ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДЕКАНАТУ МОВОЮ JAVA**»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
Коваль Б.В.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А.
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність 121 «Інженерія програманого забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного зебезпечення

Негоденко В.В.
“ ___ ” _____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

КОВАЛЯ БОГДАНА ВАСИЛЬОВИЧА
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення для автоматизації роботи деканату мовою Java»
Керівник роботи: Жебка В.В., д.т.н., доц., зав. кафедри ТЦР
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26
2. Строк подання студентом роботи «1» червня 2023 року
3. Вихідні дані до роботи:
3.1 Науково-технічна література з питань, пов‘язаних із застосування веб-додатків;
3.2 Практичній досвід розробки веб-додатків.
3.3 Концепція побудови веб-додатків;
4. Перелік демонстраційних матеріалів
4.1 Тема дипломної роботи
4.2 Мета роботи. Об’єкт дослідження. Предмет дослідження.
4.3 Результат дослідження розробки веб-додатків.
4.4 Результат дослідження фреймворків для веб-розробки.
4.5 Результати дослідження та опис реалізації програми.
4.6 Апробація результатів дослідження
4.7 Висновки
5. Дата видачі завдання 25.02.2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.2023 - 25.02.2023	виконано
2	Аналіз та дослідження існуючих аналогів	26.02.2023 - 10.03.23	виконано
3	Дослідження програмних об’єктів	11.03.2023 - 20.03.23	виконано
4	Моделювання об’єкту проектування	25.02.2023 - 01.03.23	виконано
5	Розробка веб-додатку	02.03.2023 - 20.03.23	виконано
6	Вступ, висновки, реферат	21.03.2023 - 24.04.23	виконано
7	Розробка обов’язкових демонстраційних матеріалів	25.04.2023 - 10.05.23	виконано
8	Попередній захист роботи	26.05.2023	виконано
9	Здача роботи	01.06.2023	виконано

Студент _____ Коваль Б.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жебка В.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 80 с., 1 табл., 22 рис.,, 47 джерел.

Об'єкт дослідження: автоматизація окремих процесів роботи деканату.

Предмет дослідження: програмне забезпечення для автоматизації роботи деканату.

Мета роботи: спрощення окремих робочих робочих процесів деканату за допомогою розробленого програмного забезпечення мовою Java.

Методи дослідження: методи системного аналізу, методи моделювання засновані на відомих методах системного підходу, а саме: аналітично-розрахунковий, теоретичного аналізу з використанням моделювання.

У даному дипломному проекті було виконано комплексну розробку програмного забезпечення для автоматизації роботи деканату. Проект складався з кількох етапів, включаючи аналіз предметної галузі, порівняння існуючих аналогів, визначення необхідних інструментів, проектування, реалізації та тестування.

Під час аналізу предметної галузі були вивчені основні процеси та вимоги деканату. Було проведено дослідження наявних аналогів та визначено переваги та недоліки кожного з них. Цей етап допоміг зрозуміти основні вимоги та потреби, що повинні були бути враховані в розробці нової системи.

Реалізація проекту включала розробку функціональних модулів, забезпечення взаємодії з базою даних, розробку інтерфейсу користувача та виконання основної логіки програми. Завдяки використанню визначених інструментів та технологій, розроблена система демонструє надійну та ефективну роботу, відповідає поставленим вимогам та задовольняє потреби деканату.

Галузь використання – університети які потребують автоматизацію роботи деканату.

Ключові слова: веб-додаток, фреймворк, Java, Spring Framework, Mockito, JUnit, автоматизація, розробка, проектування.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

IT	-	Information technology
MVC	-	Model-View-Controller
SQL	-	Structured query language
SPA	-	Single page application
HTML	-	HyperText Markup Language
CSS	-	Cascading Style Sheets
ВНЗ	-	Вищий навчальний заклад

ВСТУП

В сучасному світі інформаційних технологій, швидкого розвитку комп'ютерів та Інтернету, автоматизація різноманітних процесів стала необхідністю для ефективної та швидкої роботи багатьох сфер діяльності. Однією з таких сфер є деканат вищих навчальних закладів, де ведеться складна обліково-документальна робота, пов'язана зі студентами, групами, навчальними планами та багатьма іншими аспектами студентського життя.

У зв'язку зі зростанням обсягу інформації та складності процесів у деканаті, традиційні методи ведення обліку та документообігу стають недостатньо ефективними. Виникає потреба у впровадженні сучасних інформаційних технологій, зокрема програмного забезпечення та веб-додатків, які спрощують та автоматизують роботу деканату.

Метою даної дипломної роботи є розробка програмного забезпечення для автоматизації роботи деканату вищого навчального закладу. Це програмне забезпечення буде забезпечувати зручний доступ до інформації про студентів, групи, навчальні плани, успішність та інші аспекти деканатської роботи, сприяти оптимізації процесів та поліпшенню продуктивності.

Розробка цього програмного забезпечення має на меті вирішити декілька ключових проблем, з якими стикається деканат в повсякденній роботі. Перша проблема - це складність та обсяг ручного ведення обліку студентів, їх успішності та відвідуваності. Цей процес вимагає великої кількості часу, ресурсів та може бути схильним до помилок. Другою проблемою є доступність та швидкість доступу до інформації для деканатських працівників. Традиційні методи зберігання та пошуку даних можуть бути малоефективними та обмеженими. Використання програмного забезпечення та веб-додатків дозволить забезпечити централізований доступ до інформації, швидкий пошук та оновлення даних. Третя проблема полягає у складності та нетривкості процесу створення та оцінювання навчальних занять. Традиційні методи

вимагають великої кількості ручної роботи та можуть бути піддаються ризику помилок.

Програмне забезпечення дозволить викладачам створювати та оцінювати заняття в онлайн-режимі, забезпечуючи швидкість та точність.

Застосування програмного забезпечення для автоматизації роботи деканату має на увазі не тільки покращення роботи самого деканату, але й підвищення якості обслуговування студентів. Швидкий доступ до інформації, зручний інтерфейс та точність даних сприятимуть взаємодії між студентами та деканатом, а також сприятимуть вирішенню проблем та розблокуванню потенціалу студентів шляхом забезпечення точної інформації про їхній академічний прогрес.

У процесі роботи також будуть розглянуті важливі аспекти безпеки даних та конфіденційності. Застосування відповідних заходів безпеки, таких як автентифікація користувачів, контроль доступу та захист даних, дозволяє забезпечити, що інформація про студентів та навчальні процеси залишається конфіденційною та захищеною.

Об'єкт дослідження: процес розробки програмного забезпечення для автоматизації роботи деканату

Предмет дослідження: методи і моделі побудови програмного забезпечення для автоматизації роботи деканату.

Мета роботи: покращення процесу роботи деканату за допомого розробленого програмного забезпечення на мові Java

Опираючись на поставлену мету, були поставлені наступні задачі:

- проаналізувати обрану предметну галузь;
- порівняти існуючі аналоги;
- визначити інструменти які необхідні для розробки;
- спроектувати проект для автоматизації роботи деканату;
- реалізувати проект;
- протестувати роботу програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 **Веб додатки**

Проаналізувавши види програмного забезпечення визначив, що найбільш оптимальним рішенням для розробки програмного забезпечення для автоматизації деканату буде веб-додаток.

Веб-додаток є програмним забезпеченням або програмою, яка доступна через веб-браузер. Він є інтерактивним веб-застосуванням, яке надає користувачам функціональні можливості та дозволяє отримати доступ до різноманітних сервісів і ресурсів через Інтернет [1] [2].

Зазвичай зовнішній інтерфейс веб-додатків розробляється з використанням мов розмітки HTML, спеціальної мови стилів CSS і мови програмування JavaScript. [2] Ці мови є стандартними для веб-розробки і підтримуються всіма сучасними браузерами, такими як Opera, Chrome, Mozilla, Edge. [29]

При розробці серверної частини веб-додатка розробники мають можливість використовувати різні мови програмування або фреймворки, такі як Python, PHP, Ruby або Java. Це надає їм широкий вибір інструментів для створення потужних та ефективних веб-додатків, які здатні обробляти запити користувачів і забезпечувати необхідну функціональність. [2]

Аналізуючи особливості розробки та функціонування веб-додатків, можна виділити ключові переваги, які наведені нижче. Кросплатформовість - веб-додатки можуть працювати на будь-якій операційній системі (Linux, Mac, Windows), оскільки вони базуються на веб-технологіях та відкриваються через браузер. [2]

Це робить їх доступними для широкого кола користувачів. Простота підтримки - завдяки використанню єдиного коду для всіх користувачів, веб-додатки є легше підтримувати порівняно з настільними додатками. Розробка зосереджується на самому додатку, а не на платформозалежних аспектах. [2]

Складність розробки - веб-додатки полегшують процес програмування, оскільки вони не потребують роботи з апаратними компонентами, такими як ядро, процесор або відеокарта. Розробники можуть зосередитись на функціоналі додатку без необхідності вдаватися до деталей апаратного забезпечення. Легкість поширення - відмінною рисою веб-додатків є їх гнучкість у поширенні. Для запуску веб-додатків не потрібно отримувати схвалення або ліцензії від певних платформ. Вони можуть бути розміщені на серверах і надаватися користувачам як сервіс. Висока мобільність - веб-додатки можуть бути легко адаптовані для мобільних пристроїв, що робить їх доступними для користувачів з різних типів пристроїв, включаючи смартфони та планшети. Це дає можливість користувачам отримувати доступ до додатків незалежно від їх місця перебування. [25]

Веб-додатки стали невід'ємною частиною нашого сучасного світу, надаючи зручний та доступний спосіб взаємодії з інформацією та сервісами. Вони дозволяють нам здійснювати покупки, отримувати освіту, спілкуватися з іншими людьми та здійснювати безліч інших дій, просто заходячи на веб-сторінку через наш браузер. [2]

Веб-додатки значно змінили наше щоденне життя, проникнувши у різні сфери діяльності, від електронної комерції до надання державних послуг в онлайн-режимі.

Проблема підвищення ефективності в адмініструванні навчального процесу за рахунок впровадження WEB-додатків для інформаційного забезпечення організаційної структури університету є актуальною.

Виконаний аналіз специфіки і характеристик роботи деканатів дозволяє виділити цілий ряд переваг за рахунок впровадження WEB-додатків. Серед таких переваг можна визначити:

- **Ефективне керування студентськими даними**: Програмне забезпечення дозволить деканату легко зберігати та оновлювати інформацію про студентів, включаючи особисті дані, академічні

досягнення, групи та розклади занять. Це дозволить швидко знаходити потрібну інформацію та забезпечити точність та цілісність даних.

- **Забезпечення точності та надійності даних**: Автоматизована система зменшує ризик людських помилок та неуважності, що можуть виникнути під час ручного обліку даних. Програмне забезпечення гарантує консистентність та надійність даних, а також забезпечує можливість резервного копіювання та відновлення інформації.
- **Зручний доступ до інформації**: Використання веб-додатків дозволяє забезпечити централізований та зручний доступ до даних для деканатських працівників, викладачів та студентів. Це спрощує обмін інформацією, зменшує необхідність у паперовій документації та полегшує спільну роботу
- **Розширення можливостей розвитку**: Використання програмного забезпечення для автоматизації деканату створює основу для подальшого розвитку та вдосконалення системи. За допомогою модульної архітектури, програмне забезпечення може бути легко розширене та адаптоване під нові потреби та вимоги деканату. Це дає можливість впроваджувати нові функції, вдосконалювати існуючі процеси та адаптуватися до змін у вимогах та регуляторному середовищі.
- **Забезпечення безпеки даних**: При автоматизації деканату велика увага приділяється захисту конфіденційної інформації та дотриманню вимог щодо захисту даних. Використання програмного забезпечення дозволяє встановити механізми контролю доступу, резервне копіювання та шифрування даних, забезпечуючи конфіденційність, цілісність та доступність інформації.

1.2 **Особливості роботи веб-додатків на основі MVC**

Розробка веб-додатків з використанням архітектури Model-View-Controller (MVC) відіграє важливу роль у сфері програмного забезпечення. Ця архітектурна парадигма розділяє додаток на три компоненти: модель, представлення та контролер, що сприяє покращенню розподілу обов'язків та полегшує процес розробки. [6]

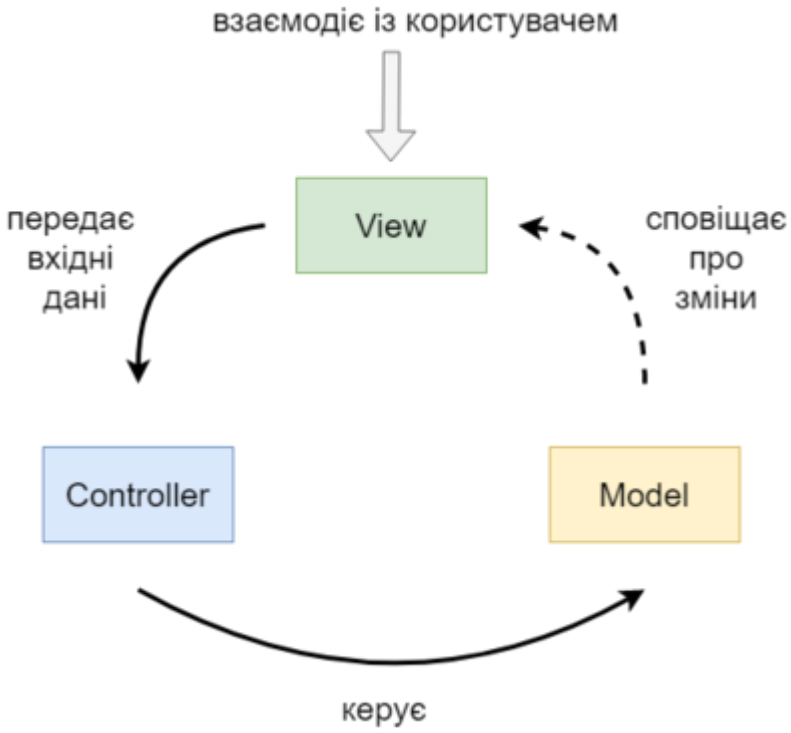


Рис. 1.2.1 – Архітектура MVC

Модель відповідає за управління даними та бізнес-логікою додатку. Вона забезпечує доступ до даних з різних джерел, таких як база даних, зовнішні служби або файлова система. Модель забезпечує обробку даних та здійснює необхідні операції для підтримки функціональності додатку. [6]

Представлення відповідає за візуалізацію даних та взаємодію з користувачем. Воно відображає дані, отримані від моделі, у зручному для користувача форматі. Представлення може бути представлено як HTML-

шаблон, який містить відповідний вигляд сторінки або інтерфейсу користувача. [25]

Контролер відповідає за обробку запитів користувача та керування взаємодією між моделлю та представленням. Він приймає запити від користувача, обробляє їх та виконує необхідні дії, включаючи взаємодію з моделлю та відправку даних до представлення. Контролер забезпечує координацію всіх компонентів додатку. [20]

Основна перевага використання архітектури MVC полягає в розділенні відповідальностей між різними компонентами, що сприяє підтримці коду, його розширенню та перевикористанню. [25]

Іншою важливою особливістю є можливість розширення та перевикористання коду. Завдяки розділенню відповідальностей, ви можете впроваджувати зміни у будь-якому компоненті без впливу на інші. Це сприяє швидкому розвитку додатку та полегшує його підтримку у майбутньому. [25]

Додатково, веб-додатки, розроблені з використанням архітектури MVC, мають велику масштабованість. Завдяки розділенню функціональності на компоненти, можливість додавання нових функцій значно полегшується, модифікувати існуючі та масштабувати додаток у відповідності до зростаючих потреб. [25]

1.1.1 **Види веб-додатків**

Веб-додатки варто класифікувати по їхньому типу на набору функцій які вони надають. [29]

Статичний веб-додаток. Першим типом веб-додатків, доступних в Інтернеті, були статичні веб-додатки, створені за допомогою HTML і CSS для полегшення відображення важливого вмісту та інформації. Зазвичай це найпростіша веб-програма, оскільки вона відображає лише обмежений вміст і не є гнучкою. Як правило, такі програми не мають функцій персоналізації та вносять зміни після повного завантаження сторінки. Хоча він дозволяє використовувати анімовані об’єкти, такі як GIF-файли, відео тощо, змінити

вміст статичної веб-програми нелегко, оскільки вам потрібно завантажити, змінити та повернути код HTML. Тому ця програма найкраще підходить для компаній-розробників програмного забезпечення та професійних веб-майстрів. Деякі з найкращих прикладів включають цифрові резюме та сторінки захоплення потенційних клієнтів у маркетингу. [29]

Динамічний веб-додаток. Динамічні веб-додатки надають дані в реальному часі на основі запитів користувачів і тому вважаються одними з найкращих типів веб-програм. Вони збільшують технічну складність порівняно зі статичними веб-додатками. Існує багато інтерактивних елементів і методів, щоб привернути увагу клієнта до послуг і продуктів, які пропонує веб-додаток. Такі веб-додатки використовують бази даних для зберігання всіх приватних і загальнодоступних даних, які відображаються на веб-сайті. Зазвичай вони мають панель адміністратора для керування внутрішніми та зовнішніми частинами, дозволяють адміністраторам змінювати вміст і включати різноманітні інтерактивні компоненти у веб-програму. [29]

