати різні фільтри для точного пошуку вакансій, включаючи місце розташування, тип зайнятості, рівень заробітної плати, вимоги до досвіду тощо.
- Поміж безлічі вакансій, користувачі можуть розмістити своє резюме, щоб залучити увагу потенційних роботодавців.

Переваги work.ua:

- Широкий вибір вакансій: Work.ua має велику базу вакансій у різних галузях та регіонах, що дозволяє користувачам знайти роботу, яка відповідає їхнім потребам та навичкам.
- Покращений пошук: Сайт пропонує розширені функції фільтрації та сортування вакансій, що дозволяє користувачам швидко знаходити вакансії за ключовими параметрами, такими як місцезнаходження, зарплата, галузь тощо.
- Професійні профілі кандидатів: Work.ua надає користувачам можливість створювати детальні профілі з інформацією про свої навички, досвід роботи та освіту, що полегшує роботодавцям знаходження потенційних кандидатів.
- Актуальність і релевантність: Платформа оновлюється регулярно, тому користувачі можуть бути впевнені, що вони отримують актуальну інформацію про вакансії та роботодавців.
- Рекомендації та підказки: Work.ua надає користувачам поради та рекомендації щодо пошуку роботи, складання резюме та проведення співбесід, що може бути корисним для кандидатів.

Недоліки work.ua:

- Платна підписка на деякі функції: Деякі функції на Work.ua, такі як перегляд контактів роботодавців, можуть бути доступні лише за платну підписку.
- Обмежені можливості безкоштовного облікового запису: Деякі функції можуть бути обмежені для користувачів з безкоштовним обліковим записом, що може обмежити їх можливості в пошуку роботи.
- Необхідність уважності при виборі вакансій: Користувачам слід бути обережними та уважними при виборі вакансій, оскільки існує ризик потрапити на недостовірні пропозиції.

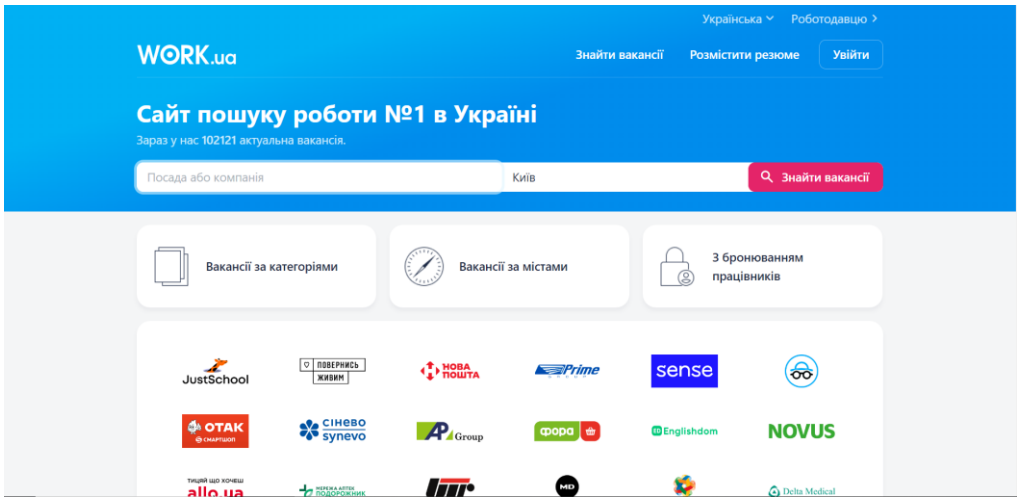


Рис 2.2. Головний екран work.ua

2.3. Таблиця порівняння існуючих аналогів

Було проведено аналіз, який спрямований на визначення переваг, недоліків та ключових можливостей під час порівняння застосунків OLX робота, work.ua та розроблений мною web-застосунок «NUMO».

Таблиця 2.2.1.

Аналіз аналогів

Застосунки / Характеристики	Work.ua	OLX робота	NUMO
Можливість знайти вакансію без спеціальних навичок	–	+	+
Можливість відфільтрувати роботу по рівню небезпеки	–	–	+
Можливість знайти роботу на годину	–	–	+
Волонтерські оголошення (безоплатні оголошення)	–	+	+

3. ІНСТРУМЕНТИ РОЗРОБКИ ДЛЯ СТВОРЕННЯ WEB-ЗАСТОСУНКУ

При розробці web-застосунку для пошуку вакансій та робітників було використано такі інструменти як, IDE PyCharm, фреймворк Django, система управління базами даних SQLite, HTML CSS, мова програмування Python, DB Browser for SQLite, GitHub.

3.1. Мова програмування Python:

Python - це потужна та популярна мова програмування, яка була розроблена Гвідо ван Россумом і вперше випущена у 1991 році. Вона відома своєю простотою в освоєнні, читабельністю коду та великою кількістю бібліотек, які роблять її дуже розширюваною та придатною для великої кількості застосувань. Нижче наведено деякі з її ключових особливостей:

Python відомий своєю простотою та легкістю в освоєнні, що робить його ідеальним вибором для початківців та досвідчених розробників. Мова має чистий та читабельний синтаксис, що нагадує англійську мову, що сприяє розробці зрозумілого коду.

Python є інтерпретованою мовою, що означає, що ви можете виконувати код без необхідності компіляції. Це робить розробку та тестування програм швидшими та більш зручними.

Мова має велику та активну спільноту розробників, що сприяє розвитку та підтримці різноманітних бібліотек та фреймворків. Наприклад, веб-розробники можуть скористатися фреймворками, такими як Django або Flask, для швидкої розробки веб-додатків, тоді як науковці можуть використовувати бібліотеки, такі як NumPy та SciPy, для наукових обчислень.

Python має широке застосування у різних галузях, включаючи веб-розробку, наукові дослідження, обробку даних, штучний інтелект, машинне навчання,

автоматизацію та багато іншого. Його універсальність та гнучкість роблять його популярним вибором серед розробників усіх рівнів.

Python має велику кількість бібліотек та фреймворків для різних цілей. Наприклад, для веб-розробки є Django та Flask, для наукових обчислень - NumPy та SciPy, для машинного навчання - TensorFlow, PyTorch та scikit-learn. Це робить Python досить потужним інструментом для розв'язання різноманітних завдань.

Python є вільним та відкритим програмним забезпеченням, що означає, що ви можете використовувати його безкоштовно, модифікувати його та сприяти його подальшому розвитку. Ця відкритість сприяє швидкому росту мови та розширенню її функціональності.

Python підтримується на багатьох операційних системах, таких як Windows, macOS, Linux та інші.

Якщо вам потрібна функціональність, якої немає у вбудованих бібліотеках Python, ви можете легко встановити сторонні бібліотеки за допомогою менеджера пакетів pip.

Що ж до можливостей, які Python надає, вони практично необмежені. Деякі з них включають:

Веб-розробка: створення веб-сайтів, веб-додатків, API за допомогою фреймворків, таких як Django та Flask.

Наукові дослідження: обробка даних, візуалізація даних, статистичний аналіз, наукові обчислення за допомогою бібліотек, таких як NumPy, SciPy та Matplotlib.

Машинне навчання та штучний інтелект: розробка моделей машинного навчання, нейронних мереж, обробка природної мови, комп'ютерний зір тощо з використанням бібліотек, таких як TensorFlow, PyTorch та scikit-learn.

Автоматизація задач: автоматизація рутинних процесів, створення скриптів для роботи з файлами, обробки тексту, роботи з API.

Розробка ігор: створення ігор за допомогою різноманітних фреймворків та бібліотек для графіки, фізики та штучного інтелекту.

Python використовується для розв'язання різноманітних завдань у різних галузях, таких як фінанси, медицина, освіта, аналіз даних та багато іншого.

В таблиці 3.1.1. переваги мови програмування Python з фреймворком Django, в порівнянні з іншими мовами програмування.

Таблиця 3.1.1

Переваги Python з Django, в порівнянні з іншими мовами програмування

Особливості / Критерії	**Python з Django**	**JavaScript з Node.js**	**Ruby з Ruby on Rails**
Експресивність	Python має простий та зрозумілий синтаксис, що полегшує розробку та зрозумілість коду. Django надає високорівневі функції, що дозволяють швидко розробляти веб-додатки.	JavaScript з Node.js має складніший синтаксис порівняно з Python, що може бути менш зрозумілим для новачків та менш експресивним.	Ruby має приємний та елегантний синтаксис, проте порівняно з Python не такий зрозумілий для більшості розробників. Ruby on Rails також має конвенції над конфігурацією, але може бути менш інтуїтивним для новачків.
Продуктивність	Python з Django надає широкий набір інструментів та бібліотек, що допомагають розробникам бути продуктивними. Django також має добре організовану документацію та спільноту, що сприяє швидкій розробці.	JavaScript з Node.js може бути менш продуктивним у розробці через асинхронний підхід та складність управління асинхронним кодом.	Ruby з Ruby on Rails, хоч і забезпечує швидку розробку завдяки готовим рішенням, проте не досягає такого рівня продуктивності як Python з Django через меншу кількість бібліотек та інструментів.

Продовження таблиці 3.1.1

Переваги Python з Django, в порівнянні з іншими мовами програмування

Особливості / Критерії	**Python з Django**	**JavaScript з Node.js**	**Ruby з Ruby on Rails**
Зручність розробки	Python з Django має чистий та лаконічний синтаксис, а також широкий набір готових бібліотек та модулів для швидкої розробки. Django також забезпечує стандартизовану структуру проекту, що полегшує розробку.	JavaScript з Node.js може бути менш зручним для розробки через асинхронний підхід та складність управління асинхронним кодом.	Ruby з Ruby on Rails, хоч і надає швидку розробку завдяки готовим рішенням, може бути менш зручним для розробників, оскільки не всі концепції та підходи є інтуїтивно зрозумілими.
Спільнота та підтримка	Python має велику та активну спільноту розробників, що надає високий рівень підтримки та допомоги. Django також має велику спільноту користувачів, що надає багато ресурсів та документації.	JavaScript також має велику спільноту розробників, а Node.js є однією з найпопулярніших технологій для серверного JavaScript.	Ruby також має активну спільноту розробників, а Ruby on Rails є одним з найпопулярніших фреймворків веб-розробки, проте спільнота не така велика, як у Python.
Безпека	Python та Django мають вбудовані засоби для забезпечення безпеки, такі як системи аутентифікації, авторизації та захисту від атак. Django також має вбудовану захист CSRF та SQL-ін'єкцій.	JavaScript та Node.js також мають різні бібліотеки та інструменти для забезпечення безпеки веб-додатків, такі як Passport.js для аутентифікації.	Ruby та Ruby on Rails також мають інструменти безпеки, але не досягають такого рівня безпеки, як Python з Django, через меншу кількість вбудованих захистів та інструментів.

Продовження таблиці 3.1.1

Переваги Python з Django, в порівнянні з іншими мовами програмування

Особливості / Критерії	**Python з Django**	**JavaScript з Node.js**	**Ruby з Ruby on Rails**
Швидкодія	Python з Django зазвичай має добру швидкодію, особливо з оптимізацією коду та використанням кешування. Django також має широкий вибір серверів WSGI, які дозволяють покращити продуктивність.	JavaScript з Node.js має високу швидкодію завдяки асинхронному підходу та використанню JavaScript на стороні сервера.	Ruby з Ruby on Rails, хоч і має досить гарну швидкодію, проте часто відстає від Python з Django та JavaScript з Node.js, особливо при обробці великих обсягів даних або високо-навантажених додатків.
Масштабованість	Python з Django дозволяє побудувати масштабні додатки завдяки можливості розподіленої розробки, горизонтальному та вертикальному масштабуванню, а також підтримці кешування та асинхронного програмування.	JavaScript з Node.js також добре масштабується завдяки асинхронному підходу та можливості розгортання на кластері серверів.	Ruby з Ruby on Rails має обмежену масштабованість порівняно з Python з Django та JavaScript з Node.js, особливо при великій кількості одночасних підключень та обробці великих обсягів даних.

Переглянувши всі плюси та мінуси, я обрав мову програмування Python для свого проекту. Це рішення виявилось оптимальним у порівнянні з іншими мовами та фреймворками.

Переглянувши всі плюси та мінуси, я обрав мову програмування Python для свого проекту. Це рішення виявилось оптимальним у порівнянні з іншими мовами та фреймворками.

3.1. Середовище розробки PyCharm

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, що розробляється компанією JetBrains. Ось деяка інформація про PyCharm:

- PyCharm надає широкий набір інструментів та функцій, які полегшують розробку Python-додатків, включаючи підтримку створення, редагування та відлагодження коду.
- PyCharm має потужний редактор коду з різноманітними функціями, такими як автодоповнення коду, перевірка синтаксису, вбудована довідка та підтримка рефакторингу коду, що сприяє збільшенню продуктивності розробників.
- PyCharm включає в себе інтегровану систему керування версіями, таку як Git, що дозволяє розробникам легко працювати зі своїми проектами у спільній робочій області.
- PyCharm має вбудовані інструменти для відлагодження коду, включаючи можливість ставити точки зупинки, спостерігати за змінними та виконувати код по кроку, що дозволяє розробникам ефективно відлагоджувати свої програми.
- PyCharm також надає підтримку віртуальних середовищ (virtual environments), що дозволяє ізолювати проекти та їх залежності для кращої управління.
- Поза цим, PyCharm має інтегровані інструменти для роботи з різними фреймворками та бібліотеками Python, такими як Django, Flask, NumPy, і багато інших, що спрощує розробку різноманітних видів програм.
- PyCharm доступний у різних варіантах, включаючи безкоштовну версію Community Edition та комерційні версії Professional Edition та Enterprise Edition, які мають додаткові функції та підтримку.

Загалом, PyCharm є потужним та інтуїтивно зрозумілим інструментом для розробки Python-додатків, що допомагає розробникам зосередитися на

творчому процесі та підвищити продуктивність роботи. В таблиці 3.2.1. зображено переваги серидовища розробки PyCharm у порівнянні з іншими серидовищами розробки.

Таблиця 3.2.1.

