Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України»

Виконав: студент 3 курсу
групи 1ПІ-19мс2
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кубай М.О.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д.І.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН Крилик Л.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кубаю Максиму Олексійовичу

1. Тема роботи – Розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України.

Керівник роботи: Катєльніков Денис Іванович, к. т. н., доц. кафедри ПЗ, затверджений наказом вищого навчального закладу від 24 березня 2022 р. № 66.

2. Строк подання студентом роботи 13 червня 2022 р.

3. Вихідні дані до роботи: середовище розробки Visual Studio 2019, мова програмування C#, операційна система – Windows 10, інформація про заклади вищої освіти України.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; порівняльний аналіз аналогів; аналіз методів розв'язання поставленої задачі; розробка алгоритмів системи; проєктування нормалізованих відношень; розробка методу вибору спеціальностей; розробка веб-інтерфейсу користувача; варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу; розробка функцій виведення інформації; розробка модуля рекомендацій спеціальностей; розробка адмін-панелі; тестування системи; висновки; додатки.

5. Перелік графічного матеріалу: мета, об’єкт та предмет дослідження; задачі дослідження; наукова новизна; практичне значення; метод і модель роботи мимтеми; графічний інтерфейс інформаційної системи; ER-модель предметної області; діаграма функціональної залежності; тестування системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Катєльніков Д. І., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання _____25 березня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
	Аналіз стану систем для пошуку закладу освіти	26.03.2022 – 14.04.2022	Вик.
	Розробка структури інформаційної системи	14.04.2022 – 01.05.2022	Вик.
	Розробка інформаційної системи	2.05.2022 – 30.06.2022	Вик.
	Тестування інформаційної системи	1.06.2022 – 10.06.2022	Вик.

Студент _____ Кубай М.О.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Катєльніков Д.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 117 сторінок формату А4, на яких є 42 рисунки, 3 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі описано розробку інформаційної системи для вибору спеціальностей у закладах вищої освіти України. Сформульовано мету досліджень – підвищення рівня обгрунтованості та спрощення процесу вибору навчального закладу шляхом розробки та використання інформаційної системи для вибору спеціальностей в закладах вищої освіти України, що дозволить оптимізувати процес прийняття рішень.

Запропоновано модель вибору спеціальності з урахуванням балів ЗНО абітурієнта, що враховує оцінку з кожної дисципліни і на основі цих даних допомагає обрати перелік спеціальностей, які найкраще підійдуть абітурієнту. Запропоновано метод підбору на основі статистичного аналізу для більш точної рекомендації спеціальностей.

Розроблено алгоритми для підбору спеціальностей та закладів вищої освіти, а також систему з веб-інтерфейсом на їх основі.

Отримані в бакалаврській дипломній роботі результати можна використати в школах та закладах вищої освіти для спрощення рекомендацій абітурієнтам щодо вступу.

Програмний продукт було створено з використанням мови програмування C# та платформи ASP.NET MVC.

Ключові слова: статистичний аналіз, система відбору, абітурієнт.

ABSTRACT

The bachelor's thesis consists of 117 A4 pages, which contain 42 figures, 3 tables, the list of sources used contains 20 titles.

The bachelor's thesis describes the development of an information system for choosing specialties in higher education institutions of Ukraine. The purpose of the research is to increase the level of validity and simplify the process of choosing an educational institution by developing and using an information system for choosing specialties in higher education institutions of Ukraine, which will optimize the decision-making process.

The model of the choice of a specialty taking into account points of EIT of the entrant which considers an estimation from each discipline and on the basis of these data helps to choose the list of specialties which will best suit the entrant is offered. The method of selection on the basis of the statistical analysis for more exact recommendation of specialties is offered.

Algorithms for selection of specialties and institutions of higher education, as well as a system with a web interface based on them have been developed.

The results obtained in the bachelor's thesis can be used in schools and institutions of higher education to simplify the recommendations for applicants for admission.

The software product was created using the C # programming language and the ASP.NET MVC platform.

Key words: statistical analysis, selection system, entrant.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Більшість абітурієнтів, визначившись із майбутньою професією, стикається з однією не менш гострою проблемою – вибором вищого навчального закладу. На сьогодні в Україні існує більше 1000 закладів вищої освіти. Навіть використовуючи найновішу та найбільш повну довідникову інформацію, абітурієнту досить складно визначити переваги та недоліки певного закладу.

В умовах розвитку сучасного суспільства інформаційні технології глибоко проникають життя людей. Вони швидко перетворилися на життєво важливий стимул розвитку не тільки світової економіки, а й інших сфер людської діяльності. Зараз важко знайти сферу, в якій не використовуються інформаційні технології.

Інформаційна система, як система управління, тісно пов’язується, як з системами збереження та видачі інформації, так і з системами, що забезпечують обмін інформацією в процесі управління. Вона охоплює сукупність засобів та методів, що дозволяють користувачу збирати, зберігати, передавати і обробляти відібрану інформацію [1]. Інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існує потреба в управлінні. Місією інформаційної системи є обробка потрібної для організації інформації, потрібної для ефективного управління всіма її ресурсами, створення інформаційного та технічного середовища для управління діяльністю. Інформаційна система може існувати і без застосування комп’ютерної техніки – це питання економічної необхідності. В будь-якій інформаційній системі управління вирішуються задачі трьох типів:

– задачі оцінки ситуації (деколи їх називають задачами розпізнавання образів);
– задачі перетворення опису ситуації (розрахункові задачі, задачі моделювання);

– задачі прийняття рішень (в тому числі і оптимізаційні).

Інформаційні системи включають в себе: технічні засоби обробки даних, програмне забезпечення і відповідний персонал. Складові частини системи утворюють внутрішню інформаційну основу:

– засоби фіксації і збору інформації;

– засоби передачі відповідних даних та повідомлень;

– засоби збереження інформації;

– засоби аналізу, обробки і подання даних.

Велика кількість закладів вищої освіти і спеціальностей обумовлює складність для абітурієнтів з визначенням майбутнього фаху й вибором навчального закладу. Наявні інформаційні системи надають обмежений функціонал для обгрунтованого прийняття рішення щодо обрання навчального закладу для отримання освіти. Тому актуальною є розробка інформаційної системи для пошуку закладів освіти . У ній передбачено необхідні засоби для визначення і обробки даних, а також для управління ними при роботі з великими об'ємами інформації.

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

Мета та завдання дослідження.

Мета виконання бакалаврської дипломної роботи – підвищення рівня обгрунтованості та спрощення процесу вибору навчального закладу шляхом розробки та використання інформаційної системи для вибору спеціальностей в закладах вищої освіти України, що дозволить оптимізувати процес прийняття рішень.

Відповідно до поставленої мети в бакалаврській дипломній роботі потрібно вирішити такі **завдання**:

– розробити метод рекомендації спеціальностей;

– розробити модель системи підбору закладів освіти для вступу та рекомендованих спеціальностей на основі введених користувачем балів ЗНО;

– розробити інтерфейс та дизайн інформаційної системи, що буде зручним у використанні непідготовленим користувачем;

– програмно реалізувати спеціалізовану інформаційну систему для вибору спеціальностей у закладах вищої освіти;

– провести тестування програмного продукту.

Об’єкт дослідження – процес розробки інформаційної системи для вибору спеціальностей у закладах вищої освіти.

Предмет дослідження – методи та засоби розробки інформаційних систем та веб-інтерфейсів.

Методи дослідження. У процесі досліджень використовувались:

- методи статистичного аналізу даних для обробки інформаційних масивів;
- методи прогнозування для системи прийняття рішень;
- методи теорії алгоритмів для побудови алгоритмів роботи програми;
- методи створення графічних зображень для візуального оформлення інтерфейсу;
- методи аналізу та синтезу для побудови ключових моделей моніторингової системи;
- методи тестування для підтвердження працездатності системи та її відповідності заданим вимогам.

Наукова новизна отриманих результатів.

- Подальшого розвитку отримав метод статистичного аналізу, який, на відміну від існуючих, реалізує багатокритеріальний аналіз спеціальностей за вказаними параметрами і пріоритетами користувача, що допомагає абітурієнтам зорієнтуватися і визначитися з вибором спеціальності і закладу вищої освіти для вступу.

- Подальшого розвитку отримала модель відбору спеціальностей та закладів освіти на основі введених користувачем балів ЗНО, що дозволить визначити найбільш перспективний напрям у розвитку абітурієнта.

Практична цінність отриманих результатів. Практична цінність полягає у реалізації інформаційної системи, що може використовуватись абітурієнтами в процесі вибору спеціальнсті та закладу вищої освіти для вступу.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій роботі [2], опублікованій у співавторстві, автору належать метод статистичного аналізу даних, модель відбору рекомендованих спеціальностей на основі введених даних, веб-інтерфейс системи.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на XLIX науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2020).

Публікації. Основні результати дослідження опубліковані в науковій роботі [2] – в тезах доповіді на XLIX науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2020).

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем для пошуку закладу освіти

У сучасних умовах розвитку в Україні ринкових відносин, ринку праці, нових форм власності, безробіття і конкуренції, своєчасний і обґрунтований вибір молоддю професії, навчального закладу, набуває все більшого економічного і соціального значення.

Як відомо, успішний вибір професії значною мірою визначає результати праці, рівень добробуту, сприяє реалізації особистісного потенціалу, а держава отримує максимальну віддачу. Якщо вибір професії молодою людиною зроблено необґрунтовано, помилково, то це може вплинути на якість його навчання у вищому навчальному закладі і значно знизити його конкурентоспроможність на ринку праці.

У цій ситуації актуально і гостро постає питання конкретної допомоги молоді в професійному самовизначенні засобами професійної орієнтації. Сьогодні формування психологічної готовності випускників школи до усвідомленого вибору професії, до праці, стало не лише гострою необхідністю, але й багатогранним і складним процесом.

Інформаційна система призначена для абітурієнтів, які хочуть отримати вищу освіту. Розроблена система також буде корисною для працівників університетів та коледжів: директорів, методистів, викладачів. Використання інформаційної системи дасть можливість у зручній та доступній формі переглянути інформацію про навчальний заклад, обрати спеціальність, переглянути статистику за різними параметрами, а також знайти заклад для вступу на основі власних балів ЗНО.

1.2 Порівняльний аналіз аналогів

На сьогоднішній день існує декілька сервісів для пошуку закладів вищої

освіти для вступу, з них найвідомішими є:

– ЄДЕБО;
– Osvita.ua;
– Education.ua.

ЄДЕБО (Єдина Державна Електронна База з питань Освіти) – електронний сервіс для перегляду інофрмації про заклади освіти з можливістю електронного подання документів на вступ (рис.1.1) [3].

Основні переваги:

– актуальність інформації про заклади освіти;
– можливість експорту даних в Excel;
– зручний пошук даних;
– можливість подати електронну заяву на вступ до закладу вищої освіти;
– містить найбільш актуальну інформацію про вступ поточного року.

Рисунок 1.1 – Інтерфейс ЄДЕБО

Недоліки:

– початкова складність в користуванні системою абітурієнтами;
– відсутність статистики по закладах вищої освіти за певний період;

– відсутність архівування даних (доступна інформація лише за поточний рік.

Osvita.ua – інформаційний ресурс, який публікує актуальні новини з питань освіти та має влаштований сервіс для вибору навчального закладу для вступу (рис.1.2) [4].

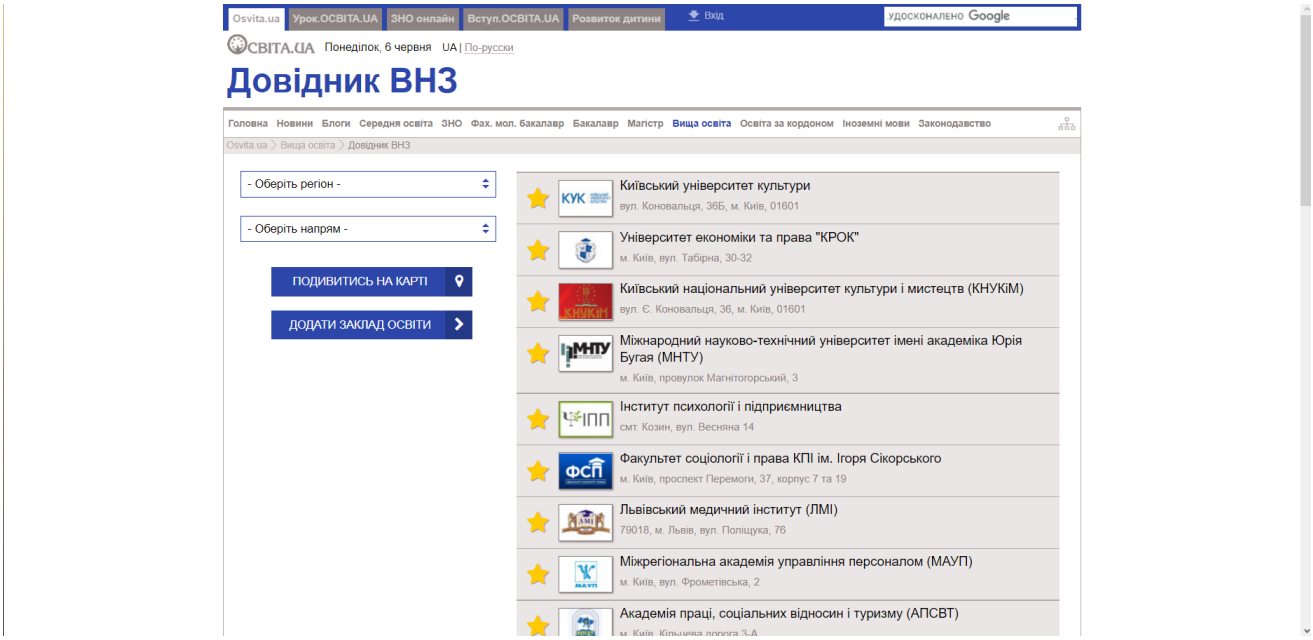


Рисунок 1.2 – Сайт Osvita.ua

Переваги:

– простий та зручний інтерфейс;
– можливість переглянути заклади освіти на карті;
– наявність актуальної інформації;
– система відгуків.

Недоліки:

– відсутність статистичної інформації;
– відсутність інформації про конкурсні пропозиції закладів освіти.

Education.ua – інформаційний ресурс, що дозволяє знайти навчальний заклад за різними параметрами, а також переглянути відгуки про заклади освіти (рис.1.3) [5].

Переваги:

– зручний і зрозумілий інтерфейс;
– велика кількість відгуків про заклади освіти;
– наявна інформація не лише про заклади вищої освіти, але й про школи і дитсадки;
– розміщена контактна інформація закладів в соцмережах.

Недоліки:

– відсутність статистичної інформації;
– відсутність інформації про конкурсні пропозиції.

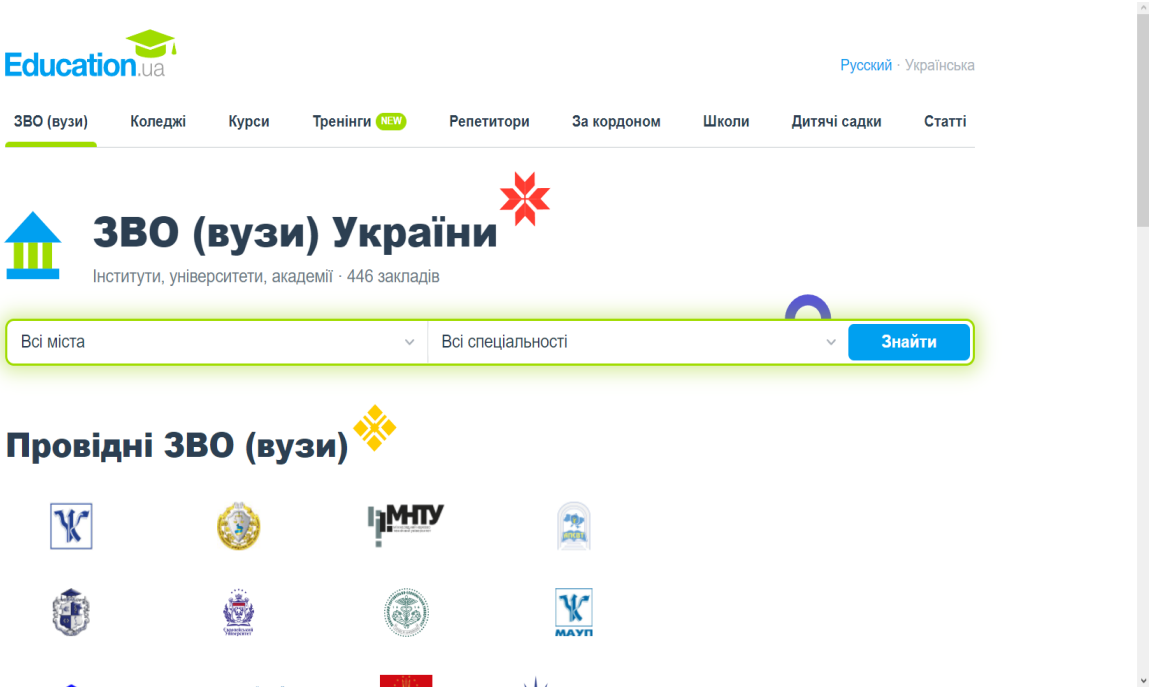


Рисунок 1.3 – Сайт Education.ua

Порівняємо всі системи між собою, результати зведемо в таблицю 1.1.

Таблиця 1.1 – Порівняльна характеристика сервісів для пошуку закладів освіти

Критерії	ЄДЕБО	Osvita.ua	Education.ua	Власна система
Зручний інтерфейс	-	+	+	+
Інформація про конкурсні пропозиції	+	-	-	+
Наявність системи відгуків	-	+	+	-
Статистична інформація про заклади	-	-	-	+
Актуальність та оновлення інформації	+	+	+	+
Рекомендації щодо вибору спеціальності	-	-	-	+
Загальна оцінка	33%	50%	50%	83%

Відповідно до порівняльних характеристик розробка власної системи для пошуку закладів освіти є доцільною. Такий програмний продукт зможе покрити недоліки існуючих рішень та забезпечити новий функціонал вибору закладів вищої освіти.

1.3 Аналіз методів розв’язання задачі

Для вирішення задачі вибору спеціальностей та закладів вищої освіти існує декілька методів. Найбільш швидким з них є розробка окремого модуля, який дозволить обрати рекомендовану спеціальність за даними результатів ЗНО. Цей спосіб є достатньо ефективним, але потрібно докласти додаткових зусиль з боку розробників веб-інтерфейсу. Недоліком є те, що значна частина роботи буде покладена на розробників веб-інтерфейсу, що буде інтегрувати модуль. Тому цей спосіб не бажано реалізовувати.

Кращим варіантом є розробка власної інформаційної системи для пошуку закладів освіти та рекомендацій щодо вибору спеціальності. Цей спосіб дозволить одразу розробити модуль та інтерфейс до нього, що значно спрощує процес розробки. Однак цей метод є складнішим через застосування різних підходів одночасно.

Врахувавши переваги і недоліки кожного методу, було прийнято рішення використати метод розробки власної інформаційної системи з влаштованим модулем відбору закладів освіти та рекомендацій щодо вибору спеціальності, що дозволить більш повноцінно та ефективно розробити програмний продукт.

1.4 Постановка задач

Завданням бакалаврської дипломної роботи є розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України.

Система буде створюватись з використанням мови програмування C# та платформи ASP.NET MVC.

Для розробки інформаційної системи потрібно розв’язати такі задачі:

– розробити метод рекомендації спеціальностей;

– розробити модель системи підбору закладів освіти для вступу та рекомендованих спеціальностей на основі введених користувачем балів ЗНО;

– розробити інтерфейс та дизайн інформаційної системи, що буде зручним у використанні непідготовленим користувачем;

– програмно реалізувати спеціалізовану інформаційну систему для вибору спеціальностей у закладах вищої освіти;

– провести тестування програмного продукту.

1.5 Висновки

У першому розділі було розглянуто стан сучасних інформаційних систем та актуальність вибору абітурієнтами закладу освіти для вступу. Було проведено

порівняння з ЄДЕБО, Osvita.ua та Education.ua, у результаті якого було визначено, що існуючі сервіси не надають статистичної інформації та рекомендацій стосовно вибору закладів освіти. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власної інформаційної системи.

2 РОЗРОБКА МЕТОДУ ВИБОРУ СПЕЦІАЛЬНОСТІ, МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка методу та алгоритмів роботи системи

Для роботи інформаційної системи потрібно розробити метод пошуку спеціальностей, який акумулює послідовність дій:

1. Користувач отримує список усіх можливих представлень даних (а саме список усіх закладів освіти та конкурсних пропозицій з можливістю фільтрації за окремим параметрами).
2. Далі користувач може обрати необхідне подання або відредагувати дані за необхідності.
3. Обравши перегляд даних, користувач обирає, в якому саме вигляді йому потрібно знайти дані: переглянути весь список закладів освіти чи знайти конкретний заклад, використовуючи фільтри, чи переглянути конкурсні пропозиції.
4. Якщо користувач не знайшов усі дані або не може визначитися з вибором закладу – він може перейти до модуля рекомендації спеціальностей, внести туди всі результати ЗНО, після чого отримає результат у вигляді списку рекомендованих спеціальностей.
5. Якщо результат його влаштовує, спеціальності його задовільняють, то можна розглядати конкурсні пропозиції щодо обраної спеціальності, після визначитися з кінцевим вибором.

Алгоритм роботи методу вибору спеціальностей наведено на рисунку 2.1. Розглянувши алгоритм, можна зробити висновок, що запропонований підхід до пошуку закладу та спеціальності до вступу дозволить швидко і просто знайти найбільш задовільний до пошукових вимог заклад освіти.

Рисунок 2.1 – Блок-схема загального алгоритму роботи системи

Для розуміння роботи методу відбору закладів освіти та спеціальностей було розроблено окремий алгоритм (рисунок 2.2). Згідно нього користувач спочатку вводить свої результати ЗНО. Система зчитує ці дані, аналізує їх, обраховує загальний та середній бал, шукає по яких дисциплінах були найвищі оцінки. Після чого результат виводиться на екран у вигляді переліку

рекомендованих спеціальностей та рекомендує переглянути конкурсні пропозиції за цією спеціальністю.

Рисунок 2.2 – Блок-схема алгоритму методу відбору закладів освіти та спеціальностей

2.2 Розробка моделі вибору спеціальностей

Для рекомендації вибору спеціальності було вирішено розробити спеціальну модель (рисунок 2.3). Вона передбачає, що користувач вводить свої оцінки з ЗНО, після чого система зчитує ці дані та виявляє, які саме дисципліни були введені, на яких з них оцінки найвищі, на яких – найнижчі. Далі на основі цих даних система пропонує одну чи декілька рекомендованих для користувача спеціальностей.

Рисунок 2.3 – Модель відбору рекомендованих спеціальностей на основі введених даних

2.3 Розробка веб-інтерфейсу користувача

Основними критеріями при розробці інтерфейсу були простота та зручність. Для зручності в навігації сайту в шапку були додані кнопки «Категорії» для вибору необхідних даних для перегляду та «Адмін-панель» для редагування даних (рисунок 2.4) [6].

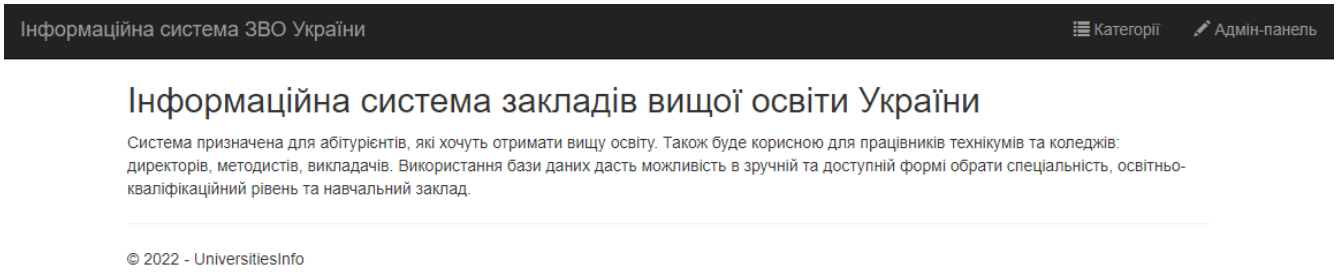


Рисунок 2.4 – Головна сторінка

Для зручності перегляду всі дані були розбиті на категорії (рисунок 2.5) [7]. Таким чином користувачеві не потрібно довго розшукувати необхідну йому інформацію. Аналогічно були розділення представлення даних кожної категорії (рисунок 2.6).

