Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії)

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Розробка методу та засобів обробки міжкорпоративних даних

Виконав: студент II курсу,
групи 2ПІ-18м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Тяпкін О. А.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Черноволик Г. О.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН Арсенюк І. Р.
(прізвище та ініціали)

ВНТУ – 2019
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень - магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
“____”_____2019 року

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

Тяпкіну Олександру Андрійовичу

1. Тема роботи – розробка методу та засобів обробки міжкорпоративних даних.

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від “___”_____2019 року №___

2. Строк подання студентом роботи
_____

3. Вихідні дані до роботи: Операційна система – Windows 10
Середовище розробки – Visual Studio
Мови програмування – C#, SQL

4. Зміст розрахунково-пояснювальної записки: аналіз предметної галузі та постановка задач дослідження; розробка методу прогнозованої оцінки, моделі сутностей і сховища даних; розробка програмних засобів додатку; тестування роботи програмного додатку; економічна частина.

5. Перелік графічного матеріалу: тема, автор, науковий керівник магістерської кваліфікаційної роботи; мета, об’єкт та предмет дослідження; аналіз предметної області; порівняльний аналіз аналогів; структура програмного додатку; алгоритм програмного додатку; обґрунтування вибору програмного середовища розробки; обґрунтування вибору мови програмування; результати роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Черноволик Г. О., к.т.н, доцент кафедри ПЗ		
5	Бальзан М. В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	07.10.2018 – 12.10.2019	Вик.
2	Аналіз методів та засобів розробки сховищ даних	13.10.2018 – 16.10.2019	Вик.
3	Розробка методу прогнозованої оцінки кандидата для конкретної вакансії	17.10.2018 – 05.11.2019	Вик.
4	Розробка сховища даних обліку вакансій і кадрів	06.11.2019 – 16.11.2019	Вик.
5	Розробка програмного додатку обліку та рекомендування вакансій та кадрів	17.11.2019 – 25.11.2019	Вик.
6	Тестування роботи програмного додатку	26.12.2019 – 30.12.2019	Вик.
7	Економічна частина	01.12.2019 – 08.12.2019	Вик.

Студент _____ **___Тяпкін О. А.__**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Черноволик Г. О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська кваліфікаційна робота присвячена розробці програмного додатку «V-Staff Advice» для обліку та рекомендування кадрів та відкритих вакансій. Програмний модуль виконує основні функції: відображення списку відкритих вакансій та статусу агентів (як вільних, так і зайнятих), пошук та фільтрація доступних вакансій та кадрів, рекомендування на основі вказаних даних найбільш сумісні для конкретного агента відкриті вакансії та підбір кандидатів (як вакантних, так і зайнятих) для конкретної вакансії.

Програмний додаток реалізований за допомогою середовища розробки − Microsoft Visual Studio 2017, мовою розробки – C#. Програмна система може працювати під керівництвом операційної системи Windows 7/8/10.

ANNOTATION

Master's qualification work is devoted to the development of a software application «V-Staff Advice» for the accounting and recommendation of the then open vacancies. The program module performs the main functions: displaying the open vacancy list and status of agents (both vacant and employed), searching and filtering available vacancies and personnel, recommendations, based on the specified data, about most compatible for the specific agent open vacancies and selection of candidates (both vacant and employed) for a specific job.

The software application is implemented using the development environment – Microsoft Visual Studio Code, the language of development – JavaScript. The software system can run Windows 7/8/10.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогоднішній день значну увагу привертає явище зміни кадрів на підприємстві та пошук нових кандидатів. Підбором персоналу займаються всі компанії. Навіть ті, в яких HR-політики, як такої, немає, а керівники займаються HRM, не приходячи до тями. Але рекрутмент є у всіх – від стартапів і квіткових кіосків до компаній-гігантів. Всім їм потрібні люди: кому постійно, кому зрідка. З підбору починали кар'єру більшість HR-ів, багато керівників час від часу проводять співбесіди. Тому ця робота нікому не здається складною [1]. Але також слід розуміти, що підбір персоналу – процес не шаблонний, він корегується відповідно до особливостей галузі, специфіки вакансії та команди, у якій потенційному працівнику доведеться працювати. Тому при співбесідах та попередній оцінці кандидата рекрутерами застосовуються різні механізми визначення відповідності кандидата розгляданій вакансії.

Підбір ведеться за перевіреною ще у 2000-х роках схемою: даємо оголошення і чекаємо зворотного зв’язку. З тих пір ринок праці сильно змінився, змінилися уявлення агентів про роботу та її пошук, виросли очікування. По-перше, на ринок праці прийшло нове покоління, у нього свої особливості. По-друге, завдяки технологіям стали доступнішими і привабливішими фріланс, свій бізнес і стартапи. Наймана робота тепер не єдиний варіант, є альтернативи. По-третє, на ринку праці майже в кожному сегменті є сильні компанії, з якими важко конкурувати за персонал. Тому кандидатів доводиться переконувати і мотивувати з самого початку.

Якщо рекрутер впродовж тривалого часу працює з обмеженим набором джерел кандидатського потоку і з часом цей набір не розширюється, існує ймовірність того, що пошук кандидатів на конкретну вакансію може зайняти значно більше часу, ніж дійсно потрібно. Як причину такого явища можна привести слідування рекрутерів стереотипу: якщо вчора з цього сайту

приходили кандидати, то прийдуть і цього разу, просто потрібно почекати. Поки вони чекають, час йде, вакансії «висять».

При цьому робота йде тільки на вхідному потоці зовнішніх кандидатів – зазвичай це вже досвідчені фахівці. Раніше це було майже єдиним способом закрити вакансію [17]. Зараз способів набагато більше, якщо їх використовувати, то підбір буде йти швидше та успішніше.

На всіх етапах підбору рекрутер все робить сам, вручну, не використовуючи допоміжний «софт». В рекрутингу багато рутинних і часозатратних процесів, наприклад, скачування резюме з робітних сайтів та їх ручного сортування за потрібними параметрами. Без використання допоміжного ПЗ подібні процеси часто можуть значно сповільнити роботу рекрутера. За словами самих рекрутерів, вони роблять вищевказані дії «на автоматі». Чи не простіше доручити роботу спеціальним автоматам, ніж перетворювати на автомат людину? Великий обсяг ручної роботи навантажує фахівця з підбору настільки сильно, що часто у нього не залишається часу на активний та ініціативний рекрутинг. Простіше кажучи, рекрутеру немає часу думати, як зробити краще, що змінити, необхідність освоєння нових технологій та методик може відходити на задній план, що може з плином часу негативно сказатись на процесі рекрутингу, так як використовувані механізми не відповідатимуть вимогам сучасності. При цьому рекрутинг стає складніше та потребує нових підходів, а вимоги до рекрутера підвищуються. І це тільки загострює проблему.

Більшість нинішніх рекрутерів раніше навчались проводити співбесіди, спостерігаючи за роботою більш досвідчених колег, потім до цього додавався власний досвід [18]. Емпіричний відбір неефективний, допускається занадто багато помилок.

На сьогодні співробітника шукають під конкретного керівника. Це породжує декілька проблем. По-перше, специфіка вакансії змушує рекрутера у цьому випадку шукати кандидатів без конкретних критеріїв відбору, оперуючи при цьому лише наявною специфікою керівництва, що може значно

сповільнювати процес та підвищує ризик у результати відібрати кандидата, особливості якого будуть іти у розріз з реальними особливостями вакансії. По-друге, на перший погляд, підбирати співробітника під керівника логічно: їм же разом працювати [19]. Але в результаті існує ймовірність стикнутись з наступною проблемою – при зміні керівника йому може бути складно адаптуватись, так як йому або доведеться приймати існуючий ритм команди, так би мовити, «прийняти правила гри», або він почне процес реорганізації робочого процесу в команді, що іноді може призвести до необхідності пошуку нових працівників, що, звісно ж, призведе до сповільнення робочих процесів.

У зв’язку з цим актуальною є проблема організації та систематизації обліку кадрів, оскільки ринок відкритих вакансій постійно оновлюється, що тягне за собою явище частого переходу працівників між різноманітними вакансіями, підприємствами та навіть професіями. Тому досить актуальним буде розробка певного рішення, що матиме на меті усунути вказані вище проблеми та зробити процес обліку кадрів більш організованим та адаптивним до високого рівня «текучки» кадрів.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення продуктивності рекрутингових механізмів за рахунок градаційного компонування сумісних вакансій та доступних агентів.

Основними задачами дослідження є:

— провести аналіз сучасного стану ринку додатків для обліку кадрів;

— проаналізувати сучасні вимоги та структуру подібних додатків;

— удосконалити існуючий метод дискретного зважування у сфері рекрутингу;

— удосконалити існуючий метод матричного порівняння у сфері сортування даних;

— проаналізувати та розробити моделі сутностей та безпосередньо сховище даних;

— розробити програмну реалізацію на основі сформованих вимог та запропонованого методу;

— протестувати програмну реалізацію за допомогою засобів модульного тестування.

Об’єктом дослідження є процес обробки міжкорпоративних даних із застосуванням технології розробки баз даних обліку кадрів.

Предметом дослідження є методи та засоби сортування та групування даних.

Методи дослідження. У процесі досліджень застосовувались: теорія алгоритмів для формального формулювання задач розробки; статистичні методи оцінки параметрів моделей для отримання шаблону оцінюваних записів; методи формальної логіки для інтерпретації висловів зі збереженням істинних значень.

Наукова новизна отриманих результатів:

— Отримав подальшого розвитку метод дискретного зважування у сфері рекрутингу, який полягає в попередній оцінці кандидата за набором компетентних критеріїв, що дозволяє зазначати без попереднього інтерв’ю прогнозовані результати аналізу особистості кандидата;

— Отримав подальшого розвитку метод матричного порівняння у сфері сортування, який полягає у застосуванні порівняльних методик для специфічних типів записів, що дозволяє оптимізувати процес відбору кандидатів для інтерв’ю.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмні засоби навігації та отримання короткої інформації про існуючий кадровий стан на підприємстві та наявність вакантних посад, а також механізму рекомендування та підбору сумісних кандидатів на вакантну посаду і навпаки.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідались на XLVII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (21 березня 2018 року, м. Вінниця) та на Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (9-10 грудня 2019 року) [25].

Публікації. Основні результати досліджень було опубліковано у тезах доповіді.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1 Аналіз предметної галузі

На сьогоднішній день актуальним є розробка додатку для обліку кадрів з можливостями додавання та видалення записів, зручного переходу між записам, можливістю доповнювати каталог, з простим та інтуїтивно зрозумілим інтерфейсом та захистом від несанкціонованого запису.

Вигідною перевагою подібного додатку є наявність можливості таргетингового підбору кандидатів для вказаної вакансії та навпаки – підбір вакансій, що найбільше відповідають конкретному кандидати. Подібний механізм оперує набором градаційно оцінених параметрів, що зіставляються з відповідними параметрами у протилежній категорії (кандидати-вакансії і навпаки).

Створення даного додатку дасть можливість ефективно використовувати його у корпоративній діяльності. Особливо важливим стає такий додаток для вакантних агентів, які не можуть визначитись з місцем роботи, де вони хочуть працювати. З іншого боку, це також розширює інструментарій HR-працівників компаній, що зможуть практикувати нові підходи для набору персоналу та полегшити існуючі.

Подібні додатки доволі популярні і в освіті. Зібрані на одному ресурсі короткі відомості про групу об’єктів спрощують пошук інформації, необхідної користувачу [14]. А механізм аналізу даних та рекомендування подальших дій розширює можливості користувача в розрізі взаємодії з даним програмним рішенням.

Рішення даного типу дають змогу керівникам підприємств та рекрутинговим агентствам (а також й іншим користувачам) зголошувати резюме разом з коротким описом до розгляду, а після перегляду і перевірки таке зголошення затверджується модератором додатку. Категоризація резюме

базується на змісті цілого додатку, а не окремих його сторінок або ключових слів. Залежно від принципів роботи додатку резюме може бути приписане до однієї чи до кількох категорій або підкатегорій.

Існує три типи подібних рішень: глобальні – як правило, до глобальних додатків збираються ресурси з усього Інтернету, часто такі додатки є багатомовними; регіональні – ресурси, зібрані у регіональному каталозі, мають відношення до певної місцевості − області, міста, району; тематичні – тематичний каталог присвячується одній певній тематиці.

Додаток для обліку кадрів створює умови для реалізації одного з головних принципів відкритого суспільства – принципу загальної доступності інформації [12]. Як правило, він містить записи про кожного працівника чи агента і показує місце його знаходження в колекції.

Аналітична складова додатку дозволяє спеціалістам порівнювати результати роботи механізму рекомендування з результатами, отриманими при використанні традиційних рекрутингових методів.

Тому наявність продуманого додатку для обліку кадрів та вакансій суттєво впливає на швидкість та якість як пошуку роботи так і навпаки – пошуку кадрів. А наявність механізму для рекомендування підібраного набору вакансій або кадрів дозволяє зекономити час рекрутерам та вакантним агентам. Отже, є доцільним розробити програмний продукт для спрощення реалізації глобальної «текучки» кадрів.

2 Аналіз методів реалізації додатку

Основне завдання додатку обліку кадрів – надати можливість користувачу (роботодавцю або шукачу роботи) отримати необхідну, систематизовану та вичерпну інформацію про наявний стан відкритих вакансій та дані про наявних відкритих агентів, а також дозволити внутрішнім механізмам зіставити резюме конкретного агента чи групи агентів з наявною відкритою вакансією (набором вакансій). В даному випадку, подібних користувачів цікавить стислість, релевантність та вичерпність інформації, що

вони хочуть отримати в процесі пошуку. Тому усе вищевказане повинно бути враховане при створенні додатку.

Програмне рішення повинне реалізовувати рішення ряду задач серед яких можна виділити:

— роботу з даними про робочі кадри (як задіяні, так і вакантні) – додання, редагування та видалення необхідної інформації;

— можливість пошуку потенційних працівників та порівняння їх з наявними, використовуючи внутрішній механізм аналізу та оцінки за конкретними параметрами;

— надання звичайним працівникам та вільним агентам змоги працювати лише з особистою інформацією та шукати нових працівників за іменем та переглядати обмежений рівень даних про них.

— надання працівникам та агентам можливості переглядати рекомендовані, відповідно до їх резюме, відкриті вакансії;

— надання керівникам та вповноваженим працівникам HR-департаментів змоги переглядати повний обсяг інформації про робочі кадри та доповнювати характеристику обраних працівників;

— надання можливості рекрутерам порівнювати резюме рекомендованих кандидатів, використовуючи внутрішній механізм аналізу резюме у відповідності до конкретної вакансії;

— користування навігацією та фільтруванням даних, що виконуватиме відсіювання зайвої інформації за критеріями, встановленими користувачем.

Відповідно до технологій створення подібних додатків необхідно також врахувати структуру, що використовується створеним додатком з метою забезпечити задоволення всіх необхідних потреб орієнтованих категорій користувачів.

Серед них можна виділити:

— облік кадрів;

— headhunting;

— різниця доступності функціоналу;

— оцінка записів за критеріями;

— фільтри.

Інтерфейс додатку для обліку кадрів повинен бути простим та інтуїтивно зрозумілим, інформація має подаватись у розміреному вигляді, не перевантажуючи користувача додатку зайвими даними. Питання лінгвістичного забезпечення є, мабуть, найбільш проблемними і трудомісткими аспектами роботи з даними рішеннями. Лінгвістичне забезпечення розуміється як сукупність пошукових засобів: різні класифікаційні системи, словники, бази ключових слів і предметних рубрик та інше. Електронна пошукова система дозволяє в лічені секунди знаходити в каталозі потрібні дані і складати списки, кожен запис в яких забезпечений базовою необхідною інформацією про шукані об’єкти.

В той же час, каталог створений так, щоб з ним легко міг працювати як досвідчений користувач, так і новачок. Функціонал електронного каталогу повинен надавати зручні засоби навігації по електронному каталогу, розвинену систему пошуку, систему оновлення каталогу та інше. Електронний каталог повинен розроблятись з урахуванням мінімальних системних вимог, щоб програма працювала коректно на будь-якому ПК (це означає, що не варто зловживати новомодними «ненажерливими» технологіями). Якщо нам необхідно знайти якийсь документ, то ми висловлюємо свою потребу у вигляді інформаційного запиту – напівформалізованого вираження інформаційної потреби. Електронний каталог є багатофункціональним. Інформація вводиться один раз, а використовуватись може в різних цілях, замінюючи інформацію цілої системи традиційних карткових каталогів і картотек.

3 Типи розподілених інформаційних систем

Інформація — сукупність відомостей (даних), які сприймають з навколишнього середовища (вхідна інформація), видають у навколишнє середовище (вихідна інформація) або зберігають всередині певної системи.

У теорії автоматизованої обробки інформації ІС розглядають як сукупність знань, що є об'єктом нагромадження, реєстрації, передачі, збереження, оброблення [10].

3.1 Поняття інформаційної системи

Інформаційна система — система, яка організовує пам'ять і маніпулювання інформацією щодо проблемної сфери (ПС).

У широкому розумінні під визначення ІС підпадає будь-яка система обробки інформації.

Ефективність функціонування інформаційної системи (ІС) багато в чому залежить від її архітектури.

Історично першими з'явилися розподілені ІС із застосуванням файл-сервера (див. рисунок 1.1). У таких ІС по запитах користувачів файли бази даних передаються на персональні комп'ютери (ПК), де і проводиться їх обробка. Недоліком такого варіанту архітектури є висока інтенсивність передачі оброблюваних даних. Причому часто передаються надмірні дані: незалежно від того скільки записів з бази даних потрібно користувачеві, файли бази даних передаються цілком.

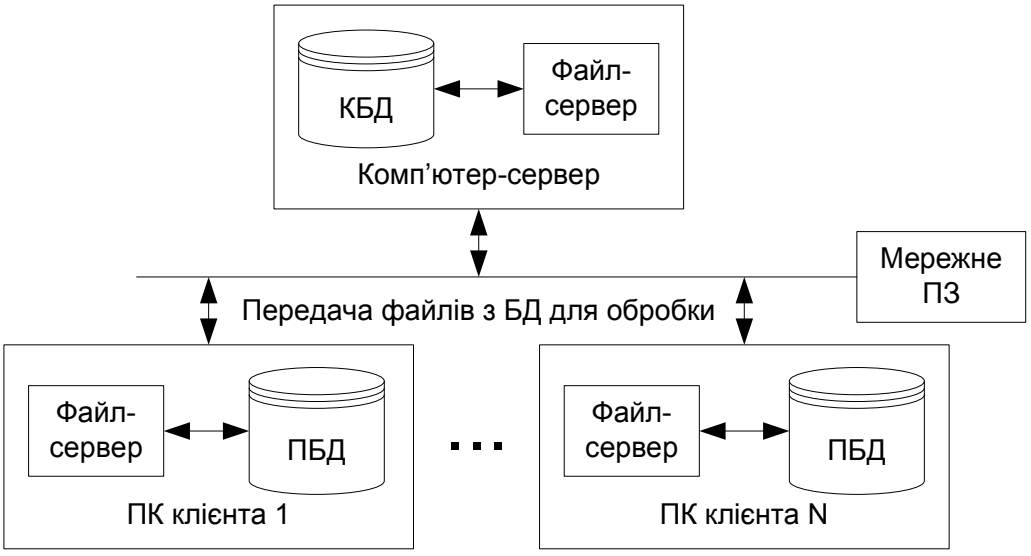


Рисунок 1.1 — Структура ІС з файл-сервером.

В даний час перспективною є архітектура клієнт-сервер. У достатньо поширеному варіанті вона припускає наявність комп'ютерної мережі і розподіленої бази даних, що включає корпоративну базу даних (КБД) і персональні бази даних (ПБД). КБД розміщується на комп'ютері-сервері, ПБД розміщуються на комп'ютерах співробітників підрозділів, що є клієнтами корпоративної БД.

Сервером певного ресурсу в комп'ютерній мережі називається комп'ютер (програма), керівник цим ресурсом, клієнтом — комп'ютер (програма), що використовує цей ресурс. Тип сервера визначається видом ресурсу, яким він управляє [13].

Наприклад, якщо керованим ресурсом є база даних, то відповідний сервер називається сервером бази даних. У якості ресурсу комп'ютерної мережі можуть виступати, наприклад, бази даних, файлові системи, служби друку, поштові служби.

Структура розподіленої ІС, побудованою по архітектурі клієнт-сервер з використанням сервера баз даних, показана на рисунку 1.2. При такій архітектурі сервер бази даних забезпечує виконання основного об'єму обробки даних. Формовані користувачем або додатком запити поступають до сервера БД у вигляді інструкцій мови SQL. Сервер бази даних виконує пошук і витягання потрібних даних, які потім передаються на комп'ютер користувача. Перевагою такого підходу в порівнянні попереднім є помітно менший об'єм даних, що передаються.

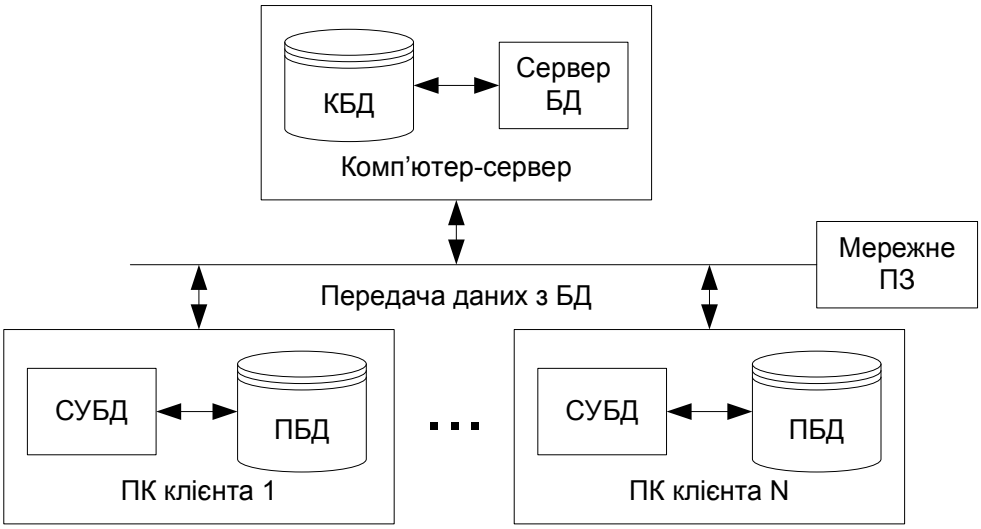


Рисунок 1.2 — Структура ІС з сервером баз даних

Перевагою організації інформаційної системи за архітектурою клієнт-сервер є вдале поєднання централізованого зберігання, обслуговування і колективного доступу до загальної корпоративної інформації з індивідуальною роботою користувачів над персональною інформацією.

3.2 Моделі архітектури клієнт-сервер

Архітектура клієнт-сервер широко використовується при побудові розподілених ІС, що працюють з БД. Її основу складають принципи організації взаємодії клієнта і серверу при управлінні БД.

Як програма, що підтримує інтерфейс з користувачем, СУБД, в загальному випадку, реалізує наступні основні функції:

— керування даними, що знаходяться в базі;

— обробку інформації за допомогою прикладних програм;

— представлення інформації в зручному для користувача вигляді.

Якщо система розміщується на одній ЕОМ, то всі функції зібрані в одній програмі і викликаються за схемою, подібною розглянутій раніше [15].

При розміщенні СУБД в мережі можливі різні варіанти розподілу функцій за вузлами. Залежно від числа вузлів мережі, між якими виконується розподіл функцій СУБД, можна виділити двохланкові моделі, трьохланкові

моделі і т.д. Місце розриву функцій з'єднується комунікаційними функціями (середовищем передачі інформації в мережі).

3.2.1 Двохланкові моделі розподілу функцій

Двохланкові моделі відповідають розподілу функцій СУБД між двома вузлами мережі. Комп'ютер (вузол мережі), на якому обов'язково присутня функція управління даними, називається комп'ютером-сервером. Комп'ютер, близький до користувача і який обов'язково займається питаннями представлення інформації, називається комп'ютером-клієнтом.

Найтиповішими варіантами (див. рисунок 1.3) розділення функцій між комп'ютером-сервером і комп'ютером-клієнтом є наступні:

— розподілене представлення;

— віддалене представлення;

— розподілена функція;

— віддалений доступ до даних;

— розподілена БД.

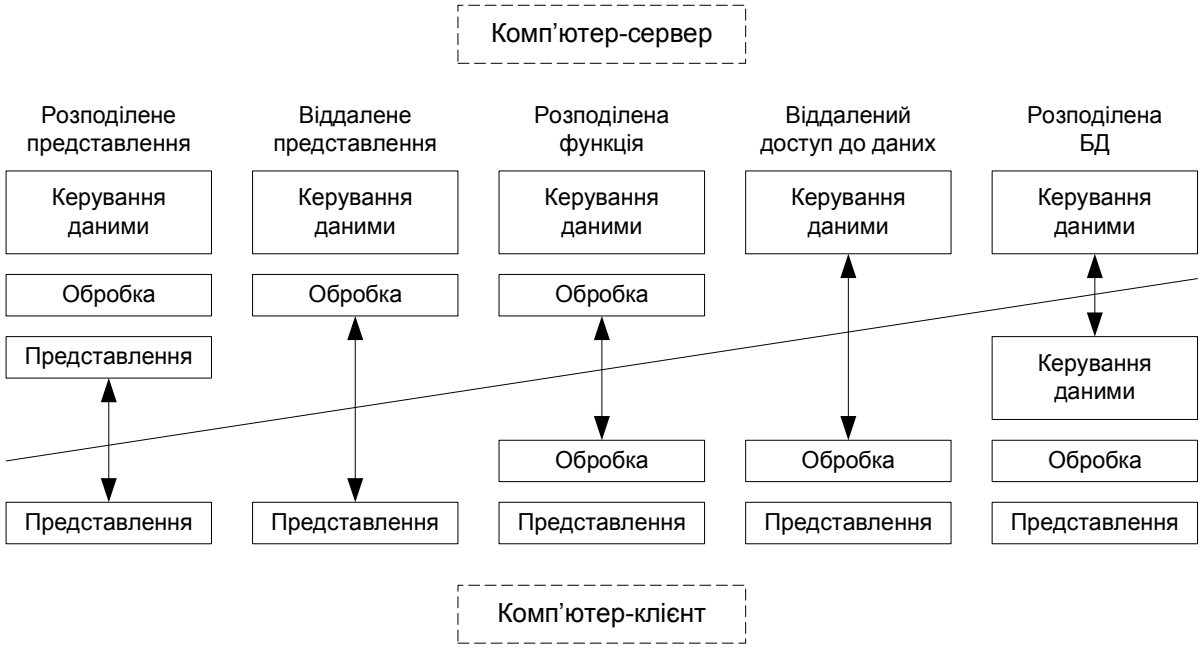


Рисунок 1.3 — Спектр моделей архітектури клієнт-сервер.

Перераховані способи розподілу функцій в системах з архітектурою клієнт-сервер ілюструють різні варіанти: від потужного серверу, коли практично вся робота проводиться на ньому, до потужного клієнта, коли велика частина функцій виконується на робочій станції, а сервер обробляє ті SQL-виклики, що поступають до нього по мережі.

В моделях віддаленого доступу до даних і віддаленого представлення проводиться строгий розподіл функцій між комп'ютером-клієнтом і комп'ютером-сервером [5].

В інших моделях має місце виконання однієї з наступних функцій одночасно на двох комп'ютерах:

— управління даними (модель розподіленої БД);

— обробки інформації (модель розподіленої функції);

— представлення інформації (модель розподіленого представлення).

Найбільш поширеними двохланковими моделями розподілу функцій є моделі віддаленого доступу до даних і віддаленого представлення (серверу БД).

