НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6. 050201 «Системна інженерія»

на тему: «Веб-застосунок для пошуку розробників програмного забезпечення на основі ASP.NET»

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-51
Поліщук Мирослав Ігорович _____

Керівник:
Професор кафедри АУТС, д.т.н., доцент Корнієнко Б.Я. _____

Рецензент:
Доцент кафедри ОТ, к.т.н., доцент Павлов В.Г.
_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Веб-застосунок для пошуку розробників програмного забезпечення на основі ASP.NET»

Київ – 2019 рік

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASP.NET – Active Server Pages для .NET платформа розробки веб-додатків

W3C – World Wide Web Consortium головна міжнародна організація, що розробляє й впроваджує технологічні стандарти.

MySQL – вільна реляційна система управління базами даних.

EF – ADO.NET Entity Framework об'єктно-орієнтована технологія доступу до даних, є object-relational mapping (ORM)

ПЗ – програмне забезпечення.

JS – JavaScript зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків.

C# - об'єктно-орієнтована мова програмування

CSS – Cascading Style формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки.

HTML – HyperText Markup Language «мова гіпертекстової розмітки»

Bootstrap – Twitter Bootstrap вільний набір інструментів для створення сайтів і веб-додатків.

NuGet - система управління пакетами для платформ розробки Microsof.

Web UI - Створення клієнтської (Front-end) части крос-браузерні Web-застосунків.

Front-end - клієнтська сторона призначеного для користувача інтерфейсу до програмно-апаратної частини сервісу.

Back-end - программно-аппаратная часть сервиса.

ВСТУП

В наш час для пошуку розробників програмного забезпечення використовую змішані ресурси по підбору майбутніх працівників, професійні соціальні мережі, звичайні соціальні мережі або ж форуми.

Створений мною продукт дає можливість знайти кваліфікованих розробників в сфері IT технологій. На даний момент для пошуку розробників програмного забезпечення існує багато різних джерел , але кожен з даних ресурсів має свої недоліки. Проаналізувавши ринок праці я вирішив створити свій продукт, який буде позбавлений основних проблем, які мають інші продукти.

Якщо розглянути проблематику даних ресурсів то можна виявити багато недоліків в цих продуктах, деякі з них просто непристосовані або погано адаптовані для пошуку, інші ж занадто складні для розуміння і тому доводиться звертатися за допомогою до спеціалістів по підбору працівників. А інші вже застаріли і взаємодія з ними уже не задовільняє потенційного користувача.

Тому я вважаю що даний продукт стане конкурентом іншим ресурсам і займе високу нішу в даній сфері. Для цього було проаналізовано різні можливі середовища, щоб обрати саме те що буде повністю задовільняти потреби проекту.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Актуальність проекту

Мій проект буде повністю орієнтуватися на ринок і буде задовольняти потреби ринку. Щоб компанії або окремі люди змогли швидко знайти потрібних IT спеціалістів. Для цього буде розроблений зручний інтуїтивний інтерфейс для взаємодії з клієнтом. Зроблений пошук по категоріям. Буде підтримка кросбраузерності, а також адаптиву .

1.2 Загальні положення

Мій продукт буде слідувати класичному підходу, який активно використовується в даний час. Я створю каталог вакансій, який користувачі можуть переглядати за категоріями і сторінками.

Крім того я створю адміністративну панель за допомогою якої можна буде керувати вмістом і при потрібності редагувати. Мій проект буде створюватися на Visual Studio з трьома проектами. Один підпроект буде містити в собі модель предметної області налаштовується на забезпечення сталості за допомогою сховища , яке створено за допомогою інфраструктури Entity Framework , другий підпроект зроблений за шаблоном MVC який містить контролери та подання і виступає в якості призначеного для користувача інтерфейсу, а третій буде використовуватися для модульних тестів.

Призначення розробки - використання проекту для пошуку розробників програмного забезпечення який буде актуальним і задовільнятиме на даний момент всі потреби користувача.

1.3 Змістовний опис і аналіз предметної області

Веб-проект повинен бути розроблений на технології ASP.NET.Як сервер бази даних буде використовуватися Microsoft SQL Server, а допуск до бази даних буде здійснюватися за допомогою Entity Framework інфраструктури

ORM для платформи .NET. Потрібно буде створити адміністративну та клієнтську частини сайту. Для адміністративної частини потрібно створити окремий інтерфейс та аутентифікацію, щоб тільки потенційний адміністратор міг авторизуватися і мати право на редагування, видалення і додавання нових вакансій.

1.4 Можливості проекта

Проект повинен задовільняти певні критерії які на даний момент актуальні і використовуються в схожих проектах.Також проект розроблений для подальшого розвитку і підтримки, для цього я використував певний шаблон, щоб в подальшому удосконаленні не виникало проблем у розробників.

1.4.1 Зручність панелі навігації

Меню і пункти розроблені таким чином щоб користувачу було інтуїтивно зрозуміло як користуватися сайтом. За допомогою CSS,а також фремворку Bootstrap зроблено лаконічний і красивий дизайн, щоб зацікавити користувача.А за допомогою фремворку зроблений адаптивний дизайн для мобільної версії.

1.4.2 Каталог вакансій

Каталог вакансій також відноситься до навігації і в ньому мають знаходитися вакансії для пошуку веб-розробників. Також за допомогою CSS покращили візуальний інтерфейс. А також розбили на окремі сторінки вакансії для зручності користувача.

1.4.3 Додавання веб-розробників

Передбачено що потенційний клієнт зможе додати кілька вакансій які його зацікавили. Та реалізовано можливість перегляду доданих вакансій та взаємодію з ними .

1.4.4 Сумісність браузерів

На даний момент майже кожен користується інтернетом і це призвело до створення великої кількості різних браузерів. З появою різних браузерів виникли проблеми сумісності проекту і відображенням його у різних браузерів. Через те що не існує єдиного стандарту. Через це було створено єдиний стандарт W3C. В рамках цього протоколу прийшли до рішення про створення єдиного стандарту набору інтрументів для браузерів. По статистиці OpenStat найбільша популярність:

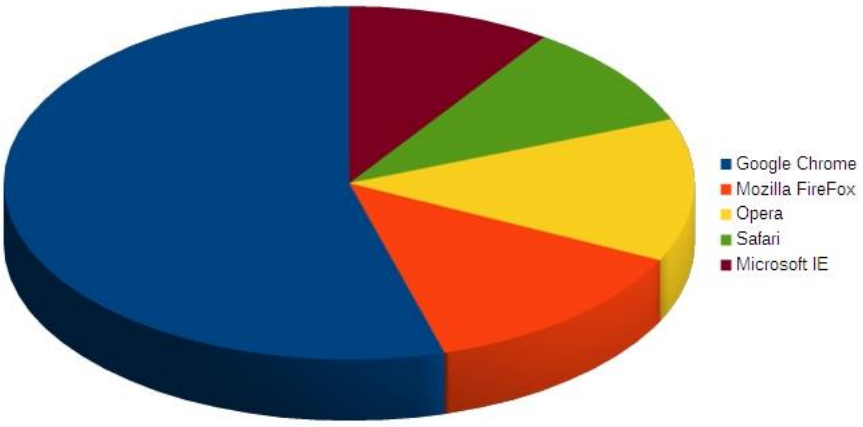

Рисунок 2.1 — Статистика використання браузерів[1]

Мій проект буде орієнтуватися на останні версії найпопулярніших браузерів.Як мобільних версій так і версій для ПК.

1.5 Аналіз успішних IT-проектів

Серед подібних продуктів на ринку найбільш серйозні конкуренти це DOU а також LinkedIn

DOU - це найбільше співтовариство розробників. Даний ресурс надає можливість порівняти для різних спеціальностей заробітну плату яка на час є актуальною, також ви можете вказати ваш досвід роботи і система це враховує,до цього ви можете вказати ваше місто і вашу спеціальність.

Ви можете переглянути як змінилася ціна за різні періоди часу. Є спеціальна функція динаміка зарплат по певним мовам програмування, де можна вибрати регіони України і подивитися якого рівня спеціалісти отримували заробітну плату. Також цей сервіс надає можливість моніторингу зарплат по різним містам по різним мовам програмування і за певний період.

Linkedin-це сервіс точного пошуку програмістів і інших IT-спеціалістів надає дані розробників за виконаними проектами, замовленнями і професійним навичкам, підтверджені грошима замовників, які оплатили роботи, а також єдині рейтинг, набір даних і пошукові фільтри для профілів з різних джерел. Вони інтегрують і регулярно оновлюють профілі розробників найбільших фріланс-бірж (upwork | odesk, elance, freelancer.com і ін.):

– регулярно обробляючи понад 100 тисяч профілів, вони відбирають для вас тільки підтверджені реальними проектами профілі з реальними сумами;

– очищають і коригують дані від похибок (різноманітності написаний одних і тих самих навичок або міст і т.д.) і різних хитрощів, що можуть завищити рейтинг розробника (таких як завищення годин або сум);

– забезпечують потрібні пошукові фільтри і повноцінне уявлення необхідних для точного вибору HR даних;

– надають додаткові пошукові можливості по GitHub і Linkedin і засоби обліку для процесу обробки профілів;

– простий функціонал статусів профілів в стилі TODO дозволяє вам ефективно класифікувати і обробляти вибрані профілі;

– додатково сервіс дає можливість моніторингу рівня оплат і активності замовників по країнах;

– єдиний рейтинг для кожного, на основі аналізу виконаних розробником проектів, дозволяє зіставити для точного вибору профілі з різних джерел.

Важливий результат пошуку програмістів - незалежна, достовірна оцінка кандидатів на базі виконаних проектів, а не тільки резюме. Для цього є рейтинг сервісу, що розраховується на основі підтверджених даних по проектам з

різних джерел. Він дозволяє точно вибирати схожих по необхідному рівню розробників, дані про яких розкидані по різних джерелах.

Висновки по розділу

У цьому розділі було описано та проаналізовано предметну область розробки, описано загальні положення для проекту. Було виділено успішні IT-проекти у даній області та порівняння їй з моїм веб-додатком.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Технологія ASP.NET

ASP.NET MVC - це структура веб-розробки від Microsoft, яка поєднує в собі ефективність і впорядкованість архітектура моделі-перегляду-контролера (MVC), найсучасніші ідеї та методи з гнучкої розробки, і найкращі частини існуючої платформи ASP.NET. Це повна альтернатива традиційним веб-формам ASP.NET, надаючи переваги для всіх, крім найбільш тривіальних проектів веб-розробки.

