

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ ВИКОНАВЦІВ
ДРІБНИХ РОБІТ

SOFTWARE FOR SEARCHING THE MINOR WORKS CONTRACTORS

Виконав: студент(ка) 4 курсу, групи КНТ-128сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Кайдаш А.В.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кайдаша Артемія Вадимовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення пошуку виконавців дрібних робіт
Software for Searching the Minor Works Contractors

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 14.05.2021

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “12” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Кайдаш А.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 83 с., 25 рис., 1 табл., 3 дод., 13 джерел.

ДРІБНІ РОБОТИ, ЗАМОВЛЕННЯ, ЗАВДАННЯ, ВЕБЗАСТОСУНОК, PHP, MODEL VIEW CONTROLLER.

Об'єкт роботи – процес виконання дрібних робіт.

Предмет роботи – програмне забезпечення пошуку виконавців дрібних робіт.

Мета роботи – розроблення програмного забезпечення пошуку виконавців дрібних робіт для забезпечення принципу делегування власних обов'язків.

Матеріали, методи та технічні засоби: мова програмування PHP, система керування базами даних MySQL.

Результати. Результати роботи полягають у виконаному аналізі проблеми, визначенні засобів розробки, вибору підходу до визначення архітектури системи, моделюванні функціональних вимог до системи, детальному проектуванні предметної області з побудовою структури даних, реалізації програми, описаних класів, що було реалізовано для вебзастосунку.

Висновки. Розроблено вебзастосунок, який призначений для забезпечення пошуку виконавців дрібних робіт замовниками, яким необхідно виконати певні тимчасові роботи, шляхом формування завдань або пошуку виконавців за їх вміннями. Кожен користувач системи може виступати і як замовник, і як виконавець, не потребуючи зміни облікового запису для зміни ролі в системі.

Галузь використання – створення платформи для взаємодії виконавців і замовників для виконання робіт на непостійній основі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 25 figures, 1 table, 3 appendixes, 13 sources.

MINOR WORKS, ORDERS, TASKS, WEB APPLICATION, PHP, MODEL VIEW CONTROLLER.

The object of work is the process of execution of micro works.

The subject of work is software for finding contractors of micro works.

The purpose of the work is to develop software for finding contractors of micro works to ensure the principle of delegation of responsibilities.

Materials, methods and technical means: PHP programming language, MySQL database management system.

Results. The results of the work are analysis of the problem, definition of development tools, choice of approach to determining the architecture of the system, modeling of functional requirements for the system, detailed design of the subject area with data structure construction, software implementation, described classes, which were implemented for web application.

Conclusions. A web application has been developed, which is designed to provide search for small contractors by customers who need to perform certain temporary works, by creating tasks or finding contractors according to their skills. Each user of the system can act as both a customer and an contractor, without having to change the account to change the role in the system.

Scope is in creating a platform for interaction between performers and customers to perform work on a non-permanent basis.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Проблема виконання дрібних робіт	9
1.2 Програмне забезпечення пошуку виконавців дрібних робіт.....	11
1.3 Огляд програмних аналогів.....	12
1.4 Висновки за розділом 1	17
2 Розробка архітектури програми.....	19
2.1 Вибір мови програмування	19
2.2 Функціональні вимоги до програмного забезпечення	20
2.3 Проектування структури даних	23
2.4 Висновки за розділом 2	33
3 Розробка програми	34
3.1 Порядок функціонування програми.....	34
3.2 Опис реалізації програми	35
3.3 Висновки за розділом 3	43
4 Експлуатація та тестування програми	44
4.1 Призначення програми	44
4.2 Умови виконання програми	44
4.3 Робота з програмою	44
Висновки	51
Перелік джерел посилання	52
Додаток А Технічне завдання	54
Додаток Б Текст програми	59
Додаток В Слайди презентації.....	77

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

HTML – HyperText Markup Language;

MVC – Model View Controller.

ВСТУП

Оголошення про виконання робіт знайомі кожному. Раніше вони зустрічалися більше в паперовому вигляді і наводнювали оточення людей всюди. З розвитком інформаційних технологій ефективність таких паперових оголошень втратилась, відповідно активного розвитку стали набувати вебсайти оголошень. Одним з таких напрямків є оголошення про пошук виконавців дрібних робіт. До дрібних робіт можна віднести і послуги няні, і послуги прибирання вдома, і послуги виконання невеликого ремонту, їх коло достатньо широке.

Якщо виконавець бажає виконати якусь роботу один раз і більше не буде робити цього, то для підтримки такої роботи достатньо простого неуніверсального ресурсу, але у випадку, якщо кількість пропозицій робіт виконавця збільшується, якщо він окрім власних пропозицій повинен ще шукати замовлення, якщо кількість запитів на його роботи збільшується, а до того ж не всі він підтримує, то для упорядкування таких даних необхідне спеціалізоване програмне забезпечення.

Важливим практичним завданням є створення програмного забезпечення, яке повинно дозволити людям (замовникам) визначати їх потреби в роботах, а іншим людям (виконавцям) – визначати їх можливості щодо здійснення робіт. До того ж останні повинні мати можливість представити свої вміння незалежно від замовлень. За рахунок виділення сфери саме дрібних робіт має бути досягнуто привертання уваги до програми за рахунок специфіки сфери використання на відміну від вебсайтів пошуку роботи, що в результаті може створити спільноту для взаємодії і додатково сприяти популяризації програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема виконання дрібних робіт

Дрібні роботи – це вид тимчасової роботи, що передбачає виконання завдань за окрему плату, такі роботи можуть включати роботу, яка виконується в Інтернеті або особисто, і можуть охоплювати такі завдання, як написання щоденників, виконання різноробочих робіт, роботи няні, роботи з дизайну вебсайтів, введення даних та виконання інших адміністративних завдань, виконання доручень, такі роботи мають багато переваг, оскільки дозволяють працювати за гнучким графіком, вибирати роботу, яку працівник хоче прийняти, та формувати навички та досвід для свого резюме [1].

Як і для будь-якого типу кар'єрних можливостей, важливо шукати та знаходити дрібні роботи, які будуть добре відповідати особливості виконавця та його інтересам: кількість часу, яку вимагає такий тип роботи, може бути невеликою, але зрештою виконавець буде набагато щасливішими, якщо знайде варіант, який йому справді сподобається [2].

Існує декілька різних типів мікроробіт:

- робота няні: можна запропонувати послуги няні батькам, яким на короткий час потрібен хтось, кому можна довірити спостереження за своїми дітьми, багато людей знаходять роботу няні, спілкуючись із знайомими людьми, у яких є діти, або розміщуючи свої послуги в місцевих комісіях із догляду за дітьми;

- вигул собак: якщо працівник має можливість ужитися з декількома різними породами собак і він доступний вранці та ввечері, то можна погодитися вигулювати собак для людей у своєму районі в обмін на невелику суму грошей кожного разу, коли сам виконавець вигулює свою собаку;

- різноробоча робота: якщо ви добре справляєтеся та ремонтуєте речі вдома, прибираєте чи виконуєте легкі домашні справи, такі як вивішування картин чи малювання, ви можете запропонувати свої послуги людям у вашому районі, які потребують допомоги у виконанні цих завдань;

– косіння газону: якщо виконавець не проти працювати на відкритому повітрі, то можна виконати стрижку газону та базовий ландшафтний дизайн для людей у своєму районі, це популярна робота для підлітків, які мають обмежений досвід роботи, при цьому деякі люди очікуватимуть від виконавця власної газонокосарки та інструментів для озеленення;

– виконання доручень: багатьом людям потрібен хтось, хто може допомогти їм виконати такі справи, як покупка продуктів, транспортування одягу в хімчистку, відвідування банку та декілька інших звичайних завдань, вони можуть запропонувати послуги зайнятим мамам, людям з обмеженими можливостями, людям похилого віку тощо;

– введення даних: створення простих завдань із введення даних для віддалених працівників в Інтернеті стало популярним серед багатьох компаній, для виконання цієї роботи знадобиться доступ до комп'ютера, Інтернет-сервісу;

– написання контенту: якщо виконавець здатний письменник, можна пропонувати свої послуги приватному бізнесу та маркетинговим агенціям, які потребують контенту, написаного для своїх вебсайтів, соціальних медіа та друкованих рекламних матеріалів;

– адміністративні завдання: багато підприємств також наймають віддалених працівників для виконання основних адміністративних завдань, таких як нарахування заробітної плати, електронна реєстрація та реагування на листування клієнтів;

– графічний дизайн: люди, які мають талант створювати унікальну графіку за допомогою комп'ютерного програмного забезпечення, можуть пропонувати свої послуги як графічні дизайнери-фрілансери для підприємств та організацій, яким потрібна допомога щодо їх вебсайту, соціальних медіа чи друкованих рекламних матеріалів;

– опитування: брати участь в опитуваннях – це мікроробота для людей, які хочуть займатися роботою у вільний час, існує декілька компаній,

що проводять онлайн опитування, які платять людям за чесні відгуки, це допомагає компаніям збирати інформацію від цільової аудиторії та ринку [1].

1.2 Програмне забезпечення пошуку виконавців дрібних робіт

Програмне забезпечення пошуку виконавців дрібних робіт повинно ґрунтуватися на наступних принципах:

- управління послугами: звичайно, що виконавці реєструються на сайті, щоб рекламувати свої послуги і отримувати замовлення, тому продавці повинні мати можливість розміщувати окремі оголошення про виконання різних робіт на основі своїх вмінь;
- управління замовленнями – це функція, яка допомагає виконавцям контролювати всі замовлення за статусом;
- ціноутворення на послуги: виходячи зі стратегії, можна встановити фіксовану ціну для всі дрібні роботи або дозволити гнучкий варіант ціноутворення, і дозволити виконавцям самостійно визначати ціну;
- система може мати різні рівні користувачів: ця система є мотивацією для виконавців добре працювати в представленні на сайті, відповідно чим вищі рівні вони досягають, тим більше ймовірність отримати нові замовлення;
- сортування: функція сортування дозволяє виконавцям і замовникам переглядати дрібні роботи у бажаному порядку, наприклад, вони можуть сортувати завдання або вміння за високими рейтингами або за часом появи таких оголошень;
- фільтрування: іншим ефективним критерієм пошуку є фільтрування, дрібні задачі можна відфільтрувати за категоріями, підкатегоріями або за певними тегами;
- створюючи сторінку з деталями дрібних робіт, потрібно, щоб кнопка замовлення була видимою та видною з інших елементів;

– завжди важко зрозуміти, якими є люди в Інтернеті, тому огляд профілів виконавців може бути розроблений як рішення для створення довіри між виконавцями та замовниками, ця функція вигідна як для обох сторін користувачів, так і для самого вебсайту, виконавці можуть отримати більше замовлень, якщо мають багато позитивних відгуків, замовникам є де висловити свої коментарі щодо послуг, які вони отримали, а активна система огляду – це перевага для привертання уваги серед користувачів, особливо нових [3]-[4].

1.3 Огляд програмних аналогів

Програмні аналоги у світі та Україні включають рішення різного рівня: частина з них є універсальними і дозволяють виконувати дрібні роботи будь-якого типу, інша частина є вузькоспеціалізованими і дозволяють, наприклад, виконувати сантехнічні роботи, роботи з прибирання тощо. Деякі вебсервіси направлені на виконання віддаленої роботи, тобто вони не містять послуг з прибирання або подібних, а містять завдання різного характеру (в цьому сенсі є універсальними засобами), але всі вони мають виконуватися віддалено (наприклад, класифікація зображень, оброблення відео, перевірка даних).

Деякі з тих програмних засобів, які було проаналізовано і які мають певну специфіку відносно інших, виділено у даному підрозділі окремо і описано нижче.

TaskRabbit [5] (рис. 1.1) – це вебсайт і застосунок (iOS та Android), що дозволяє користувачам досліджувати, наймати, оцінювати та переглядати місцевих постачальників послуг, при цьому ці послуги виражаються у вигляді завдань універсального типу, тобто існує цілий ряд категорій завдань з різних галузей [6].

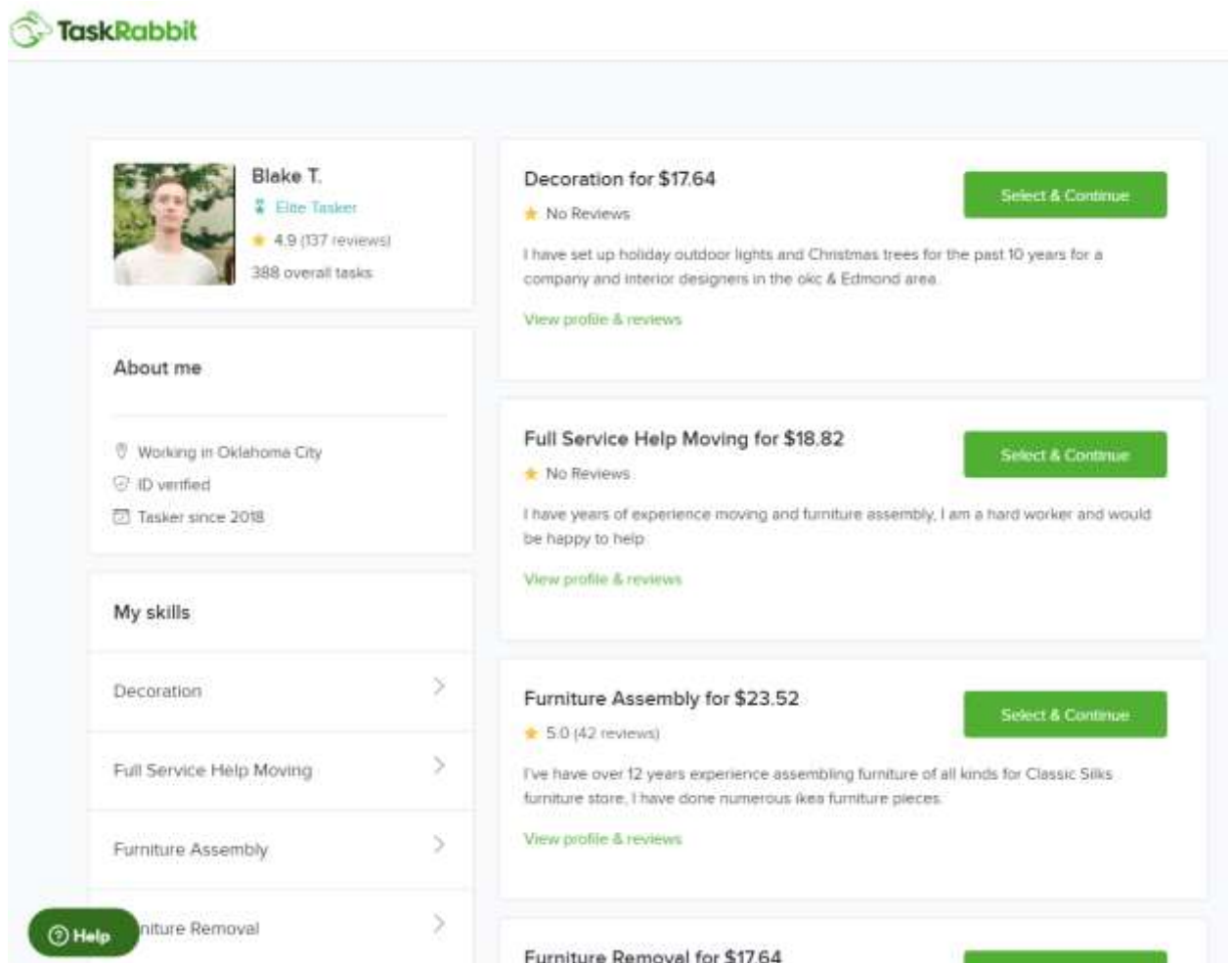


Рисунок 1.1 – Перегляд профілю виконавця на вебсайті TaskRabbit

Щодо TaskRabbit можна відзначити наступне:

- погодинна ставка включає 15 % плати за обслуговування TaskRabbit: якщо потрібні будь-які матеріали, то доведеться або придбати їх самостійно, або обговорити додавання компенсації до рахунку-фактури, на додаток до плати за послугу, можна очікувати, що до рахунку-фактури після завершення послуги буде додано 15 % комісії за довіру та підтримку від загальної суми рахунку-фактури (без урахування витрат та відшкодування);
- TaskRabbit працює шляхом з'єднання користувачів з місцевими виконавцями, сервіс при цьому надає доступ до перевірених рейтингів та відгуків;

- можна використовувати TaskRabbit, щоб знаходити завдання поруч, читати про досвід попередніх клієнтів, порівнювати погодинні тарифи та ділитися своїми відгуками;

- усі завдання поділені за категоріями, за категоріями можна переглянути перелік робіт і виконати пошук, вказавши для потрібної роботи адресу її виконання, особливості виконання;

- за виконавцем відображається набір завдань, які він здатен виконувати, при цьому для кожного завдання вказується його опис, що включає досвід виконання завдання виконавцем, набір інструментів та автомобілів, який є у розпорядженні виконавця;

- переглядати перелік робіт за містом, обраним з переліку доступних, або за введеним поштовим індексом;

- основна процедура взаємодії включає процедуру, що охоплює вибір завдання, пошук виконавця з зазначенням вартості, вибір дати і часу виконання та безпосередньо оформлення замовлення [6].

Недоліком даного застосунку є те, що за послуги стягується плата, при чому достатньо велика, що дещо збільшує поріг входження в область.

Окрім інструментів, які є універсальними засобами пошуку робіт, тобто дозволяють виконувати пошук виконавців для різних областей, існують інструменти, направлені на конкретні галузі.

Housekeeper [7] (рис. 1.2) – це вебсайт для пошуку робіт, які можуть виконуватися по дому.

