МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Тернопільський фаховий коледж
Тернопільського національного технічного університету імені Івана Пулюя»

Відділення телекомунікацій та електронних систем

Циклова комісія комп'ютерних наук

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра

на тему: **Розробка інформаційної системи Центру зайнятості**

Виконав: студент IV курсу, групи КН-421

спеціальності:

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Коцур О. Ф.
(прізвище та ініціали)

Керівник Сербін В. С.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

м. Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерних наук
Освітньо-професійний ступінь «фаховий молодший бакалавр»
Спеціальність 122 «Комп'ютерні науки»
Галузь знань 12 «Інформаційні технології»

ЗАТВЕРДЖУЮ
Голова циклової комісії комп'ютерних наук
_____ Галина МАРЦІЯШ
« _____» _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Коцур Олександр Федорович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної системи Центру зайнятості

керівник роботи Сербін Володимир Сергійович,
(прізвище, ім'я, по батькові)
затверджені наказом вищого навчального закладу № _____
2. Строк подання студентом роботи: _____ р.
3. Вихідні дані до роботи: технічне завдання на розробку програмного забезпечення, мови програмування: Java, JavaScript; фреймворки: Spring Boot, Hibernate, Svetle стандарти IEEE 830-1998, IEEE 29148-2018, IEEE 29119, ГОСТ 34.602-89.

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

1 Загальний розділ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

1.2.1 Найменування та область застосування

1.2.2 Призначення розробки

1.2.3 Вимоги до програмного забезпечення

1.2.4 Вимоги до програмної документації

1.2.5 Техніко-економічні показники

1.2.6 Стадії та етапи розробки

1.2.7 Порядок контролю та прийому

2 Розробка технічного та робочого проєкту

2.1 Розробка загальної структури і варіантів використання програми

2.2 Проєктування і опис інтерфейсу користувача

2.3 Розробка системи класів

2.4 Розробка методів

2.5 Опис файлової структури програми

2.6 Тестування програми

3 Спеціальний розділ

3.1 Інструкція з інсталяції програмного забезпечення

3.2 Інструкція з використання тестових наборів

3.3 Інструкція з експлуатації програмного комплексу

4 Економічний розділ

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

4.3 Розрахунок витрат на електроенергію

4.4 Розрахунок суми амортизаційних відрахувань

4.5 Обчислення накладних витрат

4.6 Складання кошторису витрат та визначення собівартості НДР

4.7 Розрахунок ціни НДР

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

5 Охорона праці, техніка безпеки та екологічні вимоги

5.1 Функції служби охорони праці підприємства

5.2 Чинники, що впливають на тяжкість ураження людини електричним струмом

6 Висновки (навести результати роботи по кожному розділу зокрема і загальний висновок по кваліфікаційній роботі)

Додаткові вказівки:

Виконання роботи із розробкою програмного продукту……..

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, наприклад):

1. Структурна схема взаємодії компонентів програмного забезпечення
2. Блок-схема процесу подання заявки на вакансію
3. Діаграма класів застосунку
4. Таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	**Ім'я та прізвище, посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**
Економічний розділ	Любов КАЛУШКА		
Охорона праці, техніка безпеки та екологічні вимоги	Генадій ГОРЯЧЕК		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання		
2	Збір і узагальнення інформації по роботі		
3	Написання першого розділу		
4	Розробка технічного та робочого проєкту		
5	Написання спеціального розділу		
6	Розрахунок економічної частини		
7	Написання розділу охорони праці		
8	Виконання графічної частини		
9	Оформлення роботи		
10	Погодження нормоконтролю		
11	Попередній захист роботи		
12	Захист роботи		

7. Дата видачі завдання: _____.

Студент _____ Олександр КОЦУР
(підпис) (ім'я та прізвище)

Керівник роботи _____ Володимир СЕРБІН
(підпис) (ім'я та прізвище)

ЗМІСТ

Анотація 8
Abstract 9
Вступ 10
1 Загальний розділ 11
1.1 Аналітичний огляд існуючих рішень 11
1.1.1 Загальна характеристика інформаційних систем центрів зайнятості ... 11
1.1.2 Аналіз існуючих програмних рішень 12
1.1.3 Аналіз технологій та засобів розробки 14
1.1.4 Причини вибору власного рішення 15
1.2 Технічне завдання 17
1.2.1 Найменування та область застосування 17
1.2.2 Призначення розробки 18
1.2.3 Вимоги до програмного забезпечення 18
1.2.4 Вимоги до програмної документації 23
1.2.5 Техніко-економічні показники 24
1.2.6 Стадії та етапи розробки 25
2 Розробка технічного та робочого проєкту 27
2.1 Постановка задачі на розробку програмного забезпечення 27
2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних 28
2.3 Розробка алгоритму 30
2.3.1 Зовнішнє проєктування програми 30
2.3.2 Проєктування логіки програми 34
2.4 Визначення інформаційних зв'язків 37
2.5 Написання текстів програм 41

2.6 Тестування та налагодження програм 43
3 Спеціальний розділ 45
3.1 Інструкція з інсталяції програмного забезпечення 45
3.1.1 Конфігурація технічних засобів 45
3.1.2 Файли налаштування 47
3.1.3 Технологічний тип програм та послідовність інсталяції 48
3.2 Інструкція з використання тестових наборів 49
3.3 Інструкція з експлуатації програмного комплексу 51
4 Економічний розділ 57
4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР 57
4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи 58
4.3 Розрахунок витрат на електроенергію 60
4.4 Розрахунок суми амортизаційних відрахувань для розробки інформаційної системи Центру зайнятості 61
4.5 Обчислення накладних витрат 61
4.6 Складання кошторису витрат та визначення собівартості для розробки інформаційної системи Центру зайнятості 62
4.7 Розрахунок ціни для розробки інформаційної системи Центру зайнятості 63
4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень 63
5 Охорона праці, техніка безпеки та екологічні вимоги 65
5.1 Функції служби охорони праці підприємства. 65
5.2 Чинники, що впливають на тяжкість ураження людини елекричним струмом. 67
Висновки 69
Перелік посилань 70
Додатки 72
Додаток А Лістинг файлу «entity/Vacancy.java» 72

Додаток Б Лістинг файлу «entity/JobApplication.java» 74
Додаток В Лістинг файлу «security/JwtService.java» 77
Додаток Г Лістинг файлу «security/JwtAuthenticationFilter.java» 79
Додаток Д Лістинг файлу «dto/auth/UnemployedRegisterRequest.java» 82
Додаток Е Лістинг файлу «specification/VacancySpecification.java» 84
Додаток Ж Лістинг файлу «lib/stores/auth.ts» 86
Додаток И Лістинг файлу «lib/api/client.ts» 88
Додаток К Лістинг файлу «exception/GlobalExceptionHandler.java» 91
Додаток Л Лістинг файлу «security/LoginRateLimitFilter.java» 97

АНОТАЦІЯ

Тема кваліфікаційної роботи: Розробка інформаційної системи Центру зайнятості.

Метою кваліфікаційної роботи є розробка інформаційної системи Центру зайнятості.

Пояснювальна записка складається з п 'яти розділів.

У загальній частині описуються аналітичний огляд існуючих рішень та аналіз технічного завдання аналіз популярних підходів до реалізації інформаційної системи Центру зайнятості, а також постановка задачі та формування вимог до програмного забезпечення.

У другому розділі представлено процес створення програмного продукту, опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних, опис алгоритмів, інформаційних зв’язків, зовнішнє проектування програми, тестування та налагодження програм.

В спеціальній частині описані процес інсталяції програмного продукту, інструкція з використання тестових наборів та інструкція з користування програмою її інтерфейс та налаштування

Розрахунок вартості розробки та економічної ефективності приведено в економічній частини.

Основні питання охорони праці та техніки безпеки розглянуто в п’ятому розділі, а саме функції служби охорони праці та чинники, які впливають на тяжкість ураження людини електричним струмом.

Обсяг пояснювальної записки _____ сторінок.

До складу кваліфікаційної роботи входить графічна частина, яка складається з структурної схеми програми, блок-схеми процесу подання заявки, техніко-економічних показників та діаграми класів, що виконані на окремих аркушах формату А1.

ABSTRACT

Qualification work topic: Development of an Information System for the Employment Center.

The goal of this qualification work is to develop an information system for the Employment Center.

The explanatory note consists of five sections.

The general section describes an analytical review of existing solutions and analysis of the technical specification, an analysis of popular approaches to implementing an information system for the Employment Center, as well as the problem statement and formation of software requirements.

The second section presents the process of creating the software product, a description and justification of the chosen structure and method of organizing input and output data, a description of algorithms, information relationships, external program design, testing and debugging.

The special section describes the software product installation process, instructions for using test sets, and a user manual covering the program's interface and settings.

The calculation of development costs and economic efficiency is provided in the economic section.

The main issues of labor protection and safety are covered in the fifth section, namely the functions of the labor protection service and the factors affecting the severity of electric shock to a person.

The volume of the explanatory note is _____ pages.

The qualification work includes a graphical section consisting of a structural diagram of the program, a flowchart of the application submission process, technical and economic indicators, and a class diagram, all presented on separate A1 format sheets.

ВСТУП

В теперішніх умовах поширення інформаційних технологій, автоматизація напрямку державних установ та засобів працевлаштування набуло великої важливості. Особливу роль відіграють інформаційні системи, призначені для обробки, зберігання та аналізу даних. Однією з установ, діяльність якої потребує організації інформаційних процесів, є центр зайнятості. Використання сучасних програмних рішень дозволяє значно спростити роботу працівників, пришвидшити обслуговування громадян та підвищити якість надання послуг.

Актуальність теми полягає у необхідності створення зручної та ефективної інформаційної системи для автоматизації основних процесів центру зайнятості.

Метою даної дипломної роботи є розробка інформаційної системи Центру зайнятості, яка забезпечить автоматизацію процесів реєстрації безробітних, обліку вакансій, та пошуку роботи. Система повинна забезпечувати зручний інтерфейс користувача, надійне зберігання даних та ефективну взаємодію між різними модулями програмного забезпечення.

Для досягнення поставленої мети необхідно виконати аналіз предметної області, визначити функціональні вимоги до системи, спроєктувати структуру бази даних, реалізувати програмні модулі та провести тестування розробленого програмного продукту. Окрім цього, важливим етапом є забезпечення стабільної роботи системи та захисту інформації користувачів.

Об’єктом дослідження є процеси обліку та обробки інформації у центрі зайнятості. Предметом дослідження виступають методи та засоби розробки інформаційних систем. У процесі розробки можуть використовуватися сучасні технології програмування, системи керування базами даних та веб технології.

Практичне значення роботи полягає у можливості використання розробленої інформаційної системи для підвищення ефективності роботи центрів зайнятості. Запропоноване програмне рішення може бути адаптоване до потреб установ, що займаються обробкою даних про вакансії, роботодавців та пошукачів роботи.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

1.1.1 Загальна характеристика інформаційних систем центрів зайнятості

Центри зайнятості України щороку обробляють сотні тисяч звернень від громадян та роботодавців. При відсутності єдиної інформаційної системи більша частина цієї роботи виконується вручну або у розрізнених програмних засобах, що призводить до дублювання даних, затримок в обслуговуванні та збільшення ризику помилок. Діяльність таких установ охоплює широкий спектр операцій: реєстрацію безробітних, ведення бази вакансій, взаємодію з роботодавцями та формування статистичної звітності.

Інформаційні системи центрів зайнятості призначені для автоматизації основних бізнес-процесів установи. До основних функцій таких систем належать реєстрація та авторизація користувачів, ведення бази вакансій, пошук інформації про роботодавців і кандидатів, формування аналітичної звітності, а також забезпечення швидкого доступу до актуальних даних. У більшості випадків сучасні системи реалізуються у вигляді веб застосунків із використанням серверної архітектури та систем керування базами даних.

Однією з головних переваг сучасних інформаційних систем є централізоване зберігання інформації та можливість швидкого обміну даними між різними користувачами системи. Це дозволяє значно скоротити час обробки заявок, підвищити ефективність роботи працівників центру зайнятості та покращити якість обслуговування громадян. Автоматизація також дозволяє мінімізувати кількість помилок, пов’язаних із ручним введенням даних та паперовим документообігом.

Сучасні веб технології дозволяють забезпечити доступ до системи з будь-якого пристрою, який має доступ до мережі Інтернет. Це значно спрощує процес взаємодії між роботодавцями та пошукачами роботи, а також забезпечує

мобільність користувачів. Крім того, використання сучасних засобів захисту інформації, таких як JWT-автентифікація, шифрування персональних даних та розмежування прав доступу, дозволяє забезпечити належний рівень безпеки системи.

Разом із перевагами існують і певні недоліки. Частина існуючих систем має застарілий інтерфейс користувача, обмежений функціонал або складну навігацію. Деякі рішення недостатньо адаптовані для роботи з великою кількістю користувачів або не забезпечують належного рівня інтеграції між різними модулями системи. Саме тому виникає потреба у створенні сучасних інформаційних систем, які поєднуватимуть зручний інтерфейс, розширений функціонал, засоби аналітики та високий рівень безпеки.

1.1.2 Аналіз існуючих програмних рішень

На сьогоднішній день існує значна кількість програмних рішень для автоматизації процесів пошуку роботи та управління вакансіями. Найбільш поширеними в Україні є веб платформи Work.ua та Robota.ua, які забезпечують взаємодію між роботодавцями та пошукачами роботи через мережу Інтернет.

Сервіс Work.ua є одним із найбільших українських ресурсів для пошуку роботи. Під час особистого тестування платформи було перевірено процес реєстрації роботодавця, створення вакансії та пошуку кандидатів. Інтерфейс є зрозумілим, пошук вакансій за містом, заробітною платою та типом зайнятості працює швидко. Однак у ході перевірки виявлено, що для роботодавця відсутній будь-який журнал дій — неможливо переглянути, хто і коли вносив зміни до вакансії. Також не передбачено жодного внутрішнього звіту про результати підбору: скільки заявок надійшло, скільки відхилено, яка конверсія. Для комерційного сервісу це прийнятно, однак для державної установи, яка зобов’язана звітувати — критичний недолік.

Проте Work.ua орієнтований переважно на комерційний сектор та не забезпечує функціоналу, необхідного для роботи державних центрів зайнятості. У

системі відсутні модулі внутрішнього адміністрування, ведення офіційної звітності, аудиту дій користувачів та контролю працевлаштування безробітних відповідно до вимог державних установ.

Ще одним популярним рішенням є платформа Robota.ua. При тестуванні було перевірено пошук вакансій за фільтрами, перегляд профілю роботодавця та подання відгуку на вакансію. Сервіс має сучасний дизайн і добре реалізовані фільтри. Проте виявлено суттєве обмеження: кандидат не отримує жодного повідомлення у випадку, якщо роботодавець переглянув заявку але нічого не зробив — статус залишається “на розгляді” без змін невизначений час. Крім того, жодна з цих платформ не дозволяє офіційно зафіксувати факт працевлаштування всередині системи — для державного обліку це є принциповим недоліком.

Недоліком Robota.ua, як і більшості подібних платформ, є відсутність спеціалізованого функціоналу для державних центрів зайнятості. Система не підтримує повноцінне ведення внутрішньої документації, журналів аудиту, контролю статусів працевлаштування та формування офіційної статистичної звітності. Також частина функціоналу є платною, що може бути недоцільним для використання в державних установах.

Окрім комерційних платформ, використовуються також внутрішні інформаційні системи центрів зайнятості та кадрових служб. Такі рішення забезпечують ведення баз даних безробітних, облік звернень громадян, формування звітності та контроль працевлаштування. Перевагою подібних систем є адаптація до внутрішніх бізнес-процесів установи. Проте значна частина таких рішень створена на застарілих технологіях, має складний інтерфейс та обмежені можливості інтеграції із сучасними веб сервісами.

Проведений аналіз показує, що існуючі рішення мають переваги, однак не забезпечують повного набору функцій, необхідних для ефективної роботи. Саме тому доцільним є розроблення нової інформаційної системи, яка поєднуватиме сучасний веб інтерфейс, засоби адміністрування, систему безпеки, модулі аналітики та функціонал автоматизованого підбору вакансій і кандидатів.Аналіз існуючих програмних рішень

1.1.3 Аналіз технологій та засобів розробки

Вибір технологій безпосередньо визначає, наскільки система буде швидкою, підтримуваною та здатною до масштабування. У цьому розділі обґрунтовано вибір кожного інструменту — від мови програмування до фреймворку для фронтенду — з урахуванням конкретних вимог інформаційної системи Центру зайнятості. При виборі інструментів зверталася увага на швидкість обробки даних, надійність та зручність використання програмного забезпечення.

Для реалізації серверної частини системи доцільним є використання мови програмування Java та фреймворку Spring Boot. Даний фреймворк є одним із найбільш популярних засобів для створення веб застосунків та корпоративних інформаційних систем. Spring Boot забезпечує швидку розробку програмного забезпечення завдяки автоматичній конфігурації, підтримці REST API та інтеграції з технологіями для роботи з базами даних [1].

Основною перевагою використання Java та Spring Boot є висока продуктивність, стабільність роботи та підтримка багаторівневої архітектури застосунку. Завдяки цьому систему можна розділити на окремі компоненти, що покращує її підтримку, тестування та подальшу модернізацію. Крім цього, Spring Boot має вбудовані механізми забезпечення безпеки, обробки запитів та взаємодії між серверною частиною і клієнтським застосунком.

Для зберігання інформації використовується реляційна система керування базами даних PostgreSQL. Дана система забезпечує надійне зберігання даних, підтримує механізми резервного копіювання та дозволяє працювати з великими обсягами інформації. PostgreSQL характеризується стабільністю, підтримкою SQL-стандартів та можливостями для роботи зі складними структурами даних[2].

Для взаємодії між серверною частиною та базою даних у середовищі Spring Boot можуть використовуватися технології Spring Data JPA та Hibernate. Вони реалізують механізм об’єктно-реляційного відображення, що дозволяє працювати з даними у вигляді Java-об’єктів без необхідності постійного написання SQL-запитів.

Клієнтська частина інформаційної системи реалізовується за допомогою фреймворку Svelte. Даний фреймворк використовується для створення сучасних та швидких веб інтерфейсів. На відміну від багатьох інших JavaScript-фреймворків, Svelte компілює код застосунку під час збірки, що дозволяє отримати високу швидкодію та зменшити навантаження на браузер користувача[3].

Використання Svelte дозволяє створити зручний інтерфейс користувача для працівників центру зайнятості та відвідувачів системи. Завдяки компонентному підходу спрощується розробка окремих елементів інтерфейсу, а також забезпечується повторне використання компонентів у різних частинах застосунку. Крім цього, Svelte забезпечує швидке оновлення даних на сторінці без необхідності її перезавантаження, що покращує зручність роботи із системою[4].

Для обміну даними між клієнтською та серверною частинами використовується REST API. Такий підхід дозволяє забезпечити незалежність фронтенду та бекенду, спростити масштабування системи та забезпечити можливість подальшого розширення функціоналу[5].

Під час розробки інформаційної системи важливу роль також відіграють засоби тестування та проєктування програмного забезпечення. Для опису структури системи можуть використовуватися UML-діаграми, які дозволяють візуалізувати взаємодію компонентів програми. Тестування забезпечує перевірку правильності роботи функцій, виявлення помилок та підвищення стабільності програмного продукту.

1.1.4 Причини вибору власного рішення

Після проведення аналізу існуючих інформаційних систем та програмних рішень було встановлено, що більшість із них не повністю відповідають сучасним вимогам центрів зайнятості. Частина систем має обмежений функціонал, складний інтерфейс користувача або використовує застарілі технології, що ускладнює їх використання та подальшу підтримку.

Однією з головних причин вибору власного рішення є необхідність

створення системи, яка забезпечуватиме комплексну автоматизацію процесів центру зайнятості. Розроблювана система повинна поєднувати функції реєстрації користувачів, ведення бази вакансій, обліку роботодавців, формування заявок, пошуку роботи та створення звітності в єдиному програмному середовищі. Це дозволить значно спростити роботу працівників установи та підвищити швидкість обробки інформації.

Важливим фактором є також архітектурна гнучкість власного рішення. Готові комерційні платформи не дозволяють вільно змінювати логіку роботи, додавати специфічні перевірки або інтегрувати нові модулі без залежності від вендора. Власна розробка надає повний контроль над структурою даних, бізнес-логікою та можливістю подальшого розвитку.

Ще однією причиною вибору власної розробки є потреба у створенні сучасного та зрозумілого інтерфейсу користувача. Багато існуючих систем мають перевантажений дизайн, складну навігацію або застарілий зовнішній вигляд, що потребує додаткового навчання персоналу. Власне програмне рішення дозволяє створити інтерфейс, орієнтований на потреби користувачів, та забезпечити зручну взаємодію із системою.

Окрему увагу приділено питанням безпеки та захисту даних користувачів. Інформаційна система Центру зайнятості працює з персональними даними громадян та роботодавців, тому важливим є забезпечення належного рівня захисту інформації. Використання сучасних технологій розробки дозволяє реалізувати механізми авторизації, обмеження доступу та захисту даних від несанкціонованого використання.

Таким чином, розробка власної інформаційної системи є обґрунтованим рішенням, що дозволяє врахувати специфіку роботи державних центрів зайнятості, забезпечити відповідність вимогам безпеки та звітності, і при цьому залишити можливість для подальшого розвитку без залежності від сторонніх платформ.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

У рамках кваліфікаційної роботи виконується розробка інформаційної системи Центру зайнятості. Основною метою даної системи є автоматизація процесів обліку безробітних громадян, ведення бази вакансій, взаємодії з роботодавцями та формування необхідної звітності. Використання інформаційної системи дозволяє підвищити ефективність роботи установи, скоротити час обробки інформації та забезпечити швидкий доступ до необхідних даних.