Переваги серидовища розробки PyCharm у порівнянні з іншими серидовищами розробки

Критерій	**PyCharm**	**Visual Studio Code (VS Code)**	**Sublime Text**
Підтримка Python	Повна підтримка, включаючи автодоповнення, рефакторинг, візуальний налагоджувач та інше	Підтримка Python з допомогою розширень, але менш повна в порівнянні з PyCharm	Підтримка Python, але менш розвинена у порівнянні з PyCharm
Інтеграція з Git	Інтегрований Git з підтримкою основних операцій	Інтегрований Git з підтримкою основних операцій, але менше можливостей у порівнянні з PyCharm	Підтримка Git, але можливо знадобиться встановлення додаткових плагінів
Відладка	Вбудований візуальний налагоджувач з підтримкою крокування по коду, відслідковування змін та багато іншого	Відсутня або обмежена підтримка відладки без розширень	Відсутня або обмежена підтримка відладки без розширень

Продовження таблиці 3.2.1

Переваги серидовища розробки PyCharm у порівнянні з іншими серидовищами розробки

Критерій	PyCharm	Visual Studio Code (VS Code)	Sublime
Підтримка віртуальних середовищ	Інтегрована підтримка віртуальних середовищ, включаючи venv та інші	Підтримка віртуальних середовищ через розширення, але може вимагати додаткових налаштувань	Підтримка віртуальних середовищ, але може бути складніше у налаштуванні
Спільнота	Активна спільнота користувачів та розробників, яка надає підтримку, поради та обмін ідеями	Активна спільнота користувачів, але менш активна у порівнянні з PyCharm	Активна спільнота користувачів, але може бути менш активна у порівнянні з PyCharm

На малюнку 3.1 зображено вікно PyCharm

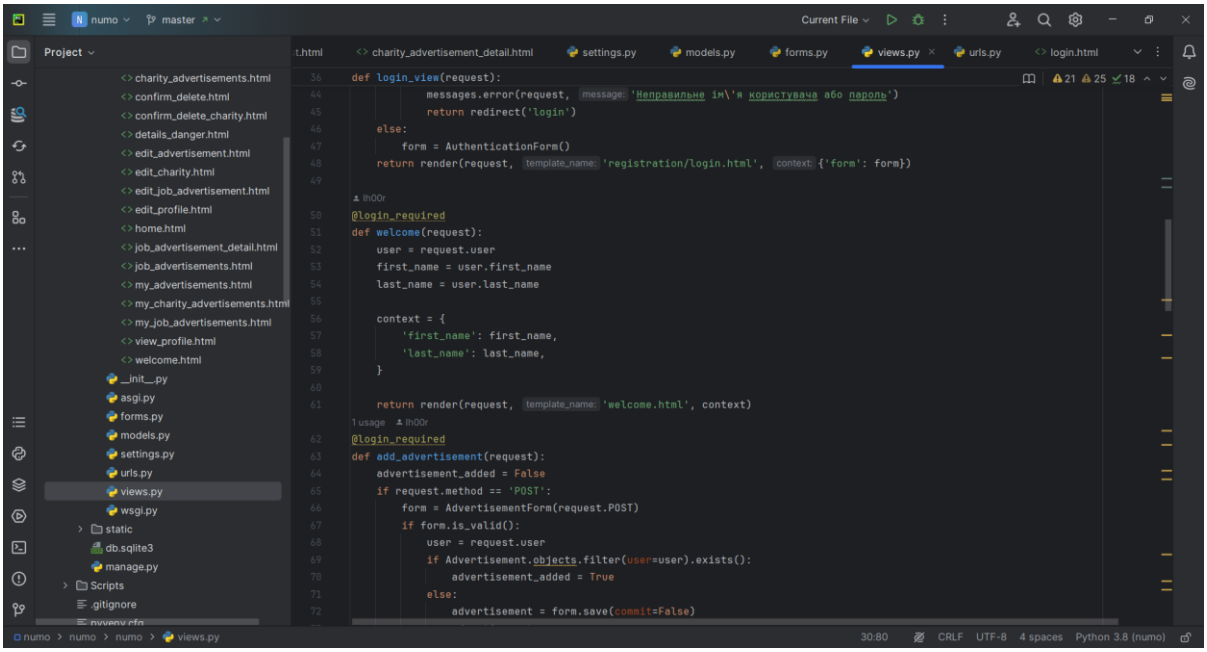


Рис. 3.1 Вікно IDE PyCharm

3.2. Фреймворк Django

Django - це високорівневий веб-фреймворк для Python, що забезпечує швидку та ефективну розробку веб-додатків. Основні особливості Django включають вбудований адміністративний інтерфейс, який автоматично створюється на основі моделей додатків; об'єктно-реляційне відображення (ORM), яке дозволяє працювати з базою даних за допомогою Python-коду; систему маршрутизації URL-адрес; потужну систему шаблонів для розділення логіки представлення та дизайну; а також вбудовані інструменти для захисту від різних видів атак, таких як атаки SQL-ін'єкції та атаки на міжсайтову поділку міжсайтову (CSRF).

Поза цим, Django має вбудовану підтримку міжнародизації, що дозволяє розробникам легко перекладати веб-сайти на різні мови для різних аудиторій. Завдяки активній та розширеній спільноті розробників, ви зможете знайти безліч ресурсів, таких як документація, пакети розширень та підтримка, що допоможе вам вирішити будь-які проблеми або питання, що виникають під час розробки.

Додатково, Django є вільним та відкритим програмним забезпеченням, що означає, що ви можете використовувати його безкоштовно, модифікувати його згідно з вашими потребами та сприяти його подальшому розвитку.

Розширюваність Django додатково підтверджується за допомогою пакетів, таких як Django REST Framework та Django Channels, які дозволяють легко розширити функціональність фреймворка для розробки різних типів додатків, включаючи API та чати в реальному часі.

При створенні web-застосунку було використано такі пакети Django:

django.shortcuts – пакет django.shortcuts містить набір зручних функцій, які допомагають скоротити код та зробити його більш читабельним.

django.contrib.auth – пакет django.contrib.auth забезпечує потужну систему аутентифікації та авторизації користувачів:

- authenticate: Перевірка облікових даних користувача.
- login: Авторизація користувача в системі.
- logout: Вихід користувача із системи.
- get_user: Отримання поточного користувача.
- get_user_model: Отримання моделі користувача, яка використовується в проекті.
- update_session_auth_hash: Оновлення сесії користувача після зміни його пароля.

django.contrib.auth.forms – пакет django.contrib.auth.forms містить готові форми для аутентифікації та управління користувачами:

- UserCreationForm: Форма для реєстрації нового користувача.
- UserChangeForm: Форма для редагування інформації про користувача.
- AuthenticationForm: Форма для аутентифікації користувача (вхід у систему).
- PasswordChangeForm: Форма для зміни пароля користувача.
- PasswordResetForm: Форма для відновлення пароля через електронну пошту.

django.contrib.auth.decorators – пакет django.contrib.auth.decorators містить декоратори для обмеження доступу до в'юх:

- login_required: Декоратор, який обмежує доступ до в'юхи лише для аутентифікованих користувачів. Якщо користувач не аутентифікований, він буде перенаправлений на сторінку входу.
- permission_required: Декоратор, який перевіряє наявність у користувача необхідних дозволів для доступу до в'юхи.

- user_passes_test: Декоратор, який перевіряє, чи задовольняє користувач певну умову.

django.contrib.auth.models – пакет django.contrib.auth.models містить моделі для роботи з користувачами та групами користувачів:

- User: Основна модель користувача, яка містить поля для зберігання імені користувача, пароля, електронної пошти та іншої інформації.
- Group: Модель групи користувачів, яка дозволяє об'єднувати користувачів у групи з певними правами.
- Permission: Модель дозволу, яка визначає права доступу користувачів та груп до різних ресурсів.
- AbstractBaseUser: Абстрактна базова модель користувача, яку можна використовувати для створення власної кастомної моделі користувача.
- BaseUserManager: Базовий клас менеджера користувачів, який можна використовувати для створення кастомних менеджерів користувачів.

Ці модулі та підмодулі забезпечують широкий спектр функціоналу для розробки веб-додатків з використанням Django, дозволяючи легко реалізовувати аутентифікацію, авторизацію, управління користувачами, а також створювати та обробляти різні форми. На рисунку 3.2 зображений логотип Django.

Рис. 3.2 Логотип Django

3.3. Система управління базами даних SQLite

SQLite - це високоефективна та надійна система управління базами даних, яка широко використовується для зберігання та обробки даних в різноманітних додатках та середовищах. Однією з ключових особливостей SQLite є те, що вона є вбудовувано. СУБД, тобто весь її код знаходиться в одному файлі бібліотеки, що робить її легкою у використанні та інтеграції в додатки.

SQLite використовується в різних областях, включаючи мобільні додатки, веб-сайти, настільні програми, системи вбудованих пристроїв та багато іншого. Багато популярних програмних продуктів, таких як браузери, операційні системи та інші, використовують SQLite для зберігання та обробки даних користувачів.

Однією з переваг SQLite є його висока продуктивність та ефективність роботи з базами даних. Вона працює дуже швидко, навіть з великими обсягами даних, та використовує мало системних ресурсів, що робить її ідеальним вибором для додатків з обмеженими ресурсами.

Переваги SQLite:

- Простота розгортання: завдяки простоті та самодостатності SQLite легко інтегрувати в будь-який проект. Це робить її чудовим вибором для розробників, які хочуть швидко та без проблем додати локальну базу даних до своїх програм.
- Зниження навантажень на сервер: SQLite може використовуватися для локального зберігання та обробки даних, що зменшує навантаження на сервер та покращує продуктивність веб-сайтів та програм. Це робить її цінним інструментом для веб-розробників, які прагнуть оптимізувати свої веб-сайти.
- Мобільність: SQLite ідеально підходить для мобільних додатків, адже її невеликий розмір та портативність не впливають на продуктивність пристроїв. Це робить її чудовим вибором для розробників мобільних додатків, які хочуть створити легкі та швидкі програми.

- Надійність: SQLite має стійку до збоїв репутацію, що робить її надійним вибором для критичних даних. Розробники можуть бути впевнені, що дані їхніх користувачів будуть у безпеці та збережуться навіть у разі збоїв.
- Безкоштовність та відкритий код: SQLite доступна безкоштовно та розповсюджується під ліцензією з відкритим кодом, що робить її доступною для будь-якого використання. Це робить її привабливим вибором для розробників з обмеженим бюджетом або тих, хто хоче мати більше контролю над своїм кодом.

Використання SQLite

SQLite можна використовувати різними способами, включаючи:

- Вбудовування в програмне забезпечення: SQLite можна вбудовувати в будь-яку програму, що потребує локального зберігання та керування даними. Це може бути корисно для таких програм, як календарі, блокноти, списки справ та ігри.
- Розробка веб-сайтів: SQLite можна використовувати для зберігання даних користувачів, кешування контенту та інших цілей у веб-сайтах. Це робить її цінним інструментом для веб-розробників, які хочуть створити динамічні та інтерактивні веб-сайти.
- Аналітика даних: SQLite можна використовувати для зберігання та аналізу даних з різних джерел. Це робить її корисним інструментом для дослідників даних та аналітиків, які хочуть вивчати та розуміти складні набори даних.
- Прототипування: SQLite можна використовувати для швидкого прототипування баз даних перед розгортанням більш масштабних рішень.
- Це економить час та ресурси розробників, дозволяючи їм тестувати та вдосконалювати свої ідеї перед тим, як вкладати значні кошти у розробку.

SQLite - це потужна, гнучка та проста у використанні база даних, яка ідеально підходить для широкого спектру завдань. Її легка вага, самодостатність, SQL-сумісність та мультиплатформність роблять її популярним вибором для

розробників програмного забезпечення та веб-сайтів, які потребують локального зберігання та керування даними.

3.4. DB Browser for SQLite

Це безкоштовний інструмент з відкритим вихідним кодом, який використовується для управління базами даних SQLite через графічний інтерфейс. Він дозволяє користувачам легко створювати, переглядати та редагувати бази даних SQLite без необхідності писати SQL-запити вручну. Ось детальний опис функцій та можливостей цієї програми:

Графічний інтерфейс користувача (GUI):

- Інтуїтивно зрозумілий інтерфейс дозволяє користувачам легко виконувати операції з базами даних.
- Зручне меню та панель інструментів для доступу до основних функцій.

Створення та управління базами даних:

- Створення нових баз даних SQLite з нуля.
- Відкриття та збереження існуючих баз даних.

Перегляд та редагування таблиць:

- Візуальний перегляд структури баз даних та даних, що зберігаються в таблицях.
- Легке редагування вмісту таблиць: додавання, видалення та оновлення записів.
- Можливість копіювання та вставки даних між таблицями та іншими програмами.

Створення та редагування структур баз даних:

- Створення нових таблиць, індексів, тригерів та переглядів.
- Редагування існуючих структур баз даних.

Імпорт та експорт даних:

- Імпорт даних з CSV-файлів, текстових файлів та інших форматів.
- Експорт даних до CSV-файлів, SQL-скриптів та інших форматів для подальшого використання або резервного копіювання.

SQL-запити:

- Виконання SQL-запитів вручну через вбудований редактор запитів.
- Перегляд результатів запитів у табличній формі.

Робота з транзакціями:

- Підтримка транзакцій для забезпечення цілісності даних.
- Можливість відкочування змін у разі помилок.

Перегляд та аналіз баз даних:

- Перегляд статистичних даних про таблиці та індекси.
- Аналіз продуктивності та оптимізація запитів.

Додаткові можливості

- Розширюваність: Підтримка розширень та плагінів для додавання нових функціональних можливостей.
- Підтримка декількох мов: Інтерфейс програми доступний на багатьох мовах, що робить її зручною для користувачів з різних країн.
- Відкрите вихідне кодування: Вихідний код програми доступний на GitHub, що дозволяє розробникам вносити свої зміни та пропонувати покращення.

Використання DB Browser for SQLite

DB Browser for SQLite може бути корисним для різних категорій користувачів:

- Розробники: Для швидкого створення прототипів баз даних та тестування SQL-запитів.
- Біоінформатики: Для аналізу великих наборів даних, збережених у форматі SQLite.
- Адміністратори баз даних: Для управління базами даних, резервного копіювання та відновлення даних.
- Аналітики даних: Для імпорту, експорту та аналізу даних у зручному графічному інтерфейсі.

DB Browser for SQLite – це потужний інструмент, який поєднує простоту використання з широкими можливостями для управління базами даних SQLite, що робить його ідеальним вибором для різних завдань, пов'язаних з базами даних.

