

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»

Факультет (інститут)

Навчально-науковий інститут математики та
інформаційних технологій

(повна назва)

Кафедра

Інформаційних технологій та систем

(повна назва)

Освітній ступень

Бакалавр

(код, назва)

Напрямок підготовки

123 Комп'ютерна інженерія

(код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС

М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ” 2024 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Курячому Олександрову Володимировичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Розробка клієнт-серверної системи «Електрон-
ний деканат» засобами Java.

Керівник кваліфікаційної роботи

Переяславська С.О. к.п.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету

від

2. Строк подання студентом проекту (роботи)

3. Вихідні дані до роботи (проекту)

Розроблено клієнт-серверну

програму для навчальних закладів «Електронний деканат» засобами Java в

середовищі IntelliJ IDEA.





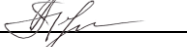
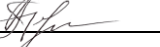
(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Розділ 1. Аналіз предметної області та технологій. Розділ 2.Проектування системи «Електронний деканат». Розділ 3. Реалізація системи «Електронний деканат».

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту/роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Переяславська С.О.		
2	Переяславська С.О.		
3	Переяславська С.О.		

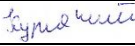
7. Дата видачі завдання „ _____ ” _____ 2024р.


КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	Другий тиждень листопада (10 листопада)	
	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 грудня	
	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 28 січня	
	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень березня	
	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 березня	
	Попередній захист роботи на кафедрі	квітень	
	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

Керівник проекту (роботи)


підпис


підпис

О. В. Курячий

(ініціали, прізвище)

С. О. Переяславська

(ініціали, прізвище)

АНОТАЦІЯ

Курячий О. В.

Тема: Розробка клієнт–серверної системи «Електронний деканат» засобами Java.

Спеціальність: 123 «Комп'ютерна інженерія»

Установа: ЛНУ імені Тараса Шевченка, 2025 р.

Бакалаврська робота містить: 143 с., 26 рис., 59 табл., 31 джерело.

Об'єкт дослідження - процес автоматизованого управління обліком студентів у закладах вищої освіти.

Предмет дослідження - клієнт–серверна інформаційна система для ведення обліку здобувачів вищої освіти, викладачів, дисциплін, оцінок.

Мета роботи - розробка клієнт–серверної системи «Електронний деканат» з використанням мови програмування Java для автоматизації освітнього процесу в ЗВО.

Результати роботи.

В результаті дослідження визначено функціональні та нефункціональні вимоги, обґрунтовано вибір технологій. Розроблено архітектуру системи, інфологічну та даталогічну модель бази даних, програмну реалізацію серверної та клієнтської частин системи. Серверна частина реалізована мовою Java з використанням TCP-з'єднання та обробки запитів. Клієнтська частина має графічний інтерфейс на основі бібліотеки Swing. Для взаємодії з базою даних використано MySQL.

Ключові слова: Java, клієнт–серверна система, електронний деканат, TCP, MySQL, Swing.

ABSTRACT

Kuryachyi Oleksandr

Theme: Development of a client-server system «Electronic Dean's Office» using Java.

Speciality: 123 «Computer Engineering»

Institution: Luhansk Taras Shevchenko National University (LTSNU), 2025.

Diploma work contains: 143 pages, 26 figures, 59 tables, 31 references.

Object of research – the process of automated student data management in higher educational institutions.

Subject of research – a client-server information system for managing records of students, teachers, courses, and grades.

Aim of work – to develop a client-server system «Electronic Dean's Office» using Java for automation of educational administrative processes.

Results.

The paper presents an analysis of existing analog systems, formulates the requirements, and justifies the choice of technologies. A three-tier architecture of the system was developed, along with an infological and datalogical model of the database. The server part was implemented in Java using TCP connections for request handling. The client part features a user-friendly GUI based on the Swing library. MySQL was used as the database management system to ensure reliable data processing and storage.

Keywords. Java, client-server system, electronic dean's office, TCP, MySQL, Swing.

РОЗРОБКА КЛІЄНТ—СЕРВЕРНОЇ СИСТЕМИ «ЕЛЕКТРОННИЙ ДЕКАНАТ» ЗАСОБАМИ JAVA

					ІТС.4КІ.0124.01-ВП				
Змн.	Арк.	№ докум.	Підпис	Дата	ВІДОМІСТЬ ПРОЄКТУ	Літ.	Арк.	Акрушів	
Розроб.		Курячий О.В.							
Керівник		Переяславська С.О.					1	1	
Реценз.		Козуб Ю.Г.				ЛНУ			
Н. Контр.						Кафедра ІТС, Гр.4КІ			
Зав. каф.		Семенов М.А.							

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»
Факультет (інститут) Навчально-науковий інститут
математики та інформаційних технологій
(повна назва)
Кафедра Інформаційних технологій та систем
(повна назва)
(код, назва)

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання програмної розробки (ПР):
РОЗРОБКА КЛІЄНТ—СЕРВЕРНОЇ СИСТЕМИ «ЕЛЕКТРОННИЙ
ДЕКАНАТ» ЗАСОБАМИ JAVA
ІТС.4КІ.0124.02-ТЗ

ПОГОДЖЕНО
Керівник кваліфікаційної роботи

Переяславська С.О.

“ _____ ” _____ 2025р

ВИКОНАВЕЦЬ
Студент групи 4КІ

Курячий О.В.

“ _____ ” _____ 2025р

Зміст

ВСТУП	3
1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ	3
2. ПРИЗНАЧЕННЯ ОБ'ЄКТУ	4
3. ВИМОГИ СИСТЕМИ	5

					ІТС.4КІ.0124.02-ТЗ						
Змн.	Арк.	№ докум.	Підпис	Дата	ТЕХНІЧНЕ ЗАВДАННЯ			Літ.	Арк.	Акрушів	
Розроб.	Курячий О.В.									2	5
Керівник	Переяславська С.О.										
Реценз.	Козуб Ю.Г.										
Н. Контр.											
Зав. каф.	Семенов М.А.										
					ЛНУ						
					Кафедра ІТС, Гр.4КІ						

ВСТУП

- 1.1. **Найменування:** Розробка клієнт—серверної системи «електронний деканат» засобами java.
- 1.2. **Підстава до виконання:** Підставою для виконання даної розробки є завдання на бакалаврську роботу.
- 1.3. **Термін розробки:**
- 1.3.1. Початок 15 жовтня 2024р.
- 1.3.2. Закінчення 10 березня 2025р.
- 1.4. **Фінансується** за рахунок коштів замовника.

1. ХАРАКТЕРИСТИКА ОБ'ЄКТУ

- 1.1. Розроблена система повинна забезпечувати взаємодію користувача з електронною базою даних навчального закладу через графічний клієнтський інтерфейс, що з'єднується з серверною частиною за допомогою клієнт-серверної архітектури.
- 1.2. До складу об'єкта, що створюється, входить:
- Клієнтська частина з графічним інтерфейсом, реалізована мовою програмування Java.
 - Серверна частина, яка приймає запити клієнта, обробляє їх та взаємодіє з базою даних.
 - СУБД MySQL для зберігання інформації про здобувачів освіти, викладачів, дисципліни, бали тощо.
- 1.3. Вхідна інформація включає:
- Дані користувачів (логін, пароль).
 - Інформацію про здобувачів, дисципліни, бали.
 - Запити користувача через GUI-клієнт.

					ІТС.4КІ.0124.02-ТЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4. Вихідна інформація включає:

- Відповіді на запити користувача (перегляд, додавання, видалення даних).
- Повідомлення про результат операцій (успіх, помилка).
- Звіти щодо успішності здобувачів освіти та інші аналітичні дані, які формуються на основі даних в базі.

2. ПРИЗНАЧЕННЯ ОБ'ЄКТУ

2.1. Призначення:

Реалізація клієнт-серверного програмного забезпечення для автоматизації процесів обліку, управління та обміну інформацією в навчальному закладі, зокрема в деканаті. Система дозволяє зручно зберігати, обробляти та переглядати дані про здобувачів, дисципліни, викладачів та результати навчання через графічний інтерфейс.

2.2. Цільова аудиторія:

- Співробітники деканату (адміністратори баз даних).
- Викладачі, які вводять та переглядають оцінки здобувачів.
- Освітняни, які отримують доступ до свого навчального прогресу.
- IT-відділ навчального закладу, що займається підтримкою і розгортанням системи.

2.3. Платформи для роботи системи:

- Клієнтська частина запускається на ОС Windows або Linux з встановленою Java Virtual Machine.
- Серверна частина також працює на ОС Windows або Linux, потребує JVM та доступ до бази даних MySQL.

					<i>ITC.4KI.0124.02-T3</i>	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

- Система не передбачає мобільної версії чи веб-доступу, оскільки реалізована як десктопний додаток із TCP-з'єднанням між клієнтом і сервером.

3. ВИМОГИ СИСТЕМИ

3.1. Загальні вимоги:

- 3.1.1. Система повинна працювати на персональних комп'ютерах з операційною системою Windows або Linux, що підтримують запуск Java-додатків.
- 3.1.2. Клієнтська частина повинна мати інтуїтивно зрозумілий графічний інтерфейс для зручної роботи користувачів різних рівнів підготовки.
- 3.1.3. Система повинна бути оптимізована для стабільної роботи при обробці великої кількості даних та одночасному підключенні кількох клієнтів.
- 3.1.4. Програмне забезпечення повинно бути сумісним з СУБД MySQL та мати можливість масштабування для розширення функціональності в майбутньому.
- 3.1.5. Повинно бути реалізовано захист даних користувачів (автентифікація) та механізми безпечної передачі інформації між клієнтом і сервером.

3.2. Функціональні вимоги

Облік студентів:

- 3.2.1. Додавання та видалення студентів.
- 3.2.2. Призначення студентів у навчальні групи.
- 3.2.3. Збереження академічної історії студента.

Облік викладачів:

- 3.2.4. Додавання та видалення викладачів.
- 3.2.5. Призначення викладачів на конкретні дисципліни.

					<i>ITS.4KI.0124.02-T3</i>	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністрування навчальних дисциплін:

3.2.6. Додавання та видалення дисциплін.

3.2.7. Призначення конкретних дисциплін для студентів та викладачів.

Облік оцінок:

3.2.8. Внесення оцінок студентів у систему.

3.2.9. Збереження історії оцінювання.

3.2.10. Можливість перегляду оцінок студентами та викладачами.

Реєстрація та автентифікація:

3.2.11. Реєстрація користувачів.

3.2.12. Авторизація користувачів.

3.2.13. Захист від несанкціонованого доступу.

3.2.14. Шифрування паролів.

Клієнт-серверна взаємодія:

3.2.15. Використання сервера бази даних.

3.2.16. Використання окремого серверного додатка для обробки запитів.

3.2.17. Клієнтський додаток для взаємодії користувачів із серверною частиною.

3.3. Нефункціональні вимоги

Безпека:

3.2.6. Шифрування даних користувачів.

Зручність використання:

3.2.7. Інтуїтивно зрозумілий графічний інтерфейс для роботи користувачів.

3.2.8. Дружній дизайн з урахуванням принципів UX/UI.

3.3. Вимоги до складу та архітектури:

3.3.1. Система повинна мати чітко визначену клієнт-серверну архітектуру з розподіленням обов'язків: клієнт відповідає за інтерфейс користувача, сервер – за обробку логіки та доступ до бази даних.

3.3.2. Клієнт і сервер мають взаємодіяти через TCP-з'єднання із власним форматом обміну.

					ITC.4KI.0124.02-T3	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3.3. У складі системи обов'язково повинні бути:

- клієнтський застосунок з графічним інтерфейсом;
- серверний застосунок із GUI для адміністрування підключень;
- база даних MySQL для зберігання навчальної інформації.

					ІТС.4КІ.0124.02-ТЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики та інформаційних
технологій

(назва факультету, інституту)

Інформаційних технологій та систем

(назва кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
БАКАЛАВРА
(освітньо-кваліфікаційний рівень)

на тему:
РОЗРОБКА КЛІЄНТ—СЕРВЕРНОЇ СИСТЕМИ «ЕЛЕКТРОННИЙ
ДЕКАНАТ» ЗАСОБАМИ JAVA

Виконав: студент 4 курсу, групи _____
напряму підготовки (спеціальності)
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)

Курячий О.В.
(прізвище та ініціали)

Керівник Переяславська С.О.
(прізвище та ініціали)

Рецензент Козуб Ю. Г.
(прізвище та ініціали)

Полтава – 2025 року

Зміст

ВСТУП.....	3
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ	5
1.1. Аналіз процесу управління обліку студентів	5
1.2. Вимоги до електронного деканату	8
1.3. Вибір технологій для розробки	9
1.4. Огляд клієнт-серверної архітектури.....	11
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ	17
2.1. Архітектура клієнт-серверної системи.....	17
2.2. Модель бази даних та її структура	19
2.3. Проектування серверної частини.....	23
2.4. Проектування клієнтської частини	24
2.5. Опис взаємодії між клієнтом і сервером	25
РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ.....	28
3.1. Опис серверної частини.....	28
3.2. Опис клієнтської частини	58
ВИСНОВКИ	90
СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ	92
ДОДАТКИ.....	95

					ІТС.4КІ.0124.03-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	ЗМІСТ	Лім.	Арк.	Акрушів
Розроб.		Курячий О.В.						
Керівник		Переяславська С.О.					2	1
Реценз.		Козуб Ю.Г.				ЛНУ Кафедра ІТС, Гр.4КІ		
Н. Контр.								
Зав. каф.		Семенов М.А.						

ВСТУП

В сучасних умовах цифровізації суспільства та активного розвитку інформаційних технологій особливу увагу приділяють автоматизації управління та обробки інформації в різних сферах діяльності. Однією з таких сфер є освітні заклади, де управління студентськими даними, ведення академічної документації та комунікація між викладачами й студентами потребують ефективного програмного забезпечення.

Традиційні методи обліку та адміністрування освітніх процесів часто передбачають використання паперових або застарілих електронних систем, які не відповідають сучасним вимогам ефективності, безпеки та зручності. Внаслідок цього виникає потреба у впровадженні інноваційних рішень для автоматизації цих процесів. Одним із таких рішень є електронний деканат — система, що дозволяє керувати студентськими даними, розкладом занять, успішністю та іншими адміністративними аспектами навчального процесу.

Об'єктом дослідження є процес управління студентськими даними в освітніх установах.

Предметом дослідження є клієнт-серверна система «Електронний деканат», розроблена з використанням мови програмування Java, технологій MySQL та TCP-з'єднання між клієнтом і сервером.

Метою роботи є розробка клієнт-серверної системи «Електронний деканат», яка забезпечує зручний і безпечний доступ до даних, їх обробку та управління освітньою документацією. Для досягнення цієї мети передбачено вирішення таких основних завдань:

- а) аналіз предметної області та існуючих аналогів систем електронного деканату;

					ІТС.4КІ.0124.03-ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	ВСТУП	Лім.	Арк.	Акрушіє	
Розроб.		Курячий О.В.							
Керівник		Переяславська С.О.					3	2	
Реценз.		Козуб Ю.Г.				ЛНУ			
Н. Контр.						Кафедра ІТС, Гр.4КІ			
Зав. каф.		Семенов М.А.							

- b) визначення вимог до програмного забезпечення та вибір оптимальних технологій для розробки;
- c) проєктування архітектури клієнт-серверної системи та моделі бази даних;
- d) реалізація серверної частини системи з урахуванням вимог до безпеки та продуктивності;
- e) розробка клієнтської частини з графічним інтерфейсом для взаємодії з користувачем;
- f) тестування та оптимізація роботи системи для забезпечення її стабільної роботи.

Запропоноване програмне рішення дозволить суттєво спростити адміністрування навчального процесу, підвищити швидкість обробки інформації та мінімізувати ймовірність помилок, пов'язаних із людським фактором. Використання клієнт-серверної архітектури забезпечить ефективний розподіл навантаження та зручність доступу до даних для користувачів системи. Таким чином, розробка даної системи є актуальним завданням у сфері автоматизації управління освітніми установами.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ

1.1. Аналіз процесу управління обліку студентів

В сучасних навчальних закладах управління обліком студентів є важливою частиною організації навчального процесу. Системи управління обліком студентів повинні забезпечити ефективне ведення та обробку даних, що стосуються академічної діяльності студентів, а також допомогти адміністрації університету, деканатам та викладачам у здійсненні поточного контролю успішності та академічних досягнень студентів [22].

Процес управління обліком студентів

Процес управління обліком студентів у загальному випадку можна поділити на кілька основних етапів:

- а) **Реєстрація студентів у системі:** Перший етап включає введення базових даних про студента в систему, що може включати особисті дані (Прізвище та ім'я, дата народження, контактні дані), академічні досягнення, інформацію про факультет та спеціальність. Цей етап є критично важливим, оскільки правильність і повнота внесених даних безпосередньо впливає на подальшу роботу з системою.
- б) **Реєстрація на курси та дисципліни:** На цьому етапі система дозволяє студенту обирати курси та дисципліни, на які він буде записаний протягом семестру. Процес автоматизує контроль за кількістю місць на курсах, дозволяє студентам бачити доступні курси та реєструватися на них, а також надає адміністрації можливість контролювати процес вибору дисциплін, реєстрацію на спеціалізації тощо.

					ІТС.4КІ.0124.03-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата	РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ			Лім.	Арк.	Акрушів		
Розроб.	Курячий О.В.									5	12	
Керівник	Переяславська С.О.							ЛНУ Кафедра ІТС, Гр.4КІ				
Реценз.	Козуб Ю.Г.											
Н. Контр.												
Зав. каф.	Семенов М.А.											

- с) **Моніторинг успішності:** Після реєстрації на курси система слідкує за результатами студентів протягом семестру. Вона зберігає дані про оцінки за різноманітні завдання, тестування, контрольні роботи, заліки та екзамени. Це дає змогу викладачам та деканатам оперативно отримувати актуальну інформацію про успішність кожного студента, а також сприяє зручності для студентів, які мають можливість постійно перевіряти свої досягнення.
- д) **Ведення академічної історії:** Протягом всього періоду навчання система автоматично формує академічну історію студента, що включає інформацію про курси, оцінки, академічні досягнення, а також історію переведення на інші факультети чи спеціальності, відпустки, відрахування або поновлення. Це дає змогу створювати повну картину успішності та кар'єри студента в межах навчального закладу.
- е) **Управління випуском студентів:** Наприкінці навчання студенти отримують дипломи та сертифікати, що підтверджують їхні досягнення. Система управління обліком студентів повинна автоматизувати процес видачі документів, а також надавати необхідну інформацію для формування державних атестатів, дипломів про вищу освіту, сертифікатів про курси тощо.

Проблеми та обмеження існуючих процесів

Хоча процеси управління обліком студентів є стандартними в більшості навчальних закладів, існують низка проблем, що супроводжують ці процеси:

- а) **Відсутність інтеграції між системами:** Багато університетів використовують окремі системи для різних етапів обліку студентів (реєстрація, оцінки, академічні досягнення). Це призводить до проблем з інтеграцією та обміном даними, що затрудняє адміністрування та роботу з інформацією.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

- b) **Неефективність ручного вводу даних:** На багатьох етапах процесів обліку студентів все ще існують мануальні операції введення даних. Це може спричиняти помилки при обробці інформації та уповільнювати роботу з даними.
- c) **Недостатня автоматизація аналізу успішності:** У багатьох системах управління обліком студентів не вистачає інструментів для глибокого аналізу успішності студентів, наприклад, для прогнозування потенційних проблем з успішністю або для генерації детальних звітів для викладачів та адміністрації.
- d) **Захист даних:** Оскільки системи містять конфіденційну інформацію про студентів, забезпечення належного рівня безпеки та захисту даних є важливою проблемою. Багато існуючих систем мають вразливості щодо захисту особистих даних студентів, що ставить під загрозу їх конфіденційність [2].

Напрями вдосконалення процесів

- a) **Інтеграція систем:** Для покращення обробки даних та уникнення дублювання важливо інтегрувати різні системи, що використовуються в навчальному процесі, такі як системи реєстрації студентів, оцінювання, бібліотечні системи та інші, в єдину централізовану платформу [26].
- b) **Автоматизація збору та обробки даних:** Замість ручного введення даних, можна використовувати автоматизовані системи збору інформації, що забезпечують точність та швидкість обробки даних, а також знижують ймовірність помилок.
- c) **Аналіз успішності з використанням ІТ-рішень:** Використання штучного інтелекту та алгоритмів машинного навчання для прогнозування успішності студентів може значно підвищити ефективність системи, а також допомогти виявляти потенційні проблеми ще до того, як вони стануть критичними.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- d) **Посилення захисту даних:** Використання сучасних технологій шифрування та захисту даних, а також постійне оновлення систем безпеки допоможуть зберегти конфіденційність даних студентів. [2]

1.2. Вимоги до електронного деканату

Розробка клієнт-серверної системи «Електронний деканат» передбачає створення інформаційної системи для автоматизації обліку студентів, викладачів, курсів та оцінок. Дана, система має відповідати наступним функціональним та нефункціональним вимогам. [18, 16, 30]

Функціональні вимоги

Облік студентів:

- a) Додавання та видалення студентів.
- b) Призначення студентів у навчальні групи.
- c) Збереження академічної історії студента.

Облік викладачів:

- a) Додавання та видалення викладачів.
- b) Призначення викладачів на конкретні дисципліни.

Адміністрування навчальних дисциплін:

- a) Додавання та видалення курсів.
- b) Призначення конкретних дисциплін для студентів та викладачів.

Облік оцінок:

- a) Внесення оцінок студентів у систему.
- b) Збереження історії оцінювання.
- c) Можливість перегляду оцінок студентами та викладачами.

Реєстрація та автентифікація:

- a) Реєстрація користувачів.
- b) Авторизація користувачів.
- c) Захист від несанкціонованого доступу.
- d) Шифрування паролів.

					ITS.4KI.0124.03-ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнт-серверна взаємодія:

- а) Використання сервера бази даних.
- б) Використання окремого серверного додатка для обробки запитів.
- с) Клієнтський додаток для взаємодії користувачів із серверною частиною.

Нефункціональні вимоги

Безпека:

- а) Шифрування даних користувачів. [2]

Зручність використання:

- а) Інтуїтивно зрозумілий графічний інтерфейс для роботи користувачів.
- б) Дружній дизайн з урахуванням принципів UX/UI.

1.3. Вибір технологій для розробки

В процесі розробки клієнт-серверної системи «Електронний деканат» важливо обрати ефективні технології, які забезпечать стабільність, продуктивність та зручність у використанні. У даному підрозділі розглянуто вибір основних технологій: Java для програмної логіки, MySQL для управління базами даних, Maven для управління залежностями та збірки проєкту, а також засоби створення графічного інтерфейсу користувача (GUI).

Огляд технологій

Java

Java — це об'єктно-орієнтована мова програмування, що широко використовується для розробки корпоративних застосунків, зокрема клієнт-серверних систем. Вона забезпечує платформонезалежність завдяки використанню віртуальної машини Java (JVM), дозволяє запускати застосунок на різних операційних системах без змін у коді [5, 28].

MySQL

					ІТС.4КІ.0124.03-ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

MySQL — це реляційна система управління базами даних (СУБД), яка відзначається високою продуктивністю, надійністю та масштабованістю. Вона підтримує мову SQL для виконання запитів і добре інтегрується з Java через драйвер JDBC [6, 7].

Maven

Maven — це система управління збіркою та залежностями, яка спрощує процес компіляції, тестування та розгортання Java-додатків. Вона автоматизує завантаження необхідних бібліотек та структурування проєкту [21].

Графічний інтерфейс користувача

Для створення зручного інтерфейсу користувача використовуються бібліотеки JavaFX або Swing. В подальшому буде використовуватися Swing [15].

Переваги вибору технологій

Java

- a) Портативність: програма може працювати на будь-якій ОС з підтримкою JVM.
- b) Висока продуктивність завдяки JIT-компіляції.
- c) Велика спільнота та підтримка.
- d) Інтеграція з різними технологіями для створення масштабованих додатків.

MySQL

- a) Оптимізована для роботи з великими обсягами даних.
- b) Підтримка складних SQL-запитів.
- c) Висока швидкість виконання операцій читання та запису.
- d) Легка інтеграція з Java через JDBC.

Maven

- a) Автоматизація процесу збірки.
- b) Управління залежностями без потреби в ручному завантаженні бібліотек.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

- с) Стандартна структура проєкту, що полегшує роботу команди розробників.

1.4. Огляд клієнт-серверної архітектури

Клієнт-серверна архітектура є одним із найпоширеніших підходів до побудови інформаційних систем. Вона базується на розподілі функціональності між двома основними компонентами: клієнтом, який запитує ресурси або послуги, і сервером, який обробляє ці запити та повертає результати. Така модель дозволяє створювати масштабовані, гнучкі та ефективні системи, які відповідають сучасним вимогам до продуктивності та безпеки [20].

Визначення клієнт-серверної архітектури

Клієнт-серверна архітектура передбачає, що клієнти взаємодіють із сервером через мережу. Сервер виконує обчислення, зберігає дані та надає необхідні ресурси клієнтам. Основні принципи цієї моделі включають:

- а) **Розподіл навантаження:** клієнти виконують лише частину обчислень, а основна робота зосереджена на сервері.
- б) **Централізоване управління даними:** всі дані зберігаються на сервері, що забезпечує їхню цілісність і безпеку.
- с) **Мережева взаємодія:** клієнти та сервери можуть бути розміщені на різних пристроях та з'єднуватися через локальну або глобальну мережу.

