

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

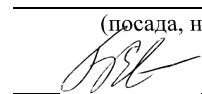
на тему: Розробка інформаційної системи для покращення роботи
приймальної комісії

Виконав: студент групи



Луценко В.Є.
(підпис)

Керівник роботи:

(посада, науковий ступінь вчене звання)


Беркунський Є.Ю.
(підпис)

м. Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студенту Луценко Володимир Євгенович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інформаційної системи для покращення роботи приймальної комісії

Керівник роботи Беркунський Є.Ю.

Затверджені наказом ректора № 270 -уч від " 31 " березня 2026 року.

2. Термін подання роботи: 20 червня 2026 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів)

1. Розробка концепції інформаційної системи для покращення роботи приймальної комісії 2. Розробка проекту інформаційної системи для покращення роботи приймальної комісії. 3. Реалізація проекту ІС 4. Охорона праці

5. Перелік презентаційних матеріалів 1 – Титульний слайд, 2 - Мета, об'єкт, предмет та актуальність дослідження, 3 - Організаційна структура приймальної комісії, 4 - Концепція інформаційної системи, 5–7 DFD-діаграми бізнес-процесів інформаційної системи, 8–10 Проектування бази даних та UML-модель системи, 11 - Алгоритм формування рейтингових списків, 12 - Використані технології розробки, 13–21 Демонстрація роботи інформаційної системи, 22 - Результати впровадження, 23 – Завершальний слайд

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунський Є.Ю.	03.02.2026	22.02.2026
2	Беркунський Є.Ю.	22.02.2026	21.03.2026
3	Беркунський Є.Ю.	21.03.2026	25.04.2026
4		25.04.2026	07.06.2026

7. Дата видачі завдання 03 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Опис об'єкта обстеження	13.02.2026	
2	Аналіз предметної галузі	15.02.2026	
3	Формування вимог до інформаційної системи	25.02.2026	
4	Розробка концепції	15.03.2026	
5	Розробка проекту ІС	15.04.2026	
6	Реалізація проекту	15.05.2026	
7	Розробка питань з охорони праці	07.06.2026	
8	Розробка графічних матеріалів	12.06.2026	
9	Подання роботи на перевірку на плагіат	10.06.2026	
10	Подання роботи рецензенту	15.06.2026	
11	Подання роботи на захист	20.06.2026	

Студент

Керівник роботи


 (підпис)

Луценко В.Є.
 (ПІБ)

Беркунський Є.Ю.
 (ПІБ)

АНОТАЦІЯ

У дипломній роботі розглянуто розробку інформаційної системи приймальної комісії закладу вищої освіти.

Об'єктом дослідження є процес організації та автоматизації роботи приймальної комісії під час проведення вступної кампанії.

Предметом дослідження є методи, моделі та програмні засоби автоматизації обробки даних про вступників, їх заяви, результати вступних випробувань, пільги, апеляції та конкурсний відбір.

Метою роботи є розробка інформаційної системи, яка забезпечує автоматизацію основних процесів вступної кампанії, зменшення обсягу ручної обробки інформації та підвищення ефективності роботи приймальної комісії.

У процесі виконання роботи проведено аналіз предметної області та особливостей організації вступної кампанії у закладах вищої освіти.

Визначено основні категорії користувачів системи, їх функціональні можливості та права доступу. Виконано аналіз інформаційних потоків предметної області, побудовано DFD-діаграми нульового, першого та другого рівнів, що відображають процеси обробки заяв, апеляцій, спеціальностей та формування звітності.

У роботі розроблено концептуальну, логічну та фізичну моделі бази даних, визначено основні сутності предметної області, їх атрибути та взаємозв'язки. Виконано проектування структури бази даних для зберігання інформації про вступників, спеціальності, факультети, результати вступних випробувань, пільги, заяви та апеляції.

Розроблено математичне забезпечення системи, що включає алгоритми розрахунку конкурсного бала, формування рейтингових списків вступників, врахування пріоритетів заяв та пільгових категорій під час конкурсного відбору.

Виконано реалізацію програмного забезпечення із використанням мови програмування Java, фреймворку Spring, системи управління базами даних MySQL та технологій створення web-додатків. Реалізовано функції реєстрації

вступників, подання та обробки заяв, керування спеціальностями, формування звітів, подання та розгляду апеляцій, а також розмежування прав доступу між різними категоріями користувачів.

У результаті виконання роботи створено інформаційну систему приймальної комісії, яка забезпечує автоматизацію основних процесів вступної кампанії та може бути використана для підтримки діяльності приймальної комісії закладу вищої освіти.

Дипломна робота викладена на 98 сторінках машинописного тексту, містить 10 рисунків, 4 таблиці та 18 джерел інформації.

Ключові слова: інформаційна система, приймальна комісія, вступна кампанія, автоматизація, конкурсний відбір, конкурсний бал, рейтинговий список, база даних, web-додаток, апеляція, Java, Spring.

ABSTRACT

This diploma thesis is devoted to the development of an information system for the admissions committee of a higher education institution.

The object of research is the process of organizing and automating the activities of an admissions committee during the admission campaign.

The subject of research is methods, models, and software tools for automating the processing of applicant data, applications, entrance examination results, benefits, appeals, and competitive selection procedures.

The purpose of the thesis is to develop an information system that automates the main processes of the admission campaign, reduces the amount of manual data processing, and increases the efficiency of the admissions committee.

During the course of the research, an analysis of the subject area and the specific features of organizing the admission campaign in higher education institutions was carried out. The main categories of system users, their functional capabilities, and access rights were identified. The information flows of the subject area were analyzed, and DFD diagrams of the zero, first, and second levels were developed to represent the processes of application processing, appeal management, specialty management, and report generation.

The thesis includes the development of conceptual, logical, and physical database models. The main entities of the subject area, their attributes, and relationships were identified. A database structure was designed to store information about applicants, specialties, faculties, entrance examination results, benefits, applications, and appeals.

Mathematical support of the system was developed, including algorithms for calculating the competitive score, generating applicant ranking lists, and considering application priorities and preferential categories during the admission process.

The software implementation was carried out using the Java programming language, the Spring framework, the MySQL database management system, and web application development technologies. The developed system provides

applicant registration, application submission and processing, specialty management, report generation, appeal submission and review, as well as access control for different categories of users.

As a result of the work, an information system for an admissions committee was developed. The system automates the main processes of the admission campaign and can be used to support the activities of admissions committees in higher education institutions.

The thesis is presented on 98 pages of typewritten text, contains 10 figures, 4 tables and 18 sources of information.

Keywords: information system, admissions committee, admission campaign, automation, competitive selection, competitive score, ranking list, database, web application, appeal, Java, Spring.

Зміст

ВСТУП.....	9
Розділ 1 – Розробка концепції інформаційної системи для покращення роботи приймальної комісії	12
1.1 Опис об'єкту обстеження.....	12
1.2 Аналіз предметної області.....	16
1.3 Формування вимог до інформаційної системи	18
1.4 Розробка концепції інформаційної системи для покращення роботи приймальної комісії	21
Розділ 2 – Розробка проекту інформаційної системи для покращення роботи приймальної комісії	25
2.1 Загальносистемні рішення.....	25
2.3 Рішення з математичного забезпечення	36
2.4 Рішення з програмного забезпечення	37
2.5 Рішення з технічного забезпечення.....	39
Розділ 3 – реалізація проекту ІС	41
3.1 Керівництво користувача системи	41
3.2 Результати експлуатації системи.....	49
Розділ 4 – Охорона праці	52
4.1 Аналіз умов праці користувача інформаційної системи.....	52
4.2 Вимоги до мікроклімату та освітлення	52
4.3 Електробезпека та пожежна безпека.....	53
4.4 Організація режиму праці та відпочинку	53
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ.....	59
Додаток А – технічне завдання.....	59
Додаток Б – фрагменти програмного коду	61
Додаток В – реалізація програмної частини та результати роботи системи ..	85

ВСТУП

Розробка інформаційної системи виконана за темою «Інформаційна система підтримки роботи приймальної комісії закладу вищої освіти».

Актуальність теми полягає у необхідності підвищення ефективності організації вступної кампанії в умовах значного обсягу інформації, що обробляється приймальною комісією. Процес вступу супроводжується прийомом та обробкою великої кількості заяв вступників, формуванням конкурсних списків, контролем плану набору, розподілом місць за спеціальностями та веденням звітної інформації. Використання традиційних або частково автоматизованих методів обробки даних може призводити до збільшення часу обробки інформації, виникнення помилок та ускладнення контролю за ходом вступної кампанії.

Метою даної роботи є підвищення ефективності роботи приймальної комісії за рахунок автоматизації процесів обробки заяв вступників, формування конкурсних списків, контролю плану набору, обліку пільгових категорій та підготовки звітної інформації.

Задача роботи полягає у розробці інформаційної системи, яка забезпечить автоматизацію основних процесів вступної кампанії, централізоване зберігання та обробку даних, а також підтримку взаємодії між користувачами системи та інформаційними ресурсами.

Наукова новизна роботи полягає у запропонованому підході до побудови інформаційної системи підтримки вступної кампанії, в якій узгоджено процеси роботи із заявами вступників, формування конкурсних списків та обробки інформації, необхідної для забезпечення діяльності приймальної комісії.

У першому розділі виконано розробку концепції інформаційної системи для покращення роботи приймальної комісії. У підрозділі 1.1 наведено опис об'єкта дослідження, організаційної структури приймальної комісії та

основних проблем її функціонування. У підрозділі 1.2 проведено аналіз предметної галузі та існуючих процесів обробки інформації під час вступної кампанії. У підрозділі 1.3 сформовано основні вимоги до інформаційної системи. У підрозділі 1.4 розроблено концепцію інформаційної системи та побудовано модель інформаційної взаємодії функціональних задач системи.

У другому розділі виконано розробку проекту інформаційної системи для покращення роботи приймальної комісії. У підрозділі 2.1 наведено загальносистемні рішення щодо архітектури системи, програмних засобів, технологій реалізації та організації взаємодії між компонентами системи. У підрозділі 2.2 розроблено рішення з інформаційного забезпечення, побудовано DFD-діаграми, концептуальну, логічну та фізичну моделі бази даних. У підрозділі 2.3 наведено рішення з математичного забезпечення, зокрема алгоритми розрахунку конкурсного бала, формування рейтингових списків та врахування пріоритетів заяв і пільгових категорій вступників. У підрозділі 2.4 обґрунтовано вибір програмних засобів та технологій реалізації інформаційної системи. У підрозділі 2.5 наведено рішення з технічного забезпечення та визначено вимоги до апаратних засобів, необхідних для функціонування системи.

У третьому розділі наведено результати реалізації інформаційної системи. У підрозділі 3.1 описано користувацький інтерфейс системи та порядок роботи користувачів із її функціональними можливостями. У підрозділі 3.2 наведено результати експлуатації та тестування розробленого програмного забезпечення, що підтверджують працездатність реалізованих функцій та відповідність системи поставленим вимогам.

У четвертому розділі розглянуто питання охорони праці під час роботи з розробленою інформаційною системою. У підрозділі 4.1 виконано аналіз умов праці користувачів інформаційної системи та наведено вимоги до організації робочого місця користувача персонального комп'ютера. У підрозділі 4.2 розглянуто вимоги до параметрів мікроклімату та освітлення приміщень, обладнаних комп'ютерною технікою. У підрозділі 4.3 наведено

основні вимоги електробезпеки та пожежної безпеки під час експлуатації комп'ютерного обладнання. У підрозділі 4.4 розглянуто питання організації режиму праці та відпочинку користувачів, спрямовані на зниження втоми та забезпечення безпечних умов праці під час роботи з інформаційною системою.

Після основної частини роботи наведено висновки, у яких узагальнено результати проведених досліджень, проектування та реалізації інформаційної системи, а також сформульовано основні результати, отримані під час виконання дипломної роботи.

У додатках наведено матеріали, що доповнюють основний зміст роботи. Зокрема, представлено сформоване технічне завдання на розробку інформаційної системи, фрагменти програмного коду, що реалізують основні функціональні можливості системи, SQL-скрипти створення бази даних, програмний код JPA-сущностей, а також скріншоти роботи інформаційної системи, які демонструють реалізацію користувацького інтерфейсу та результати функціонування розробленого програмного забезпечення.

Розділ 1 – Розробка концепції інформаційної системи для покращення роботи приймальної комісії

1.1 Опис об'єкту обстеження

На сучасному етапі розвитку економіки та технологій особливого значення набуває підготовка висококваліфікованих фахівців. Ринок праці постійно потребує спеціалістів у різних галузях діяльності, тому заклади вищої освіти відіграють важливу роль у забезпеченні суспільства професійними кадрами. Якість підготовки спеціалістів значною мірою залежить від ефективності організації вступної кампанії, адже саме на цьому етапі відбувається відбір абітурієнтів, які у майбутньому стануть студентами закладу вищої освіти.

Діяльність закладу вищої освіти охоплює широкий спектр освітніх та організаційних процесів, серед яких важливе місце займає формування контингенту студентів. Організація та проведення вступної кампанії є складним управлінським процесом, що включає прийом заяв абітурієнтів, обробку документів, проведення вступних випробувань, формування рейтингових списків та прийняття рішень щодо зарахування.

Безпосереднє управління процесом прийому кандидатів здійснює приймальна комісія. Вона організовує прийом документів, координує роботу структурних підрозділів закладу освіти під час вступної кампанії, забезпечує проведення конкурсного відбору та приймає рішення щодо зарахування вступників.

Ефективність роботи приймальної комісії має важливе значення для забезпечення якісного набору студентів, оскільки саме від результатів її діяльності залежить формування майбутнього контингенту здобувачів освіти.

Організаційна структура приймальної комісії визначається відповідними нормативними документами та включає кілька рівнів управління. Загальне керівництво діяльністю комісії здійснює її голова, яким є керівник закладу вищої освіти. До складу комісії також входять заступники голови, відповідальний секретар, заступники відповідального секретаря,

уповноважена особа з питань прийняття та розгляду електронних заяв, а також члени приймальної комісії, серед яких можуть бути директори інститутів, декани факультетів та керівники структурних підрозділів закладу освіти [11]. Оперативну організацію роботи комісії забезпечує відповідальний секретар та його заступники, які відповідають за прийом і реєстрацію заяв вступників, ведення документації та координацію інформаційних процесів вступної кампанії. Члени приймальної комісії беруть участь у розгляді документів вступників, аналізують результати конкурсного відбору та формують пропозиції щодо зарахування.

Для виконання окремих функцій у структурі приймальної комісії можуть створюватися спеціалізовані підрозділи. До них належать предметні екзаменаційні комісії, комісії для проведення співбесід, фахові атестаційні комісії, предметні комісії, апеляційна комісія та відбіркові комісії. Кожна з таких підкомісій виконує визначені завдання у межах організації та проведення вступної кампанії [11].

Організаційна структура управління приймальною комісією представлена на рисунку 1.1

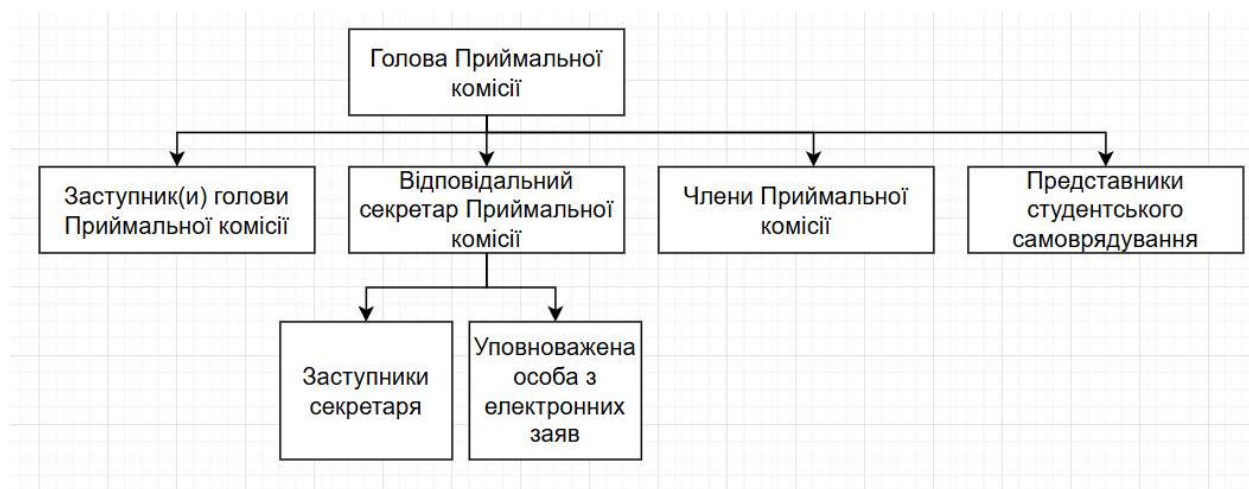


Рисунок 1.1 – Організаційна структура приймальної комісії

Крім загальної управлінської структури, у межах діяльності приймальної комісії формується система функціональних підкомісій, що забезпечують виконання окремих етапів вступної кампанії.

Для забезпечення виконання функцій приймальної комісії у її складі формуються спеціалізовані підкомісії, кожна з яких відповідає за окремі етапи вступної кампанії.

Предметні екзаменаційні комісії здійснюють організацію та проведення вступних іспитів з окремих дисциплін, а також формують результати оцінювання, які надалі використовуються при визначенні конкурсного балу вступників.

Комісії для проведення співбесід забезпечують проведення індивідуальних співбесід зі вступниками, зокрема для окремих категорій вступу або пільгових категорій, та фіксують результати проходження співбесіди.

Фахові атестаційні комісії проводять оцінювання рівня професійної підготовки вступників, які вступають на основі попередньо здобутої освіти, та формують результати фахових випробувань, що враховуються під час конкурсного відбору.

Предметні комісії відповідають за розробку програм вступних випробувань, визначення критеріїв оцінювання та методичного забезпечення процесу проведення іспитів.

Апеляційна комісія здійснює розгляд апеляцій вступників щодо результатів вступних випробувань або конкурсного відбору, приймає рішення щодо їх задоволення або відхилення та забезпечує об'єктивність оцінювання.

Відбіркові комісії виконують функції формування конкурсних та рейтингових списків вступників, здійснюють аналіз результатів вступної кампанії та готують пропозиції щодо зарахування вступників на відповідні спеціальності.

Структуру комісії наведено на рисунку 1.2



Рисунок 1.2 – Структура підкомісій приймальної комісії

У процесі організації вступної кампанії приймальна комісія опрацьовує значні обсяги інформації, що включає персональні дані вступників, результати вступних випробувань, інформацію про пільги, обрані спеціальності, план набору та статуси заяв. В умовах великого інформаційного навантаження можуть виникати певні труднощі, що знижують ефективність роботи комісії. Зокрема, процес обробки заяв вступників може відбуватися недостатньо оперативно, ускладнюється координація роботи різних структурних підрозділів, виникають труднощі з контролем виконання плану набору та формуванням статистичної звітності. Крім того, значний обсяг інформації, що підлягає обробці, створює додаткове навантаження на працівників приймальної комісії.