3.2.2 Трьохланкова модель розподілу функцій

Трьохланкова модель розподілу функцій є типовим варіантом, при якому кожна з трьох функцій застосування реалізується на окремому комп'ютері.

Варіанти розподілу функцій застосувань на більше число комп'ютерів можуть мати місце, але застосовуються дуже рідко.

Розглянемо модель, яка має назву модель серверу застосувань, або AS-модель (Application Server), і показана на рисунку 1.4.

Згідно трьохланкової AS-моделі, процес, який відповідає за організацію діалогу з кінцевим користувачем, як завжди, реалізує функції представлення інформації і взаємодіє з компонентом застосування. Компонент застосування, розташовуючись на окремому комп'ютері, у свою чергу, зв'язаний з компонентом управління даними.

Рисунок 1.4 — Трьохланкова модель серверу застосувань.

Центральною ланкою AS-моделі є сервер застосувань. На сервері застосувань реалізується декілька прикладних функцій, кожна з яких оформлена як служба надання послуг усім програмам, які їх вимагають.

Серверів застосувань може бути декілька, причому кожний з них надає свій вид сервісу. Будь-яка програма, що запрошує послугу у сервера застосувань, є для нього клієнтом. Запити, що поступають від клієнтів до серверів, поміщуються в чергу [6], з якої вибираються відповідно до деякого правила, наприклад, за пріоритетами.

Компонент, що реалізує функції представлення і є клієнтом для сервера застосувань, в цій моделі трактується більш широко, ніж звичайно. Він може використатись для організації інтерфейсу з кінцевим користувачем, забезпечувати прийом даних від пристроїв, наприклад, датчиків, або бути довільною програмою.

Перевагою AS-моделі є гнучкість і універсальність внаслідок розділення функцій застосування на три незалежні складові. У багатьох випадках ця модель виявляється більш ефективною в порівнянні з двохланковими.

Основний недолік моделі — більш високі витрати ресурсів комп'ютерів на обмін інформацією між компонентами застосування в порівнянні з двохланковими моделями.

Прикладами програмних продуктів, що реалізують середовище функціонування застосувань на комп'ютерах-серверах застосувань, є BEA

WebLogic Server (BEA Systems Corp.), Inprise Application Server (Inprise Corp.) і IBM WebSphere Application Server (IBM Corp.).

3.2.3 Складні схеми взаємодії

Можливі складніші схеми взаємодії, наприклад, схеми, в яких елемент, що є сервером для деякого клієнта, у свою чергу, виступає в ролі клієнта по відношенню до іншого серверу (див. рисунок 1.5). Приклад цього ми спостерігали в AS-моделі.

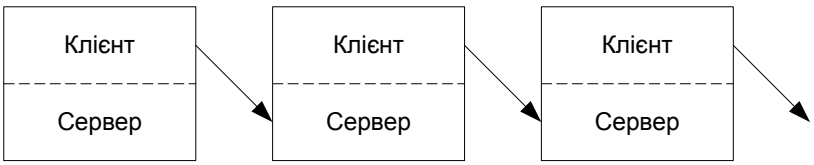


Рисунок 1.5 — Ланцюжок взаємодій типу клієнт-сервер.

Можливо також, що в розподіленій обчислювальній системі при роботі з БД є множинні зв'язки (статичні), коли один об'єкт у відношенні до одних є клієнтом, а у відношенні до інших — сервером (див. рисунок 1.6).

При розгляді взаємодії об'єктів в динаміці виходять ще складніші схеми взаємодії. Прикладом такої схеми є випадок, коли в процесі роботи ролі об'єктів змінюються [7]: об'єкт, що є в деякий момент часу клієнтом у відношенні до іншого об'єкту, в подальшому стає сервером для іншого об'єкту.

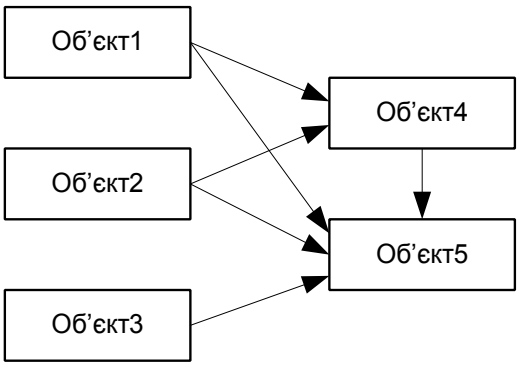


Рисунок 1.6 — Множинні зв'язки взаємодії типу клієнт-сервер

4 Принципи функціонування Web-застосувань

Програмні засоби мереж Інтернет/Інтранет включають нову категорію програм — Web-застосування.

До Web-застосувань відносять набір Web-сторінок, сценаріїв і інших програмних засобів, розташованих на одному або декількох комп'ютерах (клієнтських та серверних) і з'єднаних для виконання прикладної задачі.

Сучасні Web-застосування застосовуються для створення інформаційних систем в мережах Інтернет/Інтранет і зазвичай будуються як багатоланкові (багаторівневі) застосування, що публікують БД. Введення багаторівневої архітектури за умови великої кількості клієнтських комп'ютерів обумовлено необхідністю зменшення навантаження на сервер баз даних і лінії зв'язку.

Web-застосування мають ряд особливостей функціонування у порівнянні з іншими багаторівневими клієнт-серверними застосуваннями [12], які полягають в принципах роботи Інтернету. Стисло нагадаємо основні з цих принципів.

Web-застосування виконуються на стороні Web-серверу, який знаходиться на Web-вузлах мережі Інтернет. Web-сервер обробляє запити оглядача на отримання Web-сторінок і посилає необхідні дані оглядачу у форматі Web-документів.

Обмін даними в мережі Інтернет здійснюється на апаратному рівні на основі протоколу TCP/IP і протоколу більш високого логічного рівня HTTP. Спрощена схема функціонування Web-застосування приведена на рисунку 1.7.

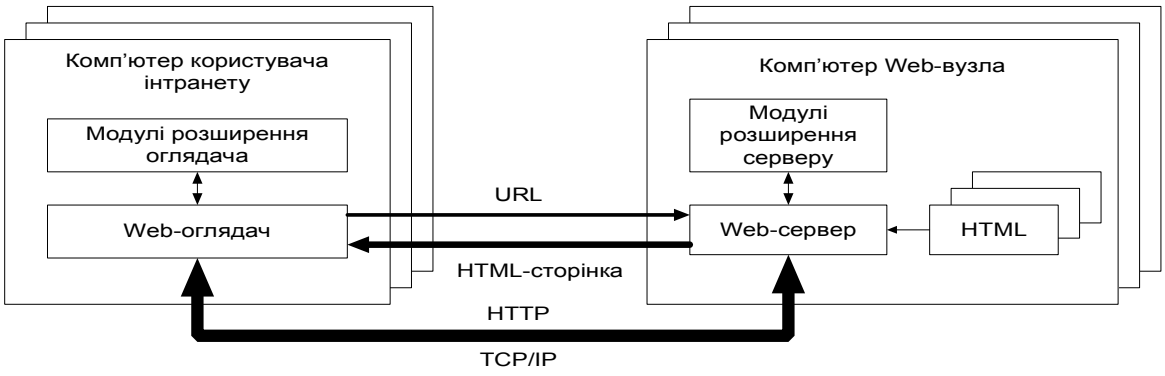


Рисунок 1.7 — Спрощена схема функціонування Web-застосування.

Нагадаємо, що під Web-документом (Web-сторінкою) розуміють документи, що використовуються в мережі Інтернет у форматах HTML, XML, шаблони у форматах ASP, HTX і т.д.

Для доступу до Web-сторінок використовуються спеціальні клієнтські програми — оглядачі Web, що знаходяться на комп'ютерах користувачів Інтернету. Оглядач формує запит на отримання необхідної сторінки або іншого ресурсу за допомогою адреси URL. Функції оглядача полягають у відображенні Web-сторінок, що згенерували сервером або модулями розширення, і відправці запитів користувача Web-застосуванню [14]. Оглядач є з’єднуючою ланкою між користувачем і Web-додатком. При цьому Web-оглядач встановлює з'єднання з необхідним Web-вузлом, використовуючи різні протоколи передачі даних, нами розглядається використання протоколу HTTP.

Розвиток технологій в мережі Інтернет привів до появи нових архітектурних і технологічних рішень для локальних корпоративних інтранет-мереж. Мережі Інтранет побудовані на тому ж апаратно-програмному забезпеченні, принципах і протоколах, що і мережа Інтернет. В загальному випадку під мережею Інтранет розуміють виділену частину мережі Інтернет, в якій виконується Web-застосування (інформаційна система).

4.1 Web-застосування в мережах Інтранет

Застосування Інтернет-технологій в корпоративних Інтранет-мережах дозволяє підвищувати ефективність функціонування мереж і інформаційних систем, що використовуються них.

Застосування, які побудовані на основі використання Інтернет-технологій, характеризуються:

— надійністю, яка обумовлена стійкістю роботи програмно-апаратних засобів мережі Інтернет, випробуваною протягом багатьох років;

— масштабованістю, що забезпечується використанням багаторівневої архітектури, яка дозволяє одне і те ж Web-застосування практично без

повторного конфігурування використовувати для Інтранет-застосувань з різною архітектурою;

— відкритістю архітектури, яка ґрунтується на гнучкій багаторівневій архітектурі і стандартизованих протоколах та форматах документів, доступних для модифікації;

— простотою вивчення та використання, що обумовлена стандартизацією призначеного для користувача інтерфейсу на основі застосування однотипного клієнтського застосування — оглядача зі стандартним графічним інтерфейсом користувача;

— значним зниженням грошових витрат (в десятки і сотні разів) на обслуговування, модернізацію і нарощування мережі Інтранет в порівнянні з традиційними корпоративними мережами, побудованими на клієнт/серверних технологіях.

Важливою перевагою мереж Інтранет є можливість розгортання на існуючій інфраструктурі корпоративних локальних і глобальних мереж [23]. Для побудови мережі Інтранет допускається просте вбудовування в існуючі корпоративні мережі з використанням існуючого апаратного забезпечення.

При використанні Web-застосувань в мережі Інтранет може використовуватися архітектура, показана на рисунку 1.8. Мережа Інтранет в загальному випадку має різну внутрішню структуру, побудовану на принципах Інтернет. Причому мережа Інтранет може і не мати виходу в Інтернет.

У якості клієнтських застосування в цій архітектурі виступають Web-оглядачі, які звертаються із запитами до серверу БД або до серверу застосувань через Web-сервер. Залежно від архітектури, що використовується, Web-сервер може знаходитися на сервері БД або на сервері застосувань.

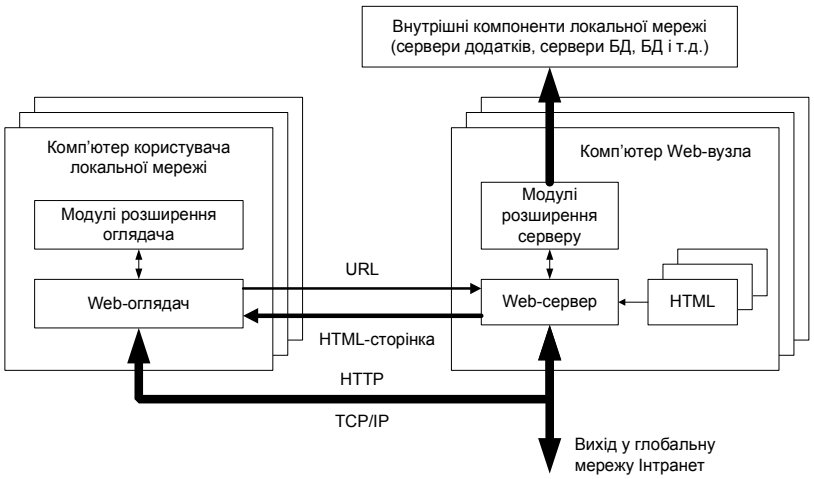


Рисунок 1.8 — Схема функціонування Web-застосування з використанням модулів розширення.

У функції Web-серверу в мережі Інтранет входить обробка запитів Web-оглядачів на отримання інформації з БД, що розділяються, перетворення цих запитів в SQL-запити або інші формати, зрозумілі для серверу БД чи серверу застосувань.

5 Порівняльний аналіз аналогів

Подібна програмна реалізація на етапі постановки задач розробки повинна порівнюватись з аналогами у двох аспектах – користувацького інтерфейсу та аналітичному, тобто необхідно розглянути існуючи реалізації у сфері роботи зі зведеними базами даних та сфері рекрутингу кандидатів для конкретної посади.

5.1 Аналіз аналогів у сфері роботи зі зведеними базами даних

На даний момент існує декілька варіантів реалізації зведеної бази даних обліку кадрів, серед яких:

— Microsoft Dynamics AX;

— E-Staff Рекрутер;

Microsoft Dynamics AX (рисунок 1.9) — це один з програмних продуктів компанії Microsoft, який належить до систем управління ресурсами підприємства. Він входить до лінійки Microsoft Dynamics. Для розширення і модифікації системи під потреби користувача в AX існує своє інтегроване середовище розробки (IDE) MorphX, до якого входять такі інструменти розробки, як debugger, аналізатор коду та інтерфейс запитів. Це середовище знаходиться в тій самій клієнтській програмі, з якою працює звичайний користувач, і таким чином дозволяє займатися розробкою на будь-якому з її екземплярів. Розробка в AX відбувається за допомогою мови програмування X++, яка дуже нагадує C# і Java.

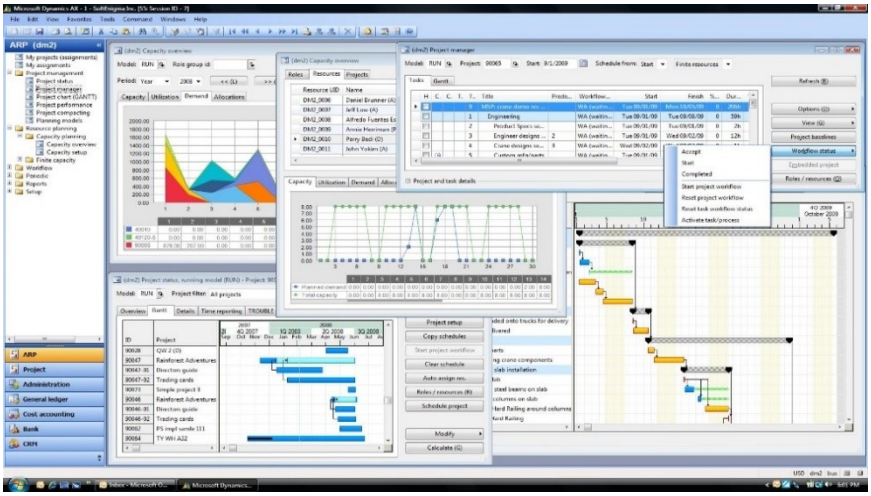


Рисунок 1.9 — Зовнішній вигляд робочої області додатку Microsoft Dynamics AX

Серед недоліків продукту можна виділити:

— велика кількість зайвого функціоналу;

— незрозумілий для вітчизняного ринку інтерфейс.

Система E-Staff Рекрутер (рисунок 1.10) існує на ринку з 2000 року. Система призначена для HR-служб компаній, що здійснюють підбір співробітників, а також для кадрових агентств. E-Staff Рекрутер – система повного циклу, яка автоматизує більшість рутинних операцій в рекрутингу.

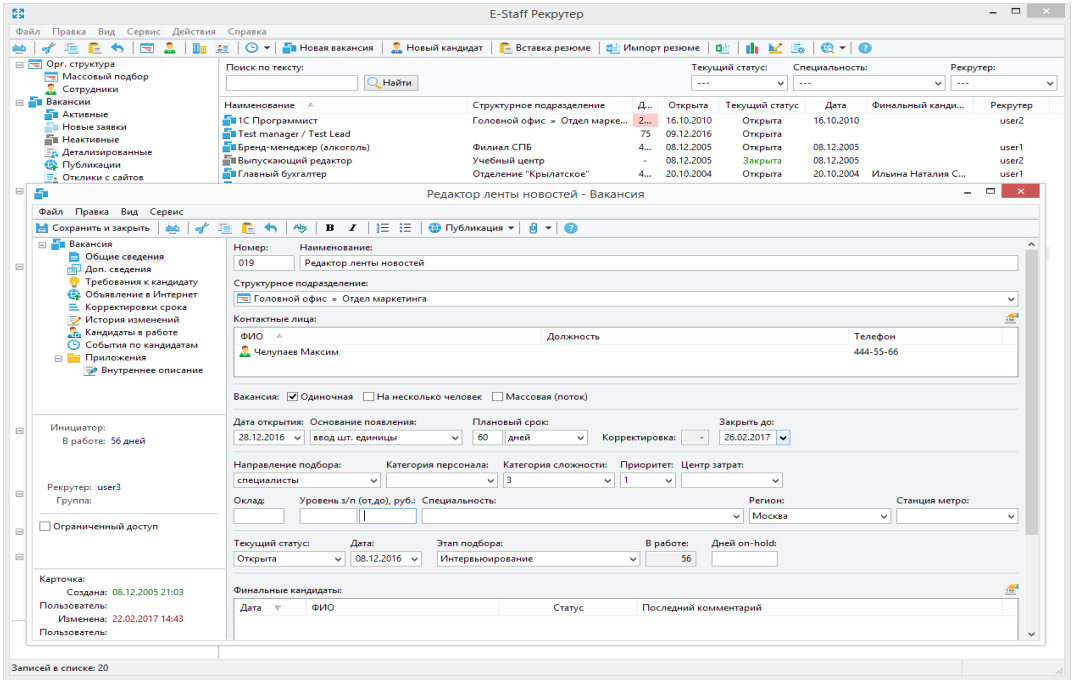


Рисунок 1.10 — Зовнішній вигляд робочої області додатку E-Staff Рекрутер

На жаль, дана система теж володіє рядом недоліків:

— низька якість пошуку;

— відсутня можливість виключати оброблених в попередніх запитах кандидатів з результатів нового пошуку.

У таблиці 1.1 наведено порівняння представлених вище аналогів з розроблюваною магістерською кваліфікаційною роботою.

Таблиця 1.1 – Порівняльний аналіз аналогів

Критерії	Microsoft Dynamics AX	E-Staff Рекрутер	Дипломна робота «VStaff Advice»
Доступні дані про задіяних працівників	+	-	+
Підбір кандидатів на вакансію	-	+	+
Присутня можливість впровадження в інші розробки	-	-	+

Продовження таблиці 1.1

Сумісність з популярними операційними системами	+	-	+
Рекомендування вакансій для кандидатів	-	-	+
Не перевантажений функціоналом інтерфейс	-	-	+
Можливість роботи з базами декількох підприємств	-	-	+

Загальним недоліком обох рішень є те, що програми є платними. Оплата необхідна для імпорту резюме з сайтів пошуку роботи інтуїтивно та швидко. За відсутності оплати база залишиться, але вносити резюме доведеться самостійно, заповнюючи бланки. Ще з мінусів: не зовсім зручний інтерфейс, багато зайвих функцій.

Тому було прийнято рішення розробити власну реалізацію додатку для обліку кадрів з переліком вакантних агентів. При цьому, передбачається реалізація синхронізації інформації між підприємствами та рекрутинговими агентствами, які в процесі роботи зможуть обмінюватись кадрами та інформацією про них.

5.2 Аналіз існуючих методів оцінки у сфері рекрутингу кандидатів для конкретної посади

На сьогоднішній день при рекрутингу кандидатів застосовується набір оціночних методик для виявлення відповідності кандидата на розглядану вакансію, серед яких можна виділити попередній аналіз та узагальнення пунктів резюме, виконання тестового завдання, що визначає відповідність рівня навичок, необхідних для посади та безпосередньо співбесіда – часто часозатратний процес для оцінки «хард» та «софт» навичок кандидата.

Серед методик, використовуваних при співбесідах кандидатів [1], можна окремо виділити:

— поведінкове інтерв’ю за моделями STAR та PARLA;

— порівняльний метод оцінки кандидата на основі регіональної приналежності.

Поведінковий інтерв'ю може бути застосовано для кандидатів з будь-якої сфери діяльності. В ході інтерв'ю рекрутер збирає повні поведінкові приклади (ППП) з досвіду кандидата. З кожного такого стають зрозумілі ситуація, з якою кандидат зіткнувся (situation); завдання, яке стояло перед ним (task); дії, до яких запобіг кандидат (action); результат, підсумок ситуації (result).

Ці компоненти легко запам'ятати за абревіатурою STAR – Situation, Task, Action, Result.

Як правило, достатньо отримати по 2-3 повних поведінкових приклади (ППП) для кожної необхідної компетенції, тоді картина досвіду виходить більш-менш ясною.

Результат поведінкового інтерв’ю повинен давати відповідь на питання [17] – чи є у кандидата достатньо успішний досвід у вирішенні ситуацій, схожих на ті, які на нього чекають при роботі?

Серед недоліків даного методу можна виділити:

— важке встановлення професійного кругозору кандидата та, відповідно, спрогнозувати можливість зміни професійних пріоритетів в майбутньому;

— досить великі затрати у часі.

При порівняльному методі оцінки кандидата на основі регіональної приналежності виконується набір послідовних дій:

1. Розробка вимог та заявки на підбір, що включає в себе визначення стандартних вимог до кандидата, специфіки роботи в конкретному регіоні та формування остаточної заявки на підбір персоналу;

2. Пошук і залучення кандидатів, що розуміє під собою визначення основних джерел залучення кандидатів;

3. Проведення інтерв’ю з кандидатом, що може означати як телефонне інтерв’ю, так і за допомогою Skype або ж, в іншому випадку – інтерв’ю при особистій зустрічі;

4. Оцінка кандидатів, при якій необхідно враховувати набір компетенцій, стратегію розвитку компанії в регіоні, трактування досягнень, рівень підготовки кандидатів в різноманітних регіонах та правильне розуміння мотивації кандидата.

Основними недоліками даного методу є:

— можливість використання лише для вакансій, що призначені для регіональних посад або віддаленої роботи;

— підвищена ймовірність зниження точності оцінки кандидата.

Загальним недоліком обох методів можна виділити відсутність фіксованої шкали оцінки розгляданих критеріїв та необхідність очного спілкування рекрутерів з кандидатами.

Тому очевидно актуальною є ідея розробки власного методу оцінки кандидатів шляхом поєднання методів, що використовувались для порівняльного аналізу. При цьому передбачається реалізація шкали оцінки для розгляданих у цих методах критеріїв для можливості визначення конкретного коефіцієнту сумісності кандидата з розгляданою посадою.

6 Постановка задач розробки

Розроблений програмний додаток повинен використовувати технологію взаємодії користувача з додатком за допомогою браузера, синхронізацію виконаних дій зі зведеною базою даних та повернення релевантного результату.

Також даний додаток повинен на основі отриманої з бази даних інформації застосовувати механізм прогнозування сумісності зіставних сутностей бази.

Для досягнення поставлених цілей потрібно вирішити такі завдання:

— розробити метод прогнозованої оцінки сумісності кандидата з розгляданою вакансією;

— реалізувати сховище даних вакансій та кадрів у вигляді бази даних під керуванням СУБД Microsoft SQL Server;

— реалізувати базову модель додатку та зв’язати її з базою даних за допомогою програмної платформи Entity Framework;

— створити розширення функціоналу у вигляді розмежування на сторінки великої кількості однотипних елементів;

— стилізувати контент за допомогою каскадних таблиць стилів (CSS) та модулів на основі них, наприклад, Bootstrap;

— створити навігацію, що виконуватиме відсіювання зайвої інформації за критеріями, встановленими користувачем;

протестувати функціонал та можливості кінцевого проекту.

7 Висновок

У результаті проведених аналізів була продемонстровано актуальність та необхідність розробки у наш час. Визначені основні вимоги та структура розробки додатку для обліку кадрів та відкритих вакансій. Визначені постановка задач та хід роботи. В ході аналізу аналогів, було прийнято, що є доцільним розробити власний додаток зі схожими властивостями і в той самий час – з якісними відмінностями.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ПРОГНОЗОВАНОЇ ОЦІНКИ, МОДЕЛІ СУТНОСТЕЙ І СХОВИЩА ДАНИХ

8 Приведення методик інтерв’ю до градаційної шкали

У попередньому розділі було описано два методи оцінки потенційних працівників при рекрутингу на вакантну посаду. Зазначалось, що дані методи дозволяють доволі ефективно оцінити придатність кандидата для розгляданої посади на етапі співбесіди. Також було відмічено наявність загального недоліку в обох розглянутих методах – відсутність еквіваленту у вигляді оцінки результатам проходження етапів співбесіди.

Також необхідно зазначити, що подібна оцінка можлива лише при проходженні кандидатом співбесіди або одразу після неї.

Для реалізації механізму рекомендування кадрів до вакансій і навпаки, потрібно привести оціночні критерії використовуваних методик у більш предметну площину [2], наприклад, конвертувати абстраговані висловлювання результатів у конкретні оцінки.

Подібна реорганізація дозволить групувати окремі оцінки за категоріями, що дасть можливість уніфікувати деталі резюме та вакансій у різних галузях пошуку роботи.

У якості основної буде розглядатись методика проведення поведінкового інтерв’ю за моделлю STAR, так як вона передбачає досить широке різноманіття критеріїв оцінки кандидатів, а метод оцінки кандидата на основі регіональної приналежності буде розширювати «батьківський» метод, надаючи йому варіативності та більшої гнучкості.

Поведінкове інтерв’ю може бути застосовано для кандидатів з будь-якої сфери діяльності. В ході інтерв'ю рекрутер збирає повні поведінкові приклади (ППП) з досвіду кандидата. З кожного такого стають зрозумілі:

— Ситуація, з якою кандидат зіткнувся (situation);

— Завдання, яке стояло перед ним (task);

— Дії, які здійснювались кандидатом (action);

— Результат, підсумок ситуації (result).

Як правило, достатньо отримати по 2-3 повних поведінкових приклади (ППП) для кожної необхідної компетенції, тоді картина досвіду виходить більш-менш ясною.

Спочатку необхідно визначити, який саме досвід вирішення задач потрібен для конкретної вакансії. Найоптимальнішим у даному випадку буде використання списку компетенцій. Наприклад, у галузі «Менеджмент робочих кадрів» можна виділити декілька компетенцій – рекрутинг, мотивація, делегування і так далі. Відповідно, у кожній компетенції можна сформулювати ряд запитань, відповіді на які характеризуватимуть досвід кандидата у даній компетенції.

Подібні запитання міститимуть ряд стандартних відповідей, серед яких кандидат в резюме обиратиме одну або ж вказуватиме відповідний варіант.

Рекрутер при описі вакансії вказує пріоритетні компетенції та для підвищення точності системи рекомендування може вказати бажані відповіді у кожній конкретній компетенції. Також обов’язковим буде вказання пріоритету для кожної компетенції за шкалою від 1 до 10. Подібний діапазон буде основним у всіх шкалах оцінювання при розробці даного методу.

Метод оцінки кандидата на основі регіональної приналежності дозволить розробленому механізму «обрости» варіативністю [20] та дозволить в залежності від групових та загального коефіцієнтів визначати не лише місце кандидата/вакансії в списку рекомендованих, але і, в залежності від значення, визначати належність до специфічних груп вакансій/кандидатів.

Рекрутер при формуванні нового запису вакансії вказує, чи є вакансія належною до категорій:

— віддалена робота;

— part-time робота;

— робота з можливістю релокейту.

При належності вакансії до однієї з вищевказаних категорій механізм рекомендування серед відібраних кандидатів надаватиме пріоритет тим, в кого в записі вказане серед особливих побажань згода на вакансію, що відмічена однією з вищерозглянутих категорій.

І навпаки – при рекомендуванні потенційному кандидату відкритих вакансій пріоритетними будуть ті, що відмічені однією з вищевказаних особливостей.