Типи клієнт-серверної архітектури

Існує кілька типів клієнт-серверної архітектури залежно від рівня розподілу функцій між клієнтом і сервером:

Однорівнева архітектура (Thick Client)

В такій архітектурі клієнт виконує більшість операцій, включаючи обробку даних, а сервер використовується переважно для збереження інформації. Цей підхід менш залежний від мережі, але важко масштабується.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11



Рис. 1.1. Однорівнева архітектура клієнт-сервер

Дворівнева архітектура (Two-Tier)

Цей підхід передбачає чіткий розподіл між клієнтом та сервером. Клієнтська частина взаємодіє із сервером бази даних через SQL-запити. Вона часто використовується у корпоративних системах завдяки простоті реалізації.

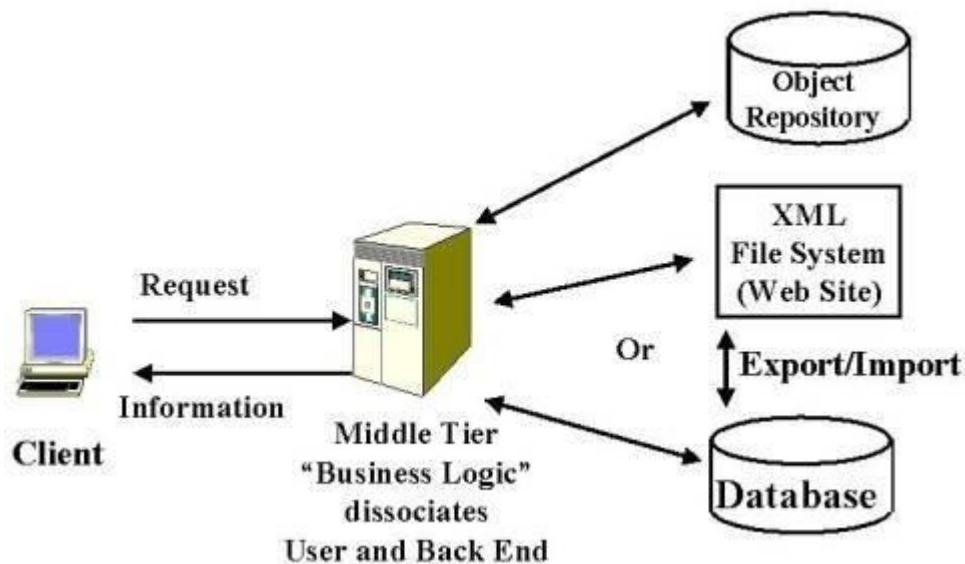


Рис. 1.2. Дворівнева архітектура клієнт-сервер

Трирівнева архітектура (Three-Tier)

Тут додається ще один рівень — програмний сервер (Application Server), який виконує бізнес-логіку та обробку запитів. Це підвищує безпеку, гнучкість і продуктивність системи [14, 17].

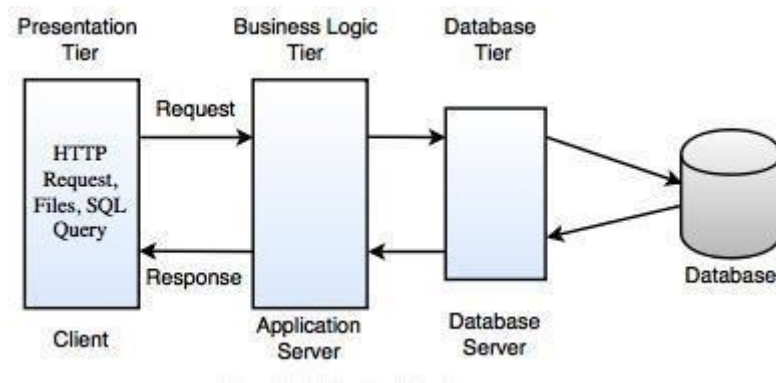


Рис. 1.3. Трирівнева архітектура клієнт-сервер

Багаторівнева архітектура (Multi-Tier)

Ця модель включає кілька серверів, кожен з яких відповідає за конкретний функціонал (база даних, бізнес-логіка, API, веб-інтерфейс). Вона використовується у великих розподілених системах, що працюють із високими навантаженнями.

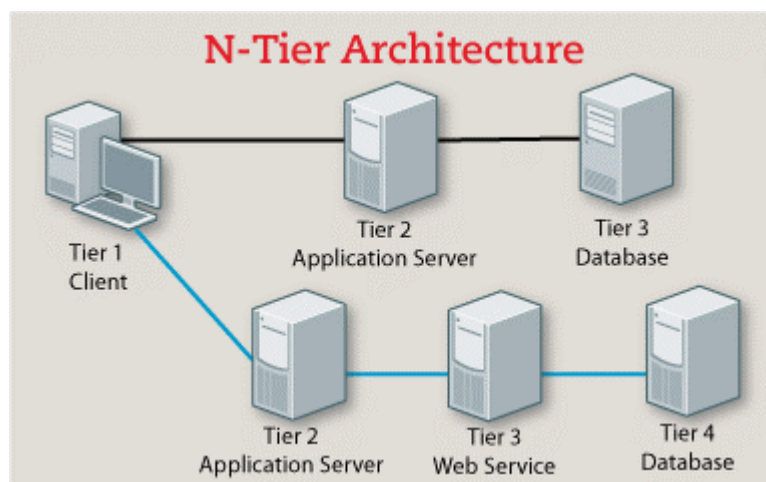


Рис. 1.4. Багаторівнева архітектура клієнт-сервер

Опис клієнт-серверної моделі в «Електронному деканаті»

Система «Електронний деканат» використовує трирівневу клієнт-серверну архітектуру, у якій клієнтська частина взаємодіє із програмним сервером, який у свою чергу обробляє запити та передає їх серверу бази даних MySQL.

Основні компоненти системи:

- а) **Клієнт** (графічний інтерфейс, реалізований за допомогою Swing) дозволяє студентам та викладачам виконувати запити щодо інформа-

ції про курси, розклад та оцінки.

- b) **Програмний сервер** (реалізований на Java) є проміжним рівнем між клієнтом та сервером бази даних, виконує бізнес-логіку та обробляє запити.
- c) **Сервер бази даних (MySQL)** зберігає інформацію про студентів, дисципліни, оцінки, викладачів та інші адміністративні дані.
- d) **Обробка запитів** відбувається через програмний сервер, який взаємодіє із сервером бази даних за допомогою JDBC з'єднання.

Переваги та недоліки клієнт-серверної архітектури

Переваги:

- a) **Централізоване управління даними** — всі дані зберігаються на сервері, що спрощує їхнє адміністрування та захист.
- b) **Розширюваність** — можна додавати нові функції без значного перероблення клієнта чи сервера.
- c) **Безпека** — централізоване зберігання даних дозволяє контролювати доступ до інформації та реалізовувати резервне копіювання.
- d) **Ефективне використання ресурсів** — клієнти виконують лише необхідний мінімум завдань, основні обчислення здійснюються сервером.

Недоліки:

- a) **Високе навантаження на сервер** — при великій кількості клієнтів сервер може стати «вузьким місцем» системи.
- b) **Залежність від мережі** — у разі проблем із підключенням система може працювати нестабільно.
- c) **Складність розгортання** — налаштування сервера та клієнтів потребує часу та технічних знань.

Клієнт-серверна архітектура є оптимальним рішенням для реалізації системи «Електронний деканат», оскільки вона забезпечує ефективний розподіл ресурсів, централізоване управління даними та гнучкість у масшта-

					ІТС.4КІ.0124.03-ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

абуванні. Використання трирівневої моделі дозволяє забезпечити зручну взаємодію між клієнтською програмою, програмним сервером та сервером бази даних, що сприяє підвищенню продуктивності та безпеки системи.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 1

В результаті аналізу предметної області та технологій для розробки клієнт-серверної системи «Електронний деканат» було зроблено низку важливих висновків.

По-перше, системи управління обліком здобувачів освіти є невід’ємною частиною сучасного навчального процесу. Ефективність таких систем значною мірою залежить від автоматизації процесів, інтеграції між різними підсистемами та зручності взаємодії для кінцевих користувачів — здобувачів, викладачів та адміністрації закладу.

По-друге, розробка інформаційної системи вимагає чіткого визначення функціональних та нефункціональних вимог, які охоплюють облік здобувачів, викладачів, дисциплін, балів, а також безпечну автентифікацію користувачів і швидку взаємодію з базою даних.

По-третє, обґрунтовано вибір технологій для реалізації системи, зокрема використання Java для логіки додатку, MySQL для зберігання даних та Swing для створення інтерфейсу. Застосування клієнт-серверної архітектури дозволяє забезпечити централізоване управління інформацією, гнучке масштабування системи та високу продуктивність.

Таким чином, врахування вищезазначених аспектів є запорукою створення ефективної та надійної системи «Електронний деканат», яка відповідатиме сучасним вимогам інформаційного забезпечення навчальних закладів.

					<i>ІТС.4КІ.0124.03-ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Архітектура клієнт-серверної системи

Загальний опис архітектури

Розроблювана система «Електронний деканат» використовує клієнт-серверну архітектуру, яка забезпечує розподіл обчислювальних навантажень між клієнтською та серверною частинами. Це дозволяє централізовано зберігати дані, забезпечувати безпеку та підтримувати масштабованість системи [3, 29].

Обґрунтування вибору клієнт-серверної моделі

Вибір клієнт-серверної архітектури зумовлений потребою в централізованому управлінні даними та безпеці доступу до них. Основні переваги цієї архітектури:

- а) **Централізоване управління даними** — усі важливі академічні дані зберігаються на сервері бази даних, що запобігає втраті чи дублюванню інформації.
- б) **Розподіл навантаження** — сервер обробляє запити від клієнтів, мінімізуючи вимоги до ресурсів клієнтських пристроїв.
- с) **Безпека** — клієнти взаємодіють із сервером через авторизацію та автентифікацію, що запобігає несанкціонованому доступу.

Основні компоненти системи

Система складається з трьох основних компонентів:

- а) **Клієнтська частина** — графічний інтерфейс користувача, через який здобувачі вищої освіти та співробітники навчального закладу взаємодіють із системою. Вона реалізована мовою Java з використанням технологій GUI (Swing).

					ІТС.4КІ.0124.03-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата	РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ			Лім.	Арк.	Акрушіє		
Розроб.		Курячий О.В.								17	11	
Керівник		Переяславська С.О.										
Реценз.		Козуб Ю.Г.										
Н. Контр.												
Зав. каф.		Семенов М.А.										
					ЛНУ Кафедра ІТС, Гр.4КІ							

- b) **Програмний сервер** — проміжний рівень, який обробляє запити клієнтів, взаємодіє з базою даних та здійснює бізнес-логіку системи. Реалізований за допомогою Java (Swing для графічного інтерфейсу).
- c) **Сервер бази даних** — зберігає інформацію про здобувачів освіти, курси, викладачів та інші сутності системи. Використовується MySQL для забезпечення швидкого доступу до даних і їхньої цілісності.

ServerSocket — це спеціальний клас, об'єкти якого виконують роль сервера. Тобто можуть обслуговувати запити, які прийшли на певний сокет [1].

Клас Socket — це фактично Socket-клієнт. За допомогою нього ми можемо надіслати повідомлення певному сокету та отримати відповідь [1].

Взаємодія між компонентами

- a) Клієнт під'єднується та надсилає запит до програмного сервера через Socket.
- b) Сервер обробляє запит, виконує необхідні операції (наприклад, перевіряє облікові дані користувача) та звертається до бази даних.
- c) Сервер бази даних виконує відповідний SQL-запит та повертає результат програмному серверу.
- d) Програмний сервер обробляє отримані дані та надсилає відповідь клієнтові у відповідному форматі (JSON).
- e) Клієнтська частина відображає отриману інформацію користувачеві.

Використовувані технології

- a) **Клієнт-серверна взаємодія** — ServerSocket, Socket.
- b) **База даних** — MySQL.
- c) **Серверна частина** — Java, Swing.
- d) **Клієнтська частина** — Swing для графічного інтерфейсу.

Клієнт-серверна архітектура забезпечує ефективний розподіл навантаження, централізоване збереження даних і підвищений рівень безпеки. Це дозволяє створити масштабовану, безпечну та продуктивну си-

стему «Електронний деканат», що відповідатиме вимогам навчального закладу.

2.2. Модель бази даних та її структура

Загальний опис моделі бази даних

Для реалізації системи «Електронний деканат» використовується реляційна база даних, створена на основі MySQL. Вона містить структуровану інформацію про студентів, викладачів, дисципліни, успішність та інші ключові аспекти навчального процесу. Використання MySQL дозволяє забезпечити ефективне збереження, швидкий доступ і підтримку цілісності даних. В даній роботі використовується дві бази даних це ElectronicDean та mysite.

БД ElectronicDean — це база про зберігання даних «Електронного деканату». База даних mysite — це база яка відповідає за автентифікацію та авторизацію даних в системі.

Для повного розуміння структури та взаємозв'язків у базі даних використовуються інфологічна та даталогічна моделі. Інфологічна модель відображає концептуальні сутності та їхні зв'язки у вигляді ER-діаграми, незалежно від конкретної СУБД. Даталогічна модель, у свою чергу, деталізує структуру таблиць, зв'язки між ними та визначає основні атрибути, враховуючи особливості реляційного підходу до збереження інформації. Інфологічна (концептуальна) модель БД ElectronicDean та даталогічна (фізична) модель бази даних ElectronicDean (див. ДОДАТОК В рис. В.1., рис.В.2.) [6, 12, 25,7].

Оскільки база даних mysite містить одну таблицю, то інфологічна та даталогічна модель цієї БД однакова (див. ДОДАТОК В рис.В.3., рис.В.4.).

Основні таблиці баз даних

Таблиця 2.1.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

Таблиця users даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
id	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор користувача
name	VARCHAR	50	NOT NULL	Ім'я користувача
surname	VARCHAR	50	NOT NULL	Прізвище користувача
email	VARCHAR	100	NOT NULL UNIQUE	Електронна пошта користувача
password	VARCHAR	255	NOT NULL	Пароль
phone	VARCHAR	15	—	Номер мобільного телефону користувача

Таблиця 2.2.

Таблиця Students даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
StudentID	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор здобувача освіти
FirstName	VARCHAR	50	NOT NULL	Ім'я здобувача освіти
LastName	VARCHAR	50	NOT NULL	Прізвище здобувача освіти
Email	VARCHAR	100	NOT NULL UNIQUE	Електронна пошта здобувача освіти

Продовження таблиці 2.2.

Phone	VARCHAR	15	—	Номер мобільного телефону здобувача освіти
BirthDate	DATE	—	—	Дата народження здобувача освіти
EnrollmentYear	YEAR	—	NOT NULL	Рік народження здобувача освіти

Таблиця 2.3.

Таблиця Teachers даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
TeacherID	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор викладача
FirstName	VARCHAR	50	NOT NULL	Ім'я викладача
LastName	VARCHAR	50	NOT NULL	Прізвище викладача
Email	VARCHAR	100	NOT NULL UNIQUE	Електронна пошта викладача
Phone	VARCHAR	15	—	Номер мобільного телефону викладача
Department	VARCHAR	100	NOT NULL	Підрозділ в якому працює викладач

Таблиця 2.4.

Таблиця Courses даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
CourseID	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор дисципліни
CourseName	VARCHAR	100	NOT NULL	Назва дисципліни
Description	TEXT	—	—	Опис дисципліни
Credits	INTEGER	—	NOT NULL	Кількість кредитів
TeacherID	INTEGER	—	NOT NULL	Ідентифікатор викладача

Таблиця 2.5.

Таблиця Groupe даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
GroupID	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор групи
GroupName	VARCHAR	50	NOT NULL UNIQUE	Назва групи
Specialty	VARCHAR	100	NOT NULL	Спеціальність
EnrollmentYear	YEAR	—	NOT NULL	Рік вступу

Таблиця 2.6.

Таблиця StudentGroups даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
StudentID	INTEGER	—	NOT NULL	Ідентифікатор здобувача освіти
GroupID	INTEGER	—	NOT NULL	Ідентифікатор групи

Таблиця 2.7.

Таблиця Grades даталогічної моделі бази даних

Найменування	Тип	Розмір	Обмеження	Коментарі
GradeID	INTEGER	—	AUTO_INCREMENT PRIMARY KEY	Ідентифікатор успішності

Продовження таблиці 2.7.

StudentID	INTEGER	—	NOT NULL	Ідентифікатор здобувача освіти
CourseID	INTEGER	—	NOT NULL	Ідентифікатор дисципліни
Grade	TINY INTEGER	Від 0 до 100	NOT NULL CHECK	Підсумок оцінки від 0 до 100 балів

2.3. Проектування серверної частини

Загальний опис серверної частини

Серверна частина клієнт-серверної системи «Електронний деканат» виконує роль посередника між клієнтським додатком і сервером бази даних. Взаємодія між клієнтом і сервером реалізована за допомогою TCP-з'єднання, що забезпечує стабільний та швидкий обмін даними. Основна функція сервера — приймати запити від клієнтів, обробляти їх відповідно до бізнес-логіки та взаємодіяти з сервером бази даних для отримання або збереження інформації.

Завдяки такому підходу клієнтська частина мінімально навантажується, оскільки всі обчислювальні процеси, перевірка даних та взаємодія з базою даних відбуваються на сервері. Це підвищує продуктивність системи та забезпечує централізований контроль за доступом і безпекою інформації.

Архітектура серверної частини

Серверна частина складається з кількох основних модулів, кожен з яких реалізований у вигляді окремих Java-класів. Серед основних компонентів можна виділити такі:

- а) **ServerWithGUI** — головний клас, який запускає сервер та забезпечує графічний інтерфейс адміністратора. Він керує прийомом з'єднань від

клієнтів, створенням сесій, а також забезпечує візуалізацію поточних процесів.

- b) **ClientHandler** — окремий потік для кожного клієнта. Цей клас відповідає за прийом та обробку запитів клієнта, їх розшифровку, виконання необхідної дії та повернення відповіді.
 - c) **db_dekanat, db_log** — класи доступу до бази даних, які реалізують з'єднання, підготовку та виконання SQL-запитів. Вони інкапсулюють логіку роботи з таблицями БД та забезпечують надійний обмін інформацією між сервером і СУБД.
 - d) **TeachersManager, StudentGroupsManager, GradesManager, CoursesManager, StGrManager** — управління відповідними сутностями (курси, оцінки, викладачі, студентські групи). Кожен менеджер містить методи для додавання, оновлення, видалення та отримання записів з БД.
 - e) **Табличні класи:** **CoursesTable, GradesTable, groupTable, StudentGroupsTable, TeachersTable, StudentTable** — відповідають за відображення та структурування даних.
 - f) **Допоміжні класи:** **StudentAdder, GroupAdder, TeachersAdder, Courses, CoursesDeleter, GradesDeleter, GroupDeleter, Groups, StudentDeleter, Students, Teacher, TeachersDeleter, StGrDeleter** — реалізують конкретні дії над даними, що викликаються через запити клієнта.
- Діаграма серверної частини (див. ДОДАТОК А рис.А.1.).

2.4. Проектування клієнтської частини

Загальний опис клієнтської частини

Клієнтська частина системи «Електронний деканат» реалізована у вигляді десктопного застосунку з графічним інтерфейсом, що взаємодіє із сервером через TCP-з'єднання. Вона написана мовою Java з використанням відповідних бібліотек для створення GUI. Клієнтський додаток забезпечує

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

автентифікацію користувачів, формування та надсилення запитів на сервер, а також обробку отриманих відповідей. Інтерфейс користувача розроблений таким чином, щоб забезпечити зручний доступ до функціоналу системи, включаючи перегляд та управління навчальним процесом [15, 31].

Основні функції клієнтського додатку включають:

- а) автентифікацію користувача (реалізовано у класі RegistrationForm);
- б) формування та передавання запитів на сервер;
- в) обробку відповідей від серверної частини;
- г) представлення навчальної інформації у структурованому вигляді.

Архітектура клієнтської частини

Клієнтський застосунок структурований за модульним принципом. Кожен функціональний компонент представлений окремим класом: Діаграма клієнтської частини (див. ДОДАТОК А рис.А.2.).

2.5. Опис взаємодії між клієнтом і сервером

Комунікація між клієнтською та серверною частинами системи «Електронний деканат» реалізована за допомогою протоколу TCP. В основі взаємодії лежить архітектура клієнт-сервер із чітким розподілом обов’язків. Клієнт формує запити, а сервер їх обробляє та повертає відповідь у відповідному форматі.

Клієнт ініціює з’єднання з сервером за допомогою класу ServerConnection, який забезпечує:

- ✓ встановлення TCP-з’єднання;
- ✓ надсилення запитів у вигляді текстових команд;
- ✓ отримання та передавання відповідей для відображення у графічному інтерфейсі.

На сервері взаємодію обслуговує головний клас ServerWithGUI, який містить графічний інтерфейс та логіку обробки клієнтських запитів.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Безпосередню обробку запитів виконує клас ClientHandler, який приймає дані з сокета, аналізує тип запиту та викликає відповідні сервіси для роботи з базою даних [31].

Приклад типової взаємодії:

- a) Клієнт надсилає команду, наприклад: GET_COURSES.
- b) Програмний сервер приймає запит через клас ClientHandler, розпізнає команду.
- c) На основі команди сервер викликає відповідний метод, наприклад, метод з класу CoursesManager.
- d) CoursesManager передає запит до сервера бази даних, де дані витягуються безпосередньо з таблиці курсів (через об'єкти CoursesTable або db_dekanat, які виконують SQL-запити).
- e) Отримані з бази дані повертаються до програмного сервера у вигляді рядка або структури.
- f) Програмний сервер формує відповідь і надсилає її назад клієнту.
- g) Клієнтська частина обробляє відповідь і відображає інформацію у відповідному вікні інтерфейсу (наприклад, CoursesForm).

Таким чином, реалізована трикомпонентна архітектура: клієнт – програмний сервер – сервер бази даних, що забезпечує чітке розділення функцій, масштабованість і зручність підтримки системи.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

Висновок до розділу 2

У другому розділі було здійснено детальне проектування клієнт-серверної системи «Електронний деканат», що дозволило визначити її архітектурні особливості, структуру бази даних, а також механізми взаємодії між основними компонентами системи.

По-перше, обґрунтовано доцільність застосування клієнт-серверної архітектури, яка забезпечує централізоване управління даними, підвищену безпеку, ефективний розподіл навантаження між клієнтською та серверною частинами та гнучкість у масштабуванні.

По-друге, розроблено структуровану модель бази даних, яка охоплює всі ключові елементи навчального процесу: здобувачів освіти, викладачів, дисципліни, оцінки, групи. Використання реляційної СУБД MySQL забезпечує збереження цілісності та швидкий доступ до інформації, а також підтримку складних міжтабличних зв'язків. Спроектовані інфологічні, даталогічні моделі баз даних, а також діаграми класів серверної та клієнтської частин.

По-третє, детально спроектовано серверну та клієнтську частини системи, де кожна реалізована у вигляді модулів із чітким функціональним призначенням. Сервер виконує основну бізнес-логіку та обробку запитів, а клієнт забезпечує зручний інтерфейс для користувача.

Реалізована ТСП-взаємодія між клієнтом і сервером гарантує стабільний обмін даними та підтримку ефективної комунікації в межах системи.

Таким чином, спроектована система «Електронний деканат» відповідає сучасним вимогам інформаційного забезпечення навчальних закладів, демонструючи високу продуктивність, безпеку та зручність у використанні.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Опис серверної частини

Серверна частина системи «Електронний деканат» реалізована на мові програмування Java. Вона виконує роль центрального компонента, що забезпечує обробку запитів клієнта, взаємодію з базою даних і повернення відповідей. Таблиця класи серверної частини системи «Електронний деканат» (див. ДОДАТОК Б табл.Б.1.).

Опис класу ServerWithGUI

Загальний опис

Клас ServerWithGUI (див. ДОДАТОК Г Код класу ServerWithGUI) відповідає за реалізацію серверної частини клієнт-серверної системи «Електронний деканат». Він забезпечує запуск сервера, управління підключеннями клієнтів та взаємодію з ними через TCP-з'єднання. Додатково, клас містить графічний інтерфейс адміністратора для запуску та зупинки сервера, а також для перегляду підключених клієнтів.

Основні компоненти класу

- a) *ServerSocket serverSocket* — серверний сокет, що приймає вхідні з'єднання від клієнтів.
- b) *boolean isRunning* — змінна для контролю стану сервера.
- c) *Thread serverThread* — потік для роботи сервера, що дозволяє обробляти клієнтські підключення у фоновому режимі.
- d) *SessionManager* — внутрішній клас, який керує активними сесіями користувачів.

					ІТС.4КІ.0124.03-ПЗ							
Змн.	Арк.	№ докум.	Підпис	Дата	РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ			Літ.	Арк.	Акрушіє		
Розроб.	Курячий О.В.									28	62	
Керівник	Переяславська С.О.											
Реценз.	Козуб Ю.Г.											
Н. Контр.												
Зав. каф.	Семенов М.А.											
					ЛНУ Кафедра ІТС, Гр.4КІ							

- е) **Графічний інтерфейс (GUI)** — реалізований на основі JFrame, JLabel, JButton, JTextArea, забезпечує керування сервером та відображення підключених клієнтів.

Внутрішній клас *SessionManager*

Цей клас використовується для управління активними сесіями користувачів.