За допомогою веб-форм корпорація Майкрософт спробувала приховати як HTTP так і HTML шляхом моделювання інтерфейсу користувача (UI) як ієрархії керування на сервері об'єктів. Кожен елемент керування відстежує своє власне стан за запитами (за допомогою засобу View State), що відображає себе як HTML при необхідності автоматично підключаючи події на стороні клієнта (наприклад, натискання кнопки) з відповідним кодом обробника подій на сервері. Фактично, Web Forms є гігантським шаром абстракції, розробленим для забезпечення класичної події графічний інтерфейс користувача (GUI) через Інтернет. Ідея полягала в тому, щоб зробити веб-розробку таким самим, як розробка Windows Forms. Розробники цього не зробили, тому що потрібно працювати з низкою незалежних HTTP-запитів і відповідей. Для створення продукту була обрана платформа ASP.NET. для відповіді. Технологія ASP.NET є розвитком Active Server Page (ASP). Дана технологія являє собою універсальну платформу для розробки веб-додатків корпоративного рівня. ASP.NET пропонує нову модель програмування і інфраструктуру, які дозволяють розробляти захищені і масштабовані рішення [2]. Рішення реалізовано за допомогою паттерна MVC 5 [3]. концепція паттерна (Шаблону) MVC (model - view - controller) передбачає поділ програми на три компонента:

Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує ведені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді представлення.

Представлення (view) - це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, html-сторінка, яку користувач бачить, зайшовши на сайт.

Модель (model) представляє клас, що описує логіку використовуваних даних. При такому підході модель є незалежним компонентом - будь-які зміни контролера або представлення не зачіпають модель. контролер і уявлення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного. Завдяки цьому реалізується концепція поділ відповідальності, в зв'язку з чим легше побудувати роботу над окремими компонентами.

Крім того,внаслідок цього додаток стає легший у тестувані. І також, важлива візуальна частина або фронтенд, то тестують уявлення незалежно від контролера. Або ж зосереджуються на тестування бекенда і тестування контролер. Конкретні реалізації та визначення даного патерну можуть відрізнятися, але в силу своєї гнучкості і простоти він став дуже популярним останнім часом, особливо в сфері веб-розробки.

Свою реалізацію паттерна представляє платформа ASP.NET MVC. 2015 рік ознаменувався виходом нової версії ASP.NET MVC - MVC 6 [4], а також релізом Visual Studio 2017, яка надає інструментарій для роботи з MVC6.

ASP.NET має наступні функціональні можливості:

– Простота розгортання. Розгортання ASP.NET додатків виконується шляхом копіювання файлів програми в спеціальну папку на веб сервері. Перезапуск web-сервера не потрібно;

– Кошти безпеки. Розробник ASP.NET може використовувати в своєму додатку будь-яку із запропонованих типових схем авторизації і аутентифікації користувачів;

– Підтримка національних мов. ASP.NET використовує Unicode і розробники мають можливість застосовувати в своїх проектах національні алфавіти;

– Висока продуктивність. ASP.NET має справу зі скомпільований кодом. Завдяки цьому ASP.NET отримує можливість ефективно використовувати різні механізми оптимізації коду;

– Підтримка мобільних пристроїв. ASP.NET підтримується будь-яким браузером, запущеним на будь-якому пристрої (заява Microsoft);

– Можливості налагодження. ASP.NET забезпечує можливість трасування і налагодження коду додатків;

– Інтеграція з .NET Framework. ASP.NET є частиною платформи .NET Framework. Розробники можуть використовувати можливості, що надаються цією платформою при створенні додатків [5];

ASP.NET містить безліч готових елементів управління, застосовуючи які, можна швидко створювати інтерактивні web-додатки. В загальному, можливості ASP.NET обмежені тільки нашою уявою.

2.2 Мова реалізації C#

В якості мови реалізації був обраний C # на увазі наступних його переваг:

– C # є об'єктно-орієнтованою мовою;

– Мова C # розроблявся паралельно з каркасом Framework .Net і в повною мірою враховує всі його можливості;

– C # є спадкоємцем мов C / C ++. Ці мови мають загальний синтаксис, що полегшує перехід від C ++ до C #;

– Потужна бібліотека каркаса підтримує зручність побудови різних типів додатків на C #, дозволяючи достатньо просто зберігати і отримувати інформацію з бази даних і інших сховищ даних.

– Простота і надійність, головним чином, пов'язані з тим, що на C # хочаі допускаються, але не заохочуються такі небезпечні властивості C ++ як адресна арифметика, адресація, розіменування, і покажчики.

2.3 Microsoft SQL Server

Microsoft SQL Server в якості мови запитів використовує версію SQL,

що отримала назву Transact-SQL [6] (скорочено T-SQL), що є реалізацією SQL-92 (стандарт ISO для SQL) з множинними розширеннями. TSQL дозволяє використовувати додатковий синтаксис для збережених процедур і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим додатком). Microsoft SQL Server і Sybase ASE для взаємодії з мережею використовують протокол рівня додатка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних). Протокол TDS також був реалізований в проект FreeTDS з метою забезпечити різним додаткам можливість взаємодії з базами даних Microsoft SQL Server і Sybase.

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. Версія SQL Server 2017 забезпечує можливість підключення користувачів через веб-сервіси, використовують протокол SOAP. Це дозволяє клієнтським програмам, які не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє додатків під управлінням Java (таким як BEA і IBM WebSphere) з'єднуватися з Microsoft SQL Server 2017.

SQL Server підтримує віддзеркалення і кластеризації баз даних. Кластер сервера SQL - це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Всі

сервера мають одне віртуальне ім'я, і дані розподіляються по IP-адресами машин кластеру протягом робочого циклу. також в випадку відмови або збою на одному з серверів кластера доступний автоматичний перенесення навантаження на інший сервер.

У SQL Server 2017 вбудована підтримка .NET Framework. Завдяки цьому, збережені процедури БД можуть бути написані на будь-якій мові платформи .NET, використовуючи повний набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних в Microsoft .NET Framework). Однак, на відміну від інших процесів, .NET Framework, будучи базисної системою для SQL Server 2017 року, виділяє додаткову пам'ять і вибудовує засоби управління SQL Server замість того, щоб використовувати вбудовані засоби Windows. Це підвищує продуктивність в порівнянні з загальними алгоритмами Windows, так як алгоритми розподілу ресурсів спеціально налаштовані для використання в структурах SQL Server

2.4 AJAX

Коли існуючих можливостей ставати мало, а вдосконалювати існуюче вже нікуди, тоді і відбувається технологічний прорив. Таким проривом і є AJAX (Asynchronous JavaScript and XML) - підхід до побудови призначених для користувача інтерфейсів веб-додатків, при якому web-сторінка, що не перезавантажуючись, сама довантажує потрібні користувачу дані. AJAX – один з компонентів концепції DHTML [7]. Що ж дає нам ця технологія. В даний час розробка WEB додатків прагне до розмежування клієнтської частини і серверної. При розробці складних проектів виникає необхідність в структурованості і легкості читання коду. Не слід засмічувати код програміста кодом розробника, а код розробника - правками дизайнера, і так далі. Виникає необхідність в розмежуванні роботи. Так, наприклад, дизайнер робитиме свою роботу, розрозбник свою, програміст свою, і при цьому ніхто один одному заважати не буде.

У підсумку кожному учаснику проекту досить буде знати тільки ті дані, з якими йому доведеться працювати. В такому випадку продуктивність групи і якість проекту підвищується в рази.

Отже, AJAX - це ідея, яка базується на двох основних принципах:

– Використання DHTML для динамічної зміни змісту сторінки.

– Використання XMLHttpRequest для звернення до сервера "на льоту".

– Використання цих двох підходів дозволяє створювати набагато більш зручні WEB-інтерфейси користувача на тих сторінках сайтів, де необхідно активна взаємодія з користувачем.

– Використання Ajax стало найбільш популярно після того, як компанія Google почала активно використовувати його пристворенні своїх сайтів, таких як Gmail, Google maps і Google suggest.

Створення цих сайтів підтвердило ефективність використання даного підходу. Клієнт, набираючи в рядку пошуку адресу цікавить його ресурсу, потрапляючи на сервер, робить до нього запит. Сервер виробляє обчислення відповідно до запитом, звертається до бази даних і так далі, після чого отримані дані йдуть клієнтові і, в разі необхідності підставляються в шаблони і обробляються браузером. Результатом є сторінка, яку ми бачимо, і яку 80% населення країни перебуває в WEB називають Інтернетом. Це класична модель, яка встигла себе зарекомендувати і заслужити собі почесне місце під сонцем. Це найпростіша модель взаємодії і, як наслідок, найпоширеніша.

При зверненні до сервера, генерується сторінка, яка буде відображатися користувачеві, і пропонувати йому зробити потрібну йому послідовність дій. При свідомому (хоча і не обов'язково) виборі клієнта, його запит буде звертатися до AJAX модулю, який і буде виробляти всі потрібні йому обчислення і роботу з сервером як таким. Основна відмінність в тому що цей метод дає нам можливість динамічно звертатися до сервера і виконувати цікаві для нас дії. Наприклад, нам потрібно виконати звернення до бази даних і отримати цікаві для нас дані, які потім будемо використовувати

2.5 Бібліотека jQuery

jQuery - це популярна javascript бібліотека, здатна істотно спростити життя веб-розробнику. Бібліотека jQuery містить функціонал, корисний для максимально широкого кола завдань. Проте, розробниками бібліотеки не ставилося завдання суміщення в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, велика частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки і плагінів.

Це дозволяє зібрати для ресурсу саме той JavaScript-функціонал, який на ньому був би затребуваний [8]. Бібліотека jQuery в першу чергу забезпечує несуперечливу роботу програмного коду у всіх типах браузерів, вирішуючи такі складні проблеми JavaScript, як очікування завантаження сторінки перед тим, як виконувати будь-які операції. На той випадок, якщо в бібліотеці виявитися недолік функціональності, розробники передбачили простий, але дуже дієвий спосіб її розширення. Багато починаючі програмісти jQuery виявляють цю гнучкість на практиці, розширюючи можливості jQuery в перший же день.

Щоб привнести динамічну функціональність на будь-яку інтернет сторінку, доводитися слідувати одному і тому ж шаблону спочатку відбирається елемент або група елементів, а потім над ними виконуються деякі дії, наприклад, приховувати або показувати, що цікавлять нас елементи, додавати до них клас CSS, створювати анімаційні ефекти або змінювати атрибути.

З звичайним JavaScript для вирішення кожного із завдань буде потрібно десятки рядків програмного коду. Творець jQuery розробив свою бібліотеку саме для того, щоб зробити найбільш загальні завдання тривіальними. Наприклад, щоб створити таблицю з різним кольором фону для парних і непарних рядків, дизайнеру потрібно написати до 10 рядків коду на мові

JavaScript, а ось з використанням jQuery цей ефект досягається з використанням не більше ніж одного рядка.