Подібне розгалуження критеріїв сприятиме більшій варіативності механізму рекомендування та дозволить не обмежуватись лише однією шкалою оцінювання.

9 Комбінування запропонованих методик інтерв'ю у вигляді коефіцієнту відповідності

Після переведення вищевказаних методів у чисельну метрику, постає питання використання даних напрацювань при прогнозуванні відповідності кандидата вакансії та навпаки.

При формуванні списку рекомендацій по вакансії, за замовчуванням дані відображатимуться, відсортовані за величиною загального коефіцієнту сумісності [2]. При необхідності перегляду і порівняння даних кандидатів за конкретними компетенціями список прийматиме вигляд таблиці даних, що зберігатиме початкове сортування та відображатиме більш детальну інформацію щодо сумісності кандидатів за компетенціями.

Коефіцієнт сумісності за конкретною компетенцією встановлюється в результаті використання формули:

$$k = \frac{S}{n} P$$

де k – коефіцієнт сумісності за компетенцією, S – сума коректних відповідей у компетенції, n – кількість запитань за компетенцією, P – пріоритет компетенції, вказаний у вакансії.

Отриманий коефіцієнт вказує на відповідність розгляданого кандидата у визначеній компетенції. За відсутності інших компетенцій, обов'язкових для вакансії, отримане значення вказуватиме, наскільки кандидат відповідає вакансії, на яку шукають працівника.

За наявності повного збігу між записами кандидата та вакансії у регіональних ознаках, фінальний коефіцієнт збільшується, як наведено у формулі:

$$K = K + R$$

,

де K – коефіцієнт сумісності кандидата з вакансією, R – ознака регіонального збігу.

Проте у більшості випадків вакансія не обмежуватиметься лише однією пріоритетною компетенцією, тому фінальний коефіцієнт повинен враховувати їх всі, однаково оцінюючи кожного кандидата, незалежно від ступеня наповненості його запису у базі. Тому загальний коефіцієнт відповідності дорівнюватиме наступній формулі:

$$K = \frac{\sum k}{N}$$

де K – загальний коефіцієнт сумісності, k – коефіцієнт сумісності за конкретною компетенцією, N – кількість компетенцій, вказаних у резюме.

Отриманий коефіцієнт дозволяє усереднити велику кількість [21] резюме, навіть якщо кожне з них розроблялось у відповідності з власним унікальним шаблоном. За бажання або необхідності коефіцієнт може не враховуватись або бути другорядним, дозволяючи рекрутерам фільтрувати вакансії вручну.

Алгоритм роботи комбінованого методу наведено на рисунку 2.1.

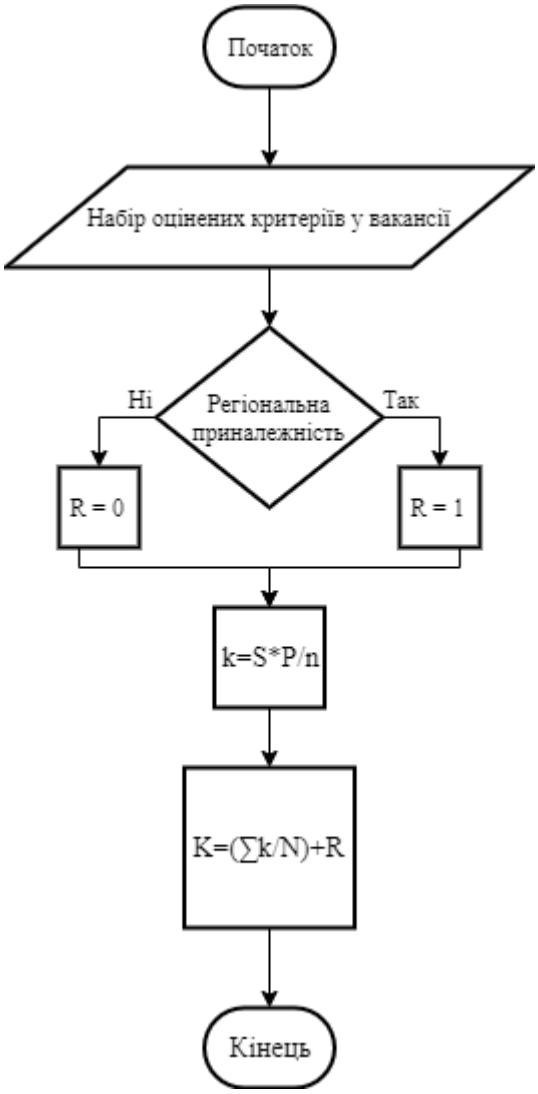


Рисунок 2.1 — Алгоритм роботи комбінованого методу визначення сумісності кандидата з вакансією.

Використання подібного механізму значно спрощує сортування та фільтрацію [22] кандидатів при пошуку під конкретну вакансію. Даний метод працює і в зворотному напрямку, дозволяючи вакантним агентам слідкувати за найбільш релевантними для себе вакансіями, відібраними та відсортованими, використовуючи вищеописаний механізм.

10 Моделювання сховища даних для вакансій та кадрів

Всі сутності в додатку прийнято виділяти в окремі моделі. Залежно від поставленого завдання і складності програми можна виділити різну кількість моделей.

Моделі представляють собою прості класи і розташовуються в проекті в каталозі Models. Моделі описують логіку даних.

Модель необов'язково складається тільки з властивостей, крім того, вона може мати конструктор [23], які-небудь допоміжні методи. Але головне не перевантажувати клас моделі і пам'ятати, що його призначення - описувати дані. Маніпуляції з даними і бізнес-логіка - це більше сфера контролера.

Дані моделей, як правило, зберігаються в базі даних. Для роботи з базою даних дуже зручно користуватися фреймворком Entity Framework, який дозволяє абстрагуватися від написання sql-запитів, від будови бази даних і повністю зосередитися на логіці програми.

Якщо при створенні проекту MVC 5 був встановлений тип аутентифікації «No Authentication», то після “” створення проекту в нього необхідно підключити Entity Framework через пакетний менеджер NuGet.

Після введення команди буде завантажений і встановлений пакет Entity Framework. Іноді цієї консоллю переважно користуються при встановленні пакетів, ніж менеджером NuGet, так як менеджер NuGet може трохи спізнюватися за випуском останніх версій пакетів. Або навпаки, нам треба встановити пакети більш ранньої версії, а NuGet може запропонувати тільки поточну версію.

Entity Framework підтримує підхід «Code first», який передбачає збереження або вилучення інформації з БД на SQL Server без створення схеми бази даних [11] або використання дизайнера в Visual Studio. Навпаки, створюються звичайні класи, а Entity Framework вже сам визначає, як і де зберігати об'єкти цих класів.

Для підключення до бази даних через Entity Framework, потрібен посередник — контекст даних. Контекст даних містить одну або кілька

властивостей типу DbSet<T>, де T представляє тип об'єкта, що зберігається в базі даних.

Схема бази даних, структурована та організована для взаємодії з Entity Framework та підходом «Code first», наведена в додатку Г.

11 Розробка моделі розширення серверу

Для розширення можливостей клієнтської частини (оглядача) і серверної частини розробляють модулі розширення оглядача і серверу, що використовуються для динамічного керування об'єктами користувацького інтерфейсу (компонентами) Web-документа.

Архітектура Web-застосувань з модулями розширення серверу (див. рисунок 2.9) може включати стандартні модулі розширення — dll-бібліотеки, що реалізують, наприклад, технології ASP, IDC/HTX, об'єкти ActiveX. Крім того, можуть підключатися додаткові модулі, розроблені з використанням інтерфейсів CGI, ISAPI та ін.

В цьому випадку у функції Web-серверу входять обробка запитів Web-оглядачів користувачів мережі, виклик (завантаження) відповідного модуля розширення серверу і передача йому параметрів запиту. В результаті обробки запиту модулями розширення серверу формується Web-документ з використанням різних HTML-шаблонів. Готовий HTML-документ відсилається Web-оглядачу у форматі протоколу HTTP.

Web-сервер формує динамічно Web-сторінки [5] різними способами і посилає готові Web-сторінки у форматі протоколу HTTP.

Розрізняють пасивний і активний стан Web-серверу. Так, Web-сервер знаходиться в пасивному стані, якщо документ, що ним формується, містить статичну інформацію, тобто на Web-сторінці відсутні засоби введення і обробки запитів до серверу (див. рисунок 2.2).

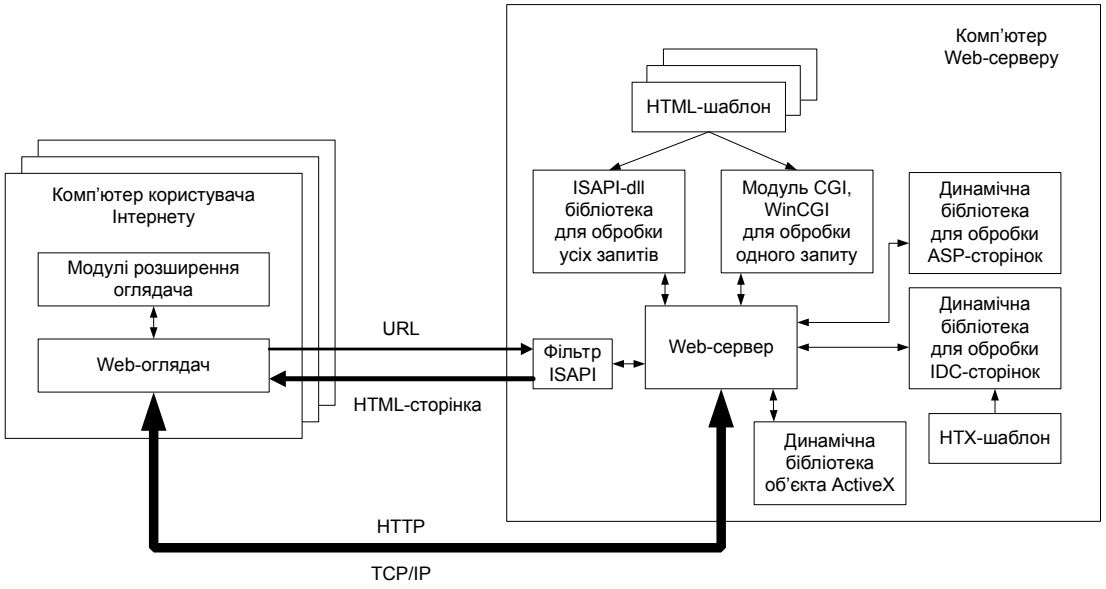


Рисунок 2.2 — Архітектура Web-застосування з модулями розширення серверу.

В активному стані Web-сервер знаходиться при динамічному створенні Web-документів у відповідь на запит користувача або у випадку, коли в оглядач завантажені різні інтерактивні елементи форми. Для публікації БД основний інтерес представляє активний Web-сервер, що реалізується за допомогою модулів розширення Web-серверу.

12 Розробка моделі розширення клієнтської частини

Для зменшення навантаження на Web-сервер частину функцій, пов'язаних з попередньою обробкою запитів і введених даних, доцільно виконувати на стороні клієнта. Цю задачу вирішують модулі розширення клієнтської частини. Архітектура Web-застосування з використанням модулів розширення клієнтської частини показана на рисунку 2.3.

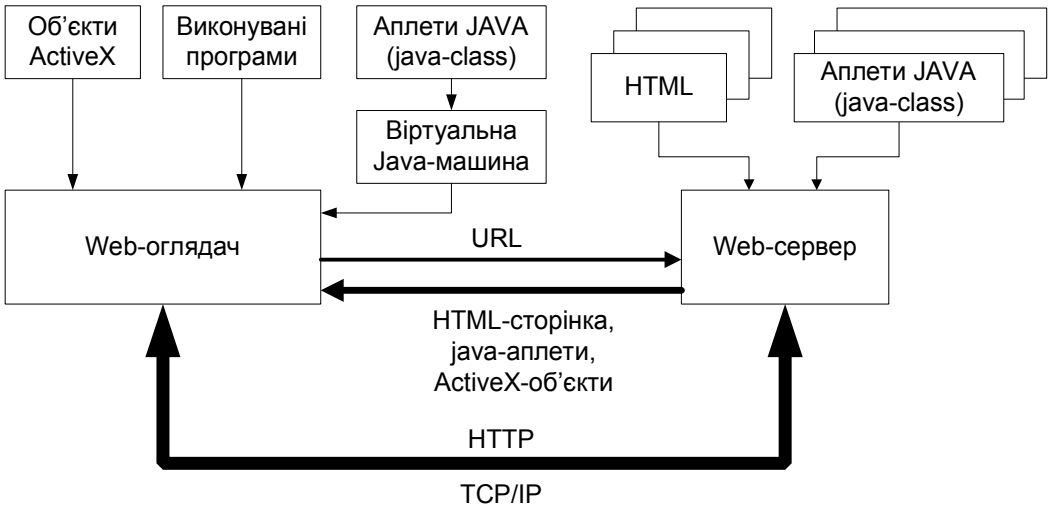


Рисунок 2.3 — Схема Web-застосування з використанням модулів

У випадку модулів розширення клієнтської частини (активність на стороні клієнта) використовують аплети, програми, що підключаються, ActiveX-об'єкти і сценарії. Ці технології можуть використатися для створення динамічних ефектів при перегляді Web-сторінки.

Елементи керування ActiveX є видом модулів розширення, які можуть використатися на стороні клієнта або на стороні серверу. Вони реалізуються за допомогою динамічних бібліотек DLL і можуть бути вбудовані у Web-документ як додаткові елементи користувацького інтерфейсу. Механізм роботи елементів керування ActiveX дозволяє з програмного коду цих об'єктів діставати необмежений доступ до локальних ресурсів комп'ютера користувача [6]. З елемента керування ActiveX є можливість передавати на сервер будь-яку інформацію з комп'ютера користувача. З погляду забезпечення безпеки локальних даних користувачів використання елементів керування ActiveX не завжди виправдане.

Виконувані програми є першим поколінням клієнтських розширень. Для організації роботи такої програми у Web-застосуванні необхідно заздалегідь встановлювати таку програму і конфігурувати оглядач. Тому при використанні механізму виконуваних програм дуже важко забезпечити сумісність такого Web-застосування для різних оглядачів. Крім того, використання такої технології порушує принцип стандартизації клієнтського інтерфейсу.

Аплет Java застосовуються для створення інтерфейсу користувача, що динамічно формується. Аплет є байтовим кодом, який інтерпретується віртуальною Java-машиною, що входить до складу оглядача. У аплетів відсутня можливість здійснювати запис на диск користувача. Тому використання механізму аплетів гарантує цілісність локальних даних користувачів.

13 Висновок

У час великого розвитку інформаційних технологій розробка досить актуальна. Зібрана в одному місці загальнодоступна коротка інформація про в наявні вакансії та вакантних агентів необхідна, тому створення подібного додатку та постійне оновлення в ньому актуальної інформації приймає велику користь.

Отже, дана розробка призначена для опублікування систематизованого каталогу вакансій та потенційних працівників. Розробка стане цінним вказівником для людей, які прагнуть не перебирати велику кількість сайтів, збираючи інформацію про конкретного працівника та набір доступних вакансій на підприємстві, а використовуючи інтуїтивно зрозумілу навігацію знайти необхідні вакантні посади та людей з основною інформацією про них.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДОДАТКУ

14 Обґрунтування вибору пакету прикладних програм для розробки додатку для обліку кадрів

Створити застосування для обліку кадрів можна, використовуючи різні технології та мови програмування. І навіть якщо визначитись з мовою, якою буде написаний даний додаток, постає питання вибору підходу логіки написання коду та застосування набору програмних пакетів та допоміжних платформ і бібліотек. А враховуючи, що процес створення електронного каталогу розділений на декілька частин, даний набір питань доведеться вирішити як мінімум двічі.

Якщо відкинути PHP, одразу на думку приходить технологія ASP .NET. Це повноцінна мова. Крім того, платформа .NET дозволяє використовувати будь-яку .NET-сумісну мову (C#, VB.NET, Python, Delphi та інші). Особливо можна відмітити зручний об’єктно-орієнтований синтаксис базових бібліотек. Також, на відміну від PHP, в .NET код компілюється, завдяки чому він виконується набагато швидше [7]. Або можна згадати про ідентичність середовища розробки та розгортання (deployment). Якщо використовується ОС Windows, то при розробці на PHP, ваше середовище розробки відрізняється від серверної (так як там стоїть Linux). При розробці на ASP .NET використовується однакове середовище і на комп’ютерах розробника, і на серверах в Інтернеті. Як перевагу ще можна виокремити повну підтримку Unicode. Всі проекти на ASP .NET автоматично підтримують UTF-8 [11], чого не скажеш про проекти на PHP. ASP .NET надає величезні можливості, включені до складу стандартних бібліотек. В PHP велика кількість аналогічних речей реалізуються сторонніми бібліотеками, але їх використання створює додаткові проблеми: відсутність бібліотек на хостингу, необхідність вивчення. І на завершення – MS SQL проти MySQL. Як PHP-розробники використовують в

більшості випадків MySQL, так розробники на .NET використовують MS SQL. Останній володіє більш широкими можливостями, ніж MySQL.

ASP.NET є єдиною моделлю для розробки веб-додатків, яка містить служби, необхідні для створення веб-додатків для підприємств. ASP. NET є частиною .NET Framework і дозволяє повною мірою використовувати можливості середовища CLR [15] (наприклад, строга типізація, спадкування, взаємодія мов і відстеження версій).

Microsoft повністю перебудувала ASP .NET, ґрунтуючись на Common Language Runtime (CLR), яка є основою всіх додатків Microsoft .NET. Розробники можуть писати код для ASP .NET, використовуючи практично будь-які мови програмування , що входять у комплект .NET Framework (C#, Visual Basic .NET і JScript .NET). ASP .NET має перевагу в швидкості в порівнянні зі скриптовими технологіями, так як при першому зверненні код компілюється і поміщається в спеціальний кеш, і згодом тільки виповнюється, не вимагаючи витрат часу на парсинг, оптимізацію, і т. д.

Платформа ASP .NET MVC являє собою фреймворк для створення сайтів і веб-додатків за допомогою реалізації паттерну MVC.

Шаблон архітектури Model-View-Controller (MVC) розділяє додаток на три основних компоненти: модель, подання і контролер.

Об'єкти моделей є частинами програми, що реалізують логіку для домену даних програми. Об'єкти моделей часто отримують і зберігають стан моделі в базі даних [6]. Наприклад, об'єкт Product може отримувати інформацію з бази даних, працювати з нею, а потім записувати оновлені дані в таблицю Products бази даних SQL Server.

У невеликих додатках ця модель передбачає концептуальне, а не фізичне розділення. Наприклад, якщо додаток тільки зчитує набір даних і відправляє його в представлення, то фізичний шар моделі і пов'язаних класів відсутня. У цьому випадку набір даних приймає роль об'єкта моделі.

Представлення служать для відображення інтерфейсу користувача додатку. Користувальницький інтерфейс зазвичай створюється на основі даних

моделі [7]. Прикладом може служити представлення для редагування таблиці Products, яке містить текстові поля, списки, що розкриваються і прапорці, значення яких засновані на поточний стан об'єкта Product.

Контролери здійснюють взаємодію з користувачем, роботу з моделлю, а також вибір представлення, що відображає користувальницький інтерфейс. У додатку MVC подання лише відображають дані, а контролер обробляє дані, що вводяться і відповідає на дії користувача. Наприклад, контролер може обробляти строкові значення запиту і передавати їх в модель, яка може використовувати ці значення для відправки запиту в базу даних.

Завдяки цьому реалізується концепція «поділ відповідальності», у зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього додаток більш придатний для тестування. І якщо, припустимо, важлива візуальна частина (frontend), то можна тестувати представлення незалежно від контролера. Або можна зосередитись на серверній (backend) частині і тестувати контролер.

Конкретні реалізації та визначення даного паттерну можуть відрізнятись, але в силу своєї гнучкості та простоти він став дуже популярним останнім часом, особливо у сфері веб-розробки.

Свою реалізацію паттерну представляє платформа ASP .NET MVC. У 2013 році вийшла нова версії ASP .NET MVC - MVC 5 [11], а також реліз Visual Studio 2013, з інструментарієм для роботи з MVC 5.

Для створення адаптивного і розширеного інтерфейсу в MVC 5 використовується css-фреймворк Bootstrap.

Платформа MVC визначається в збірці System.Web.Mvc.

В якості середовища розробки вибір одразу падає на Microsoft Visual Studio без будь-яких альтернатив, адже для вирішення поставленої задачі даний пакет продуктів підходить найкраще, усі інструменти та технології, необхідні для реалізації проекту або вже вбудовані, або доступні для підключення.

15 Розробка та реалізація структури електронного каталогу

Додаток під назвою «VStaff Advice» слідує класичному підходу, який всюди використовується в електронних каталогах. Був створений онлайн каталог кадрів та вакансій, який користувачі зможуть переглядати за категоріями та сторінками. Крім того, була створена адміністративна область, що включає в себе засоби створення, читання, оновлення та видалення (create, read, update, delete - CRUD) для керування цим каталогом, та був захищений так, щоб зміни могли вносити лише зареєстровані адміністратори.

Побудова всіх рівнів необхідної інфраструктури може здатись дещо повільною. Початкову функціональність вдалося б побудувати швидше за допомогою Web Forms, просто перетягуючи елементи керування безпосередньо на базу даних. Однак початкові труднощі розробки додатку MVC окупаються, забезпечуючи придатний для супроводу, розширюваний, добре структурований код з чудовою підтримкою модульного тестування.

15.1 Підготовка структури проекту

Було передбачено створення рішення Visual Studio з трьома проектами всередині. Один проект міститиме модель предметної області, другий – додаток MVC, а третій – модульні тести. Для початку було створено нове рішення Visual Studio під назвою VStaff-Advice з використанням шаблону Blank Solution (Порожнє рішення).

Рішення Visual Studio – це контейнер для одного і більше проектів. Для розроблюваного проекту додатку необхідні три проекти, що описані в таблиці 3.1. Для додавання проекту необхідно натиснути правою кнопкою миші на елементі Solution (Рішення) у вікні Solution Explorer (Провідник рішення) і вибрати в контекстному меню пункт Add – New Project (Додати – Новий проект).

Таблиця 3.1 – Три проекти для додатку VStaff Advice

Ім’я проекту	Шаблон проекту	Призначення
VStaff-Advice.Domain	Class Library (Бібліотека класів)	Містить сутності і логіку предметної області; налаштовується на забезпечення постійності за рахунок сховища.
VStaff-Advice.WebUI	ASP.NET MVC Web Application (Веб-додаток ASP.NET MVC 5); при виборі шаблону проекту було вказано варіант Empty (Порожнє) і відмічений прапорець MVC	Містить контролери і представлення; виступає в якості користувацького інтерфейсу для додатку VStaff-Advice
VStaff-Advice.UnitTests	Unit Test Project (Проект модульного тестування)	Містить модульні тести для інших двох проектів

Після створення трьох проектів, вікно Solution Explorer виглядає так, як показано на рисунку 3.1.

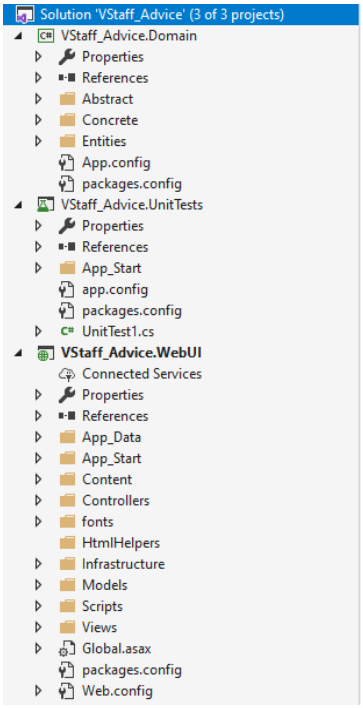


Рисунок 3.1 – Вікно Solution Explorer після додавання трьох проектів

Файл Class1.cs, доданий Visual Studio в проект VStaff-Advice.Domain, був видалений, оскільки він не використовуватиметься в подальшому. Для спрощення відлагодження проект VStaff-Advice.WebUI був встановлений в якості стартового проекту; ім'я проекту виділилось напівжирним шрифтом. Це означає, що даний проект буде завантажуватись при виборі пункту Start Debugging (Запустити відлагодження) або Start Without Debugging (Запустити без відлагодження) в меню Debug (Відлагодження).

15.2 Встановлення пакетів з інструментами

В нашому проекті застосовуватимуться інструменти Ninject і Moq. У вікні командного рядка NuGet (Tools – Library Package Manager – Package Manager Console) були введені команди, наведені у лістингу 3.1:

Лістинг 3.1 – Встановлення пакету Express та його залежностей

```
Install-Package Ninject -version 3.0.1.10 -projectname
VStaffAdvice.WebUI
Install-Package Ninject.Web.Common -version 3.0.0.7 -
projectname VStaffAdvice.WebUI
Install-Package Ninject.MVC3 -Version 3.0.0.6 -projectname
VStaffAdvice.WebUI
Install-Package Ninject -version 3.0.1.10 -projectname
VStaffAdvice.UnitTests
Install-Package Ninject.Web.Common -version 3.0.0.7 -
projectname VStaffAdvice.UnitTests
Install-Package Ninject.MVC3 -Version 3.0.0.6 -projectname
VStaffAdvice.UnitTests
Install-Package Moq -version 4.1.1309.1617 -projectname
VStaffAdvice.WebUI
Install-Package Moq -version 4.1.1309.1617 -projectname
VStaffAdvice.UnitTests
Install-Package Microsoft.Aspnet.Mvc -version 5.0.0 -
projectname VStaffAdvice.Domain
Install-Package Microsoft.Aspnet.Mvc -version 5.0.0 -
projectname VStaffAdvice.UnitTests
```

Пакети NuGet повинні встановлюватись в проекти вибірково і мають бути вказані конкретні версії пакетів, що завантажуються і встановлюються.

15.3 Додавання посилань між проектами

Необхідно налаштувати залежності між проектами, а також від певних збірок Microsoft. Для цього необхідно натиснути на кожному проекті в вікні Solution Explorer, обрати в контекстному меню пункт Add Reference (Додати посилання) і в розділах Assemblies – Framework (Збірки – Інфраструктура), Assemblies – Extensions (Збірки – Розширення) або Solution (Рішення) додати посилання, описані в таблиці 3.2:

Таблиця 3.2 – Обов'язкові залежності між проектами

Ім'я проекту	Залежності всередині рішення	Залежності від збірок
VStaffAdvice.Domain	Відсутні	System .ComponentModel .DataAnnotations
VStaffAdvice.WebUI	VStaffAdvice.Domain	Відсутні
VStaffAdvice.UnitTests	VStaffAdvice.Domain, VStaffAdvice.WebUI	System.Web, Microsoft.CSharp

Встановлення даних відношень необхідно проводити доволі ретельно, інакше за відсутності правильних бібліотек та посилань на проекти на етапі компілювання проекту виникнуть проблеми.

15.4 Налаштування контейнеру DI

Спочатку необхідно використати бібліотеку Ninject для створення спеціального механізму для розпізнавання залежностей, які MVC Framework застосовуватиме при створенні екземплярів об'єктів всередині додатку. Починається даний процес з додавання в проект VStaffAdvice.WebUI папки Infrastructure та поміщення в неї файлу класу під назвою NinjectDependencyResolver.cs. Кінцевий лістинг даного файлу наведений в додатку А.

Наступний крок передбачає створення в файлі App_Start/NinjectWebCommon.cs шлюзу між класом NinjectDependencyResolver, доданим в проект одним з NuGet-пакетів Ninject і підтримкою впровадження залежностей в MVC, згідно з лістингом 3.2:

Лістинг 3.2 – Шлюз з пакетом Ninject

```
// ...
private static void RegisterServices(IKernel kernel) {
                System.Web.Mvc.DependencyResolver.SetResolver(new
VStaffAdvice.WebUI.Infrastructure.NinjectDependencyResolver(kernel
));
}
```

Якщо вибрати пункт Start Debugging (Запустити відлагодження) в меню Debug (Відлагодження), відкриється сторінка повідомлення про помилку. Причина в тому, що був запитаний URL, зв’язаний з неіснуючим контролером.