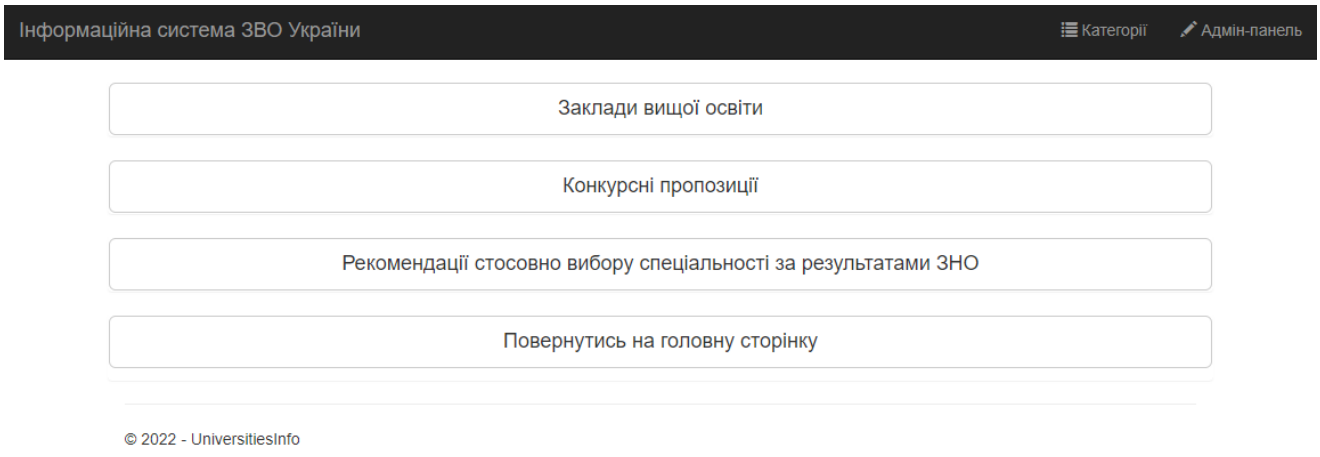


Рисунок 2.5 – Список категорій

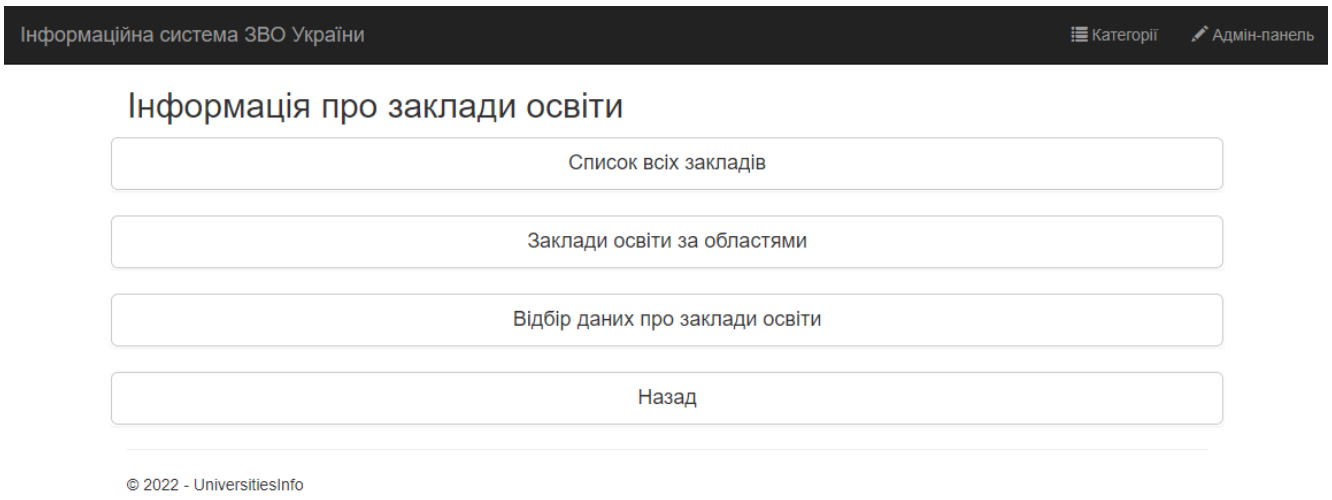


Рисунок 2.6 – Сторінка подання даних категорії «Заклади вищої освіти»

2.4 Висновки

У другому розділі було розроблено основний алгоритм роботи інформаційної системи та модель вибору рекомендованих спеціальностей. Також було розроблено метод відбору закладів освіти та спеціальностей, що дозволяє виводити список рекомендованих спеціальностей для вступу.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Для розробки програмного додатку необхідно використати об’єктно-орієнтовану парадигму програмування. Дану парадигму підтримують такі мови як C++ , C#, Java та Python.

C++ – компільована, статично типізована мова програмування, яка поєднує властивості як високорівневих та і низькорівневих мов програмування. Код написаний на C++ компілюється напряму в машинний код, що робить дану мову одною з найшвидших в світі. C++ повністю сумісний з мовою C. Мова C++ широко використовується для розробки прикладних програм, операційних систем, драйверів пристроїв, відеоігор тощо [8].

C# – об’єктно орієнтована мова, розроблена компанією Microsoft спеціально для використання можливостей .NET Framework. C# відноситься до мов програмування з C-подібним синтаксисом. Мова має строгу статичну типізацію, підтримує перевантаження операторів, анонімні функції, узагальнені типи і методи, вказівники на члени функції класів. C# не підтримує множинне спадкування класів, на відміну від C++, але підтримує множинне спадкування інтерфейсів [9].

Java – об’єктно-орієнтована мова програмування, одна з найпоширеніших мов програмування в світі. Основною особливістю є те, що її програмний код спочатку транслюється в спеціальний байт-код, незалежний від платформи, а згодом байт-код виконується віртуальною машиною JVM (Java Virtual Machine). Це дозволяє код написаний на Java запускати майже на всіх відомих системах: Windows, Mac, Android тощо. Але дана мова програмування не є компільованою, як C++, саме через це код написаний на Java працює повільніше [10].

Python – високорівнева мова програмування, як призначена для створення додатків різного типу: веб-додатки, відеоігри, настільні програми, утиліти для

роботи з базами даних тощо. Велике розповсюдження Python отримав в області машинного навчання та штучного інтелекту. Код написаний на Python інтерпретується байт-код який переводиться віртуальною машиною в набір команд, які виконуються операційною системою [11].

Проаналізувавши мови програмування, було вирішено використовувати мову програмування C#. Для розробки веб-інфтерфейсу вирішено використати платформу ASP.NET MVC. We фреймоворк для розробки веб-проектів, які народився в компанії Microsoft, але надалі перейшов в категорію проектів із відкритим вихідним кодом [12].

Розробка на ASP.NET MVC зазвичай ведеться з використанням мови C#, що дає можливість використовувати всю міць .NET Framework, зокрема асинхронність, багатопотоковість, планувальники тощо, а також використовувати величезну кількість готових бібліотек і рішень із публічних репозиторіїв, наприклад, Nuget тощо.

Інфраструктура ASP.NET MVC генерує ясний код і відповідає стандартам розмітки, в результаті цього сторінки сайту виходять компактними та швидкісними.

3.2 Розробка функцій виведення інформації

Основною задачею інформаційної системи є надання інформації про заклади освіти та конкурсні пропозиції. Для представлення даних було обрано концепцію MVC (model - view – controller) [13].

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

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

Модель (model) є класом, що описує логіку використовуваних даних.

За виведення даних відповідають такі контролери:

– Home (головна сторінка та перелік сутностей) (рисунок 3.1);
– Admin (адмін-панель) (рисунок 3.2);
– Navigation (допоміжна панель дня фільтрації даних) (рисунок 3.3);
– Universities (сторінки з даними про заклади освіти) (рисунок 3.4);
– Offers (інформація про конкурсні пропозиції) (рисунок 3.5).

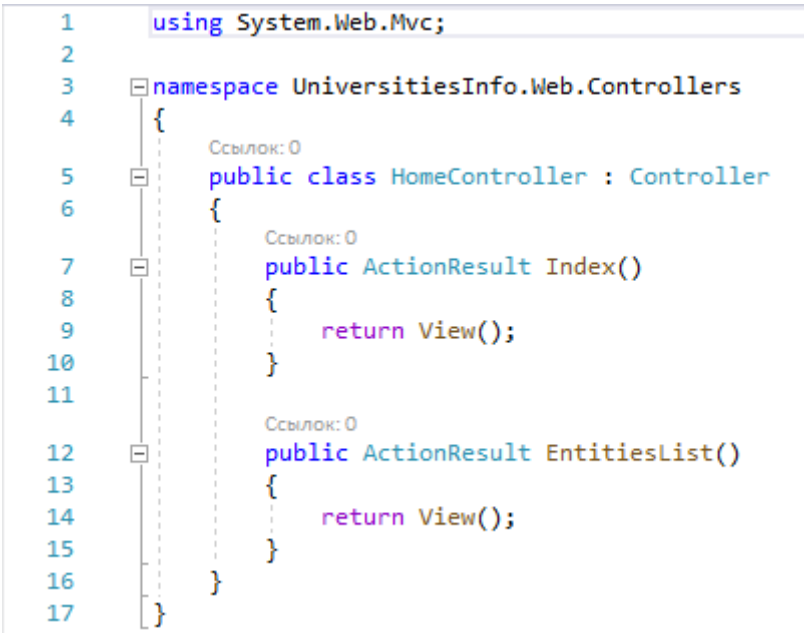


Рисунок 3.1 – Код класу HomeController

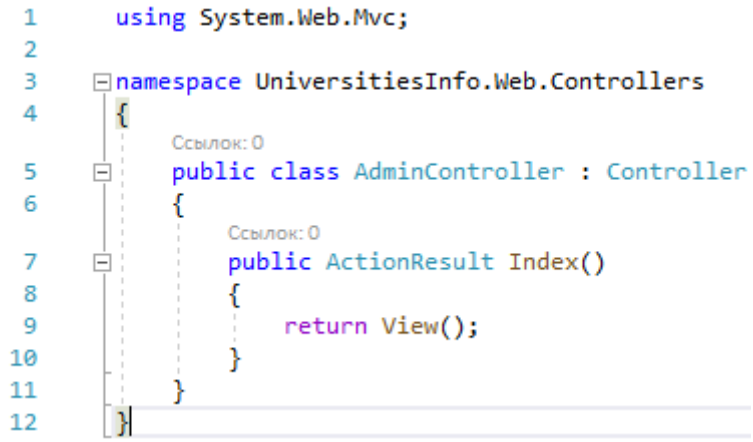


Рисунок 3.3 – Код класу AdminController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    Ссылок: 3
    public class NavigationController : Controller
    {
        Ссылок: 2
        public IEnumerable<University> Universities { get; set; }

        Ссылок: 0
        public NavigationController()
        {
            Universities = DataContext.universities;
        }

        public const string ALL_CATEGORIES = "...";

        [ChildActionOnly]
        Ссылок: 0
        public PartialViewResult UniversitiesByAreasMenu(
        string categoryName = ALL_CATEGORIES)
        {
            ViewBag.SelectedCategoryName = categoryName;
            List<string> categoryNames = new List<string>();
            categoryNames.Add(ALL_CATEGORIES);
            categoryNames.AddRange(Universities.Select(e => e.Area).Distinct().OrderBy(e => e));
            return PartialView(categoryNames);
        }
    }
}
```

Рисунок 3.3 – Код класу NavigationController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesController : Controller
    {
        private IEnumerable<University> objects;
        private IEnumerable<UniversityViewModel> viewModelObjects;


        public IEnumerable<University> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects.Select(e => (UniversityViewModel)e)
                    .OrderBy(e => e.UniversityName);
            }
        }

        Ссылок: 0
        public UniversitiesController()
        {
            Objects = DataContext.universities;
        }

        Ссылок: 0
        public ActionResult Index()
        {
            return View();
        }

        Ссылок: 0
        public ViewResult UniversitiesByAreasInfo(
        string categoryName = NavigationController.ALL_CATEGORIES)
        {
            IEnumerable<University> models = Objects.OrderBy(e => e.UniversityName);
            if (!string.IsNullOrEmpty(categoryName) &&
                categoryName != NavigationController.ALL_CATEGORIES)
            {
                models = models
                    .Where(e => e.Area == categoryName);
            }
            ViewBag.SelectedCategoryName = categoryName;
            return View(models);
        }
```

Рисунок 3.4 – Частина коду класу UniversitiesController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    ссылка: 1
    public class OffersController : Controller
    {
        Ссылок: 0
        public ActionResult Index()
        {
            return View();
        }

        private IEnumerable<Offer> objects;
        private IEnumerable<OfferViewModel> viewModelObjects;

        ссылка: 1
        public IEnumerable<Offer> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects.Select(e => (OfferViewModel)e)
                    .OrderBy(e => e.UniversityName);
            }
        }

        Ссылок: 0
        public OffersController()
        {
            Objects = DataContext.offers;
        }

        Ссылок: 0
        public ActionResult Selection()
        {
            return View(viewModelObjects);
        }

        Ссылок: 0
        public PartialViewResult _SelectData(string selUniversityName, string selSpecialityCode,
            int? PlacesFrom, int? PlacesTo, string selEducationLevel)
        {
            var model = viewModelObjects;
            if (!string.IsNullOrWhiteSpace(selUniversityName))
                model = model.Where(e => e.UniversityName.ToLower()
                    .StartsWith(selUniversityName.ToLower()));
            if (!string.IsNullOrWhiteSpace(selSpecialityCode))
                model = model.Where(e => e.SpecialityCode.ToLower()
```

Рисунок 3.5 – Частина коду класу Offers Controller

Для відображення даних використовуються моделі UniversityViewModel (інформація про заклади освіти) (рисунок 3.6) та OfferViewModel (дані про конкурсні пропозиції) (рисунок 3.7).

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
    public class UniversityViewModel
    {
        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        public string UniversityName { get; set; }

        [Display(Name = "Ідентифікаційний код")]
        public string Code { get; set; }

        [Display(Name = "Тип закладу")]
        public string Type { get; set; }

        [Display(Name = "Область")]
        public string Area { get; set; }

        [Display(Name = "Населений пункт")]
        public string Settlement { get; set; }

        [Display(Name = "Адреса")]
        public string Address { get; set; }

        [Display(Name = "Телефон")]
        public string Phone { get; set; }

        [Display(Name = "Електронна адреса")]
        public string Email { get; set; }

        [Display(Name = "Веб-сайт")]
        ссылка: 1
        public string Website { get; set; }

        public static explicit operator UniversityViewModel(University obj)
        {
            return new UniversityViewModel()
            {
                Id = obj.Id,
                UniversityName = obj.UniversityName,
                Code = obj.Code,
                Type = obj.Type,
                Area = obj.Area,
                Settlement = obj.Settlement,
```

Рисунок 3.6 – Частина коду класу UniversityViewModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
    public class OfferViewModel
    {
        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        public string UniversityName { get; set; }

        [Display(Name = "Код спеціальності")]
        public string SpecialityCode { get; set; }

        [Display(Name = "Кількість місць")]
        public int NumberofPlaces { get; set; }

        [Display(Name = "Освітньо-кваліфікаційний рівень")]
        public string EducationLevel { get; set; }

        public static explicit operator OfferViewModel(Offer obj)
        {
            return new OfferViewModel()
            {
                Id = obj.Id,
                UniversityName = obj.UniversityName,
                SpecialityCode = obj.SpecialityCode,
                NumberofPlaces = obj.NumberofPlaces,
                EducationLevel = obj.EducationLevel,
            };
        }
    }
}
```

Рисунок 3.7 – Код класу OfferViewModel

3.3 Розробка модуля рекомендацій спеціальностей

Однією з головних задач системи є рекомендації щодо вибору спеціальності. Завдяки йому користувачі зможуть отримати список спеціальностей, за якими їм краще буде навчатись, на основі балів ЗНО. На рисунку 3.8 наведено код подання модуля рекомендації спеціальностей.

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))

{
    <h2>Введіть оцінки з ЗНО для рекомендації спеціальностей</h2>
    <p>
        Математика<br />
        <input type="text" name="math" />
    </p>
    <p>
        Фізика<br />
        <input type="text" name="physics" />
    </p>
    <p>
        Українська мова<br />
        <input type="text" name="ukrainian" />
    </p>
    <p>
        Українська література<br />
        <input type="text" name="ukrliterature" />
    </p>
    <p>
        Історія України<br />
        <input type="text" name="historyofUkraine" />
    </p>
    <p>
        Біологія<br />
        <input type="text" name="biology" />
    </p>
    <p>
        Географія<br />
        <input type="text" name="geography" />
    </p>
    <p>
        Хімія<br />
        <input type="text" name="chemistry" />
    </p>
    <p>
        Англійська мова<br />
        <input type="text" name="english" />
    </p>
    <p>
        Німецька мова<br />
        <input type="text" name="deutsch" />
    </p>
    <p>
        Іспанська мова<br />
        <input type="text" name="spanish" />
    </p>
```

Рисунок 3.8 – Код подання модуля рекомендації спеціальностей

3.4 Розробка адмін-панелі

Адмін-панель дозволяє додавати, редагувати або видаляти дані з системи. Аналогічно підсистемі відображення даних, використовується модель MVC [14]. За редагування даних відповідають контролери UniversitiesCrud (рисунок 3.9) та OffersCrud (рисунок 3.10), і моделі UniversityEditingModel (рисунок 3.11) та OfferEditingModel (рисунок 3.12). Моделі, в свою чергу, місять перевірку на введені користувачем дані, щоб запобігти помилкових записів або помилок в роботі системи [15].

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesCrudController : Controller
    {
        private IList<University> objects;

        private IEnumerable<UniversityViewModel> viewModelObjects { get; set; }

        private IEnumerable<UniversityEditingModel> editingModelObjects { get; set; }


        public IList<University> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects
                    .Select(e => (UniversityViewModel)e).OrderBy(e => e.UniversityName);
                editingModelObjects = objects
                    .Select(e => (UniversityEditingModel)e).OrderBy(e => e.UniversityName);
            }
        }


        public IEnumerable<University> Areas { get; set; }


        public UniversitiesCrudController()
        {
            Objects = DataContext.universities;
            Areas = DataContext.universities;
        }


        public ActionResult Index()
        {
            return View(viewModelObjects);
        }


        public ViewResult Create()
        {
            ViewBag.Area = CreateAreasSelectList();
            return View(new UniversityEditingModel());
        }

        [HttpPost]
```

Рисунок 3.9 – Частина код класу UniversitiesCrudController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{

    public class OffersCrudController : Controller
    {
        private IList<Offer> objects;

        private IEnumerable<OfferViewModel> viewModelObjects { get; set; }

        private IEnumerable<OfferEditingModel> editingModelObjects { get; set; }


        public IList<Offer> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects
                    .Select(e => (OfferViewModel)e).OrderBy(e => e.UniversityName);
                editingModelObjects = objects
                    .Select(e => (OfferEditingModel)e).OrderBy(e => e.UniversityName);
            }
        }


        public OffersCrudController()
        {
            Objects = DataContext.offers;
        }


        public ActionResult Index()
        {
            return View(viewModelObjects);
        }

        public ViewResult Create()
        {
            return View(new OfferEditingModel());
        }

        [HttpPost]
        public ActionResult Create(OfferEditingModel model)
        {
            if (!ModelState.IsValid)
            {
```

Рисунок 3.10 – Частина коду класу OffersCrudController

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
    public class UniversityEditingModel
    {
        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Назва закладу\'")]
        [StringLength(512, MinimumLength = 2,
             ErrorMessage = "Назва закладу має містити"
             + "від 2 до 512 символів")]
        public string UniversityName { get; set; }

        [Display(Name = "Ідентифікаційний код")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Ідентифікаційний код\'")]
        [RegularExpression(@"[1-99999999]{1}", ErrorMessage =
            "Потрібно ввести число з 8-ми цифр")]
        public string Code { get; set; }

        [Display(Name = "Тип закладу")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Тип закладу\'")]
        [StringLength(128, MinimumLength = 6,
             ErrorMessage = "Тип закладу має містити"
             + "від 6 до 128 символів")]
        public string Type { get; set; }

        [Display(Name = "Область")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Область\'")]
        [StringLength(17, MinimumLength = 4,
             ErrorMessage = "Область має містити"
             + "від 4 до 17 символів")]
        public string Area { get; set; }

        [Display(Name = "Населений пункт")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Населений пункт\'")]
        [StringLength(128, MinimumLength = 2,
             ErrorMessage = "Населений пункт має містити"
             + "від 2 до 128 символів")]
        public string Settlement { get; set; }

        [Display(Name = "Адреса")]
```

Рисунок 3.11 – Частина коду класу UniversityEditingModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{

    public class OfferEditingModel
    {

        public int Id { get; set; }

        [Display(Name = "Назва закладу")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Назва закладу\'")]

        public string UniversityName { get; set; }

        [Display(Name = "Код спеціальності")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Код спеціальності\'")]
        [RegularExpression(@"[0-9]{3}", ErrorMessage =
            "Код спеціальності має складатись з трьох цифр")]

        public string SpecialityCode { get; set; }

        [Display(Name = "Кількість місць")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Кількість місць\'")]
        [Range(1, 50000, ErrorMessage = "Кількість місць"
            + "має бути в межах від 1 до 50000")]

        public int NumberofPlaces { get; set; }

        [Display(Name = "Освітньо-кваліфікаційний рівень")]
        [Required(ErrorMessage =
             "Потрібно заповнити поле \'Освітньо-кваліфікаційний рівень\'")]

        public string EducationLevel { get; set; }

        public static explicit operator OfferEditingModel(Offer obj)
        {
            return new OfferEditingModel()
            {
                Id = obj.Id,
                UniversityName = obj.UniversityName,
                SpecialityCode = obj.SpecialityCode,
                NumberofPlaces = obj.NumberofPlaces,
                EducationLevel = obj.EducationLevel
            };
        }

        public static explicit operator Offer(OfferEditingModel obj)
        {
            return new Offer()
            {
                Id = obj.Id,
```

Рисунок 3.12 – Частина коду класу OfferEditingModel

3.5 Розробка представлень даних

У шаблоні MVC представлення відповідає за відображення даних програми та взаємодію з користувачем. Подання це HTML-шаблон з вбудованою Razor розміткою. Razor розмітка – це код, що взаємодіє з розміткою HTML для створення веб-сторінки, що надсилається клієнту.

Макети дозволяють створювати однакові розділи веб-сторінок та скоротити повтори у коді. Вони часто містять верхній та нижній колонтитули, а також елементи навігації та меню. Колонтитули зазвичай містять стандартну розмітку для багатьох елементів метаданих та посилання на ресурси скриптів та стилів. Макети дозволяють уникнути використання цієї стандартної розмітки у представленнях.

На рисунку 3.13 наведено код сторінки макету _Layout.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - UniversitiesInfo</title>
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
        <a class="navbar-brand hidden-xs hidden-sm" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-sm" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-xs" href="/">Інформаційна система ЗВО України</a>
        <ul class="nav navbar-nav navbar-right">
            <li>
                <a href="/Home/EntitiesList">
                    <span class="glyphicon glyphicon-list"></span>
                    Категорії
                </a>
            </li>
            <li>
                <a href="/Admin/Index">
                    <span class="glyphicon glyphicon-pencil"></span>
                    Адмін-панель
                </a>
            </li>
        </ul>
    </div>

    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - UniversitiesInfo</p>
            @RenderSection("Footer", false)
        </footer>
    </div>

    <script src="~/Scripts/jquery-1.10.2.min.js"></script>
    <script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>
    <script src="~/Scripts/bootstrap.min.js"></script>
</body>
</html>
```

Рисунок 3.13 – Код сторінки макету _Layout

Представлення EntitiesList контролеру Home містить перелік категорій даних та посилання на перехід до модуля рекомендацій спеціальностей (рисунок 3.14).

```
<h2></h2>
<div class="row panel">
    @Html.ActionLink("Заклади вищої освіти",
                     "Index", "Universities", null,
                     new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Конкурсні пропозиції",
                     "Index", "Offers", null,
                     new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Рекомендації стосовно вибору спеціальності за результатами ЗНО",
                     "Recommendation", "Universities", null,
                     new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    <p>
        @Html.ActionLink("Повернутись на головну сторінку", "Index", null, new { @class = "btn btn-block btn-default btn-lg" })
    </p>
</div>
```

Рисунок 3.14 – Код представлення EntitiesList контролеру Home

Представлення Index контролеру Universities містить перелік представлень даних про заклади освіти (рисунок 3.15).

```
1
2   @{
3       ViewBag.Title = "Index";
4   }
5
6   <h2>Інформація про заклади освіти</h2>
7
8
9   <div class="row panel">
10      @Html.ActionLink("Список всіх закладів",
11                       "List", "Universities", null,
12                       new { @class = "btn btn-block btn-default btn-lg" })
13  </div>
14  <div class="row panel">
15      @Html.ActionLink("Заклади освіти за областями",
16                       "UniversitiesByAreasInfo", "Universities", null,
17                       new { @class = "btn btn-block btn-default btn-lg" })
18  </div>
19  <div class="row panel">
20      @Html.ActionLink("Відбір даних про заклади освіти",
21                       "Selection", "Universities", null,
22                       new { @class = "btn btn-block btn-default btn-lg" })
23  </div>
24  <div class="row panel">
25      @Html.ActionLink("Назад",
26                       "EntitiesList", "Home", null,
27                       new { @class = "btn btn-block btn-default btn-lg" })
28  </div>
```

Рисунок 3.15 – Код представлення Index контролеру Universities

Представлення Selection контролеру Universities (рисунок 3.16) використовує модель UniversityViewController, відображає інформацію про заклади освіти у вигляді таблиці та дозволяє відфільтрувати її за вказаними користувачем параметрами. Також використовуються часткові представлення _SelectionForm (відображає форму для введення параметрів відбору) (рисунок 3.17), _TableHead та _TableBody, які відповідають за відображення таблиці даних (рисунок 3.18).

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "Selection";
}

<div class="row panel">
    <div id="parameters" class="col-sm-2">
        <h3>Параметри</h3>
        @Html.Partial("_SelectionForm")
        <h3>
            @Html.ActionLink("Назад", "Index")
        </h3>
    </div>
    <div class="col-sm-10">
        <table class="table">
            <thead>
                @Html.Partial("_TableHead", @Model)
            </thead>
            <tbody id="data">
                @Html.Partial("_TableBody", @Model)
            </tbody>
        </table>
    </div>
</div>
```

Рисунок 3.16 – Код представлення Selection контролеру Universities

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))
{
    <p>
        Назва закладу<br />
        <input type="text" name="selUniversityName" />
    </p>
    <p>
        Ідентифікаційний код<br />
        <input type="text" name="selCode" />
    </p>
    <p>
        Тип закладу<br />
        <input type="text" name="selType" />
    </p>
    <p>
        Область<br />
        @Html.DropDownList("selArea", ViewBag.selArea as SelectList)
    </p>
    <p>
        Населений пункт<br />
        <input type="text" name="selSettlement" />
    </p>
    <p>
        <input type="submit" value="Відібрати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Рисунок 3.17 – Код часткового представлення _SelectionForm контролеру Universities

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>


@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.UniversityName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Code)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Type)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Area)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Settlement)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Address)
        </td>
    </tr>
}
```

Рисунок 3.18 – Код часткового представлення _TableBody контролеру Universities

Представлення BrowseByAreas контролеру Universities (рисунок 3.19) використовує модель UniversityViewController, відображає інформацію про заклади освіти з можливістю фільтрації даних її за вказаною областю. Також використовується метод часткового представлення _GetDataByArea, який виводить підменю вибору областей (рисунок 3.20).

