

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**Хмельницький політехнічний фаховий коледж**  
**Національного університету «Львівська політехніка»**

**Відділення програмної інженерії**

Циклова комісія  
програмного забезпечення

**КУРСОВИЙ ПРОЄКТ**  
**з дисципліни «Конструювання програмного забезпечення»**

**Біржа праці.**

Виконав студент *IV* курсу, групи *ПІ192*  
спеціальності *121 Інженерія програмного*  
*забезпечення*  
*Тартаковський Владислав Святославович*

Керівник \_\_\_\_\_ *Щуцький С.В.*

Члени комісії:

_____	_____ <i>Григоровський Є.С.</i>
(підпис)	(прізвище та ініціали)
_____	_____ <i>Миколок М.В.</i>
(підпис)	(прізвище та ініціали)

Оцінка: \_\_\_\_\_

\_\_\_\_\_

дата захисту

Хмельницький 2022

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**Хмельницький політехнічний фаховий коледж**  
**Національного університету «Львівська політехніка»**

Відділення *програмної інженерії*  
Циклова комісія *програмного забезпечення*  
Спеціальність *121 Інженерія програмного забезпечення*

*Затверджую*  
Голова циклової комісії

\_\_\_\_\_ **М.В. Миколук**

« 08 » вересня 2022 р.

**Індивідуальне завдання**  
на курсовий проєкт з дисципліни  
**«Конструювання програмного забезпечення»**

отримав студенту **IV** курсу групи **ПІ192**  
**Тартаковському Владиславу Святославовичу**

**Тема проєкту *Біржа праці***

Керівник курсового проєкту *Щуцький С.В.*

Затверджено на засіданні циклової комісії програмного забезпечення, протокол №7  
від 08.09. 2022 р.

**Постановка задачі**

База безробітних: піп, адреса, дн, освіта, сімейний стан, контактні координати, вимоги до майбутньої роботи, про себе. База вакансій: фірма, посада, умови праці та оплати, вимоги до фахівця. Пошук і реєстрація варіантів з того та іншого боку

**Зміст пояснювальної записки**

Реферат

Зміст

Вступ

**1. Загальний розділ**

- 1.1 Постановка завдання
- 1.2 Опис вхідної інформації
- 1.3 Опис результуючої інформації
- 1.4 Формалізований опис задачі
- 1.5 Опис існуючих методів та рішень

**2. Розробка технічного та робочого проєкту**

- 2.1 Проектування архітектури, структури даних, інтерфейсу ПЗ
- 2.2 Розробка та опис програмної реалізації та тестування задачі

**3. Спеціальний розділ**

- 3.1 Інструкція з інсталяції розробленого проєкту
- 3.2 Інструкція з експлуатації проєкту

Висновки

Список використаних джерел

Додатки

## КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапу проектування	Термін	Примітка
1.	Реферат. Зміст. Вступ. Постановка задачі. Опис вхідної інформації. Опис результуючої інформації	22.09.2022	
2.	Формалізований опис задачі. Опис існуючих методів та рішень	06.10.2022	
3.	Проектування архітектури, структур даних, інтерфейсу ПЗ	20.10.2022	
4.	Розробка та опис програмної реалізації та тестування задачі	27.10.2021	
5.	Розробка інструкцій з інсталяції та експлуатації розробленого проекту	01.11.2021	
6.	Оформлення ПЗ	03.11.2022	
7.	Оформлення графічної частини (CASE-діаграми)	03.11.2022	
8.	Підготовка презентації доповіді на захист	03.11.2022	
9.	Перевірка ПЗ курсового проекту	03.11-08.11	
10.	Захист	08.11-15.11	

Видано «08» «вересня» 2022 року

Термін виконання «08» «листопада» 2022 року

Індивідуальне завдання одержав Тартаковський В.С.

\_\_\_\_\_ «08» «вересня» 2022 року  
(підпис студента)

Керівник \_\_\_\_\_ (Щуцький С.В.)

[illegible]

					КП.ПО.09.ПІ-192.38.ВП						
Змн.	Арк.	№ докум.	Підпис	Дата	Біржа праці.  Відомість проекту.			Літ.	Арк.	Аркушів	
Розроб.		Тартаковський								1	1
Перевір.		Щуцький									
Н. Контр.											
Затверд.								ХПФК НУ “ЛП”			

## **Abstract**

The course project on the topic: "Labor Exchange" was developed according to the individual task approved at the meeting of the cyclic software commission, protocol No. 7 dated September 8, 2022.

The explanatory note contains 37 pages, 23 figures, 10 used sources of information and 6 references.

The object of development is the "Labor Exchange".

The purpose of the development is to create software to enable the employer to find employees using the job posting, and the unemployed to find the desired job among the posted vacancies by the administrator.

The development method is using the Python programming language using the Django framework in the PyCharm environment.

The result of the development is a multifunctional site with a convenient opportunity to view vacancies and enter personal data by selecting the desired vacancy.

The program will be used to search for a job among the posted vacancies and enter data among the selected vacancies, the administrator can view the user data posted by the unemployed near the selected vacancies. The administrator can add, edit and delete vacancies, and the unemployed can add, edit and delete data about themselves.

Keywords: jobs, administrator, user, Python, Django.

## **Реферат**

Курсовий проект на тему: «Біржа праці» розроблено згідно індивідуального завдання, затвердженого на засіданні циклової комісії програмного забезпечення, протокол № 7 від 08.09.2022 р.

Пояснювальна записка містить 34 сторінок, 23 рисунків, 10 використаних джерел інформації та 7 довідників.

Об'єкт розробки – це “Біржа праці”.

Метою розробки є створення програмного забезпечення для можливості роботодавцю знайти працівників за допомогою розміщення вакансії, а безробітному знайти потрібну роботу серед розміщених вакансій адміністратором.

Метод розробки – за допомогою мови програмування Python з використанням фреймворку Django в середовищі PyCharm.

Результат розробки – багатофункціональний сайт зі зручною можливістю перегляду вакансій і внесення даних про себе, вибравши потрібно вакансію.

Застосовуватися програма буде для пошуку роботи серед розміщених вакансій і внесення даних серед вибраної вакансії, адміністратор може переглянути дані про користувача, які розмістив безробітний біля вибраної вакансії. Адміністратор може додавати, редагувати та видаляти вакансії, а безробітний може додавати, редагувати та видаляти дані про себе.

Ключові слова: вакансії, адміністратор, користувач, Python, Django.

## Зміст

<b>1. Загальний розділ .....</b>	<b>3</b>
<b>1.1. Постановка завдання .....</b>	<b>3</b>
<b>1.2. Опис вхідної інформації.....</b>	<b>4</b>
<b>1.3. Опис результуючої інформації.....</b>	<b>5</b>
<b>1.4. Формалізований опис задачі.....</b>	<b>5</b>
<b>1.5. Опис існуючих методів та рішень .....</b>	<b>7</b>
<b>2. Розробка технічного та робочого проекту.....</b>	<b>14</b>
<b>2.1. Проектування архітектури, структури даних, інтерфейсу ПЗ ..</b>	<b>14</b>
<b>2.2. Розробка та опис програмної реалізації та тестування.....</b>	<b>18</b>
<b>3. Спеціальний розділ.....</b>	<b>29</b>
<b>3.1. Інструкція з інсталяції розробленого проекту .....</b>	<b>29</b>
<b>3.2. Інструкція з експлуатації проекту.....</b>	<b>30</b>
<b>Висновки .....</b>	<b>33</b>
<b>Додаток А код програми .....</b>	<b>35</b>

					КП.ПО.09.ПІ-192.38.ВП		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Тартаковський			Біржа праці.  Пояснювальна записка.	Літ.	Арк.
Перевір.		Щуцький					Аркушів
						1	34
Н. Контр.						ХФПК НУ “ЛПТ”	
Затверд.							

## Вступ

З давніх давен пошук роботи для людини був надзвичайно важливим. Біржі, тобто місця, де збирались люди і укладали угоди на продаж і купівлю великої кількості товарів виникли ще в часи середньовіччя. Пізніше на біржах стали пропонувати і робочу силу. Біржі праці стали масово створюватись з розвитком капіталістичних виробничих відносин.

Біржа праці – це організація, що спеціалізується на посередництві між робітниками та підприємцями з метою купівлі-продажу робочої сили. Не усуваючи безробіття в цілому, біржа праці дозволяє впорядкувати цей процес для підприємств, господарств, фірм і скоротити людям час пошуку місця роботи.

У наш час пошук роботи є актуальним питанням. Адже багато хто є безробітним з тих чи інших причин. Особливо коли розпочалась повномасштабна війна на території України. В наслідок цього рівень безробіття в Україні досягнув рекордної позначки 35%. Саме створений сайт може допомогти знайти роботу людині, яка їй сподобалась. Або хтось відкрив свою фірму і прагне знайти працівників. Йому важливо повідомити про відкриття фірми багатьом людям, щоб швидше знайти кваліфікованого працівника. Саме тому у мене виникла ідея створити програму “Біржа праці”. Яка дозволяє безробітним переглянути вільні вакансії та умови для працевлаштування та багато іншої інформації щодо різних вакансій. Вакансії може добавляти, змінювати або видаляти адміністратор. Згідно тих даних, які переглянуть роботодавці або з телефонної розмови, зроблять висновок чи брати людину на роботу. Перевагою проекту є те, в ньому наявний простий та зрозумілий інтерфейс, можуть користуватись користувачі будь-якого віку.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		



## 1. Загальний розділ

### 1.1. Постановка завдання

Проект має назву “Біржа праці”. Головною цілю моєї програми є можливість перегляду інформації про вакансії користувачем і внесенням даних про себе після того, як зацікавила відповідна вакансія. Адміністратор може переглянути ось ці дані, які ввів безробітний і може зробити певні висновки щодо нього. Наприклад, може зателефонувати або написати на електронну пошту безробітному. Або надіслати ось цю інформацію роботодавцю. Цей проект дозволяє роботодавцю знайти працівника для своєї роботи, а безробітному знайти роботу.

Потрібно реалізувати такі дії:

- додавання, редагування та видалення даних про вакансії з їх характеристиками: посада, фірма, місто, вулиця, заробітна плата, вимоги, умови праці;
- додавання безробітному даних про себе: прізвище, ім'я, по батькові, місто, вулиця, номер телефона, email, вік, спеціальність, вакансії;
- зрозумілий інтерфейс;
- зручність у роботі з програмою;
- реєстрація і авторизація користувачів;
- здійснити вивід вакансії за вибраною професією;
- реалізувати пошук за містом та заробітною платою;
- збереження даних в базу даних.

Система буде реалізована у вигляді веб-сайту на мові програмування Python з використанням фреймворку Django.

Отже, на цьому етапі було визначено призначення проекту та за допомогою яких технологій це буде реалізовуватись.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

## 1.2. Опис вхідної інформації

Вхідна інформація – це та інформація, що надходить у програму для опрацювання. Ця інформація може зчитуватись з файлу чи бази даних або вводиться користувачем. Вхідною інформацією будуть поля модель. Дані про вакансії і професії в категорію буде добавляти лише адміністратор. Дані про безробітного, який вибрав вакансію і для реєстрації буде вводити безробітний.