Housekeeper відзначається наступним:

- вебсайт дозволяє виконувати пошук як виконавців робіт, так і самих робіт, тобто користувач може виступати як замовником, так і виконавцем, підтримуються обидві сторони роботи;

- вебсайт дозволяє обирати місто з переліку найбільш популярних, виконувати пошук робіт, використовуючи фільтри, що включають ключові слова, місто розташування (конкретне місто, поштовий код, вказуючи радіус дії), тип виконавця (індивідуальний, агентство), категорії робіт з прибирання,

визначити необхідність або відсутність необхідності додаткового обладнання;

– за кожним виконавцем відображається їх місто проживання, вартість години роботи, опис, перелік конкретних робіт з прибирання, які надаються виконавцем, аналогічним чином відображається інформація про завдання, по суті анкети завдання та виконавця виглядають однаково;

– за виконавцями можна переглянути момент їх останньої авторизації, що дозволяє ідентифікувати таким чином неактивні оголошення і не витрачати час на застарілі оголошення.

The screenshot displays the Housekeeper.com website interface. On the left, there is a sidebar with search filters. The main content area on the right is titled "More than 10,000 Housekeepers Found" and lists three featured profiles.

Search Filters (Left Sidebar):

- Search for:**
 - ☒ Housekeepers
 - ☐ Housekeeping Jobs
- Search by Keyword:** [Search bar]
- Location:**
 - Select Country:** [Dropdown menu]
- Type of Provider:**
 - ☐ Agencies/Companies (18k)
 - ☐ Individuals (93k)
- Pay:**
 - ☐ Negotiable (61k)
- Services:**
 - ☐ Bathroom Cleaning (865k)
 - ☐ Kitchen Cleaning (874k)
 - ☐ General Room Cleaning (894k)
 - ☐ Change Bed Linens (185k)
 - ☐ Oven Cleaning (191k)
 - ☐ Refrigerator Cleaning (781k)
 - ☐ Cabinet Cleaning (176k)
 - ☐ Dishes (832k)
 - ☐ Window Washing (165k)
 - ☐ Surface Polishing (158k)
 - ☐ Carpet Shampooing (94k)
 - ☐ Wall Washing (131k)
 - ☐ Pet Clean-Up (109k)
 - ☐ Packing (127k)
 - ☐ Attic Clean-Up (95k)
 - ☐ Basement Clean-Up (110k)
 - ☐ House Sitting (146k)
 - ☐ Take Out Trash/Recycl... (184k)

Featured Housekeeper Profiles (Right):

- Rebecca K:** House keeper interested in Work. Prairie Hill, Texas | Last seen over a week ago. Resume Summary: I am a dirt detector, if your home appears to be clean, I will find the dirt. I pay attention to all areas that make for a fresh home. I can surprise you with better rates with monthly agr... \$ 20/hr.
- Christine W:** Professional housekeeper looking to fill week days and Sunday's during July and August. Haliburton, Ontario | Last seen 1 hour ago. Summary: Hello my name is Christine. I am searching for home and cottage cleaning employment opportunity in Haliburton, Ontario. I plan to utilize my experience as a great housekeeper and detail oriented wo... 30 yrs experience \$ 40/hr.
- Silvana C:** Housekeeping, organizing, detailed work, Birmingham, Bloomfield hills, Novi ,Farmington Michigan. Farmington Hills, Michigan | Last seen 1 hour ago. Hey my name is Silvana. I am professional freelance home helper in Farmington hills, Bloomfield hills Birmingham and I specialize in all organizing food prepare and cleaning. Please let me know if you are... 20 yrs experience \$ Pay negotiable.

Рисунок 1.2 – Пошук виконавців для домашніх робіт на вебсайті Housekeeper

Особливістю є те, що за виконавцями відображаються подібні на нього виконавці, що дозволяє зручним чином переглядати альтернативні варіанти знайденому.

Недоліком вебсайту є неможливість гнучкого переходу між оголошеннями, наприклад, не можна за знайденим параметром знайти відповідні оголошення, окрім того цей сервіс є вузькоспеціалізованим.

Українською альтернативою даним сервісам є сервіс Ooch [8] (рис. 1.3), який надає можливість шукати виконавців та замовників робіт на виконання завдань універсального характеру.

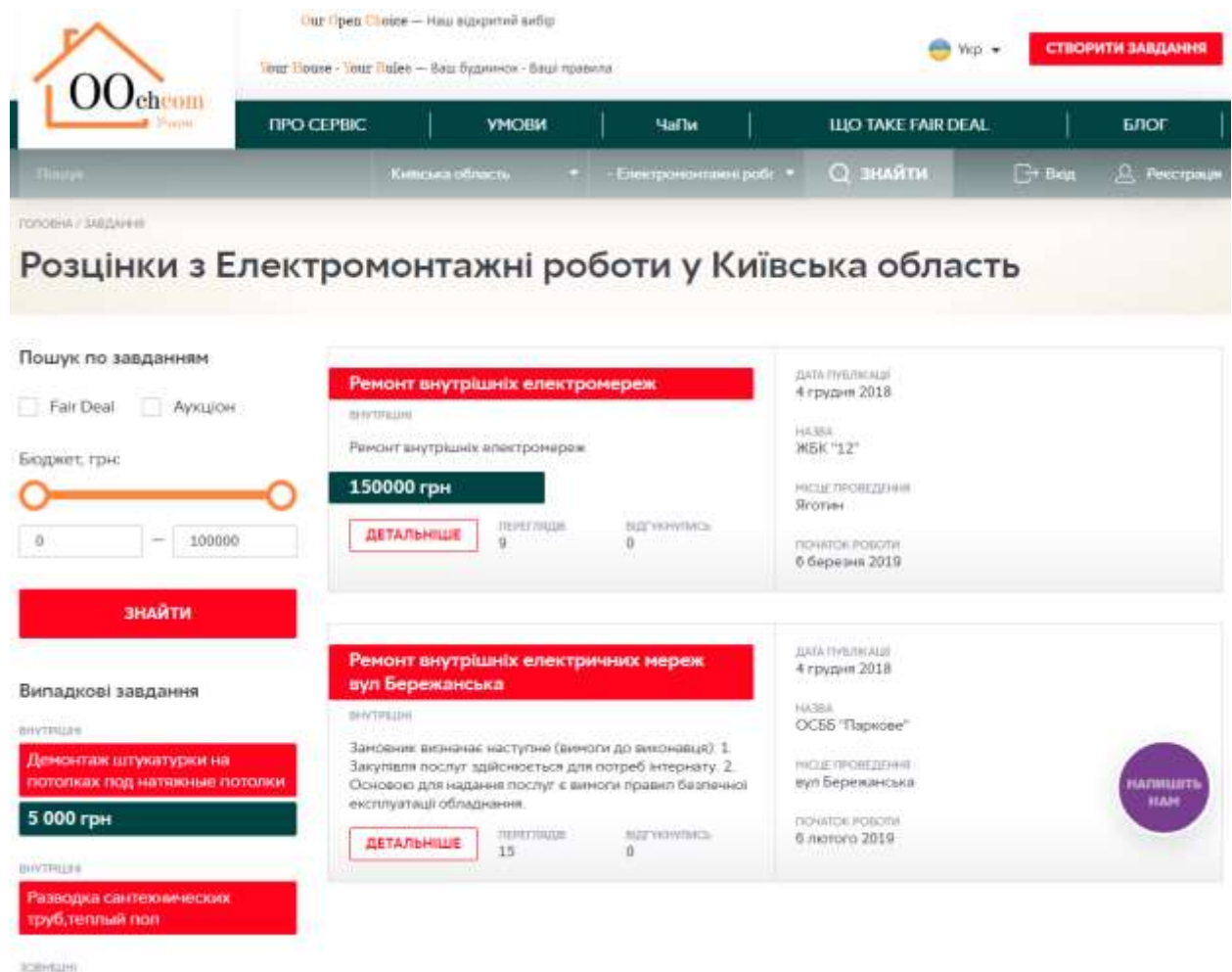


Рисунок 1.3 – Пошук виконавців робіт на вебсайті Ooch

За кожним виконавцем відображаються його послуги, інформація про нього, а також зворотна інформація у вигляді кількості відгуків, які

класифіковані на позитивні та негативні. За замовленням відображається опис, дата публікації, час виконання, кількість переглядів, звернень, запропонована вартість роботи. Окрім того надається можливість пошуку робіт за назвою, регіоном, категорією.

Особливістю застосунку є те, що вартість завдання може встановлюватися на аукціоні, а може відповідати заданій замовником. Існують і такі, і такі завдання, за необхідності можна виконати пошук потрібного.

Підводячи підсумок проведеного огляду програмних аналогів, необхідно вказати на те, що кожен такий сервіс об'єднує користувачів, які потребують виконання певних короткострокових робіт, і тих, які вміють виконувати такі роботи. При цьому існуючі сервіси за це потребують сплати деякого відсотку.

У програмному забезпеченні, що розробляється, планується привернути увагу за рахунок відсутності такої сплати, що дозволить зменшити поріг входження в дану галузь, відповідно коло потенційних виконавців та замовників за рахунок зменшення вартості послуг має розширитися.

1.4 Висновки за розділом 1

У першому розділі виконано огляд проблеми виконання дрібних робіт: розглянуто особливості, категорії і приклади дрібних робіт. Виконано огляд програмних аналогів, з виділенням їх функціональних можливостей, особливостей та недоліків. Програмні аналоги включають сервіси TaskRabbit, Housekeeper, Ooch. Дані сервіси є безкоштовними для реєстрації в них, але за оформлення замовлення потребують певної сплати, вираженої у відсотках від сплаченої суми.

Завданнями дипломної кваліфікаційної роботи, що мають бути розв'язані в наступних розділах, є:

- розробка технічного завдання і визначення вимог до програми;
- розробка архітектури програми;
- розробка програми;
- тестування програми.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір мови програмування

Аналогічно розглянутим програмним аналогам програмна реалізація буде виконуватися як веброзробка.

Першим етапом є вибір мови програмування для створення програмного забезпечення. Для цього було порівняно мови PHP [9]-[10] і Java. Результати порівняння зведено в табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування

Параметр	Java	PHP
Виконання	Компілятор	Інтерпретатор
Парадигми програмування	Об'єктно-орієнтована	Об'єктно-орієнтована, функціональна
Призначення	Загальне	Загальне, початково – веброзробка
Веброзробка, серверна частина	+	+
Менші витрати на розробку і підтримку	+—	++

За результатами порівняння було обрано мову програмування PHP для реалізації, оскільки вона активно використовується саме в галузі веброзробки, є широковідомою в цьому сенсі, має багато документації, до того ж характеризується меншими витратами на розробку і підтримку, що пов'язано з тим, що час, витрачений на розробку і подальші зміни, зазвичай менше для мови PHP ніж для Java.

2.2 Функціональні вимоги до програмного забезпечення

Функціонування програми має бути побудовано на взаємодії замовників з виконавцями. Щодо реалізації дрібних робіт, які потрібні замовникам, а виконуються виконавцями, було внесено два поняття: поняття завдання і поняття вміння. На їхній основі побудовано функціональні вимоги до програми.

Програма має забезпечувати виконання наступних функціональних вимог для підтримки взаємодії замовників з виконавцями дрібних робіт:

- створення завдання на виконання роботи;
- перегляд переліку власних завдань;
- визначення оголошення завдання;
- визначення власних вмінь;
- перегляд переліку власних вмінь;
- редагування вмінь;
- пошук завдань з можливістю сортування результатів;
- пошук вмінь з можливістю сортування;
- перегляд завдань за містом виконання;
- перегляд вмінь за містом виконання;
- перегляд переліку замовлень робіт;
- замовлення робіт на основі відповідного вміння виконавця;
- підтвердження або скасування замовлення на виконання робіт виконавцем;
- визначення вмінь виконавця, яких потребує виконання завдання;
- створення відгуку про вміння;
- перегляд профілю виконавця;
- перегляд відгуків на вміння виконавця;
- створення запиту на виконання завдання;
- перегляд переліку запитів на виконання завдання;
- підтвердження або скасування запиту на виконання завдання;

Будь-який користувач системи під час роботи з нею може виконувати одну з двох ролей: замовника (рис. 2.1) або виконавця.

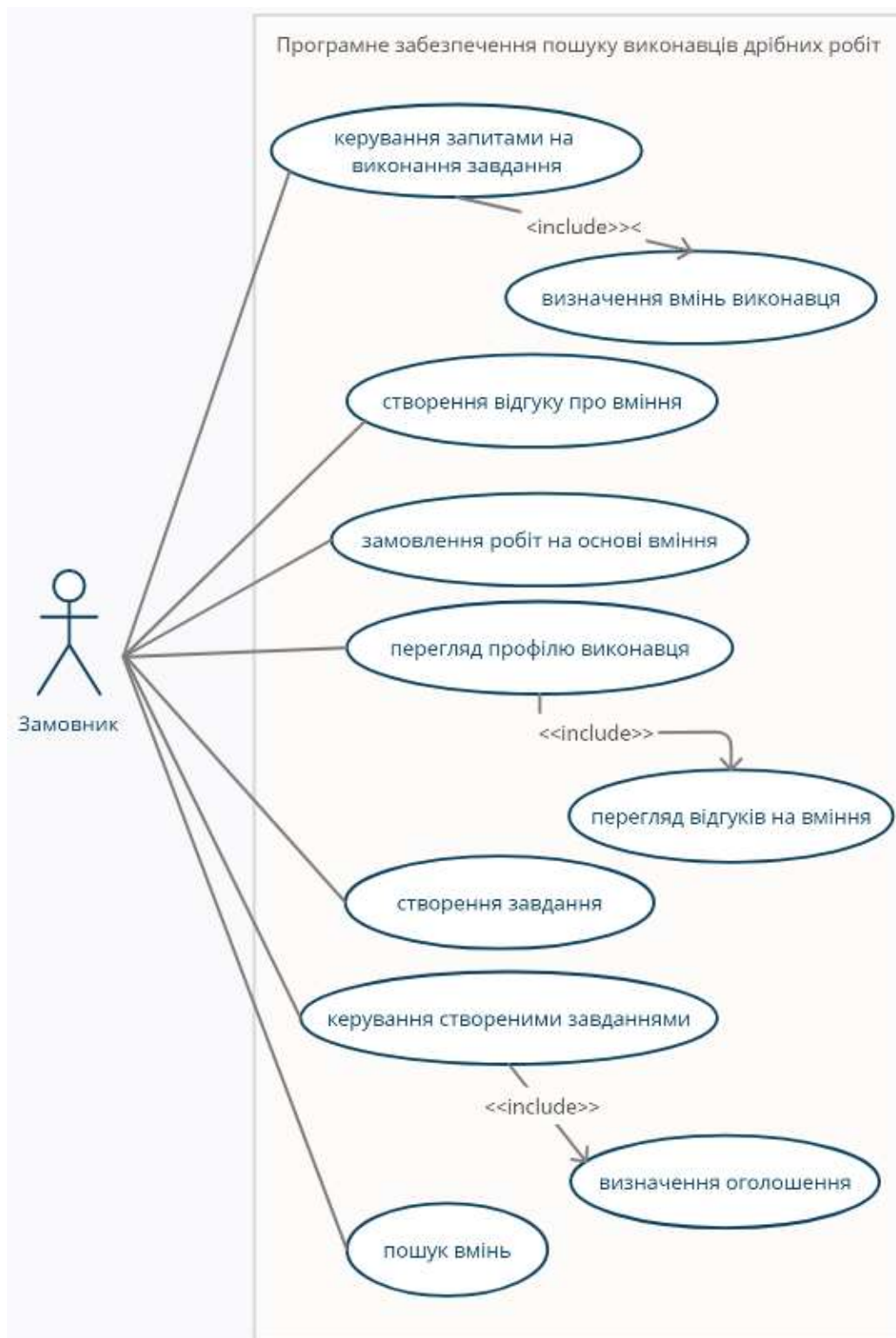


Рисунок 2.1 – Діаграма прецедентів замовника

Окрім того функціональними вимогами до програмного забезпечення є також:

- автоматичне скасування непідтверджених запитів на виконання завдання після підтвердження одного з запитів на оголошення;
- авторизація облікового запису;
- реєстрація облікового запису, яка є одночасно реєстрацією облікового запису виконавця і реєстрацією облікового запису замовника.

Під завданням розуміється опис роботи, виконання якої потребує замовник. Тобто першоджерелом взаємодії щодо виконання робіт є саме замовник.

Під вмінням розуміються здатності виконавця виконувати певні види робіт. Тобто першоджерелом взаємодії щодо виконання робіт є саме виконавець. Замовник має знайти потрібні йому вміння і продовжити взаємодію.

Замовник може скористатись пасивними сценаріями роботи з системою, коли він визначає власні завдання і очікує запитів на їх виконання, а потім керує ними, а може виконувати активну роль, для чого передбачено сценарії пошуку вмінь, перегляду профілю виконавця, замовлення робіт.

Виконавець (рис. 2.2) може керувати замовленнями на виконання робіт, отриманими від замовника, створювати вміння і керувати власними вміннями, переглядати і виконувати пошук завдань, а також створювати запити на виконання завдання.

Пошук від перегляду відрізняється тим, що пошук відбувається за заданими параметрами виконавцем, а перегляд реалізується за чітко заданими категоріями.

Програмне забезпечення виконує роль платформи, на якій мають зустрічатися замовники і виконавців дрібних робіт, пропонуючи свої варіанти такої взаємодії і за потреби шукаючи варіанти для влаштування своєї діяльності.