15.5 Модель предметної області

Всі проекти MVC Framework починаються з моделі предметної області, оскільки навколо неї обертаються майже всі компоненти в додатку MVC Framework. Була створена всередині проекту VStaffAdvice.Domain нова папка під назвою Entities, а в ній нові файли класів C#, що називаються у відповідності з назвами таблиць бази даних, з якою безпосередньо взаємодіє додаток. Кінцева структура показана на рисунку 3.2:

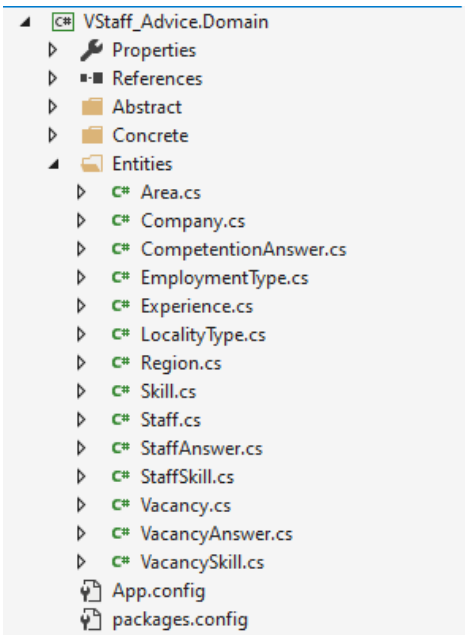


Рисунок 3.2 – Структура папки Entities проекту VStaffAdvice.Domain

Головну роль відіграє модель Vacancy, навколо якої будуються решта моделей. Зміст файлу класу Vacancy.cs приведений до вигляду, представленого в лістингу 3.3:

Лістинг 3.3 – Зміст файлу класу Vacancy.cs

```
namespace VStaffAdvice.Domain.Entities {
    public class Vacancy {
        public int VacancyId { get; set; }
        public string Name { get; set; }
        public int? CompanyId { get; set; }
        public virtual Company Company { get; set; }
        public decimal Salary { get; set; }
        public int? EmploymentTypeId { get; set; }
          public virtual EmploymentType EmploymentType { get;
set; }
        public string Description { get; set; }
        public string Advantages { get; set; }
        public string Duties { get; set; }
        public string Conditions { get; set; }
        public string Locality { get; set; }
        public int LocalityTypeId { get; set; }
        public virtual LocalityType LocalityType {get; set;}
        public int AreaId { get; set; }
        public virtual Area Area { get; set; }
         public virtual ICollection<Staff> CompetentionAnswers
{get; set;}
        public virtual ICollection<Skill> Skills {get; set;}
```

Продовження лістингу 3.3

```
public Vacancy() {
      CompetentionAnswers = new List< CompetentionAnswers>();
      Skills = new List<Skill>();
}}}
```

В даному випадку використовується підхід з визначенням моделі предметної області в окремому проекті Visual Studio, а це означає, що клас повинен бути помічений як public. Дана умова не є обов'язковою, але це допомагає зберігати модель окремо від контролерів, що корисно в великих і складних проектах.

15.6 Створення абстрактного сховища

Знадобиться якийсь спосіб отримання сутностей Vacancy з бази даних. Модель включає в себе логіку постійності для збереження і вилучення даних з постійного сховища, але навіть всередині моделі необхідно дотримуватись певного рівня розділення між сутностями моделі даних і логікою збереження та вилучення, що досягається з використанням шаблону сховища. На даному етапі не йде мова про реалізацію постійності даних, але необхідно почати процес визначення інтерфейсу для нього.

Тому було створено всередині проекту VStaffAdvice.Domain нова папка верхнього рівня під назвою Abstract, а в ній новий файл інтерфейсу IVacancyRepository, вміст якого показано в лістингу 3.4. Новий інтерфейс було додано натисканням правої кнопки миші на папці Abstract, вибором в контекстному меню пункту Add – New Item (Додати – Новий елемент) та вибором шаблону Interface.

Лістинг 3.4 – Зміст файлу інтерфейсу IVacancyRepository.cs

```
using System.Collections.Generic;
using VStaffAdvice.Domain.Entities;
namespace VStaffAdvice.Domain.Abstract {
    public interface IVacancyRepository {
        IEnumerable<Area> Areas { get; }
        IEnumerable<Company> Companies { get; }
Продовження лістингу 3.4
    IEnumerable<CompetentionAnswer> CompetentionAnswer {get;}
```

```
IEnumerable<EmploymentType> EmploymentTypes {get;}
IEnumerable<Experience> Experiences { get; }
IEnumerable<LocalityType> LocalityTypes { get; }
IEnumerable<Region> Regions { get; }
IEnumerable<Skill> Skills { get; }
IEnumerable<Staff> Staffs { get; }
IEnumerable<StaffSkill> StaffsAnswers { get; }
IEnumerable<Status> StaffsSkills { get; }
IEnumerable<Vacancy> Vacancies { get; }
IEnumerable<Vacancieskill> VacanciesAnswers { get; }
IEnumerable<Vacanciestaff> VacanciesSkills { get; }
}}
```

Даний інтерфейс використовує інтерфейс IEnumerable<T>, щоб дозволити коду, що викликається, отримувати послідовність об'єктів, нічого не повідомляючи про те, як або де зберігаються або вилучаються дані. Клас, що залежить від інтерфейсу IVacancyRepository, може отримувати об'єкти, нічого не знаючи про те, звідки вони надходять або яким чином клас реалізації буде їх доставляти. Подібним чином і визначається суть шаблону сховища.

15.7 Створення імітованого сховища

Визначивши абстрактний інтерфейс, був реалізований механізм зберігання, що був прив'язаний до бази даних, але спочатку був доданий ряд інших частин додатку. Була створена імітована реалізація інтерфейсу IVacancyRepository, що заміщала сховище даних до часу, поки воно не було до кінця реалізоване.

Імітована реалізація визначається та прив'язується до інтерфейсу IVacancyRepository в методі AddBindings() класу NinjectDependencyResolver всередині проекту VStaffAdvice.WebUI. Результат наведено в лістингу 3.5:

Лістинг 3.5 – Імітована реалізація інтерфейсу IVacancyRepository.cs

```
//...
using VStaffAdvice.Domain.Abstract;
using VStaffAdvice.Domain.Entities;
Продовження лістингу 3.5
// ...
private void AddBindings() {
```

```
                    Mock<IVacancyRepository>    mock    =    new
Mock<IVacancyRepository>();
         mock.Setup(m => m. Vacancies).Returns(new List<Vacancy>
         {
                 new Vacancy { Name = " Backend PHP Python Senior
developer", Locality = "Хмельницьке шосе, 91/2" },
               new Vacancy { Name = " Middle Front-end developer",
Locality = "вул. Червоноармійська, 5" },
                new Vacancy { Name = " Customer support specialist
with English or German", Locality = "Немирівське шосе, 78, м.
Вінниця" }});
     kernel.Bind<IVacancyRepository>().ToConstant(mock.Object);}
```

Для цього доповнення довелось додати в файл декілька просторів імен. Кожен раз, коли ядро Ninject отримує запит реалізації інтерфейсу IVacancyRepository, воно повинно повертати один і той самий імітований об’єкт, тому для встановлення відповідної області дії Ninject використовується метод ToConstant(). Замість створення нового екземпляру об’єкту реалізації в кожному випадку ядро Ninject задовольнятиме запити інтерфейсу IVacancyRepository одним і тим самим імітованим об’єктом.

15.8 Додання контролера

Для того, щоб додати контролер, необхідно натиснути правою кнопкою миші на папці Controllers в проекті VStaffAdvice.WebUI та вибрати в контекстному меню пункт Add – Controller. Вказавши варіант MVC 5 Controller – Empty (Контролер MVC 5 – Порожній) в діалоговому вікні Add Scaffold (Додання шаблону) та натиснувши на кнопці Add, в діалоговому вікні Add Controller (Додати контролер) було введено для імені контролера VacancyController і натиснута кнопка Add для створення файлу VacancyController.cs.

Окрім видалення методу дії Index() був доданий конструктор, що оголошує залежність від інтерфейсу IVacancyRepository. Це приводить до впровадження бібліотеки Ninject даної залежності для сховища вакансій та кадрів при створенні екземпляру класу контролера. В додаток також

імпортований простір імен VStaffAdvice.Domain, тому на класи сховища та моделі можна посилатись, не вказуючи їх повністю визначені імена.

Нарешті, необхідно було додати метод дії під назвою List(), який відображатиме представлення, що відображає повний список вакансій і кадрів, як наведено в лістингу 3.6:

Лістинг 3.6 – Контролер на етапі імітованого сховища

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using VStaffAdvice.Domain.Abstract;
using VStaffAdvice.Domain.Entities;
namespace VStaffAdvice.WebUI.Controllers {
    public class VacancyController : Controller {
        private IVacancyRepository repository;
        public VacancyController(IVacancyRepository repo) {
            repository = repo;
        }
        public ViewResult List() {
            return View(repository. Vacancies);
        }}}
```

Виклик методу View() подібного роду (без вказування імені представлення) повідомляє інфраструктурі про те, що потрібно відобразити стандартне уявлення для методу дії. Передаючи методу View() список об'єктів, інфраструктура постачається даними, якими необхідно заповнити об'єкт Model в строго типізованому представленні.

15.9 Компонування, файл запуску і саме представлення

Далі необхідно додати стандартне представлення для методу дії List(). Натиснувши правою кнопкою миші на методі List() в класі VacancyController та обравши в контекстному меню пункт Add View (Додати представлення) було введено в полі View Name (Ім’я представлення) ім’я List, встановлено в списку Template (Шаблон) варіант Empty (Порожній) та обрано в списку Model Class (Клас моделі) клас Vacancy, як показано на рисунку 3.3. В обов’язковому

порядку необхідно відмітити прапорець Use A Layout Page (Використовувати сторінку компонування) та натиснути кнопку Add.

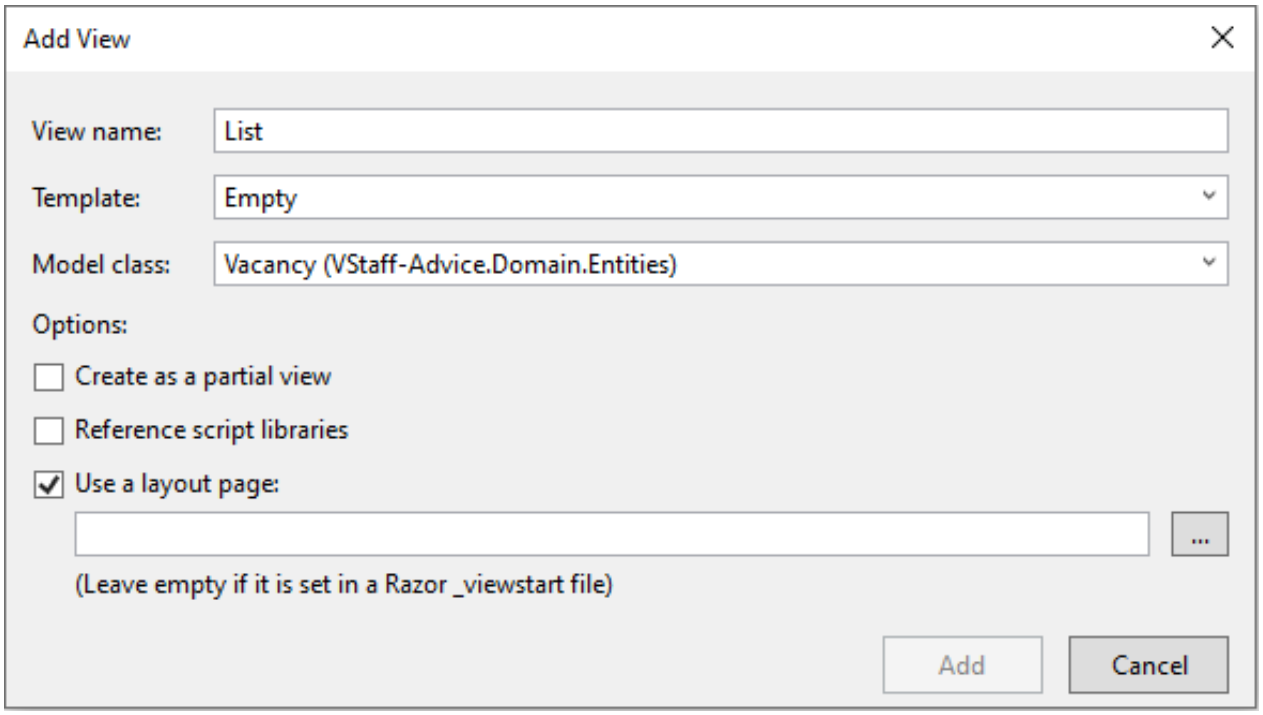


Рисунок 3.3 – Додання представлення для методу дії List

Середовище Visual Studio створило не лише файл List.cshtml, але також і файли _ViewStart.cshtml та Shared/_Layout.cshtml. Це зручна можливість, але, згідно підходу Microsoft до стандартного вмісту, в файл _Layout.cshtml включено зміст шаблону, який не використовуватиметься та є небажаним, тому компонування було модифіковано. Кінцевий варіант лістингу файлу _Layout.cshtml наведений в додатку Б.

15.10 Візуалізація даних представлення

Хоча в якості типу моделі для цього представлення був обраний клас Vacancy, в дійсності необхідно працювати з типом IEnumerable<Vacancy>, оскільки саме такі дані контролер Vacancyотримує зі сховища і передає представленню. Кінцевий варіант представлення List.cshtml наведений в додатку Б.

15.11 Встановлення стандартного маршруту

Інфраструктурі MVC Framework необхідно було вказати, що вона повинна відправляти запити, що надходять для кореневого URL додатку (http:// mysite/), методу дії List() класу VacancyController. Це виконується шляхом редагування оператора всередині методу RegisterRoutes() в файлі App_Start/RouteConfig.cs у відповідності з лістингом 3.7:

Лістинг 3.7 – Метод RegisterRoutes()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
namespace VStaffAdvice.WebUI {
    public class RouteConfig {
            public static void RegisterRoutes(RouteCollection
routes) {
     routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

                routes.MapRoute( name: "Default",
                    url: "{controller}/{action}/{id}",
                    defaults: new { controller = "Vacancy", action
= "List", id = UrlParameter.Optional });}}}
```

Дані зміни призвели до направлення запитів, що надходять до стандартного URL, методу дії List() з контролера Vacancy.

Слід звернути увагу, що в прикладі в якості значення controller вказано Vacancy, а не VacancyController, яке є іменем класу. Це частина схеми іменування ASP .NET MVC, у відповідності з якою класи контролерів завжди закінчуються словом Controller і при посиланні на клас ця частина імені опускається.

15.12 Підготовка бази даних

Для бази даних застосовується SQL Server, а доступ до неї здійснюватиметься за допомогою Entity Framework (EF) – інфраструктури ORM

для платформи .NET. Інфраструктура ORM представляє таблиці, стовпці та рядки реляційної бази даних за допомогою звичайних об’єктів C#. Слід згадати, що LINQ може працювати з різноманітними джерелами даних, і один з них – Entity Framework.

Це область, де доступний широкий діапазон інструментальних засобів і технологій. Можна взаємодіяти не тільки з різними реляційними базами даних, але також зі сховищами об’єктів, сховищами документів та іншими альтернативами. Крім того, існують інші інфраструктури ORM для .NET, кожна з яких використовує дещо відмінний підхід – подібне різноманіття дозволяє вибирати те, що найбільш підходить до конкретних проектів.

Перевага була віддана інфраструктурі Entity Framework за декількома причинами: вона проста і її легко запустити в роботу; вона прекрасно інтегрована з LINQ. Більш раннім випускам цієї інфраструктури були придатні недоліки, але поточні версії ефективні та володіють великими функціональними можливостями.

15.13 Створення бази даних

Для електронного каталогу VStaff Advice знадобиться створити базу даних в якій зберігатиметься інформація про каталог вакансій та кадрів.

Створена база даних відображена у вікні Server Explorer (Провідник Серверів) як показано на рисунку 3.4.

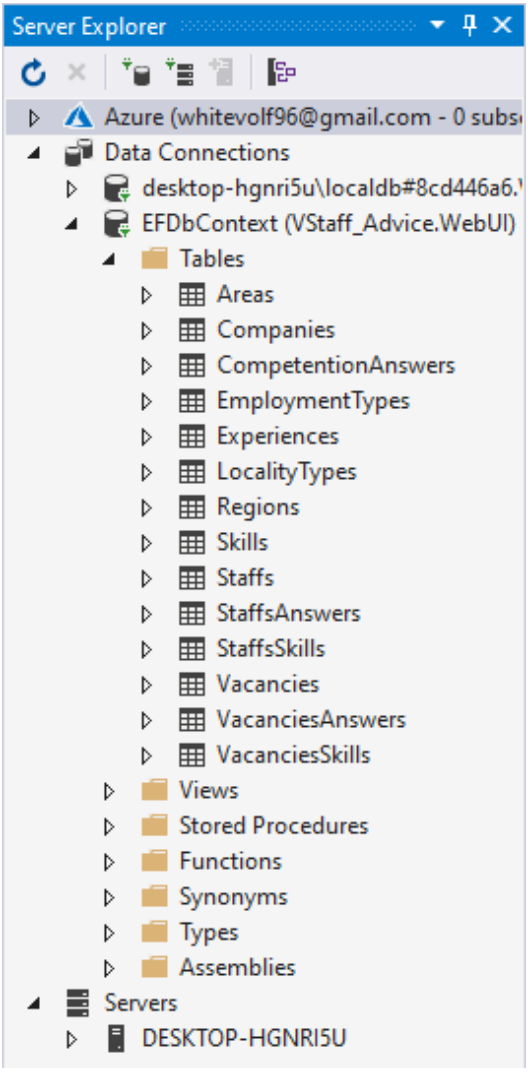


Рисунок 3.4 – База даних для електронного каталогу

В даному випадку доведеться вручну додати в базу деякі дані, щоб було з чим працювати до того, як пізніше будуть реалізовані засоби адміністрування додатку VStaff Advice (в яких можна буде мануально редагувати каталог вакансій).

15.14 Створення контексту Entity Framework

Останні версії Entity Framework включають засіб, що називається Code-First (Спочатку Код). Ідея складається в тому, що в моделі можна визначити класи, а потім на їх основі згенерувати базу даних.

Це дуже зручно для проектів, створюваних з нуля, але які представляють лише малу частину пропонованих можливостей. Тому був застосований

різновид «спочатку код», при якому класи моделі зв'язуються з існуючою базою даних. Необхідно обрати пункт меню Tools – Library Package Manager – Package Manager Console, щоб відкрити вікно командного рядка NuGet та ввести код, наведений в лістингу 3.8:

Лістинг 3.8 – Підключення Entity Framework

```
Install-Package EntityFramework -projectname
VStaffAdvice.Domain
Install-Package EntityFramework -projectname
VStaffAdvice.WebUI
```

Ці команди додають пакет Entity Framework до рішення. Один і той самий пакет необхідно додати в проекти Domain і WebUI, що дозволить створювати класи, що отримуватимуть доступ до бази даних у вказаних проектах.

Наступним кроком було створення класу контексту, котрий асоціюватиме модель з базою даних. В проекті VStaffAdvice.Domain була створена нова папка під назвою Concrete і всередині неї був поміщений новий файл класу під назвою EFDbContext.cs. Кінцевий код даного файлу наведений в додатку А.

Щоб скористатись перевагою підходу «спочатку код», необхідно створити клас, похідний від System.Data.Entity.DbContext. Цей клас потім автоматично визначає властивість для кожної таблиці бази даних, з якою необхідно буде працювати.

Ім'я властивості вказує таблицю, а параметр типу результату DbSet – тип моделі, який інфраструктура Entity Framework повинна використовувати для представлення рядків в цій таблиці. Далі Entity Framework необхідно вказати, як підключитись до бази даних, це виконується шляхом додавання в файл Web.config проекту VStaffAdvice.WebUI рядка підключення до бази даних з таким самим ім'ям, як у класу контексту, як показано в лістингу 3.9:

Лістинг 3.9 – Підключення до бази даних

```
<connectionStrings><add name = "EFDbContext" connectionString
= "Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename =
```

Продовження лістингу 3.9

```
|DataDirectory|\ VStaff-Advice.mdf; Integrated Security = True;
multipleactiveresultsets = True" providerName =
"System.Data.SqlClient"/></connectionStrings>
```

В розділі connectionStrings файлу Web.config також виявився присутнім ще один елемент add. Середовище Visual Studio створює цей елемент за замовчуванням і він може бути проігнорованим або видаленим, як було продемонстровано вище.

15.15 Створення сховища для об'єктів

Далі необхідно додати в папку Concrete проекту VStaffAdvice.Domain файл класу з назвою EFVacancyRepository.cs. Повний зміст даного файлу наведений в додатку А.

Клас EFVacancyRepository представляє необхідне сховище. Він реалізує інтерфейс IVacancyRepository і використовує екземпляр EFDbContext для вилучення даних з бази посередництвом Entity Framework.

Для застосування нового класу сховища знадобиться відредагувати прив'язки Ninject та замінити імітоване сховище даних прив'язкою до реального сховища. Для цього було змінено вміст файлу класу NinjectDependencyResolver.cs з проекту VStaffAdvice.WebUI, щоб метод AddBindings() виглядав відповідно до лістингу 3.10:

```
Лістинг 3.10 – метод AddBindings() з реальним сховищем

// ...
using VStaffAdvice.Domain.Concrete;
namespace VStaffAdvice.WebUI.Infrastructure {
     public       class       NinjectDependencyResolver       :
IDependencyResolver {
         // ...
          private void AddBindings() {
kernel.Bind<IVacancyRepository>().To<EFVacancyRepository>();}}
}
```

Нова прив’язка вказує Ninject про те, що для обслуговування запитів до інтерфейсу IVacancyRepository необхідно створювати екземпляри класу EFVacancyRepository.

Такий підхід з використанням Entity Framework для представлення бази даних SQL Server у вигляді послідовності об’єктів моделей відрізняється простотою і легкістю, дозволяючи зосередити всю увагу на інфраструктурі MVC Framework.

16 Реалізація розмежування контенту на сторінки

Після реалізації сховища даних і створення тестового представлення List.cshtml, в якому виводиться список всіх вакансій та кадрів, необхідно додати підтримку розбивки на сторінки, щоб представлення відображало на сторінці задану кількість об’єктів і користувач міг переходити від однієї сторінки до іншої з метою перегляду всього каталогу.

Для цього в метод дії List() контролера Vacancy було додано нові параметри (лістинг 3.11):

```
Лістинг 3.11 – модифікація методу дії List()

namespace VStaffAdvice.WebUI.Controllers {
    public class VacancyController : Controller {
        private IVacancyRepository repository;
        public int pageSize = 10;
        public VacancyController(IVacancyRepository repo) {
   repository = repo;}
public ViewResult List(int page = 1) {
            return View(repository. Vacancies
                .OrderBy(vacancy => vacancy. VacancyId)
  .Skip((page - 1)*pageSize)
                .Take(pageSize));
        }}}
```

Поле PageSize вказує, що на одній сторінці повинні відображатись свідчення про 10 вакансій або кадрів.

В метод List() був доданий необов’язковий параметр. Це означає, що в разі виклику методу без параметру List() виклик імітує обробку так, ніби йому

було передано значення, вказане в визначенні параметру за замовчуванням List(1). В результаті метод дії відображає першу сторінку свідчень про об’єкти, коли MVC Framework викликає його без аргументу. Всередині тіла методу дії ми отримуємо об’єкти Vacancy, впорядковуємо їх за первинним ключем, пропускаємо об’єкти, які розташовуються до початку поточної сторінки і вибираємо ту кількість вакансій або кадрів, яка вказана в полі PageSize.

16.1 Відображення посилань на сторінки

При запуску додатку на сторінці відображаються лише 10 позицій каталогу. Якщо необхідно переглянути іншу сторінку, в кінець URL необхідно додати параметри рядка запиту, наприклад http://localhost:53985/?page=2.

Певна річ, у користувачів немає ніякого способу вияснити про існування цих параметрів рядка запиту і навіть якби він був, користувачі, скоріше за все, не захотіли б виконувати навігацію в подібному стилі.

Замість цього необхідно відобразити в нижній частині кожного списку об’єктів посилання на певні сторінки, щоб користувачі могли переходити між сторінками. Для цього і було реалізовано багатократно використовуваний допоміжний метод HTML, аналогічний методам Html.TextBoxFor() і Html.BeginForm(). Цей допоміжний метод генеруватиме HTML-розмітку для необхідних навігаційних посилань.

16.2 Додання моделі представлення

Щоб забезпечити підтримку нового допоміжного методу HTML, було передано представленню інформацію про кількість доступних сторінок, поточної сторінки і загальної кількості об’єктів в сховищі, використавши модель представлення – в папку Models проекту VStaffAdvice.WebUI був доданий клас з назвою PagingInfo, код якого наведений в лістингу 3.12:

Лістинг 3.12 – Модель представлення PagingIfo

```
using System;
namespace VStaffAdvice.WebUI.Models {
    public class PagingInfo {
        public int TotalItems { get; set; }
        public int ItemsPerPage { get; set; }
        public int CurrentPage { get; set; }
        public int TotalPages {
           get {return (int)Math.Ceiling((decimal)TotalItems /
ItemsPerPage); }
        }}}
```

Модель представлення не є частиною моделі предметної області. Це всього лиш зручний клас для передачі даних між контролером і представленням. Щоб підкреслити вказану обставину, клас PagingInfo був поміщений в проект VStaffAdvice.WebUI для збереження його окремо від класів моделі предметної області.

16.3 Додання допоміжного методу HTML

Маючи модель представлення, можна реалізувати допоміжний метод HTML, який називається PageLinks(). У проекті VStaffAdvice.WebUI була створена нова папка HtmlHelpers і в неї поміщений новий файл класу під назвою PagingHelper.cs. Зміст даного файлу наведений в додатку Б.

Метод розширення PageLinks() генерує HTML-розмітку для набору посилань на сторінки з використанням інформації, наданої в об’єкті PagingInfo. Параметр Func приймає делегат, що застосовується для генерації посилань, що забезпечують перегляд інших сторінок.

Метод розширення доступний для застосування тільки коли простір імен, що його містить, знаходиться у видимій області. В файлі коду це досягається за допомогою оператора using, але для представлення Razor повинний бути доданий конфігураційний запис в файл Web.config, або оператор @using до самого представлення.

Дещо заплутує те, що проект Razor MVC містить два файли Web.config: головний файл в кореневому каталозі проекту додатку і файл, специфічний для

представлення і знаходиться в папці Views. Зміни необхідно внести в файл Views/Web.config, як наведено в лістингу 3.13:

Лістинг 3.13 – Виявлення методу розширення

```
<system.web.webPages.razor>
<host factoryType="System.Web.Mvc.MvcWebRazorHostFactory,
System.Web.Mvc, Version=5.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" />
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Routing" />
      <add namespace=" VStaffAdvice.WebUI" />
      <add namespace="VStaffAdvice.WebUI.HtmlHelpers "/>
    </namespaces>
  </pages>
</system.web.webPages.razor>
```

Таблиця рекомендацій кандидатів на посади реалізована у відповідності з рисунком 3.5.