На рисунку 3.3 зображено вікно програми DB Browser for SQLite.

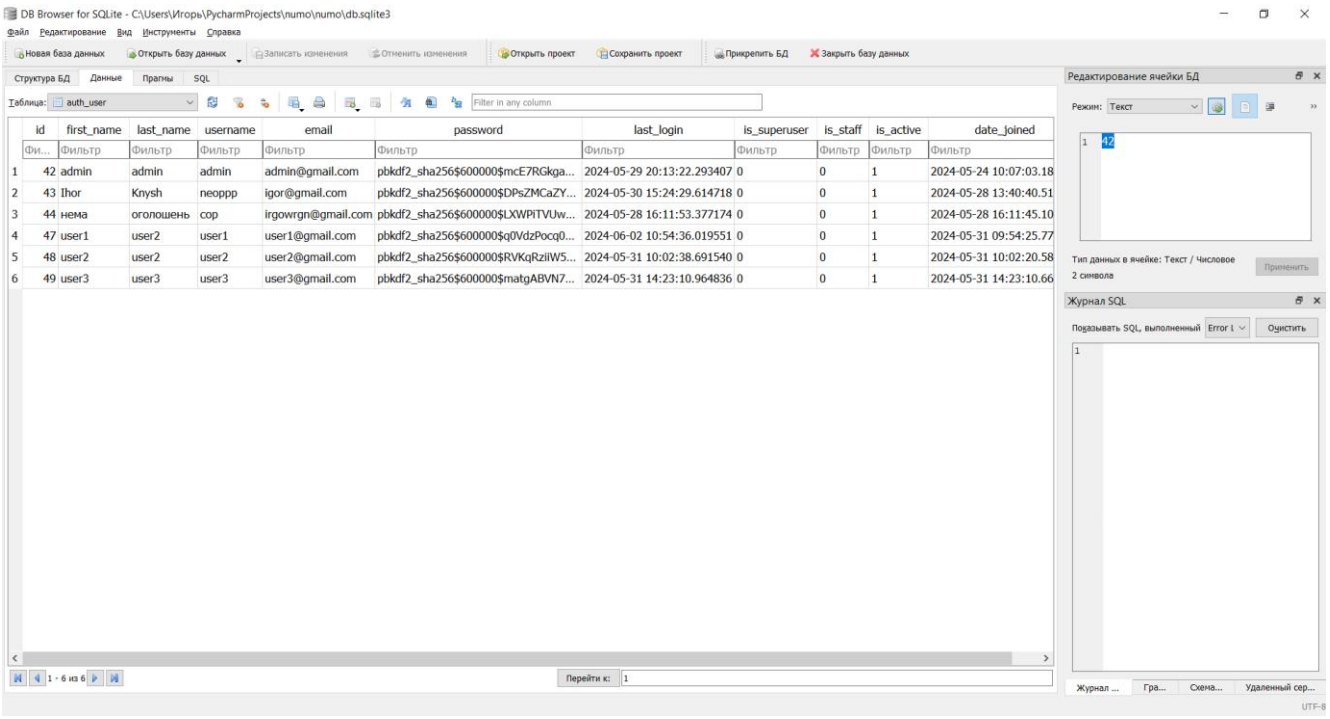


Рис. 3.3 Вікно DB Browser for SQLite

3.5. HTML та CSS

HTML (HyperText Markup Language) та CSS (Cascading Style Sheets) - це два фундаментальні блоки, з яких будується кожен веб-сайт.

HTML - це мова розмітки, яка використовується для визначення структури та вмісту веб-сторінки. Її елементи, подібні до тегів, описують заголовки, параграфи, зображення, списки та інші компоненти, з яких складається сторінка. Завдяки HTML браузер розуміє, як інтерпретувати та візуалізувати контент.

CSS - це мова стилів, яка використовується для форматування та візуального оформлення веб-сторінки. Вона додає колір, шрифти, розміри, розташування та інші стильові аспекти до елементів HTML, роблячи веб-сайти привабливими та зручними для користувачів. CSS надає гнучкість та контроль над зовнішнім виглядом веб-сторінок, дозволяючи створювати унікальні та динамічні дизайни. Разом HTML та CSS дають змогу:

- Створювати структуровані та змістовні веб-сторінки.
- Форматувати текст, зображення та інші елементи.
- Налаштувати колірну гаму, шрифти та розбивку.
- Розробляти адаптивні макети для різних пристроїв.
- Створювати інтерактивні та динамічні веб-сайти.

3.6. GitHub

GitHub — це платформа для хостингу коду та спільної розробки програмного забезпечення, яка використовує систему контролю версій Git. GitHub надає інструменти для управління версіями коду, спільної роботи над проектами, відстеження помилок та запитів на додавання функціональності, а також для автоматизації процесів розробки. GitHub в цьому проекті використовувався для збереження попередніх версій коду з можливістю відновлення, а також для відстеження невиконаних задач.

Основні функції GitHub:

1. Репозиторій (або repo) – це місце, де зберігається код вашого проекту. Репозиторії можуть бути публічними або приватними, залежно від ваших потреб щодо доступу до коду.

2. Система контролю версій Git –це розподілена система контролю версій, яка дозволяє відстежувати зміни у файлах і координувати роботу декількох розробників.

- Коміти (Commits): це окремі зміни у файлах, що зберігаються в репозиторії. Кожен коміт містить опис змін, що було зроблено.
- Гілки (Branches): дозволяють розробникам працювати над різними функціями або виправленнями незалежно один від одного. Основна гілка зазвичай називається main або master.
- Мерджі (Merges): процес об'єднання змін з однієї гілки в іншу. Це дозволяє інтегрувати нові функції або виправлення помилок в основну гілку проекту.

3. Pull Requests – це запит на злиття змін з однієї гілки в іншу. Це один з основних способів співпраці в GitHub.

- Код-рев'ю: інші розробники можуть переглядати зміни, залишати коментарі та пропозиції щодо покращення коду.
- Тести: автоматичні тести можуть бути запущені під час створення PR, що дозволяє перевірити, чи не викликають зміни нових помилок.
- Обговорення: розробники можуть обговорювати запропоновані зміни, ставити питання та давати рекомендації.

4. Issues — це інструмент для відстеження помилок, запитів на нові функції та інших задач, що виникають під час розробки проекту.

- Створення та призначення задач: можна створювати нові задачі, описувати їх та призначати відповідальних.
- Мітки (Labels): використовуються для категоризації задач, наприклад, bug, enhancement, question.

- Майлстоуни (Milestones): дозволяють групувати задачі за певними етапами проекту.

5. GitHub Actions — це платформа для автоматизації робочих процесів безпосередньо в репозиторіях GitHub.

- CI/CD (Continuous Integration/Continuous Deployment): можна налаштувати автоматичне тестування та розгортання коду при кожному коміті або pull request.
- Автоматизація задач: автоматизація рутинних задач, таких як оновлення документації, відправка повідомлень у Slack тощо.

6. GitHub Projects — це інструмент для управління завданнями та проектами.

- Канбан-дошки: можна створювати дошки для відстеження стану задач у вигляді колонок (To do, In progress, Done).
- Картки задач: задачі з issues можна перетягувати між колонками, що дозволяє легко відстежувати їх стан.
- Інтеграція з issues та pull requests: задачі з issues та pull requests можна безпосередньо пов'язувати з картками на дошці проекту.

4. РОЗРОБКА WEB-ДОДАТКУ ДЛЯ ПОШУКУ ВАКАНСІЙ ТА ПРАЦІВНИКІВ ДЛЯ ФІЗИЧНОЇ ПРАЦІ

4.1. Основні функції додатку

Основна мета дипломної роботи полягала у розробці веб-застосунку, спрямованого на сприяння пошуку роботи для користувачів, а також на знаходження потрібних працівників. Додатковою функцією була можливість створення благодійних оголошень безоплатно.

Створення такого веб-застосунку відображає актуальність проблеми безробіття та недостатньої інформованості про можливості зайнятості серед широкого кола користувачів. Він мав на меті полегшити процес пошуку роботи для безробітних, а також допомогти роботодавцям знайти кваліфікованих працівників шляхом зручного та ефективного інструменту.

У веб-застосунку повинні бути такі ключові функції:

- **Пошук працівників:** Користувачі повинні мати змогу шукати працівників з легкістю, все що потрібно зробити людині яка хоче знайти працівників, це вказати критерії які потрібні від робітників, вік та клас небезпеки. Це повинно спрощувати процес та швидкість найму персоналу.
- **Розміщення оголошень про пошук роботи:** Користувачі можуть створювати оголошення що вони шукають роботу, вони повинні вказати опис того що вони уміють та спеціальні навички якщо вони ними володіють, скільки років та місто проживання. Це спростить пошук робітників для роботодавців.
- **Благодійні оголошення:** Створення можливості для користувачів створювати благодійні оголошення в онлайн середовищі має безліч переваг. По-перше, це дає змогу залучити увагу до соціально важливих питань, що потребують негайного вирішення. Часто громадськість не має достатньої інформації про

проблеми, які виникають у суспільстві, і благодійні оголошення можуть стати ефективним інструментом у розповсюдженні цієї інформації.

По-друге, створення благодійних оголошень сприяє підтримці та розвитку благодійних ініціатив. Кожен, хто має бажання допомогти у розв'язанні певної проблеми чи підтримати конкретну благодійну програму, може легко знайти і приєднатися до відповідного оголошення. Це не лише створює можливість для людей зробити добру справу, але й сприяє збору необхідних ресурсів для реалізації благодійних проектів.

Також в web-застосунок було додано класи безпеки роботи. Вони потрібні щоб корисувачі відразу чітко розуміли яка робота їх чекає, та щоб вони могли відфільтрувати оголошення по класу небезпеки для більш зручного та швидкого пошуку роботи.

Було додано 4 класи безпеки(A, B, C, D):

Клас A (Низький ризик)

Опис: Роботи з найнижчим рівнем небезпеки, що не передбачають значних фізичних або шкідливих впливів на працівників.

Приклади:

- Моніторинг і контроль за процесами за допомогою автоматизованих систем.
- Легка фізична праця в контрольованих умовах.
- Роботи в лабораторіях без використання небезпечних хімічних речовин.

Клас B (Помірний ризик) Опис: Роботи, які мають помірний рівень небезпеки і можуть включати фізичні навантаження або роботу в умовах, що потребують певної обережності

Приклади:

- Роботи на виробництві з використанням безпечних матеріалів та обладнання.
- Роботи на відкритому повітрі в нормальних кліматичних умовах.
- Роботи, що передбачають незначний контакт з хімічними речовинами (наприклад, фарбування).

Клас C (Високий ризик) Опис: Роботи з високим рівнем небезпеки, що потребують спеціальних заходів безпеки та використання засобів індивідуального захисту.

Приклади:

- Роботи на висоті (будівництво, монтаж).
- Роботи з електрообладнанням під високою напругою.
- Роботи в умовах підвищеної температури або шуму (металургійні заводи, кування).

Клас D (Найвищий ризик)

Опис: Найнебезпечніші роботи, які пов'язані з високим ризиком для життя та здоров'я працівників і вимагають суворого дотримання норм безпеки та використання спеціальних засобів захисту.

Приклади:

- Роботи з вибуховими речовинами.
- Роботи в замкнених просторах з обмеженим доступом кисню (суднобудування, підземні роботи).
- Роботи з радіоактивними матеріалами.
- Роботи в екстремальних умовах (рятувальні операції, гасіння пожеж).

Для візуалізації послідовності дій була створена діагама використання яка зображена нижче.

Рис. 4.1 Діаграма використання

Для візуалізації зв’язків між різними таблицями в базі даних була створена діаграма структури бази даних, вона зображена на рисунку 4.2