Найменування проєкту: «Інформаційна система Центру зайнятості».

Область застосування інформаційної системи Центру зайнятості:

– Облік користувачів. Система дозволяє здійснювати реєстрацію та зберігання інформації про безробітних громадян, роботодавців та працівників центру зайнятості.

– Робота з вакансіями, таким чином забезпечується створення, редагування та пошук вакансій, а також можливість перегляду актуальних пропозицій працевлаштування.

– Аналітика та моніторинг. Програмний продукт забезпечує можливість аналізу інформації щодо кількості вакансій, працевлаштованих осіб та інших статистичних показників діяльності центру зайнятості.

– Веб доступ до системи. Використання веб технологій забезпечує доступ до інформаційної системи через браузер без необхідності встановлення додаткового програмного забезпечення.

Розроблювана інформаційна система орієнтована на підвищення ефективності взаємодії між пошукачами роботи, роботодавцями та працівниками центру зайнятості. Використання сучасних веб технологій, централізованої бази даних та засобів автоматизації дозволяє забезпечити зручність користування системою, швидке опрацювання інформації та можливість подальшого розширення функціональних можливостей програмного забезпечення.

1.2.2 Призначення розробки

Інформаційна система Центру зайнятості призначена для автоматизації основних процесів роботи установи, вакансії, роботодавців та працевлаштування. Використання системи дозволяє спростити виконання рутинних операцій, підвищити швидкість обробки даних та забезпечити ефективну взаємодію між працівниками центру зайнятості та його користувачами.

Експлуатаційне призначення програмного виробу полягає у використанні системи працівниками центру зайнятості для ведення електронного обліку інформації, формування звітності, пошуку вакансій та адміністрування даних. Система може використовуватися як внутрішній веб застосунок установи, доступ до якого здійснюється через браузер із персонального комп'ютера або іншого пристрою, підключеного до мережі Інтернет чи локальної мережі.

Функціональне призначення полягає у наступних процесах:

- реєстрація та облік користувачів системи;
- ведення бази вакансій та роботодавців;
- пошук вакансій відповідно до заданих параметрів;
- зберігання та обробка інформації про працевлаштування;
- адміністрування системи та керування доступом користувачів.

Розроблена система також забезпечує централізоване зберігання інформації та швидкий доступ до даних, що дозволяє зменшити ймовірність помилок та покращити якість обслуговування громадян. Використання сучасних веб технологій забезпечує зручний інтерфейс користувача, швидку взаємодію із системою та можливість подальшого розширення функціональних можливостей програмного продукту.

1.2.3 Вимоги до програмного забезпечення

Інформаційна система Центру зайнятості призначена для обробки даних про безробітних громадян, роботодавців, вакансії, заявки на працевлаштування та

працівників установи. Вхідні дані надходять до системи через користувацький веб інтерфейс та зберігаються у базі даних PostgreSQL.

Основними джерелами вхідної інформації є працівники центру зайнятості, роботодавці та зареєстровані користувачі системи. Дані вводяться вручну через форми веб застосунку або створюються під час виконання окремих операцій системи. Зразок повідомлень їх опис та їх перелік подано на таблиці 1.1 зображеній нижче.

Таблиця 1.1 – Перелік і опис вхідних повідомлень

Назва вхідного повідомлення	**Форма подання**	**Періодичність надходження**	**Джерело даних**
Дані безробітного	Електронна форма	За потребою	Працівник центру
Дані роботодавця	Електронна форма	За потребою	Роботодавець
Інформація про вакансію	Електронна форма	За потребою	Роботодавець
Заявка на вакансію	Електронна форма	За потребою	Користувач системи
Дані працевлаштування	Електронна форма	За потребою	Працівник центру
Дані працівників	Електронна форма	За потребою	Адміністратор

До основних вхідних даних належать:

- прізвище, ім’я та по батькові користувача;
- номер паспорта;
- адреса проживання;
- контактні дані;
- інформація про освіту та досвід роботи;
- дані про вакансії;
- інформація про роботодавців;
- статус заявок та працевлаштування.Вихідною інформацією системи є

результати обробки даних, сформовані звіти, списки вакансій, інформація про безробітних та статистичні показники діяльності центру зайнятості.

Вихідні дані можуть відображатися на екрані. Також передбачена можливість збереження та друку окремих звітів. Перелік вихідних повідомлень подано на таблиці 1.2

Таблиця 1.2 – Перелік і опис вихідних повідомлень

Назва вихідного повідомлення	**Форма подання**	**Періодичність формування**	**Користувач**
Список вакансій	Таблиця	За запитом	Користувач
Інформація про безробітних	Таблиця	За запитом	Працівник центру
Звіт про працевлаштування	Звіт	Щомісячно	Адміністрація
Дані роботодавців	Таблиця	За запитом	Працівник центру
Статистичні показники	Звіт	За запитом	Адміністрація

Вихідна інформація використовується для аналізу роботи інформаційної системи центру зайнятості, також для контролю вакансій та формування статистичних даних.

Основною функцією інформаційної системи є автоматизація роботи Центру зайнятості. Система забезпечує зберігання інформації роботодавців та шукачів роботи, обробку та пошук інформації про безробітних громадян, вакансії та роботодавців.

До основних функцій системи належать:

- реєстрація та авторизація користувачів;
- додавання, редагування та видалення вакансій;
- ведення обліку безробітних;
- ведення обліку роботодавців;

– створення заявок на вакансії;
– облік працевлаштування;
– формування звітності;
– адміністрування системи.

Під час роботи системи здійснюється обробка даних, що надходять від користувачів через вебінтерфейс. Наприклад, під час реєстрації безробітного користувача вводяться персональні дані, які після перевірки зберігаються у таблиці unemployed.

Система має такі обмеження:

– користувач не може створити запис без заповнення обов’язкових полів;
– номер паспорта повинен бути унікальним;
– доступ до окремих функцій обмежується роллю користувача;
– видалення даних виконується лише авторизованими користувачами.

Для забезпечення коректної роботи виконується перевірка введених даних, контроль форматів електронної пошти, номерів телефонів та унікальності записів у базі даних. Інформаційна система повинна забезпечувати швидку обробку запитів користувачів та стабільну роботу при одночасній роботі декількох користувачів Час відкриття сторінок системи не повинен перевищувати 3 секунд при стандартному навантаженні. Час виконання пошуку вакансій або користувачів повинен становити не більше 5 секунд. Операції додавання, редагування та видалення інформації повинні виконуватися у режимі реального часу без необхідності тривалого очікування користувача.

Для забезпечення надійної роботи інформаційної системи передбачено:

– багаторівневу архітектуру програмного забезпечення;
– контроль правильності введення даних;
– обробку виняткових ситуацій;
– захист від несанкціонованого доступу;
– резервне збереження даних;
– контроль цілісності бази даних.

Система повинна виконувати перевірку коректності введених значень перед збереженням у базі даних. У разі виникнення помилки користувач повинен отримувати зрозуміле повідомлення із поясненням причини помилки.

Для захисту даних використовується механізм авторизації та розмежування прав доступу. Доступ до адміністративних функцій мають лише авторизовані користувачі з відповідними правами.

Інформаційна система Центру зайнятості реалізована як вебзастосунок та може використовуватися через сучасні веббраузери. Для роботи із системою необхідний доступ до локальної мережі або мережі Інтернет.

Основними режимами роботи системи є:

- режим користувача;
- режим працівника центру зайнятості;
- режим адміністратора.

Система забезпечує можливість перегляду інформації, пошуку вакансій, роботи з базою даних, формування звітності та адміністрування користувачів. Користувачі системи можуть виконувати основні операції через вебінтерфейс, отримуючи швидкий доступ до необхідної інформації та функцій програмного забезпечення.

Для ефективної роботи інформаційної системи рекомендується використання персонального комп’ютера з операційною системою Windows 10 або Linux, процесором рівня Intel Core i3 або вище та оперативною пам’яттю обсягом не менше 4 ГБ. Робота із системою здійснюється через сучасні веббраузери, зокрема Google Chrome, Mozilla Firefox або Microsoft Edge. Для функціонування серверної частини застосунку використовується середовище Java Runtime Environment, сервер застосунків Spring Boot та система керування базами даних PostgreSQL.

Програмне забезпечення повинно бути сумісним із сучасними версіями операційних систем та веббраузерів, забезпечувати стабільну роботу клієнтської і серверної частин застосунку, а також підтримувати можливість подальшого оновлення та розширення функціональних можливостей системи.

1.2.4 Вимоги до програмної документації

Програмна документація інформаційної системи Центру зайнятості повинна забезпечувати можливість ефективного використання, супроводу та подальшої модернізації програмного продукту. Документація створюється відповідно до вимог державних стандартів та повинна містити необхідну інформацію для користувачів, адміністраторів і розробників системи.

До складу програмної документації входять наступні пункти:

- пояснювальна записка
- технічне завдання;
- опис структури програмного забезпечення;
- опис бази даних;
- інструкція користувача;
- інструкція адміністратора;
- текст програмного коду;
- результати тестування програмного забезпечення.

Вимоги до оформлення пояснювальної записки та її вмісту взято у методичних вказівках[16].

Технічна документація повинна містити опис призначення системи, її функціональних можливостей, структури програмних модулів та принципів взаємодії між компонентами системи. Інструкція користувача повинна описувати порядок роботи із системою, основні функції інтерфейсу та правила використання програмного продукту. Інструкція адміністратора повинна містити інформацію щодо налаштування системи, керування користувачами та обслуговування бази даних.

Текст програмного коду повинен бути структурованим, зрозумілим та відповідати сучасним вимогам програмування. Коментарі у програмному коді повинні бути стислими, зрозумілими та відображати основне призначення окремих частин програми. Їх використання необхідне для спрощення супроводу програмного забезпечення та полегшення процесу подальшої модернізації системи.

1.2.5 Техніко-економічні показники

Розробка інформаційної системи Центру зайнятості потребує використання трудових, технічних та програмних ресурсів. Основними витратами під час створення програмного продукту є витрати на оплату праці розробника, використання комп'ютерної техніки, електроенергії та програмного забезпечення, необхідного для реалізації та тестування системи. У процесі розробки також враховуються витрати, пов'язані з підтримкою робочого середовища та використанням необхідних інструментів програмування.

Трудові ресурси включають виконання аналізу предметної області, проєктування структури бази даних, реалізацію серверної та клієнтської частин застосунку, тестування програмного забезпечення та підготовку програмної документації. Значна частина часу витрачається на програмування та перевірку коректності роботи системи, оскільки інформаційна система повинна забезпечувати стабільну роботу з великим обсягом даних та підтримувати одночасну роботу декількох користувачів. Для реалізації програмного продукту використовуються сучасні технології розробки, зокрема Java, Spring Boot, PostgreSQL та Svelte. Використання цих технологій дозволяє створити масштабовану, стабільну та продуктивну систему, яка може використовуватися у реальних умовах роботи центру зайнятості. Крім цього, застосування сучасної вебархітектури забезпечує можливість подальшого розвитку програмного забезпечення та інтеграції додаткових модулів у майбутньому.

Під час оцінювання техніко-економічних показників враховуються витрати на оплату праці, які становлять основну частину собівартості програмного продукту. Це пояснюється тим, що процес створення інформаційної системи потребує значного обсягу інтелектуальної праці, пов'язаної з аналізом вимог, програмуванням та тестуванням системи. Також враховуються витрати на електроенергію, амортизацію обладнання та накладні витрати, необхідні для забезпечення процесу розробки.

Очікується, що впровадження інформаційної системи дозволить значно

підвищити ефективність роботи центру зайнятості. Автоматизація процесів обліку та пошуку інформації сприятиме скороченню часу обробки даних, зменшенню кількості помилок та покращенню якості обслуговування громадян. Крім цього, система дозволить оптимізувати процес формування звітності та забезпечить швидкий доступ до необхідної інформації.

Важливим економічним показником є прогнозований термін окупності програмного продукту. Очікується, що витрати на розробку та впровадження інформаційної системи можуть окупитися приблизно протягом двох – двох з половиною років експлуатації. Це свідчить про економічну доцільність створення програмного забезпечення та перспективність його використання у діяльності центру зайнятості.

Також очікується отримання позитивного економічного ефекту за рахунок зменшення витрат часу працівників на виконання рутинних операцій та скорочення обсягів паперової документації. Використання інформаційної системи дозволить централізувати роботу з даними та забезпечити більш ефективне використання ресурсів установи.

Зведені дані та очікування від економічної частини наведені в таблиці економічних показників у графічній частині 2026.КВР.122.421.11.00.00 ТБ.

1.2.6 Стадії та етапи розробки

Розробка інформаційної системи Центру зайнятості виконується поетапно відповідно до основних стадій створення програмного забезпечення. Кожен етап розробки включає виконання окремих робіт, пов’язаних із аналізом вимог, проєктуванням, програмною реалізацією, тестуванням та впровадженням системи.

Першим виконується аналіз предметної області та формування технічного завдання. Проводиться дослідження особливостей роботи центрів зайнятості, аналіз існуючих програмних рішень та визначення основних функціональних вимог до системи. На основі отриманої інформації формується технічне завдання, у якому визначаються цілі, функції та вимоги до програмного продукту.

Наступним етапом є проєктування системи. На даній стадії розробляється структура програмного забезпечення, визначаються основні модулі системи та проєктується база даних. Також створюються UML-діаграми, схеми взаємодії компонентів та структура користувацького інтерфейсу.

Після завершення проєктування виконується програмна реалізація системи. Тут здійснюється розробка серверної частини, клієнтської частини, а також інтеграція системи з базою даних PostgreSQL. Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

– діючу програму, яка повністю відповідає даному технічному завданню;

– вихідний програмний код, записаний разом із програмою на оптичний носій інформації.

– Прийом програмного забезпечення повинен відбуватися перед комісією у такій послідовності:

– доповідь Виконавця про виконану роботу;

– демонстрація Виконавцем роботи програми;

– контрольні випробовування роботи програми;

– відповіді на запитання і зауваження комісії.

Після завершення випробовувань комісія повинна скласти відповідний акт прийому-передачі програмного забезпечення, у якому фіксуються результати перевірки, відповідність програмного продукту вимогам технічного завдання та можливі зауваження або рекомендації щодо подальшої експлуатації системи.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Постановка задачі на розробку програмного забезпечення

Основною задачею кваліфікаційної роботи є розробка інформаційної системи Центру зайнятості, яка забезпечить автоматизацію процесів обліку користувачів, вакансій, роботодавців та працевлаштування громадян. Система повинна забезпечувати швидку обробку інформації, зручний доступ до даних та можливість централізованого керування всіма основними процесами установи.

У процесі роботи центрів зайнятості виникає необхідність постійної обробки великого обсягу інформації. До такої інформації належать дані про безробітних громадян, вакансії, роботодавців, заявки на працевлаштування та статистичні показники роботи установи. Використання паперової документації або застарілих програмних рішень значно ускладнює процес обробки даних та збільшує ризик виникнення помилок. Саме тому виникає потреба у створенні сучасної інформаційної системи, яка дозволить автоматизувати основні процеси роботи центру зайнятості.

Розроблювана система повинна забезпечувати:

- реєстрацію та авторизацію користувачів;
- ведення бази вакансій;
- облік роботодавців та пошукачів роботи;
- пошук вакансій за визначеними параметрами;
- формування статистичних звітів;
- збереження та обробку інформації у базі даних.

Для реалізації серверної логіки використовується Java та Spring Boot, а клієнтська частина реалізується за допомогою фреймворку Svelte. Обмін даними між компонентами системи здійснюється через REST API. Такий підхід дозволяє забезпечити масштабованість системи та можливість подальшого розвитку програмного продукту.

Особливістю реалізації даного класу задач на ЕОМ є необхідність

забезпечення швидкої взаємодії між клієнтською та серверною частинами системи, а також надійного зберігання даних. Для цього використовується багаторівнева архітектура програмного забезпечення, яка включає клієнтську частину, серверний застосунок та систему керування базами даних PostgreSQL.

2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Для забезпечення ефективної роботи інформаційної системи Центру зайнятості було обрано централізований метод організації вхідних та вихідних даних із використанням реляційної бази даних PostgreSQL та REST-архітектури взаємодії між клієнтською і серверною частинами застосунку. Такий підхід дозволяє забезпечити швидку обробку інформації, надійне зберігання даних та можливість масштабування системи.

Вхідні дані надходять до системи через вебінтерфейс, реалізований за допомогою фреймворку Svelte. Передача інформації між клієнтською та серверною частинами виконується через REST API у форматі JSON. Для обробки запитів використовується серверна частина, реалізована на основі Java та Spring Boot. Збереження та отримання даних виконується за допомогою ORM-технології Spring Data JPA [6] та репозиторіїв Hibernate [7].

Основними джерелами вхідної інформації є користувачі системи, а саме адміністратори, роботодавці та безробітні громадяни. Кожна категорія користувачів має власний набір функціональних можливостей та доступ до відповідних форм введення інформації.

У системі реалізовано декілька основних форм введення даних. Під час реєстрації користувача використовується двокрокова форма, у якій спочатку обирається роль користувача, після чого вводяться персональні дані. Для роботодавця вводяться назва компанії, галузь діяльності, контактна інформація та ідентифікаційні дані. Для безробітного користувача передбачене введення паспортних даних, адреси проживання, освіти та досвіду роботи.

Окремі форми використовуються для авторизації користувача, відновлення пароля, створення вакансій, редагування профілю безробітного та подання заявки на вакансію. Усі дані перед збереженням проходять перевірку коректності введення та валідацію на серверній стороні.

Вихідна інформація формується у вигляді таблиць, списків та звітів, які відображаються у вебінтерфейсі користувача. Система забезпечує виведення списку вакансій, роботодавців, безробітних громадян, заявок на працевлаштування та статистичної інформації. Для адміністратора реалізовано окремий модуль звітності та дашборд для аналізу роботи системи.

Для підвищення ефективності роботи із даними використовується механізм сортування, пошуку та пагінації. Наприклад, вакансії можуть фільтруватися за посадою, рівнем заробітної плати, містом, галуззю або статусом. Дані про безробітних можуть сортуватися за освітою, навичками, статусом або бажаною заробітною платою. Для всіх основних списків використовується пагінація із фіксованим розміром сторінки.

Організація даних у системі базується на використанні реляційної структури бази даних. Основними сутностями є користувачі, роботодавці, безробітні, вакансії, заявки на працевлаштування та записи про працевлаштування. Між сутностями реалізовано зв’язки типу «один до одного» та «один до багатьох». Наприклад, один роботодавець може мати декілька вакансій, а один безробітний користувач може подавати заявки на різні вакансії.

Для забезпечення безпеки доступу до інформації використовується JWT-автентифікація. Після успішної авторизації користувач отримує токен доступу, який використовується під час подальших запитів. Роль користувача зберігається у claims токена та використовується для контролю доступу до функцій системи.

У процесі проєктування структури даних було обрано багаторівневу архітектуру від Controller до Service, від Service до Repository, яка дозволяє розділити логіку обробки даних, бізнес-логіку та роботу з базою даних. Такий підхід забезпечує кращу підтримку програмного забезпечення, спрощує тестування та дозволяє реалізувати масштабовану структуру програмного комплексу.

2.3 Розробка алгоритму

2.3.1 Зовнішнє проєктування програми

Зовнішнє проєктування інформаційної системи Центру зайнятості передбачає визначення основних функцій програмного комплексу, структури взаємодії користувача із системою, а також розподіл функціональних можливостей відповідно до ролей користувачів. Основною метою зовнішнього проєктування є формування зручного, зрозумілого та функціонального інтерфейсу, який забезпечує ефективну взаємодію користувачів із системою.

Інформаційна система реалізована у вигляді вебзастосунку із багаторівневою структурою. Користувач взаємодіє із системою через браузерний інтерфейс, а обробка даних виконується серверною частиною застосунку. Основні функції програмного комплексу згруповані у вигляді окремих функціональних модулів.

На верхньому рівні ієрархії функцій програмного комплексу виділено модуль авторизації, модуль роботи з вакансіями, модуль заявок, модуль роботи із безробітними, модуль роботодавців, модуль працевлаштування, модуль персоналу, модуль статистики та звітності, модуль аудиту та модуль сповіщень. Основною функцією системи є забезпечення автоматизації процесів взаємодії між центром зайнятості, роботодавцями та безробітними громадянами. Для цього система підтримує реєстрацію користувачів, авторизацію, пошук вакансій, подання заявок, формування записів про працевлаштування та створення статистичної звітності.

Модуль авторизації забезпечує реєстрацію користувачів, вхід до системи та відновлення пароля. Під час реєстрації користувач обирає тип облікового запису – роботодавець або безробітний – після чого вводить необхідні персональні дані. Для забезпечення безпеки доступу використовується JWT-автентифікація[8]. Інтерфейс першого кроку сторінки авторизації подано на рисунку 2.1 нижче.