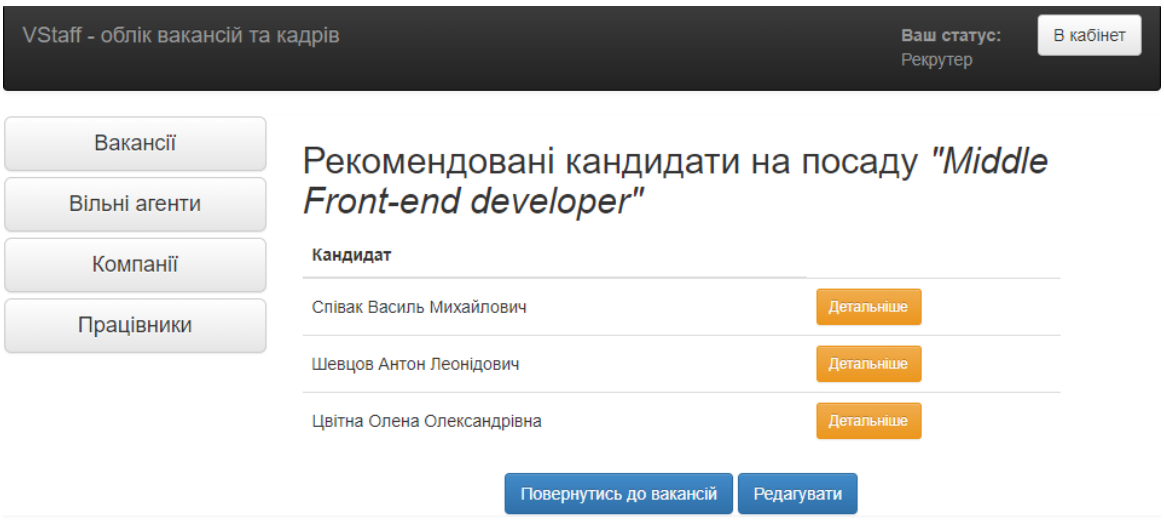


Рисунок 3.5 – Представлення для списку кандидатів

Кожен простір імен, на які необхідно посилатись в представленні Razor для явного його використання, повинен бути оголошений в файлі Views/Web.config або застосовуватись за допомогою виразу @using.

16.4 Додання даних моделі представлення

Необхідно надати представленню екземпляр класу моделі представлення PagingInfo. Це можна було б зробити з застосуванням об'єкту ViewBag, але краще помістити всі дані, що необхідно передати з контролера представленню, в один клас моделі представлення. Для цього в папку Models проекту VStaffAdvice.WebUI був доданий новий файл класу з назвою VacancyListViewModel.cs, зміст якого наведений в лістингу 3.14:

```
Лістинг 3.14 – Клас моделі представлення

using System.Collections.Generic;
using VStaffAdvice.Domain.Entities;
namespace VStaffAdvice.WebUI.Models {
    public class VacanciesListViewModel {
        public IEnumerable<Vacancy> Vacancies { get; set; }
        public PagingInfo PagingInfo { get; set; }
    }}
```

Після цього можна оновити метод дії List() класу VacancyController так, щоб він використовував клас VacanciesListViewModel для постачання представлення свідченнями про об’єкти, відображувані на сторінках, та інформацією про розбивку на сторінки.

Внесені зміни забезпечують передачу об’єкту VacanciesListViewModel представленню в якості даних моделі.

У файлі представлення List.cshtml директива @model була змінена для вказання Razor на те, що тепер робота виконується з іншим типом даних. Знадобилось також оновити цикл foreach, щоб джерелом даних виступала властивість Vacancies моделі даних.

16.5 Відображення посилань на сторінки

Необхідно викликати допоміжний метод HTML всередині представлення, як показано в лістингу 3.15:

Лістинг 3.15 – Виклик всередині представлення

```
<div>        @Html.PageLinks(Model.PagingInfo,        x        =>
Url.Action("List", new { page = x }))</div>
```

Процес додання нового користувача до бази відображений на рисунку 3.6.

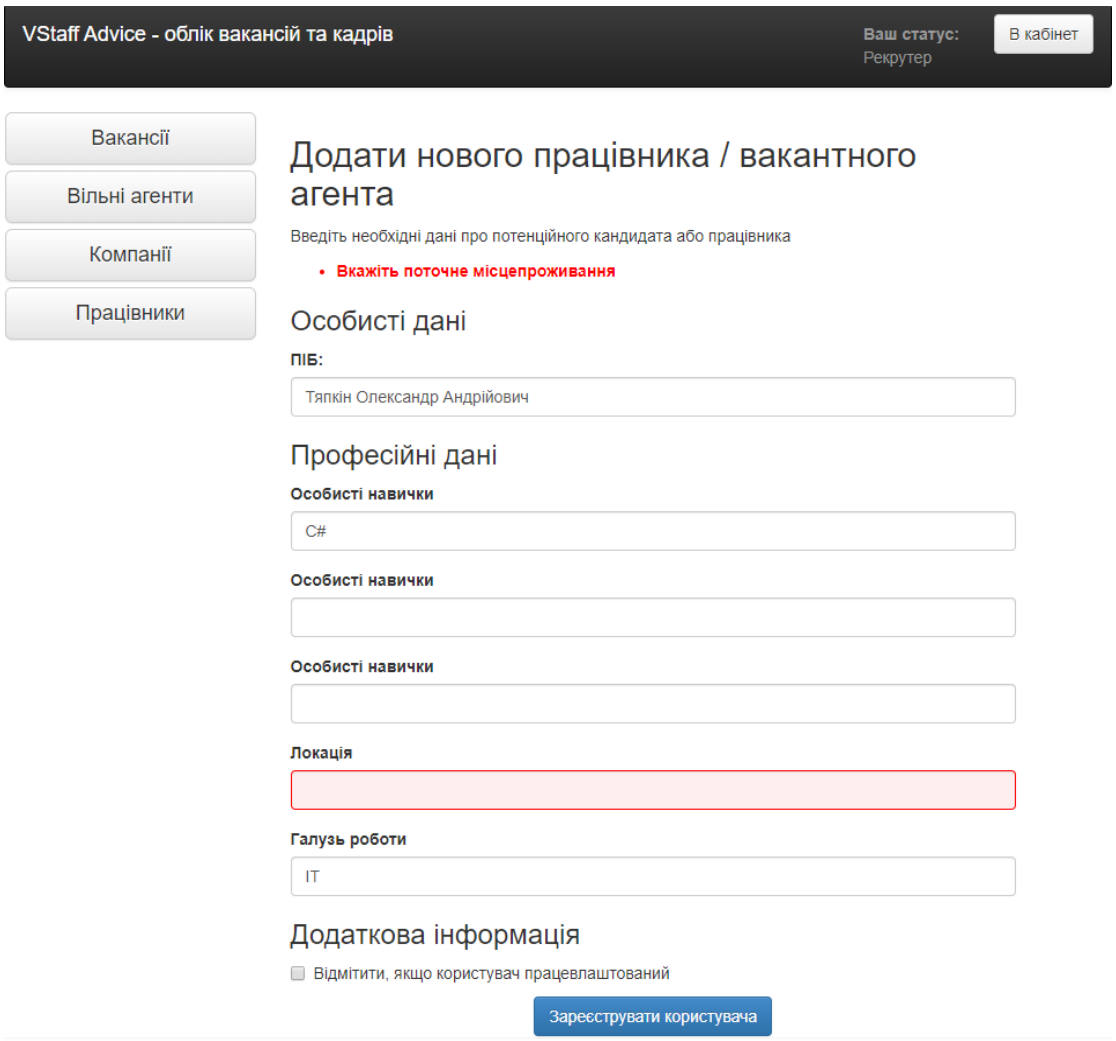


Рисунок 3.6 – Реєстрація користувача

Запустивши додаток, можна буде спостерігати нові посилання на сторінки. Спочатку стиль відображення доволі примітивний, але в подальшому він буде покращений. На даному ж етапі важливо те, що посилання дозволяють переміщуватись по сторінках каталогу та знайомитись з опублікованими вакансіями.

16.6 Покращення URL

Посилання на сторінки працюють, але для передачі інформації серверу вони досі використовують рядок запиту http://localhost:53985/?page=2.

Отримати більш привабливий URL можна, спеціально створивши схему, яка відповідає шаблону компонованих URL. Компонований URL – це URL, зрозумілий користувачу: http://localhost:53985/Page2.

Інфраструктура MVC дозволяє легко змінювати схему URL в додатку, оскільки вона користується засобом маршрутизації ASP .NET. Все, що знадобилось зробити – це додати новий маршрут до методу RegisterRoutes() в файлі RouteConfig.cs, що знаходиться в папці App_Start проекту VStaffAdvice.WebUI. Зміни, внесені в даний файл, наведені в лістингу 3.16:

Лістинг 3.16 – Додання нового маршруту до схеми URL

```
public static void RegisterRoutes(RouteCollection routes) {
          routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
          routes.MapRoute(
              name: null,
              url: "Page{page}",
              defaults: new { controller = "Vacancy", action =
"List" });
          routes.MapRoute(
              name: "Default",
              url: "{controller}/{action}/{id}",
              defaults: new { controller = "Vacancy", action =
"List", id = UrlParameter.Optional });}
```

Важливо помістити цей маршрут перед стандартним (Default) маршрутом, який вже присутній в файлі, так як система маршрутизації опрацьовує маршрути в порядку їх перерахування, в даному проекті необхідно, щоб новий маршрут мав перевагу над існуючим.

Це єдина зміна, яку довелось внести, щоб змінити схему URL для розбивки на сторінки даних про вакансії та кадри. Інфраструктура MVC Framework тісно інтегрована з функцією маршрутизації, тому додаток автоматично віддзеркалює зміни подібного роду всередині результату,

генерованого методом Url.Action(), що застосовувався в представленні List.cshtml для генерації посилань на сторінки.

17 Стилізація контенту електронного каталогу

Для надання стилів CSS, що будуть застосовані до додатку, було сплановано використання пакету Bootstrap. Для його встановлення необхідно обрати пункт Package Manager – Package Manager Console в меню Tools середовища Visual Studio. Відкриється вікно командного рядка NuGet, де необхідно ввести одну команду - Install-Package -version 3.0.0 bootstrap -projectname VStaffAdvice.WebUI.

Це базова команда NuGet для додання пакету Bootstrap з додатковим аргументом projectname, який забезпечує додання інструментом NuGet файлів в правильний проект.

17.1 Застосування стилів Bootstrap до компонування

При створенні представлення List.cshtml для контролера Vacancy був відмічений прапорець Use a layout page (Використати сторінку компонування), але поле, в якому вказувалось компонування, залишилось порожнім. Це забезпечує використання компонування, заданого в файлі Views/ _ViewStart.cshtml, який Visual Studio створює автоматично поряд з представленням. Зміст файлу запуску представлення показано в лістингу 3.17:

Лістинг 3.17 – Файл запуску представлення

```
@{ Layout = "~/Views/Shared/_Layout.cshtml"; }
```

Значення властивості Layout вказує, що представлення використовуватимуть в якості компонування файл Views/Shared/_Layout.cshtml, якщо тільки не буде явно задана альтернатива. Вміст файлу _Layout.cshtml було скинуто раніше, щоб видалити те, що додав шаблон Visual Studio. Кінцевий вміст файлу _Layout.cshtml наведений у додатку Б.

17.2 Проблема зі стилізацією елементів

Елементи HTML, генеровані додатком MVC, надходять з різних джерел (статичний зміст, вирази Razor, допоміжні методи HTML і так далі), тому класи стилів виявляються розкиданими по всьому проекту. Зміщування стилів CSS з генерацією елементів – не особливо вдала ідея, що іде в розріз з принципом розділення не пов'язаної функціональності, притаманного архітектурному шаблону MVC.

Ситуацію можна покращити за рахунок призначення класів, відмінних від Bootstrap, елементам на основі їх ролі в додатку і подальшого використання бібліотеки, подібної jQuery або LESS, для відображення спеціальних класів на класи Bootstrap.

Щоб спростити додаток, був прийнятий підхід з впровадженням класів Bootstrap всюди в додатку, навіть з урахуванням того, що це ускладнить зміну стилів в майбутньому.

17.3 Створення часткового представлення

Далі було створено часткове представлення (partial view), що є фрагментом вмісту, який можна впровадити в інше представлення подібно шаблону. Часткові представлення містяться всередині власних файлів і можуть багатократно використовуватись різноманітними представленнями, що сприяє зменшенню дублювання, особливо у випадку, коли необхідно відобразити однорідні дані в кількох місцях додатку.

Щоб додати часткове представлення, треба натиснути праву кнопку миші на папці /Views/Shared в проекті VStaffAdvice.WebUI і обрати в контекстному меню пункт Add – View (Додати – Представлення). В полі View Name (Ім'я представлення) ввести VacanciesSummary, в списку Template (Шаблон) вказати Empty (Порожній), в випадаючому списку Model Class (Клас моделі) обрати Vacancy і відмітити прапорець Create As A Partial View (Створити як часткове представлення), як показано на рисунку 3.7.

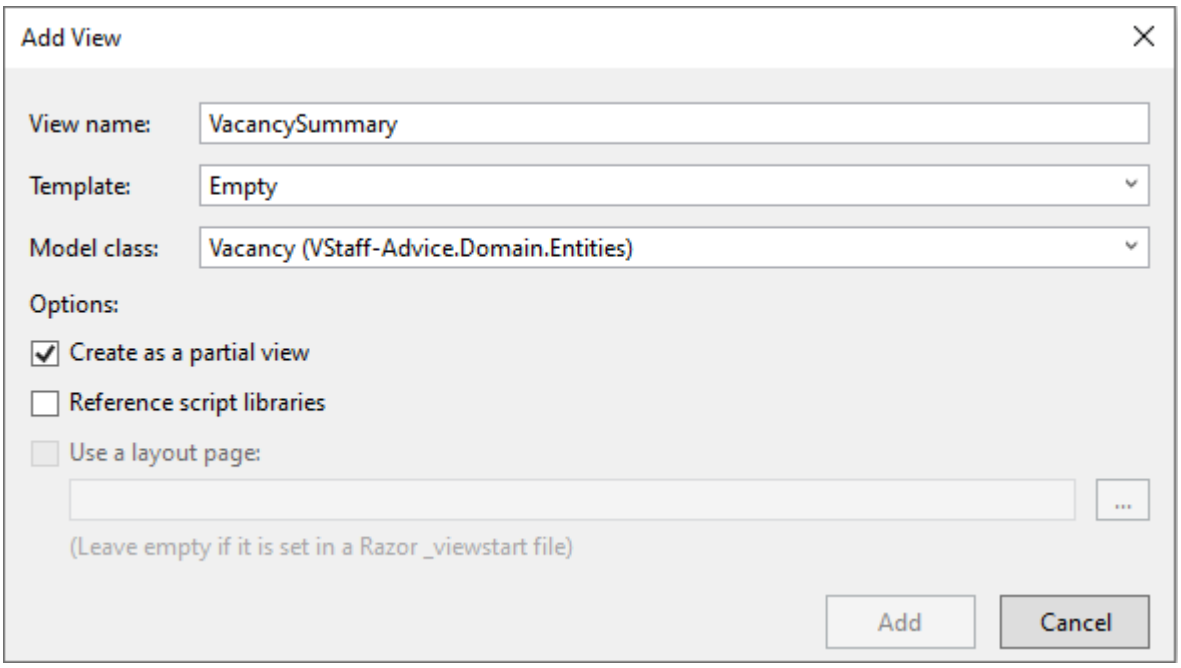


Рисунок 3.7 – Створення часткового представлення

Часткове представлення схоже на звичайне представлення за виключенням того, що воно породжує фрагмент HTML-розмітки, а не повний HTML-документ. Якщо відкрити представлення VacanciesSummary, можна помітити, що воно містить тільки директиву @model, в якій встановлений клас моделі предметної області під назвою Vacancy. Вміст файлів VacanciesSummary.cshtml та Views/Vacancies/List.cshtml наведений в додатку Б.

18 Розробка навігації для організації записів

Попередні дії будували основну інфраструктуру додатку VStaffAdvice. У наступному кроці дана інфраструктура буде використана для додавання функціональних засобів до додатку. І розпочати слід з навігаційних елементів керування – додаток VStaff Advice набагато зручніший, коли в ньому надається користувачам можливість навігації по вакансіям та кадрам з урахуванням ряду параметрів. Ця робота складається з трьох частин:

— розширення методу дії List() в класі VacancyController, щоб він отримував можливість фільтрувати об’єкти Vacancy в сховищі;

— розширення схеми URL та перегляд схеми маршрутизації;

— створення списку фільтрів, розміщених в бічній панелі сайту, які дозволятимуть зі списків обирати бажані параметри об’єктів.

18.1 Фільтрування списку вакансій та кадрів

Розпочати необхідно з розширення класу моделі представлення VacanciesListViewModel, що був доданий до проекту VStaffAdvice.WebUI. Необхідно забезпечити взаємодію поточних параметрів з представленням, щоб відобразити бічну панель. Лістинг 3.18 демонструє зміни, внесені в файл VacanciesListView.cs.

Лістинг 3.18 – Взаємодія поточних параметрів з представленням

```
using System.Collections.Generic;
using VStaffAdvice.Domain.Entities;
namespace VStaffAdvice.WebUI.Models {
public class VacancyListViewModel {
    public IEnumerable<Vacancy> Vacancies { get; set; }
    public PagingInfo PagingInfo { get; set; }
    public string CurrentRegion { get; set; }
    public string CurrentArea { get; set; }
    public string CurrentLocalityType { get; set; }
    public string CurrentBranch { get; set; }}}
```

Наступний крок полягає в оновленні класу VacancyController, щоб метод дії List() фільтрував об’єкти Vacancy за параметрами і використовував додані в модель представлення властивості для вказання необхідних користувачу параметрів, обраних в даний момент.

В даний метод дії були внесені три зміни. Перша – додані нові параметри. Вони використовуються другою зміною, що представляє собою розширення запиту LINQ. Якщо значення параметрів не рівне null, значить, обрані лише ті об’єкти Vacancy, що відповідають значенню властивостей. Остання, третя, зміна стосується встановлення значення параметрів, що були додані в клас VacanciesListViewModel.

18.2 Поліпшення схеми URL

Мало хто бажає бачити або користуватись незручними URL типу «/?company=АЛЛО». Для вирішення даної проблеми необхідно переглянути схему маршрутизації. Для реалізації нової схеми необхідно модифікувати метод RegisterRoutes() в файлі App_Start/RouteConfig.cs. Кінцевий стан файлу App_Start/RouteConfig.cs наведений в додатку Б.

В таблиці 3.3 описана схема URL, яку надають нові маршрути.

Таблиця 3.3 – Зведення по маршрутам

URL	Що робить
/	Виводить першу сторінку списку вакансій або кадрів усіх категорій
/Page2	Виводить 2 сторінку, відображаючи об’єкти всіх категорій
/Вільний_для_найму	Відображає першу сторінку елементів з вказаним параметром
/Вільний_для_найму/Page2	Відображає 2 сторінку елементів з вказаним параметром

Система маршрутизації ASP .NET застосовується інфраструктурою MVC для обробки вхідних запитів від користувачів, а також генерує вихідні URL, що відповідають схемі URL і тому можуть бути вбудовані у веб-сторінки. Використання системи маршрутизації для обробки вхідних запитів і генерації вихідних URL дозволяє гарантувати узгодженість усіх URL в додатку.

Вигляд вікна додатку для окремо взятого виду користувача представлений на рисунку 3.8.

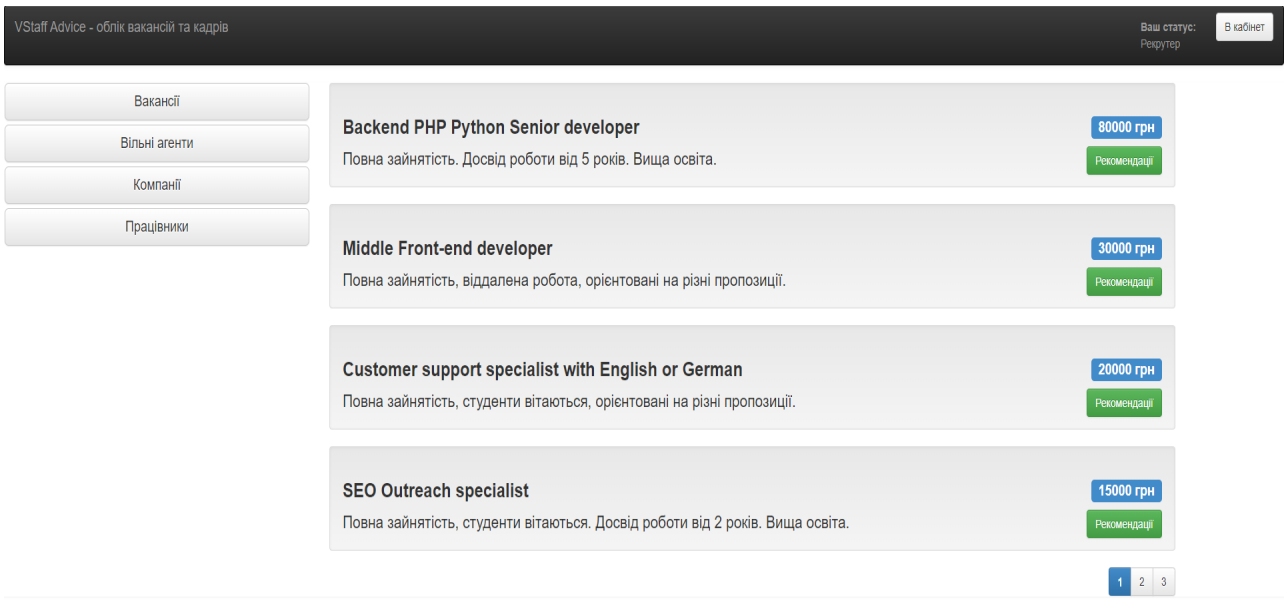


Рисунок 3.8 – Зовнішній вигляд вікна додатку «VStaff Advice»

Метод Url.Action() – це найбільш зручний спосіб генерації вихідних посилань. Цей метод застосовувався в представленні List для відображення посилань на сторінки. Тепер, коли додана підтримка фільтрації за параметрами, цьому методу необхідно передати відповідну інформацію, згідно з лістингом 3.19:

Лістинг 3.19 – Передача інформації про маршрути

```
<div class="btn-group pull-right">
  @Html.PageLinks(Model.PagingInfo, x => Url.Action("List",
  new { page = x, region = Model.CurrentRegion,
            area = Model.CurrentArea,
localityType = Model.CurrentLocalityType}))
    </div>
```

До внесення цих змін генеровані посилання на сторінки виглядали, відповідно http://<сервер>:<порт>/Page1.

Якщо користувач натисне на сторінковому посиланні по типу цього, застосований раніше фільтр за категоріями втрачається, і буде виведена сторінка, що містить усі об’єкти. За рахунок додавання поточних параметрів, що отримуємо з моделі представлення, генеруються URL по типу http://<сервер>:<порт>/Вінницький/Page1.

Коли користувач натискає на подібному посиланні, поточні параметри передаються методу дії List() і фільтрація зберігається. Після цієї зміни при відвідуванні URL вигляду «/Користувач» сторінкові посилання в нижній частині будуть коректно включати параметри.

18.3 Побудова меню навігації за категоріями

Необхідно надати користувачам можливість вибору параметрів, не передбачаючи введення будь-чого в URL. Це означає, що необхідно показати список доступних параметрів з відміченими поточними параметрами, якщо вони є. Після побудови додатку цей список параметрів буде застосовуватись в більш ніж одному контролері, тому він повинен бути самодостатнім та багатократно використовуваним.

Інфраструктура ASP .NET MVC Framework підтримує концепцію дочірніх дій, що ідеально підходять для створення компонентів, таких як багатократно використовуваний навігаційний елемент керування. Дочірня дія покладається на допоміжний метод HTML з іменем Html.Action(), який дозволяє включати в поточне представлення вивід з довільного методу дії.

В даному випадку ми можемо створити новий контролер (під назвою NavController) з методом дії (в даному випадку Menu()), котрий відображає навігаційне меню. Потім посередництвом допоміжного методу Html.Action() вивід з цього методу вбудовується в компонування.

Такий підхід дозволяє отримати реальний контролер, що може містити будь-яку необхідну прикладну логіку і який можна наражати модульному тестуванню подібно будь-якому іншому контролеру. Це дійсно зручний спосіб побудови невеликих сегментів додатку при збереженні загального підходу, прийнятого в MVC Framework.

18.4 Створення контролера навігації

Необхідно натиснути правою кнопкою миші на папці Controllers в проекті VStaffAdvice.WebUI і обрати в контекстному меню пункт Add – Controller

(Додати – Контролер), вказати варіант MVC 5 Controller – Empty (Контролер MVC 5 – Порожній) в діалоговому вікні Add Scaffold (Додання шаблону). В діалоговому вікні Add Controller (Додання контролера) ввести для імені контролера NavController. Далі треба видалити метод Index(), який Visual Studio за замовчуванням додає до нових контролерів, і додати метод дії Menu().

Необхідно відредагувати файл Views/Shared/_Layout.cshtml, додавши до нього виклик допоміжного методу Html.Action(), у відповідності з лістингом 3.20:

Лістинг 3.20 – Додання допоміжного методу до шаблону

```
<div class="row panel">
        <div id="categories" class="col-xs-3">
            @Html.Action("Menu", "Nav")
        </div>
        <div class="col-xs-8">
            @RenderBody()
        </div>
    </div>
```

В якості параметра цьому методу передається ім’я методу дії для виклику (Menu()) і контролер, що його містить (Nav).

18.5 Генерування списків параметрів

Все, що планується зробити в методі дії Menu() – це створити список параметрів, відповідно до лістингу 3.21:

Лістинг 3.21 – Створення списку параметрів в методі

```
public class NavController : Controller {
        private IVacancyRepository repository;
        public NavController(IVacancyRepository repo)
{
Продовження лістингу 3.21
            repository = repo;
        }
        public PartialViewResult Menu(string region = null){
  ViewBag.SelectedRegion = region;
           IEnumerable<string> regions = repository. Vacancies
```

```
                .Select(vacancy => vacancy.Area.Region.Name)
                .Distinct().OrderBy(x => x);
            return PartialView(regions);
        }
```

Перша зміна пов’язана доданням конструктора, який приймає в якості свого аргументу реалізацію IVacancyRepository. Результатом є оголошення залежності, яку Ninject розпізнаватиме при створенні екземплярів класу NavController. Друга зміна стосується методу дії Menu(), який тепер використовує запит LINQ для отримання списку параметрів зі сховища і передачі їх представленню. Слід звернути увагу, що оскільки робота в цьому контролері виконується з частковим представленням, в методі дії викликається метод PartialView(), а результатом є об’єкт PartialViewResult().

18.6 Створення представлення

Щоб створити представлення для методу дії Menu(), треба натиснути правою кнопкою миші на папці Views/Nav і обрати в контекстному меню пункт – MVC 5 View Page (Razor) (Додати – Сторінка представлення MVC 5 (Razor)), вказати в якості імені Menu. Видалити вміст, який середовище Visual Studio додає до нових представлень і привести представлення до відповідності з лістингом 3.22:

Лістинг 3.22 – Представлення меню навігації

```
@model IEnumerable<string>
<div    class="dropdown"><button    class="btn    btn-primary
dropdown-toggle btn-lg" type="button" data-toggle="dropdown"
style="width:  250px;  margin-bottom:  10px">Оберіть  область<span
class="caret"></span></button>
    <ul class="dropdown-menu">
        <li>@Html.ActionLink("Yci", "List", "Vacancy", null,
```

Продовження лістингу 3.22

```
        new { @class = "" })</li>
            @foreach (var link in Model) {
                <li>@Html.RouteLink(link, new {
              controller = "Vacancy", action = "List", region =
link, page = 1 }, new { @class = «»
        + (link == ViewBag.SelectedRegion ? « btn-primary» : «»)
```

```
})</li> } </ul> </div>
```

Було додано посилання під назвою «Усі», що відображатиметься зверху списку параметрів і переміщувати користувача на першу сторінку всіх об’єктів, не відфільтрованих за параметрами. Це виконується з допомогою допоміжного методу ActionLink(), що генерує HTML-елемент <a> з використанням раніше сформованої інформації маршрутизації.