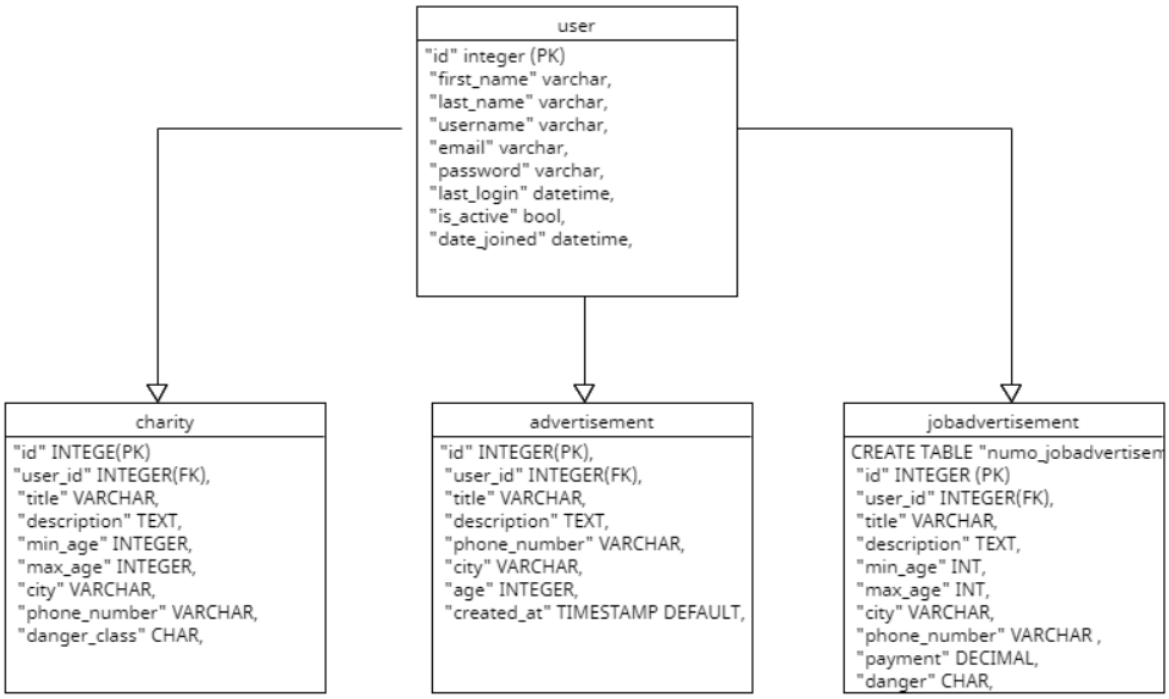
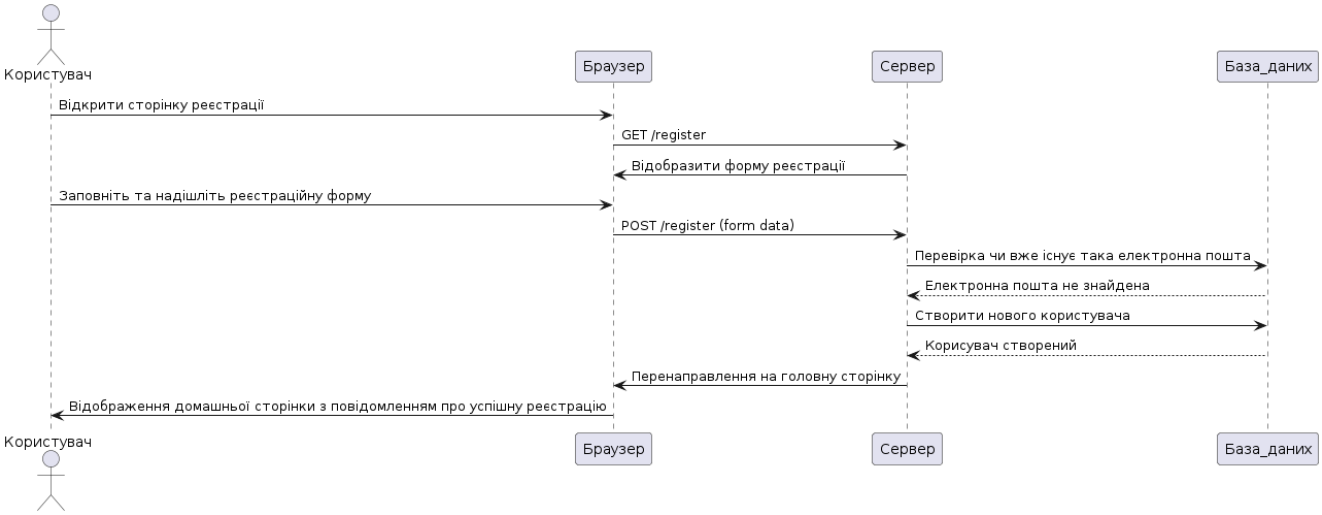
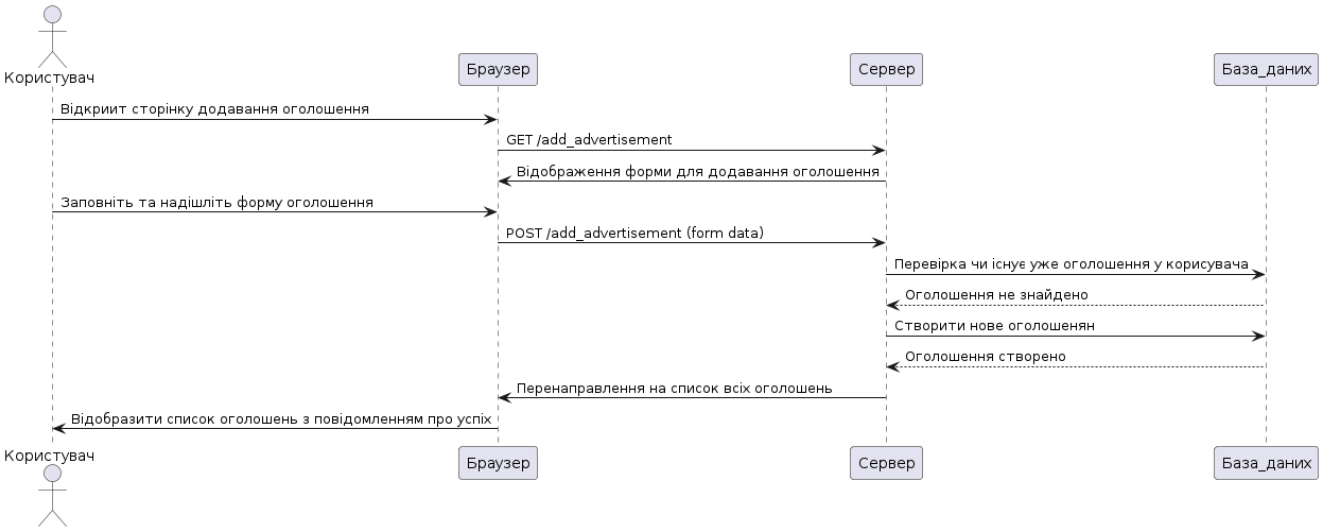


Рис. 4.2 Діаграма структури бази даних

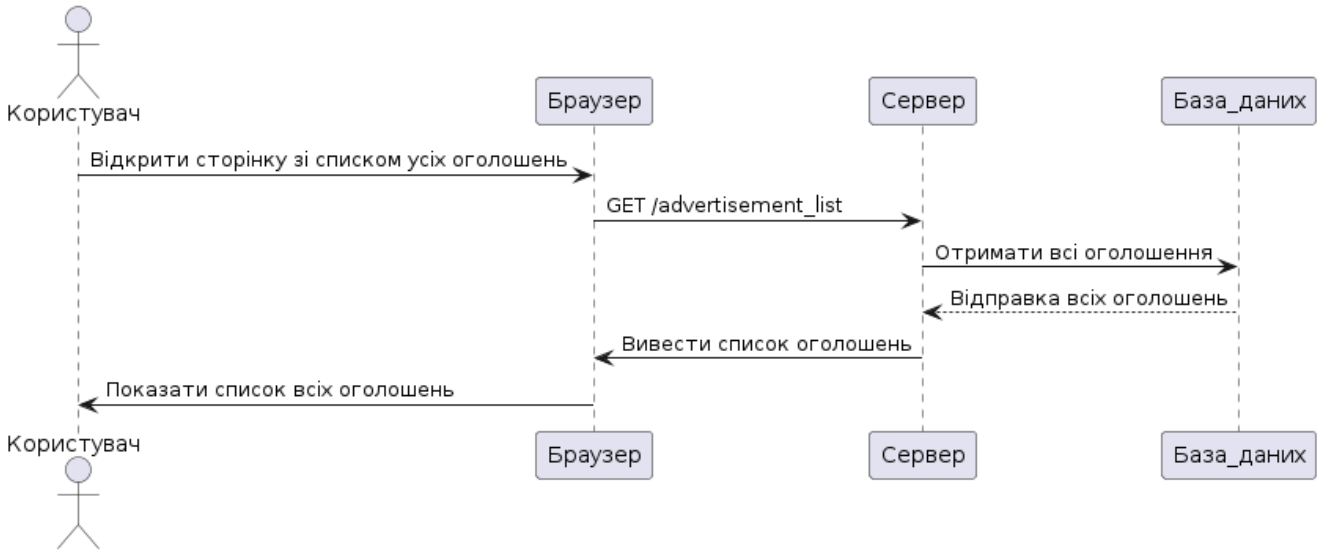
Також були створені діаграми послідовності для реєстрації(рис. 4.3), створення оголошення(рис. 4.4) та діаграма послідовності регеляду всіх оголошень(рис. 4.5). Вони використовуються для моделювання взаємодії між об’єктами в системі в конктретний момент часу, а також для аналізу, проектування та тестування програмного забезпечення.



4.3 Діаграма послідовності реєстрації



4.4 Діаграма послідовності створення оголошення



4.5 Діаграма послідовності перегляду всіх оголошень

4.2. Реалізація функціональності за допомогою коду

Для зберігання інформації в базі даних було створено 3 моделі моделі Django.

Модель Advertisement

```
class Advertisement(models.Model):
 user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    title = models.CharField(max_length=100)
  description = models.TextField()
  phone_number = models.CharField(max_length=20)
  city = models.CharField(max_length=50)
  age = models.PositiveIntegerField()
  id = models.AutoField(primary_key=True)
```

user: Одно-до-одного зв'язок з користувачем, який створив оголошення. Якщо користувач видаляє профіль, то відповідне оголошення також видаляється.

title: Заголовок оголошення, обмежений 100 символами.

description: Текстовий опис оголошення.

phone_number: Номер телефону, обмежений 20 символами.

city: Місто, де розміщено оголошення, обмежене 50 символами.

age: Вік, вказаний в оголошенні, обмежений позитивними цілими числами.

id: Первинний ключ (ідентифікатор) оголошення, який автоматично збільшується.

Також є ще 2 моделі під назвою JobAdvertisement та Charity, різниця з попереднью моделлю в тому що до моделі JobAdvertisement та Charity додано поле з варіантами вибору класу небезпеки.

```
danger_class = models.CharField(max_length=1, choices=[ ('A', 'A'), ('B', 'B'), ('C', 'C'),
('D', 'D')])
```

Для створення форм для різних типів оголошень було створено три класи.

class AdvertisementForm(forms.ModelForm):

class Meta:

model = Advertisement

fields = ['title', 'description', 'phone_number', 'city', 'age']

Цей клас розроблений для того щоб створювати форму для додавання оголошення про пошук роботи.

Другий клас для створення форми для додавання оголошення про пошук робітників.

class JobAdvertisementForm(forms.ModelForm):

class Meta:

model = JobAdvertisement

fields = ['title', 'description', 'min_age', 'max_age', 'city', 'phone_number', 'payment', 'danger']

Також є ще один клас, CharityForm, який дуже схожий на JobAdvertisementForm, за винятком відсутності поля 'payment'. Цей клас використовується для створення форми благодійних оголошень.

Для створення нового користувача був створений клас CustomUserCreationForm.

class CustomUserCreationForm(UserCreationForm):

first_name = forms.CharField(label="Ім'я", max_length=30, required=False)

last_name = forms.CharField(label="Фамілія", max_length=30, required=False)

email = forms.EmailField(max_length=254)

class Meta:

model = User

fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2')

labels = {'username': 'Нік користувача'}

Я розширив стандартну форму створення користувача, додавши поля для імені, призвіща та електронної пошти.

Опис класу AdvertisementFilterForm

Клас AdvertisementFilterForm є формою для фільтрації оголошень за кількома критеріями, включаючи вік, місто, клас небезпеки та мінімальну оплату. Він не прив'язаний безпосередньо до жодної моделі, а створений на основі forms.Form, що дозволяє задавати довільні поля та логіку.

```
class AdvertisementFilterForm(forms.Form):
age = forms.IntegerField(required=False, label='Вік')
min_age = forms.IntegerField(required=False, label='Мінімальний вік')
max_age = forms.IntegerField(required=False, label='Максимальний вік')
city = forms.ChoiceField(required=False, label='Місто')
danger = forms.ChoiceField(required=False, choices=[('', 'Всі'), ('A', 'A'), ('B', 'B'), ('C', 'C'), ('D', 'D')], label='Клас небезпеки')
min_payment = forms.DecimalField(required=False, label='Мінімальна оплата', min_value=0)
danger_class = forms.ChoiceField(
required=False,
choices=[('', 'Всі')] + [(choice, choice) for choice in ['A', 'B', 'C', 'D']],
label='Клас небезпеки')

def __init__(self, *args, **kwargs):
cities = kwargs.pop('cities', [])
super().__init__(*args, **kwargs)
self.fields['city'].choices = [('', 'Всі')] + [(city, city) for city in sorted(cities)]
```

Цей клас дозволяє створювати форму для фільтрації оголошень за кількома критеріями, зокрема за віком, містом, класом небезпеки та мінімальною оплатою. Динамічне заповнення списку міст забезпечує гнучкість при відображенні форми на сторінці, дозволяючи оновлювати список доступних міст без зміни коду форми.

Опис функції register_view, яка відповідає за обробку запитів на реєстрацію користувачів

```
def register_view(request):
if request.method == 'POST':
form = CustomUserCreationForm(request.POST)
if form.is_valid():
email = form.cleaned_data.get('email')
if User.objects.filter(email=email).exists():
messages.error(request, 'Ця електронна пошта вже зареєстрована.')
else:
password1 = form.cleaned_data.get('password1')
password2 = form.cleaned_data.get('password2')
if password1 != password2:
messages.error(request, 'Паролі не збігаються.')
else:
user = form.save()
login(request, user)
return redirect('welcome')
else:
form = CustomUserCreationForm()
return render(request, 'registration/register.html', {'form': form})
```

Ця функція виконує наступні дії:

- Перевіряє, чи метод запиту є POST.
- Якщо так, створює екземпляр форми CustomUserCreationForm із даними, що були відправлені методом POST.
- Перевіряє валідність даних у формі.
- Перевіряє, чи вказана електронна пошта вже існує у системі. Якщо так, відправляє повідомлення про помилку.
- Перевіряє, чи обидва введені паролі співпадають. Якщо ні, відправляє повідомлення про помилку.

- Якщо всі дані є валідними і користувача можна зареєструвати, то зберігає дані користувача, автоматично входить у систему і перенаправляє користувача на домашню сторінку.
- Якщо метод запиту не є POST, створює пустий екземпляр форми CustomUserCreationForm.
- Повертає шаблон сторінки реєстрації, передаючи йому створений екземпляр форми.

Опис функції add_advertisement, яка відповідає за створення оголошень про пошук роботи

```
def add_advertisement(request):
advertisement_added = False
if request.method == 'POST':
form = AdvertisementForm(request.POST)
if form.is_valid():
user = request.user
if Advertisement.objects.filter(user=user).exists():
advertisement_added = True
else:
advertisement = form.save(commit=False)
advertisement.user = user
advertisement.save()
return redirect('advertisement_list')
else:
form = AdvertisementForm()
return render(request, 'add_advertisement.html', {'form': form, 'advertisement_added': advertisement_added})
```

Ця функція виконує наступні дії:

- Ініціалізує змінну advertisement_added як False.
- Перевіряє, чи метод запиту є POST.
- Якщо так, створює екземпляр форми AdvertisementForm із даними, що були відправлені методом POST.
- Перевіряє валідність даних у формі.
- Отримує поточного користувача.
- Перевіряє, чи користувач вже додавав оголошення. Якщо так, встановлює advertisement_added як True.
- Якщо оголошення ще не було додано, зберігає дані форми для оголошення, прив'язуючи його до поточного користувача, і перенаправляє користувача на сторінку зі списком оголошень.
- Якщо метод запиту не є POST, створює пустий екземпляр форми AdvertisementForm.
- Повертає шаблон сторінки для додавання оголошення, передаючи йому створений екземпляр форми та змінну advertisement_added.