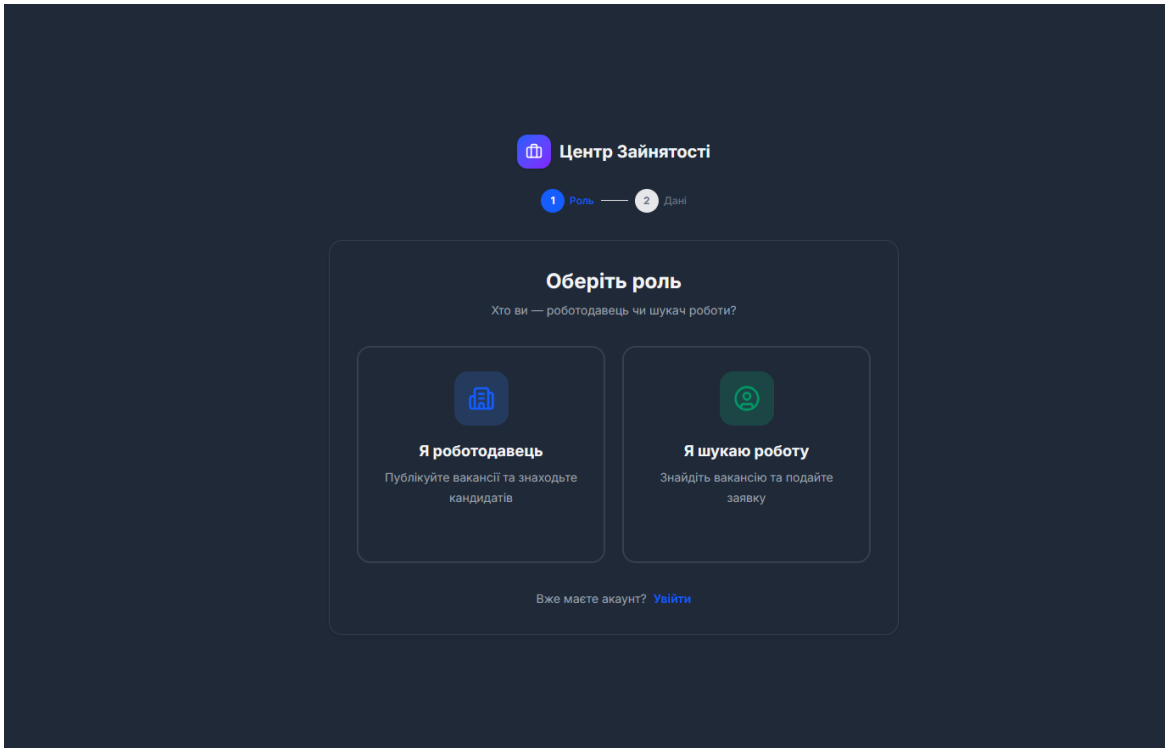


Рисунок 2.1 – Інтерфейс сторінки авторизації крок 1

Другий крок, аналогічний для обох ролей подано на рисунку 2.2.

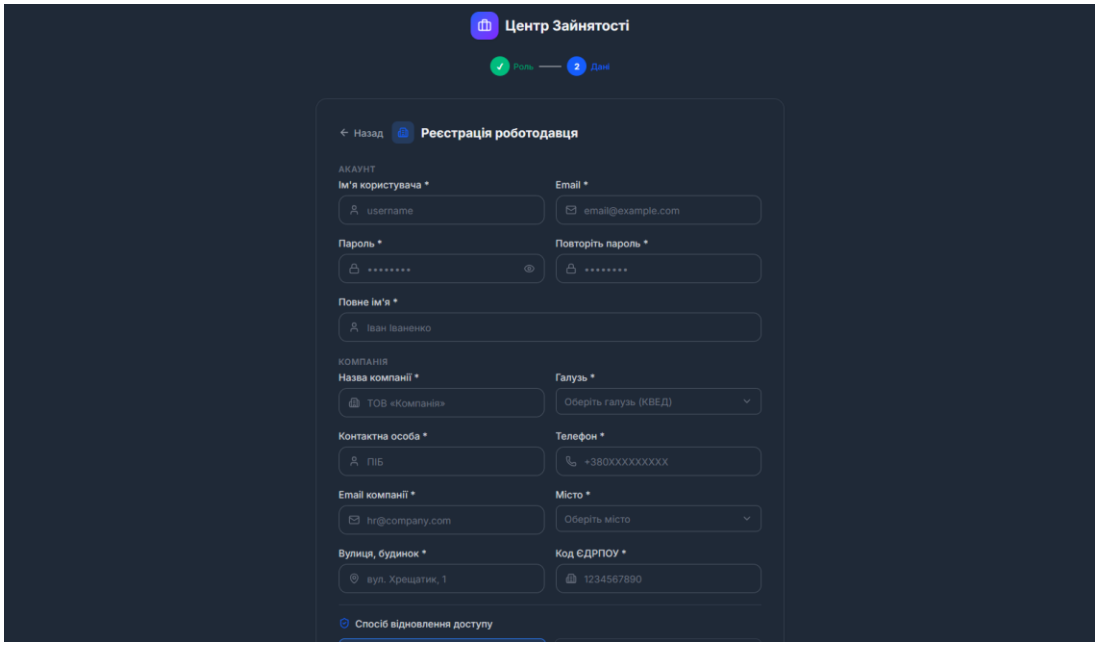


Рисунок 2.2 – Інтерфейс сторінки авторизації крок 2

Модуль вакансій забезпечує створення, редагування, перегляд та видалення вакансій. Роботодавці мають можливість керувати лише власними вакансіями, тоді

як адміністратор може виконувати повний контроль усіх записів. Система підтримує пошук вакансій за назвою посади, рівнем заробітної плати, містом, статусом та галуззю діяльності. Інтерфейс сторінки створення вакансії подано на рисунку 2.3.

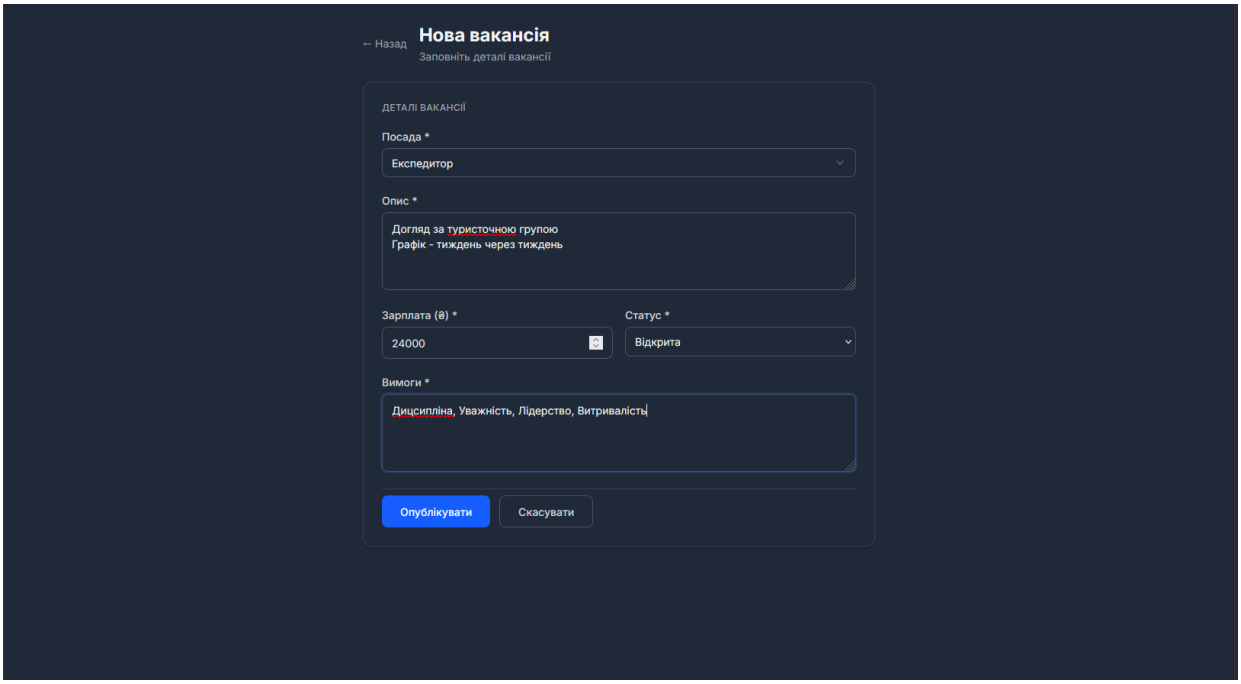


Рисунок 2.3 – Інтерфейс сторінки призначеної для створення нової вакансії роботодавцем

Для безробітних користувачів реалізовано можливість перегляду списку вакансій, використання фільтрів та подання заявки на вакансію. Під час подання заявки користувач може додати супровідний лист та переглянути інформацію про роботодавця. Роботодавець має можливість переглядати кандидатів, змінювати статус заявки, формувати пропозицію роботи або відхиляти кандидата. Безробітний користувач може переглядати власні заявки, відкликати їх або приймати пропозицію роботодавця. Інтерфейс сторінки перегляду вакансії та подання заявки показано на рисунку 2.4.

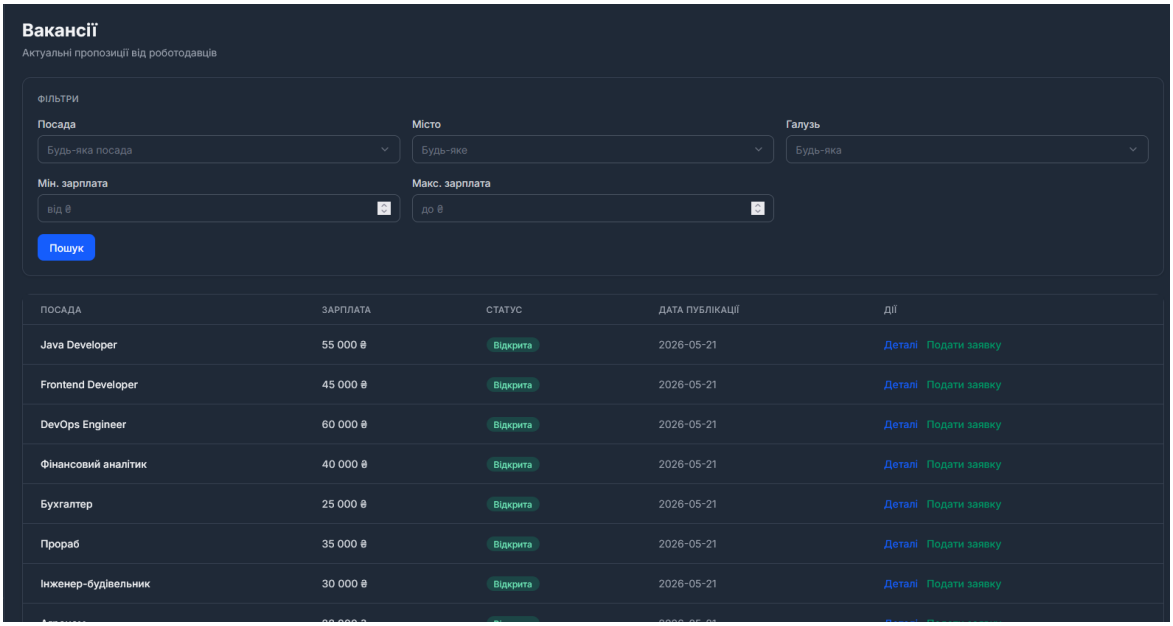


Рисунок 2.4 – Інтерфейс перегляду вакансій безробітнім

Модуль безробітних забезпечує ведення профілів користувачів, редагування особистої інформації, перегляд навичок, освіти та бажаного рівня заробітної плати. Система також підтримує механізм підбору рекомендованих вакансій на основі характеристик профілю користувача. Зображення сторінки профілю безробітного подано на рисунку 2.5.

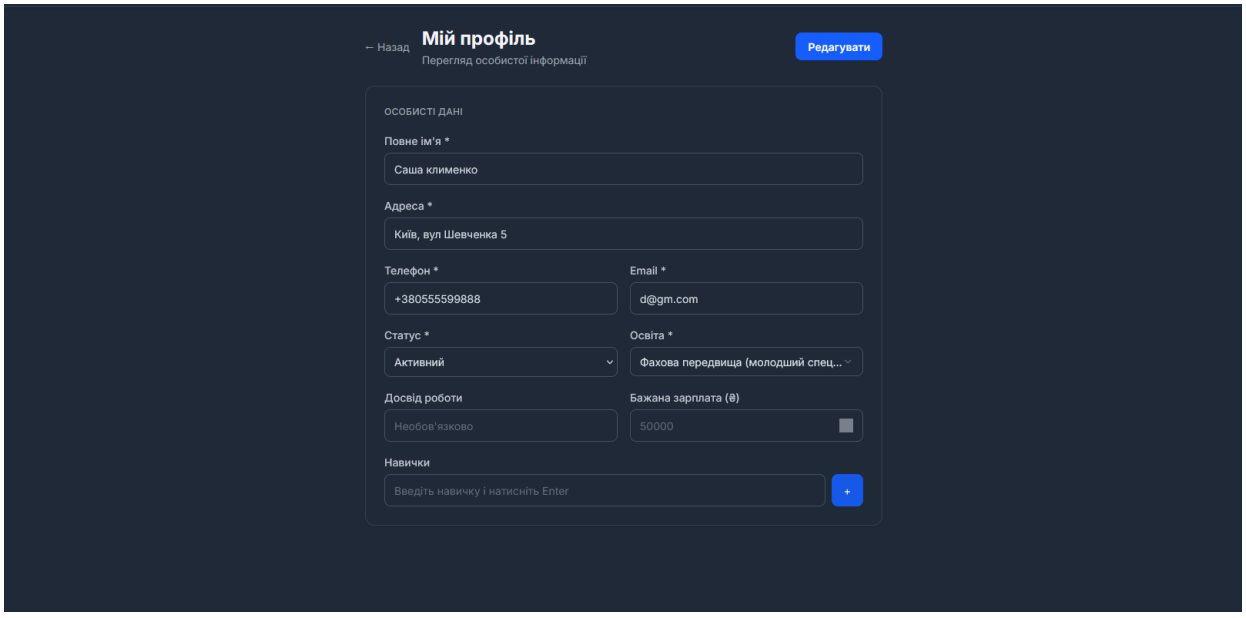


Рисунок 2.5 – Інтерфейс профілю безробітнього

Модуль роботодавців забезпечує керування профілем компанії, перегляд кандидатів та роботу із записами про працевлаштування. Роботодавець може формувати список потенційних кандидатів та переглядати інформацію про їх навички, освіту та досвід роботи.

Модуль працевлаштування використовується для ведення записів про прийняття на роботу та звільнення працівників. У системі підтримується можливість збереження історії працевлаштувань, інформації про заробітну плату та причин звільнення.

Окремо реалізовано модуль сповіщень, який забезпечує інформування користувачів про зміни статусу заявки, нові вакансії або результати розгляду кандидатури. Сповіщення можуть відображатися у вебінтерфейсі після авторизації користувача.

Усі функціональні можливості системи розподіляються відповідно до ролей користувачів. Адміністратор має повний доступ до всіх функцій програмного комплексу. Роботодавці працюють із вакансіями, кандидатами та працевлаштуванням, а безробітні користувачі можуть переглядати вакансії, подавати заявки та редагувати власний профіль.

2.3.2 Проєктування логіки програми

Проєктування логіки інформаційної системи Центру зайнятості виконувалося з урахуванням необхідності забезпечення стабільної роботи, контролю доступу до функцій системи та обробки великої кількості користувацьких запитів. Логіка системи реалізована на основі багаторівневої архітектури із розподілом відповідальності між клієнтською та серверною частинами застосунку.

Основою взаємодії користувача із системою є вебінтерфейс, реалізований із використанням фреймворку SvelteKit. Передача даних між клієнтом та сервером здійснюється через REST API із використанням HTTP-запитів у форматі JSON. Серверна частина реалізована за допомогою Spring Boot та Spring

Security.Структурна схема взаємодії компонентів програмного комплексу подана на графічній частині 2026.КВР.122.421.11.00.00 СС.

Одним із сценаріїв роботи системи є процес авторизації користувача. Під час входу до системи користувач вводить логін та пароль на сторінці авторизації. Дані передаються на серверну частину через POST-запит до API. Після перевірки облікових даних формується JWT-токен, який містить інформацію про роль користувача та використовується для подальшої авторизації запитів.

Після успішної авторизації JWT-токен зберігається у клієнтській частині застосунку та автоматично додається до всіх наступних HTTP-запитів. Під час обробки запиту серверна частина перевіряє дійсність токена, визначає роль користувача та надає доступ лише до дозволених функцій системи.

Процес реєстрації користувача реалізований у вигляді форми. Користувач обирає тип облікового запису, вводить персональні дані. Під час реєстрації виконується перевірка унікальності логіна та електронної пошти.

Для безробітних користувачів передбачено додатковий захист персональних даних. Паспортна інформація проходить процедуру шифрування. Також реалізовано механізм відновлення доступу до облікового запису.

Одним із процесів системи є створення вакансії. Після заповнення форми дані надходять на сервер, там виконується перевірка ролі користувача та його прав доступу. Після успішної перевірки вакансія зберігається у базі даних. Для пошуку вакансій система підтримує механізми фільтрації та сортування. Для зменшення навантаження на систему використовується механізм пагінації даних.

Подання заявки на вакансію є одним із найбільш складних процесів програмного комплексу. Перед створенням заявки система виконує перевірку статусів та дублюючих заявок. Після проходження перевірок заявка створюється зі статусом pending, а роботодавець отримує відповідне сповіщення. Блок-схему процесу подання заявки на вакансію подано у графічній частині 2026.КВР.122.421.11.00.00 БС

У процесі розгляду заявки роботодавець може змінювати її статус відповідно до визначеної логіки переходів між станами. Для кожного статусу визначено

допустимі переходи, що дозволяє уникнути помилок у роботі системи.

У разі прийняття заявки система автоматично створює запис про працевлаштування. Одночасно змінюється статус безробітного користувача на робітника, а вакансія переводиться у статус закритої. Усі операції виконуються в межах однієї транзакції, що забезпечує цілісність даних.

Процес звільнення працівника також реалізовано у вигляді окремого сценарію. Роботодавець або адміністратор можуть завершити працевлаштування користувача із зазначенням причини звільнення та дати завершення роботи. Безробітний користувач також має можливість самостійно завершити поточне працевлаштування через функцію resign.

Для забезпечення коректності введених даних у системі використовується механізм Bean Validation. Для основних полів визначені обмеження щодо довжини тексту, формату електронної пошти, номера телефону, ідентифікаційного номера та дат. Наприклад, дата подання заявки або не може бути більшою за поточну дату.

Крім стандартної валідації, у системі реалізовано додаткові бізнес-перевірки. Зокрема, перевіряється унікальність користувачів, допустимість переходів між статусами заявок, доступ користувача лише до власних записів та можливість створення заявки лише на активну вакансію.

Логіка роботи серверної частини побудована за принципом багаторівневої архітектури. REST-контролери відповідають за прийом HTTP-запитів та обробку JSON-даних. Шар сервісів реалізує бізнес-логіку, репозиторії забезпечують взаємодію із базою даних за допомогою технології Spring та Hibernate.

Для забезпечення безпеки програмного комплексу використовується Spring Security. Контроль доступу до функцій системи виконується за допомогою анотацій, які перевіряють роль користувача та його права.

У системі також реалізовано додаткові службові модулі. AuditLogService автоматично фіксує всі основні дії користувачів, пов’язані зі створенням, редагуванням або видаленням даних. NotificationService забезпечує формування сповіщень про зміну статусу заявки або інші важливі події системи.

Для реалізації функції підбору вакансій використовується окремий

MatchingService, який виконує аналіз навичок користувача, освіти та досвіду роботи. Результати підбору можуть кешуватися для підвищення продуктивності роботи системи. Додатково система підтримує використання AI-модуля для формування пояснення відповідності вакансії профілю користувача.

Для взаємодії клієнтської та серверної частин використовуються HTTP-методи GET, POST, PUT, PATCH та DELETE. Метод GET використовується для отримання інформації, POST – для створення нових записів та виконання окремих дій, PUT – для повного оновлення інформації, PATCH – для часткових змін, а DELETE – для видалення даних.

Основні функції програмного комплексу реалізовані у вигляді REST API endpoint-ів. Кожен endpoint має обмеження доступу відповідно до ролі користувача та виконує окрему функцію системи. Такий підхід забезпечує модульність програмного комплексу, спрощує підтримку системи та дозволяє масштабувати програмне забезпечення у майбутньому.

2.4 Визначення інформаційних зв’язків

У процесі розробки інформаційної системи Центру зайнятості було визначено взаємозв’язки між програмними компонентами, які забезпечують узгоджену роботу всіх функціональних модулів системи. Архітектура програмного забезпечення побудована за багаторівневим принципом із розділенням логіки представлення, бізнес-логіки та роботи з даними.

Програмний комплекс реалізовано у вигляді клієнт-серверного вебзастосунку. Взаємодія між клієнтом та сервером здійснюється через REST API із передачею даних у форматі JSON за протоколом HTTP/HTTPS.

Для організації інформаційних зв’язків між компонентами системи використовується багаторівнева архітектура, у якій кожен рівень виконує окремі функції. Клієнтський рівень відповідає за інтерфейс користувача та обробку взаємодії із системою. Серверний рівень забезпечує виконання бізнес-логіки, перевірку прав доступу, обробку запитів та взаємодію з базою даних. Рівень

збереження даних реалізований за допомогою PostgreSQL та Hibernate ORM. Структурна схема взаємодії компонентів програмного забезпечення подана нижче (див. рис.2.6).