Наявність зазначених труднощів свідчить про необхідність більш детального дослідження інформаційних процесів, що супроводжують вступну кампанію. З метою визначення причин виникнення цих проблем та пошуку

шляхів підвищення ефективності роботи приймальної комісії доцільно провести аналіз предметної області [12].

1.2 Аналіз предметної області

Функціонування приймальної комісії закладу вищої освіти пов'язане з обробкою значних обсягів інформації, що включає персональні дані вступників, результати вступних випробувань, інформацію про пільги, обрані спеціальності, план набору та статуси поданих заяв. У таких умовах ефективність організації вступної кампанії значною мірою залежить від рівня інформаційної підтримки процесів управління.

Аналіз діяльності об'єкта управління показує, що існуючі підходи до організації інформаційних процесів спричиняють надмірні фінансові та ресурсні витрати. Значна частина операцій виконується вручну або з використанням розрізнених інструментів обробки даних, що призводить до збільшення витрат часу та ресурсів на виконання типових операцій. У результаті зростають витрати на організацію роботи приймальної комісії, знижується оперативність прийняття управлінських рішень та ускладнюється контроль за виконанням плану набору.

Однією з основних причин такої ситуації є недостатній рівень автоматизації процесів обробки інформації. Значний обсяг даних ускладнює їх оперативну систематизацію та аналіз. Використання паперових носіїв або розрізнених електронних таблиць призводить до необхідності багаторазового введення однієї і тієї ж інформації, що збільшує тривалість виконання операцій та підвищує ризик виникнення помилок.

Додатковою проблемою є трудомісткість формування конкурсних та рейтингових списків. Розрахунок конкурсного балу потребує врахування результатів вступних випробувань, коефіцієнтів, пільг і квот, що при ручному або частково автоматизованому підході вимагає значних часових витрат. У таких умовах зростає ймовірність арифметичних помилок або некоректного ранжування вступників.

Також ускладнюється контроль виконання плану набору за спеціальностями та формами навчання. За відсутності єдиного інформаційного середовища дані про кількість поданих заяв, доступні бюджетні та контрактні місця можуть оновлюватися із затримкою, що ускладнює оперативне управління вступною кампанією.

Суттєві труднощі виникають і під час обробки змін у даних вступників, зокрема при уточненні персональної інформації, розгляді апеляцій або зміні статусів заяв. За умов ручної обробки інформації такі операції потребують значних часових витрат і створюють ризик втрати або дублювання даних.

Окремою проблемою є формування статистичної та аналітичної звітності. У разі відсутності централізованої системи обліку підготовка звітів потребує додаткового узагальнення інформації з різних джерел, що значно знижує оперативність отримання необхідних даних для прийняття управлінських рішень.

Разом із цим аналіз сучасних типових автоматизованих рішень показує, що навіть при використанні спеціалізованих інформаційних систем можуть виникати певні обмеження. Зокрема, недостатня гнучкість бізнес-логіки окремих систем ускладнює оперативну зміну формул розрахунку конкурсного балу відповідно до змін нормативних вимог. Крім того, складність реалізації механізмів квотного відбору та врахування різних пільгових категорій вступників може обмежувати адаптивність таких систем до різних сценаріїв конкурсного відбору.

Ще одним обмеженням є недостатні можливості формування нестандартної аналітичної звітності. Багато типових систем підтримують лише фіксований набір звітів, що ускладнює отримання комбінованих вибірок даних або проведення детального аналізу стану вступної кампанії.

Таким чином, проведений аналіз показує, що існуючі підходи до організації інформаційних процесів під час вступної кампанії характеризуються недостатньою оперативністю обробки інформації, значними витратами ресурсів та обмеженою гнучкістю застосовуваних

інструментів. Це зумовлює необхідність удосконалення бізнес-процесів управління вступною кампанією на основі використання сучасних інформаційних технологій.

На основі проведеного аналізу можна зробити висновок, що для усунення виявлених недоліків доцільно створити інформаційну систему, яка забезпечить централізоване зберігання даних, автоматизацію обробки інформації, підтримку механізмів конкурсного відбору та формування аналітичної звітності [12].

1.3 Формування вимог до інформаційної системи

Інформаційна система підтримки вступної кампанії призначена для підвищення ефективності роботи приймальної комісії закладу вищої освіти шляхом автоматизації процесів обробки заяв вступників, формування конкурсних списків, контролю виконання плану набору та підготовки звітної інформації. Використання такої системи дозволяє скоротити час обробки інформації, зменшити ймовірність виникнення помилок при формуванні рейтингових списків та забезпечити більш оперативне прийняття управлінських рішень.

Користувачами інформаційної системи є абітурієнти та працівники приймальної комісії, кожна з яких взаємодіє із системою відповідно до своїх функціональних потреб.

Абітурієнт використовує систему для подання та супроводу власних заяв на вступ. У зв'язку з цим інформаційна система повинна забезпечувати можливість внесення та редагування персональних даних, зокрема контактної інформації (номер телефону, електронна пошта), що необхідна для комунікації та ідентифікації вступника. Система повинна забезпечувати автоматичне збереження та відображення результатів вступних випробувань і пов'язаних документів, що дозволяє уникнути повторного введення інформації. Крім того, абітурієнт повинен мати можливість подавати заяви на обрані спеціальності, переглядати їх статус та у разі необхідності подавати апеляції щодо результатів вступних випробувань або конкурсного відбору.

Працівники приймальної комісії використовують систему для обробки інформації про вступників, формування конкурсних списків та прийняття управлінських рішень. У зв'язку з цим система повинна забезпечувати швидкий доступ до інформації про вступників за різними критеріями відбору. Зокрема, працівники приймальної комісії повинні мати можливість отримати список усіх абітурієнтів, які подали заяви на обрану спеціальність, сформувати вибірку вступників за факультетом, а також отримати списки вступників за квотами пільг, зокрема за квотою 1 або квотою 2. Крім того, система повинна забезпечувати можливість формування вибірки за конкретною пільговою категорією, що дозволяє окремо аналізувати вступників, які мають певний тип пільги. Важливою функцією є можливість сортування списків вступників за конкурсним (середнім) балом для формування рейтингових списків, що використовуються при прийнятті рішень щодо зарахування. Необхідною є також можливість пошуку конкретного вступника за ключовими даними, такими як ідентифікатор, електронна пошта або номер телефону, а також перегляду всіх поданих ним заяв.

Система повинна забезпечувати формування різних типів звітів, необхідних для аналізу вступної кампанії. Зокрема, повинна бути реалізована можливість отримання звітів за всіма факультетами та спеціальностями із зазначенням кількості поданих заяв, конкурсного рейтингу та статусів вступників. Окремо повинна підтримуватися можливість формування звітів щодо плану набору, включаючи кількість бюджетних, контрактних та квотних місць, а також контроль їх заповнення в режимі реального часу.

Важливою функцією є можливість аналізу даних за попередні роки. Система повинна забезпечувати збереження історичних даних та формування звітів за минулі вступні кампанії, що дозволяє порівнювати показники набору, аналізувати динаміку та використовувати отриману інформацію для прийняття управлінських рішень.

Інформаційна система повинна функціонувати у вигляді web-додатку та забезпечувати роботу на комп'ютерах із сучасними операційними системами.

Для взаємодії із системою користувач повинен мати доступ до веб-браузера, що підтримує сучасні веб-стандарти, зокрема Google Chrome, Mozilla Firefox або Microsoft Edge. Наявність доступу до локальної мережі або мережі Інтернет забезпечує можливість взаємодії робочих місць користувачів із серверною частиною системи.

Основними функціями інформаційної системи повинні бути реєстрація та зберігання інформації про вступників, облік результатів вступних випробувань, зберігання інформації про пільги вступників, ведення довідкової інформації про факультети та спеціальності, зберігання даних про план набору, автоматичний розрахунок конкурсного балу вступників, формування рейтингових списків, зміна статусів заяв та підготовка статистичних і аналітичних звітів. Крім того, система повинна забезпечувати можливість пошуку та фільтрації інформації за різними параметрами, що дозволить підвищити ефективність роботи працівників приймальної комісії.

Інформаційна система повинна забезпечувати зберігання та обробку різних категорій інформації, зокрема персональних даних вступників, результатів вступних випробувань, інформації про спеціальності та факультети, даних про план набору, статусів заяв вступників та інформації про пільгові категорії. Уся інформація повинна зберігатися у структурованому вигляді в реляційній базі даних, що забезпечить цілісність даних, уникнення дублювання інформації та можливість швидкого доступу до необхідних записів.

Для реалізації інформаційної системи необхідно використати програмні засоби, що забезпечують розробку серверної частини, користувацького веб-інтерфейсу та системи управління базами даних. Серверна частина системи може бути реалізована із використанням мови програмування Java та фреймворку Spring, що забезпечує зручну реалізацію логіки обробки даних та взаємодії з базою даних. Для відображення інформації у веб-інтерфейсі може використовуватися шаблонізатор Thymeleaf. Для зберігання даних доцільно

використати систему управління базами даних MySQL, яка підтримує реляційну модель даних та забезпечує ефективне виконання запитів.

Функціонування інформаційної системи передбачає використання серверного обладнання та робочих місць користувачів. Серверна частина системи може бути розгорнута на одному комп'ютері, який виконуватиме функції сервера додатку та сервера бази даних. Мінімальні технічні характеристики серверного обладнання повинні включати процесор із тактовою частотою не менше 2 GHz, оперативну пам'ять обсягом не менше 8 GB, жорсткий диск обсягом не менше 256 GB та підключення до локальної мережі. Робочі місця користувачів представлені персональними комп'ютерами працівників приймальної комісії. Для роботи із системою достатньо комп'ютера з процесором рівня Intel Core i3 або аналогічним, оперативною пам'яттю не менше 4 GB, встановленим веб-браузером та доступом до мережі.

Таким чином, сформовані вимоги до інформаційної системи визначають основні характеристики її функціонування, склад програмних і технічних засобів, а також функціональні можливості, необхідні для підтримки процесів вступної кампанії. На основі сформованих вимог у подальшому може бути розроблена концепція та модель інформаційної системи підтримки вступної кампанії [7].

1.4 Розробка концепції інформаційної системи для покращення роботи приймальної комісії

На основі сформованих вимог до інформаційної системи розроблено концептуальну модель, що відображає основні функціональні задачі та інформаційні потоки в межах вступної кампанії. Запропонована модель передбачає реалізацію системи у вигляді web-орієнтованого додатку, що забезпечує взаємодію користувачів із системою шляхом формування та обробки відповідних запитів.

У межах розроблюваного проекту передбачається вирішення всіх поставлених задач, пов'язаних із організацією та проведенням вступної кампанії, на основі обробки вхідних даних та формування відповідних результатів.

Концептуальна модель інформаційної системи представлена на рисунку

1.3

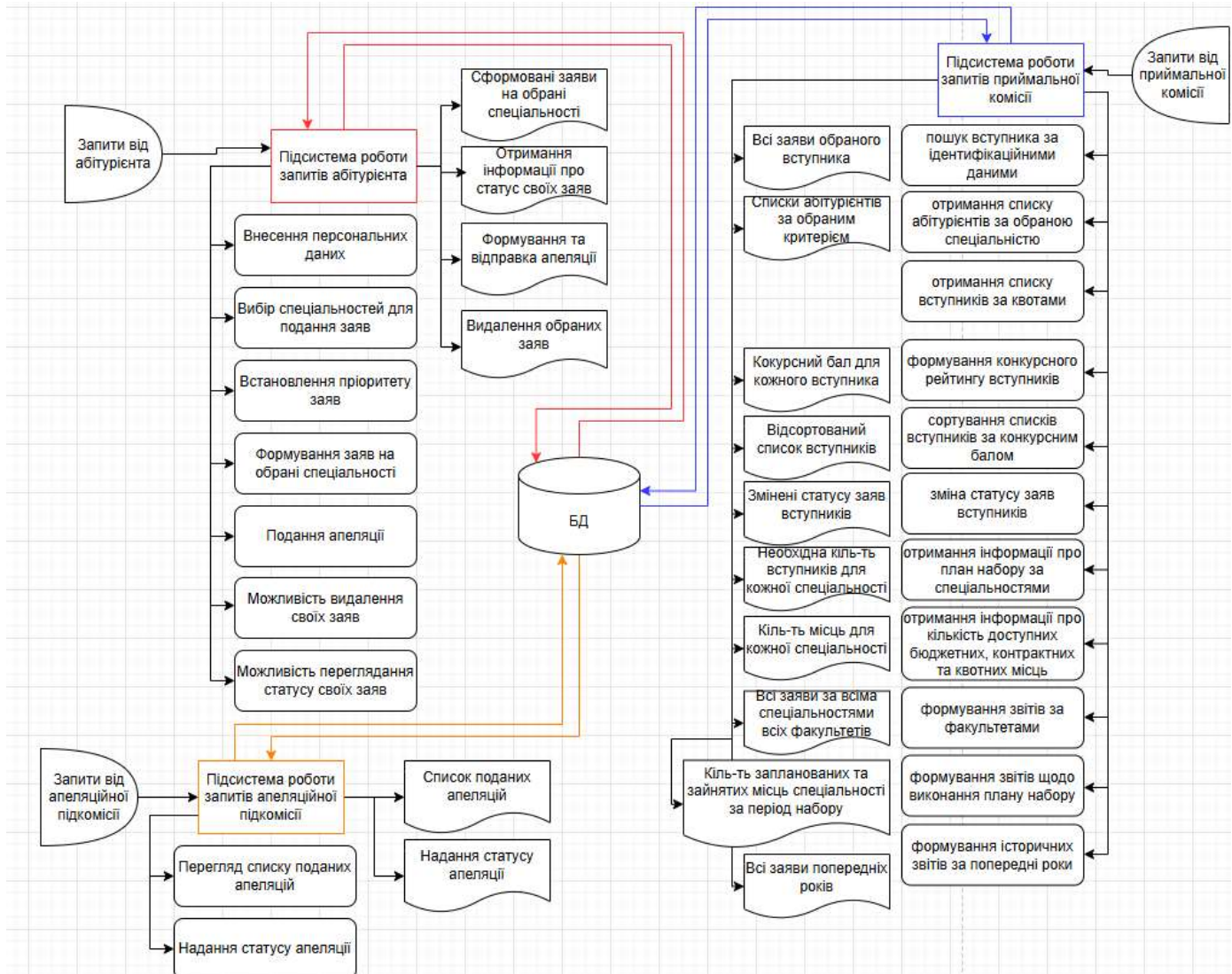


Рисунок 1.3 – Концептуальна модель інформаційної системи

Абітурієнт взаємодіє із системою з метою подання заяв та отримання інформації про хід вступної кампанії.

Запитом абітурієнта є внесення персональних даних, відповіддю системи є збережені персональні дані у базі даних.

Запитом абітурієнта є вибір спеціальностей для подання заяв, відповіддю системи є перелік обраних спеціальностей.

Запитом абітурієнта є встановлення пріоритету заяв, відповіддю системи є збережений порядок пріоритетності заяв.

Запитом абітурієнта є подання заяви на обрану спеціальність, відповіддю системи є сформована та зареєстрована заява.

Запитом абітурієнта є отримання інформації про статус своєї заяви, відповіддю системи є поточний статус заяви (допущено, у конкурсі, рекомендовано, відхилено тощо).

Запитом абітурієнта є подання апеляції, відповіддю системи є підтвердження прийняття апеляції та її збереження у системі.

Приймальна комісія виконує основні функції обробки даних вступної кампанії, формування конкурсних списків та підготовки управлінських рішень.

Запитом приймальної комісії є пошук вступника за ідентифікаційними даними, відповіддю системи є інформація про вступника та всі подані ним заяви.

Запитом є отримання списку абітурієнтів за обраною спеціальністю, відповіддю є список вступників за відповідною спеціальністю.

Запитом є отримання списку вступників за квотами, відповіддю є список вступників, що належать до квоти 1 або квоти 2.

Запитом є отримання списку вступників за пільговими категоріями, відповіддю є список вступників відповідної пільгової категорії.

Запитом є формування конкурсного рейтингу вступників, відповіддю є розрахований конкурсний бал для кожного вступника.

Запитом є сортування списків вступників за конкурсним балом, відповіддю є відсортований рейтинговий список вступників.

Запитом є зміна статусу заяв вступників, відповіддю є оновлений статус заяв у системі.

Запитом є отримання інформації про план набору за спеціальностями, відповіддю є дані про кількість місць для кожної спеціальності.

Запитом є отримання інформації про кількість доступних бюджетних, контрактних та квотних місць, відповіддю є актуальні дані про наявність місць.

Запитом є формування звітів за факультетами, відповіддю є звіт з узагальненими даними по факультетах.

Запитом є формування звітів щодо виконання плану набору, відповіддю є звіт про кількість заповнених та вільних місць.

Запитом є формування історичних звітів за попередні роки, відповіддю є архівні дані вступних кампаній попередніх років.

Апеляційна підкомісія займається спеціалізованими запитами, пов'язаними з розглядом апеляцій вступників, тому її доцільно виділити окремо від загальної приймальної комісії.

Запитом апеляційної підкомісії є перегляд списку поданих апеляцій, відповіддю системи є список усіх зареєстрованих апеляцій.

Запитом є перегляд апеляції конкретного вступника, відповіддю є дані про апеляцію та результати вступних випробувань вступника.

Запитом є надання статусу апеляції, відповіддю є зафіксований результат розгляду апеляції (задоволено або відхилено) із відповідним рішенням.

Висновок

Розробка запропонованого проекту інформаційної системи дозволить підвищити ефективність роботи приймальної комісії за рахунок автоматизації обробки даних, скорочення часу виконання основних операцій та підвищення точності формування конкурсних списків і звітної інформації. Це сприятиме покращенню якості організації вступної кампанії та прийняття управлінських рішень.

Розділ 2 – Розробка проекту інформаційної системи для покращення роботи приймальної комісії

2.1 Загальносистемні рішення

Для реалізації інформаційної системи приймальної комісії передбачається використання клієнт-серверної архітектури, що забезпечує розподіл функцій між користувацьким інтерфейсом, серверною частиною та системою управління базами даних [3].

Користувачі взаємодіють із системою через веб-інтерфейс, що реалізується у вигляді web-додатку. Обробка запитів користувачів здійснюється на серверній стороні, яка забезпечує реалізацію бізнес-логіки системи, формування відповідей та взаємодію з базою даних. Зберігання та обробка інформації здійснюється централізовано за допомогою системи управління базами даних.

Серверна частина системи реалізується із використанням мови програмування Java та фреймворку Spring, що забезпечує зручну реалізацію веб-додатків, обробку HTTP-запитів, організацію бізнес-логіки та інтеграцію з базою даних [4].