```
@using UniversitiesInfo.Web.Controllers
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "BrowseByAreas";
}

<h3>
    @Html.ActionLink("Назад", "Index")
</h3>

<div class="ajaxLink">
    @foreach (string area in @ViewBag.Areas as IEnumerable<string>)
    {
        @Ajax.ActionLink(
                area,
                "_GetDataByArea",
                new { selArea = area },
                new AjaxOptions
                {
                    UpdateTargetId = "data",
                    LoadingElementId = "loading",
                    LoadingElementDuration = 1000
                },
                new { @class = "btn btn-default" }
            )
    }
</div>
<div id="loading" class="load" style="display:none">
    <p>Завантаження даних...</p>
</div>

<div id="data">
    @Html.Action("_GetDataByArea",
            new { selNumber = @UniversitiesController.ALL_PAGE_LINK_NAME })
</div>
```

Рисунок 3.19 – Код представлення BrowseByAreas контролеру Universities

```
Ссылок: 0
public ActionResult BrowseByAreas()
{
    ViewBag.Areas = new[] { ALL_PAGE_LINK_NAME }
        .Concat(Objects.Select(e => e.Area.ToString()).Distinct().OrderBy(e => e));
    return View(viewModelObjects);
}

Ссылок: 0
public PartialViewResult _GetDataByArea(string selArea)
{
    var model = viewModelObjects;
    if (selArea != null && selArea != ALL_PAGE_LINK_NAME)
        model = model.Where(e => e.Area == selArea);
    System.Threading.Thread.Sleep(2000);
    return PartialView("_BrowseData", model);
}
```

Рисунок 3.20 – Код методу представлення _GetDataByArea контролеру Universities

3.6 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки програмного продукту. Проаналізувавши потреби програмного продукту, було обрано мову програмування C#. Для зручності розробки інтерфейсу було обрано платформу ASP.NET MVC. Програмно реалізовано базові модулі інформаційної системи.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Тестування відображення та фільтрації даних

При відкритті головної сторінки користувач спостерігає дані про інформаційну систему та меню навігації в шапці сайту (рисунок 4.1). Для пошуку даних потрібно перейти у меню «Категорії», для редагування даних – у меню «Адмін-панель» [16].

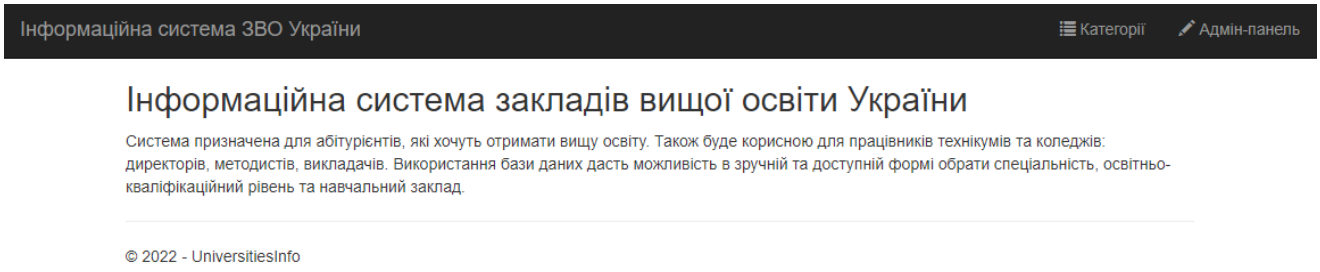


Рисунок 4.1 – Головна сторінка інформаційної системи

Після переходу у меню «Категорії» користувач може обрати потрібну йому категорію, в якій потрібно знайти певну інформацію, або перейти до підсистеми рекомендації спеціальностей (рисунок 4.2).

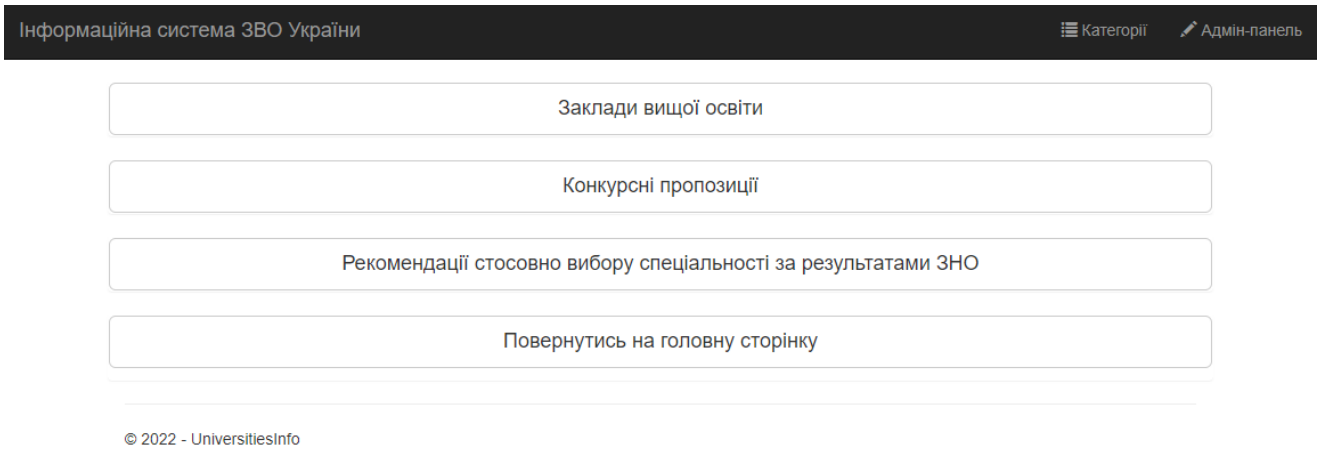


Рисунок 4.2 – Сторінка з списком категорій відображення даних

Обравши певну категорію, користувач може обрати, яким саме поданням потрібно відобразити дані – переглянути весь список або відфільтрувати дані за певними критеріями (рисунок 4.3) [17].

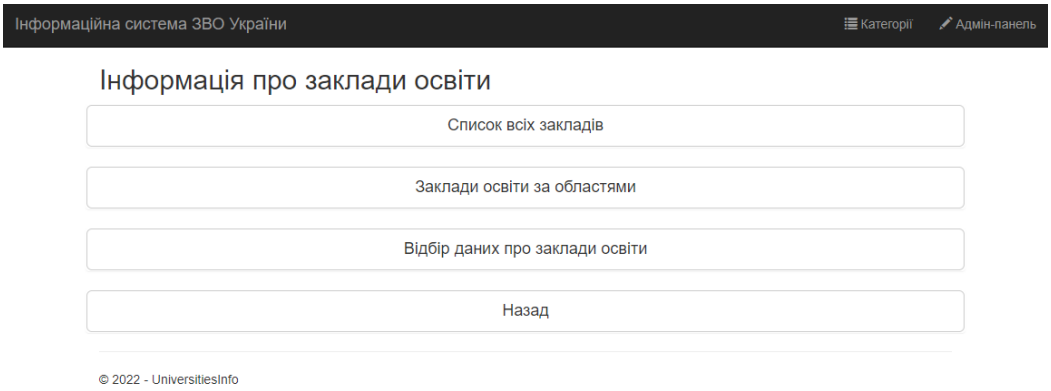


Рисунок 4.3 – Сторінка з списком представлення даних

Список всіх закладів відображає основну інформацію про навчальний заклад у вигляді таблиці (рисунок 4.4). Для відображення більш детальної інформації користувач може натиснути на відповідну кнопку, після чого відкриється сторінка з переліком усіх даних про обраний заклад освіти (рисунок 4.5) [18].

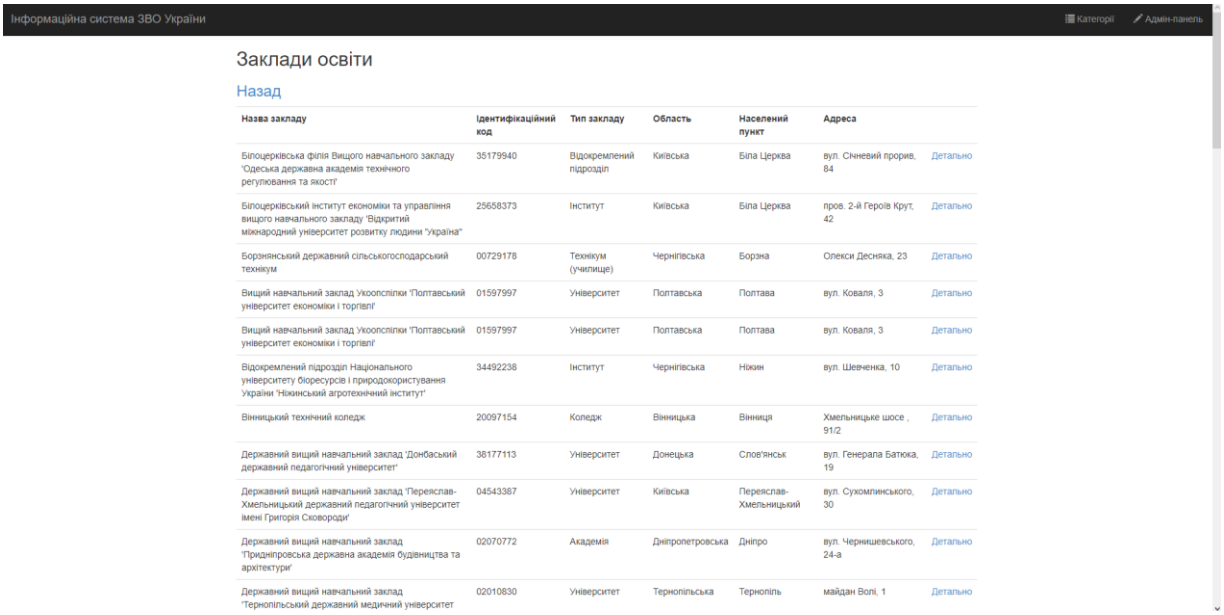


Рисунок 4.4 – Сторінка зі списком всіх даних про заклади освіти

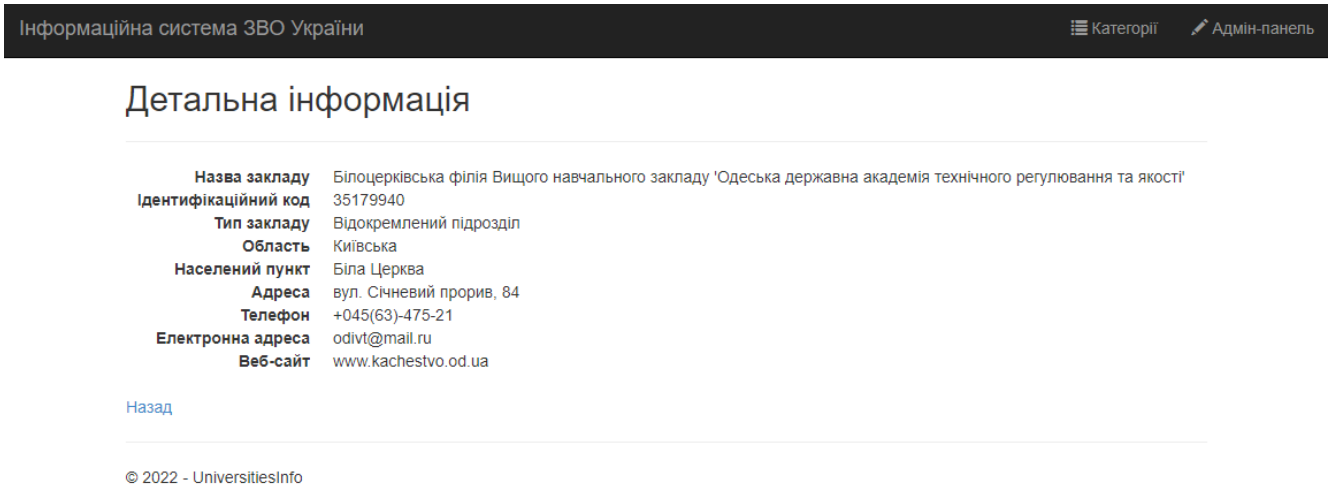


Рисунок 4.5 – Сторінка з детальною інформацією про заклад освіти

Найбільш популярною категорією для пошуку є область, тому було розроблене окреме подання даних з відокремленням закладів по вказаній області (рисунок 4.6) [19].

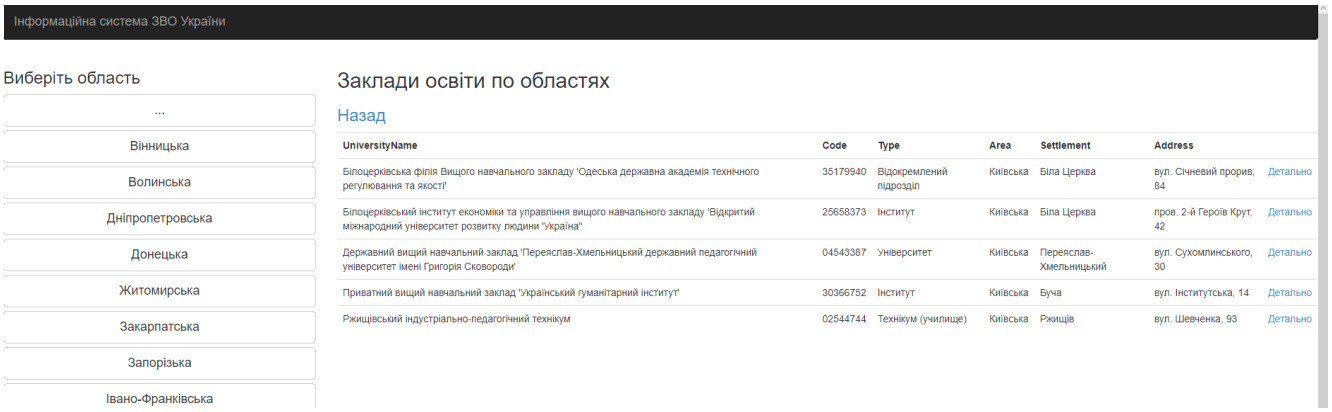


Рисунок 4.6 – Сторінка з відображенням списку закладів освіти за вказаним параметром

Для зручності користувач може вказати певні дані, за якими потрібно знайти один або декілька закладів освіти. Для цього потрібно перейти на відповідну сторінку та вказати один або декілька параметрів при необхідності (рисунок 4.7) [20].

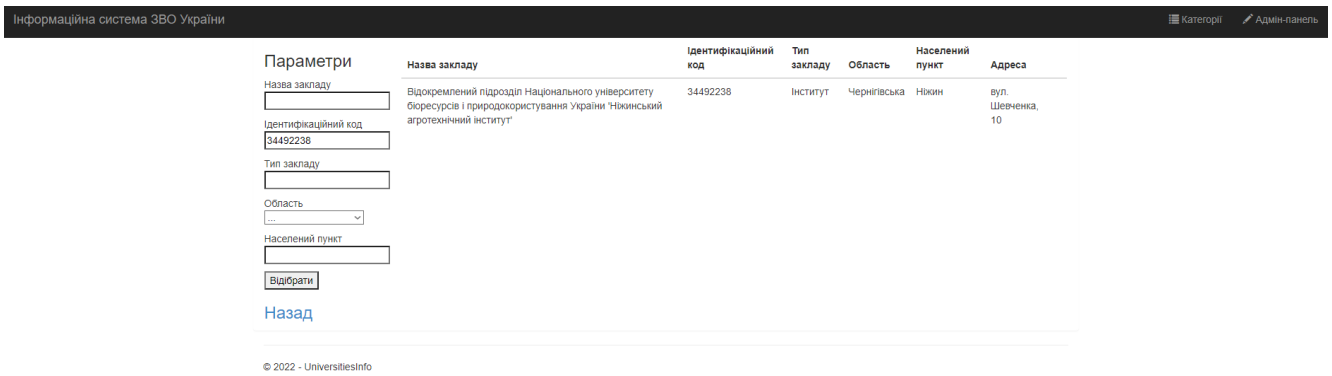


Рисунок 4.7 – Сторінка з фільтрами даних про заклади освіти

4.2 Тестування адмін-панелі

Перейшовши у адмін-панель користувач може обрати категорію, по якій потрібно змінити дані (рисунок 4.8).

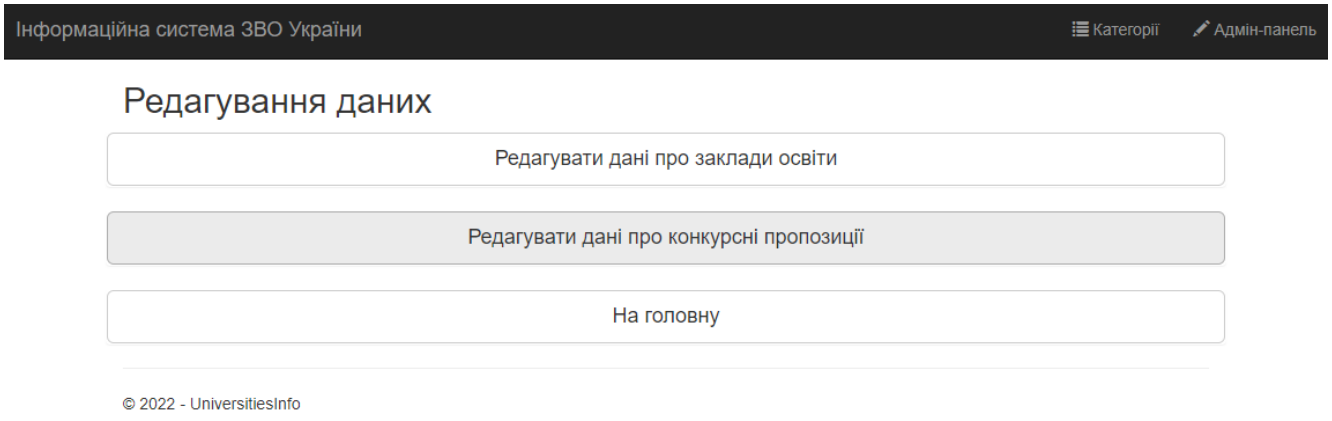


Рисунок 4.8 – Сторінка адмін-панелі

Обравши потрібну категорію, користувач може переглянути весь список даних, які зберігаються в системі (рисунок 4.9), додати новий навчальний заклад або конкурсну пропозицію (рисунок 4.10), відредагувати вже існуючі дані (рисунок 4.11) або видалити непотрібні (рисунок 4.12). При введені некоректних даних система виведе на екран повідомлення про помилки, допущені при редагуванні (рисунок 4.13).

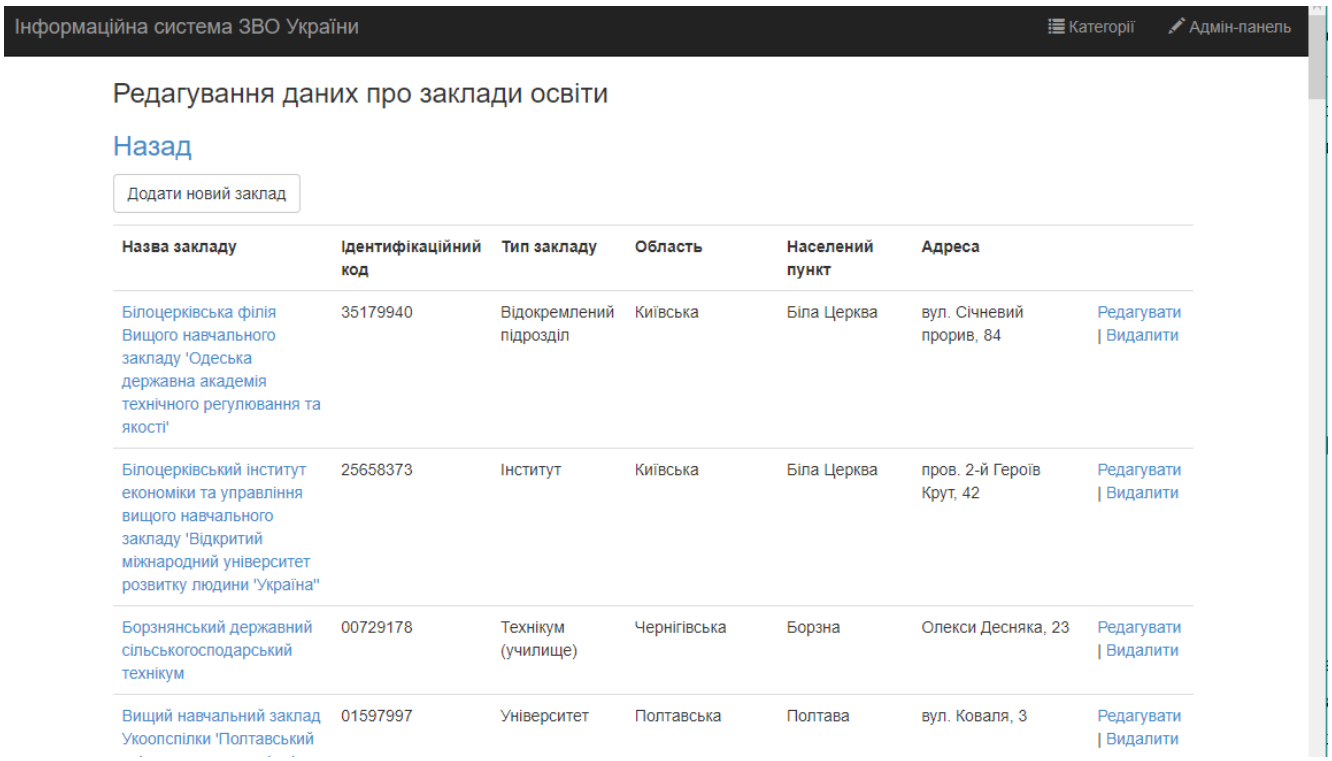


Рисунок 4.9 – Сторінка перегляду даних категорії в адмін-панелі

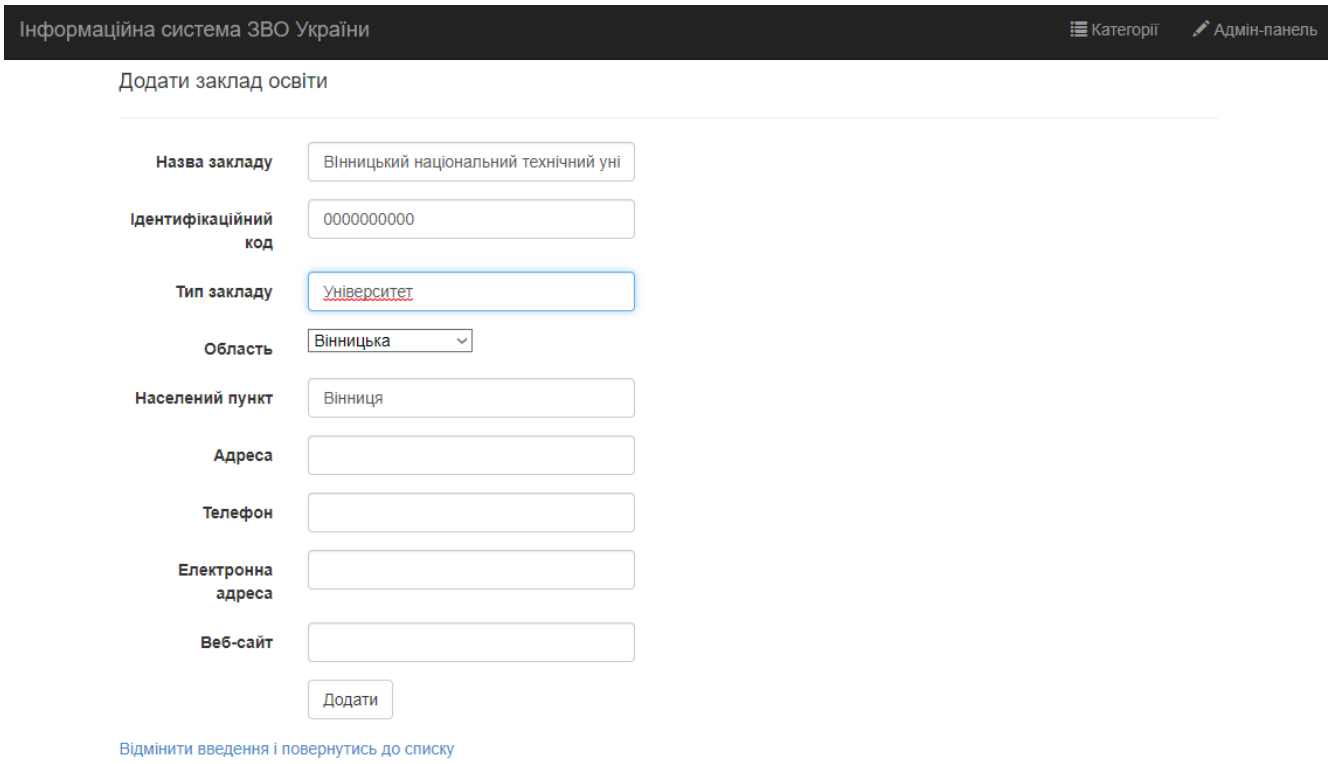


Рисунок 4.10 – Сторінка додання нового закладу освіти в систему

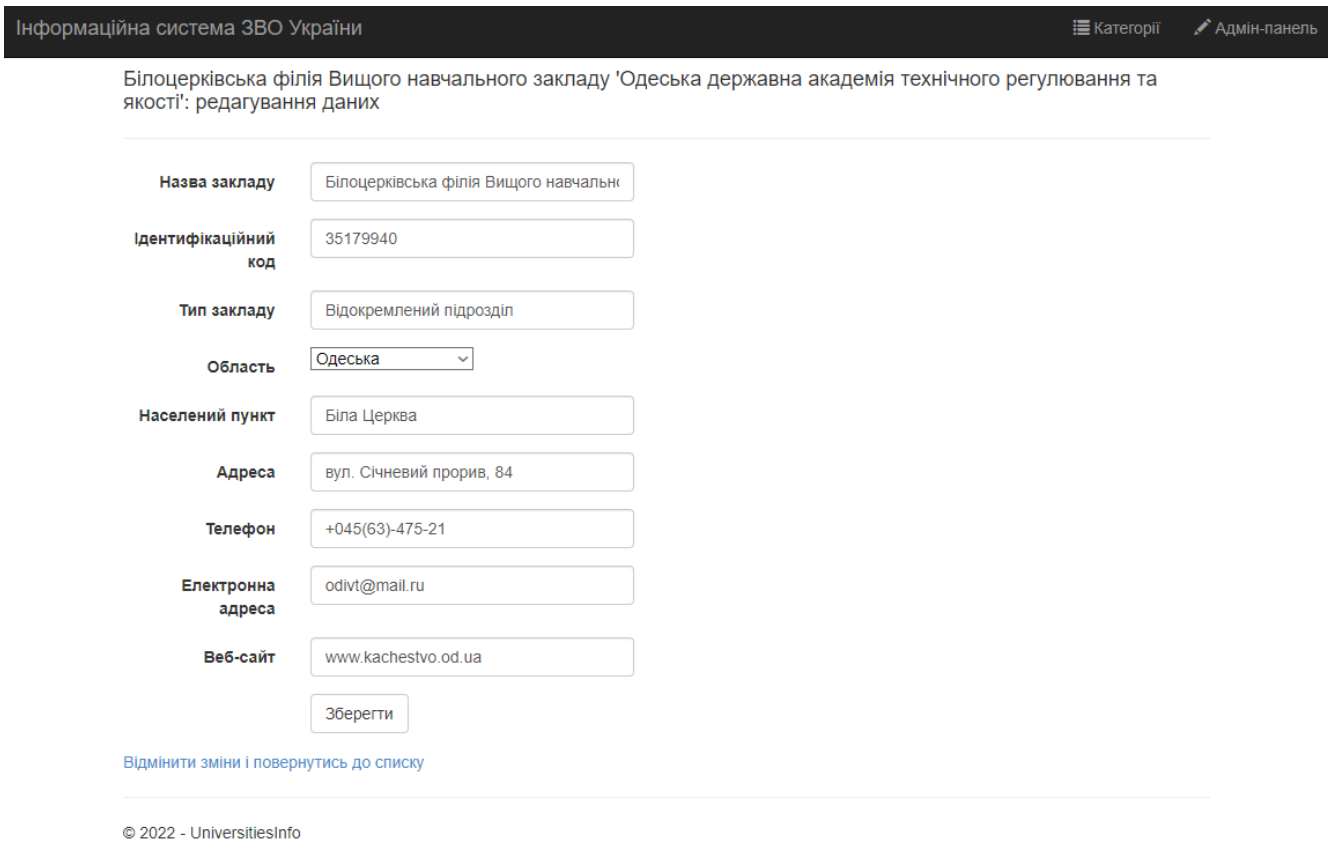


Рисунок 4.11 – Сторінка редагування даних про заклад освіти

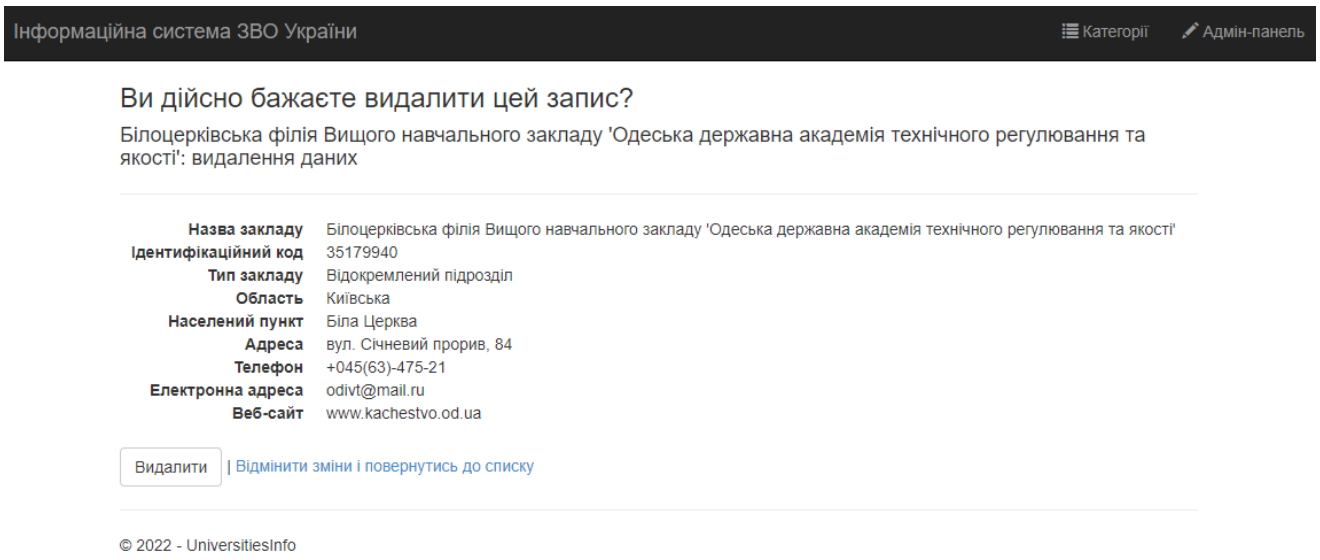


Рисунок 4.12 – Сторінка видалення даних про заклад освіти

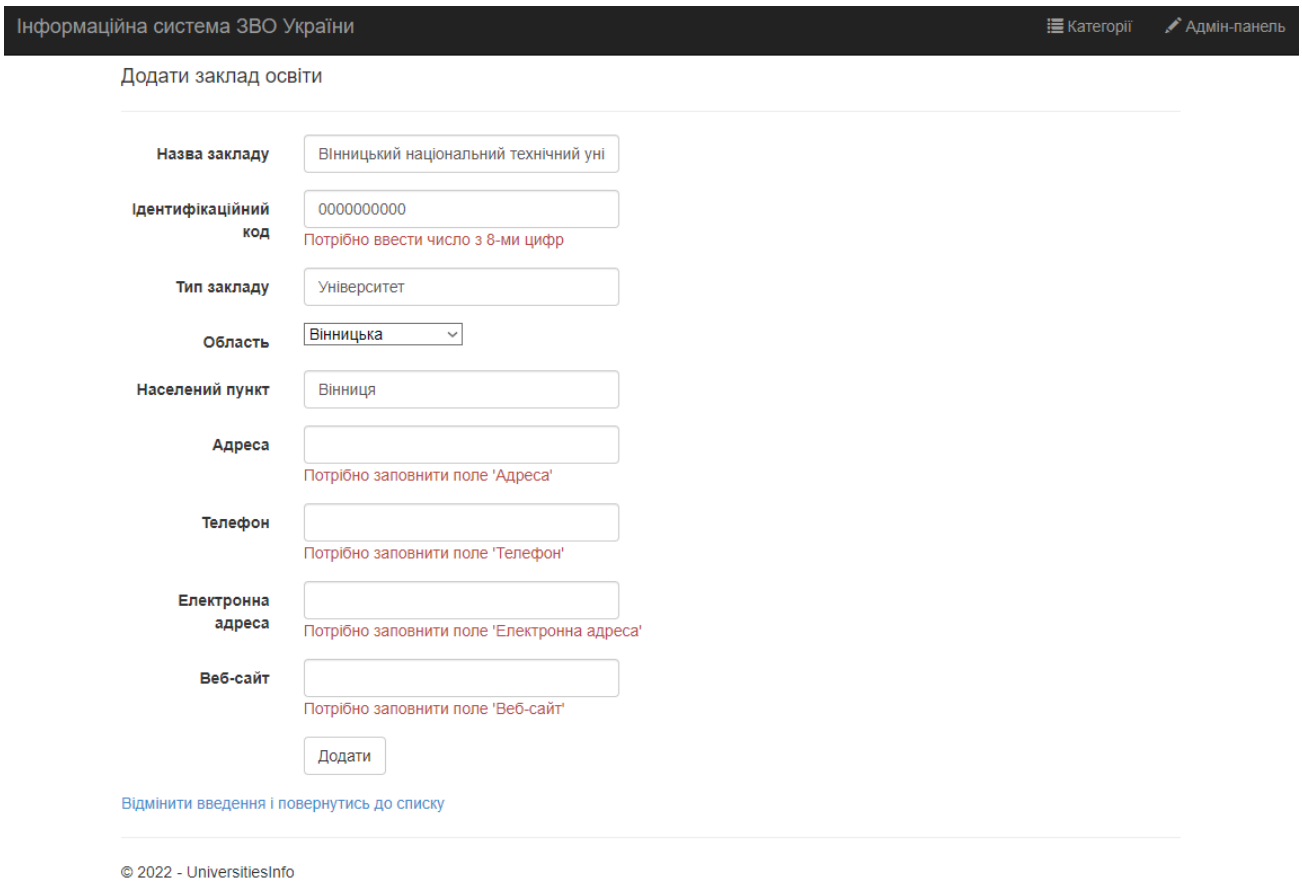


Рисунок 4.13 – Повідомлення про введення некоретктних даних та незаповнені поля

4.3 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) двоядерний процесор з тактовою частотою 1 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-розрядної системи
Операційна система	Windows XP та вище

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) двоядерний процесор з тактовою частотою 2 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-разрядної системи
Операційна система	Windows 7 та вище

Після запуску системи користувач може обрати категорію, в якій йому треба знайти інформацію, після чого обрати представлення даних для пошуку. При необхідності дані можна виправити через адмін-панель, додати нові або видалити неактуальні записи.

4.4 Висновки

У четвертому розділі було проведено тестування інформаційної системи для вибору спеціальностей в закладах вищої освіти України. Було перевірено відображення даних на сторінках, фільтрацію даних та адмін-панель. Також було розроблено інструкцію користувача для встановлення і початку експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено інформаційну систему для вибору спеціальностей в закладах вищої освіти України. Для розробки було використано середовище програмної розробки Microsoft Visual Studio 2019.

Під час виконання бакалаврської дипломної роботи було проаналізовано стан питання на сьогоднішній день. Було розглянуто основні аналоги програмного продукту, визначено їхні переваги і недоліки у порівнянні з власним програмним продуктом. Базуючись на результатах порівняння було встановлено основні задачі бакалаврської дипломної роботи.

Під час аналізу технологій розробки було обґрунтовано вибір мови програмування C# та веб-платформи ASP.NET MVC.

У бакалаврській дипломній роботі було зроблено:

- розроблено метод рекомендації спеціальностей;
- розроблено модель системи підбору закладів освіти для вступу та рекомендованих спеціальностей на основі введених користувачем балів ЗНО;
- розроблено інтерфейс та дизайн інформаційної системи, що буде зручним у використанні непідготовленим користувачем;
- програмно реалізовано спеціалізовану інформаційну систему для вибору спеціальностей у закладах вищої освіти;
- проведено тестування програмного продукту.

Було розроблено схему загального алгоритму роботи системи та модуля рекомендації спеціальностей.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Також було розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поняття інформаційної системи. Мета та завдання створення управлінських інформаційних систем. [Електронний ресурс] – Режим доступу до ресурсу: https://helpiks.org/558940.html

2. Войтко В.В. Інформаційно-пошукова система для абітурієнта вищих навчальних закладів України / В.В. Войтко, А.В. Денисюк, М.О. Кубай, А.М. Безверхний Матеріали XLIX науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2020) : збірник доповідей. – Вінниця : ВНТУ, 2020 - С.1144-1146.

3. Єдина державна електронна база з питань освіти. [Електронний ресурс] – Режим доступу до ресурсу: https://info.edbo.gov.ua/

4. Вузи України – Освіта.UA. [Електронний ресурс] – Режим доступу до ресурсу: https://osvita.ua/vnz/guide/

5. ЗВО (вузи) в Україні – інститути, університети, академії. [Електронний ресурс] – Режим доступу до ресурсу: https://www.education.ua/universities/

6. Інтерфейс користувача – [Електронний ресурс] – Режим доступу: https://studopedia.com.ua/1_369672_Interfeys-koristuvacha.html

7. C# Fundamentals - C# 10 and .NET 6 using Visual Studio 2022: Course in a book : навч. посіб. / Adam Seebeck – unQbd, 2021. – 168 ст.

8. Мова програмування C++ [Електронний ресурс] – Режим доступу до ресурсу: https://life-prog.com/ukr/.

9. Мова програмування C# [Електронний ресурс] – Режим доступу до ресурсу: http://progopedia.com/language/csharp/.

10. Мова програмування Java [Електронний ресурс] – Режим доступу до ресурсу: http://progopedia.ru/language/java/.

11. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org.

12. Створення сайту на ASP.NET MVC [Електронний ресурс] – Режим доступу до ресурсу: https://luxsite.ua/ua/aspnetmvc/

13. ASP.NET MVC | Введення в створення сайтів [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/mvc5/1.1.php

14. Andrew Troelsen Pro C# 8 with .NET Core 3 : навч. посіб. / Andrew Troelsen, Phil Japikse – Apress, 2020. – 1881 ст.

15. Beginning C# Web Applications with Visual Studio .NET 1st Edition : навч. посіб. / Daniel Cazzulino, Victor Garcia Aprea, James Greenwood, Chris Goode – Apress, 2002. – 580 ст.

16. Testing ASP.NET Core services and web apps [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps

17. Testing ASP.NET Web Applications навч. посіб. / Jeff McWherter, Ben Hall – ISBN: 978-1-118-08122-8, 2011. – 432 ст.

18. Web Application Testing Complete Guide (How To Test A Website) [Електронний ресурс] – Режим доступу до ресурсу: https://www.softwaretestinghelp.com/web-application-testing/

19. Web Application Testing: 8 Step Guide to Website Testing [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/web-application-testing.html

20. Programming ASP.NET Second Edition : навч. посіб. / Jesse Liberty, Dan Hurwitz – O'Reilly Media, 2003. – 1008 ст.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

31 березня 2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ доц. Д.І. Катєльніков

31 березня 2022 р.

Виконав:

_____студент гр. 1ПІ-19мс2 М.О. Кубай

31 березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України».

Галузь застосування – інформаційні веб-системи.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 12 від « 7 » лютого 2022 р.

3. Мета та призначення розробки.

Метою роботи є підвищення рівня обгрунтованості та спрощення процесу вибору навчального закладу шляхом розробки та використання інформаційної системи для вибору спеціальностей в закладах вищої освіти України, що дозволить оптимізувати процес прийняття рішень.

Призначення роботи – спрощення абітурієнтам пошуку закладу освіти для вступу.

4. Вихідні дані для проведення БДР

1. Andrew Troelsen Pro C# 8 with .NET Core 3 : навч. посіб. / Andrew Troelsen, Phil Japikse – Apress, 2020. – 1881 ст.

2. Programming ASP.NET Second Edition : навч. посіб. / Jesse Liberty, Dan Hurwitz – O'Reilly Media, 2003. – 1008 ст.

3. Beginning C# Web Applications with Visual Studio .NET 1st Edition : навч. посіб. / Daniel Cazzulino, Victor Garcia Aprea, James Greenwood, Chris Goode – Apress, 2002. – 580 ст.

5. Технічні вимоги

Вхідні дані – інформація про навчальні заклади, перелік спеціальностей, кількість місць на спеціальності; вихідні дані – графічний інтерфейс зі списком закладів та конкурсних пропозицій.

6. Конструктивні вимоги

Дизайн системи повинен відповідати естетичним та ергономічним вимогам, система має бути зручною у використанні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інформаційної системи для вибору спеціальностей в закладах вищої освіти України

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Катєльніков Д. І.

Оригінальність	**94,1%**
Схожість	5,9%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

 Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

 Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кубай М. О.

Керівник роботи _____ Катєльніков Д. І.

Додаток В – Лістинг програми

Програмний код класу DataContextIO