2.6 NuGet

Важливим інструментом для будь-якої сучасної платформи розвитку є механізм, за допомогою якого розробники можуть створювати, обмінюватися та споживати корисний код. Часто такий код поєднується з "пакетами", які містять скомпільований код (як DLL), а також інший вміст, необхідний для проектів, які споживають ці пакети. Для .NET (включаючи .NET Core) механізм спільного використання, який підтримує Microsoft, - це NuGet, який визначає, як пакунки для .NET створюються, розміщуються та споживаються, і надає інструменти для кожної з цих ролей. Простіше кажучи, пакет NuGet є єдиним ZIP-файлом з розширенням .nupkg, який містить скомпільований код (DLL), інші файли, пов'язані з цим кодом, і описовий маніфест, що містить інформацію, подібну до номера версії пакета. Розробники з кодом для спільного використання створюють пакети та публікують їх у загальнодоступному або приватному хості. Споживачі пакетів отримують ці пакети від відповідних хостів, додають їх до своїх проектів, а потім викликають функціональність пакета в коді проекту. Потім сама NuGet обробляє всі проміжні деталі.

Оскільки NuGet підтримує приватні хости разом з загальнодоступним вузлом nuget.org, ви можете використовувати пакети NuGet для спільного використання коду, який є ексклюзивним для організації або робочої групи. Ви також можете використовувати пакунки NuGet як зручний спосіб обчислювати свій код для використання лише у ваших власних проектах. Коротше кажучи, пакет NuGet є спільним кодом, але не вимагає і не передбачає будь-яких конкретних засобів обміну[9].

Потік пакетів між творцями, хостами та споживачами. У своїй ролі як публічний хост, NuGet сама підтримує центральне сховище з більш ніж 100

000 унікальних пакунків на nuget.org. Ці пакети використовуються мільйонами розробників .NET / .NET Core щодня. NuGet також дозволяє розміщувати пакунки приватно в хмарі (наприклад, у Azure DevOps), у приватній мережі або навіть у вашій локальній файловій системі. Роблячи це, ці пакети доступні лише для тих розробників, які мають доступ до хоста, що дає вам можливість робити пакети доступними для певної групи споживачів. Варіанти пояснюються на хостингу власних каналів NuGet. За допомогою параметрів конфігурації можна також контролювати, до яких вузлів можна звертатися будь-яким комп'ютером, забезпечуючи тим самим, що пакунки отримуються з конкретних джерел, а не з публічного сховища, як nuget.org. Незалежно від його природи, хост служить точкою зв'язку між пакувальниками і споживачами пакетів. Творці створюють корисні пакети NuGet і публікують їх на вузлі. Споживачі потім шукають корисні та сумісні пакети на доступних хостах, завантажуючи та включаючи ці пакети в свої проекти. Після інсталяції в проекті API інших пакунків доступний для решти коду проекту. Зв'язок між розробниками пакету, пакунками та споживачами пакетів.

Сумісність з пакетом націлювання. Пакет "сумісний" означає, що він містить збірки, побудовані для принаймні одного цільового каркасу .NET, сумісного з цільовим середовищем споживача проекту. Розробники можуть створювати пакунки, специфічні для одного фреймворку, як і для елементів управління UWP, або ж вони можуть підтримувати більш широкий діапазон цілей. Щоб максимізувати сумісність пакета, розробники націлюються на .NET Standard, які можуть використовувати всі проекти .NET і .NET Core. Це найбільш ефективний засіб як для творців, так і для споживачів, оскільки єдиний пакет (зазвичай містить єдину збірку) працює для всіх споживчих проектів[10].

З іншого боку, розробники пакетів, які вимагають API за межами .NET Standard, створюють окремі збірки для різних цільових фреймворків, які вони хочуть підтримувати, і включають всі ці збірки в одному пакеті (який

називається "багатоцільовим"). Коли споживач встановлює такий пакет, NuGet витягує лише ті збірки, які потрібні проекту. Це зводить до мінімуму відбиток пакета в кінцевому додатку та / або збірки, що виробляються цим проектом. Багатоцільовий пакет є, звичайно, більш складним для його творця для підтримки.

Інструменти NuGet. На додаток до підтримки хостингу, NuGet також надає різноманітні інструменти, які використовуються як творцями, так і споживачами. Див. Розділ Встановлення інструментів клієнта NuGet для отримання спеціальних інструментів. Все створення, споживання nuget.exe CLI Забезпечує всі можливості NuGet, з деякими командами, що застосовуються спеціально до авторів пакунків, деякі застосовуються лише до споживачів, а інші - до обох. Наприклад, автори пакунків використовують команду nuget pack для створення пакета з різних збірок і пов'язаних файлів, споживачі пакунків використовують nuget install для включення пакетів в папку проекту, і кожен використовує nuget config для встановлення змінних налаштування NuGet.

2.7 Ninject

Ninject - це інжектор залежностей для .NET, практична реалізація шаблону Injection Dependency (форма інверсії шаблону управління). Ninject дозволяє впроваджувати не тільки ін'єкції в типи даних, але і додавати додатковий функціонал в методи. Вносити аспекти.

2.8 Boostrap

Bootstrap - це CSS фреймворк, який спочатку створювався для внутрішнього використання компанією «Twitter» з робочою назвою «Twitter Blueprint», але в підсумку був опублікований у відкритий доступ і став хорошим набором інструментів для front-end розробки під назвою «Bootstrap».

Переваги фреймворка Bootstrap:

Висока швидкість розробки макетів сторінок сайту. Bootstrap містить величезний набір готових рішень і елементів.

– Кросбраузерність і адаптивність сайту. Всі елементи фреймворка адаптивні під всі пристрої і коректно відображаються у всіх сучасних браузерах.

– Легкість у використанні. Навіть людина, що має базові знання про HTML і CSS, може вільно створювати web-сторінки з використанням фреймворку.

– Простота в навчанні. У Bootstrap дуже хороша документація з великою кількістю прикладів готового коду.

– Про якість фреймворка говорить те, що безліч тем оформлення для переважної більшості популярних CMS, таких як WordPress, Joomla і т.д., розроблені із застосуванням Bootstrap.

Хоча Bootstrap і називають CSS фреймворком, але це не зовсім вірно. На мій погляд, правильніше його називати WEB фреймворком, так як він містить готові CSS, HTML і JavaScript компоненти, а третя версія має власний іконочний шрифт. Шрифт містить більше 250 іконок. Кількість іконок, звичайно, не таке велике, як у Font Awesome, але все базові іконки присутні. З четвертої версії фреймворк відмовився від власного іконочние шрифту на користь використання сторонніх бібліотек, які необхідні користувачу для конкретного проекту[11].

Сітка Bootstrap.При верстці адаптивного класичного макета: шапка сайту (header), основна частина (content), бічна колонка (sidebar) і підвал сайту (footer), для коректного відображення нам потрібно розрахувати ширину у відсотках кожного елемента і привласнити обтікання. Якщо з шапкою і футером все зрозуміло, в більшості випадків ширина буде 100%, то для основної частини контенту і бічний колонки може бути 70/30 або 85/25, але при зменшенні екрану нас це не влаштує, потрібно буде робити по 100% і скидати обтікання. Ось для таких цілей і потрібна сітка Bootstrap. Просто

задаються класи для блоків, які вказують, яку ширину повинен займати елемент і як він буде відображатися на різних пристроях. Сітка функціонує як таблиця, в якій є свої ряди і стовпці, максимальна кількість стовпців 12[12].

Сітку можна робити всередині іншої сітки скільки завгодно. Якщо робити все блоки сайту з використанням сітки, то самостійно писати медіа запити для їх адаптивності взагалі не доведеться. Крім сітки існує величезна кількість всіляких компонентів: навігаційні меню, форми, таблиці, модальні вікна, вкладки, оповіщення, спливаючі підказки і т.д. Ще дуже зручно, що платформа Bootstrap дозволяє як завантажити весь фреймворк цілком, так і тільки ті компоненти, які потрібні.

2.9 Razor

Механізм перегляду використовує вміст ASP.NET і шукає інструкції, як правило, для вставки динамічного вмісту . Вихідні дані, відправлені в браузер і Razor - це назва механізму перегляду MVC Framework. У Razor немає змін у MVC 5, і якщо ви вже знайомі з синтаксисом з попередніх версій, ви можете пропустити його вперед.

2.10 Entity Framework

Структура Entity (EF) - це фреймворк ORM (об'єктно-реляційне відображення), який компанія Microsoft робить доступною як частина розробки .NET (версія 3.5 SP1 і новішої версії). Його мета полягає в абстрактних зв'язках з реляційною базою даних таким чином, що розробник може ставитися до об'єкта бази даних як до набору об'єктів, а потім до класів на додаток до їх властивостей. По суті, ми говоримо про розрив між нашими додатками і логікою доступу до даних, що виявляється головним плюсом. Наприклад: якщо нам потрібно перемістити - в контексті єдиної програми - базу даних різних виробників, потрібно буде переглянути спосіб і інструкції, з якими ми зв'язуємо чергового менеджера даних.

В даний час EF переважно допускає два типи підходів, пов'язаних з цим використанням. Вони є базами даних First і Code-First (перша відсутня в EF7, але все ще діє до версії 6.1.3). Різниця між двома підходами очевидна з їх назви, як у Database-First, ми опиняємося в позиції, де ми повинні моделювати вже існуючу базу даних (і, отже, виводити з неї наші об'єкти), в той час як в Code -mode По-перше, нам доведеться підготувати, надавши їм властивості, що представляють поля таблиці, щоб визначити структуру бази даних. Не обов'язково, що Code-First зобов'язаний спочатку працювати за відсутності бази даних, оскільки ми можемо моделювати класи існуючої бази даних і підключатися до неї для виконання звичайних операцій I / OR. Можна сказати, що два підходи, поза деякими інструментальними особливостями, являють собою свого роду індекс пріоритетів порівняно з тими, що при владі, при визначенні структури даних, з яким додатку доведеться робити "перед базою даних" (з якої вони похідні класи) або "перед" кодом (з якого модель бази даних може бути текстурована)[13].

Він був розроблений, щоб бути легким, розширюваним і підтримувати розвиток крос-платформних систем як частину основи Microsoft .NET. Він також був розроблений, щоб його було простіше використовувати, а також для поліпшення продуктивності порівняно з попередніми версіями Entity Framework.

EF - об'єктно-реляційний картограф (ORM). Об'єктно-реляційне відображення - це техніка, яка дозволяє розробникам працювати з даними в об'єктно-орієнтованому способі, виконуючи роботу, необхідну для зіставлення об'єктів, визначених у мові програмування програми, і даних, що зберігаються в реляційних джерелах даних.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами. Тому EF вдало підходить.

Таблиця 2.1 – Опис інтерфейсів системи

Інтерфейс	**Опис**
IVacancyRepository	Інтерфейс, який отримує послідовність об'єктів Vacancy.
IDetailsProcessor	Інтерфейс, який буде обробляти інформацію про вибрані вакансії.
IAutorizProvider	Інтерфейс, що використовується для провірення авторизації.