- а) *ConcurrentHashMap<String, Socket> activeSessions* — асоціативний масив, що зберігає підключені сесії (email → socket).
- б) *isUserLoggedIn(String email)* — перевіряє, чи є користувач з таким email в активних сесіях.
- в) *loginUser(String email, Socket socket)* — додає нового користувача в список активних.
- г) *logoutUser(String email)* — видаляє користувача із сесії.

Графічний інтерфейс сервера

- а) **Створення вікна** (*JFrame frame*) — головне вікно сервера.
- б) **Кнопки керування сервером** (*JButton startButton, JButton stopButton*) — запуск та зупинка сервера.
- в) **Текстова область** (*JTextArea connectedClientsArea*) — відображає інформацію про підключених клієнтів.
- г) **Механізм керування подіями** — обробники кнопок для запуску та зупинки сервера.

Методи класу

startServer(JLabel statusLabel, JTextArea connectedClientsArea):

- Запускає сервер у новому потоці, створюючи *ServerSocket* на порту 12345.
- Очікує підключень клієнтів у циклі.

- Приймає вхідні з'єднання (*Socket clientSocket*).
- Запускає окремий потік *ClientHandler* для кожного підключеного клієнта.
- Виводить інформацію про підключених клієнтів у текстовому полі GUI.

stopServer() — Зупиняє сервер шляхом закриття *ServerSocket* і завершення потоків.

Опис класу *ClientHandler*

Загальний опис

Клас *ClientHandler* (див. ДОДАТОК Г Код класу *ClientHandler*) є складовою частиною серверної частини клієнт-серверної системи «Електронний деканат». Його головне призначення — забезпечення паралельної обробки запитів кожного клієнта, підключеного до сервера через TCP-з'єднання. Кожен екземпляр цього класу створюється при підключенні нового клієнта та виконується в окремому потоці.

У рамках своєї роботи клас *ClientHandler* виконує обробку широкого спектру клієнтських запитів, які охоплюють основні операції з базою даних, авторизацію, реєстрацію, а також інформаційні запити. Нижче представлено структурований опис підтримуваних запитів, розподілених за категоріями для зручності використання та супроводу системи. Таблиця Додавання даних (ADD) (див. ДОДАТОК А табл.Б.2.). Таблиця Отримання даних (GET) (див. ДОДАТОК А табл.Б.3.).

Таблиця 3.1.

Таблиця: Видалення даних (DELETE)

Команда	Формат запиту	Опис дії
DELETE_STUDENT	ELETE_STUDENT;Email	Видалення інформації відповідного здобувача освіти з бази та системи

Продовження таблиці 3.1.

DELETE_COURSE	DELETE_COURSE;Назва_курсу	Видалення відповідної дисципліни
DELETE_GROUP	DELETE_GROUP;Назва_групи	Видалення відповідної академічної групи
DELETE_TEACHERS	DELETE_TEACHERS;Email	Видалення інформації про відповідного викладача

Таблиця 3.2.

Таблиця: JSON-запити (спеціальні)

JSON Команда	Формат JSON-запиту	Опис дії
DELETE_STUDENT_FROM_GROUP	{ "command": "DELETE_STUD ENT_FROM_GROUP", "studentID":ID, "groupID": ID }	Видаляє інформацію про відповідного здобувача освіти з академічної групи
DELETE_GRADES	{ "grades": " DELETE_GRADES", "studentID":ID, "courseID":ID }	Видаляє підсумкові бали відповідного здобувача освіти з відповідної дисципліни

Опис класу Courses

Загальний опис

Клас *Courses* є частиною модельного рівня клієнт-серверної системи «Електронний деканат» і представляє сутність «Дисциплін». Цей клас використовується для зберігання базової інформації про дисципліни, такої як його унікальний ідентифікатор та назва. Клас є простим контейнером даних (POJO), який забезпечує зручний доступ до інформації про дисципліни в межах системи.

Таблиця 3.3.

Основні компоненти класу

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

Поле	Тип	Опис
courseID	int	Унікальний ідентифікатор дисципліни
courseName	String	Назва курсу (наприклад, «Програмування», «Математика»)

Таблиця 3.4.

Методи класу

Метод	Тип повернення	Опис
getCourseID()	int	Повертає ідентифікатор дисципліни
getCourseName()	String	Повертає назву дисципліни
toString()	String	Повертає текстове представлення дисципліни — лише її назву. Застосовується при виведенні об'єкта у GUI або логах.

Опис класу CoursesDeleter

Загальний опис

Клас *CoursesDeleter* є службовим компонентом системи «Електронний деканат» і використовується для видалення дисциплін із бази даних. Його основна функція — реалізація логіки видалення запису дисципліни за назвою (*CourseName*) з таблиці Courses. Клас працює з базою даних через JDBC-підключення.

Таблиця 3.5.

Основні компоненти класу

Компонент	Тип	Опис
deleteCourse(String CourseName)	boolean	Метод, який виконує SQL-запит на видалення дисципліни за назвою. Повертає true, якщо курс видалено успішно, або false, якщо курс не знайдено чи сталася помилка.

Алгоритм роботи методу deleteCourse(String CourseName)

Встановлюється з'єднання з базою даних за допомогою *db_dekanat.getConnection()*.

- a) Формується SQL-запит: `DELETE FROM Courses WHERE CourseName = ?`
- b) Використовується *PreparedStatement* для безпечної передачі параметру.
- c) Виконується команда *executeUpdate()* і перевіряється кількість змінених рядків (Якщо *rowsAffected* > 0 — дисципліну успішно видалено, Якщо *rowsAffected* == 0 — дисципліну не знайдено).
- d) У разі помилки виводиться повідомлення *SQLException*.

Опис класу **CoursesManager**

Загальний опис

Клас **CoursesManager** є частиною логіки взаємодії з базою даних у системі «Електронний деканат». Він виконує функції управління дисциплінами: додавання нових дисциплін до таблиці *Courses*, отримання списку викладачів та пошук імені викладача за його ID. Клас використовує JDBC-з'єднання з базою даних для виконання SQL-запитів.

Таблиця 3.6.

Основні компоненти класу

Метод	Тип повернення	Опис
<i>getTeachers()</i>	<code>List<Teacher></code>	Повертає список усіх викладачів з таблиці <i>Teachers</i>
<i>addCourse(String courseName, String description, int credits, int teacherID)</i>	<code>boolean</code>	Додає нову дисципліну до бази даних. Повертає <code>true</code> у разі успіху
<i>getTeacherNameById(int teacherID)</i>	<code>String</code>	Повертає повне ім'я викладача (Ім'я + Прізвище) за його ID

Опис методу *getTeachers()*

- a) Формує SQL-запит `SELECT TeacherID, FirstName, LastName FROM Teachers`.
- b) Повертає список об'єктів `Teacher` зі всіма викладачами.
- c) Може використовуватися для заповнення випадających списків або зв'язування курсів з викладачами.

Опис методу `addCourse(...)`

- a) Приймає параметри курсу: назва, опис, кредити, ID викладача.
- b) Перевіряє наявність викладача в базі (через `getTeacherNameById`).
- c) Виконує `INSERT INTO Courses(...)` для додавання нового запису.
- d) Повертає `true`, якщо запис додано успішно.

Опис методу `getTeacherNameById(int teacherID)`

- a) Пошуковий метод, який виконує SQL-запит до таблиці `Teachers` за `TeacherID`.
- b) Формує повне ім'я викладача у форматі `Ім'я Прізвище`.

Опис класу `CoursesTable`

Загальний опис

Клас ***CoursesTable*** реалізує функціональність отримання інформації про всі курси з бази даних системи «Електронний деканат». Він виконує запит до таблиці `Courses` та формує масив об'єктів `Object[][]`, який може бути використаний для подальшого відображення дисциплін у графічному інтерфейсі або для передавання на клієнтську частину.

Алгоритм роботи методу `coursesTab()`

- a) Встановлюється з'єднання з базою даних за допомогою `db_dekanat.getConnection()`.
- b) Формується SQL-запит: `SELECT * FROM Courses`.

- с) Виконується запит через Statement та зчитується ResultSet.
- д) Для кожного запису зчитуються дані: CourseName, Description, Credits, TeacherID.
- е) Записи зберігаються у двовимірному масиві Object[][] з максимальною довжиною 100 рядків (обмеження умовне).
- ф) Порожні значення (null) замінюються на «NULL» для уникнення помилок при подальшій обробці.

Таблиця 3.7.

Основні компоненти класу

Метод	Тип повернення	Опис
coursesTab()	Object[][]	Повертає таблицю курсів у вигляді масиву об'єктів: courseName, description, credits, teacherID

Таблиця 3.8.

Структура повертаємого масиву

Індекс стовпця	Значення	Тип
0	Назва дисципліни (courseName)	String
1	Опис дисципліни (description)	String
2	Кількість кредитів (credits)	int
3	Ідентифікатор викладача (teacherID)	int

Опис класу db_dekanat

Загальний опис

Клас db_dekanat (див. ДОДАТОК Г Код класу db_dekanat) реалізує функціонал підключення до бази даних системи «Електронний деканат» за допомогою JDBC. Він відповідає за встановлення та закриття з'єднання з СУБД MySQL. Клас є утилітним (static utility class) та надає зручний інтер-

фейс для отримання Connection в інших класах системи.

Алгоритм роботи методу *getConnection()*

- a) Використовує DriverManager.getConnection(...) для встановлення з'єднання з MySQL-сервером.
- b) Параметри (URL, USER, PASSWORD) жорстко задані в класі.
- c) У разі виникнення SQLException помилка передається викликаючому коду.

Алгоритм роботи методу *closeConnection(Connection connection)*

- a) Перевіряє, чи з'єднання не є null.
- b) Викликає connection.close() для його завершення.
- c) У разі помилки при закритті — виводить повідомлення в консоль.

Таблиця 3.9.

Основні компоненти класу

Компонент	Тип	Опис
URL	String	Шлях до бази даних (MySQL)
USER	String	Ім'я користувача бази даних
PASSWORD	String	Пароль для підключення до БД
getConnection()	Connection	Метод для встановлення з'єднання з базою даних
closeConnection(Connection connection)	void	Метод для безпечного закриття з'єднання з базою даних

Опис класу db_log

Загальний опис

Клас db_log (див. ДОДАТОК Г Код класу db_log) реалізує функціональність підключення до окремої бази даних mysite системи «Електронний деканат», яка використовується для автентифікації користувачів, збереження логів або інших даних, пов'язаних з логуванням. Клас є утилітним (static utility class) і забезпечує централізований доступ

доступ до БД через JDBC.

Алгоритм роботи

Ініціалізація драйвера MySQL:

- a) Через `Class.forName(...)` у статичному блоці.
- b) Виконується при першому використанні класу.
- c) У разі помилки — повідомлення виводиться в консоль.

Метод `getConnection()`

- a) Використовує `DriverManager.getConnection(URL, USER, PASSWORD)` для встановлення з'єднання.

Метод `closeConnection(Connection connection)`

- b) Закриває активне з'єднання, якщо воно існує.
- c) Обробляє можливі виключення (`SQLException`).

Таблиця 3.10.

Основні компоненти класу

Компонент	Тип	Опис
URL	String	Шлях до бази даних mysite
USER	String	Ім'я користувача для підключення
PASSWORD	String	Пароль для підключення
<code>getConnection()</code>	Connection	Метод для створення з'єднання з БД
<code>closeConnection(Connection connection)</code>	void	Метод для завершення з'єднання з БД
static блок ініціалізації	—	Завантаження драйвера MySQL <code>com.mysql.cj.jdbc.Driver</code>

Опис класу `GradesDeleter`

Загальний опис

Клас `GradesDeleter` є частиною функціональності керування оцінками в системі «Електронний деканат». Його основне завдання — реалізація логіки видалення оцінок студентів з таблиці `Grades` у базі даних.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

Перед видаленням оцінки клас перевіряє наявність відповідного студента та курсу.

Алгоритм роботи методів

isStudentExists(studentID)

- a) Виконує SQL-запит `SELECT COUNT(*) FROM Students WHERE StudentID = ?`.
- b) Повертає `true`, якщо студент існує, інакше — `false`.

isCourseExists(courseID)

- a) Аналогічно перевіряє існування дисципліни в таблиці `Courses`.

deleteStudentByID(studentID, courseID)

- b) Спочатку перевіряє існування студента та курсу.
- c) Якщо обидва існують — виконує SQL-запит: `DELETE FROM Grades WHERE StudentID = ? AND CourseID = ?`
- d) Повертає `true`, якщо запис видалено успішно.

Таблиця 3.11.

Основні компоненти класу

Метод	Тип повернення	Опис
<code>isStudentExists(int studentID)</code>	<code>boolean</code>	Перевіряє, чи існує здобувач освіти із заданим ID у таблиці <code>Students</code>
<code>isCourseExists(int courseID)</code>	<code>boolean</code>	Перевіряє, чи існує дисципліна із заданим ID у таблиці <code>Courses</code>
<code>deleteStudentByID(int studentID, int courseID)</code>	<code>boolean</code>	Видаляє запис оцінки здобувача з таблиці <code>Grades</code> за відповідними ID

Опис класу `GradesManager`

Загальний опис

Клас `GradesManager` відповідає за взаємодію з таблицями здобувачів освіти та дисциплін в базі даних, а також за додавання здобувачів до академічних дисциплін та ведення записів про їх бали.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас також перевіряє наявність освітян та дисципліни за їхніми ідентифікаторами або іменами перед виконанням операцій.

Алгоритм роботи методів

getStudents()

- a) Виконує SQL-запит для отримання всіх здобувачів з таблиці Students.
- b) Повертає список здобувачів, де кожен освітянин є об'єктом класу Students.

getCourses()

- a) Виконує SQL-запит для отримання всіх дисциплін з таблиці Courses.
- b) Повертає список дисциплін, де кожна дисципліна є об'єктом класу Courses.

addStcu(studentID, courseID)

- a) Перевіряє існування здобувача та дисципліни за допомогою методів *getStudentNameById()* та *getCourseNameById()*.
- b) Якщо освітянин та дисципліна існують, виконується SQL-запит для додавання запису про здобувача в таблицю Grades.
- c) Повертає true, якщо запис додано успішно, інакше — false.

addGrade(studentID, courseID, grade)

- a) Перевіряє існування здобувача та дисципліни за допомогою методів *getStudentNameById()* та *getCourseNameById()*.
- b) Якщо освітянин та дисципліна існують, виконується SQL-запит для додавання балів в таблицю Grades.
- c) Повертає true, якщо оцінка додана успішно, інакше — false.

getCourseNameById(courseID)

- a) Виконує SQL-запит для отримання назви дисципліни за її ID.

- b) Повертає назву дисципліни або порожній рядок, якщо дисципліна не знайдена.

getStudentNameById(studentID)

- a) Виконує SQL-запит для отримання повного імені здобувача освіти за його ID.
- b) Повертає повне ім'я освітянина або порожній рядок, якщо здобувача не знайдено.

getStudentIdByName(studentName)

- a) Виконує SQL-запит для отримання ID здобувача за його повним ім'ям.
- b) Повертає ID здобувача або -1, якщо здобувач не знайдений.

getCourseIdByName(courseName)

- a) Виконує SQL-запит для отримання ID дисципліни за її назвою.
- b) Повертає ID дисципліни або -1, якщо дисципліна не знайдена.

Таблиця 3.12.

Основні компоненти класу

Метод	Тип повернення	Опис
getStudents()	List<Students>	Повертає список всіх здобувачів освіти з таблиці Students.
getCourses()	List<Courses>	Повертає список всіх дисциплін з таблиці Courses.
addStcu(int studentID, int courseID)	boolean	Додає здобувача до дисципліни за його ID.
addGrade(int studentID, int courseID, int grade)	boolean	Додає бали освітянину з відповідної дисципліни.
getCourseNameById(int courseID)	String	Повертає назву дисципліни за її ID.
getStudentNameById(int studentID)	String	Повертає повне ім'я та прізвище здобувача освіти за його ID

Продовження таблиці 3.12.

getStudentIdByName(String studentName)	int	Повертає ID освітянина за його повним ім'ям.
getCourseIdByName(String courseName)	int	Повертає ID дисципліни за її назвою.

Опис класу GradesTable

Загальний опис

Клас GradesTable відповідає за отримання даних балів здобувачів з таблиці Grades бази даних. Він виконує SQL-запит для вибірки всіх записів із цієї таблиці, що містять інформацію про здобувачів, дисципліни та їхні бали, і формує двовимірний масив для подальшого використання.

Таблиця 3.13.

Основні компоненти класу

Метод	Тип повернення	Опис
gradesTab()	Object[][]	Виконує SQL-запит до таблиці Grades, отримує всі записи та формує двовимірний масив з результатами.

Алгоритм роботи методу *gradesTab()*

- Виконується підключення до бази даних за допомогою методу getConnection().
- Виконується SQL-запит: SELECT * FROM Grades.
- Для кожного результату запиту отримуються значення колонок StudentID, CourseID, та Grade, дані додаються до масиву gradesData.
- Повертається масив gradesData, що містить всі отримані записи.

Опис класу GroupAdder

Загальний опис

Клас GroupAdder відповідає за додавання нових груп до таблиці Groupe в базі даних. Він виконує SQL-запит для вставки нових записів, що

містять інформацію про назву групи, спеціальність та рік зарахування, в базу даних.

Таблиця 3.14.

Основні компоненти класу

Метод	Тип повернення	Опис
addGroup()	boolean	Додає нову групу до таблиці Groupe за допомогою SQL-запиту. Повертає true, якщо групу успішно додано, інакше — false.

Алгоритм роботи методу *addGroup()*

- Виконується підготовлений SQL-запит для додавання нового запису до таблиці Groupe.
- Параметри запиту заповнюються переданими значеннями (groupName — назва групи, speciality — спеціальність, enrollmentYear — рік зарахування).
- Якщо вставка запису в таблицю успішна (кількість вставлених рядків більше 0), метод повертає true, інакше — false.

Опис класу GroupDeleter

Загальний опис

Клас GroupDeleter відповідає за видалення груп із таблиці Groupe в базі даних. Він виконує SQL-запит для видалення групи за її назвою, перевіряючи наявність такої групи перед виконанням операції.

Алгоритм роботи методу *deleteGR()*

- Виконується підготовлений SQL-запит для видалення групи за назвою з таблиці Groupe.
- Параметр запиту заповнюється значенням groupName, яке передається методу.

- с) Якщо видалення групи з таблиці успішне (кількість змін у таблиці більше 0), метод повертає true. Якщо ж групи з такою назвою не знайдено, виводиться повідомлення про помилку, а метод повертає false.

Таблиця 3.15.

Основні компоненти класу

Метод	Тип повернення	Опис
deleteGR()	boolean	Видаляє групу з таблиці Groups за її назвою. Повертає true, якщо групу успішно видалено, інакше — false.

Опис класу Groups

Загальний опис

Клас Groups використовується для представлення інформації про групу в системі «Електронний деканат». Він містить дані про ідентифікатор групи та її назву. Цей клас дозволяє зберігати та отримувати інформацію про групи освітян, що є частиною системи керування навчальним процесом.

Таблиця 3.16.

Основні компоненти класу

Метод	Тип повернення	Опис
getGroupID()	int	Повертає ідентифікатор групи.
getGroupName()	String	Повертає назву групи.
toString()	String	Повертає назву групи у вигляді рядка (перевизначення методу toString() для зручності).

Алгоритм роботи методів

- Конструктор класу приймає два параметри: groupID та groupName, і ініціалізує відповідні поля.
- Метод getGroupName() повертає значення назви групи.

- с) Метод toString() перевизначено для того, щоб при виклику цього методу об'єкт класу відображався як назва групи.

Опис класу groupTable

Загальний опис

Клас groupTable використовується для отримання даних про групи з бази даних та представлення їх у вигляді двовимірного масиву, що підходить для подальшого використання в таблицях або інтерфейсах користувача. Він здійснює з'єднання з базою даних, виконує SQL-запит для отримання інформації про групи та повертає результат у форматі, зручному для подальшої обробки.

Таблиця 3.17.

Основні компоненти класу

Метод	Тип повернення	Опис
groupTab()	Object[][]	Повертає двовимірний масив, який містить дані про групи. Для кожної групи зберігається її назва, спеціальність та рік набору.

Алгоритм роботи методу groupTab()

- Метод створює двовимірний масив groupData розміром 100x3, що містить інформацію про групи: назву групи, спеціальність і рік набору.
- Виконується SQL-запит для вибірки всіх груп з таблиці Groups. Для кожного рядка результату запиту (отримуються значення для GroupName, Specialty та enrollmentYear; якщо значення не є NULL або порожнім, воно додається в масив; якщо значення є NULL, то додається рядок «NULL»).
- Після отримання всіх даних результат виводиться в консоль (загальний підсумок по кількості груп).

- d) Метод повертає двовимірний масив `groupData`, який можна використовувати для відображення даних на інтерфейсі користувача або іншій обробці.

Опис класу `StGrDeleter`

Загальний опис

Клас `StGrDeleter` використовується для перевірки існування здобувачів та груп у базі даних, а також для видалення зв'язку між здобувачем та групою в таблиці `StudentGroups`. Він забезпечує правильну логіку видалення освітян із відповідних груп, гарантуючи цілісність даних та мінімізуючи помилки. Перед видаленням виконується перевірка наявності здобувача та групи в базі даних за їхніми ідентифікаторами.

Таблиця 3.18.

Основні компоненти класу

Метод	Тип повернення	Опис
<code>isStudentExists()</code>	boolean	Перевіряє, чи існує здобувач в таблиці <code>Students</code> за <code>StudentID</code> .
<code>isGroupExists()</code>	boolean	Перевіряє, чи існує група в таблиці <code>Groupe</code> за <code>GroupID</code> .
<code>deleteStudentByID()</code>	boolean	Видаляє запис про належність здобувача до групи у таблиці <code>StudentGroups</code> за <code>StudentID</code> та <code>GroupID</code> . Повертає <code>true</code> , якщо видалення було успішним.

Алгоритм роботи методу `deleteStudentByID()`

- Виконується перевірка, чи існує здобувач з вказаним `StudentID` у таблиці `Students`.
- Якщо здобувач існує, перевіряється наявність групи з вказаним `GroupID` у таблиці `Groupe`.
- Якщо обидві перевірки успішні, виконується SQL-запит на видалення

запису з таблиці StudentGroups, де StudentID і GroupID відповідають заданим.

- d) У разі успішного видалення виводиться повідомлення про успішне завершення операції.
- e) Метод повертає логічне значення true або false залежно від результату операції.

Опис класу StGrManager

Загальний опис

Клас StGrManager реалізує функціонал керування здобувачами освіти та академічними групами в інформаційній системі «Електронний деканат». Основні задачі класу — отримання списків здобувачів освіти та груп, додавання здобувача освіти до групи, а також пошук імен або ідентифікаторів за відповідними даними.

Клас забезпечує зв'язок між таблицями Students (здобувачі освіти), Groupe (академічні групи) та StudentGroups у базі даних, дозволяючи інтегрувати ці дані для подальшого використання в інтерфейсі користувача або інших модулях системи.

Алгоритм роботи:

- a) З'єднання з базою даних через клас db_dekanat.
- b) Отримання даних про здобувачів освіти або групи за допомогою SQL-запитів.
- c) Перевірка наявності записів перед виконанням дій (наприклад, перед додаванням у групу).
- d) Додавання зв'язку між здобувачем освіти та групою в таблиці StudentGroups.

Таблиця 3.19.

Основні компоненти класу

					ІТС.4КІ.0124.03-ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

Метод	Тип повернення	Опис
getStudents()	List<Students>	Повертає список здобувачів освіти з бази даних.
getGroups()	List<Groups>	Повертає список академічних груп з бази даних.
addStGr(studentID, groupID)	boolean	Додає здобувача освіти до групи за відповідними ідентифікаторами.
getStudentNameById(id)	String	Повертає прізвище та ім'я здобувача освіти за його ID.
getGroupNameById(id)	String	Повертає назву групи за її ID.
getStudentIdByName(name)	int	Повертає ID здобувача освіти за його прізвищем та ім'ям.
getGroupIdByName(name)	int	Повертає ID групи за її назвою.

Опис класу StudentAdder

Загальний опис

Клас StudentAdder призначений для додавання нових здобувачів освіти до бази даних інформаційної системи «Електронний деканат». Він реалізує функціонал вставки нового запису до таблиці Students, зберігаючи основну інформацію про здобувача освіти: ім'я, прізвище, електронну пошту, номер телефону, дату народження та рік вступу.

Алгоритм роботи методу *addStudent()*

- Формується SQL-запит для вставки даних у таблицю Students.
- Використовується підготовлений запит (PreparedStatement), щоб запобігти SQL-ін'єкціям.
- Параметри заповнюються значеннями, переданими до методу.
- Виконується вставка нового запису в базу даних.
- У випадку успішної вставки виводиться повідомлення в консоль.

Якщо виникає помилка — відображається відповідне повідомлення про помилку SQL.

- f) Метод повертає true, якщо запис було додано, або false — якщо виникла помилка чи запис не було вставлено.

Таблиця 3.20.

Основні компоненти класу

Метод	Тип повернення	Опис
addStudent(firstName, lastName, email, phone, birthDate, enrollmentYear)	boolean	Додає нового здобувача освіти до таблиці Students. Повертає true, якщо додавання успішне, і false — у разі помилки або невдалої спроби вставки.