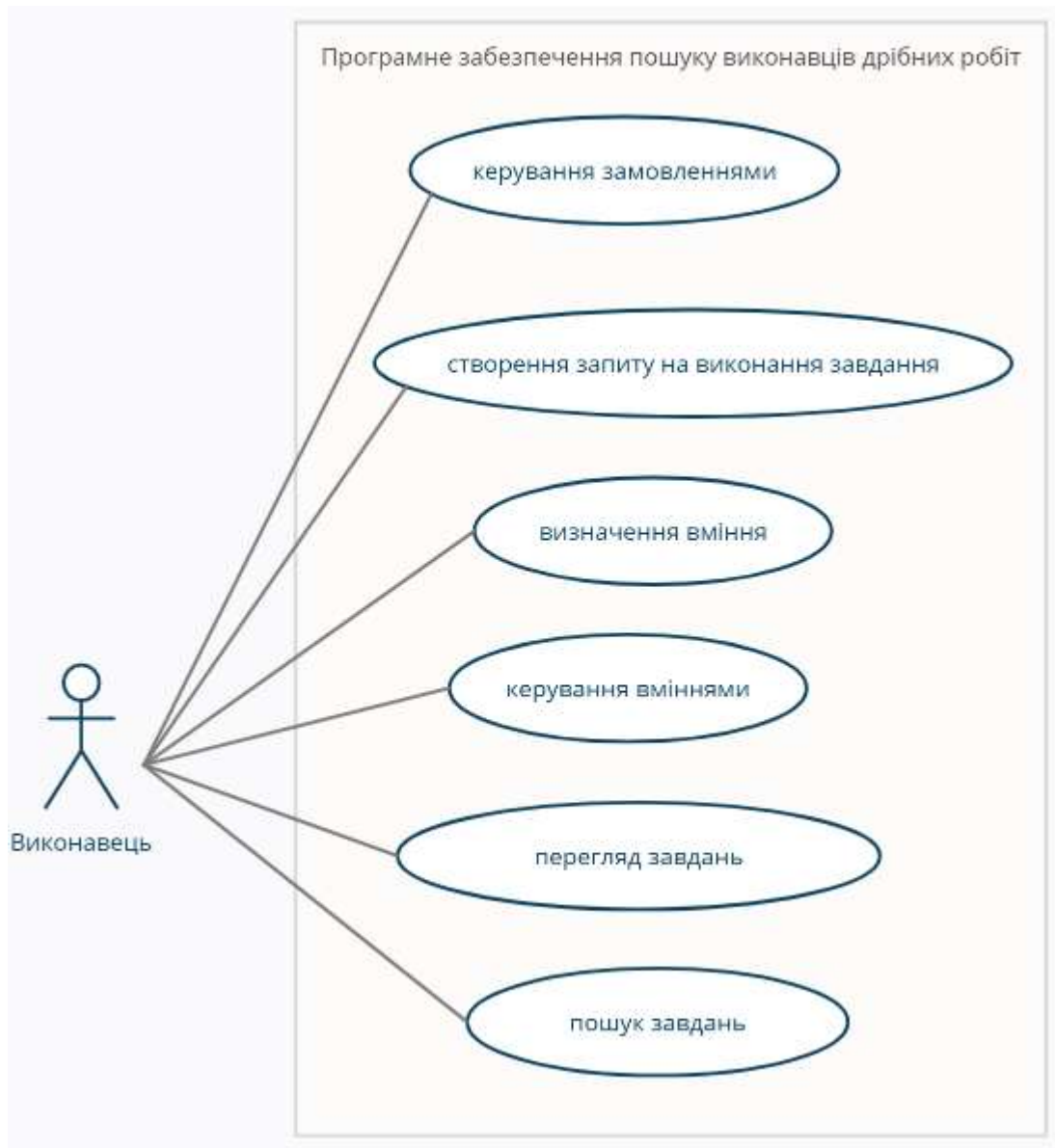


Рисунок 2.2 – Діаграма прецедентів виконавця

2.3 Проєктування структури даних

Таблиця Account визначає користувача програми, який може виступати як виконавцем, так і замовником, тобто користувач несе в собі подвійну роль. Роль визначається конкретними функціями програми, якими він користується. В одну мить користувач може бути замовником, а в іншу – виконавцем.

Таблиця характеризується полями:

– id – номер користувача;

- login – логін користувача;
- password – пароль користувача;
- surname – прізвище користувача;
- name – ім'я користувача;
- information – представлення себе, що буде використовуватися у профілі, який будуть переглядати інші користувачі;
- city_id – місто проживання;
- phone – телефон користувача;
- firstvisit – час реєстрації користувача;
- lastvisit – час останньої авторизації користувача;
- score – сумарна кількість балів, виставлених усім роботам користувача;
- dec – порядок оцінки (для роботи з великими числами зберігається ступінь десяти);
- reviews – кількість оцінок, виставлених за роботу користувача;
- photo – фотографія користувача.

Оцінювання відбувається тільки відносно виконаних робіт, тобто оцінювання авторів завдань не здійснюється. Це виходить зі специфіки проблеми, що розглядається. Оскільки це проблема пошуку виконавців робіт, то саме виконавцям робіт ставляться оцінки, щоб користувачі, які ставлять завдання або виконують пошук виконавців, могли з оцінки отримати перше враження про користувача, який претендує стати виконавцем робіт.

Оцінка не зберігається безпосередньо в базі даних значенням від 0 до 5, оцінка зберігається як сумарна кількість балів та кількість виставлених оцінок, відповідно для виведення оцінки користувача потрібно поділити сумарну кількість балів на кількість оцінок.

Оцінка, сформована для користувача, як і кількість оцінок є результатом агрегації оцінок, виставлених вмінням користувача. Тобто оцінюється не сам користувач, а його вміння замовником після виконання

поставленого ним завдання. А оцінка користувача отримується шляхом знаходження суми оцінок усіх вмінь працівника.

Поле `city_id` є зовнішнім ключем таблиці `City`, таким чином визначаючи, де проживає користувач. Передбачається, що це основа для визначення місцезнаходження виконання завдань, відповідно і пошуку виконавців і послуг. Але фактично користувач може проживати в одному місті, а замовляти реалізацію завдань в іншому, наприклад, через те, що виконує замовлення в поїзді або має будинок в іншому місті. Тому місто визначається не тільки для користувача, але і для завдання. Для вміння, тобто тих робіт, що може виконати користувач, місто не визначається, оскільки їх має виконувати користувач (виконавець) в місті проживання.

Зв'язок між таблицями `City` та `Account` один до багатьох, оскільки в одному місті може проживати велика кількість користувачів, при цьому кожен користувач проживає в одному й тому самому місті.

Таблиця `View` визначає перегляди вмінь виконавців. Перегляди фіксуються саме для вмінь. Для завдань перегляди не фіксуються. Перегляд кожного вміння має бути унікальним, тобто належати різним користувачам.

Таблиця характеризується полями:

- `id` – номер перегляду;
- `account_id` – користувач, який переглядав вміння працівника;
- `skill_id` – номер вміння, перегляд якого виконано;
- `created` – дата і час перегляду.

Поле `account_id` визначає зовнішній ключ таблиці `Account` і встановлює відношення між таблицями `Account` і `View` один до багатьох, оскільки кожне вміння може переглядатися багатьма користувачами, при цьому один користувач може переглядати певне вміння тільки один раз (максимально) або нуль разів.

Поле `skill_id` визначає зовнішній ключ таблиці `Skill` і встановлює відношення між таблицями `Skill` і `View` один до багатьох, оскільки кожне вміння може мати багато переглядів, при цьому перегляд певного вміння

певним користувачем може виконуватися тільки один раз (максимально) або нуль разів.

Фактично в системі фіксуються унікальні перегляди, тобто якщо один і той самий користувач буде заново переглядати одне й те саме вміння, то від цього кількість переглядів не збільшиться.

Таблиця Skill визначає вміння виконавця і характеризується полями:

- id – номер вміння працівника;
- jobcategory_id – категорія робіт, до якої відноситься дане вміння;
- title – назва вміння;
- price – вартість виконання даної роботи за годину;
- holidayprice – вартість виконання даної роботи за годину вихідного дня;
- experience – досвід щодо даного вміння у працівника;
- account_id – номер користувача, який має дане вміння;
- created – дата і час, коли було внесено інформацію про дане вміння;
- score – сумарна оцінка даного вміння працівника;
- dec – порядок оцінки (для роботи з великими числами зберігається ступінь десяти);
- reviews – кількість оцінок, виставлених даному вмінню користувача;
- city_id – місто, в якому може бути реалізовано це вміння.

Skill передбачає саме вміння конкретного працівника, тобто узагальненням вміння є тільки категорія роботи Jobcategory, а вміння представляють реалізації цих категорій у конкретних працівників.

Поле jobcategory_id визначає зовнішній ключ таблиці Jobcategory і встановлює відношення між таблицями Skill і Jobcategory один до багатьох, оскільки кожне вміння може належать тільки одній категорії, при цьому кожна категорія може мати велику кількість конкретних вмінь.

Jobcategory характеризує і завдання, і вміння, тобто згруповані завдання і вміння мають бути однаковою чиною, це єдина система

класифікації, яка є основою визначення потрібних працівників під наявні завдання.

Поле `account_id` визначає зовнішній ключ таблиці `Account` і встановлює відношення між таблицями `Account` і `Skill` один до багатьох, оскільки користувач може мати багато вмінь, при цьому конкретне вміння відноситься тільки до одного користувача, адже є реалізацією вміння саме конкретного користувача.

Поле `city_id` є зовнішнім ключем таблиці `City` і встановлює відношення між таблицями `City` і `Skill` один до багатьох, оскільки у кожному місті може реалізовуватися багато вмінь різних виконавців, при цьому конкретне вміння реалізовується тільки в одному місті.

Таблиця `SkillRequest` визначає запит замовника на послуги виконавця, які визначаються його вмінням. Запити на вміння використовуються тільки для інформування виконавця про наявність інтересу зі сторони замовника.

Таблиця характеризується полями:

- `id` – номер запиту на вміння;
- `skill_id` – номер вміння працівника;
- `account_id` – номер замовника;
- `duration` – тривалість годин роботи;
- `information` – інформація запиту, якщо її бажає надати замовник;
- `period` – дата і час початку робіт;
- `mode` – стан запиту.

Поле `account_id` визначає зовнішній ключ таблиці `Account` і встановлює відношення між таблицями `Account` і `SkillRequest` один до багатьох, оскільки користувач може мати багато запитів, при цьому конкретний запит відповідає тільки одному користувачу, тобто не може бути у одного запиту декілька авторів.

Поле `skill_id` визначає зовнішній ключ таблиці `Skill` і встановлює відношення між таблицями `Skill` і `SkillRequest` один до багатьох, оскільки кожне вміння може мати багато запитів, при цьому кожен запит виконується

тільки на одне вміння. Для оформлення запиту на декілька вмінь потрібно сформулювати окремі запити.

Таблиця Task визначає завдання, яке ставить замовник перед виконавцями і для якого шукає виконавця. Таблиця характеризується полями:

- id – номер завдання;
- jobcategory_id – категорія робіт, якій належить завдання;
- title – назва завдання;
- requirements – вимоги, які необхідно задовольнити для виконання завдання;
- information – опис завдання;
- account_id – номер користувача, який визначив завдання;
- created – дата і час створення завдання;
- active – активне або неактивне завдання;
- mode – режим роботи з завданням (завдання на один раз, повторюване завдання);
- costs – максимальна сума, яку може витратити замовник на виконання завдання.

Поле account_id визначає зовнішній ключ таблиці Account і встановлює відношення між таблицями Account і Task один до багатьох, оскільки один користувач може мати багато завдань в системі, при цьому одне завдання може належати тільки одному користувачу.

Поле jobcategory_id визначає зовнішній ключ таблиці Jobcategory і встановлює відношення між таблицями Task і Jobcategory один до багатьох, оскільки кожне завдання повинно належати конкретній категорії робіт, при цьому кожна категорія може містити багато завдань.

Таблиця TaskPeriod необхідна для визначення конкретної активності завдання, тобто фактично оголошення про виконання завдання, побудованого на основі завдання, описаного в таблиці Task. Дана таблиця головним чином потрібна для організації завдань, які можуть повторюватися.

У такому випадку потрібно для кожного варіанту активності завдання зберігати період цієї активності. Оскільки кожне завдання може характеризуватися різними періодами, то зберегти все це в таблиці Task неможливо, тому введено таблицю TaskPeriod.

Таблиця характеризується полями:

- id – номер періоду завдання;
- task_id – номер завдання;
- date1 – дата і час початку періоду виконання завдання;
- date2 – дата і час завершення періоду виконання завдання;
- active – активне або неактивне оголошення.

Поле task_id визначає зовнішній ключ таблиці Task і встановлює відношення між таблицями Task і TaskPeriod один до багатьох, оскільки одне завдання може мати багато періодів реалізації (багато оголошень), при цьому кожен період належить тільки одному завданню.

Таблиця JobCategory визначає категорії робіт, що є узагальнюючим поняттям, до якого належать і завдання, і вміння. Таблиця характеризується полями:

- id – номер категорії;
- title – назва категорії.

Таблиця TaskRequest визначає запит виконавця на реалізацію завдання. Цей запит може бути підтверджений або скасований замовником виконання завдання.

Таблиця характеризується полями:

- id – номер запиту;
- account_id – номер виконавця, який є претендентом на виконання завдання;
- taskperiod_id – оголошення (період виконання завдання), на яке виконано запит;
- created – дата і час створення запиту;
- information – інформація запиту (використовується як додаткова);

- mode – стан запиту;
- price – вартість, яку пропонує виконавець за свої роботи.

Поле `task_id` визначає зовнішній ключ таблиці `Task` і встановлює відношення між таблицями `Task` і `TaskRequest` один до багатьох, оскільки одне завдання може мати багато запитів, при цьому кожен запит стосується тільки одного завдання.

Поле `account_id` визначає зовнішній ключ таблиці `Account` і встановлює відношення між таблицями `Account` і `TaskRequest` один до багатьох, оскільки один виконавець може мати багато запитів на виконання, при цьому кожен запит належить тільки одному користувачу (виконавцю).

Окремо було виділено таблицю `Order`, хоча вона має фактично відношення один до одного з таблицею `TaskRequest`. Таблиця `Order` визначає замовлення на виконання завдання, що реалізується фактично. Це контракт між замовником та виконавцем. У цю таблицю зберігаються запити `TaskRequest`, які отримали підтримку від замовника. Підтримку на одне завдання може отримати тільки один запит, він і зберігається в таблицю `Order`. Дана таблиця також необхідна для прискорення роботи програми, адже якщо зберігати ці укладені угоди в таблиці `TaskRequest`, то для пошуку всіх замовлень потрібно буде постійно переглядати всю таблицю `TaskRequest`, що за збільшення кількості записів у ній призводить до додаткових часових витрат.

Таблиця характеризується полями:

- id – номер замовлення;
- taskrequest_id – номер запиту на виконання завдання;
- created – дата і час укладання замовлення (підтвердження замовником).

Таблиця `ReviewOrderSkill` визначає відгуки на вміння з замовлення.

Таблиця характеризується полями:

- id – номер відгуку;
- orderskill_id – номер вміння, яке реалізується в замовленні;

- account_id – замовник, який створив відгук;
- score – оцінка відгуку;
- created – дата і час формування відгуку;
- information – текст відгуку.

Поле account_id визначає зовнішній ключ таблиці Account і встановлює відношення між таблицями Account і ReviewOrderSkill один до багатьох, оскільки один замовник може залишити багато відгуків, при цьому кожен відгук на певне вміння з замовлення належить тільки одному замовнику.

Поле orderskill_id визначає зовнішній ключ таблиці OrderSkill і встановлює відношення між таблицями OrderSkill і ReviewOrderSkill один до одного, оскільки одне вміння з замовлення може мати тільки один відгук.

Таблиця ReviewSkill визначає відгуки на вміння виконавця. Вони формуються у тих випадках, коли замовник сам замовив виконання робіт, знайшовши оголошення про вміння виконавця.

Таблиця характеризується полями:

- id – номер відгуку;
- skill_id – номер вміння;
- account_id – замовник, який створив відгук;
- score – оцінка відгуку;
- created – дата і час формування відгуку;
- information – текст відгуку.

Поле account_id визначає зовнішній ключ таблиці Account і встановлює відношення між таблицями Account і ReviewSkill один до багатьох, оскільки один замовник може залишити багато відгуків, при цьому кожен відгук на певне вміння належить тільки одному замовнику.

Поле skill_id визначає зовнішній ключ таблиці Skill і встановлює відношення між таблицями Skill і ReviewSkill один до багатьох, оскільки одне вміння може мати багато відгуків, при цьому один відгук характеризує тільки одне вміння.

За описаною структурою бази даних було побудовано структурну схему бази даних (рис. 2.3).

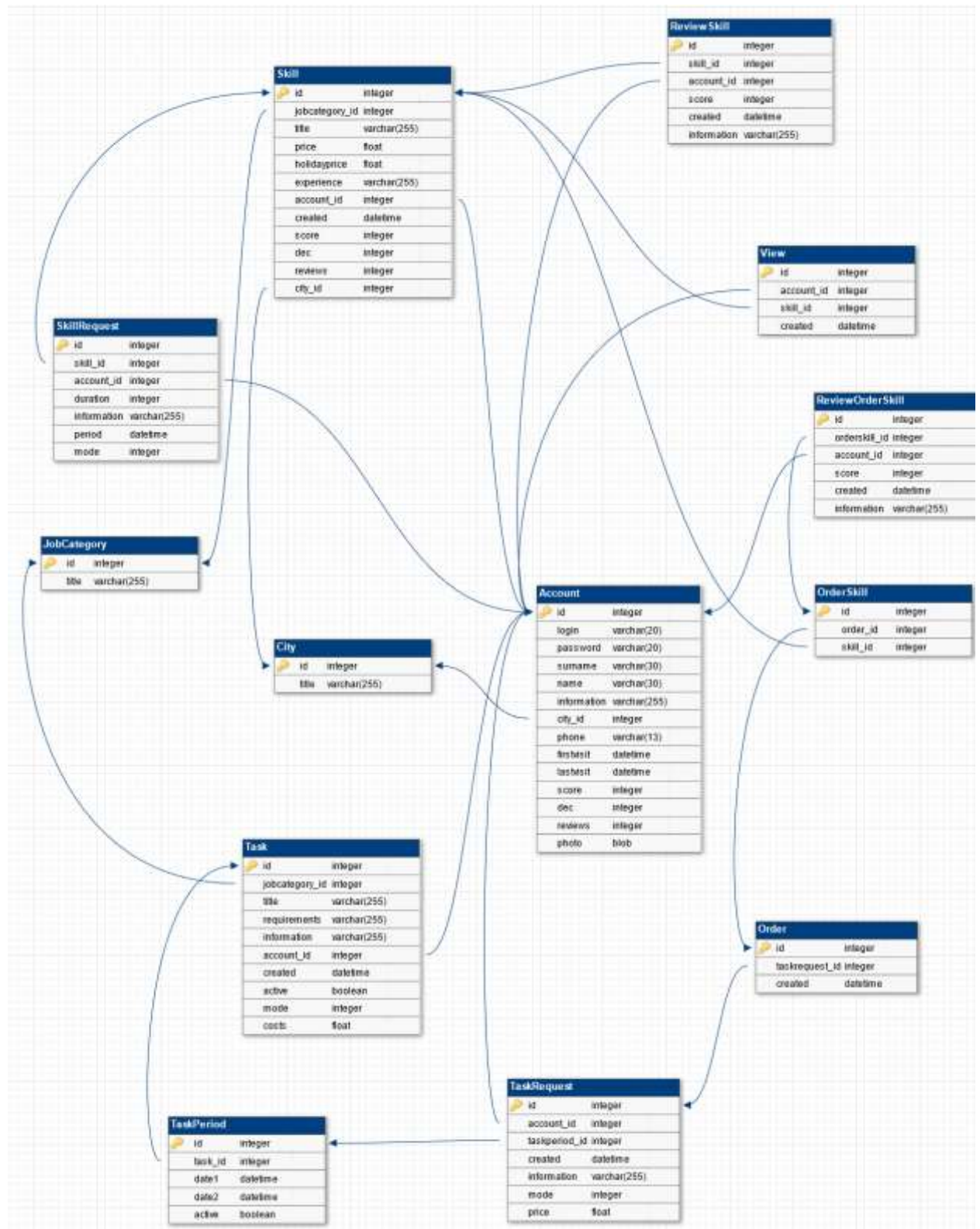


Рисунок 2.3 – Структурна схема бази даних

Таблицю ReviewOrderSkill виділено окремо від таблиці OrderSkill, оскільки не кожне вміння з замовлення може мати відгук, відповідно зі збільшенням кількості замовлень кількість пропущених полів може значно збільшуватися.