Таблиця 2.2 – Опис класів (структур) системи

Клас (структура)	**Опис**
NinjectDependencyResolver	Клас, який реалізує інтерфейс IDependencyResolver який належить простору імен System.Web.Mvc і застосовується MVC Framework для отримання необхідних об'єктів.
NinjectWebCommon	Клас, який створює зв'язок між класом NinjectDependencyResolver.
Vacancy	Клас, який реалізує модель предметної області, в моєму проекті це вакансія .
VacancyController	Клас, контроллер який буде відображати інформацію по вакансіям і реалізовувати представлення.
RouteConfig	Клас, який визначає маршрути проекту .
EFDbContexts	Клас, який асоціюватиме модель Vacancy з базою даних.

Продовження таблиці 2.2

Клас (структура)	**Опис**
EFVacancyRepositorys	Клас, який створює сховище для об'єкту Vacancy.
PagingInf	Клас, який передає інформацію про кількість доступних сторінок.
PagingHelp	Клас, допоміжний клас для реалізації кількості сторінок для вакасій.
VacancyListViewModel	Клас, який передає дані з контролера в представлення.
NavigetController	Клас, який створює контролер для навігаційної панелі.
VacancyCard	Клас, який реалізує модель предметної області в частості це додання ваканцій.
VacancyCardController	Клас, який створює контролер для VacancyCard.
VacancyCardListViewModel	Клас, який передає дані з контролера для відображення VacancyCard.
VacancyCardModeBinders	Клас, який зв'язує модель шляхом реалізації інтерфейсу.
VacancyAddDetails	Клас, який буде використовуватися для уявлення про додану вакансію.
AdminContoller	Клас, який створює контролер для відображення представлення адмін-панелі.
AutorizConcrete	Клас, який реалізує інтерфейс для авторизації.

Продовження таблиці 2.2

Клас (структура)	**Опис**
LogViewModel	Клас, який передає дані котролеру і обробляє їх для авторизації.
AcController	Клас,який реалізує контролер для відображення форми авторизації.
EmailSet	Клас,виконує роль відправки для поданої заявки на вакансії.

Таблиця 2.3 – Опис методів класів та інтерфейсів системи

Клас/Інтерфейс	**Метод**	**Опис**
IVacancyRepository	DeleteVacancy(int VacancyId)	Видаляє обрану вакансію.
IVacancyRepository	SaveVacancy(Vacancy vacancy)	Зберігає додану вакансію у базі даних.
IDetailsProcessor	DetailsProcessor(VacancyCard Vacancycard)	Обробка за вибраними вакансіями.
VacancyAddDetails	EmailOrderProc()	Відправлює дані на вказану пошту.
VacancyAddDetails	ProcOrder(VacancyCard Vacancycard)	Вказує кількість обраних вакансій.
EFVacancyRepositorys	DeleteVacancy(int vacancyId)	Видаляє вакансію з бд.
EFVacancyRepository	EditVacancy(int vacancyId)	Дозволяє зберігати змінені дані щодо обраної вакансії в базу даних.

Продовження таблиці 2.3

Клас/Інтерфейс	**Метод**	**Опис**
EFVacancyRepositorys	SaveVacancy(Vacancy vacancy)	Додає вакансію в базу даних.
VacancyCard	AddItem(Vacancy vacancy, int quantity)	Додає в список обрану вакансію.
VacancyCard	RemoveLin(Vacancy vacancy)	Обновлює статус доданих вакансій і надає дані користувачеві.
VacancyCard	Clear()	Видалення обраних вакансій з .
NinjectWebCommon	Stop()	Зберігає репозиторій у базі даних.
NinjectWebCommon	Start()	Знаходить репозиторій у базі даних по назві.
NinjectWebCommon	RegisterServices(Ikernel kernel)	Знаходить репозиторій у базі даних по ідентифікатору.
NinjectWebCommon	CreateKernel()	Створює репозиторій з бази даних і додає залежність між ними.
AcController	TransController()	Метод який перенаправлює користувача на нове представлення.

Продовження таблиці 2.3

Клас/Інтерфейс	Метод	Опис
RouteConfig	RegisterRouter(RouteCollection routes)	Метод який реалізує всю маршрутизацію та роботу контролерів.
AcController	AccountController(IAuthProvider auth)	Перевіряє чи авторизувався користувач.
AcController	ActionResult Login(LoginViewModel model, string returnUrl)	Виконує показ представлення.
AcController	Login()	Перенаправляє на форму авторизації .
AdminContoller	AdminController(IVacancyRepository repo)	Показує список всіх вакансій у вигляді таблиці
AdminContoller	Create()	Метод ,який створює нову колонку для створення нової вакансії
AdminContoller	Save()	Метод який зберігає дані введені користувачем.
AdminContolle	SaveTo(LoginViewModel)	Метод який передає дані, щоб дізнатися чи прийшов користувач аутентифікацію

Продовження таблиці 2.3

Клас/Інтерфейс	**Метод**	**Опис**
AdminContoller	Delete(int vacancyId)	Видяляє з бази даних за ідентифікатором обрану вакансії та зберігає зміни в бд.
AdminContoller	Edit(Vacancy , HttpPostedFileBase image = null)	Знаходить та додає налаштування для картинки у базі даних
AdminContoller	Edit(int vacancyId)	Знаходить кількість налаштувань для вакансій у базі даних
AdminContoller	Index()	Перенаправляє на інше представлення, а саме на представлення форми введення даних для авторизації.
VacancyCardController	AddToCart(VacancyCart Vacancycart, int VacancyId, string returnUrl)	Додає кілька вакансій в спеціальну форму для обраних вакансій і повертає користуваче кількість і статус доданих вакансій
VacancyCardController	CartController(IVacancyRepository repo, IDetailsProcessor processor)	Записує скільки і яких вакансій було додано.

Продовження таблиці 2.3

Клас/Інтерфейс	**Метод**	**Опис**
VacancyCardController	Checkout()	Знаходить по ідентифікатору і робить перевірку по доданій вакансії.
VacancyCardController	Checkout(VacancyCard vacancycard)	Знаходить по індентифікатору та перевіряє чи є обрана вакансія в базі даних .
VacancyCardController	Index(Vacancy vacancy, string returnUrl)	Перенаправляє на інше представлення, а саме на сторінку з відображенням всіх вакансій які добавив користуач.
VacancyCardController	RemoveFromCart(VacancyCard vacancycard, int VacancyId, string returnUrl)	Знаходить усі ревізії і повертає у базі даних.
VacancyCardController	Summary(VacancyCard Vacancycard)	Знаходить кількість усіх доданих вакансій та оброблює ці дані для виведення інформації.
VacancyController	List(string category,int page = 1)	Зберігає і водображає кількість вакансій з базі даних.

Продовження таблиці 2.3

Клас/Інтерфейс	**Метод**	**Опис**
VacancyController	GetImag(int VacancyId)	Створює побудову картинок у базі даних.
VacancyController	VacancyController(IVacancyRepository repo)	Знаходить усі вакансії у базі даних.
NavigetController	Menu(string category = null, bool horizontalNav = false)	Відображає меню і виводить необхідне меню.
NavigetController	NavigetController(IVacancyRepository repo)	Створює артефакт побудови у базі даних.
PagingHelp	MvcHtmlStringPageLink(this HtmlHelperhtml,PagingInfo pagingInfo,Func<int,string> pageUrl)	Знаходить і відображає потрібну кількість сторінок.
VacancyCardModeBinders	BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)	Знаходить побудови у базі даних за ідентифікатором.
PagingHelp	Create(paginInfo,Func<int,string> pageUrl)	При потрібності додає і розбиває на певну кількість сторінки для розміщення вакансій, щоб не перевищувало норму.

Продовження таблиці 2.3

Клас/Інтерфейс	**Метод**	**Опис**
IAutorizProvider	Authenticate(string username, string password)	Знаходить побудову у базі даних за ідентифікаторомта робить провірку.
AutorizConcrete	Authenticate(string username, string password)	Знаходить кількість усіх побудов у базі даних.
NinjectDependencyResolver	AddBindings()	Додає всі модулі для створення зв'язку між певними модулями.
NinjectDependencyResolver	GetService(Type serviceType)	Надіслати повідомлення для запуску процесу інтеграції.
NinjectDependencyResolver	IEnumerable<object> GetServices(Type serviceType)	Створити канал для отримання повідомлень для запуску процесу.
NinjectDependencyResolver	NinjectDependencyResolver(IKernel kernelParam)	Створити залежність для запуску процесу.

2.11 Аналіз безпеки даних

Аутентифікація налаштовується за використанням елемента <authentication>.. Дотримуватися аутентифікації за допомогою форм, оскільки вона працює з локальними користувача обліковими даними і проста в налаштуванні і управлінні.

Головними альтернативами аутентифікації за допомогою форм є аутентифікація Windows, при якій для ідентифікації користувачів застосовуються облікові дані операційної системи, і аутентифікація через обліковий запис організації (Organizational Account), коли користувач ідентифікується з використанням хмарної служби, подібної Windows Azure.

Атрибут loginUrl повідомляє ASP.NET, куди перенаправляти користувачів, коли вони потребують аутентифікації (URL в даному випадку виглядає як ~ / Account / Login), а в атрибуті timeout зазначено період, протягом якого користувач залишається аутентифицироваться після успішного входу, виражений в хвилинах (2890 хвилин, тобто 49 годин).

Незважаючи на неприйнятність для реальних проектів, використання файлу Web.config для зберігання облікових даних дозволяє зосередитися на засобах MVC, не відволікаючись на аспекти ядра платформи ASP.NET. В результаті додавань в файл Web.config ми маємо жорстко закодовані ім'я користувача (admin) і пароль (123456).

Використання засобу аутентифікації за допомогою форм вимагає виклику двох статичних методів класу System.Web.Security.FormsAuthentication:

– метод Authenticate () дозволяє перевірити облікові дані, надані користувачем;

– метод SetAuthCookie () додає cookie до відповіді, призначеному для браузера, щоб користувачам не довелося проходити аутентифікацію кожен раз, коли вони роблять запит.

ВИСНОВКИ ПО РОЗДІЛУ

У даному розділі було розроблено архітектуру проекту комплексу задач. Обрано операційну систему, що найкраще підходить у якості платформи для розробки. Було обрано найкращу мову для написання та веб інтерфейс, вирішено метод та протокол обміну інформацією між модулями. Обрано архітектурні паттерни. Описано інтерфейси, класи, структури даних комплексу та описано їх методи. Проаналізовано безпеку даних у комплексі. Та виділено всі перпективи обраних рішень, а саме чому притримувалося певний тип аутентифікації і можливі його недостатки. Описано чому вибрані технології підходять під задачі для проекту, а також описано всі методи які використовуються в даному проекті. Та чому розроблений інтерфейс зручний як для потенційного користувача.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Етап тестування є важливим в процесі розробки даного продукту у зв`язку з великою складністю системи. Також продукт передбачає постійну взаємодію із зовнішніми онлайн сервісами.