```
using UniversitiesInfo.Domain;
using System.IO;
using System.Text;
using System.Xml;

namespace UniversitiesInfo.Data
{
   partial class DataContext
   {
      public static void Save()
      {
         XmlTextWriter writer = null;
         writer = new XmlTextWriter(fileName, Encoding.Unicode);
         writer.WriteStartDocument();
         writer.WriteStartElement("UniversitiesInfo");
         WriteUniversities(writer);
         WriteOffers(writer);
         writer.WriteEndElement();
         writer.WriteEndDocument();
         writer.Close();
      }

      private static void WriteUniversities(XmlTextWriter writer)
      {
         writer.WriteStartElement("Universities");
         foreach (var obj in universities)
         {
            writer.WriteStartElement("University");
            writer.WriteAttributeString("Id", obj.Id.ToString());
            writer.WriteAttributeString("UniversityName", obj.UniversityName);
            writer.WriteAttributeString("Code", obj.Code);
            writer.WriteAttributeString("Type", obj.Type);
            writer.WriteAttributeString("Area", obj.Area);
            writer.WriteAttributeString("Settlement", obj.Settlement);
            writer.WriteAttributeString("Address", obj.Address);
            writer.WriteAttributeString("Phone", obj.Phone);
            writer.WriteAttributeString("Email", obj.Email);
            writer.WriteAttributeString("Website", obj.Website);
            writer.WriteEndElement();
         }
         writer.WriteEndElement();
      }

      private static void WriteOffers(XmlTextWriter writer)
      {
         writer.WriteStartElement("Offers");
         foreach (var obj in offers)
         {
            writer.WriteStartElement("Offers");
            writer.WriteAttributeString("Id", obj.Id.ToString());
            writer.WriteAttributeString("UniversityName", obj.UniversityName);
            writer.WriteAttributeString("SpecialityCode", obj.SpecialityCode);
            writer.WriteAttributeString("NumberofPlaces", obj.NumberofPlaces.ToString());
            writer.WriteAttributeString("EducationLevel", obj.EducationLevel);
            writer.WriteEndElement();
         }
         writer.WriteEndElement();
```

```
        }

        public static void Load()
        {
            if (!File.Exists(fileName)) return;
            XmlTextReader reader = null;
            reader = new XmlTextReader(fileName)
            {
                WhitespaceHandling = WhitespaceHandling.None
            };
            while (reader.Read())
            {
                if (reader.NodeType == XmlNodeType.Element)
                {
                    if (reader.Name == "University")
                    {
                        ReadUniversity(reader);
                    }
                    else if (reader.Name == "Offer")
                    {
                        ReadOffer(reader);
                    }
                }
            }
            reader.Close();
        }

        private static void ReadUniversity(XmlTextReader reader)
        {
            University obj = new University();
            string s = reader.GetAttribute("Id");
            if (!string.IsNullOrEmpty(s)) obj.Id = int.Parse(s);
            obj.UniversityName = reader.GetAttribute("UniversityName");
            obj.Code = reader.GetAttribute("Code");
            obj.Type = reader.GetAttribute("Type");
            obj.Area = reader.GetAttribute("Area");
            obj.Settlement = reader.GetAttribute("Settlement");
            obj.Address = reader.GetAttribute("Address");
            obj.Phone = reader.GetAttribute("Phone");
            obj.Email = reader.GetAttribute("Email");
            obj.Website = reader.GetAttribute("Website");
            universities.Add(obj);
        }

        private static void ReadOffer(XmlTextReader reader)
        {
            Offer obj = new Offer();
            string s = reader.GetAttribute("Id");
            if (!string.IsNullOrEmpty(s)) obj.Id = int.Parse(s);
            obj.UniversityName = reader.GetAttribute("UniversityName");
            obj.SpecialityCode = reader.GetAttribute("SpecialityCode");
            s = reader.GetAttribute("NumberofPlaces");
            if (!string.IsNullOrEmpty(s)) obj.NumberofPlaces = int.Parse(s);
            obj.EducationLevel = reader.GetAttribute("EducationLevel");
            offers.Add(obj);
        }
    }
}
```

Програмний код класу RouteConfig

```
namespace UniversitiesInfo.Web
{
   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 = "Home", action = "Index", id = UrlParameter.Optional }
         );
      }
   }
}
```

Програмний код класу AdminController

```
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class AdminController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }
   }
}
```

Програмний код класу HomeController

```
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }

      public ActionResult EntitiesList()
      {
         return View();
      }
   }
}
```

Програмний код класу NavigationController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
```

```
namespace UniversitiesInfo.Web.Controllers
{
   public class NavigationController : Controller
   {
      public IEnumerable<University> Universities { get; set; }

      public NavigationController()
      {
         Universities = DataContext.universities;
      }

      public const string ALL_CATEGORIES = "...";

      [ChildActionOnly]
      public PartialViewResult UniversitiesByAreasMenu(
      string categoryName = ALL_CATEGORIES)
      {
         ViewBag.SelectedCategoryName = categoryName;
         List<string> categoryNames = new List<string>();
         categoryNames.Add(ALL_CATEGORIES);
         categoryNames.AddRange(Universities.Select(e => e.Area).Distinct().OrderBy(e => e));
         return PartialView(categoryNames);
      }
   }
}
```

Програмний код класу OffersController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
   public class OffersController : Controller
   {
      public ActionResult Index()
      {
         return View();
      }

      private IEnumerable<Offer> objects;
      private IEnumerable<OfferViewModel> viewModelObjects;

      public IEnumerable<Offer> Objects
      {
         get { return objects; }
         set
         {
            objects = value;
            viewModelObjects = objects.Select(e => (OfferViewModel)e)
               .OrderBy(e => e.UniversityName);
         }
      }

      public OffersController()
      {
         Objects = DataContext.offers;
      }
```

```
        public ActionResult Selection()
        {
            return View(viewModelObjects);
        }

        public PartialViewResult _SelectData(string selUniversityName, string selSpecialityCode,
            int? PlacesFrom, int? PlacesTo, string selEducationLevel)
        {
            var model = viewModelObjects;
            if (!string.IsNullOrWhiteSpace(selUniversityName))
                model = model.Where(e => e.UniversityName.ToLower()
                    .StartsWith(selUniversityName.ToLower()));
            if (!string.IsNullOrWhiteSpace(selSpecialityCode))
                model = model.Where(e => e.SpecialityCode.ToLower()
                    .StartsWith(selSpecialityCode.ToLower()));
            if (PlacesFrom.HasValue)
                if (PlacesFrom.HasValue)
                    model = model.Where(e => e.NumberofPlaces >= PlacesFrom.Value);
            if (PlacesTo.HasValue)
                model = model.Where(e => e.NumberofPlaces <= PlacesTo.Value);
            if (!string.IsNullOrWhiteSpace(selEducationLevel))
                model = model.Where(e => e.EducationLevel.ToLower()
                    .StartsWith(selEducationLevel.ToLower()));
            System.Threading.Thread.Sleep(1000);
            return PartialView("_TableBody", model);
        }

        public ViewResult List()
        {
            return View(viewModelObjects);
        }
    }
}
```

Програмний код класу OffersCrudController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class OffersCrudController : Controller
    {
        private IList<Offer> objects;
        private IEnumerable<OfferViewModel> viewModelObjects { get; set; }
        private IEnumerable<OfferEditingModel> editingModelObjects { get; set; }

        public IList<Offer> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects
                    .Select(e => (OfferViewModel)e).OrderBy(e => e.UniversityName);
                editingModelObjects = objects
                    .Select(e => (OfferEditingModel)e).OrderBy(e => e.UniversityName);
            }
```

```
}

public OffersCrudController()
{
   Objects = DataContext.offers;
}

public ActionResult Index()
{
   return View(viewModelObjects);
}

public ViewResult Create()
{
   return View(new OfferEditingModel());
}

[HttpPost]
public ActionResult Create(OfferEditingModel model)
{
   if (!ModelState.IsValid)
   {
      return View(model);
   }
   Objects.Add((Offer)model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Дані \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

[HttpGet]
public ActionResult Edit(int id)
{
   OfferEditingModel model = editingModelObjects.First(e => e.Id == id);
   return View(model);
}

[HttpPost]
public ActionResult Edit(OfferEditingModel model)
{
   if (!ModelState.IsValid)
   {
      return View(model);
   }
   var entityObject = Objects.First(e => e.Id == model.Id);
   UpdateEntityObject(entityObject, model);
   DataContext.Save();
   TempData["message"] = string.Format(
      "Зміни даних \"{0}\" збережено", model.UniversityName);
   return RedirectToAction("Index");
}

private void UpdateEntityObject(Offer entityObject,
      OfferEditingModel model)
{
   entityObject.UniversityName = model.UniversityName;
   entityObject.SpecialityCode = model.SpecialityCode;
   entityObject.NumberofPlaces = model.NumberofPlaces;
   entityObject.EducationLevel = model.EducationLevel;
}
```

```
        public ActionResult Delete(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }

        [HttpPost]
        public ActionResult Delete(Offer model)
        {
            Offer entityObject = Objects.First(e => e.Id == model.Id);
            Objects.Remove(entityObject);
            DataContext.Save();
            return RedirectToAction("Index");
        }

        public ActionResult Details(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }
    }
}
```