Для зберігання даних використовується система управління базами даних MySQL, яка підтримує реляційну модель даних, що дозволяє ефективно організувати зберігання структурованої інформації про вступників, заяви, результати вступних випробувань та інші дані. Крім того, MySQL характеризується високою продуктивністю та широкою підтримкою у сучасних веб-додатках.

Користувацький інтерфейс реалізується у вигляді веб-сторінок із використанням HTML та шаблонізатора Thymeleaf, що дозволяє динамічно відображати дані, отримані із серверної частини. Для оформлення інтерфейсу можуть використовуватися CSS-фреймворки, зокрема Bootstrap, що забезпечує швидке створення зручного та зрозумілого інтерфейсу користувача [4].

Для забезпечення взаємодії між клієнтською та серверною частинами може використовуватися технологія REST, що дозволяє організувати обмін даними у вигляді запитів та відповідей без необхідності повного перезавантаження сторінок.

Для забезпечення функціонування інформаційної системи необхідним є використання операційної системи, що підтримує роботу із сучасними засобами розробки веб-додатків, мовою програмування Java, а також системами управління базами даних.

Операційна система повинна забезпечувати:

- підтримку виконання Java-додатків;
- можливість роботи із серверними фреймворками;
- доступ до мережі Інтернет для взаємодії клієнтської та серверної частин;
- підтримку встановлення та роботи із системами управління базами даних;
- стабільність та надійність при обробці великого обсягу даних.

У якості операційної системи для розробки та функціонування інформаційної системи обрано Windows, оскільки вона повністю відповідає зазначеним вимогам, має широку підтримку програмних засобів розробки, а також є однією з найбільш розповсюджених операційних систем серед користувачів.

Використання Windows забезпечує сумісність із більшістю інструментів розробки, зокрема середовищем програмування IntelliJ IDEA, фреймворком Spring та системою управління базами даних MySQL, що спрощує процес розробки та тестування інформаційної системи.

Для забезпечення повноцінного функціонування інформаційної системи передбачається використання сервісних програм та зовнішніх інформаційних систем, що забезпечують виконання допоміжних функцій та розширюють можливості обробки даних.

До сервісних програм належить електронна пошта, яка використовується для інформування вступників про результати обробки їх

заяв, зміну статусу, а також результати розгляду апеляцій. Крім того, доступ до інформаційної системи здійснюється за допомогою веб-браузера, що забезпечує взаємодію користувачів із web-інтерфейсом системи.

Окрему роль відіграють зовнішні інформаційні системи, зокрема Єдина державна електронна база з питань освіти (ЄДЕБО), яка забезпечує централізоване зберігання та обробку даних про вступників. Використання таких систем дозволяє здійснювати обмін інформацією, зокрема передавати дані про подані заяви, результати вступних випробувань та статуси вступників.

Обрані загальносистемні рішення забезпечують ефективну реалізацію інформаційної системи, її масштабованість, зручність використання та відповідність сучасним вимогам до веб-додатків.

2.2 Рішення з інформаційного забезпечення

Для визначення основних подій інформаційної системи, користувачів, які їх ініціюють, та відповідних процесів обробки даних доцільно побудувати матрицю подій. Матриця подій представлена у таблиці 2.1.

Таблиця 2.1 – матриця подій

Подія	Ініціатор	Процес	Вхідна інформація	Вихідна інформація
Подання заяви	Абітурієнт	Формування заяви	Персональні дані, обрана спеціальність, пріоритет	Створена заява
Встановлення пріоритету заяви	Абітурієнт	Формування пріоритетів заяв	Дані про заяву, значення пріоритету	Збережений пріоритет заяви
Розгляд заяви	Приймальна комісія	Надання статусу заяви	Дані заяви	Оновлений статус заяви
Подання апеляції	Абітурієнт	Формування апеляції	Дані заяви, текст апеляції	Створена апеляція
Розгляд апеляції	Апеляційна комісія	Надання статусу апеляції	Дані апеляції	Оновлений статус апеляції
Зміна плану набору	Приймальна комісія	Заміна кількості	Дані спеціальності,	Оновлені дані спеціальності

		бюджетних та контрактних місць	нові значення місць	
Формування звіту за спеціальностями	Приймальна комісія	Формування звіту	Дані спеціальностей та факультетів	Звіт по спеціальностях
Формування звіту за заявами	Приймальна комісія	Формування звіту	Дані заяв вступників	Звіт по заявах

З метою формалізованого опису потоків даних, відображення процесів обробки інформації, джерел та приймачів даних доцільно побудувати DFD-діаграми 0–2 рівнів, щоб відобразити загальну структуру функціонування інформаційної системи та деталізувати кожен окремий процес обробки даних.

Щоб визначити загальну структуру взаємодії користувачів із інформаційною системою, побудуємо DFD-діаграму нульового рівня.

Основними користувачами системи є абітурієнт, приймальна комісія та апеляційна комісія, які взаємодіють із системою шляхом формування запитів та отримання результатів обробки даних.

DFD-діаграма нульового рівня представлена на рисунку 2.2.

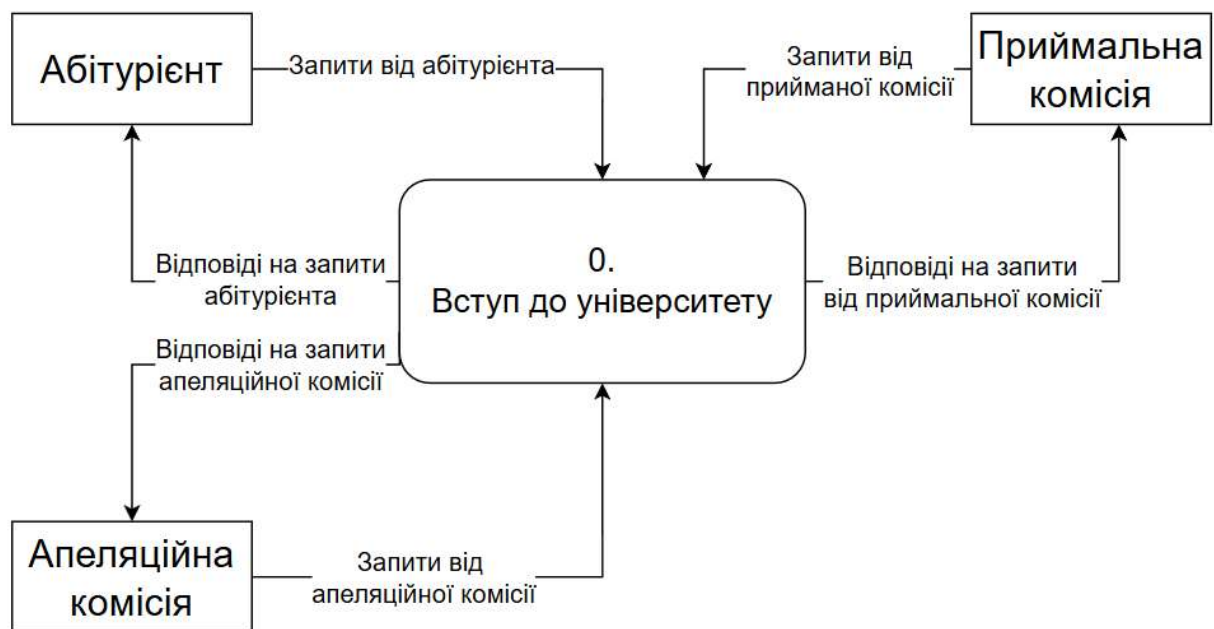


Рисунок 2.2 – DFD-діаграма нульового рівня

Наступним етапом здійснюється декомпозиція загального процесу функціонування інформаційної системи на основні функціональні процеси.

Основними інформаційними процесами є обробка заяв, обробка апеляцій, обробка спеціальностей та формування звітів.

Результати декомпозиції відображено на DFD-діаграмі першого рівня, представлений на рисунку 2.3.

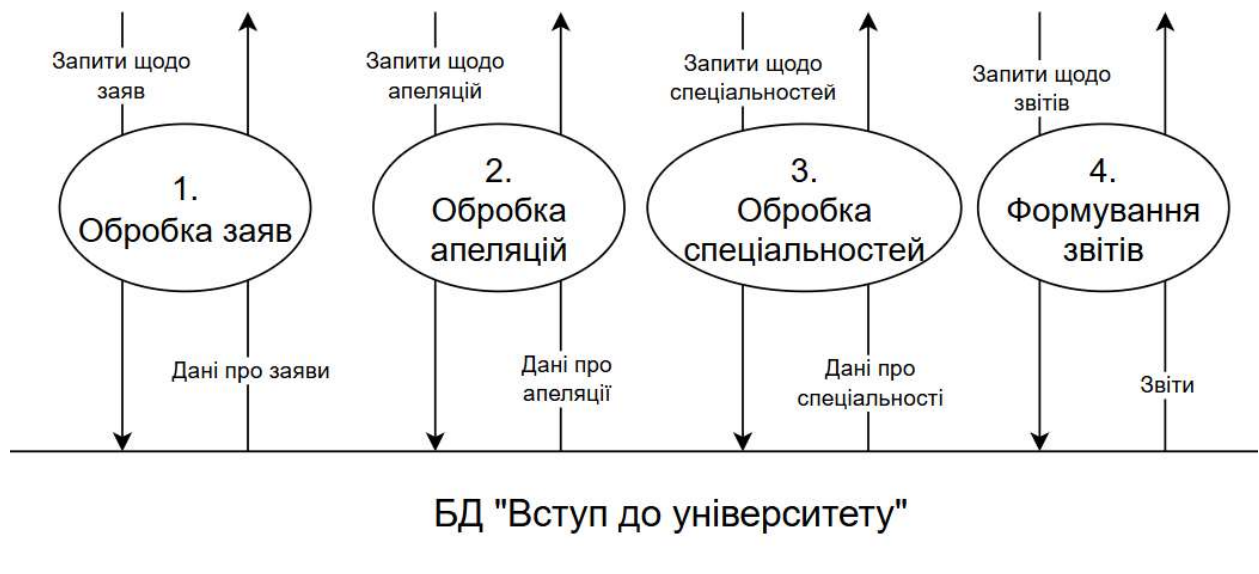


Рисунок 2.3 – DFD-діаграма першого рівня

Для більш детального відображення основних інформаційних процесів виконується виділення основних підпроцесів, що забезпечують їх функціонування.

Підпроцесами обробки заяв є формування заяви, формування пріоритетів заяв та надання статусу заяви. Підпроцес формування заяви забезпечує створення нової заяви вступником на обрану спеціальність. Підпроцес формування пріоритетів заяв дозволяє встановлювати пріоритетність поданих заяв відповідно до побажань вступника. Підпроцес надання статусу заяви забезпечує зміну та відображення поточного стану заяви працівниками приймальної комісії.

Підпроцесами обробки апеляцій є формування апеляції та надання статусу апеляції. Підпроцес формування апеляції забезпечує подання

вступником апеляційного звернення щодо результатів вступної кампанії. Підпроцес надання статусу апеляції дозволяє апеляційній комісії фіксувати результати розгляду апеляції та відображати її поточний стан.

Підпроцесом обробки спеціальностей є зміна кількості бюджетних та контрактних місць. Даний підпроцес забезпечує коригування плану набору за спеціальностями відповідно до потреб закладу вищої освіти та вимог вступної кампанії.

Підпроцесами обробки звітів є формування звіту про всі спеціальності за факультетами та формування звіту про всі заявки за спеціальностями обраного року. Перший підпроцес забезпечує отримання узагальненої інформації про спеціальності та їх належність до факультетів. Другий підпроцес призначений для отримання інформації про подані заяви вступників у розрізі спеціальностей за визначений рік вступної кампанії.

Результат наведено на DFD-діаграмі другого рівня на рисунку 2.4.

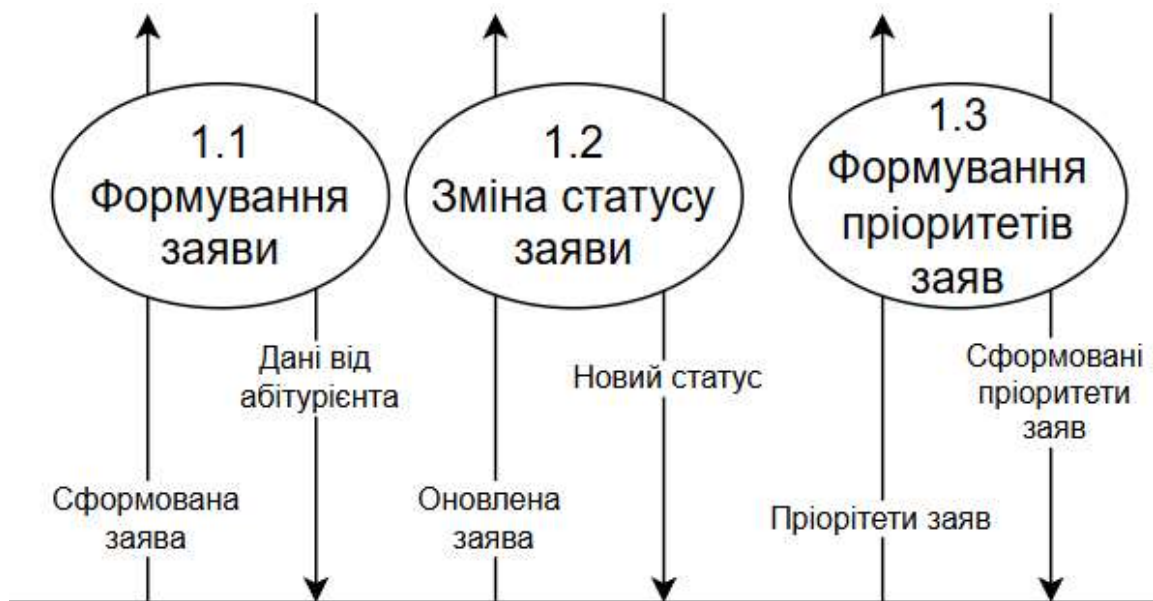


Рисунок 2.4 - DFD-діаграма другого рівня для першого та другого процесів

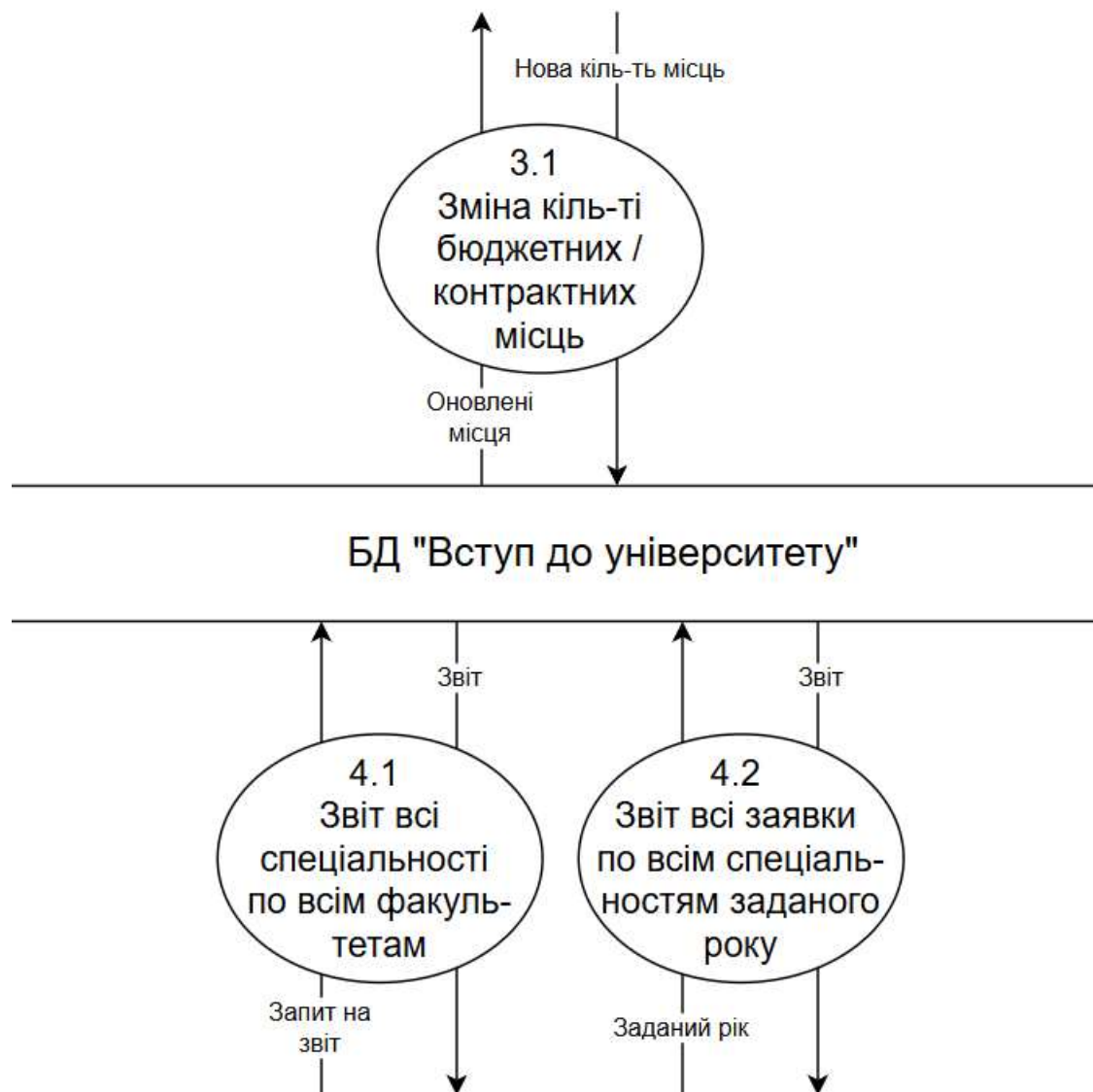


Рисунок 2.5 – DFD-діаграма другого рівня для третього та четвертого процесів

Для узагальнення результатів DFD-моделювання та відображення взаємозв'язку між процесами різних рівнів деталізації доцільно побудувати таблицю відповідності процесів. У таблиці 2.6 наведено відповідність між інформаційними потоками нульового рівня, основними процесами першого рівня та підпроцесами другого рівня.