Опис класу StudentDeleter

Загальний опис

Клас StudentDeleter призначений для реалізації функціоналу видалення здобувачів освіти з бази даних у межах інформаційної системи «Електронний деканат». Він дозволяє здійснювати вилучення записів з таблиці Students за вказаними критеріями — електронною адресою або унікальним ідентифікатором (ID) здобувача освіти.

Алгоритм роботи методів:

- Підключення до бази даних через db_dekanat.getConnection().
- Формується SQL-запит на видалення запису з таблиці Students відповідно до параметра (за електронною адресою (email) — метод deleteStudent; за унікальним ідентифікатором (ID) — метод deleteStudentById).
- Підготовка запиту (PreparedStatement) для запобігання SQL-ін'єкціям.
- Виконується запит і перевіряється кількість змінених рядків (rowsAffected).

- е) Якщо хоча б один запис було видалено, система виводить повідомлення про успішну операцію.
- ф) Якщо записів не знайдено — повідомлення про відсутність відповідного здобувача освіти.
- г) Метод повертає true, якщо видалення відбулося, або false — у разі невдачі чи помилки.

Таблиця 3.21.

Основні компоненти класу

Метод	Тип повернення	Опис
deleteStudent(email)	boolean	Видаляє здобувача освіти з бази даних за вказаною електронною поштою. Повертає true, якщо запис успішно видалено.
deleteStudentById(studentId)	boolean	Видаляє здобувача освіти з бази даних за його ID. Повертає true, якщо запис успішно видалено.

Опис класу StudentGroupsManager

Загальний опис

Клас StudentGroupsManager призначений для керування зв'язками між здобувачами освіти та академічними групами в інформаційній системі «Електронний деканат». Він реалізує функціонал отримання списків здобувачів та груп, а також додавання записів до таблиці StudentGroups, що відображає належність здобувачів до певних академічних груп.

Алгоритм роботи основних методів

Метод *addStGroup1(studentID)*:

- а) Отримує прізвище та ім'я здобувача освіти за переданим ID.
- б) Формує SQL-запит для вставки StudentID та GroupID у таблицю StudentGroups.

- с) Використовує PreparedStatement для захисту від SQL-ін'єкцій.
- d) Виконує запит.
- е) Виводить повідомлення про успіх або помилку.
- ф) Повертає true, якщо вставку здійснено успішно.

Метод *addStGroup2(groupID)* — Аналогічний до *addStGroup1*, але вставляє лише *GroupID*.

Таблиця 3.22.

Основні компоненти класу

Метод	Тип повернення	Опис
<i>getStudents()</i>	List<Students>	Повертає список усіх здобувачів освіти з таблиці <i>Students</i> .
<i>getStudentsNameById(studentID)</i>	String	Повертає ім'я та прізвище здобувача освіти за його ідентифікатором.
<i>addStGroup1(studentID)</i>	boolean	Додає запис до таблиці <i>StudentGroups</i> , пов'язуючи здобувача з групою. Повертає true, якщо додавання успішне
<i>addStGroup2(groupID)</i>	boolean	Додає запис до таблиці <i>StudentGroups</i> , додаючи групу (без конкретного здобувача). Повертає true, якщо додавання успішне.
<i>getGroupsNameById(groupID)</i>	String	Повертає назву групи за її ідентифікатором.
<i>Grid.getGroups()</i>	List<Groups>	Повертає список усіх академічних груп з таблиці <i>Groups</i> .

Опис класу StudentGroupsTable

Загальний опис

Клас *StudentGroupsTable* призначений для отримання інформації про здобувачів освіти та їх групи з таблиці *StudentGroups* бази даних інформаційної системи «Електронний деканат». Клас надає метод для

отримання всіх записів здобувачів освіти та їхніх груп і зберігає ці дані у двовимірному масиві, що дозволяє зручно працювати з інформацією.

Алгоритм роботи методу *stgroupsTab()*

- a) Формується SQL-запит для отримання всіх записів з таблиці StudentGroups.
- b) Використовується підготовлений запит (Statement) для виконання запиту.
- c) Для кожного запису з результату запиту, значення StudentID та GroupID зберігаються в масиві stgrData.
- d) Після обробки всіх записів масив повертається для подальшої роботи.

Таблиця 3.23.

Основні компоненти класу

Метод	Тип повернення	Опис
stgroupsTab()	Object[][]	Метод отримує всі записи зі зв'язку між здобувачами освіти та їх групами з таблиці StudentGroups і повертає їх у вигляді двовимірного масиву.

Опис класу Students

Загальний опис

Клас Students представляє модель для збереження та обробки даних про здобувачів освіти в інформаційній системі «Електронний деканат». Клас включає основні атрибути, такі як ID здобувача освіти, та ім'я. Цей клас використовується для зберігання та передачі даних про здобувачів освіти, а також для їх подальшої обробки в різних частинах системи.

Алгоритм роботи методу *toString()*

Метод toString() повертає повне ім'я здобувача освіти у вигляді рядка, складаючи його з імені та прізвища.

Таблиця 3.24.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

Основні компоненти класу

Метод	Тип повернення	Опис
getStudentID()	int	Метод для отримання ID здобувача освіти.
getFirstName()	String	Метод для отримання імені здобувача освіти.
getLastName()	String	Метод для отримання прізвища здобувача освіти.
toString()	String	Метод для отримання повного імені здобувача освіти (ім'я + прізвище) у вигляді рядка.

Опис класу StudentTable

Загальний опис

Клас StudentTable призначений для отримання інформації про здобувачів освіти з таблиці Students бази даних інформаційної системи «Електронний деканат». Він надає метод для отримання всіх записів про здобувачів освіти і зберігає їх у двовимірному масиві, що дозволяє зручно працювати з даними. Клас також обробляє можливі значення NULL для кожного з атрибутів здобувачів освіти.

Алгоритм роботи методу *studtab()*

- Підключення до бази даних за допомогою методу `db_dekanat.getConnection()`.
- Формується SQL-запит для отримання всіх записів з таблиці Students.
- Використовується Statement для виконання запиту та обробки результатів.
- Перевірка на NULL для кожного значення, що дозволяє уникнути помилок при роботі з даними.
- Результати запиту додаються до двовимірного масиву `studentsData`.

- g) Після виконання запиту виводиться кількість отриманих записів у консоль.

Таблиця 3.25.

Основні компоненти класу

Метод	Тип повернення	Опис
studtab()	Object[][]	Метод для отримання всіх записів про здобувачів освіти з таблиці Students і збереження їх у двовимірному масиві.

Опис класу Teacher

Загальний опис

Клас Teacher призначений для зберігання інформації про викладачів у системі «Електронний деканат». Він містить основні атрибути викладача: ID, ім'я та прізвище, а також відповідні методи для доступу до цих даних. Клас реалізує метод toString(), що дозволяє представляти викладача у вигляді рядка, що містить його ім'я та прізвище.

Алгоритм роботи класу

- У конструкторі класу ініціалізуються поля: teacherID, firstName та lastName.
- Метод getTeacherID() повертає ID викладача.
- Метод getFirstName() повертає ім'я викладача.
- Метод getLastName() повертає прізвище викладача.
- Перевизначений метод toString() формує рядок у форматі «ім'я прізвище», що використовується, наприклад, у компонентах GUI, таких як JComboBox.

Таблиця 3.26.

Основні компоненти класу

Метод	Тип повернення	Опис
getTeacherID()	int	Метод для отримання ID викладача.

Продовження таблиці 3.26.

getFirstName()	String	Метод для отримання імені викладача.
getLastName()	String	Метод для отримання прізвища викладача.
toString()	String	Перевизначений метод toString(), що повертає ім'я та прізвище викладача у вигляді рядка.

Опис класу TeachersAdder

Загальний опис

Клас TeachersAdder призначений для додавання нових викладачів до бази даних інформаційної системи «Електронний деканат». Клас містить метод для вставки нового запису в таблицю Teachers, зберігаючи основну інформацію про викладача, таку як ім'я, прізвище, електронну пошту, номер телефону та відділ.

Таблиця 3.27.

Основні компоненти класу

Метод	Тип повернення	Опис
addTeacher(firstName, lastName, email, phone, department)	boolean	Метод для додавання нового викладача до таблиці Teachers. Повертає true, якщо додавання успішне, і false — у разі помилки або невдалої спроби вставки.

Алгоритм роботи методу *addTeacher()*

- Формується SQL-запит для вставки даних у таблицю Teachers.
- Використовується підготовлений запит (PreparedStatement), щоб уникнути SQL-ін'єкцій.
- Параметри заповнюються значеннями, переданими до методу.
- Виконується вставка нового запису в базу даних.
- Якщо кількість вставлених рядків більше 0, виводиться повідомлення
- про успішне додавання викладача. В іншому випадку — повідомлення про невдачу.

- g) Метод повертає true, якщо запис було додано, або false, якщо виникла помилка.

Опис класу TeachersDeleter

Загальний опис

Клас TeachersDeleter призначений для видалення викладачів з бази даних інформаційної системи «Електронний деканат». Клас містить метод для видалення викладача з таблиці Teachers за його електронною поштою.

Таблиця 3.28.

Основні компоненти класу

Метод	Тип повернення	Опис
deleteTeacher(email)	boolean	Метод для видалення викладача з таблиці Teachers за електронною поштою. Повертає true, якщо викладач був успішно видалений, і false — у разі помилки або якщо викладач не знайдений.

Алгоритм роботи методу *deleteTeacher()*

- Формується SQL-запит для видалення викладача за допомогою його електронної пошти.
- Використовується підготовлений запит (PreparedStatement), щоб уникнути SQL-ін'єкцій.
- Параметр запиту заповнюється значенням електронної пошти викладача.
- Виконується операція видалення в базі даних.
- Якщо кількість вплинутих рядків більше 0, виводиться повідомлення про успішне видалення викладача. Якщо викладач не знайдений, виводиться відповідне повідомлення.

- f) Метод повертає true, якщо викладач був видалений, або false, якщо виникла помилка чи викладач не був знайдений.

Опис класу TeachersManager

Загальний опис

Клас TeachersManager призначений для управління даними викладачів в інформаційній системі «Електронний деканат». Клас надає методи для отримання імені та прізвища викладача за його ID, а також для отримання ID викладача за його повним ім'ям.

Алгоритм роботи методів

getTeacherNameById(teacherId)

- Формується SQL-запит для отримання імені та прізвища викладача за його ID.
- Використовується підготовлений запит (PreparedStatement), щоб уникнути SQL-ін'єкцій.
- Параметр запиту заповнюється ID викладача.
- Виконується запит, і якщо викладач знайдений, формується та повертається його повне ім'я.
- Якщо викладача не знайдено, виводиться повідомлення про помилку.

getTeacherIdByName(teacherFullName)

- Формується SQL-запит для отримання ID викладача за його повним ім'ям.
- Встановлюється параметр запиту як повне ім'я викладача.
- Виконується запит, і якщо викладач знайдений, повертається його ID. Якщо викладач не знайдений, повертається -1.

Таблиця 3.29.

Основні компоненти класу

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

Метод	Тип повернення	Опис
getTeacherNameById(teacherId)	String	Отримує ім'я та прізвище викладача за його ID. Повертає повне ім'я викладача або повідомлення, що викладач не знайдений.
getTeacherIdByName(teacherFullName)	int	Отримує ID викладача за його повним ім'ям. Повертає ID викладача або -1, якщо викладач не знайдений.

Опис класу TeachersTable

Загальний опис

Клас TeachersTable призначений для отримання інформації про викладачів з таблиці Teachers бази даних інформаційної системи «Електронний деканат». Клас надає метод для отримання всіх записів викладачів та зберігає їх у двовимірному масиві, що дозволяє зручно працювати з даними.

Алгоритм роботи методу *teachtab()*

- Формується SQL-запит для отримання всіх записів з таблиці Teachers.
- Використовується підготовлений запит (PreparedStatement) для виконання SQL-запиту, щоб уникнути SQL-ін'єкцій.
- Виконується запит, і результати обробляються за допомогою ResultSet.
- Для кожного викладача з результату запиту отримуються значення таких полів: ім'я, прізвище, електронна пошта, телефон, відділ. Якщо значення виявляються відсутніми (NULL), замість них вставляється значення «NULL».
- Отримані дані зберігаються в двовимірному масиві teachData.
- Після обробки всіх записів метод повертає масив даних.

Основні компоненти класу

Метод	Тип повернення	Опис
teachtab()	Object[][]	Отримує дані всіх викладачів з таблиці Teachers і повертає їх у вигляді двовимірного масиву об'єктів.

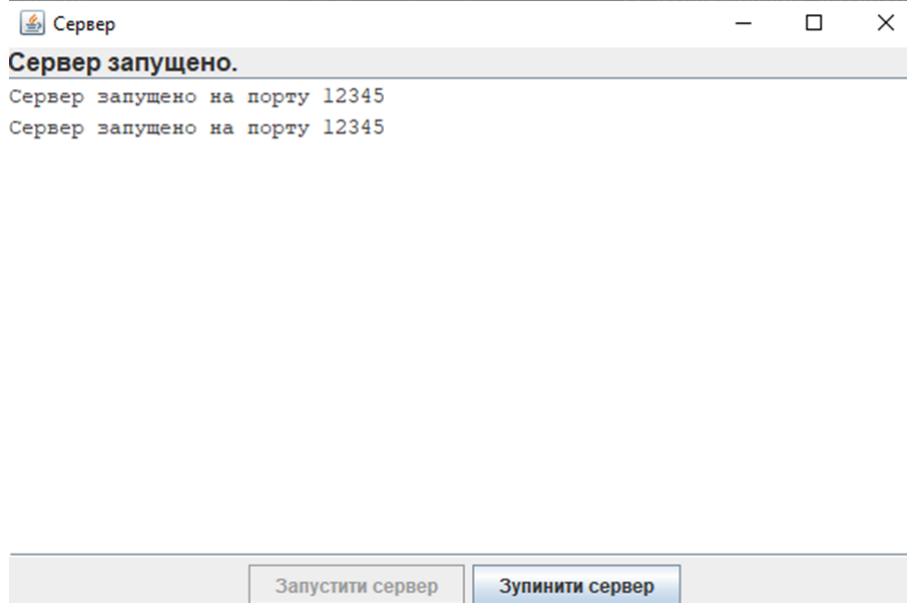


Рис. 3.1. – Графічний інтерфейс запущеного сервера

3.2. Опис клієнтської частини

Клієнтська частина системи «Електронний деканат» розроблена мовою Java та реалізована у вигляді настільного додатку з графічним інтерфейсом (GUI). Вона забезпечує зручний доступ до функціональності системи, включаючи перегляд курсів, студентів, викладачів, оцінок та управління цими даними. Таблиця класи клієнтської частини системи «Електронний деканат» (див. ДОДАТОК Б табл.Б.4.).

Опис класу Client**Загальний опис**

Клас Client відповідає за реалізацію клієнтської частини програми, що дозволяє користувачу зареєструватися, взаємодіючи з сервером.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		58

Включає графічний інтерфейс користувача (GUI), обробку подій для кнопок входу та реєстрації, а також хешування паролів для безпечної передачі через мережу.

Таблиця 3.31.

Основні компоненти класу

Метод	Тип повернення	Опис
hashPassword(String password)	String	Метод для хешування пароля за допомогою алгоритму SHA-256.
main(String[] args)	void	Головний метод програми, що ініціалізує графічний інтерфейс і обробляє події для кнопок входу та реєстрації.
isLoggedIn()	boolean	Метод для перевірки, чи користувач успішно увійшов до системи.

Алгоритм роботи

Ініціалізація графічного інтерфейсу

У головному методі main створюється вікно з полями для введення email та пароля, а також кнопками для входу та реєстрації. Всі ці елементи додаються до JPanel, а потім — до основного вікна.

Обробка події кнопки «Вхід»

При натисканні на кнопку «Вхід», клієнт:

- Підключається до сервера, використовуючи методи з класу ServerConnection.
- Отримує введені користувачем email та пароль.
- Хешує пароль за допомогою методу hashPassword.
- Відправляє запит на сервер у форматі LOGIN;email;hashedPassword.
- Читає відповідь від сервера і в залежності від відповіді або відображає повідомлення про успішний вхід, або про невірний логін/пароль.

- f) Якщо вхід успішний, вікно входу закривається і відкривається головне меню.

Обробка події кнопки «Реєстрація»

При натисканні на кнопку «Реєстрація», відкривається нове вікно для реєстрації користувача. Для цього викликається клас RegistrationForm.

Метод хешування пароля

Метод `hashPassword` використовує алгоритм SHA-256 для перетворення введеного пароля в хеш, що забезпечує безпеку передачі пароля через мережу. Пароль після хешування передається на сервер.

Рис. 3.2 – Форма клієнта

Опис класу Configuration

Загальний опис

Клас Configuration є службовим класом конфігурації, який зберігає константні параметри підключення клієнта до сервера, зокрема IP-адресу сервера (HOST) та порт (PORT), через який здійснюється обмін даними між клієнтом і сервером.

Таблиця 3.32.

Основні компоненти класу

Поле	Тип	Значення	Опис
HOST	String	«192.168.186.146»	IP-адреса сервера, до якого підключається клієнт.
PORT	Int	12345	Порт, на якому працює сервер і приймає з'єднання.

Опис класу CourseDetailsForm

					<i>ITC.4KI.0124.03-ПЗ</i>	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Загальний опис

Клас `CourseDetailsForm` реалізує графічний інтерфейс користувача для перегляду, додавання та управління даними про навчальні дисципліни. Він дозволяє користувачу бачити перелік наявних дисциплін, додавати нові дисципліни та вибирати викладача зі списку. Клас здійснює обмін даними із сервером через сокет-з'єднання, використовуючи стандартні запити у форматі команд. Таблиця Основні компоненти класу (див. ДОДАТОК Б табл.Б.5.).

Алгоритм роботи

Ініціалізація графічного інтерфейсу

В конструкторі `CourseDetailsForm` створюється вікно з таблицею дисциплін, полями введення для назви дисципліни, опису, кількості кредитів та випадаючим списком викладачів. Всі елементи додаються до панелі з відповідним розташуванням (`BorderLayout`).

Отримання списку викладачів з сервера

Метод `fetchTeachersDataFromServer` надсилає запит `GET_TEACHERS` на сервер. Сервер повертає JSON-масив викладачів. Кожен викладач додається до випадаючого списку `teacherComboBox`.

Отримання переліку дисциплін з сервера

Метод `fetchCoursesFromServer` надсилає запит `GET_COURSES` на сервер. Отримана відповідь (JSON-масив дисциплін) додається до таблиці, що відображається користувачу.

Додавання нової дисципліни

Після натискання кнопки «Додати дисципліну», дані з полів форми зчитуються, формується запит `ADD_COURSE;courseName;description;credits;teacherName;` який надсилається на сервер. Після відповіді — таблиця оновлюється.

					<i>ITS.4KI.0124.03-ПЗ</i>	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Методика взаємодії з сервером

Всі запити відправляються через сокет-з'єднання до сервера за адресою Configuration.HOST та портом Configuration.PORT. Команди передаються у вигляді рядків, відповіді — у вигляді рядків або JSON.

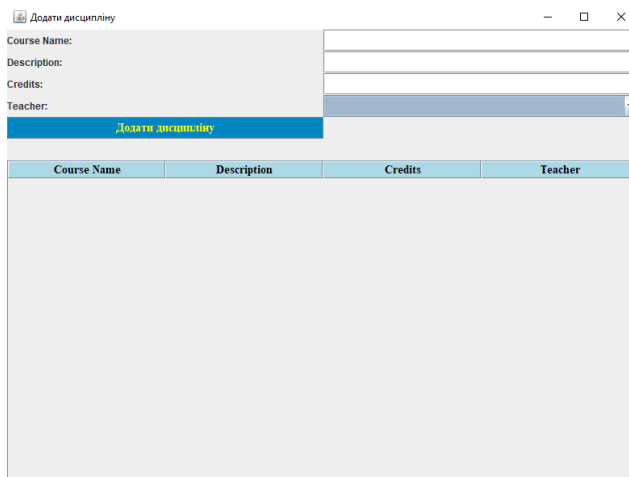


Рис. 3.3 – Форма для додавання дисциплін в систему

Опис класу CoursesForm

Загальний опис

Клас CoursesForm реалізує графічний інтерфейс користувача для перегляду, оновлення та видалення дисциплін у системі управління навчальним процесом. Клас надає зручний інтерфейс для відображення наявних дисциплін, з можливістю вибору дисципліни для видалення. Взаємодія з сервером здійснюється через сокет-з'єднання, а запити й відповіді обробляються у форматі команд.

Таблиця 3.33.

Основні компоненти класу

Метод	Тип повернення	Опис
CoursesForm()	Конструктор	Ініціалізує графічний інтерфейс форми, створює таблицю дисциплін, кнопки для оновлення та видалення дисциплін.

Продовження таблиці 3.33.

fetchCoursesFromServer()	void	Надсилає запит на сервер для отримання переліку дисциплін та відображає їх у таблиці.
sendDeleteRequest(String request)	void	Надсилає запит на видалення дисципліни на сервер, після чого оновлює таблицю.

Алгоритм роботи

- a) **Ініціалізація графічного інтерфейсу:** У конструкторі класу створюється вікно з таблицею для відображення дисциплін, кнопками для оновлення та видалення дисциплін, а також прогрес-баром для індикації процесів. Всі елементи додаються до панелі із відповідним розташуванням.
- b) **Отримання переліку дисциплін з сервера:** Метод fetchCoursesFromServer надсилає запит GET_COURSES до сервера для отримання списку дисциплін. Отримана відповідь (JSON-масив дисциплін) обробляється та відображається у таблиці.
- c) **Видалення дисципліни:** Користувач може вибрати дисципліну для видалення через таблицю. Після натискання кнопки «Видалити дисципліну» формується запит для видалення дисципліни, який надсилається на сервер. Після успішного видалення таблиця оновлюється.

Методика взаємодії з сервером

Запити та відповіді між клієнтом і сервером здійснюються через сокет-з'єднання на адресу, вказану в Configuration.HOST, та порт Configuration.PORT. Команди передаються у вигляді рядків, а відповіді — у вигляді рядків або JSON-масивів.

Опис класу GForm

Загальний опис

					ITC.4KI.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

Клас GForm реалізує графічний інтерфейс користувача для перегляду, оновлення та видалення груп у системі управління навчальним процесом. Клас надає зручний інтерфейс для відображення наявних груп, з можливістю вибору групи для видалення. Взаємодія з сервером здійснюється через сокет-з'єднання, а запити й відповіді обробляються у форматі команд.

Алгоритм роботи

- а) **Ініціалізація графічного інтерфейсу:** У конструкторі класу створюється вікно з таблицею для відображення груп, кнопками для оновлення та видалення груп, а також прогрес-баром для індикації завантаження даних. Всі елементи додаються до панелі із відповідним розташуванням.
- б) **Отримання переліку груп з сервера:** Метод fetchDataFromServer надсилає запит GET_GROUP до сервера для отримання списку груп. Отримана відповідь (JSON-масив груп) обробляється та відображається у таблиці.
- в) **Видалення групи:** Користувач може вибрати групу для видалення через таблицю. Після натискання кнопки «Видалити групу» формується запит для видалення групи, який надсилається на сервер. Після успішного видалення таблиця оновлюється.

Таблиця 3.34.

Основні компоненти класу

Метод	Тип повернення	Опис
GForm()	Конструктор	Ініціалізує графічний інтерфейс форми, створює таблицю для груп, кнопки для оновлення та видалення груп.

Продовження таблиці 3.34.

fetchDataFromServer()	void	Надсилає запит на сервер для отримання переліку груп та відображає їх у таблиці.
sendDeleteRequest(String request)	void	Надсилає запит на видалення групи на сервер, після чого оновлює таблицю.

Методика взаємодії з сервером

Запити та відповіді між клієнтом і сервером здійснюються через сокет-з'єднання на адресу, вказану в Configuration.HOST, та порт Configuration.PORT. Команди передаються у вигляді рядків, а відповіді — у вигляді рядків або JSON-масивів.

Опис класу GradesAdd

Загальний опис

Клас GradesAdd створює графічний інтерфейс для виставлення балів здобувачам освіти за різними дисциплінами. Клас надає можливість вибору освітянина та дисципліни, введення оцінки та відправлення цих даних на сервер. Окрім того, вікно також відображає таблицю з наявними балами здобувачів за дисциплінами. Дані отримуються та відправляються на сервер через сокет-з'єднання.

Алгоритм роботи

- а) **Ініціалізація графічного інтерфейсу:** У конструкторі класу створюється форма з таблицею для відображення балів, випадючими списками для вибору здобувача та дисципліни, а також полем для введення оцінки. Дані для здобувачів та дисциплін отримуються з сервера при завантаженні форми.
- б) **Виставлення балів:** Користувач вибирає здобувача, дисципліну та вводить оцінку, після чого натискає кнопку для відправлення даних на сервер. Оцінка разом із вибраними даними відправляється на сервер, і після отримання відповіді таблиця оновлюється.

- с) Отримання даних з сервера: Дані для здобувачів, дисциплін та балів отримуються за допомогою методів, що надсилають відповідні запити на сервер, а отримані дані відображаються в інтерфейсі. Підсумки відображаються в таблиці.

Методика взаємодії з сервером

Запити до сервера надсилаються через сокет-з'єднання на адресу, вказану в Configuration.HOST, та порт Configuration.PORT. Запити формуються як рядки, а відповіді повертаються у вигляді рядків. Для обробки кожної відповіді використовується функціональний інтерфейс ResponseHandler.