Опис функції edit_profile, яка відповідає за редагування профілю

```
def edit_profile(request):
user = request.user
if request.method == 'POST':
form = ProfileEditForm(request.POST, request.FILES, instance=user)
if form.is_valid():
username = form.cleaned_data['username']
email = form.cleaned_data['email']
if User.objects.exclude(pk=user.pk).filter(username=username).exists():
messages.error(request, 'Цей нік користувача вже використовується.')
elif User.objects.exclude(pk=user.pk).filter(email=email).exists():
messages.error(request, 'Користувач з такою електронною поштою вже існує.')
else:
form.save()
return redirect('welcome')
else:
form = ProfileEditForm(instance=user)
```

return render(request, 'edit_profile.html', {'form': form})

Функція виконує настуані дії:

- Отримує поточного користувача.
- Перевіряє, чи метод запиту є POST.
- Якщо так, створює екземпляр форми ProfileEditForm з даними, що були відправлені методом POST, але без зображення, оскільки воно не було вказане. Цей екземпляр форми використовується для редагування профілю користувача.
- Перевіряє валідність даних у формі.
- Отримує новий нік користувача та нову електронну пошту.
- Перевіряє, чи інші користувачі не використовують вже введений нік або електронну пошту. Якщо так, відправляє повідомлення про помилку.
- Якщо дані є валідними і унікальними, зберігає дані профілю користувача і перенаправляє користувача на вітальну сторінку.
- Якщо метод запиту не є POST, створює пустий екземпляр форми ProfileEditForm для відображення інформації профілю користувача.
- Повертає шаблон сторінки для редагування профілю, передаючи йому створений екземпляр форми.

Опис функції delete_advertisement_view, вона відповідає за видалення оголошень про пошук роботи.

def delete_advertisement_view(request, advertisement_id):
advertisement = get_object_or_404(Advertisement, id=advertisement_id)
if request.method == 'POST':
advertisement.delete()
return redirect('my_advertisements')
return render(request, 'home.html', {'advertisement': advertisement})

Ця функція виконує наступні дії:

- Отримує поточного користувача, який зараз залогінений, за допомогою request.user.
- Перевіряє, чи метод запиту є POST. Це означає, що користувач відправив форму для редагування профілю.
- Якщо метод запиту є POST, створює екземпляр форми ProfileEditForm з даними, що були відправлені методом POST. Використовується instance=user, щоб форма знала, який користувач редагується.
- Перевіряє валідність даних у формі за допомогою методу is_valid(). Це означає, що всі поля форми повинні відповідати визначеним правилам валідації.
- Отримує нові значення username та email з форми.
- Перевіряє, чи існує інший користувач з таким самим username, виключаючи поточного користувача за допомогою exclude(pk=user.pk). Якщо такий користувач існує, відправляє повідомлення про помилку.
- Аналогічно перевіряє унікальність email.
- Якщо дані є валідними і унікальними, зберігає дані профілю користувача за допомогою form.save().
- Після збереження перенаправляє користувача на вітальну сторінку (redirect('welcome')).
- Якщо метод запиту не є POST, створює екземпляр форми ProfileEditForm з даними поточного користувача (instance=user). Це дозволяє відобразити поточні дані користувача у формі для редагування.
- Повертає шаблон сторінки для редагування профілю (edit_profile.html), передаючи в нього створений екземпляр форми для відображення у веб-інтерфейсі.

Опис функції job_advertisements_view, ця функція відповідає за відображення та фільтрацію оголошень про пошук робітників.

```
def job_advertisements_view(request):
job_advertisements = JobAdvertisement.objects.all()
```

```
if request.method == 'GET':
form = AdvertisementFilterForm(request.GET,
cities=JobAdvertisement.objects.values_list('city', flat=True).distinct())
if form.is_valid():
age = form.cleaned_data.get('age')
city = form.cleaned_data.get('city')
danger = form.cleaned_data.get('danger')
min_payment = form.cleaned_data.get('min_payment')
if age is not None:
job_advertisements=job_advertisements.filter(min_age__lte=age,
max_age__gte=age)
if city:
job_advertisements = job_advertisements.filter(city=city)
if danger:
job_advertisements = job_advertisements.filter(danger=danger)
if min_payment is not None:
job_advertisements = job_advertisements.filter(payment__gte=min_payment)
else:
form =
AdvertisementFilterForm(cities=JobAdvertisement.objects.values_list('city',
flat=True).distinct())
return render(request, 'job_advertisements.html', {'form': form, 'job_advertisements':
job_advertisements})
```

Ця функція виконує наступні дії:

- Спочатку отримуються всі записи з моделі JobAdvertisement за допомогою JobAdvertisement.objects.all().

- Якщо метод запиту є GET, це означає, що користувач хоче переглянути або відфільтрувати оголошення про роботу.
- Створюється екземпляр форми AdvertisementFilterForm з даними, що були відправлені методом GET. Також передаються унікальні значення міст, що зберігаються в оголошеннях про роботу (cities=JobAdvertisement.objects.values_list('city', flat=True).distinct()).
- Перевіряється валідність даних у формі за допомогою методу is_valid(). Якщо форма є валідною, витягуються значення з форми (age, city, danger, min_payment).
- Якщо в формі вказаний вік (age), виконується фільтрація оголошень за віком, з врахуванням мінімального та максимального віку (min_age__lte=age, max_age__gte=age).
- Якщо вказано місто (city), виконується фільтрація за містом (city=city).
- Якщо вказаний клас небезпеки (danger), виконується фільтрація за класом небезпеки (danger=danger).
- Якщо вказано мінімальну оплату (min_payment), виконується фільтрація за оплатою (payment__gte=min_payment).
- Якщо метод запиту не є GET, створюється порожня форма AdvertisementFilterForm з унікальними значеннями міст для відображення в інтерфейсі користувача.
- Повертається шаблон сторінки job_advertisements.html, в якому передаються форма (form) та відфільтровані оголошення про роботу (job_advertisements).

5. ТЕСТУВАННЯ

5.1. Кросбраузерність

У сучасному цифровому пейзажі користувачі використовують різноманітні пристрої та браузери для відвідування веб-сайтів. Для того, щоб забезпечити належне функціонування та зручність використання веб-сайтів незалежно від пристрою чи браузера, розробники повинні ретельно налаштовувати їх з урахуванням кросбраузерності.

Кросбраузерність - це властивість веб-сайту, яка забезпечує однакову коректну роботу та відображення на різних браузерах і платформах. Це означає, що веб-сторінки мають відображатися і працювати належним чином у всіх популярних браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Opera тощо.

Для забезпечення кросбраузерності розробники використовують різні стратегії. Вони перевіряють вигляд та функціональність веб-сайту на різних браузерах, враховуючи їх особливості та можливі обмеження. Це означає, що код веб-сайту повинен бути написаний таким чином, щоб він правильно інтерпретувався кожним браузером.

Тестування кросбраузерності є важливим етапом розробки веб-сайту. Під час тестування розробники перевіряють, як веб-сайт виглядає та працює на різних пристроях та браузерах. Вони вирішують будь-які проблеми з відображенням або функціональністю, щоб забезпечити позитивний користувацький досвід для всіх користувачів.

Одними з популярних інструментів для тестування кросбраузерності є перевірка веб-сайту у різних браузерах та на різних пристроях, використання сервісів перевірки сумісності браузерів, а також використання віртуальних машин для тестування на різних операційних системах.

Кросбраузерність відіграє важливу роль у створенні веб-сайтів, які забезпечують зручне та коректне користування для всіх користувачів, незалежно від їхніх вподобань щодо браузерів та пристроїв.