Програмний код класу UniversitiesController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesController : Controller
    {
        private IEnumerable<University> objects;
        private IEnumerable<UniversityViewModel> viewModelObjects;

        public IEnumerable<University> Objects
        {
            get { return objects; }
            set
            {
                objects = value;
                viewModelObjects = objects.Select(e => (UniversityViewModel)e)
                    .OrderBy(e => e.UniversityName);
            }
        }

        public UniversitiesController()
        {
            Objects = DataContext.universities;
        }

        public ActionResult Index()
        {
            return View();
        }

        public ViewResult UniversitiesByAreasInfo(
```

```
string categoryName = NavigationController.ALL_CATEGORIES)
{
   IEnumerable<University> models = Objects.OrderBy(e => e.UniversityName);
   if (!string.IsNullOrEmpty(categoryName) &&
      categoryName != NavigationController.ALL_CATEGORIES)
   {
      models = models
         .Where(e => e.Area == categoryName);
   }
   ViewBag.SelectedCategoryName = categoryName;
   return View(models);
}

public ActionResult Selection()
{
   ViewBag.selArea = CreateAreasSelectList();
   return View(viewModelObjects);
}

const string ALL_VALUES = "...";

List<SelectListItem> CreateAreasSelectList()
{
   List<string> values = new List<string>();
   values.Add(ALL_VALUES);
   values.AddRange(Objects.Select(e => e.Area).Distinct());
   List<SelectListItem> list = new List<SelectListItem>();
   foreach (string e in values)
   {
      list.Add(new SelectListItem
      {
         Text = e,
         Value = e
      });
   }
   return list;
}

public PartialViewResult _SelectData(string selUniversityName, string selCode,
   string selType, string selArea, string selSettlement)
{
   var model = viewModelObjects;
   if (!string.IsNullOrWhiteSpace(selUniversityName))
      model = model.Where(e => e.UniversityName.ToLower()
         .StartsWith(selUniversityName.ToLower()));

   if (!string.IsNullOrWhiteSpace(selCode))
      model = model.Where(e => e.Code.ToLower()
         .StartsWith(selCode.ToLower()));

   if (!string.IsNullOrWhiteSpace(selType))
      model = model.Where(e => e.Type.ToLower()
         .StartsWith(selType.ToLower()));

   if (selArea != null && selArea != ALL_VALUES)
      model = model.Where(e => e.Area == selArea);

   if (!string.IsNullOrWhiteSpace(selSettlement))
      model = model.Where(e => e.Settlement.ToLower()
         .StartsWith(selSettlement.ToLower()));
```

```
            System.Threading.Thread.Sleep(1000);
            return PartialView("_TableBody", model);
        }

        public const string ALL_PAGE_LINK_NAME = "...";

        public ActionResult BrowseByAreas()
        {
            ViewBag.Areas = new[] { ALL_PAGE_LINK_NAME }
                .Concat(Objects.Select(e => e.Area.ToString()).Distinct().OrderBy(e => e));
            return View(viewModelObjects);
        }

        public PartialViewResult _GetDataByArea(string selArea)
        {
            var model = viewModelObjects;
            if (selArea != null && selArea != ALL_PAGE_LINK_NAME)
                model = model.Where(e => e.Area == selArea);
            System.Threading.Thread.Sleep(2000);
            return PartialView("_BrowseData", model);
        }

        public ViewResult List()
        {
            return View(viewModelObjects);
        }

        public ViewResult _Details(int id)
        {
            var obj = viewModelObjects.First(e => e.Id == id);
            return View(obj);
        }

        public ActionResult Recommendation()
        {
            return View();
        }

    }
}
```

Програмний код класу UniversitiesCrudController

```
using UniversitiesInfo.Data;
using UniversitiesInfo.Domain;
using UniversitiesInfo.Web.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace UniversitiesInfo.Web.Controllers
{
    public class UniversitiesCrudController : Controller
    {
        private IList<University> objects;
        private IEnumerable<UniversityViewModel> viewModelObjects { get; set; }
        private IEnumerable<UniversityEditingModel> editingModelObjects { get; set; }

        public IList<University> Objects
        {
            get { return objects; }
            set
```

```
    {
        objects = value;
        viewModelObjects = objects
            .Select(e => (UniversityViewModel)e).OrderBy(e => e.UniversityName);
        editingModelObjects = objects
            .Select(e => (UniversityEditingModel)e).OrderBy(e => e.UniversityName);
    }
}

public IEnumerable<University> Areas { get; set; }

public UniversitiesCrudController()
{
    Objects = DataContext.universities;
    Areas = DataContext.universities;
}

public ActionResult Index()
{
    return View(viewModelObjects);
}

public ViewResult Create()
{
    ViewBag.Area = CreateAreasSelectList();
    return View(new UniversityEditingModel());
}

[HttpPost]
public ActionResult Create(UniversityEditingModel model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Area = CreateAreasSelectList();
        return View(model);
    }
    Objects.Add((University)model);
    DataContext.Save();
    TempData["message"] = string.Format(
        "Дані \"{0}\" збереженo", model.UniversityName);
    return RedirectToAction("Index");
}

[HttpGet]
public ActionResult Edit(int id)
{
    UniversityEditingModel model = editingModelObjects.First(e => e.Id == id);
    ViewBag.Area = CreateAreasSelectList(model.Area);
    return View(model);
}

[HttpPost]
public ActionResult Edit(UniversityEditingModel model)
{
    if (!ModelState.IsValid)
    {
        ViewBag.Area = CreateAreasSelectList(model.Area);
        return View(model);
    }
    var entityObject = Objects.First(e => e.Id == model.Id);
    UpdateEntityObject(entityObject, model);
```

```
            DataContext.Save();
            TempData["message"] = string.Format(
               "Зміни даних \"{0}\" збережено", model.UniversityName);
            return RedirectToAction("Index");
        }

        private void UpdateEntityObject(University entityObject,
              UniversityEditingModel model)
        {
            entityObject.UniversityName = model.UniversityName;
            entityObject.Code = model.Code;
            entityObject.Type = model.Type;
            entityObject.Area = model.Area;
            entityObject.Settlement = model.Settlement;
            entityObject.Address = model.Address;
            entityObject.Phone = model.Phone;
            entityObject.Email = model.Email;
            entityObject.Website = model.Website;
        }

        public ActionResult Delete(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }

        [HttpPost]
        public ActionResult Delete(University model)
        {
            University entityObject = Objects.First(e => e.Id == model.Id);
            Objects.Remove(entityObject);
            DataContext.Save();
            return RedirectToAction("Index");
        }

        public ActionResult Details(int id)
        {
            var model = editingModelObjects.First(e => e.Id == id);
            return View(model);
        }

        List<SelectListItem> CreateAreasSelectList(string selectedValue = "")
        {
            List<string> values = new List<string>();
            values.AddRange(Objects.Select(e => e.Area).Distinct());
            List<SelectListItem> list = new List<SelectListItem>();
            foreach (string e in values)
            {
                list.Add(new SelectListItem
                {
                    Text = e,
                    Value = e
                });
            }
            return list;
        }
    }
}
```

Програмний код класу OfferEditingModel

```
using UniversitiesInfo.Domain;
```

```
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
   public class OfferEditingModel
   {
      public int Id { get; set; }

      [Display(Name = "Назва закладу")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Назва закладу\'")]
      public string UniversityName { get; set; }

      [Display(Name = "Код спеціальності")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Код спеціальності\'")]
      [RegularExpression(@"[0-9]{3}", ErrorMessage =
         "Код спеціальності має складатись з трьох цифр")]
      public string SpecialityCode { get; set; }

      [Display(Name = "Кількість місць")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Кількість місць\'")]
      [Range(1, 50000, ErrorMessage = "Кількість місць"
         + "має бути в межах від 1 до 50000")]
      public int NumberofPlaces { get; set; }

      [Display(Name = "Освітньо-кваліфікаційний рівень")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Освітньо-кваліфікаційний рівень\'")]
      public string EducationLevel { get; set; }

      public static explicit operator OfferEditingModel(Offer obj)
      {
         return new OfferEditingModel()
         {
            Id = obj.Id,
            UniversityName = obj.UniversityName,
            SpecialityCode = obj.SpecialityCode,
            NumberofPlaces = obj.NumberofPlaces,
            EducationLevel = obj.EducationLevel
         };
      }

      public static explicit operator Offer(OfferEditingModel obj)
      {
         return new Offer()
         {
            Id = obj.Id,
            UniversityName = obj.UniversityName,
            SpecialityCode = obj.SpecialityCode,
            NumberofPlaces = obj.NumberofPlaces,
            EducationLevel = obj.EducationLevel
         };
      }
   }
}
```

Програмний код класу OfferViewModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;
```

```
namespace UniversitiesInfo.Web.Models
{
   public class OfferViewModel
   {
      public int Id { get; set; }

      [Display(Name = "Назва закладу")]
      public string UniversityName { get; set; }

      [Display(Name = "Код спеціальності")]
      public string SpecialityCode { get; set; }

      [Display(Name = "Кількість місць")]
      public int NumberofPlaces { get; set; }

      [Display(Name = "Освітньо-кваліфікаційний рівень")]
      public string EducationLevel { get; set; }

      public static explicit operator OfferViewModel(Offer obj)
      {
         return new OfferViewModel()
         {
            Id = obj.Id,
            UniversityName = obj.UniversityName,
            SpecialityCode = obj.SpecialityCode,
            NumberofPlaces = obj.NumberofPlaces,
            EducationLevel = obj.EducationLevel,
         };
      }
   }
}
```

Програмний код класу UniversityEditingModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
   public class UniversityEditingModel
   {
      public int Id { get; set; }

      [Display(Name = "Назва закладу")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Назва закладу\'")]
      [StringLength(512, MinimumLength = 2,
          ErrorMessage = "Назва закладу має містити"
          + "від 2 до 512 символів")]
      public string UniversityName { get; set; }

      [Display(Name = "Ідентифікаційний код")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Ідентифікаційний код\'")]
      [RegularExpression(@"[1-99999999]{1}", ErrorMessage =
         "Потрібно ввести число з 8-ми цифр")]
      public string Code { get; set; }

      [Display(Name = "Тип закладу")]
      [Required(ErrorMessage =
          "Потрібно заповнити поле \'Тип закладу\'")]
      [StringLength(128, MinimumLength = 6,
          ErrorMessage = "Тип закладу має містити"
```

```
    + "від 6 до 128 символів")]
public string Type { get; set; }

[Display(Name = "Область")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Область\'")]
[StringLength(17, MinimumLength = 4,
    ErrorMessage = "Область має містити"
    + "від 4 до 17 символів")]
public string Area { get; set; }

[Display(Name = "Населений пункт")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Населений пункт\'")]
[StringLength(128, MinimumLength = 2,
    ErrorMessage = "Населений пункт має містити"
    + "від 2 до 128 символів")]
public string Settlement { get; set; }

[Display(Name = "Адреса")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Адреса\'")]
[StringLength(512, MinimumLength = 2,
    ErrorMessage = "Населений пункт має містити"
    + "від 2 до 512 символів")]
public string Address { get; set; }

[Display(Name = "Телефон")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Телефон\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Телефон має містити"
    + "від 2 до 64 символів")]
public string Phone { get; set; }

[Display(Name = "Електронна адреса")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Електронна адреса\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Електронна адреса має містити"
    + "від 2 до 64 символів")]
public string Email { get; set; }

[Display(Name = "Веб-сайт")]
[Required(ErrorMessage =
    "Потрібно заповнити поле \'Веб-сайт\'")]
[StringLength(64, MinimumLength = 2,
    ErrorMessage = "Веб-сайт має містити"
    + "від 2 до 64 символів")]
public string Website { get; set; }


public static explicit operator UniversityEditingModel(University obj)
{
   return new UniversityEditingModel()
   {
      Id = obj.Id,
      UniversityName = obj.UniversityName,
      Code = obj.Code,
      Type = obj.Type,
      Area = obj.Area,
```

```
            Settlement = obj.Settlement,
            Address = obj.Address,
            Phone = obj.Phone,
            Email = obj.Email,
            Website = obj.Website
        };
    }

    public static explicit operator University(UniversityEditingModel obj)
    {
        return new University()
        {
            Id = obj.Id,
            UniversityName = obj.UniversityName,
            Code = obj.Code,
            Type = obj.Type,
            Area = obj.Area,
            Settlement = obj.Settlement,
            Address = obj.Address,
            Phone = obj.Phone,
            Email = obj.Email,
            Website = obj.Website
        };
    }
  }
}
```

Програмний код класу UniversityViewModel

```
using UniversitiesInfo.Domain;
using System.ComponentModel.DataAnnotations;

namespace UniversitiesInfo.Web.Models
{
  public class UniversityViewModel
  {
    public int Id { get; set; }

    [Display(Name = "Назва закладу")]
    public string UniversityName { get; set; }

    [Display(Name = "Ідентифікаційний код")]
    public string Code { get; set; }

    [Display(Name = "Тип закладу")]
    public string Type { get; set; }

    [Display(Name = "Область")]
    public string Area { get; set; }

    [Display(Name = "Населений пункт")]
    public string Settlement { get; set; }

    [Display(Name = "Адреса")]
    public string Address { get; set; }

    [Display(Name = "Телефон")]
    public string Phone { get; set; }

    [Display(Name = "Електронна адреса")]
    public string Email { get; set; }

    [Display(Name = "Веб-сайт")]
```

```
        public string Website { get; set; }

        public static explicit operator UniversityViewModel(University obj)
        {
            return new UniversityViewModel()
            {
                Id = obj.Id,
                UniversityName = obj.UniversityName,
                Code = obj.Code,
                Type = obj.Type,
                Area = obj.Area,
                Settlement = obj.Settlement,
                Address = obj.Address,
                Phone = obj.Phone,
                Email = obj.Email,
                Website = obj.Website
            };
        }
    }
}
```

Програмний код скрипта Admin.Index

```
@{
    ViewBag.Title = "Index";
}

<h2>Редагування даних</h2>

<div class="row panel">
    @Html.ActionLink("Редагувати дані про заклади освіти",
            "Index", "UniversitiesCrud", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Редагувати дані про конкурсні пропозиції",
            "Index", "OffersCrud", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("На головну",
              "Index", "Home", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта HomeEntitiesList

```
<h2></h2>
<div class="row panel">
    @Html.ActionLink("Заклади вищої освіти",
              "Index", "Universities", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Конкурсні пропозиції",
              "Index", "Offers", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Рекомендації стосовно вибору спеціальності за результатами ЗНО",
              "Recommendation", "Universities", null,
              new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
```

```
    <p>
      @Html.ActionLink("Повернутись на головну сторінку", "Index", null, new { @class = "btn btn-block btn-default btn-lg" })
    </p>
</div>
```

Програмний код скрипта HomeIndex

```
@{
    ViewBag.Title = "Index";
}

<h2>Інформаційна система закладів вищої освіти України</h2>

<div>
    Система призначена для абітурієнтів, які хочуть отримати вищу освіту.
    Також буде корисною для працівників технікумів та коледжів: директорів, методистів, викладачів.
    Використання бази даних дасть можливість в зручній та доступній формі обрати спеціальність, освітньо-кваліфікаційний рівень та навчальний заклад.
</div>
```

Програмний код скрипта Navigation

```
@model IEnumerable<string>

<h3>Виберіть область</h3>

@foreach (var link in Model)
{
    @Html.RouteLink(
        link.Length < 20 ? link : link.Substring(0, 17) + "...",
        new
        {
            controller = "Universities",
            action = "UniversitiesByAreasInfo",
            categoryName = link
        },
        new
        {
            @class = "btn btn-block btn-default btn-lg btn-nav_menu"
                + (link == ViewBag.SelectedCategoryName ? " btn-primary" : "")
        })
}
```

Програмний код скрипта Offers _Details

```
@model UniversitiesInfo.Web.Models.OfferViewModel

@{
    ViewBag.Title = "Details";
}

<h2>Детальна інформація</h2>

<div>
    <h4>Конкурсна пропозиція</h4>
    <hr />
    <dl class="dl-horizontal">

        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.SpecialityCode)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumberofPlaces)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EducationLevel)
        </dd>
    </dl>
</div>

<p>
    @Html.ActionLink("Назад", "List")
</p>
```

Програмний код скрипта Offers _SelectionForm

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))
{
    <p>
        Початок назви закладу<br />
        <input type="text" name="selUniversityName" />
    </p>
    <p>
        Код спеціальності<br />
        <input type="text" name="selSpecialityCode" />
    </p>
    <p>
        Кількість місць<br />
        від <input type="text" name="PlacesFrom" />
        до <input type="text" name="PlacesTo" />
    </p>
    <p>
        Освітньо-кваліфікаційний рівень<br />
        <input type="text" name="selEducationLevel" />
```

```
    </p>
    <p>
        <input type="submit" value="Відібрати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта Offers _TableBody

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>


@foreach (var item in Model)
{
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.UniversityName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.SpecialityCode)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.NumberofPlaces)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
</tr>
}
```

Програмний код скрипта Offers _TableHead

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

<tr>
    <th>
        @Html.DisplayNameFor(model => model.UniversityName)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.SpecialityCode)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.NumberofPlaces)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.EducationLevel)
    </th>
</tr>
```

Програмний код скрипта Offers Index

```
@{
    ViewBag.Title = "Index";
}

<h2>Конкурсні пропозиції</h2>

<div class="row panel">
    @Html.ActionLink("Список конкурсних пропозицій",
                "List", "Offers", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
```

```
    @Html.ActionLink("Відбір даних про конкурсні пропозиції",
                "Selection", "Offers", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
    @Html.ActionLink("Назад",
                "EntitiesList", "Home", null,
                new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта Offers List

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

@{
    ViewBag.Title = "List";
}

<h2>Конкурсні пропозиції</h2>
<h3>
    @Html.ActionLink("Назад", "Index")
</h3>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.DisplayFor(modelItem => item.UniversityName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.SpecialityCode)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.NumberofPlaces)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
</tr>
    }
</table>
```

Програмний код скрипта Offers Selection

```
@model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

@{
    ViewBag.Title = "Selection";
}
```

```
<div class="row panel">
  <div id="parameters" class="col-sm-2">
    <h3>Параметри</h3>
    @Html.Partial("_SelectionForm")
    <h3>
      @Html.ActionLink("Назад", "Index")
    </h3>
  </div>
  <div class="col-sm-10">
    <h2>Конкурсні пропозиції</h2>
    <table class="table">
      <thead>
        @Html.Partial("_TableHead", @Model)
      </thead>
      <tbody id="data">
        @Html.Partial("_TableBody", @Model)
      </tbody>
    </table>
  </div>
</div>
```

Програмний код скрипта OffersCrudCreate

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
  ViewBag.Title = "Create";
}

@using (Html.BeginForm())
{
  @Html.AntiForgeryToken()

<div class="form-horizontal">
  <h4>Додати конкурсну пропозицію</h4>
  <hr />
  @Html.ValidationSummary(true, "", new { @class = "text-danger" })
  <div class="form-group">
    @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-control" } })
      @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.SpecialityCode, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
      @Html.EditorFor(model => model.SpecialityCode, new { htmlAttributes = new { @class = "form-control" } })
      @Html.ValidationMessageFor(model => model.SpecialityCode, "", new { @class = "text-danger" })
    </div>
  </div>

  <div class="form-group">
    @Html.LabelFor(model => model.NumberofPlaces, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.NumberofPlaces, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NumberofPlaces, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EducationLevel, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EducationLevel, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.EducationLevel, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Додати" class="btn btn-default" />
            </div>
        </div>
    </div>
    }

    <div>
        @Html.ActionLink("Відмінити введення і повернутись до списку", "Index")
    </div>
```

Програмний код скрипта OffersCrudDelete

```
    @model UniversitiesInfo.Web.Models.OfferEditingModel

    @{
        ViewBag.Title = "Delete";
    }

    <h3>Ви дійсно хочете видалити цю пропозицію?</h3>
    <div>
        <h4>Видалення даних</h4>
        <hr />
        <dl class="dl-horizontal">
            <dt>
                @Html.DisplayNameFor(model => model.UniversityName)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.UniversityName)
            </dd>

            <dt>
                @Html.DisplayNameFor(model => model.SpecialityCode)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.SpecialityCode)
            </dd>

            <dt>
                @Html.DisplayNameFor(model => model.NumberofPlaces)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.NumberofPlaces)
```

```
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EducationLevel)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Видалити" class="btn btn-default" /> |
            @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
        </div>
    }
</div>
```

Програмний код скрипта OffersCrudDetails

```
@model UniversitiesInfo.Web.Models.OfferEditingModel

@{
    ViewBag.Title = "Details";
}

<div>
    <h4>Конкурсна пропозиція</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.SpecialityCode)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.SpecialityCode)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.NumberofPlaces)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumberofPlaces)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EducationLevel)
        </dt>
```

```
            <dd>
                @Html.DisplayFor(model => model.EducationLevel)
            </dd>

        </dl>
    </div>
    <p>
        @Html.ActionLink("Редагувати", "Edit", new { id = Model.Id }) |
        @Html.ActionLink("Повернутись до списку", "Index")
    </p>
```

Програмний код скрипта OffersCrudEdit

```
    @model UniversitiesInfo.Web.Models.OfferEditingModel

    @{
        ViewBag.Title = "Edit";
    }

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Редагування даних</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)
        <p>
            @Html.ActionLink("Повернутись до списку", "Index")
        </p>
        <div class="form-group">
            @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.SpecialityCode, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.SpecialityCode, new { htmlAttributes = new { @class = "form-control"
} })
                @Html.ValidationMessageFor(model => model.SpecialityCode, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.NumberofPlaces, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumberofPlaces, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.NumberofPlaces, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.EducationLevel, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
              @Html.EditorFor(model => model.EducationLevel, new { htmlAttributes = new { @class = "form-control" } })
              @Html.ValidationMessageFor(model => model.EducationLevel, "", new { @class = "text-danger" })
            </div>
          </div>

          <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
              <input type="submit" value="Зберегти" class="btn btn-default" />
            </div>
          </div>
        </div>
        }
```

Програмний код скрипта OffersCrudIndex

```
        @model IEnumerable<UniversitiesInfo.Web.Models.OfferViewModel>

        @{
          ViewBag.Title = "Index";
        }

        <h3>Редагування даних</h3>
        <h3>
          @Html.ActionLink("Назад", "Index", "Admin")
        </h3>

        <p>
          @Html.ActionLink("Додати конкурсну пропозицію", "Create",
                null, new { @class = "btn btn-default" })
        </p>
        <table class="table">
          <tr>
            <th>
              @Html.DisplayNameFor(model => model.UniversityName)
            </th>
            <th>
              @Html.DisplayNameFor(model => model.SpecialityCode)
            </th>
            <th>
              @Html.DisplayNameFor(model => model.NumberofPlaces)
            </th>
            <th>
              @Html.DisplayNameFor(model => model.EducationLevel)
            </th>
            <th></th>
          </tr>

          @foreach (var item in Model)
          {
        <tr>
          <td>
            @Html.ActionLink(item.UniversityName, "Details", new { id = item.Id })
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.SpecialityCode)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.NumberofPlaces)
          </td>
```

```
    <td>
        @Html.DisplayFor(modelItem => item.EducationLevel)
    </td>
    <td>
        @Html.ActionLink("Редагувати", "Edit", new { id = item.Id }) |
        @Html.ActionLink("Видалити", "Delete", new { id = item.Id })
    </td>
</tr>
    }
</table>
```

Програмний код скрипта Shared _CategoryLayout

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <link href="~/Content/Styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="navbar navbar-inverse" role="navigation">
        <a class="navbar-brand hidden-xs" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-xs" href="#">Інформаційна система ЗВО України</a>
    </div>
    <div class="row panel">
        <div id="categories" class="col-sm-5 col-md-4 col-lg-3">
            @RenderSection("NavMenu", false)
        </div>
        <div class="col-sm-7 col-md-8 col-lg-9">
            @RenderBody()
        </div>
    </div>

    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

Програмний код скрипта Shared _Layout

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <script src="~/Scripts/modernizr-2.6.2.js"></script>
    <link href="~/Content/Styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="navbar navbar-inverse" role="navigation">
        <a class="navbar-brand hidden-xs" href="/">
            Інформаційна система ЗВО України
        </a>
        <a class="navbar-brand visible-xs" href="#">Інформаційна система ЗВО України</a>
    </div>
    <div class="row panel">
```

```
        <div id="categories" class="col-sm-5 col-md-4 col-lg-3">
            @RenderSection("NavMenu", false)
        </div>
        <div class="col-sm-7 col-md-8 col-lg-9">
            @RenderBody()
        </div>
    </div>

    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

Програмний код скрипта Shared _NoteBlock

