Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Буслова Дмитра Юрійовича*
(ПІБ)

академічної групи *121-19-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка інформаційної платформи для працевлаштування студентів на основі технологічного стеку Python/Django framework/TelegramBot Api/PostgreSQL DB*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-1* *Буслова Дмитра Юрійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інформаційної платформи для працевлаштування студентів на основі технологічного стеку Python/Django framework/TelegramBot Api/PostgreSQL DB*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Буслов Д.Ю.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 91 с., 40 рис., 11 табл., 3 дод., 27 джерел.

Об'єкт розробки: веборієнтована інформаційна платформа для працевлаштування студентів.

Мета кваліфікаційної роботи: розробка інформаційної платформи для працевлаштування студентів на основі технологічного стеку Python/Django framework/TelegramBot Api/PostgreSQL DB

У вступі розглядається можливість роботи студентів і молодих людей, порушується проблема важливості наявності сервісу для роботи, а також переваги використання цієї вебплатформи для пошуку роботи.

У першому розділі описується в подробицях нюанси роботи молодих людей з боку законодавства України в галузі праці, призначення розробки та постановка задачі - постановка технологій, що використовуються, і поверхневий опис можливостей різних типів користувачів.

У другому розділі обговорюється в подробицях призначення платформи, права доступу користувачів, використовувані архітектури та шаблони програмування, послідовність алгоритмів функціонування сервера платформи, а також опис і пояснення вибору технологій для написання проекту.

В економічному розділі визначають рівень складності проекту, проводять розрахунок вартості розроблення вебплатформи залежно від різних параметрів (кількість рядків коду, ресурсів, електрики, Інтернету тощо), а також визначають запланований час на його створення.

Практичне значення роботи полягає у розробці вебплатформи, на якій роботодавець може подати оголошення про надання роботи, і зацікавлені молоді люди і студенти (особливо з НТУ "ДП") можуть знайти підробіток у вільний час і набути як досвіду, так і додаткових грошей.

Актуальність платформи обумовлюється великим попитом підлітків і студентів заробити кишенькові гроші, а також роботодавців, яким потрібна допомога в неважких завданнях.

Список ключових слів: ВЕБПЛАТФОРМА, БРАУЗЕР, ПЛАТФОРМА, ЗАРОБІТОК, СТУДЕНТИ, PYTHON, DJANGO, ПРАЦЕВЛАШТУВАННЯ.

ABSTRACT

Explanatory note: 91 p., 40 figures, 11 tables, 3 app., 27 sources.

Object of development: web-oriented information platform for student employment.

The purpose of the qualification work: development of an information platform for student employment based on the Python/Django framework/TelegramBot Api/PostgreSQL DB technology stack

The introduction discusses the possibility of employment for students and young people, raises the issue of the importance of having a service for work, as well as the benefits of using this web platform for job search.

The first section describes in detail the nuances of young people's work from the point of view of Ukrainian labor legislation, the purpose of the development and the task statement - the statement of the technologies used and a superficial description of the capabilities of different types of users.

The second section discusses in detail the purpose of the platform, user access rights, architectures and programming templates used, the sequence of algorithms for the platform server, as well as a description and explanation of the choice of technologies for the project.

The economic section determines the level of complexity of the project, calculates the cost of developing a web platform depending on various parameters (number of lines of code, resources - electricity, Internet, etc.), and determines the planned time for its creation.

The practical significance of the work is to develop a web platform where employers can post job announcements, and interested young people and students (especially from Dnipro University of Technology) can find part-time work in their free time and gain both experience and extra money.

The relevance of the platform is due to the high demand of teenagers and students to earn pocket money, as well as employers who need help with simple tasks.

List of keywords: WEB PLATFORM, BROWSER, PLATFORM, EARNINGS, STUDENTS, PYTHON, DJANGO, EMPLOYMENT.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML - це мова розмітки, яка використовується для визначення структури та вмісту вебсторінки.

CSS - це мова стилів, яка використовується для оформлення та візуального представлення вебсторінки.

JS - це мова програмування, яка дозволяє додавати інтерактивність та динамічну поведінку на вебсторінках.

SSL - протокол, який забезпечує шифрування даних та автентифікацію сервера, що передаються. Це робить зв'язок між клієнтом та сервером конфіденційним, запобігає перехопленню та заміні даних.

SQL (Structured Query Language) - це мова програмування, що використовується управління та взаємодії з реляційними базами даних.

XSS (Cross-Site Scripting) - це тип атаки на вебзастосунки, при якій зловмисник впроваджує шкідливий скрипт на вебсторінку, який виконується в браузері користувача.

CSRF (Cross-Site Request Forgery) - це тип атаки на вебзастосунки, при якій зловмисник змушує авторизованого користувача неусвідомлено виконати небажану дію на вебсайті, на якому користувач має активну сесію.

ACID-транзакції - атомарність, узгодженість, ізоляція, довговічність.

ЗМІСТ

ВСТУП

Платформи для пошуку роботи для студентів – це онлайнсистеми, які надають широку підтримку у процесі пошуку роботи. Вони дозволяють претендентам знаходити відповідні вакансії, ефективно уявляти себе роботодавцям та встановити зв'язок з потенційними роботодавцями. На цих платформах студенти можуть створювати профілі, які представляють свої контактні дані та навички. Вони також можуть завантажувати свої фотографії, щоб у роботодавців викликати більше довіри та симпатії.

За допомогою вебінтерфейсу та інструментів пошуку, платформи для пошуку роботи дозволяють студентам знаходити доступні вакансії, що відповідають їхнім інтересам та можливостям. Платформи цього роду часто пропонують різні фільтри, щоб пошук став простішим і зручнішим.

Після знаходження відповідної вакансії студент може використовувати функціонал платформи для надсилання заявки безпосередньо роботодавцю. В основному це текстове повідомлення в особистому чаті, який доступний лише студенту та роботодавцю. Роботодавцю також в основному доступна можливість одразу прийняти на позицію кандидата та закрити вакансію (приховати на платформі).

В цілому, платформи для пошуку роботи надають зручне та ефективне середовище для студентів, допомагаючи їм знаходити, відгукуватися та взаємодіяти з роботодавцями. Вони спрощують процес пошуку роботи, роблячи його більш доступним та структурованим як для студентів, так і для роботодавця.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Одним з підходів, який набуває великої популярності, є платформи для роботи студентів (особливо закордоном). Студенти набувають навичок, розширюють знання та отримують цінний досвід, необхідний для досягнення успіху у суспільному житті та в подальшій кар'єрі.

Студенти – це люди віком від 16-17 років. Вони не є повнолітніми (1 курс). Відповідно до "КЗПП України глава 13-14: праця молоді" молоді люди цього віку мають інші права [1].

Оскільки ми розглядаємо студентів, ми не братимемо до уваги закони кодексу, які поширюються на учнів коледжів. Внизу буде наведено короткі описи важливих законів, які регулюють роботу неповнолітніх [2].

Особи, які ще не досягли 18 років, мають однакові права у трудових правовідносинах з повнолітніми особами, але вони також мають право на певні пільги в сфері робочого часу, строку відпусток та інших умов праці, визначених законодавством України. Заборонено приймати на роботу осіб, які ще не досягли 16 років. Однак, за згодою одного з батьків або опікуна, такі особи, які вже досягли 15 років, можуть бути винятком і прийняті на роботу.

Кожне підприємство, установа або організація зобов'язані вести окремий облік працівників, які ще не досягли 18 років, вказуючи їх дату народження.

Застосування праці осіб, які не досягли 18 років, заборонено у таких випадках [1]:

1. нічні роботи (починаючи з 22:00 до 6:00);
2. роботи у вихідні дні;
3. важкі роботи;
4. роботи з небезпечними умовами праці;

5. підземні роботи;

6. понаднормові роботи;

7. роботи, пов'язані із переміщенням важких (за нормою) речей.

Заробітна плата неповнолітнього працівника не може бути нижчою за встановлений законом рівень мінімальної заробітної плати.

Працівники, які ще не досягли 18 років, отримують скорочену тривалість щоденної робочої зміни, але їх заробітна плата зберігається на тому ж рівні, що й для працівників відповідних категорій, які працюють повний робочий день.

Неповнолітнім зазвичай не надається право на випробувальний термін під час працевлаштування. Випробувальний термін - це певний період, коли працівник і роботодавець мають можливість оцінити взаємну сумісність і прийнятність умов роботи. У разі неповнолітніх працівників, цей період може бути пропущений, оскільки вони зазнають особливих обмежень та правового захисту у сфері праці.

Для працівників, які ще не досягли повнолітнього віку, передбачена скорочена тривалість робочого часу згідно зі статтею 51 КЗпП [2]: для осіб у віці від 16 до 18 років - 36 годин на тиждень, а для осіб у віці від 15 до 16 років (особливо учнів у віці від 14 до 15 років, які працюють під час канікул) - 24 години на тиждень.

Для роботи неповнолітніх потрібно:

1. Письмовий дозвіл одного з батьків або особи, яка його замінює

2. Проф. медичний огляд за наказом 246 (хірург, окуліст, терапевт, отоларинголог, невропатолог, гінеколог – для жінок + АНАЛІЗИ)

3. Паспорт

4. Довідка про РНОКПП або ідентифікаційний номер

5. Трудова книжка

6. Документ про освіту (якщо передбачено законом)

7. Трудовий договір у письмовій формі

8. Наказ (розпорядження) форми №П-1 про прийом неповнолітнього громадянина на роботу (після укладання трудового договору)

9. документ про освіту (спеціальність, кваліфікацію)

При вирішенні роботодавця про звільнення неповнолітнього працівника віком менше 18 років треба мати згоду міської служби у справах дітей. Ця вимога є обов'язковою для здійснення такого звільнення та спрямована на забезпечення захисту інтересів та благополуччя неповнолітнього працівника.

1.2. Призначення розробки та галузь застосування

В якості об'єкту розробки розглядається вебплатформа, на якій роботодавець може подати оголошення про надання роботи, та зацікавлені молоді люди можуть відгукнутися на цю вакансію та заробити як гроші, так і досвід.

Дана платформа повинна містити в собі можливість створювати особистий обліковий запис, додавати контактні дані, створювати оголошення про роботу, які будуть перебувати в публічному доступі, надавати можливість оповіщення через месенджер «Telegram» як роботодавців, так і претендентів про відгуки на вакансії та появу нових вакансій відповідно.

Для забезпечення безпеки та усвідомленості роботодавців та претендентів необхідно надати сторінки з розширеною інформацією про закони України щодо роботи неповнолітніх, а також додаткову інформацію про документи, необхідні для прийому на роботу молодих людей.

Крім цього важливо мати адмінпанель та команду людей, яка буде верифікувати як нові новостворені оголошення про роботу, так і самих роботодавців. Роботодавцями можуть бути організації або підприємці, або фізичні особи.

Оскільки цей сервіс також розроблявся для студентів нашого університету НТУ "Дніпровська Політехніка", нам потрібно надати додаткову можливість

роботи для студентів нашого університету, а саме: зробити окрему сторінку та додатковий функціонал для швидкого та зручного прийому на роботу.

Платформа розрахована на студентів у ролі претендентів, та викладачів чи звичайних громадян у ролі роботодавців. Це дозволить зробити працю молоді більш доступною та сучасною. Студенти мають можливість заробити гроші та навички, а роботодавці мають швидко зроблену роботу.

1.3. Підстави для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

− освітня програма 121 Інженерія програмного забезпечення;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;

− завдання на кваліфікаційну роботу на тему «Розробка інформаційної платформи для працевлаштування студентів на основі технологічного стеку Python/Django framework/TelegramBot Api/PostgreSQL DB».

1.4. Постановка завдання

Завдання: розробити вебплатформу мовою програмування Python із використанням фреймворку Django (бекенд), HTML & CSS & JS. Розробити Телеграм бота для оповіщення студентів та роботодавців.

Функціонал:

У табл. 1.1 наведено можливості користувачів (об’єктів) при роботі з вебплатформою.

Також передбачається, що платформа налічуватиме певні об’єкти з певними властивостями (наведено у табл. 1.2):

Таблиця 1.1

Можливості користувачів (об’єктів) при роботі з вебплатформою.

Об’єкт	Можливості
Роботодавець	реєстрація вхід до системи додавання оголошень перегляд оголошень зміна контактних даних облік найнятих людей додавання та видалення людей зі списку найнятих подавання заявки на верифікацію профілю спілкування зі студентами за допомогою онлайнчату
Студент	реєстрація вхід до системи перегляд оголошень відгук на оголошення спілкування з роботодавцями зміна контактних даних
Адміністратор	реєстрація вхід до системи створення, зміна та видалення групи користувачів створення, зміна та видалення користувачів створення, зміна, верифікація та видалення оголошень верифікація користувачів
Гість	реєстрація перегляд оголошень додавання оголошень

Таблиця 1.2

Об’єкти платформи та їх властивості

Об’єкт	Властивості
Користувач (студент, работодавець, адміністратор)	електронна пошта ім'я користувача номер телефону дата приєднання (дата) останній вхід (дата) є адмін є активним це персонал є суперкористувачем є офіційним є заблокованим ім'я прізвище унікальний код зображення вік людини
Оголошення	назва посилання вміст ціна опубліковано (дата) рубрика місто вік валюта кількість працівників автор ім'я автора номер телефону електронна пошта погляди обрані робітників
Чат:	назва учасники повідомлення

1.5. Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Вебплатформа “Teenwork” повинна бути реалізована мовою програмування Python у вигляді вебсайту, перейти до якого можна за допомогою браузера (Chrome. Mozilla, Microsoft Edge, Opera, Internet Explorer та ін.).

Для написання платформи передбачається використання наступних технологій: фреймворк Django, HTML, CSS, JS, Aiogram, Telegram Bot API.

Вся інформація, яка відображається користувачам і адміністратору, а також обмінюються додатками всередині сервера, повинна знаходитися в базі даних PostgresSQL на самому сервері.

Платформа має давати можливість роботодавцям задавати оголошення, студентам переглядати їх.

1.5.2 Вимоги до інформаційної безпеки

Інформаційна безпека вебплатформи є критичним аспектом, який включає в себе заходи захисту інформації, запобігання несанкціонованому доступу та збереження конфіденційності даних.

Платформа “Teenwork” повинна мати такі моменти безпеки:

Аутентифікація та авторизація – вебплатформа повинна мати механізми автентифікації користувачів, щоб переконатися, що лише зареєстровані користувачі чи адміністратори мають доступ до системи.

Захист від злому – платформа має бути захищена від злому. Це може включати застосування оновлень безпеки, захист від відомих уразливостей, використання сильних паролів, захист від атак переповнення буфера та інші методи.

Шифрування – вся інформація, що передається між вебплатформою та її користувачами, має бути зашифрована для захисту від перехоплення та прослуховування. Платформа має використовувати такий протокол як SSL.

Обробка вхідних даних – платформа повинна включати фільтрацію і валідацію вхідних даних, щоб запобігти атакам типу SQL-ін'єкції, міжсайтового скриптингу (XSS) і CSRF-атак.

Резервне копіювання та відновлення – важливо мати механізми резервного копіювання даних, щоб запобігти втраті інформації у разі збоїв або атак.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для доступу до платформи підійде будь-який пристрій: ноутбук, планшет, комп'ютер, телефон. Важливі такі характеристики браузера (наприклад візьмемо популярний - Chrome і ноутбук):

- версія браузера Chrome 81.0.4044.20 або вище;
- підтримка 32- або 64-розрядного процесора та операційної системи;
- операційна система Windows 7 чи вище;
- наявність маніпулятора-міші/тачпаду та клавіатури;
- оперативна пам'ять об'ємом 2 ГБ або вище
- процесор із частотою 1 ГГц або вище;
- жорсткий диск HDD чи SSD об'ємом 10 ГБ або вище;

1.5.4 Вимоги до інформаційної та програмної сумісності

Для ефективної роботи програми необхідно, щоб програмне забезпечення працювало на різних пристроях, таких як: ноутбук, планшет, комп'ютер або телефон, і відповідало таким вимогам:

- операційна система Windows (7+) / Linux / MacOs/Android ;
- веб-браузер Google Chrome / Opera / Safari/ Firefox / Microsoft Edge.

Застосунок має бути реалізовано на мові програмування JavaScript, мові програмування Python із використанням фреймворку Django (бекенд), HTML & CSS та бази даних PostgreSQL.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

У ході виконання кваліфікаційної роботи було розроблено вебплатформу, яка дає змогу молодим людям і студентам (особливо з НТУ "ДП") знайти підробіток у вільний час і набути як досвіду, так і додаткових грошей.

Призначення платформи:

- надати можливість підліткам і студентам просто, швидко і надійно отримати підробіток за допомогою зв'язку з роботодавцем через оголошення на платформі;
- дозволити роботодавцям отримати якісну і швидку допомогу в їхній роботі від підлітків і студентів;
- вести облік найнятих людей (з боку роботодавця);
- дати можливість студентам знаходити оголошення, що їх цікавлять, за заданими фільтрами (ключові слова, вік, місто/область, тип зайнятості, оплата від/до);
- надсилати повідомлення особисто роботодавцям/студентам про зміни в їхніх вакансіях і про появу нових вакансій відповідно.

Вебплатформа реалізує такі функції:

- Загальний функціонал (як для зареєстрованих, так і не для зареєстрованих користувачів):
  - перегляд оголошень;
  - пошук оголошень за заданими критеріями (описано вище);
  - можливість логіна/реєстрації.
- Функціонал зареєстрованого користувача - студент:
  - відгук на вакансію, що цікавить;
  - спілкування з роботодавцем через онлайнчат;

- отримання повідомлень про нові вакансії за заданими параметрами (описано вище) через Телеграм Бота;
- зміна/видалення особистого профілю;
- перегляд профілю роботодавця;
- налаштування параметрів особистих повідомлень.

- Функціонал зареєстрованого користувача - роботодавець:
  - публікація/зміна/видалення/архівація вакансій на платформі;
  - спілкування з працівником (потенційним) через онлайнчат;
  - додавання людини до списку "найнятих людей";
  - перегляд і управління найнятими співробітниками;
  - зміна/видалення особистого профілю;
  - перегляд профілю студента;
  - найм студентів з НТУ "ДП";
  - налаштування параметрів особистих повідомлень.
- Функціонал зареєстрованого користувача - адміністратор:
  - створення груп користувачів ;
  - видача прав доступу певним користувачам;
  - створення/зміна/видалення оголошень на платформі;
  - створення/зміна/видалення особистих профілів користувачів на платформі;
  - створення/зміна/видалення чатів і повідомлень на платформі;
  - створення/зміна/видалення особистих повідомлень на платформі;
  - налаштування розсилки email'ів на платформі;
  - створення/зміна/видалення публікацій блогу на платформі;
  - створення/зміна/видалення певних сторінок про загальну інформацію, що може бути корисною для роботодавців і студенатів на платформі.

2.2. Опис застосованих математичних методів

Розроблена платформа надає можливості створювати особистий обліковий запис, додавати контактні дані, створювати оголошення про роботу, які будуть перебувати у публічному доступі, надавати можливість оповіщення через месенджер Telegram як роботодавців, так і претендентів про відгуки на вакансії та появу нових вакансій відповідно.

У зв'язку з цим, функціональність платформи не включає рішення математичних завдань або використання математичних формул.

2.3. Опис використаної архітектури та шаблонів проектування

Платформа, призначена для молодих людей і студентів, була розроблена з використанням архітектурного шаблону проектування Model-View-Controller (MVC). Однак складно стверджувати, що Django суворо слідує моделі MVC, оскільки обробка контролера відбувається в самому середовищі розробки. А найбільш важливі можливості цього фреймворку пов'язані з моделями, шаблонами та уявленнями. Слід зазначити, що Django використовує модель Model-View-Template (MVT). Цей шаблон є дочірнім, і його вважають специфікою шаблону MVC.