Таблиця OrderSkill визначає вміння виконавця, які знадобляться для виконання завдання. Внесення цих даних має виконувати замовник. Після підтвердження він має визначити, які вміння знадобляться. Саме на них після виконання завдання замовник зможе написати відгук.

Таблиця характеризується полями:

- id – номер запису;
- order_id – замовлення, до якого стосується вміння;
- skill_id – вміння, необхідне для виконання завдання.

Таблиця OrderSkill визначає зв'язок багато до багатьох між таблицями Order і Skill, оскільки одне замовлення може включати багато вмінь, а кожне вміння може входити в різні замовлення.

Таблиця City визначає місто і характеризується полями:

- id – номер міста;
- title – назва міста.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення обрано мову програмування PHP, що спрощує і пришвидшує процес веброзробки. Виконано моделювання функціональних вимог до програми на основі діаграм прецедентів. Визначено структуру предметної області з виділенням основних сутностей у відповідності з базовими принципами, визначеними для програми, виконано проєктування бази даних.

3 РОЗРОБКА ПРОГРАМИ

3.1 Порядок функціонування програми

До основних сутностей предметної області, які характеризуються станами, належать замовлення та запити. Замовлення пов'язані з іншою основною сутністю – вміннями. Запити пов'язані з завданнями.

Для обох цих сутностей використано однакову систему станів, що відповідно впливає і на процес взаємодії користувача з програмою. Опис буде виконано на прикладі запиту на виконання завдання.

Запит виникає в той момент, коли його створює виконавець. Одразу після цього запит отримує непідтверджений стан. До цього моменту завдання вважається активним і всі запити, які оформлені були на завдання, знаходяться також в початковому стані.

Як тільки замовник приймає рішення про запит, він одразу переводиться в один з двох станів:

- підтверджений, якщо запит підтверджено замовником;
- скасований, якщо замовник відхилив запит виконавця.

Якщо хоча би один з запитів на оголошення завдання є підтвердженим у будь-який момент підтвердженим, то усі інші автоматично переводяться в стан скасованих. Тоді існування об'єкту завершується, відповідно об'єкт більше не відображається замовнику для вибору.

Якщо запит знаходиться в підтвердженому стані, то одночасно створюється об'єкт замовлення, в який вноситься інформація про підтверджений запит.

Стан, в якому знаходиться запит, визначає, користувач в якій саме ролі має працювати з ним. Замовник працює тільки з непідтвердженими замовленнями, змінюючи їх стан тільки до підтвердженого або скасованого. Виконавець працює з запитом, тільки створюючи його і переводячи таким чином до непідтвердженого.

На рис. 3.1 представлено діаграму станів.

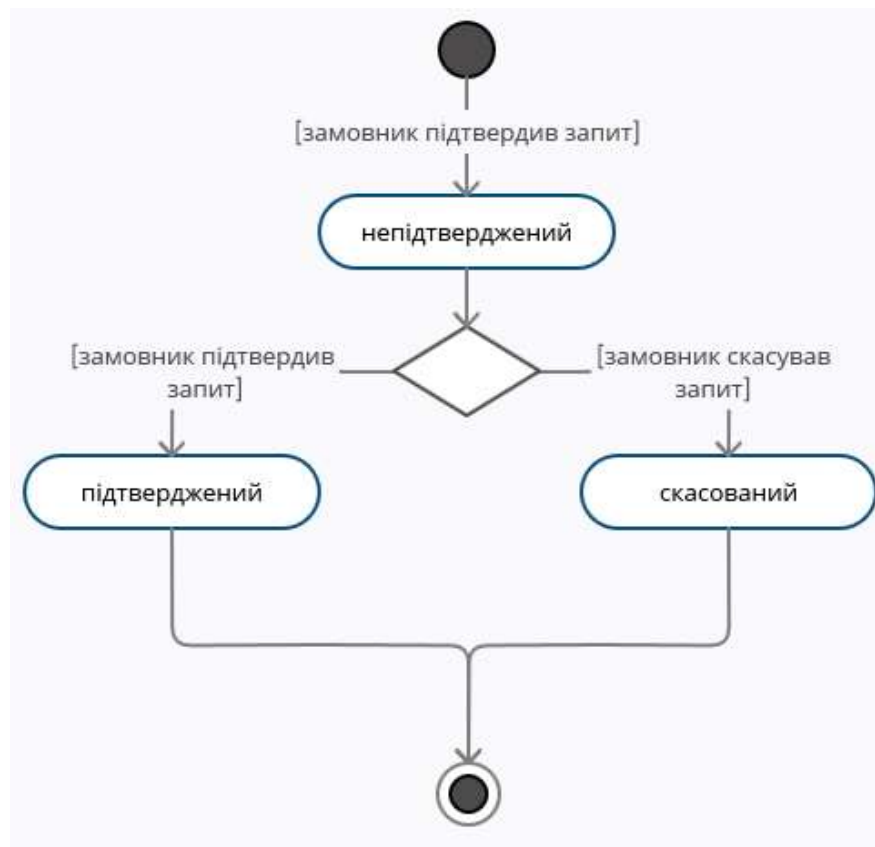


Рисунок 3.1 – Діаграма станів

3.2 Опис реалізації програми

Для побудови архітектури програмного забезпечення використано шаблон Model-View-Control (MVC).

Шаблон MVC – це шаблон програмної архітектури, побудований на основі збереження подання даних окремо від методів, що взаємодіють з даними, теоретично, добре розроблена система MVC повинна дозволити розробнику та проєктувальнику інтерфейсів (дизайнеру) працювати в одній системі, не втручаючись в роботу один одного, не надаючи спільний доступ та не редагуючи файли, над якими працює будь-яка із сторін, при цьому незважаючи на те, що MVC спочатку був розроблений для персональних сторінок, він був адаптований і широко використовується веброзробниками завдяки своєму акценту на розділенні проблем, а отже, і побічно коду, що використовується багаторазово, шаблон заохочує розробку модульних

систем, дозволяючи розробникам швидко оновлювати, додавати або навіть видаляти функціональність, складається з наступних елементів:

- модель – це елемент, що визначає постійне сховище даних, він повинен надавати доступ до даних для перегляду або збору та запису до них, і є містком між елементом View та елементом Controller у загальному шаблоні, одним із важливих аспектів моделі є те, що вона не має зв'язку або знань про те, що відбувається з даними, коли вони передаються елементам View або Controller, єдиною метою моделі є обробка даних з перетворенням їх у постійне сховище або пошук та підготовка даних для передачі до інших частин, однак модель не може бути просто підсумована як база даних або шлюз до іншої системи, модель повинна виступати відправником даних, не задаючи питань, але приймаючи всі запити, що надходять, часто найскладніша частина системи MVC, елемент Model також є вершиною всієї системи, оскільки без неї не існує зв'язку між контролером та поданням;

- подання – це елемент, який забезпечує перегляд даних, запитуваних у моделі, традиційно у вебпрограмах, побудованих за допомогою MVC, View є частиною системи, де HyperText Markup Language (HTML) формується та відображається, подання також визначає реакції користувача, який потім взаємодіє з контролером (основним прикладом є кнопка, згенерована поданням, яку користувач натискає та запускає дію в контролері), для правильного застосування архітектури MVC не повинно бути взаємодії між моделями та представленнями, вся логіка обробляється контролерами;

- контролер – це кінцева складова тріади, його робота полягає в обробці даних, які користувач вводить або подає, і він відповідно оновлює модель, це єдина частина шаблону, з якою повинен взаємодіяти користувач, контролер може бути підсумований просто як збирач інформації, який потім передає її моделі, яка буде організована для зберігання, і не містить жодної логіки, крім тієї, що потрібна для збору вхідних даних, контролер також підключений лише до одного подання та до однієї моделі, що робить його односторонньою системою потоку даних, контролеру даються завдання, які

потрібно виконувати лише тоді, коли користувач взаємодіє з поданням першим, кожна функція контролера є тригером, що визначається взаємодією користувача з поданням [11]-[12].

У відповідності з даним шаблоном було реалізовано структуру системи (рис. 3.2).

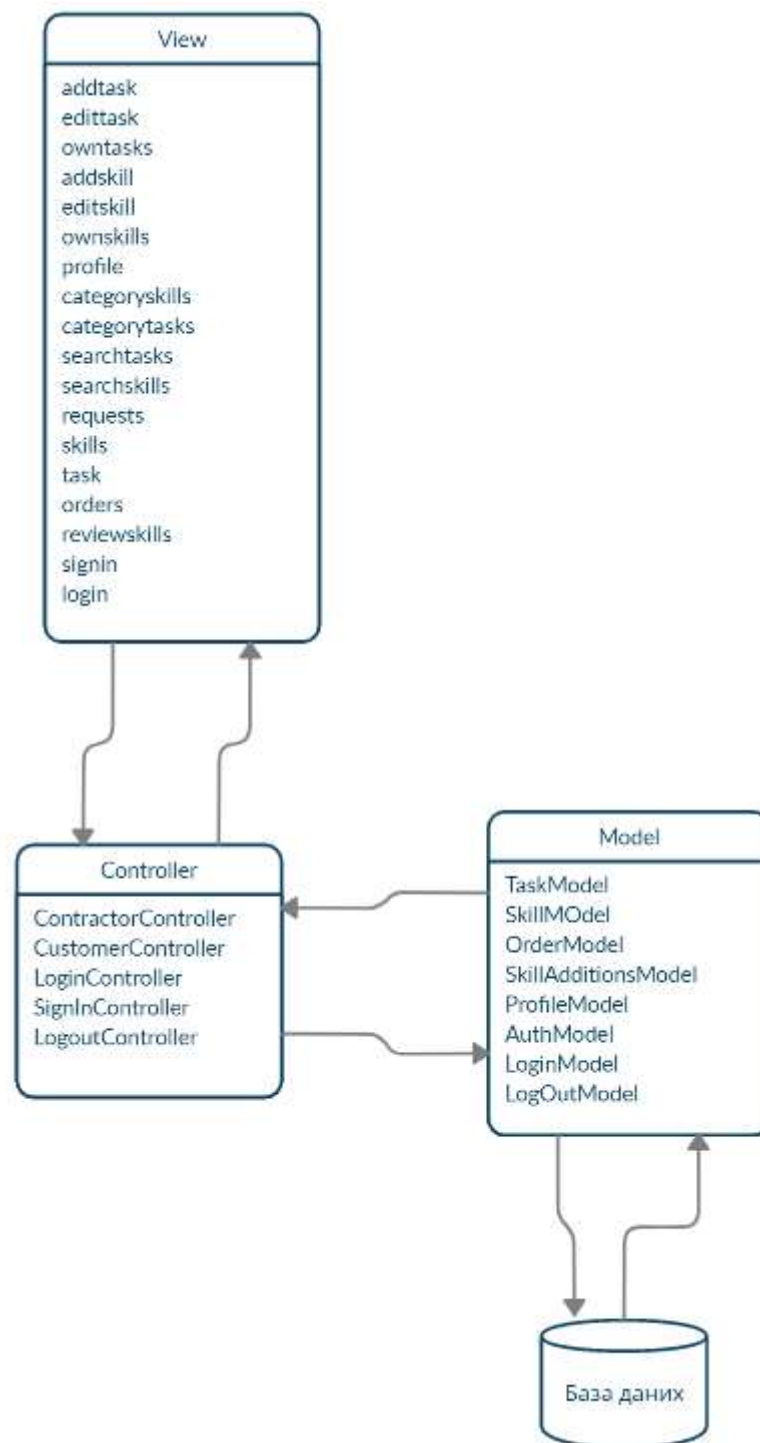


Рисунок 3.2 – Структура системи

Даний спосіб визначення структури системи було використано, враховуючи те, що не всі елементи проєкту є класами (подання реалізуються окремими сторінками, а не класами).

До структури системи входять наступні контролери:

- ContractorController – клас контролеру, що підтримує роботу виконавця, відповідно охоплює всі дії, зокрема обробку натискання кнопок, на всіх сторінках, які належать даному режиму роботи;
- CustomerController – клас контролеру, який підтримує роботу замовника і оброблює всі його дії на сторінках відповідного режиму роботи;
- LoginController – клас контролеру, який виконує підтримку роботи над авторизацією в системі;
- SignInController – клас контролеру, який виконує реєстрацію в системі;
- LogoutController – клас контролеру, який виконує підтримку виходу з облікового запису.

Класи моделей охоплюють:

- клас TaskModel – модель, що підтримує роботу з даними завдань;
- клас SkillModel – модель, що підтримує роботу з даними, які стосуються вмінь;
- клас OrderModel – модель, що підтримує роботу з даними замовлень на виконання робіт на основі вмінь;
- клас SkillAdditionsModel – модель, що підтримує роботу з додатковими даними відносно вмінь (відгуки, замовлення виконання робіт);
- клас ProfileModel – модель, яка формує дані, необхідні для роботи з профілем користувача;
- клас AuthModel – модель, що підтримує базову роботу щодо авторизації в програмі;
- клас LoginModel – модель, створена на основі AuthModel, що надає дані користувача;

– клас `LogoutModel` – модель виходу з облікового запису і визначає відповідно завершення сесії.

Клас `TaskModel` включає такі методи до свого складу.

Метод `addTask($task)` забезпечує внесення нового завдання в систему, отримуючи дані від замовника з користувацької форми через параметр `task`, заповнюючи автоматично дату створення завдання, вносячи дані власника завдання через отримання поточного авторизованого облікового запису та перетворюючи текстові описи категорії роботи, режиму та активності в числові представлення для збереження в базу даних:

```
$task['account_id'] = $_SESSION['user'];
$task['jobcategory_id'] = $this->getJobCategoryId($task['jobcategory_id']);
$task['created'] = date('Y-m-d H:i:s');
$task['mode'] = $this->getMode($task['mode']);
$task['active'] = $this->getActive($task['active']);
```

Метод `getOwnerTasks()` повертає перелік завдань, які належать авторизованому в даний момент користувачу шляхом виконання `SELECT`-запиту до бази даних.

Метод `getOwnerActiveTasks()` повертає перелік активних на даний момент завдань, які належать авторизованому в даний момент користувачу шляхом виконання `SELECT`-запиту до бази даних:

```
$data = $db->prepare('SELECT Task.id as task_id,
    Task.title as task_title,
    Task.requirements as task_requirements,
    Task.information as task_information,
    Task.created as task_created,
    Task.costs as task_costs,
    JobCategory.title
```

```

FROM Task
INNER JOIN JobCategory on JobCategory.id = Task.jobcategory_id
WHERE account_id = :account_id AND Task.active = FALSE');
$data->bindParam('account_id', $_SESSION['user']);

```

Метод `getOwnerUnactiveTasks()` повертає перелік активних на даний момент завдань, які належать авторизованому в даний момент користувачу. Необхідний для підтримки роботи щодо створення нових оголошень для роботи з завданнями, які вже були виконані раніше.

Метод `editTask($task_id, $task2)` виконує редагування даних завдання, включаючи до даного процесу редагування основних параметрів (редагувати таким чином, наприклад, автора завдання або дату створення завдання неможливо).

Метод `activateTask($task_id)` виконує оновлення завдання в базі даних шляхом зміни активності його на активовану.

Метод `getCategoryTasks($categ)` повертає перелік завдань, що відносяться до категорії `categ`.

Метод `getCityTasks($city)` повертає перелік завдань, які виконуються в заданому місті `city`.

Метод `searchTasks($title, $city)` забезпечує реалізацію пошуку, що виконується за двома фільтрами: назвою або частиною назви завдання `title`, містом виконання завдання `city`.

Метод `getOwnerTaskPeriods()` повертає всі активні на даний момент оголошення, що стосуються завдань, автором яких є авторизований користувач.

Метод `getTaskPeriods($taskid)` повертає перелік всіх оголошень, що стосуються завдання з номером `taskid`.

Метод `getTask($taskid)` повертає повну інформацію про завдання з номером `taskid`:

```

$data = $db->prepare('SELECT Task.id as task_id, Task.title as task_title,
Task.requirements as task_requirements, Task.information as task_information,
Task.created as task_created, Task.active as task_active, Task.mode as task_mode,
Task.costs as task_costs, JobCategory.title as jobcategory_title, Account.login as
account_login)

FROM Task

INNER JOIN JobCategory on JobCategory.id = Skill.jobcategory_id

INNER JOIN Account on Account.id = Skill.account_id

WHERE task_id = :task_id');

```

Метод `getMode($mode)` виконує перетворення текстового опису стану завдання на цифрове представлення.