Веб-додаток для електронної комерції. Коли веб-програма рекламує продукт або послугу безпосередньо вашим потенційним клієнтам, її прийнято назвати веб-програмою електронної комерції, не на відміну від онлайн-магазину. Багато основних функцій веб-програми для електронної комерції включають додавання нових продуктів, видалення застарілих і старих продуктів, керування платежами, спрощення електронних платежів і зручні інтерфейси. Для виконання всіх цих завдань дуже потрібна ефективна панель керування. Професійні розробники сайтів можуть налаштувати такі програми для зручності користувачів. Деякі з найпоширеніших прикладів веб-додатків для електронної комерції включають Flipkart, Amazon, Ajio та багато інших. [29; 30]

Односторінкова програма, також відома як SPA, — це динамічна веб-програма, яка не потребує перезавантаження браузера та функціонує як єдина сутність веб-програми. Ці веб-програми є швидкими та динамічними, оскільки вони забезпечують дотримання всіх ділових і технічних політик у браузері

клієнта. Процес розробки та впровадження SPA простий і швидкий. Оскільки спілкування відбувається в асинхронному режимі навігації, процес обробки запитів і відповідей користувача відбувається швидше. Крім того, будь-який тип веб-додатку SPA можна переналаштувати для досягнення бажаного результату. Однак головна проблема SPA полягає в тому, що вони не відповідають інструкціям SEO. Найкращими прикладами односторінкових веб-додатків є Netflix, Twitter і Gmail. [29; 2]

Додаток порталу. Веб-додатки порталу надають певним типам користувачів єдину точку доступу до важливих даних. Це веб-програма, яка надає доступ до різних розділів на головній сторінці. Портали є найкращим варіантом для організацій і компаній, у яких є потреба створювати індивідуальні інтерфейси відповідно до потреб своєї цільової аудиторії. Доступ доступний лише зареєстрованим користувачам, і після входу в систему постачальник послуг може відстежувати дії користувача. [2; 25]

1.2 Вразливості веб-додатків

Зі збільшенням популярності веб додатків також збільшується кількість людей які хочуть отримати доступ до цінних даних. З'являються загрози різноманітних вразливостей, які можуть бути використані зловмисниками для незаконного доступу до системи та виконання шкідливих дій. Зрозуміння та усунення цих вразливостей є критично важливим завданням для забезпечення безпеки та конфіденційності веб-додатків. [2]

Перш за все, важливо зрозуміти, що вразливості веб-додатків можуть існувати на різних рівнях їх архітектури та функціональності. Наприклад, вразливості можуть виникати на рівні введення даних, аутентифікації та авторизації, конфіденційності, управління сесіями, обробки файлів, керування даними та багато інших. Кожна з цих вразливостей може мати свої унікальні наслідки та потенційні загрози, які ми детально розглянемо. [2]

Однією з найпоширеніших вразливостей веб-додатків є вразливість введення даних. Недостатня перевірка та фільтрація введених користувачем даних може призвести до таких атак, як SQL-ін'єкція та XSS (міжсайтовий скриптинг). SQL-ін'єкція дозволяє зловмисникам виконувати шкідливі SQL-запити до бази даних, що може призвести до розголошення конфіденційної інформації або навіть зміни структури бази даних. XSS атаки дозволяють вбудовувати шкідливий скрипт, що дозволяє зловмисникам отримувати доступ до сесій користувачів або крадіжку конфіденційних даних. [2]

Іншою важливою вразливістю є недостатня аутентифікація та авторизація. Неналежно налаштовані механізми аутентифікації можуть дозволити зловмисникам незаконно отримати доступ до системи, використовуючи крадені або підбирані паролі. Недостатні обмеження прав доступу (авторизація) можуть дозволити несанкціонованим користувачам отримати доступ до конфіденційної інформації або змінювати дані в системі. [20]

Вразливості на рівні конфіденційності також становлять серйозну загрозу. Недостатнє шифрування або незахищений обмін інформацією можуть дозволити зловмисникам перехоплювати та розшифровувати конфіденційні дані, такі як особисті дані користувачів, кредитні картки або медичні записи. Це може призвести до серйозного порушення приватності та витоку конфіденційної інформації. [21]

Іншою вразливістю веб-додатків є недостатня безпека сесій. Неналежно налаштовані механізми управління сесіями можуть дозволити зловмисникам викрадати аутентифікаційні дані та використовувати їх для незаконного доступу до системи. Це може включати в себе використання підроблених сесійних ідентифікаторів або атаки перехоплення сесій. [20; 21]

Помилки в обробці файлів також можуть призвести до вразливостей веб-додатків. Недостатнє перевірка та фільтрація завантажених файлів можуть дозволити зловмисникам завантажувати та виконувати шкідливий код на

сервері, що призводить до компрометації системи або розповсюдження шкідливих програм. [20; 21]

Зазначені вразливості лише невелика частина широкого спектру можливих загроз для веб-додатків. Залежно від конкретного контексту і архітектури додатка, можуть існувати інші вразливості, такі як уразливості у сторонніх компонентах, використання старих версій програмного забезпечення з відомими вразливостями, недостатня безпека мережевих протоколів тощо. [20; 21]

З метою зменшення ризиків, пов'язаних з вразливостями веб-додатків, необхідно вживати широкий спектр заходів захисту. До них входять регулярні оновлення та патчі для програмного забезпечення, використання безпечних механізмів аутентифікації та авторизації, належна перевірка та фільтрація введених даних, захищений обмін інформацією шляхом шифрування та використання відповідних криптографічних протоколів, імплементація механізмів обмеження прав доступу та встановлення політик безпеки, а також проведення регулярних аудитів та тестувань безпеки. [20; 21]

Один із найефективніших підходів до захисту веб-додатків - це розробка з урахуванням принципів безпеки від самого початку. Це включає в себе використання безпечних фреймворків та бібліотек, які мають вбудовані механізми захисту, такі як перевірка типів даних, захист від XSS та CSRF атак, автоматична перевірка валідності даних, контроль доступу та інші. [20; 21]

Крім того, варто звернути увагу на забезпечення безпеки веб-сервера та інфраструктури, на якій працює веб-додаток. Це охоплює налаштування правильних прав доступу, моніторинг системи на виявлення потенційних атак, використання захищених мережевих протоколів, налаштування брандмауерів та регулярне оновлення всього програмного забезпечення. [20; 21]

1.3 Аналіз призначення та застосування веб додатків

Веб-додаток — це програмний інструмент який надає доступ до різноманітних функцій і послуг через Інтернет. Вони мають широкий спектр використання та застосування, а також дозволяють користувачам виконувати різні завдання з будь-якого місця, де є доступ до Інтернету. [2]

Однією з головних цілей веб-додатків є надання користувачам можливості взаємодіяти з веб-службами та зберігати, обробляти та передавати дані через Інтернет. Наприклад, веб-програми можуть допомогти вам керувати електронною поштою, зберігати та ділитися документами, створювати та редагувати файли в хмарному сховищі, робити покупки в Інтернеті, здійснювати банківські операції тощо. [2]

Веб-додатки також використовуються для розробки та представлення інформаційних ресурсів у вигляді веб-сайтів, блогів, новинних порталів, інтернет-магазинів, соціальних мереж та інших онлайн-платформ. Вони дозволяють створити інтерактивний та зручний веб-інтерфейс для користувачів, що полегшує доступ до інформації та взаємодію з контентом. [2]

Крім того, веб-додатки також можна використовувати в бізнесі для автоматизації бізнес-процесів, запису та аналізу даних, створення інструментів для управління проектами, спілкування та командної співпраці, забезпечення безпеки даних тощо. Веб-додаток можна налаштувати відповідно до конкретних потреб організації та допомагає впроваджувати ефективні та зручні рішення для покращення робочого процесу. [2; 29; 30]

Веб-додатки також поширені в ігровій індустрії, що дозволяє користувачам грати в онлайн-ігри без встановлення спеціального програмного забезпечення. Це дозволяє гравцям отримувати доступ до розваг з будь-якого пристрою, підключеного до Інтернету. [29; 30]

Нарешті, веб-додатки широко використовуються в сфері розваг і культури, і вони надають можливості для потокової передачі музики, фільмів,

телешоу, відеоігор, а також для організації віртуальних заходів, виставок і конференцій.

Взагалі кажучи, веб-додатки є важливими інструментами, які допомагають людям оптимізувати їхні робочі процеси, забезпечують доступ до інформації та послуг, покращують спілкування та співпрацю, а також надають можливості для розваг і навчання через Інтернет. Кількість та різноманітність веб-додатків обчислити майже неможливо, і їхнє значення в сучасному світі стає невід'ємною частиною нашого повсякденного життя. Завдяки веб-додаткам ми можемо зручно і ефективно виконувати різноманітні завдання, спілкуватися, навчатися, розважатися та працювати з будь-якого місця, де є доступ до Інтернету. [2]

1.4 Системи автоматизації освітнього процесу

На основі дослідження систем автоматизації навчального процесу та їх аналізу було підвищено рівень розуміння їх ролі у покращенні різних аспектів освітнього середовища. В освіті постійне прагнення до ефективності та покращення результатів призводить до інноваційних рішень. Одним з таких рішень є система автоматизації навчального процесу. Ці системи використовують технології для автоматизації та спрощення різних адміністративних, навчальних та організаційних завдань у навчальних закладах. Зменшуючи ручну працю та впорядковуючи процеси, системи автоматизації освітніх процесів мають на меті підвищити продуктивність, сприяти прийняттю рішень на основі даних і, зрештою, покращити загальний освітній досвід. [43; 44; 45]

Ключові компоненти системи автоматизації навчального процесу:

- Управління інформацією про студентів. Системи, що використовуються для автоматизації освітніх процесів, часто включають можливості управління інформацією про студентів. Це включає такі завдання, як реєстрація, облік, відстеження академічного прогресу, створення

транскриптів та управління відвідуванням. Автоматизувавши ці процеси, викладачі та адміністратори можуть витрачати більше часу та ресурсів на залучення студентів та персоналізацію навчання. [43; 44; 45]

- Навчальні програми та плани уроків**.** Автоматизовані системи дозволяють освітянам ефективно розробляти та організовувати зміст курсів і плани уроків. Ці системи часто надають шаблони, інструменти для співпраці та інтеграції з навчальними ресурсами, що полегшує вчителям створення цікавих та інтерактивних навчальних матеріалів. Автоматизація також може допомогти узгодити навчальну програму зі стандартами та контролювати її виконання в усіх класах і предметах. [43; 44; 45]
- Оцінювання та виставлення оцінок. Система автоматизації навчального процесу спрощує процес оцінювання та виставлення оцінок. Вони пропонують такі функції, як онлайн-рейтинги, автоматизоване виставлення балів та миттєвий зворотній зв'язок. Ці системи допомагають своєчасно надавати результати оцінювання, дозволяючи викладачам виявляти прогалини в навчанні та відповідно коригувати навчання. [43; 44; 45]
- Комунікація та співпраця. Ефективна комунікація та співпраця мають вирішальне значення в освітніх установах. Автоматизована система забезпечує платформу для безперешкодної комунікації між вчителями, учнями, батьками та адміністрацією. Вони пропонують такі функції, як сповіщення електронною поштою, дискусійні форуми та спільні календарі, що сприяють зміцненню стосунків і забезпечують своєчасне поширення важливої інформації. [43; 44; 45]

Проводячи огляд систем автоматизації освітнього процесу було визначено ключові переваги:

- Оптимізація часу та ресурсів. Системи автоматизації навчального процесу заощаджують цінний час викладачів та адміністраторів, автоматизуючи рутинні адміністративні завдання. Це дозволяє їм зосередитися

на більш важливих видах діяльності, таких як підтримка студентів, планування уроків та професійний розвиток. [43; 44; 45]

- Удосконалене управління даними. Автоматизовані системи збирають, аналізують і зберігають великі обсяги освітніх даних. Ці дані можна використовувати для виявлення тенденцій, відстеження успішності учнів та прийняття рішень на основі фактичних даних. Централізувавши управління даними, навчальні заклади можуть отримати цінну інформацію про свою діяльність і впроваджувати вдосконалення на основі даних. [43; 44; 45]
- Можливості персоналізованого навчання. Автоматизовані системи можуть підтримувати персоналізоване навчання, надаючи вчителям інструменти для диференціації уроків на основі індивідуальних потреб учнів. Аналізуючи дані про учнів, ці системи можуть генерувати індивідуальні навчальні траєкторії, адаптивне оцінювання та цілеспрямовані втручання, сприяючи більш персоналізованому освітньому досвіду. [43; 44; 45]

1.5 Аналіз аналогів

Для того щоб врахувати проблеми, недоліки а також переваги існуючих рішень, було проведено дослідження в якому було проаналізовано та проведено порівняння на основі якого було складено таблицю порівняння та наведено недоліки та переваги кожного з представлених продуктів. Першим продуктом був сервіс «Портал автоматизованої системи управління навчальним процесом ДУТ». [45]

1. Портал автоматизованої системи управління навчальним процесом ДУТ:

   Переваги:

   1. Даний веб-сайт працює швидко, забезпечуючи комфортне користування його відвідувачам.
   2. Зручний інтерфейс для перегляду розкладу для користувачів.

   Недоліки:

1. Помилка сервера при виборі журналу відвідуваності.
2. Немає можливості обліку успішності студентів.
3. Немає функціональності обліку відомостей груп.

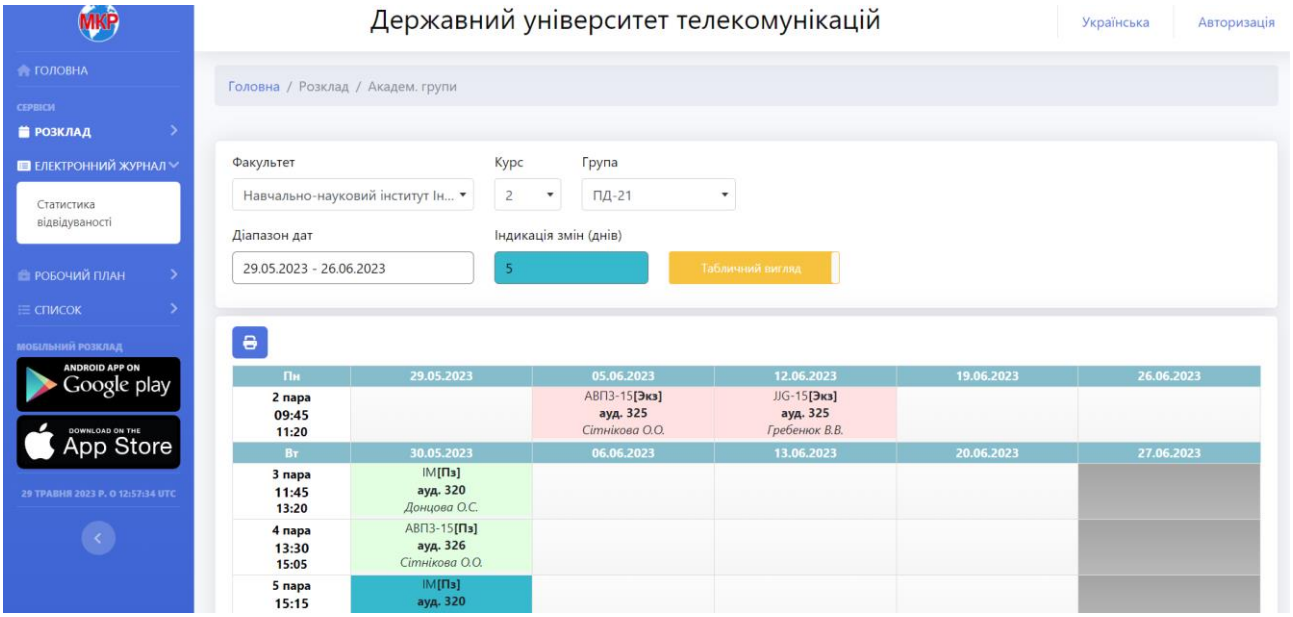


Рис. 1.5.1 Інтерфейс системи «Порталу автоматизованої системи управління навчальним процесом ДУТ»

2. Easy Grade Pro:

   Переваги:

   - Швидкість роботи: Програмне забезпечення Easy Grade Pro працює досить швидко, забезпечуючи оперативність введення оцінок та обробку даних.
   - Локальне збереження: Easy Grade Pro може працювати в режимі офлайн, що дозволяє зберігати та обробляти дані без підключення до Інтернету. [43]

   Недоліки:

   - Обмежений функціонал: Easy Grade Pro може бути обмеженим у функціоналі порівняно з іншими програмами. Наприклад, він може не мати розширених можливостей для планування розкладу або інтеграції з іншими системами.