Тестування програмного забезпечення — техніка контролю якості, що перевіряє відповідність між реальною і очікуваною поведінкою програми завдяки кінцевому набору тестів, які обираються певним чином[14].

Характеристика програмного забезпечення, ступінь відповідності ПЗ до вимог. При цьому вимоги можуть трактуватись по-різному, що породжує декілька незалежних визначень терміну. Якість ПЗ – набір властивостей продукту (сервісу або програм), що характеризують його здатність задовольнити встановлені або передбачувані потреби замовника. Поняття якості має різні інтерпретації залежно від конкретної програмної системи і вимог до неї.

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення й забезпечення якості програмного забезпечення, до якого належать всі складові ділового процесу, а не тільки тестування.

Зазвичай, поняття якості обмежується такими поняттями як коректність, надійність, практичність, безпечність, але може містити більше технічних вимог, котрі описані у стандарті ISO 9126. Склад та зміст супутньої документації процесу тестування визначається стандартом IEEE 829—1998 Standard for Software Test Documentation [15]. Існує багато підходів до

тестування програмного забезпечення, але ефективне тестування складних продуктів — це по суті дослідницький та творчий процес, а не тільки створення та виконання рутинної процедури.

План тестування ПЗ включає в себе обсяг, підхід, ресурси та план усіх методів тестування. План описує об’єкти та функції що будуть протестовані, тип тестів, ресурси та план необхідний для виконання тестування.

Маючи автоматизовані тести, це відмінний спосіб гарантувати, що програмне забезпечення робить те, що мають намір зробити його автори. Існує кілька типів тестів для програмних додатків. До них належать інтеграційні тести, веб-тести, навантажувальні тести та інші. Блокові тести перевіряють окремі компоненти та методи програмного забезпечення. Тестові одиниці повинні перевіряти код лише в контролі розробника. Вони не повинні перевіряти проблеми інфраструктури. До питань інфраструктури належать бази даних, файлові системи та мережеві ресурси.

Також майте на увазі, що є кращі практики написання тестів. Наприклад, Test Driven Development (TDD) - це тестовий модуль, який записується перед тим, як він перевіряється. TDD подібний до того, як створити план для книги, перш ніж ми її напишемо. Він покликаний допомогти розробникам писати простіший, більш читаний і ефективний код.

3.2 Модульне тестування

Користувався вбудованою підтримкою модульного тестування, існуючої в Visual Studio, хоча доступні і інші пакети модульного тестування. Найбільш популярним з них є NUnit, однак все пакети тестування в основному справі схожі. Вибір інструментів тестування Visual Studio пов'язана з привабливістю інтеграції з залишеними частками IDE-середов

Ціль модульного тестування - відокремлювати окремі частини програми і показувати, що ці окремі частини програмного коду повністю працездатні і їх робота задовільнає розробника.

Таблиця 3.1 – Модульне тестування

Мета тесту	Результат
Розбиття на сторінки	Модульне тестування розбиття на сторінки провести, створивши імітована сховище, запровадивши його в конструктор класу VacancyController і викликавши метод List (), щоб запросити конкретну сторінку. Потім отримані об'єкти Vacancy можна порівняти з тими, які очікувалися від тестових даних в імітованої реалізації .Звертаємося до властивості Model об'єкта результату, щоб отримати послідовність IEnumerable <Vacancy>, згенерувала методом List (). Потім виконується перевірка, чи є ці дані очікуваними. В цьому випадку перетворили послідовність в колекцію за допомогою LINQ-методу ToList () і перевірили довжину і значення окремих об'єктів.
Метод дії Index ()	Нас цікавить поведінка методу дії Index () в контролері Admin, яке полягає в коректному поверненні об'єктів Vacancy, що знаходяться в базі даних. Це можна протестувати шляхом створення імітованої реалізації сховища і порівняння тестових даних з даними, що повертаються методом дії. Це дуже важливий компонент і тому для тестування даного методу дії окремо протестуємо кожем метод для Index(). Потрібно переконатися чи виводиться правильна кількість вакансій з моделі Vacancy. Повернені дані мають також пройти перевірку, щоб уникнути неправильності виводу інформації користувачеві та зберігтися. А також протестувати як поводить імітована реалізації сховища.

Продовження таблиці 3.1

Створення посилань на сторінки	Щоб протестувати допоміжний метод PageLinks (), ми викликаємо метод з тестовими даними й порівнюємо результати з очікуваної HTML-розміткою.Цей тест перевіряє висновок допоміжного методу з використанням літерального строкового значення, що містить подвійні лапки. Мова C # дозволяє успішно працювати з такими рядками. Потрібно тільки не забувати випереджати рядок символом @ і застосовувати два набори подвійних лапок ("") замість одного. Необхідно також пам'ятати, що не можна розносити літерально рядок в кількох рядках файлу, якщо тільки рядок, з якої проводиться порівняння, що не рознесена аналогічним чином. Наприклад, літерал, який використовується в тестовому методі, був розміщений в двох рядках через недостатньої ширини сторінки сайту. Ми не додавали символ нового рядка - інакше тест не пройшов би.
Дані розбиття на сторінки для моделі представлення	Необхідно упевнитися, що контролер відправляє за поданням правильну інформацію про розбиття на сторінки і впевнетися що його робота коректно відображає кількість вакансій на одній сторінці і не перевищувала певну кількість вакансій на одній сторінці, а саме не більше 4 вакансій на одній сторінці. Також упевнитися що правильна взаємодія з класом VacancyCard і чи поступають збережені або змінені дані до даного методу та їх обробка для створення відповідної кількості сторінок. А також перевірити чи даний метод приймає дані про категорії і відображає та розбиває на сторінки певні вакансії в даній категорії.

Продовження таблиці 3.1

Фільтрація по категорії	Модульний тест для перевірки функціональності фільтрації по категорії, щоб упевнитися в тому, що фільтр може коректно генерувати відомості про товари зазначеної категорії. Цей тест створює імітована сховище, що містить об'єкти Vacancy, які 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.Потрібно дізнатися чи відбувається правильне отримання даних з бази даних та чи інтерфейс правильно взаємодії з даним модулем. А також чи зберігаються ці дані і оновлюються при додавані нових вакансій через адмін панель.
Контролер VacancyCard	Модульне тестування класу VacancyCartController здійснюється за рахунок створення об'єктів VacancyCart і передачі їх методам дій. Нам необхідно протестувати три різних аспекти цього контролера. Метод AddToVacancyCart () повинен додавати обраний товар в корзину для вакансій користувача.Після додавання вакансії в кошик для вакансій має відбутися перенаправлення на представлення Index і правильному відображенні доданої вакансії .Методу дії Index () повинен бути коректно переданий URL, за яким користувач може повернутися в каталог. Також контролер має взаємодіяти з класом Vacancy для того щоб оновлена інформація по доданих або видалених вакансіях відображалася. Також перевірим роботу додавання кількох вакансій і правильне їх відображення.

Продовження таблиці 3.1

Генерація списку категорії	Модульний тест, призначений для перевірки можливості генерації списку категорій. Мета полягає в створенні списку, який відсортований в алфавітному порядку і не містить дублікатів. Для цього найпростіше побудувати тестові дані, які мають дубльовані категорії і не відсортовані належним чином, передати їх в NavigetController і встановити твердження, що дані будуть відповідним чином очищені. А також потрібно дізнатися чи даний модуль коректно взаємодіє з модулем Vacancy і які саме екземпляри беруться з бази даних. Також в тесті потрібно перевірити як саме буде поводити себе генерація списку по категоріям якщо змінити дані через адміністративну панель, або якщо адміністратор при внесенні даних зробить помилку. Усередині тесту створюється імітована реалізація сховища, яка містить повторювані категорії і категорії, які не відсортовані в алфавітному порядку. Потім визначається твердження про те, що дублікати будуть видалені і алфавітний порядок відновлений.
Повідомлення про обрану категорію	Для виконання перевірки того, що метод дії Menu () коректно додав деталі про обраної категорії, в модульному тесті можна прочитати значення властивості ViewBag, яке доступне через клас ViewResult.Також потрібно протестувати чи метод дії Menu () відобразив правильні категорії з списка вакансій та правильні екземпляри з класу VacancyCard були взяті. І чи правильні дані надходять контролеру для подальшого відображення користувачу.

Продовження таблиці 3.1

Метод дії Edit()	У методі дії Edit () потрібно протестувати два аспекти поведінки. Перший з них полягає в тому, що ми отримуємо запитуваний товар, коли надаємо допустиме значення ідентифікатора. Очевидно, необхідно упевнитися, що ми редагуємо товар, який очікували. Другий аспект поведінки полягає в тому, що ми не повинні отримувати товар при запиті значення ідентифікатора, який відсутній в сховище У методі дії Edit () потрібно протестувати два аспекти поведінки. Перший з них полягає в тому, що ми отримуємо запитуваний товар, коли надаємо допустиме значення ідентифікатора. Очевидно, необхідно упевнитися, що ми редагуємо товар, який очікували. Другий аспект поведінки полягає в тому, що ми не повинні отримувати товар при запиті значення ідентифікатора, який відсутній в сховище У методі дії Edit () потрібно протестувати два аспекти поведінки. Перший з них полягає в тому, що ми отримуємо запитуваний товар, коли надаємо допустиме значення ідентифікатора. Очевидно, необхідно упевнитися, що ми редагуємо товар, який очікували. Другий аспект поведінки полягає в тому, що ми не повинні отримувати товар при запиті значення ідентифікатора, який відсутній в сховище
Відправки, пов'язані з редагуванням	У методі дії Edit (), що обробляє запити POST, ми повинні упевнитися, що сховища вакансій для збереження передаються допустимі поновлення об'єкта Vacancy, створеного звязувачем для моделі. Крім того, необхідно перевірити, що неприпустимі обновлення (тобто містять помилки моделі) в сховище не передаються і чи сховище коректно приймає ці дані на подальшу роботу

Продовження таблиці 3.1

Лічильник вакансій певної категорії	Протестувати можливість генерації коректних лічильників вакансій для різних категорій необхідно створити імітована сховище, яке містить відомі дані в діапазоні категорій, і потім викликати метод дії List (), запитуючи кожну категорію по черзі в модульному тесті також викликається метод List () без вказівки категорії, щоб упевнитися в правильності підрахунку загальної кількості товарів.
Опрацювання замовлення	Замовлення має оброблятися, тільки якщо в кошику присутні елементи, і користувач надав достовірні деталі про доставку. При будь-яких інших обставин користувачеві повинно бути повідомлено про помилку.Цей тест дозволяє перевірити неможливість переходу до звязку при порожньому кошику для вакансій. Для цього ми переконуємося, що метод ProcessOrder () імітованої реалізації IOrderProcessor ніколи не викликається, що метод повертає стандартне уявлення (яке відобразить дані, введені користувачем, і дозволить відкоригувати їх), а також що стан моделі, передане поданням, позначено як неприпустиме.. Наступний тестовий метод працює в основному так само, але впроваджує в модель уявлення помилку, емулює проблему, про яку повідомляє контролеру моделі (подібне відбувається у виробничій версії, коли користувач вводить некоректні дані про доставку.Упевнившись, що порожня кошик або некоректні дані про додану вакансію запобігають обробку замовлення, необхідно перевірити.