Таблиця 2.6 – відповідності процесів

Користувач	DFD 0 рівня	DFD 1 рівня	DFD 2 рівня
Абітурієнт	Запити від абітурієнта	Обробка заяв	Формування заяви

Абітурієнт	Запити від абітурієнта	Обробка заяв	Формування пріоритетів заяв
Приймальна комісія	Запити від приймальної комісії	Обробка заяв	Надання статусу заяви
Абітурієнт	Запити від абітурієнта	Обробка апеляцій	Формування апеляції
Апеляційна комісія	Запити від апеляційної комісії	Обробка апеляцій	Надання статусу апеляції
Приймальна комісія	Запити від приймальної комісії	Обробка спеціальностей	Заміна кіль-ті бюджетних / контрактних місць
Приймальна комісія	Запити від приймальної комісії	Обробка звітів	Запит на звіт всі спеціальності по всім факультетам
Приймальна комісія	Запити від приймальної комісії	Обробка звітів	Запит на звіт всі заявки по всім спеціальностям заданого року

Для подальшого проектування структури бази даних інформаційної системи необхідно спочатку визначити основні сутності предметної області та взаємозв'язки між ними. Результат представлено у вигляді концептуальної моделі даних, наведеної на рисунку 2.7

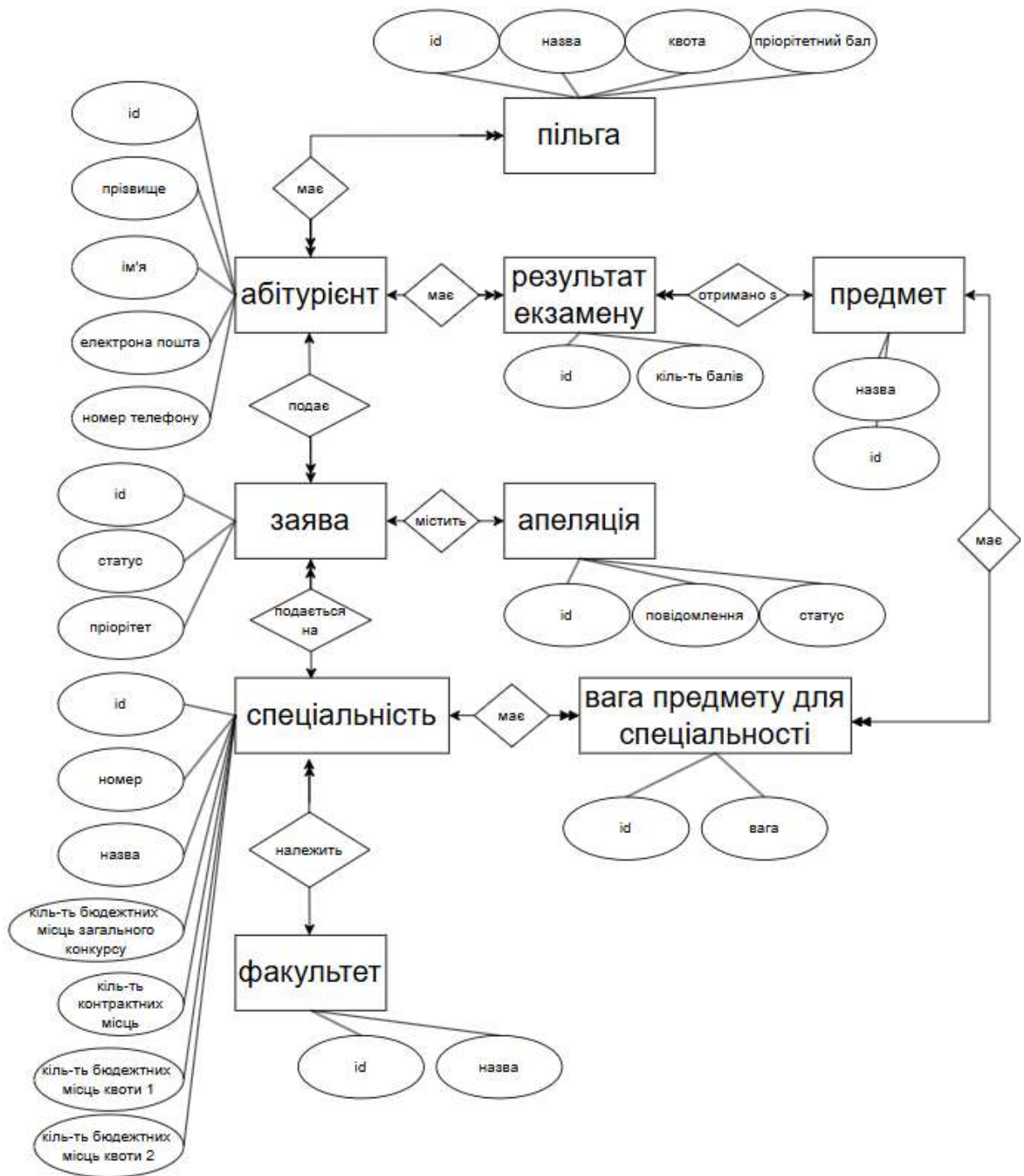


Рисунок 2.7 – Концептуальна модель бази даних

Після визначення основних сутностей предметної області та зв'язків між ними з метою подальшого переходу на фізичну модель, за якою вже буде створена бд, доцільно побудувати логічну модель, що первинно описує типи даних, відображає зв'язуючі таблиці, первинні та зовнішні ключі.

Логічна модель бази даних представлена на рисунку 1.4.

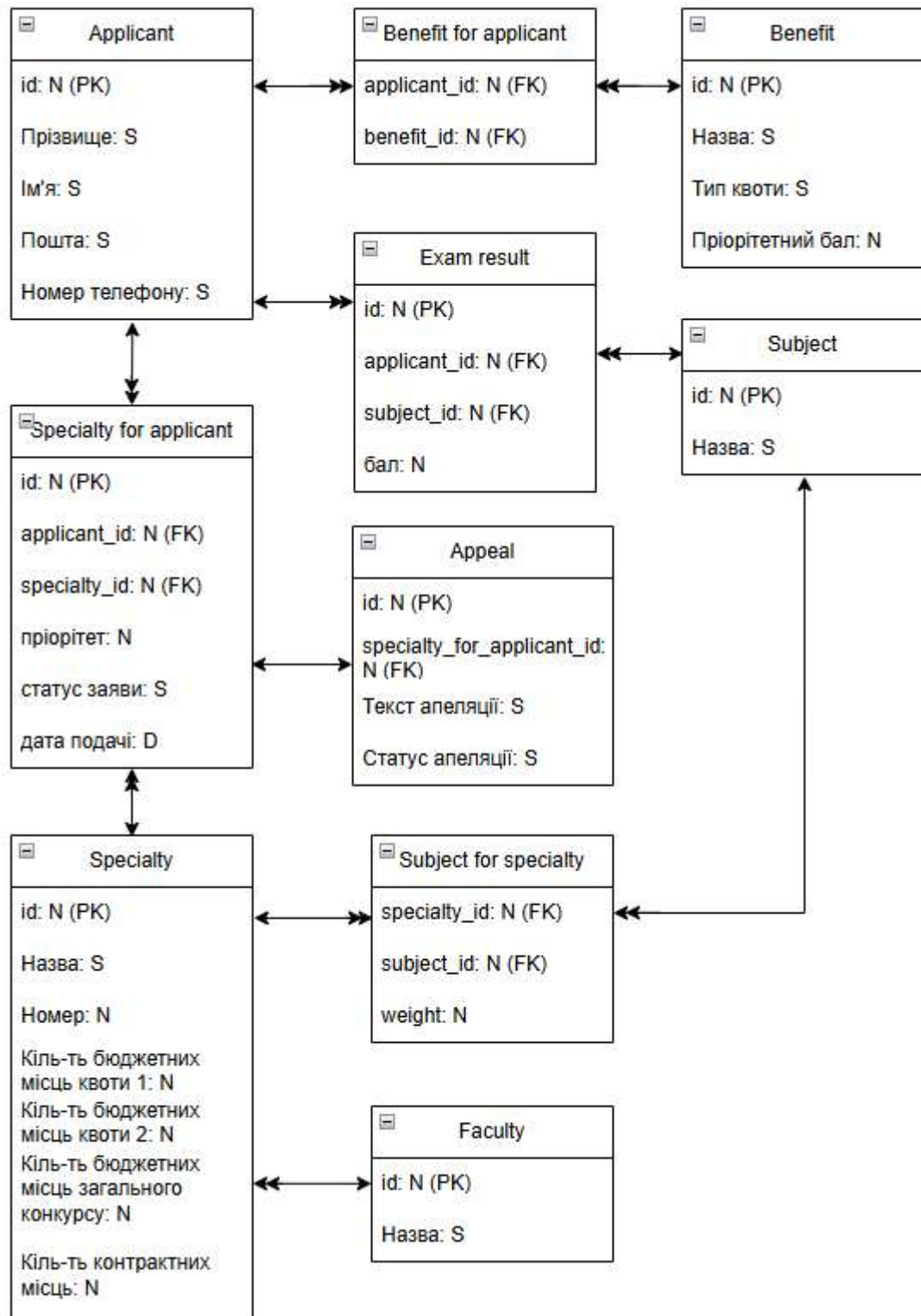


Рисунок 2.8 – Логічна модель бази даних

Для практичної реалізації інформаційної системи необхідно побудувати фізичну модель бази даних, яка визначає структуру таблиць, типи полів, обмеження цілісності та механізми реалізації зв'язків у середовищі MySQL.

Фізична модель бази даних представлена на рисунку 2.9.

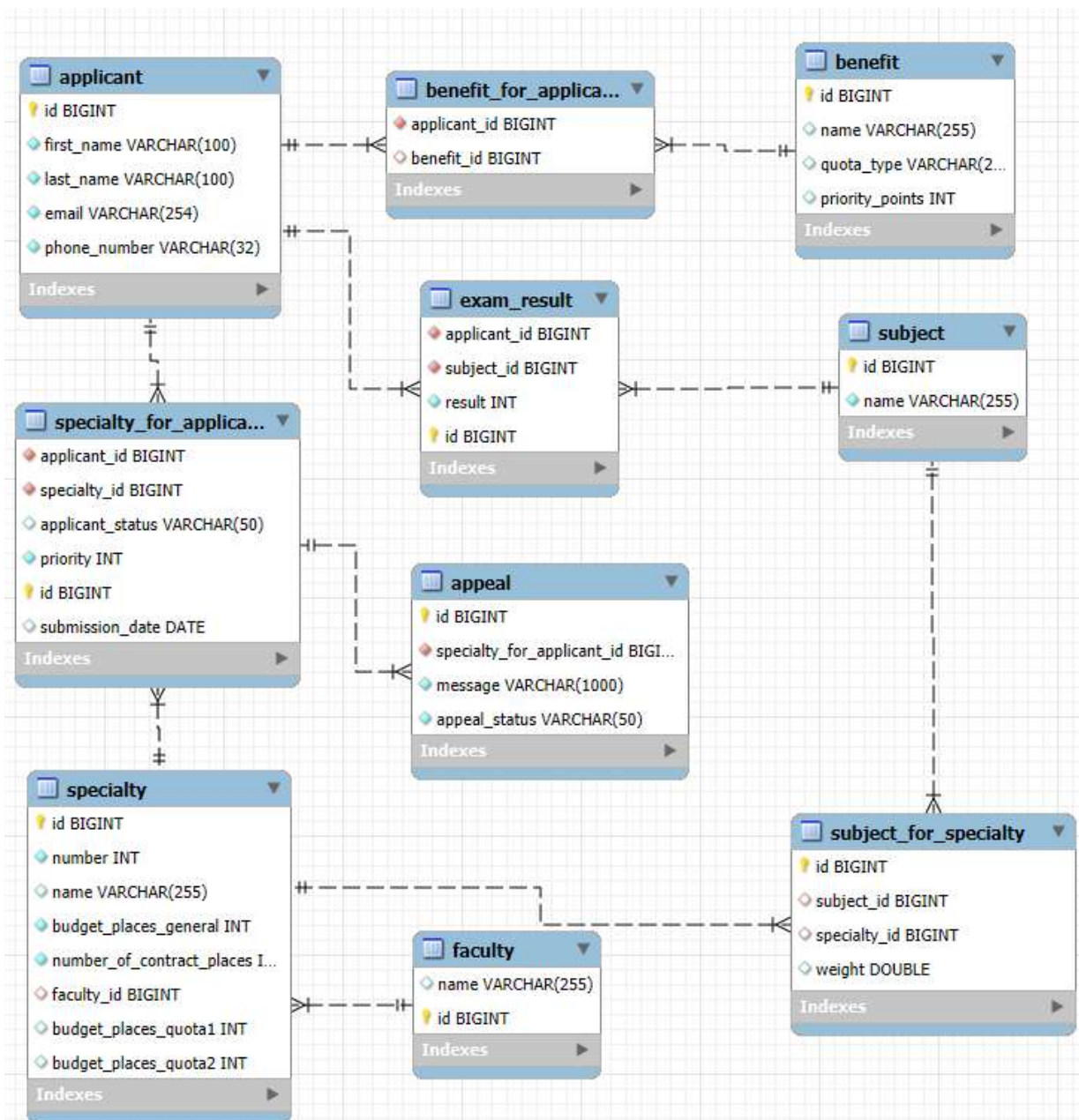


Рисунок 2.9 – Фізична модель бази даних

2.3 Рішення з математичного забезпечення

Математичне забезпечення інформаційної системи призначене для автоматизації процесів обробки даних вступників, формування конкурсних балів, побудови рейтингових списків та підтримки прийняття рішень щодо зарахування абітурієнтів.

Основою математичного забезпечення системи є алгоритм розрахунку конкурсного балу вступника. Конкурсний бал формується на основі результатів вступних випробувань із урахуванням вагових коефіцієнтів

предметів для конкретної спеціальності. Розрахунок конкурсного балу здійснюється за формулою:

$$KB = \sum(B_i \times W_i)$$

де:

KB — конкурсний бал вступника;

B_i — результат i -го предмета;

W_i — ваговий коефіцієнт i -го предмета для обраної спеціальності.

Використання вагових коефіцієнтів дозволяє враховувати значущість окремих предметів для різних спеціальностей.

У випадку однакового конкурсного балу система виконує додаткове порівняння вступників за пріоритетами пільг. Для кожного вступника формується впорядкований за спаданням список значень `'priority_points'`, що відповідають усім наявним пільгам вступника.

Порівняння здійснюється послідовно, починаючи з найбільшого значення `'priority_points'`. У випадку, якщо найбільші значення збігаються, система переходить до порівняння наступних значень списку. Процес продовжується до моменту виявлення відмінності між значеннями.

У випадку, якщо всі порівнювані значення є однаковими, перевага надається вступнику, список пільг якого містить більшу кількість елементів. Якщо після виконання порівняння відмінності між вступниками не виявлено, система переходить до наступного критерію сортування — пріоритету заяви.

Якщо після порівняння конкурсного балу та пріоритетів пільг вступники залишаються рівними, система виконує додаткове сортування за пріоритетом заяви. Кожна заява вступника містить значення пріоритету, яке визначає важливість обраної спеціальності для вступника.

Під час порівняння перевага надається заяві з вищим пріоритетом. У системі менше числове значення відповідає більшому пріоритету заяви. Наприклад, заява з пріоритетом 1 має перевагу над заявою з пріоритетом 3.

2.4 Рішення з програмного забезпечення

Програмне забезпечення інформаційної системи призначене для забезпечення взаємодії користувачів із системою, обробки запитів, автоматизації бізнес-процесів вступної кампанії та організації роботи з базою даних.

Для реалізації серверної частини інформаційної системи використовується мова програмування Java, яка є об'єктно-орієнтованою мовою загального призначення, що забезпечує платформонезалежність, надійність та високу продуктивність, що дозволяє реалізовувати складну бізнес-логіку та інтеграцію з базою даних.

Для організації обробки запитів та взаємодії між компонентами системи використовується фреймворк Spring, який забезпечує зручну розробку web-додатків та інтеграцію з системами управління базами даних, дозволяючи реалізувати багаторівневу архітектуру та обробку HTTP-запитів.

Взаємодія між клієнтською та серверною частинами здійснюється за допомогою REST-підходу, який є архітектурним стилем розподілених систем, що дозволяє обмінюватися даними через HTTP-запити та відповіді без повного перезавантаження сторінок, забезпечуючи динамічне оновлення інформації.

Для зберігання даних використовується MySQL, яка є реляційною системою управління базами даних, що забезпечує підтримку реляційної моделі, цілісність інформації, ефективне виконання запитів та легку інтеграцію з фреймворком Spring.

Для оформлення користувацького інтерфейсу застосовується CSS-фреймворк Bootstrap, який надає готові стилі, компоненти та іконки, що дозволяє швидко створювати адаптивний та зрозумілий інтерфейс користувача.

Для розробки програмного забезпечення використовується інтегроване середовище IntelliJ IDEA, яке забезпечує підтримку Java, Spring, систем управління базами даних та надає інструменти генерації коду, налагодження та управління проектом.

Для контролю версій та спільної роботи над кодом використовується Git, що дозволяє відслідковувати зміни, вести історію розробки, повертатися до попередніх станів проекту та координувати роботу декількох розробників.

2.5 Рішення з технічного забезпечення

Технічне забезпечення інформаційної системи призначене для забезпечення стабільної роботи серверної частини, функціонування бази даних, обробки запитів користувачів та підтримки роботи web-додатку приймальної комісії.

Інформаційна система реалізується за клієнт-серверною архітектурою, у якій користувачі взаємодіють із системою через веб-браузер, а основна обробка даних виконується на серверній частині системи.

Для функціонування інформаційної системи необхідне використання серверного комп'ютера, на якому розміщуються:

- * серверна частина web-додатку;
- * система управління базами даних MySQL;
- * програмні компоненти обробки запитів та формування рейтингових списків.

Серверний комп'ютер може бути розміщений у локальній мережі закладу вищої освіти або на окремому хостингу із постійним доступом до мережі Інтернет.

Мінімальні технічні характеристики серверного комп'ютера:

- * процесор Intel Core i3 або аналогічний;
- * оперативна пам'ять не менше 8 ГБ;
- * накопичувач SSD обсягом не менше 256 ГБ;
- * мережеве підключення до Інтернету;
- * операційна система Windows.

Користувачі взаємодіють із системою через web-інтерфейс без необхідності встановлення додаткового програмного забезпечення.

Мінімальні технічні характеристики клієнтських пристроїв:

- * процесор Intel Pentium або аналогічний;
- * оперативна пам'ять не менше 4 ГБ;

- * наявність веб-браузера;
- * доступ до мережі Інтернет;
- * монітор із роздільною здатністю не менше 1366×768.

Для забезпечення роботи приймальної комісії передбачається використання декількох робочих місць працівників, що забезпечують:

- * обробку заяв вступників;
- * формування рейтингових списків;
- * роботу з апеляціями;
- * формування звітності;
- * контроль плану набору.

Кількість робочих місць визначається масштабом вступної кампанії та кількістю працівників приймальної комісії.

Для забезпечення надійності функціонування інформаційної системи передбачається використання засобів резервного копіювання бази даних та стабільного підключення до мережі Інтернет.

Обране технічне забезпечення забезпечує стабільне функціонування інформаційної системи, підтримку роботи web-додатку та ефективну обробку даних вступної кампанії.