Як було сказано, архітектура MVT частково відрізняється від MVC. Основна відмінність між цими двома шаблонами полягає в тому, що в Django весь Controller (код, що відповідає за взаємодію між моделлю та поданням) вже реалізований фреймворком Django, тому робота, що залишилася – це робота з шаблонизатором. Шаблон є файлом HTML, доповненим синтаксисом шаблонної мови Django.

Нижче наведена діаграма (рис. 2.1.) ілюструє взаємодію компонентів шаблону MVT при обробці запиту користувача.

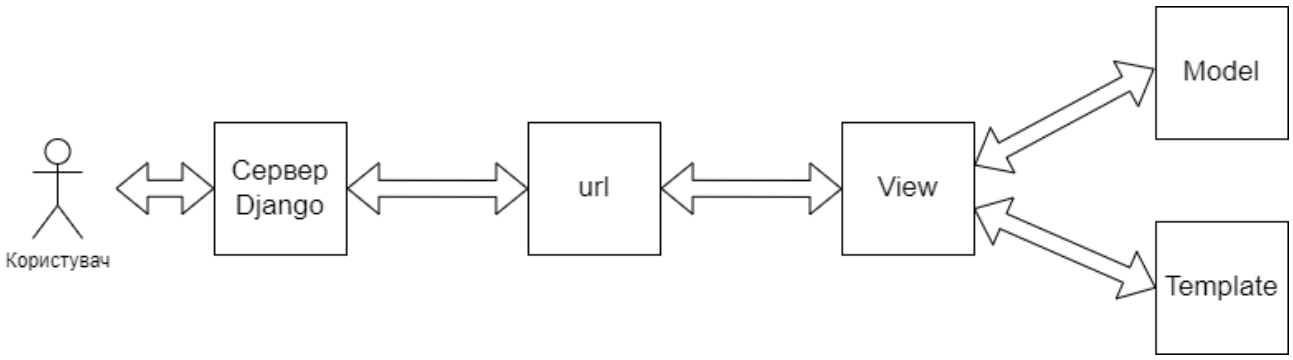


Рис. 2.1. Взаємодія компонентів шаблону MVT при обробці запиту користувача.

У процесі розробки із застосуванням шаблону MVT в Django розробник надає модель (Model), представлення (View) і шаблон (Template), а потім здійснює їх зв'язування з певними URL-адресами в додатку. Django, у свою чергу, забезпечує механізм обробки запитів користувачів та надає відповідні компоненти, пов'язані з певними URL-адресами користувачам.

При побудові архітектури Views у розробників є 2 підходи побудови запитань, що обробляють, уявлень: подання на основі функцій – Function based views (FBV) і подання на основі класів – Class based views (CBV).

Views мають такі особливості [17]:

- Доступні для виклику. CBV успадковують метод as_view(), який використовує dispatch() метод для виклику відповідного методу класа в залежності від методу HTTP (get, post, і т.д.)
- Повинні приймати об'єкт HttpRequest як перший позиційний аргумент.
- Повинні повернути об'єкт HttpResponse або викликати виняток (exception).

Спочатку Django мав стандартний підхід – FBV, який використовує функції Python як уявлення. І тут кожна функція обробляє запит і повертає відповідь. Розробник самостійно визначає логіку обробки запитів та взаємодії з моделлю та шаблоном.

У проекті даної платформи використовувався переважно підхід FBV, тому що була логіка, яку можна було написати в уявленнях виду функцій (рис. 2.2.).

```python
@staff_member_required
def add_blog_post(request):
    if request.method =='POST':
        form = TeenworkBlogForm(data=request.POST)
        if form.is_valid():
            # ...
            return redirect('board:tw_blog')
        else:
            return redirect('board:add_blog_post')
    else:
        form = TeenworkBlogForm()
        # ...
        return render(request, 'others/blog_post_add.html', {})
```

Рис. 2.2. Функція, написана за допомогою підходу FBV

CBV в Django є класами Python. Фреймворк Django поставляється з набором наперед визначених CBV, які містять заздалегідь налаштовані функціональності. Їх можна перевикористовувати та розширювати за потребою. Цим класам надано описові імена, що відображають їх функціональність. Вони часто називаються "універсальними уявленнями", оскільки вони надають рішення для загальних вимог (рис. 2.3.).

```python
class RequestResetEmailView(View):
    def get(self, request):
        return render(request, 'registration/reset_email.html')

    def post(self, request):
        email = request.POST.get('email', None)
        # ...
        user = Account.objects.filter(email=email)
        if user.exists():
            # ...
            return render(request, 'registration/reset_email_success.html')
        else:
            return redirect('/registration/')
```

Рис. 2.3. Клас, написаний за допомогою підходу CBV

Django є фреймворком. Фреймворки мають сувору структуру для організації проекту. Django надає певні правила організації файлів та коду у проекті. Цей фреймворк рекомендує використовувати модульну структуру, де програми розділені на логічні компоненти, такі як моделі, уявлення та шаблони. Кожна програма Django має свою власну директорію, де зберігаються відповідні файли (рис. 2.4.).

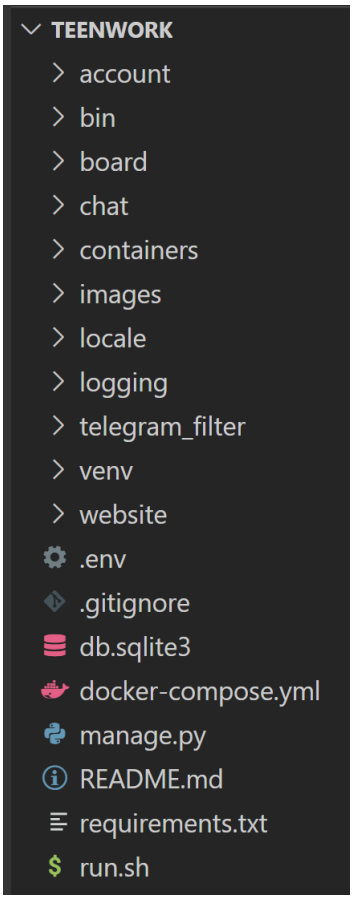


Рис. 2.4. Лістинг файлів директорії проекту Teenwork

2.4. Опис використаних технологій та мов програмування

Розроблену вебплатформу для пошуку роботи було реалізовано на мові Python з використанням фреймворка Django.

За світовими рейтингами TIOBE [4] (рис. 2.5.) - індекс PyPL (PopularitY of Programming Language), що оцінює популярність мов програмування, на основі підрахунку результатів пошукових запитів, що містять назву мови, і Stack

Overflow (всесвітній сайт з тематики програмування) серед усіх мов програмування явно виділяється мова Python. За індексами TIOBE [4] та PyPL у 2021 році мова Python вперше вийшла на перші місця [5]. Її позиції ще не зовсім впевнені - 11,77% у Python проти 10,72% у Java та C (рис. 2.5.) але через стрімкий розвиток Data Science лідерство Python у 2022 за прогнозами стане незаперечним.

Nov 2021	Nov 2020	Change	Programming Language	Ratings	Change
1	2	↑	Python	11.77%	-0.35%
2	1	↓	C	10.72%	-5.49%
3	3		Java	10.72%	-0.96%
4	4		C++	8.28%	+0.69%
5	5		C#	6.06%	+1.39%
6	6		Visual Basic	5.72%	+1.72%
7	7		JavaScript	2.66%	+0.63%
8	16	⇈	Assembly language	2.52%	+1.35%
9	10	↑	SQL	2.11%	+0.58%
10	8	↓	PHP	1.81%	+0.02%

Рис. 2.5. Світовий рейтинг мов програмування TIOBE

Звичайний сайт або вебсервіс, може бути створений за допомогою «голої» мови, наприклад, php – де всі методи та класи потрібно створювати самому. Але в мові Python існують так звані фреймворки – шаблони готових рішень програмних компонентів, на основі яких можна дописати власний код. Таким чином, при створенні сайту можливо буде вказувати готові методи та класи.

Як видно з діаграми на рис. 2.6. Django є фаворитом серед інших фреймворків мови Python [8].

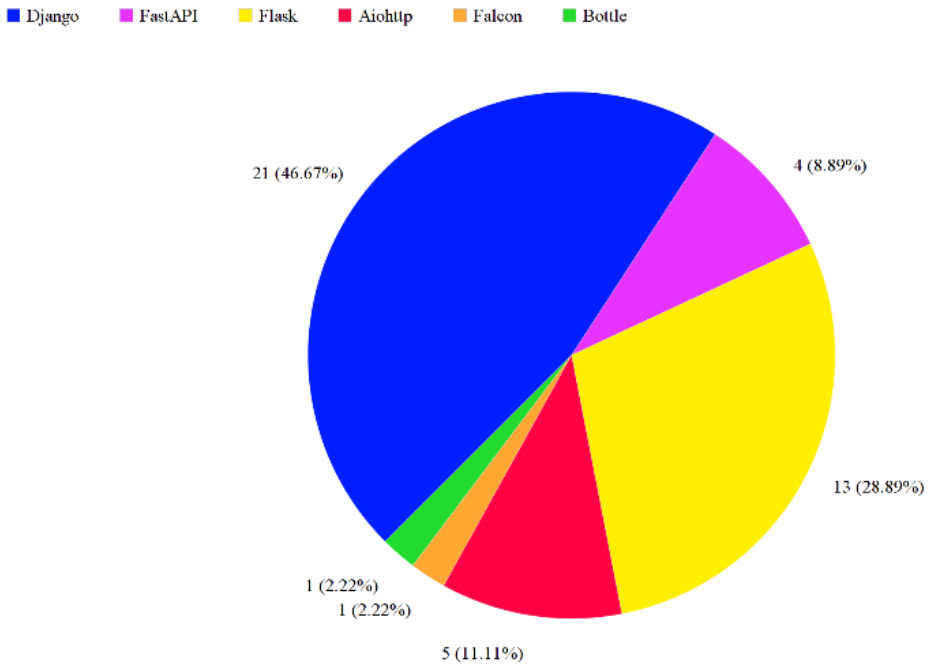


Рис. 2.6. Популярність фреймворків Flask та Django

Django є популярним відкритим середовищем Python, яке має багато переваг. Воно оптимізоване для пошукової оптимізації (SEO), що сприяє просуванню сайту в Інтернеті. Django є універсальним за своєю природою і має високий рівень надійності завдяки ретельному тестуванню.

Одна з ключових переваг Django - це безпека. Воно вбудовує захист від підробки CSRF-запитів і SQL-ін'єкцій, що дозволяє запобігти багатьом типам атак. Django також надає швидкий процес розробки завдяки вбудованим модулям і функціональності. Він пропонує багато розширених функцій, таких як карти сайту для адміністрування та rss-контент, які полегшують роботу з сайтом.

Ще однією перевагою Django є його ефективність при обробці великого обсягу трафіку. Він добре масштабується і забезпечує стабільну роботу навіть при високому навантаженні. Django є популярним фреймворком, який використовують багато великих компаній. Деякі з відомих компаній, що використовують Django, включають: YouTube, Пошук Google, Dropbox, Instagram, Pinterest, Quora, Yahoo Maps, Spotify, The Onion, Disqus, Washington Post, Bitbucket, Eventbrite, Mozilla [6], що доводить надійність цього фреймворка.

Оскільки Django є фреймворком, він має строгу структуру для організації проектів, що сприяє чистоті, модульності та легкості розробки. В основі цієї структури лежить концепція "компонування". Компонування означає, що проект розбивається на декілька додатків, які виконують конкретні функції і можуть бути повторно використані.

У стандартній структурі Django проекту, основні елементи включають:

- кореневу директорію проекту (містить налаштування та глобальні файли проекту);
- додатки (apps) - кожен функціональний блок проекту організований як окремий додаток. Кожен додаток містить свою модель даних, представлення, URL-шаблони та статичні файли;
- моделі (models) - описують структуру бази даних проекту та використовуються для зберігання та отримання даних за допомогою Django ORM [14] (у переважній кількості випадків) або, в рідкості, raw-sql запитів;
- представлення (views) - обробляють HTTP-запити та визначають, які дані потрібно показати користувачеві;
- шаблони (templates) - файли HTML, які відображають дані, створені views;
- URL-шаблони (patterns): вони визначають зв'язок між URL-адресами та views, які мають обробляти запити;
- статичні файли (static files): це файли, такі як зображення, CSS-стилі та JS-скрипти, які використовуються для оформлення веб-сторінок.

Ця строга структура Django проектів допомагає розробникам легко орієнтуватись у коді, покращує його читабельність та підтримуваність, а також сприяє швидкому розвитку проектів за рахунок повторного використання компонентів.

Загалом, Django - це потужне середовище, яке може значно зекономити час на кожному етапі від формування ідеї до релізу продукту. Воно надає широкий функціонал і забезпечує безпеку та надійність ваших програм.

За замовчуванням Django налаштований на зберігання даних у полегшеному файлі бази даних - SQLite. Хоча це добре працює за деяких навантажень, більш традиційна система керування базами даних може підвищити продуктивність у виробничому середовищі.

Django офіційно підтримує наступні бази даних:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

Найпопулярнішим рішенням для вибору БД є PostgreSQL [15].Вона також відома як Postgres - це реляційна база даних з відкритим вихідним кодом, яка підтримує як SQL (реляційну), так і JSON (нереляційну) моделі даних. Розробка проєкту PostgreSQL розпочалася 1986 року в Каліфорнійському університеті в Берклі під керівництвом професора Майкла Стоунбрейкера. Спочатку проект називався POSTGRES на честь попередньої бази даних Ingres, також створеної в Берклі. POSTGRES була розроблена з метою включення тільки необхідних функцій для повної обробки різних типів даних.

PostgreSQL має потужні можливості, включно з підтримкою ACID-транзакцій (атомарність, узгодженість, ізоляція, довговічність), уявлення, що автоматично оновлюються, матеріалізовані уявлення, тригери, зовнішні ключі та збережені процедури. Ці можливості забезпечують надійність, цілісність і гнучкість роботи з даними в PostgreSQL.

Ключові особливості PostgreSQL:

- масштабованість. PostgreSQL призначениа для управління широким спектром робочих навантажень, від окремих комп'ютерів до сховищ даних або веб-додатків з безліччю одночасних користувачів;

- реляційна структура. PostgreSQL заснована на реляційній моделі даних, де дані організовані у вигляді таблиць зі зв'язками між ними. Це дозволяє ефективно зберігати та управляти даними;
- гігантська база користувачів. Багато фірм створили продукти і рішення на основі PostgreSQL. Серед них Apple, Red Hat, Cisco, Instagram та ін.;
- безпека. Група глобального розвитку PostgreSQL (PGDG) дуже серйозно ставиться до безпеки. Це дає змогу користувачам довіряти PostgreSQL для збереження критично важливих даних;
- розширена підтримка SQL. PostgreSQL підтримує стандартну мову запитів SQL з великою кількістю розширень і додаткових функцій, що дозволяє зручно взаємодіяти з базою даних.

Основні причини використання інтеграції з Django PostgreSQL:

- включає в себе безліч типів даних, які виключно сумісні з PostgreSQL;
- надає *django.contrib.postgres* для операцій з базою даних PostgreSQL;
- підтримує більшість функцій PostgreSQL;
- пропонує важливі специфічні для PostgreSQL функції, такі як функції агрегування, обмеження бази даних, віджети полів форми, пошук, повнотекстовий пошук, засоби перевірки та багато іншого;

Однією з найпотужніших функцій Django є його Object-Relational Mapper (ORM), який дає змогу вам взаємодіяти з вашою базою даних так само, як із SQL.

Основна мета ORM [14] - передавати дані між базою даних і моделями в застосунку. Він відображає зв'язок між базою даних і моделлю. Перевага використання ORM полягає в тому, що він робить весь процес розроблення швидким і безпомилковим. По суті, це позбавляє від необхідності писати код SQL. Також можна легше змінити базу даних, якщо це необхідно.

Щоб подолати проблему жорстко закодованих записів у базі даних, було розроблено концепцію ORM. ORM автоматично створюють базу даних з певних

моделей або схем, що означає, що розробнику не потрібно знати SQL, який використовується в базі даних.

Нижче наведено приклад застосування Django ORM query (рис. 2.7.-2.8.):

```
if request.user.is_authenticated:
    if Account.objects.get(email=request.user).person_age != 0 or Account.objects.get(email=request.user).person_age != None:
        person_age = Account.objects.get(email=request.user).person_age

        board_obj1 = Board.objects.filter(Q(age=person_age), Q(status='published')|Q(status='24hour')).order_by('?') # order_by('-published')
        board_obj_other = Board.objects.filter(~Q(age=person_age), Q(status='published')|Q(status='24hour')).order_by('?') # ? - random

        arr = []
        for val in board_obj1:
            arr.append(val.id)
        for val in board_obj_other:
            arr.append(val.id)

        preserved = Case(*[When(pk=pk, then=pos) for pos, pk in enumerate(arr)])
        board_obj = Board.objects.filter(pk__in=arr).order_by(preserved)
    else:
        board_obj = Board.objects.filter(Q(status='published')|Q(status='24hour')).order_by('?')
else:
    board_obj = Board.objects.filter(Q(status='published')|Q(status='24hour')).order_by('?')
```

Рис. 2.7. Приклад застосування Django ORM query.

```
if Account.objects.filter(username=username).exists():
    board_obj = Board.objects.filter(Q(author=Account.objects.get(username=username).pk), Q(status='published')|Q(status='24hour'))
```

Рис. 2.8. Приклад 2 застосування Django ORM query.

На додачу до ORM є чудовий механізм, який надає безпечну роботу з БД - міграції. Django міграції - це інструмент, який дає змогу керувати змінами в структурі бази даних у процесі розробки застосунку. Він автоматично створює і застосовує зміни в базі даних, як-от створення таблиць, додавання або видалення полів та інші зміни, щоб відповідати моделям Django. Це дає змогу розробникам легко вносити зміни в базу даних і підтримувати її актуальною під час розроблення програми.

2.5. Опис структури системи та алгоритмів її функціонування

Структура вебплатформи наведена на рис. 2.4.

У подробицях структуру проєкту можна описати таким чином [19]:

- account - Django-додаток, відповідає за логіку користувача;

- bin - .sh-файли, використовуються в production розгортанні проєкту на сервері;
- board - Django-додаток, відповідає за логіку оголошень. Містить статичні файли і templates;
- chat - Django-додаток, відповідає за логіку чату;
- containers - містить Dockerfiles та інші конфігураційні файли для Docker'a;
- images - містить завантажені медіа-файли користувачами;
- locale - містить django.mo і django.po файли для підтримки багатомовності на сайті;
- logging - містить .log-файли (для відстеження помилок та іншої інформації, що генерується під час взаємодії із сервером (проєктом);
- telegram_filter - Django-додаток, що відповідає за логіку фільтрації та надсилання повідомлень через Телеграм Бота;
- venv - папка віртуального оточення (для локальної розробки та debugging'a);
- website - коренева папка проєкту, відповідає за конфігурацію проєкту і є стартовою точкою для прийняття запитів;
- .env - файл із конфіденційними конфігураційними даними;
- .gitignore - файл із директоріями та файлами, які потрібно упустити під час відправлення на хостинг проєктів (пр. Github);
- db.sqlite3 - sqlite-база даних, яка використовується в локальній розробці;
- docker-compose.yml - Docker Compose файл, який робить build контейнерів і об'єднує їх для спільної взаємодії;
- manage.py - це виконуваний файл у Django, який являє собою точку входу для виконання різних команд і завдань у проєкті Django;
- README.md - файл-tutorial, який пояснює, як взаємодіяти з додатком;

- requirements.txt - містить необхідні бібліотеки та версії для встановлення;
- run.sh - .sh скрипт для запуску проєкту.