Потім здійснюється прохід по іменам параметрів і створення посилань для кожного параметру з застосуванням методу RouteLink().

Зовнішній вигляд панелі адміністрування, створеної на основі CRUD-контролера, наведений на рисунку 3.9.

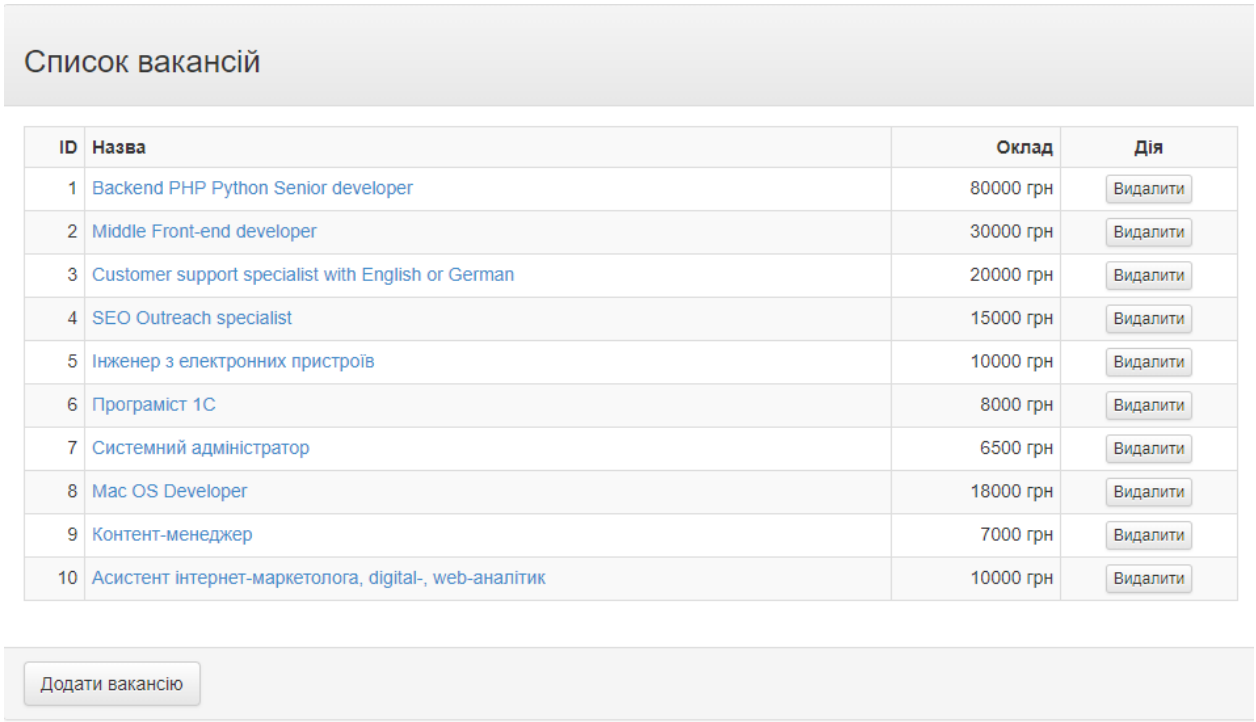


Рисунок 3.9 – Панель адміністрування

Створений адміністративний блок дозволяє користувачу з розширеним рівнем доступу (адміністратору) керувати контентом додатку та взаємодіяти з даними, що вносяться іншими користувачами.

19 Висновок

Базуючись на інформації, отриманій під час аналізу проблематики розробки додатку та визначившись з теоретичною базою для роботи, використовуючи обрані програмні засоби та інструменти, було спроектовано та реалізовано базу даних з можливістю корпоративної інтеграції і додаток для обліку вакансій та кадрів в якості представлення для бази даних.

РОЗДІЛ 4
ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО ДОДАТКУ

20 Модульне тестування функціоналу розбиття на сторінки

Модульне тестування засобу розбиття на сторінки можна провести, створивши імітоване сховище, впровадивши його в конструктор класу VacancyController та викликавши метод List(), щоб запитати конкретну сторінку. Потім отримані об’єкти Vacancy можна порівняти з тими, що очікувались від тестових даних в імпортованій реалізації.

Створений для цієї мети модульний тест, що знаходиться в файлі UnitTest1.cs проекту VStaffAdvice.UnitTests, наведений в додатку В.

Щоб отримати дані, що повертаються методом контролера, було виконано звернення до властивості Model об’єкту результату, щоб отримати послідовність IEnumerable<Vacancy>, сформовану методом List(). Потім виконується перевірка на відповідність цих даних очікуваним. В даному випадку було перетворено послідовність в колекцію за допомогою ToList() та перевірено довжину і значення окремих об’єктів.

Щоб протестувати допоміжний метод PageLinks(), ми викликаємо метод з тестовими даними та порівнюємо результати з очікуваною HTML-розміткою [16]. Метод модульного тесту має вигляд, наведений в додатку В.

Цей тест перевіряє виведення допоміжного методу з використанням літерального рядкового значення, що містить подвійні лапки. Мова C# дозволяє успішно працювати з подібними рядками. Перед рядком необхідно вставити символ @ та застосовувати два набори подвійних лапок («») замість одного. В даному випадку також забороняється розносити літеральний рядок по декількох рядках файлу, якщо лише рядок, з яким відбувається порівняння, не рознесений аналогічним способом.

Далі необхідно впевнитись, що контролер відправляє представленню правильну інформацію про розбиття на сторінки. Модульний тест, доданий в тестовий проект для забезпечення даної перевірки, наведений в додатку В.

Крім того, потрібно змінити раніше створений модульний тест для перевірки розбиття на сторінки, що міститься в методі Can_Paginate(). Він покладається на метод дії List(), що повертає об'єкт ViewResult, властивістю Model якого є послідовність об'єктів Staff, але ці дані були поміщені всередину іншого типу моделі представлення. Після переробки тест отримує вигляд, представлений в лістингу 4.1:

Лістинг 4.1 – Оновлений вигляд методу Can_Paginate()

```
[TestMethod]
public void Can_Paginate()
{
    // Організація (arrange)
                Mock<IStaffRepository>    mock    =    new
Mock<IStaffRepository>();
     mock.Setup(m => m.Staffs).Returns(new List<Staff>
     {
         new Staff { StaffId = 1, Name = "Працівник1"},
         new Staff { StaffId = 2, Name = "Працівник2"},
         new Staff { StaffId = 3, Name = "Працівник3"},
         new Staff { StaffId = 4, Name = "Працівник4"},
         new Staff { StaffId = 5, Name = "Працівник5"}
     });
                  StaffController    controller    =    new
StaffController(mock.Object);
     controller.pageSize = 3;
     // Дія (act)
                          StaffsListViewModel      result      =
(StaffsListViewModel)controller.List(2).Model;
    // Твердження
     List<Staff> staffs = result. Staffs.ToList();
     Assert.IsTrue(staffs.Count == 2);
     Assert.AreEqual(staffs[0].Name, "Игра4");
     Assert.AreEqual(staffs[1].Name, "Игра5");
}
```

Враховуючи деяку схожість між цими двома тестовими методами, можна було б створити загальний метод початкового встановлення.

21 Модульне тестування функціоналу категорій записів

Було змінено сигнатуру методу дії List(), тому деякі існуючі методи модульного тестування перестали компілюватись. Для вирішення цієї проблеми

в модульних тестах, що працюють з контролером, методу дії List() необхідно передавати в першому параметрі значення null [16]. В тестовому методі Can_Paginate() розділ дії повинен виглядати у відповідності до лістингу 4.2:

Лістинг 4.2 – Адаптований метод Can_Paginate()

```
[TestMethod]
public void Can_Paginate()
{
    // ... Організація (arrange)
    // Дія (act)
                        StaffsListViewModel      result      =
(StaffsListViewModel)controller.List(null, 2).Model;
    // ... Твердження
}
```

За рахунок використання null отримуються всі об’єкти Staff, які контролер вилучає зі сховища, що відтворює ситуацію, що існувала перед доданням нового параметру. Такого ж роду зміни доведеться внести і в тестовий метод Can_Send_Pagination_View_Model() (див. лістинг 4.3):

Лістинг 4.3 – Скорегований метод Can_Send_Pagination_View_Model()

```
[TestMethod]
public void Can_Send_Pagination_View_Model()
{
    // ...
    // Act
    StaffsListViewModel result
        = (StaffsListViewModel)controller.List(null, 2).Model;
// ... }
```

Звичка проводити тестування з забезпеченням синхронізації модульних тестів з внесенням змін в код дозволяє тестувати розробки динамічно та відслідковувати правильність роботи на різних етапах розробки.

Також необхідний модульний тест для перевірки функціональності фільтрації за категорією, щоб впевнитись в тому, що фільтр може коректно

генерувати свідчення про об’єкти вказаної категорії. Тестовий метод відповідає вигляду, наведеному в додатку В.

Цей тест створює імітоване сховище, що містить об’єкти Staff, що відносяться до певного діапазону категорій. З використанням методу дії List() запитується одна специфічна категорія, а результати перевіряються на предмет вмісту коректних об’єктів в правильному порядку.

Модульний тест, призначений для перевірки можливості генерування списку категорій, відносно простий. Мета полягає в створенні списку, що відсортований в алфавітному порядку та не містить дублікатів. Для цього найпростішим є побудова тестових даних, що мають дубльовані категорії та не відсортовані необхідним чином, передати їх в NavController та встановити твердження, що дані будуть відповідним чином очищені.

Вигляд тестового методу наведений в додатку В.

Всередині тесту створюється імітована реалізація сховища, що містить повторювані категорії та категорії, що не відсортовані в алфавітному порядку. Потім визначається твердження про те, що дублікати будуть видалені та алфавітний порядок буде відновлено.

22 Модульне тестування нотифікацій та лічильників фільтрів

Для виконання перевірки того, що метод дії Menu() правильно додав деталі про обрану категорію, в модульному тесті можна прочитати значення властивості ViewBag, що доступно через клас ViewResult. В додатку В показано даний тестовий метод.

Присутня можливість генерування коректних лічильників товарів для різних категорій – для цього в модульному тесті було створено імітоване сховище, що містить відомі дані в діапазоні категорій і потім викликається метод дії List(), що запитує кожну категорію по черзі [16]. Вигляд модульного тесту наведений в додатку В.

В модульному тесті також викликається метод List() без вказання категорії, щоб впевнитись в правильності підрахунку загальної кількості об’єктів.

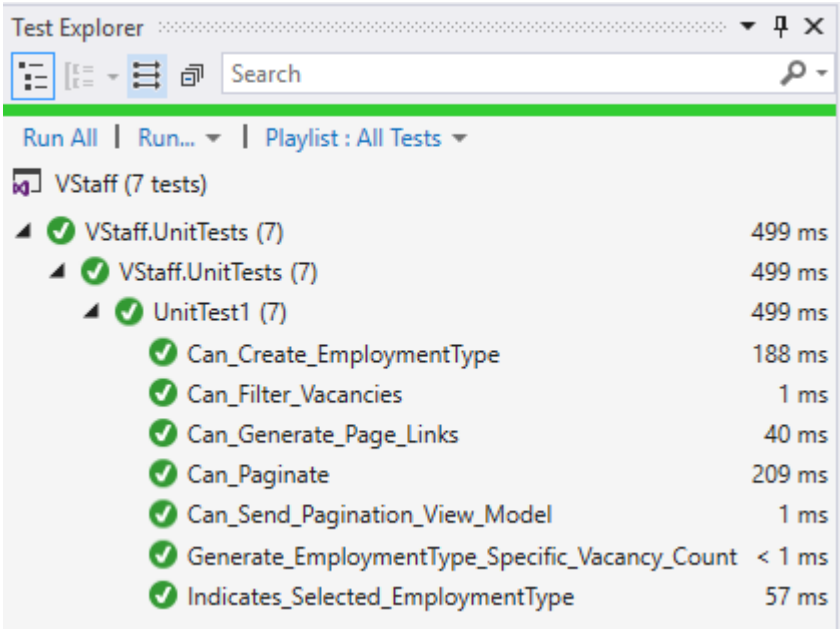


Рисунок 4.1 – Результати виконання модульних тестів з вказанням часу проходження тестів

Результати тестів свідчать про правильність компонування робочих компонентів та підтвердження факту виконання ними своїх обов’язків.

23 Висновок

Тестування роботи програмного додатку було виконано за допомогою модульного тестування та Unit-тестів зокрема для перевірки правильності роботи функціоналу та для впевнення того, що усі протестовані процеси повертають прийнятний результат. Подібне тестування дозволяє імітувати роботу окремих частин проекту та перевіряти їх як разом, так і ізольовано одне від одного.

РОЗДІЛ 5
ЕКОНОМІЧНА ЧАСТИНА

24 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Результатом магістерської кваліфікаційної роботи «Розробка методу та засобів обробки міжкорпоративних даних» є розробка програмного додатку для обліку та рекомендування кадрів та відкритих вакансій. Для проведення технологічного аудиту залучено трьох незалежних експертів. У даному випадку такими експертами є: керівник магістерської роботи – доц. каф. ПЗ Черноволик Галина Олександрівна, к.т.н., доц. каф. ПЗ – Ракитянська Ганна Борисівна, к.т.н., доц. каф. ПЗ – Майданюк Володимир Павлович.

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в додатку Б.

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Черноволик	2 – Ракитянська	3 – Майданюк
	Бали, виставлені експертами:		
1	4	3	4
Ринкові переваги (недоліки):			
2	3	2	2
3	4	4	4
4	4	3	4
5	3	4	3
Ринкові перспективи			
6	2	2	2
7	3	4	3
Практична здійсненність			

Продовження таблиці 5.1

8	4	3	4

9	3	2	3
10	4	4	4
11	3	4	3
12	4	4	4
Сума балів	$СБ_1 = 41$	$СБ_2 = 39$	$СБ_3 = 40$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{120}{3} = 40$		

За даними таблиці 5.1 можна зробити висновок щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.2.

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 40 балів, що відповідає рівню «вище середнього».

В першому розділі магістерської роботи проведено порівняння сучасних систем обліку та рекомендування кадрів.

В якості аналога для розробки було обрано систему E-Staff Рекрутер, так як вона, як і VStaff Advice, спрямована на підбір кадрів та автоматизацію рекрутингових процесів.

Основними недоліками аналога є те, що ця система володіє низькою якістю пошуку, не дозволяючи користувачу керувати режимом пошуку та працюючи в режимі повної відповідності. Також слід відмітити відсутність можливість виключати оброблених в попередніх запитах кандидатів з результатів нового пошуку.

У новому програмному засобі VStaff Advice подібні недоліки відсутні, а окрім того, доданий функціонал для розширення механізму автоматизації процесів рекрутингу, а саме – присутній механізм рекомендування кадрів та вакансій, що значно зменшує затрати часу на пошук кадрів.

У таблиці 5.3 наведені основні технічні показники аналога і нового програмного продукту [4].

Таблиця 5.3 – Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Функціональність	80%	70%	0.875
Сумісність	80%	80%	1
Супровід	70%	70%	1
Економія ресурсів і часу	60%	80%	1.3
Простота використання	40%	70%	1.75

Функціональність нової розробки виграє в зв’язку з тим, що у новій розробці присутні функції, яких немає у аналога (розширення механізму автоматизації процесів рекрутингу – рекомендування кадрів та вакансій), що забезпечує конкурентоздатність розробки. У зв’язку з впровадженням у систему нової функції, яка набагато зменшує витрати часу та зусиль на рекрутинг нових працівників, очікується підвищений попит, так як аналог, наведений вище, подібною функцією не володіє.

Рекрутер може зробити більше за допомогою однієї програми, що робить його кращим за аналог і у другому показнику – «економія ресурсів і часу».

Надійність двох розробок однакова, так як це веб-додаток і якщо у рекрутера немає доступу до Інтернету – він не може користуватись програмою, проте це означає, що обидва продукти досить легкі у використанні на різних платформах (показник «сумісність»), так як для використання даного продукту

не потрібно нічого завантажувати та встановлювати – достатньо мати на платформі браузер.

Здатність модифікувати програму і в новій розробці, і в аналога досить низька, тому що обидві програми написані за допомогою комп'ютерного коду, і якщо користувач хоче змінити тему або шрифт – в нього немає для цього можливості, тому що для цього потрібно редагувати код. Більшість користувачів аналога «E-Staff Рекрутер» скаржаться на те, що програмою досить важко користуватись через незрозумілий інтерфейс та завелику кількість полів. При розробці нової програми «VStaff Advice» ці недоліки були враховані і був розроблений простий інтерфейс для користування програмою.

Зважаючи на стабільно високу актуальність рекрутингових послуг в IT-сфері, можна зробити висновок, що розробка модифікованої системи обліку та рекомендування кадрів є досить перспективною.

Для користування системою не потрібно купувати ліцензію – достатньо зареєструватись у системі.

Головними користувачами та потенційними замовниками даного продукту є підприємства, які хочуть відслідковувати статус та інформацію про потенційних кандидатів на інших підприємствах, а також ділитись аналогічною інформацією у себе з іншими користувачами системи, тобто дана система буде реалізовуватись на IT-ринку України.

Розробка та впровадження нового програмного продукту автоматизує та спростить процес рекрутингу кандидатів на відкриті вакансії за рахунок зручної системи обліку та механізму рекомендування кадрів. Також нова розробка дозволить зменшити затрати часу та вартість самого процесу рекрутингу.

Просування розробленого програмного продукту відбуватиметься шляхом розміщення інформації про продукт на окремому веб-сайті, рекламних інтеграцій на сайтах пошуку роботи.

Окремим елементом просування програмного продукту є переговори з невеликими компаніями та підприємствами для впровадження розробки у їх механізмах рекрутингу та обліку кадрів. Це дозволить на старті сформувати

зведені дані з декількох підприємств та цим самим зацікавити більш рейтингові компанії.

Розробка повністю готова до реалізації. Є наявними фінансові та трудові ресурси [4], а також фахівці відповідної кваліфікації для впровадження на ринок нового програмного продукту.

В зв’язку з тим, що існує потенційно велика кількість зацікавлених осіб у новому програмному забезпеченні необхідно провести заходи щодо проведення переговорів з потенційними інвесторами для можливості подальшої комерціалізації розробки.

25 Прогнозування витрат на виконання наукової роботи та впровадження її результатів

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення [4], яке складається з таких етапів:

1. розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;
2. розрахунок загальних витрат на виконання даної роботи;
3. прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

1-й етап. Виконаємо розрахунок витрат приймаючи до уваги те, що для розробки інформаційної технології було залучено одного розробника програмного забезпечення.

Основна заробітна розробника-дослідника $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \qquad (5.1)$$

де М – місячний посадовий оклад = 8500 грн, $Т_р$ – число робочих днів в місяці, приблизно $Т_р$ = 22 дні, t – число робочих днів роботи розробника-дослідника (розробка програмного забезпечення триває 2 місяці) = 44.

$$З_о = \frac{8500}{22} \cdot 44 = 17000{,}00 \text{ (грн)}$$

Додаткова заробітна плата $З_д$ розробника розраховується як 10% від основної заробітної плати:

$$З_д = (0{,}1 \ldots 0{,}12) \cdot З_о \text{ [грн]}. \quad (5.2)$$

$$З_д = 0{,}1 \cdot 17000{,}00 = 1700{,}00 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{зп}$ розробника становить:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (5.3)$$

де $З_о$ – основна заробітна плата розробника, $З_д$ – додаткова заробітна плата розробника, β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 22%.

$$Н_{зп} = (17000{,}00 + 1700{,}00) \cdot 0{,}22 = 4114{,}00 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи, розраховується по кожному виду обладнання, приміщенням тощо.

Розмір амортизаційних відрахувань розраховується за формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ [грн]}, \quad (5.4)$$

де *Ц* – загальна балансова вартість обладнання, приміщення тощо, грн, $Н_а$ — річна норма амортизаційних відрахувань, %, Т – термін використання обладнання, приміщень тощо.

Розробка програмного забезпечення проводилася протягом 2 місяців.

Зроблені розрахунки зведено до таблиці 5.4.

Таблиця 5.4 – Амортизаційні відрахування

Найменування програмного забезпечення, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Приміщення	100000	20	2	3333
Комп’ютер	21000	25	2	875
Всього:				4208

Інформацію про матеріали, що використовуються при виготовленні даного продукту, вказано у таблиці 5.5.

Таблиця 5.5 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн
Папір (пачка)	100,00	1	100,00
Канцтовари	5,00	4	20,00
Всього			120,00

Під час створення програмного продукту використовувалось безкоштовне програмне забезпечення.

Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п [грн], \quad (5.5)$$

де В — вартість 1кВт-години електроенергії (В = 0,9 грн/кВт), П — установлена потужність комп'ютера (П = 0,3 кВт), Ф — фактична кількість годин роботи комп'ютера (Ф = 352 год), $К_п$ — коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,8$).

Потужність комп'ютера складає 230 Вт/год = 0,23 кВт/год + потужність на освітлення, тоді П = 0,3 кВт/год.

На проведення дослідження та розробку програмного забезпечення витрачено загалом 2 місяці, то з урахуванням кількості робочих днів та робочого часу - 8 годин, маємо 8·22·3 = 352 годин роботи комп'ютера.

$$В_е = 0,9 \cdot 0,3 \cdot 352 \cdot 0,5 = 48 \text{ (грн)}.$$

Також потрібно врахувати витрати на доступ до мережі Інтернет, що використовувався під час виконання роботи.

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$В_{ді} = Ц_{ді} \cdot Т [грн], \quad (5.6)$$

де $Ц_{ді}$ – це ціна доступу за місяць, Т – кількість місяців використання доступу до мережі.

Отже, витрати на доступ до мережі Інтернет становлять:

$$В_{ді} = 120 \cdot 2 = 240 \text{ (грн)}.$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot З_о \, [грн]. \quad (5.7)$$

$$В_{ін} = 2 \cdot 17000{,}00 = 34000{,}00 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даного етапу роботи — В:

$$В = З^{о}+З^{д}+Н^{зп}+А + В^{ді} + В^{е} + В^{ін} \text{ [грн]}, \quad (5.8)$$

$$В = 17000{,}00 + 1700{,}00 + 4114{,}00 + 4208{,}00 + 240{,}00 + 48{,}00 + 34000 = 61310{,}00 \text{ (грн)}.$$

2-й етап. Загальна вартість всієї наукової роботи $В_{заг}$ визначається за формулою:

$$В_{заг} = \frac{В}{\alpha} \, [грн], \quad (5.9)$$

де α — частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях (так, як над роботою задіяна одна людина, якою виконується уся робота, то α становить 1).

Підставивши дані у формулу, отримуємо:

$$В_{заг} = \frac{61310}{1} = 61310 \text{ грн.}$$

3-й етап. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи проводиться за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} [грн], \quad (5.10)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи(так, як розробка знаходиться на стадії впровадження, то $\beta \approx 0,9$), $В_{заг}$ – загальна вартість всієї наукової роботи (оскільки наукова робота завершена, $В_{заг}$ = 61310,00 грн.).

$$ЗВ = \frac{61310,00}{0,9} = 68122,22 \text{ (грн).}$$

Отже, прогноз загальних витрат на виконання та впровадження результатів становить 68122,22 грн.

26 Прогнозування комерційних ефектів від реалізації результатів розробки

Магістерська робота містить економічну частину обґрунтування економічної доцільності розробки програмного засобу для обліку та рекомендування кадрів. Для того, щоб виконати дану розробку потрібно 44 робочі дні. Дана розробка вважається економічно вигідною, якщо її окупність становитиме 1 рік.

Зростання чистого продукту для даного методу можна оцінити у теперішній вартості грошей. Зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Має сенс спроба кількісного прогнозування вигоди, яку можна отримати у майбутньому від впровадження результатів виконаної наукової роботи [4]. При цьому слід врахувати приблизність виконуваних розрахунків, що не передбачають деталізації.

Збільшення чистого прибутку підприємства $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_і = \sum_і^n (\Delta П_о \cdot N + Ц_о \cdot \Delta N) \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) [грн], \quad (5.11)$$

де $\Delta П_о$ – покращення основного оціночного показника від впровадження результатів розробки у даному році (зазвичай таким показником виступає ціна одиниці нової розробки); N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки; ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки; $Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки; n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; λ – коефіцієнт, який враховує сплату податку на додану вартість; ρ – коефіцієнт, який враховує рентабельність продукту (рекомендується приймати $\rho = 0{,}25$); υ – ставка податку на прибуток (18%).

Реалізація продукції до впровадження результатів наукової розробки не здійснювалась, тому приймаємо 1. Ціна – 2500 грн.

В результаті впровадження результатів наукової розробки покращується якість продукту, що дозволяє підвищити ціну його реалізації на 1000 грн.

Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 100 шт., протягом другого року – ще на 150 шт., протягом третього року – ще на 200 шт.

Вищевказані дані дозволяють спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом першого року складе:

$$\Delta П_1 = [2500 \cdot 1 + 3500 \cdot 100] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 63591{,}00 \text{ (грн).}$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = [500 \cdot 1 + 650 \cdot (100 + 150))] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 158301{,}00 \text{ грн.}$$

Аналогічним чином здійснюється розрахунок чистого прибутку протягом третього року:

$$\Delta П_3 = [500 \cdot 1 + 600 \cdot (100 + 150 + 200))] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 284581{,}00 \text{ грн.}$$

27 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає:

1) Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку (прогнозована величина загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV);

2) Розрахунок очікуваного збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження;

3) Побудова вісі часу з нанесенням всіх платежів (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі вказуються у ті терміни, коли вони здійснюються.

Має місце припущення, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнює 68122,22 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятись протягом трьох років. У першому році підприємство отримає збільшення чистого прибутку на 63591 грн відносно базового року, у другому році – збільшення чистого прибутку на 158301 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 284581 грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде представлений у вигляді, наведеному на рисунку 5.1.