- Відсутність регулярних оновлень: Easy Grade Pro може не надавати регулярних оновлень та підтримку, що може призвести до відсутності нових функцій та виправлення помилок.
- Обмежена підтримка: У порівнянні з іншими програмами для управління оцінками, Easy Grade Pro може мати обмежену підтримку або обмежений доступ до веб-ресурсів та допомоги з боку розробника.
- Застарілий інтерфейс [43]

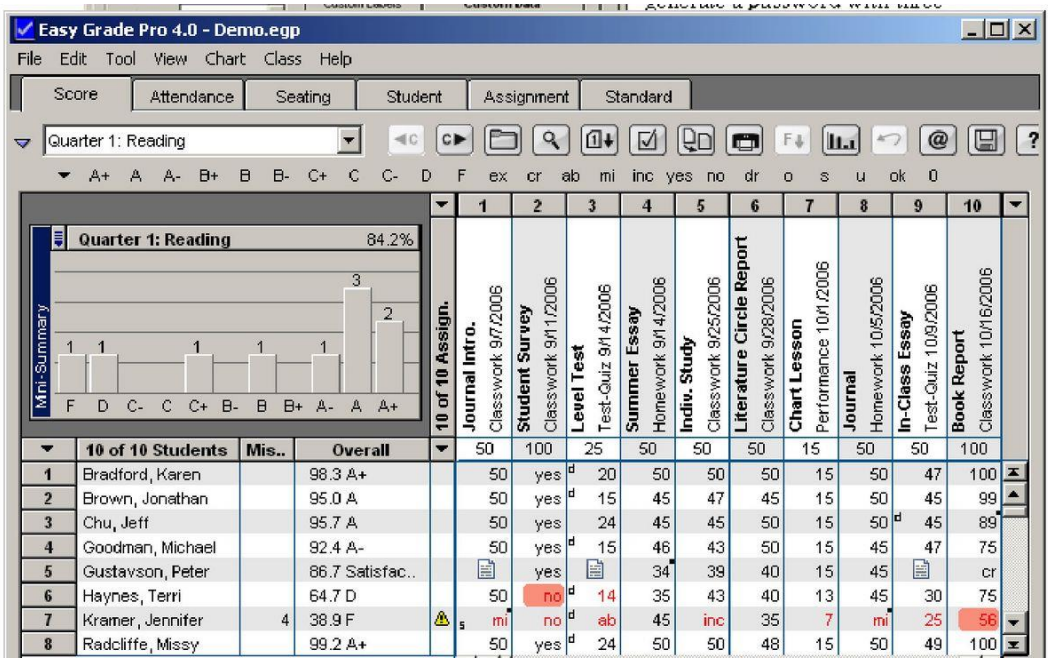


Рис. 1.5.2 Інтерфейс Easy Grade Pro

3. PeopleSoft Campus Solutions від Oracle.

PeopleSoft Campus Solutions - це комплексний набір додатків у складі програмного комплексу PeopleSoft, спеціально розроблений для задоволення потреб ВНЗ. Він містить низку інтегрованих модулів, які автоматизують і спрощують різні адміністративні процеси в університетах і коледжах. [44]

Переваги:

- Комплексне рішення: PeopleSoft Campus Solution охоплює широкий спектр адміністративних процесів, забезпечуючи інтегроване рішення, яке впорядковує операції та сприяє підвищенню ефективності. [44]

- Масштабованість та кастомізація: Вона також пропонує широкі можливості налаштування, що дозволяє установам адаптувати систему до своїх унікальних вимог і процесів. [44]
- Підтримка PeopleSoft існує на ринку вже багато років і широко використовується багатьма вищими навчальними закладами по всьому світу. Система підтримується Oracle, авторитетним постачальником програмного забезпечення, який забезпечує постійне обслуговування, оновлення та підтримку для забезпечення надійності та безпеки системи. [44]

Недоліки:

- Складність впровадження: Впровадження рішень PeopleSoft для університетів може бути складним і тривалим процесом. Він часто вимагає значного планування, конфігурації, міграції даних та навчання, що може створити проблеми для установ, особливо тих, що мають обмежені ресурси або досвід. [44]
- Вартість: Витрати на ліцензування, впровадження та обслуговування, пов'язані з PeopleSoft Campus Solutions, можуть бути значними, особливо для невеликих установ. Бюджетні обмеження можуть обмежити доступність програмного забезпечення для деяких організацій. [44]
- Користувацький інтерфейс та досвід користувачів: Хоча PeopleSoft протягом багатьох років докладає зусиль для покращення інтерфейсу користувача та зручності роботи, деякі користувачі все ще вважають, що система має круту криву навчання та дещо застарілий або складний інтерфейс. Це може вплинути на адаптацію та задоволеність користувачів. [44]
- Обмеження гнучкості: Хоча PeopleSoft пропонує можливості кастомізації, рівень гнучкості може бути не таким широким, як у деяких інших програмних рішень. Установи з дуже унікальними або

складними процесами можуть зіткнутися з обмеженнями в повному пристосуванні програмного забезпечення до їхніх конкретних вимог. [44]

- Проблеми інтеграції: Інтеграція PeopleSoft Campus Solutions з існуючими системами та сторонніми додатками може бути складним завданням. Забезпечення безперебійного потоку даних і підтримка інтеграції може вимагати додаткових ресурсів і досвіду. [44]

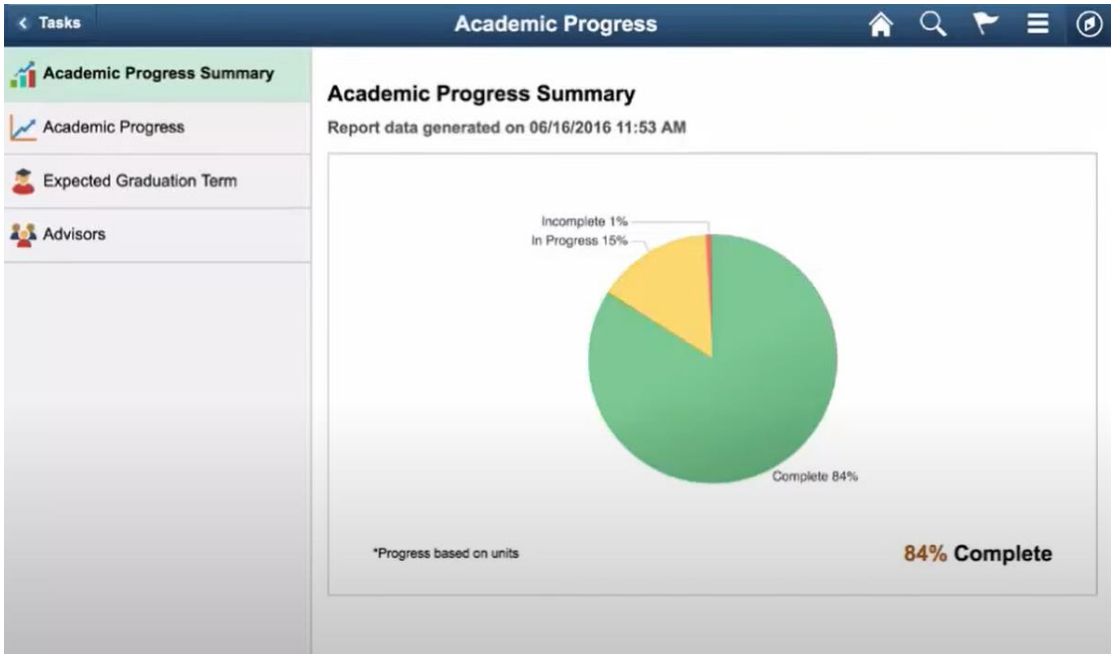


Рис. 1.5.3 Інтерфейс PeopleSoft Campus Solution

Дослідження програмного забезпечення для автоматизації управління навчальним процесом є важливим завданням у сучасному освітньому середовищі. Рішення, які вибираються для впровадження в установу, можуть значно впливати на ефективність, продуктивність та зручність управління освітнім процесом.

Було проведено порівняльний аналіз чотирьох систем управління навчальним процесом: "Актуальна АСУ", "Easy Grade Pro", "PeopleSoft Campus Solution" та розроблений додаток.

Для більш зрозумілого порівняння цих систем, наведена нижче таблиця, яка показує деякі ключові параметри.

Таблиця надає загальний огляд функціональності кожної системи та допоможе прийняти обґрунтоване рішення щодо вибору системи управління навчальним процесом, з урахуванням потреб та вимог конкретного навчального закладу.

Таблиця 1.5.1 – Таблиця порівняння аналогів

Назва	Актуальна АСУ	Easy Grade Pro	PeopleSoft Campus Solution	**Розроблений додаток**
Кросплатформеність	+	-	+	+
Українська локалізація	+	-	-	+
Облік успішності студентів	-	+	+	+
Облік відомостей груп	-	-	-	+
Облік контактних даних студентів та викладачів	-	-	+	+
Облік відвідуваності студентів	-	-	+	+
Управління розкладом	+	-	-	-

2 ДОСЛІДЖЕННЯ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Мова програмування Java

Java, розроблена Джеймсом Гослінгом та його командою в Sun Microsystems (зараз Oracle Corporation), з'явилася в середині 1990-х років як революційна мова програмування. Вона як і будь-яка мова програмування була розроблена щоб вирішити проблеми, з якими стикалися розробники при створенні надійного, незалежного від платформи програмного забезпечення. [1]

Своїм корінням Java сягає мови програмування Oak, розробленої Гослінгом та його командою на початку 1990-х років. Черпаючи під впливом таких мов, як C та C++, Java успадкувала їхній синтаксис, додавши при цьому вдосконалення та нові можливості. Спочатку мова була розроблена для побутової електроніки, але її потенціал для веб-розробки став очевидним з появою Інтернету. [1; 17]

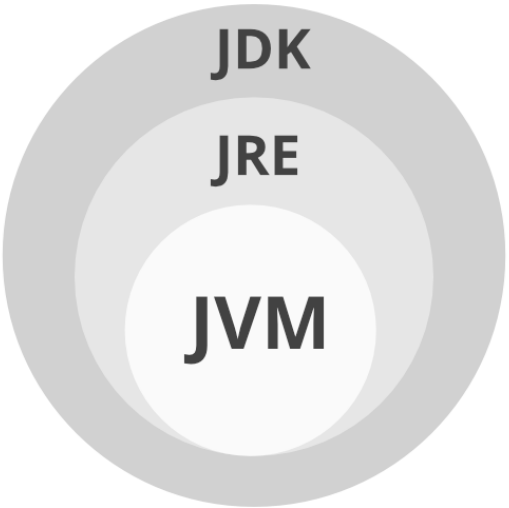


Рис. 2.1.1 – Структура JDK

З моменту свого створення Java зазнала низки значних змін. Мова еволюціонувала через різні версії, з кожною ітерацією додаючи нові функції, підвищуючи продуктивність та безпеку. Серед найважливіших віх - поява Java

Development Kit (JDK), віртуальної машини Java (JVM) та платформи Java Standard Edition (Java SE). [5; 7]

JDK, або Java Development Kit, - це комплект для розробки програмного забезпечення, що надається корпорацією Oracle. Він включає в себе набір інструментів та утиліт, які розробники використовують для створення, компіляції, налагодження та розгортання Java-додатків. JDK складається з компілятора Java (javac), який перетворює вихідний код Java у байт-код, та інших інструментів, таких як віртуальна машина Java (JVM), відладчик, генератор документації та різні бібліотеки. На додаток до основних інструментів розробки, JDK також включає API-інтерфейси розробки та фреймворки, які полегшують створення Java-додатків. [1; 27]

JRE, або середовище виконання Java, - це програмний пакет, який надає необхідні компоненти для запуску Java-додатків. Він включає JVM, бібліотеки класів та інші файли, необхідні для виконання байт-коду Java. JRE не містить інструментів розробки (таких як компілятор), які містяться в JDK. Коли ви встановлюєте JRE у вашій системі, ви отримуєте можливість запускати програми на Java, але ви не можете розробляти або компілювати нові програми без JDK. [40]

JVM, або віртуальна машина Java, є важливим компонентом платформи Java. Вона відповідає за виконання байт-коду Java, який є скомпільованою формою вихідного коду Java. JVM забезпечує середовище, яке абстрагується від базового обладнання та операційної системи, дозволяючи програмам Java узгоджено працювати на різних платформах. Вона інтерпретує байт-код або, в деяких випадках, компілює його в машинний код для ефективного виконання. JVM також займається управлінням пам'яттю, збиранням сміття та обробкою винятків у програмі на Java. [40]

Мова Java має синтаксис, подібний до мов програмування C та C++, що робить її доступною для розробників, знайомих з цими мовами. Її структура базується на класах, об'єктах та методах, що сприяє створенню модульного та багаторазового коду. Синтаксис Java забезпечує дотримання суворих правил

організації коду, таких як використання пакетів, класів та модифікаторів доступу. [26]

Java відома своєю сильною підтримкою об'єктно-орієнтованої парадигми програмування. Вона втілює такі ключові принципи, як інкапсуляція, успадкування та поліморфізм. За допомогою класів та об'єктів розробники можуть моделювати реальні сутності та взаємодії, що полегшує повторне використання коду, його супровід та розширення. [26]

У Java реалізовано автоматичне керування пам'яттю за допомогою механізму збору сміття. Ця функція звільняє розробників від тягаря ручного розподілу пам'яті, зменшуючи ризик витоку пам'яті та підвищуючи загальну стабільність програми. Віртуальна машина Java (JVM) керує розподілом і звільненням пам'яті, оптимізуючи продуктивність під час виконання. [26; 28]

Обробка винятків є важливим аспектом надійності Java. Мова надає структурований механізм для перехоплення та обробки винятків, запобігаючи раптовому завершенню програми. Використовуючи блоки перехоплення винятків, розробники можуть ефективно обробляти помилки та вживати відповідних заходів, підвищуючи надійність програми. [26]

Однією з визначальних особливостей Java є її незалежність від платформи. Програми на Java компілюються у байт-код, який можна виконати на будь-якій системі зі встановленою віртуальною машиною Java (JVM). Така переносимість відіграє важливу роль у полегшенні крос-платформної розробки, позбавляючи розробників від необхідності переписувати код для різних операційних систем. [40]

Java надає пріоритет надійності та безпеці, що робить її популярним вибором для розробки захищених додатків. Мова включає в себе такі функції, як сильна перевірка типів, обробка винятків та управління пам'яттю, які сприяють загальній стабільності та надійності програм на Java. Застосовуючи суворі правила та забезпечуючи комплексну систему безпеки, Java допомагає розробникам створювати додатки з вбудованим захистом від поширених вразливостей, таких як переповнення буферів та пошкодження пам'яті. [26; 40]

Java може похвалитися великою і всеосяжною екосистемою бібліотек, яка пропонує розробникам широкий спектр інструментів, фреймворків та API для прискорення процесу розробки. Стандартна бібліотека Java надає багату колекцію класів та утиліт для виконання найпоширеніших завдань програмування, починаючи від операцій вводу/виводу і закінчуючи мережею та підключенням до баз даних. Крім того, сторонні бібліотеки та фреймворки, такі як Spring, Hibernate та Apache Commons, ще більше розширюють можливості Java і дозволяють розробникам використовувати вже існуючі рішення для різних вимог додатків. [26; 40]

Java відіграє важливу роль у веб-розробці, особливо у серверних додатках. Такі технології, як JavaServer Pages (JSP) і Java Servlets дозволяють створювати динамічні та інтерактивні веб-сторінки. Надійність, масштабованість і функції безпеки мови Java роблять її добре придатною для створення веб-додатків корпоративного рівня. [21]

Крос-платформенна сумісність Java та багаті фреймворки графічного інтерфейсу користувача (GUI), такі як Swing та JavaFX, зробили її популярним вибором для розробки десктопних додатків. Можливість створювати незалежні від платформи десктопні додатки була особливо вигідною для постачальників програмного забезпечення, орієнтованих на різні операційні системи. [21; 1; 28]

Java широко використовується у розробці мобільних додатків, насамперед завдяки її інтеграції з платформою Android. Android SDK використовує Java як основну мову для створення додатків для Android, що дозволяє розробникам використовувати розгалужену екосистему Java та писати додатки, які працюють на великій базі користувачів. [1; 40]

2.2 Поняття фреймворку

У широкій області розробки програмного забезпечення фреймворки перетворилися на важливі інструменти, які визначають спосіб створення додатків. Ці фреймворки забезпечують фундаментальну та структуровану

методологію розробки, надаючи розробникам заздалегідь визначений набір правил, бібліотек та компонентів, які спрощують створення програмних рішень. У цій статті розглядається визначення фреймворку та його величезне значення у розробці програмного забезпечення. Підкреслюється їхня роль у підвищенні продуктивності, масштабованості та якості коду. [22]

Визначення фреймворку. У світі розробки програмного забезпечення фреймворк можна визначити як структурований і багаторазовий набір інструментів, бібліотек і рекомендацій, які допомагають розробникам ефективно створювати додатки. Він забезпечує заздалегідь визначену структуру і набір стандартизованих компонентів для полегшення процесу розробки. [22]

Фреймворки слугують каркасом, на якому розробники створюють свої додатки, абстрагуючись від складних низькорівневих деталей і дозволяючи їм зосередитися на основній функціональності свого програмного забезпечення. [22; 23]

Фреймворки відіграють центральну роль у сучасній розробці програмного забезпечення з кількох вагомих причин:

- **Оптимізація процесу розробки**. Фреймворки забезпечують структурований підхід до розробки програмного забезпечення, надаючи заздалегідь визначені шаблони та методи. Вони направляють розробників через весь процес, забезпечуючи узгодженість і скорочуючи час, необхідний для прийняття рішень. Усуваючи необхідність вигадувати «велосипед» для загальних завдань, фреймворки дозволяють розробникам зосередитися на вирішенні унікальних бізнес-завдань, а не турбуватися про повторювані деталі реалізації. [22; 23]
- **Підвищення продуктивності та ефективності.** Фреймворки призначені для максимізації продуктивності розробників, заохочуючи повторне використання коду, модульність та автоматизацію. Вони

надають велику кількість готових компонентів, бібліотек та API для прискорення виконання завдань розробки. Використовуючи ці ресурси, розробники можуть прискорити реалізацію функцій і можливостей, скоротити час виходу на ринок і підвищити загальну ефективність проекту. [22; 23]

- **Масштабованість та підтримка.** Фреймворки формують основу для масштабованих і підтримуваних додатків. Надаючи архітектурні шаблони та найкращі практики, вони допомагають розробникам створювати модульні, слабко пов'язані системи, які можуть легко адаптуватися до майбутніх змін та зростання. Фреймворки сприяють розподілу завдань, забезпечують дотримання принципів чистоти коду та полегшують інтеграцію нових функцій, роблячи додатки більш надійними та зручними для обслуговування з часом. [22; 23]
- **Абстракція та багаторазове використання.** Фреймворки забезпечують абстракції, які захищають розробників від низькорівневих складнощів і дозволяють їм працювати на більш високих рівнях абстракції. Ця абстракція дозволяє розробникам зосередитися на бізнес-логіці, не турбуючись про складні деталі реалізації. Крім того, фреймворки сприяють повторному використанню коду, надаючи репозиторій готових компонентів і модулів. Таке повторне використання не лише економить час розробки, але й покращує узгодженість коду та зменшує ймовірність помилок. [22; 23]
- **Стандартизація та кращі практики.** Фреймворк втілює найкращі практики та встановлені галузеві стандарти. Вони поєднують в собі багаторічний досвід і знання спільноти, щоб гарантувати, що розробники дотримуються найкращих практик. Дотримання стандартів покращує якість кодової бази та полегшує сумісність між різними компонентами та системами. [22; 23]

2.2.1 Які існують фреймворки?