Алгоритм функціонування платформи:

1. Для локальної розробки сервер запускається за допомогою команди: *python manage.py runserver 0.0.0.0:80*. На стадії production запускається за допомогою gunicorn: *gunicorn --log-level info --log-file=- --workers 4 --timeout 300 --name teenwork_gunicorn -b 0.0.0.0:8000 --reload core.wsgi:application*. Django (у раз і локальної) розробки створює HTTP-сервер, який прослуховує певний порт і чекає на вхідні запити від клієнтів.

2. Обробка запитів: Коли клієнт відправляє HTTP-запит на сервер, Django отримує цей запит і починає його обробку. Відбувається наступне:

3. Маршрутизація запиту: Django використовує файл urls.py, щоб визначити, який View повинен бути викликаний для даної URL-адреси.

4. Виклик View: Django викликає відповідий View, яке обробляє запит і повертає HTTP-відповідь.

5. Робота з моделями: Під час роботи з базою даних Django використовує Models для представлення даних та взаємодії з базою даних. Моделі визначені у файлах models.py і представляють таблиці в базі даних. Django створює SQL-запити для роботи з базою даних за допомогою Django ORM.

6. Обробка форм: Якщо в поданні потрібна робота з формами, Django обробляє надіслані дані форми, перевіряє їх на валідність і зберігає / оновлює дані в базі даних.

7. Надсилання HTTP-відповіді: Після опрацювання запиту Django формує HTTP-відповідь, що містить статус код, заголовки та вміст. Відповідь може бути HTML-сторінкою або JSON-даними. У нашій платформі це переважно HTML-сторінки.

8. Відправлення відповіді клієнту: Django відправляє сформовану HTTP-відповідь назад клієнту, який потім відображає її.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Оскільки платформа використовує базу даних для зберігання інформації різних моделей, ми можемо як приймати дані, тобто зберігати їх, так і виводити користувачеві в разі відповідних запитів.

База даних містить у собі наступну структуру таблиць, зображених на рис. 2.9.

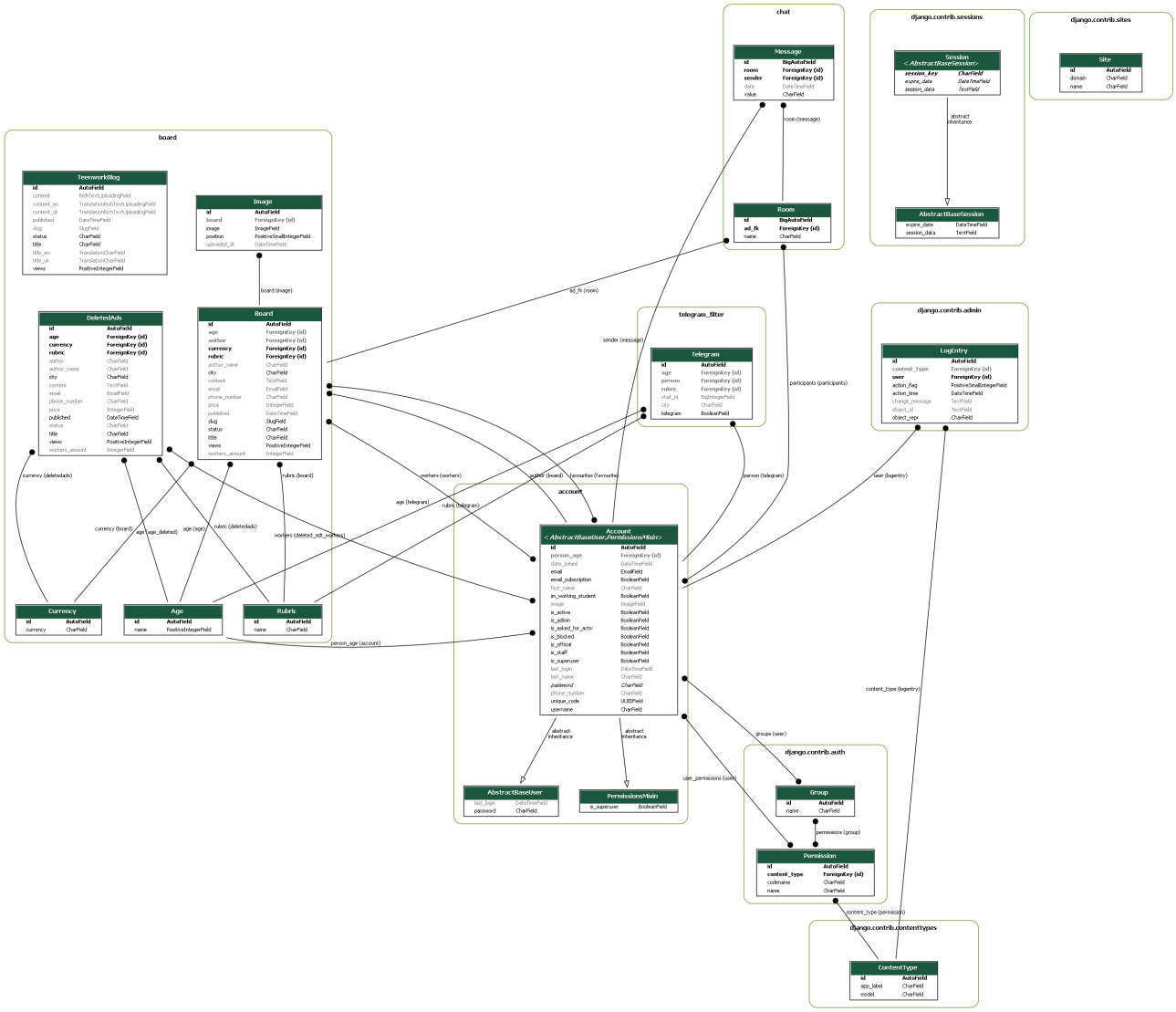


Рис. 2.9. ER-діаграма БД

Структура таблиць БД (типи даних вказано відповідно до Django ORM форматів типів даних). Опис полів усіх таблиць БД наведено у таблицях 2.1-2.11.

Таблиця 2.1

Опис полів таблиці Account

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Унікальний ідентифікатор	unique=True primaryKey
username	CharField	Ім'я користувача	max_length=30 unique=True
email	EmailField	Електронна пошта	max_length=60 unique=True
phone_number	CharField	Номер телефону	max_length=13 unique=True null=True
date_joined	DateTimeField	Дата реєстрації	auto_now_add =True
last_login	DateTimeField	Дата останнього входу	auto_now_add =True
is_admin	BooleanField	Чи є людина адміністратором	default=False
is_active	BooleanField	Чи є людина активним користувачем	default=True
is_staff	BooleanField	Чи є людина частиною робочого персоналу платформи	default=False
is_superuser	BooleanField	Чи є людина суперкористувачем	default=False
is_official	BooleanField	Чи є людина офіційною особою / організацією	default=False
is_blocked	BooleanField	Чи є людина заблокованим користувачем на майданчику	default=False
first_name	CharField	Ім'я	max_length=30 default=''

Продовж. табл. 2.1

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
last_name	CharField	Прізвище	max_length=30 default=''
unique_code	UUIDField	Унікальний код (використовується для надсилання повідомлень через телеграм бота)	default=random unique=True editable=False
email_subscription	BooleanField	Чи підписана людина на розсилку	default=True
im_working_student	BooleanField	Чи є людина студентом, який активно шукає роботу	default=False
is_asked_for_activ	BooleanField	Чи була людина повідомлена про лист, що прийшов на пошту через 3 дні, у разі, якщо вона не верифікувала свій особистий профіль	default=False
image	ImageField	Зображення профілю (фото людини)	null=True
person_age	ForeignKey	Вік людини	null=True on_delete=models.PROTECT

Таблиця 2.2

Опис полів таблиці Board

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор публікації	unique=True primaryKey
title	CharField	Заголовок публікації	max_length=70

Продовж. табл. 2.2

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
slug	SlugField	Посилання (строковий ідентифікатор)	max_length=150 unique = True
content	TextField	Зміст публікації	max_length=5500 null=True
price	IntegerField	Оплата	null=True
published	DateTimeField	Дата публікації	auto_now_add=True db_index=True
rubric	ForeignKey	Рубрика	on_delete=models.PROTECT
city	CharField	Місто	max_length=100
age	ForeignKey	Вік (діапазон)	on_delete=models.PROTECT null=True
currency	ForeignKey	Валюта	on_delete=models.PROTECT null=True
workers_amount	IntegerField	Кількість працівників	null=True
author	ForeignKey	Автор	on_delete=models.CASCADE null=True
author_name	CharField	Ім'я автора	max_length=50
phone_number	CharField	Номер телефону	max_length=13
email	EmailField	Електронна пошта	-
views	PositiveIntegerField	Кількість переглядів	default=0
favourites	ManyToManyField	До яких людей додано публікацію в обране	default=None
workers	ManyToManyField	Які молоді люди працюють на цій вакансії	default=None
status	CharField	Статус оголошення	max_length=20 default='draft'

Таблиця 2.3.

Опис полів таблиці Room

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор чату	unique=True primaryKey
name	CharField	Назва чату	max_length = 500
ad_fk	ForeignKey	Fk на оголошення	on_delete=models.CASCADE default=None null=True
participants	ManyToManyField	Учасники чату	default=None

Таблиця 2.4.

Опис полів таблиці Message

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор повідомлення	unique=True primaryKey
value	CharField	Текст повідомлення	max_length = 10000
room	ForeignKey	Fk кімнати	on_delete=models.CASCADE
date	DateTimeField	Дата відправлення	default = datetime.now
sender	ForeignKey	Відправник	on_delete=models.SET_NULL null=True

Таблиця 2.5.

Опис полів таблиці Image

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор зображення	unique=True primaryKey
image	ImageField	Відносний шлях до зображення	max_length=300
board	ForeignKey	Fk до оголошення	null=True on_delete=models.CASCADE
position	PositiveSmallIntegerField	Позиція	default=1
uploaded_at	DateTimeField	Дата завантаження	auto_now_add=True

Таблиця 2.6.

Опис полів таблиці Rubric

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор рубрики	unique=True primaryKey
name	CharField	Назва рубрики	max_length=50 db_index=True

Таблиця 2.7.

Опис полів таблиці Age

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор віку	unique=True primaryKey
name	PositiveIntegerField	Вік	db_index=True

Таблиця 2.8.

Опис полів таблиці Currency

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор валюти	unique=True primaryKey
name	CharField	Назва валюти	max_length=10

Таблиця 2.9.

Опис полів таблиці DeletedAds

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор видаленого оголошення	unique=True primaryKey
title	CharField	Заголовок публікації	max_length=70
content	TextField	Зміст публікації	max_length=5500 null=True

Продовж. табл. 2.9

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
price	IntegerField	Оплата	null=True
published	DateTimeField	Дата публікації	db_index=True editable=False
rubric	ForeignKey	Рубрика	null=True on_delete=models.PROTECT
city	CharField	Місто	max_length=100 null=True
age	ForeignKey	Вік (діапазон)	null=True on_delete=models.PROTECT
currency	ForeignKey	Валюта	null=True on_delete=models.PROTECT
workers_amount	IntegerField	Кількість працівників	null=True
author	CharField	Автор	max_length=50 null=True
author_name	CharField	Ім'я автора	max_length=50
phone_number	CharField	Номер телефону	max_length=13
email	EmailField	Електронна пошта	-

Продовж. табл. 2.9

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
views	PositiveIntegerField	Кількість переглядів	default=0
workers	ManyToManyField	Які молоді люди працювали на цій вакансії	default=None
status	CharField	Статус оголошення	max_length=20

Таблиця 2.10

Опис полів таблиці TeenworkBlog

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор публікації	unique=True primaryKey
title	CharField	Заголовок	max_length=200
slug	SlugField	Посилання (строковий ідентифікатор)	max_length=150 unique = True null=True
content	RichTextUploadingField	Зміст	null=True
published	DateTimeField	Опубліковано	auto_now_add=True
views	PositiveIntegerField	Перегляди	default=0
status	CharField	Статус	max_length=20 default='draft'

Таблиця 2.11.

Опис полів таблиці Telegram

Назва поля	Тип даних поля	Ціль (значення)	Обмеження (constraints)
id	BigIntegerField	Ідентифікатор публікації	unique=True primaryKey
telegram	BooleanField	Чи хоче людина отримувати повідомлення в телеграм	null=True default=False
rubric	ForeignKey	Рубрика	null=True on_delete=models.PROTECT
age	ForeignKey	Вік	null=True on_delete=models.PROTECT
city	CharField	Місто	null=True max_length=100
person	ForeignKey	Людина	null=True on_delete=models.CASCADE
chat_id	BigIntegerField	Id чату	null=True editable=False

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для роботи вебплафторми не потрібно великих комп'ютерних потужностей. Цю платформу було запущено на сервері з мінімальними характеристиками потужностей (нижче), але це все ж давало змогу обробляти певну кількість клієнтських запитів. Для розробки та роботи платформи підходять такі характеристики системи:

- операційна система: Linux, Windows, MacOS та інші, які підтримують інтерпретацію мови програмування Python;
- процесор з тактовою частотою 1 ГГц або вище;
- RAM об'ємом 1ГБ або вище;
- жорсткий диск об’ємом 10 ГБ або вище;
- монітор, миша (тачпад) і клавіатура - для розробки.

2.7.2. Використані програмні засоби

Під час платформи були використані наступні програмні засоби:

- браузер Chrome версії 81.0.4044.20 (стара версія для тестування правильності роботи UI);
- Microsoft Visual Studio Code;
- PostrgeSQL;

Як бекенд було використано мову Python версії 3.9, фреймворк Django версії 3.4.2.

Як фронтенд було використано мову гіпертекстової розмітки HTML5, каскадні таблиці стилів CSS3 і мову програмування JS (Javascript).

Крім цього було використано такі бібліотеки: aiogram, aiohttp, asgiref, bcrypt, celery, django-celery-beat, django-celery-results, django-cleanup, django-filter, django-modeltranslation, django-rosetta, Pillow, psycopg2, django-extensions, python-dotenv, python-slugify, pytils, six, slugify, googletrans, django-ckeditor (більше бібліотек описано в requirements.txt).

2.7.3. Виклик та завантаження програми

Щоб запустити сервер платформи, потрібно:

- відкрити термінал;
- перейти в кореневу директорію проєкту;

- запустити сервер;
  - для локальної розробки: python manage.py runserver 0.0.0.0:80;
  - для локальної розробки і деплою в production:
    - встановити Docker на свою ОС;
    - docker-compose build (для побудови образу проєкту);
    - docker-compose up (для запуску контейнера).

2.7.4. Опис інтерфейсу користувача

Коли гість або зареєстрований користувач відкривають сайт, їм відображається головна сторінка. На цій сторінці вони можуть переглядати оголошення, змінювати мови (доступні: українська, англійська), змінювати тему платформи (доступні: світла, темна), перейти в Телеграм Бота, щоб подивитися останні повідомлення. Також гість може перейти на сторінку логіна/реєстрації, а зареєстрований користувач перейти на свій особистий акаунт або подати оголошення.

Основні сторінки та елементи цих сторінок зображено на рис. 2.10. – 2.39.

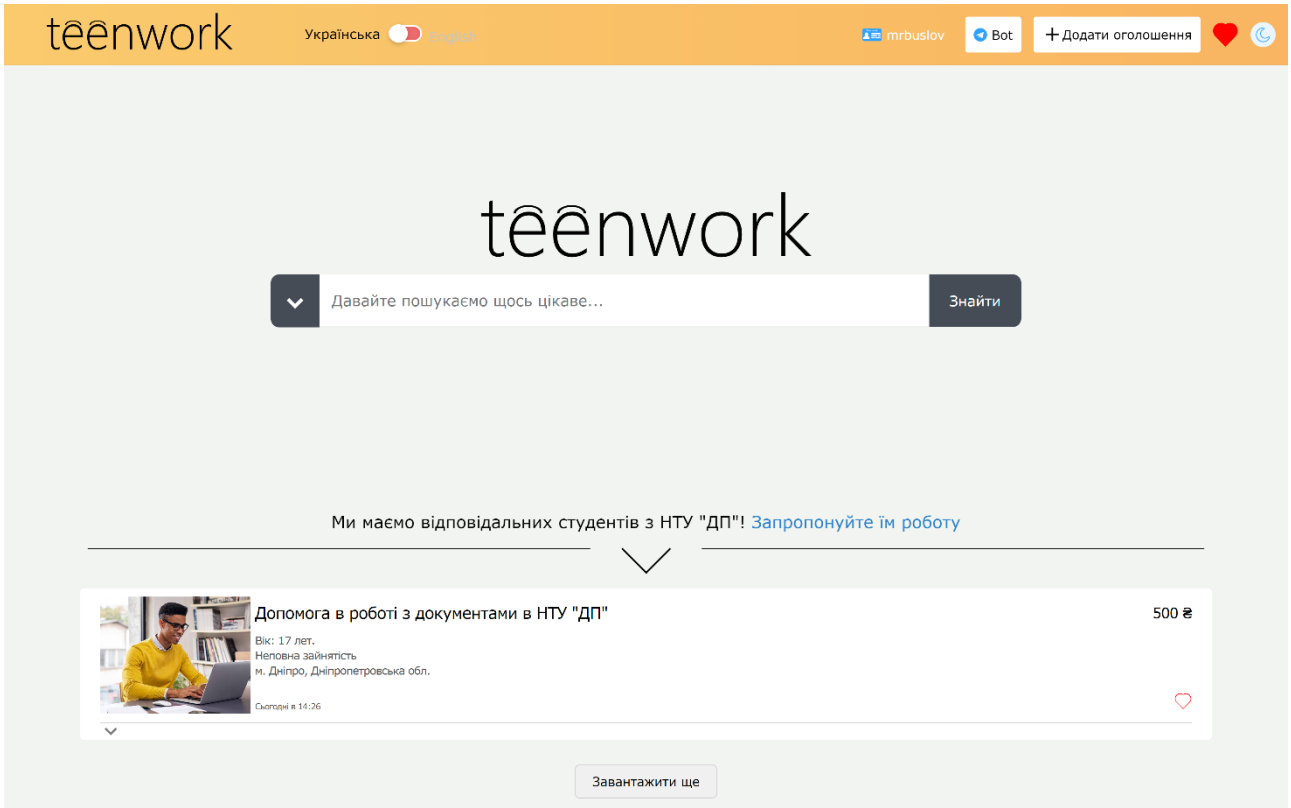


Рис. 2.10. Головна сторінка

При натисканні на кнопку пошуку (стрілка вниз) відкривається меню, що випадає, з фільтрами пошуку (рис. 2.11.)