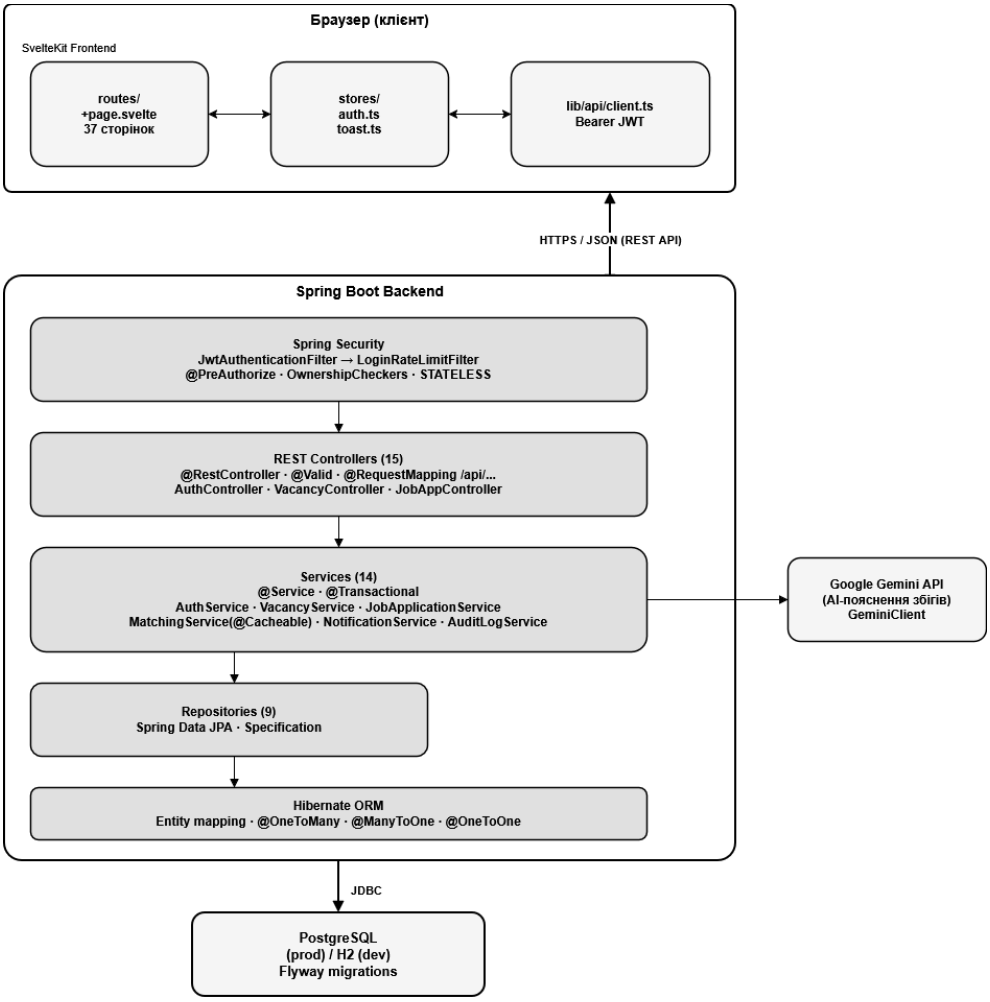


Рисунок 2.6 – схема взаємодії компонентів.

У серверній частині програмного забезпечення реалізовано окремі функціональні модулі, кожен із яких відповідає за певну групу задач. Для організації REST API використовуються контролери, що приймають HTTP-запити від клієнтської частини. Контролери передають дані до сервісного шару, де виконується основна бізнес-логіка. Після обробки інформації сервіси взаємодіють із репозиторіями, які забезпечують доступ до бази даних через Spring Data JPA.

Основним компонентом системи безпеки є модуль Spring Security. Він забезпечує авторизацію та автентифікацію користувачів на основі JWT-токенів.

Після успішного входу користувача система генерує токен доступу, який надалі передається у заголовку Authorization під час кожного HTTP-запиту. Обробка токенів здійснюється за допомогою компонента JwtAuthenticationFilter, який перевіряє валідність токена та заповнює SecurityContext даними авторизованого користувача[9].

Додатково у системі реалізовано механізм перевірки прав доступу на рівні окремих функцій. Для цього використовуються анотації, які визначають доступ до методів залежно від ролі користувача. Наприклад, створення вакансій доступне лише роботодавцям та адміністраторам, а керування користувачами – виключно адміністраторам системи.

Модуль AuthService забезпечує реєстрацію користувачів, вхід у систему та відновлення пароля. VacancyService реалізує функції створення, редагування та пошуку вакансій. JobApplicationService відповідає за цикл заявок. EmploymentService забезпечує ведення обліку працевлаштувань та звільнень.

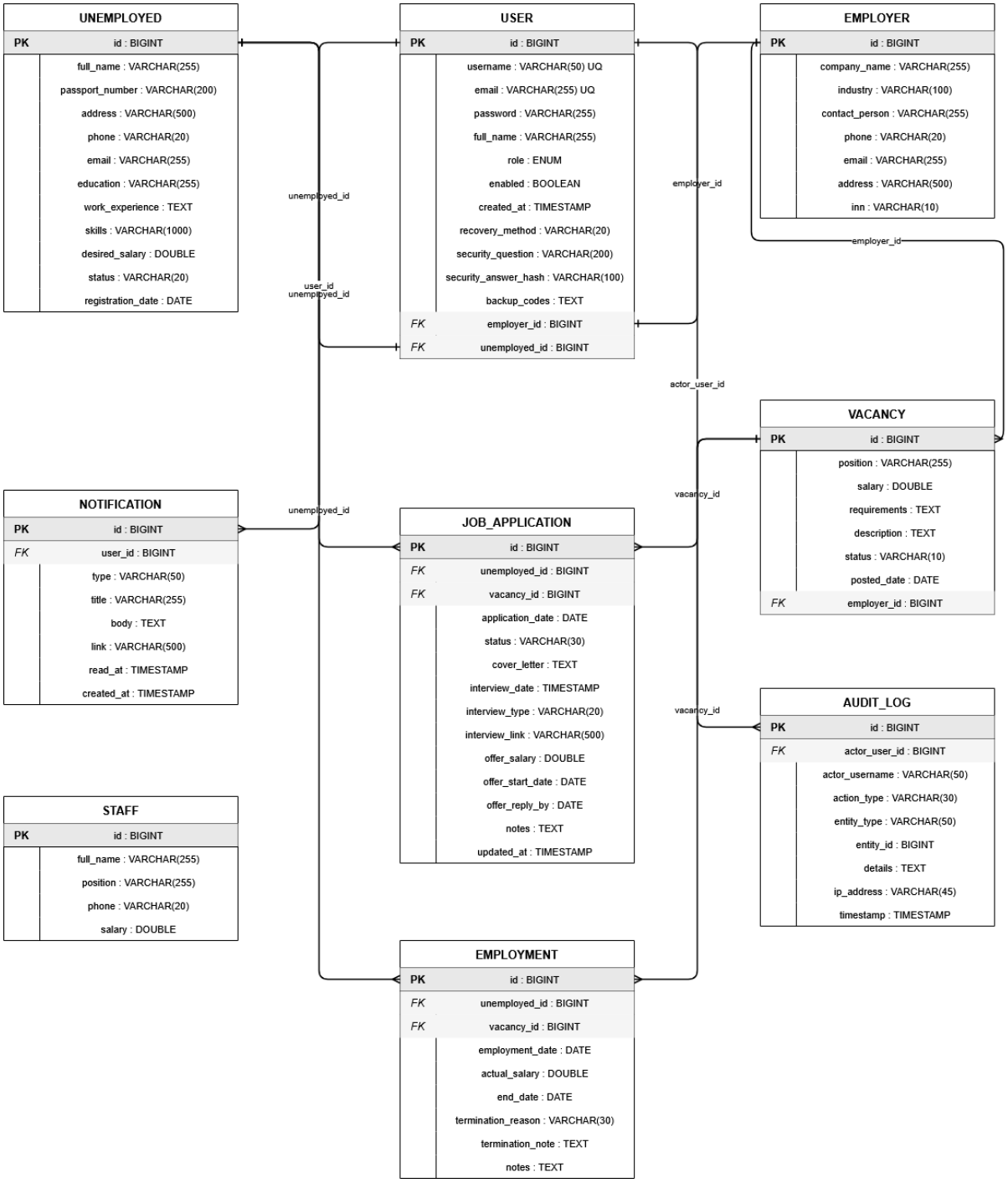
Для автоматичного інформування користувачів про зміни в системі використовується NotificationService. Сервіс створює сповіщення при зміні статусу заявки, отриманні офера, прийнятті рішення роботодавцем або інших важливих подіях. Повідомлення передаються до клієнтської частини через REST API та відображаються в інтерфейсі користувача.

Клієнтська частина програмного забезпечення побудована за принципом file-based routing. Для кожної сторінки системи створюється окремий маршрут, який відповідає певній функції користувача. Наприклад, сторінка /vacancies використовується для перегляду вакансій.

Для покращення взаємодії користувача із системою створено набір багаторазових UI-компонентів. До них належать таблиці, елементи пагінації, компоненти сортування, діалогові вікна підтвердження дій, повідомлення про помилки та інші елементи інтерфейсу. Це дозволяє забезпечити єдиний стиль оформлення системи та спростити підтримку клієнтської частини.

Зв’язки між програмними компонентами також реалізовані на рівні бази даних. Основними сутностями системи є користувачі, роботодавці, безробітні,

вакансії, заявки на вакансії, працевлаштування, персонал центру зайнятості, сповіщення та журнал аудиту. Між таблицями використовуються зв'язки типу One-to-One та One-to-Many, які реалізовані засобами Hibernate ORM. На рисунку 2.7 нижче подано ER-діаграму бази даних



Риснуок 2.7 – ER-діаграма бази даних.

2.5 Написання текстів програм

При написанні текстів програми виконано реалізацію програмного забезпечення інформаційної системи Центру зайнятості відповідно до сформованої архітектури та вимог технічного завдання. Розробка здійснювалася із використанням сучасних засобів програмування та принципів побудови багаторівневих вебзастосунків. Основна увага під час кодування приділялася забезпеченню зрозумілості програмного коду, дотриманню стандартів оформлення, модульності та можливості подальшого супроводження системи.

Серверна частина програмного забезпечення реалізована мовою Java із використанням фреймворку Spring Boot. Для створення клієнтської частини застосовано SvelteKit та мову TypeScript. Такий підхід дозволив забезпечити чітке розділення клієнтської та серверної логіки, спростити підтримку системи та реалізувати взаємодію через REST API у форматі JSON. Використовувався багаторівневий підхід до організації структури проєкту. Контролери відповідають за обробку HTTP-запитів, сервіси реалізують бізнес-логіку системи, репозиторії забезпечують доступ до бази даних, а сутності представляють структуру таблиць PostgreSQL. Лістинги файлів сутностей вакансії та заявки на вакансію Vacancy.java і JobApplication.java подано в додатку А та додатку Б, відповідно. Додатково використовувалися окремі пакети dto, mapper, specification, security та exception. Назви класів та інтерфейсів оформлювалися у стилі PascalCase, наприклад VacancyService або JobApplicationController. Назви методів і полів реалізовувалися у стилі camelCase, наприклад generateToken() або companyName. Діаграма класів подана у графічній частині 2026.КВР.122.421.11.00.00 ДК.

Константи описувалися великими літерами у форматі UPPER_CASE. Для складних ділянок логіки використовувалися пояснювальні коментарі, а для окремих класів – JavaDoc-коментарі. У процесі реалізації програмного забезпечення активно використовувалися можливості сучасних бібліотек та фреймворків. Для організації безпеки застосовано Spring Security із JWT-автентифікацією файли JwtService.java та JwtAuthenticationFilter.java подані у

додатках В та Г. Для взаємодії з базою даних використано Hibernate та Spring Data JPA. Механізми Bean Validation дозволили реалізувати перевірку вхідних даних через анотації @NotBlank, @Email, @Pattern та @Valid зразок файлу запиту з записом даних UnemployedRegisterRequest.java подано у додатку Д. Файл VacancySpecification.java що демонструє динамічні JPA-предикати, LIKE/equal/range фільтри без зміни коду репозиторію подано у додатку Е. Для автоматизації генерації допоміжного коду використовувалася бібліотека Lombok.

Клієнтська частина застосунку реалізована з використанням компонентного підходу. Інтерфейс побудований із незалежних UI-компонентів, що забезпечує повторне використання елементів та спрощує супроводження фронтенду. Для стилізації інтерфейсу використано Tailwind CSS[10], а для побудови графіків у модулі звітності – бібліотеку Chart.js[11]. Взаємодія фронтенду з бекендом здійснюється через централізований API-клієнт, який автоматично додає JWT-токен до HTTP-запитів та обробляє помилки авторизації. Обробка процесу авторизації файл auth.ts подано у додатку Ж, а логіка клієнтської частини файл client.ts у додатку И.

У процесі розробки використовувалися сучасні засоби автоматизації програмування. Розробка серверної частини здійснювалася у середовищі IntelliJ IDEA, а клієнтської – у Visual Studio Code. Для керування версіями проєкту використовувалася система Git та сервіс GitHub. Автоматичне тестування та CI/CD реалізовано через GitHub Actions. Для тестування REST API застосовувалися Postman та Swagger UI.

Здійснювалась обробка помилок і забезпеченню надійності програмного забезпечення. Для цього реалізовано глобальний обробник виключень GlobalExceptionHandler, який обробляє помилки валідації, порушення доступу, конфлікти даних та внутрішні помилки сервера лістинг файлу GlobalExceptionHandler.java поданий у додатку К. На клієнтській частині реалізовано механізми повідомлення користувача про помилки та автоматичний вихід із системи при завершенні терміну дії JWT-токена. Захист від brute-force, Caffeine cache лістинг поданий у файлі LoginRateLimitFilter.java додаток Л.

Для підвищення якості програмного забезпечення було реалізовано багаторівневе тестування системи. Unit-тести створювалися із використанням JUnit 5 та Mockito, integration-тести – через MockMvc, а E2E-тестування – за допомогою Playwright.

2.6 Тестування та налагодження програм

Тестування програмного комплексу інформаційної системи Центру зайнятості проводилося з метою виявлення та усунення помилок у роботі системи.

Відбувалось тестування коректності переходів між статусами заявок, автоматичному створенню запису про працевлаштування при прийнятті офера, а також механізмам авторизації та розмежування доступу між ролями. Теж перевірялася поведінка системи при некоректних вхідних даних – введенні невалідних форматів полів, спробах виконати недозволені дії та зверненнях до захищених ресурсів без автентифікації. У таблиці 2.1 наведено результати тестування сценаріїв з некоректними вхідними даними.

Таблиця 2.1 – Результати тестування з помилкою

Дія користувача	**Очікуваний результат**	**Фактичний результат**	**Перевірка пройдена**
Вхід з невірним паролем	Повідомлення про помилку автентифікації	Відображено повідомлення про помилку	так
Подання заявки на закриту вакансію	Повідомлення про неможливість подання	Відображено відповідне повідомлення	так
Реєстрація з існуючим логіном	Повідомлення «Логін вже зайнятий»	Відображено повідомлення про дублювання	так

Виявлені у процесі тестування помилки стосувалися переважно крайових

випадків бізнес-логіки та коректності обробки граничних значень вхідних даних. Зокрема, було виявлено та виправлено помилку некоректного визначення дати при використанні UTC замість локального часового поясу, а також помилку повторної ініціалізації запису про працевлаштування при повторному виклику методу прийняття офера. Усі виявлені помилки були усунені до завершення розробки. У таблиці 2.2 наведено верифікацію точності результатів при коректних вхідних даних у штатних сценаріях роботи системи.

Таблиця 2.2 – Верифікація точності результатів

Дія користувача	**Очікуваний результат**	**Фактичний результат**	**Перевірка пройдена**
Вхід з коректними даними	Перехід до особистого кабінету	Виконано перехід до кабінету	так
Створення вакансії роботодавцем	Вакансія з'являється у списку	Вакансія відображена у списку	так
Прийняття офера кандидатом	Запис про працевлаштування створено автоматично	Запис створено, статуси оновлено	так

За результатами тестування система коректно обробляє як штатні сценарії роботи, так і некоректні вхідні дані, повертаючи користувачу зрозумілі повідомлення про помилки. Автоматичний запуск тестів при кожному оновленні репозиторію через CI pipeline забезпечує безперервний контроль якості програмного забезпечення.

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з інсталяції програмного забезпечення

3.1.1 Конфігурація технічних засобів

Для забезпечення стабільної роботи інформаційної системи Центру зайнятості необхідно використовувати сучасні технічні засоби, які відповідають мінімальним системним вимогам. Серверна частина програмного забезпечення потребує наявності процесора з тактовою частотою не нижче 2.0 ГГц та підтримкою щонайменше двох обчислювальних ядер. Мінімальний обсяг оперативної пам’яті повинен становити 2 ГБ, проте для стабільної роботи Java Virtual Machine, Hibernate ORM та системи кешування рекомендовано використовувати не менше 4 ГБ оперативної пам’яті. Для зберігання бази даних, журналів аудиту, резервних копій та виконуваних файлів необхідно не менше 10 ГБ вільного дискового простору.

Для роботи користувачів із вебінтерфейсом достатньо персонального комп’ютера або ноутбука із сучасним процесором та оперативною пам’яттю обсягом від 512 МБ вільного простору. Доступ до системи здійснюється через веббраузер, тому обов’язковою умовою є використання сучасних версій Google Chrome, Mozilla Firefox, Microsoft Edge або Safari. Рекомендована роздільна здатність екрана становить не менше 1280×720 пікселів, що забезпечує коректне відображення елементів інтерфейсу та таблиць даних.

Для функціонування серверної частини можуть використовуватись операційні системи Linux, Windows Server або macOS. Найбільш доцільним варіантом для виробничого середовища є Ubuntu 22.04 LTS, оскільки дана система забезпечує високу стабільність роботи серверних застосунків та підтримку сучасних технологій Java. Для розробки програмного забезпечення можуть використовуватись Windows 10/11, Ubuntu або macOS за умови підтримки Java Development Kit та Node.js. Зведенні дані показані у таблиці 3.1.

Таблиця 3.1 – Зведені дані системних вимог

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**
СЕРВЕР		
Процесор	2.0 ГГц, 2 ядра	2.5 ГГц, 4 ядра
Оперативна пам'ять	2 ГБ	4 ГБ
Дисковий простір	10 ГБ	20 ГБ
Операційна система	Linux / Windows Server / macOS	Ubuntu 22.04 LTS
Java	JDK 21	JDK 21
СУБД	PostgreSQL 14+	PostgreSQL 16
КЛІЄНТ		
Процесор	Будь-який сучасний	–
Оперативна пам'ять	512 МБ вільного простору	1 ГБ
Браузер	Chrome / Firefox / Edge / Safari (актуальні версії)	Google Chrome
Роздільна здатність екрана	1280×720 пікселів	1920×1080 пікселів
Операційна система	Windows 10/11 / Ubuntu / macOS	Windows 11 / Ubuntu 22.04
Java Development Kit	JDK 21	JDK 21
Node.js	18+	20 LTS
Оперативна пам'ять	8 ГБ	16 ГБ

Для запуску серверної частини інформаційної системи необхідно встановити Java Development Kit версії 21 LTS, який використовується для виконання Spring Boot застосунку. Як система керування базами даних використовується PostgreSQL версії 14 або новішої. Для запуску та збирання клієнтської частини необхідно використовувати Node.js версії 24 LTS.

Під час розробкипрограмного забезпечення додатково використовуються Apache Maven для автоматизованого збирання Java-проєкту, а також H2 Database, яка застосовується у режимі локальної розробки та тестування. Для автоматизованого тестування інтерфейсу вебзастосунку використовується Playwright, що дозволяє перевіряти роботу клієнтської частини у різних сценаріях використання.

Для забезпечення сумісності компонентів програмного забезпечення необхідно використовувати актуальні версії браузерів та бібліотек. Усі компоненти системи повинні працювати у єдиному програмному середовищі, що забезпечує коректну взаємодію між фронтендом, серверною частиною та базою даних.

3.1.2 Файли налаштування

Для забезпечення роботи програмного комплексу використовуються спеціальні конфігураційні файли. Основним файлом налаштування серверної частини є application.properties, у якому задаються параметри підключення до бази даних, налаштування JWT-автентифікації, параметри кешування та інтеграції із зовнішніми сервісами. Для режиму локальної розробки використовується файл application-dev.properties, який містить конфігурацію H2 Database, SQL-логування та параметри тестового середовища.

У клієнтській частині використовується файл .env, у якому задається адреса REST API серверної частини через змінну VITE_API_URL. Для автоматизованого тестування використовується файл playwright.config.ts, який містить параметри запуску end-to-end тестів.

Під час розгортання програмного забезпечення у виробничому середовищі необхідно визначити змінні оточення. До них належать ключ шифрування конфіденційних даних, секретний ключ для підпису JWT-токенів, а також API-ключ сервісу Google Gemini. Застосування змінних оточення дозволяє підвищити рівень безпеки системи та запобігти зберіганню критичних параметрів безпосередньо у вихідному коді.

3.1.3 Технологічний тип програм та послідовність інсталяції

Програмне забезпечення інформаційної системи має декілька технологічних типів представлення. Вихідний код серверної частини представлений файлами .java, які після компіляції перетворюються у .class файли та упаковуються у виконуваний JAR-архів. Клієнтська частина представлена файлами .svelte та .ts, які після процесу збирання компілюються у статичні JavaScript, HTML та CSS ресурси.

Міграції бази даних реалізовані у вигляді SQL-файлів, які автоматично виконуються Flyway під час першого запуску програмного забезпечення. Автоматизовані тести представлені файлами TypeScript та виконуються через середовище Node.js[12].

Для локальної розробки серверна частина запускається через Spring Boot із використанням вбудованої H2 Database, а клієнтська частина – через Vite Development Server із налаштованим proxy-з'єднанням до REST API. Такий підхід забезпечує зручність тестування та швидку розробку окремих компонентів системи.