А саме буде інформація про:

Безробітний:

- Прізвище (текстовий тип);
- Ім'я (текстовий тип);
- По батькові (текстовий тип);
- Місто (текстовий тип);
- Вулиця (текстовий тип);
- Вік (числовий тип);
- Номер телефона (текстовий тип);
- Email (email тип);
- Спеціальність (текстовий тип).

Вакансії:

- фірма (текстовий тип);
- місто (текстовий тип);
- вулиця (текстовий тип);
- заробітна плата (числовий тип);
- вимоги (текстовий тип);
- умови праці (текстовий тип).

Професія:

- Назва (текстовий тип);

Реєстрація:

- логін (текстовий тип);

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

- email (email тип);
- пароль (тип пароля);
- повтор пароля (тип пароля).

Дані про вакансії обов'язкові до заповнення адміністратором якщо не буде добавлено жодної вакансії, то не буде необхідності користуватись програмою. Для входу з можливістю внесення даних про себе, якщо вибрана відповідна вакансія, на сайт обов'язково потрібно здійснити реєстрацію та авторизацію, а дані про безробітного необов'язкові до заповнення, тому що програма буде працювати і без заповнення даних.

### 1.3. Опис результуючої інформації

Результуюча інформація – це та інформація, яка отримується в результаті опрацювання та збереження даних.

Результуючою інформацією буде розміщення вакансій на веб сторінці у відсортованому по даті додання. При пошуку за містом та заробітною платою виведеться лише ті вакансії де знаходиться робоче місце за потрібним містом та шуканою заробітною платою. Можна вивести лише ті вакансії, які відповідають вибраній професії так і загалом всі. Також до результуючої інформації можна віднести дані, які вводить безробітний, якщо вибрав вакансію про себе і ті дані, які вводить користувач під час реєстрації. Ці дані збережуться в базі даних і вони будуть результуючими для адміністратора. І адміністратор може з ними працювати.

### 1.4. Формалізований опис задачі

При розробці проекту потрібно розділити задачу на певні блоки. Це дозволить зрозуміти, як потрібно виконувати дане завдання і пришвидшити розробку. Тому було поділено задачу на ось такі блоки:

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

- визначення мети та завдання;
- розробка алгоритму розв'язання;
- розробка математичної моделі;
- написання програмного коду;
- тестування;
- впровадження.

Для формалізованого опису задачі можна побудувати UML діаграму Use Case. Use Case діаграма [КП.П09.ПІ.192.38.Д2] демонструє загальні поняття системи на початкових етапах проектування. Ну і вимоги до функціональної поведінки [2].

Діаграма показує, що користувач може виконувати такі дії:

- Реєстрацію та авторизацію;
- перегляд вакансій;
- здійснення пошуку за містом;
- додати дані про себе, вибравши вакансію;
- вихід з акаунту.

Адмін може виконувати ось такі дії:

- додавати, редагувати та видаляти вакансії;
- переглядати дані по безробітних;
- додавати, редагувати та видаляти дані по безробітних;
- здійснювати контроль за користувачами.
- Але переглядати вакансії може користувач та адміністратор лише після того, як вакансії були добавлені користувачем.

Також можна побудувати діаграму послідовності. Діаграма послідовності [КП.П09.ПІ.192.38.Д3] – забезпечує розуміння того, як саме буде функціонувати програма. Вона повідомляє розробнику у якій послідовності треба реалізовувати програму [2].

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, саме на цьому етапі задачу було розділено на блоки для кращого розуміння розробки проект.

## 1.5. Опис існуючих методів та рішень

Розробка програмного забезпечення – це досить складний процес, який потребує здійснення аналізу вже існуючих програм такого типу. Перегляд схожих програм дозволить переглянути існуючі методи та рішення. Що у свою чергу полегшить розробку власного програмного продукту. Наприклад було переглянуто сайт “Work.ua”. Перевагою цього проекту є те що можна знайти вільні вакансії до більшості професії по всій Україні. Можна переглянути як і всі наявні вакансії так і можна переглянути вакансії за певною професією або за певним містом. Можна переглянути актуальну інформацію про цю вакансію. Також безробітній може додати до певної вакансії файл з резюме. А роботодавець може додавати, редагувати та видаляти вакансії. Недоліком цього проекту є те, що біля вакансії можна лише додати резюме, а такі дані які можна без резюме вказати немає можливості і ще неможливо здійснити пошук за заробітною платою.

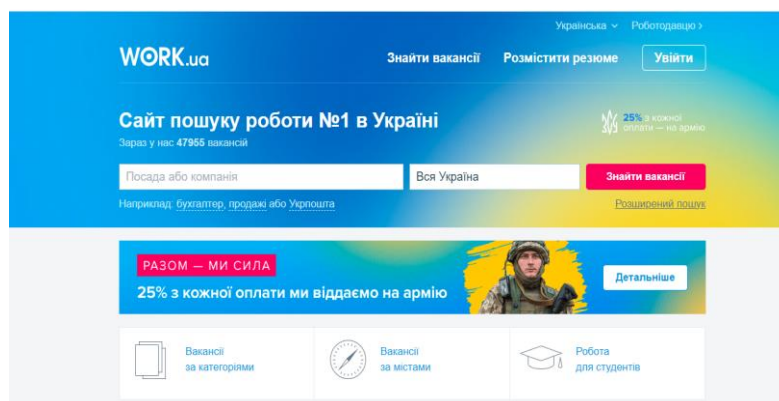


Рисунок 1.1 – “Сайт пошуку роботи”

Можна було створити десктопний проект, використовуючи мови програмування C#, C++ та інші. Але вона буде більш громісткою і до того ж

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

розробка сайтів набирає все більшої популярності. Адже сайтом можна користуватись в будь-якому місці де є доступ до мережі інтернет і з використанням комп'ютерів, смартфонів та інших пристроїв де наявний браузер. Ще можна було створити Android додаток, використовуючи мову програмування Java. Але тоді зможуть користуватися моїм додатком лише користувачі телефонів з операційною системою Android. Тому я вирішив створити сайт. Цей сайт можна було написати на мові програмування PHP, Python та з використанням технології ASP.NET.

PHP – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веброзробок (разом із Java, .NET, JavaScript, Python, Ruby) [9].

Python - дуже гнучка інтерпретована об'єктно-орієнтована мова програмування, на ній пишуть ігри, сервіси, веб-додатки, крон-скрипти для бекапу [1].

ASP.NET — платформа розробки веб-сайтів, до складу якої входять: веб-сервіси, програмна інфраструктура, модель програмування, від компанії Майкрософт [9].

Було вирішено створити сайт, використовуючи мову програмування Python. Тому що ось ця мова має ось такі переваги: проста в розумінні, легко виправляти і оновлювати код.

Під час створення проектів доцільно використовувати фреймворки. Фреймворк – це програмне середовище, яке спрощує та прискорює створення програмного забезпечення. У фреймворках зазвичай реалізуються задачі, які спільні для різних веб-сайтів, веб-програм та інших продуктів. Створювати код для виконання цих завдань щоразу – це невиправдана трата часу та зусиль. Саме для цього і призначені фреймворки. Існують ось такі фреймворки Python: Django, Flask, Tornado та багато інших.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Django — це високорівневий веб-фреймворк Python, який дозволяє швидко створювати безпечні веб-сайти. Створений досвідченими розробниками, Django бере на себе більшу частину веб-розробки, тому можна зосередитись на написанні код без необхідності придумувати велосипед. Він безкоштовний і з відкритим вихідним кодом, має динамічне та активне співтовариство, відмінну документацію та безліч варіантів як безкоштовної, так і платної підтримки. Підходить для масштабних інтерактивних проєктів, так як підтримує шаблон проектування MVC, що дозволяє сайтам швидко реагувати на дії користувачів з використанням його фреймворку та з використанням HTML, CSS та JS [4].

Flask — фреймворк для створення веб-прикладень на мові програмування Python, що використовує набір інструментів Werkzeug, а також шаблонизатор Jinja2. Відноситься до категорії так званих мікрофреймворків — мінімалістичних каркасів веб-прикладень, відповідно надають лише самі базові можливості. Flask залежить від деяких зовнішніх бібліотек — таких як Werkzeug і Jinja2. Werkzeug — це інструментарій для

WSGI, стандартного інтерфейсу Python між веб-додатками та різними серверами. Він призначений як для розробки, так і розгортання. Jinja2 — шаблонизатор [6].

Tornado — порівняно швидкий веб-фреймворк, який може обробляти тисячі одночасних постійних підключень. Зазвичай він використовується для веб-сокетів та інших додатків, що потребують тривалого з'єднання з кожним користувачем [6].

Було вирішено використати фреймворк Django, тому що це найпопулярніший фреймворк Python і використовується архітектурний шаблон проектування MVT.

Архітектурний шаблон - це узагальнене рішення поширеної проблеми в архітектурі програмного забезпечення в заданому контексті. Шаблон - це

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

вирішення завдання в певному контексті. Часто розробники не до кінця розуміють різницю між архітектурними шаблонами, а іноді взагалі мало що про них знають. Існують ось такі архітектурні шаблони: багаторівнева архітектура, канали і фільтри, клієнт-сервер, модель-представлення-контролер, керована подіями архітектура, архітектура на основі мікросервісів. Було використано архітектурний шаблон “модель-представлення-контролер”.

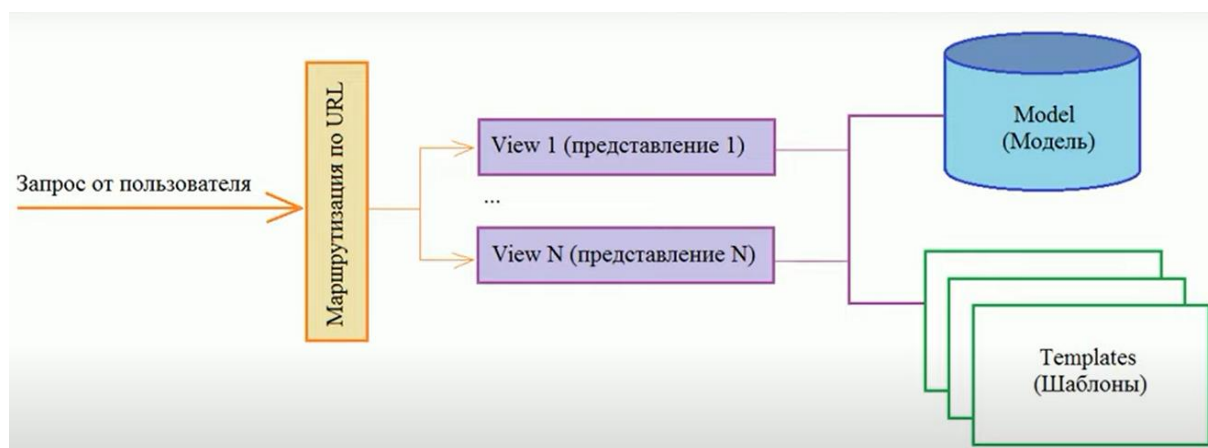


Рисунок 1.2 – “Архітектурний шаблон MVT”

Між архітектурним шаблоном MVT та MVC різниці немає, адже суть така сама. Шаблон “модель - представлення – контролер” (MVC) розділяє функціональність програми на компоненти трьох видів. Модель - містить дані програми. У середині моделі створюються класи, які складаються з властивостей (тобто характеристик даного об’єкта). На основі створеного класу створюються таблиці в базі даних з такими полями, які вказані у класі і при цьому фреймворк автоматично додає поле id. І нам не потрібно створювати це поле у класі, що є доволі зручно. Вибір способу організації всіх даних має величезне значення, через те, що саме від цього буде залежати подальший функціонал та можливості нашої програми, як зможе користувач взаємодіяти з ними, та яким чином вони будуть оброблятися в подальшому.