Таблиця 3.35.

Основні компоненти класу

Метод	Тип повернення	Опис
GradesAdd()	Конструктор	Ініціалізує графічний інтерфейс форми для виставлення балів, з таблицею та полями для введення даних.
addGrade()	void	Виставляє бали здобувачу, перевіряючи наявність всіх необхідних даних, та надсилає їх на сервер.
fetchStudentDataFromServer()	void	Отримує список здобувачів з сервера та заповнює випадаючий список для вибору здобувача.
fetchGroupDataFromServer()	void	Отримує список дисциплін з сервера та заповнює випадаючий список для вибору дисципліни.
fetchSortedStudentsFromServer()	void	Отримує бали освітян за дисциплінами з сервера і відображає їх у таблиці.

Продовження таблиці 3.35.

sendRequestToServer(String, ResponseHandler)	void	Надсилає запит на сервер та обробляє відповідь через переданий обробник.
--	------	--

Рис. 3.4 – Форма для виставлення підсумків в системі

Опис класу GradesForm

Загальний опис

Клас GradesForm реалізує графічний інтерфейс користувача для перегляду та управління балами здобувачів освіти за дисциплінами. Клас дозволяє користувачу побачити таблицю з балами здобувачів освіти, дисциплін та можливістю видаляти бали для конкретного здобувача освіти. Клас здійснює обмін даними із сервером через сокет-з'єднання за допомогою стандартних запитів у форматі команд.

Алгоритм роботи

Ініціалізація графічного інтерфейсу

У конструкторі GradesForm створюється вікно з таблицею для відображення дисциплін, полями для відображення балів здобувачів освіти, прогрес-баром та кнопками для оновлення таблиці та видалення балів.

Отримання даних про бали здобувачів освіти з сервера

Метод fetchFromServer надсилає запит GET_GRADES на сервер, отримує список балів здобувачів освіти та дисциплін у форматі JSON і заповнює таблицю відповідними даними.

Видалення балів здобувача освіти

					ІТС.4КІ.0124.03-ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

Після натискання кнопки «Видалити бали студента» дані з вибраного рядка таблиці передаються на сервер для видалення балів здобувача освіти за відповідною дисципліною. Запит на сервер формується в JSON-форматі.

Методика взаємодії з сервером

Усі запити відправляються через сокет-з'єднання до сервера за адресою Configuration.HOST та портом Configuration.PORT. Запити передаються у вигляді рядків, відповіді — у вигляді рядків або JSON.

Таблиця 3.36.

Основні компоненти класу

Метод	Тип повернення	Опис
GradesForm()	Конструктор	Ініціалізує графічний інтерфейс форми, створює таблицю для відображення балів, прогрес-бар, кнопки для оновлення таблиці та видалення балів здобувачів освіти.
deletegrades()	void	Видаляє бали здобувача освіти за вибраною дисципліною з таблиці.
sendDeleteRequest(String request)	void	Надсилає запит на сервер для видалення балів здобувача освіти за дисципліною.
fetchFromServer()	void	Надсилає запит на сервер для отримання даних про бали здобувачів освіти та дисциплін і відображає їх у таблиці.

Опис класу GroupDetailsForm

Загальний опис

Клас GroupDetailsForm реалізує графічний інтерфейс для додавання нових груп та перегляду існуючих груп у системі. Клас забезпечує взаємодію з сервером для отримання та додавання даних про групи здобувачів освіти, відображаючи їх у таблиці, а також надає можливість додавати нові групи через форму введення.

Алгоритм роботи

Ініціалізація графічного інтерфейсу

У конструкторі GroupDetailsForm створюється вікно для введення нової групи, таблиця для відображення груп та прогрес-бар для індикації завантаження даних.

Отримання даних про групи з сервера

Метод fetchDataFromServer надсилає запит до сервера для отримання даних про групи здобувачів освіти, отримує відповідь у форматі JSON і оновлює таблицю.

Таблиця 3.37.

Основні компоненти класу

Метод	Тип повернення	Опис
GroupDetailsForm()	Конструктор	Ініціалізує графічний інтерфейс для додавання групи та перегляду даних про групи, налаштовує таблицю та форми введення.
sendAddStudentRequest(String groupName, String specialty, int enrollmentYear)	void	Відправляє запит на сервер для додавання нової групи з відповідними даними.
fetchDataFromServer()	void	Отримує список груп здобувачів освіти з сервера та оновлює таблицю.

Отримання даних про групи з сервера

Метод fetchDataFromServer надсилає запит до сервера для отримання даних про групи здобувачів освіти, отримує відповідь у форматі JSON і оновлює таблицю.

Додавання нової групи

При натисканні кнопки «Створити групу», введені дані (назва групи, спеціальність та рік вступу) відправляються на сервер для додавання нової групи через метод sendAddStudentRequest.

Методика взаємодії з сервером

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

Запити відправляються через сокет-з'єднання до сервера за адресою Configuration.HOST і портом Configuration.PORT. Запит на отримання груп містить команду «GET_GROUP», а для додавання групи — «ADD_GROUP» із відповідними параметрами.

GroupName	Specialty	EnrollmentYear

Рис. 3.5 – Форма для додавання груп в системі

Опис класу MenuForm

Загальний опис

Клас MenuForm реалізує головне вікно програми з вкладками для управління різними аспектами системи. У вікні є вкладки для роботи з формамиздобувачів, викладачів, дисциплін, груп, балів і загальних дій (вихід). Кожна вкладка містить набори кнопок, що дозволяють відкривати відповідні форми для взаємодії з даними.

Алгоритм роботи

Ініціалізація інтерфейсу

У конструкторі MenuForm налаштовуються вкладки для роботи з різними формами, такими якздобувачі, викладачі, дисципліни тощо. Для кожної вкладки створюється окрема панель з кнопками, кожна з яких викликає відповідну дію.

Відкриття форм

Для кожної вкладки на панелі є кнопки, що відкривають різні форми, наприклад, для додавання або перегляду даних проздобувачів, викладачів, дисципліни тощо. Кожна кнопка викликає відповідний метод, який відкриває нову форму.

Методика взаємодії з сервером

У даному класі немає безпосередньої взаємодії з сервером, але всі виклики методів відкривають відповідні форми для керування даними.

Таблиця 3.38.

Основні компоненти класу

Метод	Тип повернення	Опис
MenuForm()	Конструктор	Ініціалізує головне вікно з вкладками для різних розділів системи (освітня, викладачі, дисципліни, групи, оцінки, загальні дії).
createPanelWithButtons()	JPanel	Створює панель з кнопками для певної вкладки, кожна кнопка має свою дію, що викликає відповідну форму.
styleButton()	void	Налаштовує стиль для кнопок, встановлюючи шрифт, кольори та інші параметри.
Методи для виклику різних форм	void	Кожен метод відкриває відповідну форму для роботи з даними (наприклад, openStudentForm() для відкриття форми студентів, addStudentForm() для додавання здобувачів).

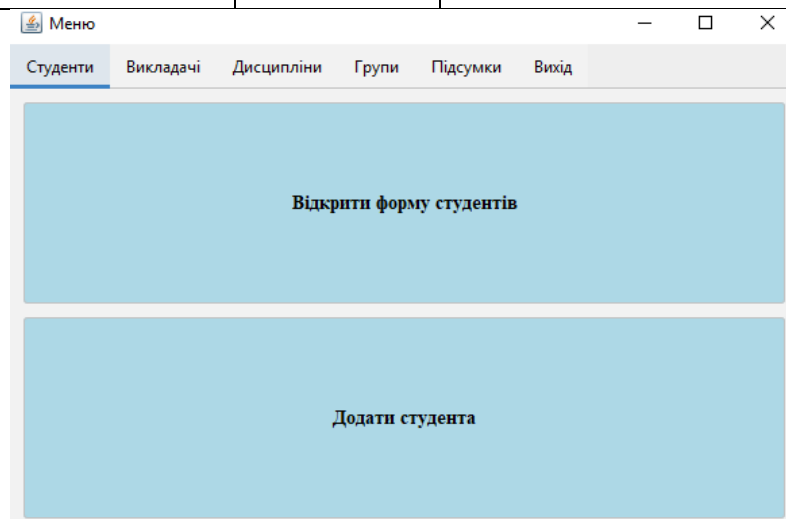


Рис. 3.6 – Форма меню системи

Опис класу RegistrationForm

Загальний опис

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

Клас `RegistrationForm` відповідає за створення форми для реєстрації нового користувача в системі. Форма включає поля для введення імені, прізвища, електронної пошти, пароля, телефону та дати народження. Після заповнення форми та натискання кнопки «Зареєструватися», відбувається перевірка введених даних, хешування пароля та відправка даних на сервер для реєстрації користувача.

Таблиця 3.39.

Основні компоненти класу

Метод	Тип повернення	Опис
<code>RegistrationForm(String host, int port)</code>	Конструктор	Ініціалізує форму для реєстрації нового користувача, включаючи поля для введення даних та кнопку для відправки інформації на сервер.
<code>hashPassword(String password)</code>	String	Хешує пароль користувача за допомогою алгоритму SHA-256 для забезпечення безпеки пароля перед відправкою на сервер.

Алгоритм роботи

Ініціалізація інтерфейсу

У конструкторі класу створюється форма з полями для введення даних користувача (ім'я, прізвище, email, пароль, телефон, дата народження) та кнопка для відправки цих даних на сервер.

Валідація введених даних

Після натискання кнопки «Зареєструватися» перевіряється, чи заповнені всі поля. Якщо хоча б одне поле порожнє, з'являється повідомлення про помилку. Також перевіряється формат email за допомогою регулярного виразу.

Хешування пароля

Для забезпечення безпеки пароля, перед відправкою на сервер, він хешується за допомогою алгоритму SHA-256.

Відправка даних на сервер

За допомогою сокетів здійснюється з'єднання з сервером. Дані про користувача (ім'я, прізвище, email, хешований пароль, телефон, дата народження) відправляються на сервер у вигляді рядка. Сервер відповідає результатом реєстрації, який відображається користувачу.

Методика взаємодії з сервером

Дані відправляються на сервер за допомогою сокетів через TCP-з'єднання. Клієнт надсилає запит на реєстрацію у форматі «REGISTER;name;surname;email;hashedPassword;phone;birthdate», а сервер відповідає повідомленням про успіх або помилку.

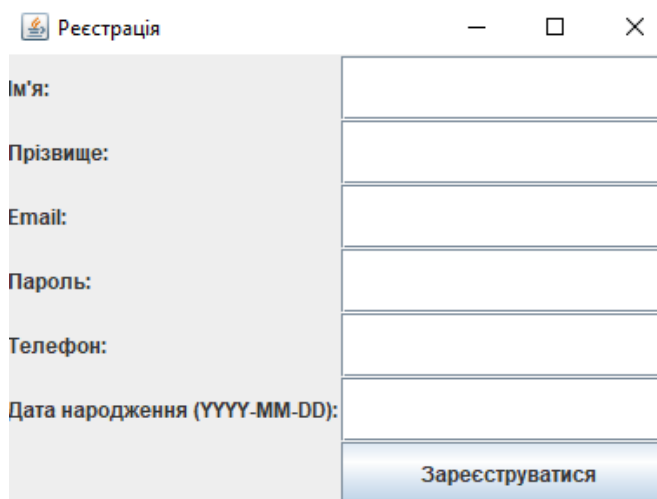


Рис. 3.7 – Форма реєстрації в системі

Опис класу ServerConnection

Загальний опис

Клас ServerConnection відповідає за ініціалізацію з'єднання з сервером та управління сокетами для обміну даними між клієнтом та сервером. Цей клас дозволяє підключитися до сервера, отримати потоки вводу/виводу для обміну даними, а також закрити з'єднання.

Таблиця 3.40.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		73

Основні компоненти класу

Метод	Тип повернення	Опис
initialize(String serverHost, int serverPort)	void	Ініціалізує параметри сервера (адресу та порт), якщо вони ще не були ініціалізовані.
getSocket()	Socket	Отримує активне з'єднання до сервера або створює нове, якщо з'єднання ще не існує.
getOut()	PrintWriter	Отримує об'єкт для виведення даних на сервер (потік виводу).
getIn()	BufferedReader	Отримує об'єкт для вводу даних від сервера (потік вводу).
closeConnection()	void	Закриває з'єднання з сервером, якщо воно відкрите.

Алгоритм роботи

Ініціалізація параметрів сервера

Метод `initialize()` використовується для налаштування параметрів сервера (хосту та порту). Якщо сервер вже ініціалізований, нові параметри не встановлюються, і виводиться повідомлення про це.

Підключення до сервера

Метод `getSocket()` відповідає за підключення до сервера. Якщо з'єднання вже існує, воно використовується повторно. Якщо ж сокет ще не створено або закрито, здійснюється підключення до сервера через задану адресу та порт.

Отримання потоку виводу

Метод `getOut()` повертає об'єкт `PrintWriter`, який використовується для відправки даних на сервер. Потік створюється при підключенні до сервера.

Отримання потоку вводу

Метод `getIn()` повертає об'єкт `BufferedReader`, який використовується для отримання даних від сервера. Потік також створюється при підключен-

ні.

Закриття з'єднання

Метод `closeConnection()` закриває сокет та звільняє ресурси, якщо з'єднання більше не потрібне.

Методика взаємодії з сервером

Клас взаємодіє з сервером через сокети. Після отримання активного з'єднання з сервером можна відправляти дані через `PrintWriter` та отримувати відповіді через `BufferedReader`. Підключення здійснюється один раз за допомогою методу `initialize()`, а після цього використовується для обміну даними через відповідні потоки.

Опис класу Sform

Загальний опис

Клас `Sform` є графічним інтерфейсом для управління даними здобувачів освіти. Він забезпечує відображення таблиці здобувачів, а також дозволяє виконувати операції оновлення таблиці та видалення здобувачів через сервер. Всі операції здійснюються через сокети, і дані передаються у форматі JSON. Таблиця Основні компоненти класу (див. ДОДАТОК Б табл.Б.6.).

Алгоритм роботи

Ініціалізація форми

Конструктор `Sform()` налаштовує інтерфейс користувача, створюючи таблицю для відображення даних студентів і додаючи кнопки для оновлення таблиці та видалення здобувачів. Прогрес-бар встановлюється невидимим і відображається лише під час завантаження даних.

Отримання даних зі сервера

Метод `fetchDataFromServer()` виконується в окремому потоці через `SwingWorker` і надсилає запит «GET_STUDENTS» на сервер. Відповідь, що містить дані здобувачів у форматі JSON, обробляється і виводиться в таблицю.

					<i>ITC.4KI.0124.03-ПЗ</i>	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

Оновлення таблиці

Коли дані про студентів отримано, таблиця очищається, і в неї додаються нові рядки з інформацією, отриманою з сервера. Прогрес-бар ховається після завершення завантаження.

Видалення здобувача

Метод `sendDeleteRequest()` дозволяє видалити здобувача з бази даних за допомогою запиту «DELETE_STUDENT». Після успішного видалення таблиця оновлюється, щоб відобразити зміни.

Методика взаємодії з сервером

Комунікація з сервером відбувається через сокети, використовуючи протокол текстових повідомлень. Для отримання даних використовуються запити типу «GET_STUDENTS», а для видалення освітян — «DELETE_STUDENT». Сервер повертає дані у форматі JSON, що дозволяє надійно обробляти інформацію про здобувачів.

Опис класу Stgradd

Загальний опис

Клас Stgradd є графічним інтерфейсом для відсортування здобувачів освіти за групами. Він забезпечує можливість вибору здобувача освіти та групи зі списків, а також додає здобувача освіти до вибраної групи. Інтерфейс використовує сокетні з'єднання для комунікації з сервером, отримуючи дані здобувачів освіти, груп і інформацію про вже відсортованих здобувачів освіти.

Алгоритм роботи

Ініціалізація форми

Конструктор `Stgradd()` налаштовує інтерфейс користувача, створюючи таблицю для відображення здобувачів освіти, випадючі списки для вибору здобувача освіти та групи, а також кнопку для додавання здобувача освіти до групи.

Прогрес-бар використовується для візуалізації завантаження даних.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		76

Отримання даних зі сервера

Метод `fetchStudentDataFromServer()` отримує список здобувачів освіти, `fetchGroupDataFromServer()` — список груп, а `fetchCoursesFromServer()` — інформацію про здобувачів освіти, що вже належать до груп. Ці дані отримуються через сокетні з'єднання з сервером у форматі JSON.

Вибір здобувача освіти та групи

Користувач вибирає здобувача освіти і групу з відповідних списків. При натисканні кнопки «Відсортувати», виконується запит до сервера для додавання здобувача освіти до вибраної групи.

Оновлення таблиці

Таблиця оновлюється після успішного додавання здобувача освіти до групи або після отримання нових даних від сервера. У таблиці відображаються імена здобувачів освіти та назви груп, до яких вони належать.

Методика взаємодії з сервером

Взаємодія з сервером здійснюється через сокети. Запити на сервер можуть бути такими:

- a) `GET_STUDENTS` — отримати список здобувачів освіти.
- b) `GET_GROUP` — отримати список груп.
- c) `GET_STUDGROUP` — отримати дані про здобувачів освіти і групи.
- d) `ADD_STUDENT_TO_GROUP` — додати здобувача освіти до групи.

Сервер відповідає у форматі JSON, який обробляється і відображається в інтерфейсі користувача.

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77

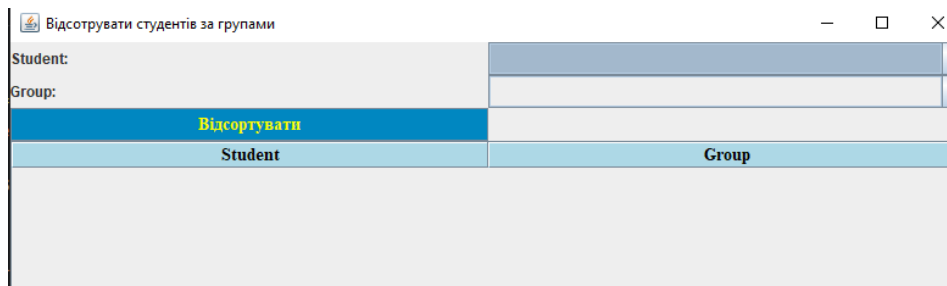


Рис. 3.8 – Форма відсортування здобувачів за академічними групами

Опис класу StGrForm

Загальний опис

Клас StGrForm є графічним інтерфейсом для відображення здобувачів освіти за групами. Він надає можливість вибору здобувача освіти і групи, а також дає можливість видаляти здобувачів освіти з груп. Клас використовує сокетні з'єднання для комунікації з сервером для отримання та оновлення даних.

Таблиця 3.41.

Основні компоненти класу

Метод	Тип повернення	Опис
studentIDs	ArrayList <Integer>	Список ідентифікаторів здобувачів освіти, отриманих з сервера.
groupIDs	ArrayList <Integer>	Список ідентифікаторів груп, отриманих з сервера.

Алгоритм роботи

Ініціалізація форми

Конструктор StGrForm() налаштовує інтерфейс користувача, створюючи таблицю для відображення здобувачів освіти та груп, додаткові кнопки для оновлення таблиці та видалення здобувача освіти з групи. Прогрес-бар використовується для індикації процесу завантаження даних.

Отримання даних зі сервера

Метод fetchFromServer() отримує список здобувачів освіти, груп та інформацію про вже відсортованих здобувачів освіти за групами через

сокетні з'єднання у форматі JSON.

Видалення здобувача освіти з групи

Користувач може вибрати здобувача освіти у таблиці і натиснути кнопку для видалення здобувача освіти з групи. Запит на видалення відправляється на сервер через сокетне з'єднання.

Оновлення таблиці

Після видалення здобувача освіти з групи або отримання нових даних, таблиця оновлюється з новою інформацією про здобувачів освіти та їх групи.

Методика взаємодії з сервером

Запити на сервер для взаємодії з даними можуть бути наступними:

- a) GET_STUDGROUP — отримати список здобувачів освіти та їх груп.
- b) DELETE_STUDENT_FROM_GROUP — видалити здобувача освіти з групи.

Сервер відповідає у форматі JSON, який обробляється для відображення результатів на інтерфейсі.

Опис класу StudentDetailsForm

Загальний опис

Клас StudentDetailsForm є графічним інтерфейсом для додавання нового здобувача освіти до системи та перегляду списку існуючих здобувачів освіти. Клас використовує сокетне з'єднання для комунікації з сервером, зокрема для надсилання запитів на додавання студента та отримання списку студентів. Прогрес-бар показує статус завантаження даних або виконання запитів.

Алгоритм роботи

Ініціалізація форми

Конструктор StudentDetailsForm() налаштовує інтерфейс користувача, додаючи таблицю для відображення даних про здобувачів освіти, а також форму для введення нового студента.

					ITC.4KI.0124.03-ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

Прогрес-бар спочатку невидимий і з'являється під час виконання запитів до сервера.

Додавання студента

Після заповнення форми і натискання кнопки «Додати студента», дані з полів форми відправляються на сервер через сокетне з'єднання за допомогою методу `sendAddStudentRequest()`. Запит на сервер форматується як рядок і відправляється за допомогою методу `PrintWriter`.

Отримання даних про студентів

Метод `fetchDataFromServer()` надсилає запит на сервер для отримання списку здобувачів освіти у форматі JSON. Отримані дані перетворюються у формат, зручний для відображення в таблиці.

Оновлення таблиці

Після отримання відповіді від сервера або додавання нового студента, таблиця оновлюється з новими даними.

Методика взаємодії з сервером

Запити на сервер для взаємодії з даними можуть бути наступними:

- а) `ADD_STUDENT` — додати нового студента.
- б) `GET_STUDENTS` — отримати список здобувачів освіти.

Сервер відповідає у форматі JSON, який обробляється для відображення результатів на інтерфейсі.

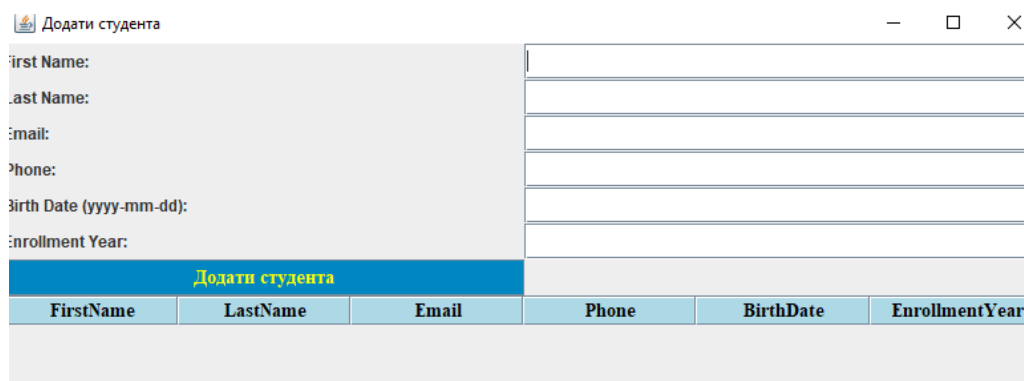


Рис. 3.9 – Форма для додавання даних про здобувачів освіти в систему

Опис класу TableStyle

Загальний опис

Клас TableStyle відповідає за налаштування стилю таблиці та кнопок у графічному інтерфейсі. Це дозволяє забезпечити гармонійний вигляд таблиць із даними та кнопок для взаємодії з користувачем. В ньому реалізовані методи для налаштування фону таблиці, сітки, вирівнювання тексту, а також стилізації кнопок [23].

Таблиця 3.42.

Основні компоненти класу

Метод	Тип повернення	Опис
applyTableStyle(JTable table)	void	Налаштовує стилі для таблиці: фон, сітка, заголовки та вирівнювання тексту.
styleButton(JButton button)	void	Налаштовує стилі для кнопок: шрифт, фон, колір тексту та відсутність фокусу.

Алгоритм роботи

Налаштування таблиці

Метод *applyTableStyle(JTable table)* налаштовує:

- Фон таблиці: задається світлий фон для таблиці.
- Сітка таблиці: включається відображення сітки з чорним кольором.
- Заголовки таблиці: встановлюється світло-блакитний фон, чорний колір тексту та жирний шрифт для заголовків.
- Вирівнювання тексту: усі клітинки таблиці будуть відображатися з вирівнюванням тексту по центру.
- Чередування кольорів рядків: для парних і непарних рядків таблиці встановлюються різні фони — білий та світло-блакитний.
- Вибраний рядок: фон вибраного рядка змінюється на світло-блакитний, а текст — на чорний.

Налаштування кнопок

Метод `styleButton(JButton button)` налаштовує:

- a) Шрифт: задається жирний шрифт «Times New Roman» розміру 14.
- b) Фон кнопки: встановлюється темно-синій фон.
- c) Колір тексту кнопки: текст на кнопці буде жовтого кольору.
- d) Фокусування: вимикається відображення фокусу при натисканні на кнопку.

Методика взаємодії з сервером

Цей клас не взаємодіє безпосередньо з сервером, але забезпечує естетичний вигляд компонентів інтерфейсу, що полегшує взаємодію з користувачем.

Опис класу `TeachersDetailsForm`

Загальний опис

Клас `TeachersDetailsForm` створює графічний інтерфейс для додавання викладачів до системи. Він включає таблицю для відображення списку викладачів, форму для введення нових даних про викладачів, а також прогрес-бар для індикації процесу завантаження даних або додавання інформації. Взаємодія з сервером здійснюється через сокети, що дозволяє отримувати та надсилати дані.