```
@model string

@if (string.IsNullOrWhiteSpace(@Model))
{
    @:(дані відсутні)
}
else
{
    <blockquote>
        <p class="lead">
            @Model
        </p>
    </blockquote>
}
```

Програмний код скрипта Shared _NoteInfo

```
@model IEnumerable<string>

@if (@Model != null)
{
    <hr />
    foreach (var m in @Model)
    {
        <p class="lead">@m</p>
    }
    <hr />
}
```

Програмний код скрипта Universities _BrowseData

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@foreach (var m in @Model)
{
    <div class="row panel">
        <div class="col-xs-12">
            <h3>
                <strong>@m.UniversityName</strong>
            </h3>
            <strong class="lead">Ідентифікаційний код: @m.Code</strong><br />
            <strong class="lead">Тип закладу: @m.Type</strong><br />
            <strong class="lead">Область: @m.Area</strong><br />
            <strong class="lead">Населений пункт: @m.Settlement</strong><br />
            <strong class="lead">Адреса: @m.Address</strong><br />
            <strong class="lead">Телефон: @m.Phone</strong><br />
            <strong class="lead">Електронна адреса: @m.Email</strong><br />
            <strong class="lead">Веб-сайт: @m.Website</strong><br />
        </div>
    </div>
}
```

Програмний код скрипта Universities _Details

```
@model UniversitiesInfo.Web.Models.UniversityViewModel

@{
    ViewBag.Title = "Details";
}

<h2>Детальна інформація</h2>

<div>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Code)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Code)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Type)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Type)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Area)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Area)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Settlement)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Settlement)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Address)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Address)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Phone)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Phone)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Website)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Website)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Назад", "List")
</p>
```

Програмний код скрипта Universities _SelectionForm

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))
{
    <p>
        Назва закладу<br />
        <input type="text" name="selUniversityName" />
    </p>
    <p>
        Ідентифікаційний код<br />
        <input type="text" name="selCode" />
    </p>
    <p>
        Тип закладу<br />
        <input type="text" name="selType" />
    </p>
    <p>
        Область<br />
        @Html.DropDownList("selArea", ViewBag.selArea as SelectList)
    </p>
    <p>
        Населений пункт<br />
        <input type="text" name="selSettlement" />
    </p>
    <p>
        <input type="submit" value="Відібрати" />
```

```
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта Universities _TableBody

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>


@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.UniversityName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Code)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Type)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Area)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Settlement)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Address)
        </td>
    </tr>
}
```

Програмний код скрипта Universities _TableHead

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

<tr>
    <th>
        @Html.DisplayNameFor(model => model.UniversityName)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Code)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Type)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Area)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Settlement)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Address)
    </th>
    <th></th>
</tr>
```

Програмний код скрипта UniversitiesBrowseByAreas

```
@using UniversitiesInfo.Web.Controllers
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>
```

```
@{
   ViewBag.Title = "BrowseByAreas";
}

<h3>
   @Html.ActionLink("Назад", "Index")
</h3>

<div class="ajaxLink">
   @foreach (string area in @ViewBag.Areas as IEnumerable<string>)
   {
      @Ajax.ActionLink(
            area,
            "_GetDataByArea",
            new { selArea = area },
            new AjaxOptions
            {
               UpdateTargetId = "data",
               LoadingElementId = "loading",
               LoadingElementDuration = 1000
            },
            new { @class = "btn btn-default" }
         )
   }
</div>
<div id="loading" class="load" style="display:none">
   <p>Завантаження даних...</p>
</div>

<div id="data">
   @Html.Action("_GetDataByArea",
         new { selNumber = @UniversitiesController.ALL_PAGE_LINK_NAME })
</div>
```

Програмний код скрипта UniversitiesIndex

```
@{
   ViewBag.Title = "Index";
}

<h2>Інформація про заклади освіти</h2>


<div class="row panel">
   @Html.ActionLink("Список всіх закладів",
            "List", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Заклади освіти за областями",
            "UniversitiesByAreasInfo", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Відбір даних про заклади освіти",
            "Selection", "Universities", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
<div class="row panel">
   @Html.ActionLink("Назад",
```

```
            "EntitiesList", "Home", null,
            new { @class = "btn btn-block btn-default btn-lg" })
</div>
```

Програмний код скрипта UniversitiesList

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
   ViewBag.Title = "List";
}

<h2>Заклади освіти</h2>

<h3>
   @Html.ActionLink("Назад", "Index")
</h3>

<table class="table">
   <tr>
      <th>
         @Html.DisplayNameFor(model => model.UniversityName)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Code)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Type)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Area)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Settlement)
      </th>
      <th>
         @Html.DisplayNameFor(model => model.Address)
      </th>
      <th></th>
   </tr>

   @foreach (var item in Model)
   {
      <tr>
         <td>
            @Html.DisplayFor(modelItem => item.UniversityName)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Code)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Type)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Area)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Settlement)
         </td>
         <td>
            @Html.DisplayFor(modelItem => item.Address)
         </td>
```

```
            <td>
                @Html.ActionLink("Детально", "_Details", new { id = item.Id })
            </td>
        </tr>
    }
</table>
```

Програмний код скрипта UniversitiesRecommendation

```
@using (Ajax.BeginForm(
    "_SelectData",
        new AjaxOptions
        {
            UpdateTargetId = "data",
            LoadingElementId = "loading",
            LoadingElementDuration = 1000
        }
        ))

{
    <h2>Введіть оцінки з ЗНО для рекомендації спеціальностей</h2>
    <p>
        Математика<br />
        <input type="text" name="math" />
    </p>
    <p>
        Фізика<br />
        <input type="text" name="physics" />
    </p>
    <p>
        Українська мова<br />
        <input type="text" name="ukrainian" />
    </p>
    <p>
        Українська література<br />
        <input type="text" name="ukrliterature" />
    </p>
    <p>
        Історія України<br />
        <input type="text" name="historyofUkraine" />
    </p>
    <p>
        Біологія<br />
        <input type="text" name="biology" />
    </p>
    <p>
        Географія<br />
        <input type="text" name="geography" />
    </p>
    <p>
        Хімія<br />
        <input type="text" name="chemistry" />
    </p>
    <p>
        Англійська мова<br />
        <input type="text" name="english" />
    </p>
    <p>
        Німецька мова<br />
        <input type="text" name="deutsch" />
    </p>
    <p>
```

```
        Іспанська мова<br />
        <input type="text" name="spanish" />
    </p>
    <p>
        Французька мова<br />
        <input type="text" name="french" />
    </p>
    <p>
        <input type="submit" value="Аналізувати" />
    </p>
    <div id="loading" class="load" style="display:none">
        <p>Завантаження даних...</p>
    </div>
}
```

Програмний код скрипта UniversitiesSelection

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "Selection";
}

<div class="row panel">
    <div id="parameters" class="col-sm-2">
        <h3>Параметри</h3>
        @Html.Partial("_SelectionForm")
        <h3>
            @Html.ActionLink("Назад", "Index")
        </h3>
    </div>
    <div class="col-sm-10">
        <table class="table">
            <thead>
                @Html.Partial("_TableHead", @Model)
            </thead>
            <tbody id="data">
                @Html.Partial("_TableBody", @Model)
            </tbody>
        </table>
    </div>
</div>
```

Програмний код скрипта UniversitiesUniversitiesByAreasInfo

```
@model IEnumerable<UniversitiesInfo.Domain.University>

@{
    ViewBag.Title = "UniversitiesByAreasInfo";
    Layout = "~/Views/Shared/_CategoryLayout.cshtml";
}

<h2>Заклади освіти по областях</h2>
<h3>
    @Html.ActionLink("Назад", "Index")
</h3>


<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
```

```
        <th>
            @Html.DisplayNameFor(model => model.Code)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Type)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Area)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Settlement)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Address)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UniversityName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Code)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Type)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Area)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Settlement)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Address)
            </td>
            <td>
                @Html.ActionLink("Детально", "_Details", new { id = item.Id })
            </td>
        </tr>
    }
</table>

@section NavMenu {
    @Html.Action("UniversitiesByAreasMenu", "Navigation",
            new { categoryName = ViewBag.SelectedCategoryName })
}
```

Програмний код скрипта UniversitiesCrud Create

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Create";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
```

```
        <div class="form-horizontal">
            <h4>Додати заклад освіти</h4>
            <hr />
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            <div class="form-group">
                @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-md-2"
})
                <div class="col-md-10">
                    @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
                    @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Code, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Code, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Code, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Type, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Type, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Type, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Area, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownList("Area", ViewBag.Area as SelectList)
                    @Html.ValidationMessageFor(model => model.Area, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Settlement, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Settlement, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Settlement, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Phone, htmlAttributes: new { @class = "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Phone, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Phone, "", new { @class = "text-danger" })
                </div>
```

```
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.Website, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.Website, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Website, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Додати" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Відмінити введення і повернутись до списку", "Index")
</div>
```

Програмний код скрипта UniversitiesCrud Delete

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Delete";
}

<h3>Ви дійсно бажаєте видалити цей запис?</h3>
<div>
    <h4>@Model.UniversityName: видалення даних</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.UniversityName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.UniversityName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Code)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Code)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Type)
        </dt>
```

```
    <dd>
        @Html.DisplayFor(model => model.Type)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Area)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Area)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Settlement)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Settlement)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Address)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Address)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Phone)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Phone)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Email)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Email)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Website)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Website)
    </dd>
</dl>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-actions no-color">
        <input type="submit" value="Видалити" class="btn btn-default" /> |
        @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
```

```
    </div>
  }
</div>
```

Програмний код скрипта UniversitiesCrud Details

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
  ViewBag.Title = "Details";
}

<div>
  <h4>@Model.UniversityName: перегляд даних</h4>
  <hr />
  <dl class="dl-horizontal">
    <dt>
      @Html.DisplayNameFor(model => model.UniversityName)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.UniversityName)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Code)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Code)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Type)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Type)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Area)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Area)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Settlement)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Settlement)
    </dd>

    <dt>
      @Html.DisplayNameFor(model => model.Address)
    </dt>

    <dd>
      @Html.DisplayFor(model => model.Address)
```

```
            </dd>

            <dt>
                @Html.DisplayNameFor(model => model.Phone)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.Phone)
            </dd>

            <dt>
                @Html.DisplayNameFor(model => model.Email)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.Email)
            </dd>

            <dt>
                @Html.DisplayNameFor(model => model.Website)
            </dt>

            <dd>
                @Html.DisplayFor(model => model.Website)
            </dd>

        </dl>
    </div>
    <p>
        @Html.ActionLink("Редагувати", "Edit", new { id = Model.Id }) |
        @Html.ActionLink("Повернутись до списку", "Index")
    </p>
```

Програмний код скрипта UniversitiesCrud Edit

```
@model UniversitiesInfo.Web.Models.UniversityEditingModel

@{
    ViewBag.Title = "Edit";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

<div class="form-horizontal">
    <h4>@Model.UniversityName: редагування даних</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.Id)

    <div class="form-group">
        @Html.LabelFor(model => model.UniversityName, htmlAttributes: new { @class = "control-label col-md-2"
})
        <div class="col-md-10">
            @Html.EditorFor(model => model.UniversityName, new { htmlAttributes = new { @class = "form-
control" } })
            @Html.ValidationMessageFor(model => model.UniversityName, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
```

```
    @Html.LabelFor(model => model.Code, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Code, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Code, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Type, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Type, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Type, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Area, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.DropDownList("Area", ViewBag.Area as SelectList)
        @Html.ValidationMessageFor(model => model.Area, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Settlement, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Settlement, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Settlement, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Address, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Address, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Address, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Phone, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Phone, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Phone, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    @Html.LabelFor(model => model.Website, htmlAttributes: new { @class = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.EditorFor(model => model.Website, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Website, "", new { @class = "text-danger" })
    </div>
```

```
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Зберегти" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Відмінити зміни і повернутись до списку", "Index")
</div>
```

Програмний код скрипта UniversitiesCrud Index

```
@model IEnumerable<UniversitiesInfo.Web.Models.UniversityViewModel>

@{
    ViewBag.Title = "Index";
}

<h3>Редагування даних про заклади освіти</h3>
<h3>
    @Html.ActionLink("Назад", "Index", "Admin")
</h3>

<p>
    @Html.ActionLink("Додати новий заклад", "Create",
          null, new { @class = "btn btn-default" })
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.UniversityName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Code)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Type)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Area)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Settlement)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Address)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
<tr>
    <td>
        @Html.ActionLink(item.UniversityName, "Details", new { id = item.Id })
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.Code)
```

```
            </td>
            <td>
              @Html.DisplayFor(modelItem => item.Type)
            </td>
            <td>
              @Html.DisplayFor(modelItem => item.Area)
            </td>
            <td>
              @Html.DisplayFor(model => item.Settlement)
            </td>
            <td>
              @Html.DisplayFor(model => item.Address)
            </td>
            <td>
              @Html.ActionLink("Редагувати", "Edit", new { id = item.Id }) |
              @Html.ActionLink("Видалити", "Delete", new { id = item.Id })
            </td>
          </tr>
            }
          </table>
```

Програмний код скрипта Web.config

```
          <?xml version="1.0"?>

          <configuration>
            <configSections>
              <sectionGroup name="system.web.webPages.razor" type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">
                <section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
                <section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection, System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
              </sectionGroup>
            </configSections>

            <system.web.webPages.razor>
              <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.2.4.0, 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="UniversitiesInfo.Web" />
                </namespaces>
              </pages>
            </system.web.webPages.razor>

            <appSettings>
              <add key="webpages:Enabled" value="false" />
            </appSettings>

            <system.webServer>
              <handlers>
                <remove name="BlockViewHandler"/>
                <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode" type="System.Web.HttpNotFoundHandler" />
```