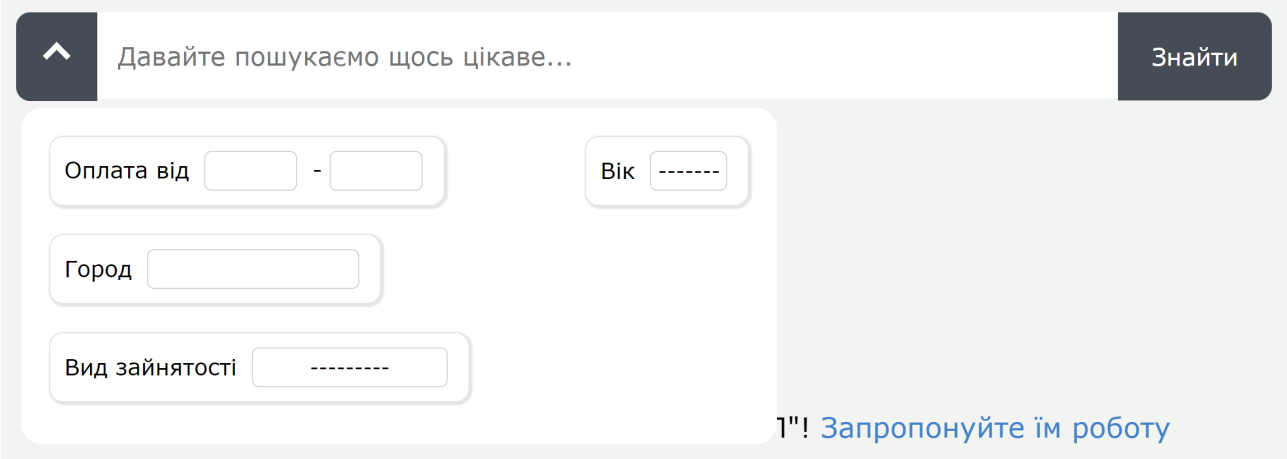


Рис. 2.11. Меню фільтрів

Нижче наведено приклад пошуку вакансій. Якщо ми введемо дані, але за такими даними вакансій не буде знайдено, нам відобразиться відповідне повідомлення (рис. 2.12.-2.13.).

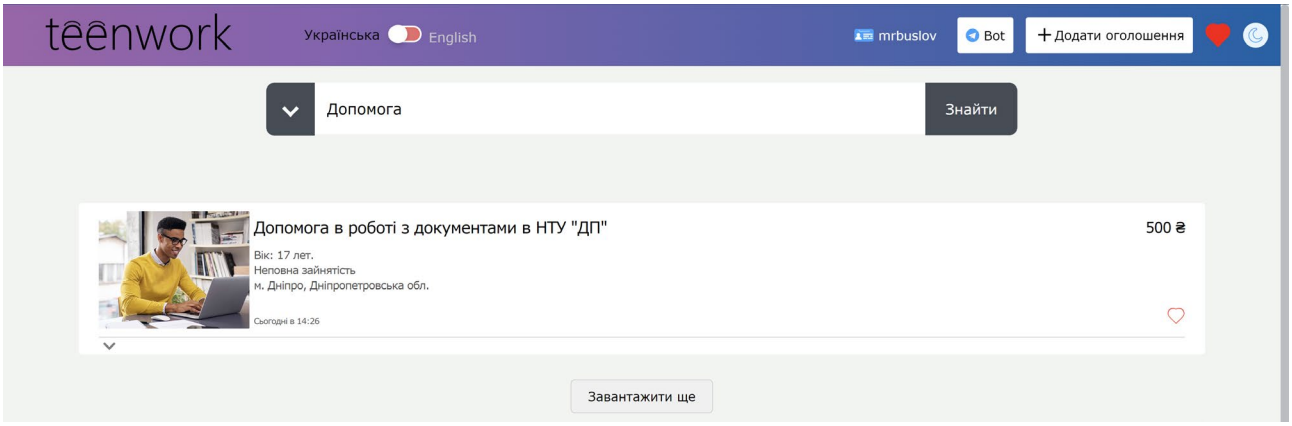


Рис. 2.12. Результати пошуку 1

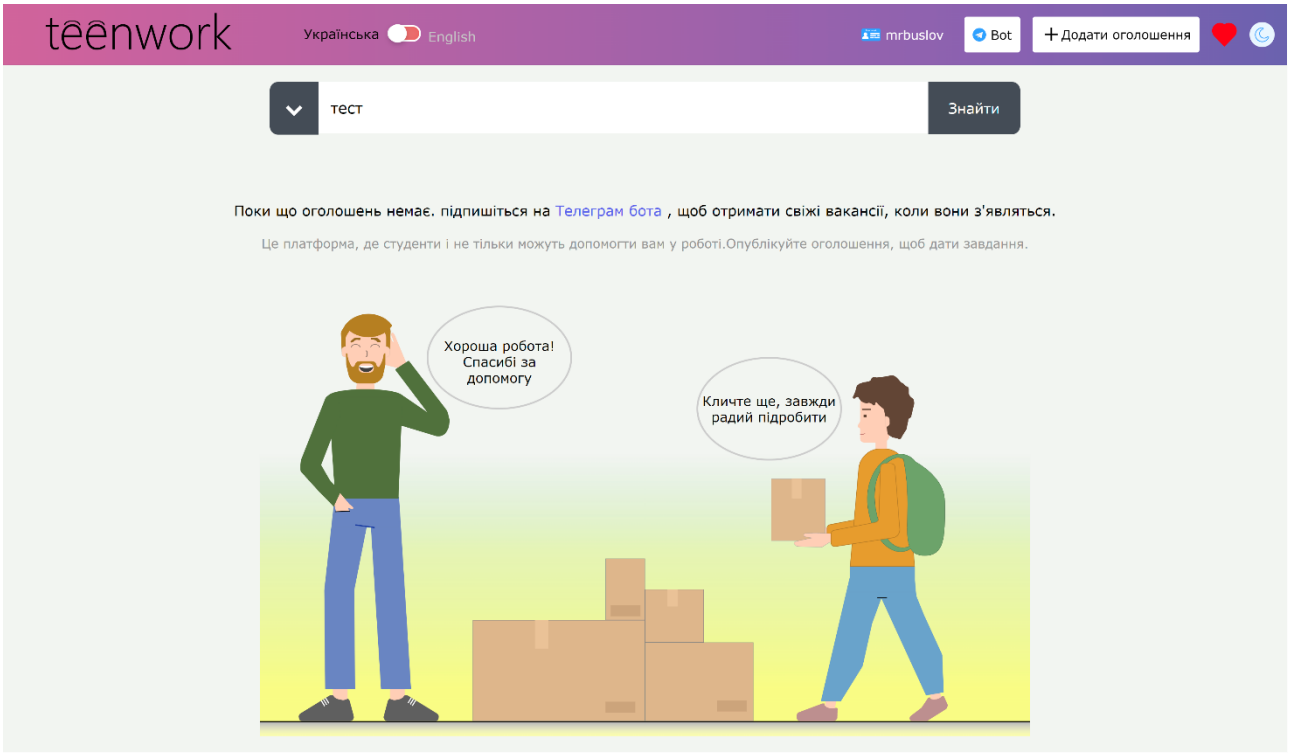
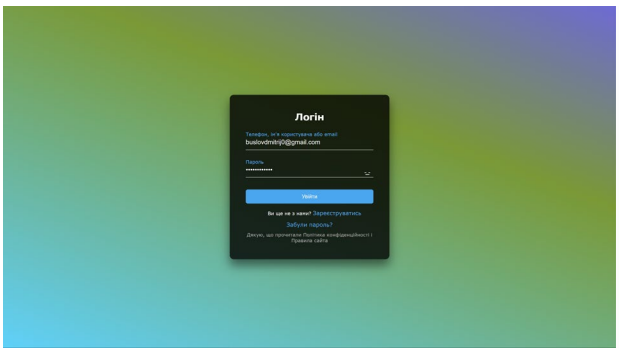


Рис. 2.13. Результати пошуку 2

При натисканні на посилання "Увійти до профілю" користувача перенаправляє на сторінку входу. На рис. 2.14 – 2.15 представлені сторінки входу та реєстрації.



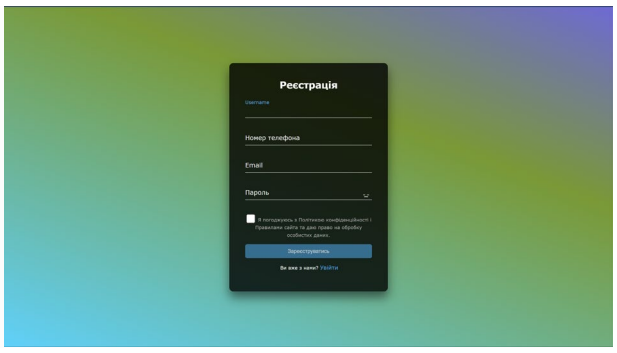


Рис. 2.14.-2.15. Сторінки входу та реєстрації

Якщо користувач введе невалідні дані, йому відобразяться відповідні помилки, представлені на рис. 2.16.

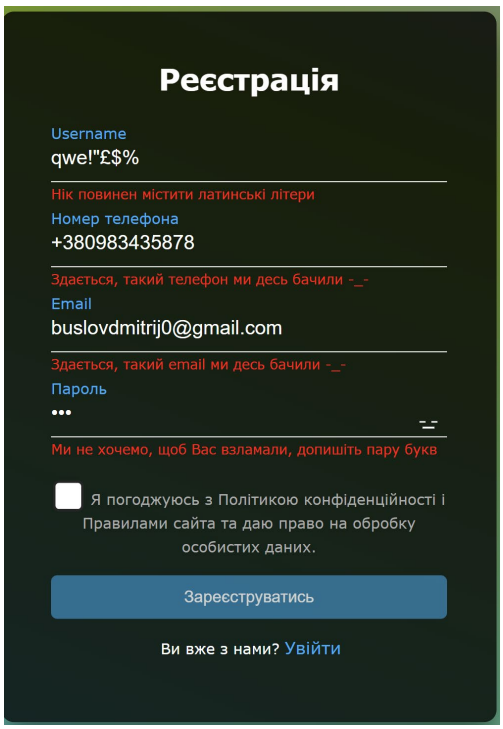


Рис. 2.16. Помилки, що відображаються в разі невалідних даних

Після реєстрації / входу користувача перенаправляє на сторінку зміни особистого профілю (рис. 2.17.).

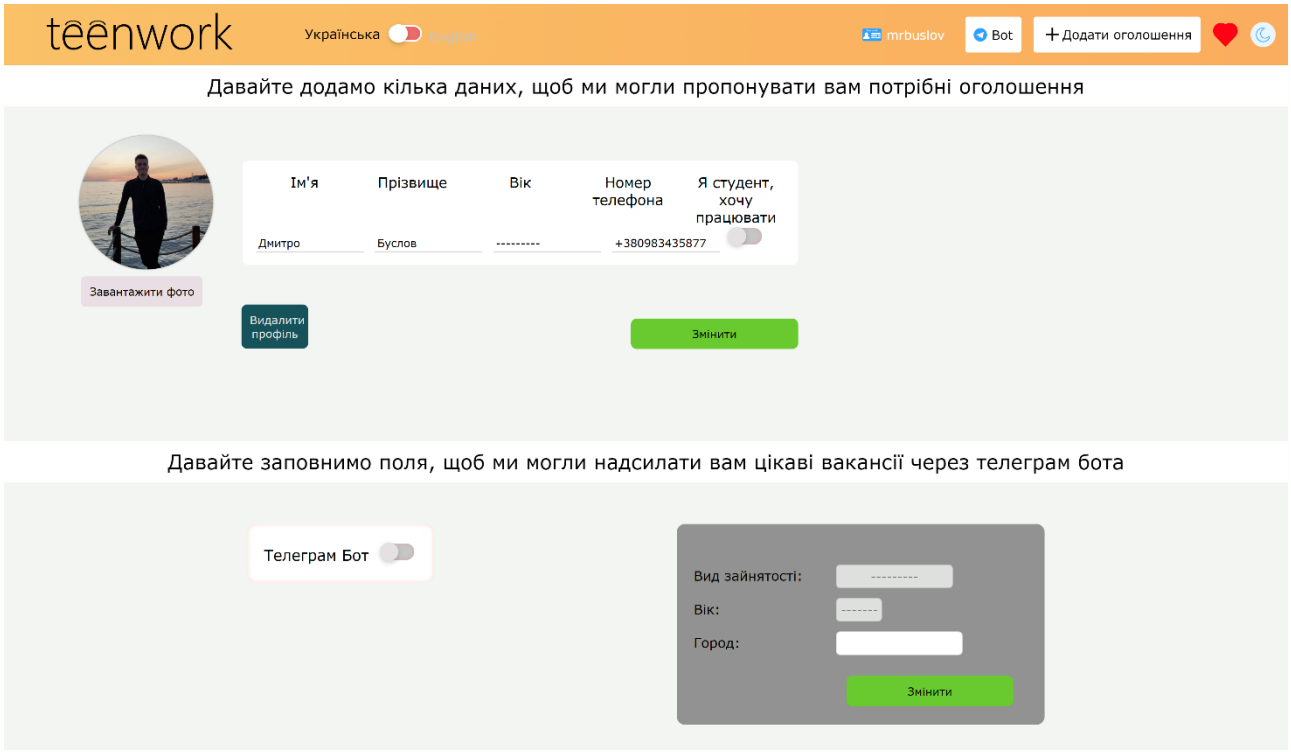


Рис. 2.17. Сторінка зміни особистого профілю

При натисканні на кнопку "Bot", користувача перенаправляє на сторінку пропозиції взаємодії з Телеграм Ботом (рис. 2.18.)

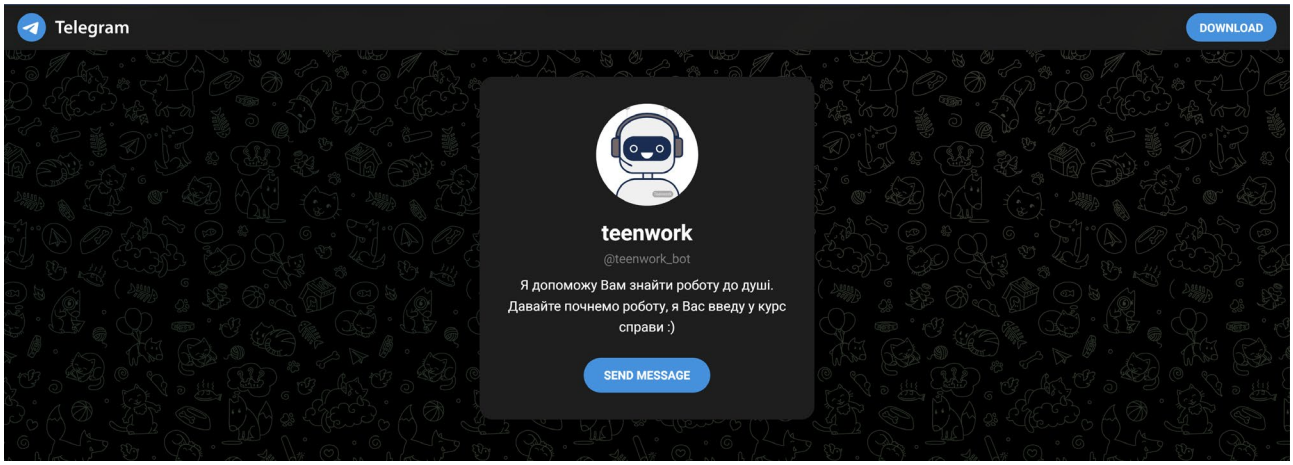


Рис. 2.18. Сторінка взаємодії з Телеграм Ботом

Як зареєстрований, так і незареєстрований користувачі можуть додати оголошення на платформу. Єдина відмінність - оголошення незареєстрованого користувача буде на майданчику 24 години, після чого буде видалено (рис. 2.19.).

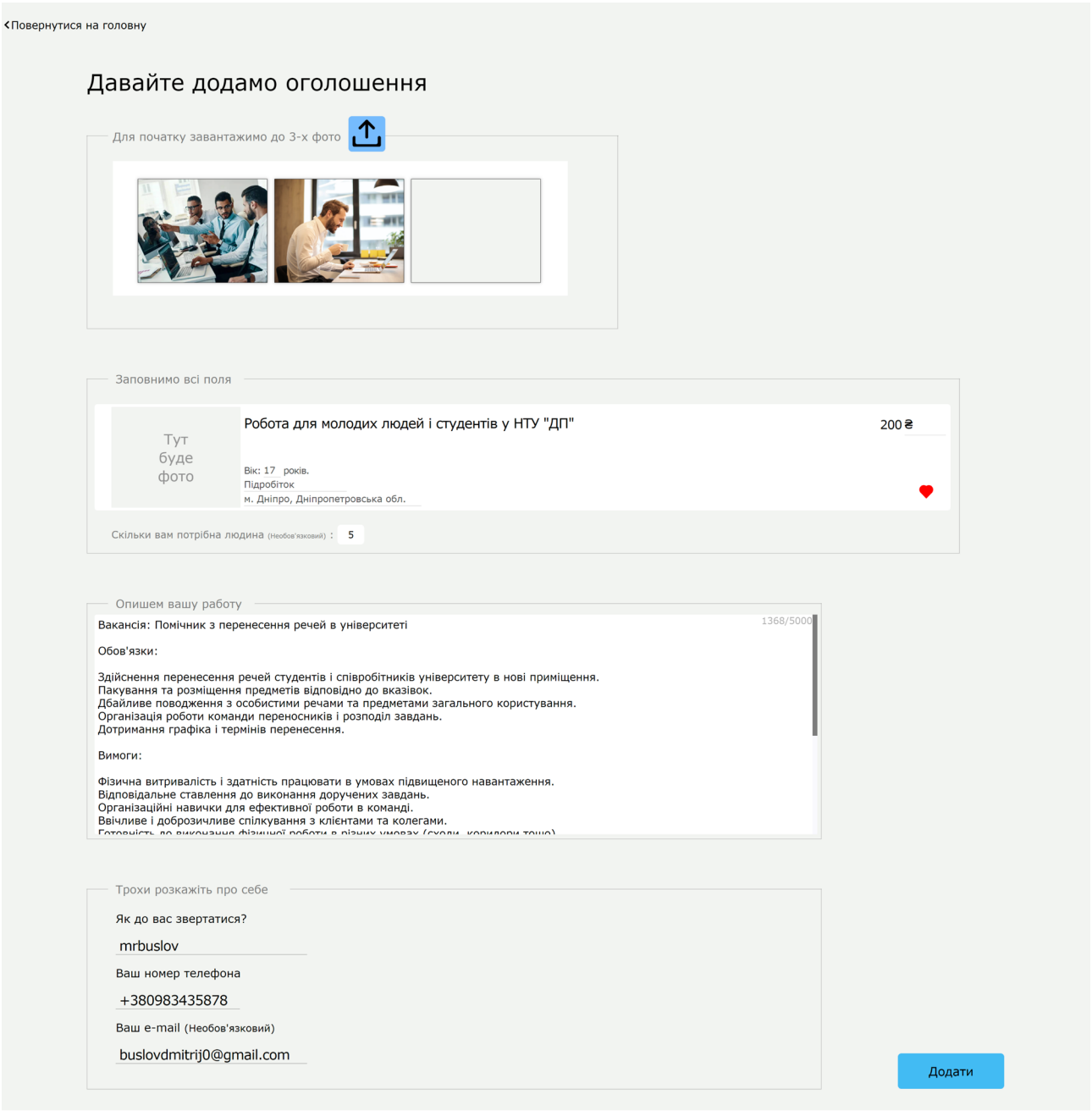


Рис. 2.19. Сторінка додавання оголошення

Якщо користувач перейде на вкладку "Мої оголошення", йому відкриється можливість також перейти на наступні сторінки (рис. 2.20.):

- Повідомлення
- Налаштування
- Телеграм Бот
- Прийняті працівники

Також користувачеві будуть доступні його оголошення.