Алгоритм роботи

Конструктор `TeachersDetailsForm()`:

- a) Налаштовує інтерфейс користувача з таблицею для відображення викладачів, формою для введення нових даних та прогрес-баром для індикації завантаження.
- b) Кнопка «Додати викладача» ініціює відправку введених даних на сервер.

Метод `sendAddTeacherstRequest(...)`:

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		82

- a) Створює запит для додавання викладача через сокет-з'єднання з сервером.
- b) Оновлює таблицю після отримання відповіді від сервера.

Метод `fetchDataFromServer()`:

- a) Надсилає запит на сервер для отримання списку викладачів.
- b) Отримує відповідь у форматі JSON, яку перетворює в об'єкти та відображає в таблиці.

Методика взаємодії з сервером

- a) Для взаємодії з сервером використовується сокетне з'єднання.
- b) У методах `sendAddTeacherstRequest` та `fetchDataFromServer` створюються сокетні з'єднання з сервером, після чого надсилаються запити («ADD_TEACHERS», «GET_TEACHERS»), і отримується відповідь.
- c) Відповідь сервера обробляється: у разі отримання даних — вони додаються в таблицю, у разі помилки — виводиться повідомлення про помилку.

Таблиця 3.43.

Основні компоненти класу

Метод	Тип повернення	Опис
<code>TeachersDetailsForm()</code>	Конструктор	Конструктор, який ініціалізує інтерфейс форми для додавання викладача.
<code>sendAddTeacherstRequest(...)</code>	void	Надсилає запит на сервер для додавання викладача до бази даних.
<code>fetchDataFromServer()</code>	void	Завантажує список викладачів з сервера та відображає його в таблиці.

Рис.3.10 – Форма для додавання даних про викладачів в систему

Опис класу Tform

Загальний опис

Клас Tform створює графічний інтерфейс для управління викладачами в системі. Він включає таблицю для відображення списку викладачів, кнопки для оновлення даних та видалення викладача з бази даних. Взаємодія з сервером здійснюється через сокети для запитів щодо викладачів.

Таблиця 3.44.

Основні компоненти класу

Метод	Тип повернення	Опис
Tform()	Конструктор	Конструктор, що ініціалізує інтерфейс для роботи з викладачами.
fetchDataFromServer()	void	Завантажує дані викладачів із сервера та відображає їх у таблиці.
sendDeleteRequest(String)	void	Надсилає запит на сервер для видалення викладача за його email.

Алгоритм роботи

Конструктор Tform():

- Ініціалізує вікно з таблицею для відображення викладачів, кнопками для оновлення даних та видалення викладачів.
- При натисканні кнопки «Оновити таблицю» відправляється запит для завантаження нових даних викладачів.

- с) При натисканні кнопки «Видалити викладача» видалається вибраний викладач із системи за допомогою його email.

Метод `fetchDataFromServer()`

- а) Відправляє запит на сервер для отримання списку викладачів.
- б) Отримує відповідь у форматі JSON, яку перетворює в об'єкти та відображає у таблиці.

Метод `sendDeleteRequest(String)`:

- а) Відправляє запит на сервер для видалення викладача, використовуючи його email.
- б) Після успішного видалення перезавантажує таблицю з оновленими даними.

Методика взаємодії з сервером

- а) Для взаємодії з сервером використовується сокетне з'єднання.
- б) У методах `fetchDataFromServer` та `sendDeleteRequest` створюються сокетні з'єднання для надсилання запитів на сервер (запит на отримання викладачів: «GET_TEACHERS»; запит на видалення викладача: «DELETE_TEACHERS;email»).

Додатково були розроблені, два універсальні класи захисту паролів які працюють як на серверній так і на клієнтській частинах. Це класи `SecureEncryption` який потім зібраний в jar файл та в подальшому використовується як бібліотека в проєкті, та клас `ConfigReader`.

Опис класу `SecureEncryption`

Загальний опис

Клас `SecureEncryption` забезпечує функціональність симетричного шифрування та дешифрування текстових даних з використанням алгоритму AES у режимі CBC. Ключ генерується на основі пароля користувача та випадкової «солі» за допомогою алгоритму `PBKDF2WithHmacSHA256`.

					<i>ITC.4KI.0124.03-ПЗ</i>	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

Клас забезпечує надійний захист даних за допомогою сучасних криптографічних стандартів. Таблиця Основні компоненти класу (див. ДОДАТОК Б табл.Б.7.).

Алгоритм роботи

Конструктор SecureEncryption()

Ініціалізує об'єкт класу без жодних параметрів. У даній реалізації не виконує додаткових дій.

Метод generateKeyFromPassword(String password, byte[] salt)

- a) Використовує алгоритм PBKDF2WithHmacSHA256 для генерації ключа.
- b) На основі пароля та солі створюється 256-бітний ключ AES.

Метод encryptAES(String data, String password)

- a) Генерується випадкова сіль (16 байт) і IV (ініціалізаційний вектор, 16 байт).
- b) Генерується ключ на основі пароля та солі.
- c) Дані шифруються з використанням AES у режимі CBC.
- d) У результаті формується один масив із солі + IV + зашифрованих даних, який кодується у формат Base64 для передачі або збереження.

Метод decryptAES(String encryptedData, String password)

- a) Декодує дані з Base64 у байтовий масив.
- b) Виділяє сіль, IV та зашифроване повідомлення.
- c) Генерує ключ за тією ж методикою.
- d) Дешифрує дані та повертає оригінальний текст.

Методика взаємодії з даними

- a) Шифрування відбувається локально, без взаємодії з сервером або мережею.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

- б) Клас не залежить від зовнішніх систем і може використовуватись у будь-яких клієнт-серверних або автономних системах для забезпечення захисту даних (наприклад, паролів).

Опис класу ConfigReader

Загальний опис

Клас ConfigReader відповідає за зчитування конфігураційних параметрів із зовнішнього файлу config.properties. Основною метою є отримання значення головного криптографічного ключа MASTER_KEY, який використовується у системі для шифрування або інших захищених операцій.

Таблиця 3.45.

Основні компоненти класу

Метод	Тип повернення	Опис
getMasterKey()	String	Зчитує значення MASTER_KEY із конфігураційного файлу та повертає його. У разі помилки повертає null

Алгоритм роботи

Метод getMasterKey()

- Створюється об'єкт Properties для зчитування параметрів.
- Відкривається файл config.properties, шлях до якого вказано у змінній CONFIG_FILE.
- Завантажуються властивості з файлу.
- Проводиться пошук параметра MASTER_KEY.
- Якщо параметр знайдено — метод повертає його значення.
- Якщо параметра не знайдено або виникла помилка зчитування файлу — виводиться повідомлення про помилку і повертається null.

Методика взаємодії з системою

- а) Клас працює автономно та не взаємодіє з мережею чи сервером.
- б) Основна роль — забезпечити централізоване зчитування конфігураційного параметра (MASTER_KEY), що використовується іншими класами системи, наприклад, у класі SecureEncryption.
- в) Залежність: клас очікує, що файл config.properties міститиме ключ MASTER_KEY у форматі: MASTER_KEY=ваш_ключ.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 3

В третьому розділі було здійснено повну реалізацію клієнт-серверної інформаційної системи «Електронний деканат», яка охоплює всі основні аспекти розробки програмного забезпечення: серверну частину, клієнтську частину та їхню взаємодію.

По-перше, реалізовано серверну частину системи на мові Java з використанням архітектури ТСП-сервер – клієнт. Сервер забезпечує прийом і обробку запитів, а також взаємодіє з базами даних для збереження, оновлення та вилучення інформації. Розроблений графічний інтерфейс адміністратора дозволяє зручно керувати роботою сервера та переглядати активні підключення користувачів.

По-друге, розроблена клієнтська частина реалізована у вигляді десктопного додатку з дружнім графічним інтерфейсом, що дозволяє користувачам реєструватися, входити до системи, переглядати та редагувати інформацію про студентів, викладачів, дисципліни, групи та оцінки.

По-третє, реалізовано ефективну взаємодію між клієнтом і сервером за допомогою протоколу TCP та обміну командами у форматі текстових запитів або JSON. Усі компоненти системи забезпечують логічну цілісність, узгодженість даних і стабільну роботу в умовах багатокористувацького середовища.

Таким чином, реалізація системи «Електронний деканат» демонструє можливості побудови сучасного програмного забезпечення для автоматизації навчального процесу з використанням Java та баз даних MySQL, забезпечуючи ефективне управління освітнім процесом.

ВИСНОВКИ

В ході виконання даної бакалаврської роботи було здійснено розробку клієнт-серверної системи «Електронний деканат» засобами Java, що забезпечує автоматизацію управління академічним процесом у закладах вищої освіти.

Перший розділ містив аналіз предметної області, визначення вимог до системи та вибір технологій для її реалізації. Було розглянуто основні проблеми існуючих систем управління обліком здобувачів освіти та запропоновано підхід для їх усунення шляхом створення централізованої інформаційної системи на основі клієнт-серверної архітектури. Було обґрунтовано вибір мови програмування, бази даних та фреймворків, які забезпечують надійну та ефективну роботу системи.

Другий розділ був присвячений процесу проектування системи, включаючи визначення архітектури клієнт-серверної взаємодії, структури бази даних, проектування серверної та клієнтської частин. Було розроблено UML-діаграми для відображення зв'язків між основними компонентами системи, а також описано механізми автентифікації користувачів, обробки запитів та управління даними.

У третьому розділі розглянуто безпосередню реалізацію системи. Детально описано структуру серверної та клієнтської частин, механізм обробки запитів клієнта, взаємодію з базою даних та реалізацію заходів безпеки. Була впроваджена система авторизації користувачів, що гарантує безпечний доступ до інформації, а також оптимізовано запити до бази даних для підвищення продуктивності системи.

					ІТС.4КІ.0124.03-ПЗ					
Змн.	Арк.	№ докум.	Підпис	Дата	ВИСНОВКИ		Лім.	Арк.	Акрушів	
Розроб.		Курячий О.В.							90	2
Керівник		Переяславська С.О.								
Реценз.		Козуб Ю.Г.								
Н. Контр.										
Зав. каф.		Семенов М.А.								
					ЛНУ Кафедра ІТС, Гр.4КІ					

В результаті виконаної роботи була створена повнофункціональна клієнт-серверна система «Електронний деканат», яка дозволяє автоматизувати управління академічними процесами, зменшити адміністративне навантаження та підвищити ефективність роботи викладачів, здобувачів освіти та адміністрації навчального закладу. Реалізована система відповідає сучасним вимогам інформаційної безпеки та зручності використання, що робить її придатною для впровадження у реальних навчальних закладах.

					ІТС.4КІ.0124.03-ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JavaRush. Класи Socket та ServerSocket, або Алло, сервер, ти мене чуєш [Електронний ресурс]. – Режим доступу: <https://javarush.com/ua/groups/posts/uk.654.klasi-socket-ta-serversocket-abo-allo-server-ti-mene-chush>.
2. Блох Дж. Effective Java. 3rd ed. Boston: Addison-Wesley, 2018. 416 с.
3. Гетц Б. Java. Паралельне програмування в реальному житті. Київ: Діалектика, 2012. 512 с.
4. Фрімен Е., Робсон Е. Шаблони проектування. Head First. Київ: Діалектика, 2016. 688 с.
5. Шилдт Г. Java. Повний довідник. 11-е вид. Київ: Діалектика, 2022. 1344 с.
6. DuBois P. MySQL Cookbook. 3rd ed. Sebastopol: O'Reilly Media, 2014. 866 p.
7. Bell C., Kindahl M., Thalmann L. MySQL High Availability. Sebastopol: O'Reilly, 2014. 576 p.
8. Oaks S. Java Performance: The Definitive Guide. O'Reilly Media, 2014. 426 p.
9. Bauer C., King G. Hibernate in Action. Manning Publications, 2005. 400 p.
10. Walls C. Spring in Action. 6th ed. Manning Publications, 2018. 520 p.
11. Vos J. Pro JavaFX 9. Apress, 2017. 600 p.
12. MySQL Reference Manual [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc/>.
13. MySQL Connector/J Developer Guide [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc/connector-j/en/>.

					ІТС.4КІ.0124.03-ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Курячий О.В.			СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ		
Керівник		Переяславська С.О.					
Реценз.		Козуб Ю.Г.					
Н. Контр.							
Зав. каф.		Семенов М.А.					
					Лім.	Арк.	Акрушів
						92	3
					ЛНУ Кафедра ІТС, Гр.4КІ		

14. Java SE 8 API Specification [Електронний ресурс]. – Режим доступу:
<https://docs.oracle.com/javase/8/docs/api/>.
15. Java Swing Tutorial – Oracle [Електронний ресурс]. – Режим доступу:
<https://docs.oracle.com/javase/tutorial/uiswing/>.
16. Java Security Overview – Oracle [Електронний ресурс]. – Режим доступу:
<https://docs.oracle.com/javase/8/docs/technotes/guides/security/>.
17. Java Networking and Proxies – Oracle [Електронний ресурс]. – Режим доступу:
<https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html>.
18. NIST. Data Security and Privacy [Електронний ресурс]. – Режим доступу:
<https://www.nist.gov/topics/data-security>.
19. OWASP Foundation. Web Application Security Guide [Електронний ресурс]. – Режим доступу:
<https://owasp.org>.
20. Client-server architecture basics [Електронний ресурс]. – Режим доступу:
<https://training.qatestlab.com/blog/technical-articles/client-server-architecture/>.
21. Maven – Dependency Management. Apache Maven Project [Електронний ресурс]. – Режим доступу:
<https://maven.apache.org/>.
22. Educational Management Information Systems (EMIS). UNESCO [Електронний ресурс]. – Режим доступу:
<https://uis.unesco.org/en/topic/education-management-information-systems>.
23. TableStyle – Оформлення таблиць у Java GUI [Електронний ресурс]. – Режим доступу:
<https://stackoverflow.com/questions/38143306/java-swing-jdbc-with-mysql-client-server-lan-setup>.

24. Конференція «Проблеми інформатизації» / ЧДТУ, 2018 [Електронний ресурс]. – Режим доступу: https://er.chdtu.edu.ua/bitstream/ChSTU/4508/1/Conference_NTU_KhPI_2018_Problemy_informatyzatsii.pdf.
25. Пояснювальна записка: особливості СУБД MySQL / НУ «Запорізька політехніка» [Електронний ресурс]. – Режим доступу: https://eir.zp.edu.ua/bitstream/123456789/9073/1/MR_Kondratenko.pdf.
26. Додаткові роз'яснення щодо вибору навчальних дисциплін [Електронний ресурс]. – Режим доступу: https://ipz.khmnpu.edu.ua/wp-content/uploads/sites/48/dodatkovy-rozjasnyvalni-materialy-dlya-vyboru-dysczyplin_fit_2023_2024_n_r_fin-2.pdf.
27. Open Source Software Conference Materials. ХНЕУ [Електронний ресурс]. – Режим доступу: <https://repository.hneu.edu.ua/bitstream/123456789/35624/1/foss-2025-theses.pdf>.
28. Sierra K., Bates B. Head First Java. 2nd ed. O'Reilly Media, 2005. 688 p.
29. Java Concurrency Utilities – Oracle [Електронний ресурс]. – Режим доступу: <https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html>.
30. OWASP Secure Coding Practices [Електронний ресурс]. – Режим доступу: <https://owasp.org/www-project-secure-coding-practices/>.
31. Java Database Connectivity (JDBC) Tutorial – Oracle [Електронний ресурс]. – Режим доступу: <https://docs.oracle.com/javase/tutorial/jdbc/>.

ДОДАТОК А Діаграми класів серверної та клієнтської частин



ДОДАТКИ

Літ.		Арк.		Акрушіє	
		95		0	
<p style="text-align: center;">ЛНУ Кафедра ІТС, Гр.4КІ</p>					

ДОДАТОК Б Таблиці серверної та клієнтської частин

Таблиця Б.1.

Класи серверної частини системи «Електронний деканат»

Клас	Призначення
ServerWithGUI	Головний клас серверної частини. Запускає сервер, забезпечує графічний інтерфейс для адміністратора, управляє сесіями.
ClientHandler	Обробка підключень від клієнтів у окремих потоках, взаємодія із запитами клієнтів.
db_dekanat	Здійснює підключення до бази даних, обробляє основні SQL-запити.
db_log	Ведення журналу дій користувачів, логування системних подій.
CoursesManager	Управління курсами: додавання, редагування, видалення.
GradesManager	Управління оцінками: CRUD-операції.
TeachersManager	Управління інформацією про викладачів.
StudentGroupsManager	Управління даними про зв'язки студентів з групами.
StGrManager	Управління взаємозв'язками студент-група.
CoursesTable	Таблиця для відображення і структурування даних про курси.
GradesTable	Таблиця для представлення оцінок.
groupTable	Таблиця для управління групами.
StudentGroupsTable	Таблиця для зв'язків студентів з групами.
TeachersTable	Таблиця для представлення даних викладачів.
StudentTable	Таблиця студентів системи.
StudentAdder	Клас для додавання нового студента.

Продовження таблиці Таблиця Б.1.

GroupAdder	Додавання нової групи в базу даних.
TeachersAdder	Додавання нового викладача.
Courses	Модельний клас для об'єктів «курс».
CoursesDeleter	Видалення записів про курси.
GradesDeleter	Видалення оцінок із бази.
GroupDeleter	Видалення груп.
Groups	Модельний клас для груп.
StudentDeleter	Видалення студентів із бази даних.
Students	Модельний клас для студентів.
Teacher	Модельний клас для викладачів.
TeachersDeleter	Видалення даних про викладачів.
StGrDeleter	Видалення зв'язків студент-група.

Таблиця Б.2.

Таблиця: Додавання даних (ADD)

Команда	Формат запиту	Опис дії
ADD_COURSE	ADD_COURSE;Назва;Опис; Кредити; ID/Ім'я_викладача	Додає нову дисципліну в систему
ADD_STUDENT	ADD_STUDENT;Ім'я;Прізвище; Email;Телефон; Дата_народження;Рік_вступу	Додає нового здобувача освіти
ADD_GROUP	ADD_GROUP;Назва_групи; Спеціальність; Рік_вступу	Додає нову академічну групу
ADD_TEACHERS	ADD_TEACHERS;Ім'я; Прізвище;Email; Телефон;Кафедра	Додає викладача
ADD_STUDENT_TO_GROUP	ADD_STUDENT_TO_GROUP; ID_студента/Ім'я; ID_групи/Назва	Призначає здобувача освіти до відповідної групи

Продовження таблиці Таблиця Б.2.

ADD_GRADES	ADD_GRADES;ID_студента/Ім'я; ID_курсу/Назва;Оцінка (0–100)	Додає підсумкову оцінку здобувачу освіти
------------	---	--

Таблиця Б.3.

Таблиця: Отримання даних (GET)

Команда	Формат запиту	Опис дії
GET_STUDENTS	GET_STUDENTS	Отримання списку здобувачів освіти
GET_GROUP	GET_GROUP	Отримання списку навчальних груп
GET_COURSES	GET_COURSES	Отримання переліку навчальних дисциплін
GET_STUDGROUP	GET_STUDGROUP	Отримання інформації про в якій групі навчається відповідний здобувач освіти
GET_GRADES	GET_GRADES	Отримання підсумкових балів здобувачів освіти
GET_TEACHERS	GET_TEACHERS	Отримання списку викладачів

Таблиця Б.4.

Класи клієнтської частини системи «Електронний деканат»

Клас	Призначення
Client	Головний клас клієнтської частини, що ініціалізує графічний інтерфейс і встановлює з'єднання з сервером.
Configuration	Клас для збереження та завантаження налаштувань підключення до сервера.
CourseDetailsForm	Форма для перегляду та редагування деталей конкретної дисципліни.

Продовження таблиці Таблиця Б.4.

CoursesForm	Інтерфейс для перегляду списку доступних дисциплін.
Gform	Форма для перегляду академічних груп
GradesAdd	Форма для виставлення підсумкових балів здобувачів освіти
GradesForm	Форма для перегляду підсумків здобувачів
GroupDetailsForm	Форма для додавання академічних груп в систему
MenuForm	Форма меню
RegistrationForm	Форма реєстрації в системі
ServerConnection	Клас в якому зберігається адреса та порт сервера для підключення
Sform	Форма для візбображення всіх здобувачів освіти в системі
Stgradd	Форма для розподілення здобувачів освіти за відповідними академічними групами
StGrForm	Форма для відображення освітян за розподіленими відповідними групами
StudentDetailsForm	Форма для додавання здобувачів освіти в систему
TableStyle	Клас зі стилями для таблиць та кнопок
TeachersDetailsForm	Форма для додавання викладачів освітньої установи в систему
Tform	Форма для візбображення всіх викладачів в системі

Таблиця Б.5.

Основні компоненти класу CourseDetailsForm

Метод	Тип повернення	Опис
CourseDetailsForm()	Конструктор	Ініціалізує графічний інтерфейс форми, створює таблицю дисциплін, форму для додавання нових дисциплін, прогрес-бар та випадаючий список викладачів.
sendAddCourseRequest(String courseName, String description, int credits, String teacherName)	void	Надсилає запит на сервер для додавання нової дисципліни. Після успішної відповіді оновлює таблицю дисциплін.
fetchTeachersDataFromServer()	void	Надсилає запит на отримання списку викладачів із сервера та заповнює комбінований список у формі.
fetchCoursesFromServer()	void	Надсилає запит на отримання списку дисциплін із сервера та відображає їх у таблиці.

Таблиця Б.6.

Основні компоненти класу Sform

Компонент	Тип	Опис
table	Jtable	Таблиця для відображення інформації про студентів (FirstName, LastName, Email, Phone, BirthDate, EnrollmentYear).
tableModel	DefaultTableModel	Модель для таблиці, яка дозволяє додавати та оновлювати дані таблиці.
progressBar	JprogressBar	Прогрес-бар для індикації завантаження даних з сервера.
Sform()	Конструктор	Ініціалізує форму, налаштовує таблицю та кнопки для управління даними здобувачів, а також запускає процес завантаження даних з сервера.

Продовження таблиці Таблиця Б.6.

Sform()	Конструктор	Ініціалізує форму, налаштовує таблицю та кнопки для управління даними здобувачів, а також запускає процес завантаження даних з сервера.
fetchDataFromServer()	void	Отримує дані про здобувачів з сервера за допомогою сокетів і оновлює таблицю за допомогою JSON формату.
sendDeleteRequest(String request)	void	Надсилає запит на сервер для видалення студента з таблиці на основі його email.

Таблиця Б.7.

Основні компоненти класу SecureEncryption

Метод	Тип повернення	Опис
SecureEncryption()	Конструктор	Порожній конструктор класу SecureEncryption.
generateKeyFromPassword(String, byte[])	SecretKey	Генерує криптографічний ключ на основі пароля користувача та «солі» з використанням PBKDF2.
encryptAES(String, String)	String	Шифрує текстові дані за допомогою AES (CBC/PKCS5Padding) та повертає результат у форматі Base64.
decryptAES(String, String)	String	Дешифрує текст, зашифрований методом encryptAES, використовуючи той самий пароль.