Метод `getActive($mode)` виконує перетворення текстового опису активності завдання на цифрове представлення.

Метод `getActiveStr($mode)` повертає текстове представлення активності завдання у відповідь на його цифрове представлення.

Метод `getJobCategory()` повертає перелік всіх категорій робіт з бази даних.

Метод `getCity()` повертає перелік всіх міст, про які є згадка в базі даних.

Метод `getJobCategoryId($jc)` повертає номер категорії роботи з відповідної таблиці бази даних за назвою категорії `jc`.

Метод `getCityId($city)` повертає номер міста з назвою `city`.

Метод `addTaskPeriod($taskperiod)` виконує створення нового оголошення про виконання завдання.

Метод `unactivateTask($task_id)` виконує оновлення завдання в базі даних шляхом зміни активності його на неактивну.

Кожен клас має базову сутність предметної області, навколо якої організовується робота з даними. Наприклад, у класі `TaskModel` окрім роботи

з таблицею Task в основі виконується також робота зокрема з таблицею TaskPeriod бази даних.

Подання створено шляхом застосування HTML-шаблону [13], при цьому серед подань розроблено зокрема наступні:

- addtask.phtml – подання внесення нового завдання замовника;
- edittask.phtml – подання редагування існуючого завдання замовника;
- owntasks.phtml – подання перегляду і керування замовником усіх власних завдань;
- addskill.phtml – подання внесення нового вміння виконавця;
- editskill.phtml – подання редагування існуючого вміння виконавця;
- ownskills.phtml – подання перегляду і керування виконавцем усіх власних вмінь;
- profile.phtml – подання перегляду профілю виконавця;
- categoryskills.phtml – подання перегляду вмінь, які пропонуються виконавцями, за заданою категорією;
- categorytasks.phtml – подання перегляду завдань, які пропонуються замовниками до виконання, за заданою категорією;
- searchtasks.phtml – подання пошуку завдань, які пропонуються замовниками до виконання;
- searchskills.phtml – подання пошуку вмінь, які надаються виконавцями;
- requests.phtml – подання керування запитами виконавців на виконання завдань, використовується замовником;
- skills.phtml – подання перегляду повних даних вміння разом з відгуками щодо реалізації цих вмінь;
- task.phtml – подання перегляду детальних даних завдання;
- orders.phtml – подання керування замовленнями щодо реалізації вмінь виконавця, використовується виконавцем;
- reviewskills.phtml – подання створення відгуку за вмінням виконавця;

- signin.phtml – подання реєстрації облікового запису;
- login.phtml – подання авторизації.

3.3 Висновки за розділом 3

Розробку програмного забезпечення виконано на основі керування даним процесом за допомогою архітектурного шаблону MVC, що лежить в основі визначення структури програми. Описано основні створені класи, що реалізують вебзастосунок і підтримують шаблон MVC. Описано порядок функціонування програми.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для забезпечення пошуку виконавців дрібних робіт замовниками, яким необхідно виконати певні тимчасові роботи. Такі роботи можуть повторюватися з часом, але вони не передбачають певної циклічності.

4.2 Умови виконання програми

Серверна сторона має відповідати таким вимогам:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

4.3 Робота з програмою

У режимі роботи неавторизованого користувача можна переглядати тільки загальну інформацію (опис) про сервіс. Для того щоб скористатися функціями системи, потрібно авторизуватися або зареєструватися. Це має спонукати розширення користувацької бази.

Для кожного користувача надається весь спектр можливих функцій, але під час роботи з програмою в будь-яку мить користувач повинен обрати: він є в дану мить виконавцем чи замовником.

Виконавцю для роботи доступний весь набір функцій виконавця та можливість перейти до функцій замовника, а замовнику – відповідно весь набір його функцій та можливість перейти до функцій виконавця. Таким чином підтримується скорочення кількості можливостей для спрощення

роботи і чітке позиціонування набору функцій, щоб користувачу було зручно обирати з потрібного набору функцій.

Пункт меню замовника «Завдання» відображає окрему сторінку, на якій виводяться всі завдання, які були створені даним користувачем (рис. 4.1). За кожним завданням виводиться його назва, опис, категорія завдання (виділяється синім кольором), дата створення, активність та максимальна сума, яку готовий заплатити замовник за виконання даного завдання.

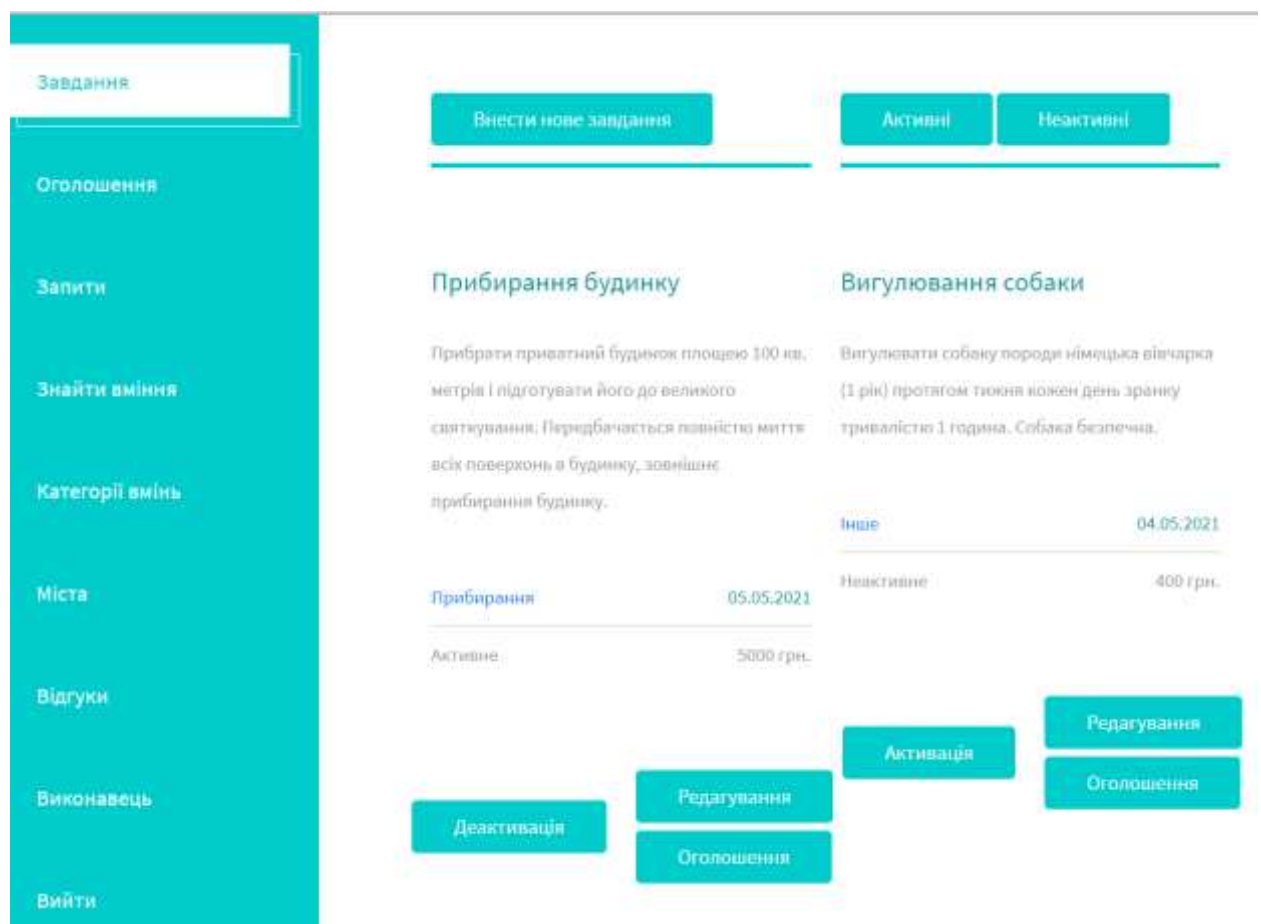


Рисунок 4.1 – Перегляд власних завдань

Натисканням на назву завдання можна перейти до повної інформації про завдання на окремій сторінці. Також можна перейти до переліку вмінь за категорією, якщо натиснути на її назву.

За кожним завданням доступні також кнопки «Деактивація» або «Активація» для того щоб змінити активність завдання. Відповідно, якщо завдання активне, то воно відображається виконавцям, якщо ні, то ні.

За допомогою кнопок «Редагування» і «Оголошення» можна відповідно відкрити сторінку редагування інформації замовлення і оголошень, тобто періодів виконання завдань, які було визначено замовником за весь час існування даного завдання.

Також на верхній панелі доступна кнопка «Внести нове завдання», що дозволяє перейти до сторінки створення завдання. Кнопка «Неактивні» надає можливість відобразити тільки завдання користувача, які в поточний момент є неактивними. Навпаки працює кнопка «Активні».

Пункт меню «Знайти вміння» дозволяє замовнику самостійно шукати, що на ринку пропонують виконавці. У кожного вміння є свій виконавець, який його пропонує. Натиснувши на його ім'я, можна перейти до сторінки профілю користувача (рис. 4.2).

На сторінці профілю відображається інформація про виконавця, що включає його ім'я, середню оцінку (обчислюється на основі середніх оцінок всіх вмінь), інформацію про місто проживання (при натисканні можна перейти на перелік вмінь, які надаються в цьому місті), опис виконавця, а також перелік його вмінь. Щодо кожного вміння надається його назва, середня оцінка, інформація про вартість години роботи зазвичай та у вихідні дні.

Замовник має наступні пункти меню:

- «Завдання» для керування переліком завдань, які було створено замовником;
- «Оголошення» для керування активними в даний момент оголошеннями на виконання завдань;
- «Запити» для керування запитами виконавців на реалізацію завдань, які визначив замовник;
- «Знайти вміння» для пошуку вмінь виконавців;

- «Категорії» для перегляду вмінь, обираючи одну з категорій з переліку;
- «Міста» для перегляду вмінь, обираючи одне з міст з переліку в якості міста реалізації вміння;
- «Відгуки» для внесення відгуків про вміння виконавців за виконаними завданнями;
- «Виконавець» для переходу в режим роботи виконавця;
- «Вийти» для виходу з облікового запису.

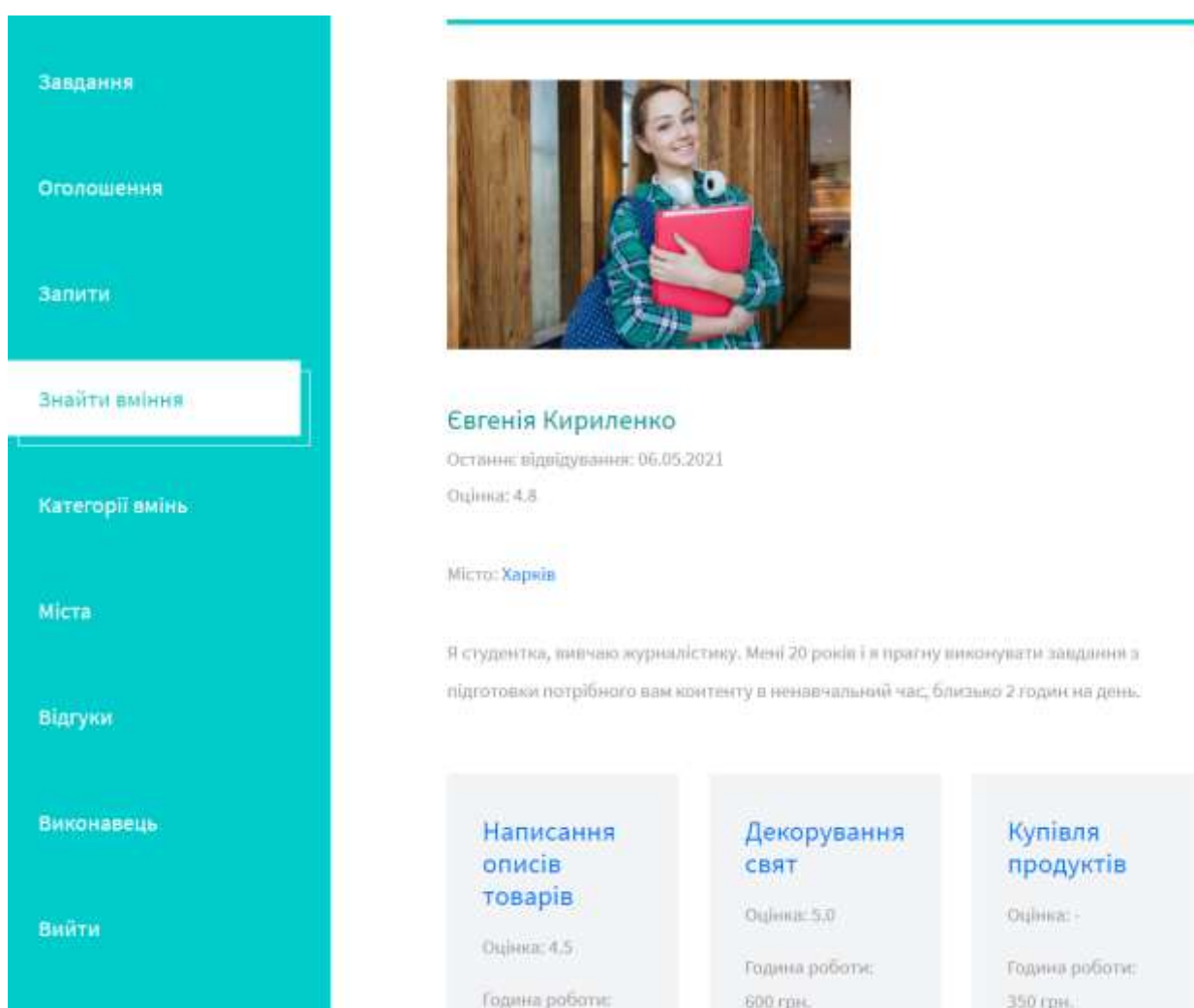


Рисунок 4.2 – Перегляд профілю виконавця

Якщо перейти до конкретного вміння, то відображається сторінка з інформацією про нього (рис. 4.3). Вона включає його назву, середню оцінку, кількість переглядів сторінки, опис досвіду робіт виконавця за цим вмінням,

вартість години роботи у звичайні та у вихідні дні, місто, де виконавець готовий працювати, а також категорію робіт.

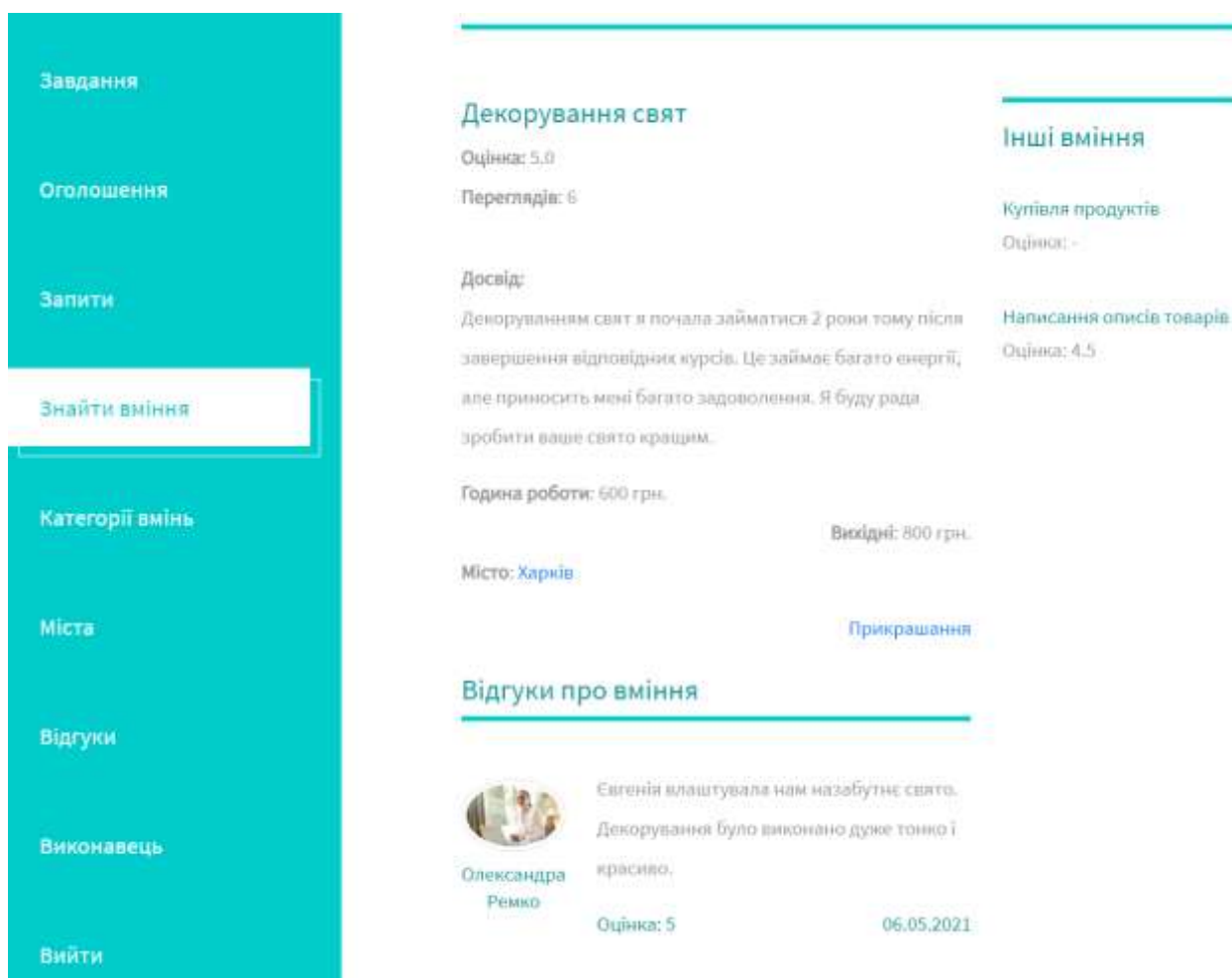


Рисунок 4.3 – Перегляд вміння виконавця

На сторінці також відображено набір інших вмінь користувача (у даному випадку відсортовані за алфавітом) та відгуків за даним вмінням. За кожним відгуком виводиться його автор, оцінка та дата публікації.