Фреймворки можуть бути специфічними для певних мов програмування. Наприклад, є фреймворки, розроблені для Python, такі як Django або Flask, є фреймворки, розроблені для Ruby (Ruby on Rails), JavaScript (Angular, React, Node.js), PHP (Laravel, Symfony) та багатьох інших мов програмування. [22; 23]

Фреймворки для веб-розробки надають інструменти та шаблони для створення веб-додатків. Вони часто містять компоненти для маршрутизації, керування базами даних, візуалізації сторінок, автентифікації та інших важливих аспектів веб-розробки. [22; 23]

Front-end фреймворки призначені для полегшення розробки користувацьких інтерфейсів і підвищення інтерактивності веб-додатків. Вони зосереджені переважно на клієнтській стороні розробки додатків, включаючи технології HTML, CSS та JavaScript. Ці фреймворки надають розробникам структурований підхід, багаторазові компоненти та ефективні механізми зв'язування даних для створення динамічних та адаптивних користувацьких інтерфейсів. [22; 23] Було проведено аналіз найпопулярніших фреймворків для Front-end частини додатків:

1. **Angular**, розроблений Google, є потужним фреймворком, який дозволяє розробникам створювати односторінкові додатки (SPA). Він дотримується компонентної архітектури і використовує TypeScript (статично типізовану підмножину JavaScript) для створення багатофункціональних додатків. Angular надає такі функції, як двостороннє зв'язування даних, ін'єкція залежностей та потужні можливості маршрутизації. [22] Завдяки багатому набору бібліотек, всеосяжному інструментарію та широкій підтримці спільноти, Angular дозволяє розробникам створювати складні та масштабовані веб-додатки. [22; 23]
2. **React** - це бібліотека JavaScript, яку підтримує Facebook і яка є дуже популярною для створення користувацьких інтерфейсів. Вона дотримується компонентного підходу і використовує віртуальну DOM

(Document Object Model) для ефективного представлення змін в інтерфейсі. Декларативний синтаксис React, повторне використання компонентів та ефективний рендеринг роблять його ідеальним для створення інтерактивних інтерфейсів, керованих даними. Завдяки великій екосистемі бібліотек та інструментів, таких як Redux для управління станами, React став найкращим вибором для розробників, які шукають потужний та модульний фронтенд. [23]

3. **Vue.js** - це просунутий JavaScript фреймворк, який вирізняється простотою та універсальністю. Він дозволяє розробникам поступово впроваджувати його функціональність в існуючі проекти або створювати повноцінні односторінкові додатки. Vue.js пропонує м'яку криву навчання, архітектуру на основі компонентів та реактивну систему прив'язки даних. Він надає гнучкі можливості для розширення програми та безперешкодної інтеграції в існуючі проекти. [23]

Back-end фреймворки зосереджені на серверній розробці, обробці даних, бізнес-логіці та зв'язку з базами даних і зовнішніми сервісами. Ці фреймворки забезпечують основу для створення надійних і масштабованих серверних додатків, обслуговування запитів, управління даними та забезпечення безпеки. [23] Слід навести наступні приклади найпопулярніших Back-end фреймворків:

1. **Spring Framework** - це найпопулярніший фреймворк для Java, який широко використовується для створення додатків корпоративного рівня. Він надає такі функції, як інверсія управління (IoC), ін'єкція залежностей (DI) та аспектно-орієнтоване програмування (AOP). Spring надає модулі для різних функцій, включаючи веб-розробку (Spring MVC), інтеграцію з базами даних (Spring Data) та безпеку (Spring Security). Завдяки своїй легкій та модульній структурі Spring спрощує розробку додатків, покращує тестування та полегшує розширюваність і супроводжуваність. [22; 23; 21]
2. **Django** - це високорівневий веб-фреймворк на Python, який орієнтований на швидку розробку і дотримується підходу "battery-

inclusive". Він містить багато вбудованих функцій, таких як ORM, маршрутизація URL-адрес, обробка форм та автентифікація. Дотримання Django принципу DRY (Don't Repeat Yourself - не повторюй себе) та його сильні функції безпеки роблять його придатним для створення масштабованих та безпечних веб-додатків. [23]

3. **Ruby on Rails**, яку зазвичай називають Rails, - це повностековий фреймворк веб-додатків, написаний на Ruby. Rails сприяє простоті та продуктивності розробників, дотримуючись принципу "конвенції над конфігурацією". Акцент робиться на використанні чистого коду, автоматизованому тестуванні та архітектурному шаблоні MVC (Model-View-Controller). Rails надає такі функції, як ORM (Active Record), вбудована автентифікація та великі бібліотеки для швидкої розробки. Елегантний синтаксис і акцент на задоволенні потреб розробників роблять його популярним серед стартапів і малих та середніх проектів. [22]
4. **Laravel** - це PHP-фреймворк для веб-додатків, який набрав обертів в останні роки. Акцент на елегантному та виразному синтаксисі, з такими функціями, як ORM (Eloquent), маршрутизація та кешування. та планування завдань. Laravel спрощує типові завдання розробки, такі як міграція баз даних та автентифікація, пропонуючи при цьому надійну екосистему пакетів для розширення функціональності. Завдяки чистій та зрозумілій структурі коду, Laravel дозволяє розробникам створювати безпечні, зручні в обслуговуванні та високопродуктивні веб-додатки на PHP. [23]

2.2 Hibernate

У сфері програмної інженерії управління та збереження даних відіграють ключову роль в успіху будь-якої програми. Фреймворки об'єктно-реляційного відображення (ORM) слугують мостом між об'єктно-орієнтованою парадигмою програмування та реляційними базами даних, забезпечуючи безперебійну взаємодію та збереження даних. Hibernate, відомий фреймворк ORM, який бв створений для покращення процесу розробки програмного забезпечення. [17]

Об'єктно-реляційне відображення (ORM) слугує важливим мостом між об'єктно-орієнтованою парадигмою програмування та реляційними базами даних, пом'якшуючи невідповідність, що виникає через їхні різні принципи. В той час як об'єктно-орієнтоване програмування фокусується на інкапсуляції даних і поведінки в об'єктах, реляційні бази даних організовують дані в таблицях з чітко визначеними взаємозв'язками і обмеженнями. Ця розбіжність вимагає механізму для трансляції та синхронізації даних між об'єктно-орієнтованою моделлю програми та реляційною моделлю даних. [4; 18]

ORM-фреймворки, такі як Hibernate, спрощують процес відображення об'єктів у таблиці реляційної бази даних і сприяють безперешкодному маніпулюванню даними. Надаючи рівень абстракції, ORM-фреймворки захищають розробників від складнощів низькорівневої взаємодії з базами даних і дозволяють їм працювати з об'єктами в природній, об'єктно-орієнтованій манері. [4; 18]

ORM-фреймворки забезпечують об'єктно-реляційне відображення, використовуючи метадані, виражені через конфігураційні файли XML або анотації, для визначення зв'язків між об'єктами і таблицями бази даних. Ці метадані містять інформацію про атрибути об'єкта, зв'язки та ієрархії успадкування. Маючи цю інформацію, ORM-фреймворк може автоматично генерувати відповідні SQL-запити для збереження об'єктів у базі даних або їхнього вилучення за потреби. [17; 31]

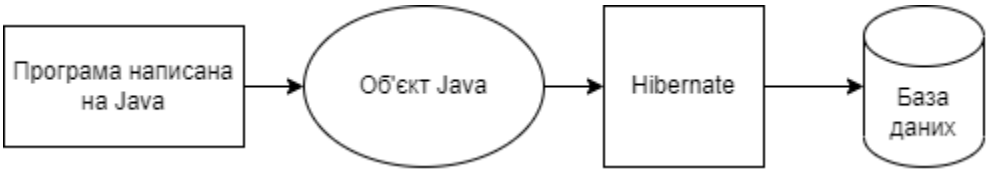


Рис. 2 – Взаємодія Hibernate між програмою та БД

Основна мета ORM - забезпечити прозору персистентність, гарантуючи, що розробники можуть маніпулювати об'єктами, не турбуючись про основні операції з базою даних. Завдяки використанню ORM розробники можуть виконувати CRUD-операції (Create, Read, Update, Delete) над об'єктами, а фреймворк піклується про перетворення цих операцій у відповідні SQL-оператори. [4; 17; 18]

Однією з суттєвих переваг ORM є підвищення продуктивності. Усуваючи необхідність писати низькорівневі SQL-запити та керувати з'єднаннями з базами даних вручну, розробники можуть зосередитися на бізнес-логіці своїх додатків. ORM-фреймворки також обробляють специфічні для бази даних операції та оптимізації, дозволяючи розробникам писати код для діагностики бази даних, який можна легко перенести на різні системи баз даних. [4; 17; 18]

Ліниве завантаження та кешування - дві важливі функції, які пропонує Hibernate, фреймворк об'єктно-реляційного відображення (ORM), для оптимізації продуктивності та підвищення ефективності пошуку та маніпулювання даними. [4; 17; 18] Давайте розглянемо ці можливості більш детально:

Ліниве завантаження. Ліниве завантаження - це механізм, який використовується Hibernate для відкладення завантаження пов'язаних об'єктів або асоціацій, поки програма не отримає до них явний доступ або не запитає їх. Замість того, щоб негайно завантажувати всі пов'язані об'єкти, Hibernate спочатку завантажує лише основний об'єкт і затримує отримання пов'язаних об'єктів, доки вони не знадобляться. [4; 17; 18]

Ліниве завантаження має кілька переваг:

- Зменшення споживання пам'яті: Завантажуючи пов'язані об'єкти на вимогу, Hibernate уникає зайвого заповнення пам'яті даними, які можуть не використовуватися у поточному потоці програм. [4; 17; 18]
- Покращена продуктивність: Ліниве завантаження зменшує кількість запитів до бази даних, завантажуючи пов'язані об'єкти лише за необхідності, що призводить до пришвидшення часу відгуку та покращення загальної продуктивності програми. [4; 17; 18]
- Покращена масштабованість: Завдяки лінивому завантаженню програма може ефективніше працювати з великими наборами даних, оскільки вона отримує необхідні дані лише за потреби. [17]

Для ефективної роботи лінивого завантаження потрібен активний сеанс Hibernate. Коли до асоційованого об'єкта вперше звертаються поза межами сеансу, Hibernate прозоро запускає запит до бази даних, щоб отримати дані. Це забезпечує безперешкодну інтеграцію з кодом програми, оскільки ліниво завантажені об'єкти поводяться так само, як і інші об'єкти програми. [18]

Кешування в Hibernate передбачає зберігання в пам'яті об'єктів або результатів запитів, до яких часто звертаються, що зменшує потребу в повторному доступі до бази даних. Hibernate надає кеш першого рівня (також відомий як кеш сеансу) і кеш другого рівня для підвищення продуктивності додатків. [4]

Кеш першого рівня пов'язаний із сеансом Hibernate. Коли об'єкти отримуються або завантажуються за допомогою Hibernate, вони автоматично зберігаються в кеші першого рівня. Будь-які наступні запити на той самий об'єкт в межах того самого сеансу обслуговуються безпосередньо з кешу, усуваючи потребу в додаткових запитах до бази даних. Це мінімізує накладні витрати на повторні обходи бази даних і покращує час відгуку. [4]

Кеш другого рівня - це спільний кеш, який можна налаштувати для декількох сеансів Hibernate. Він працює в більш широкому масштабі, дозволяючи кешувати об'єкти в різних сеансах і навіть різних екземплярах

програми. Кеш другого рівня зазвичай реалізується за допомогою зовнішніх провайдерів кешування, таких як Ehcache або Infinispan. Кешуючи об'єкти або результати запитів, до яких часто звертаються, кеш другого рівня ще більше зменшує кількість звернень до бази даних і підвищує загальну продуктивність програми. [4]

Hibernate забезпечує гнучкість у налаштуванні стратегій кешування для конкретних об'єктів, колекцій або запитів. Це дозволяє розробникам контролювати деталізацію кешування та визначати механізми кешування, які найкраще відповідають вимогам їхніх додатків. [4; 18]

Кешування в Hibernate має кілька переваг:

- Покращений час відгуку: Зменшуючи доступ до бази даних, кешування мінімізує час, необхідний для отримання даних, до яких часто звертаються, що призводить до прискорення часу відгуку. [4]
- Зменшення навантаження на базу даних: Кешування допомагає зменшити навантаження на сервер бази даних, обслуговуючи дані безпосередньо з пам'яті, зменшуючи потребу в частих запитах до бази даних. [4]
- Масштабованість: Кешування підвищує масштабованість програми, дозволяючи їй обробляти більше одночасних користувачів і більші набори даних, не перевантажуючи базу даних. [4; 18]

Однак кешування потрібно використовувати розумно, враховуючи такі фактори, як закінчення терміну дії кешу, узгодженість кешу та характер шаблонів доступу до даних програми. Неправильне керування кешем може призвести до втрати даних або збільшення споживання пам'яті. [4; 17]

2.3 Spring Framework

Аналізуючи фреймворки для Java які могли б стати найкращим рішенням, яке покращить якість та збільшить швидкість розробки програмного

забезпечення, я обрав Spring Framework. Аналізуючи Spring Framework можна відзначити, що це один з найрозвиненіших фреймворків для розробки ПЗ сьогодення. Мною було розглянуто фундаментальні концепції, принципи проектування, ключові особливості та архітектурні компоненти Spring Framework. Крім того,було проаналізовано переваги та варіанти використання Spring в контексті сучасної розробки програмного забезпечення. [6; 25]

Завдяки поглибленому вивченню його модулів та можливостей, я приведу важливість Spring Framework як потужного рішення для створення додатків. Результати цього дослідження висвітлюють вплив Spring Framework на індустрію програмного забезпечення та його внесок у підвищення ефективності, масштабованості та ремонтопридатності сучасних програмних систем. [20]

Spring Framework став провідним середовищем розробки додатків з відкритим вихідним кодом для корпоративного програмного забезпечення на основі Java. Він докорінно змінює спосіб проектування, створення та розгортання додатків, надаючи цілісну, модульну та розширювану платформу. [20; 21]

Для того, щоб досконально зрозуміти Spring Framework, необхідно розібратися з його основними концепціями. Інверсія управління (IoC) та ін'єкція залежностей (DI). Було проведено аналіз, як ці концепції забезпечують вільний зв'язок, тестування та масштабованість при розробці додатків. [29]

Spring Framework втілює набір принципів проектування, які керують його архітектурою та практикою розробки. У цьому розділі описано принципи SOLID та патерни проектування, що використовуються у Spring, зокрема патерни Factory, Singleton та Proxy. Розуміння цих принципів проектування може дати уявлення про гнучкість, ремонтопридатність та розширюваність додатків на основі Spring. [39]

2.3.1 Шаблони проектування які використовує Spring Framework

Spring Framework слідує принципам SOLID, набору рекомендацій для написання чистого, зручного для обслуговування та розширюваного коду. Ці принципи допомагають розробникам створювати програмне забезпечення, яке легко розуміти, тестувати та змінювати. [16]

Spring Framework використовує різні патерни проектування для вирішення поширених проблем проектування програмного забезпечення. До найпоширеніших патернів відносяться. [16]

Фабричний метод використовується у Spring для створення та керування екземплярами об'єктів. Він інкапсулює логіку створення об'єктів і забезпечує центральну точку для конкретизації об'єктів. [16]

Одинак. Паттерн Singleton гарантує, що у всьому додатку буде лише один екземпляр класу. У Spring біни «одинаки» є областю видимості за замовчуванням, і один екземпляр спільно використовується декількома компонентами. [16]