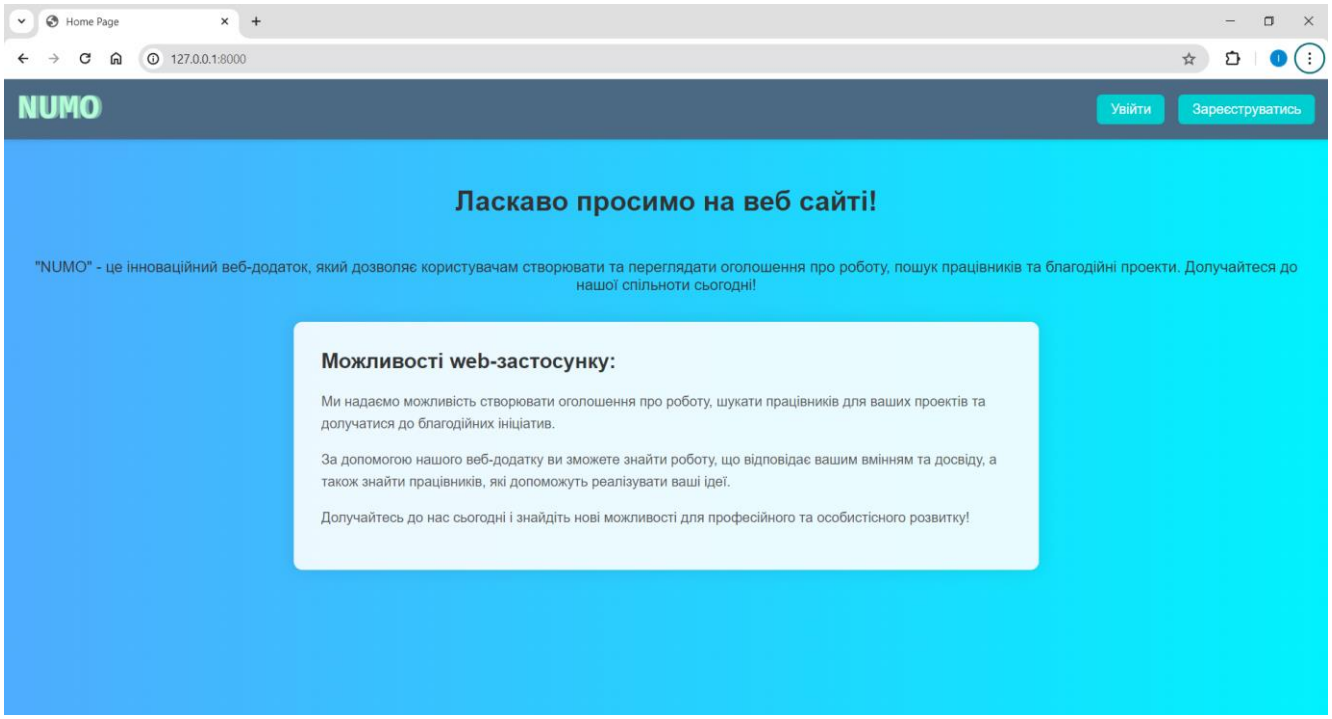


Рис. 5.1 Вигляд застосунку в браузері Google Chrome

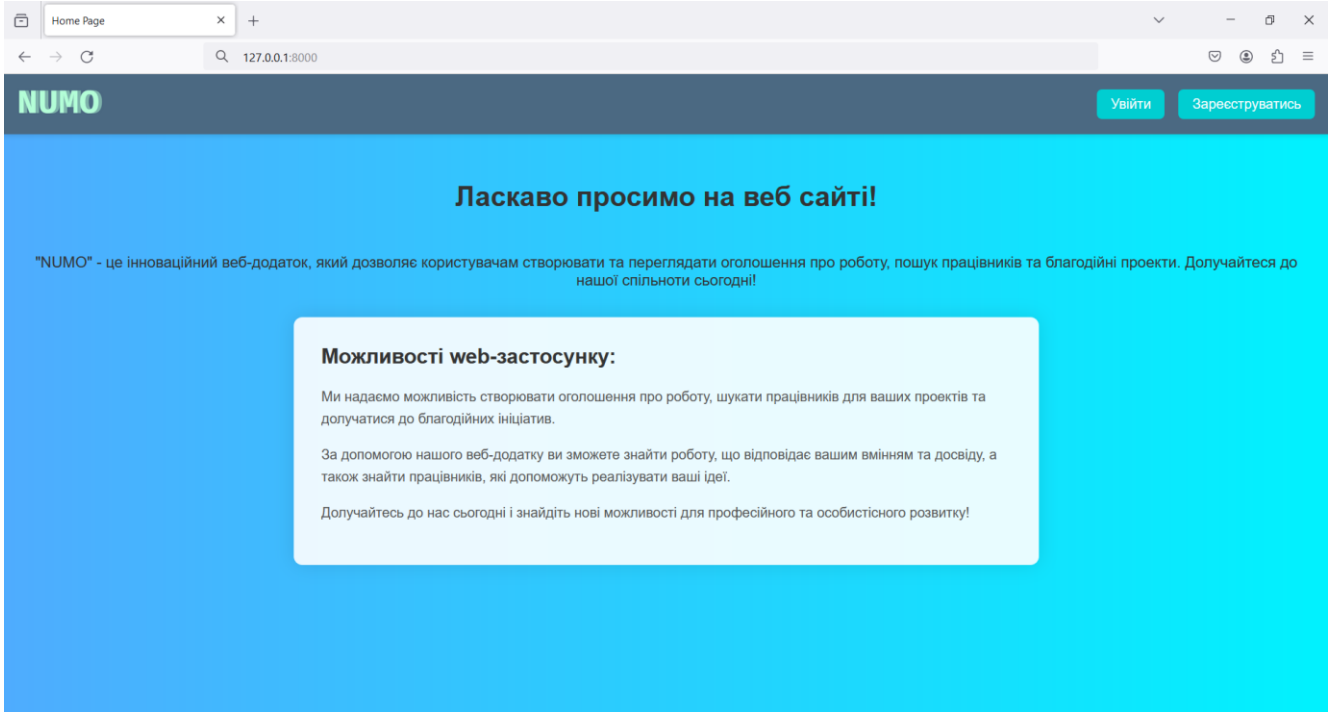


Рис. 5.2 Вигляд застосунку в браузері Mozilla Firefox

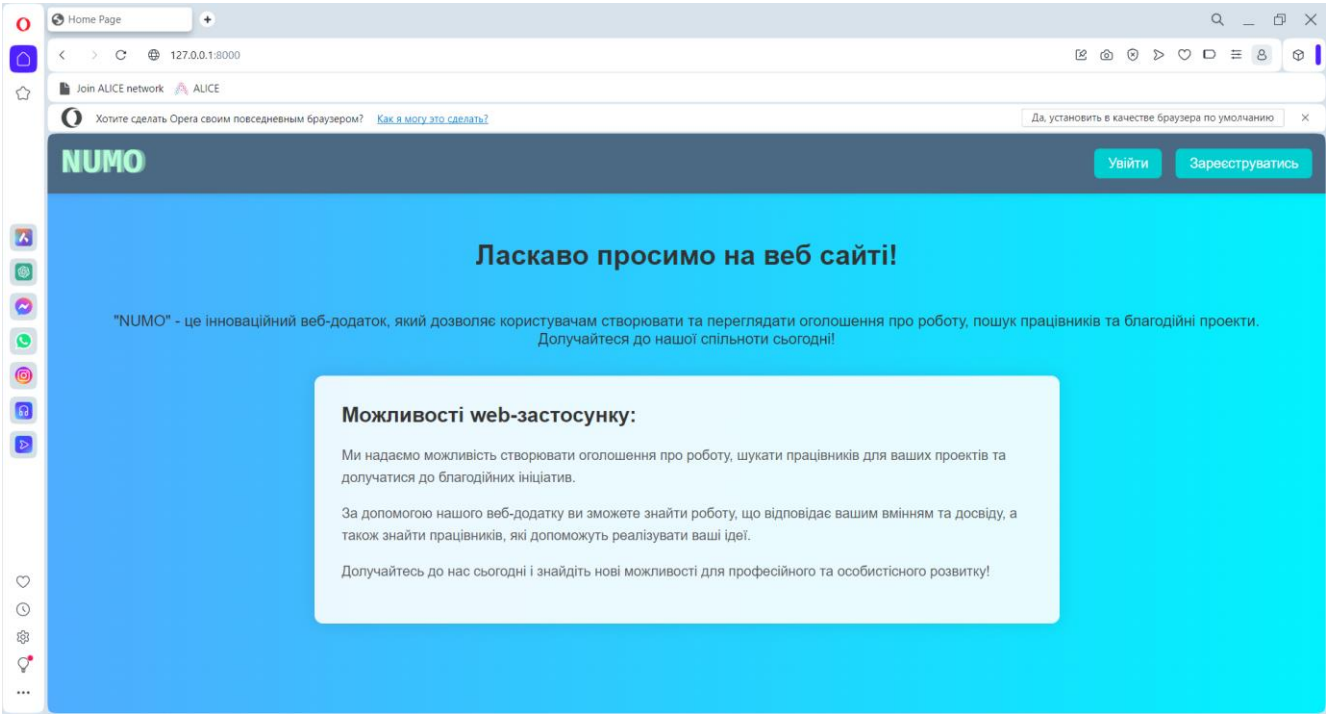


Рис. 5.3 Вигляд застосунку в браузері Opera

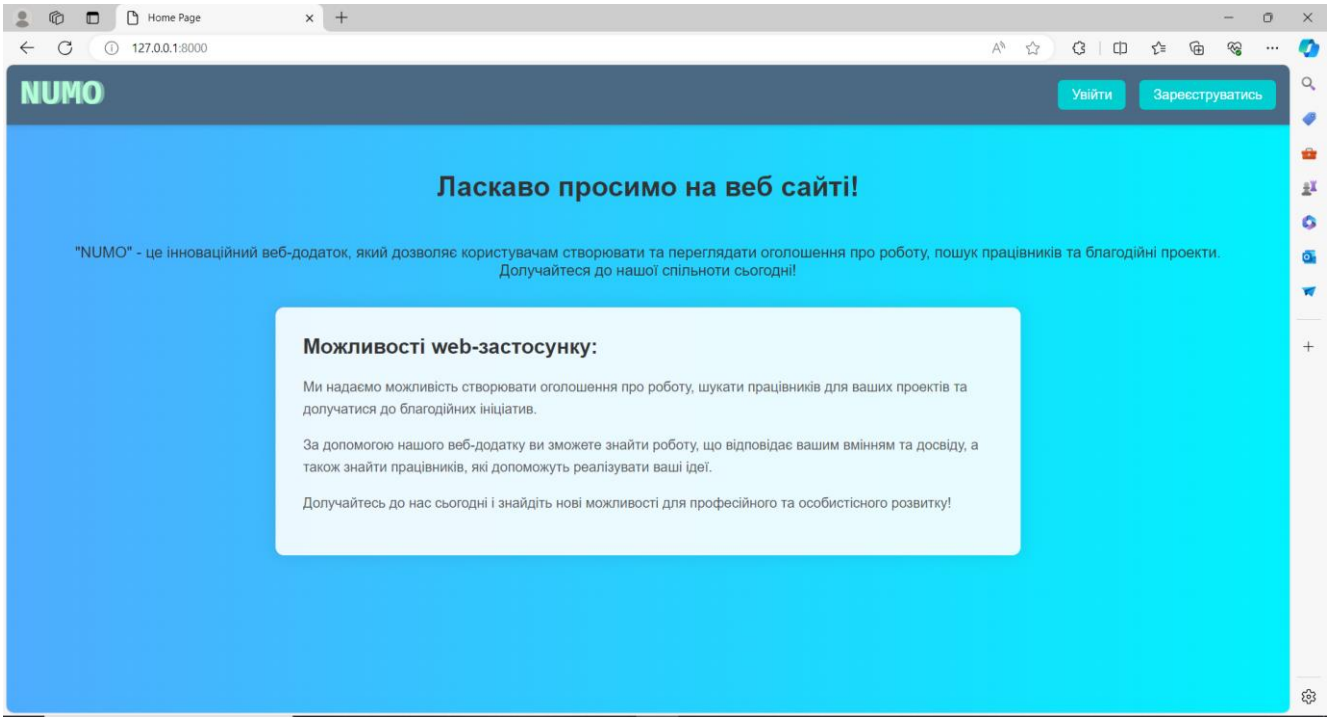


Рис. 5.4 Вигляд застосунку в браузері Microsoft Edge

5.2. Адаптивність web-застосунку

Адаптивність веб-сайтів - це їх здатність оптимально відображатися на різних пристроях з різними розмірами екрану і орієнтаціями. З урахуванням широкого спектру пристроїв, які використовуються для доступу до Інтернету, важливо мати сайт, який може пристосовуватися до будь-якого екрану і забезпечує комфортне використання. Більшість людей сьогодні використовують веб-сайти через свої смартфони, тому важливо мати адаптивний дизайн, який підходить для будь-якого пристрою, а не лише для смартфонів. Адаптивний дизайн забезпечує коректне відображення сайту на різних пристроях, включаючи комп'ютери, планшети, смартфони та інші.

Основна перевага респонсивного дизайну полягає в його здатності автоматично адаптуватися до розміру екрану, що дозволяє забезпечити зручну інтеракцію для користувачів будь-якого пристрою.

Такий підхід не лише поліпшує користувацький досвід, але й сприяє покращенню оптимізації для пошукових систем. Адаптивні сайти мають кращі шанси на підвищення рейтингу в пошукових системах і залучення більшої аудиторії.

Крім того, адаптація сайту за допомогою респонсивного дизайну спрощує процес управління та підтримки, оскільки не потрібно створювати окремі версії сайту для різних пристроїв. Для перевірки адаптивності сайту можна використовувати різні методи, такі як зміна розміру вікна браузера, використання онлайн-ресурсів для тестування та фізичне тестування на різних пристроях.

Одним із ключових аспектів адаптивності є реагування веб-сайту на зміни розміру екрану. Наприклад, веб-сайт повинен автоматично переставляти та масштабувати елементи і контент таким чином, щоб вони зберігали зручність читання та використання, незалежно від розміру пристрою.

Для досягнення адаптивності веб-сайту використовуються такі техніки, як медіа-запити (media queries), які дозволяють змінювати стилізацію та розміщення

елементів в залежності від характеристик екрану пристрою, таких як ширина і висота. Крім того, використання гнучких сіток (flexbox) і сіток згортання (grid) дозволяє створювати гнучкі та адаптивні макети для різних пристроїв. На малюнках 5.5-5.7 показано результати відображення сторінку реєстрації сайту на для різних розмірів екрану.