Для правильної структуризації даних необхідна їх обов'язкова ідентифікація, тобто прив'язка до певних моделей даних. Це є необхідним для подальшої швидкої та коректної обробки системою. Представлення - відображає деяку частину базових даних і взаємодіє з користувачем. Контролер - діє як посередник між моделлю і представленням і керує повідомленнями про зміну стану. Але в ось такого архітектурного шаблону є ще недоліки. Наприклад ось така для простих користувацьких інтерфейсів складність може бути надмірною. Абстракції «модель», «представлення» і «контролер» можуть не дуже добре підходити в разі деяких наборів інструментів для розробки користувацького інтерфейсу. Архітектурний шаблон MVC зазвичай використовується в мобільних і веб-застосунках при розробці користувацьких інтерфейсів [4].

Після того, як було вибрано мову програмування та фреймворк для створення сайту потрібно визначити редактор чи IDE на якому можна писати код. IDE має більший функціонал, ніж редактори, однак потребує більш міцної системи. Існує багато середовищ розробки наприклад: PyCharm, Visual Studio Code, Atom та багато інших.

PyCharm основний редактор Python його використовують 33% програмістів. Перевага IDE — вбудований термінал для запуску коду під час роботи. PyCharm підтримує веб-розробку завдяки інтеграції JavaScript, HTML і CSS. Також він підтримує фреймворки Python (Django) або бібліотек, які використовуються в наукових дослідженнях і візуалізації (NumPy, Anaconda, Matplotlib). PyCharm доступний для Windows, macOS і Linux [3].

Visual Studio Code - це текстовий редактор з відкритим кодом. Він розроблений в Microsoft на базі фреймворка Electron. Додаток практично підтримує всі мови програмування та інтеграцію з Git і GitHub. У редакторі є бібліотека елементів коду. Також можна додати в пам'ять сніпети — власні

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

фрагменти коду. Visual Studio Code підтримує одночасну роботу з кількома проектами [3].

Atom редактор коду теж побудований на базі Electron, але був запущений раніше, ніж Visual Studio Code. Серед можливостей обох редакторів — додавання плагінів, розширень і тем, яких в об'єднаних програмах більше 10 тисяч. У Visual Studio Code контролюється базовий набір функцій, а плагіни додають поверхневий функціонал програми. У випадку, якщо в основі програми стоять плагіни Atom'а, що дозволяє робити персоналізовану настройку. Atom підтримує кросплатформене редагування. Також є можливість розділити вікно програми для окремого відображення коду та файлів [3]. Було вибрано середовище для розробки проекту PyCharm. Тому що, через термінал можна запустити сервер, записати дані в базу даних, створити міграції та багато інших речей за допомогою спеціальних команд.

Оскільки у своєму проекті я буду використовувати базу даних, то потрібно вибрати СУБД. Існують ось такі СУБД: MongoDB, SQLite, MySQL.

MongoDB — система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потрібні таблиці, схеми або окрема мова запитів. Інформація зберігається у вигляді документів або колекцій [5].

SQLite – це вбудована реляційна база даних, з відкритим вихідним кодом. Тобто вона не використовує звичну нам модель роботи бази даних клієнт-сервер і не є окремим працюючим процесом. Наприклад, база даних MySQL. Іншими словами движок SQLite стає як би частиною нашого веб-додатки. При такому підході база даних SQLite (з усіма таблицями) являє собою звичайний текстовий файл, який ви можете розташувати в зручному для вас місці [5].

MySQL - це система керування базами даних. У реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць [5].

Я вирішив використати СУБД SQLite, тому що вона містить наступні переваги: самодостатня (не потрібний окремий сервер для роботи), легко встановити та продуктивна.

При роботі з базою даних потрібно здійснювати нормалізацію даних. Нормальна форма – властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних.

Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення. Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми.

Перша нормальна форма утворює ґрунт для структурованої схеми бази даних. Кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис. Необхідно уникати повторень груп правильно визначаючи неключові атрибути. Кожен атрибут повинен мати лише одне значення, а не множину значень (атомарність).

Друга нормальна форма вимагає, аби дані, що зберігаються в таблицях із композитним ключем, не залежали лише від частини ключа. Схема бази даних повинна відповідати вимогам першої нормальної форми. Дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці.

Третя нормальна форма вимагає, аби дані в таблиці залежали винятково від основного ключа.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

## 2. Розробка технічного та робочого проєкту

### 2.1. Проектування архітектури, структури даних, інтерфейсу ПЗ

Спочатку було вирішено здійснити нормалізацію даних для наявної бази даних. Для того, щоб були відсутні аномалії даних. Для цього потрібно звести базу даних до першої нормальної форми. У таблиці не повинні бути неатомарні атрибути. У моєму випадку в таблиці є поля ППП та адреса. Замість одного поля ППП потрібно розбити на атрибути прізвище, ім'я, по батькові. Замість поля адреса потрібно розбити на місто та вулиця. Для даного відношення повинні бути унікальні поля, в даному випадку – це код вакансії. Далі потрібно звести базу даних до другої нормальної форми. Для цього має виконуватись перша нормальна форма. Далі потрібно звести функціональну залежність не ключових атрибутів від частин ключа. Наприклад якщо в мене буде одне відношення з полями: код вакансії, фірма, місто, вулиця, заробітна плата, вимоги, умови праці, дата додання, код професії, назва, прізвище, ім'я, по батькові, місто, вулиця, вік, номер телефона, email, спеціальність. Деякі атрибути залежать від частини первичного ключа. Наприклад, фірма залежить від ключа код вакансії, ніяк не від ключа код професії. А так в другій нормальній формі не повинно бути. Тому потрібно здійснити декомпозицію. Для цього спроектовано два відношення. Одне відношення, яке має назву вакансії присутні ось такі атрибути: код вакансії, фірма, місто, вулиця, заробітна плата, вимоги, умови праці, дата додання, код професії, назва, прізвище, ім'я, по батькові, місто, вулиця, вік, номер телефона, email, спеціальність. А друге відношення, яке має назву професія присутні ось такі атрибути: код професії, назва. І вже після цього виконується друга нормальна форма. Далі потрібно виконати третю нормальну форму. Спочатку повина виконуватись друга нормальна форма. Потім потрібно, щоб дані в таблиці залежали винятково від основного ключа. У відношенні вакансії фірма

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

залежить від ключа код вакансії, а ось номер телефона не залежить від цього ключа, а більше залежить від прізвища. Тому потрібно створити третє відношення, яке буде мати назву безробітні і до якого будуть вносити ось такі атрибути, які раніше належали таблиці вакансії: код безробітного, прізвище, ім'я, по батькові, місто, вулиця, вік, номер телефона, email, спеціальність, код вакансії. І вже після цього буде виконуватись третя нормальна форма. Цього вже достатньо для нормалізації даних. Далі можна побудувати інфологічну схему бази даних [КП.П09.ПІ.192.38.Д1]. Між таблицею вакансії та безробітні один до багатьох. Для того, щоб реалізувати ось цей зв'язок потрібно в таблицю безробітні додати поле код вакансії. І здійснити зв'язок між полем код з таблиці вакансії та код вакансії з таблиці безробітні. Між таблицею професії та вакансії також зв'язок один до багатьох.

Оскільки було зрозуміло, які в мене будуть таблиці в базі даних, то значить і можна спроектувати, які в мене будуть моделі. Для цього потрібно побудувати діаграму класів. На діаграмі класів [КП.П09.ПІ.192.38.Д4] представлено класи, які наявні в програмі та їх зв'язки між собою. Тобто було на діаграмі зображено лише ті класи, які мають бути створені в моделі.

У своєму проекті було використано три класи: “Вакансії”, “Професії” та “Безробітні”. Зв'язок між класом “Вакансії” та “Професії” асоціація і між класом “Вакансії” та “Безробітні” також асоціація. Відношення асоціації відповідає наявності деякого відношення між класами. Дане відношення позначається суцільною лінією з додатковими спеціальними символами, які характеризують окремі властивості конкретної асоціації.

В діаграмі компонентів [КП.П09.ПІ.192.38.Д7], вказується компоненти програмного забезпечення і зв'язки між собою. Тобто дана діаграма показує з яких частин складається дана програма та їх взаємодія на фізичному рівні [2]. Даний проєкт використовує компоненту класів, які використовуються у проєкті. Компонента PyCharm це компонента середовища розробки з

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

інтегрованими бібліотеками та функціями, які входять у фреймворк Django без яких програма працювати не зможе.

Великим значенням для розробки сайту є його інтерфейс. Він має бути інтуїтивно зрозумілим для користувача та мати гарний вигляд. Користувач, поглянувши на сайт має розуміти, що потрібно робити, щоб отримати результат виконання певної дії сайту. Сайти складаються з ось таких блоків: header, content, footer. Header – верхня частина сайту. Content – це дані, які наявні в сайті. Footer – нижня частина сайту. Content у сайті зазвичай змінюється динамічно при переході на різні сторінки сайту, чи при виконанні певних дій. А ось стилі можуть мати однаковий вигляд при переході на різні сторінки сайту або змінюватись. В моєму випадку стилі кожної сторінки сайту матимуть однаковий вигляд.

Рисунок 2.1 – Макет авторизації

Ось сторінка авторизації матиме такий вигляд.

Рисунок 2.2 – Макет реєстрації

Ось сторінка реєстрації матиме ось такий вигляд.

Рисунок 2.3 – Макет основної сторінки

Ось тут користувач може переглядати вакансії, здійснювати пошук за містом та заробітною платою і ще переглядати вакансії за певною професією. Як видно з макетів footer в мене буде однаковий, header за стилем в мене буде однаковим, однак будуть змінюватись меню на header. Наприклад якщо користувач зареєструється, то header вже буде мати ось такий вигляд.

Рисунок 2.4 – Макет header при реєстрації користувача

А ось content на кожній сторінці згідно створених макетів буде різним. Наприклад якщо я буду знаходитись на головній сторінці, то в мене не буде відображатись форма реєстрації, а будуть лише відображатись вакансії. Оскільки було визначено, які в будуть макети, то можна вважати, що спроектовано templates (шаблони). Далі можна спроектувати views (представлення). Для цього спочатку потрібно визначити, які будуть url адреса. Було визначено, що будуть ось такі url адреса:

```
path('', VacanciesHome.as_view(), name='home'),
path('about/', about, name='about'),
path('admin/', AddPage.as_view(), name='add_page'),
path('search/', Search.as_view(), name='search'),
path('sear/', Sear.as_view(), name='sear'),
```

```

path('contact/', contact, name='contact'),
path('login/', LoginUser.as_view(), name='login'),
path('logout/', logout_user, name='logout'),
path('register/', RegisterUser.as_view(), name='register'),
path('post/<int:post_id>', show_post, name='post'),
path('profession/<int:pr_id>', show_profession, name='profession')

```

Де в якості параметрів функції path буде: адрес, функція або клас представлення, назва адреси. У функціях або класах представлення буде реалізовуватись логіка сайту.