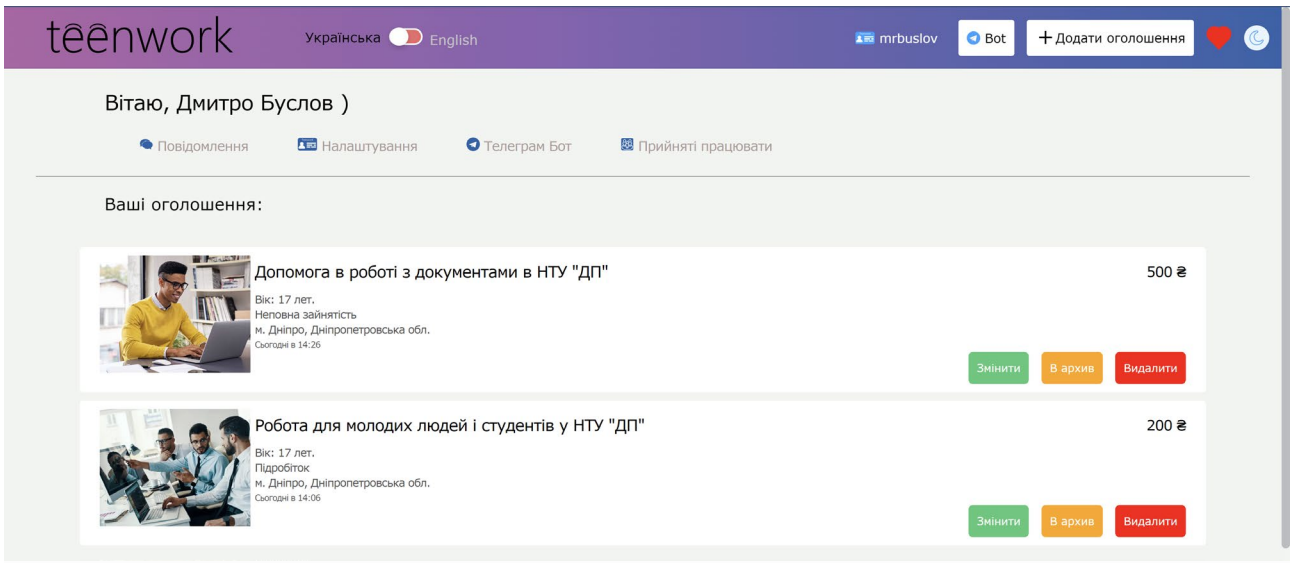


Рис. 2.20. Оголошення користувача

Якщо перейти по одному з оголошень, ми побачимо сторінку з повною інформацією про вакансію (рис. 2.21.).

Якщо оголошення переглядає не той користувач, який його створив, буде доступне поле для зв'язку через онлайнчат із роботодавцем (рис. 2.22.).

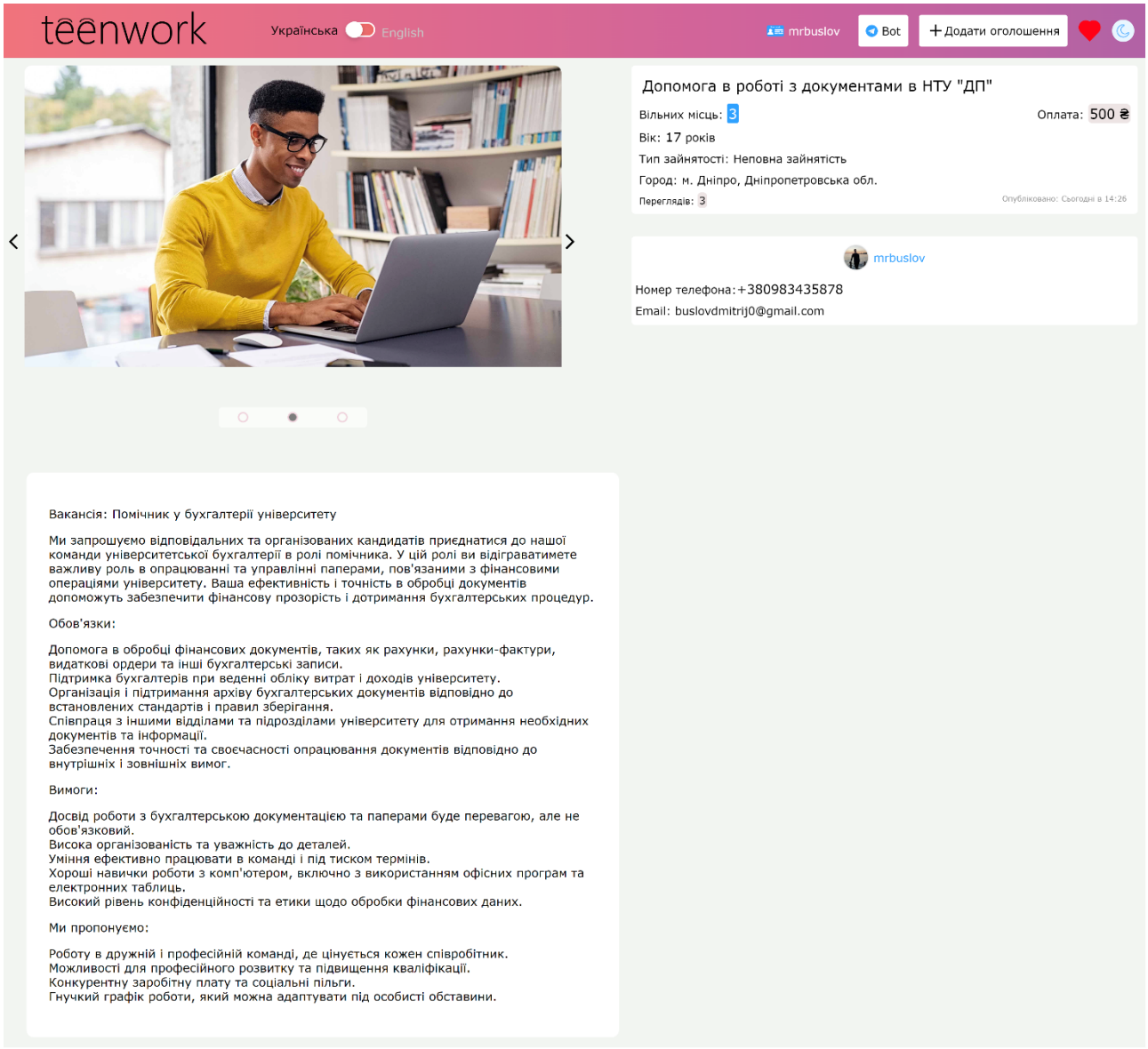


Рис. 2.21. Сторінка з повною інформацією про вакансію

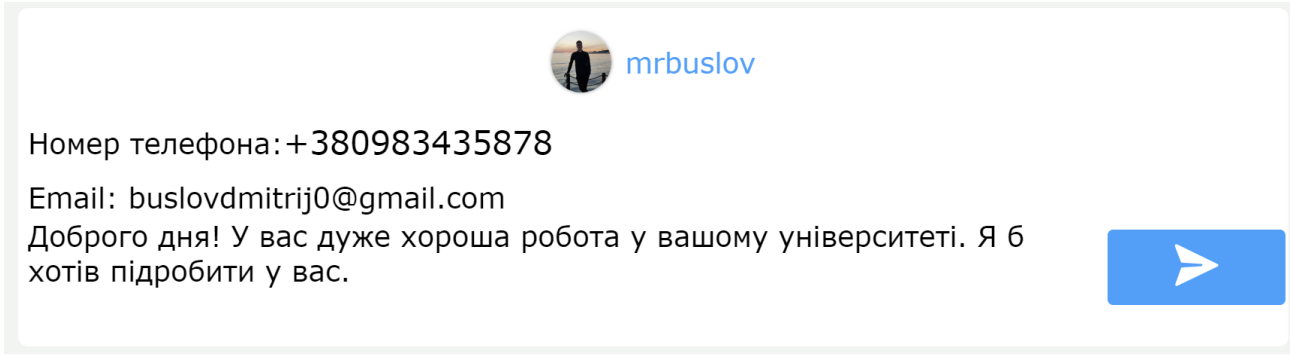


Рис. 2.22. Поле для зв'язку через онлайнчат із роботодавцем

Після надсилання повідомлення в обох користувачів (студента і роботодавця) з'являється чат у списку чатів (рис. 2.23.).

Рис. 2.23. Список чатів

Коли ми переходимо на чат, у роботодавця присутня зелена галочка, натиснувши на яку надсилається повідомлення від імені роботодавця, що людину прийнято на роботу (рис. 2.24.-2.25.).

Цей користувач автоматично додається до списку працівників на цій вакансії - для зручного менеджменту людьми (рис. 2.26.)

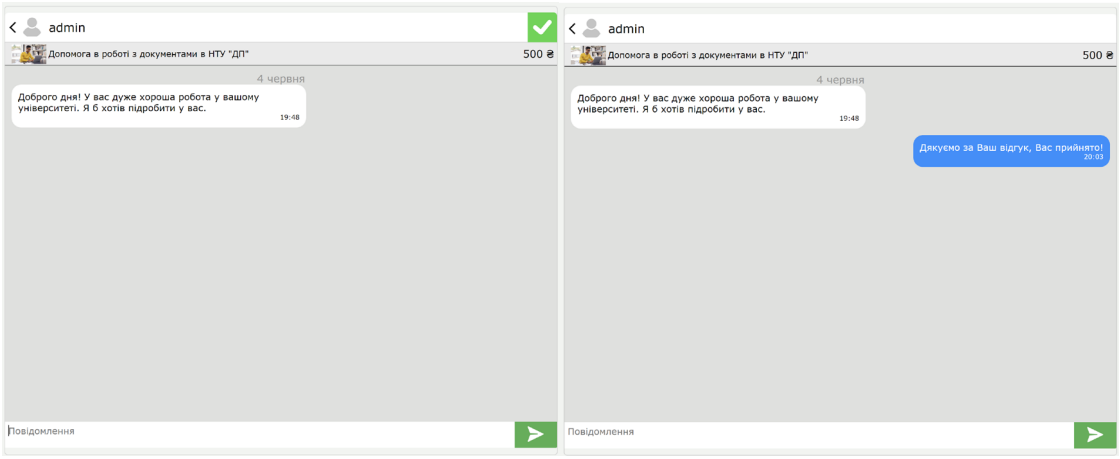


Рис. 2.24.-2.25. Повідомлення чату

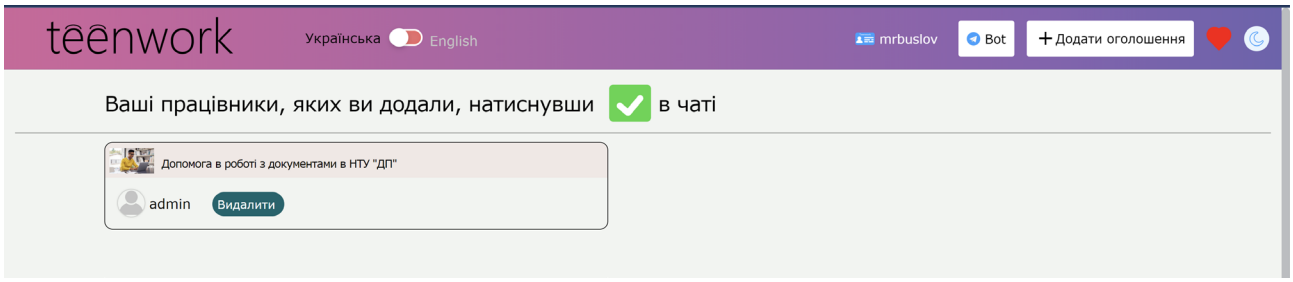


Рис. 2.26. Список найнятих працівників

Зареєстрований користувач може додати вподобані вакансії собі в акаунт (рис. 2.27.-2.28.)

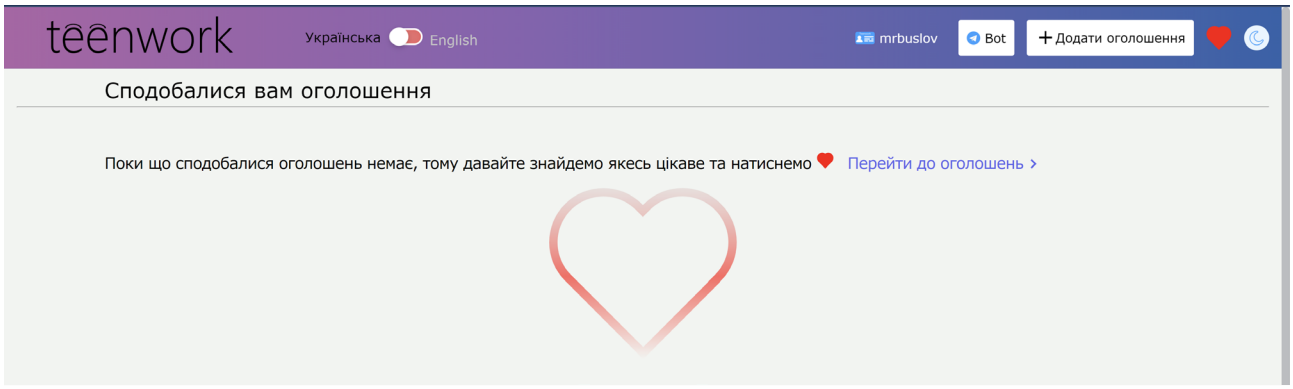


Рис. 2.27. Сторінка з вподобаними вакансіями (порожня)

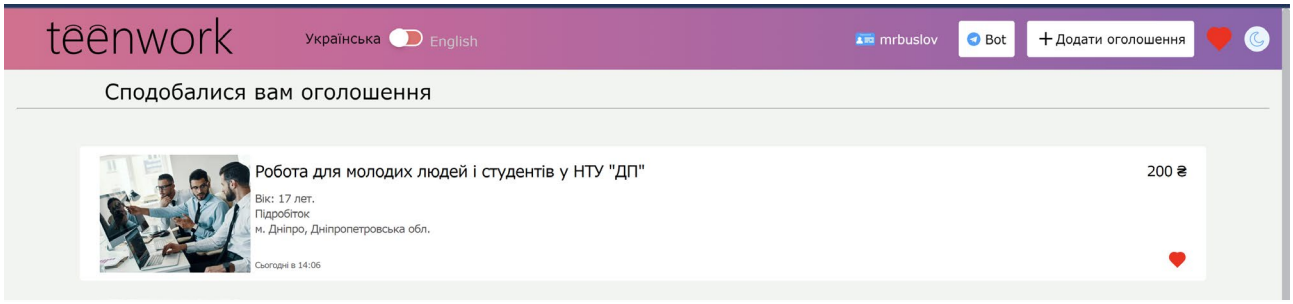


Рис. 2.28. Сторінка з вподобаними вакансіями (повна)

Адміністраторам платформи за допомогою Телеграм Бота надходить повідомлення про те, що додано нову вакансію (рис. 2.29.) [20].

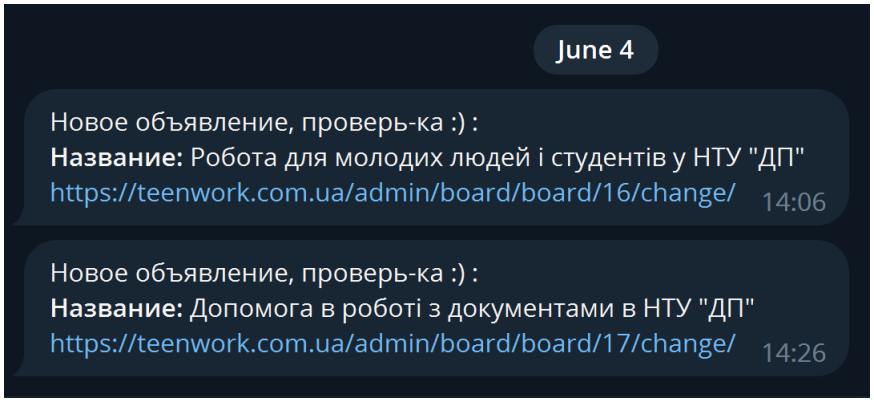


Рис. 2.29. Повідомлення про те, що додано нову вакансію

Можна поспілкуватися з ботом, запитавши, що він уміє робити (рис. 2.30.)

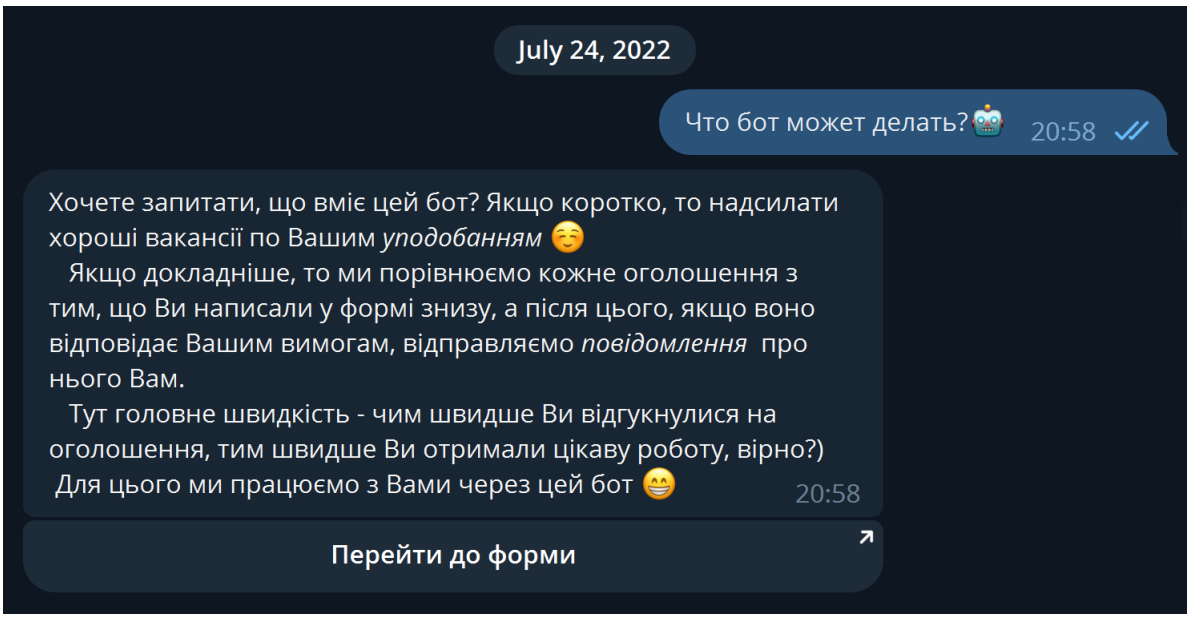


Рис. 2.30. Спілкування з ботом

Таке повідомлення приходить користувачам, які підписалися на розсилку нових вакансій відповідно до їхніх вимог через Телеграм бота (рис. 2.31.).

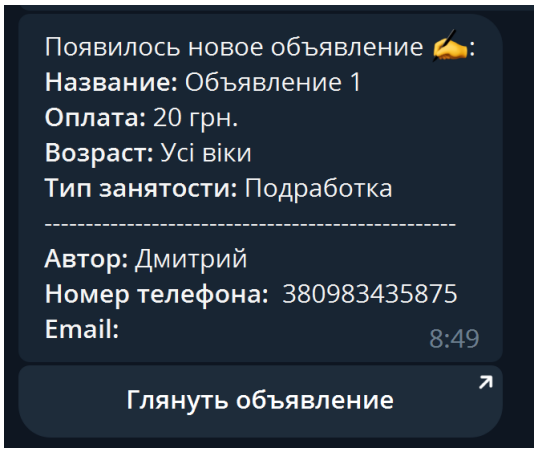


Рис. 2.31. Повідомлення про нову вакансію в Телеграм Боті

Також є вбудована у фреймворк Django адмінпанель, яка дає змогу адміністраторам керувати всіма моделями та об'єктами в них (рис 2.32.- 2.34.).