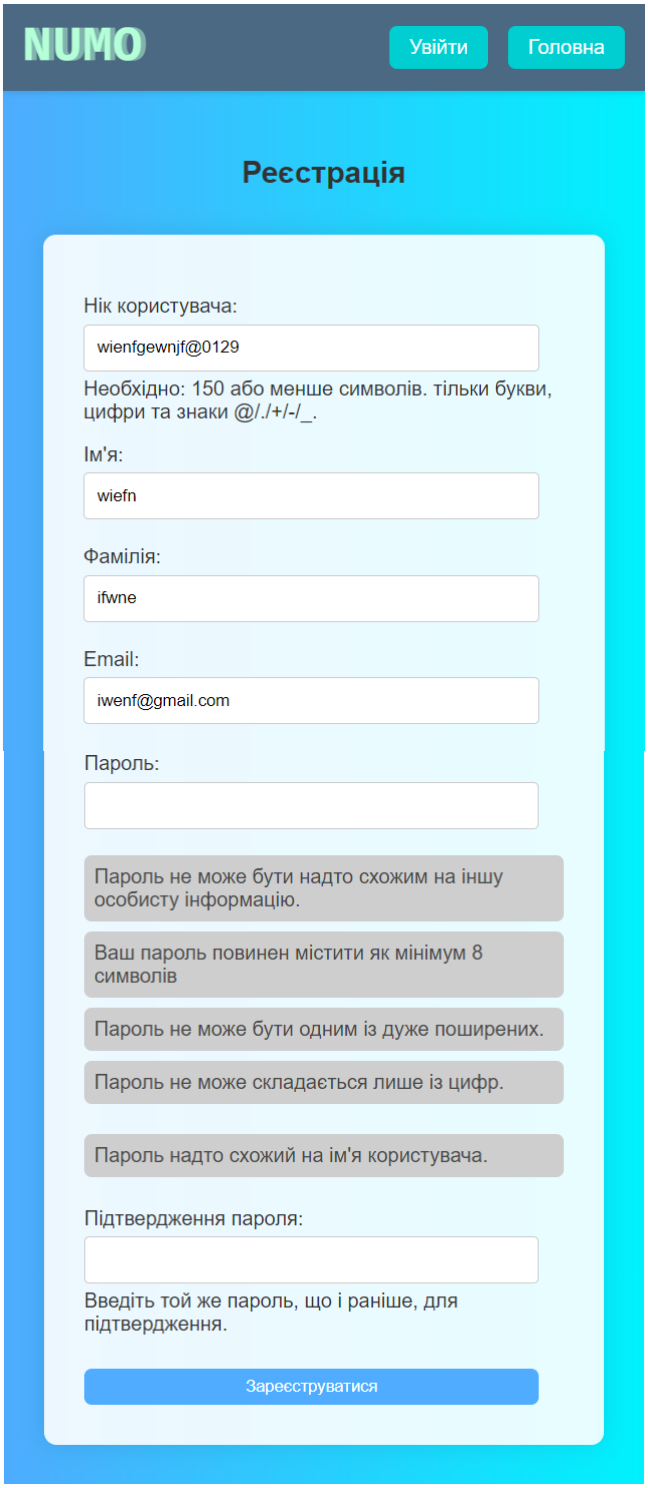


Рис. 5.5 сторінка реєстрації для пристрою шириною 510px

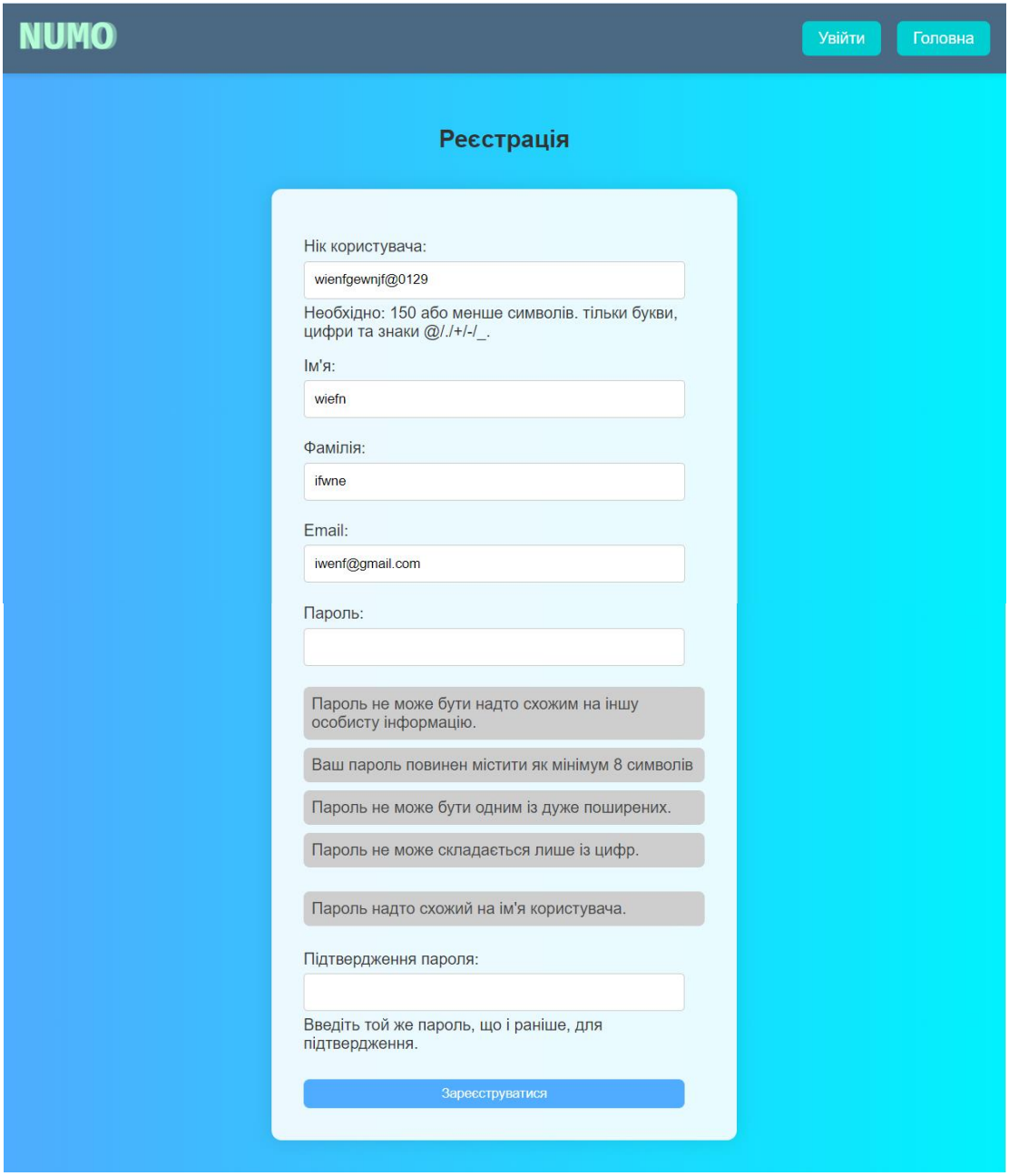


Рис. 5.6 сторінка реєстрації для пристрою шириною 1000px

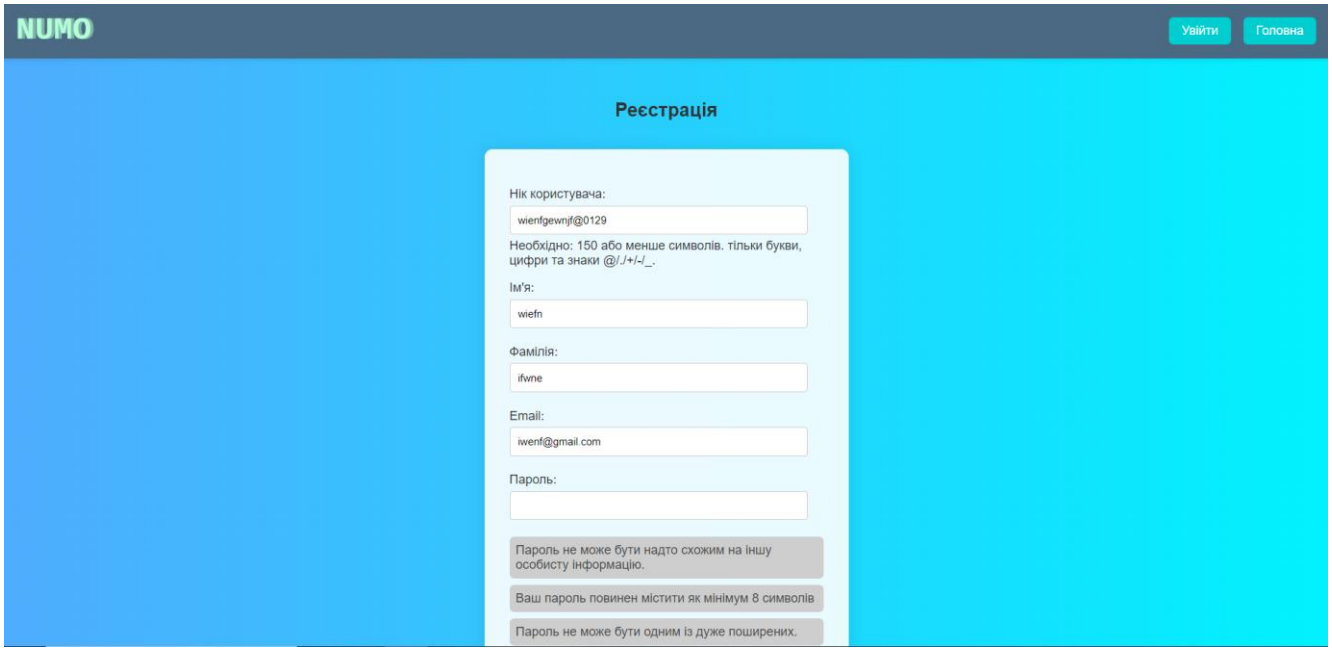


Рис. 5.7 сторінка реєстрації для пристрою шириною 1920px

5.3. Тестування функціоналу web-застосунку

Була сформована таблиця тест-кейсів яка демонструє тест-кейс, очікуваний результат та результат тестування.

Таблиця 5.3.1

Тест-кейси для додатку

№	Тест-кейс	Очікуваний результат	Результат тестування
1	Введення паролю при реєстрації схожий на особисті дані користувача	Вивід помилки що пароль надто схожий на нік користувача	Пройдено
2	При реєстрації введення ніку корисувача який уже існує в базі даних	Вивід помилки що такий корисувач уже існує	Пройдено
3	При реєстрації введення пошти яка уже зареєстрована	Вивід помилки про те що користувач з такою поштою вже існує	Пройдено

Продовження таблиці 5.3.1

Тест-кейси для додатку

№	Тест-кейс	Очікуваний резльтат	Результат тестування
4	При реєстрації введення паролю який не підходить по критеріям(мінімум 8 символів, не може бути дуже поширеним, не може складатись лише з цифр)	Вивід помилки що пароль не підходить по критеріям	Пройдено
5	Створення оголошення про пошук роботи, коли користувач його уже створив	Виведенян помилки що оголошення вже існує	Пройдено
6	При фільтрації оголошення вказати однаковий мінімальний та максимальний вік	Виведення оголошень які підходять під ці критерії, якщо такі існують	Пройдено
7	При редагуванні профілю вказати пошту або нік користувача які уже існують в базі даних	Виведення помилки про те що користувач з такою поштою або ніком уже існують	Пройдено
8	При редагуванні профілю вказати не валідні дані для електронної пошти	Виведення помилки про невалідну електронну пошту	Пройдено
9	Відфільтрувати список роботи по класу небезпеки	Виведення оголошень які мають цей клас небезпеки	Пройдено
10	Перехід до редагування оголошення через посилання вказавши id оголошення яке не належить користувачеві	Немає кнопки зберегти відредаговане оголошення та виведення помилки «ви не можете редагувати це оголошення»	Пройдено
11	Видалення профілю користувача	Разом з профілем видаляються всі оголошення користувача	Пройдено
12	Перехід до редагування профілю корисувача через посилання вказавши id користувача	Немає кнопки зберегти відредаговане оголошення та виведення помилки «ви не можете редагувати цей профіль»	Пройдено

ВИСНОВКИ

У даній дипломній роботі розглянуто процес розробки та впровадження веб-застосунку "NUMO" для пошуку роботи. Основною метою роботи було створення інструменту, що дозволяє користувачам ефективно та зручно знаходити вакансії, відповідно до їхніх професійних навичок якщо такі маються та особистих уподобань.

Під час розробки була створена модульна архітектура, що забезпечує гнучкість та масштабованість системи для подальшого розвитку та оновлення. У ході реалізації проекту були впроваджені основні функціональні модулі, які забезпечують зручний та ефективний пошук роботи.

1. Розробка web-застосунку NUMO має на меті спростити процес пошуку вакансій та робітників для фізичної праці. Об’єктом дослідження є сам процес пошуку вакансій та робітників, а предметом – веб-застосунок, який виконує цей процес.

2. В ході роботи був розроблений web-застосунок NUMO, який має широкий функціонал для користувачів. Користувачі можуть створювати, редагувати та видаляти свій профіль, а також додавати, редагувати та видаляти оголошення про роботу. Окремою можливістю є розміщення благодійних оголошень. Під час створення оголошень, користувач може вказати клас небезпеки роботи.

3. NUMO забезпечує платформу для користувачів, яка допомагає знаходити вакансії та робітників для фізичної праці. Користувачі можуть швидко та зручно знаходити необхідні оголошення та взаємодіяти з ними.

4. NUMO сприяє підвищенню доступності робочих місць для робітників та вакансій для роботодавців, що сприяє зменшенню безробіття та підвищенню ефективності ринку праці.

5. Користувачі мають можливість редагувати не лише особисту інформацію, а й редагувати дані оголошень, такі як опис вакансій, розмір оплати, клас небезпеки та інші параметри. Це дозволяє користувачам швидко адаптувати та оновлювати інформацію в оголошеннях, щоб відповідати змінюючимся потребам та умовам на ринку праці.

6. Розробка web-застосунку NUMO має на меті спростити процес пошуку вакансій та робітників для фізичної праці. Об’єктом дослідження є сам процес пошуку вакансій та робітників, а предметом – веб-застосунок, який виконує цей процес.

Книш І.О., Гаманюк І.М. ВЕБ ЗАСТОСУНОК ДЛЯ ПОШУКУ РОБІТНИКІВ ДЛЯ ФІЗИЧНОЇ ПРАЦІ НА ОСНОВІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024 С.61.

Книш І.О., Гаманюк І.М. АНАЛІЗ WEB–ЗАСТОСУНКІВ ДЛЯ ПОШУКУ РОБІТНИКІВ ДЛЯ ФІЗИЧНОЇ ПРАЦІ: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024 С.153.

ПЕРЕЛІК ПОСИЛАНЬ

1. Aidas Bendoraitis, “Web Development with Django Cookbook- Second Edition”, 2016, 384 с.

2. Antonio Melé “Django 4 By Example - Fourth Edition”, 2022, 766 с.

3. Ben Shaw , Saurabh Badhwar , Chris Guest, “Web Development with Django - Second Edition”, 2023, 764 с.

4. Денні Лане, Кендіз Леймон, Джозеф Шмелцер. “UML 2.0 in a Nutshell”, 2005, 236 с.

5. Джон Дакетт **“** HTML & CSS: Design and Build Web Sites

6. Книга, Джон Дакетт**”**, 2011, 514 с.

7. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення 06.04.2024)

8. Django framework documentation. URL: https://docs.djangoproject.com/en/5.0/ (дата звернення 15.04.2024)

9. Lewis Coulson , Brett Jephson , Rob Larsen “The HTML and CSS Workshop”, 2019, 700 с.

10. mdn web docs, “CSS: Cascading Style Sheets”, URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення 29.03.2024)

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Створення Web-застосунку для пошуку вакансій та робітників для фізичної праці мовою Python з використанням Django

Виконав студент 4 курсу
групи ПД-43
Книш Ігор Олександрович
Керівник роботи
Ст. викладач кафедри ІПЗ Гаманюк Ігор Михайлович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи –** спрощення процесу пошуку вакансій та робітників за допомогою web-застосунку NUMO.
- **Об'єкт дослідження –** процес пошуку вакансій та робітників для фізичної праці.
- **Предмет дослідження –** web-застосунок для пошуку вакансій та робітників для фізичної праці.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести огляд застосунків для пошуку вакансій та робітників для фізичної праці.
2. Провести огляд засобів, які можуть бути використані при розробці web-застосунку для пошуку вакансій та робітників для фізичної праці.
3. Визначивши основні функції та можливості, які будуть надані користувачам.
4. Розробити візуальну частину та інтерфейс користувача.
5. Реалізувати веб-застосунок.
6. Провести тестування застосунку.

3

АНАЛІЗ АНАЛОГІВ

Застосунки / Характеристики	Work.ua	OLX робота	NUMO
Можливість знайти вакансію без спеціальних навичок	–	+	+
Можливість відфільтрувати роботу по рівню небезпеки	–	–	+
Можливість знайти роботу на годину	–	–	+
Волонтерські оголошення (безоплатні оголошення)	–	+	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Реєстрація нових користувачів, вхід та вихід з акаунту.
2. Редагування інформації в профілі користувача.
3. Створення оголошень з рівнем безпечності роботи.
4. Редагування та видалення оголошень.
5. Шифрування паролів користувача.
6. Перегляд деталей оголошення.
7. Особистий кабінет користувача.
8. Пошук волонтерів для фізичної праці.
9. Фільтр рівня безпечності роботи.

Нефункціональні вимоги:

1. Система повинна бути сумісною з різними пристроями, браузерами та операційними системами, що дозволяє користувачам зручно використовувати її з будь-якого пристрою.

5

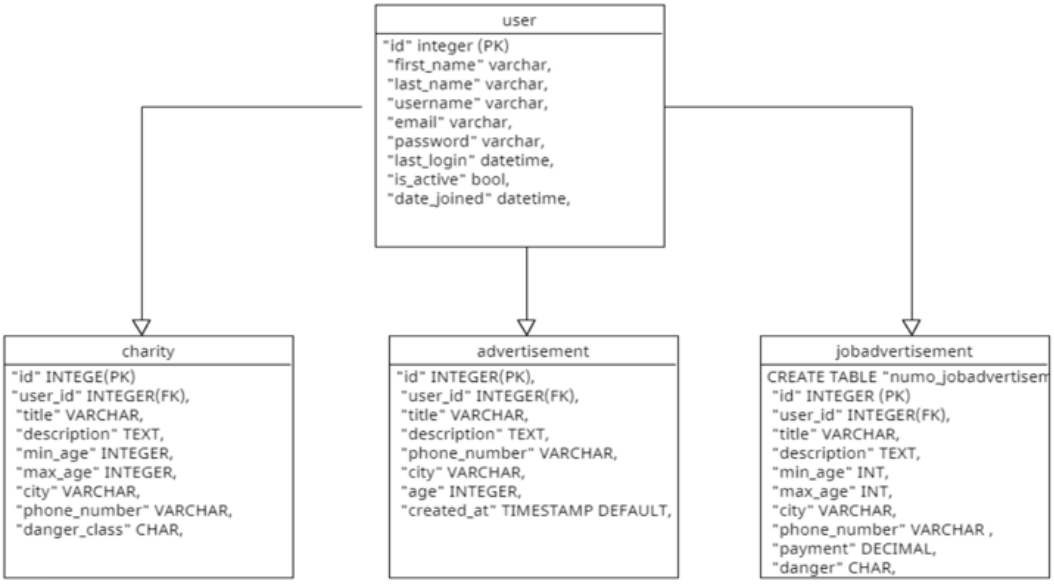
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

PyCharm

DB Browser (SQLite)