Висновок

У другому розділі було обґрунтовано основні проектні рішення щодо створення інформаційної системи для підтримки роботи приймальної комісії. Розглянуто архітектуру системи, інформаційне, математичне, програмне та технічне забезпечення, необхідні для її стабільного функціонування. Запропоновані рішення забезпечують централізовану обробку даних, автоматизацію формування конкурсних списків, роботу із заявами, апеляціями та звітністю. Отже, розроблений проект створює основу для подальшої реалізації інформаційної системи та її практичного використання під час вступної кампанії.

Розділ 3 – реалізація проекту ІС

3.1 Керівництво користувача системи

Після виконання етапів аналізу, проектування та розробки інформаційної системи доцільно розглянути результати її реалізації. У даному розділі наведено опис основних елементів користувацького інтерфейсу, порядок роботи із системою та результати функціонування реалізованого програмного забезпечення.

Після запуску інформаційної системи користувачеві відображається сторінка авторизації, представлена на рисунку __. Дана сторінка призначена для визначення ролі користувача та надання доступу до відповідного функціоналу системи.

Для входу до системи користувачеві необхідно ввести значення у поле авторизації. Для працівників приймальної комісії, апеляційної комісії та системного адміністратора у якості значення використовується відповідний пароль доступу. Для абітурієнтів у даному полі вводиться адреса електронної пошти, яка використовується системою для ідентифікації конкретного вступника та надання доступу до пов'язаних із ним даних.

У випадку введення некоректного значення система відображає повідомлення про помилку авторизації та пропонує повторити спробу входу.

Нижче поля введення розташовано кнопки керування. Кнопка «Login» використовується для входу до системи із зазначеними даними. Кнопка «Logout» призначена для завершення поточного сеансу роботи та виходу з обраної ролі. Кнопка «Registration» використовується для переходу до форми реєстрації нових абітурієнтів, які ще не внесені до системи.

У верхній частині сторінок системи розташовано навігаційну панель (header), яка використовується для переходу між основними розділами інформаційної системи. У лівій частині навігаційної панелі відображається поточна роль користувача, що дозволяє швидко визначити набір доступних функцій та прав доступу. Склад навігаційної панелі залежить від поточної ролі користувача та визначається його правами доступу.

Для користувача, який ще не виконав авторизацію та не має призначеної ролі, у навігаційній панелі відображається лише посилання на сторінку входу до системи. Після успішної авторизації склад навігаційної панелі автоматично змінюється відповідно до ролі користувача, надаючи доступ лише до тих сторінок та функцій, які передбачені для даної категорії користувачів.

Після натискання кнопки «Registration» користувач переходить на сторінку реєстрації абітурієнта.

Основним елементом сторінки реєстрації є форма додавання абітурієнта, призначена для внесення персональних даних вступника, результатів вступних випробувань та формування переліку обраних спеціальностей. Дана форма використовується для створення нового абітурієнта в системі та надалі може застосовуватися в інших функціональних модулях інформаційної системи.

Форма містить текстові поля для введення імені, прізвища, адреси електронної пошти та номера телефону. Усі поля є обов'язковими для заповнення, а спроба збереження форми з порожніми значеннями призводить до відображення повідомлення про помилку.

Для вибору пільг використовується набір прапорців вибору (checkbox), де для кожної пільги відображається її назва та відповідна квота вступу. Абітурієнт може обрати одну або декілька пільг залежно від власного статусу.

Також форма містить поля для введення результатів вступних випробувань. Перелік предметів формується автоматично на підставі даних, що зберігаються в базі даних системи. Після введення результатів система виконує перевірку відповідності предметів вимогам спеціальностей та автоматично формує перелік доступних для вступу спеціальностей.

Доступні спеціальності відображаються у вигляді набору прапорців вибору. Під час вибору спеціальностей система автоматично визначає та відображає їх пріоритетність. Для кожної обраної спеціальності встановлюється відповідний пріоритет заяви, який використовується під час конкурсного відбору.

Для успішного завершення роботи з формою необхідно обрати щонайменше одну спеціальність. У випадку відсутності вибору система відображає повідомлення про помилку та не дозволяє зберегти введені дані.

Після успішного заповнення форми дані абітурієнта зберігаються в базі даних, користувач автоматично авторизується в системі з роллю «Абітурієнт» та перенаправляється на головну сторінку для подальшої роботи.

Після успішної авторизації з роллю «Абітурієнт» у навігаційній панелі додатково відображається посилання на сторінку «Абітурієнти», яка призначена для роботи із власними заявами та апеляціями.

У верхній частині сторінки розташовано форму перегляду заяв абітурієнта, що містить кнопку «Переглянути всі заявки». Після натискання кнопки система виконує пошук усіх заяв, що належать поточному абітурієнту, та відображає їх у таблиці результатів.

Нижче розташовано форму видалення заяви, яка містить поле для введення ідентифікатора заяви та кнопку підтвердження операції. Абітурієнт має право видаляти лише власні заяви. У випадку введення ідентифікатора чужої заяви або неіснуючого запису система відображає повідомлення про помилку та не виконує видалення.

Також на сторінці розміщено форму подання апеляції, що складається з поля введення ідентифікатора заяви, текстового поля для введення змісту апеляції та кнопки надсилення. Подання апеляції можливе лише для заяв поточного абітурієнта. Для кожної заяви може бути створена лише одна апеляція. Порожній текст апеляції не допускається, а при порушенні правил введення система відображає повідомлення про помилку.

У нижній частині сторінки розташована таблиця заяв поточного року вступної кампанії. Для спрощення пошуку та аналізу інформації передбачено засоби фільтрації за спеціальністю та квотою вступу. Фільтри реалізовано у вигляді випадаючих списків, а вибір значення автоматично оновлює вміст таблиці відповідно до обраного критерію. Для одночасного застосування обох критеріїв передбачено кнопку «Застосувати фільтри».

Таблиця містить основну інформацію про заяви абітурієнта, зокрема ідентифікатор заяви, ідентифікатор абітурієнта, ім'я, прізвище, номер телефону, адресу електронної пошти, середній конкурсний бал, дату подання та поточний статус заяви. Крім того, для кожної заяви відображається перелік предметів, необхідних для вступу на відповідну спеціальність, а також результати вступних випробувань з цих предметів.

Для користувача з роллю «Приймальна комісія» у навігаційній панелі, окрім сторінки «Абітурієнти», додатково відображаються сторінки «Спеціальності» та «Звіти». Доступ до даних сторінок обумовлений необхідністю виконання основних функцій приймальної комісії під час проведення вступної кампанії.

На сторінці «Абітурієнти» розташовано форму додавання абітурієнта, призначену для реєстрації нового вступника в систему. Дана форма має аналогічну структуру до форми реєстрації абітурієнта та дозволяє працівнику приймальної комісії самостійно внести необхідні дані до системи.

Для видалення абітурієнта передбачено окрему форму, що містить поле введення ідентифікатора абітурієнта та кнопку підтвердження операції. Після виконання операції із системи видаляється інформація про абітурієнта та всі пов'язані з ним заяви.

Також на сторінці розташовано форму пошуку заяв абітурієнта за його ідентифікатором. На відміну від аналогічної функції для ролі «Абітурієнт», де пошук виконується автоматично для поточного користувача, працівник приймальної комісії може переглядати інформацію щодо будь-якого вступника, вказавши його ідентифікатор.

Для прийняття рішень за результатами конкурсного відбору передбачено форму зміни статусу заяви. Форма містить поля для введення ідентифікатора абітурієнта та ідентифікатора спеціальності, після чого працівник приймальної комісії може встановити один із доступних статусів заяви. Для новостворених заяв за замовчуванням використовується статус «Невизначено».

У нижній частині сторінки розташована таблиця заяв поточного року вступної кампанії. Для спрощення роботи з великим обсягом даних передбачено механізми фільтрації за спеціальністю та квотою вступу. Фільтрація може виконуватися як за окремими критеріями, так і за їх комбінацією за допомогою кнопки «Застосувати фільтри». Структура таблиці відповідає таблиці, доступній для ролі «Абітурієнт».

Сторінка «Спеціальності» призначена для перегляду та редагування інформації про спеціальності, що беруть участь у вступній кампанії.

У верхній частині сторінки розташовано форму пошуку спеціальностей за ключовими атрибутами. Пошук може виконуватися за ідентифікатором спеціальності, її назвою або унікальним кодом. Для виконання пошуку необхідно заповнити щонайменше одне поле форми. У випадку, якщо спеціальність за вказаними критеріями не знайдена, система відображає повідомлення про помилку.

Нижче розташовано форму редагування кількості місць для спеціальності. Дана форма дозволяє змінювати кількість бюджетних місць за квотою 1, бюджетних місць за квотою 2, бюджетних місць загального конкурсу та кількість контрактних місць. Користувач може змінити як один із зазначених показників, так і декілька одночасно. Для запобігання некоректним операціям система контролює наявність хоча б одного заповненого поля, а спроба залишити всі поля порожніми супроводжується повідомленням про помилку.

У нижній частині сторінки розташована таблиця спеціальностей, яка містить інформацію про ідентифікатор спеціальності, її назву, унікальний код, кількість бюджетних місць за квотою 1, кількість бюджетних місць за квотою 2, кількість бюджетних місць загального конкурсу, кількість контрактних місць та загальну кількість місць.

Для спрощення роботи з переліком спеціальностей передбачено механізм фільтрації за факультетами. Фільтр реалізовано у вигляді випадаючого списку, який автоматично формується на підставі даних, що зберігаються в системі.

Після вибору факультету таблиця автоматично оновлюється та відображає лише спеціальності, що належать до обраного факультету.

Сторінка «Звіти» призначена для формування узагальненої інформації про результати вступної кампанії та перегляду даних, що зберігаються в системі.

У верхній частині сторінки розташовано форму формування звіту за заявами абітурієнтів. Форма містить випадаючий список років та кнопку «Абітурієнти». Перелік років формується автоматично на підставі даних, наявних у системі, починаючи з року подання першої заяви та закінчуючи поточним роком. Після вибору року та натискання кнопки «Абітурієнти» відображається звіт, що містить усі заявки вступників за обраний рік незалежно від спеціальності. Повторне натискання кнопки приховує сформований звіт.

Структура таблиці звіту відповідає таблиці, що використовується на сторінці «Абітурієнти», та містить інформацію про заяви, дані вступників, результати вступних випробувань, конкурсний бал, дату подання та поточний статус заяв. Нижче розташовано форму формування звіту за спеціальностями, яка містить кнопку для відображення переліку всіх спеціальностей, наявних у системі. Після натискання кнопки відображається таблиця спеціальностей за всіма факультетами. Повторне натискання кнопки приховує сформований звіт.

Структура таблиці відповідає таблиці сторінки «Спеціальності» та містить інформацію про ідентифікатор спеціальності, її назву, унікальний код, кількість бюджетних місць за квотою 1, кількість бюджетних місць за квотою 2, кількість бюджетних місць загального конкурсу, кількість контрактних місць та загальну кількість місць.

Для користувача з роллю «Апеляційна комісія» у навігаційній панелі відображається сторінка «Апеляції», призначена для розгляду та обробки апеляційних звернень вступників.

У верхній частині сторінки розташовано форму зміни статусу апеляції. Форма містить поле для введення ідентифікатора апеляції та випадаючий список

доступних статусів. Для апеляції можуть бути встановлені статуси «Схвалено» або «Відхилено». Для новостворених апеляцій за замовчуванням використовується статус «Невизначено». Після підтвердження операції система оновлює статус відповідної апеляції.

Нижче розташовано форму пошуку апеляції за її ідентифікатором. Після виконання пошуку система відображає інформацію про знайдену апеляцію або повідомлення про помилку у випадку відсутності відповідного запису.

У нижній частині сторінки розміщено таблицю апеляцій, що містить інформацію про ідентифікатор апеляції, ідентифікатор абітурієнта, прізвище, ім'я та по батькові вступника, ідентифікатор заяви, ідентифікатор спеціальності, назву спеціальності, кнопку перегляду детальної інформації, статус заяви та статус апеляції.

Для кожного запису таблиці передбачено кнопку «Переглянути», натискання якої відкриває додаткове інформаційне вікно. У вікні відображаються ідентифікатор апеляції, ідентифікатор абітурієнта, прізвище, ім'я та по батькові вступника, ідентифікатор заяви, ідентифікатор спеціальності, назва спеціальності та текст поданої апеляції.

Для спрощення пошуку та аналізу інформації передбачено механізми фільтрації апеляцій за спеціальністю та категорією конкурсу. Фільтри реалізовано у вигляді випадаючих списків із можливістю відбору апеляцій за спеціальністю, квотою 1, квотою 2 або загальним конкурсом. Для одночасного застосування декількох критеріїв використовується кнопка «Застосувати фільтри», після натискання якої таблиця оновлюється відповідно до обраних параметрів.

Для користувача з роллю «Системний адміністратор» доступний повний набір функцій керування даними інформаційної системи. У навігаційній панелі відображаються сторінки «Абітурієнти», «Спеціальності», «Апеляції» та «Звіти».

На сторінці «Абітурієнти» адміністратору доступна форма додавання нового абітурієнта, форма пошуку заяв абітурієнта за ключовими атрибутами, форма

видалення абітурієнта разом із усіма пов'язаними заявами, форма видалення окремої заяви за її ідентифікатором та форма зміни статусу заяви. Крім того, на сторінці розташована таблиця заяв із можливістю фільтрації за спеціальністю та категорією конкурсу, аналогічна сторінці ролі «Приймальна комісія».

На сторінці «Спеціальності» адміністратору доступні всі функції, передбачені для працівника приймальної комісії, а також додаткові засоби керування переліком спеціальностей. Для створення нової спеціальності використовується окрема форма, яка містить поля для введення назви спеціальності, унікального коду, кількості бюджетних місць за квотою 1, кількості бюджетних місць за квотою 2, кількості місць загального конкурсу та кількості контрактних місць.

Також форма містить випадуючий список факультетів, що дозволяє визначити факультет, до якого належатиме нова спеціальність. Для налаштування вимог до вступу відображається перелік предметів, доступних у системі, для кожного з яких може бути вказаний коефіцієнт значущості. Значення коефіцієнтів вводяться лише для тих предметів, які використовуються під час розрахунку конкурсного бала для відповідної спеціальності.

Під час створення спеціальності система виконує перевірку коректності введених даних. Не допускається створення спеціальностей із уже існуючими назвами або кодами, залишення обов'язкових полів порожніми, а також створення спеціальності без визначення жодного предмета вступного випробування. У разі порушення зазначених правил система відображає повідомлення про помилку та не виконує збереження даних.

Для видалення спеціальності передбачено окрему форму, що містить поле введення ідентифікатора спеціальності та кнопку підтвердження операції. Видалення спеціальності можливе лише за відсутності заяв, пов'язаних із нею. Якщо в системі існують заявки на відповідну спеціальність, система блокує операцію видалення та відображає повідомлення про необхідність попереднього видалення або перенесення пов'язаних записів.

Сторінка «Звіти» для ролі «Системний адміністратор» має аналогічну структуру та функціональні можливості до сторінки звітів ролі «Приймальна комісія». Користувач може формувати звіти за заявами вступників для обраного року вступної кампанії, а також переглядати звіт про всі спеціальності, наявні в системі. Звіти відображаються у вигляді таблиць та можуть бути приховані шляхом повторного натискання відповідної кнопки.

На сторінці «Апеляції» адміністратору доступні всі функції, передбачені для ролі «Апеляційна комісія», а також додаткові засоби керування апеляціями. Для створення нової апеляції використовується форма додавання апеляції, що дозволяє вказати необхідні дані та зберегти новий запис у системі.

Для видалення апеляції передбачено окрему форму, яка містить поле для введення ідентифікатора апеляції та кнопку підтвердження операції. Після успішного виконання операції відповідна апеляція видаляється із системи.

Також адміністратору доступні форма пошуку апеляції за її ідентифікатором та форма зміни статусу апеляції. Оновлення статусу виконується шляхом вибору одного з доступних значень статусу для вказаної апеляції.

У нижній частині сторінки розташована таблиця апеляцій, структура якої відповідає таблиці, доступній для ролі «Апеляційна комісія». Для спрощення роботи з даними передбачено механізми фільтрації за спеціальністю та категорією конкурсу. Фільтрація може виконуватися як за окремими критеріями, так і за їх комбінацією за допомогою кнопки «Застосувати фільтри».

3.2 Результати експлуатації системи

Для перевірки працездатності розробленої інформаційної системи було проведено тестування її основних функціональних можливостей. У процесі тестування перевірялася коректність роботи механізмів авторизації та реєстрації користувачів, обробки заяв вступників, роботи зі спеціальностями, обробки апеляцій, формування звітів та розмежування прав доступу між різними категоріями користувачів.

Результати тестування підтвердили коректну роботу механізму авторизації. Система успішно визначає роль користувача на підставі введених даних, надає доступ до відповідного функціоналу та відображає повідомлення про помилку у випадку введення некоректних даних. Також підтверджено коректну роботу механізму реєстрації абітурієнтів, збереження персональних даних та автоматичного надання ролі «Абітурієнт» після завершення реєстрації.

У ході тестування ролі «Абітурієнт» підтверджено коректну роботу механізмів перегляду власних заяв, подання апеляцій та видалення заяв. Система успішно контролює належність заяв поточному користувачу, не допускає виконання операцій над чужими записами та здійснює перевірку коректності введених даних під час створення апеляцій.

Перевірка функціональних можливостей ролі «Приймальна комісія» підтвердила коректну роботу механізмів додавання та видалення абітурієнтів, пошуку заяв вступників, зміни статусів заяв та роботи із переліком спеціальностей. Також підтверджено коректне функціонування механізмів зміни кількості бюджетних та контрактних місць, а також засобів фільтрації інформації за різними критеріями.

Під час тестування сторінки спеціальностей встановлено коректну роботу механізмів пошуку спеціальностей, оновлення плану набору та відображення інформації про спеціальності відповідно до обраного факультету. Система забезпечує контроль введених даних та запобігає виконанню некоректних операцій.

Результати тестування сторінки звітів підтвердили коректність формування звітів за заявами вступників та спеціальностями. Система успішно виконує відбір інформації за заданими критеріями та відображає актуальні дані, що зберігаються в базі даних.

Для ролі «Апеляційна комісія» підтверджено коректну роботу механізмів пошуку апеляцій, перегляду їх детальної інформації, зміни статусів та застосування засобів фільтрації. Система забезпечує збереження результатів розгляду апеляцій та їх подальше відображення користувачам.

У процесі тестування ролі «Системний адміністратор» підтверджено працездатність функцій повного керування даними інформаційної системи. Коректно працюють механізми створення, редагування та видалення спеціальностей, керування заявами та апеляціями, а також перевірки цілісності даних під час виконання операцій видалення.