Розгортання серверної частини здійснюється шляхом збирання проєкту у виконуваний JAR-файл за допомогою Maven. Для цього виконується команда:

mvn clean package -DskipTests

Після успішного збирання запускається серверна частина в production-режимі. Після запуску автоматично ініціалізується база даних, оскільки система використовує Flyway для виконання міграцій. Це дозволяє створити необхідну структуру таблиць без додаткових ручних налаштувань.

Розгортання клієнтської частини виконується у директорії frontend. Спочатку встановлюються всі залежності проєкту. Далі виконується збірка оптимізованої версії вебзастосунку: npm run build.

Отримані статичні файли можуть бути розміщені на вебсервері або запущені у режимі попереднього перегляду.

Для режиму розробки використовується одночасний запуск двох середовищ. Серверна частина стартує через Spring Boot із вбудованою базою даних H2: mvn

spring-boot:run.

Клієнтська частина запускається окремо у режимі dev-сервера: cd frontend && npm run dev.

У цьому режимі фронтенд працює на порту 5173 із проксуванням запитів до бекенду, який працює на порту 8080. Така конфігурація дозволяє зручно виконувати розробку та тестування системи в реальному часі.

3.2 Інструкція з використання тестових наборів

Для перевірки коректності функціонування програмного комплексу реалізовано багаторівневу систему тестування, що охоплює серверну та клієнтську частини системи. Тестування проводилося із застосуванням фреймворків JUnit 5, Mockito, MockMvc та Playwright.

Основні тести перевіряють базову функціональність кожного модуля в штатних умовах. Для сервісного шару реалізовано 18 тест-класів із використанням JUnit 5 та Mockito, що охоплюють AuthService, VacancyService, JobApplicationService, MatchingService, EmploymentService та інші. Запуск виконується командою: mvn test[13]. Результат виконання тестів показано на рисунку 3.1 нижче.

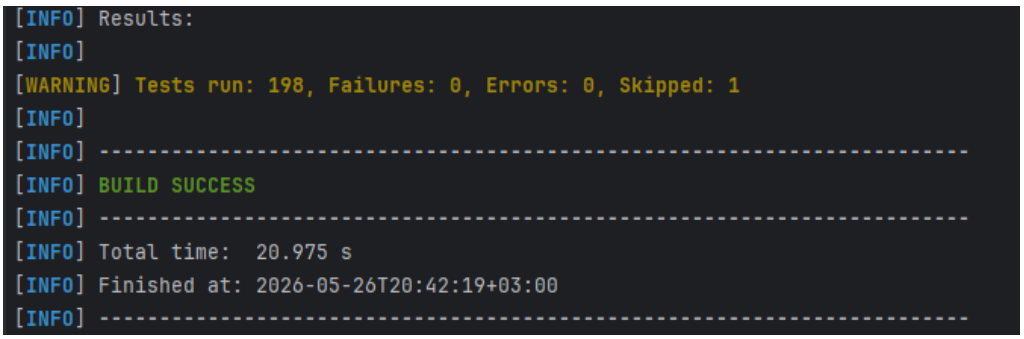


Рисунок 3.1 – Результати виконання unit-тестів у терміналі

Функціональне тестування перевіряє відповідність поведінки системи функціональним вимогам. Integration-тести через MockMvc охоплюють усі основні REST-контролери: AuthControllerTest, VacancyControllerTest,

JobApplicationControllerTest, UnemployedControllerTest, EmployerControllerTest. Кожен тест імітує реальний HTTP-запит та перевіряє коректність відповіді сервера, HTTP-статус та структуру JSON.

Структурне тестування перевіряє внутрішню логіку окремих компонентів. Зокрема, тестується машина станів заявок – коректність дозволених та заборонених переходів між статусами, алгоритм підбору кандидатів MatchingService (розрахунок балів за освітою, досвідом, навичками та зарплатою), а також VacancyOwnershipChecker і ApplicationOwnershipChecker – перевірка права власності на ресурс.

Стикові тести перевіряють взаємодію між компонентами системи. E2E-тести[14] на базі Playwright охоплюють сценарії від авторизації до кінцевого результату. Тести організовані у BDD-форматі через .feature-файли показано на рисунку 3.2 нижче.

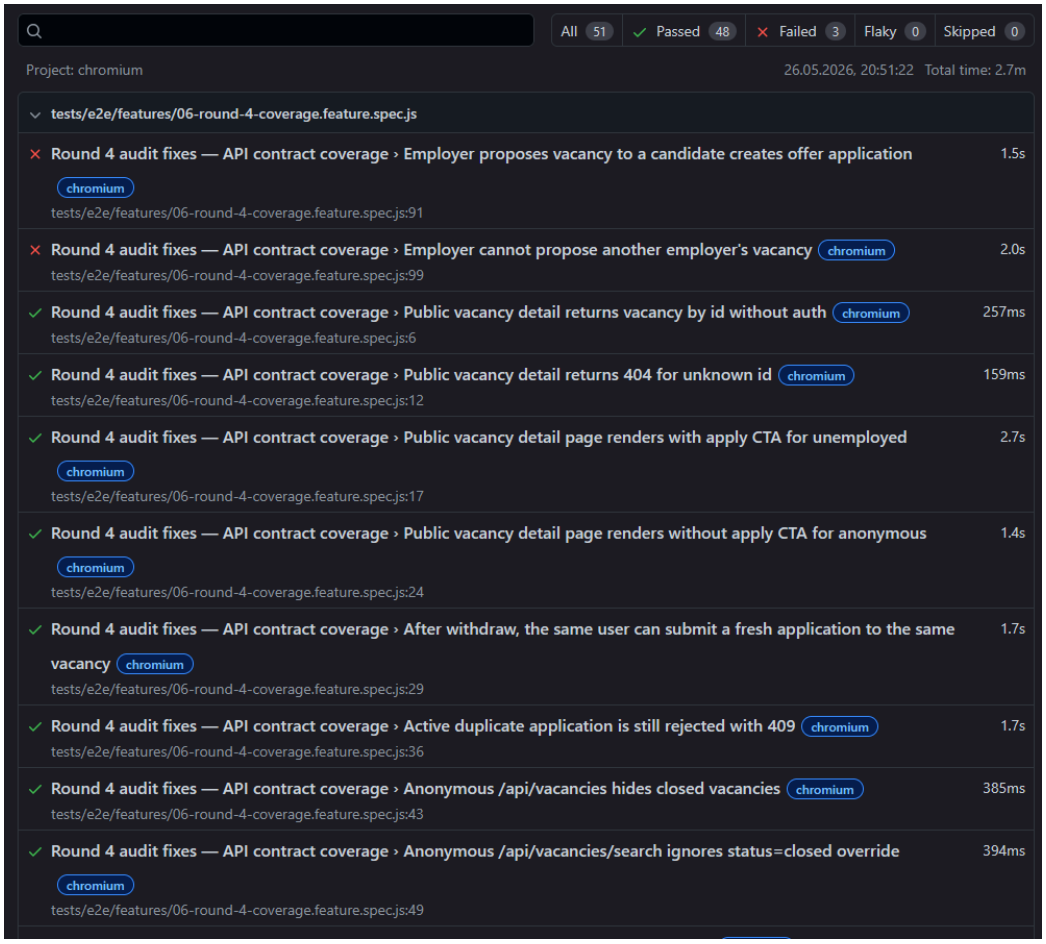


Рисунок 3.2 – Звіт Playwright після виконання E2E-тестів

Аварійні тести перевіряють поведінку системи при некоректних або граничних вхідних даних. Тестуються сценарії передачі порожніх значень обов'язкових полів, введення некоректного формату email, телефону та ІНН, спроби входу з неіснуючим логіном або хибним паролем, а також звернення до захищених ендпоінтів без токена або з простроченим токеном. У всіх випадках система повертає відповідний HTTP-статус та повідомлення про помилку через GlobalExceptionHandlerx[15].

3.3 Інструкція з експлуатації програмного комплексу

Програмний комплекс інформаційної системи Центру зайнятості є вебзастосунком і доступний через браузер без встановлення додаткового програмного забезпечення на робочій станції користувача. Для роботи із системою рекомендується використовувати браузер Google Chrome, Mozilla Firefox або Microsoft Edge актуальних версій. Доступ до системи здійснюється за адресою серверного розгортання.

Система підтримує чотири ролі користувачів: адміністратор, роботодавець, безробітний та базовий користувач. Набір доступних функцій визначається роллю, що призначається під час реєстрації. Головна сторінка вебзастосунку подана на рисунку 3.3

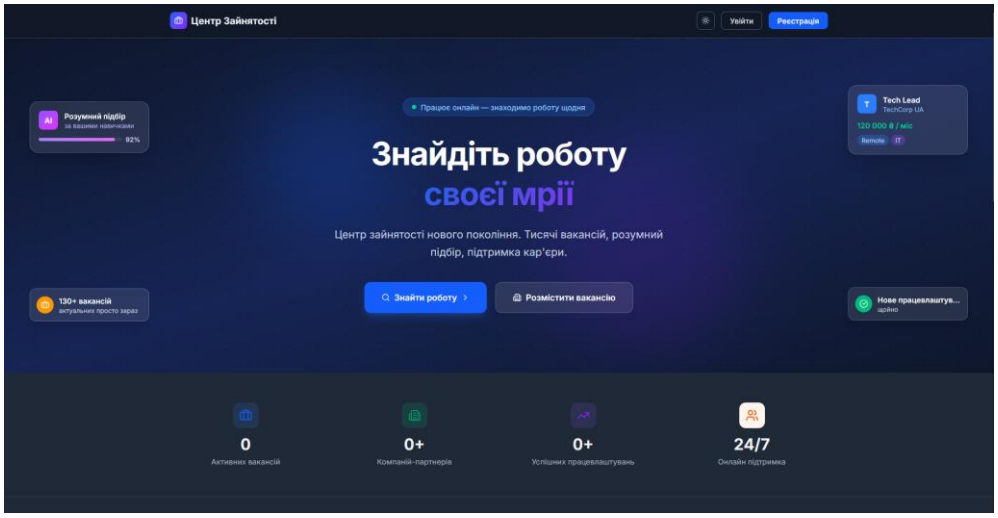


Рисунок 3.3 – Головна сторінка системи

Реєстрація та авторизація. Для входу до системи користувач переходить до розділу «Увійти» та вводить логін і пароль. У разі першого звернення необхідно пройти реєстрацію, обравши роль – роботодавець або безробітний. Реєстрація роботодавця передбачає введення назви компанії, галузі, контактних даних та ІНН. Реєстрація безробітного – введення особистих даних, номера паспорта, адреси, освіти та досвіду роботи подано на рисунку 3.4. Під час реєстрації налаштовується метод відновлення доступу: контрольне запитання або одноразові резервні коди.

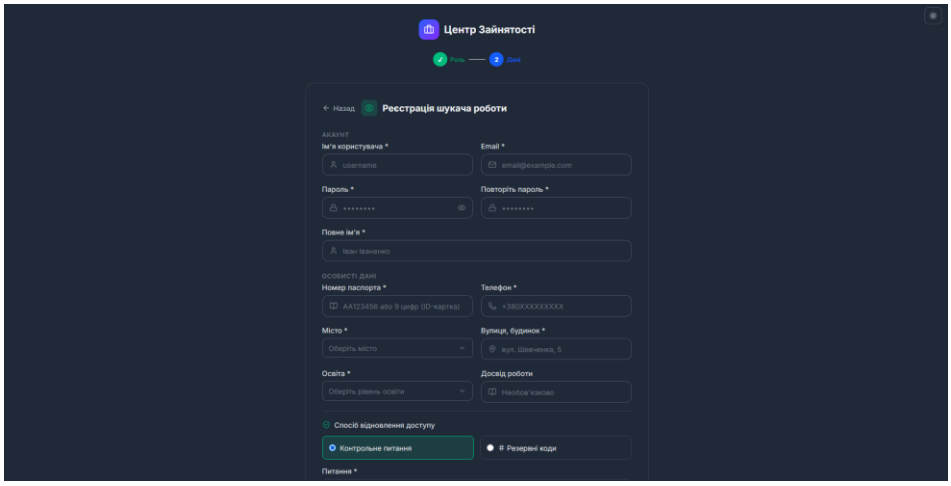


Рисунок 3.4 – Форма реєстрації (введення даних)

Після успішного входу система зберігає сеанс протягом 24 годин, після чого автоматично перенаправляє користувача на сторінку входу подано на рисунку 3.5.

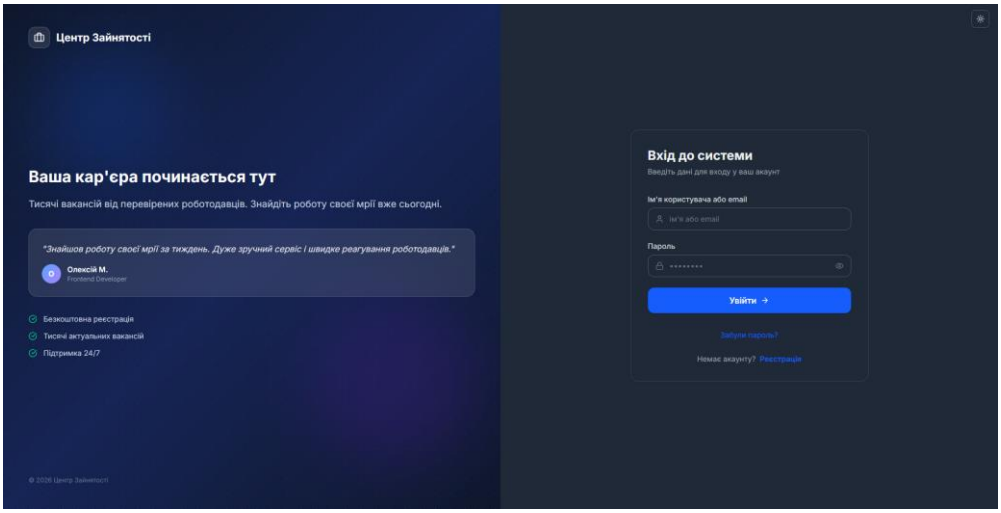


Рисунок 3.5 – Сторінка входу

Робота безробітного. Після входу безробітний потрапляє на особистий дашборд, показано на рисунку 3.6 де відображається поточний статус, активні заявки на вакансії та рекомендовані пропозиції. У розділі «Профіль» користувач може редагувати особисті дані, вказати навички, бажану заробітну плату та рівень освіти – ці дані впливають на результати автоматичного підбору вакансій.

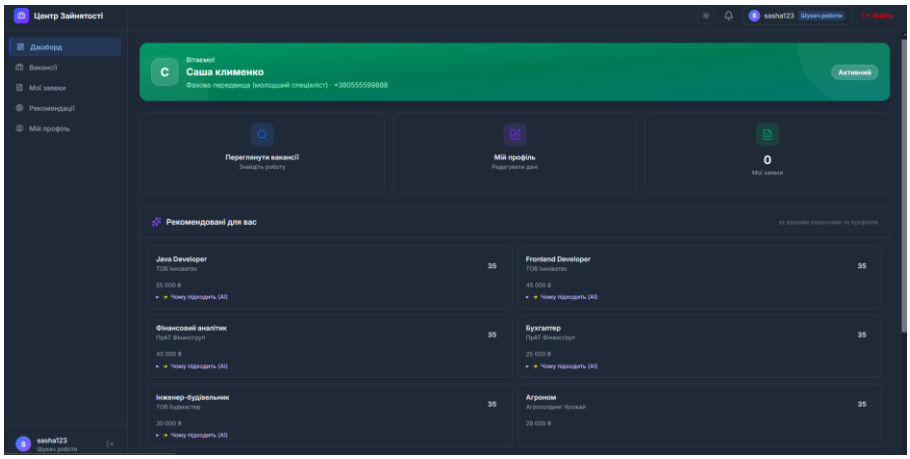


Рисунок 3.6 – Особистий дашборд безробітного

У розділі «Вакансії» відображається список вакансій із можливістю фільтрації за посадою, галуззю, містом та діапазоном заробітної плати. Для подання заявки необхідно відкрити сторінку вакансії, натиснути кнопку «Подати заявку», процес подано на риснку 3.7, після чого з'являється форма введення супровідного листа.

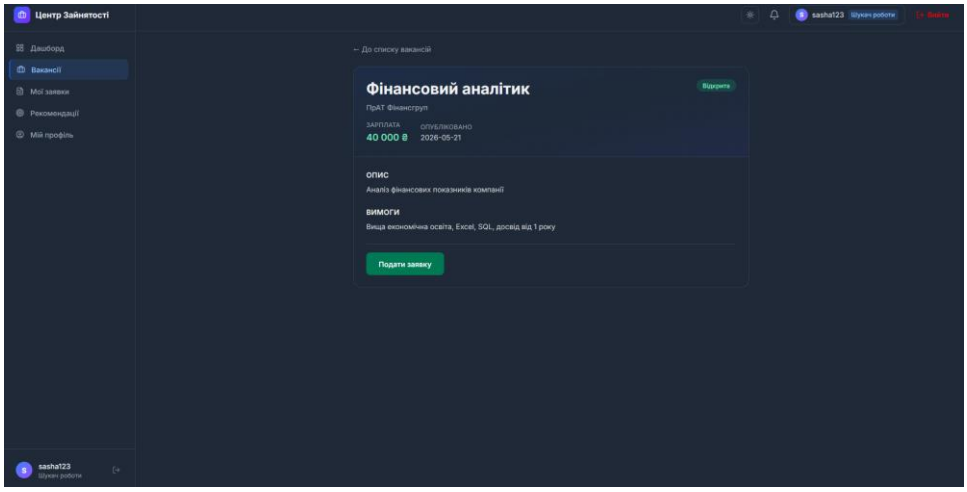


Рисунок 3.7 – Список вакансій з фільтрами

Розділ «Рекомендовані» містить вакансії, підібрані алгоритмом відповідності на основі навичок, освіти та бажаної зарплати. Кожному збігу присвоюється рейтинг від 0 до 100 балів. У розділі «Мої заявки» відображається повна історія звернень із поточним статусом кожної заявки. При переведенні заявки у статус «Оффер» безробітний отримує сповіщення та може прийняти або відхилити пропозицію безпосередньо в системі. При прийнятті офера автоматично формується запис про працевлаштування, статус користувача змінюється на «Працевлаштований», а вакансія закривається. Для завершення трудових відносин передбачена функція самостійного звільнення через кнопку «Звільнитися» на дашборді.

Робота роботодавця. Після входу роботодавець отримує доступ до власного кабінету. У розділі «Мої вакансії» відображається перелік створених оголошень із поточним статусом, сторінка показана на рисунку 3.8 нижче.

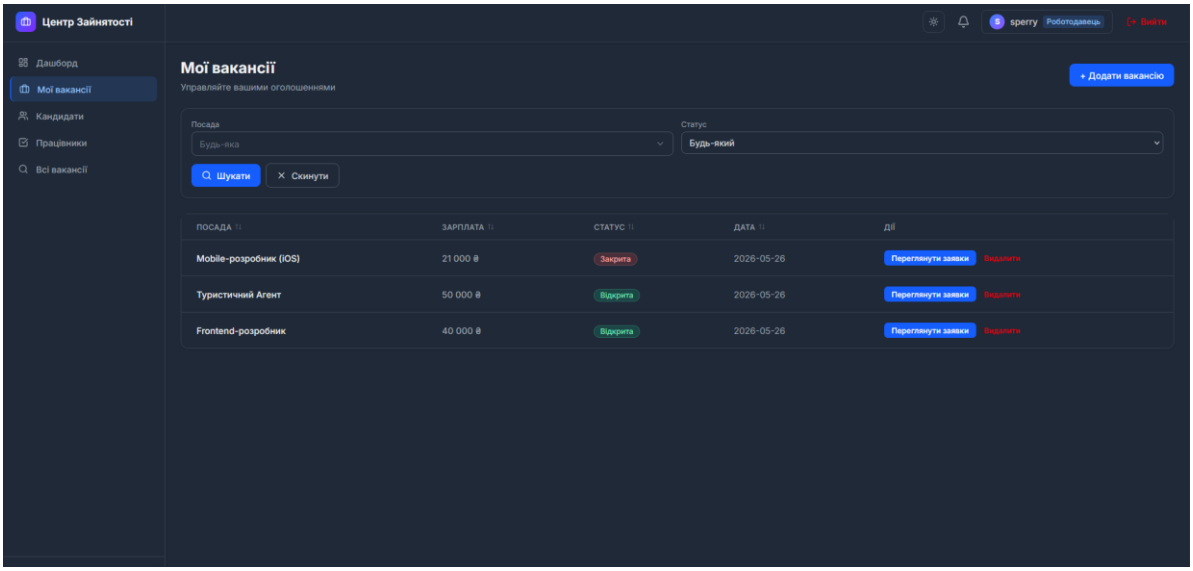


Рисунок 3.8 – Список вакансій роботодавця

Для створення нової вакансії необхідно натиснути кнопку «Додати вакансію» та заповнити форму як на рисунку 3.9, посада, заробітна плата, вимоги до кандидата та опис. Редагування та видалення вакансій доступне через відповідні кнопки у рядку кожної вакансії.