У фреймворку Spring патерн «Proxy» використовується для створення проксі-об'єкта, який виступає посередником між клієнтським кодом і власне цільовим об'єктом. Ці проксі-об'єкти контролюють доступ до цільового об'єкта і дозволяють застосовувати додаткову функціональність без безпосередньої зміни коду цільового об'єкта. Це особливо корисно для реалізації наскрізних проблем у модульний і ненав'язливий спосіб. [16]

Шаблон проксі у Spring можна зрозуміти в контексті аспектно-орієнтованого програмування (АОП), парадигми програмування, призначеної для відокремлення основної бізнес-логіки програми від загальних проблем, таких як ведення журналів, кешування, безпека і т.д. та управління транзакціями. AOP досягає такого розділення за допомогою проксі-серверів, які перехоплюють виклики методів та застосовують іншу поведінку навколо них. [16]

Spring надає два типи проксі-серверів: динамічні JDK-проксі-сервери та CGLIB-проксі-сервери. [16]

Динамічні JDK-проксі. Ці проксі використовуються, коли ціль реалізує хоча б один інтерфейс. Spring динамічно створює проксі-об'єкт, який реалізує той самий інтерфейс, що й цільовий об'єкт. Коли метод викликається на проксі-об'єкті, він перехоплює виклик і делегує його реальному цільовому об'єкту, дозволяючи виконувати інші операції до і після виклику методу. [16; 15]

Проксі об'єкти CGLIB. Ці проксі використовуються, коли цільовий об'єкт не реалізує жодного інтерфейсу. У цьому випадку Spring використовує бібліотеку CGLIB для створення підкласів цільового об'єкта, що дозволяють перевизначати методи. Створені підкласи діють як проксі і перехоплюють виклики методів, щоб застосувати бажану додаткову поведінку. Використовуючи проксі, Spring застосовує різні наскрізні проблеми до цільових об'єктів, не змінюючи свою реалізацію. Такий підхід забезпечує розділення проблем і сприяє створенню модульного коду, що легко підтримується. Крім того, AOP Spring на основі проксі дозволяє застосовувати аспекти декларативно за допомогою конфігурації, анотацій і навіть динамічно під час виконання. [14]

Спостерігач. Паттерн спостерігач використовується у моделі подієво-керованого програмування Spring. Він дозволяє об'єктам підписуватися на події, опубліковані іншими об'єктами, та прослуховувати їх, полегшуючи слабко зв'язану та роз'єднану комунікацію між компонентами. [16]

Шаблонний метод. Режим шаблонного методу використовується у шаблонних класах Spring, таких як шаблони B. Jdbc. Він забезпечує основу для алгоритму і дозволяє розробникам налаштовувати окремі частини, зберігаючи загальний процес незмінним. [13; 16]

3 РОЗРОБКА ВЕБ ДОДАТКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДЕКАНАТУ

3.1 Опис етапів розробки програмного забезпечення

Для початку було проведено аналіз та зібрано вимоги до веб-додатку для автоматизації роботи деканату. Було вивчено поточні процеси та потреби деканату. На основі цього аналізу, було складено план, в якому описав всі вимоги до функціональності та можливостей веб-додатку. Далі, я перейшов до етапу проектування. Було визначено архітектуру веб-додатку, а також компоненти, їх взаємодію та спосіб організації бази даних. Також, було розроблено дизайн користувацького інтерфейсу, приділяючи увагу зручності використання та естетичним аспектам. Після завершення проектування, почалася приступив до фази реалізації. За допомогою відповідних технологій, мов програмування та фреймворків, я написав програмний код мого веб-додатку. При реалізації, було дотримано кращих практик програмування, про які буде наведено далі

3.1.1 Короткий опис вимог

Перед початком розробки веб-додатку для автоматизації роботи деканату було визначено ряд вимог, а саме:

Функціональні вимоги:

- **Облік студентів.** Можливість додавання, редагування та видалення студентів з бази даних. Зберігання інформації про студентів, включаючи особисті дані, контактну інформацію та історію навчання. Автоматизована генерація унікальних студентських номерів та ідентифікаційних даних.
- **Облік груп.** Створення та управління групами студентів. Прив'язка студентів до конкретних груп. Оновлення та зберігання інформації про групи, включаючи назву, спеціальність та кількість студентів.

- **Облік успішності.** Введення оцінок студентів за кожен предмет. Розрахунок середнього балу та кількості нарахованих кредитів для кожного студента. Збереження історії оцінок та успішності студента протягом навчального процесу

Нефункціональні вимоги:

- **Безпека**. Захист конфіденційної інформації про студентів та їх успішність. Автентифікація та авторизація користувачів з використанням надійних механізмів аутентифікації і контролю доступу.
- **Швидкодія**. Швидкий та ефективний доступ до бази даних для оптимальної продуктивності системи.Мінімальні часові затримки при виконанні операцій, щоб забезпечити швидкий відгук системи на дії користувачів.
- **Надійність**. Забезпечення стабільності та надійності системи в умовах високого навантаження. Резервне копіювання даних та можливість відновлення системи в разі виникнення помилок або збоїв.
- **Інтерфейс.** Забезпечення зручного та легкого користування веб-додатком без потреби в особливих технічних навичках.Інтуїтивний інтерфейс, що дозволяє користувачам швидко зорієнтуватися та виконувати необхідні дії.
- **Масштабованість**. Забезпечення гнучкості системи для масштабування в разі збільшення кількості студентів, груп та навчальних планів. Підтримка великого обсягу даних та одночасних користувачів.

3.2 Опис роботи та компонентів веб-додатку

Програмний продукт, що розробився, є комплексною системою автоматизації деканату, яка надає широкий спектр функціональних можливостей для спрощення та полегшення всіх аспектів діяльності деканату.

Основною метою даного програмного продукту є забезпечення ефективного управління даними студентів, групами, курсами, успішністю та

розкладом занять. Завдяки інтуїтивно зрозумілому інтерфейсу та розширеним функціям користувачі можуть легко орієнтуватися в системі та виконувати необхідні завдання.

Для досягнення цих цілей у програмному продукті використовується сучасна архітектурна парадигма Model-View-Controller (MVC), яка дозволяє логічно відокремити компоненти системи та забезпечити їхню взаємодію. Модель відповідає за управління даними та бізнес-логікою, представлення забезпечує відображення інформації користувачеві, а контролер керує взаємодією між моделлю та представленням. [25]

3.2.1 UML діаграми

Мова моделювання UML (Уніфікована мова моделювання) - це інструмент, що широко використовується у розробці програмного забезпечення. Вона дозволяє представити та передати різні аспекти структури, поведінки та взаємодії системи. [46]

Діаграми UML забезпечують стандартизований спосіб представлення компонентів системи, взаємозв'язків та процесів. У цій статті ми розглянемо три основні типи діаграм UML: діаграми варіантів використання, діаграми класів та діаграми зв'язків між базами даних. Було проаналізовано та створено кілька типів таких діаграм для даного проекту. [46]

Діаграми варіантів використання допомагають зафіксувати функціональні вимоги до системи, ілюструючи взаємодію між акторами (користувачами, зовнішніми системами або іншими сутностями) і самою системою. Діаграми надають загальний огляд того, як система працює та допомагають зрозуміти її поведінку. [46] Діаграма варіантів використання складається з наступних основних складових елементів:

Актори. Актори представляють зовнішні сутності, які взаємодіють з системою. Це можуть бути користувачі, інші системи або навіть апаратні пристрої. Актори зображуються у вигляді паличок і з'єднуються з варіантами використання за допомогою ліній зв'язку. [46]

- **Варіант використання**. Варіант використання представляє конкретну функцію або завдання, яке може виконувати система. Варіанти використання зображуються у вигляді еліпсів і з'єднуються з акторами асоціативними лініями. [46]
- **Взаємозв'язки.** Встановлюють зв'язки між акторами і варіантами використання, щоб проілюструвати їх взаємодію і залежності. Найважливіші типи зв'язків включають асоціацію, узагальнення та відносини, що містять/розширюють зв'язки. [46]

Було розроблено дві діаграми використання так як у нас в системі існують 2 ролі з різним функціоналом. Один з них адміністратор в якого буде можливість додавати нових користувачів до системи, створювати групи, створювати дані про викладачів, предмети та створювати зв’язки між ними. Інша роль «Викладач» в якої буде можливість додавати інформацію про заняття які проходили а також вести облік відвідування та успішності студентів.

Рис. 3.2.1.1 - Діаграма варіантів використання користувача «Вчитель»

В даній діаграма можемо спостерігати як і було сказано вище наступні функції для ролі «Викладач».

Крім того, це дослідження підкреслює важливість орієнтованого на користувача дизайну в розробці освітнього програмного забезпечення. Зосереджуючись на викладачеві як головній дійовій особі, діаграми варіантів використання допомагають забезпечити адаптацію системи до його унікальних вимог, сприяючи інтуїтивному та ефективному користувацькому досвіду. Взаємозв'язки варіантів використання, включаючи асоціації, включають і розширюють гнучкість і модульність системи, пристосовуючи її до різноманітних сценаріїв і потреб, що змінюються.

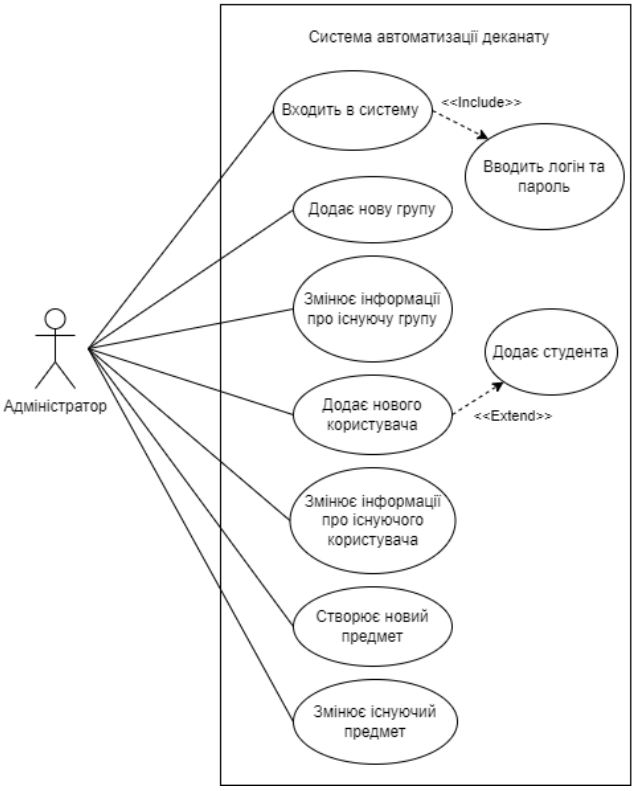


Рис. 3.2.1.2 – Діаграма варіантів використання користувача «Адміністратор»

В даній діаграма визначено деякий функціонал який доступний для ролі «Адміністратор».

Діаграми класів зосереджені на представленні статичної структури системи шляхом моделювання класів, їхніх атрибутів, методів і зв’язків. Ці діаграми є основою для об'єктно-орієнтованого аналізу та проектування. [46] Діаграма класів складається з таких ключових елементів:

- **Клас.** Клас — це будівельний блок системи, який інкапсулює об’єкти зі схожими атрибутами, поведінкою та зв’язками. Кожен клас представлений у вигляді прямокутника, що містить три частини: назву класу, властивості та методи. [46]

- **Асоціації.** Асоціації встановлюють стосунки між класами та вказують, як вони пов’язані та взаємодіють один з одним. Асоціації можуть бути «один-до-одного», «один-до-багатьох» або «багато-до-багатьох» і представлені у вигляді ліній, що з’єднують пов’язані класи. [46]

- **Успадкування.** Успадкування використовується для представлення зв’язку «є» між класами, де один клас успадковує атрибути та поведінку від іншого суперкласу. Спадкування представлено стрілкою, що вказує на суперклас. [46]

- **Множинність.** множинність визначає кількість екземплярів, які можуть існувати у зв’язку між класами. Він вказує на кардинальність зв’язку та на те, чи є він зв’язком «один до одного», «один до багатьох» або «багато до багатьох». [46]

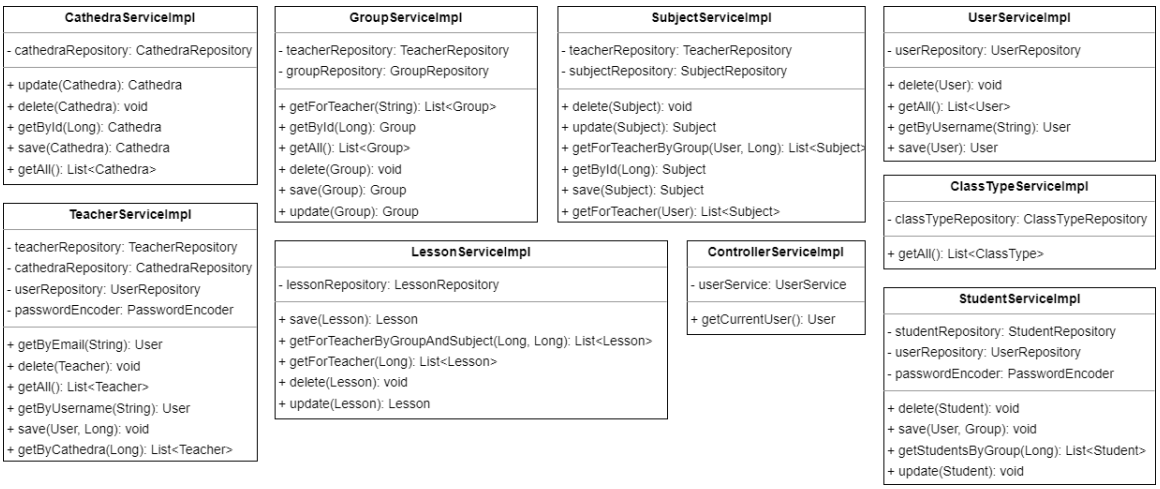


Рис. 3.2.1.3 – Діаграма класів сервісів

Вивчаючи діаграми класів було також створено діаграму класів для мого додатку. Варто відзначити, що дані класи не мають між собою зв’язків. Дана діаграма класів представляє лише частину класів, а саме класи імплементації сервісів.

Також надзвичайно важливою частиною цього програмного продукту є база даних, де зберігаються всі необхідні дані про студентів, групи, курси та інша інформація. Реляційні діаграми бази даних показують взаємозв'язки між таблицями та полями, які допомагають краще зрозуміти структуру бази даних та взаємозв'язки між сутностями. [32]

Щоб реалізувати функції додатку було створено 10 таблиць з різними зв’язками між собою, включаючи один до одного та один до багатьох.

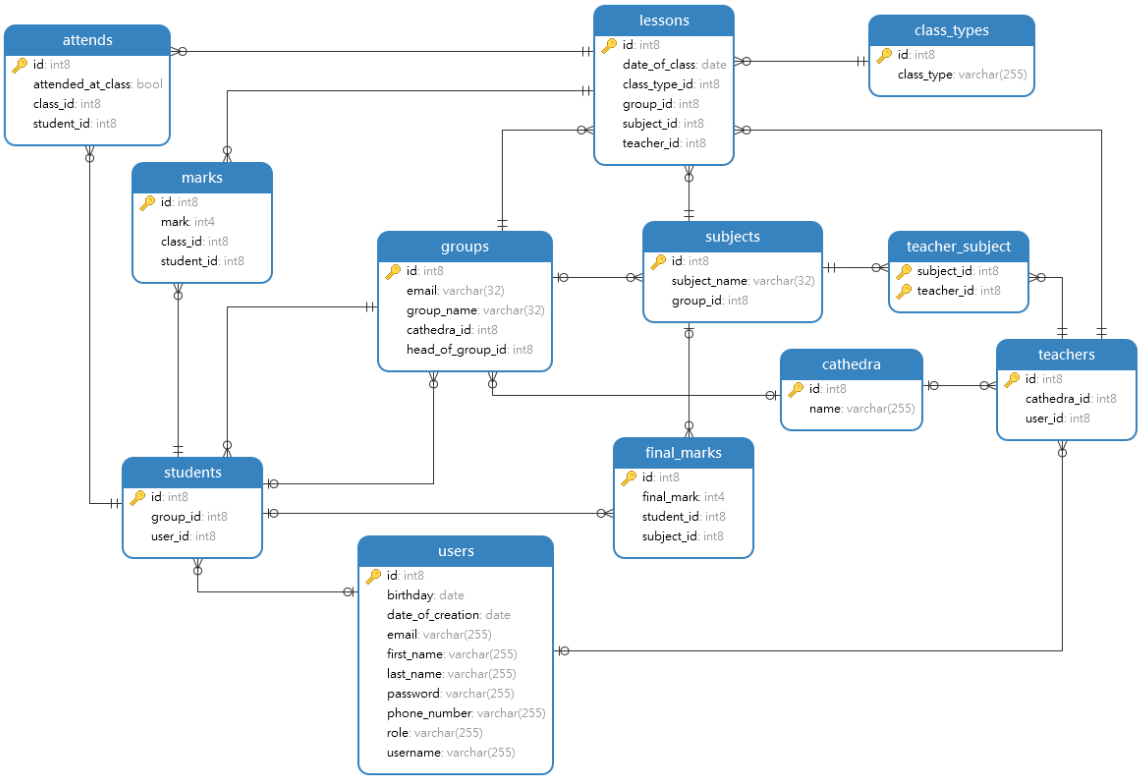


Рис. 3.2.1.4 – Діаграма схеми бази даних

Добре розроблена схема організовує дані в логічний та ефективний спосіб. Вона визначає, як дані зберігаються, індексуються і витягуються, що впливає на продуктивність операцій з базою даних. Оптимізований дизайн

схеми може значно підвищити продуктивність запитів і скоротити час пошуку даних.