Замовник може керувати запитами виконавців на виконання завдань. Для цього передбачено сторінку «Запити», що містить спочатку перелік всіх завдань, потрібно обрати необхідне, а далі відображається (рис. 4.4) перелік запитів. Для кожного запиту міститься інформація про виконавця з можливістю переходу до профілю за його іменем. Для кожного запиту можна

натиснути на кнопку «Обрати» або «Скасувати». При натисканні кнопки «Обрати» одразу деактивуються всі інші запити.

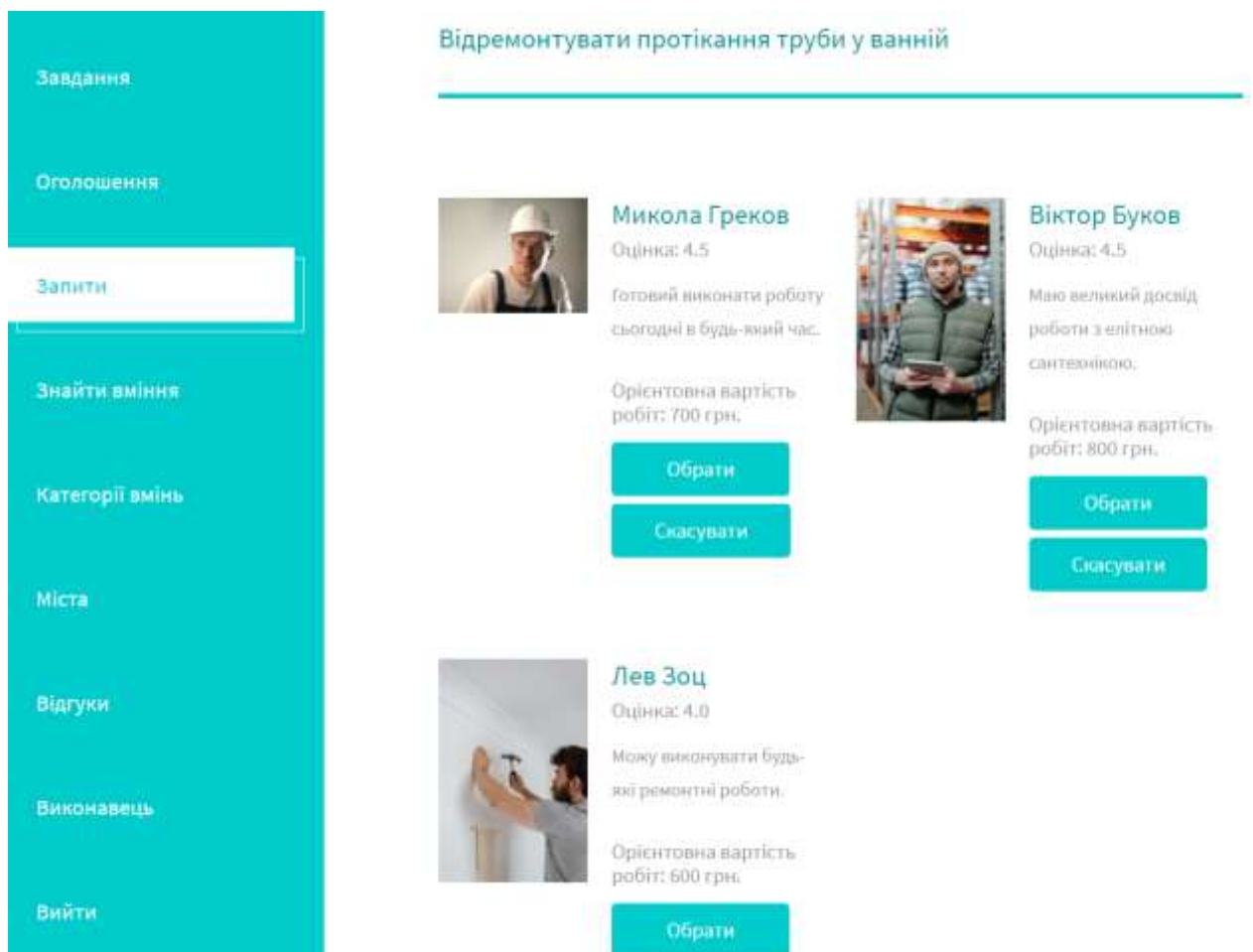


Рисунок 4.4 – Керування запитами

Виконавець має наступні пункти меню:

- «Вміння» для керування переліком вмінь, які для себе визначив виконавець;
- «Замовлення» для керування замовленнями, оформленими замовниками на виконання потрібних їм робіт;
- «Знайти завдання» для пошуку завдань, які виставлені замовником;
- «Категорії» для перегляду завдань, обираючи одну з категорій з переліку;
- «Міста» для перегляду завдань, обираючи одне з міст з переліку в якості міста виконання завдання;

– «Відгуки» для перегляду відгуків замовників на вміння виконавця за виконаними завданнями;

– «Замовник» для переходу в режим роботи замовника;

– «Вийти» для виходу з облікового запису.

Під час пошуку завдань виконавцем надається можливість обрати місто реалізації завдання і ввести назву завдання або її частину (рис. 4.5). За натискання на кнопку «Знайти» відображаються всі завдання. За кожним завданням міститься його назва (натисканням на неї можна перейти до сторінки з інформацією про оголошення, тобто датами виконання, а також можливістю подати запит), опис, вимоги, категорія, активність.

За допомогою кнопки «За вартістю» можна відсортувати всі знайдені завдання за максимальною вартістю. За замовчуванням відбувається сортування від найраніших.

Вміння

Замовлення

Знайти завдання

Категорії завдань

Міста

Відгуки

Замовник

Вийти

Місто

Назва

За вартістю **Знайти**

Прибирання снігу

Розчистити територію приватного котеджу від снігу. Роботу потрібно виконати саме сьогодні. Територія котеджу складає 630 кв.м.

Вимоги: наявність власної техніки або інструменту.

Прибирання	04.05.2021
Активне	3000 грн.

Рисунок 4.5 – Пошук завдань виконавцем

ВИСНОВКИ

Дипломну кваліфікаційну роботу виконано у відповідності з метою, що полягала в розробленні програмного забезпечення пошуку виконавців дрібних робіт для забезпечення принципу делегування власних обов'язків. Для забезпечення мети спочатку було виконано аналіз проблеми виконання дрібних робіт: розглянуто особливості, категорії і приклади дрібних робіт, виконано огляд програмних аналогів, з виділенням їх функціональних можливостей, особливостей та недоліків. Програмні аналоги включають сервіси TaskRabbit, Housekeeper, Ooch. Дані сервіси є безкоштовними для реєстрації в них, але за оформлення замовлення потребують певної сплати, вираженої у відсотках від сплаченої суми.

Для реалізації програмного забезпечення обрано мову програмування PHP, що спрощує і пришвидшує процес веброзробки. Виконано моделювання функціональних вимог до програми на основі діаграм прецедентів. Визначено структуру предметної області з виділенням основних сутностей у відповідності з базовими принципами, визначеними для програми, виконано проєктування бази даних.

Розробку виконано на основі підходу, керованого архітектурою. Реалізацію програмного забезпечення описано в роботі з представленням структури рішення.

Розроблена програма дозволяє взаємодіяти виконавцям з замовниками і навпаки шляхом створення завдань замовниками і визначення власних вмінь виконавцями. Виконавцям надається можливість оформлення запиту на виконання завдання, а замовник вирішує, якому саме запиту віддати перевагу. У свою чергу замовник може сформулювати замовлення на реалізацію вмінь для виконання необхідних йому робіт.

Усі завдання дипломної кваліфікаційної роботи повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Micro Jobs [Electronic resource] : Everything You Need To Know. – Access mode : <https://www.indeed.com/career-advice/finding-a-job/micro-jobs>
2. Kolmar, C. How to Find Small Tasks or Micro Jobs Near You [Electronic resource] / C. Kolmar. – Access mode : <https://www.zippia.com/advice/microjobs/>
3. Design Your Micro Job Site following These 4 Principles (Part 1) [Electronic resource]. – Access mode : <https://www.enginethemes.com/design-micro-job-site-following-4-principles-part-1/>
4. Design Your Micro Job Site following These 4 Principles (Part 2) [Electronic resource]. – Access mode : <https://www.enginethemes.com/design-micro-job-site-following-4-principles-part-2/>
5. TaskRabbit [Electronic resource] : Same Day Handyman, Moving & Delivery Services. – Access mode : <https://www.taskrabbit.com/>
6. Cox, D. TaskRabbit Review: 5 Things To Know Before Signing Up [Electronic resource] / D. Cox. – Access mode : <https://clark.com/save-money/taskrabbit-review/>
7. Housekeeping and House Cleaning Services | Housekeeper.com [Electronic resource]. – Access mode : <https://housekeeper.com/>
8. Oochcom – Сервіс з пошуку будівельних та ремонтних робіт [Електрон. ресурс]. – Режим доступу : <https://ooch.com.ua/>
9. Ларри, У. PHP и MySQL [Текст] : создание интернет-магазинов / У. Ларри. – СПб. : Вильямс, 2015. – 544 с.
10. Кузнецов, М. Самоучитель PHP 7 [Текст] / М. Кузнецов, И. Симдянов. – СПб. : БХВ-Петербург, 2018. – 448 с.
11. PHP Master | The MVC Pattern and PHP, Part 1 [Electronic resource]. – Access mode : <https://www.sitepoint.com/the-mvc-pattern-and-php-1/>
12. Pitt, C. Pro PHP 8 MVC [Text] : Model View Controller Architecture-Driven Application Development / Christopher Pitt. – New York : Apress, 2021. –

385 p.

13. Free Template 553 Xtra Blog [Electronic resource]. – Access mode :
<https://templatemo.com/tm-553-xtra-blog>

ДОДАТОК А
Технічне завдання

Вступ

Виконання дрібних робіт іншими людьми є виявленням принципу делегування, коли людина замість того, щоб витратити власний час на виконання завдань, які потребують багато часу, а спричиняють малий ефект в результаті, або у випадках, коли людина просто не має часу на виконання цих завдань, делегує їх на виконання іншим людям. Саме для забезпечення делегування і призначене програмне забезпечення, яке необхідно розробити.

A.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Програмне забезпечення пошуку виконавців дрібних робіт» і затверджене у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для забезпечення пошуку виконавців дрібних робіт замовниками, яким необхідно виконати певні тимчасові роботи. Такі роботи можуть повторюватися з часом, але вони не передбачають певної циклічності.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати виконання наступних функціональних характеристик:

- створення завдання на виконання роботи;

- перегляд переліку власних завдань;
- визначення оголошення завдання;
- визначення власних вмінь;
- перегляд переліку власних вмінь;
- редагування вмінь;
- пошук завдань з можливістю сортування результатів;
- пошук вмінь з можливістю сортування;
- перегляд завдань за містом виконання;
- перегляд вмінь за містом виконання;
- перегляд переліку замовлень робіт;
- замовлення робіт на основі відповідного вміння виконавця;
- підтвердження або скасування замовлення на виконання робіт виконавцем;
- визначення вмінь виконавця, яких потребує виконання завдання;
- створення відгуку про вміння;
- перегляд профілю виконавця;
- перегляд відгуків на вміння виконавця;
- створення запиту на виконання завдання;
- перегляд переліку запитів на виконання завдання;
- підтвердження або скасування запиту на виконання завдання;
- автоматичне скасування непідтверджених запитів на виконання завдання після підтвердження одного з запитів на оголошення;
- авторизація облікового запису;
- реєстрація облікового запису, яка є одночасно реєстрацією облікового запису виконавця і реєстрацією облікового запису замовника.

A.3.2 Вимоги до надійності

Для забезпечення надійності має виконуватися валідація даних, що було введено користувачем в поля форм.

A.3.3 Умови експлуатації

Експлуатація програмного забезпечення має виконуватися на основі клієнт-серверної моделі. Для цього потрібно виділити окремий комп'ютер або скористатися виділеним сервером, який буде забезпечувати підтримку реалізації вебсервера. Вебсервер має бути доступним для запитів користувачів постійно.

A.3.4 Вимоги до складу та параметрів технічних засобів

Склад та параметри технічних засобів вебсервера було визначено для реалізації можливості доступу 300 користувачів одночасно. Вимоги можуть бути знижені, якщо потрібне забезпечення меншої кількості запитів, а для більшої кількості одночасних запитів вимоги навпаки мають бути збільшені. Мінімальні системні вимоги до вебсервера включають:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

A.3.5 Вимоги до транспортування та збереження

Безпосередньо програма має транспортуватися та зберігатися через репозиторій вебсервера. Збереження на окремих фізичних носіях не передбачається.

A.3.6 Вимоги до програмної документації

Основою розробки даного програмного проєкту є технічне завдання на нього.

Опис прийнятих рішень виконується в пояснювальній записці. У розділі «Експлуатація та тестування програми» має бути представлено опис процесу роботи з програмою. Інших програмних документів не передбачається.

А.4 Порядок контролю та приймання

Календарний план завдання на дипломну кваліфікаційну роботу визначає набір етапів, за якими має забезпечуватися виконання, а також результати, які мають бути отримані за виконання відповідного етапу. Саме на основі даних результатів має забезпечуватися приймання роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла TaskModel.php

```
<?php

class TaskModel
{

    public function addTask($task)
    {
        $task['account_id'] = $_SESSION['user'];
        $task['jobcategory_id'] = $this->getJobCategoryId($task['jobcategory_id']);
        $task['created'] = date('Y-m-d H:i:s');
        $task['mode'] = $this->getMode($task['mode']);
        $task['active'] = $this->getActive($task['active']);
        unset($task['submit']);
        $db = DB::getConnection();
        $data = $db->prepare('INSERT INTO Task
            (jobcategory_id,
            title,
            requirements,
            information,
            account_id,
            created,
            active,
            mode,
            costs)
            VALUES (:jobcategory_id,
            :title,
            :requirements,
            :information,
            :account_id,
            :created,
            :active,
            :mode,
            :costs)');
        $data->execute($data);
    }

    public function getOwnerTasks()
    {
        $db = DB::getConnection();
        $data = $db->prepare('SELECT Task.id as task_id,
```

```

        Task.title as task_title,
        Task.requirements as task_requirements,
        Task.information as task_information,
        Task.created as task_created,
        Task.active as task_active,
        Task.costs as task_costs,
        JobCategory.title
        FROM Task
        INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
        WHERE account_id = :account_id');
$data->bindParam('account_id', $_SESSION['user']);
$data ->execute();
$data = $tasks->fetchAll(PDO::FETCH_ASSOC);
$data = $this->getActiveStr($data);
return $data;
}

public function getOwnerActiveTasks()
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,
Task.information as task_information,
Task.created as task_created,
Task.costs as task_costs,
JobCategory.title
        FROM Task
        INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
        WHERE account_id = :account_id AND Task.active
= TRUE');
    $data->bindParam('account_id', $_SESSION['user']);
    $data ->execute();
    $data = $tasks->fetchAll(PDO::FETCH_ASSOC);
    $data = $this->getActiveStr($data);
    return $data;
}

public function getOwnerUnactiveTasks()
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,

```

```

        Task.information as task_information,
        Task.created as task_created,
        Task.costs as task_costs,
        JobCategory.title
    FROM Task
    INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
    WHERE account_id = :account_id AND Task.active
= FALSE');

$data->bindParam('account_id', $_SESSION['user']);
$data ->execute();
$data = $tasks->fetchAll(PDO::FETCH_ASSOC);
$data = $this->getActiveStr($data);
return $data;
}

public function editTask($task_id, $task2)
{
    unset($task2['submit']);
    $task2['active'] = $this-
>getActive($task['active']);
    $task['jobcategory_id'] = $this-
>getJobCategoryId($task['jobcategory_id']);
    $task['mode'] = $this->getMode($task['mode']);
    $db = DB::getConnection();
    $data = $db->prepare('UPDATE Task SET
jobcategory_id = :jobcategory_id,
title = :title,
requirements = :requirements,
information = :information,
active = :active,
mode = :mode,
costs = :costs)
VALUES (:jobcategory_id,
:title,
:requirements,
:information,
:account_id,
:created,
:active,
:mode,
:costs)
WHERE id = :id');
    return $result = $data ->execute($task2);
}

```

```

public function activateTask($task_id)
{
    $db = DB::getConnection();
    $data = $db->prepare('UPDATE Task SET
    active = TRUE
    WHERE id = :id');
    return $result = $data ->execute($task_id);
}

public function unactivateTask($task_id)
{
    $db = DB::getConnection();
    $data = $db->prepare('UPDATE Task SET
    active = FALSE
    WHERE id = :id');
    return $result = $data ->execute($task_id);
}

public function getCategoryTasks($categ)
{
    $category = this->getCategoryId($categ);
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Task.id as task_id,
    Task.title as task_title,
    Task.requirements as task_requirements,
    Task.information as task_information,
    Task.created as task_created,
    Task.jobcategory_id as task_jobcategoryid,
    Task.active as task_active,
    Task.costs as task_costs,
    FROM Task
    INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
    WHERE task_jobcategoryid = :task_jobcategoryid
    AND task_active = TRUE
    ORDER BY task_id DESC');
    $data->bindParam('task_jobcategoryid', $category);
    $data ->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}

public function getCategorySortedTasks($categ)
{
    $category = this->getCategoryId($categ);
    $db = DB::getConnection();