Результати проведеного тестування підтвердили працездатність розробленої інформаційної системи та коректність реалізації всіх передбачених функціональних можливостей. Система забезпечує автоматизацію основних процесів вступної кампанії, підтримує роботу різних категорій користувачів та дозволяє ефективно виконувати обробку інформації, необхідної для діяльності приймальної та апеляційної комісії.

Висновок

У третьому розділі було наведено результати реалізації інформаційної системи для підтримки роботи приймальної комісії. Розглянуто користувацький інтерфейс системи, порядок роботи з основними сторінками та функціональні можливості для абітурієнта, приймальної комісії, апеляційної комісії та системного адміністратора. Проведене тестування підтвердило коректність роботи механізмів авторизації, реєстрації, обробки заяв, керування спеціальностями, розгляду апеляцій, формування звітів і розмежування прав доступу. Отже, реалізоване програмне забезпечення відповідає поставленим вимогам і може використовуватися для автоматизації основних процесів вступної кампанії.

Розділ 4 – Охорона праці

4.1 Аналіз умов праці користувача інформаційної системи

Робота з розробленою інформаційною системою здійснюється за допомогою персонального комп'ютера у приміщенні адміністративного або навчального призначення. Основними користувачами системи є абітурієнти, працівники приймальної комісії, працівники апеляційної комісії та системний адміністратор.

Під час роботи користувачів із персональним комп'ютером можуть виникати такі шкідливі фактори: тривале статичне навантаження, зорове напруження, недостатній рівень освітлення, несприятливі параметри мікроклімату, а також електрична небезпека під час експлуатації комп'ютерної техніки [13].

Робоче місце користувача повинно бути організоване відповідно до вимог чинних нормативних документів. Робочий стіл має забезпечувати достатню площу для розміщення монітора, клавіатури, миші та супутнього обладнання. Конструкція робочого крісла повинна забезпечувати можливість регулювання висоти сидіння та кута нахилу спинки [14].

Монітор необхідно розташовувати на відстані 50–70 см від очей користувача. Верхня межа екрана повинна знаходитися на рівні очей або трохи нижче, що дозволяє зменшити навантаження на шийний відділ хребта та органи зору [14].

4.2 Вимоги до мікроклімату та освітлення

Для забезпечення комфортних умов праці параметри мікроклімату повинні відповідати санітарним нормам. Температура повітря у приміщенні має підтримуватися в межах 22–24 °С у холодний період року та 23–25 °С у теплий період року. Відносна вологість повітря повинна становити 40–60 %, а швидкість руху повітря не повинна перевищувати 0,1 м/с [15].

Освітлення робочих місць повинно забезпечувати можливість тривалої роботи з документами та екраном монітора без надмірного навантаження на органи зору. Для приміщень, обладнаних персональними комп'ютерами,

рекомендований рівень освітленості робочої поверхні становить 300–500 лк [16].

Для запобігання появі відблисків на екрані монітора джерела природного та штучного освітлення необхідно розташовувати таким чином, щоб світлові потоки не потрапляли безпосередньо на екран. Бажано використовувати комбіноване освітлення, що поєднує природне та штучне світло [16].

4.3 Електробезпека та пожежна безпека

Комп'ютерне обладнання повинно експлуатуватися відповідно до вимог електробезпеки. Усі технічні засоби повинні бути справними, мати цілу ізоляцію електричних проводів та підключатися до електромережі через справні розетки із захисним заземленням [17].

Під час роботи забороняється використовувати обладнання з пошкодженими кабелями живлення, виконувати самостійний ремонт електрообладнання або торкатися струмопровідних частин пристроїв [17].

Приміщення, у якому експлуатується інформаційна система, повинно бути забезпечене первинними засобами пожежогасіння. Для гасіння електрообладнання рекомендується використовувати вуглекислотні або порошкові вогнегасники [18].

У разі виникнення аварійної ситуації або загоряння необхідно негайно відключити обладнання від електромережі, повідомити відповідальних осіб та діяти відповідно до плану евакуації [18].

4.4 Організація режиму праці та відпочинку

Тривала робота за персональним комп'ютером може призводити до перевтоми, зниження працездатності та погіршення стану здоров'я працівників. Для зменшення негативного впливу рекомендується дотримуватися раціонального режиму праці та відпочинку [14].

Під час роботи необхідно періодично виконувати перерви для відпочинку очей та зміни положення тіла. Доцільним є виконання простих

вправ для очей, рук та шийного відділу хребта. Такі заходи сприяють зниженню втоми та підвищенню продуктивності праці [14].

Таким чином, дотримання вимог охорони праці під час роботи з розробленою інформаційною системою дозволяє забезпечити безпечні та комфортні умови праці користувачів, зменшити ризик виникнення професійних захворювань та підвищити ефективність виконання робочих завдань.

Висновок

У четвертому розділі було розглянуто основні питання охорони праці під час роботи користувачів із розробленою інформаційною системою. Проаналізовано умови праці користувача персонального комп'ютера, вимоги до організації робочого місця, параметрів мікроклімату та рівня освітлення приміщень. Окрему увагу приділено дотриманню вимог електробезпеки, пожежної безпеки та організації раціонального режиму праці й відпочинку. Отже, виконання наведених вимог дозволяє забезпечити безпечні та комфортні умови роботи користувачів інформаційної системи, знизити ризик перевтоми та підвищити ефективність виконання робочих завдань.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено інформаційну систему приймальної комісії закладу вищої освіти, призначену для автоматизації основних процесів вступної кампанії. Проведений аналіз предметної області дозволив визначити основні категорії користувачів системи, інформаційні потоки та функціональні вимоги до програмного забезпечення.

У процесі виконання роботи було досліджено особливості організації роботи приймальної комісії, процеси подання та обробки заяв вступників, проведення конкурсного відбору, обробки апеляцій та формування звітності. На основі результатів аналізу побудовано DFD-діаграми нульового, першого та другого рівнів, що відображають основні процеси функціонування інформаційної системи та взаємодію користувачів із нею.

Для забезпечення зберігання та обробки інформації розроблено концептуальну, логічну та фізичну моделі бази даних. Визначено основні сутності предметної області, їх атрибути та взаємозв'язки, а також спроектовано структуру бази даних для зберігання інформації про вступників, спеціальності, факультети, результати вступних випробувань, заяви, пільги та апеляції.

У роботі розроблено математичне забезпечення інформаційної системи, що включає алгоритми розрахунку конкурсного бала, формування рейтингових списків вступників, врахування пріоритетів заяв та пільгових категорій під час конкурсного відбору. Запропоновані алгоритми дозволяють автоматизувати процес прийняття рішень та зменшити вплив людського фактора під час обробки результатів вступної кампанії.

Для реалізації програмного забезпечення було обрано мову програмування Java, фреймворк Spring, систему управління базами даних MySQL та сучасні технології створення web-додатків. Реалізовано функціональні можливості для різних категорій користувачів, зокрема

абітурієнтів, працівників приймальної комісії, працівників апеляційної комісії та системного адміністратора. Система забезпечує реєстрацію вступників, подання та обробку заяв, керування спеціальностями, подання та розгляд апеляцій, формування звітів і контроль доступу до функціональних можливостей відповідно до ролі користувача.

Проведене тестування підтвердило працездатність розробленого програмного забезпечення та коректність реалізації передбачених функціональних можливостей. Усі основні операції, пов'язані з обробкою даних вступників, спеціальностей, заяв та апеляцій, виконуються відповідно до визначених вимог.

Отже, поставлену мету дипломної роботи досягнуто. Розроблена інформаційна система забезпечує автоматизацію основних процесів вступної кампанії, підвищує ефективність роботи приймальної комісії та може бути використана як основа для подальшого розвитку і впровадження подібних систем у закладах вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білов В.В., В.І. Чистякова — Проектування інформаційних систем: підручник для студ. установ вищої. проф. Освіта; за ред. В.В. Білова М: Видавничий центр Академія, 2013. 352с.
2. Гагаріна Л.Г., Кокорєва Є.В., Віснадул Б.Д. Технологія розробки програмного забезпечення: навчальний посібник/під ред. Л.Г. Гагаріна. – М.: ВД «ФОРУМ»: ІНФА-М, 2008. – 400 с.: іл. – (Вища освіта).
3. Shklar L., Rosen R. Web Application Architecture: Principles, Protocols and Practices. — 2nd ed. — Wiley, 2009
4. Craig Walls - Spring in Action 4th Edition – 2014
5. Грекул В.І., Г.М. Денищенко. Проектування інформаційних систем: курс лекцій. М. Інтернет - Ун-т Інформ. технологій, 2005. 304с.
6. Клевцов С.І. Аналіз та формування вимог до програмного забезпечення інформаційних систем збору та обробки даних: навчальний посібник. Таганрог: Вид-во ТТЮФ, 2007. 100с.
7. Марченко А.В. Проектування інформаційних систем 2016 90с. Режим доступу:
https://elearning.sumdu.edu.ua/free_content/lectured:delc9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf
8. Пальмов С.В. Проектування інформаційних систем. М: ІСТ-72, 2011. 51 с.
9. Ременяк Л.В. Проектування інформаційних систем: конспект лекцій. Одеса, Одеський державний екологічний університет, 2016. 152.
10. Хомоненко О.Д., Циганков В.М., Мальцев М.Г. Бази даних: Підручник для вищих навчальних закладів/Під ред. Проф. А.Д. Хомоненко. – 6-те вид.дод. – СПб.: КОРОНА – Вік. 2009. – 736 с.

11. Положення про приймальну комісію вищого навчального закладу Наказ Міністерства освіти і науки України 15.10.2015 № 1085 –

<https://zakon.rada.gov.ua/laws/main/z1353-15>

12. Авраменко В. С., Авраменко А. С. Проектування інформаційних систем : навчальний посібник. – Черкаси : ЧНУ ім. Б. Хмельницького, 2017. – 434 с.

13. Закон України «Про охорону праці».

<https://zakon.rada.gov.ua/laws/main/2694-12>

14. НПАОП 0.00-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин.

<https://zakon.rada.gov.ua/laws/main/z0293-10>

15. ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень.

<https://zakon.rada.gov.ua/rada/show/va042282-99#Text>

16. ДБН В.2.5-28:2018. Природне і штучне освітлення.

https://zakon.isu.net.ua/sites/default/files/normdocs/dbn_v_2.5-28_2018.pdf

17. НПАОП 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів.

<https://dnaop.com/html/2029/doc->

[_%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_40.1-1.21-98](https://dnaop.com/html/2029/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_40.1-1.21-98)

18. Кодекс цивільного захисту України. Розділ щодо пожежної безпеки та забезпечення евакуації людей.

<https://zakon.rada.gov.ua/laws/show/5403-17#Text>

ДОДАТКИ

Додаток А – технічне завдання

Призначення інформаційної системи:

- автоматизація процесів прийому та обробки заяв вступників;
- автоматизація формування рейтингових списків абітурієнтів;
- забезпечення зручного доступу до інформації про результати вступної кампанії;
- підтримка процесів обробки апеляцій та прийняття рішень за результатами їх розгляду;
- надання працівникам приймальної комісії інструментів для керування даними про вступників, спеціальності, пільги та результати конкурсного відбору;
- автоматизація формування звітної інформації щодо результатів вступної кампанії.

Мета розробки:

- створення єдиного програмного засобу для автоматизації діяльності приймальної комісії закладу вищої освіти;
- зменшення обсягу ручної обробки інформації під час проведення вступної кампанії;
- підвищення швидкості обробки заяв вступників та формування рейтингових списків;
- забезпечення централізованого зберігання даних про вступників, спеціальності та результати вступної кампанії;
- підвищення ефективності роботи приймальної та апеляційної комісій;
- отримання практичних навичок розробки web-додатків із використанням технологій Java, Spring, MySQL, HTML, CSS та REST.

Користувачі системи:

Абітурієнт — користувач системи, який має можливість реєструватися в системі, переглядати власні заяви, подавати апеляції та виконувати інші операції, пов'язані з участю у вступній кампанії.

Працівник приймальної комісії — користувач, який здійснює реєстрацію вступників, обробку заяв, зміну статусів заяв, керування інформацією про спеціальності та формування звітів.

Працівник апеляційної комісії — користувач, який здійснює розгляд апеляцій, перегляд інформації щодо поданих апеляцій та прийняття рішень за результатами їх розгляду.

Системний адміністратор — користувач із повними правами доступу до інформаційної системи. До його функцій належать керування довідковими даними, спеціальностями, заявами, апеляціями, контроль цілісності інформації та адміністрування системи.

Додаток Б – фрагменти програмного коду

На лістингу 1 наведено SQL-скрипт створення бази даних інформаційної системи. Скрипт містить команди створення таблиць, визначення первинних та зовнішніх ключів, а також інших обмежень цілісності даних, необхідних для функціонування системи.

Лістинг 1 – скрипт для створення бд у MySQL

```
-- MySQL dump 10.13  Distrib 8.0.40, for Win64 (x86_64)
--
-- Host: 127.0.0.1    Database: university_admissionscommittee
-- -----
-- Server version      8.0.40

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0
*/;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `appeal`
--

DROP TABLE IF EXISTS `appeal`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `appeal` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `specialty_for_applicant_id` bigint NOT NULL,
  `message` varchar(1000) NOT NULL,
  `appeal_status` varchar(50) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_appeal_specialty_for_applicant`
  (`specialty_for_applicant_id`),
  CONSTRAINT `fk_appeal_specialty_for_applicant` FOREIGN KEY
  (`specialty_for_applicant_id`) REFERENCES
  `specialty_for_applicant` (`id`) ON DELETE CASCADE,
```

```

        CONSTRAINT `appeal_chk_1` CHECK ((`appeal_status` in
(_utf8mb4'PENDING',_utf8mb4'APPROVED',_utf8mb4'REJECTED'))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `applicant`
--

DROP TABLE IF EXISTS `applicant`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `applicant` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `first_name` varchar(100) NOT NULL,
  `last_name` varchar(100) NOT NULL,
  `email` varchar(254) NOT NULL,
  `phone_number` varchar(32) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  UNIQUE KEY `unique_phone_number` (`phone_number`)
) ENGINE=InnoDB AUTO_INCREMENT=96 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `benefit`
--

DROP TABLE IF EXISTS `benefit`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `benefit` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `quota_type` varchar(20) DEFAULT 'NONE',
  `priority_points` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `benefit_chk_1` CHECK ((`quota_type` in
(_utf8mb4'NONE',_utf8mb4'QUOTA_1',_utf8mb4'QUOTA_2'))
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `benefit_for_applicant`
--

DROP TABLE IF EXISTS `benefit_for_applicant`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;

```

```

/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `benefit_for_applicant` (
  `applicant_id` bigint NOT NULL AUTO_INCREMENT,
  `benefit_id` bigint DEFAULT NULL,
  KEY `fk_bfa_applicant` (`applicant_id`),
  KEY `fk_bfa_benefit` (`benefit_id`),
  CONSTRAINT `fk_bfa_applicant` FOREIGN KEY (`applicant_id`)
REFERENCES `applicant` (`id`),
  CONSTRAINT `fk_bfa_benefit` FOREIGN KEY (`benefit_id`)
REFERENCES `benefit` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=94 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `exam_result`
--

DROP TABLE IF EXISTS `exam_result`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `exam_result` (
  `applicant_id` bigint NOT NULL,
  `subject_id` bigint NOT NULL,
  `result` int NOT NULL,
  `id` bigint NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_exam_pair` (`applicant_id`,`subject_id`),
  KEY `fk_exam_applicant` (`applicant_id`),
  KEY `fk_exam_subject` (`subject_id`),
  CONSTRAINT `fk_exam_applicant` FOREIGN KEY (`applicant_id`)
REFERENCES `applicant` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT `fk_exam_subject` FOREIGN KEY (`subject_id`)
REFERENCES `subject` (`id`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=192 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `faculty`
--

DROP TABLE IF EXISTS `faculty`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `faculty` (
  `name` varchar(255) DEFAULT NULL,
  `id` bigint NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)

```

```

) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `specialty`
--

DROP TABLE IF EXISTS `specialty`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `specialty` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `number` int NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `budget_places_general` int NOT NULL,
  `number_of_contract_places` int NOT NULL,
  `faculty_id` bigint DEFAULT NULL,
  `budget_places_quota1` int DEFAULT NULL,
  `budget_places_quota2` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `unique_number` (`number`),
  UNIQUE KEY `unique_name` (`name`),
  KEY `fk_specialty_faculty` (`faculty_id`),
  CONSTRAINT `fk_specialty_faculty` FOREIGN KEY (`faculty_id`)
REFERENCES `faculty` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `specialty_for_applicant`
--

DROP TABLE IF EXISTS `specialty_for_applicant`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `specialty_for_applicant` (
  `applicant_id` bigint NOT NULL,
  `specialty_id` bigint NOT NULL,
  `applicant_status` varchar(50) DEFAULT NULL,
  `priority` int NOT NULL,
  `id` bigint NOT NULL AUTO_INCREMENT,
  `submission_date` date DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_sfa_pair` (`applicant_id`,`specialty_id`),
  KEY `applicant_id` (`applicant_id`),
  KEY `specialty_id` (`specialty_id`),
  CONSTRAINT `fk_sfa_applicant` FOREIGN KEY (`applicant_id`)
REFERENCES `applicant` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE,

```

```

        CONSTRAINT `fk_sfa_specialty` FOREIGN KEY (`specialty_id`)
REFERENCES `specialty` (`id`) ON DELETE RESTRICT ON UPDATE
CASCADE,
        CONSTRAINT `chk_applicant_status` CHECK ((`applicant_status`
in
(_utf8mb4'PENDING',_utf8mb4'ACCEPTED_BUDGET',_utf8mb4'ACCEPTED_C
ONTRACT'))))
) ENGINE=InnoDB AUTO_INCREMENT=37 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `subject`
--

DROP TABLE IF EXISTS `subject`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `subject` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uq_subject_name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `subject_for_specialty`
--

DROP TABLE IF EXISTS `subject_for_specialty`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `subject_for_specialty` (
  `subject_id` bigint DEFAULT NULL,
  `specialty_id` bigint DEFAULT NULL,
  `id` bigint NOT NULL AUTO_INCREMENT,
  `weight` double DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `subject_for_specialty_fk1` (`subject_id`),
  KEY `fk_sfs_specialty` (`specialty_id`),
  CONSTRAINT `fk_sfs_specialty` FOREIGN KEY (`specialty_id`)
REFERENCES `specialty` (`id`),
  CONSTRAINT `subject_for_specialty_fk1` FOREIGN KEY
(`subject_id`) REFERENCES `subject` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=30 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

```

```

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
*/;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

```

На лістингу 2 наведено програмний код JPA-сутностей, що реалізують відображення таблиць бази даних на об'єкти предметної області.