Далі я вирішив спроектувати можливі виконання дій зареєстрованим безробітним за виконанням певних умов. Для цього я побудував діаграму діяльності. Діаграма діяльності [КП.ПО9.ПІ.192.38.Д7], — візуальне представлення графу, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Тобто ця діаграма характеризує не лише зміну станів, а також уточнює деякі частини та елементи реалізації виконуваних команд [2].

Отже, в ось цьому пункті я визначив, який в мене буде архітектурний шаблон, спроектував базу даних

## 2.2. Розробка та опис програмної реалізації та тестування

Після того, як встановлено Python і PyChar потрібно встановити фреймворк Django. Але його потрібно встановлювати на віртуальному оточенні. Для того, щоб перейти на віртуальне оточення потрібно вказати команду в командному рядку операційної системи Windows “python venv venv”. Спочатку потрібно вибрати папку де буде встановлюватись віртуальне оточення. Запустити віртуальне оточення можна за допомогою команди “.\venv\Scripts\activate”. Для виходу з віртуального оточення потрібно набрати команду “deactivate”. Дії по відношенню до віртуального оточення можна здійснювати і в терміналі PyCharm. Ось це середовище автоматично відкриває віртуальне оточення. Далі вже можна здійснити встановлення фреймворку

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		



Django, знаходячись в кореневій папці проекту. Встановлення здійснюється за допомогою команди “`pip install django`”. Для того, щоб створити проект сайту потрібно ввести ось таку команду “`django-admin startproject coolsite`”. Де “coolsite” назва проекту. В нашому проекті є доволі важливий файл “`manage.py`”, за допомогою якого відбувається керування даним сайтом. Наприклад, можна створювати програми, міграції бази даних, запускати тестовий веб сервер і т.д. Це все буде відбуватися через ось цей файл. Для того, щоб запустити тестовий сервер потрібно спочатку перейти в папку “coolsite”. А вже потім можна запустити сервер можна за допомогою команди “`python manage.py runserver`”. Після цього запустився сервер і головна сторінка буде доступна за вказаним в терміналі адресом. Якщо натиснути на ось те посилання, то відкриється браузер з ось таким сайтом.

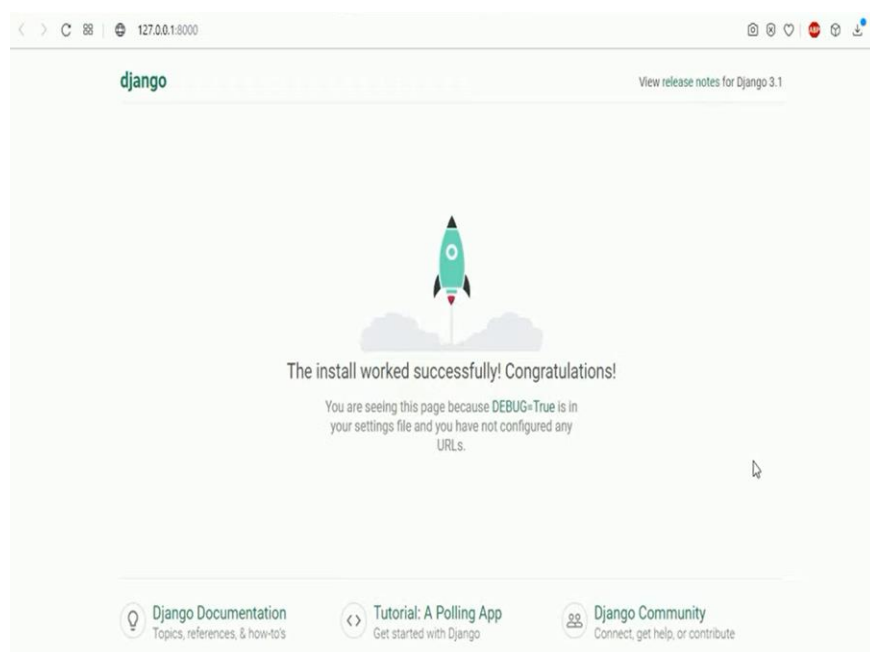


Рисунок 2.5 – Запуск сайту після встановлення Django

При цьому веб сервер автоматично перезапускається, якщо відбулися зміни в коді, що є доволі зручно. Потім потрібно створити програму з назвою “vacancies”. Я це зробив за допомогою команди “`python manage.py startapp vacancies`”. Після цього в мене з’явився каталог “vacancies”. Оскільки в цьому

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

каталозі є файл `__init__.py`. Це говорить про те, що в цьому каталозі є пакет. Потім потрібно його зареєструвати в проєкті нашого сайту. Щоб фреймворк Django знав про його існування і коректно працював. Для цього потрібно увійти в пакет конфігурації і відкрити файл `settings.py`. У цьому файлі є список `INSTALLED_APPS`. Ось тут прописуємо всі програми, які встановлені в нашому пакеті. Оскільки я створив програму, то потрібно додати в ось цей список. Список `INSTALLED_APPS` має ось такий вигляд:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'vacancies.apps.VacanciesConfig',
]
```

Далі я створив у файлі `models.py` створив класи. Клас “Вакансії” має ось такий вигляд:

```
class Vacancies(models.Model):
    posada=models.CharField(max_length=255, verbose_name="Посада")
    firma = models.CharField(max_length=255, verbose_name="Фірма")
    city=models.CharField(max_length=255,null=True, verbose_name="Місто")
    street=models.CharField(max_length=255,null=True, verbose_name="Вулиця")    vumogu =
models.TextField(blank=True, verbose_name="Вимоги")
    responsibilities=models.TextField(blank=True, verbose_name="Умови праці")
    time_create=models.DateTimeField(auto_now_add=True, verbose_name="Дата опублікування")
    pr = models.ForeignKey('Profession', on_delete=models.PROTECT, verbose_name="Професії")

    def __str__(self):
        return self.posada

    def get_absolute_url(self):
        return reverse('post', kwargs={'post_id': self.pk})
```

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class Meta:

    verbose_name = 'Вільні вакансії'
    verbose_name_plural = 'Вільні вакансії'
ordering = ['-time_create']

```

Клас “Професії” має ось такий вигляд:

```

class Profession(models.Model):
    name = models.CharField(max_length=100, db_index=True, verbose_name='Професія')

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('profession', kwargs={'pr_id': self.pk})

class Meta:
    verbose_name = 'Професія'
    verbose_name_plural = 'Професія'
ordering = ['id']

```

Клас “Безробітні” має ось такий вигляд:

```

class Unemployed(models.Model):
    priz=models.CharField(max_length=100, verbose_name="Прізвище")
    name = models.CharField(max_length=50, verbose_name="Ім'я")
    pobat=models.CharField(max_length=100, verbose_name="По батькові")
    city = models.CharField(max_length=100, verbose_name="Місто")
    street = models.CharField(max_length=50, verbose_name="Вулиця")
    year =models.IntegerField(verbose_name="Вік") nomer_phone=models.CharField(max_length=10,
verbose_name="Номер телефону")
    special=models.CharField(max_length=100, verbose_name="Спеціальність")
vc=models.ForeignKey('Vacancies',on_delete=models.PROTECT, verbose_name="Вакансії", null=True)

```

Далі після того, як створено моделі можна здійснити міграцію для того, щоб створились таблиці і в базі даних. Це можна здійснити за допомогою команди “python manage.py makemigrations” і потім я написав в терміналі ще одну команду “python manage.py migrate”. Після виконання цих команд

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

з'явилися таблиці в базі даних. Поки що вони порожні. Тому було вирішено додати дані в таблицю “Вакансії” через термінал. Спочатку потрібно написати команду “python manage.py shell”. Потім я здійснив виконання команди “from vacancies.models import Vacancies”. Для того, щоб була доступна модель “Вакансії”. Адже потрібно взаємодіяти з цією моделлю, а саме хочеться додати дані в таблицю “Vacancies”. Тому для того, щоб додати дані в ось цю таблицю потрібно написати ось таку команду “Vacancies() в дужках вказується поле, яке є в таблиці та його значення. Але не потрібно вказувати значення поля “time-create”, тому що дата додавання вакансії буде визначатись автоматично. Далі потрібно створювати екземпляр класу, використовуючи команду “w1 \_”. Потім треба здійснити викликання методу w1.save().

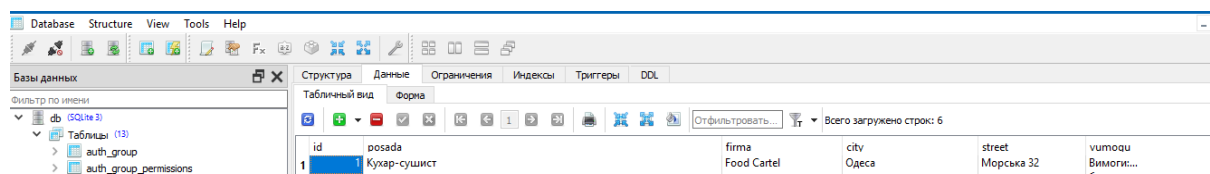


Рисунок 2.6 – Додані дані в таблиці

Після того, як вже додані дані в таблицю “Вакансії” потрібно відобразити на сайті. Для цього потрібно використовувати “templates”, яка призначена для відображення інтерфейсів для користувачів. Тому я в пакеті “vacancies” я створив папку “templates”. Далі в папці “templates” я створив папку “vacancies”. Далі в ось цій папці можна створювати html файли. Де буде відображатись інформація на сторінці. Створено urls.py у пакеті “Vacancies”. Тут будуть знаходитись маршрути. Це тому що коли на сторінку приходить запит він пропускається через блок маршрутизації. Фіксується URL адрес і в списку шаблонів шукається перше співпадіння. І якщо знаходиться співпадіння по шаблону та активізується представлення для обробки цього запиту. Якщо жоден зі шаблонів не співпадає, то генерується помилка 404.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Представлення бере шаблон наповнює цей шаблон конкретною інформацією тобто з моделі і в кінцевому випадку вийде html сторінка, яка відається кінцевому користувачу. Тому файл urls.py містить ось такий вигляд:

```
from django.urls import path, re_path
from . import views

from .views import *

urlpatterns = [
    path("", VacanciesHome.as_view(), name='home'),
    path('about/', about, name='about'),
    path('admin/', AddPage.as_view(), name='add_page'),
    path('search/', Search.as_view(), name='search'),

    path('contact/', contact, name='contact'),
    path('login/', LoginUser.as_view(), name='login'),
    path('logout/', logout_user, name='logout'),
    path('register/', RegisterUser.as_view(), name='register'),
    path('post/<int:post_id>/', show_post, name='post'),
    path('profession/<int:pr_id>/', show_profession, name='profession')
]
```

У пакеті “vacancies” є файл views.py. Призначений для збереження представлень даного проекту. Представлення у Django можна реалізовувати як у вигляді функції так і у вигляді класів. Приклад реалізації представлення у вигляді класу:

```
class VacanciesHome(DataMixin, ListView):
    paginate_by = 3
    model = Vacancies
    template_name = 'vacancies/index.html'
    context_object_name = 'posts'
    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['user_menu'] = menu.copy()
        context['title'] = "Головна сторінка"
```