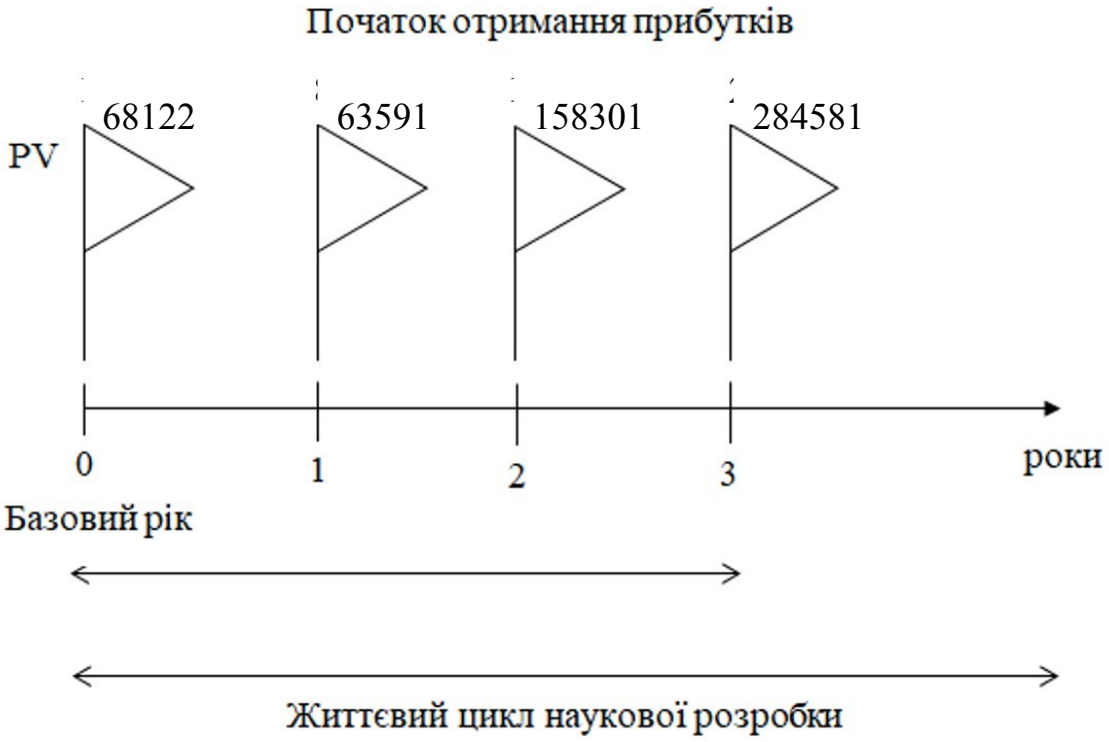


Рисунок 5.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4) Розрахунок абсолютної ефективності $E_{абс}$ вкладених інвестицій за формулою:

$$E_{абс} = (ПП - PV)\ [грн], \quad (5.12)$$

де ПП — приведена вартість чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.; PV — теперішня вартість інвестицій PV=ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1+\tau)^t} [грн], \quad (5.13)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн; т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні (для України цей показник знаходиться на рівні 0,1); t – період часу (в роках) від моменту отримання чистого прибутку до точки «0».

Якщо $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту.

Тоді розрахунок абсолютної ефективності інвестицій, вкладених у реалізацію проекту, виглядатиме наступним чином:

$$ПП = \frac{63591{,}00}{(1+0{,}1)^1} + \frac{158301{,}00}{(1+0{,}1)^2} + \frac{284581{,}00}{(1+0{,}1)^3} = 402447{,}19\ (грн).$$

З огляду на отриманий результат можна визначити абсолютну ефективність вкладених інвестицій $Е_{абс}$:

$$Е_{абс} = 402447,19 – 68122,22 = 334324,97$$

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки програмного продукту та їх впровадження принесе прибуток, тобто є доцільним.

5) Розрахунок відносної (щорічної) ефективності вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.14)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV – теперішня вартість інвестицій PV = ЗВ, грн; $Т_ж$ – життєвий цикл наукової розробки, роки.

Використавши наявні дані, отримаємо:

$$E_в = \sqrt[3]{1 + \frac{334324,97}{68122,22}} - 1 = \sqrt[3]{5,91} - 1 = 0,81 \text{ або } 81\%$$

Наступним кроком буде порівняння розрахованої величини $Е_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$ [4], яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \quad (5.15)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках (d = 0,2); f — показник, що характеризує ризикованість вкладень.

Якщо величина $E_в > \tau$ мін, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде. Спочатку спрогнозуємо величину τ мін. Припустимо, що за даних умов τ мін = 0,2 + 0,35 = 0,55.

Оскільки $E_в$ =81% > $\tau_{мін}$ = 55%, тоді припустимий висновок, що інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

6) Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [грн]} \qquad (5.16)$$

Для розгляданої розробки термін окупності вкладених у реалізацію проекту інвестицій $T_{ок}$ складатиме:

$$T_{ОК} = \frac{1}{0,81} = 1,23 \text{ р.}$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок}$< 3 років), то фінансування розгляданої розробки є доцільним.

28 Висновки

В даному розділі було виконано оцінювання комерційного потенціалу розробки програмного додатку для обліку та рекомендування кадрів «VStaff Advice».

Проведено технологічний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки вище

середнього. Дослідження комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою.

Існуючі переваги нової розробки дозволять швидко її поширити та популяризувати.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 68122,22 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 334324,97 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 81 %, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 55%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 1,23 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

В магістерській кваліфікаційній роботі було створено додаток для обліку кадрів та відкритих вакансій на основі платформи ASP .NET. Вважається, що даний проект повинен допомогти людям у пошуку бажаної роботи для одних і якісних працівників – для інших, маючи при цьому можливість формувати списки об’єктів за однотипними параметрами. Крім того було створено зручний інтерфейс з інтуїтивно зрозумілими навігаційним меню.

В результаті виконання магістерської кваліфікаційної роботи було досягнуто усіх поставлених цілей:

— проаналізовано сучасний стан ринку додатків для обліку кадрів;

— проаналізовано сучасні вимоги та структуру подібних додатків;

— вдосконалено існуючий метод дискретного зважування у сфері рекрутингу;

— вдосконалено існуючий метод матричного порівняння у сфері сортування даних;

— проаналізовано та розроблено моделі сутностей та безпосередньо сховище даних;

— розроблено програмну реалізацію на основі сформованих вимог та запропонованого методу;

— протестовано функціонал та можливості кінцевого проекту за допомогою засобів модульного тестування.

Це дозволило не лише розробити швидкий та водночас об’ємний ресурс, покращити навички програмування, ознайомитись з розробкою веб-каталогів, а і створити конкурентоспроможний ресурс на ринку веб-розробки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Спивак В.А. Управление персоналом для менеджеров: учебное пособие. Москва, 2012. 790 с.

2. Оценка качества разбиений алгоритмов на подалгоритмы с использованием весовой функции. Материалы межрегиональной научно-технической конференции «Интеллектуальные и информационные системы» (Интеллект-2005). Тула, 2005. 30 с.

3. Матвієнко М.П. Теорія алгоритмів. Навчальний посібник. Київ, 2017. 340 с.

4. Козловський В. О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ, 2012. 22 с.

5. Джозеф Албахари, Бен Албахари. C# в двух словах. Вильямс, 2016. 1040 с.

6. Джеффри Рихтер. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. СПб., 2013. 895 с.

7. Ендрю Троелсен. Мова програмування C# 5.0 и платформа .NET 4.5. СПб., 2013. 592 с.

8. Михайлов Ю. М. Охрана труда в офисе. Москва, 2010. 256 с.

9. Ткачук К. Н., Халімовський М. О. Основи охорони праці. Київ, 2006. 446 с.

10. Оптимизация скриптов. URL: http://professorweb.ru/my/ASP_NET/mvc/level1/ (дата звернення 28.10.2019).

11. Бучек Г. ASP.NET. Учебный курс. СПб., 2002. 512с.

12. Петлюшкин А. В. HTML в Web-дизайне. СПб., 2004. 400 с.

13. Офіційний сайт Майданюка Володимира Павловича. URL: http://maydanyuk.vk.vntu.edu.ua/pub (дата звернення 30.10.2019).

14. Реферат на тему: «Засоби створення Web-додатків». URL: http://ifreestore.net/2267/ (дата звернення 15.10.2019).

15. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C#. СПб., 1999. 372 с.

16. Майерс Г. Искусство тестирования программ. Москва, 1982. 172 с.

17. Иванова С. Искусство подбора персонала. Москва, 2004. 160 с.

18. Беленко П. Headhunting: принципы и технологии. Питер, 2006. 192с.

19. Мансуров Р. HR-брендинг. Как повысить эффективность персонала. Питер, 2011. 224 с.

20. Cormen T., Leiserson C., Rivest R. Introduction to Algorithms. Montana, 1990. 1048 с.

21. Левитин А. Алгоритмы: введение в разработку и анализ. Вильямс, 2006. 576 с.

22. Макконнелл Дж. Основы современных алгоритмов. Москва, 2004, 368 с.

23. Скиена С. Алгоритмы. Руководство по разработке. Питер, 2013. 720с.

24. Романюк Оксана Володимирівна – Публікації. URL: http://oromanuk.vk.vntu.edu.ua/pub (дата звернення 25.11.2019).

25. Електронні інформаційні ресурси: створення, використання, доступ: Збірник матеріалів Міжнародної науково-практичної Інтернет конференції. Пам’яті А. М. Петуха. Вінниця : ВНТУ, 2019. – 292 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

_____ Романюк О. Н.

«____» _____2019 року

Технічне завдання

на магістерську кваліфікаційну роботу

«Розробка методу та засобів обробки міжкорпоративних даних»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. Г. О. Черноволик

"____"_____2019 р.

Виконав:

_____студент гр. 2ПІ-18м О. А. Тяпкін

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та засобів обробки міжкорпоративних даних».

Галузь застосування – рекрутингові процеси.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №___ ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності рекрутингових механізмів за рахунок градаційного компонування сумісних вакансій та доступних агентів.

Призначення роботи – розробка методу та засобів обробки міжкорпоративних даних.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Спивак В.А. Управление персоналом для менеджеров: учебное пособие Москва, 2012. 790 с.

2. Оценка качества разбиений алгоритмов на подалгоритмы с использованием весовой функции. Материалы межрегиональной научно-технической конференции «Интеллектуальные и информационные системы» (Интеллект-2005). Тула, 2005. 30 с.

3. Матвієнко М.П. Теорія алгоритмів. Навчальний посібник. Київ, 2017. 340 с.

4. Джеффри Рихтер Программирование на платформе Microsoft .NET Framework 4.5 на языке C#, 2013. 895 с.

4. Технічні вимоги

Операційна система – Windows 10; середовище розробки – Microsoft Visual Studio; мови програмування – C#, SQL.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз предметної галузі	07.10.2018 – 12.10.2019
2	Аналіз методів та засобів розробки бази даних	13.10.2018 – 16.10.2019
3	Розробка методу прогнозованої оцінки кандидата для конкретної вакансії	17.10.2018 – 05.11.2019
4	Розробка бази даних обліку вакансій і кадрів	06.11.2019 – 16.11.2019
5	Розробка програмного додатку обліку та рекомендування вакансій та кадрів	17.11.2019 – 25.11.2019
6	Тестування роботи програмного додатку	26.12.2019 – 30.12.2019
7	Економічна частина	01.12.2019 – 08.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці

9	Потрібні значні	Потрібні	Потрібні значні	Потрібні	Не потребує

	фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	незначні фінансові ресурси. Джерела фінансування відсутні	фінансові ресурси. Джерела фінансування є	незначні фінансові ресурси. Джерела фінансування є	додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Додаток В. Лістинг програми

Лістинг В.1 – Код файлу Domain/Abstract/IVacancyRepository.cs

```
using System.Collections.Generic;
using VStaff.Domain.Entities;
namespace VStaff.Domain.Abstract {
    public interface IVacancyRepository {
        IEnumerable<Area> Areas { get; }
        IEnumerable<Company> Companies { get; }
        IEnumerable<CompanyBranch> CompetentionAnswers {get;}
        IEnumerable<EmploymentType> EmploymentTypes { get; }
        IEnumerable<Experience> Experiences { get; }
        IEnumerable<LocalityType> LocalityTypes { get; }
        IEnumerable<Region> Regions { get; }
        IEnumerable<Skill> Skills { get; }
        IEnumerable<Staff> Staffs { get; }
        IEnumerable<StaffSkill> StaffsAnswers { get; }
        IEnumerable<Status> StafsSkills { get; }
        IEnumerable<Vacancy> Vacancies { get; }
        IEnumerable<Vacancieskill> VacanciesAnswers { get; }
        IEnumerable<Vacanciestaff> VacanciesSkills { get; }
    }
}
```

Лістинг В.2 – Код файлу Domain/Concrete/EFVacancyRepository .cs

```
using System.Collections.Generic;
using VStaff.Domain.Abstract;
using VStaff.Domain.Entities;
namespace VStaff.Domain.Concrete {
    public class EFVacancyRepository : IVacancyRepository {
        EFDbContext context = new EFDbContext();
        public IEnumerable<Area> Areas {
            get { return context.Areas; }
        }
        public IEnumerable<Company> Companies {
            get { return context.Companies; }
        }
                    public IEnumerable<CompetitionAnswer>
CompetitionAnswers {
            get { return context. CompetitionAnswers; }
        }
        public IEnumerable<EmploymentType> EmploymentTypes {
            get { return context.EmploymentTypes; }
        }
        public IEnumerable<Experience> Experiences {
            get { return context.Experiences; }
        }
        public IEnumerable<LocalityType> LocalityTypes {
```

```
                get { return context.LocalityTypes; }
            }
            public IEnumerable<Region> Regions {
                get { return context.Regions; }
            }
            public IEnumerable<Skill> Skills {
                get { return context.Skills; }
            }
            public IEnumerable<Staff> Staffs {
                get { return context.Staffs; }
            }
            public IEnumerable<StaffAnswer> StaffsAnswers {
                get { return context. StaffsAnswers; }
            }
            public IEnumerable< StaffSkill > StaffsSkills {
                get { return context. StaffsSkills; }
            }
            public IEnumerable<Vacancy> Vacancies {
                get { return context.Vacancies; }
            }
            public IEnumerable<VacancyAnswer> VacanciesAnswers {
                get { return context. VacanciesAnswers; }
            }
            public IEnumerable< VacancieSkill > VacanciesSkills {
                get { return context. VacanciesSkills; }
            }
        }
    }
```

Лістинг В.3 – Код файлу Domain/Concrete/EFDbContext.cs

```
using System.Data.Entity;
using VStaff.Domain.Entities;
namespace VStaff.Domain.Concrete {
    public class EFDbContext : DbContext {
        public DbSet<Area> Areas { get; set; }
        public DbSet<Company> Companies { get; set; }
          public DbSet< CompetitionAnswer > CompetitionAnswers
{get; set;}
             public DbSet<EmploymentType> EmploymentTypes {get;
set;}
        public DbSet<Experience> Experiences { get; set; }
        public DbSet<LocalityType> LocalityTypes { get; set; }
        public DbSet<Region> Regions { get; set; }
        public DbSet<Skill> Skills { get; set; }
        public DbSet<Staff> Staffs { get; set; }
        public DbSet<StaffAnswer> StaffsAnswers { get; set; }
        public DbSet< StaffSkill > StaffsSkills { get; set; }
        public DbSet<Vacancy> Vacancies { get; set; }
          public DbSet<VacancyAnswer> VacanciesAnswers { get;
set; }
```

```
            public DbSet< VacancySkill > VacanciesSkills { get;
set; }
        protected override void OnModelCreating(DbModelBuilder
modelBuilder) {
                    modelBuilder.Entity<Staff>().HasMany(c =>
c.Answers)
                .WithMany(s => s.Staffs)
                .Map(t => t.MapLeftKey("StaffId")
                .MapRightKey("AnswerId")
                .ToTable("StaffsAnswers"));
                    modelBuilder.Entity<Staff>().HasMany(c =>
c.Skills)
                .WithMany(s => s.Staffs)
                .Map(t => t.MapLeftKey("StaffId")
                .MapRightKey("SkillId")
                .ToTable("StaffsSkills"));
                  modelBuilder.Entity<Vacancy>().HasMany(c =>
c.Skills)
                .WithMany(s => s.Vacancies)
                .Map(t => t.MapLeftKey("VacancyId")
                .MapRightKey("SkillId")
                .ToTable("VacanciesSkills"));
                  modelBuilder.Entity<Vacancy>().HasMany(c =>
c.Answers)
                .WithMany(s => s.Vacancies)
                .Map(t => t.MapLeftKey("VacancyId")
                .MapRightKey("AnswerId")
                .ToTable("VacanciesAnswers"));
        }
    }
}
```

Лістинг В.4 – Код файлу Domain/Entities/Area.cs

```
using System.Collections.Generic;
namespace VStaff.Domain.Entities {
    public class Area {
        public int AreaId { get; set; }
        public string Name { get; set; }
        public List<Vacancy> Vacancies { get; set; }
        public List<Staff> Staffs { get; set; }
        public int RegionId { get; set; }
        public virtual Region Region { get; set; }}}
```

Лістинг В.6 – Код файлу Domain/Entities/Company.cs

```
using System.Collections.Generic;
namespace VStaff.Domain.Entities {
    public class Company {
        public int CompanyId { get; set; }
        public string Name { get; set; }
        public string Status { get; set; }
```

```
        public string Director { get; set; }
        public string Branch { get; set; }
        public string Description { get; set; }
        public List<Vacancy> Vacancies { get; set; }
        public List<Staff> Staffs { get; set; }
        public Company() {
            Branches = new List<Branch>();
        }
    }
}
```

Лістинг В.7 – Код файлу Domain/Entities/CompanyBranch.cs

```
namespace VStaff.Domain.Entities {
    public class CompanyBranch {
        public int CompanyBranchId { get; set; }
        public int CompanyId { get; set; }
        public virtual Company Company { get; set; }
        public int BranchId { get; set; }
        public virtual Branch Branch { get; set; }
    }
}
```

Лістинг В.8 – Код файлу Domain/Entities/Director.cs

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
namespace VStaff.Domain.Entities {
    public class Director {
        [ForeignKey("CompanyId")]
        public int DirectorId { get; set; }
        public Company CompanyId { get; set; }
        public string Surname { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public DateTime? BirthDate { get; set; }
        public string Phone { get; set; }
    }
}
```

Лістинг В.9 – Код файлу Domain/Entities/EmploymentType.cs

```
using System.Collections.Generic;
namespace VStaff.Domain.Entities {
    public class EmploymentType {
        public int EmploymentTypeId { get; set; }
        public string Name { get; set; }
        public List<Vacancy> Vacancies { get; set; }
    }
}
```

Лістинг В.10 – Код файлу Domain/Entities/Experience.cs

```
using System;
namespace VStaff.Domain.Entities {
    public class Experience {
        public int ExperienceId { get; set; }
        public int? StaffId { get; set; }
        public virtual Staff Staff { get; set; }
        public int? CompanyId { get; set; }
        public virtual Company Company { get; set; }
        public string CurrentPosition { get; set; }
        public DateTime? Begin { get; set; }
        public DateTime? End { get; set; }
    }
}
```

Лістинг В.11 – Код файлу Domain/Entities/LocalityType.cs

```
using System.Collections.Generic;

namespace VStaff.Domain.Entities {
    public class LocalityType {
        public int LocalityTypeId { get; set; }
        public string Name { get; set; }
        public List<Vacancy> Vacancies { get; set; }
        public List<Staff> Staffs { get; set; }
    }
}
```

Лістинг В.12 – Код файлу Domain/Entities/Region.cs

```
using System.Collections.Generic;
namespace VStaff.Domain.Entities {
    public class Region {
        public int RegionId { get; set; }
        public string Name { get; set; }
        public List<Area> Areas { get; set; }
    }
}
```

Лістинг В.13 – Код файлу Domain/Entities/Skill.cs

```
using System.Collections.Generic;
namespace VStaff.Domain.Entities {
    public class Skill {
        public int SkillId { get; set; }
        public string Name { get; set; }
        public virtual ICollection<Staff> Staffs { get; set; }
          public virtual ICollection<Vacancy> Vacancies {get;
set;}
```

```
            public Skill() {
                Staffs = new List<Staff>();
                Vacancies = new List<Vacancy>();
            }
        }
    }
```

Лістинг В.14 – Код файлу WebUI/App_Start/NinjectWebCommon.cs

```
    [assembly:
WebActivator.PreApplicationStartMethod(typeof(VStaff.WebUI.App_Sta
rt.NinjectWebCommon), "Start")]
    [assembly:
WebActivator.ApplicationShutdownMethodAttribute(typeof(VStaff.WebU
I.App_Start.NinjectWebCommon), "Stop")]
    namespace VStaff.WebUI.App_Start {
        using System;
        using System.Web;
        using Microsoft.Web.Infrastructure.DynamicModuleHelper;
        using Ninject;
        using Ninject.Web.Common;
        public static class NinjectWebCommon {
              private static readonly Bootstrapper bootstrapper =
new Bootstrapper();
            public static void Start() {
    DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpMo
dule));
DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
                bootstrapper.Initialize(CreateKernel);
            }
            public static void Stop() {
                bootstrapper.ShutDown();
            }
            private static IKernel CreateKernel() {
                var kernel = new StandardKernel();
                kernel.Bind<Func<IKernel>>().ToMethod(ctx => () =>
new                                    Bootstrapper().Kernel);
kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpMod
ule>();
                RegisterServices(kernel);
                return kernel;
            }
            private static void RegisterServices(IKernel kernel) {
                 System.Web.Mvc.DependencyResolver.SetResolver(new
VStaff.WebUI.Infrastructure.NinjectDependencyResolver(kernel));
            }
        }
    }
```

Лістинг В.15 – Код файлу WebUI/App_Start/RouteConfig.cs

```
using System.Web.Mvc;
using System.Web.Routing;
namespace VStaff.WebUI {
    public class RouteConfig {
     public static void RegisterRoutes(RouteCollection routes)
{
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(null,
                "",
                new {
                    controller = "Vacancy",
                    action = "List",
                    region = (string)null,
                    area = (string)null,
                    locality Type = (string)null,
                    page = 1
                }
            );
            routes.MapRoute(
                name: null,
                url: "Page{page}",
               defaults: new { controller = "Vacancy", action =
"List",
                    region = (string)null,
                    area = (string)null,
                    locality Type = (string)null
                },
                constraints: new { page = @"\d+" }
            );
            routes.MapRoute(null, "{region}",
        new { controller = "Vacancy", action = "List", page = 1
}
            );
            routes.MapRoute(null, "{area}",
        new { controller = "Vacancy", action = "List", page = 1
}
            );
            routes.MapRoute(null, "{locality Type}",
        new { controller = "Vacancy", action = "List", page = 1
}
            );
            routes.MapRoute(null,
                "{region}/{area}/{locality Type}/Page{page}",
                      new { controller = "Vacancy", action =
"List" },
                new { page = @"\d+" }
            );
            routes.MapRoute(null, "{controller}/{action}");
         }
      }}
```

Лістинг В.16 – Код файлу WebUI/Controllers/VacancyController .cs

```
using System.Linq;
using System.Web.Mvc;
using VStaff.Domain.Abstract;
using VStaff.WebUI.Models;
namespace VStaff.WebUI.Controllers {
   public class VacancyController : Controller {
      private IVacancyRepository repository;
      public int pageSize = 10;
      public VacancyController(IVacancyRepository repo) {
          repository = repo;
      }
      public ViewResult List(string region, string area,
         string localityType, string branch,
         int page = 1) {
         VacancyListViewModel model = new VacancyListViewModel
{
                Vacancies = repository.Vacancies
               .Where(p=>region==null || p.Area.Region.Name==
region)
             .Where(p=>branch==null || p.Branches.Where(s =>
s.Name == branch) == branch)
           .Where(p=>localityType==null || p.LocalityType.Name
== localityType)
           .Where(p=>area== null || p.Area.Name == area)
                .OrderBy(Vacancy => Vacancy.VacancyId)
                .Skip((page - 1) * pageSize)
                .Take(pageSize),
                PagingInfo = new PagingInfo {
                    CurrentPage = page,
                    ItemsPerPage = pageSize,
                           TotalItems = ((region == null) &&
(localityType == null)) ?
                repository.Vacancies.Count() :
                          repository.Vacancies.Where(Vacancy=>
Vacancy.Area.Region.Name == region)
                  .Where(Vacancy => Vacancy.Area.Name == area)
.Where(Vacancy=>Vacancy.LocalityType.Name==localityType).Count()
                },
                CurrentRegion = region,
                CurrentArea = area,
                CurrentLocalityType = localityType
            };
            return View(model);
        }
    }
}
```

Лістинг В.17 – Код файлу WebUI/Controllers/NavController.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using VStaff.Domain.Abstract;
namespace VStaff.WebUI.Controllers {
   public class NavController : Controller {
       private IVacancyRepository repository;
       public NavController(IVacancyRepository repo) {
            repository = repo;
       }
       public PartialViewResult Menu(string region = null) {
            ViewBag.SelectedRegion = region;
            IEnumerable<string> regions = repository.Vacancies
                .Select(Vacancy => Vacancy.Area.Region.Name)
                .Distinct()
                .OrderBy(x => x);
            return PartialView(regions);
       }
          public PartialViewResult Menu2(string area = null,
string region = null) {
           ViewBag.SelectedArea = area;
           ViewBag.SelectedRegion = region;
           IEnumerable<string> areas = repository.Vacancies
               .Where(Vacancy=>Vacancy.Area.Region.Name==regio
n)
               .Select(Vacancy => Vacancy.Area.Name)
               .Distinct()
               .OrderBy(x => x);
            return PartialView(areas);
        }
         public PartialViewResult Menu3(string localityType =
null) {
          ViewBag.SelectedLocalityType = localityType;
                          IEnumerable<string>  localityTypes  =
repository.Vacancies
                .Select(Vacancy => Vacancy.LocalityType.Name)
                .Distinct()
                .OrderBy(x => x);
            return PartialView(localityTypes);
        }
    }
}
```

Лістинг В.18 – Код файлу WebUI/HtmlHelpers/PagingHelpers.cs

```
using System;
using System.Text;
using System.Web.Mvc;
using VStaff.WebUI.Models;
namespace VStaff.WebUI.HtmlHelpers {
    public static class PagingHelpers {
```

```
            public static MvcHtmlString PageLinks(this HtmlHelper
html, PagingInfo pagingInfo, Func<int, string> pageUrl) {
                StringBuilder result = new StringBuilder();
                for (int i = 1; i <= pagingInfo.TotalPages; i++) {
                    TagBuilder tag = new TagBuilder("a");
                    tag.MergeAttribute("href", pageUrl(i));
                    tag.InnerHtml = i.ToString();
                    if (i == pagingInfo.CurrentPage) {
                        tag.AddCssClass("selected");
                        tag.AddCssClass("btn-primary");
                    }
                    tag.AddCssClass("btn btn-default");
                    result.Append(tag.ToString());
                }
                return MvcHtmlString.Create(result.ToString());
            }
        }
    }
```

Лістинг В.19 – Код файлу WebUI/Infrastructure/ NinjectDependencyResolver.cs

```
    using System;
    using System.Collections.Generic;
    using System.Web.Mvc;
    using Ninject;
    using VStaff.Domain.Abstract;
    using VStaff.Domain.Concrete;
    namespace VStaff.WebUI.Infrastructure {
      public class NinjectDependencyResolver : IDependencyResolver
{
            private IKernel kernel;
             public NinjectDependencyResolver(IKernel kernelParam)
{
                kernel = kernelParam;
                AddBindings();
            }
            public object GetService(Type serviceType) {
                return kernel.TryGet(serviceType);
            }
          public IEnumerable<object> GetServices(Type serviceType)
{
                return kernel.GetAll(serviceType);
            }
            private void AddBindings() {
      kernel.Bind<IVacancyRepository>().To<EFVacancyRepository>();
            }
        }
    }
```

Лістинг В.20 – Код файлу WebUI/Models/VacancyListViewModel.cs

```
using System.Collections.Generic;
using VStaff.Domain.Entities;
namespace VStaff.WebUI.Models {
    public class VacancyListViewModel {
        public IEnumerable<Vacancy> Vacancies { get; set; }
        public PagingInfo PagingInfo { get; set; }
        public string CurrentRegion { get; set; }
        public string CurrentArea { get; set; }
        public string CurrentLocalityType { get; set; }
        public string CurrentBranch { get; set; }
    }
}
```

Лістинг В.21 – Код файлу WebUI/Models/PagingInfo.cs

```
using System;
namespace VStaff.WebUI.Models {
    public class PagingInfo {
        public int TotalItems { get; set; }
        public int ItemsPerPage { get; set; }
        public int CurrentPage { get; set; }
        public int TotalPages {
            get { return (int)Math.Ceiling((decimal)TotalItems
/ ItemsPerPage); }
        }
    }
}
```

Лістинг В.22 – Код файлу WebUI/Views/Vacancy/List.cshtml

```
@using VStaff.WebUI.Models
@using VStaff.WebUI.HtmlHelpers;
@model VacancyListViewModel
@{ ViewBag.Title = "Вакансіяі";
}
    @foreach (var p in @Model.Vacancies) {
        @Html.Partial("Vacanciesummary", p)
    }
    <div class="btn-group pull-right">
                            @Html.PageLinks(Model.PagingInfo,
x=>Url.Action("List",
        new { page = x,
            region = Model.CurrentRegion,
            area = Model.CurrentArea,
            localityType = Model.CurrentLocalityType}))
    </div>
```