Лістинг 2 – основні JPA-сутності, що відображають таблиці з бд та за допомогою яких реалізовується робота із таблицями бд

```

@Getter
@Setter
@Entity
@Table(name = "appeal")
public class Appeal {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @NotNull
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JoinColumn(name = "specialty_for_applicant_id", nullable =
false)
    private SpecialtyForApplicant specialtyForApplicant;

    @Size(max = 1000)
    @NotNull
    @Column(name = "message", nullable = false, length = 1000)
    private String message;

    @Enumerated(EnumType.STRING)
    @Column(name = "appeal_status", nullable = false, length =
50)
    private AppealStatus appealStatus = AppealStatus.PENDING;

    protected Appeal() {}

    public Appeal(SpecialtyForApplicant specialtyForApplicant,
String message) {
        this.specialtyForApplicant = specialtyForApplicant;
        this.message = message;
    }
}

```



```

@Entity
@Table(
    name = "applicant",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_applicant_email",
columnNames = "email"),
        @UniqueConstraint(name = "uq_applicant_phone",
columnNames = "phone_number")
    },
    indexes = {
        @Index(name = "idx_applicant_email", columnList
= "email"),
        @Index(name = "idx_applicant_phone", columnList
= "phone_number")
    }
)

public class Applicant {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank
    @Size(max = 100)
    @Column(name = "first_name", nullable = false, length = 100)
    private String firstName;

    @NotBlank
    @Size(max = 100)
    @Column(name = "last_name", nullable = false, length = 100)
    private String lastName;

    @NotBlank
    @Email
    @Column(name = "email", nullable = false, length = 254)
    private String email;

    @NotBlank
    @Size(max = 32)
    @Column(name = "phone_number", nullable = false, length =
32)
    private String phoneNumber;

    @OneToMany(
        mappedBy = "applicant",
        cascade = CascadeType.ALL,
        orphanRemoval = true,
        fetch = FetchType.LAZY
    )
    private Set<ExamResult> examResults = new HashSet<>();
}

```

```

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(
        name = "benefit_for_applicant",
        joinColumns = @JoinColumn(name = "applicant_id"),
        inverseJoinColumns = @JoinColumn(name =
"benefit_id")
    )
    private Set<Benefit> benefits = new HashSet<>();

    @OneToMany(
        mappedBy = "applicant",
        cascade = CascadeType.ALL,
        orphanRemoval = true,
        fetch = FetchType.LAZY
    )
    private Set<SpecialtyForApplicant> specialties = new
HashSet<>();

    protected Applicant() {}

    public Applicant(String firstName, String lastName, String
email, String phoneNumber) {
        this.firstName = Objects.requireNonNull(firstName,
"firstName");
        this.lastName = Objects.requireNonNull(lastName,
"lastName");
        this.email = Objects.requireNonNull(email, "email");
        this.phoneNumber = Objects.requireNonNull(phoneNumber,
"phoneNumber");
    }

    /* =====
    Bidirectional helpers
    ===== */

    public void addExamResult(ExamResult examResult) {
        Objects.requireNonNull(examResult, "examResult");
        examResults.add(examResult);
        examResult.setApplicant(this);
    }

    public void removeExamResult(ExamResult examResult) {
        Objects.requireNonNull(examResult, "examResult");
        if (examResults.remove(examResult)) {
            examResult.setApplicant(null);
        }
    }

    public void addBenefit(Benefit benefit) {

```

```

        Objects.requireNonNull(benefit, "benefit");
        benefits.add(benefit);
    }

    public void removeBenefit(Benefit benefit) {
        Objects.requireNonNull(benefit, "benefit");
        benefits.remove(benefit);
    }

    public void addBenefits(Collection<Benefit> toAdd) {
        Objects.requireNonNull(toAdd, "benefits");
        toAdd.forEach(this::addBenefit);
    }

    public void addSpecialty(SpecialtyForApplicant sfa) {
        Objects.requireNonNull(sfa, "specialtyForApplicant");
        specialties.add(sfa);
        sfa.setApplicant(this);
    }

    public void removeSpecialty(SpecialtyForApplicant sfa) {
        Objects.requireNonNull(sfa, "specialtyForApplicant");
        if (specialties.remove(sfa)) {
            sfa.setApplicant(null);
        }
    }

    /* =====
       Domain operations
       ===== */

    public int getResultBySubjectId(Long subjectId) {
        Objects.requireNonNull(subjectId, "subjectId");
        return examResults.stream()
            .filter(er -> {
                var subject = er.getSubject();
                return subject != null &&
subjectId.equals(subject.getId());
            })
            .mapToInt(ExamResult::getResult)
            .findFirst()
            .orElseThrow(() -> new
IllegalStateException("Exam result not found for subjectId=" +
subjectId));
    }

    public Optional<ExamResult> findExamResultBySubjectId(Long
subjectId) {
        Objects.requireNonNull(subjectId, "subjectId");
        return examResults.stream()
            .filter(er -> {

```

```

        var subject = er.getSubject();
        return subject != null &&
subjectId.equals(subject.getId());
    })
    .findFirst();
}

/* =====
   Getters / Setters
   ===== */

public Long getId() { return id; }

public String getFirstName() { return firstName; }
public void setFirstName(String firstName) {
    this.firstName = Objects.requireNonNull(firstName,
"firstName");
}

public String getLastName() { return lastName; }
public void setLastName(String lastName) {
    this.lastName = Objects.requireNonNull(lastName,
"lastName");
}

public String getEmail() { return email; }
public void setEmail(String email) {
    this.email = Objects.requireNonNull(email, "email");
}

public String getPhoneNumber() { return phoneNumber; }
public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = Objects.requireNonNull(phoneNumber,
"phoneNumber");
}

public Set<ExamResult> getExamResults() {
    return examResults;
}

public Set<Benefit> getBenefits() {
    return benefits;
}

public Set<SpecialtyForApplicant> getSpecialties() {
    return specialties;
}

/* =====
   equals / hashCode / toString
   ===== */

```

```

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Applicant other)) return false;
    return id != null && id.equals(other.id);
}

@Override
public int hashCode() {
    return getClass().hashCode();
}

@Override
public String toString() {
    return "Applicant{id=" + id +
        ", firstName='" + firstName + '\'' +
        ", lastName='" + lastName + '\'' +
        ", email='" + email + '\'' +
        ", phoneNumber='" + phoneNumber + '\'' +
        '}';
}
}

@Entity
@Table(
    name = "benefit",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_benefit_name",
columnNames = "name")
    }
)
public class Benefit {

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

    @NotBlank
    @Size(max = 255)
    @Column(name = "name", nullable = false, length = 255)
    private String name;

    @Min(0)
    @Column(name = "priority_points", nullable = true)
    private int priorityPoints;

    @Enumerated(EnumType.STRING)
    @Column(name = "quota_type", nullable = true, length = 32)
    private QuotaType type = QuotaType.NONE;

```

```

protected Benefit() {}

public Benefit(String name, int priorityPoints) {
    this.name = Objects.requireNonNull(name, "name");
    this.priorityPoints = priorityPoints;
}

public Long getId() {
    return id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = Objects.requireNonNull(name, "name");
}

public int getPriorityPoints() {
    return priorityPoints;
}

public void setPriorityPoints(int additionalPoints) {
    this.priorityPoints = additionalPoints;
}

public QuotaType getType() {
    return type;
}

public void setType(QuotaType type) {
    this.type = type;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Benefit other)) return false;
    return id != null && id.equals(other.id);
}

@Override
public int hashCode() {
    return getClass().hashCode();
}

@Override
public String toString() {
    return "Benefit{id=" + id +

```

```

        ", name='" + name + '\'' +
        ", additionalPoints=" + priorityPoints +
        '}';
    }
}

```

```

@Entity
@Table(
    name = "exam_result",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_exam_pair",
columnNames = {"applicant_id", "subject_id"})
    },
    indexes = {
        @Index(name = "idx_exam_applicant", columnList =
"applicant_id"),
        @Index(name = "idx_exam_subject", columnList =
"subject_id")
    }
)
public class ExamResult {

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

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "applicant_id", nullable = false)
    private Applicant applicant;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "subject_id", nullable = false)
    private Subject subject;

    @Min(0)
    @Max(200)
    @Column(name = "result", nullable = false)
    private int result;

    protected ExamResult() {
    }

    public ExamResult(Applicant applicant, Subject subject, int
result) {
        this.applicant = Objects.requireNonNull(applicant,
"applicant");
        this.subject = Objects.requireNonNull(subject,
"subject");
    }
}

```

```

        this.result      = result;
    }

    public static ExamResult link(Applicant applicant, Subject
subject, int result) {
        ExamResult er = new ExamResult(applicant, subject,
result);
        applicant.addExamResult(er);
        subject._addExamResult(er);
        return er;
    }

    public void linkApplicant(Applicant applicant) {
        this.applicant = Objects.requireNonNull(applicant,
"applicant");
        applicant.addExamResult(this);
    }

    public void linkSubject(Subject subject) {
        this.subject = Objects.requireNonNull(subject,
"subject");
        subject._addExamResult(this);
    }

    /* ===== Getters / Setters ===== */

    public Long getId() { return id; }

    public Applicant getApplicant() { return applicant; }
    public void setApplicant(Applicant applicant) {
        this.applicant = Objects.requireNonNull(applicant,
"applicant");
    }

    public Subject getSubject() { return subject; }
    public void setSubject(Subject subject) {
        this.subject = Objects.requireNonNull(subject,
"subject");
    }

    public int getResult() { return result; }
    public void setResult(int result) { this.result = result; }

    /* ===== equals/hashCode/toString ===== */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof ExamResult other)) return false;
        return id != null && id.equals(other.id);
    }

```



```

@Override
public int hashCode() { return getClass().hashCode(); }

@Override
public String toString() {
    return "ExamResult{id=" + id +
        ", applicantId=" + (applicant != null ?
applicant.getId() : null) +
        ", subjectId=" + (subject != null ?
subject.getId() : null) +
        ", result=" + result + "}";
}
}

@Entity
@Table(
    name = "faculty",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_faculty_name",
columnNames = "name")
    }
)
public class Faculty {

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

    @NotBlank
    @Size(max = 255)
    @Column(name = "name", nullable = false, length = 255)
    private String name;

    @OneToMany(
        mappedBy = "faculty",
        cascade = { CascadeType.PERSIST, CascadeType.MERGE
},
        orphanRemoval = false,
        fetch = FetchType.LAZY
    )
    private Set<Specialty> specialties = new HashSet<>();

    protected Faculty() {}

    public Faculty(String name) {
        this.name = Objects.requireNonNull(name, "name");
    }
}

```

```

    public void addSpecialty(Specialty specialty) {
        Objects.requireNonNull(specialty, "specialty");
        specialties.add(specialty);
        specialty.setFaculty(this);
    }

    public void removeSpecialty(Specialty specialty) {
        Objects.requireNonNull(specialty, "specialty");
        if (specialties.remove(specialty)) {
            specialty.setFaculty(null);
        }
    }

    public Long getId() { return id; }

    public String getName() { return name; }
    public void setName(String name) {
        this.name = Objects.requireNonNull(name, "name");
    }

    public Set<Specialty> getSpecialties() {
        return Collections.unmodifiableSet(specialties);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Faculty other)) return false;
        return id != null && id.equals(other.id);
    }

    @Override
    public int hashCode() { return getClass().hashCode(); }

    @Override
    public String toString() {
        return "Faculty id=" + id + ", name='" + name;
    }
}

@Entity
@Table(
    name = "specialty",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_specialty_name",
columnNames = "name"),
        @UniqueConstraint(name = "uq_specialty_number",
columnNames = "number")
    }
)

```

```

)
public class Specialty {

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

    @NotBlank
    @Size(max = 255)
    @Column(name = "name", nullable = false, length = 255)
    private String name;

    @Column(name = "number", nullable = false)
    private int number;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "faculty_id", nullable = false)
    private Faculty faculty;

    @Column(name = "budget_places_quota1", nullable = false)
    private int budgetPlacesQuota1;

    @Column(name = "budget_places_quota2", nullable = false)
    private int budgetPlacesQuota2;

    @Column(name = "budget_places_general", nullable = false)
    private int budgetPlacesGeneral;

    @Column(name = "number_of_contract_places", nullable =
false)
    private int numberOfContractPlaces;

    @OneToMany(mappedBy = "specialty", cascade =
CascadeType.ALL, orphanRemoval = true)
    private Set<SubjectForSpecialty> subjectWeights = new
HashSet<>();

    @OneToMany(mappedBy = "specialty", cascade =
CascadeType.ALL, orphanRemoval = true, fetch = FetchType.LAZY)
    private Set<SpecialtyForApplicant> applicants = new
HashSet<>();

    protected Specialty() {
    }

    public Specialty(String name, int number, Faculty faculty,
int budgetPlacesGeneral, int
numberOfContractPlaces) {
        this.name = Objects.requireNonNull(name, "name");
        this.number = number;
    }

```

```

        this.faculty = Objects.requireNonNull(faculty,
"faculty");
        this.budgetPlacesGeneral = budgetPlacesGeneral;
        this.numberOfContractPlaces = numberOfContractPlaces;
    }

```

```

    public void addSubject(Subject subject, Double weight) {
        SubjectForSpecialty sfs = new
SubjectForSpecialty(weight, this, subject);
        subjectWeights.add(sfs);
        subject.getSpecialtyWeights().add(sfs);
    }

```

```

    public void removeSubject(Subject subject) {
        for (Iterator<SubjectForSpecialty> iterator =
subjectWeights.iterator(); iterator.hasNext(); ) {
            SubjectForSpecialty sfs = iterator.next();
            if (sfs.getSpecialty().equals(this) &&
sfs.getSubject().equals(subject)) {
                iterator.remove();

```

```

sfs.getSubject().getSpecialtyWeights().remove(sfs);
                sfs.setSpecialty(null);
                sfs.setSubject(null);
                break;
            }
        }
    }

```

```

    public boolean hasNoApplicants() {
        return applicants.isEmpty();
    }

```

```

    public Long getId() { return id; }

```

```

    public String getName() { return name; }
    public void setName(String name) {
        this.name = Objects.requireNonNull(name, "name");
    }

```

```

    public int getNumber() { return number; }
    public void setNumber(int number) { this.number = number; }

```

```

    public Faculty getFaculty() { return faculty; }
    public void setFaculty(Faculty faculty) {
        this.faculty = Objects.requireNonNull(faculty,
"faculty");
    }

```

```

    }

    public int getBudgetPlacesGeneral() { return
budgetPlacesGeneral; }
    public void setBudgetPlacesGeneral(int numberOfBudgetPlaces)
{
        this.budgetPlacesGeneral = numberOfBudgetPlaces;
    }

    public int getNumberOfContractPlaces() { return
numberOfContractPlaces; }
    public void setNumberOfContractPlaces(int
numberOfContractPlaces) {
        this.numberOfContractPlaces = numberOfContractPlaces;
    }

    public Set<SubjectForSpecialty> getSubjectWeights() {
        return subjectWeights;
    }

    public Set<SpecialtyForApplicant> getApplicants() {
        return Collections.unmodifiableSet(applicants);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Specialty other)) return false;
        return id != null && id.equals(other.id);
    }

    @Override
    public int hashCode() { return getClass().hashCode(); }

    @Override
    public String toString() {
        return "Specialty{id=\"" + id + "\", name='" + name + "',
number=\"" + number + "\"}";
    }
}

@Entity
@Table(
    name = "specialty_for_applicant",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_sfa_pair",
columnNames = {"applicant_id", "specialty_id"}),
        @UniqueConstraint(name = "uq_sfa_priority",
columnNames = {"applicant_id", "priority"})
    }
)

```

```

        }
    )
    public class SpecialtyForApplicant {

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

        @ManyToOne(fetch = FetchType.LAZY, optional = false)
        @JoinColumn(name = "specialty_id", nullable = false)
        private Specialty specialty;

        @ManyToOne(fetch = FetchType.LAZY, optional = false)
        @JoinColumn(name = "applicant_id", nullable = false)
        private Applicant applicant;

        @NotNull
        @Min(1)
        @Column(name = "priority", nullable = false)
        private Integer priority;

        @Enumerated(EnumType.STRING)
        @Column(name = "applicant_status", nullable = false, length
= 32)
        private ApplicantStatus applicantStatus =
ApplicantStatus.PENDING;

        @Column(name = "submission_date", nullable = false, length =
32)
        private LocalDate date = LocalDate.now();

        protected SpecialtyForApplicant() {
        }

        public SpecialtyForApplicant(Applicant applicant, Specialty
specialty, Integer priority) {
            this.applicant = Objects.requireNonNull(applicant,
"applicant");
            this.specialty = Objects.requireNonNull(specialty,
"specialty");
            this.priority = Objects.requireNonNull(priority,
"priority");
        }

        public static SpecialtyForApplicant link(Applicant
applicant, Specialty specialty, Integer priority) {
            SpecialtyForApplicant sfa = new
SpecialtyForApplicant(applicant, specialty, priority);
            applicant.addSpecialty(sfa);
        }
    }
}

```

```

        return sfa;
    }

    /* ===== Getters / Setters ===== */

    public Long getId() { return id; }

    public Specialty getSpecialty() { return specialty; }
    public void setSpecialty(Specialty specialty) {
        this.specialty = Objects.requireNonNull(specialty,
"specialty");
    }

    public Applicant getApplicant() { return applicant; }
    public void setApplicant(Applicant applicant) {
        this.applicant = Objects.requireNonNull(applicant,
"applicant");
    }

    public Integer getPriority() { return priority; }
    public void setPriority(Integer priority) {
        this.priority = Objects.requireNonNull(priority,
"priority");
    }

    public ApplicantStatus getApplicantStatus() { return
applicantStatus; }
    public void setApplicantStatus(ApplicantStatus
applicantStatus) {
        this.applicantStatus =
Objects.requireNonNull(applicantStatus, "applicantStatus");
    }

    public LocalDate getDate() {
        return date;
    }

    /* ===== equals/hashCode/toString ===== */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof SpecialtyForApplicant other)) return
false;
        return id != null && id.equals(other.id);
    }

    @Override
    public int hashCode() { return getClass().hashCode(); }

    @Override

```

```

        public String toString() {
            return "SpecialtyForApplicant{id=" + id +
                ", applicantId=" + (applicant != null ?
applicant.getId() : null) +
                ", specialtyId=" + (specialty != null ?
specialty.getId() : null) +
                ", priority=" + priority +
                ", status=" + applicantStatus +
                "}";
        }
    }

@Entity
@Table(
    name = "subject",
    uniqueConstraints = {
        @UniqueConstraint(name = "uq_subject_name",
columnNames = "name")
    },
    indexes = {
        @Index(name = "idx_subject_name", columnList =
"name")
    }
)
public class Subject {

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

    @NotBlank
    @Size(max = 255)
    @Column(name = "name", nullable = false, length = 255)
    private String name;

    @OneToMany(mappedBy = "subject", cascade = CascadeType.ALL,
orphanRemoval = true)
    private Set<SubjectForSpecialty> specialtyWeights = new
HashSet<>();

    @OneToMany(mappedBy = "subject", fetch = FetchType.LAZY)
    private final Set<ExamResult> examResults = new HashSet<>();

    protected Subject() { /* JPA */ }

    public Subject(String name) {
        this.name = Objects.requireNonNull(name, "name");
    }
}

```



```

        public Set<SubjectForSpecialty> getSpecialtyWeights() {
            return specialtyWeights;
        }

        public void setSpecialtyWeights(Set<SubjectForSpecialty>
specialtyWeights) {
            this.specialtyWeights = specialtyWeights;
        }

        void _addExamResult(ExamResult examResult) {
            examResults.add(Objects.requireNonNull(examResult,
"examResult"));
        }

        void _removeExamResult(ExamResult examResult) {
            examResults.remove(Objects.requireNonNull(examResult,
"examResult"));
        }

        /* ===== Getters / Setters ===== */

        public Long getId() { return id; }

        public String getName() { return name; }

        public void setName(String name) { this.name =
Objects.requireNonNull(name, "name"); }

        public Set<ExamResult> getExamResults() {
            return Collections.unmodifiableSet(examResults);
        }

        /* ===== equals/hashCode/toString ===== */

        @Override
        public boolean equals(Object o) {
            if (this == o) return true;
            if (!(o instanceof Subject other)) return false;
            return id != null && id.equals(other.id);
        }

        @Override
        public int hashCode() { return getClass().hashCode(); }

        @Override
        public String toString() {
            return "Subject{id=" + id + ", name='" + name + "'}";
        }
    }
}

```

```

@Getter
@Setter
@Entity
@Table(name = "subject_for_specialty")
public class SubjectForSpecialty {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "subject_id")
    private Subject subject;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "specialty_id")
    private Specialty specialty;

    @Column(name = "weight")
    private Double weight;

    public SubjectForSpecialty(Double weight,
                               Specialty specialty,
                               Subject subject) {
        this.weight = weight;
        this.specialty = specialty;
        this.subject = subject;
    }

    protected SubjectForSpecialty() {
    }
}

```

Додаток В – реалізація програмної частини та результати роботи системи

Хто я?