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if not self.request.user.is_authenticated:

    user_menu.pop(2)

    context['menu'] = user_menu

    if not self.request.user.is_superuser:
        user_menu.pop(1)
        context['menu'] = user_menu
    return dict(list(context.items()) + list(c_def.items()))

```

### Приклад реалізації представлення у вигляді функції:

```

def contact(request):

    if request.method == 'POST':
        form = UnemployedF(request.POST)

        user_menu = menu.copy()
        if not request.user.is_superuser:
            user_menu.pop(1)
        if form.is_valid():
            # print(form.cleaned_data)

    try:
        Unemployed.objects.create(**form.cleaned_data)

        return redirect('home')

    except:
        form.add_error(None, 'Помилка додавання')

    else:
        form = UnemployedF()

    return render(request, 'vacancies/contact.html', {'form': form, 'menu': menu, 'title': 'Додавання даних про користувача'})

```

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Функція `render` є вбудованим шаблонізатором. І саме вона виконує обробку наших шаблонів. Файли шаблонів будуть знаходитись у пакеті `'vacancies'`. У тільки що створеній папці `templates`. Але у папці `templates` я створив папку `vacancies`. І саме в цій папці будуть знаходитись файли шаблонів. Наприклад шаблон `contact.html` має ось такий вигляд:

```
{% extends 'vacancies/base.html' %}

{% block vumogu %}

{% if request.user.is_superuser %}

<h1>{{ title }}</h1>

{% endif %}

<form action="{% url 'contact' %}" method="post">

    {% csrf_token %}


<div class="form-error">{{ form.non_field_errors }}</div>

{% for f in form %}

    <p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label }}: </label>{{ f }}</p>

    <div class="form-error">{{ f.errors }}</div>


{% endfor %}

<button type="submit">Добавити</button>

</form>

{% endblock %}
```

Тобто шаблоном вважаються `html` файли з використанням стилів `CSS`. А у представлені просто викликається ось цей шаблон за вказаним адресом. При чому в мене цей шаблон, як і інші, окрім, `base.html` наслідується від шаблону `base.html`. Це потрібно для того, щоб в при створені кожного шаблону не повторювався код, що є в `base.html`. До шаблонів можна підключати статичні файли `CSS`, `JavaScript` і т.д. У пакеті `"vacancies"` я створив папку `static`. А в цьому каталозі я створив папку `vacancies`. Це для того, щоб не виникав конфлікт імен. А в цьому каталозі я створив папку `css`. Для збереження каскадних таблиць стилів. Можна створити ще папку `js`. Але я не буду використовувати файли `JavaScript`. Ну і ще я створив каталог `images` для

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

зберігання зображення. У папці css я створив файл style.css. Ось цей файл виглядає ось таким чином:

Тестування є важливим етапом розробки програмного забезпечення. Саме на цьому етапі можна зрозуміти чому виникають помилки та знайти способи їх виправлення. Тестування також дозволяє виявляти помилки під час роботи, які можуть виникнути на етапі проектування та написання коду. Під час тестування перевіряється, як буде поводити сайт при введенні різних текстових даних у текстове поле.

### Реєстрація

Логін:	<input type="text" value="VladTartakov"/>
Email:	<input type="text" value="vlad3@gmail.com"/>
Пароль:	<input type="password" value="****"/>
Повтор пароля:	<input type="password" value="****"/>

Пароль надто короткий. Він повинен містити як мінімум 8 символів  
Цей пароль повністю складається із цифр.

### Рисунок 2.7 – Результат введення занадто короткого пароля

Ось тут переглянуто, що буде якщо я введу замалий пароль. Як видно зі скріна було виведено відповідне повідомлення. При створенні форми для відправки даних потрібно здійснити валідацію даних.

```
class UnemployedF(forms.Form):
    priz = forms.CharField(max_length=100, label="Прізвище")
    name = forms.CharField(max_length=50, label="Ім'я")
    pobat = forms.CharField(max_length=100, label="По батькові")
    city = forms.CharField(max_length=100, label="Місто")
    street = forms.CharField(max_length=50, label="Вулиця")
    nomer_phone = forms.CharField(max_length=10, label="Номер телефону")
    year = forms.IntegerField(label="Вік")

    special = forms.CharField(max_length=100, label="Спеціальність")
    vc = forms.ModelChoiceField(queryset=Vacancies.objects.all(), label="Вакансії")
```

Тобто біля кожного поля, окрім поля year та vc, я вказав максимальну довжину

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		



рядка. При введенні даних у формі користувачу буде неможливо ввести рядок який буде більшим за вказану довжину рядка. Було створено власну валідацію для поля year та номер\_phone.

Для поля year було здійснено перевірку на коректність введення віку безробітних. Вік може бути більшим рівним 14 і меншим рівним 80 років.

Ось ця перевірка відбувається таким чином:

```
def clean_year(self):  
  
    year = self.cleaned_data['year']  
    if year<=14 or year>=80:  
        raise ValidationError('Некоректно введено вік безробітного')  
  
    return year
```

Після створення власних валідацій було вирішено переглянути як працює програма.

Прізвище:

Ім'я:

По батькові:

Місто:

Вулиця:

Номер телефону:

Вік:

Некоректно введено вік безробітного

Спеціальність:

Вакансії:

Рисунок 2.8 – Результат ведення не коректного віку

Згідно цього скріна видно, що перевірка відбувається і коли користувач вводить вік, який не відповідає заданій умові, то з'являється повідомлення про те, що ввели не коректно дані. А якщо користувач б ввів коректна значення віку, то тоді б дані про безробітного записалися в базі даних. Під час тестування було виявлено певні незручності.

Змінено:

- шрифт і розмір тексту;
- добавив плагінацію з можливістю розміщення лише 3 вакансії на одній сторінці, якщо користувач переглядає всі наявні вакансії;

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

- здійснив адаптацію сайту;

Отже на даному етапі я здійснив реалізацію програмної частини та тестування готового програмного продукту.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

### 3. Спеціальний розділ

#### 3.1. Інструкція з інсталяції розробленого проекту

“Біржа праці” реалізована у вигляді модель-представлення-контролер. Клієнтом виступатиме браузер будь-якого пристрою, який має доступ до серверу. Для встановлення серверної частини необхідно мати фізичний або хмарний сервер, який відповідатиме таким вимогам:

- рекомендована – Windows 7 і вище;
- об’єм вільного простору – від 410 Мб;
- оперативна пам’ять – від 700 Мб.

Діаграма розгортання [КП.П09.ПІ.192.38.Д6] - застосовується для уявлення загальної топології розподіленої програмної системи з наявністю компонентів у вигляді вузлів з фізичними зв’язками, які забезпечують передачу інформації між пристроями [2].

Перед інсталяцією проекту необхідно встановити SQLite. Для цього потрібно перейти на офіційний сайт SQLite.

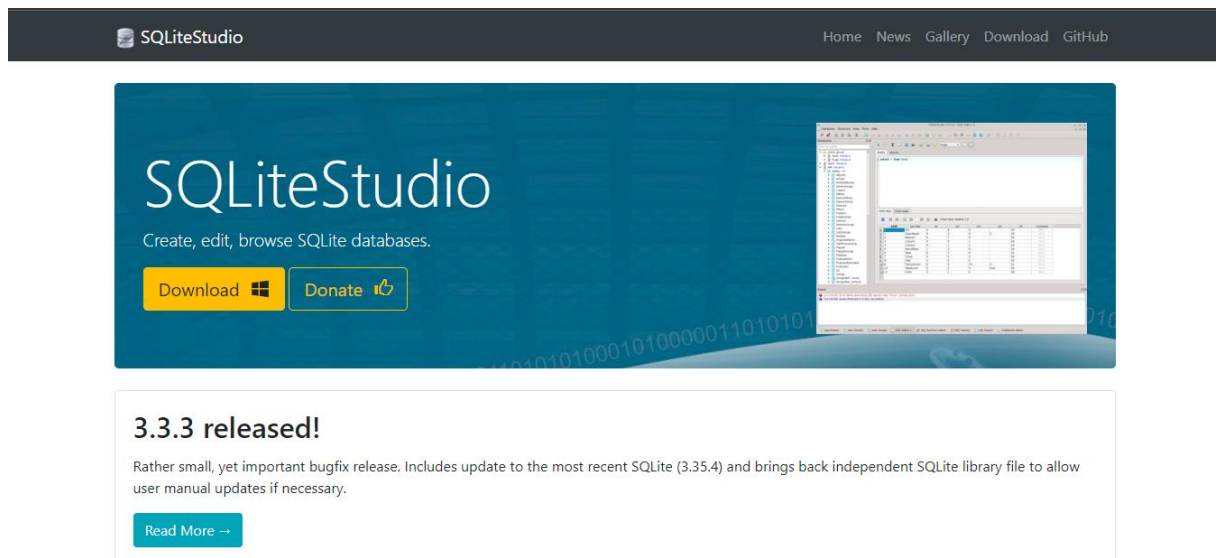


Рисунок 3.1 – Офіційний сайт SQLite

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі при запуску PyCharm потрібно перейти на папку програми у терміналі і запустити локальний сервер за допомогою команди “python manage.py runserver”. І після переходу за вказаним URL адресом з’явиться головна сторінка.

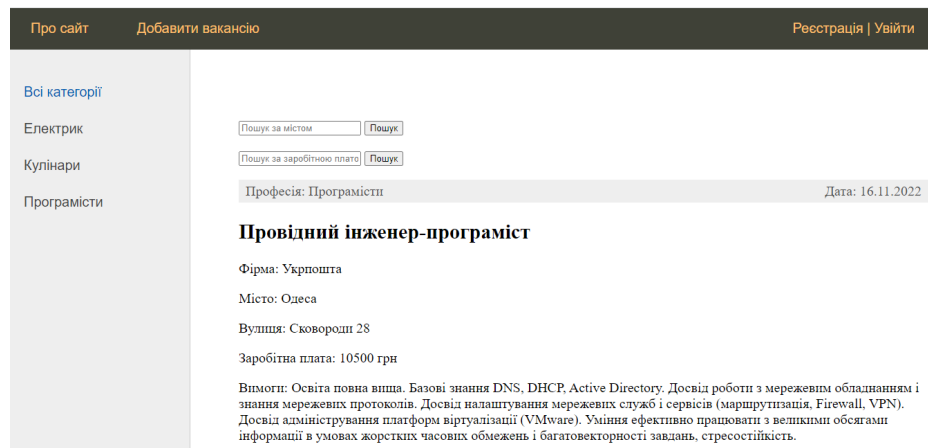


Рисунок 3.2 – Головна сторінка сайту

Отже, на цьому етапі я запустив сайт на локальному сервері.

### 3.2. Інструкція з експлуатації проекту

Для роботи з системою можна перейти на сайт і не реєструватись та авторизуватись. Але тоді можна просто переглядати наявні вакансії, здійснювати пошук за містом знаходженням робочого місця та переглядати вакансії за певною професією.

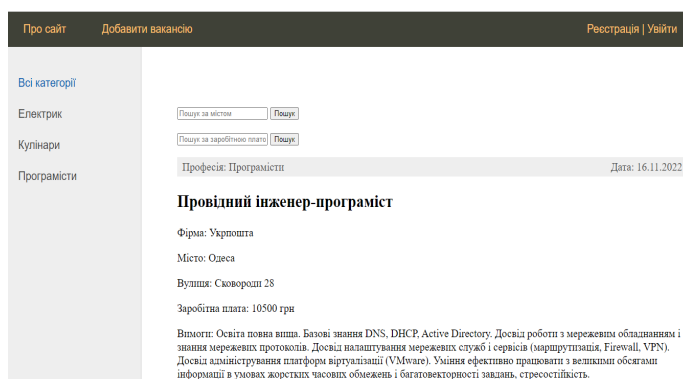


Рисунок 3.3 – Головна сторінка сайту

Ось тут відображено головну сторінку сайту.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Про сайт

Добавити вакансію

Рєєстрація | Увійти

Всі категорії

Електрик

Кулінари

Програмісти

Пошук за містом

Пошук

Пошук за заробітною платою

Пошук

Професія: Електрик

Дата: 06.11.2022

Електрик, інсталятор систем електропостачання

Фірма: Територія Сервісу

Місто: Івано-Франківськ

Вулиця: Грушевського 22

Заробітна плата: 7000 грн

Вимоги: технічна освіта (бажано профільна) допуск до 1000 V досвід електриком від 5 років розуміння усіх процесів сучасного монтажу електропостачання пунктуальність, відповідальність, організованість відсутність шкідливих звичок, які впливають на якість роботи бажання працювати і заробляти

Пропонуємо: високий рівень доходів роботу в сучасній прогресивній компанії додаткові навчання

Рисунок 3.4 – Здійснення пошуку за містом Івано-Франківськ  
Продemonстровано як працює пошук.

Про сайт

Добавити вакансію

Відгукнутись на вакансію

rootadmin | Вийти

Всі категорії

Електрик

Кулінари

Програмісти

Пошук за містом знаходження роботи

Пошук

Пошук

Професія: Кулінари

Дата: 05.11.2022

Кухар-сушист

Food Cartel

Одеса

Морська 32

Вимоги: бажання працювати витривалість відданість справі

Обов'язки: готувати відповідно до стандартів

Рисунок 3.5 – Перегляд вакансій для кулінарів

Якщо користувач зарєєструється або авторизується, то він може ще написати дані про себе і відгукнутись на вакансію. Ці дані, які користувач введе може записати в базу даних.

Про сайт	Добавити вакансію	Відгукнутись на вакансію	VladTartakovsky   Вийти
Всі категорії Електрик Кулінари Програмісти	Прізвище: <input type="text"/> Ім'я: <input type="text"/> По батькові: <input type="text"/> Місто: <input type="text"/> Вулиця: <input type="text"/> Номер телефону: <input type="text"/> Вік: <input type="text"/> Спеціальність: <input type="text"/> Вакансії: <input type="text"/>	<input type="button" value="Добавити"/>	

Рисунок 3.6 – В ось цій формі можна написати дані про себе

І адміністратор може переглянути ці дані і зателефонувати безробітному або проаналізувати введену інформацію. Якщо увійти в систему, як адміністратор, то можна додавати, редагувати та видаляти дані про вакансії, професії та дані безробітних за власним бажанням. Ще будь які користувачі можуть переглянути, як працює сайт для цього достатньо перейти на вкладку “Про сайт”.

Про сайт	Добавити вакансію	Відгукнутись на вакансію	VladTartakovsky   Вийти
Всі категорії Електрик Кулінари Програмісти	Інформація про сайт  Сайт призначений для розміщення вакансій, які потребують знаходження працівників. А безробітні, які шукають роботу можуть знайти відповідну і зв'язатись з роботодавцем. Користувач може не реєструватись і авторизуватись, але тоді він може лише переглядати вакансії, якщо він зареєструється, то може ще вибрати вакансію і написати дані про себе. Адміністратор перегляне ваші дані і зателефонує вам.		

Рисунок 3.7 – Перегляд як працює сайт

Ось тут продемонстровано як працює сайт у текстовій формі.

Отже на цьому етапі я продемонстрував, як працює сайт.

## Висновки

Під час виконання курсового проекту було створено програму “Біржа праці”. Даний проект був створений з використанням фреймворку Django на мові програмування Python. Перед створенням програми була вивчена предметна область. Далі перегляд схожих програми, які розробили інші розробники і визначення їх недоліків. Потім я побудував UML діаграми для визначення функціональних можливостей і кращого розуміння предметної області. Для реалізації цього продукту було використана середовище PyCharm.

Важливим етапом для розробки “Біржа праці” було тестування. Саме під час тестування я виявив деякі помилки і зрозумів чому вони виникали. Також на цьому етапі я оптимізував код даного проекту. В результаті виконанні цих дій я зумів реалізувати цей проект. Переваги мого проекту:

- нескладний інтерфейс;
- доступний для будь якої операційної системи де наявний браузер;
- безпечний сайт;
- не потрібно встановлювати так як десктопні чи мобільні додатки.
- Недоліки мого проекту:
- потрібний доступ до мережі інтернет;
- занадто простий стиль сайту;
- може добре працювати лише при невеликому навантаженню.

Отже, після виконання усіх етапів розробки програмного забезпечення за порядком, вдалося створити “Біржа праці”.

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

### Список використаних джерел

1. Беррі Пол. Вивчаємо програмування на Python. – 2-ге видання, 2021 – 520 с..
2. Буч Г., Блаха М. – UML 2.0 Об'єктно-орієнтоване моделювання і розробка. – Питер, 2005 – 545 с.
3. Доусон Майкл. Програмуємо на Python. – Питер, 2014 – 416 с.
4. Дронов Володимир. – Django практика створення Web-сайтів на Python– bhv, 2010 - 480 с.
5. Осипов Д. Л. – Технології проектування баз даних – ДМК-Пресс, 2012 – 720 с.
6. <https://codecademy.com/catalog/language/python>.-Вивчення Python [Електронний ресурс].
7. <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/>-Веб-фреймворк Django (Python) [Електронний ресурс].
8. <https://learnpython.org/> Інтерактивне видання по Python [Електронний ресурс].
9. <https://metanit.com/> Інтернет-ресурс для вивчення мови програмування Python [Електронний ресурс].
10. <https://ror-development.com/uk/8-osnovnyh-mov-dlya-bekenda/> Основні мови для бекенду [Електронний ресурс].
11. ДСТУ 3008-95 «Документація. Звіти в сфері науки і техніки. Структура і правила оформлення» - Державний стандарт України

					КП.ПО.09.ПІ-192.38.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34



## Додаток А код програми

### models.py