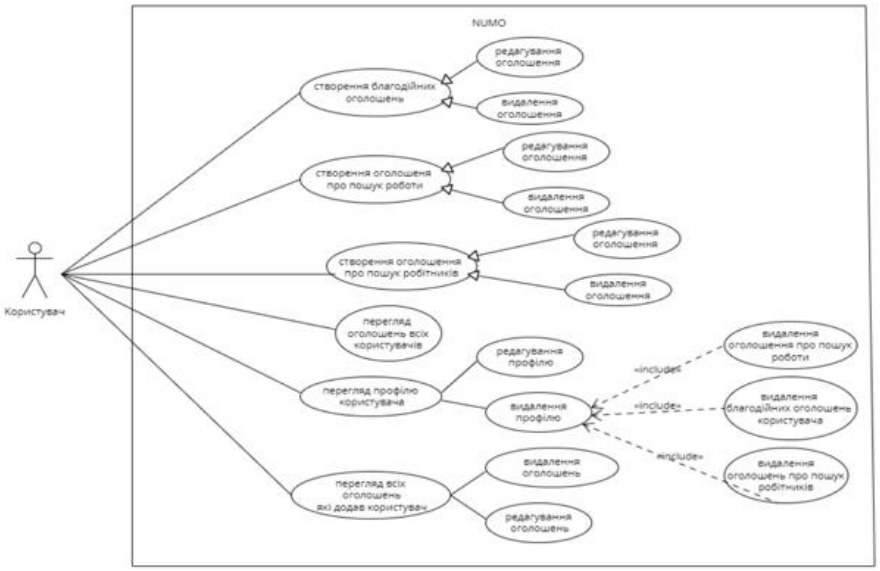
6

СТРУКТУРА БАЗИ ДАНИХ



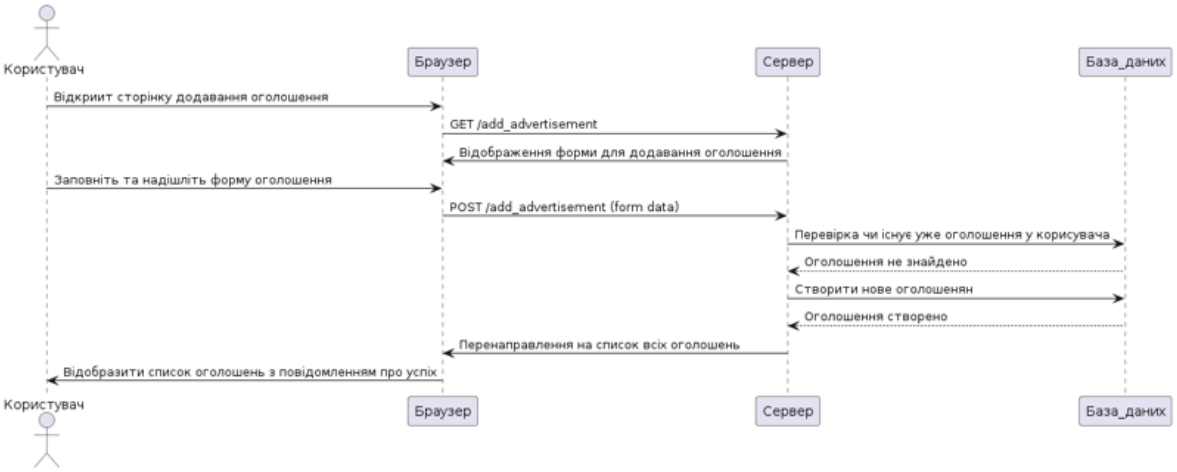
7

ДІАГРАМА ВИКОРИСТАННЯ



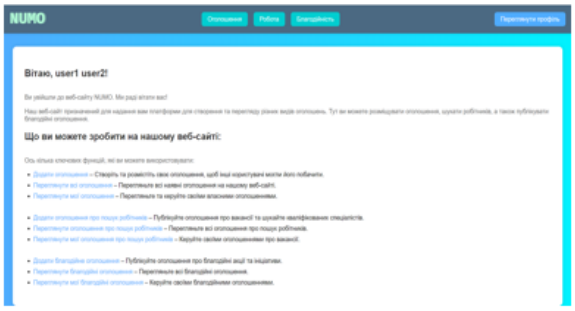
8

Діаграма послідовності створення оголошення

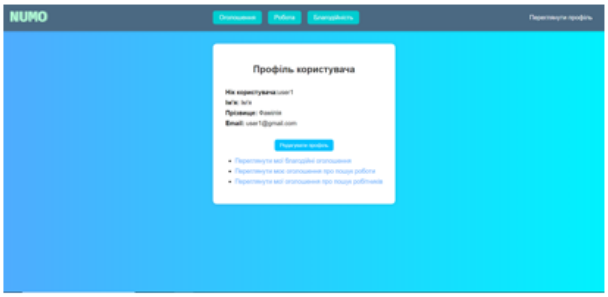


9

ЕКРАННІ ФОРМИ



Головна сторінка

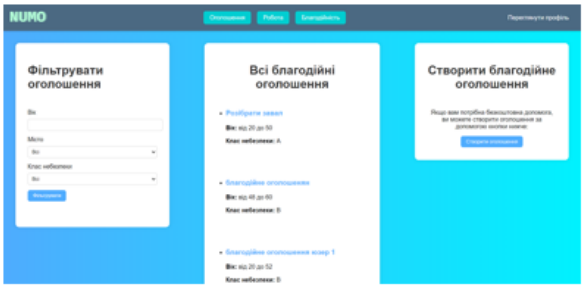


Перегляд профілю користувача

10

ЕКРАННІ ФОРМИ

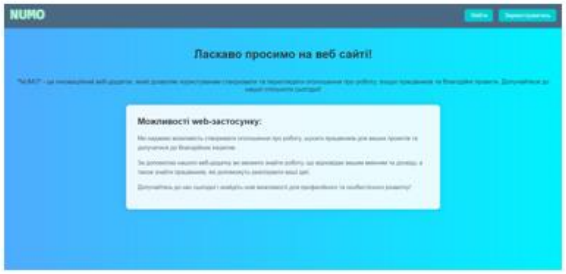
Сторінка редагування оголошення



Сторінка перегляду благодійних оголошень

11

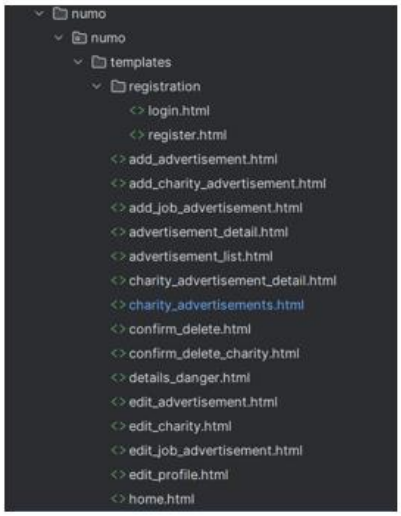
ЕКРАННІ ФОРМИ

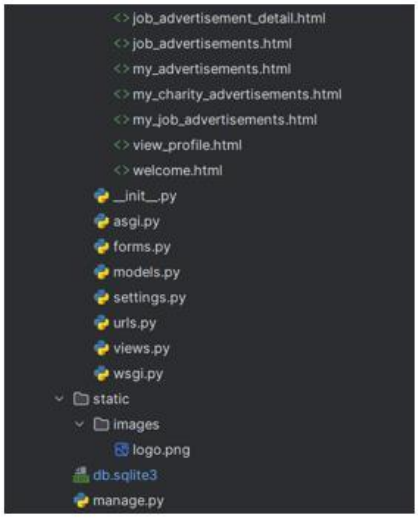


Головна сторінка для користувачів
які не увійшли в аккаунт

12

ФАЙЛОВА СТРУКТУРА ПРОЄКТУ





13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Книш І.О., Гаманюк І.М. ВЕБ ЗАСТОСУНОК ДЛЯ ПОШУКУ РОБІТНИКІВ ДЛЯ ФІЗИЧНОЇ ПРАЦІ НА ОСНОВІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, ДУІКТ, м.Київ. К.:ДУІКТ

2. Книш І.О., Гаманюк І.М. АНАЛІЗ WEB–ЗАСТОСУНКІВ ДЛЯ ПОШУКУ РОБІТНИКІВ ДЛЯ ФІЗИЧНОЇ ПРАЦІ: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, ДУІКТ, м.Київ. К.:ДУІКТ

14

ВИСНОВКИ

1. В ході роботи було проведено аналіз аналіз аналогів web-застосунків для пошуку вакансій та робітників для пошуку фізичної роботи
2. Визначено вимоги та спроектовано інтерфейс web-застосунку.
3. Розроблено функціональні та нефункціональні вимоги.
4. Було створено web-застосунок для пошуку вакансій та робітників.
5. Проведено тестування web-застосунку для пошуку вакансій та робітників для фізичної праці.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Фрагмент коду CSS

```
body {
        font-family: 'Arial', sans-serif;
        background: linear-gradient(to right, #4facfe, #00f2fe);
        margin: 0;
        display: flex;
        flex-direction: column;
        min-height: 100vh;
    }
    .header {
        background-color: #4B6982;
        color: white;
        padding: 1em;
        display: flex;
        justify-content: space-between;
        align-items: center;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }
    .header .logo {
        font-size: 1.5em;
        font-weight: bold;
    }
    .header .logo img {
        max-width: 100px;
    }
    .nav-buttons {
        display: flex;
        align-items: center;
    }
    .nav-buttons a {
        color: white;
        text-decoration: none;
        margin-left: 1em;
        padding: 0.5em 1em;
        border-radius: 6px;
        transition: background 0.3s ease;
    }
    .nav-buttons a.advertisement {
        background: #00ced1;
    }
    .nav-buttons a:hover {
        background: #00c1fe;
    }
    .main-content {
        display: flex;
        flex-direction: row;
        width: 100%;
        justify-content: space-between;
    }
    .create-container {
        background: white;
        padding: 2em;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        margin: 2em;
        max-height: 32vh;
        overflow-y: auto;
    }
    .filter-container {
        flex: 0.3;
        background: white;
        padding: 2em;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
```

```
    margin: 2em;
    max-height: 66vh;
    overflow-y: auto;
}
.advertisement-container {
    flex: 0.35;
    background: white;
    padding: 2em;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    margin: 2em;
    text-align: center;
}
.create-container {
    flex: 0.3;
    text-align: center;
}
h1 {
    margin-bottom: 1.5em;
    color: #333;
}
form {
    margin-bottom: 2em;
}
label {
    display: block;
    margin-bottom: 0.5em;
}
input[type="number"],
select {
    width: 100%;
    padding: 0.5em;
    margin-bottom: 1em;
    border-radius: 6px;
    border: 1px solid #ccc;
}
button[type="submit"],
.create-container button {
    background: #4facfe;
    color: white;
    border: none;
    padding: 0.5em 1em;
    border-radius: 6px;
    cursor: pointer;
    transition: background 0.3s ease;
}
button[type="submit"]:hover,
.create-container button:hover {
    background: #00c1fe;
}
ul {
    padding-left: 1.5em;
    text-align: left;
}
li {
    margin: 0.5em 0;
}
a {
    color: #4facfe;
    text-decoration: none;
    transition: color 0.3s ease;
}
a:hover {
    color: #00c1fe;
}
```

Фрагмент HTML коду

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Пошук працівників</title>
</head>
<body>
    <div class="header">
        <div class="logo">
            <img src="{% static 'images/logo.png' %}"
alt="Логотип">
        </div>
        <div class="nav-buttons">
            <a class="advertisement" href="{% url
'advertisement_list' %}">Оголошення </a>
            <a class="advertisement" href="{% url
'job_advertisements' %}">Робота</a>
            <a class="advertisement" href="{% url
'charity_advertisements' %}">Благодійність</a>
        </div>
        <div class="nav-buttons">
            <a class="view-profile" href="{% url
'view_profile' %}">Переглянути профіль</a>
        </div>
    </div>
    <div class="main-content">
        <div class="filter-container">
            <h1>Фільтрувати оголошення</h1>
            <form method="get">
                <label for="id_age">Вік</label>
                <input type="number" name="age"
id="id_age" value="{{ form.age.value }}">

                <label for="id_city">Місто</label>
                <select name="city" id="id_city">
                    {% for value, label in form.city.field.choices
%}
                        <option value="{{ value }}" {% if
form.city.value == value %}selected{% endif %}>{{
label }}</option>
                    {% endfor %}
                </select>

                <label for="id_danger">Клас
небезпеки</label>
                <select name="danger" id="id_danger">
                    {% for value, label in
form.danger.field.choices %}
                        <option value="{{ value }}" {% if
form.danger.value == value %}selected{% endif %}>{{
label }}</option>
                    {% endfor %}
                </select>

                <label for="id_min_payment">Мінімальна
оплата</label>
                <input type="number" step="0.01"
name="min_payment" id="id_min_payment" value="{{
form.min_payment.value }}">

                <button
type="submit">Фільтрувати</button>
            </form>
        </div>
        <div class="advertisement-container">
            <h1>Всі оголошення</h1>
            <ul>
                {% for advertisement in job_advertisements
%}
                    <li>
                        <a href="{% url
'job_advertisement_detail' advertisement.id %}">
                            <h2>{{ advertisement.title }}</h2>
                        </a>
                        <p><strong>Опис:</strong></p>
                        <p>{{ advertisement.description }}</p>
                        <p><strong>Вік:</strong> {{
advertisement.min_age }} - {{ advertisement.max_age
}}</p>
                        <p><strong>Рівень небезпеки:</strong>
{{ advertisement.danger }}</p>
                        <p><strong>Оплата:</strong> {{
advertisement.payment }}</p><br><br><br>
                    </li>
                {% endfor %}
```

```
        </ul>
    </div>
    <div class="create-container">
        <h1>Створити оголошення</h1>
        <p>Якщо ви хочете створити оголошення, натисніть на кнопку нижче:</p>
        <button onclick="location.href='{% url 'add_job_advertisement' %}'">Створити оголошення</button>
    </div>
</div>
</body>
</html>
```

Фрагмент коду на мові програмування Python

```
def charity_advertisements(request):
    charities = Charity.objects.all()

    if request.method == 'GET':
        form = AdvertisementFilterForm(request.GET, cities=Charity.objects.values_list('city', flat=True).distinct())
        if form.is_valid():
            age = form.cleaned_data.get('age')
            city = form.cleaned_data.get('city')
            danger_class = form.cleaned_data.get('danger_class')

            if age is not None:
                charities = charities.filter(min_age__lte=age, max_age__gte=age)
            if city:
                charities = charities.filter(city=city)
            if danger_class:
                charities = charities.filter(danger_class=danger_class)
    else:
        form = AdvertisementFilterForm(cities=Charity.objects.values_list('city', flat=True).distinct())
    return render(request, 'charity_advertisements.html', {'form': form, 'charities': charities})
```