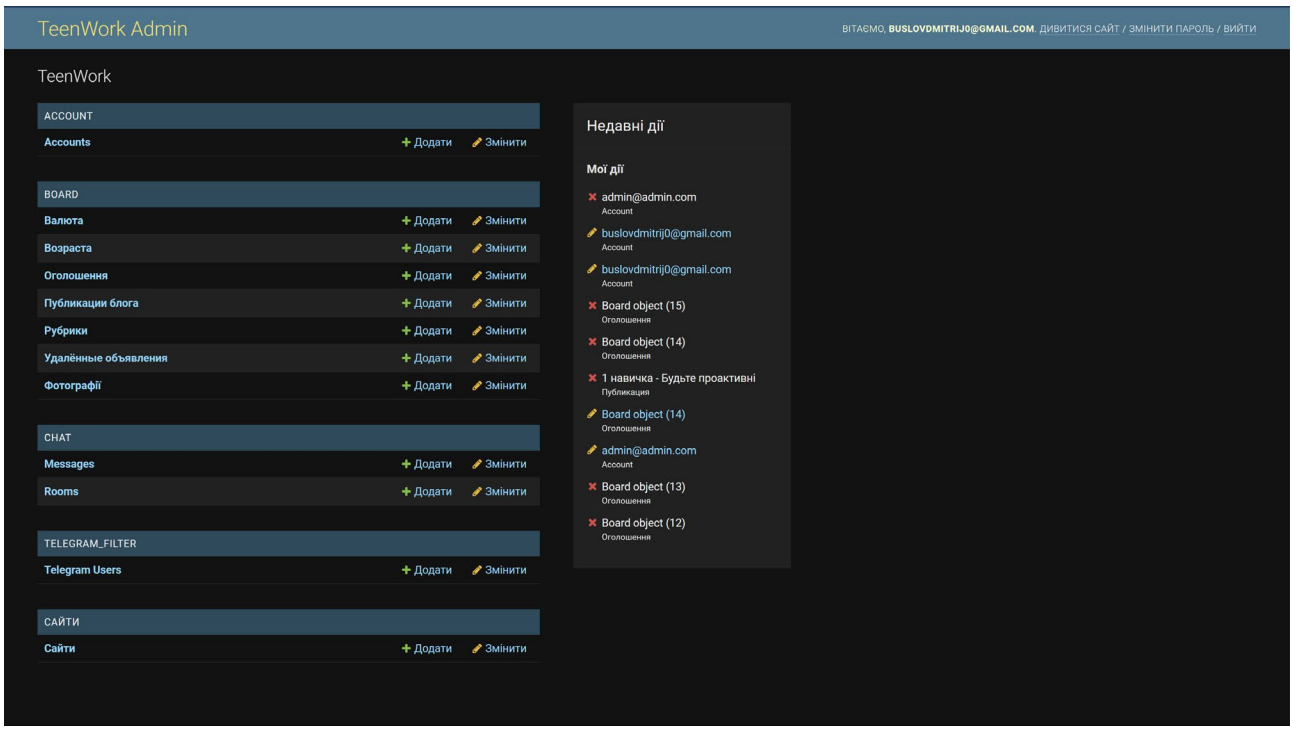


Рис. 2.32. Головна сторінка адмін-панелі.

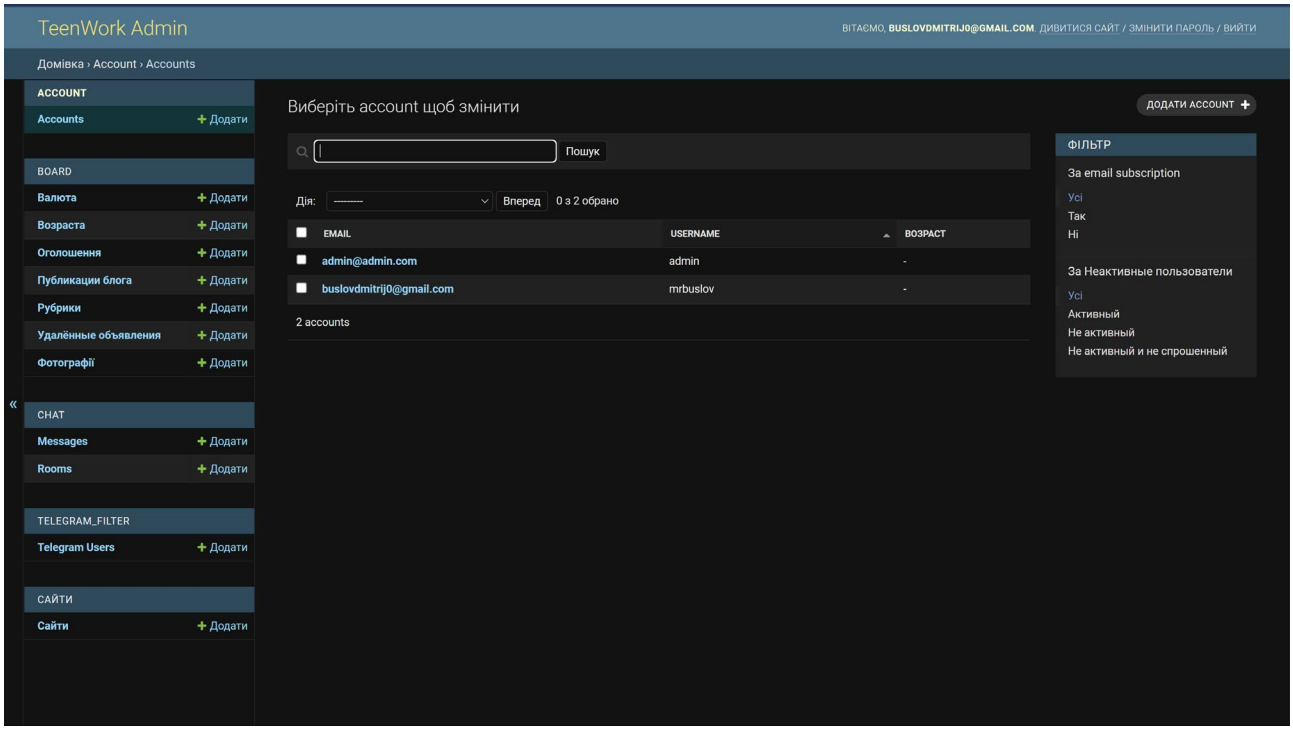


Рис. 2.33. Сторінка зі списком акаунтів в адмін-панелі.

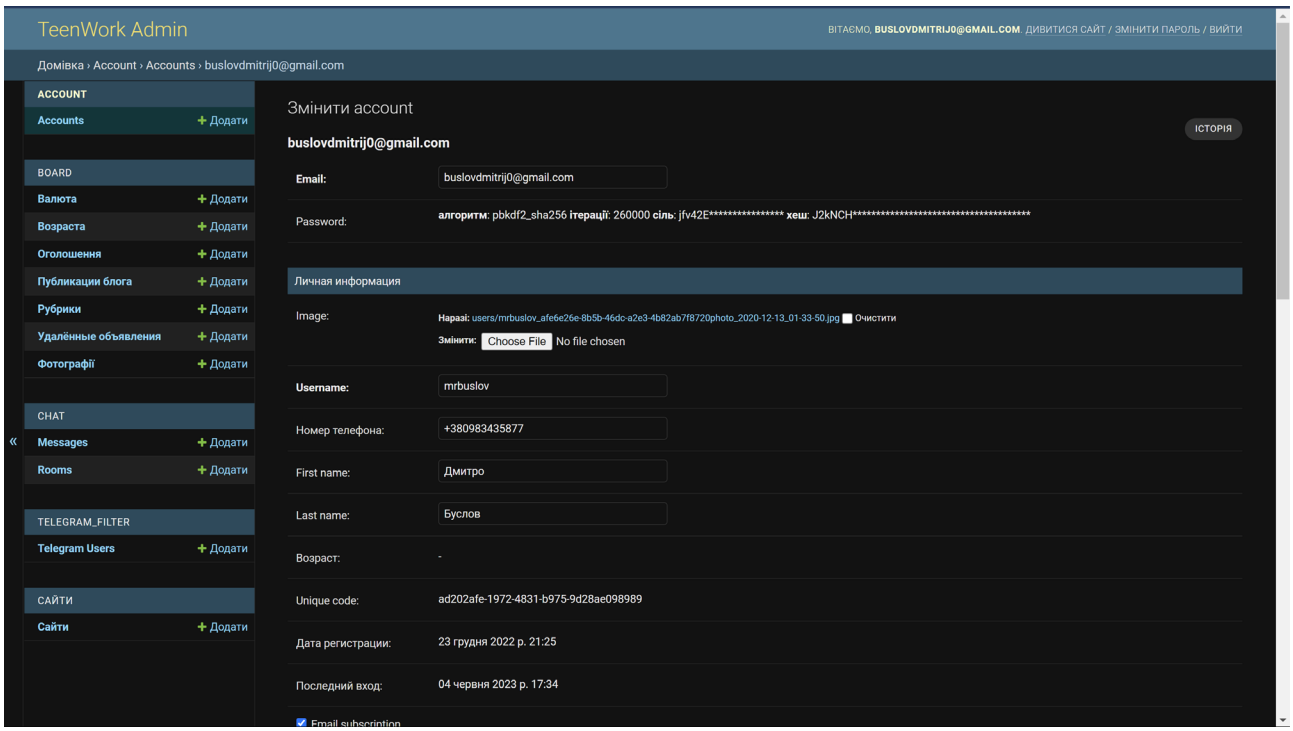


Рис. 2.34. Сторінка з деталями одного з акаунтів в адмін-панелі.

Є можливість вести блог платформи (додавання доступне тільки для адміністраторів) (рис. 2.35.- 2.36.)

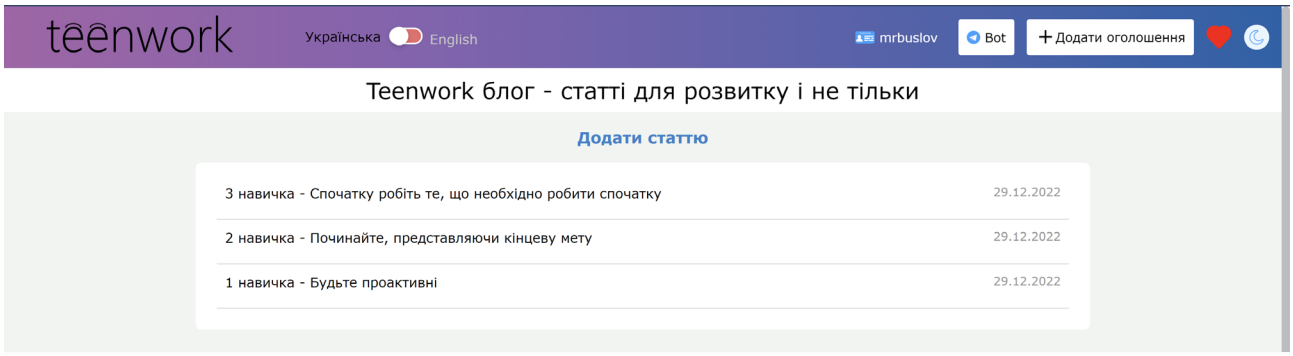


Рис. 2.35. Список усіх статей у блозі.

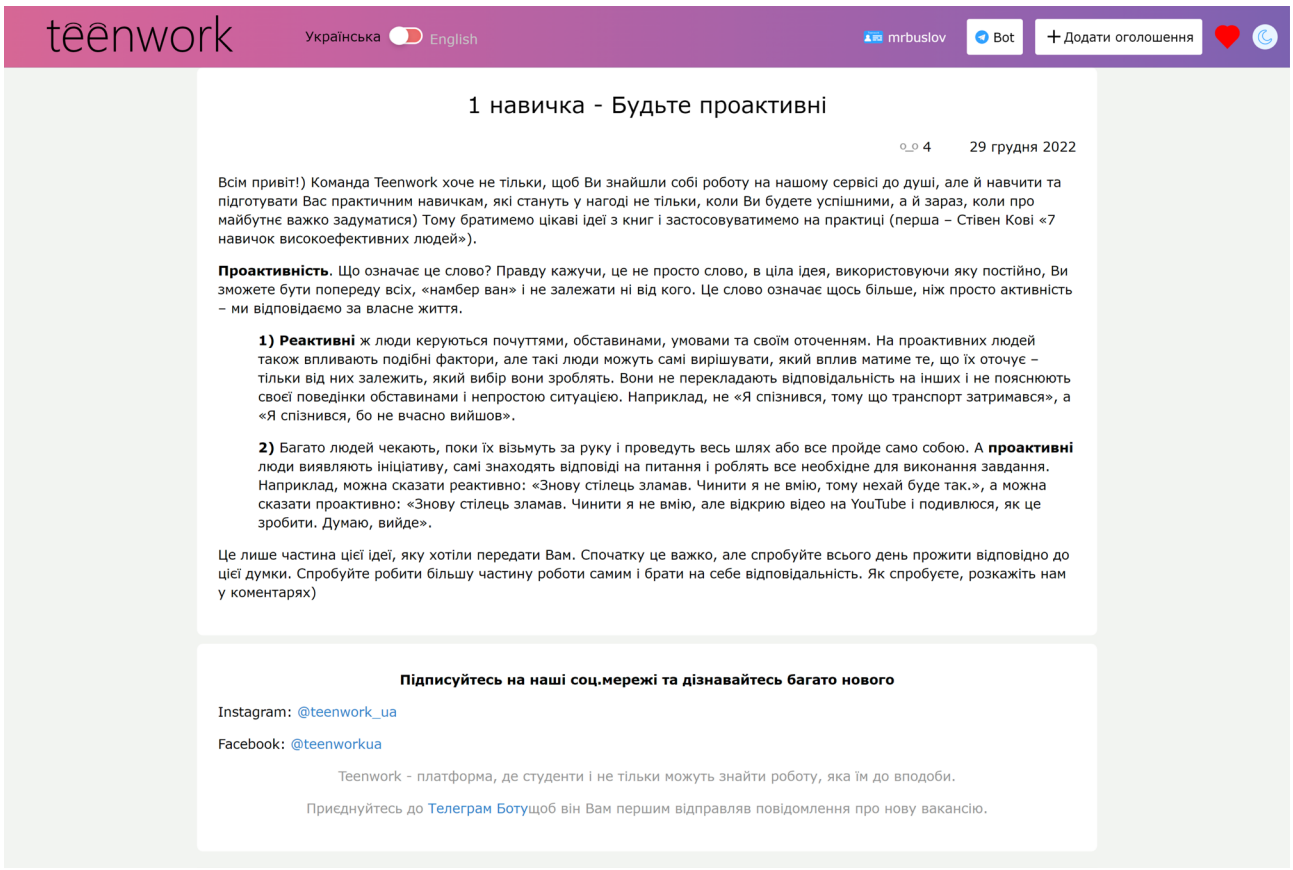


Рис. 2.36. Сторінка однієї статті.

Також є такі сторінки:

- для роботодавців;
- політика конфіденційності;
- правила сайта;
- офіційне працевлаштування молоді;
- як працює "публікація на 24ч" (рис. 2.37.);
- cookies (надкусане печиво) (рис. 2.38.);
- про нас (рис. 2.39.).

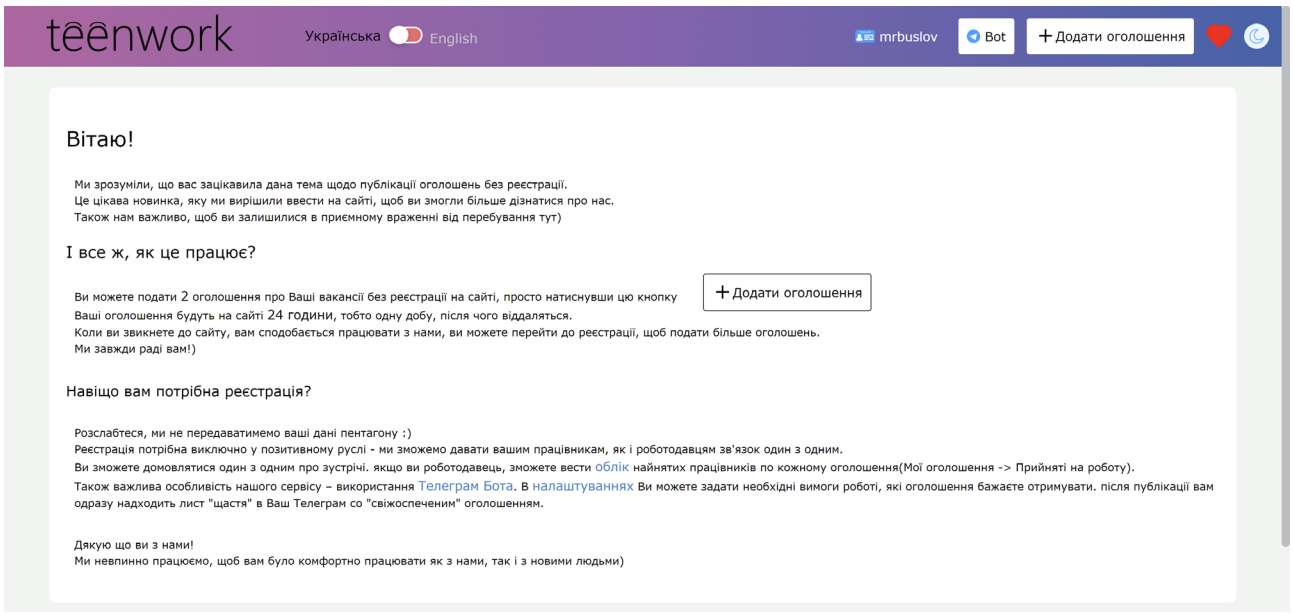


Рис. 2.37. Сторінка «як працює "публікація на 24ч"»

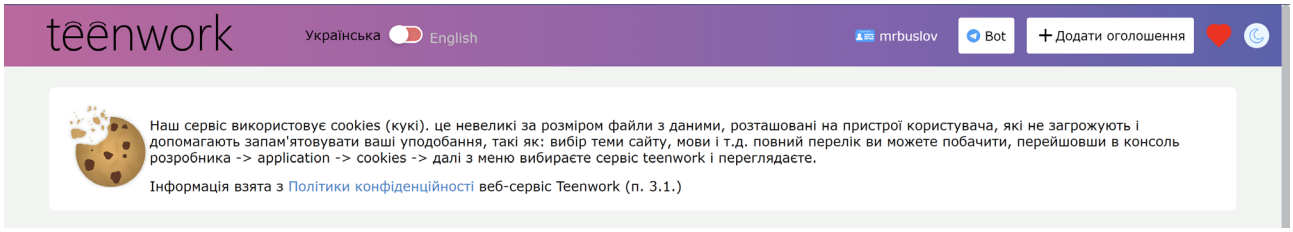


Рис. 2.38. Сторінка cookies (надкусане печиво)

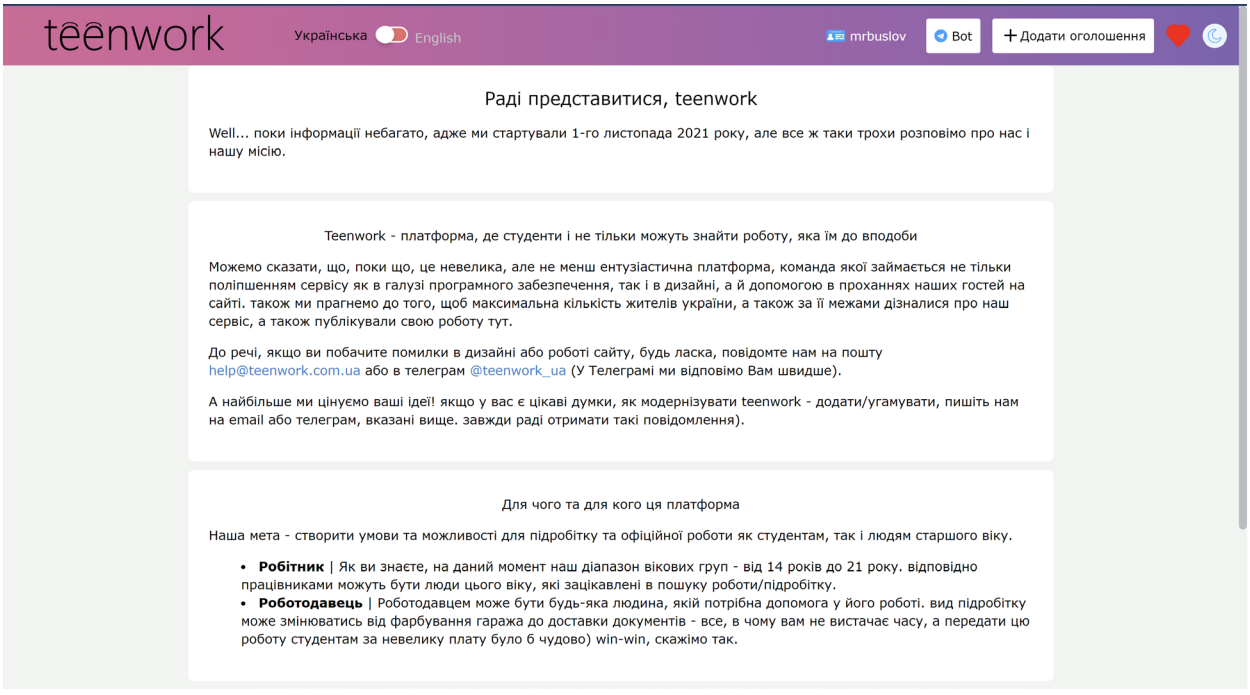


Рис. 2.39. Сторінка про Teenwork

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості розробки програмного забезпечення

Вихідні дані:

1. передбачуване число операторів програми – 2241;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста – 220 грн/год [22];

Вивчивши інформацію, представлену на сайті qubit-labs.com (https://qubit-labs.com/developer-salary-ukraine/) і прочитавши відповідну статтю, можна зробити висновок, що середня заробітна плата Junior Python Developer'а становить 1000 доларів. На момент написання цієї записки, курс долара Національного банку України становить 36,5 гривень за 1 долар (див. рис. 3.1.) [25]. Отже, при заробітній платі в розмірі 1000 доларів, це еквівалентно 36 500 гривень на місяць. У програміста зазвичай робочий графік, що складається з 21 робочого дня по 8 годин на день. Отже, погодинна ставка становитиме (36 500 гривень / 8 годин) / 21 робочий день = 217 гривень на годину.