```
from django.db import models

from django.urls import reverse

class Vacancies(models.Model):

    posada = models.CharField(max_length=255, verbose_name="Посада")

    firma = models.CharField(max_length=255, verbose_name="Фірма")

    city = models.CharField(max_length=255, null=True, verbose_name="Місто")

    street = models.CharField(max_length=255, null=True, verbose_name="Вулиця")

    vumogu = models.TextField(blank=True, verbose_name="Вимоги")

    responsibilities = models.TextField(blank=True, verbose_name="Умови праці")

    time_create = models.DateTimeField(auto_now_add=True, verbose_name="Дата опублікування")

    pr = models.ForeignKey('Profession', on_delete=models.PROTECT, verbose_name="Професії")

    def __str__(self):

        return self.posada

    def get_absolute_url(self):

        return reverse('post', kwargs={'post_id': self.pk})

    class Meta:

        verbose_name = 'Вільні вакансії'

        verbose_name_plural = 'Вільні вакансії'

        ordering = ['-time_create']

class Profession(models.Model):

    name = models.CharField(max_length=100, db_index=True, verbose_name='Професія')

    def __str__(self):

        return self.name

    def get_absolute_url(self):

        return reverse('profession', kwargs={'pr_id': self.pk})
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class Meta:

    verbose_name = 'Професія'

    verbose_name_plural = 'Професія'

    ordering = ['id']


class Unemployed(models.Model):

    priz = models.CharField(max_length=100, verbose_name="Прізвище")

    name = models.CharField(max_length=50, verbose_name="Ім'я")

    pobat = models.CharField(max_length=100, verbose_name="По батькові")

    city = models.CharField(max_length=100, verbose_name="Місто")

    street = models.CharField(max_length=50, verbose_name="Вулиця")

    year = models.IntegerField(verbose_name="Рік")

    nomer_phone = models.CharField(max_length=10, verbose_name="Номер телефону")

    special = models.CharField(max_length=100, verbose_name="Спеціальність")

    vc = models.ForeignKey('Vacancies', on_delete=models.PROTECT, verbose_name="Вакансії",
null=True)

```

## urls.py

```

from django.urls import path, re_path

from . import views

from .views import *

urlpatterns = [

    path("", VacanciesHome.as_view(), name='home'),

    path('about/', about, name='about'),

    path('admin/', AddPage.as_view(), name='add_page'),

    path('search/', Search.as_view(), name='search'),

    path('contact/', contact, name='contact'),

    path('login/', LoginUser.as_view(), name='login'),

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

```

path('logout/', logout_user, name='logout'),

path('register/', RegisterUser.as_view(), name='register'),

path('post/<int:post_id>/', show_post, name='post'),

path('profession/<int:pr_id>/', show_profession, name='profession')

]