```

```

        $data = $db->prepare('SELECT Task.id as task_id,
        Task.title as task_title,
        Task.requirements as task_requirements,
        Task.information as task_information,
        Task.created as task_created,
        Task.jobcategory_id as task_jobcategoryid,
        Task.active as task_active,
        Task.costs as task_costs,
            FROM Task
            INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
            WHERE task_jobcategoryid = :task_jobcategoryid
AND task_active = TRUE
            ORDER BY task_costs DESC');
        $data->bindParam('task_jobcategoryid', $category);
        $data ->execute();
        $data = $data->fetchAll(PDO::FETCH_ASSOC);
        return $data;
    }

    public function getCityTasks($city)
    {
        $city_id = this->getCityId($city);
        $db = DB::getConnection();
        $data = $db->prepare('SELECT Task.id as task_id,
        Task.title as task_title,
        Task.requirements as task_requirements,
        Task.information as task_information,
        Task.created as task_created,
        Task.jobcategory_id as task_jobcategoryid,
        Task.active as task_active,
        Task.costs as task_costs,
        Account.city_id as account_cityid
            FROM Task
            INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
            INNER JOIN Account on Account.id =
Task.account_id
            WHERE account_cityid = :account_cityid AND
task_active = TRUE
            ORDER BY task_id DESC');
        $data->bindParam('account_cityid', $city_id);
        $data ->execute();
        $data = $data->fetchAll(PDO::FETCH_ASSOC);
        return $data;
    }

```

```

public function getCitySortedTasks($city)
{
    $city_id = this->getCityId($city);
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,
Task.information as task_information,
Task.created as task_created,
Task.jobcategory_id as task_jobcategoryid,
Task.active as task_active,
Task.costs as task_costs,
Account.city_id as account_cityid
FROM Task
INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
INNER JOIN Account on Account.id =
Task.account_id
WHERE account_cityid = :account_cityid AND
task_active = TRUE
ORDER BY task_costs DESC');
    $data->bindParam('account_cityid', $city_id);
    $data->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}

public function searchTasks($title, $city)
{
    if (!is_null($city))
    {
        $city_id = this->getCityId($city);
        $db = DB::getConnection();
        $data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,
Task.information as task_information,
Task.created as task_created,
Task.jobcategory_id as task_jobcategoryid,
Task.active as task_active,
Task.costs as task_costs,
Account.city_id as account_cityid
FROM Task
INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id

```

```

INNER JOIN Account on Account.id =
Task.account_id
WHERE account_cityid = :account_cityid AND
task_active = TRUE AND
task_title LIKE ?
ORDER BY task_id DESC');
$data->bindParam('account_cityid', $city_id);
$data->bindParam('s', $title);
$data->execute();
$data = $data->fetchAll(PDO::FETCH_ASSOC);
return $data;
}

else
{
$db = DB::getConnection();
$data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,
Task.information as task_information,
Task.created as task_created,
Task.jobcategory_id as task_jobcategoryid,
Task.active as task_active,
Task.costs as task_costs,
FROM Task
INNER JOIN JobCategory on JobCategory.id =
Task.jobcategory_id
INNER JOIN Account on Account.id =
Task.account_id
WHERE task_active = TRUE AND
task_title LIKE ?
ORDER BY task_id DESC');
$data->bindParam('s', $title);
$data->execute();
$data = $data->fetchAll(PDO::FETCH_ASSOC);
return $data;
}

}

public function getOwnerTaskPeriods()
{
$db = DB::getConnection();
$data = $db->prepare('SELECT Task.id as task_id,
Task.title as task_title,
Task.requirements as task_requirements,
Task.information as task_information,

```

```

        Task.created as task_created,
        Task.costs as task_costs,
        Task.account_id as account_id,
        TaskPeriod.date1 as tp_d1,
        TaskPeriod.date2 as tp_d2,
        TaskPeriod.active as tp_active
        FROM TaskPeriod
        INNER JOIN Task on Task.id = TaskPeriod.task_id
        WHERE account_id = :account_id
        AND tp_active = TRUE
        ORDER_BY tp_d2 DESC');
$data->bindParam('account_id', $_SESSION['user']);
$data ->execute();
$data = $tasks->fetchAll(PDO::FETCH_ASSOC);
return $data;
}

public function getTaskPeriods($taskid)
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Task.id as task_id,
        TaskPeriod.date1 as tp_d1,
        TaskPeriod.date2 as tp_d2,
        TaskPeriod.active as tp_active
        FROM TaskPeriod
        INNER JOIN Task on Task.id = TaskPeriod.task_id
        WHERE task_id = :task_id
        ORDER_BY tp_d2 DESC');
    $data->bindParam('task_id', $taskid);
    $data ->execute();
    $data = $tasks->fetchAll(PDO::FETCH_ASSOC);
    $data = $this->getActiveStr($data);
    return $data;
}

public function getTask($taskid)
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT
        Task.id as task_id,
        Task.title as task_title,
        Task.requirements as task_requirements,
        Task.information as task_information,
        Task.created as task_created,
        Task.active as task_active,
        Task.mode as task_mode,

```

```

        Task.costs as task_costs,
        JobCategory.title as jobcategory_title,
        Account.login as account_login)
        FROM Task
        INNER JOIN JobCategory on JobCategory.id =
Skill.jobcategory_id
        INNER JOIN Account on Account.id =
Skill.account_id
        WHERE task_id = :task_id');
$data->bindParam('task_id', $taskid);
$data->execute($data);
}

public function getMode($mode)
{
    if($mode == 'повторюваний') {
        return 1;
    }
    if($mode == 'одноразовий') {
        return 0;
    }
}

public function getActive($mode)
{
    if($mode == 'активний') {
        return 1;
    }
    if($mode == 'неактивний') {
        return 0;
    }
}

public function getActiveStr($mode)
{
    if($mode == 1) {
        return 'активний';
    }
    if($mode == 0) {
        return 'неактивний';
    }
}

public function getJobCategory()
{
    $db = DB::getConnection();

```

```

        return $db->query('SELECT * FROM JobCategory')-
>fetchAll();
    }

    public function getCity()
    {
        $db = DB::getConnection();
        return $db->query('SELECT * FROM City')-
>fetchAll();
    }

    public function getJobCategoryId($jc)
    {
        $jca = $this->getJobCategory();
        foreach ($jca as $categ) {
            if($categ['title'] == $jc) {
                return $categ['id'];
            }
        }
    }

    public function getCityId($city)
    {
        $cit = $this->getCity ();
        foreach ($cit as $c) {
            if($c['title'] == $city) {
                return $c['id'];
            }
        }
    }

    public function addTaskPeriod($taskperiod)
    {
        $taskperiod ['active'] = $this-
>getActive($taskperiod ['active']);
        unset($taskperiod ['submit']);
        $db = DB::getConnection();
        $data = $db->prepare('INSERT INTO TaskPeriod
            (task_id,
            datel,
            date2,
            active)
            VALUES (:task_id,
            :datel,
            :date2,
            :active) ');
    }

```

```

        $data->execute($task);
    }

}

```

Б.2 Текст файла SkillModel.php

```

<?php

class SkillModel
{

    public function addSkill($skill)
    {
        $skill ['account_id'] = $_SESSION['user'];
        $skill      ['jobcategory_id']      =      $this->
>getJobCategoryId($skill ['jobcategory_id']);
        $skill ['created'] = date('Y-m-d H:i:s');
        $skill      ['city_id']      =      $this->getCityId($skill
['city_id']);

        unset($task['submit']);
        $db = DB::getConnection();
        $data = $db->prepare('INSERT INTO Skill
        (jobcategory_id,
        title,
        price,
        holidayprice,
        experience,
        account_id,
        created,
        score,
        dec,
        reviews,
        city_id)
        VALUES (:jobcategory_id,
        :title,
        :price,
        :holidayprice,
        :experience,
        :account_id,
        :created,
        :score,
        :dec,
        :reviews,

```

```

        :city_id)');
        $data->execute($skill);
    }

    public function getOwnerSkills()
    {
        $db = DB::getConnection();
        $data = $db->prepare('SELECT Skill.id as skill_id,
        Skill.title as skill_title,
        Skill.price as skill_price,
        Skill.holidayprice as skill_holidayprice,
        Skill.created as skill_created,
        Skill.experience as skill_experience,
        Skill.score as skill_score,
        Skill.dec as skill_dec,
        Skill.reviews as skill_reviews,
        JobCategory.title,
        City.title
        FROM Skill
        INNER JOIN City on City.id = Skill.city_id
        INNER JOIN JobCategory on JobCategory.id =
Skill.jobcategory_id
        WHERE account_id = :account_id');
        $data->bindParam('account_id', $_SESSION['user']);
        $data->execute();
        $data = $skills->fetchAll(PDO::FETCH_ASSOC);
        return $data;
    }

    public function editSkill($skill_id, $skill2)
    {
        unset($skill2['submit']);
        $skill2['jobcategory_id'] = $this->getJobCategoryId($skill2['jobcategory_id']);
        $skill2['city_id'] = $this->getCityId($skill2['city_id']);
        $db = DB::getConnection();
        $data = $db->prepare('UPDATE Skill SET
        jobcategory_id,
        title,
        price,
        holidayprice,
        experience,
        city_id)
        VALUES (:jobcategory_id,
        :title,

```

```

        :price,
        :holidayprice,
        :experience,
        :city_id)
        WHERE id = :id');
    return $result = $data ->execute($skill2);
}

public function getCategorySkills($categ)
{
    $category = this->getCategoryId($categ);
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Skill.id as skill_id,
Skill.jobcategory_id as skill_jobcategoryid,
Skill.title as skill_title,
Skill.price as skill_price,
Skill.holidayprice as skill_holidayprice,
Skill.created as skill_created,
Skill.experience as skill_experience,
Skill.score as skill_score,
Skill.dec as skill_dec,
Skill.reviews as skill_reviews,
Skill.score/Skill.reviews as skill_review,
City.title
        FROM Skill
        INNER JOIN City on City.id = Skill.city_id
        WHERE
            skill_jobcategoryid
:skill_jobcategoryid
        ORDER BY skill_id DESC');
    $data->bindParam('skill_jobcategoryid', $category);
    $data ->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}

public function getCategorySortedSkills($categ)
{
    $category = this->getCategoryId($categ);
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Skill.id as skill_id,
Skill.jobcategory_id as skill_jobcategoryid,
Skill.title as skill_title,
Skill.price as skill_price,
Skill.holidayprice as skill_holidayprice,
Skill.created as skill_created,
Skill.experience as skill_experience,

```

```

        Skill.score as skill_score,
        Skill.dec as skill_dec,
        Skill.reviews as skill_reviews,
        Skill.score/Skill.reviews as skill_review,
        City.title
        FROM Skill
        INNER JOIN City on City.id = Skill.city_id
        WHERE skill_jobcategoryid =
:skill_jobcategoryid
        ORDER BY skill_review DESC');
$data->bindParam('skill_jobcategoryid', $category);
$data ->execute();
$data = $data->fetchAll(PDO::FETCH_ASSOC);
return $data;
}

public function getCitySkills($city)
{
    $city_id = this->getCityId($city);
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Skill.id as skill_id,
Skill.jobcategory_id as skill_jobcategoryid,
Skill.title as skill_title,
Skill.price as skill_price,
Skill.holidayprice as skill_holidayprice,
Skill.created as skill_created,
Skill.experience as skill_experience,
Skill.score as skill_score,
Skill.dec as skill_dec,
Skill.reviews as skill_reviews,
Skill.score/Skill.reviews as skill_review,
Skill.city_id as skill_cityid
        FROM Skill
        INNER JOIN City on City.id = Skill.city_id
        WHERE skill_cityid = :skill_cityid
        ORDER BY skill_id DESC');
    $data->bindParam('skill_cityid', $city_id);
    $data ->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}

public function getCitySortedSkills($city)
{
    $city_id = this->getCityId($city);
    $db = DB::getConnection();

```

```

        $data = $db->prepare('SELECT Skill.id as skill_id,
        Skill.jobcategory_id as skill_jobcategoryid,
        Skill.title as skill_title,
        Skill.price as skill_price,
        Skill.holidayprice as skill_holidayprice,
        Skill.created as skill_created,
        Skill.experience as skill_experience,
        Skill.score as skill_score,
        Skill.dec as skill_dec,
        Skill.reviews as skill_reviews,
        Skill.score/Skill.reviews as skill_review,
        Skill.city_id as skill_cityid
        FROM Skill
        INNER JOIN City on City.id = Skill.city_id
        WHERE skill_cityid = :skill_cityid
        ORDER BY skill_review DESC');
        $data->bindParam('skill_cityid', $city_id);
        $data ->execute();
        $data = $data->fetchAll(PDO::FETCH_ASSOC);
        return $data;
    }

    public function searchSkills($title, $city)
    {
        if (!is_null($city))
        {
            $city_id = this->getCityId($city);
            $db = DB::getConnection();
            $data = $db->prepare('SELECT Skill.id as skill_id,
            Skill.jobcategory_id as skill_jobcategoryid,
            Skill.title as skill_title,
            Skill.price as skill_price,
            Skill.holidayprice as skill_holidayprice,
            Skill.created as skill_created,
            Skill.experience as skill_experience,
            Skill.score as skill_score,
            Skill.dec as skill_dec,
            Skill.reviews as skill_reviews,
            Skill.score/Skill.reviews as skill_review,
            Skill.city_id as skill_cityid
            FROM Skill
            WHERE skill_cityid = :skill_cityid AND
            skill_title LIKE ?
            ORDER BY skill_review DESC');
            $data->bindParam('skill_cityid', $city_id);
            $data->bindParam('s', $title);

```

```

    $data ->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}

else
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT Skill.id as skill_id,
Skill.jobcategory_id as skill_jobcategoryid,
Skill.title as skill_title,
Skill.price as skill_price,
Skill.holidayprice as skill_holidayprice,
Skill.created as skill_created,
Skill.experience as skill_experience,
Skill.score as skill_score,
Skill.dec as skill_dec,
Skill.reviews as skill_reviews,
Skill.score/Skill.reviews as skill_review,
Skill.city_id as skill_cityid,
City.title as city_title
        FROM Skill
        WHERE Skill.city_id = City.id AND
        skill_title LIKE ?
        ORDER BY skill_review DESC');
    $data->bindParam('s', $title);
    $data ->execute();
    $data = $data->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}
}

public function getSkill($skillid)
{
    $db = DB::getConnection();
    $data = $db->prepare('SELECT
Skill.id as skill_id,
Skill.jobcategory_id as skill_jobcategoryid,
Skill.title as skill_title,
Skill.price as skill_price,
Skill.holidayprice as skill_holidayprice,
Skill.account_id as skill_accountid,
Skill.created as skill_created,
Skill.experience as skill_experience,
Skill.score as skill_score,
Skill.dec as skill_dec,

```

```
Skill.reviews as skill_reviews,  
Skill.score/Skill.reviews as skill_review,  
Skill.city_id as skill_cityid,  
City.title as city_title)  
    FROM Skill  
    INNER JOIN City on City.id = Skill.city_id  
    WHERE skill_id = :skill_id ');  
$data->bindParam('skill_id', $skillid);  
$data->execute($data);  
}  
  
}
```

ДОДАТОК В
Слайди презентації

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ ВИКОНАВЦІВ ДРІБНИХ РОБІТ

Дипломна кваліфікаційна робота

Виконав:

студент групи КНТ-128сп

А.В. Кайдаш

Керівник роботи:

к.т.н., доцент

С.Ю. Скрупський

2021

Рисунок В.1 – Слайд № 1

Об'єкт, предмет, мета роботи

Об'єкт роботи – процес виконання дрібних робіт.

Предмет роботи – програмне забезпечення пошуку виконавців дрібних робіт.

Мета роботи – розроблення програмного забезпечення пошуку виконавців дрібних робіт для забезпечення принципу делегування власних обов'язків.

Рисунок В.2 – Слайд № 2

Завдання роботи

- ✓ розробка технічного завдання і визначення вимог до програми;
- ✓ розробка архітектури програми;
- ✓ розробка програми;
- ✓ тестування програми.