Увійти

Авторизація

Введіть пароль

password

login

logout

register

Тестові облікові записи

admin: admin

committee: committee

appeal-committee: appeal-committee

applicant: enter email

Рисунок В.1 – початкова сторінка із визначенням користувача, тестові облікові записи та паролі до них, можливість вийти через пароль, вийти із ролі, зареєструвати нового абітурієнта та автоматично виконати вхід

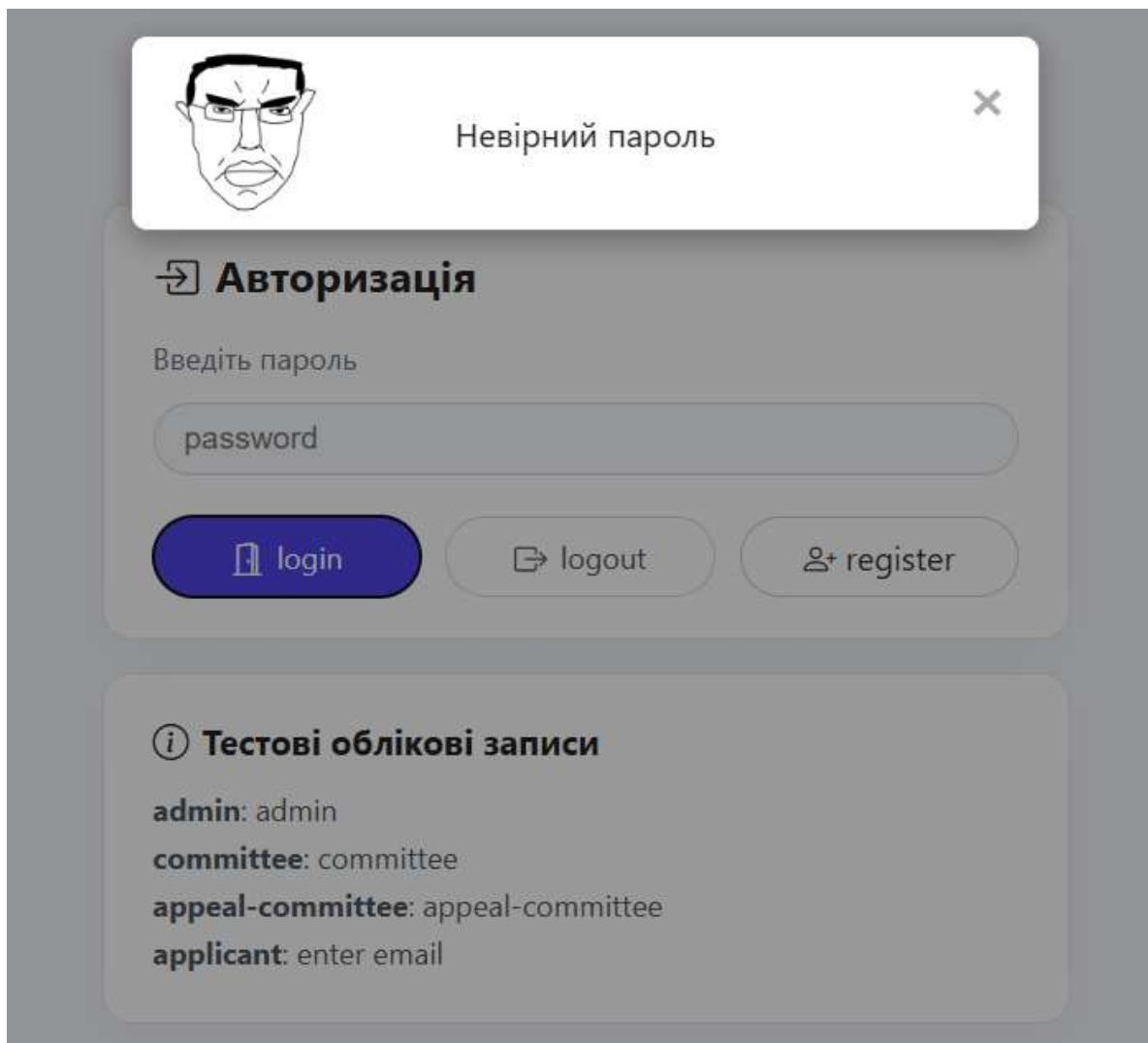


Рисунок В.2 – повідомлення про помилку при неправильному
паролі

👤

Реєстрація

Ім'я

First name

Прізвище

Second name

Email

Email

Телефон

+380XXXXXXXXX

Екзамени і результати:

Англійська мова

Біологія

Географія

Історія України

Математика

Українська мова

Фізика

Хімія

Пільги:

☐ Дитина учасника бойових дій (Загальний конкурс)

☐ Переможець олімпіади (Загальний конкурс)

☐ Особа з інвалідністю (Загальний конкурс)

Доступні спеціальності:

Додати абітурієнта

Рисунок В.3 – сторінка для реєстрацій, на якій розміщено форму додання нового абітурієнта, після додання до бд автоматично виконається у систему

Реєстрація

Ім'я:

Прізвище:

Email:

Телефон:

Екзамен і результати:

Англійська мова 160	Біологія
Географія 180	Історія України 192
Математика 190	Українська мова 192
Фізика 	Хімія

Пільги:

- ☐ Дитина учасника бойових дій (Загальний конкурс)
- ☐ Переможець олімпіади (Загальний конкурс)
- ☒ Особа з інвалідністю (Загальний конкурс)

Доступні спеціальності:

- ☒ Кібербезпека (пріоритет: 1)
- ☒ Комп'ютерні науки (пріоритет: 2)
- ☒ Міжнародне право (пріоритет: 3)
- ☐ Правознавство

[Додати абітурієнта](#)

Рисунок В.4 – форма для додання абітурієнта із динамічним формуванням списку доступних спеціальностей в залежності від результатів екзаменів, динамічним формуванням пріоритетів на обрані спеціальності



номер телефона вже зайнятий



Така пошта вже зайнята



Оберіть хоча б 1 спеціальність

Рисунок В.5 – валідація при введенні вже зайнятої пошти, номера телефону, спробі не обрати жодної спеціальності

🔍 Переглянути свої заяви

🗑️ Видалити заяву

ID

🗑️ Видалити

🛡️ Подати апеляцію

Id заяви

Текст апеляції

🛡️ Подати апеляцію

📊 Таблиця абітурієнтів

Фармація

▼

Квота 1

▼

Застосувати фільтри

Фармація

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус	Біологія	Хімія
----------	----------------	------	----------	---------	-------	--------------	-------------	--------	----------	-------

Рисунок В.6 – сторінка для ролі абітурієнт із формою перегляду власних заяв, видалення своїх заяв, формою подачі апеляції

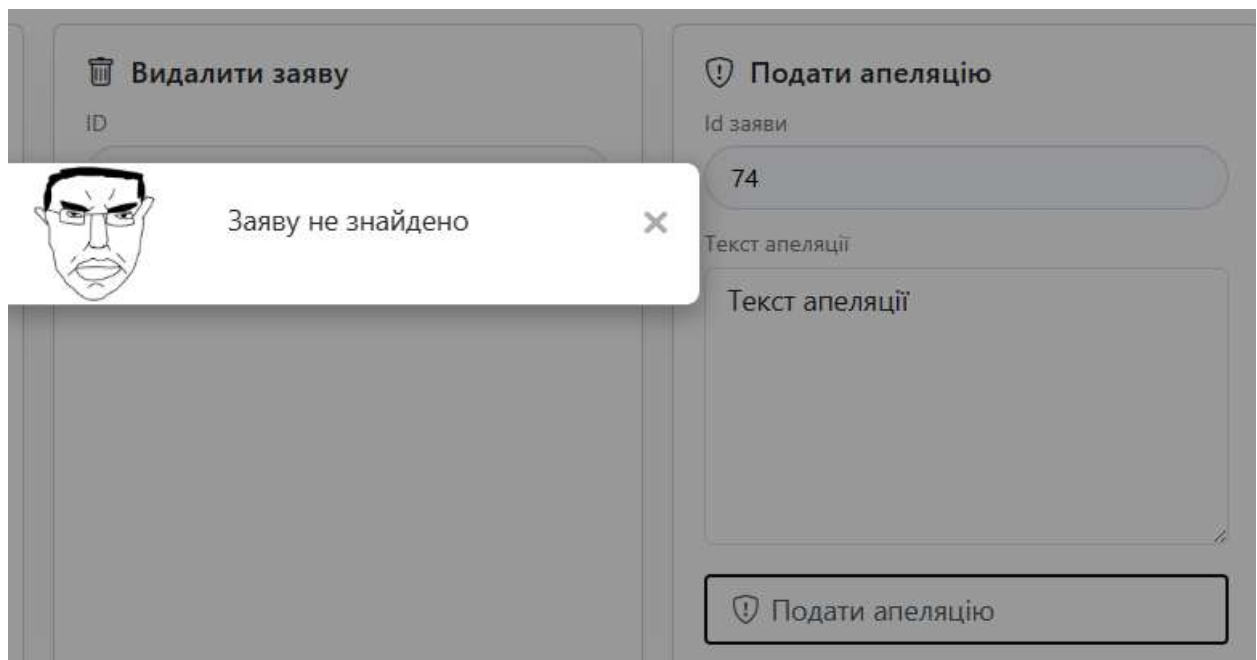


Рисунок В.9 – помилка при спробі подати апеляцію не на свою заяву

Таблиця абітурієнтів

Комп'ютерні науки Загальний конкурс Застосувати фільтри

Комп'ютерні науки

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус
4	74	0000000000	0000000000	+380502929633	00@gmail.com	0.00	2026-05-19	Прийнято на бюджет
5	76	000222	00022	+380995695902	02@gmail.com	0.00	2026-05-05	Невизначено
8	77	000333	000333	+380995695933	03@gmail.com	0.00	2026-05-20	Невизначено
27	90	4141	Луценко	+380003242578	lucar@gmail.com	0.00	2026-06-13	Невизначено
16	83	dfsfsdf	xc	+380003242145	13539@nuos.edu.ua	0.00	2026-05-01	Невизначено

Рисунок В.10 – фільтрація заяв у вигляді випадаючих списків спеціальностей та квот, на кожну зміну значення одного з випадаючих списків виконується фільтрація, при натисканні на кнопку “Застосувати фільтри” виконується фільтрація за значеннями 1-го і 2-го випадаючих списків

Додати абітурієнта

Ім'я

First name

Прізвище

Second name

Email

Email

Телефон

+380XXXXXXXX

Екзамени і результати:

Англійська мова

Біологія

Географія

Історія України

Математика

Українська мова

Фізика

Хімія

Пільги:

☐ Дитина учасника бойових дій (Загальний конкурс)
 ☐ Переможець олімпіади (Загальний конкурс)
 ☐ Особа з інвалідністю (Загальний конкурс)

Доступні спеціальності:

Додати абітурієнта

Знайти абітурієнта

ID

id

Email

Email

Телефон

+380XXXXXXXX

Знайти

Видалити абітурієнта

ID

id

Видалити

Оновити статус

ID абітурієнта

ID спеціальності

Статус

Прийнято на контракт

Оновити

Таблиця абітурієнтів

Фармація

Загальний конкурс

Застосувати фільтри

Фармація

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус	Біологія	Хімія
----------	----------------	------	----------	---------	-------	--------------	-------------	--------	----------	-------

Рисунок В.11 – роль приймальної комісії, сторінка абітурієнтів із формою додавання, пошуку за ключовими атрибутами, формою видалення всіх заяв абітурієнта, оновлення статусу окремої заявки

Рисунок В.12 – сторінка із спеціальностями із формою пошуку за ключовими атрибутами, оновлення кіль-ті місць, фільтрації за факультетами

Рисунок В.13 – сторінка для отримання звітів всіх заявок за квотою та обраним роком (від року найпершої поданої заявки до поточного року) та звіту всіх спеціальностей по всім факультетам

2025

Загальний конкурс

абітурієнти

Комп'ютерні науки

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус	Українськ мова
21	86	4141	Луценко	+380003242449	lucenkov@gmail.com	0.00	2025-06-13	Невизначено	1.00

Кібербезпека

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус	Українськ мова	Математика	Англійсь мова
-------------	-------------------	------	----------	---------	-------	-----------------	----------------	--------	-------------------	------------	------------------

Правознавство

id заяви	id абітурієнта	Ім'я	Прізвище	Телефон	Email	Середній бал	Дата подачі	Статус	Українськ мова	Математика	Історія України
-------------	-------------------	------	----------	---------	-------	-----------------	----------------	--------	-------------------	------------	--------------------

2025

Загальний конкурс

абітурієнти

спеціальності

Факультет комп'ютерних наук

ID	Назва спеціальності	Код спеціальності	Бюджетні місця квоти 1	Бюджетні місця квоти 2	Бюджетні місця загальний конкурс	Контрактні місця	Всього місць
1	Комп'ютерні науки	121	1	0	-1	3	3
2	Кібербезпека	122	1	0	4	5	10
13	тест	156	0	0	2	2	4
17	35235	11	0	0	1	1	2

Юридичний факультет

ID	Назва спеціальності	Код спеціальності	Бюджетні місця квоти 1	Бюджетні місця квоти 2	Бюджетні місця загальний конкурс	Контрактні місця	Всього місць
3	Правознавство	221	0	0	21	2	23
4	Міжнародне право	222	0	0	10	20	30
15	466	12	0	0	2	3	5

Рисунок В.14, В.15 – звіти по заявкам та спеціальностям, при повторному натисканні на кнопки звіти згортаються та розгортаються

Оновити статус

ID апеляції

Статус

Апеляція прийнята

Оновити

Знайти апеляцію

ID апеляції

Знайти

Видалити апеляцію

ID апеляції

Видалити

Таблиця апеляцій

Комп'ютерні науки

Квота 1

Застосувати фільтри

Комп'ютерні науки

Id Апеляції	Id Абитурієнта	Прізвище	Id Заяви	Подивитись	Статус заяви	Статус апеляції
1	96	Володимир	37	Подивитись	Невизначено	Невизначено

Рисунок В.16 – роль апеляційна комісія, сторінка для перегляду спеціальностей із формою оновлення статусу, пошуку, видалення та фільтрації за квотою та спеціальностями

Апеляційна комісія

Увійти

Апеляції

Оновити статус

ID апеляції

Статус

Апеляція прийнята

Оновити

Знайти апеляцію

ID апеляції

Знайти

Видалити апеляцію

ID апеляції

Видалити

Таблиця апеляцій

Комп'ютерні науки

Квота 1

Застосувати фільтри

Комп'ютерні науки

Id Апеляції	Id Абитурієнта	Прізвище	Id Заяви	Подивитись	Статус заяви	Статус апеляції
1	96	Володимир	37	Подивитись	Невизначено	Невизначено

Апеляція за заявою #37

ID Абитурієнта: 96

ПІБ: Володимир

ID Заяви: 37

Повідомлення апеляції:

Текст апеляції

Рисунок В.17 – додаткове вікно для перегляду інформації щодо поданої апеляції

Рисунок В.18 – роль системний адміністратор із найширшими
можливостями, на сторінці із спеціальностями окрім форм пошуку
та оновлення місць присутні форми додавання та видалення
спеціальностей

Add

Оновити місце

ID спеціальності

Бюджетні місця

Контрактні місця

Update

Знайти спеціальність

ID

Назва

Код

Find

Видалити спеціальність

ID спеціальності

delete

Звіт по спеціальностях за факультетами

Факультет
Юридичний факультет

Юридичний факультет

ID	Назва спеціальності	Код спеціальності	Бюджетні місця квоти 1	Бюджетні місця квоти 2	Бюджетні місця загальний конкурс	Контрактні місця	Всього місць
3	Правознавство	221	0	0	21	2	23
4	Міжнародне право	222	0	0	10	20	30
15	466	12	0	0	2	3	5
16	1	1	0	0	1	1	2

Рисунок В.19 – система не дає можливості видалити спеціальності, на які є заявки від абітурієнтів

Адміністратор
Абітурієнти
Увійти
Спеціальності
Апеляції
Звіти

Оновити статус

ID апеляції

Статус

Апеляція прийнята
Оновити

Знайти апеляцію

ID апеляції

Знайти

Видалити апеляцію

ID апеляції

Видалити

Подати апеляцію

Id заяви

Текст апеляції

Подати апеляцію

Таблиця апеляцій

Комп'ютерні науки
Квота 1
Застосувати фільтри

Id Апеляції	Id Абітурієнта	Прізвище	Id Заяви	Подивитись	Статус заяви	Статус апеляції
-------------	----------------	----------	----------	------------	--------------	-----------------

Рисунок В.20 – сторінка апеляцій для системного адміністратора із формою додання з метою створення тестових апеляцій та роботи з ними

Інші сторінки для системного адміністратора аналогічні сторінкам для ролі приймальна комісія.