```

## views.py

```

from django.contrib.auth import logout, login

from django.contrib.auth.views import LoginView

from django.http import HttpResponse, HttpResponseNotFound, Http404

from django.shortcuts import render, redirect, get_object_or_404

from django.urls import reverse_lazy

from django.views.generic import ListView, CreateView

from .forms import AddPostForm, RegisterUserForm, LoginUserForm, UnemployedF

from .utils import *

class VacanciesHome(DataMixin, ListView):

    paginate_by = 3

    model = Vacancies

    template_name = 'vacancies/index.html'

    context_object_name = 'posts'

    def get_context_data(self, *, object_list=None, **kwargs):

        context = super().get_context_data(**kwargs)

        c_def = self.get_user_context(title="Головна сторінка")

        user_menu = menu.copy()

        if not self.request.user.is_authenticated:

            user_menu.pop(2)

        context['menu'] = user_menu

        if not self.request.user.is_superuser:

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        user_menu.pop(1)

        context['menu'] = user_menu

        return dict(list(context.items()) + list(c_def.items()))

def about(request):

    user_menu = menu.copy()

    if not request.user.is_authenticated:

        user_menu.pop(2)

    return render(request, 'vacancies/about.html', {'menu': user_menu, 'title': 'Про сайт'})

class AddPage(CreateView):

    form_class = AddPostForm

    template_name = 'vacancies/post.html'

    def get_context_data(self, *, object_list=None, **kwargs):

        context = super().get_context_data(**kwargs)

        context['title'] = 'Добавлення вакансії'

        context['menu'] = menu

        return context

def contact(request):

    if request.method == 'POST':

        form = UnemployedF(request.POST)

        user_menu = menu.copy()

        if not request.user.is_superuser:

            user_menu.pop(1)

        if form.is_valid():

            try:

                Unemployed.objects.create(**form.cleaned_data)

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return redirect('home')

    except:

        form.add_error(None, 'Помилка додавання')

    else:

        form = UnemployedF()

    return render(request, 'vacancies/contact.html', {'form': form, 'menu': menu, 'title': 'Додавання даних про користувача'})

def pageNotFound(request, exception):

    return HttpResponseNotFound('<h1>Сторінка не знайдена</h1>')

def show_post(request, post_id):

    post = get_object_or_404(Vacancies, pk=post_id)

    context = {

        'post': post,

        'menu': menu,

        'title': post.posada,

        'cat_selected': post.pr_id,

    }

    return render(request, 'vacancies/post.html', {'title': 'Додати дані про себе'})

def show_profession(request, pr_id):

    posts = Vacancies.objects.filter(pr_id=pr_id)

    user_menu = menu.copy()

    if not request.user.is_authenticated:

        user_menu.pop(2)

    context = {

        'posts': posts,

        'menu': user_menu,

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'title': 'Відображення по професіях',

        'cat_selected': pr_id,

    }

    return render(request, 'vacancies/index.html', context=context)

class RegisterUser(DataMixin, CreateView):

    form_class = RegisterUserForm

    template_name = 'vacancies/register.html'

    success_url = reverse_lazy('login')

    def get_context_data(self, *, object_list=None, **kwargs):

        context = super().get_context_data(**kwargs)

        c_def = self.get_user_context(title="Реєстрація")

        return dict(list(context.items()) + list(c_def.items()))

    def form_valid(self, form):

        user = form.save()

        login(self.request, user)

        return redirect('home')

class LoginUser(DataMixin, LoginView):

    form_class = LoginUserForm

    template_name = 'vacancies/login.html'

    def get_context_data(self, *, object_list=None, **kwargs):

        context = super().get_context_data(**kwargs)

        c_def = self.get_user_context(title="Авторизація")

        return dict(list(context.items())+list(c_def.items()))

    def get_success_url(self):

        return reverse_lazy('home')

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def logout_user(request):
```

```
    logout(request)
```

```
    return redirect('login')
```

```
class Search(ListView):
```

```
    model = Vacancies
```

```
    template_name = 'vacancies/index.html'
```

```
    context_object_name = 'posts'
```

```
    def get_queryset(self):
```

```
        return Vacancies.objects.filter(city=self.request.GET.get("search"))
```

```
    def get_context_data(self, *args, **kwargs):
```

```
        context = super().get_context_data(*args, **kwargs)
```

```
        context["search"] = self.request.GET.get("search")
```

```
        user_menu = menu.copy()
```

```
        if not self.request.user.is_superuser:
```

```
            user_menu.pop(2)
```

```
        context['menu'] = user_menu
```

```
        return context
```

## forms.py

```
from django import forms
```

```
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
```

```
from django.contrib.auth.models import User
```

```
from .models import *
```

```
class AddPostForm(forms.ModelForm):
```

```
    class Meta:
```

```
        model = Vacancies
```

```
        fields = ['posada', 'firma', 'city', 'street', 'vumogu', 'responsibilities', 'pr']
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

widgets = {

    'posada': forms.TextInput(attrs={'class': 'form-input'}),

    'vumogu': forms.Textarea(attrs={'cols': 60, 'rows': 10})

}

class RegisterUserForm(UserCreationForm):

    username = forms.CharField(label='Ім'я', widget=forms.TextInput(attrs={'class': 'form-input'}))

    email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={'class': 'form-input'}))

    password1 = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class': 'form-input'}))

    password2 = forms.CharField(label='Повтор пароля', widget=forms.PasswordInput(attrs={'class': 'form-
input'})))

    class Meta:

        model = User

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

class LoginUserForm(AuthenticationForm):

    username = forms.CharField(label='Ім'я', widget=forms.TextInput(attrs={'class': 'form-input'}))

    password = forms.CharField(label='Пароль', widget=forms.PasswordInput(attrs={'class': 'form-input'}))

class UnemployedF(forms.Form):

    priz = forms.CharField(max_length=100, label="Прізвище")

    name = forms.CharField(max_length=50, label="Ім'я")

    pobat = forms.CharField(max_length=100, label="По батькові")

    city = forms.CharField(max_length=100, label="Місто")

    street = forms.CharField(max_length=50, label="Вулиця")

    nomer_phone = forms.CharField(max_length=10, label="Номер телефону")

    year = forms.IntegerField(label="Вік")

    special = forms.CharField(max_length=100, label="Спеціальність")

    vc = forms.ModelChoiceField(queryset=Vacancies.objects.all(), label="Вакансії")

```

## apps.py

```
from django.apps import AppConfig
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		



```
class VacanciesConfig(AppConfig):

    default_auto_field = 'django.db.models.BigAutoField'

    name = 'vacancies'

    verbose_name = 'Вакансії:'
```

## admin.py

```
from django.contrib import admin

from .models import *

class VacanciesAdmin(admin.ModelAdmin):

    list_display = ('id', 'posada', 'firma', 'city', 'street', 'vumogu', 'responsibilities', 'time_create', 'pr')

    list_display_links = ('id', 'posada')

    search_fields = ('posada', 'city')

class ProfessionAdmin(admin.ModelAdmin):

    list_display = ('id', 'name')

    list_display_links = ('id', 'name')

    search_fields = ('name', )

class UnemployedAdmin(admin.ModelAdmin):

    list_display = ('id', 'priz', 'name', 'pobat', 'city', 'street', 'year', 'nomer_phone', 'special')

    list_display_links = ('id', 'priz')

    search_fields = ('priz', )

admin.site.register(Vacancies, VacanciesAdmin)

admin.site.register(Profession, ProfessionAdmin)

admin.site.register(Unemployed, UnemployedAdmin)
```

## utils.py

```
from .models import *

menu = [{ 'title': "Про сайт", 'url_name': 'about'},

        { 'title': "Добавити вакансію", 'url_name': 'add_page'},

        { 'title': "Відгукнутись на вакансію", 'url_name': 'contact'},

        ]

class DataMixin:

    def get_user_context(self, **kwargs):

        context = kwargs
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

cats = Profession.objects.all()

user_menu = menu.copy()

if not self.request.user.is_authenticated:

    user_menu.pop(2)

context['menu'] = user_menu

context['cats'] = cats


if 'cat_selected' not in context:

    context['cat_selected'] = 0

return context

```

## about.html

```

{% extends 'vacancies/base.html' %}

{% block vumogu %}

<h1>{{ posada }}</h1>

<p>Інформація про сайт</p>

<p>Сайт призначений для розміщення вакансій, які потребують знаходження працівників. А безробітні, які шукають роботу можуть знайти відповідну і зв'язатись з роботодавцем. Користувач може не реєструватись і авторизуватись, але тоді він може лише переглядати вакансії, якщо він зареєструється, то може ще вибрати вакансію і написати дані про себе. Адміністратор перегляне ваші дані і зателефонує вам. </p>

{% endblock %}

```

## base.html

```

{% load static %}

{% load vacancies_tag %}

<!DOCTYPE html>

<html lang="en">

<head>

    <title>{{ title }}</title>

    <link type="text/css" href="{% static 'vacancies/css/styles.css' %}" rel="stylesheet" />

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

    <!--    <link rel="shortcut icon" href="{% static 'vacancies/images/main.ico' %}" type="image/x-icon"/>-->

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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

</head>

<body>

<table class="table-page" border=0 cellpadding="0" cellspacing="0">

<tr><td valign=top>

{% block mainmenu %}

        <div class="header">

                <ul id="mainmenu" class="mainmenu">

{% for m in menu %}

<li><a href="{% url m.url_name %}">{{ m.title }}</a></li>

{% endfor %}

                                {% if request.user.is_authenticated %}

                                <li class="last"> {{ user.username }} | <a href="{% url 'logout'

                                {%}">Вийти</a></li>

                                {% else %}

                                <li class="last"> <a href="{% url 'register' %}">Реєстрація</a> | <a

                                href="{% url 'login' %}">Увійти</a></li>

                                {% endif %}

                </ul>

        </div>

{% endblock mainmenu %}

<table class="table-content" border=0 cellpadding="0" cellspacing="0">

<tr>

        <td valign="top" class="left-chapters">

                <ul id="leftchapters">

                        {% if cat_selected == 0 %}

                        <li class="selected">Всі категорії</li>

                        {% else %}

                        <li><a href="{% url 'home' %}">Всі категорії</a></li>

                        {% endif %}

                {% show_profession 'name' pr_selected %}

                </ul>

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</td>

<td valign="top" class="content">

    { % block breadcrumbs % }

    { % endblock % }

    <div class="content-text">

        <p> </p>

    { % block vumogu % }

    { % endblock % }

    { % if page_obj.has_other_pages % }

    <nav class="list-pages">

        <ul>

            { % if page_obj.has_previous % }

            <li class="page-num">

                <a href="?page={{ page_obj.previous_page_number }}">&lt;</a>

            </li>

            { % endif % }

            { % for p in paginator.page_range % }

                { % if page_obj.number == p % }

                <li class="page-num page-num-selected">{{ p }}</li>

                { % elif p >= page_obj.number|add:-2 and p <= page_obj.number|add:2 % }

                <li class="page-num">

                    <a href="?page={{ p }}">{{ p }}</a>

                </li>

                { % endif % }

            { % endfor % }

            { % if page_obj.has_next % }

            <li class="page-num">

                <a href="?page={{ page_obj.next_page_number }}">&gt;</a>

            </li>

            { % endif % }

        </ul>

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</nav>

{% endif %}

</div>

</td></tr></table>

</td></tr>

<tr><td valign=top>

    <div id="footer">

        <p>&copy; 2022 Вільні вакансії</p>

    </div>

</td></tr></table>

</body>

</html>

```

## contact.html

```

{% extends 'vacancies/base.html' %}

{% block vumogu %}

{% if request.user.is_superuser %}

<h1>{{ title }}</h1>

{% endif %}

<form action="{% url 'contact' %}" method="post">

    {% csrf_token %}

    <div class="form-error">{{ form.non_field_errors }}</div>

    {% for f in form %}

    <p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label }}: </label>{{ f }}</p>

    <div class="form-error">{{ f.errors }}</div>

    {% endfor %}

    <button type="submit">Добавити</button>

</form>

{% endblock %}

```

## index.html

```

{% extends 'vacancies/base.html' %}

{% block vumogu %}

<ul class="list-articles">

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<p> Пошук за містом знаходження роботи </p>

<form class="form-inline my-2 my-lg-0" action="{% url 'search' %}" method="get">

  <input class="form-control mr-sm-2" type="search" placeholder="Пошук" aria-label="Пошук",
name="search">

  <button class="btn btn-online-succes my -2 my -sm -0" type="submit">Пошук</button>

</form>

<p> </p>

{% for p in posts %}

<li><div class="article-panel">

  <p class="first">Професія: {{p.pr}}</p>

  <p class="last">Дата: {{p.time_create|date:"d.m.Y"}}</p>

</div>

  <h2>{{p.posada}}</h2>

  {% autoescape on %}

  {{p.firma|linebreaks|truncatewords:50}}

  {{p.city|linebreaks|truncatewords:50}}

  {{p.street|linebreaks|truncatewords:50}}

  {{p.vumogu|linebreaks|truncatewords:50}}

  {{p.responsibilities|linebreaks|truncatewords:50}}

  {% endautoescape %}

  <div class="clear"></div>

</li>

{% endfor %}

</ul>

{% endblock %}

```

## list\_categories.html

```

{% for pr in cats %}

  {% if pr.pk == cat_selected %}

    <li class="selected">{{pr.name}}</li>

  {% else %}

    <li><a href="{% pr.get_absolute_url %}">{{pr.name}}</a> </li>

  {% endif %}


```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

{% endfor %}

## login.html