Лістинг В.23 – Код файлу WebUI/Views/Nav/Menu.cshtml

```
@model IEnumerable<string>
<div class="dropdown">
```

```
          <button class="btn btn-primary dropdown-toggle btn-lg"
type="button" data-toggle="dropdown" style="width: 250px; margin-
bottom: 10px">
              Оберіть область
              <span class="caret"></span>
          </button>
          <ul class="dropdown-menu">
              <li>@Html.ActionLink("Усі", "List", "Vacancy", null,
          new { @class = "" })</li>
              @foreach (var link in Model) {
                  <li>@Html.RouteLink(link, new
         {
             controller = "Vacancy",
             action = "List",
             region = link,
             page = 1
         }, new {
             @class = ""
          + (link == ViewBag.SelectedRegion ? " btn-primary" : "")
         })
     </li>
              }
          </ul>
     </div>
```

Лістинг В.24 – Код файлу WebUI/Views/Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
         <meta  name="viewport"  content="width=device-width,
initial-scale=1.0">
    <link href="~/Content/bootstrap.css" rel="stylesheet" />
                  <link    href="~/Content/bootstrap.min.css"
rel="stylesheet" />
    <script src="~/scripts/jquery-1.10.2.js"></script>
    <script src="~/scripts/main.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/
1.12.2/jquery.min.js"></script>
                                                          <script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.m
in.js"></script>
    <title>@ViewBag.Title</title>
</head>
<body>
    <div class="navbar navbar-inverse" role="navigation">
         <a class="navbar-brand" href="#">Електронний каталог
обліку кадрів та вакансій</a>
    </div>
    <div class="row panel">
        <div id="categories" class="col-xs-3">
```

```
                @Html.Action("Menu", "Nav")
                @Html.Action("Menu2", "Nav")
                @Html.Action("Menu3", "Nav")
            </div>
                <div class="col-xs-8">
                    @RenderBody()
                </div>
            </div>
    </body>
    </html>
```

Лістинг В.25 – Код файлу WebUI/Views/Shared/CollegSummary .cshtml

```
    @model VStaff.Domain.Entities.Vacancy
    <div class="well">
             <table  cellpadding="0"  cellspacing="0"  border="0"
width="100%" class="found">
        <tbody><tr class="sf desc f_th">
                <td width="1%"></td>
                  <td width="81%" style="padding-left:5px">заклад
/ опис</td>
                   <td style="padding-left:2px;padding-right:3px">
населений пункт</td>
            </tr>
            <tr>
                <td class="nf f_pad"></td>
                <td>
                                  <a  href="@Model.Site"  name="143"
class="h4_link"> <strong>@Model.Name</strong></a>
                    <div class="sf desc">
                        Рівень акредитації @Model.AccrLevel.
                    </div>
                </td>
                       <td  class="nf  f_pad"  nowrap=""> <strong>
@Model .Locality</strong></td>
            </tr>
            <tr class="f_bor">
                <td></td>
                <td colspan="2" class="">
                    <p><b>Адреса:</b> @Model.Address</p>
         <p style="margin-top:-10px;">
                        <b class="">Телефон:</b>
                                                                <span
class="">@Model.Phone</span></p><p></p>
    </td>
    </tr>
    </tbody>
    </table>
    </div>
```

Лістинг В.26 – Код файлу UnitTests/UnitTest1.cshtml

```
using System;
using System.Collections.Generic;
using System.Web.Mvc;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using VStaff.Domain.Abstract;
using VStaff.Domain.Entities;
using VStaff.WebUI.Controllers;
using VStaff.WebUI.Models;
using VStaff.WebUI.HtmlHelpers;
namespace VStaff.UnitTests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Can_Paginate()
        {
                        Mock<IVacancyRepository> mock = new
Mock<IVacancyRepository>();
                        mock.Setup(m=>m.Vacancies).Returns(new
List<Vacancy>
            {
                        new Vacancy { VacancyId = 1, Name =
"Вакансія1"},
                        new Vacancy { VacancyId = 2, Name =
"Вакансія2"},
                        new Vacancy { VacancyId = 3, Name =
"Вакансія3"},
                        new Vacancy { VacancyId = 4, Name =
"Вакансія4"},
                        new Vacancy { VacancyId = 5, Name =
"Вакансія5"}
            });
                         VacancyController controller = new
VacancyController(mock.Object);
            controller.pageSize = 3;
                              VacancyListViewModel result =
(VacancyListViewModel)controller.List(null, 2).Model;
            List<Vacancy> games = result.Vacancies.ToList();
            Assert.IsTrue(games.Count == 2);
            Assert.AreEqual(games[0].Name, "Вакансія4");
            Assert.AreEqual(games[1].Name, "Вакансія5");
        }
        [TestMethod]
        public void Can_Generate_Page_Links()
        {
            HtmlHelper myHelper = null;
            PagingInfo pagingInfo = new PagingInfo
            {
                CurrentPage = 2,
```

```
                    TotalItems = 28,
                    ItemsPerPage = 10
                };
                Func<int, string> pageUrlDelegate = i => "Page" +
i;
                                        MvcHtmlString   result   =
myHelper.PageLinks(pagingInfo, pageUrlDelegate);
```

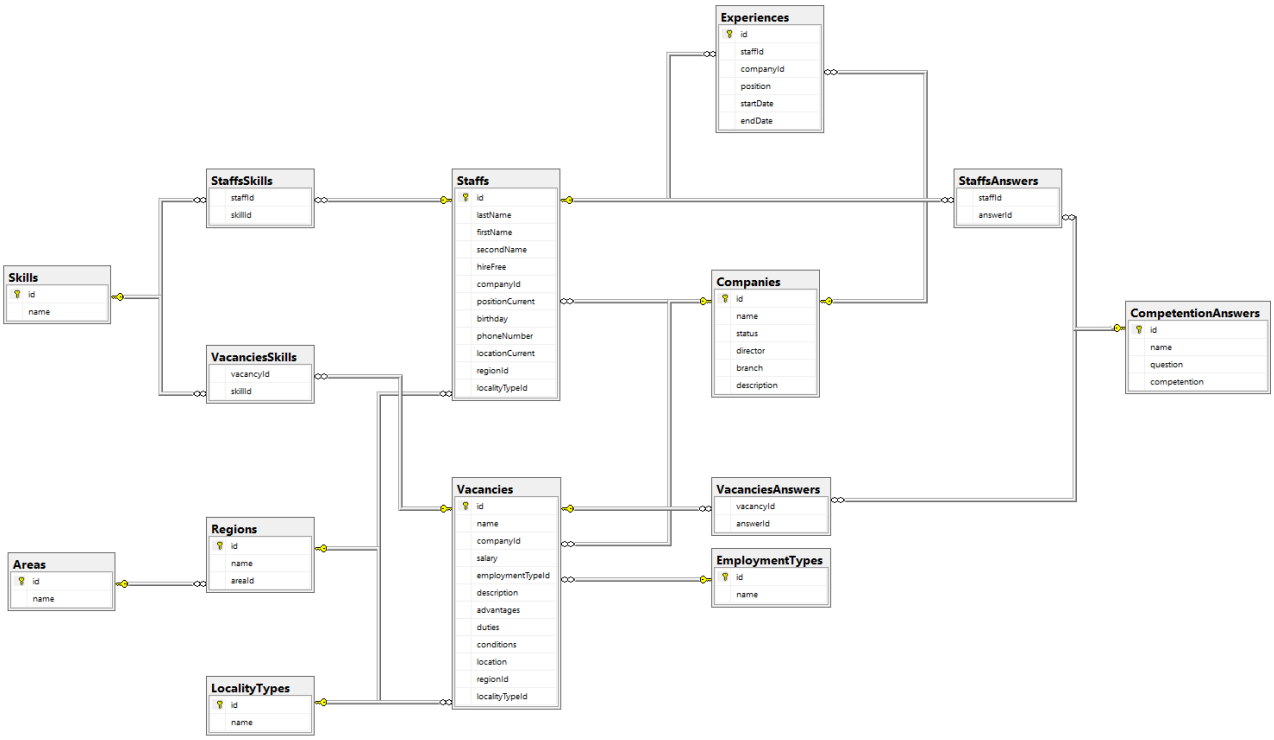
Продовження лістингу

```
                Assert.AreEqual(@"<a class=""btn btn-default""
href=""Page1"">1</a>"
                    + @"<a class=""btn btn-default btn-primary
selected"" href=""Page2"">2</a>"
                        +  @"<a  class=""btn  btn-default""
href=""Page3"">3</a>",
                    result.ToString());
            }
            [TestMethod]
            public void Can_Send_Pagination_View_Model()
            {
                        Mock<IVacancyRepository>  mock  =  new
Mock<IVacancyRepository>();
                    mock.Setup(m  =>  m.Vacancies).Returns(new
List<Vacancy>
        {
            new Vacancy { VacancyId = 1, Name = "Вакансія1"},
            new Vacancy { VacancyId = 2, Name = "Вакансія2"},
            new Vacancy { VacancyId = 3, Name = "Вакансія3"},
            new Vacancy { VacancyId = 4, Name = "Вакансія4"},
            new Vacancy { VacancyId = 5, Name = "Вакансія5"}
        });
                        VacancyController  controller  =  new
VacancyController(mock.Object);
                controller.pageSize = 3;

                VacancyListViewModel result
                    = (VacancyListViewModel)controller.List(null,
2).Model;
                PagingInfo pageInfo = result.PagingInfo;
                Assert.AreEqual(pageInfo.CurrentPage, 2);
                Assert.AreEqual(pageInfo.ItemsPerPage, 3);
                Assert.AreEqual(pageInfo.TotalItems, 5);
                Assert.AreEqual(pageInfo.TotalPages, 2);}
                [TestMethod]
                public void Can_Filter_Vacancies()
                {
                        Mock<IVacancyRepository> mock = new
Mock<IVacancyRepository>();
                    mock.Setup(m => m.Vacancies).Returns(new
List<Vacancy>
            {
                new Vacancy { VacancyId = 1, Name = "Вакансія1",
Area.Name="Area1"},
```

```
                new Vacancy { VacancyId = 2, Name = "Вакансія2",
Area="Area2"},

    new Vacancy { VacancyId = 3, Name = "Вакансія3",
Area="Area1"},
                new Vacancy { VacancyId = 4, Name = "Вакансія4",
Area="Area2"},
                new Vacancy { VacancyId = 5, Name = "Вакансія5",
Area="Area3"}
            });
                                VacancyController controller = new
VacancyController(mock.Object);
                    controller.pageSize = 3;
                                        List<Vacancy> result =
((VacancyListViewModel)controller.List("Area2", 1).Model)
                        .Vacancies.ToList();
                    Assert.AreEqual(result.Count(), 2);
                    Assert.IsTrue(result[0].Name == "Вакансія2" &&
result[0].Area.Name == "Area2");
                    Assert.IsTrue(result[1].Name == "Вакансія4" &&
result[1].Area.Name == "Area2");
                }
                [TestMethod]
                public void Can_Create_Areas(){
                                Mock<IVacancyRepository> mock = new
Mock<IVacancyRepository>();
                            mock.Setup(m => m.Vacancies).Returns(new
List<Vacancy> {
                new Vacancy { VacancyId = 1, Name = "Вакансія1",
Area.Name="Вінницький"},
                new Vacancy { VacancyId = 2, Name = "Вакансія2",
Area.Name="Вінницький"},
                new Vacancy { VacancyId = 3, Name = "Вакансія3",
Area.Name="Барський"},
                new Vacancy { VacancyId = 4, Name = "Вакансія4",
Area.Name="Гайсинський"},        });
                                    NavController target = new
NavController(mock.Object);
                                        List<string> results =
((IEnumerable<string>)target.Menu().Model).ToList();
                    Assert.AreEqual(results.Count(), 3);
                    Assert.AreEqual(results[0], "Барський");
                    Assert.AreEqual(results[1], "Вінницький");
                    Assert.AreEqual(results[2], "Гайсинський");
                }
                [TestMethod]
                public void Indicates_Selected_Area()
                {
                                Mock<IVacancyRepository> mock = new
Mock<IVacancyRepository>();
                            mock.Setup(m => m.Vacancies).Returns(new
Vacancy[] {
```

```
                new Vacancy { VacancyId = 1, Name = "Вакансія1",
Area.Name="Вінницький"},
                new Vacancy { VacancyId = 2, Name = "Вакансія2",
Area.Name="Барський"}
            });
                                        NavController target = new
NavController(mock.Object);
                    string areaToSelect = "Барський";
                                                string result =
target.Menu(areaToSelect).ViewBag.SelectedArea;
                    Assert.AreEqual(areaToSelect, result);
                }
            [TestMethod]
            public void Generate_Area_Specific_Vacancy_Count(){
                            Mock<IVacancyRepository> mock = new
Mock<IVacancyRepository>();
                        mock.Setup(m => m.Vacancies).Returns(new
List<Vacancy>
        {
                new Vacancy { VacancyId = 1, Name = "Вакансія1",
Area.Name="Area1"},
                new Vacancy { VacancyId = 2, Name = "Вакансія2",
Area.Name="Area2"},
                new Vacancy { VacancyId = 3, Name = "Вакансія3",
Area.Name="Area1"},
                new Vacancy { VacancyId = 4, Name = "Вакансія4",
Area.Name="Area2"},
                new Vacancy { VacancyId = 5, Name = "Вакансія5",
Area.Name="Area3"}
        });
                            VacancyController controller = new
VacancyController(mock.Object); controller.pageSize = 3;
                                                int res1 =
((VacancyListViewModel)controller.List("Area1").Model).PagingInfo.
TotalItems;
                                                    int res2 =
((VacancyListViewModel)controller.List("Area2").Model).PagingInfo.
TotalItems;
                                                    int res3 =
((VacancyListViewModel)controller.List("Area3").Model).PagingInfo.
TotalItems;
                                                  int resAll =
((VacancyListViewModel)controller.List(null).Model).PagingInfo.Tot
alItems;
                Assert.AreEqual(res1, 2);
                Assert.AreEqual(res2, 2);
                Assert.AreEqual(res3, 1);
                Assert.AreEqual(resAll, 5);}}}
```

Додаток Г. Схема даних



Додаток Д. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор_____О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____Г. О. Черноволик

Рецензент, к. т. н., доцент кафедри КН_____І. Р. Арсенюк

Нормоконтроль, к. т. н., доцент кафедри ПЗ_____Г. О. Черноволик

Виконавець, студент групи 2ПІ-18м_____О. А. Тяпкін

АКТУАЛЬНІСТЬ

Актуальність даної роботи обумовлена тим, що спеціалісти з підбору кадрів знаходяться в постійному пошуку та оновленні інструментів, що допомагали б їм в процесі підбору та найму працівників, а також для систематизації існуючих кадрів.

Слайд 1 – Актуальність магістерської кваліфікаційної роботи

Мета, об’єкт та предмет дослідження

- **Мета дослідження** – підвищення продуктивності рекрутингових механізмів за рахунок градаційного компонування сумісних вакансій та доступних агентів.
- **Об’єкт дослідження** – процес обробки міжкорпоративних даних із застосуванням технології розробки сховища даних обліку кадрів.
- **Предмет дослідження** – методи та засоби сортування та групування даних.

Слайд 2 – Мета, об’єкт та предмет дослідження

ОСНОВНІ ЗАДАЧІ

- проаналізувати сучасний стан ринку додатків для обліку кадрів;
- проаналізувати сучасні вимоги та структуру подібних додатків;
- удосконалити існуючий метод дискретного зважування у сфері рекрутингу;
- удосконалити існуючий метод матричного порівняння у сфері сортування даних;
- проаналізувати та розробити моделі сутностей та безпосередньо сховище даних;
- розробити програмну реалізацію на основі сформованих вимог та запропонованого методу;
- протестувати програмну реалізацію за допомогою засобів модульного тестування.

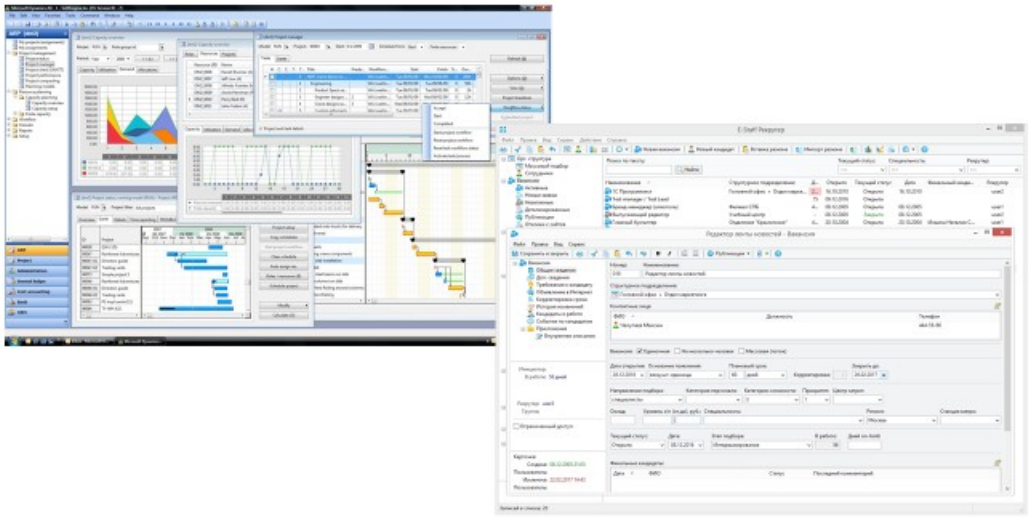
Слайд 3 – Основні задачі дослідження

НАУКОВА НОВИЗНА

- Отримав подальшого розвитку метод дискретного зважування у сфері рекрутингу, який полягає в попередній оцінці кандидата за набором компетентних критеріїв, що дозволяє зазначати без попереднього інтерв'ю прогнозовані результати аналізу особистості кандидата;
- Отримав подальшого розвитку метод матричного порівняння у сфері сортування, який полягає у застосуванні порівняльних методик для специфічних типів записів, що дозволяє оптимізувати процес відбору кандидатів для інтерв'ю.

Слайд 4 – Наукова новизна дослідження, що проводиться

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ



Слайд 5 – Порівняльний аналіз аналогів

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

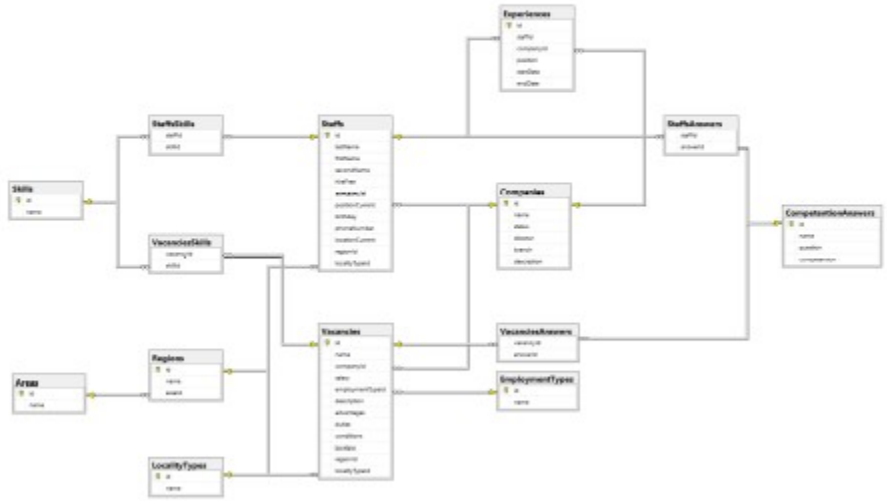
Критерії	Microsoft Dynamics AX	E-Staff Рекрутер	Дипломна робота «VStaff Advice»
Доступні дані про задіяних працівників	+	-	+
Підбір кандидатів на вакансію	-	+	+
Присутня можливість впровадження в інші розробки	-	-	+
Сумісність з популярними операційними системами	+	-	+
Рекомендування вакансій для кандидатів	-	-	+
Не перевантажений функціоналом інтерфейс	-	-	+
Можливість роботи з базами декількох підприємств	-	-	+

Слайд 6 – Таблиця результатів порівняння аналогів з роботою

Слайд 7 – Перевага приведення до градаційної шкали

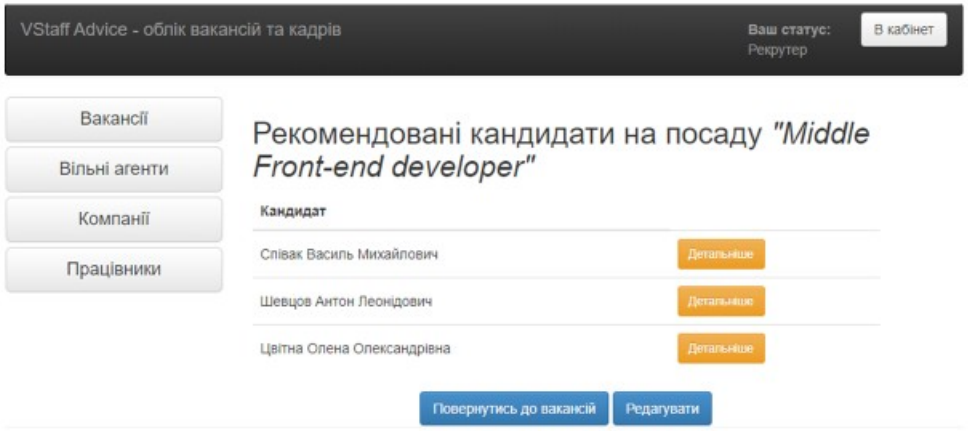
Слайд 8 – Алгоритм роботи методу для визначення сумісності

СТРУКТУРА БАЗИ ДАНИХ



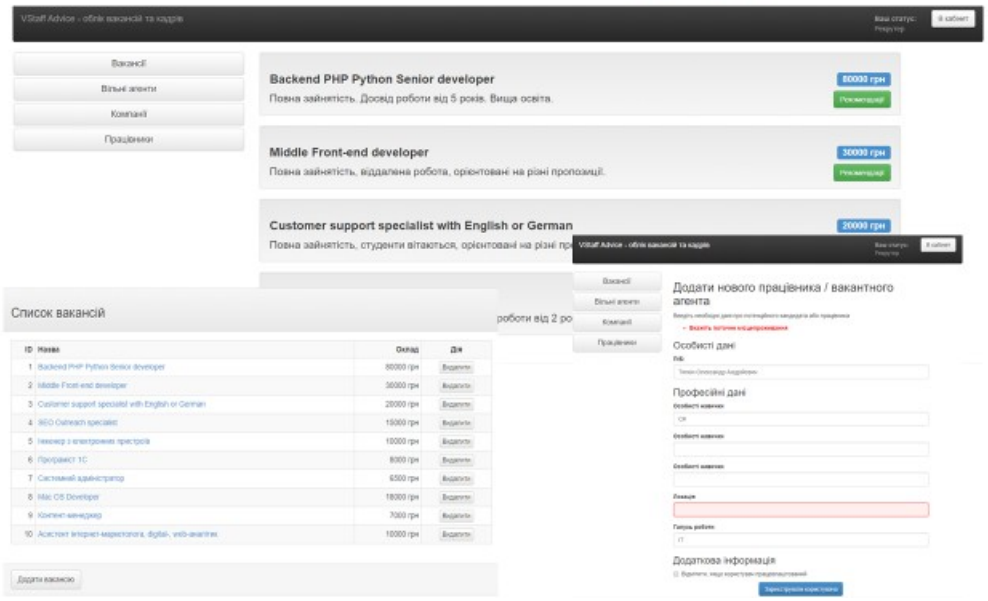
Слайд 9 – Структура бази даних

РЕЗУЛЬТАТ РОБОТИ АЛГОРИТМУ



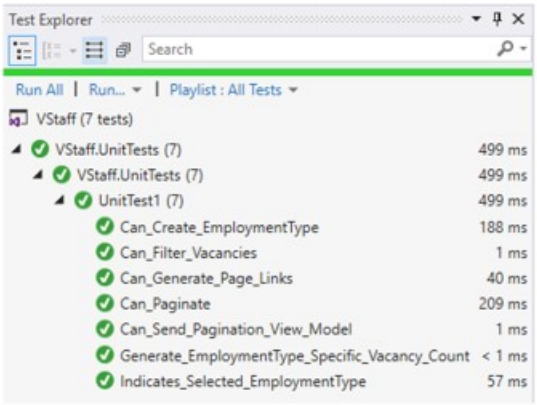
Слайд 10 – Результат роботи алгоритму

ГОЛОВНЕ ВІКНО ДОДАТКУ

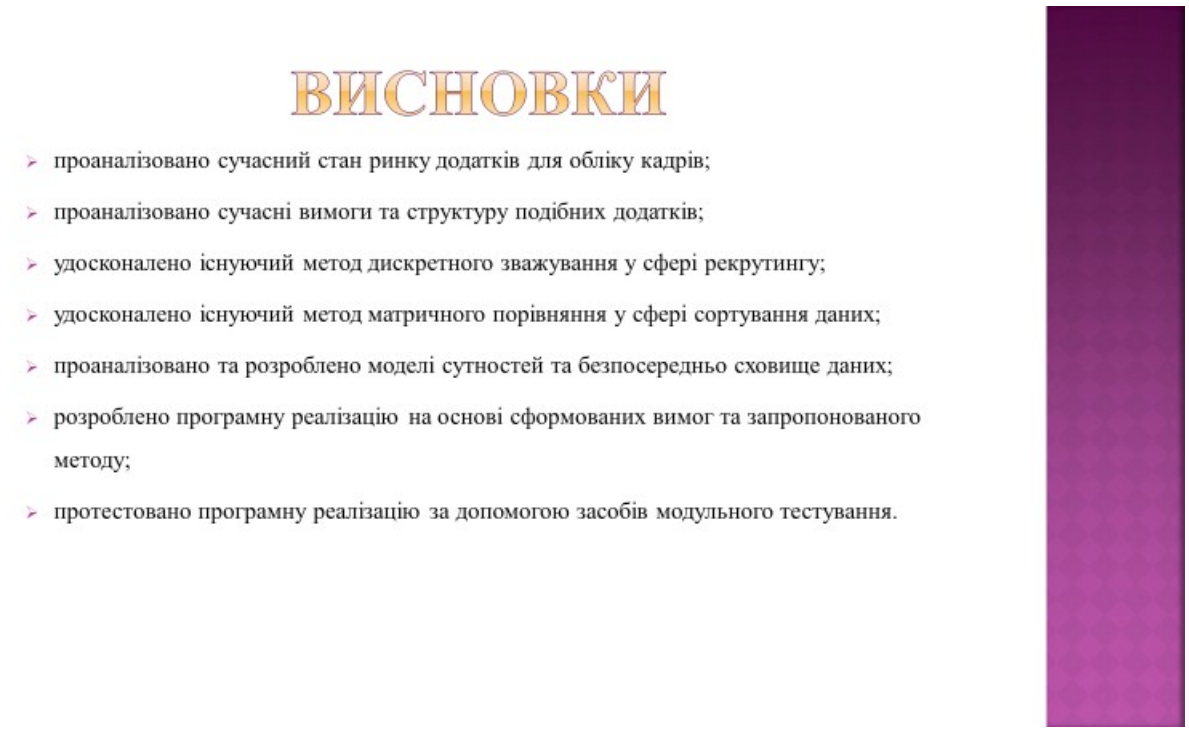


Слайд 11 – Основні вікна додатку

РЕЗУЛЬТАТИ ТЕСТУВАННЯ



Слайд 12 – Результати тестування



Слайд 13 - Висновки