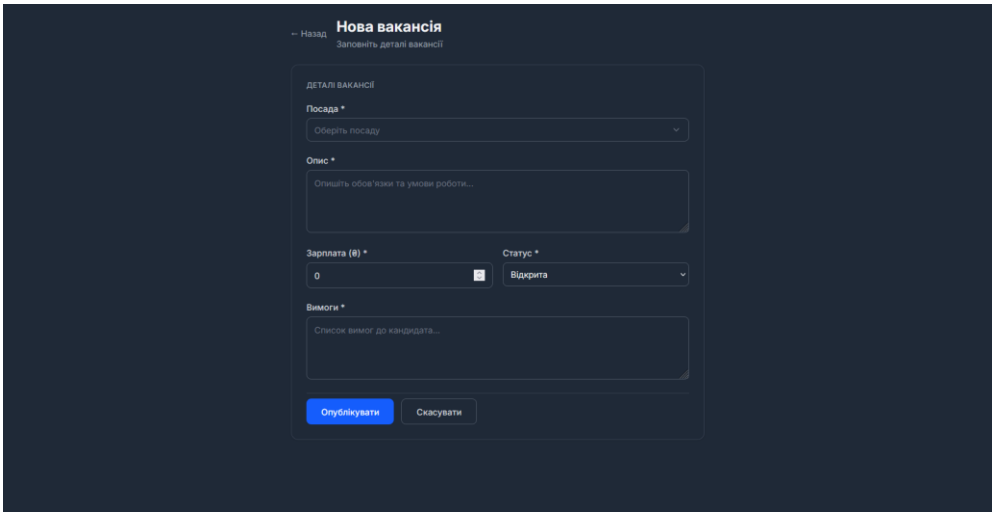


Рисунок 3.9 – Форма створення вакансії

У розділі «Заявки» відображаються всі звернення кандидатів на вакансії роботодавця. Для кожної заявки роботодавець може змінювати статус відповідно до етапу розгляду: переглянуто, запрошено на співбесіду, співбесіда пройдена, оффер. При виставленні офера вказується запропонована заробітна плата, дата початку роботи та кінцевий термін відповіді кандидата. Розділ «Кандидати» містить список безробітних із можливістю перегляду профілю та надсилання прямої пропозиції вакансії.

Робота адміністратора. Адміністратор має повний доступ до всіх розділів системи. У розділі «Користувачі», як подано на рисунку 3.10, доступне керування обліковими записами: зміна ролі, блокування, скидання пароля та видалення.

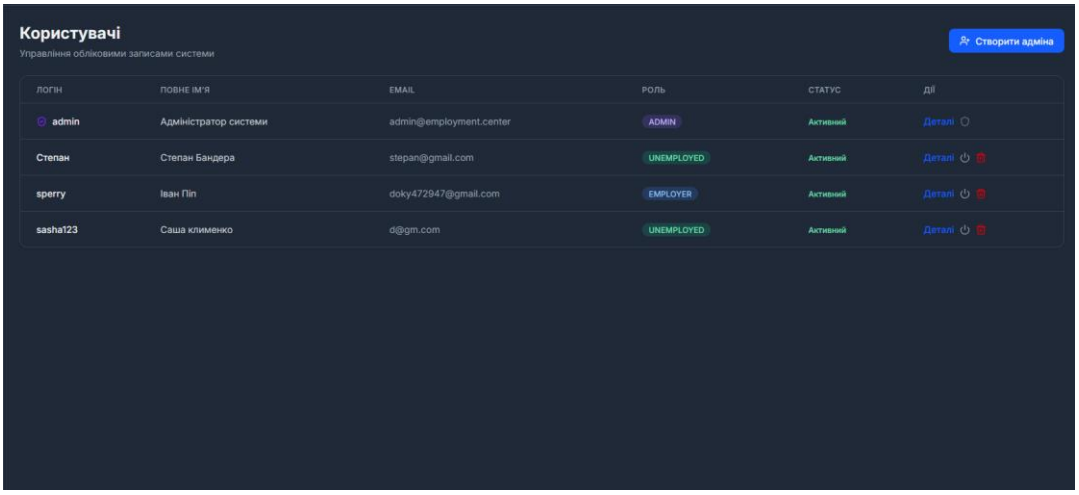


Рисунок 3.10 – Панель адміністратора / керування користувачами

Розділи «Безробітні», «Роботодавці» та «Вакансії» дозволяють переглядати, редагувати та видаляти будь-які записи. Розділ «Звіти» містить аналітичні дашборди: динаміка працевлаштувань по місяцях, розподіл за галузями та містами, середня заробітна плата за посадами, конверсія воронки заявок та рейтинг роботодавці, подано на рисунку 3.11. Журнал аудиту фіксує всі дії користувачів із зазначенням часу, IP-адреси та типу операції.

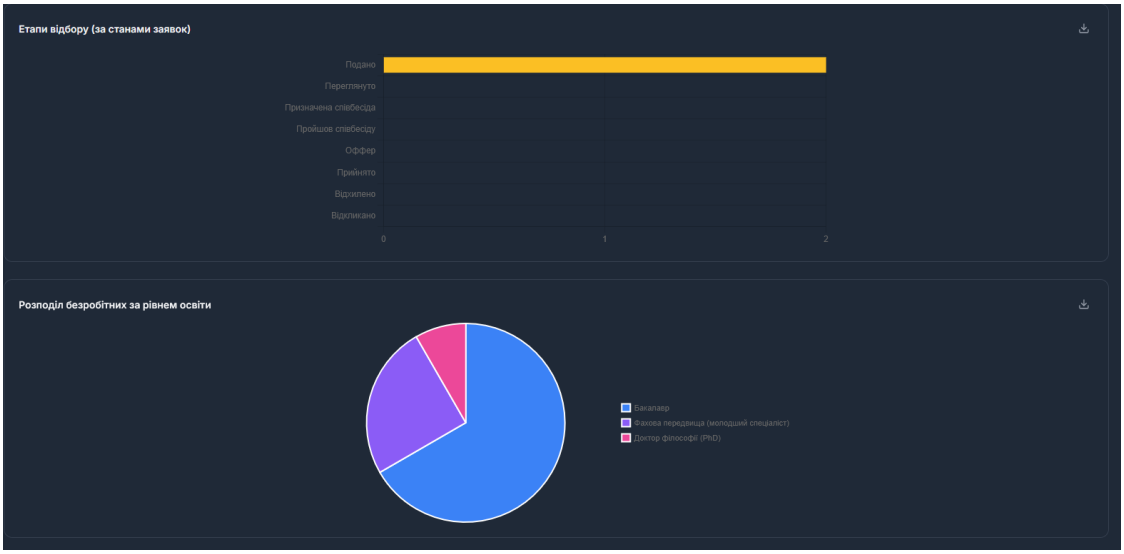


Рисунок 3.11 – Сторінка звітів з графіками

Усі ролі отримують сповіщення про зміни, що їх стосуються. Лічильник непрочитаних сповіщень відображається у верхній панелі навігації. Сповіщення надходять при зміні статусу заявки, надходженні нової заявки на вакансію роботодавця, а також при підтвердженні або відхиленні офера.

Відомості про використання ресурсів. Клієнтська частина системи є статичним вебзастосунком і не висуває суттєвих вимог до ресурсів робочої станції користувача. Зібрані статичні файли клієнта займають близько 2–5 МБ. Серверна частина функціонує у вигляді виконуваного JAR-файлу розміром близько 65–80 МБ. Для стабільної роботи серверу рекомендується не менше 512 МБ оперативної пам'яті, оптимальний обсяг – 1 ГБ. Вбудований кеш Caffeine зберігає до 500 записів із терміном зберігання 5 хвилин. Файл бази даних H2 у режимі розробки з тестовими даними займає від 1 до 3 МБ.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини даного дипломного проєкту є проведення економічних розрахунків, спрямованих на визначення економічної ефективності розробки інформаційної системи Центру зайнятості, прийняття рішення про подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки.

Об’єктом розробки є інформаційна система Центру зайнятості

Розрахунок вартості розробки виконується в декілька етапів:

- описати технологічний процес розробки із зазначенням трудомісткості кожної операції;
- визначити суму витрат на оплату праці основного і допоміжного персоналу, включаючи відрахування на соціальні заходи;
- обчислити витрати на електроенергію;
- нарахувати суму амортизаційних відрахувань;
- визначити суму накладних витрат;
- скласти кошторис та визначити собівартість робіт;
- розрахувати ціну робіт;
- визначити економічну ефективність та термін окупності.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

В цьому підрозділі розглянемо основні етапи технологічного процесу для розробки інформаційної системи Центру зайнятості. Розробка програмного комплексу відбувається поетапно відповідно до методології життєвого циклу програмного забезпечення. Для визначення загальної тривалості проведення робіт доцільно дані витрат часу по окремих операціях технологічного процесу звести у таблицю. В таблиці 4.1 подано середній час виконання робіт по обслуговуванню та стадій(операції) технологічного процесу.

Таблиця 4.1 - Середній час виконання робіт по обслуговуванню та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Планування та аналіз	Кер. Проєкту(Pm)	12
		Інженер (l1)	6
2	Розробка технічного завдання	Кер. Проєкту(Pm)	6
		Інженер (l1)	6
3	Дизайн інтерфейсу	Інженер (l1)	12
		Інженер (l2)	24
4	Розробка функціоналу	Інженер (l1)	44
5	Тестування та відладка	Тестувальник	8
6	Документування	Інженер (l1)	8
7	Розгортання та підтримка	Інженер (l2)	16
8	Управління проєктом	Кер. Проєкту(Pm)	24
Разом			166

Сумарний час виконання операцій технологічного процесу становить 166 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

У даному підрозділі проводиться аналіз і розрахунок витрат, пов’язаних із оплатою праці та нарахуванням обов’язкових соціальних внесків, які виникають у процесі розробки інформаційної системи Центру зайнятості. Розмір заробітної плати розробників залежить від складності та обсягу виконуваної роботи, кваліфікаційного рівня працівника, досвіду, та умов функціонування підприємства.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с * К_г \quad (4.1)$$

де: T_c – тарифна ставка, грн. (приймаємо для керівника проєкту (Pm) – 350 грн./год, інженера (l2) – 220 грн./год., інженера (l1) – 180 грн./год., тестувальник – 100 грн./год.; $К_г$ – кількість відпрацьованих годин.

Отже, основна заробітна плата для:

Керівника проекту (Pm) $З_{осн1}$= 42 * 350 = 14 700 грн.

Інженера (l1) $З_{осн2}$= 76 * 180 = 13 680 грн.

Інженера (l2) $З_{осн3}$= 40 * 220 = 8 800 грн.

Тестувальник $З_{осн4}$= 8 * 100 = 800 грн.

Сумарна основна заробітна плата становить

$З_{осн}$ = 14 700 + 13 680 + 8 800 + 800 = 37 980 грн.

Додаткова заробітна плата становить 10 – 15 % від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.} \quad (4.2)$$

де: $К_{допл.}$ – коефіцієнт додаткових виплат працівникам.

Отже додаткова заробітна плата по категоріях працівників становить:

Керівника проекту $З_{дод1}$= 14 700 * 0,1 = 1 470 грн.

Інженера (l1) $З_{дод2}$= 13 680 * 0,1 = 1 368 грн.

Інженера (l2) $З_{дод3}$= 8 800 * 0,1 = 880 грн.

Тестувальник $З_{дод4}$= 800 * 0,1 = 80 грн.

Загальна додаткова заробітна плата становить:

$З_{дод}$= 1 470 + 1 368 + 880 + 80 = 3 798 грн.

Звідси загальні витрати на оплату праці ($B_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (4.3)$$

$В_{о.п}$= 37 980 + 3 798 = 41 778 грн.

Єдиний соціальний внесок (ЄСВ – 22%) визначається за формулою:

$$В_{єсв}=Воп*0,22 \quad (4.4)$$

$В_{єсв}$ = 41 778 * 0,22 = 9 192 грн.

Проведені розрахунки витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	ЄСВ, грн.	Всього витрати на оплату праці, грн.
		Тарифна ставка, грн.	К-сть годин	Фактичного нарах. зарплати, грн.			
1	Кер. проєкту (Pm)	350	42	14 700	1 470	3558	19 728
2	Інженера (l1)	180	76	13 680	1 368	3 310	18 358
3	Інженера (l2)	220	40	8 800	880	2130	11810
4	Тестувальник	100	8	800	80	194	1 074
Разом				37 980	3 798	9 192	50 970

Отже, загальні витрати на оплату праці становлять 50 970 грн

4.3 Розрахунок витрат на електроенергію

Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_в = W * T * S \quad (4.5)$$

де: W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії (приймаємо 15,94 грн).

В нашій системі є 1 ПК. Витрати на електроенергію для цього комп’ютера обчислимо окремо, взявши за основу, що час роботи обладнання обчислюється в залежності від виконуваних робіт (згідно табл. 4.1) і споживані потужності наступні: комп’ютер – 0,85 кВт/год.

$З_{ек}$= 0,85*166*15,94= 2 249 грн.

Витрати на електроенергію становлять 2 249 грн.

4.4 Розрахунок суми амортизаційних відрахувань для розробки інформаційної системи Центру зайнятості

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення

Комп'ютери та оргтехніка належать до четвертої групи основних фондів.

Амортизація на них нараховується лише в випадку, якщо мінімально допустимі строки їх корисного використання 2 роки. Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = \frac{Б_В * Н_А}{100\%} * T, \qquad (4.6)$$

де: А – амортизаційні відрахування за звітний період, грн.;

$Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_А$ – норма амортизації, 0,04 %.

Оскільки для написання програми та її тестування використовується один ПК, вартістю 50000,00 грн., то сума амортизаційних відрахувань становитиме:

$$А = \frac{50\,000*0{,}04}{150} * 166 = 2\,213 \text{грн.}$$

4.5 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$Н_в=В_{о.п.} * \ 0,2..0,6 \qquad (4.7)$$

де: $Н_В$ – накладні витрати.

$Н_В$= 41 778 *0,4= 16 711 грн.

4.6 Складання кошторису витрат та визначення собівартості для розробки інформаційної системи Центру зайнятості

Для складення кошторису витрат та визначення собівартості, результати проведених вище розрахунків зведемо у таблиці 4.4.

Таблиця 4.4 - Кошторис витрат розробки інформаційної системи Центру зайнятості

№	Зміст витрат	Сума, грн.	В % до загальної суми
1	Витрати на оплату праці (основну і додаткову заробітну плату)	41 778	57
2	Єдиний соціальний внесок	9 192	13
3	Витрати на електроенергію	2 249	3
4	Амортизаційні відрахування	2 213	4
5	Накладні витрати	16 711	23
	Собівартість	72 143	100

Собівартість ($С_В$) НДР розрахуємо за формулою:

$$Св = Во.п + Вс.з + Зе + А + Нв \qquad (4.8)$$

Отже, собівартість дорівнює $С_В$= 72 143грн.

4.7 Розрахунок ціни для розробки інформаційної системи Центру зайнятості

Розрахунок ціни науково-дослідної роботи включає в себе урахування різноманітних факторів, таких як рівень рентабельності, собівартість та податкова ставка.

Ціну робіт можна визначити за формулою:

$$Ц=Св*(1+Р_{рен})*(1+ПДВ), \quad (4.9)$$

де: Св – собівартість; $Р_{рен.}$ – рівень рентабельності, 30 %; ПДВ – ставка податку на додану вартість, 20 %

Ц = 72 143 * (1+0,3) * (1+0,2)= 112 543грн.

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Розрахунок прибутку визначається з а формулою:

$$П = Ц – Св \quad (4.10)$$

де: Ц – ціна розробки, грн.; Св – собівартість розробки, грн.

П = 112 543 – 72 143 = 40 400 грн.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності (Ток).

$$ЧТВ = -Св + \sum_{i=1}^{t} \frac{Г_П}{(1+i)^t} \quad (4.11)$$

де: Св – собівартість розробки; $Г_п$ – грошовий потік за t – ий рік; t – відповідний рік проекту; і – величина дисконтної ставки (10…15%).

$$\text{ЧТВ} = -72\,143 + \frac{40\,400}{(1+0{,}1)^1} + \frac{40\,400}{(1+0{,}1)^2} + \frac{40\,400}{(1+0{,}1)^3}$$

$$= -72\,143 + 36\,727 + 33\,388 + 30\,353 = 28\,325 \text{ грн}$$

Якщо ЧТВ ≥ 0, то проект може бути рекомендований до впровадження.

Термін окупності визначається за формулою:

$$T_{\text{ОК}} = T_{\text{ПВ}} + \frac{\text{Н}_\text{В}}{\Gamma_{\text{ПР}}} \qquad (4.12)$$

де: $\text{Т}_{\text{ПВ}}$ – період до повного відшкодування витрат, років; $\text{Н}_\text{В}$ – невідшкодовані витрати на початок року, грн.; $\Gamma_{\text{ПР}}$ – грошовий потік на початок року, грн.

$$\text{Нв} = \left(\frac{40\,400}{(1+0{,}1)^3}\right) - 28\,325 = 30\,353 - 28\,325 = 2\,028 \text{ грн}$$

$$\text{Т}_{\text{ок}} = 2 + \frac{2\,028}{40\,400} = 2{,}05 \text{ р.}$$

Всі дані внесемо в зведену таблицю 4.5.

Таблиця 4.5 – Техніко-економічні показники веб застосунку

№ п/п	**Показник**	**Значення**
1.	Собівартість, грн.	72 143
2.	Плановий прибуток або грошовий потік, грн.	40 400
3.	Ціна, грн.	112 543
4.	Чиста теперішня вартість, грн.	28 325
5.	Термін окупності, рік	2,05

Прибутковість проєкту та термін окупності свідчать про його фінансову ефективність та здатність повернути капітальні вкладення протягом 2,05 року. Отже, на основі отриманих показників можна зробити висновок, що розробка веб застосунку є доцільною з економічної точки зору.

5 ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

5.1 Функції служби охорони праці підприємства.

Служба охорони праці є структурним підрозділом підприємства, що створюється відповідно до Закону України «Про охорону праці» з метою забезпечення безпечних та здорових умов праці, запобігання нещасним випадкам і професійним захворюванням. Відповідно до статті 15 зазначеного Закону, служба охорони праці створюється на підприємствах з кількістю працюючих 50 і більше осіб. На підприємствах з меншою чисельністю працівників функції служби охорони праці можуть виконуватися в порядку сумісництва особами, які мають відповідну підготовку[17].

Служба охорони праці підпорядковується безпосередньо роботодавцю і прирівнюється за своїм статусом до основних виробничо-технічних служб підприємства. Працівники служби не можуть залучатися до виконання функцій, не передбачених Законом України «Про охорону праці» та Типовим положенням про службу охорони праці.

Розробка та впровадження заходів з охорони праці. Служба розробляє комплексні заходи щодо досягнення встановлених нормативів та підвищення існуючого рівня охорони праці, готує розділи колективного договору щодо охорони праці та бере участь у розробці положень, інструкцій та інших нормативних актів з охорони праці, що діють у межах підприємства.

Служба здійснює контроль за станом умов праці на робочих місцях, за дотриманням працівниками вимог законодавства про охорону праці, за використанням засобів індивідуального та колективного захисту, за проходженням працівниками медичних оглядів. Служба бере участь у розслідуванні нещасних випадків, профзахворювань і аварій відповідно до встановленого порядку, аналізує їх причини та розробляє заходи щодо запобігання повторенню подібних випадків.

Проводиться навчання та інструктаж з питань охорони праці, перевірку вимог охорони праці посадовими особами та іншими працівниками підприємства.

Інформаці працівників про основні вимоги законів та інших нормативних актів з охорони праці, забезпечує підрозділи підприємства нормативними документами та засобами пропаганди з питань охорони праці. Служба проводить з працівниками бесіди, лекції, огляди з питань охорони праці, організовує виставки, конкурси та інші заходи, спрямовані на підвищення рівня обізнаності персоналу у питаннях безпеки праці.

У сфері розробки програмного забезпечення служба охорони праці виконує ті самі законодавчо визначені функції, проте їх зміст суттєво відрізняється від виробничих підприємств через специфіку діяльності. Основним виробничим обладнанням є комп'ютерна техніка, а більшість загроз здоров'ю працівників пов'язана не з механічними або хімічними чинниками, а з тривалою роботою за монітором, неправильною організацією робочого місця та психоемоційним навантаженням.

Розробка нормативних документів. Служба охорони праці IT-компанії розробляє інструкції з безпечної роботи з комп'ютерною технікою, положення про режим праці та відпочинку для операторів ПК відповідно до ДСанПіН 3.3.2.007-98, а також інструкції з пожежної безпеки для приміщень з великою кількістю електронного обладнання. Контроль умов праці. Служба здійснює контроль за відповідністю робочих місць розробників санітарно-гігієнічним нормам: рівень освітленості (не менше 300–500 лк на поверхні столу), мікроклімат приміщення (температура 21–23°C, відносна вологість 40–60%), рівень шуму (не більше 50 дБА), а також розташування моніторів (відстань від очей не менше 50–70 см, кут нахилу екрана 10–20°). Для IT-компанії атестація робочих місць передбачає оцінку ергономічних характеристик: висота столу та крісла, наявність підставки для ніг, підлокітників, можливість регулювання висоти монітора. Особлива увага приділяється організації робочого місця з метою запобігання розвитку професійних захворювань – синдрому зап'ястного каналу, остеохондрозу та порушень зору[19].

Режим праці та відпочинку. Відповідно до нормативних вимог, при роботі з комп'ютером понад 4 години на день обов'язковим є дотримання регламентованих перерв: 10–15 хвилин після кожних 2 годин роботи. Служба охорони праці

контролює дотримання цих вимог та організовує проведення виробничої гімнастики.

5.2 Чинники, що впливають на тяжкість ураження людини елекричним струмом.

Електричний струм є одним із найнебезпечніших чинників виробничого середовища. На відміну від більшості інших небезпечних чинників, електричний струм не має запаху, кольору та не виявляється органами чуття людини до моменту безпосереднього контакту. Це робить електричну небезпеку особливо підступною і вимагає суворого дотримання правил електробезпеки.