ДОДАТОК В Інфологічна та даталогічна моделі БД ElectronicDean та mysite

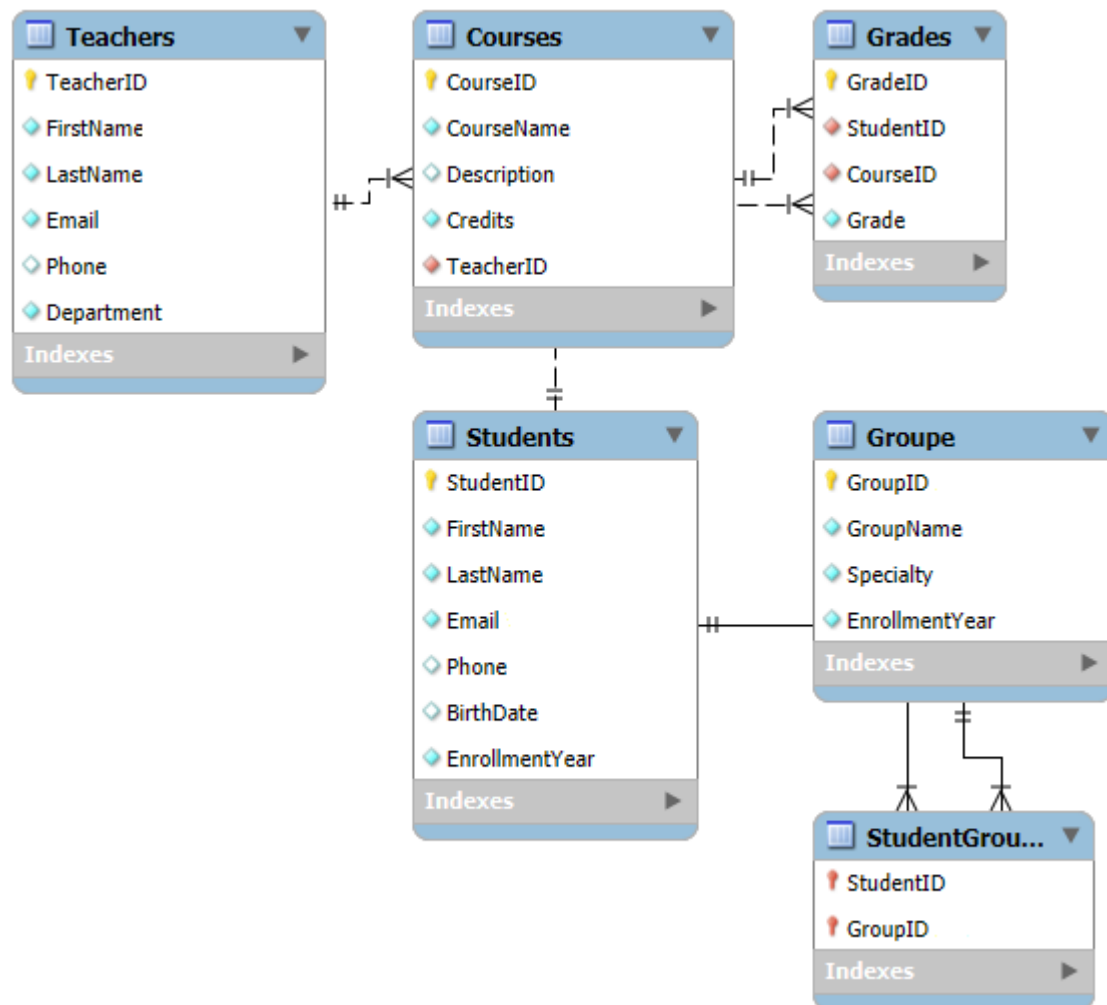


Рис. В.1. – Інфологічна модель БД ElectronicDean

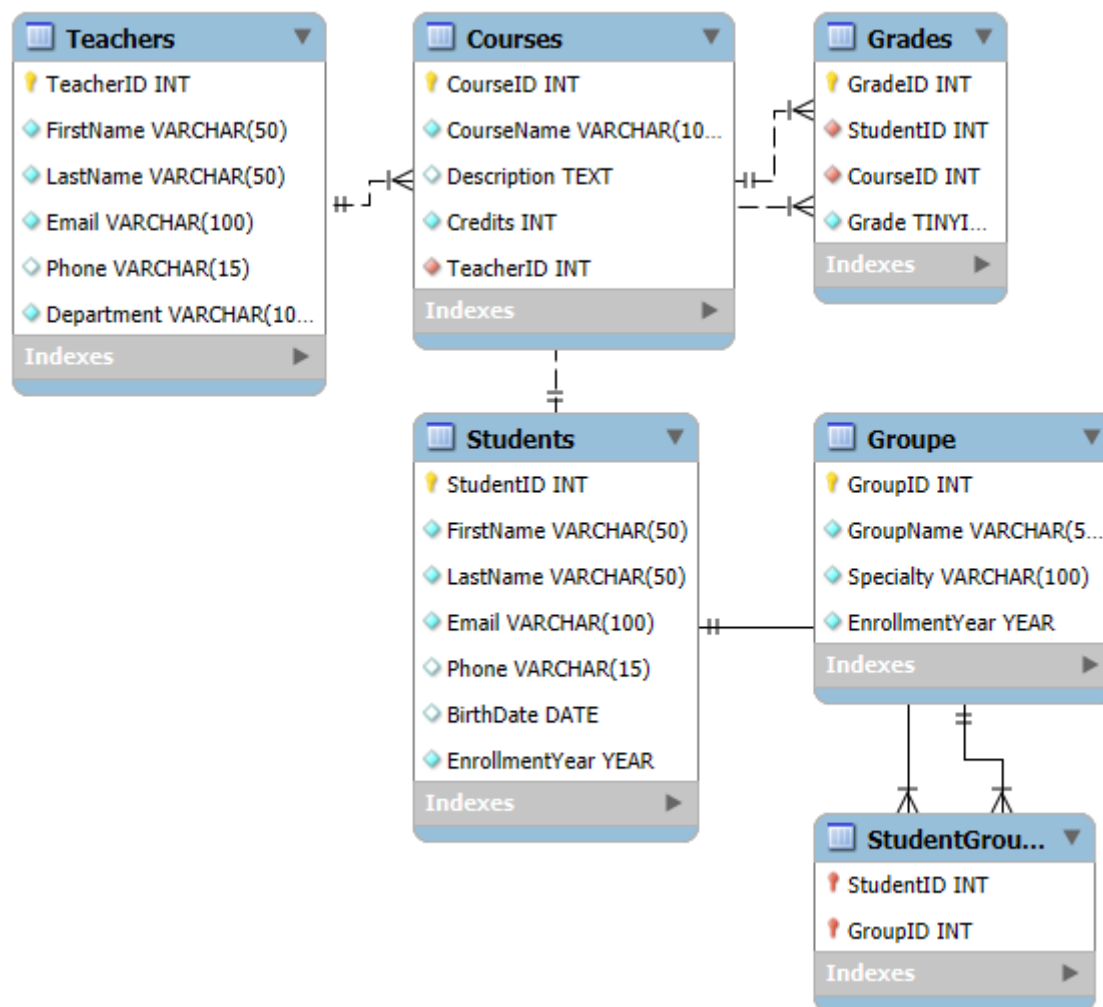


Рис. В.2. – Даталогічна модель БД ElectronicDean

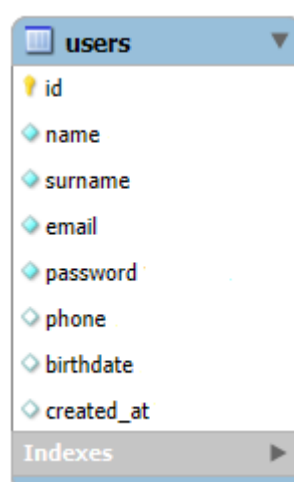


Рис. В.3. – Інфологічна модель БД mysite

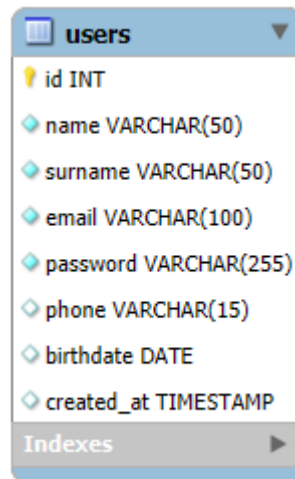


Рис. В.4. – Даталогічна модель БД mysite

ДОДАТОК Г Лістинг програми

Код класу ServerWithGUI

```
package org.example;
import org.json.JSONObject;
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.Arrays;
import java.util.concurrent.ConcurrentHashMap;
public class ServerWithGUI {
    private static ServerSocket serverSocket;
    private static boolean isRunning = false;
    private static Thread serverThread;
    // Створюємо клас для управління сесіями
    public static class SessionManager {
        private static final ConcurrentHashMap<String, Socket> activeSessions = new
        ConcurrentHashMap<>();
        public static boolean isUserLoggedIn(String email) {
            return activeSessions.containsKey(email);
        }
        public static void loginUser(String email, Socket socket) {
            activeSessions.put(email, socket);
        }
        public static void logoutUser(String email) {
            activeSessions.remove(email);
        }
    }
    public static void main(String[] args) {
        // Створюємо вікно
        JFrame frame = new JFrame("Сервер");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

					ITC.4KI.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		106

```

frame.setSize(600, 400);
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
frame.add(panel);
JLabel statusLabel = new JLabel("Сервер зупинено.");
statusLabel.setFont(new Font("Arial", Font.BOLD, 16));
panel.add(statusLabel, BorderLayout.NORTH);
JPanel buttonPanel = new JPanel();
JButton startButton = new JButton("Запустити сервер");
JButton stopButton = new JButton("Зупинити сервер");
stopButton.setEnabled(false); // Сервер ще не запущений, кнопка неактивна
buttonPanel.add(startButton);
buttonPanel.add(stopButton);
panel.add(buttonPanel, BorderLayout.SOUTH);

// Текстова область для відображення IP-адрес
JTextArea connectedClientsArea = new JTextArea();
connectedClientsArea.setEditable(false);
connectedClientsArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
JScrollPane scrollPane = new JScrollPane(connectedClientsArea);
panel.add(scrollPane, BorderLayout.CENTER);

// Обробник для кнопки "Старт"
startButton.addActionListener(e -> {
    if (!isRunning) {
        isRunning = true;
        startButton.setEnabled(false);
        stopButton.setEnabled(true);
        statusLabel.setText("Сервер запущено.");
        startServer(statusLabel, connectedClientsArea);
    }
});

// Обробник для кнопки "Стоп"
stopButton.addActionListener(e -> {

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						107
Змн.	Арк.	№ докум.	Підпис	Дата		


```

if (isRunning) {
    isRunning = false;
    startButton.setEnabled(true);
    stopButton.setEnabled(false);
    statusLabel.setText("Сервер зупинено.");
    stopServer();
}
});
frame.setVisible(true);
}

// Метод запуску сервера
private static void startServer(JLabel statusLabel, JTextArea connectedClientsArea) {
    serverThread = new Thread(() -> {
        try {
            serverSocket = new ServerSocket(12345);
            System.out.println("Сервер запущено на порту 12345");
            // Оновлюємо статус на сервері в текстовому полі
            SwingUtilities.invokeLater(() -> {
                connectedClientsArea.append("Сервер запущено на порту 12345\n");
            });
        } while (isRunning) {
            try {
                // Приймаємо новий клієнтський запит
                Socket clientSocket = serverSocket.accept();
                String clientIP = clientSocket.getInetAddress().getHostAddress();
                System.out.println("Новий клієнт підключений: " + clientIP);
            } // Додаємо повідомлення про новий підключений клієнт в текстове поле
            SwingUtilities.invokeLater(() -> {
                connectedClientsArea.append("Новий клієнт підключений: " + clientIP +
"\n");
            });
        } / Створюємо новий потік для обробки клієнта
        new Thread(new ClientHandler(clientSocket, connectedClientsArea)).start();
    });
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		108

```

    } catch (IOException e) {
        if (isRunning) {
            e.printStackTrace();
        }
    }
}

} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        if (serverSocket != null && !serverSocket.isClosed()) {
            serverSocket.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

});

serverThread.start();
}

// Метод зупинки сервера
private static void stopServer() {
    try {
        if (serverSocket != null && !serverSocket.isClosed()) {
            serverSocket.close();
        }
        if (serverThread != null) {
            serverThread.interrupt();
        }
    }

    System.out.println("Сервер зупинено.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						109
Змн.	Арк.	№ докум.	Підпис	Дата		

```
}  
}
```

Код класу ClientHandler

// Обробник клієнтів

```
class ClientHandler implements Runnable {  
    private Socket clientSocket;  
    private JTextArea connectedClientsArea;  
    public ClientHandler(Socket socket, JTextArea connectedClientsArea) {  
        this.clientSocket = socket;  
        this.connectedClientsArea = connectedClientsArea;  
    }  
    @Override  
    public void run() {  
        try (BufferedReader in = new BufferedReader(new  
InputStreamReader(clientSocket.getInputStream()));  
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {  
            String request;  
            while ((request = in.readLine()) != null) {  
                System.out.println("Запит від клієнта: " + request);  
                // Обробка запиту  
                String response = processRequest(request, out); // Передаємо 'out' в  
processRequest  
                out.println(response);  
            }  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
    // Передаємо 'PrintWriter out' як параметр  
    private String processRequest(String request, PrintWriter out) {  
        try (Connection connection = db_log.getConnection()) {  
            if (request.startsWith("REGISTER")) {
```

					ITC.4KI.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		110

```

String[] parts = request.split(";");

String query = "INSERT INTO users (name, surname, email, password, phone,
birthdate) VALUES (?, ?, ?, ?, ?, ?)";

try (PreparedStatement statement = connection.prepareStatement(query)) {
    statement.setString(1, parts[1]);
    statement.setString(2, parts[2]);
    statement.setString(3, parts[3]);
    statement.setString(4, parts[4]);
    statement.setString(5, parts[5]);
    statement.setString(6, parts[6]);
    statement.executeUpdate();
    return "Реєстрація успішна!";
} catch (SQLException e) {
    e.printStackTrace();
    return "Помилка при реєстрації.";
}

} else if (request.startsWith("LOGIN")) {
    String[] parts = request.split(";");
    // Перевірка правильності кількості параметрів
    if (parts.length != 3) {
        return "Невірний формат запиту.";
    }

    String email = parts[1];
    String password = parts[2];

    String query = "SELECT * FROM users WHERE email = ? AND password = ?";
    try (PreparedStatement statement = connection.prepareStatement(query)) {
        statement.setString(1, email);
        statement.setString(2, password);
    }

    try (ResultSet resultSet = statement.executeQuery()) {
        if (resultSet.next()) {
            // Успішний вхід

            ServerWithGUI.SessionManager.loginUser(email, clientSocket); // Записуємо в сесію
        }
    }
}

```

```

System.out.println("Користувач успішно увійшов: " + email); // Логування
        return "Вхід успішний!";
    } else {
        System.out.println("Невірний логін або пароль для: " + email); //
Логування
        return "Неправильний логін або пароль.";
    }
}
} catch (SQLException e) {
    e.printStackTrace();
    return "Помилка при обробці запиту.";
}
} else if (request.startsWith("LOGOUT")) {
    String[] parts = request.split(";");
    // Перевірка наявності правильного формату для логування
    if (parts.length != 2) {
        return "Невірний формат запиту.";
    }
    String email = parts[1];
    // Перевірка, чи користувач вже увійшов
    if (!ServerWithGUI.SessionManager.isUserLoggedIn(email)) {
        return "Користувач не увійшов.";
    }
    // Вихід з сесії
    ServerWithGUI.SessionManager.logoutUser(email);
    System.out.println("Користувач вийшов: " + email); // Логування
    return "Вихід успішний!";
} else if (request.startsWith("GET_STUDENTS")) {
    // Якщо запит містить "GET_STUDENTS", викликаємо метод для отримання
    списку студентів
    StudentTable studentTable = new StudentTable();
    Object[][] studentsData = studentTable.studtab(); // Отримуємо масив студентів
    // Перетворюємо масив студентів в формат JSON

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

String jsonResponse = convertToJson(studentsData);

    // Відправляємо клієнту у форматі JSON
    out.println(jsonResponse);
    out.flush();
    return jsonResponse;
} else if (request.startsWith("GET_GROUP")) {

    // Якщо запит містить "GET_STUDENTS", викликаємо метод для отримання
    списку студентів

    groupTable GstudentTable = new groupTable();
    Object[][] groupData = GstudentTable.groupTab();
    // Перетворюємо масив студентів в формат JSON
    String jsonResponse = convertToJson3(groupData);
    // Відправляємо клієнту у форматі JSON
    out.println(jsonResponse);
    out.flush();
    return jsonResponse;
}

}

else if (request.startsWith("GET_COURSES")) {

    // Якщо запит містить "GET_COURSES", викликаємо метод для отримання
    списку курсів

    CoursesTable coursesTable = new CoursesTable();
    Object[][] courseData = coursesTable.coursesTab(); // Отримуємо масив курсів
    // Відправляємо клієнту у форматі JSON
    out.println(jsonResponse);
    out.flush();
    return jsonResponse;
}

else if (request.startsWith("GET_STUDGROUP")) {

    // Якщо запит містить "GET_COURSES", викликаємо метод для отримання
    списку курсів

    StudentGroupsTable studentGroupsTable = new StudentGroupsTable();
    Object[][] stgrData = studentGroupsTable.stgroupsTab(); // Отримуємо масив курсів

```

```

// Перетворюємо масив курсів у формат JSON
String jsonResponse = convertToJson4(stgrData);
// Відправляємо клієнту у форматі JSON
out.println(jsonResponse);
out.flush();
return jsonResponse;
}
else if (request.startsWith("GET_GRADES")) {
    // Якщо запит містить "GET_COURSES", викликаємо метод для отримання
    // списку курсів
    GradesTable gradesTable = new GradesTable();
    Object[][] gradesData = gradesTable.gradesTab(); // Отримуємо масив курсів
    // Перетворюємо масив курсів у формат JSON
    String jsonResponse = convertToJson5(gradesData);
    // Відправляємо клієнту у форматі JSON
    out.println(jsonResponse);
    out.flush();
    return jsonResponse;
}
else if (request.startsWith("ADD_COURSE")) {
    // Якщо запит містить "ADD_COURSE", викликаємо метод для додавання
    // курсу
    String[] parts = request.split(";");
    // Перевіряємо, чи правильно передано всі дані
    if (parts.length != 5) {
        out.println("Невірний формат запити для додавання курсу.");
        out.flush();
        return "Невірний формат запити для додавання курсу.";
    }
    String courseName = parts[1];    // Назва курсу
    String description = parts[2];    // Опис курсу
    int credits;

    String teacherName = parts[4];    // Ім'я викладача або ID

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// Перевірка та обробка кількості кредитів
try {
    credits = Integer.parseInt(parts[3]); // Кількість кредитів
} catch (NumberFormatException e) {
    out.println("Невірне значення для кількості кредитів.");
    out.flush();
    return "Невірне значення для кількості кредитів.";
}

// Використовуємо TeachersManager для отримання імені викладача
TeachersManager teachersManager = new TeachersManager();
int teacherID = -1; // Початкове значення для ID викладача
String teacherFullName = "";

try {
    // Спробуємо обробити teacherName як ID викладача
    teacherID = Integer.parseInt(teacherName); // Якщо це ID викладача
    teacherFullName = teachersManager.getTeacherNameById(teacherID); //
Отримуємо ім'я викладача за ID
} catch (NumberFormatException e) {
    // Якщо це не число, вважаємо, що це ім'я викладача
    teacherFullName = teacherName;
}

// Створюємо об'єкт класу CoursesManager для роботи з курсами
CoursesManager coursesManager = new CoursesManager();

// Викликаємо метод для додавання курсу, передаючи ID викладача
boolean isAdded = false;
if (teacherID != -1) {
    isAdded = coursesManager.addCourse(courseName, description, credits,
teacherID); // Якщо передано ID
} else {
    // Якщо передано ім'я викладача, потрібно знайти ID
    teacherID = teachersManager.getTeacherIdByName(teacherFullName); //
Додайте метод для отримання ID за ім'ям
    if (teacherID != -1) {

```

					ITC.4KI.0124.03-ПЗ	Арк.
						115
Змн.	Арк.	№ докум.	Підпис	Дата		


```

isAdded = coursesManager.addCourse(courseName, description, credits, teacherID);
    } else {
        out.println("Не вдалося знайти викладача за ім'ям: " + teacherFullName);
        out.flush();
        return "Не вдалося знайти викладача.";
    }
}

// Перевіряємо результат і відправляємо відповідь клієнту
if (isAdded) {
    out.println("Курс додано успішно: " + courseName);
    out.flush();
    return "Курс додано успішно!";
} else {
    out.println("Помилка при додаванні курсу.");
    out.flush();
    return "Помилка при додаванні курсу.";
}
}

else if (request.startsWith("ADD_STUDENT_TO_GROUP")) {
    // Якщо запит містить "ADD_STUDENT_TO_GROUP", викликаємо метод для
    додавання студента до групи
    String[] parts = request.split(";");
    // Перевіряємо, чи правильно передано всі дані
    if (parts.length != 3) {
        out.println("Невірний формат запиту для додавання студента до групи.");
        out.flush();
        return "Невірний формат запиту для додавання студента до групи.";
    }

    String studentIdentifier = parts[1]; // Ідентифікатор студента (ID або ім'я)
    String groupIdentifier = parts[2]; // Ідентифікатор групи (ID або назва)
    // Менеджери для роботи зі студентами та групами
    StGrManager stGrManager = new StGrManager();

    int studentID = -1;

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						116
Змн.	Арк.	№ докум.	Підпис	Дата		

```

int groupID = -1;

    // Спроба обробити студентський ідентифікатор
    try {
        studentID = Integer.parseInt(studentIdentifier); // Якщо це ID
    } catch (NumberFormatException e) {
        studentID = stGrManager.getStudentIdByName(studentIdentifier); // Якщо це
ім'я

        if (studentID == -1) {
            out.println("Студент з таким ім'ям не знайдений: " + studentIdentifier);
            out.flush();
            return "Студент не знайдений.";
        }
    }

    // Спроба обробити ідентифікатор групи
    try {
        groupID = Integer.parseInt(groupIdentifier); // Якщо це ID
    } catch (NumberFormatException e) {
        groupID = stGrManager.getGroupIdByName(groupIdentifier); // Якщо це назва
        if (groupID == -1) {
            out.println("Група з такою назвою не знайдена: " + groupIdentifier);
            out.flush();
            return "Група не знайдена.";
        }
    }

    // Перевіряємо, чи існують студент і група
    String studentFullName = stGrManager.getStudentNameById(studentID);
    if (studentFullName.isEmpty()) {
        out.println("Студент з таким ID не знайдений: " + studentID);
        out.flush();
        return "Студент не знайдений.";
    }

    String groupName = stGrManager.getGroupNameById(groupID);
    if (groupName.isEmpty()) {

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

out.println("Група з таким ID не знайдена: " + groupID);
    out.flush();
    return "Група не знайдена.";
}
// Викликаємо метод для додавання студента до групи
boolean isAdded = stGrManager.addStGr(studentID, groupID);
// Перевіряємо результат і надсилаємо відповідь клієнту
if (isAdded) {
    out.println("Студент " + studentFullName + " успішно доданий до групи " +
groupName + ".");
    out.flush();
    return "Студент доданий до групи успішно!";
} else {
    out.println("Помилка при додаванні студента до групи.");
    out.flush();
    return "Помилка при додаванні студента до групи.";
}
}
else if (request.startsWith("ADD_GRADES")) {
    // Якщо запит містить "ADD_GRADES", викликаємо метод для додавання
оцінки
    String[] parts = request.split(";");
    // Перевіряємо, чи правильно передано всі дані
    if (parts.length != 4) {
        out.println("Невірний формат запиту для додавання оцінки. Очікується:
ADD_GRADES;<StudentID>;<CourseID>;<Grade>");
        out.flush();
        return "Невірний формат запиту.";
    }
    String studentIdentifier = parts[1]; // Ідентифікатор студента
    String courseIdentifier = parts[2]; // Ідентифікатор курсу
    String gradeStr = parts[3];        // Оцінка
    // Менеджери для роботи з оцінками

```

```

GradesManager gradesManager = new GradesManager();

    int studentID = -1;

    int courseID = -1;

    int grade = -1;

// Спроба обробити студентський ідентифікатор
    try {
        studentID = Integer.parseInt(studentIdentifier); // Якщо це ID
    } catch (NumberFormatException e) {
        studentID = gradesManager.getStudentIdByName(studentIdentifier);
        if (studentID == -1) {
            out.println("Студент з таким ім'ям не знайдений: " + studentIdentifier);
            out.flush();
            return "Студент не знайдений.";
        }
    }

// Спроба обробити ідентифікатор курсу
    try {
        courseID = Integer.parseInt(courseIdentifier); // Якщо це ID
    } catch (NumberFormatException e) {
        courseID = gradesManager.getCourseIdByName(courseIdentifier); // Якщо це
назва
        if (courseID == -1) {
            out.println("Курс з такою назвою не знайдений: " + courseIdentifier);
            out.flush();
            return "Курс не знайдений.";
        }
    }

// Перевірка оцінки
    try {
        grade = Integer.parseInt(gradeStr);
        if (grade < 0 || grade > 100) {
            out.println("Оцінка має бути числом від 0 до 100.");
            out.flush();
        }
    }

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		119

```

return "Невірне значення оцінки.";
    }
} catch (NumberFormatException e) {
    out.println("Невірний формат оцінки: " + gradeStr);
    out.flush();
    return "Невірний формат оцінки.";
}

// Перевіряємо, чи існують студент і курс
String studentFullName = gradesManager.getStudentNameById(studentID);
if (studentFullName.isEmpty()) {
    out.println("Студент з таким ID не знайдений: " + studentID);
    out.flush();
    return "Студент не знайдений.";
}

String courseName = gradesManager.getCourseNameById(courseID);
if (courseName.isEmpty()) {
    out.println("Курс з таким ID не знайдений: " + courseID);
    out.flush();
    return "Курс не знайдений.";
}

// Додавання оцінки
boolean isAdded = gradesManager.addGrade(studentID, courseID, grade); //
Викликаємо метод для додавання оцінки

// Перевірка результату і надсилаємо відповідь клієнту
if (isAdded) {
    out.println("Оцінку успішно додано для студента " + studentFullName + " по
курсу " + courseName + ".");
    out.flush();
return "Оцінку успішно додано.";
} else {
    out.println("Помилка при додаванні оцінки.");
    out.flush();
    return "Помилка при додаванні оцінки.";
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    }

    else if (request.startsWith("ADD_STUDENT")) {
        // Розділяємо запит для отримання даних студента
        String[] parts = request.split(";");
        if (parts.length != 7) {
            return "Невірний формат запиту для додавання студента.";
        }

        // Отримуємо дані студента
        String firstName = parts[1];
        String lastName = parts[2];
        String email = parts[3];
        String phone = parts[4];
        String birthDate = parts[5];
        int enrollmentYear = Integer.parseInt(parts[6]);

        // Викликаємо метод для додавання студента
        StudentAdder studentAdder = new StudentAdder();

        boolean isAdded = studentAdder.addStudent(firstName, lastName, email, phone,
        birthDate, enrollmentYear);
        if (isAdded) {
            System.out.println("Студента додано успішно: " + firstName + " " +
        lastName);
            return "Студента додано успішно!";
        } else {
            return "Помилка при додаванні студента.";
        }
    }

    else if (request.startsWith("ADD_GROUP")) {
        // Розділяємо запит для отримання даних студента
        String[] parts = request.split(";");
        if (parts.length != 4) {
            return "Невірний формат запиту для додавання студента.";
        }
    }

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		121