У схемах баз даних встановлюють зв'язки між різними сутностями або таблицями в базі даних. Вони визначають первинний ключ, зовнішній ключ і обмеження посилальної цілісності, які посилюють зв'язки і забезпечують узгодженість даних між таблицями. Правильно визначені зв'язки уможливлюють ефективний пошук даних і підтримують складні багатотабличні запити.

Схеми відіграють життєво важливу роль у забезпеченні безпеки даних та контролю доступу. Вони визначають дозволи, привілеї та ролі, які обмежують несанкціонований доступ до конфіденційних даних. [31]

Добре розроблена схема дозволяє базі даних розвиватися з часом. Коли бізнес-потреби змінюються, схему можна модифікувати, щоб врахувати нові елементи даних, зв'язки або обмеження, не впливаючи на існуючі дані або додатки. Гнучкий дизайн схеми дозволяє базі даних масштабуватися та адаптуватися до мінливих бізнес-потреб. [31; 32]

Діаграми зв’язків бази даних використовуються як артефакти документації, що описують структуру і семантику бази даних. Вони надають чіткий і стислий огляд моделі даних, зв'язків між сутностями та атрибутами.

3.3 Огляд коду частини додатку

Виходячи з назви теми, у нас було завдання використовувати мову програмування Java, приклади якої буде далі наведено далі. Вище ми розглянули діаграму класів, тож розглянемо реалізацію одного з них.

```java
@Service
@RequiredArgsConstructor
public class SubjectServiceImpl implements SubjectService {

    private final SubjectRepository subjectRepository;
    private final TeacherRepository teacherRepository;

    1 usage  new *
    @Override
    public List<Subject> getSubjectsForTeacher(User user) {
        Teacher teacher = teacherRepository.findByUserUsername(user.getUsername())
                .orElseThrow(() -> new UsernameNotFoundException("User with username: " + user.getUsername() + " not found"));
        return subjectRepository.findAllByTeacher(teacher.getId());
    }

    1 usage  new *
    @Override
    public List<Subject> getSubjectsForTeacherByGroup(User user, Long groupId) {
        Teacher teacher = teacherRepository.findByUserUsername(user.getUsername())
                .orElseThrow(() -> new UsernameNotFoundException("User with username: " + user.getUsername() + " not found"));
        return subjectRepository.findAllByTeacherAndGroup(teacher.getId(), groupId);
    }

    1 usage  new *
    @Override
    public Subject getSubjectById(Long subjectId) {
        return subjectRepository.findById(subjectId)
                .orElseThrow(() -> new SubjectNotFoundException("Subject with id:" + subjectId + " not found"));
    }
}
```

Рис. 3.3.1 – Реалізація сервісу для роботи з предметами

Наведений код представляє клас реалізації сервісу під назвою «SubjectServiceImpl», який реалізує інтерфейс «SubjectService». Давайте розберемо код і зрозуміємо його функціональність.

Код передбачає наявність наступних залежностей, що вводяться за допомогою ін'єкції конструктора:

```java
private final SubjectRepository subjectRepository;
private final TeacherRepository teacherRepository;
```

Рис. 3.3.2 – Залежності «SubjectServiceImpl»

Ці репозиторії використовуються для взаємодії зі сховищем даних для виконання CRUD-операцій над сутностями «Subject» та «Teacher».

Клас SubjectServiceImpl надає наступні методи:

- **Метод «getSubjectsForTeacher»** отримує об'єкт «User» як параметр і повертає список предметів, пов'язаних з викладачем, що відповідає даному користувачеві. Він робить внутрішній запит до «TeacherRepository», щоб знайти

сутність вчителя на основі наданого імені користувача. Якщо вчителя знайдено, він використовує знайдений ідентифікатор вчителя для запиту до сховища предметів і отримує всі предмети, пов'язані з цим вчителем.

- **Метод «getSubjectsForTeacherByGroup»** схожий на попередній, але додатково фільтрує предмети на основі вказаного ідентифікатора групи. Він знаходить вчителя за наданим іменем користувача, а потім знаходить предмети, пов'язані з вчителем і заданим ідентифікатором групи.
- **Метод «getSubjectById»** отримує ідентифікатор предмета як параметр і знаходить відповідний предмет зі сховища предметів. Якщо предмет знайдено, він повертається, інакше генерується спеціальне виключення «SubjectNotFoundException».

Анотації @Service і @RequiredArgsConstructor вказують на те, що цей клас є сервісним компонентом і що конструктор буде автоматично згенеровано для фінальних полів відповідно. Давайте розглянемо, що саме являють собою ці анотації. [25]

Анотація @Service - це стереотипна анотація з Spring Framework. Зазвичай вона використовується для анотування класів, які виконують сервісні завдання, такі як бізнес-логіка, маніпулювання даними або зовнішні системні взаємодії. Використовуючи цю анотацію, ви вказуєте, що анотований клас слугує сервісним компонентом у вашому додатку. [25]

Коли Spring-контейнер сканує компоненти вашого додатку, він розпізнає класи з анотацією @Service і автоматично реєструє їх як біни. Потім ці біни можуть бути використані в інших компонентах за допомогою ін'єкції залежностей. [25]

У наведеному фрагменті коду «Service» використовується для анотації класу SubjectServiceImpl, вказуючи на те, що він слугує сервісним компонентом, відповідальним за обробку операцій, пов'язаних з предметом.

Анотація «RequiredArgsConstructor» є частиною бібліотеки Lombok, яка надає різні анотації для зменшення шаблонного коду в класах Java. Ця

конкретна анотація генерує конструктор для класу з параметрами для всіх кінцевих полів.

У наведеному коді «RequiredArgsConstructor» використовується для анотації класу «SubjectServiceImpl». В результаті Lombok автоматично генерує конструктор, який приймає «SubjectRepository» та «TeacherRepository» як параметри. Оскільки ці поля позначені як фінальні, згенерований конструктор ініціалізує ці поля в самому конструкторі, усуваючи потребу в явній реалізації конструктора. [38]

Використовуючи @RequiredArgsConstructor, ми можемо уникнути ручного написання конструкторів для класів з великою кількістю фінальних полів, зменшуючи захараженість коду і покращуючи його читабельність. [38]

Далі розглянемо один із методів одного з контролерів нашого додатку. Фрагмент коду, наведено нижче, являє собою метод обробника для «GET» запиту, який отримує дані для заповнення подання, пов'язаного з групою та суб'єктом. Давайте пройдемося по коду і зрозуміємо його функціональність.

```
@GetMapping("/subjects/{groupId}/{subjectId}/table")
public String getGroupListForSubject(Model model, @PathVariable Long groupId, @PathVariable Long subjectId) {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

    String currentPrincipalName = authentication.getName();
    User user = userService.getByUsername(currentPrincipalName);
    model.addAttribute( attributeName: "user", user);

    Subject subject = subjectService.getSubjectById(subjectId);
    model.addAttribute( attributeName: "subject", subject);

    List<Student> students = studentService.getStudentByGroup(groupId);
    model.addAttribute( attributeName: "students", students);
    model.addAttribute( attributeName: "numbers", rangeClosed(1, students.size()).boxed().toList());

    Group group = groupService.getGroupById(groupId);
    model.addAttribute( attributeName: "group", group);

    List<ClassType> classTypes = classTypeService.getAllClassTypes();
    model.addAttribute( attributeName: "classTypes", classTypes);
    model.addAttribute( attributeName: "lesson", new Lesson());

    List<Lesson> lessons = lessonService.getLessonsForTeacherByGroupAndSubject(groupId, subjectId);
    model.addAttribute( attributeName: "lessons", lessons);

    return "teacher/group-subject";
}
```

Рис. 3.3.3 – Метод обробки «GET» запиту

Метод починається з отримання поточного автентифікованого користувача за допомогою «SecurityContextHolder». Він отримує об'єкт Authentication і витягує ім'я користувача, який наразі автентифікований. Потім використовується userService для отримання об'єкта «User», пов'язаного з цим іменем користувача. Отриманий об'єкт «User» додається до атрибуту моделі з ключем «user». [20]

```
Subject subject = subjectService.getSubjectById(subjectId);
model.addAttribute( attributeName: "subject", subject);
```

Рис. 3.3.4 – Код пошуку предмету по його «id»

Код отримує суб'єкт за допомогою сервісу «subjectService» шляхом виклику методу «getSubjectById» з наданим суб'єктом ідентифікатором «subjectId». Потім отриманий суб'єкт додається до атрибуту моделі з ключем "subject".

```
List<Student> students = studentService.getStudentByGroup(groupId);
model.addAttribute( attributeName: "students", students);
model.addAttribute( attributeName: "numbers", rangeClosed(1, students.size()).boxed().toList());
```

Рис. 3.3.5 – Код пошуку студентів по «id» групи

Код отримує список студентів, пов'язаних із заданим «groupId» за допомогою «studentService», і додає їх до атрибутів моделі з ключами "students" і "numbers". Атрибут "numbers" містить список чисел від 1 до розміру списку студентів, який може бути корисним для відображення номерів рядків у поданні.

3.3.1 Огляд безпекової складової додатку

Швидкий розвиток веб-технологій призвів до розробки складних веб-додатків, які обробляють конфіденційні дані. Таким чином, захист цих програм

стає критично важливим. Було проаналізовано способи захисту додатків Spring MVC, які широко використовуються в корпоративних середовищах завдяки їх гнучкості, масштабованості та простоті розробки. [25]

Автентифікація — це процес перевірки особи користувача або організації, яка отримує доступ до програми. Spring Security — це потужна структура безпеки, яку можна легко інтегрувати з Spring MVC для забезпечення надійних механізмів автентифікації, таких як автентифікація на основі імені користувача/паролю, автентифікація на основі маркерів та інтеграція із зовнішніми постачальниками автентифікації, такими як LDAP або OAuth). [20; 21]

Після перевірки особи користувача авторизація визначає, до яких дій та ресурсів він може отримати доступ у додатку. Spring Security підтримує авторизацію на основі ролей та дозволів, що дозволяє розробникам визначати деталізовані політики контролю доступу. Налаштовуючи правила безпеки за допомогою анотацій або конфігурації на основі XML, розробники можуть застосовувати обмеження доступу на основі ролей користувачів, гарантуючи, що тільки авторизовані користувачі можуть виконувати певні дії. [20; 21]

Перевірка вхідних даних відіграє вирішальну роль у запобіганні різним типам атак, включаючи міжсайтовий скриптинг (XSS) та SQL-ін'єкції. Spring MVC надає надійні механізми зв'язування та валідації даних, включаючи використання валідаційних анотацій, кастомних валідаторів та утиліт перетворення даних. [20; 21] Перевіряючи дані, що вводяться користувачем, як на стороні клієнта, так і на стороні сервера, розробники можуть зменшити ризик впровадження шкідливого коду або несанкціонованих даних у додаток. [20; 21]

Керування сеансами є життєво важливим для підтримки стану та захисту від атак, пов'язаних з сеансами, таких як фіксація та перехоплення сеансів. Spring MVC пропонує вбудовану підтримку управління сеансами, включаючи можливість генерувати безпечні ідентифікатори сеансів, контролювати таймаут сеансу та застосовувати безпечні атрибути cookie. Розробники також можуть

використовувати функції управління сеансами Spring Security, такі як одночасний контроль сеансів та захист від фіксації сеансів, щоб підвищити безпеку своїх додатків. [20; 21; 25]

Проаналізувавши можливості фреймворку було прийнято рішення про потребу шифрування паролів користувачів в базі даних. Для цього було використано «BCryptPasswordEncoder». [20; 47]

```
BodyakovDeveloper
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder( strength: 12);
}

BodyakovDeveloper
@Bean
public AuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authenticationProvider = new DaoAuthenticationProvider();
    authenticationProvider.setPasswordEncoder(passwordEncoder());
    authenticationProvider.setUserDetailsService(userDetailsService);
    return authenticationProvider;
}
```

Рис. 3.3.1.1 – Конфігурація «PasswordEncoder» та «AuthenticationProvider»

Безпека паролів є критично важливим аспектом будь-якої програми, яка має справу з автентифікацією користувачів. Надійне зберігання паролів та використання надійних методів шифрування є життєво важливими для захисту облікових записів користувачів від несанкціонованого доступу. [20; 21]

«BcryptPasswordEncoder» - це широко використовуваний алгоритм хешування паролів, що надається «Spring Security». Він використовує функцію хешування bcrypt, яка включає в себе соління та декілька ітерацій для створення безпечного хешу. Використовуючи «BcryptPasswordEncoder» у «Spring MVC» додатку, паролі можна зберігати у хешованому та посоленому

форматі, що робить надзвичайно складним для зловмисника отримати оригінальні паролі. [24 ;47]

Фрагмент коду містить метод з анотацією «Bean», який створює боб «PasswordEncoder». Цей боб відповідає за кодування та перевірку паролів у додатку. Повертаючи екземпляр «BcryptPasswordEncoder» з коефіцієнтом стійкості 12, метод гарантує, що процес хешування паролів є обчислювально дорогим, що збільшує складність атак грубої сили. [20; 47]

Ще однією важливою конфігурацією в методі «authenticationProvider()» є налаштування «userDetailsService». «UserDetailsService» відповідає за отримання інформації про користувача, включаючи хешовані паролі, з джерела даних. Дуже важливо правильно реалізувати інтерфейс «userDetailsService», щоб отримати інформацію про користувача безпечно та ефективно. [19]

Використовуючи «BcryptPasswordEncoder» та «DaoAuthenticationProvider» у «Spring MVC» додатку, розробники можуть підвищити безпеку паролів та захистити облікові записи користувачів від несанкціонованого доступу. Безпечне зберігання паролів у хешованому форматі з сіллю гарантує, що навіть якщо база даних буде скомпрометована, оригінальні паролі залишаться захищеними. Крім того, дотримуючись рекомендованих найкращих практик безпеки паролів, таких як використання стійких алгоритмів шифрування та застосування належних політик складності паролів, розробники можуть ще більше посилити загальний рівень безпеки своїх «Spring MVC» додатків. [19; 24]

Рис. 3.3.1.2 – Закодований пароль за допомогою «BcryptPasswordEncoder»

Зашифрований рядок пароля складається з декількох частин, розділених символами "$". Насправді було закодовано пароль «qwerty». Нижче наведено розбивку різних компонентів:

- "$2a$" вказує на версію bcrypt, що використовується. В даному випадку це версія 2a, яка широко використовується в різних реалізаціях.
- "12$" - фактор вартості, або кількість ітерацій, які використовує алгоритм. В даному випадку це означає, що алгоритм був запущений 12 разів, що робить процес хешування повільнішим, але більш безпечним.
- Решта рядка "MsUWqEJOnVlFdv0eMOzHoeTWDYXR4z2BgXvjGebRN Mwl621uiYqQ" являє собою власне пароль, зашифрований bcrypt.

Bcrypt є односторонньою функцією, що означає, що відновити оригінальний пароль із зашифрованого рядка обчислювально неможливо. Ця властивість допомагає захистити паролі користувачів, надійно зберігаючи їх у базі даних або системі, оскільки навіть якщо зашифрований пароль буде зламано, зловмиснику буде дуже важко знайти оригінальний пароль. [47]

3.3.2 Огляд «front-end» частини додатку

У контексті Spring Boot і Thymeleaf папка "resources" слугує важливим компонентом для організації статичних ресурсів і шаблонів «Thymeleaf». [14]

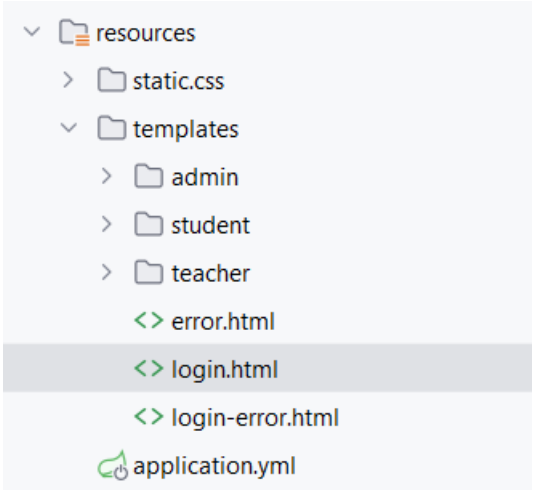


Рис. 3.3.2.1 – Організація тек в «Spring Boot» для «front-end» частини

Папка "static", що знаходиться в каталозі "resources", слугує місцем зберігання статичних ресурсів, таких як файли CSS, JavaScript, зображення та інші файли, які безпосередньо обслуговуються веб-сервером. [14] Вона є

невід'ємною частиною шляху класів веб-додатку, і будь-який ресурс, розміщений у цій папці, автоматично стає доступним для клієнтів. Наприклад, файл CSS з назвою "styles.css", розміщений у папці "resources/static/css", може бути доступний за адресою "/css/styles.css". [14]

Папка "templates": Папка "templates", також розташована в каталозі "resources", призначена для зберігання шаблонів Thymeleaf. Thymeleaf, надійний серверний движок шаблонів Java, дозволяє створювати динамічні подання, використовуючи синтаксис, подібний до HTML. [13] Ці шаблони використовують заповнювачі, вирази та логіку, які вирішуються та відображаються сервером для створення остаточного HTML, що надсилається клієнтам. При використанні Thymeleaf з Spring Boot тека "templates" автоматично налаштовується як місце за замовчуванням для пошуку файлів шаблонів. Підкаталоги в папці "templates" можуть бути використані для організації шаблонів на основі структури програми. [13; 14]

Використовуючи папку "templates" і Thymeleaf, розробники можуть ефективно відокремити логіку презентації від решти додатку, полегшуючи генерацію динамічних HTML-сторінок на основі даних, наданих Spring Boot-додатком. [13]

Крім того, варто зазначити, що папка ресурсів не обмежується лише "static" та "templates". Розробники можуть створювати додаткові підкаталоги в папці ресурсів для розміщення інших типів ресурсів, включаючи файли властивостей, скрипти баз даних або конфігураційні файли. Така гнучкість дозволяє організувати ресурси відповідно до специфічних вимог конкретної програми. [13; 14]

Як було написано вище, додаток використовує Spring Security, за допомогою якого імплементує рольову модель доступу до ресурсів. Це означає, що коли ми введено логін та пароль користувача «Вчитель» то у нас буде можливість користуватися функціями які доступні тільки для вчителя, навіть якщо додаток має набагато більше функцій. Так саме можемо сказати і про роль «Адміністратор». Давайте розглянемо сторінку входу в систему.