Тяжкість ураження людини електричним струмом визначається сукупністю чинників, які умовно поділяються на електричні та неелектричні[18].

Сила струму є основним чинником, що визначає тяжкість ураження. Сила Значення струму відповідно до фізіологічної дії, показано на таблиці 5.1.

Таблиця 5.1 – Значення струму відповідно до фізіологічної дії

Значення струму	**Змінний струм (50 Гц)**	**Постійний струм**	**Фізіологічна дія**
0,5–1,5 мА	Відчутний	5–7 мА	Легке поколювання
2–3 мА	–	–	Сильне поколювання
5–10 мА	Невідпускаючий	50–80 мА	Судоми, неможливість відірватись
10–50 мА	Небезпечний	100–300 мА	Фібриляція серця
понад 100 мА	Смертельний	понад 300 мА	Зупинка серця, опіки

Зі збільшенням напруги зменшується опір тіла людини внаслідок пробою

шкіри, що призводить до різкого зростання струму через тіло. Напруга до 36 В вважається безпечною для сухих приміщень, до 12 В – для вологих. Змінний струм промислової частоти (50 Гц) є більш небезпечним порівняно з постійним струмом аналогічної величини приблизно у 3–5 разів, оскільки спричиняє судомне скорочення м'язів і унеможливлює самостійне звільнення від дії струму. Зі збільшенням часу дії струму тяжкість ураження зростає, оскільки знижується опір тіла людини, збільшується ймовірність збігу моменту дії з найбільш вразливою фазою серцевого циклу (фаза Т на кардіограмі). Струм тривалістю менше 0,1 секунди значно менш небезпечний, ніж той самий струм тривалістю 1–2 секунди.

Найнебезпечнішими є петлі, що проходять через серце та головний мозок. На тяжкість ураження також впливають стан здоров'я, втома, вживання алкоголю та наявність захворювань серцево-судинної системи. Підвищена температура та вологість знижують опір шкіри, збільшуючи небезпеку ураження.

Робочі місця розробників програмного забезпечення відносяться до приміщень без підвищеної електричної. Незважаючи на це, тяжкість ураження електричним струмом, є обов'язковим для кожного працівника.

Основними джерелами електричної небезпеки на робочому місці розробника є комп'ютерна техніка периферійні пристрої, зарядні пристрої, подовжувачі та електрична мережа 220 В. Найбільш поширеними причинами ураження є пошкодження ізоляції кабелів, несправні розетки та подовжувачі, порушення правил експлуатації електрообладнання.

Для зниження ризику ураження електричним струмом на робочих місцях розробників необхідно забезпечити справний стан електропроводки та розеток, використовувати лише сертифіковані зарядні пристрої та периферію, не допускати попадання рідини на електронне обладнання, не виконувати самостійного ремонту електрообладнання, а також забезпечити заземлення серверного обладнання та технічних засобів у серверних кімнатах. Особливу увагу слід приділяти організації кабельного господарства: кабелі не повинні перетинати проходи, звисати з робочих поверхонь або мати пошкодження ізоляції.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інформаційну систему Центру зайнятості, призначену для автоматизації процесів взаємодії між безробітними, роботодавцями та працівниками центру зайнятості. У процесі роботи було проведено аналіз предметної області, досліджено сучасні технології веброзробки та обґрунтовано вибір архітектурних і програмних рішень для створення програмного забезпечення.

У ході виконання роботи було реалізовано клієнтську та серверну частини вебзастосунку. Серверна частина створена із використанням мови програмування Java та фреймворку Spring Boot, а клієнтська частина – на основі SvelteKit та TypeScript. Для зберігання даних використано систему керування базами даних PostgreSQL. У системі реалізовано REST API, JWT-автентифікацію, рольову модель доступу, механізми валідації та шифрування чутливих даних.

Під час розробки були використані сучасні принципи програмної інженерії. Для перевірки працездатності програмного забезпечення виконано unit-, integration- та end-to-end тестування, що дозволило забезпечити стабільність роботи системи та коректність реалізації бізнес-логіки.

Отримані результати відповідають вимогам технічного завдання та дозволяють автоматизувати основні процеси діяльності центру зайнятості. Розроблена система має можливість подальшого масштабування та розширення функціоналу. У перспективі можливим є впровадження мобільного застосунку, інтеграції з державними електронними сервісами, а також розширення AI-функціоналу для більш точного аналізу вакансій і кандидатів.

Під час виконання кваліфікаційної роботи автором самостійно було виконано аналіз предметної області, проєктування структури бази даних, розробку серверної та клієнтської частин застосунку, реалізацію REST API, механізмів авторизації, системи ролей, модулів вакансій, заявок, працевлаштування, статистики та аудиту. Також самостійно було виконано тестування програмного забезпечення та підготовлено програмну документацію.

ПЕРЕЛІК ПОСИЛАНЬ

1) Єфімов О.В., Кузьменко Є.В. Розробка веб-застосунків на платформі Java : навч. посібник. Харків : ХНУРЕ, 2019. 212 с. (дата звернення: 08.05.2026).

2) Silberschatz A., Korth H., Sudarshan S. Database System Concepts. 7th Edition. New York : McGraw-Hill, 2021. 1376 с. (дата звернення: 08.05.2026).

3) Svelte Documentation: вебсайт. URL: https://svelte.dev/docs/svelte (дата звернення: 09.05.2026).

4) SvelteKit Documentation: вебсайт. URL: https://svelte.dev/docs/kit/introduction (дата звернення: 09.05.2026).

5) Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: dissertation. University of California, Irvine, 2000. URL: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення: 10.05.2026).

6) Walls C. Spring Boot in Action. 2nd Edition. Shelter Island : Manning Publications, 2022. 480 . (дата звернення: 10.05.2026).

7) Bauer C., King G., Gregory G. *Java Persistence with Hibernate*. 3rd Edition. Shelter Island : Manning Publications, 2023. 864 с. (дата звернення: 10.05.2026).

8) Jones M., Bradley J., Sakimura N. RFC 7519: JSON Web Token (JWT). IETF, 2015. URL: https://www.rfc-editor.org/rfc/rfc7519.html (дата звернення: 11.05.2026).

9) Pollock C. Spring Security: Securing Spring Applications. Birmingham : Packt Publishing, 2022. 350 с. (дата звернення: 11.05.2026).

10) Tailwind CSS Documentation: вебсайт. URL: https://tailwindcss.com/docs (дата звернення: 12.05.2026).

11) Murray S. Interactive Data Visualization for the Web. 3rd Edition. Sebastopol : O'Reilly Media, 2022. (дата звернення: 12.05.2026).

12) Flyway Documentation. Redgate: вебсайт. URL:

https://documentation.red-gate.com/fd (дата звернення: 13.05.2026).

13) Spring Boot: Building an Application with Spring Boot. Getting Started Guide: вебсайт. URL: https://spring.io/guides/gs/spring-boot (дата звернення: 14.05.2025).

14) Gopinath P. End-to-End Testing with Playwright. Birmingham : Packt Publishing, 2023. (дата звернення: 13.05.2026).

15) Mockito Framework Documentation: вебсайт. URL: https://site.mockito.org/ (дата звернення: 15.05.2026).

16) Методичні вказівки до виконання кваліфікаційної роботи за спеціальністю 122. Комп’ютерні науки (дата звернення: 15.05.2026).

17) Закон України «Про охорону праці» від 14.10.1992 № 2694-XII : вебсайт. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 22.05.2026).

18) НПАОП 0.00-1.28-10. Правила охорони праці для експлуатації електронно-обчислювальних машин : вебсайт. URL: https://zakon.rada.gov.ua/laws/show/z0293-10 (дата звернення: 01.06.2026).

19) Жидецький В.Ц. Охорона праці користувачів комп'ютерів : навч. посібник. Львів : Афіша, 2000. 176 с. (дата звернення: 01.06.2026).

ДОДАТКИ

Додаток А Лістинг файлу «entity/Vacancy.java»

```
package com.employmentcenter.entity;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

import java.time.LocalDate;

/**
 * Entity для вакансії
 * Зберігає інформацію про робочі місця від роботодавців
 */
@Entity
@Table(name = "vacancy")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(of = "id")
@ToString(exclude = "employer")
public class Vacancy {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @NotNull(message = "Position cannot be null")
    @Size(min = 2, max = 255, message = "Position must be between 2 and 255
characters")
    @Column(nullable = false)
    private String position; // Посада

    @NotNull(message = "Salary cannot be null")
    @Positive(message = "Salary must be positive")
    @Column(nullable = false)
    private Double salary; // Зарплата

    @Column(columnDefinition = "TEXT")
    private String requirements; // Вимоги

    @Column(columnDefinition = "TEXT")
    private String description; // Опис

    @NotNull(message = "Status cannot be null")
    @Pattern(regexp = "open|closed", message = "Status must be: open or
closed")
    @Column(nullable = false)
    private String status; // Статус: "open", "closed"

    @NotNull(message = "Posted date cannot be null")
    @PastOrPresent(message = "Posted date cannot be in the future")
    @Column(name = "posted_date", nullable = false)
    private LocalDate postedDate; // Дата публікації

    @NotNull(message = "Employer cannot be null")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "employer_id", nullable = false)
    private Employer employer; // Зв'язок з роботодавцем
}
```

Додаток Б Лістинг файлу «entity/JobApplication.java»

```
package com.employmentcenter.entity;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

import java.time.LocalDate;
import java.time.LocalDateTime;

/**
 * Entity для заявки на вакансію.
 *
 * Lifecycle: pending → reviewed → interview_scheduled → interview_passed →
 *            offer → accepted | rejected | withdrawn.
 */
@Entity
@Table(name = "job_application")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(of = "id")
@ToString(exclude = {"unemployed", "vacancy"})
public class JobApplication {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;

    @NotNull(message = "Unemployed cannot be null")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "unemployed_id", nullable = false)
    private Unemployed unemployed;

    @NotNull(message = "Vacancy cannot be null")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "vacancy_id", nullable = false)
    private Vacancy vacancy;

    @NotNull(message = "Application date cannot be null")
    @PastOrPresent(message = "Application date cannot be in the future")
    @Column(name = "application_date", nullable = false)
    private LocalDate applicationDate;

    @NotNull(message = "Status cannot be null")
    @Pattern(
        regexp                                                              =
"pending|reviewed|interview_scheduled|interview_passed|offer|accepted|rejec
ted|withdrawn",
        message = "Invalid status"
    )
    @Column(nullable = false)
    private String status;

    @Column(name = "cover_letter", columnDefinition = "TEXT")
    private String coverLetter;

    @Column(name = "interview_date")
    private LocalDateTime interviewDate;

    @Column(name = "interview_type", length = 20)
```

```
    private String interviewType;

    @Column(name = "interview_link", length = 500)
    private String interviewLink;

    @Column(name = "offer_salary")
    private Double offerSalary;

    @Column(name = "offer_start_date")
    private LocalDate offerStartDate;

    @Column(name = "offer_reply_by")
    private LocalDate offerReplyBy;

    @Column(columnDefinition = "TEXT")
    private String notes;

    @Column(name = "updated_at")
    private LocalDateTime updatedAt;

    @PrePersist
    @PreUpdate
    void touchUpdatedAt() {
        this.updatedAt = LocalDateTime.now();
    }
}
```

Додаток В Лістинг файлу «security/JwtService.java»

```
package com.employmentcenter.security;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Service;

import javax.crypto.SecretKey;
import java.nio.charset.StandardCharsets;
import java.util.Date;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtService {

    @Value("${app.jwt.secret}")
    private String secretKey;

    @Value("${app.jwt.expiration:86400000}")
    private long expiration; // default 24 hours

    public String generateToken(UserDetails userDetails) {
        return generateToken(Map.of(), userDetails);
    }

    public String generateToken(Map<String, Object> extraClaims, UserDetails
userDetails) {
        return Jwts.builder()
                .claims(extraClaims)
```

```
                .subject(userDetails.getUsername())
                .issuedAt(new Date())
                .expiration(new      Date(System.currentTimeMillis()      +
expiration))
                .signWith(getSigningKey())
                .compact();
    }
    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }
    public boolean isTokenValid(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return           username           !=           null           &&
username.equals(userDetails.getUsername()) && !isTokenExpired(token);
    }
    private boolean isTokenExpired(String token) {
        return    extractClaim(token,    Claims::getExpiration).before(new
Date());
    }
    private   <T>   T   extractClaim(String   token,   Function<Claims,   T>
claimsResolver) {
        final Claims claims = Jwts.parser()
                .verifyWith(getSigningKey())
                .build()
                .parseSignedClaims(token)
                .getPayload();
        return claimsResolver.apply(claims);
    }
    private SecretKey getSigningKey() {
        byte[] keyBytes = secretKey.getBytes(StandardCharsets.UTF_8);
        return Keys.hmacShaKeyFor(keyBytes);
    }
}
```

Додаток Г Лістинг файлу «security/JwtAuthenticationFilter.java»

```
package com.employmentcenter.security;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.lang.NonNull;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationT
oken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSou
rce;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;
import java.time.Instant;
@Slf4j
@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final JwtService jwtService;
    private final UserDetailsService userDetailsService;
    @Override
    protected void doFilterInternal(
            @NonNull HttpServletRequest request,
            @NonNull HttpServletResponse response,
            @NonNull FilterChain filterChain
```

```
    ) throws ServletException, IOException {
        final String authHeader = request.getHeader("Authorization");
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }
        try {
            final String jwt = authHeader.substring(7);
            final String username = jwtService.extractUsername(jwt);
            if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
                UserDetails userDetails;
                try {
                    userDetails =
userDetailsService.loadUserByUsername(username);
                } catch
(org.springframework.security.core.userdetails.UsernameNotFoundException e)
{
                    log.warn("JWT references a non-existent user: {}",
username);
                    writeJsonError(response,
HttpServletResponse.SC_UNAUTHORIZED,
                            "TOKEN_EXPIRED", "Сесія завершилась, увійдіть
знову");
                    return;
                }

                if (jwtService.isTokenValid(jwt, userDetails)) {
                    UsernamePasswordAuthenticationToken authToken =
                            new UsernamePasswordAuthenticationToken(
                                    userDetails,
                                    null,
                                    userDetails.getAuthorities()
                            );
```

```
                    authToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authToken);
                    log.debug("Authenticated user: {}", username);
                }
            }
        } catch (io.jsonwebtoken.ExpiredJwtException e) {
            log.warn("JWT token is expired: {}", e.getMessage());
            writeJsonError(response, HttpServletResponse.SC_UNAUTHORIZED,
                    "TOKEN_EXPIRED", "Сесія завершилась, увійдіть знову");
            return;
        } catch (io.jsonwebtoken.JwtException e) {
            log.warn("JWT authentication failed: {}", e.getMessage());
            writeJsonError(response, HttpServletResponse.SC_UNAUTHORIZED,
                    "INVALID_TOKEN", "Невалідний токен авторизації");
            return;
        }
        filterChain.doFilter(request, response);
    }
    private void writeJsonError(HttpServletResponse response, int status,
String code, String message) throws IOException {
        response.setStatus(status);
        response.setContentType("application/json;charset=UTF-8");
        String body = String.format(
"{\"status\":%d,\"error\":\"Unauthorized\",\"code\":\"%s\",\"message\":\"%s
\",\"timestamp\":\"%s\"}",
                status, code, escapeJson(message), Instant.now());
        response.getWriter().write(body);    }
    private String escapeJson(String s) {
        return s == null ? "" : s.replace("\\", "\\\\").replace("\"",
"\\\"");
    }
}
```

Додаток Д Лістинг файлу «dto/auth/UnemployedRegisterRequest.java»

```
package com.employmentcenter.dto.auth;

import io.swagger.v3.oas.annotations.media.Schema;
import jakarta.validation.constraints.*;

@Schema(description = "Запит на реєстрацію безробітного")
public record UnemployedRegisterRequest(
        @Schema(description = "Ім'я користувача (логін)", example =
"unemployed1")
        @NotBlank @Size(min = 3, max = 50) String username,

        @Schema(description = "Email користувача", example =
"user@example.com")
        @NotBlank @Email String email,

        @Schema(description = "Пароль (мінімум 8 символів)", example =
"password123")
        @NotBlank @Size(min = 8, max = 100) String password,

        @Schema(description = "Повне ім'я", example = "Іванов Іван Іванович")
        @NotBlank String fullName,

        @Schema(description = "Номер паспорта", example = "АА123456")
        @NotBlank @Pattern(regexp = "^(\\d{9}|[А-ЯІЇЄҐ]{2}\\d{6})$") String
passportNumber,

        @Schema(description = "Адреса проживання", example = "м. Київ, вул.
Хрещатик, 1")
        @NotBlank @Size(min = 2, max = 500) String address,

        @Schema(description = "Номер телефону", example = "+380501234567")
        @NotBlank @Pattern(regexp = "^\\+?380\\d{9}$") String phone,
```

```
        @Schema(description = "Освіта", example = "Вища")
        @NotBlank @Size(min = 2, max = 255) String education,

        @Schema(description = "Досвід роботи", example = "5 років в IT")
        String workExperience,

        @Schema(description = "Метод відновлення доступу: 'question' або
'codes' (опційно)", example = "question")
        String recoveryMethod,

        @Schema(description = "Контрольне питання (якщо
recoveryMethod=question)", example = "Дівоче прізвище матері")
        String securityQuestion,

        @Schema(description = "Відповідь на контрольне питання (plain text,
буде захешована)", example = "Іваненко")
        String securityAnswer
) {}
```

Додаток Е Лістинг файлу «specification/VacancySpecification.java»

```
package com.employmentcenter.specification;

import com.employmentcenter.entity.Vacancy;
import org.springframework.data.jpa.domain.Specification;

/**
 * Specification для динамічної фільтрації вакансій
 */
public class VacancySpecification {

    public static Specification<Vacancy> hasPosition(String position) {
        return (root, query, cb) ->
                position == null ? null :
                        cb.like(cb.lower(root.get("position")),    "%"    +
position.toLowerCase() + "%");
    }

    public static Specification<Vacancy> hasStatus(String status) {
        return (root, query, cb) ->
                status == null ? null :
                        cb.equal(root.get("status"), status);
    }

    public static Specification<Vacancy> salaryGreaterThan(Double minSalary)
{
        return (root, query, cb) ->
                minSalary == null ? null :
                        cb.greaterThanOrEqualTo(root.get("salary"),
minSalary);
    }

    public static Specification<Vacancy> salaryLessThan(Double maxSalary) {
```

```
        return (root, query, cb) ->
                maxSalary == null ? null :
                        cb.lessThanOrEqualTo(root.get("salary"),
maxSalary);
    }

    public static Specification<Vacancy> cityContains(String city) {
        return (root, query, cb) ->
                city == null || city.isBlank() ? null :

cb.like(cb.lower(root.get("employer").get("address")),         "%"         +
city.toLowerCase() + "%");
    }

    public static Specification<Vacancy> industryContains(String industry)
{
        return (root, query, cb) ->
                industry == null || industry.isBlank() ? null :

cb.like(cb.lower(root.get("employer").get("industry")),         "%"         +
industry.toLowerCase() + "%");
    }

    public static Specification<Vacancy> employerId(Long employerId) {
        return (root, query, cb) ->
                employerId == null ? null :
                        cb.equal(root.get("employer").get("id"),
employerId);
    }
}
```

Додаток Ж Лістинг файлу «lib/stores/auth.ts»

```
import { writable, derived } from 'svelte/store';

interface AuthState {
    token: string | null;
    username: string | null;
    role: string | null;
    entityId: number | null;
}

function getInitialState(): AuthState {
    if (typeof window !== 'undefined') {
        const token = localStorage.getItem('token');
        const username = localStorage.getItem('username');
        const role = localStorage.getItem('role');
        const entityIdRaw = localStorage.getItem('entityId');
        const entityId = entityIdRaw ? Number(entityIdRaw) : null;
        if (token) return { token, username, role, entityId };
    }
    return { token: null, username: null, role: null, entityId: null };
}

export const auth = writable<AuthState>(getInitialState());

export const isAuthenticated = derived(auth, ($auth) => !!$auth.token);
export const isAdmin = derived(auth, ($auth) => $auth.role === 'ADMIN');
export const isEmployer = derived(auth, ($auth) => $auth.role ===
'EMPLOYER');
export const isUnemployed = derived(auth, ($auth) => $auth.role ===
'UNEMPLOYED');
export const token = derived(auth, ($auth) => $auth.token);

export function setAuth(data: { token: string; username: string; role:
```

```
string; entityId?: number | null }) {
    localStorage.setItem('token', data.token);
    localStorage.setItem('username', data.username);
    localStorage.setItem('role', data.role);
    if (data.entityId != null) {
        localStorage.setItem('entityId', String(data.entityId));
    } else {
        localStorage.removeItem('entityId');
    }
    auth.set({ token: data.token, username: data.username, role: data.role,
entityId: data.entityId ?? null });
}

export function clearAuth() {
    localStorage.removeItem('token');
    localStorage.removeItem('username');
    localStorage.removeItem('role');
    localStorage.removeItem('entityId');
    auth.set({ token: null, username: null, role: null, entityId: null });
}
```