```

// Отримуємо дані студента

String groupName = parts[1];
String speciality = parts[2];
int enrollmentYear = Integer.parseInt(parts[3]);
// Викликаємо метод для додавання студента
GroupAdder groupAdder = new GroupAdder();
boolean isAdded = groupAdder.addGroup(groupName, speciality,
enrollmentYear);
if (isAdded) {
    System.out.println("Групу додано успішно: " + groupName);
    return "Групу додано успішно!";
} else {
    return "Помилка при додаванні групи.";
}
}

else if (request.startsWith("ADD_TEACHERS")) {
    // Розділяємо запит для отримання даних студента
    String[] parts = request.split(";");
    if (parts.length != 6) {
        return "Невірний формат запиту для додавання студента.";
    }

    // Отримуємо дані студента
    String firstName = parts[1];
    String lastName = parts[2];
    String email = parts[3];
    String phone = parts[4];
    String department = parts[5];

    // Викликаємо метод для додавання студента
    TeachersAdder teachersAdder = new TeachersAdder();
    boolean isAdded = teachersAdder.addTeacher(firstName, lastName, email, phone,
if (isAdded) {department);
    System.out.println("Викладача додано успішно: " + firstName + " " + lastName);
    return "Викладача додано успішно!";
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		122

```

    } else {

        return "Помилка при додаванні студента.";

    }

}

else if (request.startsWith("DELETE_STUDENT")) {

    String[] parts = request.split(";");

    if (parts.length != 2) {

        return "Невірний формат запиту для видалення студента.";

    }

String email = parts[1];

    // Викликаємо метод для видалення студента

    StudentDeleter studentDeleter = new StudentDeleter();

    boolean isDeleted = studentDeleter.deleteStudent(email);

if (isDeleted) {

    System.out.println("Студента видалено успішно: " + email);

    return "Студента видалено успішно!";

    } else {

        return "Помилка при видаленні студента.";

    }

}

else if (request.startsWith("DELETE_COURSE")) {

    String[] parts = request.split(";");

    if (parts.length != 2) {

        return "Невірний формат запиту для видалення курсу.";

    }

String CourseName = parts[1];

    // Викликаємо метод для видалення студента

    CoursesDeleter coursesDeleter = new CoursesDeleter();

    boolean isDeleted = coursesDeleter.deleteCourse(CourseName);

if (isDeleted) {

    System.out.println("Курс видалено успішно: " + CourseName);

    return "Курс видалено успішно!";

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		123


```

    } else {

        return "Помилка при видаленні курсу.";

    }

}

else if (request.startsWith("DELETE_GROUP")) {

    String[] parts = request.split(";");

    if (parts.length != 2) {

        return "Невірний формат запиту для видалення курсу.";

    }

    String GroupName = parts[1];

    // Викликаємо метод для видалення студента

    GroupDeleter groupDeleter = new GroupDeleter();

    boolean isDeleted = groupDeleter.deleteGR(GroupName);

    if (isDeleted) {

        System.out.println("Назву групи видалено успішно: " + GroupName);

        return "Назву групи видалено успішно!";

    } else {

        return "Помилка при видаленні групи.";

    }

}

else if (request.startsWith("{}")) { // Перевірка на JSON запит

    System.out.println("Отриманий JSON запит: [" + request + "]");

    try {

        // Парсинг JSON

        JSONObject jsonRequest = new JSONObject(request);

        System.out.println("Парсинг JSON запиту: " + jsonRequest);

        // Обробка команди DELETE_STUDENT_FROM_GROUP

        if (jsonRequest.has("command") &&
            jsonRequest.getString("command").equals("DELETE_STUDENT_FROM_GROUP")) {

            // Перевірка параметрів studentID та groupID

            if (!jsonRequest.has("studentID") || !jsonRequest.has("groupID")) {

```

```

System.out.println("Невірний формат: відсутні studentID або groupID.");
        return "Невірний формат запиту: потрібно вказати studentID та
groupID.";
    }
    int studentID = jsonRequest.getInt("studentID");
    int groupID = jsonRequest.getInt("groupID");
    System.out.println("studentID: " + studentID + ", groupID: " + groupID +
    "");
    StGrDeleter stGrDeleter = new StGrDeleter();
    if (!stGrDeleter.isStudentExists(studentID)) {
        return "Студента з ID " + studentID + " не існує.";
    }
    if (!stGrDeleter.isGroupExists(groupID)) {
        return "Групи з ID " + groupID + " не існує.";
    }
    boolean isDeleted = stGrDeleter.deleteStudentByID(studentID, groupID);
    return isDeleted ? "success" : "Помилка при видаленні студента з групи.";
}

// Обробка команди DELETE_GRADES
    if (jsonRequest.has("grades") && jsonRequest.getString("grades").equals
("DELETE_GRADES")) {
/ Перевірка параметрів studentID та courseID
        if (!jsonRequest.has("studentID") || !jsonRequest.has("courseID")) {
            System.out.println("Невірний формат: відсутні studentID або courseID.");
            return "Невірний формат запиту: потрібно вказати studentID та
courseID.";
        }
        int studentID = jsonRequest.getInt("studentID");
        int courseID = jsonRequest.getInt("courseID");
        System.out.println("studentID: " + studentID + ", courseID: " + courseID + "");
        GradesDeleter gradesDeleter = new GradesDeleter();
        if (!gradesDeleter.isStudentExists(studentID)) {

```

					ИТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		125

```

return "Студента з ID " + studentID + " не існує.";
    }
    if (!gradesDeleter.isCourseExists(courseID)) {
        return "Курсу з ID " + courseID + " не існує.";
    }
    boolean isDeleted = gradesDeleter.deleteStudentByID(studentID, courseID);
    return isDeleted ? "success" : "Помилка при видаленні студента з курсу.";
}

// Якщо команда не розпізнана
System.out.println("Невідома команда.");
return "Невідома команда або формат запиту.";
} catch (Exception e) {
    System.out.println("Помилка при обробці запиту: " + e.getMessage());
    e.printStackTrace();
    return "Невідома помилка при обробці запиту.";
}
}

else if (request.startsWith("DELETE_TEACHERS")) {
    String[] parts = request.split(";");
    if (parts.length != 2) {
        return "Невірний формат запиту для видалення викладача.";
    }

    String email = parts[1];

    // Викликаємо метод для видалення студента
    TeachersDeleter teachersDeleter = new TeachersDeleter();
    boolean isDeleted = teachersDeleter.deleteTeacher(email);

    if (isDeleted) {
        System.out.println("Викладача видалено успішно: " + email);
        return "Викладача видалено успішно!";
    } else {
        return "Помилка при видаленні викладача.";
    }
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						126
Змн.	Арк.	№ докум.	Підпис	Дата		

```

else if (request.startsWith("GET_TEACHERS")) {
    // Якщо запит містить "GET_STUDENTS", викликаємо метод для отримання
    списку студентів

    TeachersTable teachersTable = new TeachersTable();

    Object[][] teachData = teachersTable.teachtab(); // Отримуємо масив студентів
    // Перетворюємо масив студентів в формат JSON
    String jsonResponse = convertToJson2(teachData);
    // Відправляємо клієнту у форматі JSON
    out.println(jsonResponse);
    out.flush();
    return jsonResponse;
} else {
    return "Невідомий запит.";
}
} catch (SQLException e) {
    e.printStackTrace();
    return "Помилка при підключенні до бази даних.";
}
}

private String convertToJson5(Object[][] gradesData) {
    GradesManager gradesManager = new GradesManager();// Ініціалізуємо менеджер для
    доступу до БД

    StringBuilder json = new StringBuilder();
    json.append("[");
    for (int i = 0; i < gradesData.length; i++) {
        if (gradesData[i][0] == null) continue; // Пропускаємо порожні рядки
        json.append("{");
        json.append("\"studentID\":").append(gradesData[i][0]).append("\", ");
        json.append("\"courseID\":").append(gradesData[i][1]).append("\", ");
        // Отримуємо ім'я студента та групи за їх ID
        int studentID = (int) gradesData[i][0];

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

String studentName = gradesManager.getStudentNameById(studentID);

    json.append("\"studentName\":\"").append(studentName).append("\", ");

    int courseID = (int) gradesData[i][1];

    String groupName = gradesManager.getCourseNameById(courseID);

    json.append("\"courseName\":\"").append(groupName).append("\", ");

    int grade = (int) gradesData[i][2];

    json.append("\"grade\":\"").append(grade);

    json.append("}");

// Додаємо кому, лише якщо це не останній об'єкт
    if (i < gradesData.length - 1) {

        json.append(",");

    }

}

json.append("]");

return json.toString();

}

private String convertToJson4(Object[][] stgrData) {

    StudentGroupsManager studentGroupsManager = new StudentGroupsManager(); //
    Ініціалізуємо менеджер для доступу до БД

    StringBuilder json = new StringBuilder();

    json.append("[");

    for (int i = 0; i < stgrData.length; i++) {

        if (stgrData[i][0] == null) continue; // Пропускаємо порожні рядки

        json.append("{");

        json.append("\"studentID\":\"").append(stgrData[i][0]).append("\", ");

        json.append("\"groupID\":\"").append(stgrData[i][1]).append("\", ");

        // Отримуємо ім'я студента та групи за їх ID

        int studentID = (int) stgrData[i][0];

        String studentName = studentGroupsManager.getStudentsNameById(studentID);

        json.append("\"studentName\":\"").append(studentName).append("\", ");

        int groupID = (int) stgrData[i][1];

        String groupName = studentGroupsManager.getGroupsNameById(groupID)

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		128

```

json.append("\"groupName\":\").append(groupName).append("\"");
    json.append("}");
    // Додаємо кому, лише якщо це не останній об'єкт
    if (i < stgrData.length - 1) {
        json.append(",");
    }
}
json.append("]");
return json.toString();
}

private String convertToJson3(Object[][] groupData) {
    StringBuilder json = new StringBuilder();
    json.append("[");
    for (int i = 0; i < groupData.length; i++) {
        if (groupData[i][0] == null) continue; // Пропускаємо порожні рядки
        json.append("{");
        //json.append("\"studentID\":").append(studentsData[i][0]).append("\", ");
        json.append("\"GroupName\":").append(groupData[i][0]).append("\", ");
        json.append("\"Specialty\":").append(groupData[i][1]).append("\", ");
        json.append("\"enrollmentYear\":").append(groupData[i][2]).append("\"");
        json.append("},");
    }
    // Видаляємо останню кому, якщо така є
    if (json.charAt(json.length() - 1) == ',') {
        json.deleteCharAt(json.length() - 1);
    }
    json.append("]");
    return json.toString();
}

// Метод для перетворення масиву даних студентів в JSON
private String convertToJson(Object[][] studentsData) {

```

```

StringBulder json = new StringBulder();

json.append("[");

for (int i = 0; i < studentsData.length; i++) {

    if (studentsData[i][0] == null) continue; // Пропускаємо порожні рядки

    json.append("{");

    //json.append("\"studentID\":").append(studentsData[i][0]).append("\", ");
    json.append("\"firstName\":").append(studentsData[i][0]).append("\", ");
    json.append("\"lastName\":").append(studentsData[i][1]).append("\", ");
    json.append("\"email\":").append(studentsData[i][2]).append("\", ");
    json.append("\"phone\":").append(studentsData[i][3]).append("\", ");
    json.append("\"birthDate\":").append(studentsData[i][4]).append("\", ");
    json.append("\"enrollmentYear\":").append(studentsData[i][5]).append("\", ");

    json.append("},");

}

// Видаляємо останню кому, якщо така є

if (json.charAt(json.length() - 1) == ',') {

    json.deleteCharAt(json.length() - 1);

}

json.append("]");

return json.toString();

}

private String convertToJson2(Object[][] teachData) {

    StringBulder json = new StringBulder();

    json.append("[");

    for (int i = 0; i<teachData.length; i++) {

        if (teachData[i][0] == null) continue; // Пропускаємо порожні рядки

        json.append("{");

        //json.append("\"studentID\":").append(studentsData[i][0]).append("\", ");
        json.append("\"firstName\":").append (teachData[i][0]).append("\", ");
        json.append("\"lastName\":").append (teachData[i][1]).append("\", ");
        json.append("\"email\":").append( teachData[i][2]).append("\", ");
    }
}

```

```

        json.append("\"phone\":").append( teachData[i][3]).append("\", ");
        json.append("\"department\":").append (teachData[i][4]).append("\", ");
        json.append("},");
    }

    // Видаляємо останню кому, якщо така є
    if (json.charAt(json.length() - 1) == ',') {
        json.deleteCharAt(json.length() - 1);
    }
    json.append("]");
    return json.toString();
}

private String convertToJsonCourses(Object[][] courseData) {
    CoursesManager coursesManager = new CoursesManager(); // Ініціалізуємо менеджер
    для доступу до БД

    StringBuilder json = new StringBuilder();
    json.append("[");
    for (int i = 0; i < courseData.length; i++) {
        if (courseData[i][0] == null) continue; // Пропускаємо порожні рядки
        json.append("{");
        json.append("\"courseName\":").append(courseData[i][0]).append("\", ");
        json.append("\"description\":").append(courseData[i][1]).append("\", ");
        json.append("\"credits\":").append(courseData[i][2]).append(", ");
        // Отримуємо ім'я та прізвище викладача за ID
        int teacherID = (int) courseData[i][3];
        String teacherName = coursesManager.getTeacherNameById(teacherID);
        json.append("\"teacherName\":").append(teacherName).append("\",");
        json.append("},");
    }

    // Видаляємо останню кому, якщо така є
    if (json.charAt(json.length() - 1) == ',') {
        json.deleteCharAt(json.length() - 1); }

```



```

json.append("]");
    return json.toString();
}
}

```

Код класу db_dekanat

```

package org.example;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class db_dekanat {
    //private static final String URL = "jdbc:mysql://192.168.100.119:3306/ElectronicDean";
    private static final String URL = "jdbc:mysql://192.168.100.131:3306/ElectronicDean";
    private static final String USER = "admin";
    private static final String PASSWORD = "sasha1575";
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
    // Закриття з'єднання
    public static void closeConnection(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                System.out.println("Помилка при закритті з'єднання: " + e.getMessage());
            }
        }
    }
}

```

Код класу db_log

```

package org.example;
import java.sql.Connection;

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						132
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import java.sql.DriverManager;
import java.sql.SQLException;
public class db_log {
    //private static final String URL = "jdbc:mysql://192.168.100.119:3306/mysite";
    private static final String URL = "jdbc:mysql://192.168.100.131:3306/mysite";
    private static final String USER = "admin";
    private static final String PASSWORD = "sasha1575";
    static {
        try {
            // Ініціалізація драйвера
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            System.out.println("Помилка при завантаженні драйвера: " + e.getMessage());
        }
    }
    // Отримання з'єднання з базою даних
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
    // Закриття з'єднання
    public static void closeConnection(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                System.out.println("Помилка при закритті з'єднання: " + e.getMessage());
            }
        }
    }
}

```

Код класу Client

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		133

```

package org.example;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
import java.net.Socket;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Client {

    private static final String HOST = Configuration.HOST;
    private static final int PORT = Configuration.PORT;
    private static boolean isLoggedIn = false;

    // Метод для хешування пароля
    private static String hashPassword(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.digest(password.getBytes());
            StringBuilder hexString = new StringBuilder();
            for (byte b : hash) {
                hexString.append(String.format("%02x", b));
            }
            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("Помилка хешування пароля", e);
        }
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Клієнт");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);

        JPanel panel = new JPanel();

```

					ITC.4KI.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		134

```

frame.add(panel);

JTextField emailField = new JTextField(20);
JPasswordField passwordField = new JPasswordField(20);
JButton loginButton = new JButton("Вхід");
JButton registerButton = new JButton("Реєстрація");
panel.add(new JLabel("Email:"));
panel.add(emailField);
panel.add(new JLabel("Пароль:"));
panel.add(passwordField);
panel.add(loginButton);
panel.add(registerButton);

// Обробник події для кнопки "Вхід"
loginButton.addActionListener(e -> {
    try {
        // Ініціалізація підключення до сервера
        ServerConnection.initialize(HOST, PORT);
        Socket socket = ServerConnection.getSocket();
        PrintWriter out = ServerConnection.getOut();
        BufferedReader in = ServerConnection.getIn();
        String email = emailField.getText();
        String password = new String(passwordField.getPassword());
        // Хешуємо введений пароль перед відправкою
        String hashedPassword = hashPassword(password);
        out.println("LOGIN;" + email + ";" + hashedPassword);
        // Читання відповіді з серверу з таймаутом
        String response = in.readLine();
        if (response == null) {
            JOptionPane.showMessageDialog(frame, "Час на з'єднання вичерпано. Спробуйте ще раз.");
            return;
        }
    }
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		135

```

JOptionPane.showMessageDialog(frame, response);

    if ("Вхід успішний!".equals(response)) {
        isLoggedIn = true;
        SwingUtilities.invokeLater(() -> {
            frame.setVisible(false);
            frame.dispose();
            MenuForm menuForm = new MenuForm();
            menuForm.setVisible(true);
        });
    } else {
        JOptionPane.showMessageDialog(frame, "Невірний логін чи пароль");
    }
} catch (IOException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(frame, "Помилка при підключенні до сервера:
" + ex.getMessage());
}

});

// Обробник події для кнопки "Реєстрація"
registerButton.addActionListener(e -> {
    // Відкрити нове вікно для реєстрації
    new RegistrationForm(HOST, PORT);
});

frame.setVisible(true);
}

public static boolean isLoggedIn() {
    return isLoggedIn;
}
}

```

Код класу Configuration

					ІТС.4КІ.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		136

```

package org.example;

public class Configuration {

    public static final String HOST = "192.168.186.147";

    //public static final String HOST = "192.168.99.122";

    public static final int PORT = 12345;

}

```

Код класу ServerConnection

```

package org.example;

import java.io.*;

import java.net.Socket;

public class ServerConnection {

    private static Socket socket;

    private static PrintWriter out;

    private static BufferedReader in;

    private static String host; // Адреса сервера

    private static int port; // Порт сервера

    public static void initialize(String serverHost, int serverPort) {

        if (host == null && port == 0) {

            host = serverHost;

            port = serverPort;

            System.out.println("Параметри сервера ініціалізовані: " + host + ":" + port);

        } else {

            System.out.println("Сервер вже ініціалізовано: " + host + ":" + port);

        }

    }

    public static Socket getSocket() throws IOException {

        if (host == null || port == 0) {

            throw new IllegalStateException("Сервер не ініціалізовано. Використовуйте метод initialize().");

        }

        if (socket == null || socket.isClosed()) {

```

```

System.out.println("Спроба підключення до сервера: " + host + ":" + port);

    socket = new Socket(host, port);

    System.out.println("Підключення до сервера успішне!");

    out = new PrintWriter(socket.getOutputStream(), true);

    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

    System.out.println("Створено потоки введення/виведення.");

} else {

    System.out.println("Повторне використання існуючого сокета.");

}

return socket;

}

public static PrintWriter getOut() throws IOException {

    if (out == null) {

        throw new IllegalStateException("Сокет не ініціалізований. Використовуйте getSocket().");

    }

    return out;

}

public static BufferedReader getIn() throws IOException {

    if (in == null) {

        throw new IllegalStateException("Сокет не ініціалізований. Використовуйте getSocket().");

    }

    return in;

}

public static void closeConnection() {

    try {

        if (socket != null && !socket.isClosed()) {

            System.out.println("Закриття з'єднання з сервером...");

            socket.close();

            System.out.println("З'єднання успішно закрито.");

        }

    }

}

```

```

    }

    } catch (IOException e) {

        System.err.println("Помилка при закритті з'єднання: " + e.getMessage());

    }

}

```

Код класу SecureEncryption

```

package org.example;
import java.security.SecureRandom;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
public class SecureEncryption {
    public SecureEncryption() {

    }

    public static SecretKey generateKeyFromPassword(String password, byte[] salt) throws
    Exception {
        SecretKeyFactory factory =
        SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
        PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt, 10000, 256);
        return new SecretKeySpec(factory.generateSecret(spec).getEncoded(), "AES");
    }

    public static String encryptAES(String data, String password) throws Exception {
        SecureRandom secureRandom = new SecureRandom();
        byte[] salt = new byte[16];
        secureRandom.nextBytes(salt);
        byte[] iv = new byte[16];
        secureRandom.nextBytes(iv);
        SecretKey key = generateKeyFromPassword(password, salt);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        IvParameterSpec ivSpec = new IvParameterSpec(iv);
        cipher.init(1, key, ivSpec);

        byte[] encryptedData = cipher.doFinal(data.getBytes());
        byte[] result = new byte[salt.length + iv.length + encryptedData.length];
        System.arraycopy(salt, 0, result, 0, salt.length);

        System.arraycopy(iv, 0, result, salt.length, iv.length);
        System.arraycopy(encryptedData, 0, result, salt.length + iv.length, encryptedData.length);
        return Base64.getEncoder().encodeToString(result);
    }
}

```

					ITC.4KI.0124.03-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		139


```

public static String decryptAES(String encryptedData, String password) throws Exception {
    byte[] encryptedBytes = Base64.getDecoder().decode(encryptedData);
    byte[] salt = new byte[16];
    byte[] iv = new byte[16];
    System.arraycopy(encryptedBytes, 0, salt, 0, salt.length);
    System.arraycopy(encryptedBytes, salt.length, iv, 0, iv.length);
    byte[] encryptedMessage = new byte[encryptedBytes.length - salt.length - iv.length];
    System.arraycopy(encryptedBytes, salt.length + iv.length, encryptedMessage, 0,
encryptedMessage.length);
    SecretKey key = generateKeyFromPassword(password, salt);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(2, key, ivSpec);
    byte[] decryptedData = cipher.doFinal(encryptedMessage);
    return new String(decryptedData);
}
}

```

Код класу ConfigReader

```

package org.example;

import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;

public class ConfigReader {

    private static final String CONFIG_FILE = "S:\\Криптографія\\Java\\1\\config.properties";
    // Відносний шлях до конфігураційного файлу

    // Читає значення MASTER_KEY з конфігураційного файлу

    public static String getMasterKey() {

        Properties properties = new Properties();

        try (FileInputStream fis = new FileInputStream(CONFIG_FILE)) {

            properties.load(fis);

            String masterKey = properties.getProperty("MASTER_KEY");

            if (masterKey == null) {

                System.out.println("Ключ MASTER_KEY не знайдено в конфігураційному файлі!");

                return null;

            }

            return masterKey; // Отримуємо значення MASTER_KEY
        }
    }
}

```

```

} catch (IOException e) {
    System.out.println("Помилка при зчитуванні конфігураційного файлу: " +
e.getMessage());
    e.printStackTrace();
    return null; // Повертаємо null, якщо не вдається зчитати файл
}
}
}

```

					ІТС.4КІ.0124.03-ПЗ	Арк.
						141
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК Д Приклад роботи програми

Реєстрація

Ім'я: Адміністратор

Прізвище: Системи

Email: 1@gmail.com

Пароль:

Телефон: +380101010101

Дата народження (YYYY-MM-DD): 1999-09-15

Зареєструватися

Рис. Д.1. – Реєстрація користувача методом хешування SH-256

Клієнт

Email: 1@gmail.com

Пароль:

Вхід Реєстрація

Рис. Д.2. – Вхід в систему

Меню

Студенти Викладачі Дисципліни Групи Підсумки Вихід

Відкрити форму студентів

Додати студента

Рис. Д.3. – Меню програми

Додати викладача

First Name: Віталій

Last Name: Кузнецов

Email: vyk110@gmail.com

Phone: +380507560709

Department: IMIT

Додати викладача

FirstName	LastName	Email	Phone	Department
Микола	Семенов	vyk11@gmail.com	+380507560714	IMIT
Жучок	Юрій	vyk12@gmail.com	+380507560703	IMIT
Сергєєва	Вікторія	vyk13@gmail.com	+380507560701	IMIT
Донченко	Світлана	vyk14@gmail.com	+380507560702	IMIT
Світлана	Переяславська	vyk15@gmail.com	+380507560704	IMIT
Володимир	Матієвський	vyk16@gmail.com	+380507560705	IMIT
Ольга	Смагіна	vyk17@gmail.com	+380507560706	IMIT
Геннадій	Могилий	vyk18@gmail.com	+380507560707	IMIT
Ліна	Бондаренко	vyk19@gmail.com	+380507560708	IMIT
Віталій	Кузнецов	vyk110@gmail.com	+380507560709	IMIT

Рис. Д.4. – Додавання викладача

Реєстрація

Ім'я: Адміністратор

Прізвище: Системи

Email: 2@gmail.com

Пароль:

Телефон: +380101010101

Дата народження (YYYY-MM-DD): 1999-09-15

Зареєструватися

Рис. Д.5. – Реєстрація користувача за допомогою влачного методу захисту паролів

ID	Ім'я	Прізвище	Email	Пароль	Телефон	Дата народження	Дата реєстрації
98	Адміністратор	Системи	1@gmail.com	bbc5e661e106c6dcd8dcdd186454c2fcb3c71...	+380101010101	1999-09-15	2025-03-24 07:37:45
100	Адміністратор	Системи	2@gmail.com	+5ZhVzQbaLYVuDyhY74kiDC19/62hHWR7BuH4...	+380101010101	1999-09-15	2025-03-24 08:18:59
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. Д.6. – Зашифровані дані методом хеш функції та власним методом