3

Рисунок В.3 – Слайд № 3

Вибір мови програмування

Параметр	Java	PHP
Виконання	Компілятор	Інтерпретатор
Парадигми програмування	Об'єктно-орієнтована	Об'єктно-орієнтована, функціональна
Призначення	Загальне	Загальне, початково – веброзробка
Веброзробка, серверна частина	+	+
Менші витрати на розробку і підтримку	+–	++

4

Рисунок В.4 – Слайд № 4

Діаграми прецедентів замовника та виконавця

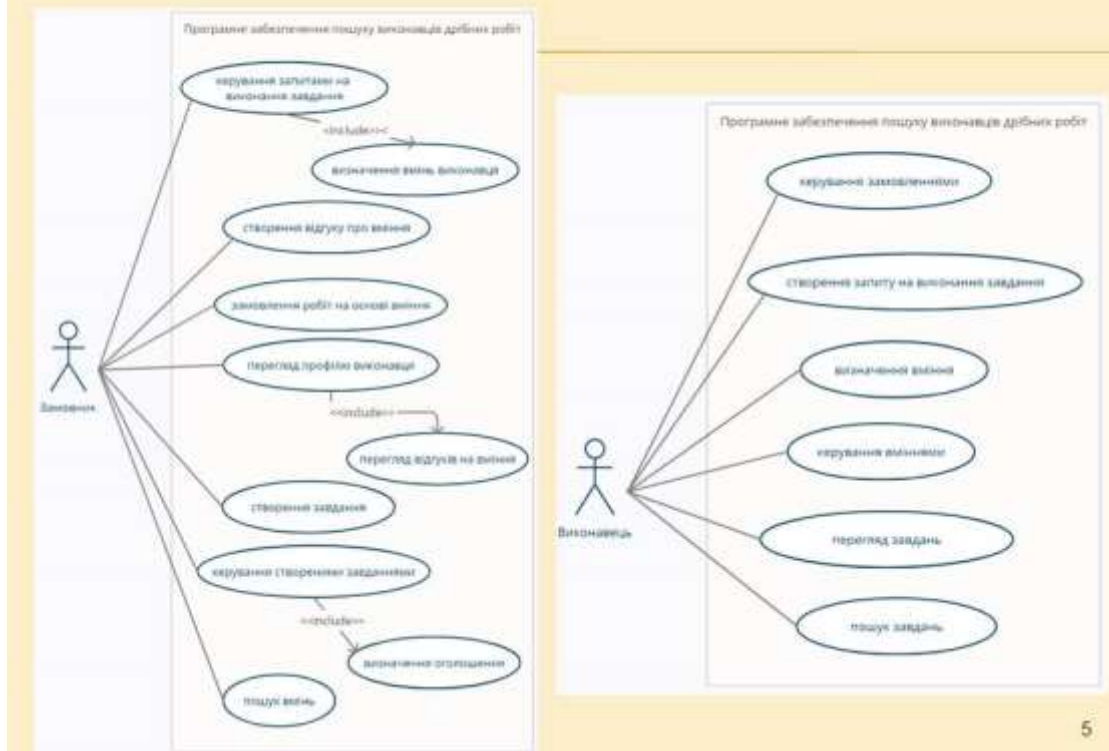


Рисунок В.5 – Слайд № 5

Схема бази даних

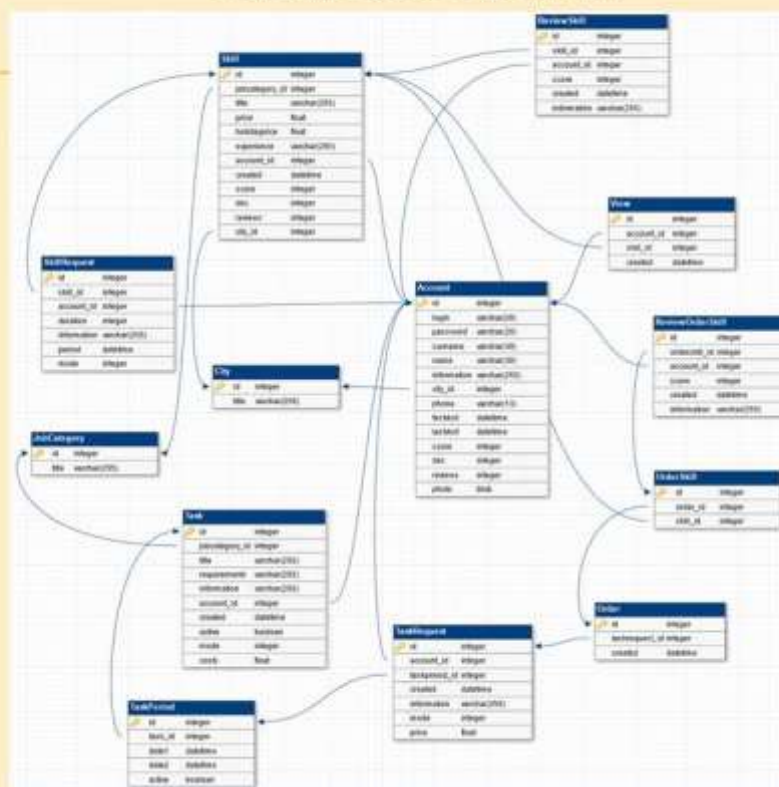
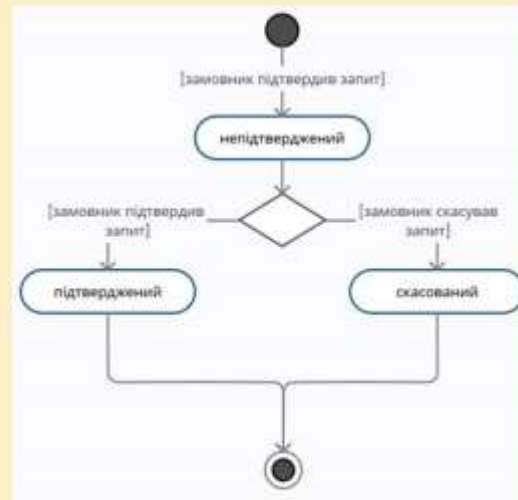


Рисунок В.6 – Слайд № 6

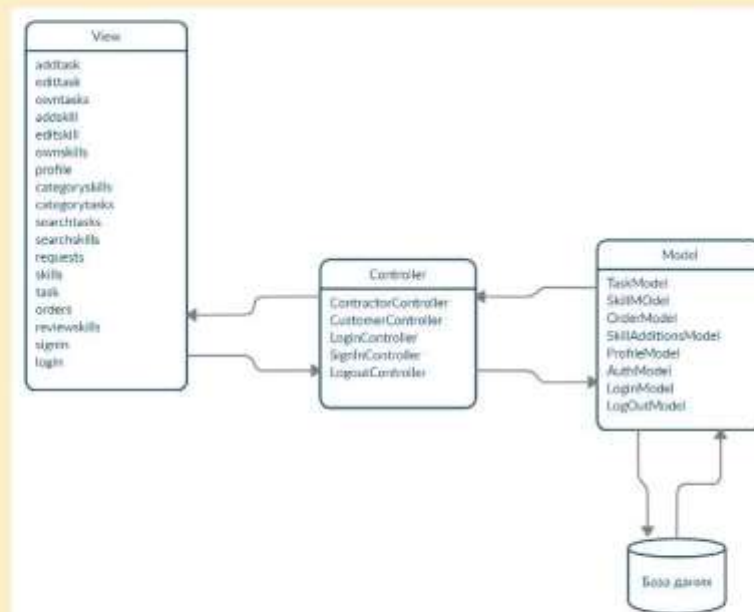
Діаграма станів запиту на виконання робіт



7

Рисунок В.7 – Слайд № 7

Структура системи



8

Рисунок В.8 – Слайд № 8

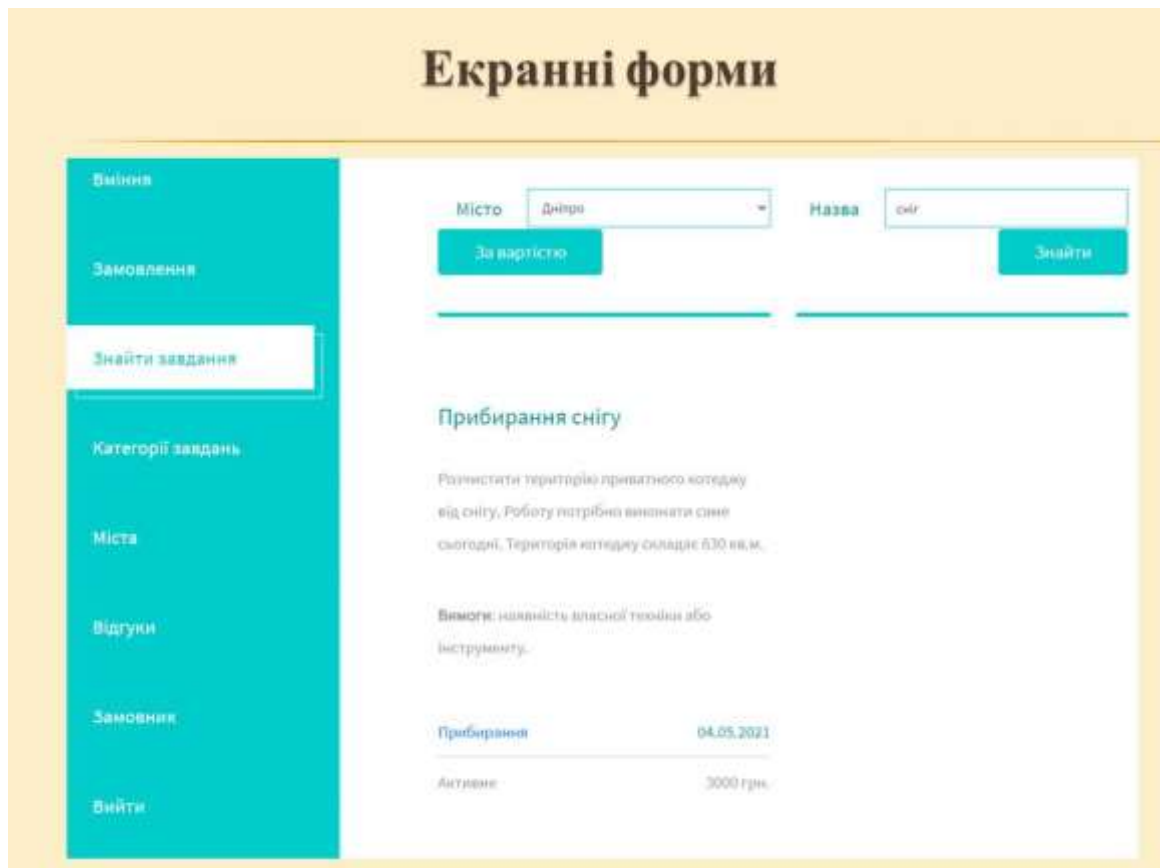


Рисунок В.9 – Слайд № 9

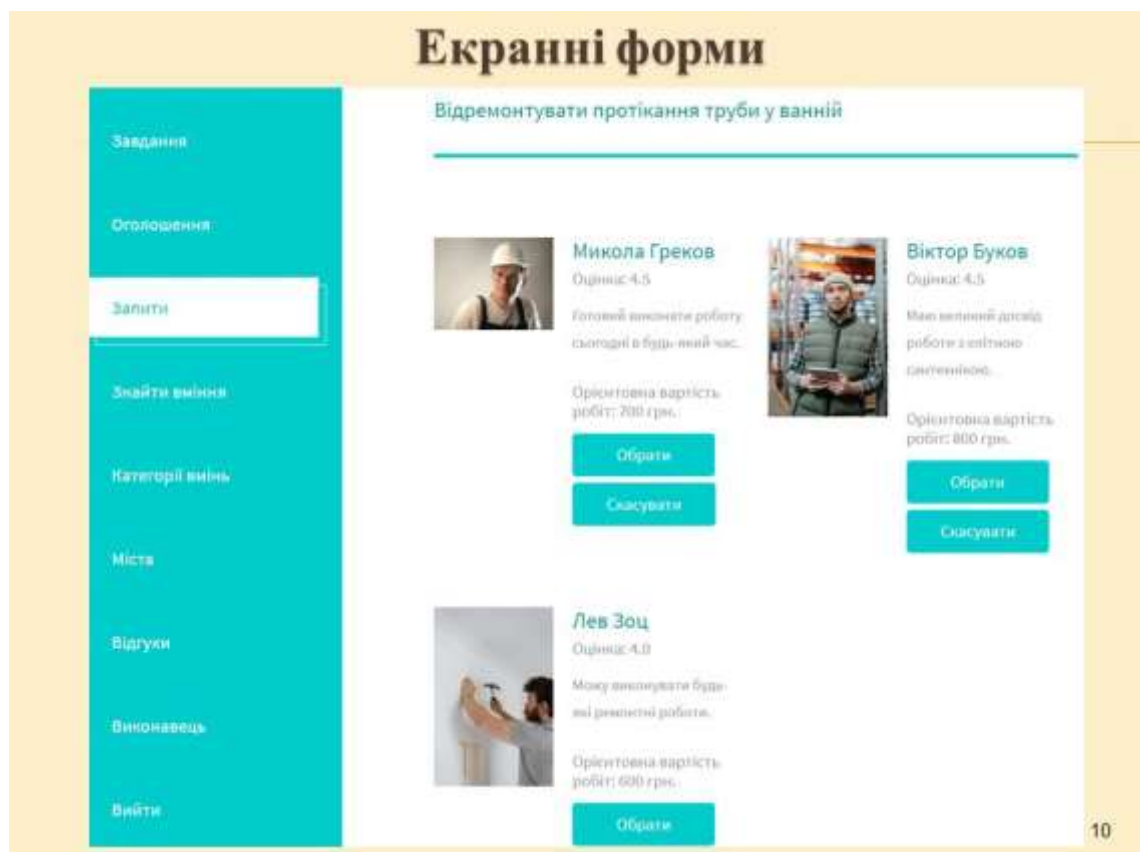


Рисунок В.10 – Слайд № 10

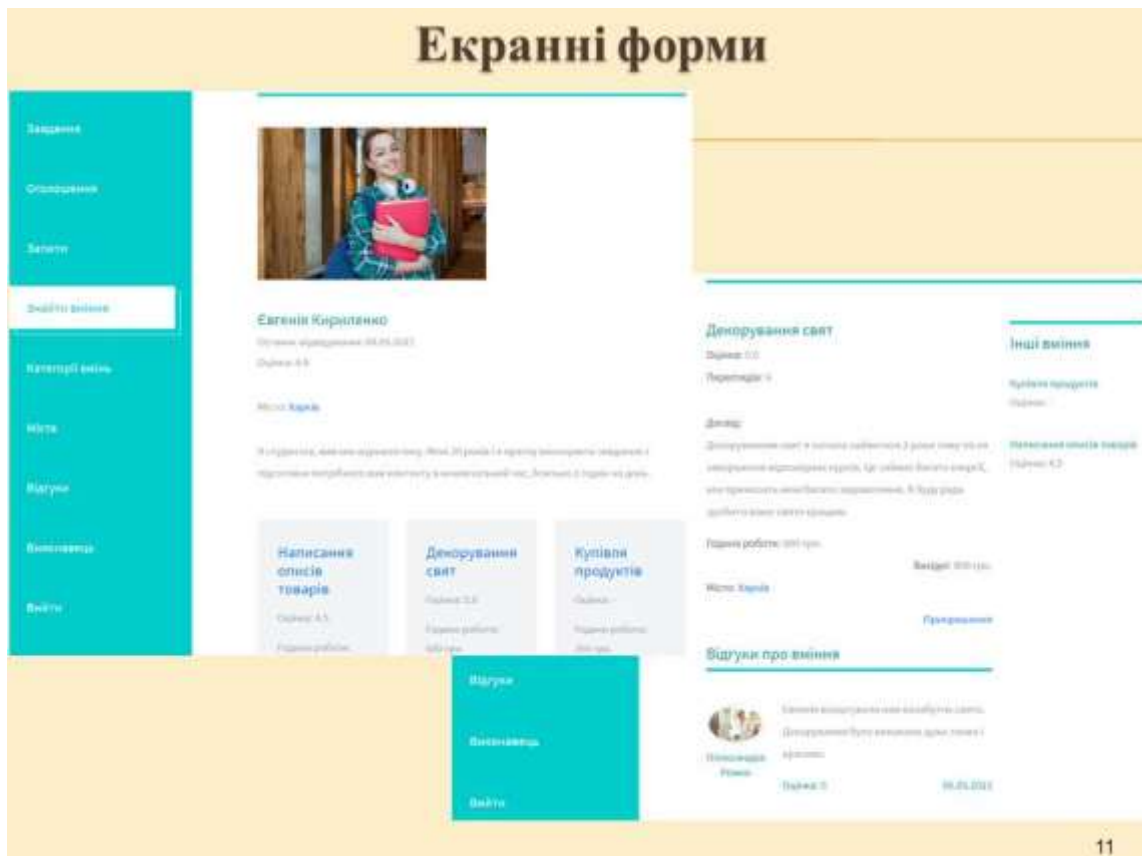


Рисунок В.11 – Слайд № 11

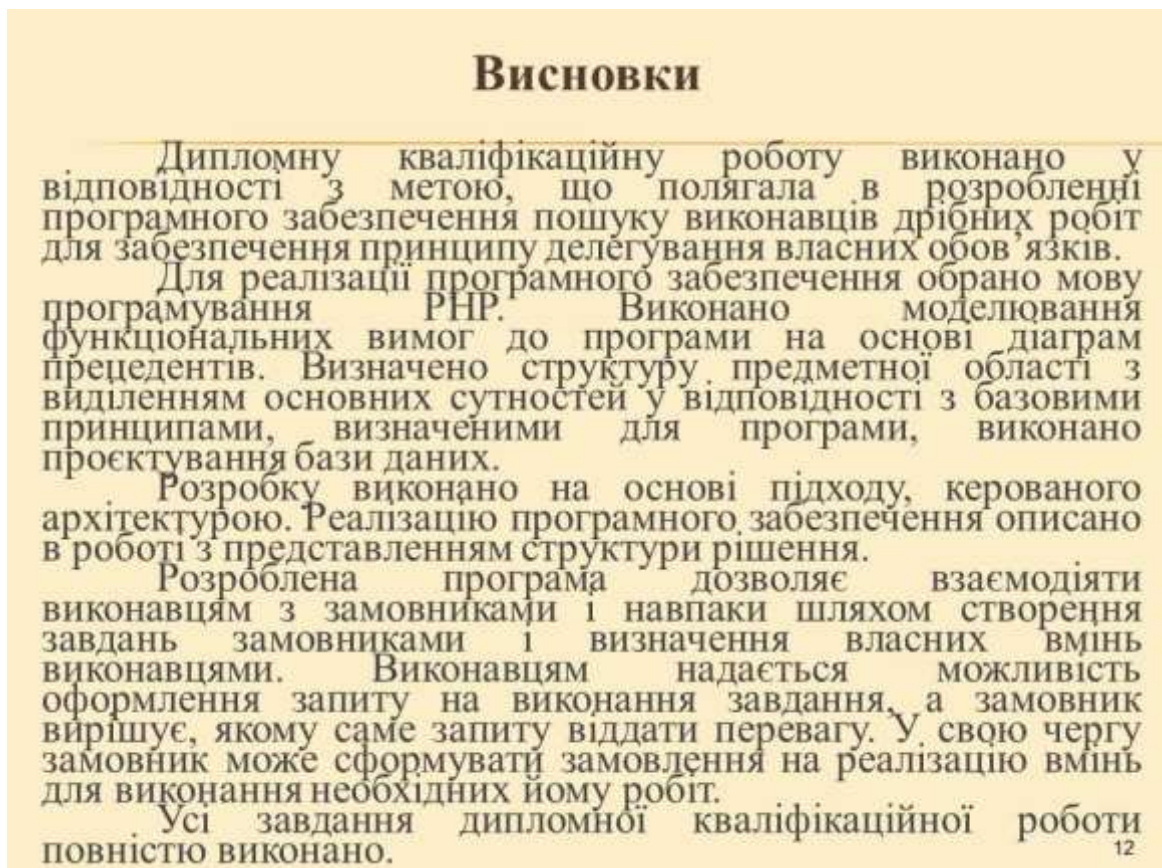


Рисунок В.12 – Слайд № 12