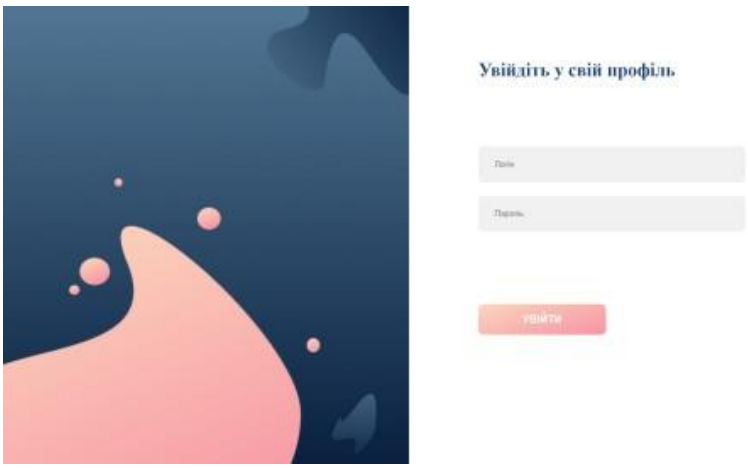


Рис. 3.3.2.2 – Сторінка входу в систему

Сторінку входу було реалізовано за допомогою мови розмітки HTML та стилів CSS, також щоб ми могли зайти в систему, були використанні інструменти Spring Security та Thymeleaf.

```
<div id="create-subject-modal" class="modal">
    <div class="modal-content">
        <span class="close-subject">×</span>
        <h2>Create Subject</h2>

        <form th:action="@{/admin/subjects}" th:object="${subject}" method="post">
            <label for="subjectName">Subject Name:</label>
            <input type="text" id="subjectName" th:field="*{subjectName}" required/><br/>

            <label for="group">Group:</label>
            <select id="group" th:field="*{group}" required>
                <option th:each="group : ${groups}" th:value="${group.id}" th:text="${group.groupName}"></option>
            </select><br/>

            <label for="teachers">Teachers:</label>
            <select id="teachers" th:field="*{teachers}" required multiple>
                <option th:each="teacher : ${teachers}" th:value="${teacher.id}"
                        th:text="${teacher.user.firstName + ' ' + teacher.user.lastName}"></option>
            </select><br/>

            <button type="submit">Create</button>
        </form>
    </div>
</div>
```

Рис. 3.3.2.3 – Частина коду головної сторінку адміністратора

Наведений блок коду представляє модальне діалогове вікно у форматі HTML для створення предмета в навчальному додатку або додатку для управління курсами. Модальне вікно містить елементи форми, зокрема поля для введення назви предмета, випадаюче меню для вибору групи предмета і ще одне випадаюче меню для вибору декількох викладачів, пов'язаних з цим предметом. Після заповнення форми дані надсилаються на вказану URL-адресу, що дозволяє серверному додатку створити предмет на основі наданої інформації.

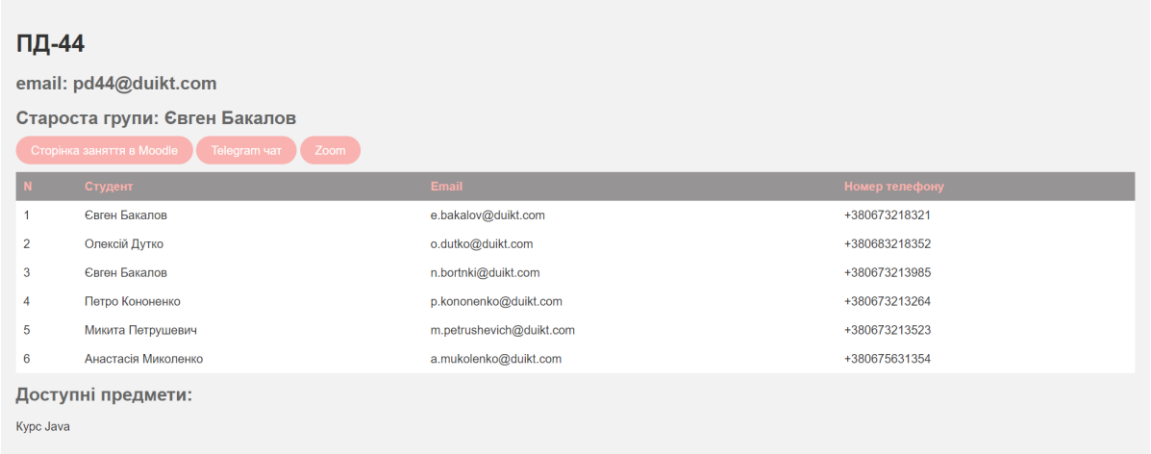


Рис. 3.3.2.4 – Сторінка групи для викладача

На рисунку 3.3.2.4 представлено сторінку групи для викладача на якій він може бачити контактні дані студента, а також перейти на сторінку предмету який є спільним як для групи так і для викладача.

3.4 Застосування фреймворків JUnit та Mockito для тестування веб-додатку

При розробці додатку я використав Mockito, популярний фреймворк, щоб полегшити процес тестування. Mockito дозволив мені створювати макетні об'єкти, які є імітаційними версіями залежностей, що дозволяє проводити ізольоване тестування окремих компонентів.

Використовуючи Mockito, я зміг визначати поведінку цих об'єктів, вказуючи відповіді, які вони повинні надавати при взаємодії з ними під час тестування. Це дозволило мені змоделювати різні сценарії і переконатися, що додаток поводився так, як очікувалося, за різних умов.

В деяких тестах було створено імітаційні об'єкти, які представляли зовнішні сервіси або бази даних, на які покладався наш додаток. Визначивши бажані реакції цих об'єктів, ми могли протестувати поведінку нашого додатку без реальних викликів API або доступу до бази даних. Це не тільки спростило процес тестування, але й зробило його швидшим та надійнішим.

Mockito також надав нам потужні функції, такі як перевірка викликів методів. Я можу стверджувати, що конкретні методи були викликані з очікуваними параметрами, гарантуючи, що мій додаток коректно взаємодіє зі своїми залежностями. [2]

Крім того, Mockito дозволив мені імітувати певні винятки, що дало нам змогу ретельно протестувати обробку помилок та сценарії винятків. З його допомогою можливо змоделювати виняткові умови і переконатися, що додаток реагує належним чином, забезпечуючи надійну обробку помилок. [11]

Загалом, використання Mockito в нашій стратегії тестування підвищило ефективність та результативність наших тестів. Це дозволило мені ізолювати компоненти, моделювати різні сценарії та перевіряти взаємодію між різними частинами нашого додатку. В результаті я зміг впевнено гарантувати якість і надійність нашого програмного забезпечення перед розгортанням. [11]

```
@Test
public void testGetSubjectsForTeacher() {
    // Mock data
    User user = new User();
    user.setUsername("username");
    Teacher teacher = new Teacher();
    teacher.setId(1L);

    when(teacherRepository.findByUserUsername(user.getUsername())).thenReturn(Optional.of(teacher));
    when(subjectRepository.findAllByTeacher(teacher.getId()))
            .thenReturn(Arrays.asList(
                    MockDataProvider.getSubjectOne(),
                    MockDataProvider.getSubjectTwo()
            )
            );

    // Invoke the method
    List<Subject> subjects = subjectService.getSubjectsForTeacher(user);

    // Verify interactions and assertions
    verify(teacherRepository).findByUserUsername(user.getUsername());
    verify(subjectRepository).findAllByTeacher(teacher.getId());
    assertEquals( expected: 2, subjects.size());
}
```

Рис. 3.4.1 – Метод тестування

Наведений фрагмент коду є модульним тестом, написаним з використанням фреймворку Junit. Цей конкретний тест зосереджений на перевірці функціональності методу «getSubjectsForTeacher()» у класі «subjectService».

Для налаштування тестового сценарію створюються макетні дані. Це включає створення об'єкта User з конкретним іменем користувача та об'єкта Teacher з наперед визначеним «Id». Потім «teacherRepository» і «subjectRepository» налаштовуються за допомогою Mockito для надання певних відповідей при виклику певних методів.

Метод «when()» використовується для визначення поведінки імітованих репозиторіїв. У цьому випадку «teacherRepository.findByUserUsername(user.getUsername())» налаштовано на повернення Optional, що містить об'єкт імітації викладача. Крім того, «subjectRepository.findAllByTeacher(teacher.getId())» повертає список об'єктів

Subject, отриманих від MockDataProvider. Це імітує пошук предметів, пов'язаних з викладачем.

Далі викликається метод «getSubjectsForTeacher()» на екземплярі «subjectService», передаючи об'єкт користувача як аргумент. Очікується, що цей метод поверне список предметів, пов'язаних з викладачем.

Щоб забезпечити очікувану поведінку, взаємодія з імітованими сховищами перевіряється за допомогою методу «verify()». Тест перевіряє, що методи «teacherRepository.findByUserUsername()» і «subjectRepository.findAllByTeacher()» були викликані з очікуваними параметрами.

Нарешті, тест стверджує результат, перевіряючи, що список предметів, отриманий методом «getSubjectsForTeacher()», містить два елементи, які вказують на те, що були знайдені правильні предмети для вчителя. Цей тестовий приклад надає комплексний спосіб перевірки функціональності методу «getSubjectsForTeacher()», гарантуючи, що він коректно взаємодіє зі сховищами і повертає очікувані результати. Це допомагає забезпечити надійність і коректність логіки пошуку предметів у додатку.

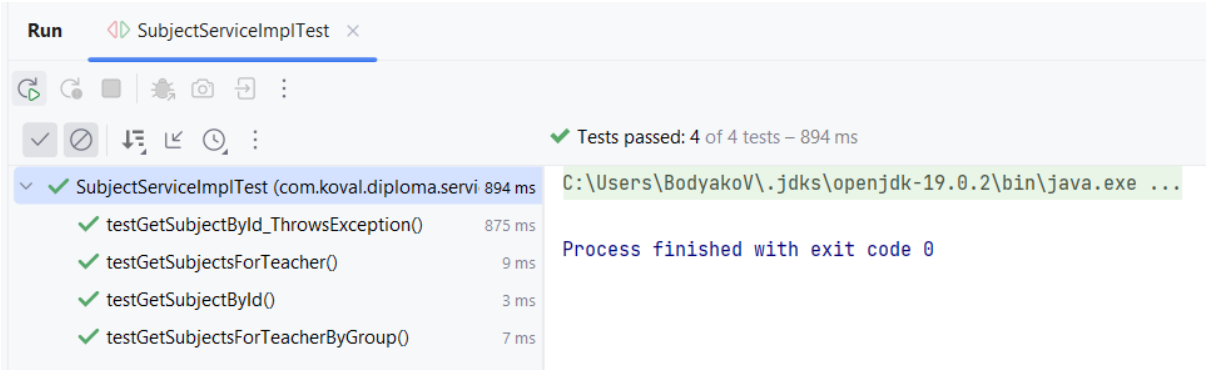


Рис. 3.4.2 – Результат тестування

Як ми можемо бачити на рисунку 3.4.2 всі тести проходять успішно. Це означає, що ми можемо стверджувати, що методи цього сервісу працюють коректно. Також варто відзначити, що якість написання тестових методів можуть впливати на результат роботи в різних сценаріях.

ВИСНОВКИ

У даному дипломному проекті було виконано комплексну розробку програмного забезпечення для автоматизації роботи деканату. Проект складався з кількох етапів, включаючи аналіз предметної галузі, порівняння існуючих аналогів, визначення необхідних інструментів, проектування, реалізації та тестування.

Під час аналізу предметної галузі були вивчені основні процеси та вимоги деканату. Було проведено дослідження наявних аналогів та визначено переваги та недоліки кожного з них. Цей етап допоміг зрозуміти основні вимоги та потреби, що повинні були бути враховані в розробці нової системи.

Визначення необхідних інструментів було важливим етапом перед розробкою. Були вибрані відповідні технології, мови програмування та інструментарій для реалізації системи. Враховуючи потреби проекту, було вибрано підходящі інструменти для роботи з базами даних, розробки інтерфейсу користувача та забезпечення безпеки даних.

Проектування системи було важливим кроком у розробці. Була створена архітектура програмного забезпечення, включаючи структуру бази даних, логіку програми та інтерфейс користувача. Під час проектування були враховані принципи ефективності, безпеки та зручності використання, що сприяло створенню функціонального та зручного інструменту для деканату.

Реалізація проекту включала розробку функціональних модулів, забезпечення взаємодії з базою даних, розробку інтерфейсу користувача та виконання основної логіки програми. Завдяки використанню визначених інструментів та технологій, розроблена система демонструє надійну та ефективну роботу, відповідає поставленим вимогам та задовольняє потреби деканату.

Після завершення реалізації було проведено тестування роботи програмного забезпечення. З метою забезпечення якості та надійності системи, були розроблені тестові сценарії, які включали в себе позитивні та негативні

тести, імітуючи різні сценарії використання системи. Застосовуючи такі інструменти, як Mockito, було проведено автоматизоване тестування функціональності та надійності розробленої системи. Це дозволило виявити та виправити можливі помилки та проблеми, забезпечуючи коректну роботу програмного забезпечення.

У результаті дипломного проекту було успішно розроблено програмне забезпечення для автоматизації роботи деканату. Реалізація проекту дозволила поліпшити ефективність та точність роботи деканату, спростити процеси обробки даних та забезпечити легкий доступ до необхідної інформації. Розроблена система відповідає потребам та вимогам деканату, є стабільною та надійною, а також має потенціал для подальшого розширення та вдосконалення.

У ході виконання даної дипломної роботи були отримані цінні навички в аналізі вимог, проектуванні та розробці програмного забезпечення. Практичне застосування відповідних інструментів та технологій, а також проведення тестування дозволило підтвердити правильність виконання поставлених завдань. Результатом роботи став стабільний та функціональний продукт, який може бути використаний для поліпшення роботи деканату та підвищення його ефективності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Lee R., Chan P. Java Class Libraries Vol. 2: Java. Applet, Java. AWT, Java. Beans. Pearson Education, Limited, 2020. 1712 с.
2. Juneau J. Debugging and Unit Testing. Java 8 Recipes. Berkeley, CA, 2014. С. 249–262.
3. Linwood J., Minter D., Ottinger J. B. Beginning Hibernate: For Hibernate 5. Apress, 2016. 223 с.
4. Sharma R., Gulati S. Java Unit Testing with JUnit 5: Test Driven Development with JUnit 5. Apress, 2017. 151 с.
5. Spring | Why Spring. Why Spring. URL: https://spring.io/why-spring (дата звернення: 16.05.2023).
6. Sharan K. Streams API Updates. *Java 9 Revealed*. Berkeley, CA, 2017. С. 429–439.
7. PostgreSQL 15.3 Documentation. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/ (дата звернення: 16.05.2023).
8. Sharan K., Davis A. L. Beginning Java 17 Fundamentals. Berkeley, CA : Apress, 2022.
9. Mockito framework site. *Mockito framework site*. URL: https://site.mockito.org/ (дата звернення: 16.05.2023).
10. Mockito and JUnit 5 - Using ExtendWith | Baeldung. *Baeldung*. URL: https://www.baeldung.com/mockito-junit-5-extension (дата звернення: 16.05.2023).
11. Porter J., Bueno A. S., Gumbrecht A. Testing Java Microservices: Using Arquillian, Hoverfly, AssertJ, JUnit, Selenium, and Mockito. Manning Publications, 2018. 296 с.
12. Garcia B. Mastering Software Testing with JUnit 5: Comprehensive guide to develop high quality Java applications. Packt Publishing, 2017. 350 с.
13. Walls C. Spring Boot in Action. Manning Publications, 2016. 264 с.