Код цифровий	Код літерний	Кількість одиниць валюти	Назва валюти	Офіційний курс
840	USD	1	Долар США	36,5686

Рис. 3.1. Курс долар/гривня

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,5;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 0,69 грн/год.

Для створення кваліфікованої роботи не було потрібно додаткового обладнання або приміщення. Натомість для розрахунку вартості машино-години комп'ютера були враховані витрати на електроенергію та домашній інтернет. Згідно з даними Мінфіну на момент написання роботи, вартість 1 кВт/год становила 2,64 гривень (починаючи з 1 червня) [23]. Домашній інтернет коштував 80 гривень на місяць. Ноутбук споживав 80 Вт [15], тож вартість електроенергії за місяць використання в робочий час становила 80 * 2,64 * 8 * 21 / 1000 = 35,48 гривень. Таким чином, вартість машино-години комп'ютера становила (80 + 35,48) / 168 = 0,69 гривень на місяць.

У зв'язку з творчим характером роботи програміста, нормування праці в процесі розробки програмного забезпечення суттєво ускладнене. Тому трудомісткість розроблення ПЗ може бути розрахована з використанням системи моделей, що пропонують різні рівні точності оцінки. Трудомісткість розроблення програмного забезпечення може бути визначена за такою формулою:

$$t = t_о + t_и + t_а + t_п + t_{отл} + t_д \text{, людино-годин,}$$

де $t_о$ - витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

$t_а$ - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

Після підставлення значень умовне число операторів дорівнює:

$$Q = 2370 \cdot 1{,}4 \cdot (1 + 0{,}2) = 3764{,}88$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_и = \frac{Q*B}{(75..85)*k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності, стаж роботи 1 рік, тому коефіцієнт кваліфікації програміста = 1,1.

Будемо вважати збільшення витрат праці внаслідок недостатнього опису завдання як 1,5 (B = 1,5).

Після підставлення значень маємо:

$$t_u = (3764{,}88 \cdot 1{,}5) / (85 \cdot 1{,}1) = 73{,}1 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_а = \frac{Q}{(20..25)*k}, \text{ людино-годин.}$$

Підставивши значення:

$$t_a = 3764{,}88 / (23 \cdot 1{,}1) = 180{,}1 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі розраховується за формулою:

$$t_п = \frac{Q}{(20..25)*k}, \text{ людино-годин.}$$

Підставивши значення отримуємо витрати на складання програми по готовій блок-схемі:

$$t_n = 3764{,}88 / (25 \cdot 1{,}1) = 165{,}65 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин.}$$

Підставивши значення:

$t_{отл}$ = 3764,88/ (5 · 1,1) = 828,27 людино-годин.

– за умови комплексного налагодження завдання:

$$t^{k}_{отл} = 1{,}5 \cdot t_{отл}$$, людино-годин.

$t^{k}_{отл}$ = 1,5 · 828,27 = 1242,4 людино-годин.

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15..20) \cdot k}$$, людино-годин.

$t_{др}$ = 3764,88/ (20 · 1,1) = 207,1 людино-годин.

$T_{до}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}$$, людино-годин.

$t_{до}$ = 0,75 · 207,1 = 155,3 людино-годин.

Виходить що витрати праці на підготовку документації:

$t_{д}$ = 207,1 + 155,3 = 362,4 людино-годин.

Отже підставивши всі найдені значення у першу формулу маємо:

t = 73,1 + 180,1 + 165,65 + 828,3 + 362,4 + 50 = 1659,4 людино-години.

3.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$K_{ПО} = З_{ЗП} + З_{МВ}$$, грн.

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн},$$

де: t - загальна трудомісткість, людино-годин;

$C_{пр}$ - середня годинна заробітна плата програміста, грн/година

Середня плата за одну годинну роботи програміста становить 217 грн., тому:

$$З_{ЗП} = 1659{,}4 \cdot 217 = 365076{,}9019 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч} \text{ грн},$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (0,69 грн/год).

Підставивши значення:

$$З_{мв} = 828{,}27 * 0{,}687390476 = 569{,}35 \text{ грн.}$$

Отже витрати на створення програмного продукту будуть складати:

$$К_{ПО} = 365076{,}9019 + 569{,}34 = 365646{,}2492 \text{ грн.}$$

Формула для розрахунку очікуваного періоду створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс},$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Очікуваний період створення ПЗ:

$$T = 1659{,}440463 / 176 \approx 9{,}4 \text{ міс.}$$

Висновок: Остаточні витрати на розробку цього програмного продукту становлять 360667,93 гривень, що є мінімальною вартістю. Розрахунки також показують, що термін створення становить 9,4 місяців. У цей період включено час на виправлення помилок або поліпшення завдань у зв'язку з їхньою неточністю.

ВИСНОВКИ

Під час роботи було розроблено інформаційну платформу для працевлаштування та підробітку студентів і молодих людей. Платформу реалізовано на мовах програмування Python і JS з використанням фреймворку Django і API TelegramBot.

Відповідно до завдання платформа реалізує такий функціонал і виконує таку роль:

Забезпечити молодим людям, а саме підліткам та студентам зручну, ефективну та надійну можливість знаходити підробіток шляхом встановлення контакту з роботодавцем через оголошення на платформі.

- Забезпечити роботодавцям швидку та якісну допомогу з боку підлітків та студентів.
- Запровадити систему обліку працівників, найнятих роботодавцем.
- Забезпечити студентам можливість знаходити оголошення, що відповідають їхнім інтересам і вимогам, за допомогою встановлення фільтрів, таких як ключові слова, вік, місто/область, тип зайнятості та оплата.
- Надсилати особисті повідомлення роботодавцям/студентам про зміни у вакансіях і появу нових вакансій відповідно.

Реляційна база даних забезпечує швидкий, надійний і структурований пошук необхідних записів за ключовими словами.

Під час виконання даної кваліфікаційної роботи також було встановлено складність розробленої системи і здійснено розрахунок вартості роботи зі створення програми, використовуючи середню заробітну плату FullStack Django/JS розробника.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стаття 188. Вік, з якого допускається прийом на роботу Розділ XIII. Праця молоді (ст. 187–200) Кодекс законів про працю України | Нормативна база України Factor. URL: https://i.factor.ua/law-39/section-194/article-3066/ (дата звернення: 29.05.2023)

2. Кодекс законів про працю | від 10.12.1971 № 322-VIII. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 06.06.2023)

3. Top 13 Programming Languages Trends in 2023 | Fireart. URL: https://fireart.studio/blog/top-programming-languages-that-will-rule-in-2021/ (дата звернення: 26.05.2023)

4. TIOBE Index - TIOBE. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 06.05.2023)

5. Top 10 Programming Languages to Learn in 2023 - GeeksforGeeks. URL: https://www.geeksforgeeks.org/top-10-programming-languages-to-learn-in-2022/ (дата звернення: 03.06.2023)

6. Paul Francis. Django Website Examples: 10 Popular Websites Built With Django. URL: https://uvik.net/blog/django-website-examples/ (дата звернення: 26.04.2023)

7. Attention Required! | Cloudflare. URL: https://www.affde.com/ru/pros-and-cons-of-django-web-framework-for-app-development.html (дата звернення: 03.06.2023)

8. Django vs flask - full comparison in 5 min - YouTube. URL: https://www.youtube.com/watch?v=NwKB4HZnIHM&ab_channel=Jelvix (дата звернення: 09.05.2023)

9. Django with Abstract Base Classes & Composition - Getting Started - Django Forum. URL: https://forum.djangoproject.com/t/django-with-abstract-base-classes-composition/17427 (дата звернення: 06.05.2023)

10. Moving from Inheritance to Composition in Django | by Rowan Hale | Skilljar Engineering | Medium. URL: https://medium.com/skilljar-

engineering/moving-from-inheritance-to-composition-in-django-a268563089d7 (дата звернення: 22.05.2023)

11. How To Use PostgreSQL with your Django Application on Ubuntu 20.04 | DigitalOcean. URL: https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-django-application-on-ubuntu-20-04 (дата звернення: 28.05.2023)

12. Databases | Django documentation | Django. URL: https://docs.djangoproject.com/en/4.2/ref/databases/ (дата звернення: 14.05.2023)

13. Django PostgreSQL Connection: 5 Easy Steps - Learn | Hevo. URL: https://hevodata.com/learn/django-postgresql/ (дата звернення: 24.05.2023)

14. What is django ORM. URL: https://www.tutorialspoint.com/what-is-django-orm (дата звернення: 25.04.2023)

15. Огляд та тестування ноутбука Acer Nitro 5 AN515-45 на базі нового процесора AMD Ryzen 7 5800H та відеокарти Nvidia GeForce RTX 3080 / Overclockers.ua. URL: https://www.overclockers.ua/notebook/acer-nitro-5-an515-45/all/#:~:text=Acer%20Nitro%205%20AN515-45%20%D0%BE%D1%81%D0%BD%D0%B0%D1%89%D0%B5%D0%BD%20%D0%B2%D0%B8%D0%B4%D0%B5%D0%BE%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D0%B9%20NVIDIA%20GeForce%20RTX,3080%20%D1%81%20%D0%BC%D0%BE%D1%89%D0%BD%D0%BE%D1%81%D1%82%D1%8C%D1%8E%2085%20%D0%92%D1%82 (дата звернення: 07.06.2023)

16. Django - Overview. URL: https://www.tutorialspoint.com/django/django_overview.htm (дата звернення: 28.05.2023)

17. Sarthak Kumar. Django : Class Based Views vs Function Based Views| Medium. URL: https://medium.com/@ksarthak4ever/django-class-based-views-vs-function-based-view-e74b47b2e41b (дата звернення: 23.04.2023)

18. The Structure of a Django Application - YouTube. URL: https://www.youtube.com/watch?v=jmX27FrCqqs (дата звернення: 03.05.2023)

19. Mastering Django: Structure - Mastering Django. URL: https://masteringdjango.com/django-tutorials/mastering-django-structure/ (дата звернення: 20.04.2023)

20. Telegram Bot API. URL: https://core.telegram.org/bots/api#authorizing-your-bot (дата звернення: 05.05.2023)

21. Security in Django | Django documentation | Django. URL: https://docs.djangoproject.com/en/4.2/topics/security/ (дата звернення: 05.05.2023)

22. Elena Kononchuk. How Much Does It Cost to Hire Developers in Ukraine?. URL: https://qubit-labs.com/developer-salary-ukraine/ (дата звернення: 29.05.2023)

23. Данило Крамаренко. Світло подорожчало. Яким буде тариф на електроенергію та як зростуть платіжки. URL: https://www.rbc.ua/rus/news/svitlo-podorozhchalo-kim-bude-tarif-elektroenergiyu-1685533449.html (дата звернення: 18.04.2023)

24. Julia Korsun. Why We Use Django Framework & What Is Django Used For | Django Stars. URL: https://djangostars.com/blog/why-we-use-django-framework/ (дата звернення: 22.05.2023)

25. Офіційний курс гривні щодо іноземних валют. URL: https://bank.gov.ua/ua/markets/exchangerates (дата звернення: 01.06.2023)

26. 77-а студентська науково-технічна конференція "Тиждень Студентської Науки" (16-20 травня 2022 р.), Секція 11 – Інформаційні та телекомунікаційні технології / Буслов. Д. Ю. Наукові керівники: к.т.н., доц. Спірінцев В. В. к.т.н., доц. Ширін А. Л.: НТУ «Дніпровська політехніка», 2022

27. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 121 Інженерія програмного забезпечення / В.В. Спірінцев, О.С. Шевцова, І.М. Удовик; Д : НТУ «Дніпровська політехніка», 2022. – 60 с.