Додаток И Лістинг файлу «lib/api/client.ts»

```
import { browser } from '$app/environment';
import { goto } from '$app/navigation';
import { clearAuth } from '$lib/stores/auth';

// Dev: '/api' (Vite proxy to localhost:8080). Prod: full backend URL via
env.
const API_BASE = import.meta.env.VITE_API_URL ?? '/api';

interface ApiOptions {
    method?: string;
    body?: unknown;
    token?: string;
    params?: Record<string, string | number | undefined>;
}

interface ApiError {
    status: number;
    message: string;
    code?: string;
    errors?: Record<string, string>;
}

export class ApiClientError extends Error {
    status: number;
    code?: string;
    errors?: Record<string, string>;

    constructor(error: ApiError) {
        super(error.message);
        this.status = error.status;
        this.code = error.code;
        this.errors = error.errors;
```

```
    }
}
let expiredHandled = false;
function handleExpiredToken() {
    if (expiredHandled || !browser) return;
    expiredHandled = true;
    clearAuth();
    const next = window.location.pathname + window.location.search;
    const params = new URLSearchParams({ expired: '1' });
    if (next && next !== '/' && !next.startsWith('/login'))
params.set('next', next);
    goto(`/login?${params.toString()}`);
    // Let next navigation reset the guard.
    setTimeout(() => { expiredHandled = false; }, 1000);
}
export async function api<T>(endpoint: string, options: ApiOptions = {}):
Promise<T> {
    const { method = 'GET', body, token, params } = options;

    let url = `${API_BASE}${endpoint}`;

    if (params) {
        const searchParams = new URLSearchParams();
        for (const [key, value] of Object.entries(params)) {
            if (value !== undefined) {
                searchParams.set(key, String(value));
            }
        }
        const qs = searchParams.toString();
        if (qs) url += `?${qs}`;
    }

    const headers: Record<string, string> = {
        'Content-Type': 'application/json'
```

```
    };
    if (token) {
        headers['Authorization'] = `Bearer ${token}`;
    }
    const res = await fetch(url, {
        method,
        headers,
        body: body ? JSON.stringify(body) : undefined
    });

    if (!res.ok) {
        let errorData: ApiError;
        try {
            const json = await res.json();
            errorData = {
                status: res.status,
                code: json.code,
                message: json.message || json.error || res.statusText,
                errors: json.errors
            };
        } catch {
            errorData = {
                status: res.status,
                message: res.statusText
            };
        }
        if (res.status === 401 && errorData.code === 'TOKEN_EXPIRED') {
            handleExpiredToken();
        }
        throw new ApiClientError(errorData);
    }
    if (res.status === 204) return undefined as T;
    return res.json();
}
```

Додаток К Лістинг файлу «exception/GlobalExceptionHandler.java»

```
package com.employmentcenter.exception;

import com.employmentcenter.dto.ErrorResponse;
import lombok.extern.slf4j.Slf4j;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import jakarta.validation.ConstraintViolationException;
import org.springframework.data.mapping.PropertyReferenceException;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.core.AuthenticationException;

import java.util.HashMap;
import java.util.Map;

/**
 * Глобальний обробник виключень
 */
@Slf4j
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<ErrorResponse> handleValidationExceptions(
            MethodArgumentNotValidException ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getAllErrors().forEach((error) -> {
```

```
            String fieldName = ((FieldError) error).getField();
            String errorMessage = error.getDefaultMessage();
            errors.put(fieldName, errorMessage);
        });

        ErrorResponse response = new ErrorResponse(
            HttpStatus.BAD_REQUEST.value(),
            "Validation Failed",
            "Invalid input data",
            errors
        );

        log.warn("Validation error: {}", errors);
        return ResponseEntity.badRequest().body(response);
    }

    @ExceptionHandler(ConstraintViolationException.class)
    public ResponseEntity<ErrorResponse> handleConstraintViolation(
            ConstraintViolationException ex) {
        Map<String, String> errors = new HashMap<>();
        ex.getConstraintViolations().forEach(violation -> {
            String paramName = violation.getPropertyPath().toString();
            //     Extract     just     the     parameter     name     (e.g.,
"getAllUnemployed.page" -> "page")
            if (paramName.contains(".")) {
                paramName  =  paramName.substring(paramName.lastIndexOf('.')
+ 1);
            }
            errors.put(paramName, violation.getMessage());
        });
        ErrorResponse response = new ErrorResponse(
            HttpStatus.BAD_REQUEST.value(),
            "Validation Failed",
            "Invalid request parameters",
```

```
            errors
        );

        log.warn("Constraint violation: {}", errors);
        return ResponseEntity.badRequest().body(response);
    }

    @ExceptionHandler(DuplicateApplicationException.class)
    public ResponseEntity<ErrorResponse> handleDuplicateApplication(
            DuplicateApplicationException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.CONFLICT.value(),
            "Conflict",
            ex.getMessage()
        );

        log.warn("Duplicate application: {}", ex.getMessage());
        return ResponseEntity.status(HttpStatus.CONFLICT).body(response);
    }

    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity<ErrorResponse> handleEntityNotFound(
            EntityNotFoundException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.NOT_FOUND.value(),
            "Not Found",
            ex.getMessage()
        );
        log.warn("Entity not found: {}", ex.getMessage());
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(response);
    }
    @ExceptionHandler(DataIntegrityViolationException.class)
    public ResponseEntity<ErrorResponse> handleDataIntegrityViolation(
            DataIntegrityViolationException ex) {
```

```
        String message = "Database constraint violation";
        if (ex.getMessage() != null &&
ex.getMessage().contains("passport_number")) {
            message = "Passport number already exists";
        }

        ErrorResponse response = new ErrorResponse(
            HttpStatus.CONFLICT.value(),
            "Conflict",
            message
        );

        log.warn("Data integrity violation: {}", message);
        return ResponseEntity.status(HttpStatus.CONFLICT).body(response);
    }

    @ExceptionHandler(AccessDeniedException.class)
    public ResponseEntity<ErrorResponse> handleAccessDenied(
            AccessDeniedException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.FORBIDDEN.value(),
            "Forbidden",
            "Insufficient permissions"
        );
        log.warn("Access denied: {}", ex.getMessage());
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(response);
    }
    @ExceptionHandler(AuthenticationException.class)
    public ResponseEntity<ErrorResponse> handleAuthentication(
            AuthenticationException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.UNAUTHORIZED.value(),
            "Unauthorized",
            "Authentication required"
```

```
        );
        log.warn("Authentication failed: {}", ex.getMessage());
        return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(response);
    }
    @ExceptionHandler(PropertyReferenceException.class)
    public ResponseEntity<ErrorResponse> handlePropertyReference(
            PropertyReferenceException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.BAD_REQUEST.value(),
            "Bad Request",
            "Invalid sort field: " + ex.getPropertyName()
        );

        log.warn("Invalid property reference: {}", ex.getPropertyName());
        return ResponseEntity.badRequest().body(response);
    }

    @ExceptionHandler(IllegalStateException.class)
    public ResponseEntity<ErrorResponse> handleIllegalState(
            IllegalStateException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.BAD_REQUEST.value(),
            "Bad Request",
            ex.getMessage()
        );

        log.warn("Illegal state: {}", ex.getMessage());
        return ResponseEntity.badRequest().body(response);
    }
    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<ErrorResponse> handleIllegalArgument(
            IllegalArgumentException ex) {
        ErrorResponse response = new ErrorResponse(
```

```
            HttpStatus.BAD_REQUEST.value(),
            "Bad Request",
            ex.getMessage()
        );
        log.warn("Illegal argument: {}", ex.getMessage());
        return ResponseEntity.badRequest().body(response);
    }
    @ExceptionHandler(RuntimeException.class)
    public ResponseEntity<ErrorResponse> handleRuntimeException(
            RuntimeException ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.INTERNAL_SERVER_ERROR.value(),
            "Internal Server Error",
            "An unexpected error occurred"
        );
        log.error("Unhandled runtime exception: {}", ex.getMessage(), ex);
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(response);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorResponse> handleGenericException(
            Exception ex) {
        ErrorResponse response = new ErrorResponse(
            HttpStatus.INTERNAL_SERVER_ERROR.value(),
            "Internal Server Error",
            "An unexpected error occurred"
        );

        log.error("Unexpected error", ex);
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(response);
    }
}
```

Додаток Л Лістинг файлу «security/LoginRateLimitFilter.java»

```
package com.employmentcenter.security;

import com.github.benmanes.caffeine.cache.Cache;
import com.github.benmanes.caffeine.cache.Caffeine;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.lang.NonNull;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;
import java.time.Duration;
import java.time.Instant;
import java.util.concurrent.atomic.AtomicInteger;
@Slf4j
@Component
public class LoginRateLimitFilter extends OncePerRequestFilter {

    @Value("${security.login-rate-limit.max-attempts:5}")
    private int maxAttempts;

    @Value("${security.login-rate-limit.trust-forwarded-only:false}")
    private boolean trustForwardedOnly;

    private static final Duration WINDOW = Duration.ofMinutes(5);

    private final Cache<String, AtomicInteger> attempts =
Caffeine.newBuilder()
```

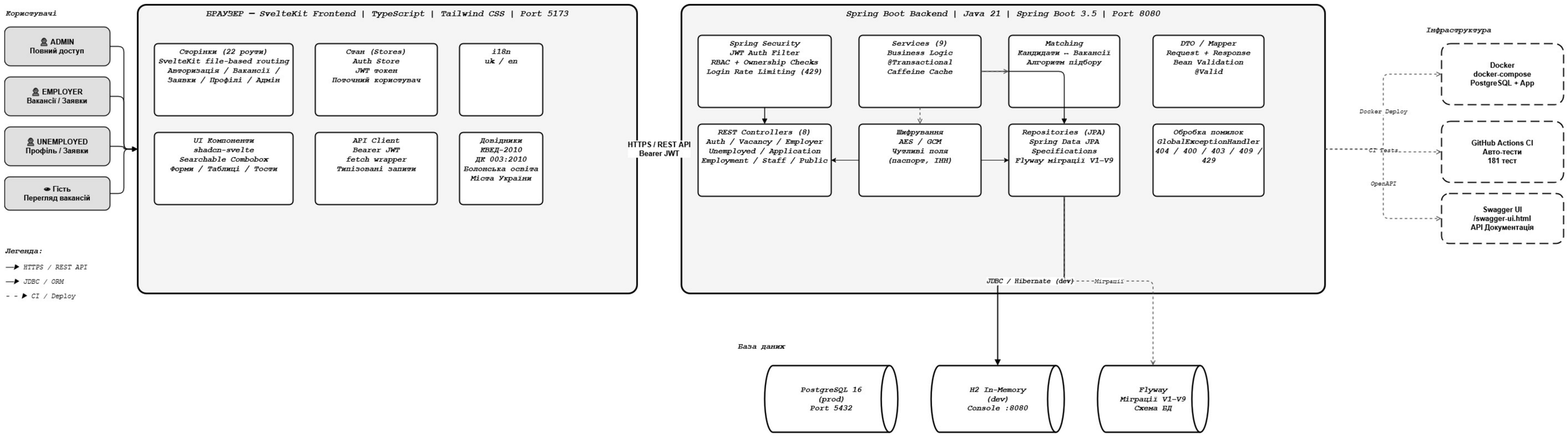
```
            .expireAfterWrite(WINDOW)
            .maximumSize(10_000)
            .build();

    @Override
    protected void doFilterInternal(@NonNull HttpServletRequest request,
                                    @NonNull HttpServletResponse response,
                                    @NonNull FilterChain chain)
            throws ServletException, IOException {

        if (!isLoginAttempt(request)) {
            chain.doFilter(request, response);
            return;
        }

        // Dev/E2E convenience: when trust-forwarded-only=true, requests without
        // an explicit X-Forwarded-For header are exempt. Lets local tooling
        // make many logins from the loopback interface while still enforcing
        // the limit for spoofed/forwarded IPs (covered by the dedicated test).
        if (trustForwardedOnly) {
            String forwarded = request.getHeader("X-Forwarded-For");
            if (forwarded == null || forwarded.isBlank()) {
                chain.doFilter(request, response);
                return;
            }
        }

        String ip = clientIp(request);
        AtomicInteger counter = attempts.get(ip, k -> new AtomicInteger());
        int current = counter.incrementAndGet();

        if (current > maxAttempts) {
```

```
            log.warn("Login rate limit exceeded for IP {}: {} attempts in
window", ip, current);
            response.setStatus(429);
            response.setContentType("application/json;charset=UTF-8");
            response.getWriter().write(String.format(
                    "{\"status\":429,\"error\":\"Too                    Many
Requests\",\"code\":\"RATE_LIMIT\","
                            + "\"message\":\"Забагато спроб входу. Спробуйте
через кілька хвилин.\","
                            + "\"timestamp\":\"%s\"}",
                    Instant.now()));
            return;
        }

        chain.doFilter(request, response);
    }

    private boolean isLoginAttempt(HttpServletRequest request) {
        return "POST".equalsIgnoreCase(request.getMethod())
                && "/api/auth/login".equals(request.getRequestURI());
    }

    private String clientIp(HttpServletRequest request) {
        String forwarded = request.getHeader("X-Forwarded-For");
        if (forwarded != null && !forwarded.isBlank()) {
            return forwarded.split(",")[0].trim();
        }
        return request.getRemoteAddr();
    }
}
```

Таблиця техніко-економічних показників

№ п/п	Показник	Одиниці вимірювання	Значення
1	Мова програмування	–	Java 21, TypeScript 5, JavaScript (ES6+)
2	Інтерфейс	–	SvelteKit, Tailwind CSS
3	Інструменти збірки та публікації	–	Apache Maven, npm, Vite, Flyway
4	Бібліотеки та фреймворки	–	Spring Boot, Spring Security, Hibernate, Chart.js, JUnit 5, Mockito, Playwright
5	Середовище програмування	–	IntelliJ IDEA, Visual Studio Code
6	СУБД	–	PostgreSQL 16
7	Собівартість	грн.	72 143
8	Плановий прибуток або грошовий потік	грн.	40 400
9	Ціна	грн.	112 543
10	Чиста теперішня вартість	грн.	28 325
11	Термін окупності	рік	2,05

Схема структурна взаємодії компонентів програмного забезпечення



Зм.	Арк.	№ докум.	Підп.	Дата	2026.КВР.122.421.11.00.00 СС			
Розроб.		Кощур О. Ф.			Розробка інформаційної системи Центру зайнятості	Літ.	Маса	Масштаб
Перев.		Сердін В. С.			Схема структурна взаємодії компонентів програмного забезпечення			
Т.контр.						Аркуш	Аркушів 1	
Реценз.						ВСП ТФК ТНТУ КН-421		
Н.контр.		Прийомак В.А.				м. Тернопіль		
Затв.								

Копіював

Формат А1

Перш. застос.

Справ. №

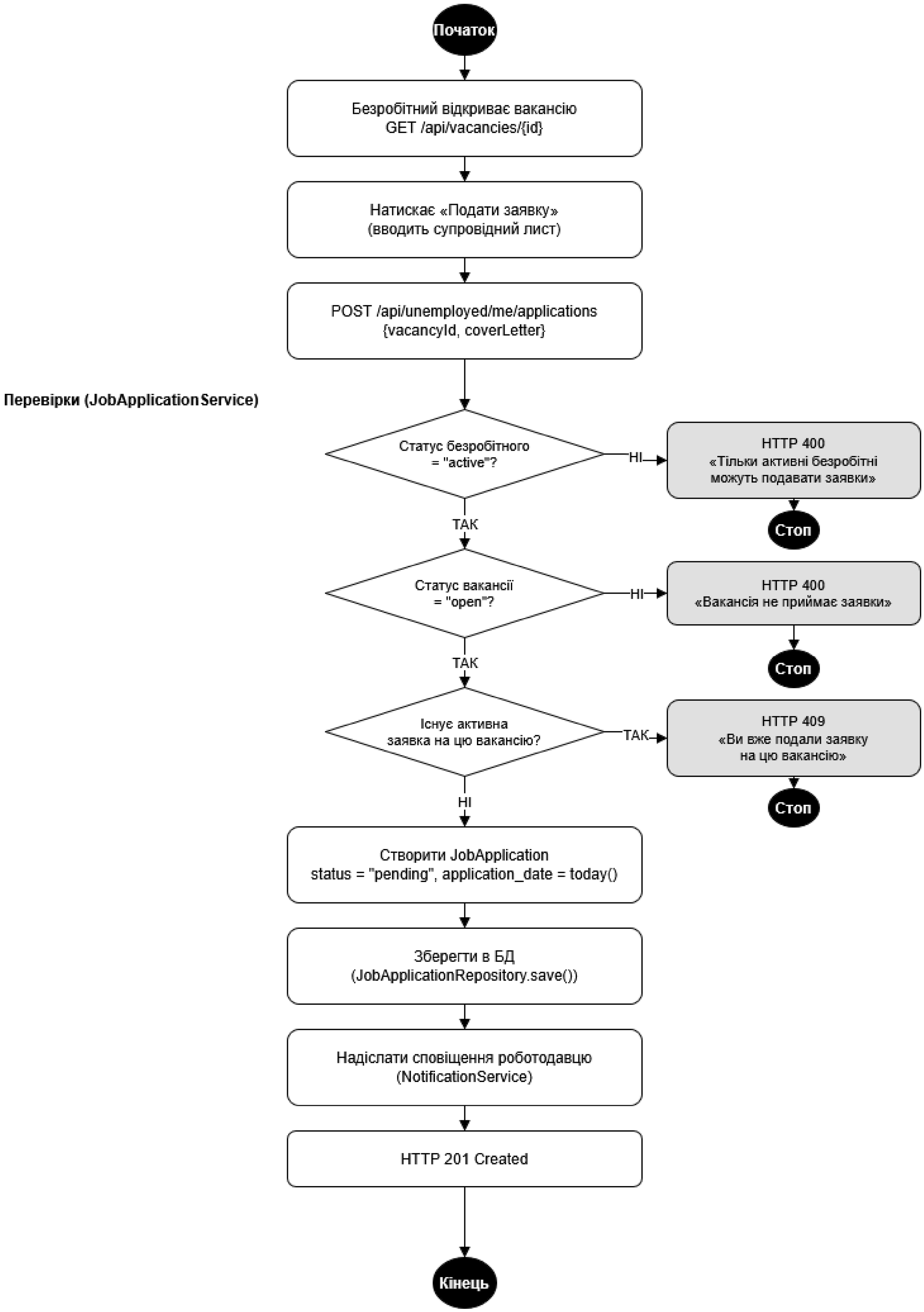
Інв. № дубл.

Взам. інв. №

Підп. і дата

Інв. № подл.

Блок-схема процесу подання заявки на вакансію



Зм.	Арк.	№ докум.	Підп.	Дата	2026.КВР.122.421.11.00.00 БС	
Розроб.		Коцур О. Ф.			Розробка інформаційної системи Центру зайнятості Блок-схема процесу подання заявки на вакансію	Літ. / Маса / Масштаб
Перев.		Сердін В. С.				Аркуш / Аркушів 1
Т.контр.						ВСП ТФК ТНТУ КН-421 м. Тернопіль
Рецензент						
Н.контр.		Приймак В.А.				
Затв.						

Копіював Формат А1

Перш. застос.
Справ. №
Підп. і дата
Інв. № дубл.
Зам. інв. №
Підп. і дата
Інв. № ориг.

Діаграма класів застосунку

Сервісний шар

```
«service»
AuthService
---------------
+ registerEmployer()
+ registerUnemployed()
+ login()
+ forgotPassword()
```

```
«service»
VacancyService
---------------
+ createVacancy()
+ updateVacancy()
+ deleteVacancy()
+ search()
```

```
«service»
UnemployedService
---------------
+ createUnemployed()
+ updateUnemployed()
+ deleteUnemployed()
+ getAll()
```

```
«service»
EmployerService
---------------
+ createEmployer()
+ updateEmployer()
+ deleteEmployer()
+ findByInn()
```

```
«service»
JobApplicationService
---------------
+ createApplication()
+ transitionStatus()
+ deleteApplication()
```

```
«service»
EmploymentService
---------------
+ createEmployment()
+ terminateEmployment()
+ getStats()
```

```
«service»
MatchingService
---------------
+ findMatchingVacancies()
+ findMatchingCandidates()
+ scoreSkills()
```

```
«service»
NotificationService
---------------
+ notify()
+ findForUser()
+ markRead()
```

Доменна модель (JPA Entities)

```
«entity»
Employer
---------------
- id: Long
- companyName: String
- industry: String
- phone: String
- email: String
- address: String
- inn: String
```

```
«entity»
User
---------------
- id: Long
- username: String
- email: String
- password: String
- role: Role
- enabled: boolean
- createdAt: LocalDateTime
---------------
enum Role: USER | ADMIN
EMPLOYER | UNEMPLOYED
```

```
«entity»
JobApplication
---------------
- id: Long
- applicationDate: LocalDate
- status: String
- coverLetter: String
- interviewDate: LocalDateTime
- offerSalary: Double
- updatedAt: LocalDateTime
```

```
«entity»
AuditLog
---------------
- id: Long
- timestamp: LocalDateTime
- actorUsername: String
- actionType: String
- entityType: String
- ipAddress: String
```

```
«entity»
Staff
---------------
- id: Long
- fullName: String
- position: String
- phone: String
- salary: Double
```

```
«entity»
Vacancy
---------------
- id: Long
- position: String
- salary: Double
- requirements: String
- status: String {open|closed}
- postedDate: LocalDate
```

```
«entity»
Unemployed
---------------
- id: Long
- fullName: String
- passportNumber: String
- education: String
- skills: String
- desiredSalary: Double
- status: String
- registrationDate: LocalDate
```

```
«entity»
Notification
---------------
- id: Long
- type: String
- title: String
- body: String
- readAt: LocalDateTime
- createdAt: LocalDateTime
```

```
«entity»
Employment
---------------
- id: Long
- employmentDate: LocalDate
- actualSalary: Double
- endDate: LocalDate
- terminationReason: String
- notes: String
```