14. Learning Spring Boot 3. 0: Simplify the Development of Production-Grade Applications Using Java and Spring / D. Syer та ін. Packt Publishing, Limited, 2022.
15. Rajput D. Spring 5 Design Patterns: Master efficient application development with patterns such as proxy, singleton, the template method, and more. Packt Publishing, 2017. 396 с.
16. Tudose C. Java Persistence with Spring Data and Hibernate. Manning, 2022. 625 с.
17. Janssen T., Ebersole S. Hibernate Tips: More than 70 solutions to common Hibernate problems. Createspace Independent Publishing Platform, 2017. 256 с.
18. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 624 с.
19. Spilca L. Spring Security in Action. Manning Publications Co. LLC, 2020.
20. Scarioni C., Nardone M. Pro Spring Security: Securing Spring Framework 5 and Boot 2-based Java Applications. Apress, 2019. 428 с.
21. What is a framework? Understanding their purpose, value, development and use - Journal of Environmental Studies and Sciences. *SpringerLink*. URL: https://link.springer.com/article/10.1007/s13412-023-00833-w (дата звернення: 16.05.2023).
22. Top 10 Backend Frameworks In 2023: Which One Is The Best?. *Back4App Blog*. URL: https://blog.back4app.com/backend-frameworks/ (дата звернення: 16.05.2023).
23. Documentation - Thymeleaf. *Thymeleaf*. URL: https://www.thymeleaf.org/documentation.html (дата звернення: 16.05.2023).
24. Deblauwe W. Taming Thymeleaf: Practical Guide to Building a Webapplication with Spring Boot and Thymeleaf. Lulu Press, Inc., 2021.

25. Love C. Progressive Web Application Development by Example: Develop fast, reliable, and engaging user experiences for the web. Packt Publishing, 2018. 354 с.
26. Schildt H. Java: A Beginner's Guide, Ninth Edition. McGraw-Hill Education, 2022. 800 с.
27. Niemeyer P., Loy M., Leuck D. Learning Java: An Introduction to Real-World Programming with Java. O'Reilly Media, Incorporated, 2020. 600 с.
28. Späth P., Sharan K. More Java 17: An in-Depth Exploration of the Java Language Its Features. Apress L. P., 2022.
29. Subramanian H., Raj P. Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs. Packt Publishing, 2019. 378 с.
30. Getting Started | Building a RESTful Web Service. *Getting Started | Building a RESTful Web Service*. URL: https://spring.io/guides/gs/rest-service/ (дата звернення: 16.05.2023).
31. Schonig H.-J. Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13, 4th Edition. Packt Publishing, 2020. 476 с.
32. Riggs S., Ciolli G. PostgreSQL 14 Administration Cookbook: Over 175 Proven Recipes for Database Administrators to Manage Enterprise Databases Effectively. Packt Publishing, Limited, 2022.
33. Viescas J. L. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL (4th Edition). Addison-Wesley Professional, 2018. 960 с.
34. C M. R. Clean code: A Handbook of Agile Software Craftsmanship. Upper Saddle River, NJ : Prentice Hall, 2008. 431 с.
35. Head First Design Patterns: A Brain-Friendly Guide / K. Sierra та ін. O'Reilly Media, Incorporated, 2004. 694 с.
36. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson, 2017. 432 с.

37. Getting started | IntelliJ IDEA. *IntelliJ IDEA Help*. URL: https://www.jetbrains.com/help/idea/getting-started.html (дата звернення: 16.05.2023).
38. Introduction to Project Lombok | Baeldung. *Baeldung*. URL: https://www.baeldung.com/intro-to-project-lombok (дата звернення: 16.05.2023).
39. Rubio D., Long J., Deinum M. Spring 5 Recipes: A Problem-Solution Approach. Apress, 2017. 831 с.
40. Bloch J. Effective Java. Addison-Wesley Professional, 2018. 412 с.
41. Accessing PowerSchool - St. Ignatius College Prep. *St. Ignatius College Preparatory, San Francisco*. URL: https://www.siprep.org/about-us/offices/academic-office/powerschool/accessing-powerschool (дата звернення: 18.05.2023).
42. What is a framework? Understanding their purpose, value, development and use - Journal of Environmental Studies and Sciences. *SpringerLink*. URL: https://link.springer.com/article/10.1007/s13412-023-00833-w (дата звернення: 16.05.2023).
43. Orbis Software Inc.. Easy Grade Pro 4.0® User Manual. URL: https://link.springer.com/article/10.1007/s13412-023-00833-w (дата звернення: 16.05.2023).
44. Oracle. Campus Solutions. *Moved*. URL: https://docs.oracle.com/cd/E52319_01/infoportal/cs.html (дата звернення: 18.05.2023).
45. PowerSchool / Overview. *Charlotte-Mecklenburg Schools / Homepage*. URL: https://www.cmsk12.org/powerschool (дата звернення: 18.05.2023).
46. Unified Modeling Language – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 18.05.2023).

47. Usage and principle of BCryptPasswordEncoder. *Spring Cloud*. URL: https://www.springcloud.io/post/2022-08/bcryptpasswordencoder/#gsc.tab=0 (дата звернення: 18.05.2023).

ДОДАТКИ

Додаток 1

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для автоматизації роботи деканату мовою Java

Виконав студент 4 курсу
групи ПД-44
Коваль Богдан Васильович
Керівник роботи
д.т.н., доц., зав. кафедри ТЦРЖебкаВікторія Вікторівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення окремих робочих процесів деканату за допомогою розробленого програмного забезпечення мовою Java.
- **Об’єкт дослідження** – автоматизація роботи деканату.
- **Предмет дослідження** – програмне забезпечення для автоматизації роботи деканату.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз процесів роботи деканатів та визначити процеси які негативно впливають на ефективність та якість ведення обліку інформації.
2. Проаналізувати існуючі програмні рішення, що використовуються для ведення обліку необхідної інформації для деканатів. Визначити їх переваги і недоліки.
3. Сформулювати вимоги до програмного забезпечення з урахуванням недоліків існуючих засобів.
4. Провести аналіз інструментальних засобів необхідних для створення веб - додатку.
5. Спроектувати схему бази даних для зберігання необхідної інформації для деканату.
6. Розробити механізм ведення обліку кафедр, груп, викладачів, студентів та предметів.
7. Провести тестування розробленої системи .

3

АНАЛІЗ АНАЛОГІВ

Назва	Актуальна АСУ	Easy Grade Pro	PeopleSoft Campus Solution	**Розроблений додаток**
Кросплатформеність	+	-	+	+
Українська локалізація	+	-	-	+
Облік успішності студентів	-	+	+	+
Облік відомостей груп	-	-	-	+
Облік контактних даних студентів та викладачів	-	-	+	+
Облік відвідуваності студентів	-	-	+	+
Управління розкладом	+	-	-	-

4

ВИМОГИ ДО ВЕБ-ДОДАТКУ

1. Рольовий контроль доступу до ресурсів.
2. Облік студентів, викладачів, включаючи їх особисту інформацію, дані про групи, предмети та кафедри.
3. Перегляд даних про групи та студентів, а також ведення обліку занять та успішності студентів .
4. Облік занять та успішності студентів .
5. Зберігання посилань на необхідні ресурси для кожного предмету групи.
6. Захищений доступ до облікових даних.

5

ПРОГРАМНІ ТА ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

ДІАГРАМИ ВАРІАНТІВ ВИКОРИСТАННЯ

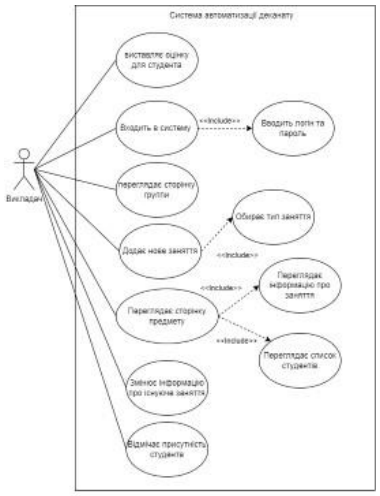
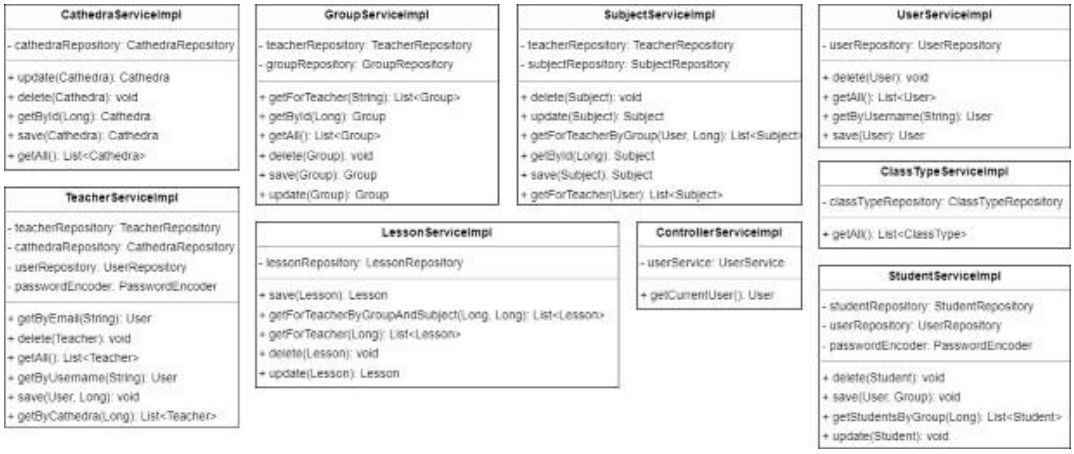


Рис. 7.1 Діаграма варіантів використання користувача «Викладач»

Рис. 7.2 Діаграма варіантів використання користувача «Адміністратор»

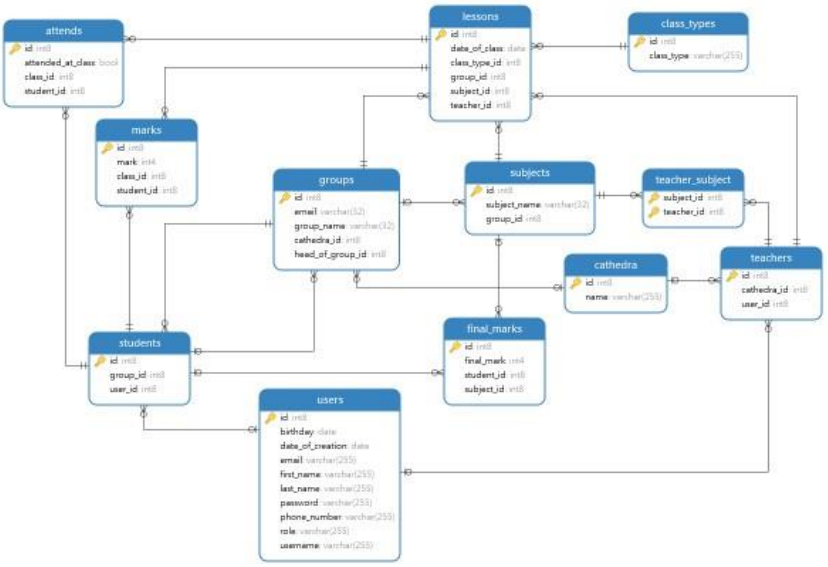
7

ДІАГРАМА КЛАСІВ СЕРВІСІВ



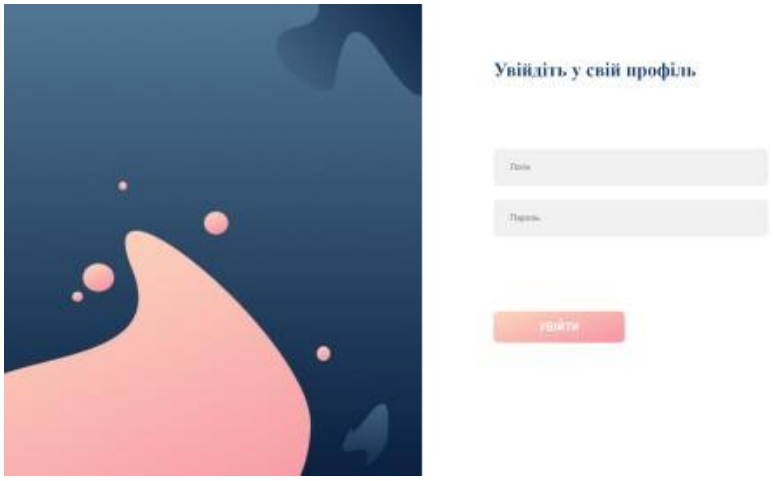
8

ДІАГРАМА СХЕМИ БАЗИ ДАНИХ



9

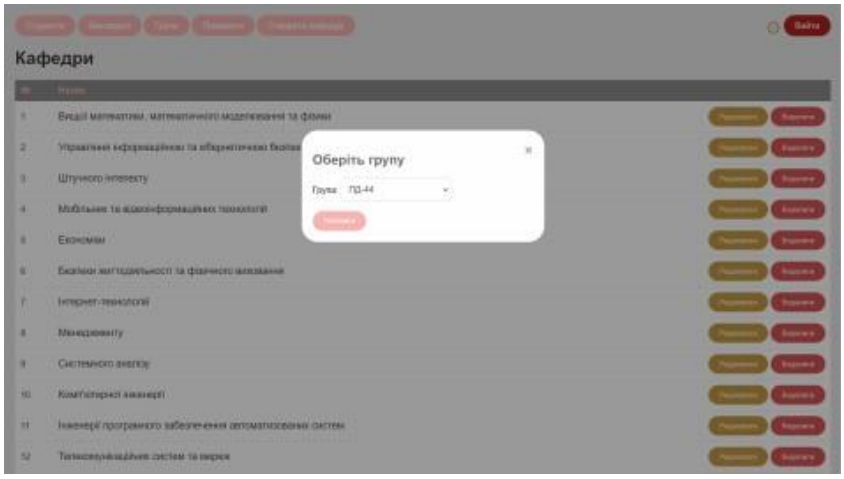
ЕКРАННА ФОРМА



Сторінка входу

10

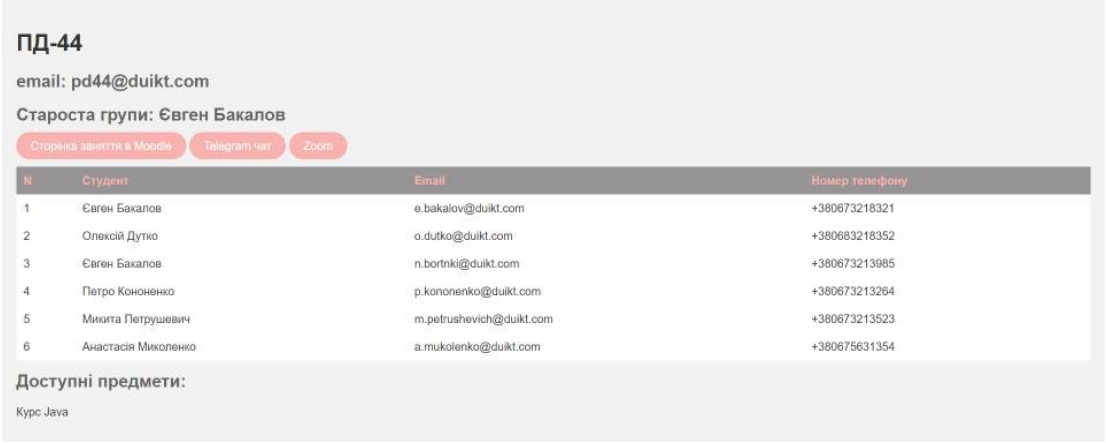
ЕКРАННА ФОРМА



Модальне вікно на головній сторінці адміністратора

11

ЕКРАННА ФОРМА



Сторінка групи для викладача

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Коваль Б.В. Розробка програмного забезпечення для автоматизації роботи деканату мовою програмування Java / В.В. Жебка , Б.В. Коваль // Сучасні інтелектуальні інформаційні технології в науці та світі: Матеріали третьої Всеукраїнської науково-практичної конференції . Збірник тез, 16.05.22, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 25 – 26.
- Коваль Б.В. Аналіз Spring Framework як найпопулярнішого фреймворку для Java / В.В. Жебка , Б.В. Коваль // Сучасні інтелектуальні інформаційні технології в науці та світі: Матеріали третьої Всеукраїнської науково-практичної конференції . Збірник тез, 16.05.22, ДУТ, м. Київ – К.: ДУТ, 2023. – С. 33 – 34.

ВИСНОВКИ

1. Проаналізовано існуючі аналоги та їх переваги і недоліки.
2. Проведено аналіз засобів розробки веб-додатків. Одним з засобів було обрано Spring Framework як ключовий фреймворк для серверної частини додатку.
3. Спроектовано архітектуру веб-додатку та застосовано патерни проектування Ключовим патерном розробленого веб-додатку є MVC.
4. Спроектовано схему бази даних яка включає в себе 11 таблиць.
5. Розроблено механізм ведення обліку викладачів, студентів, груп, кафедр та предметів, а також забезпечено захист цих даних.
6. Розроблено механізм ведення обліку успішності студентів.
7. Реалізовано рольовий контроль доступу для викладачів а адміністраторів.
8. Протестовано роботу веб-додатку за допомогою бібліотеки Junit 5 та фреймворку Mockito.