Продовження таблиці 3.1

Перевірка доданих вакансій	Клас VacancyCart відносно простий, але в ньому є кілька важливих аспектів поведінки, в коректній роботі яких необхідно упевнитися. Невірно функціонуючий кошик для вакансій порушить роботу всієї програми. Протестувати кожен засіб окремо. Перше відноситься до додавання елемента в кошик для вакансій. При найпершому додаванні в кошик об'єкта Vacancy повинен бути доданий новий екземпляр VacancyCartLine. однак якщо користувач вже додав об'єкт Vacancy в кошик для вакансій, необхідно збільшити кількість у відповідному екземплярі VacancyCartLine, а не створювати новий.Перевірити, що користувачі мають можливість змінювати своє рішення і видаляти товари з кошика для вакансій. Це засіб реалізовано у вигляді методу RemoveLine(). Та протестувати чи зменшується кількість вакансій у відповідному екземплярі VacancyCartLine, а не відбудеться створювання нового. Наступне перевіряється поведінка стосується можливості обчислення загальної вартості елементів в кошику для вакансій.В результаті очищення кошика для вакансій її вміст коректно видаляється. А також протестувати чи екземпляри класу VacancyCartLine додаються і правильно виводиться вся вартість обраних вакансій і як буде взаємодіяти при видаленні обраної вакансії через адміністративну панель, а саме як буде поводити себе екземпляри Vacancy і чи не буде створюватися окремий каталог.

Продовження таблиці 3.1

Видалення вакансій	Тестуванню підлягає основне поведінку методу дії Delete (), яке полягає в тому, що при передачі в якості параметра допустимого ідентифікатора VacancyId метод дії повинен викликати метод DeleteVacancy () сховища і передати йому коректне значення VacancyId видаляється товару. Необхідно протестувати яке саме значення надходить з VacancyId та як ці значення перезаписуються. Крім цього цей тест проводиться для того щоб дізнатися скільки разів користувач зможе видаляти вакансію і чи може це якось призвести до неполадок.
Аутентифікація	Тестування контролера Account вимагає перевірки двох аспектів поведінки: користувач повинен бути аутентифікований, якщо надав правильні облікові дані, і користувач не повинен бути аутентифікований, якщо надав некоректні облікові дані. Ці тести можна виконати за рахунок створення імітованих реалізацій інтерфейсу IAuthProvider і перевірки типу і природи результату, повернутого методом Login () контролера. Також потрібно перевірити чи дані надходять у базу даних та правильність їх обробітку.Важливий фактор також є повідомлення користувача про неправильність введеня даних та вказівки на якому саме полі в формі було допущено помилку, та як все це оброблює контролер і які дані поступають йому для подальшої обробітки. Необхідно зазначити кількість спроб авторизуватися та протестувати як буде поводити себе контролер.

Продовження таблиці 3.1

Зображення	Ми повинні упевнитися в тому, що метод GetImage () повертає коректний тип MIME зі сховища, і що ніякі дані не повертаються, якщо запитаний ідентифікатор товару не існує. Нижче показані необхідні тестові методи. Коли ми маємо справу з допустимим ідентифікатором товару, ми перевіряємо отримання результату FileResult з методу дії і відповідність типу вмісту типу імітованих даних. Клас FileResult не дозволяє мати доступ до двійкового вмісту файлу, тому доведеться задовольнятися таким недостатньо надійним тестом. У разі запиту неприпустимого ідентифікатора товару ми просто перевіряємо, що результатом є null.

ВИСНОВКИ ПО РОЗДІЛУ

В даному розділі проводилося модульне тестування найважливіших компонентів в проекті. За допомогою них було виясненно що всі модулі функціонують правильно і не порушують роботу всієї системи. Також розглянуто та проаналізовано, які саме тести використовувалися і чому.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для повного розгортання даного комплексу задач потрібно виконати наступні етапи:

– Встановлення IIS і налаштування;

– Створити і налаштувати базу даних;

– Встановіть права облікового запису Windows для користувача робочого процесу;

– Конфігурація файлів в проекті;

4.1.1 Встановлення IIS

IIS (Internet Information Services - інформаційні служби Інтернету), а його поточною версією - IIS 8. Коли він був вперше реалізований, IIS був базовий веб-сервер. З роками IIS розвинувся в складний сервер додатків, що надає широке безліч функціональних засобів, найбільш важливим з яких є підтримка хостингу додатків ASP.NET.

Для встановлення IIS потрібно:

– Відкрити панель програм;

– Вибрати ”Програми”;

– Нажати кнопку “Включення компонентів Windows”;

– Знайдіти елемент Internet Information Services (Служби IIS) у верхній частині списку і натисніть на галочку щоб включити його;

– Перевірити чи вибрана підтримка ASP.NET

4.1.2 Створення і налаштування бази даних

Важливо не тільки створити базу даних і її таблиці, але і налаштувати облікові записи для входу на сервер бази даних і користувачів бази даних. Не забувайте, що в разі застосування вбудованої аутентифікації для підключення до бази даних SQL Server обліковий запис, під якою виконується ASP.NET (обліковий запис пулу додатків або обліковий запис aspnet_wp.exe), буде потрібно конфігурувати як користувач бази даних програми. Підхід веб-розгортання може спростити розгортання бази даних..

4.1.3 Встановлення права облікового запису Windows для користувача робочого процесу

Права облікового запису Windows для користувача робочого процесу. Користувач, від імені якого запускається робочий процес (w3wp.exe), потребує доступу для читання до каталогів додатки. Якщо додаток звертається до інших ресурсів, наприклад, до системного реєстру або до журналу подій, для облікового запису робочого процесу знадобиться настроїти дозвіл на доступ до цих ресурсів

4.1.4 Конфігурація файлів в проекті

Якщо потрібно обробляти будь-які URL-адреси з розширеннями імен файлів, які відрізняються від розширень, зареєстрованих при установці ASP.NET за замовчуванням, додайте відображення файлів IIS.

Налаштуйте ASP.NET (і налаштування IIS 8.0) в файлі web.config для виробничих середовищ. Іншими словами, додайте (або змінюйте) будь-які потрібні рядки з'єднання і параметри налаштування програми, а також параметри безпеки і авторизації, параметри налаштування стану сеансу і параметри налаштування глобалізації.У деяких випадках необхідно також модифікувати файл machine.config. Наприклад, якщо робота виконується в середовищі веб-хостингу, а з метою балансування навантаження додаток

працює на безлічі веб-серверів, необхідно синхронізувати будь-які ключі шифрування, які використовуються для шифрування мандатів аутентифікації за допомогою форм або стану подання на всіх цих комп'ютерах. Ці ключі зберігаються в файлі machine.config і повинні бути однаковими на кожному комп'ютері веб-ферми з тим, щоб один комп'ютер міг розшифрувати інформацію, зашифровану іншим комп'ютером, раніше обробляли запит

4.2 Розгортання проекту

Перемістіть файли веб-сайту на сервер будь-яким підходящим способом - за допомогою загального мережевого диска, знімного диска USB, скопіюйте файли .aspx і. aspx.cs в каталог FileCopy, створений на сервері.

Коли файли будуть скопійовані, поверніться у вікно IIS Manager на сервері, натисніть правою кнопкою на папці FileCopy в деревовидному поданні і в контекстному меню виберіть пункт Refresh. У нижній частині екрана натисніть на кнопці Content View (Перегляд вмісту). FileCopy в IIS Manager і натисніть на посиланні Browse (Огляд) в правій частині вікна. Відкриється веб-браузер з завантаженим URL-адресою створеної папки.

4.3 Інструкція користувача

4.3.1 Головна сторінка

Коли користувач вперше заходить на мій веб проект перед ним відображається головна сторінка .На цій сторінці користувач може бачити список з усіх вакансій і доданим логотипом до кожної з ним. Перше що бачить користувач це назва проекту яка знаходиться по ліву сторону сторінки. Також перед ним можливий вибір яка саме категорія з вакансіями його цікавить, якщо наприклад шукають конкретного спеціаліста в певній області. Також в правому куті сторінки є відображення кількості доданих вакансій. Також є можливість переглянути кожну з них. Також користувач може переходити на інші сторінки з вакансіями.