```
      </handlers>
    </system.webServer>

    <system.web>
      <compilation>
        <assemblies>
          <add assembly="System.Web.Mvc, Version=5.2.4.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" />
        </assemblies>
      </compilation>
    </system.web>
  </configuration>
```

Додаток Г – Графічна частина

ГРАФІЧНА ЧАСТИНА
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИБОРУ СПЕЦІАЛЬНОСТЕЙ В ЗАКЛАДАХ ВИЩОЇ ОСВІТИ УКРАЇНИ

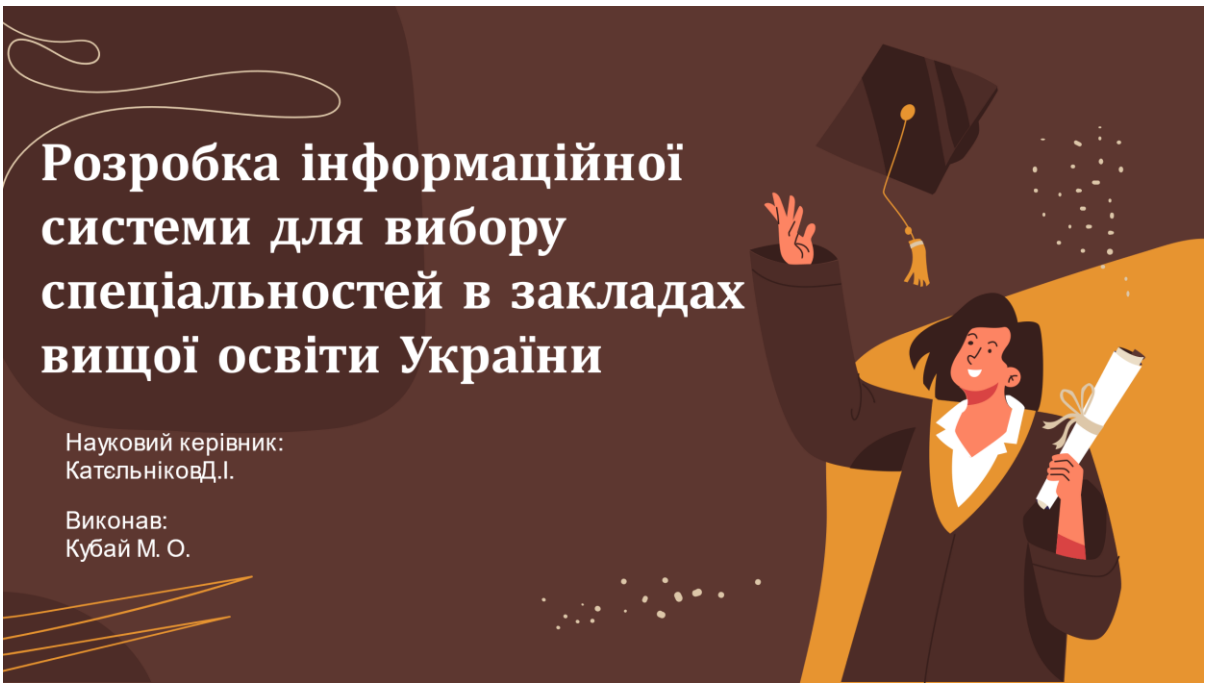


Рисунок Г.1 – Назва роботи

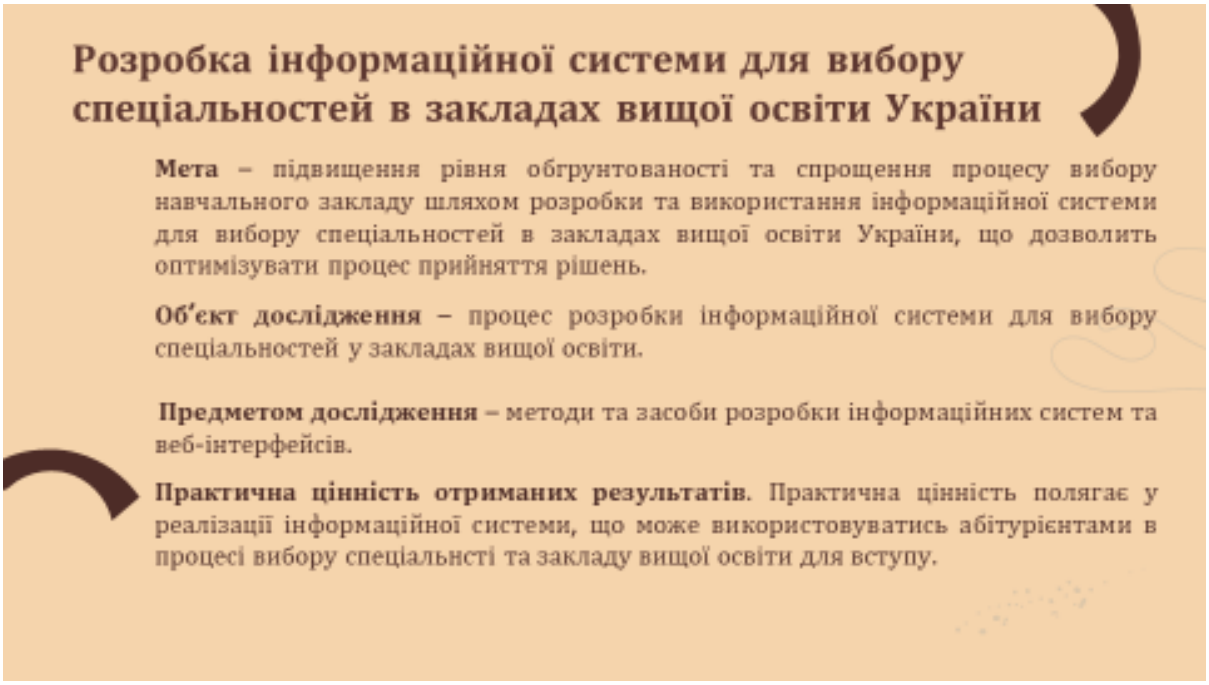


Рисунок Г.2 – Мета, об'єкт і предмет дослідження

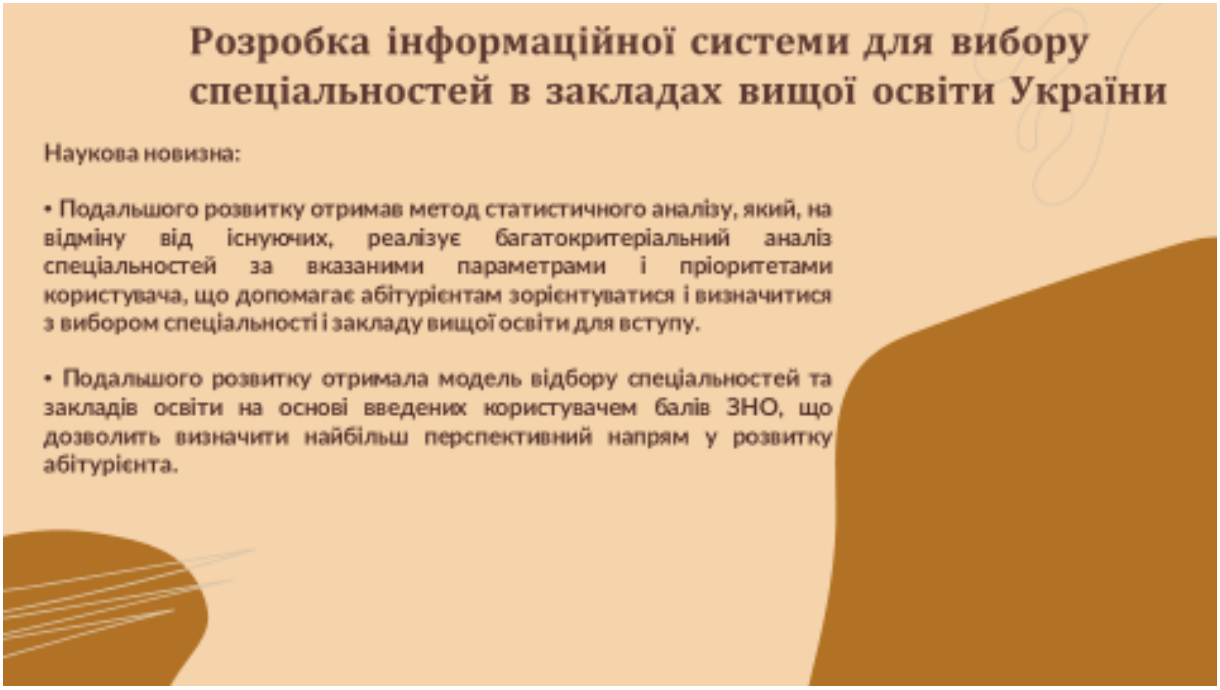


Рисунок Г.3 – Наукова новизна

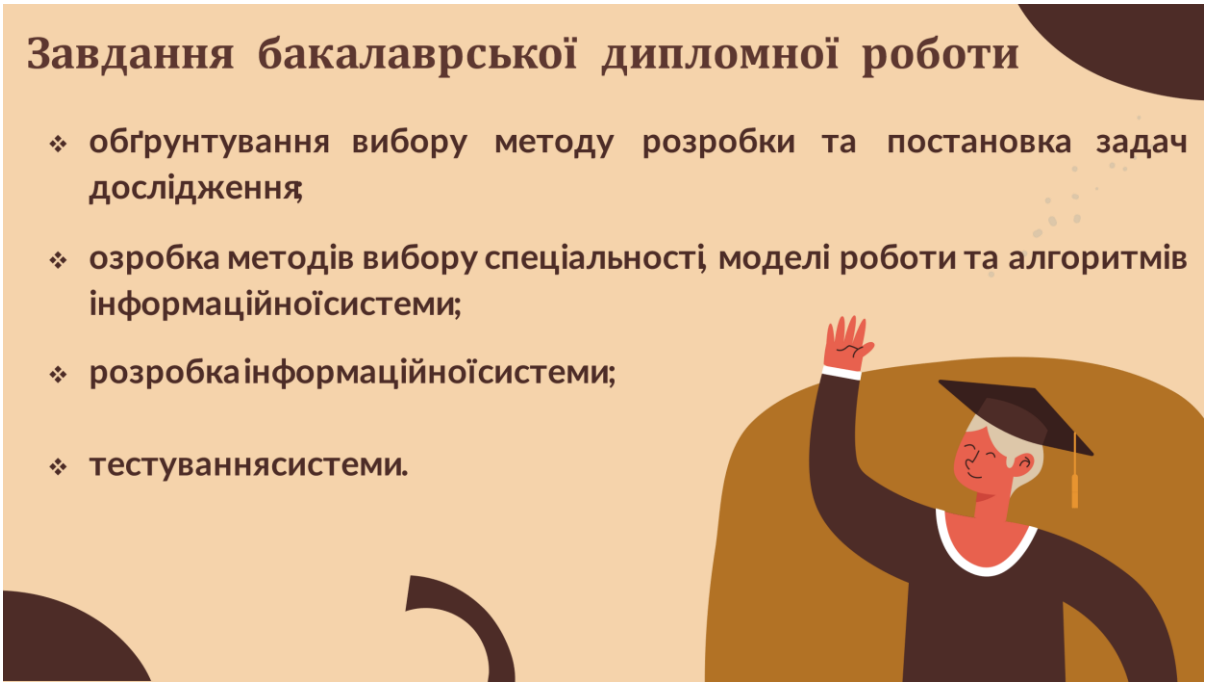
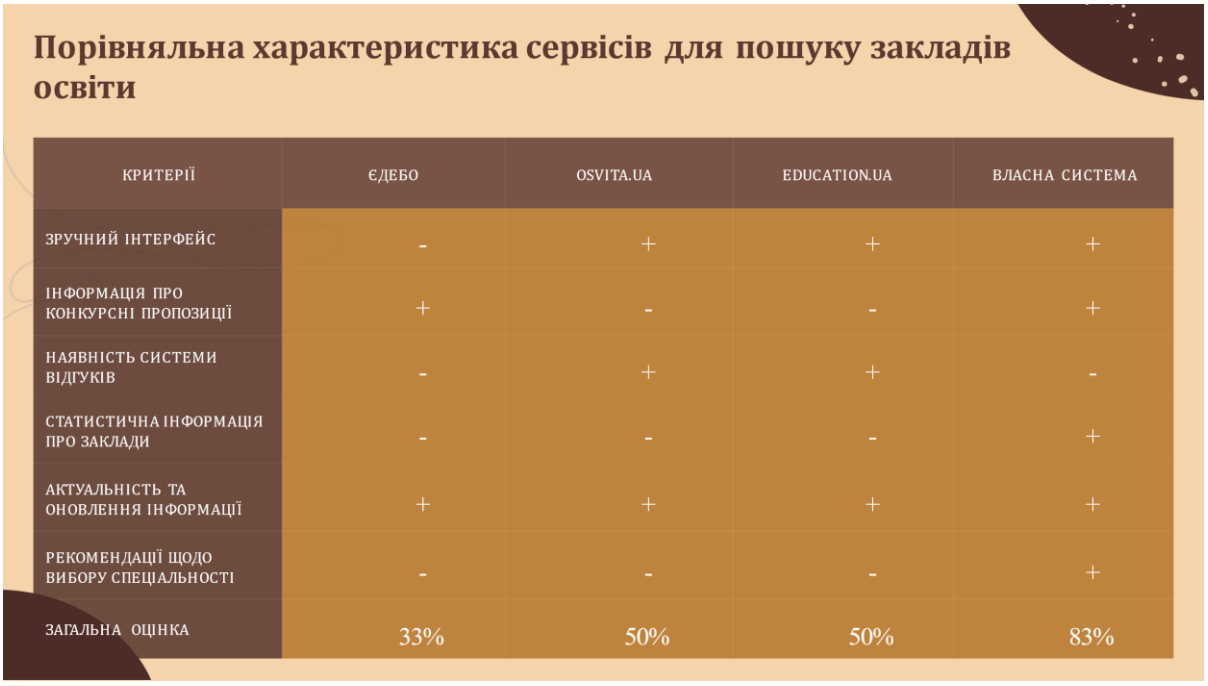


Рисунок Г.4 – Задачі бакалаврської дипломної роботи

Рисунок Г.5 – Аналоги



КРИТЕРІЇ	ЄДЕБО	OSVITA.UA	EDUCATION.UA	ВЛАСНА СИСТЕМА
ЗРУЧНИЙ ІНТЕРФЕЙС	-	+	+	+
ІНФОРМАЦІЯ ПРО КОНКУРСНІ ПРОПОЗИЦІЇ	+	-	-	+
НАЯВНІСТЬ СИСТЕМИ ВІДГУКІВ	-	+	+	-
СТАТИСТИЧНА ІНФОРМАЦІЯ ПРО ЗАКЛАДИ	-	-	-	+
АКТУАЛЬНІСТЬ ТА ОНОВЛЕННЯ ІНФОРМАЦІЇ	+	+	+	+
РЕКОМЕНДАЦІЇ ЩОДО ВИБОРУ СПЕЦІАЛЬНОСТІ	-	-	-	+
ЗАГАЛЬНА ОЦІНКА	33%	50%	50%	83%

Рисунок Г.6 – Порівняльна характеристика сервісів для пошуку закладів освіти

Метод роботи інформаційної системи

1. Користувач отримує список всіх можливих представлень даних (а саме списком всіх закладів освіти та конкурсних пропозицій з можливістю фільтрації за окремим параметрами).
2. Далі він може обрати необхідне представлення або відредагувати дані при необхідності.
3. Обравши перегляд даних, користувач обирає в якому саме вигляді йому потрібно знайти дані: переглянути весь список закладів освіти, чи знайти конкретний заклад, використовуючи фільтри, чи переглянути конкурсні пропозиції.
4. Якщо користувач не знайшов всі дані, або не може визначитись з вибором закладу – він може перейти до модулю рекомендації спеціальностей, внести туди всі результати ЗНО, після чого отримає результат у вигляді списку рекомендованих спеціальностей.
5. Якщо результат його влаштовує, спеціальності його задовільняють – можна розглядати конкурсні пропозиції на обрану спеціальність, після чого подавати документи до вступу.