ДОДАТОК А

КОД ПРОГРАМИ

account\views.py

```
from django.shortcuts import render
from board.models import Board
from account.models import Account
from board.models import TeenworkBlog
from django.shortcuts import get_object_or_404, redirect, render
from django.contrib.auth import authenticate, login, logout
from django.contrib.sites.shortcuts import get_current_site
from .utils import token_generator
from django.conf import settings
from django.db.models import Q

from django.urls import reverse, reverse_lazy
from django.utils.encoding import force_bytes, force_text, DjangoUnicodeDecodeError
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
import datetime
from datetime import datetime
from django.contrib.auth.decorators import login_required
from django.core.mail import EmailMultiAlternatives
import re
from .forms import ProfileEditForm
from telegram_filter.models import Telegram
from telegram_filter.forms import TelegramForm
from django.views.decorators.csrf import csrf_exempt
from django.template.loader import get_template



import threading
class EmailThreading(threading.Thread):
    def __init__(self, email_message):
        self.email_message = email_message
        threading.Thread.__init__(self)
```

```
    def run(self):
        self.email_message.send()




@login_required(login_url='/login/')
def profile(request):
    board_obj = Board.objects.filter(author=request.user)
    time_now1 = str(datetime.today().date().day).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)
    time_now2 = str(datetime.today().date().day-1).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)


    context={
        'board_obj':board_obj,
        'time_now1':time_now1,
        'time_now2':time_now2,
    }


    return render(request, 'user/profile.html', context)


def account(request, username):
    if Account.objects.filter(username=username).exists():
        board_obj = Board.objects.filter(Q(author=Account.objects.get(username=username).pk), Q(status='published')|Q(status='24hour'))
        user = Account.objects.get(username=username)
        time_now1 = str(datetime.today().date().day).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)
        time_now2 = str(datetime.today().date().day-1).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)


        context={
            'board_obj':board_obj,
            'requested_user':user,
            'time_now1':time_now1,
            'time_now2':time_now2,
        }


        if user == request.user:
```

```
            return redirect('/profile/')


        return render(request, 'user/account.html', context)
    else:
        return HttpResponse('Такий Юзер зник о_о')


@login_required(login_url='/login/')
@csrf_exempt
def profile_edit(request):

    if request.is_ajax():
        data = request.POST.get('data', None)
        isTelegram = request.POST.get('isTelegram', None)


        tel = Telegram.objects.get(person = request.user)
        if data:
            if data == 'checked':
                tel.telegram = True
                data = 'true'
            else:
                tel.telegram = False
                data = 'false'
            tel.save()
            return JsonResponse(data, safe=False)
        elif isTelegram:
            if tel.telegram == True:
                isTelegram = 'true'
            else:
                isTelegram = 'false'
            return JsonResponse(isTelegram, safe=False)


    username = request.user.username
```

```
    account = Account.objects.get(username=username)
    form = ProfileEditForm(instance=account)
    if request.method == 'POST' and 'profile_edit_btn' in request.POST:
        form = ProfileEditForm(request.POST, instance=account)
        if form.is_valid():
            post = form.save(commit=False)
            if request.FILES.get('img'):
                post.image = request.FILES.get('img')
            post.save()
        return redirect('account:profile_edit')

    tlg_obj = Telegram.objects.get(person=request.user)
    tlg_form = TelegramForm(instance=tlg_obj)
    if request.method =='POST' and 'tlg_btn' in request.POST:
        tlg_form = TelegramForm(request.POST, instance=tlg_obj)
        if tlg_form.is_valid():
            tel = tlg_form.save(commit=False)
            tel.person = request.user
            tlg_form.save()
            return redirect('account:profile_edit')
        else:
            return redirect('account:profile_edit')

    if request.method == 'POST' and 'delete_account_btn' in request.POST:
        for board_obj in Board.objects.filter(author=request.user):
            board_obj.delete()
        Account.objects.get(id=request.user.id).delete()
        return redirect('board:index')

    return render(request, 'user/profile_edit.html', {'form':form, 'account':account, 'tlg_form':tlg_form,})




def registration(request):
    if request.method == 'POST':
```

```
    username = request.POST.get('username')
    email = request.POST.get('email')
    password = request.POST.get('password')
    phone_number = request.POST.get('phone_number')

    if request.is_ajax():
        data=''
        username_check = request.POST.get('username', None)
        email_check = request.POST.get('email', None)
        phone_number_check = request.POST.get('phone_num', None)

        if username_check:
            if Account.objects.filter(username=username_check.lower()).exists():
                data='username_taken'
            else:
                data='username_free'
        elif email_check:
            if Account.objects.filter(email=email_check.lower()).exists():
                data='email_taken'
            else:
                data='emil_free'
        elif phone_number_check:
            if Account.objects.filter(phone_number=phone_number_check).exists():
                data='phone_taken'
            else:
                data='phone_free'



        return JsonResponse(data, safe=False)


    if Account.objects.filter(username=username.lower()).exists() or
Account.objects.filter(email=email.lower()).exists() or Account.objects.filter(phone_number=phone_number).exists():
        return redirect('account:registration')


    if re.match("^[a-z0-9_]+$", username.lower()) == None:
        return redirect('account:registration')
```

```
        user = Account.objects.create_user(username=username.lower(), password=password, email=email.lower(), phone_number=phone_number)
        user.is_active = False
        user.save()

        tlg_user = Telegram.objects.create(person=user)
        tlg_user.save()

        uidb64 = urlsafe_base64_encode(force_bytes(user.pk))

        domain = get_current_site(request).domain
        link = reverse('account:activate', kwargs={'uidb64':uidb64,'token':token_generator.make_token(user)})

        activate_url='https://' + domain + link

        # email_context = {
        #     'user':user,
        #     'activate_url': activate_url,
        # }

        # email_subject = ''
        # # email_body = ''
        # email_body = render_to_string('registration/email.html', email_context)
        # email_msg = EmailMessage(
        #     email_subject,
        #     email_body,
        #     settings.EMAIL_HOST_USER,
        #     [email],
        # )

        # # email_msg.send(fail_silently=False)

        # EmailThreading(email_msg).start()
```

```
        mail_title = "Активація"
        variables = {
            'activate_url': activate_url,
            'watch_here': 'https://' + domain + '/rules/',
        }
        html = get_template('registration/email.html').render(variables)
        text = f'https://teenwork.com.ua/static/img/icons/teenwork.png - ( наше лого :) )\nДавайте підтвердимо Ваш аккаунт\nЩоб почати користуватися Teenwork, просто натисніть кнопку підтвердження адреси електронної пошти нижче:\nПідтвердити\n{activate_url}\nУ вас є питання? Подивіться тут.\nhttps://teenwork.com.ua/rules/\nTeenwork - платформа, де підлітки і не тільки можуть знайти роботу, яка їм до вподоби.'

        msg = EmailMultiAlternatives(
            mail_title,
            text,
            settings.EMAIL_HOST_USER,
            [email])
        msg.attach_alternative(html, "text/html")
        # msg.send(fail_silently=False)
        EmailThreading(msg).start()


        return render(request, 'registration/registration_success.html')
    else:
        return render(request, 'registration/registration.html')


def resend_activation_email(request):
    if request.method == 'POST':
        email = request.POST.get('email')

        user = Account.objects.get(email=email)

        uidb64 =urlsafe_base64_encode(force_bytes(user.pk))
```

```
        domain = get_current_site(request).domain
        link = reverse('account:activate', kwargs={'uidb64':uidb64,'token':token_generator.make_token(user)})

        activate_url='https://' + domain + link



        mail_title = "Активація"
        variables = {
            'activate_url': activate_url,
            'watch_here': 'https://' + domain + '/rules/',
        }
        html = get_template('registration/email.html').render(variables)
        text = get_template('registration/email.html').render(variables)

        msg = EmailMultiAlternatives(
            mail_title,
            text,
            settings.EMAIL_HOST_USER,
            [email])
        msg.attach_alternative(html, "text/html")
        EmailThreading(msg).start()



        return render(request, 'registration/resend_activation_email_success.html')
    else:
        return render(request, 'registration/resend_activation_email.html')




from django.views.generic import View
class VerificationView(View):
    def get(self,request, uidb64, token):
        try:
            uid = force_text(urlsafe_base64_decode(uidb64))
            user = Account.objects.get(pk=uid)
```

```
        except Exception as e:
            user = None

        if user is not None:
            if not token_generator.check_token(user,token):
                return redirect('account:login')



            if user.is_active:
                return redirect('account:login')
            user.is_active = True
            user.save()

            return redirect('account:login')
        else:
            raise Exception



def user_login(request):
    if request.user.is_authenticated:
        return redirect('board:index')
    else:
        if request.method == 'POST':

            if request.is_ajax():
                data=''
                email_check = request.POST.get('email_login', None)

                if email_check:
                    if Account.objects.filter(email=email_check.lower()).exists():
                        if Account.objects.get(email=email_check.lower()).is_active == False:
                            data='not_active'
                        if Account.objects.get(email=email_check.lower()).is_blocked == True:
                            data='blocked'
                    elif Account.objects.filter(phone_number=email_check).exists():
                        if Account.objects.get(phone_number=email_check).is_active == False:
```

```
            data='not_active'
        if Account.objects.get(phone_number=email_check).is_blocked == True:
            data='blocked'
    elif Account.objects.filter(username=email_check.lower()).exists():
        if Account.objects.get(username=email_check.lower()).is_active == False:
            data='not_active'
        if Account.objects.get(username=email_check.lower()).is_blocked == True:
            data='blocked'
    else:
        data='not_exists'

  return JsonResponse(data, safe=False)

username = request.POST.get('username')
password = request.POST.get('password')

continue_execution = True
try:
  user = authenticate(username=Account.objects.get(username=username.lower()).email, password=password)
  continue_execution = False
except:
  pass
if continue_execution:
  try:
    # номер телефона
    user = authenticate(username=Account.objects.get(phone_number=username).email, password=password)
  except:
    # email
    user = authenticate(email=username.lower(), password=password)


if user is not None:
  if user.is_blocked:
    return redirect('account:logout')
  login(request, user)
  if 'next' in request.POST:
```

```
                return redirect(request.POST.get('next'))
            return redirect('board:index')
        else:
            return render(request, 'registration/login.html')
    else:
        return render(request, 'registration/login.html')
def user_logout(request):
    logout(request)
    return redirect('board:index')


from django.contrib.auth.views import PasswordResetView
from django.contrib.auth.forms import PasswordResetForm
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
class PasswordResetPSWRDView(PasswordResetView):
    form_class = PasswordResetForm
    success_url = reverse_lazy('account:password_reset_done')




@csrf_exempt
@login_required(login_url='/login/', redirect_field_name='/')
def favourite_add(request, pk):
    pk = int(pk)
    if request.user.is_authenticated:
        post = get_object_or_404(Board, id=pk)
        if post.favourites.filter(id=request.user.pk).exists():
            post.favourites.remove(request.user)
        else:
            post.favourites.add(request.user)

        return HttpResponseRedirect(request.META['HTTP_REFERER'])
    # else:
    #     return redirect('/login/')
```

```
@csrf_exempt
@login_required(login_url='/login/')
def favourite_list(request):
    new = Board.objects.filter(favourites=request.user)

    time_now1 = str(datetime.today().date().day).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)
    time_now2 = str(datetime.today().date().day-1).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' + str(datetime.today().date().year).zfill(2)

    context = {
        'new':new,
        'time_now1':time_now1,
        'time_now2':time_now2,
    }

    return render(request, 'board/favourites.html', context)




class RequestResetEmailView(View):
    def get(self, request):
        return render(request, 'registration/reset_email.html')

    def post(self, request):
        email = request.POST.get('email', None)

        if email is None:
            # messages.error(request, '')
            return render(request, 'registration/reset_email.html')

        user = Account.objects.filter(email=email)
        if user.exists():
            uidb64 =urlsafe_base64_encode(force_bytes(user[0].pk))
```

```
        domain = get_current_site(request).domain
        link = reverse('account:set_new_pswrd',
kwargs={'uidb64':uidb64,'token':token_generator.make_token(user[0])})


        activate_url='https://' + domain + link


        # email_context = {
        #     'user':user,
        #     'activate_url': activate_url,
        # }


        # email_subject = ''
        # # email_body = ''
        # email_body = render_to_string('registration/reset_password.html', email_context)
        # email_msg = EmailMessage(
        #     email_subject,
        #     email_body,
        #     settings.EMAIL_HOST_USER,
        #     [email],
        # )


        # # email_msg.send(fail_silently=False)




        # EmailThreading(email_msg).start()









        mail_title = "Оновлення пароля"
        variables = {
           'activate_url': activate_url,
        }
        html = get_template('registration/reset_password.html').render(variables)
        text = f'https://teenwork.com.ua/static/img/icons/teenwork.png - ( наше лого :) )\nЗабули пароль?\nЩоб
скинути пароль, натисніть кнопку нижче.\nСкинути\n{activate_url}\nЯкщо Ви не бажаєте змінювати свій пароль
або не запитували скидання паролю, Ви можете проігнорувати або видалити цей лист.\nTeenwork - платформа,
де підлітки і не тільки можуть знайти роботу, яка їм до вподоби.'
```

```
            msg = EmailMultiAlternatives(
                mail_title,
                text,
                settings.EMAIL_HOST_USER,
                [email])
            msg.attach_alternative(html, "text/html")
            # msg.send(fail_silently=False)
            EmailThreading(msg).start() # мы быстрее отправляем email


            return render(request, 'registration/reset_email_success.html')
        else:
            return redirect('/registration/')


from django.contrib.auth.tokens import PasswordResetTokenGenerator
from django.contrib import messages
from django.utils.html import strip_tags
from django.utils.translation import get_language
class SetNewPswrdView(View):
    def get(self, request, uidb64, token):
        context = {
            'uidb64':uidb64,
            'token':token,
        }

        try:
            user_id = force_text(urlsafe_base64_decode(uidb64))

            user = Account.objects.get(pk=user_id)

            if PasswordResetTokenGenerator().check_token(user, token):
                return render(request, 'registration/reset_email.html')
        except DjangoUnicodeDecodeError as identifier:
            return render(request, 'registration/reset_email.html')
```

```
        return render(request, 'registration/set_new_pswrd.html', context)

    def post(self, request, uidb64, token):
        context = {
            'uidb64':uidb64,
            'token':token,
        }

        password = request.POST.get('password', None)
        if password is None or len(password) < 4 or len(password) > 20:
            return render(request, 'registration/set_new_pswrd.html', context)

        try:
            user_id = force_text(urlsafe_base64_decode(uidb64))

            user = Account.objects.get(pk=user_id)
            user.set_password(password)
            user.is_active = True
            user.save()

            return redirect('/login/')
        except DjangoUnicodeDecodeError as identifier:
            return render(request, 'registration/set_new_pswrd.html', context)


class UnsubscribeView(View):
    def get(self,request, uidb64, token):
        try:
            uid = force_text(urlsafe_base64_decode(uidb64))
            user = Account.objects.get(pk=uid)
        except Exception as e:
            user = None

        if user is not None:
            if not token_generator.check_token(user,token):
                return render(request, 'others/user_unsubscribed.html')
```

```
            if user.email_subscription == False:
                return render(request, 'others/user_unsubscribed.html')
            user.email_subscription = False
            user.save()

            return render(request, 'others/user_unsubscribed.html')
        else:
            raise Exception
```

board\views.py

```
from django.shortcuts import redirect, render
from .models import Board, TeenworkBlog
from .forms import BoardForm, TeenworkBlogForm
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt
from .services import *
from django.templatetags.static import static
from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator
from django.contrib.admin.views import decorators
import functools
from googletrans import Translator


def staff_member_required(view_func):
    def _checklogin(request, *args, **kwargs):
        if request.user.is_active and request.user.is_staff:
            return view_func(request, *args, **kwargs)
        else:
            return redirect('board:index')
    return functools.wraps(view_func)(_checklogin)

decorators.staff_member_required = staff_member_required
```

```
def index(request):
    return show_index(request)
@csrf_exempt
def load_more(request):
    return load_more_index_adts(request)

def advertisement(request,slug):
    return show_advertisement(request,slug)

@csrf_exempt
def add_save(request):
    if request.method =='POST':
        form = BoardForm(data=request.POST, files=request.FILES)
        if form.is_valid():
            add_advert(request, form)
            if request.user.is_authenticated:
                return redirect('account:profile')
            return redirect('board:index')
        else:
            print(form.errors)
            if request.user.is_authenticated:
                context = {
                    'form':form,
                    'not_registered':False,
                }
            else:
                context = {
                    'form':form,
                    'not_registered':True,
                }
            return render(request, 'board/add.html', context)
    else:
        form = BoardForm()
```

```
        if request.user.is_authenticated:
            context={
                'form':form,
                'not_registered':False,
            }
        else:
            context={
                'form':form,
                'not_registered':True,
            }
        return render(request, 'board/add.html', context)

def add_get_user_data(request):
    return get_user_data(request)



@login_required(login_url='/login/')
def edit(request, pk):
    return edit_adt(request, pk)

@login_required(login_url='/login/')
def delete(request,pk):
    if request.user == Board.objects.get(pk=pk).author:
        Board.objects.get(pk=pk).delete()
        return redirect('/profile/')
    else:
        return redirect('/profile/')

@login_required(login_url='/login/')
def archive(request, pk):
    return archive_adt(request, pk)

@login_required(login_url='/login/')
def get_ntu_students(request):
    # also we can search students, who are registered in out service
    # data = Account.objects.filter(im_working_student=True, email__contains="@nmu.one")
```

```
data = [
    {
        'image': static('img/students/dmytro.jfif'),
        'name': 'Дмитро',
        'surname': 'Буслов',
        'patronymic': 'Юрійович',
        'speciality': '121-19-1',
        'faculty': 'ФІТ',
        'email': 'buslov.d.y@nmu.one',
        'city': 'Дніпро',
        'birthDate': '01.01.2001',
    },
    {
        'image': static('img/students/kristina.jfif'),
        'name': 'Кристина',
        'surname': 'Андрющенко',
        'patronymic': 'Сергіївна',
        'speciality': '121-19-1',
        'faculty': 'ФІТ',
        'email': 'andriushchenko.k.s@nmu.one',
        'city': 'Дніпро',
        'birthDate': '01.01.2001',
    },
    {
        'image': static('img/students/vladislav.jfif'),
        'name': 'Владислав',
        'surname': 'Федорець',
        'patronymic': 'Романович',
        'speciality': '121-19-1',
        'faculty': 'ФІТ',
        'email': 'fedorets.v.r@nmu.one',
        'city': 'Дніпро',
        'birthDate': '01.01.2001',
    },
    {
```

```
            'image': static('img/students/andriy.jfif'),
            'name': 'Андрій',
            'surname': 'Балян',
            'patronymic': 'Давідович',
            'speciality': '121-19-1',
            'faculty': 'ФІТ',
            'email': 'balian.a.d@nmu.one',
            'city': 'Дніпро',
            'birthDate': '01.01.2001',
        },
        {
            'image': static('img/students/karina.jfif'),
            'name': 'Карина',
            'surname': 'Жовнір',
            'patronymic': 'Олександрівна',
            'speciality': '121-19-1',
            'faculty': 'ФІТ',
            'email': 'zhovnir.k.o@nmu.one',
            'city': 'Дніпро',
            'birthDate': '01.01.2001',
        },
    ]
    university_info = {
        'image': 'https://lh3.googleusercontent.com/p/AF1QipM1veB6FAnZWcPP4jLBLpv1j1wkqJx4kiIpErtd=s1360-w1360-h1020',
        'title': 'Національний ТУ «Дніпровська політехніка»',
        'body': 'Національний технічний університет «Дніпровська політехніка» — провідний вищий навчальний заклад профілю України, заснований 16 червня 1899 р., розташований у місті Дніпро, Україна. Найстаріший вищий гірничий навчальний заклад в Україні, заснований як Катеринославський інститут.',
        'address': 'проспект Дмитра Яворницького, 19, Дніпро',
        'phone': '056 744 1411',
    }
    # return JsonResponse(data, status=200, safe=False)
    # return HttpResponse(data)
    return {'students': data, 'university': university_info}


@login_required(login_url='/login/')
def get_students(request, university):
```

```
    if university == 'ntu':
        data = get_ntu_students(request)
        return render(request, 'board/students.html', {'students':data['students'], 'university': data['university']})
    return render(request, 'board/students.html', {'students':[], 'university':[]})

def how_24h_works(request):
    return render(request, 'others/24_hour.html')
def about_cookies(request):
    return render(request, 'others/about_cookies.html')
def others_page(request):
    return render(request, 'board/adaptive_others_page.html')
def privacy_policy(request):
    return render(request, 'others/privacy_policy.html')
def website_rules(request):
    return render(request, 'others/website_rules.html')
def lawbook(request):
    return render(request, 'others/lawbook.html')
def about_us(request):
    return render(request, 'others/about_us.html')
def for_employers(request):
    return render(request, 'others/for_employers.html')

def error_to_admin(request):
    a=1/0
    return render(request, 'others/exception.html')

def handler400_error(request, *args, **argv):
    return render(request, "others/400.html")
def handler403_error(request, *args, **argv):
    return render(request, "others/403.html")
def handler404_error(request, *args, **argv):
    return render(request, "others/404.html")
def handler500_error(request, *args, **argv):
    return render(request, "others/500.html")

def tw_blog(request):
```

```
    blog_obj = TeenworkBlog.objects.filter(status = 'published')
    paginator = Paginator(blog_obj, 20)

    page = request.GET.get('page')
    try:
        response = paginator.page(page)
    except PageNotAnInteger:
        response = paginator.page(1)
    except EmptyPage:
        response = paginator.page(paginator.num_pages)

    time_now1 = str(datetime.today().date().day).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' +
str(datetime.today().date().year).zfill(2)
    time_now2 = str(datetime.today().date().day-1).zfill(2) + '.' + str(datetime.today().date().month).zfill(2) + '.' +
str(datetime.today().date().year).zfill(2)

    context={
        'blog_obj':response,
        'time_now1': time_now1,
        'time_now2': time_now2,
    }

    return render(request, 'others/blog.html', context)

from django.db.models import F
def tw_blog_post(request, slug):
    if TeenworkBlog.objects.filter(slug=slug).exists:
        blog_obj = TeenworkBlog.objects.get(slug=slug)
        TeenworkBlog.objects.filter(slug=slug).update(views=F('views')+1)
        return render(request, 'others/blog_post.html', {'blog_obj':blog_obj})
    else:
        return redirect('account:tw_blog')



'''
pip install googletrans==3.1.0a0
```

'''

```
def translate_str(string):
    string_trans = ''
    if get_language() == 'uk':
        try:
            # You can get about 1000 requests / hour without hitting the req/IP block limit. Also, individual requests are limited to less than 5000 characters per request
            translator = Translator()
            string_trans = translator.translate(string, dest='ru').text
        except Exception as e: print(e)

        return {'uk':string, 'ru':string_trans}
    elif get_language() == 'ru':
        try:
            # You can get about 1000 requests / hour without hitting the req/IP block limit. Also, individual requests are limited to less than 5000 characters per request
            translator = Translator()
            string_trans = translator.translate(string, dest='uk').text
        except Exception as e: print(e)

        return {'ru':string, 'uk':string_trans}

@staff_member_required
def add_blog_post(request):
    if request.method =='POST':
        form = TeenworkBlogForm(data=request.POST)
        if form.is_valid():
            post = form.save(commit=False)
            post.status = 'published'
            trans_title = translate_str(post.title)
            trans_content = translate_str(post.content)
            post.title_uk = trans_title['uk']
            post.title_ru = trans_title['ru']
            post.content_uk = trans_content['uk']
            post.content_ru = trans_content['ru']
```

```
            post.save()


            return redirect('board:tw_blog')
        else:
            return redirect('board:add_blog_post')
    else:
        form = TeenworkBlogForm()
        context={
            'form':form,
        }
        return render(request, 'others/blog_post_add.html', context)




@staff_member_required
def add_blog_post(request):
    if request.method =='POST':
        form = TeenworkBlogForm(data=request.POST)
        if form.is_valid():
            # ...
            return redirect('board:tw_blog')
        else:
            return redirect('board:add_blog_post')
    else:
        form = TeenworkBlogForm()
        # ...
        return render(request, 'others/blog_post_add.html', {})
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра на тему:
«Розробка інформаційної платформи для працевлаштування студентів на основі технологічного стеку Python/Django framework/TelegramBot Api/PostgreSQL DB»

студента групи 121-19-1 Буслова Дмитра Юрійовича

Керівник економічного розділу доц. Л.В. Касьяненко
каф. ПЕП та ПУ, к.е.н

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Буслов Д.Ю.диплом.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Буслов Д.Ю.диплом.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Буслов Д.Ю.диплом.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Буслов Д.Ю.диплом.ppt	Презентація кваліфікаційної роботи