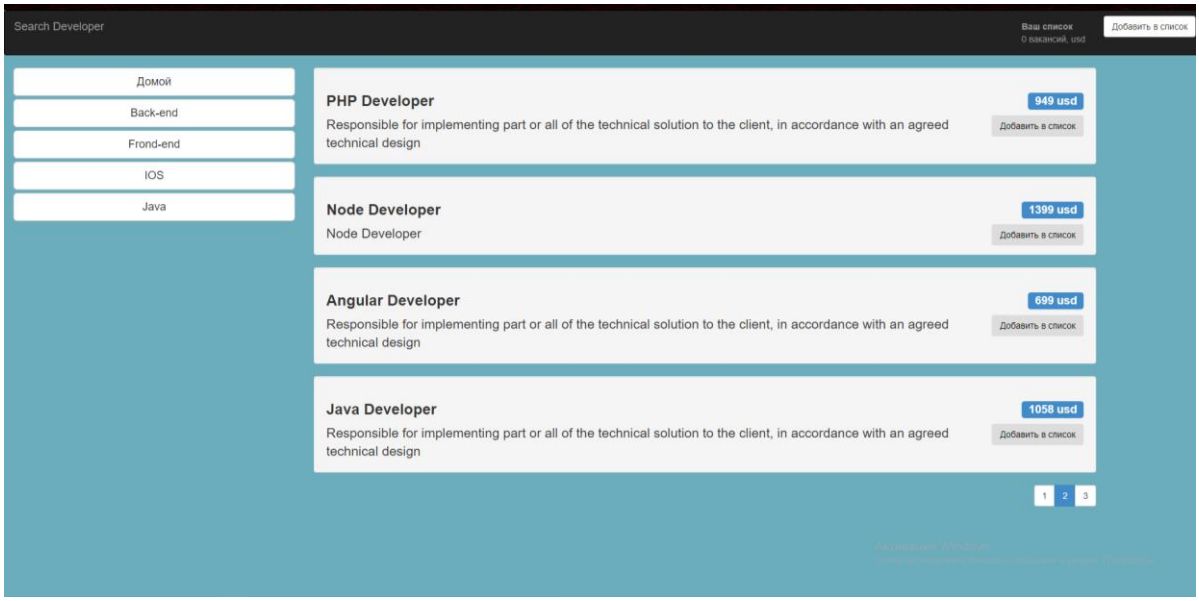

Рисунок 4.1 — Головна сторінка

4.3.2 Вибір по категоріям

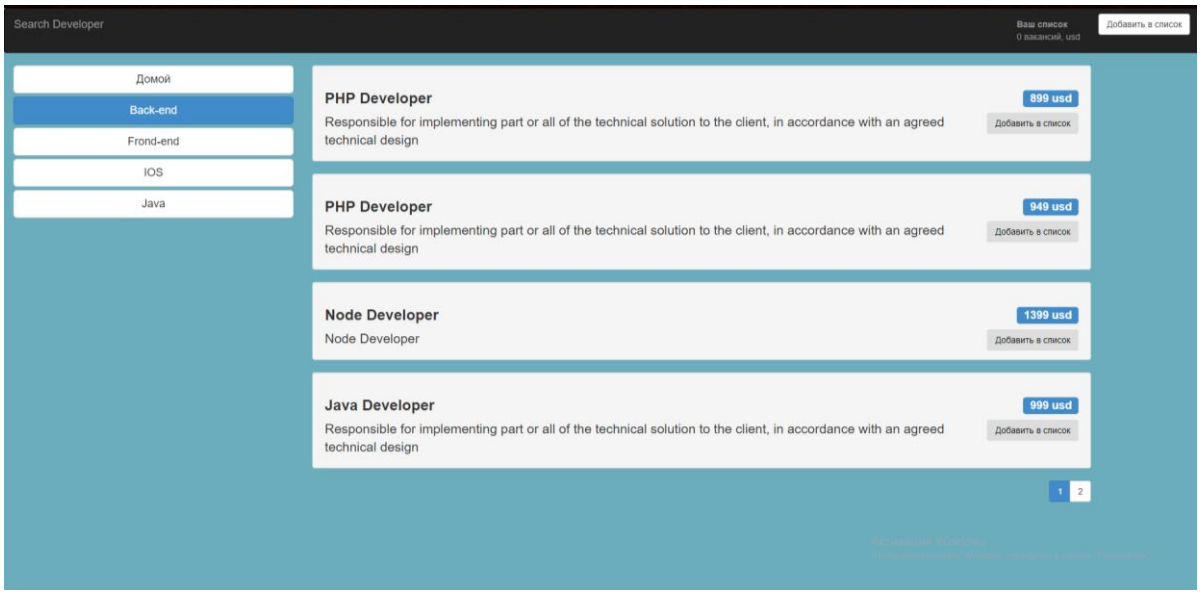

Рисунок 4.2 — Категорії

Користувач має змогу обирати вакансії за категоріями, при натисканні на обрану категорію відкривається список з вакансіями даної категоріями по 4 вакансії на одній сторінці. Також є можливість додавання вакансій у список, якщо користувачу це потрібно .

4.3.3 Список вакансій

Користувач може додавати кілька вакансій в список, якщо йому це потрібно. Додана вакансія додається до списку і користувач може переглянути додані вакансії.

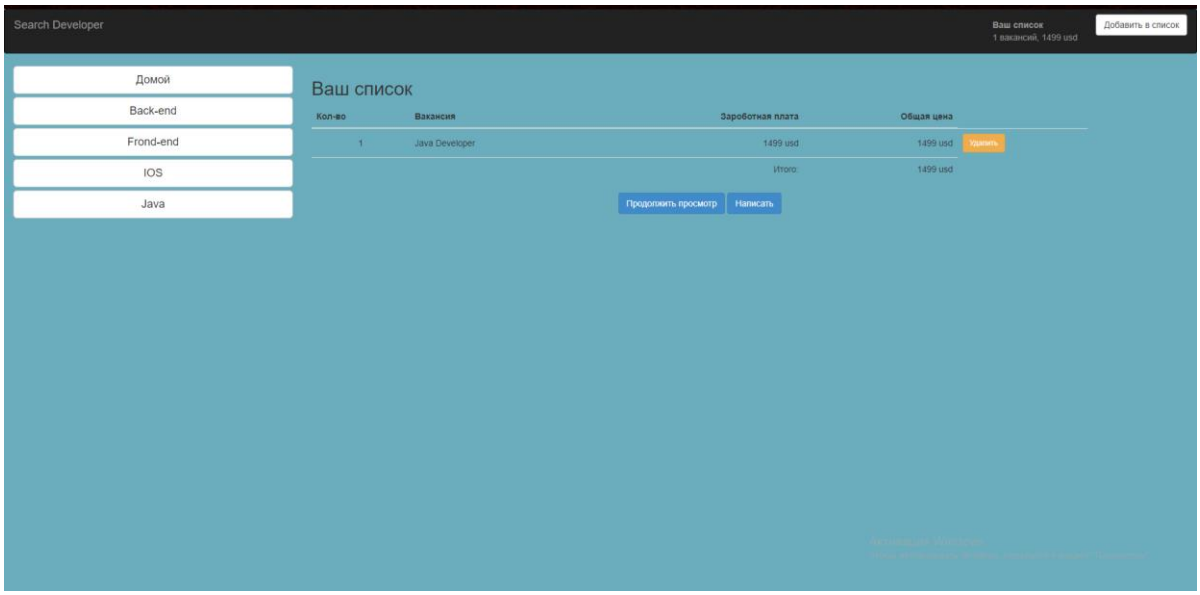

Рисунок 4.3 — Список вакансій

На даній сторінці користувача чекає новий функціонал, а саме він може видаляти додані ним вакансії. Може повернутися до головної сторінки для подальшого вибору вакансії.

Також при додаванні вакансій сумується вся сума заротної плати кожної вакансії.

Можна видаляти обрані вакансії, якщо на те є причина і відбудеться зміна в списку вакансій. При видаленні доданої вакансії потрібно натиснути відповідну кнопку.

Розроблений мною інтерфейс має інтуїтивний інтерфейс і спрямований на те, щоб користувачу було досить швидко ознайомитися з функціоналом і зручно користуватися проектом.

Також передбачений інтерфейс для мобільних пристроїв. Тому це дозволяє розширити базу користувачів, на даний момент багато людей використовують

4.3.4 Зв'язок з робітником

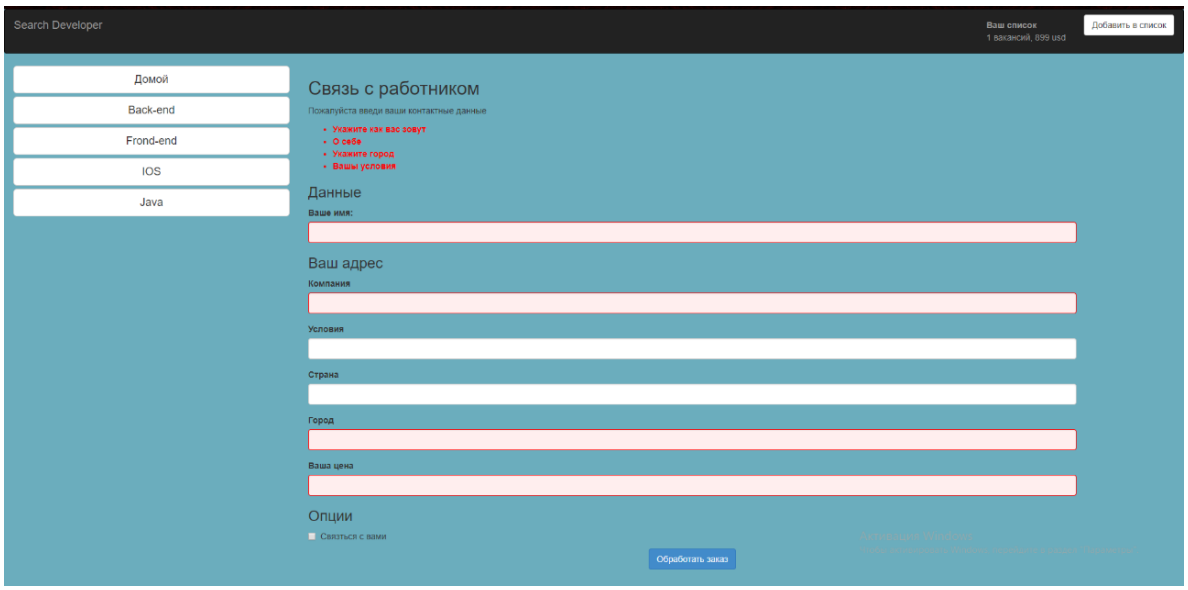

Рисунок 4.4 — Форма для звязку

Коли користувач переходить на дану форму йому потрібно заповнити форму для зв'язку з майбутнім робітником. Якщо користувач неправильно заповнив форму ті поля які були неправильно заповнені буде видно для корегування даних. Після заповнення даної форми як користувача так і розробника повідомлять по електронній пошті.

4.4 Інструкція адміністратора

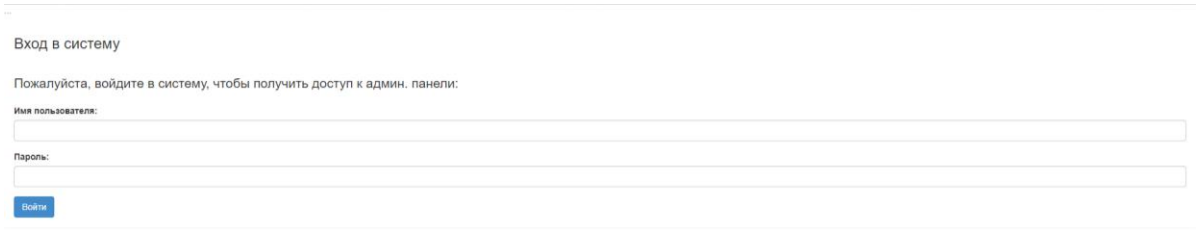

Рисунок 4.5 — Форма для авторизації

Для доступу до адміністративної панелі користувачеві потрібно авторизуватися і правильно ввести дані. Якщо дані були введені неправильно то користувачеві буде надано інформацію в якому з полів для вводу інформації була зроблена помилка.