Рисунок Г.7 – Метод роботи інформаційної системи

Блок- схема загального алгоритму системи

Рисунок Г.8 – Блок-схема загального алгоритму роботи системи

Рисунок Г.9 – Блок-схема методу відбору закладів освіти та спеціальностей

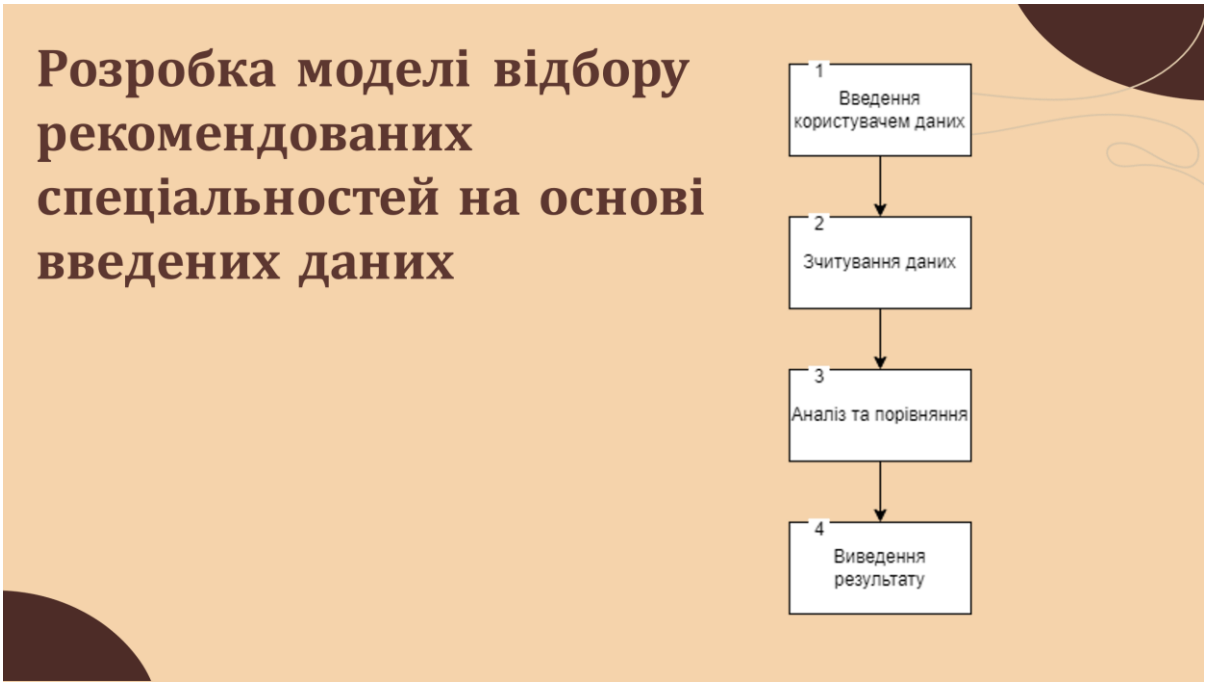


Рисунок Г.10 – Розробка моделі відбору рекомендованих спеціальностей на основі введених даних

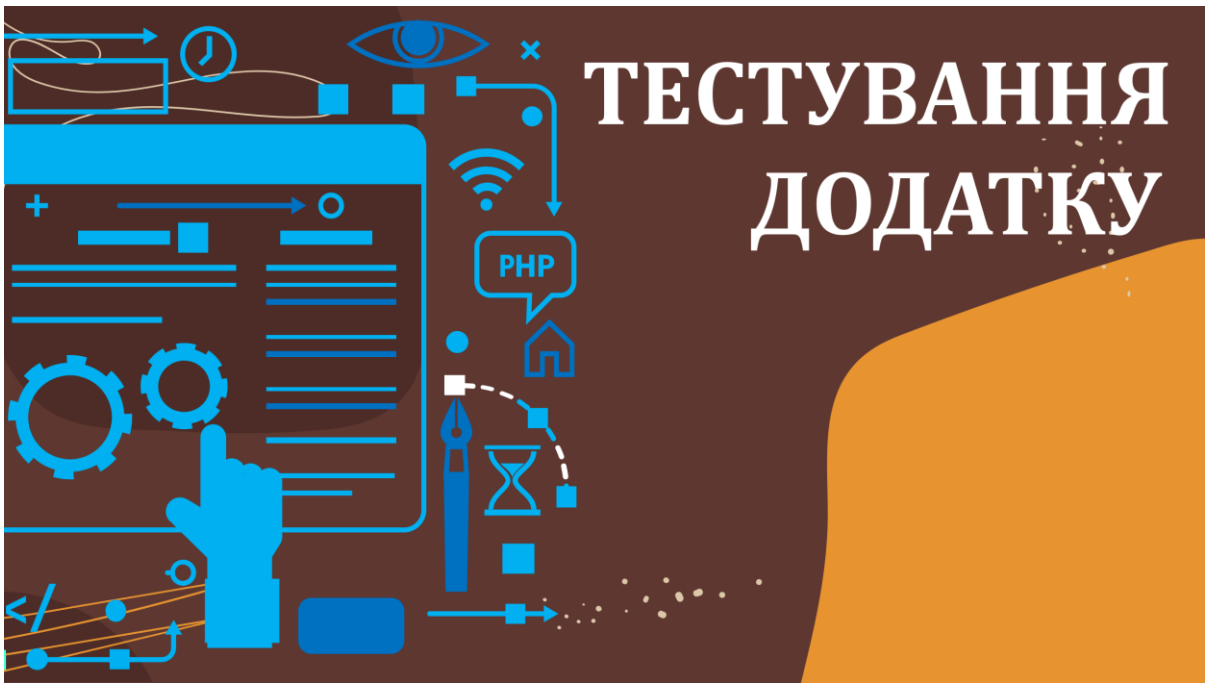


Рисунок Г.11 – Тестування додатку

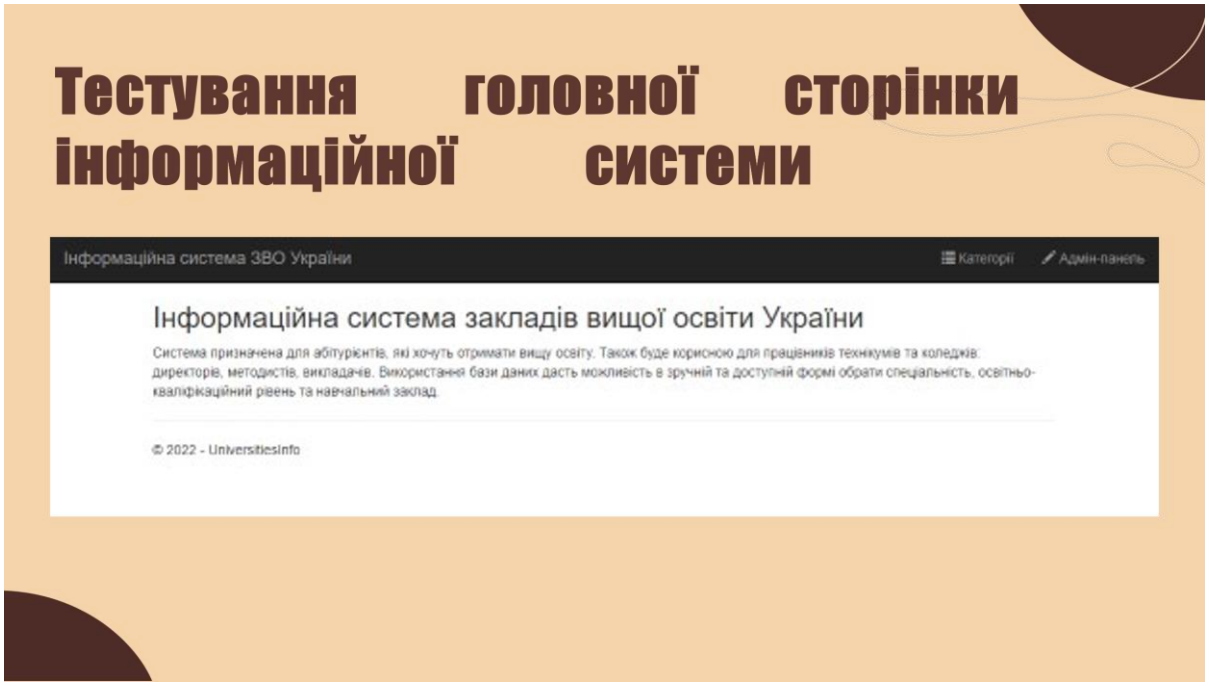


Рисунок Г.12 – Тестування головної сторінки інформаційної системи

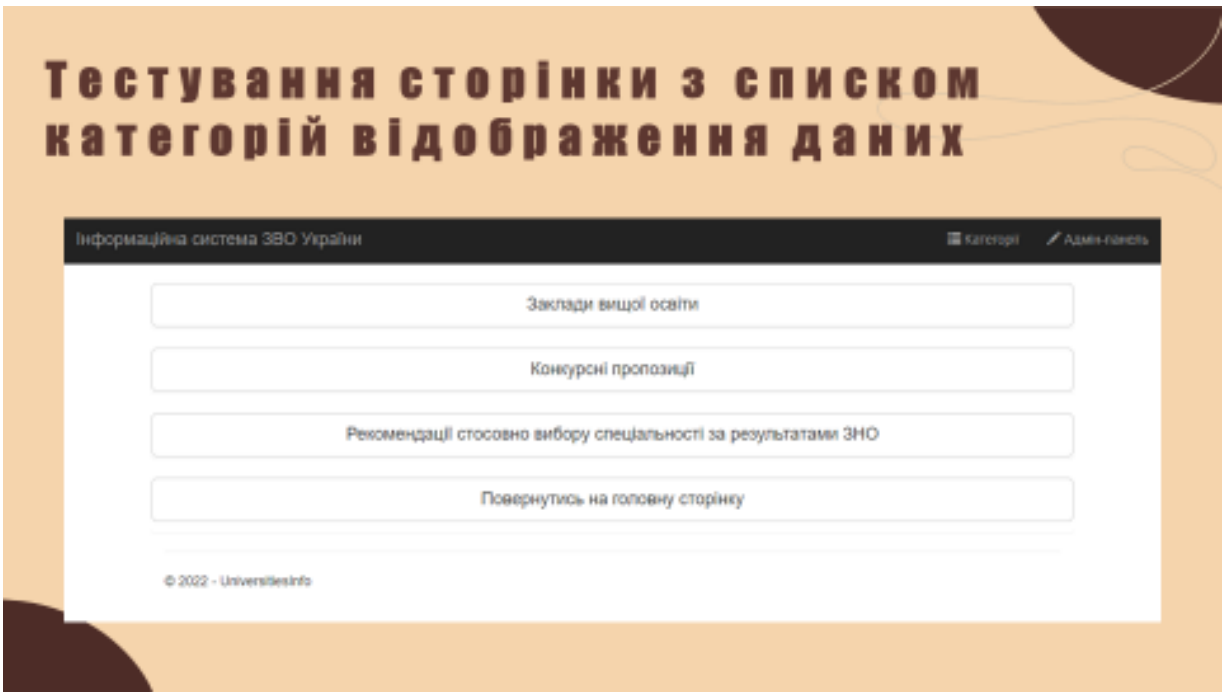


Рисунок Г.13 – Тестування сторінки з списком категорій відображення даних

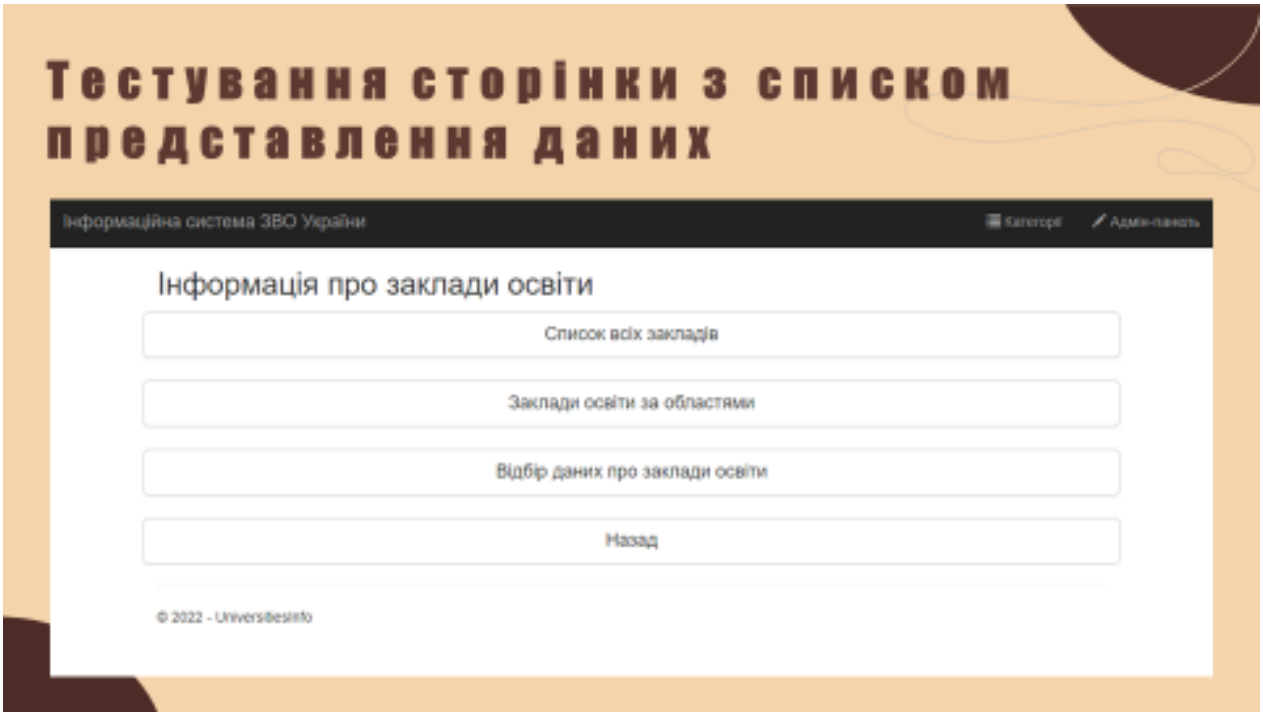


Рисунок Г.14 – Тестування сторінки з списком представлення даних

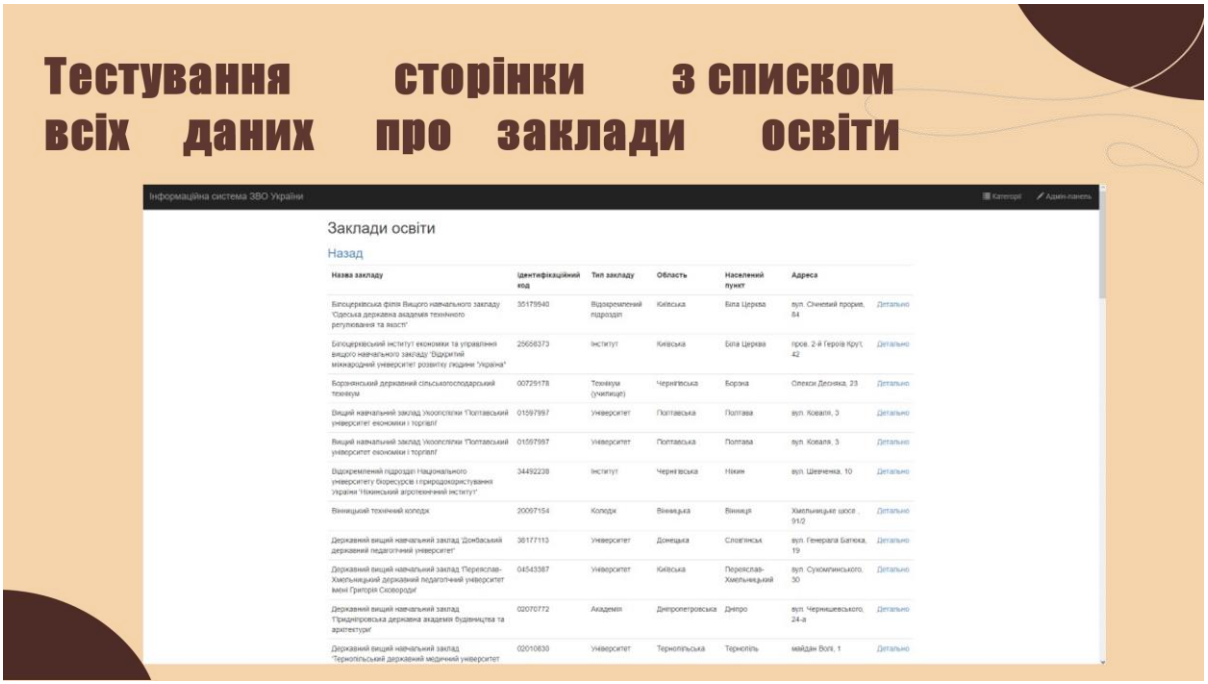


Рисунок Г.15 – Тестування сторінки зі списком всіх даних про заклади освіти

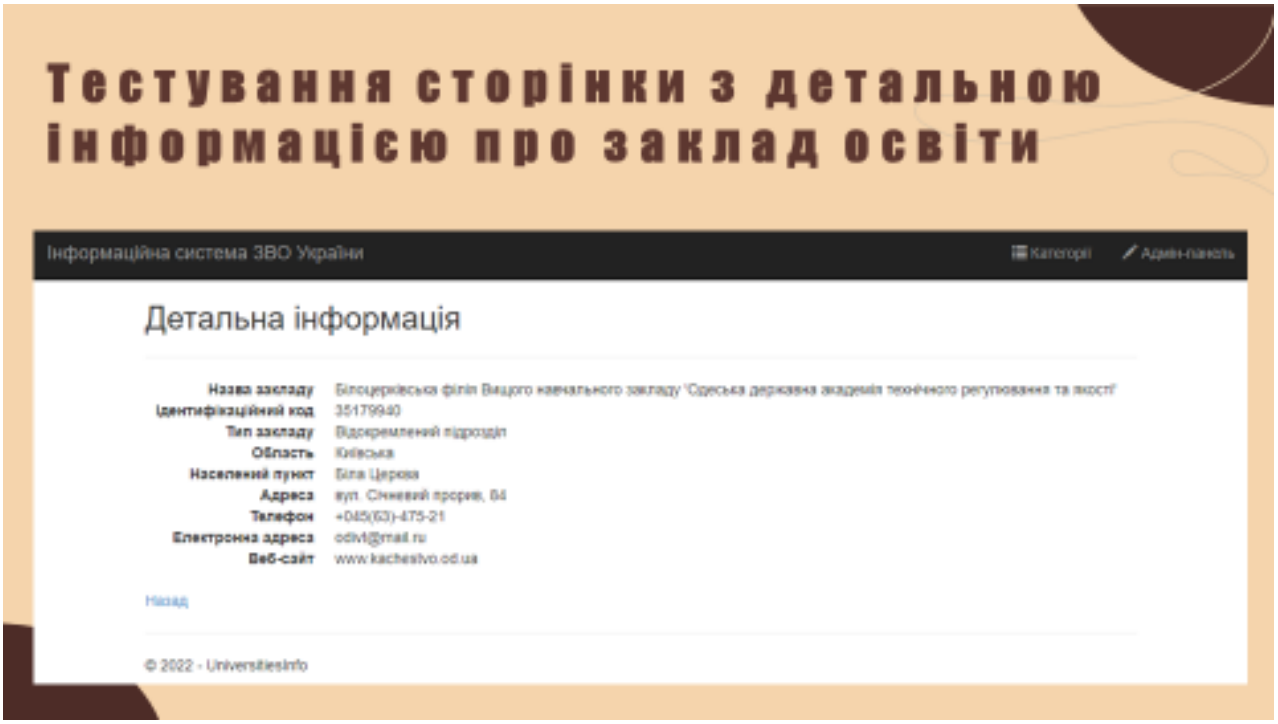


Рисунок Г.16 – Тестування сторінки з детальною інформацією про заклад освіти

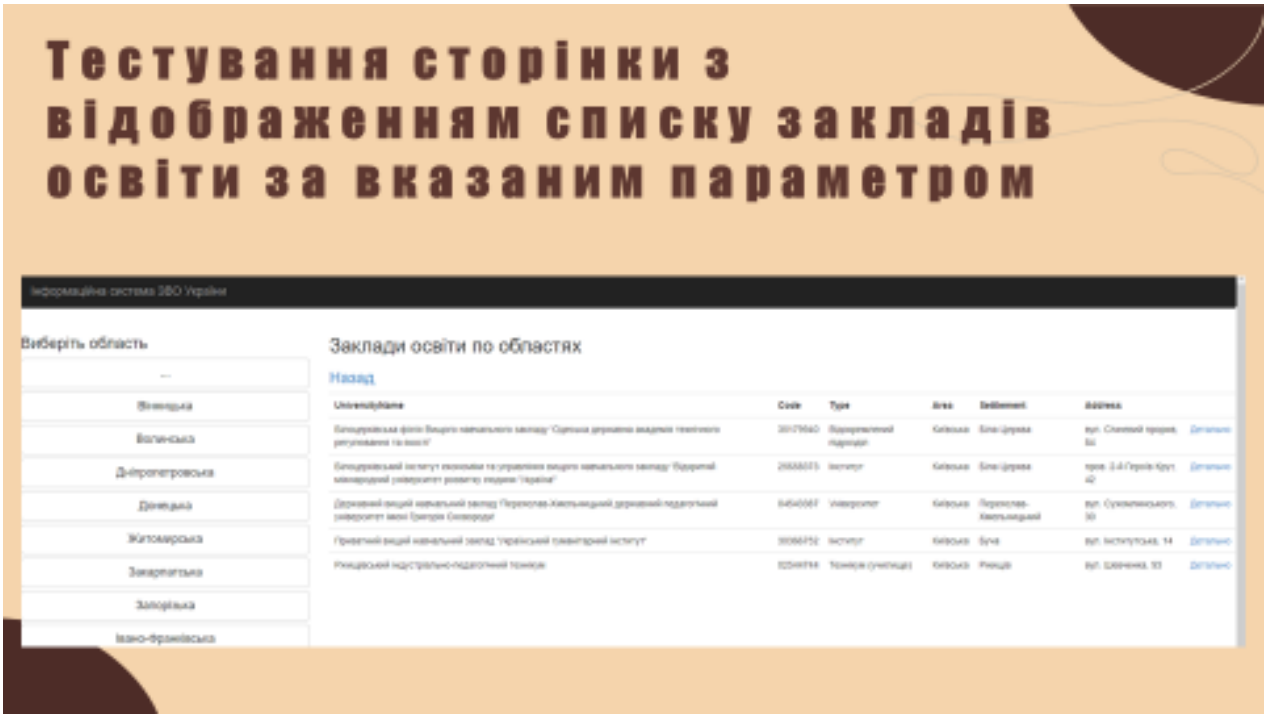


Рисунок Г.17 – Тестування сторінки з відображенням списку закладів освіти за заданим параметром

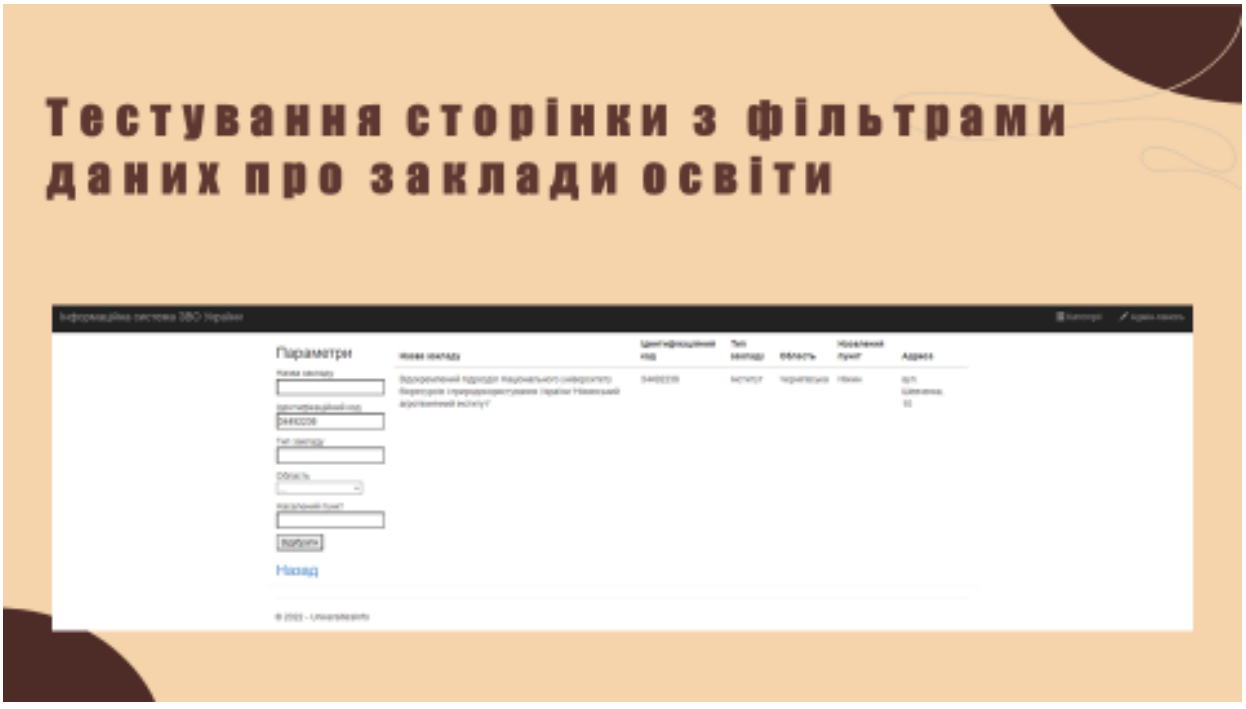


Рисунок Г.18 – Тестування сторінки з фільтрами даних про заклади освіти

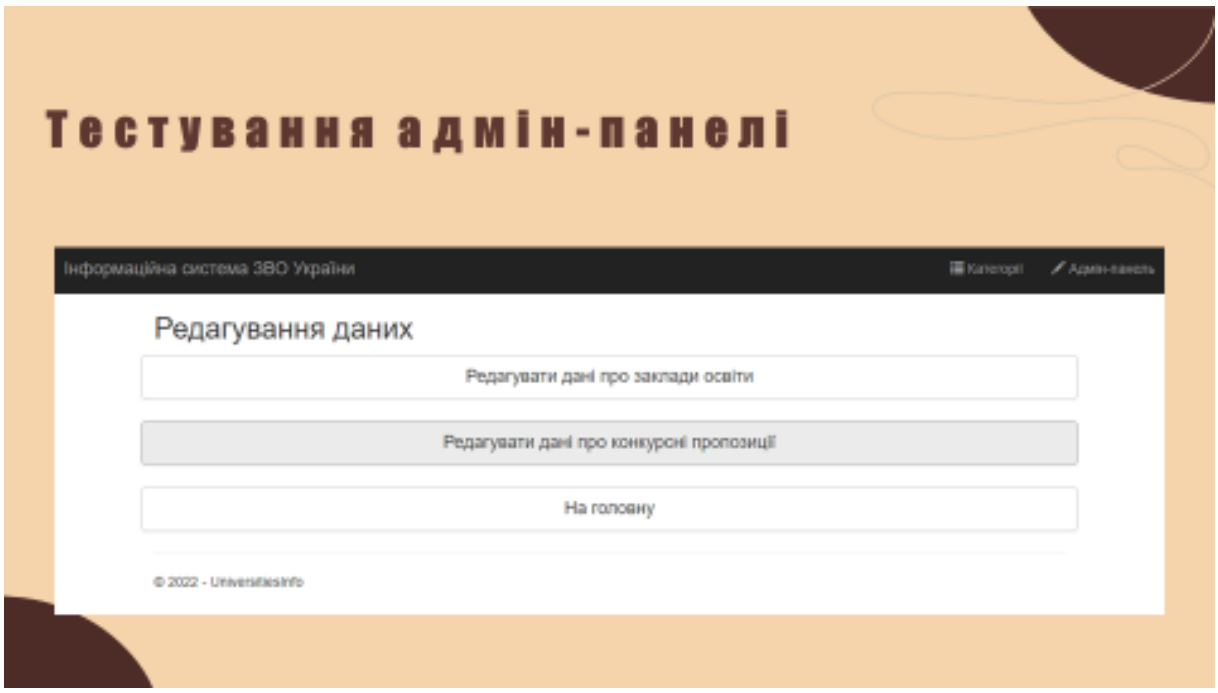


Рисунок Г.19 – Тестування адмін-панелі

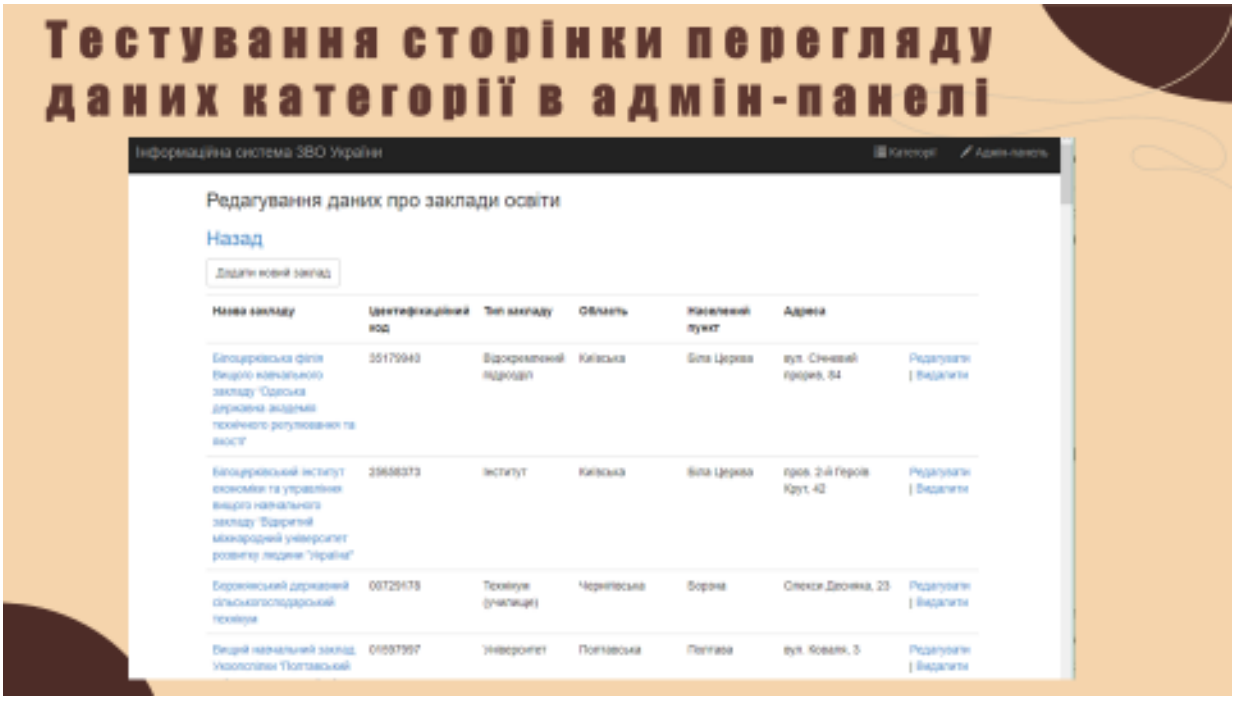


Рисунок Г.20 – Тестування сторінки перегляду даних категорії в адмін-панелі

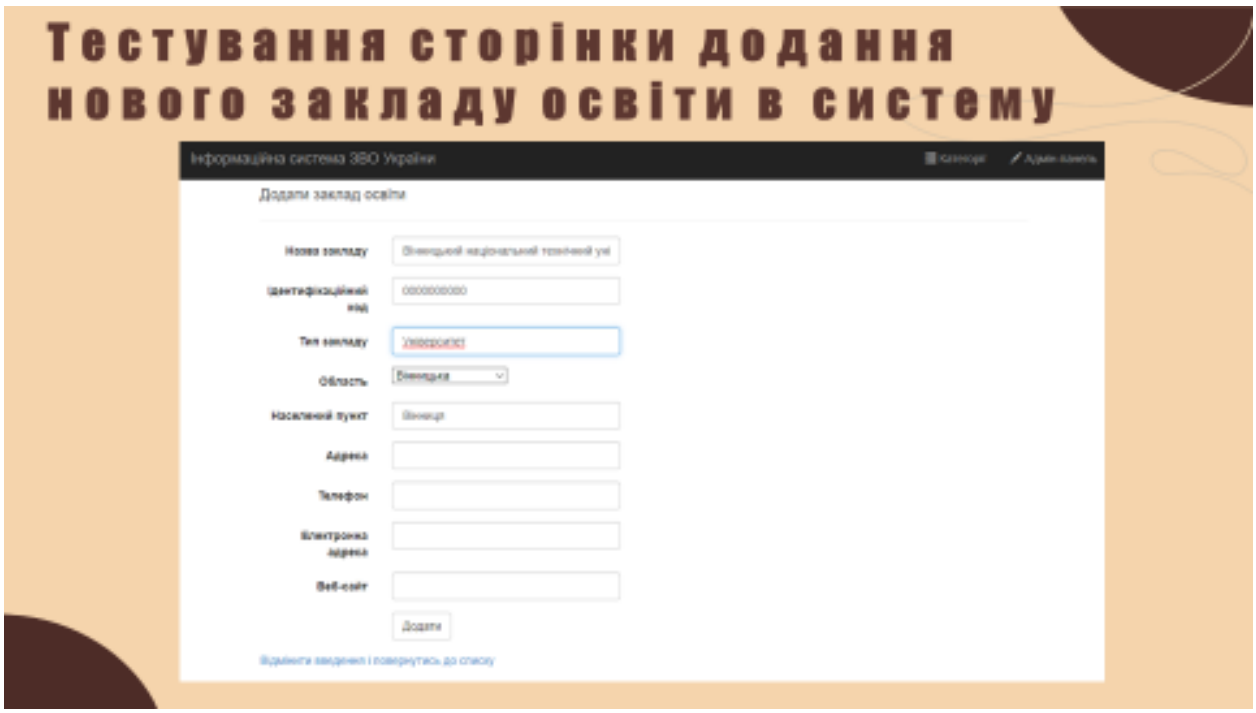


Рисунок Г.21 – Тестування сторінки додання нового закладу освіти в систему

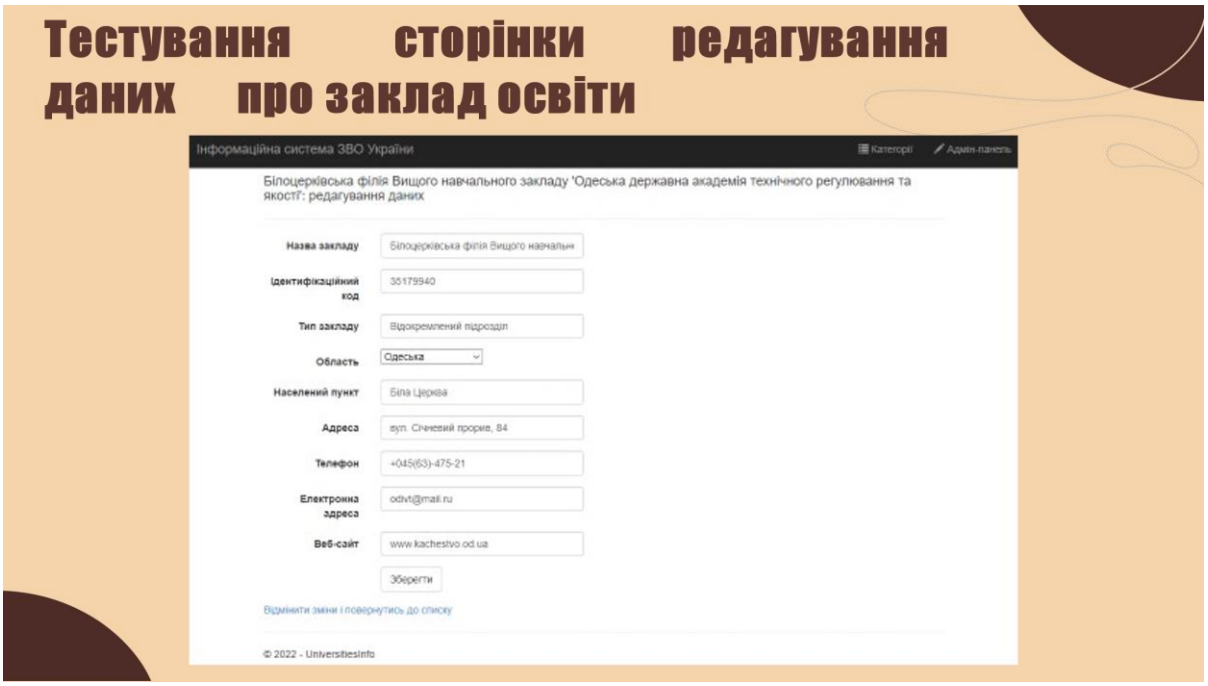


Рисунок Г.22 – Тестування сторінки редагування даних про заклад освіти

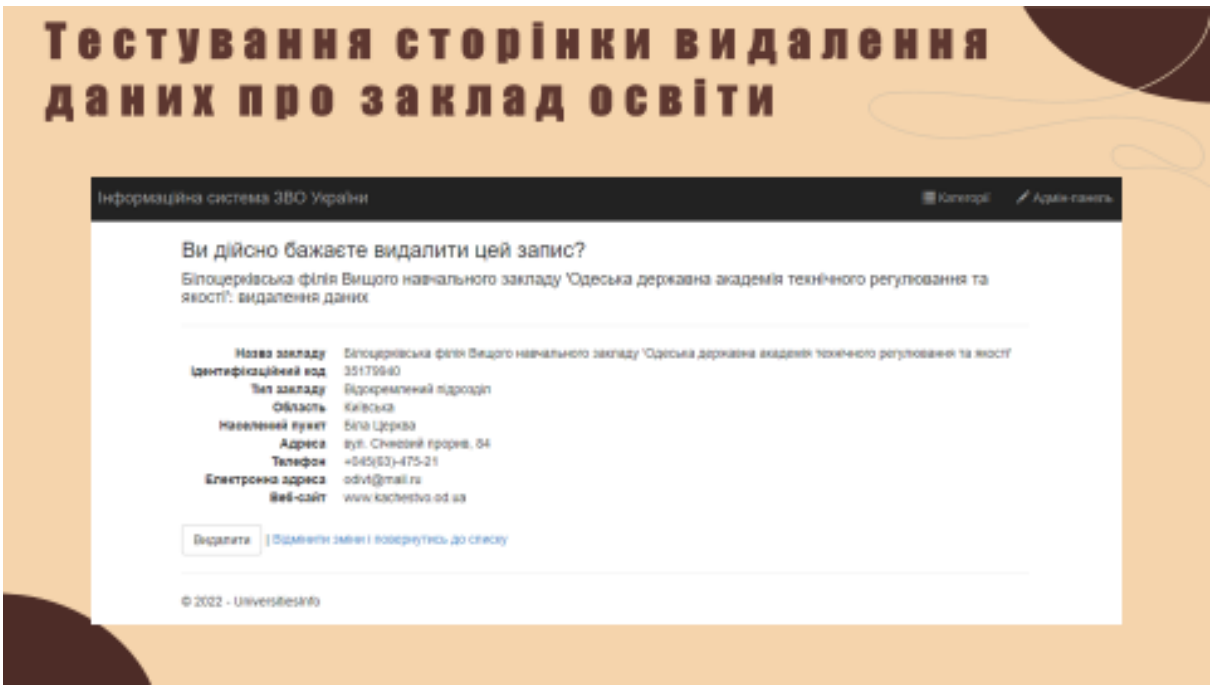


Рисунок Г.23 – Тестування сторінки видалення даних про заклад освіти

Рисунок Г.24 – Тестування повідомлення про введення некоректних даних та незаповнені поля

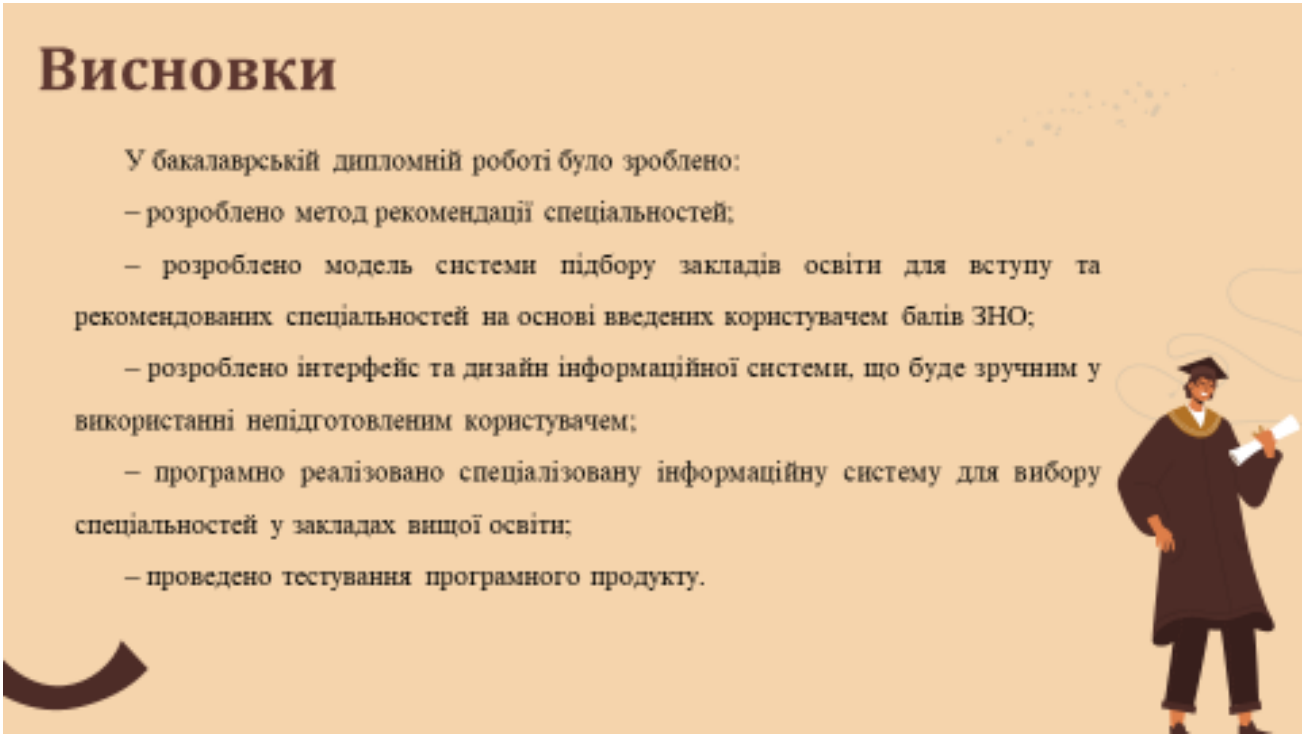


Рисунок Г.25 – Висновки

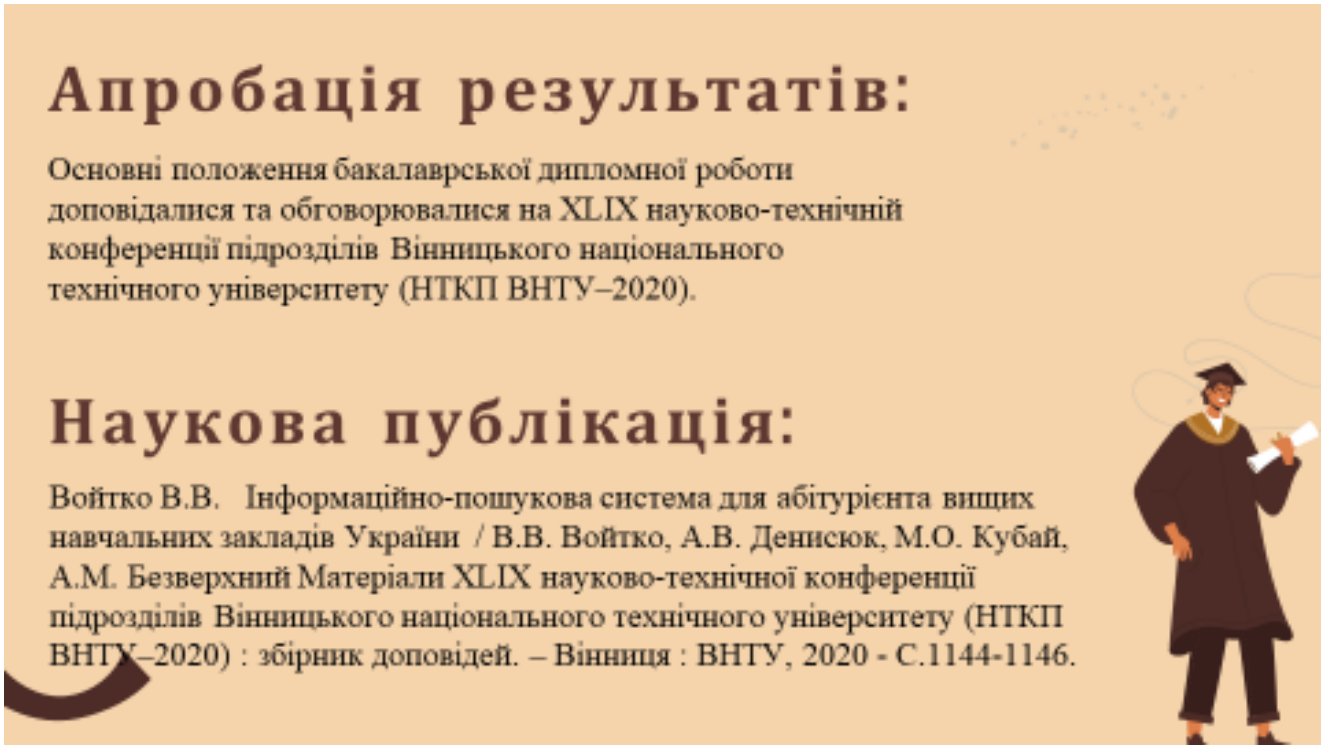


Рисунок Г.26 – Апробація результатів і публікація