```
{% extends 'vacancies/base.html' %}

{% block vumogu %}

<h1>{{ title }}</h1>

<form method="post">

    {% csrf_token %}

    <div class="form-error">{{ form.non_field_errors }}</div>

{% for f in form %}

<p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label }}: </label>{{ f }}</p>

<div class="form-error">{{ f.errors }}</div>

{% endfor %}

    <button type="submit">Увійти</button>

</form>

{% endblock %}
```

## post.html

```
{% extends 'vacancies/base.html' %}

{% block vumogu %}

<h1>{{ title }}</h1>

{{ post.firma|linebreaks }}

<form action="{{ url 'about' }}" method="post">

    {% csrf_token %}

    <div class="form-error">{{ form.non_field_errors }}</div>

{% for f in form %}

    <p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label }}: </label>{{ f }}</p>

<div class="form-error">{{ f.errors }}</div>

    {% endfor %}

    <button type="submit">Добавити</button>

</form>

{% endblock %}
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

## register.html

```
{% extends 'vacancies/base.html' %}

{% block vumogu %}

<h1>{{ title }}</h1>

<form method="post">

    {% csrf_token %}

    {% for f in form %}

    <p><label class="form-label" for="{{ f.id_for_label }}">{{ f.label }}: </label>{{ f }}</p>

    <div class="form-error">{{ f.errors }}</div>

    {% endfor %}

    <button type="submit">Реєстрація</button>

</form>

{% endblock %}
```

## style.css

```
html, body {

    font-family: 'Arial';

    margin: 0;

    padding: 0;

    height: 100%;

    width: 100%;

    color: #444;

}

a {

    color: #0059b2;

    text-decoration: none;

}

a:hover {

    color: #CC0000;

    text-decoration: underline;

}

img {max-width: 600px; height: auto;}
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		



```
img.img-article-left {
    max-width: 300px;
    height: auto;
    float: left;
    padding: 0 10px 10px 0;
}
```

```
img.img-article-left.thumb {
    max-width: 150px;
    height: auto;
}
```

```
p.link-read-post {
    text-align: right;
}
```

```
p.link-read-post a {
    padding: 10px;
    min-width: 100px;
    background: #333671;
    color: #fff;
    text-decoration: none;
}
```

```
div.article-panel {
    background: #eee;
    padding: 5px 10px 5px 10px;
    box-sizing: border-box;
    overflow: hidden;
    color: #555;
}
```

```
div.article-panel p.first {
    margin: 0;
    float: left;
}
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

div.article-panel p.last {
    margin: 0;
    float: right;
}

ul.list-articles {
    list-style: none;
    margin: 0;
    padding: 0;
}
ul.list-articles li {
    border-bottom: 1px solid #ccc;
}

.table-page {
    width: 100%;
    height: 100%;
    vertical-align: top;
}

.table-page tr {height: 100%;}

.clear {clear: both;}

.header {
    background: #3F4137;
    height: 60px;
}

.logo {
    background: url('../images/logo.png') no-repeat 10px 5px;
    width: 70px;
    height: 60px;
}

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

ul.mainmenu {
    list-style: none;
    margin: 0;
    padding: 0;
    height: 60px;
    color: #fdc073;
    font-size: 20px;
    overflow: hidden;
}

ul.mainmenu li {
    float: left;
    margin: 18px 40px 0 30px;
}

ul.mainmenu li.logo {margin: 0;}
ul.mainmenu li.last {float: right;}

ul.mainmenu li a {
    color: #fdc073;
    text-decoration: none;
}

ul.mainmenu li a:hover {
    color: #FDA83D;
}

.panelitems {
    text-align: center;
}

ul.langitem {
    list-style: none;
    display: inline-block;
    margin: 30px;
    padding: 0;
    max-width: 300px

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

ul.langitem li {
    margin: 10px 0 0 0;
    width: 100%;
}

ul.langitem li.image {
    margin: 0;
    text-align: center;
    width: 100%;
    height: 250px;
    overflow: hidden;
}

ul.langitem li.descr {
    color: #777;
    height: 170px;
    overflow: hidden;
}

ul.langitem li a {
    color: #fdc073;
    text-decoration: none;
    font-size: 20px;
}

ul.langitem li a:hover {
    color: #FDA83D;
}

div.button {
    background: #3F4137;
    max-width: 200px;
    margin: 0 auto;
    padding: 5px 20px 5px 20px;
    border-radius: 10px;
    font-size: 20px;
}

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		54

```

.table-content {
    width: 100%;
    min-height: 100%;
    height: 100%;
}

.left-chapters {
    margin: 0;
    background: #eee;
    border-right: 1px solid #ccc;
    min-height: 100%;
}

.left-chapters ul {
    width: 230px;
    min-height: 100%;
    margin: 0;
    padding: 20px 10px 10px 20px;
    list-style: none;
    font-size: 20px;
}

.left-chapters ul li {
    margin: 30px 0 0 0;
    color: #999;
}

.left-chapters ul li.selected { color: #0059b2; }

.left-chapters ul li a {
    color: #444;
    text-decoration: none;
}

.left-chapters ul li a:hover { color: #CC0000; }

.left-chapters ul li.share { margin: 60px 0 0 0; }

.left-chapters ul li.share p { margin: 30px 0 30px 0; }

.left-chapters ul li.share a {
    display: inline-block;

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

width: 40px;
    height: 40px;
    margin: 0 20px 0 0;
}

.left-chapters ul li.share a.share-yt {
    width: 100%;
    height: 30px;
    background: url('../images/share_yt.png') no-repeat 0 0
}

.left-chapters ul li.share a.share-yt:hover {
    width: 100%;
    height: 30px;
    background: url('../images/share_yt.png') no-repeat 0 -30px
}

.content {
    padding: 40px 20px 20px 40px;
    width: 100%;
    box-sizing: border-box;
}

ul.breadcrumbs {
    margin: 0;
    padding: 0;
    list-style: none;
    font-size: 20px;
    font-family: Sans-serif, Arial, Tahoma;
}

ul.breadcrumbs li {
    display: inline-block;
}

ul.breadcrumbs li a {
    color: #0059b2;
    text-decoration: none;

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

ul.breadcrumbs li a:hover { color: #CC0000; }

ul.breadcrumbs li:after{
    padding: 0 10px 0 10px;
    content: "\2192";
}

ul.breadcrumbs li.last:after {content: "";}

ul.control {
    margin: 0;
    padding: 60px 0 0 0;
    list-style: none;
    font-family: Sans-serif, Arial, Tahoma;
    font-size: 18px;
    width: 100%;
}

ul.control li { float: left; padding: 0 0 0 10px;}
ul.control li.first:before {content: "\2190"; padding: 0 10px 0 0;}
ul.control li.last {float: right; padding: 0 40px 0 0;}
ul.control li.last:after {content: "\2192"; padding: 0 0 0 10px;}
ul.control li a { color: #0059b2; text-decoration: none; }
ul.control li a:hover { color: #CC0000; }

ul.content-table {
    margin: 0;
    padding: 60px 0 0 30px;
    list-style: none;
    font-family: Sans-serif, Arial, Tahoma;
    font-size: 28px;
}

ul.content-table li {
    margin: 0 0 40px 0;
}

ul.content-table li a {

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

color: #BB5715;

    text-decoration: none;

}

ul.content-table li a:hover {color: #F0711C;}


.content-text {

    margin: 60px 0 0 30px;

    color: #000;

    font-size: 20px;

    font-family: Yandex Sans Display Light;

}

.content-text h1 {font-size: 32px;}


ul.lang-list {

    list-style: none;

    margin: 0;

    padding: 14px 0 0 0;

    background: #3F4137;

    width: 100%;

    height: 50px;

    border-top: 1px solid #959A82;

    box-sizing: border-box;

    font-size: 18px;

    font-family: Sans-serif, Arial, Tahoma;

}

ul.lang-list li {

    display: inline-block;

    margin-left: 40px;

}

ul.lang-list li.selected {color: #fdc073; border-bottom: 1px solid #fdc073;}

ul.lang-list li a {

    color: #eee;

    text-decoration: none;

}

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		



```
ul.lang-list li a:hover {color: #fdc073;}
```

```
.topic-subject .topic-line {  
    border-bottom: 1px solid #CC0000;  
    margin-top: -16px;  
}
```

```
.topic-subject .topic-text {  
    display: inline-block;  
    font-size: 28px;  
    color: #777;  
    padding: 0 10px 0 10px;  
    margin-left: 30px;  
    background: #fff;  
    font-family: Sans-serif, Arial, Tahoma;  
}
```

```
.list-topic {  
    margin: 40px 0 60px 0;  
    font-family: Sans-serif, Arial, Tahoma;  
}
```

```
.list-topic p {  
    margin: 0;  
    font-size: 28px;  
}
```

```
.list-topic ol {  
    columns: 2;  
    -webkit-columns: 2;  
    column-gap: 40px;  
}
```

```
.list-topic ol li {  
    display: list-item;  
    margin: 10px 10px 0 0px;  
    padding: 0 0 0 10px;
```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.list-topic ol li a {
    color: #0059b2;
    text-decoration: none;
}

.list-topic ol li a:hover {color: #CC0000;}

.highlight {
    max-width: 350px;
    padding: 0 10px 0 10px;
    margin: 0;
    overflow: auto;
    overflow-y: hidden;
    background: #f0f0f0;
}

.highlight p { margin: 0; }

.highlight .block {
    width: 100%;
    margin: 0;
}

.highlight .block p { margin: 0; }

.quote {
    background: url('../images/blockquote.png') no-repeat 0 10px;
    padding: 5px 5px 5px 70px;
    font-family: Sans-serif, Arial, Tahoma;
    font-style: italic;
}

#footer {
    box-sizing: border-box;

    background: #3F4137;
    padding: 5px;
    color: #aaa;

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        font-size: 14px;

        font-family: Verdana, Geneva, Arial, Helvetica;

        text-align: left;

        overflow: hidden;

    }

    #footer a {

        text-decoration: underline;

        color: #aaa;

    }


.form-input {

    width: 300px;

    font-size: 16px;

}


.form-label {

    display: inline-block;

    min-width: 150px;

    vertical-align: top;

}


.form-error ul {

    list-style: none;

    padding: 0;

    color: #CC0000;

}


.form-button {

    min-width: 200px;

    font-size: 16px;

}


.list-pages {

    text-align: center;

    margin: 0 0 20px 0;

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.list-pages ul {
    margin: 20px 0 0 0;
    padding: 0;
    list-style: none;
}

.list-pages ul li {
    display: inline-block;
    margin: 0 20px 0 0;
}

.list-pages a {
    color: #000;
    font-size: 24px;
    text-decoration: none;
}

.list-pages .page-num, .page-num-selected {
    display: inline-block;
    width: 60px;
    height: 44px;
    padding: 16px 0 0 0;
    border: 1px solid #d0d0d0;
    border-radius: 30px;
}

.list-pages .page-num:hover {
    box-shadow: 3px 3px 1px #d0d0d0;
}

.list-pages .page-num-selected {
    border: none;
    color: #000;
    font-size: 20px;
}

.list-pages .page-num-selected:hover {
    box-shadow: none;
}

```

					КП.ПО.02.ПІ-192.39.Д1	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		