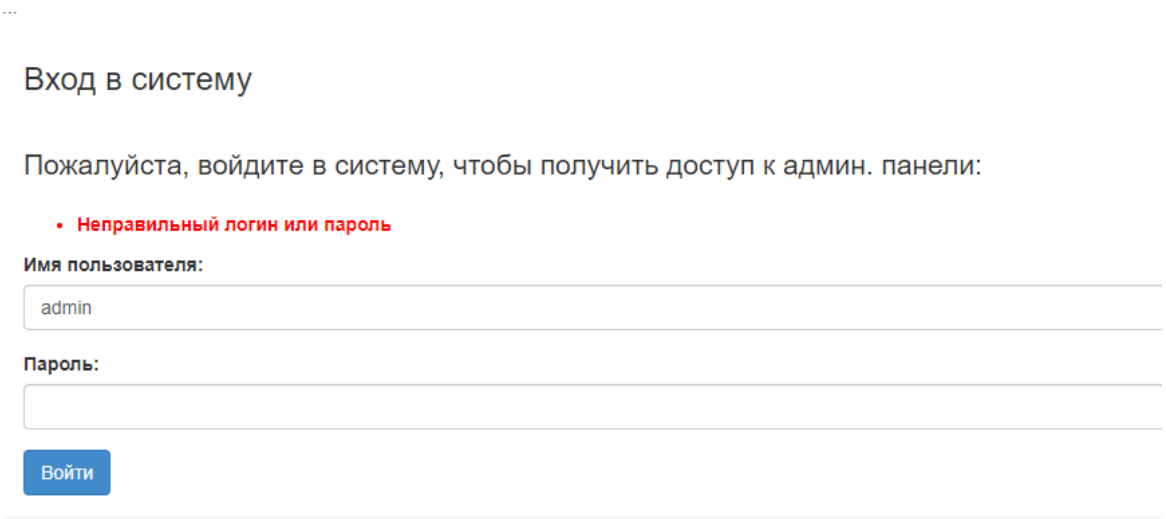

Рисунок 4.6 — Помилка при авторизації

4.4.1 Головна сторінка адміністративної панелі

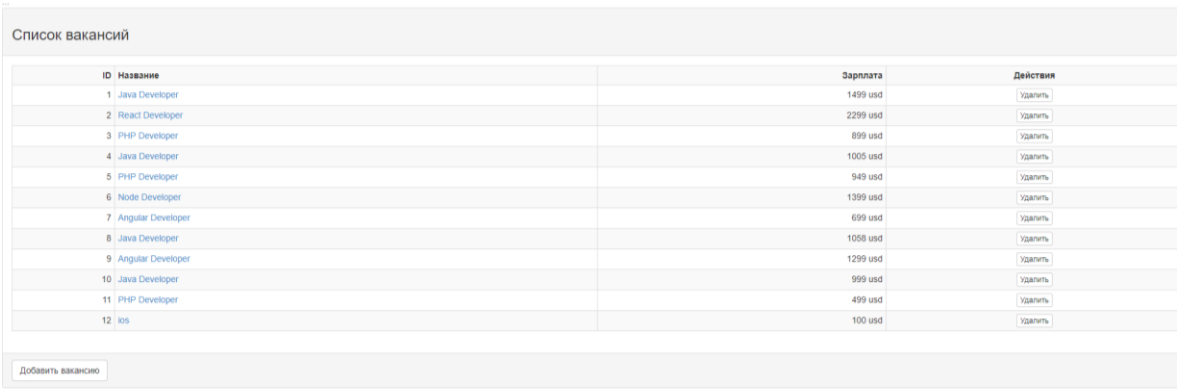

Рисунок 4.7 — Головна сторінка адміністративної панелі

Після проходження авторизації користувач попадає до головного меню адміністративної панелі.

На цій сторінці можливий процес додавання нових вакансій, видалення, редагування.

4.4.2 Редагування вакансій

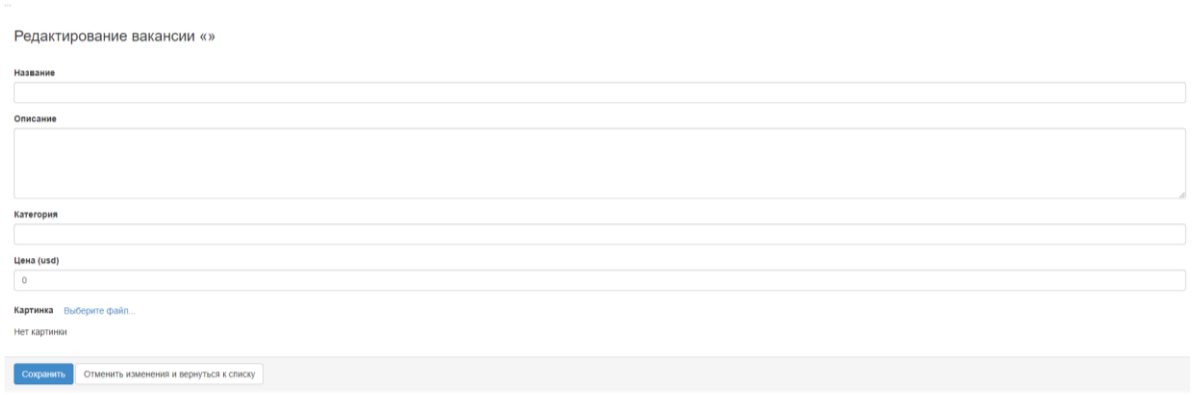

Рисунок 4.8 — Зміна вакансій

При натисненні на будь яку вакансію яка доступна на головній сторінці, користувач відкриває нову форму. На цій формі можна редагувати дані. Якщо вносити зміни і зберігати то користувачу повідомлять про те що дані збережені.

Рисунок 4.9 — Редагування вакансій

При натисненні на кнопку відмінити зміни відбудеться переадресація на головну сторінку.Також можна завантажувати картинки.

4.4.3 Додання вакансії

Рисунок 4.10 — Додання вакансії

При натисненні кнопки для добавлення нової вакансії перед користувачем буде відповідна форма де потрібно заповнити потрібно інформацію, а також пройти провірку на коректність заповнення даних.

4.4.4 Видалення вакансій

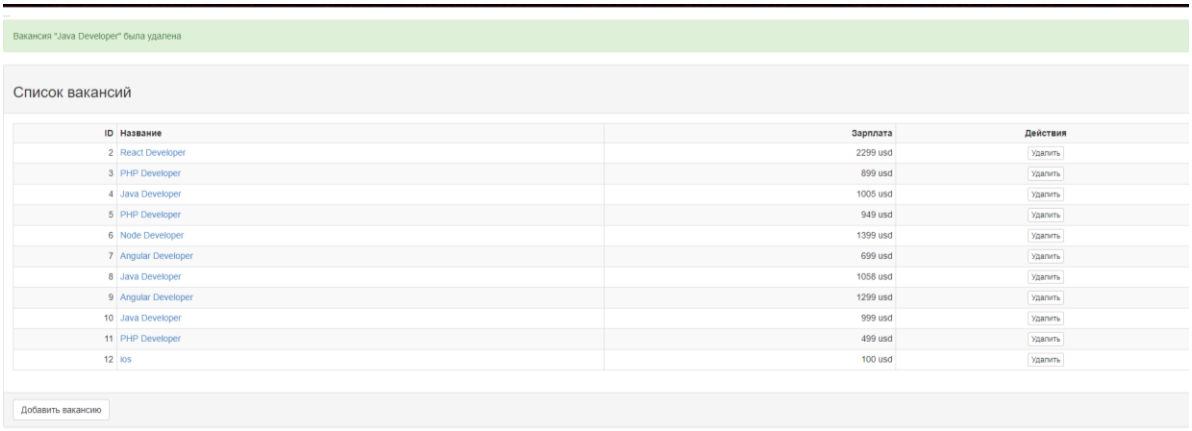

Рисунок 4.11 — Видалення вакансії

ВИСНОВКИ ПО РОЗДІЛУ

В даному розділі було описано дії які потрібно зробити для того, щоб розгорнути проект.Вказані послідовні дії які потрібно зробити для правильної роботи і запуску проекта. Описано інтерфейс всього проекту та всіх частин. Весь функціонал доступний як користувачеві так і адміністратору також було описано.

ВИСНОВКИ

У розділі “Аналіз вимог до програмного забезпечення” було описано та проаналізовано конкурентів і описана структура проекту, а також описано інтерфейс.

У розділі “Моделювання та конструювання програмного забезпечення” було розроблено архітектуру прогрманого забезпечення. І описано предметну область розробки.

У розділі “Аналіз якості та тестування програмного забезпечення” було розроблено план тестування комплексу задач та виконано приклад тестування. І зроблено обгрунтування тестів.

У розділі “Впровадження та супровід програмного забезпечення” було описано процес встановлення комплексу на апаратну платформу. Розроблено інструкції для користувача-програміста та для адміністратора. З детальною покроковою інтрукцією.

У рамках даного дипломного проєкту було застосовано набуті знання з розробки баз даних, архітектури програмного забезпечення, безпеки даних, об’єктно-орієнтованого програмування, створення зручного та функціонального інтерфейсу.

Розроблений комплекс задач є повноцінним програмним продуктом, готовим для використання, який легко може бути доповнений новим функціоналом та масштабований.

ПЕРЕЛІК ПОСИЛАНЬ

1. Triaservise http://triaservice.com.ua/notes/145-brauzer-google-chrome.html - офіційна сторінка у мережі Інтернет.
2. ItProger https://itproger.com - офіційна сторінка у мережі Інтернет.
3. Helm https://dotnet.today/ru - офіційна сторінка у мережі Інтернет.
4. dotNet https://github.com/kubesail/deploy-node-app - офіційна сторінка у мережі Інтернет.
5. METANIT https://metanit.com - офіційний опис стандарту.
6. Oracle https://www.oracle.com/ru– офіційний опис стандарту.
7. Javascript https://learn.javascript.ru- офіційна сторінка у мережі Інтернет.
8. jQuery https://learn.jquery.com/- офіційна сторінка у мережі Інтернет.
9. Microsoft https://docs.microsoft.com/en-us/nuget - офіційний опис стандарту.
10. Wikipedia https://ru.wikipedia.org/wiki/NuGet - офіційна сторінка у мережі Інтернет.
11. Zybin https://zyubin.ru - офіційна сторінка у мережі Інтернет.
12. Tproger https://tproger.ru/translations/bootstrap-short-intro/ - офіційна сторінка у мережі Інтернет.
13. Metanit https://metanit.com/sharp/entityframework/1.1.php- - офіційна сторінка у мережі Інтернет.
14. Wikipedia https://uk.wikipedia.org/wiki- - офіційна сторінка у мережі Інтернет.
15. ПроТестинг http://www.protesting.ru/testing/ - офіційна сторінка у мережі Інтернет.

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

Все эти листы не печатаются! Они нужны только для того, чтоб сделать правильное содержание

ДОДАТОК Б ОПИС ПРОГРАМИ

ДОДАТОК Е ГРАФІЧНИЙ МАТЕРІАЛ

ЛИСТ 1. СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАНЬ

ЛИСТ 2. СХЕМА СТРУКТУРНА СТАНІВ СИСТЕМИ

ЛИСТ 3. СХЕМА БАЗИ ДАНИХ