НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

До захисту допущено:

Завідувач кафедри

_____ Надія БУРАУ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерно - інтегровані технології та системи навігації і керування»

спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: Веб-додаток «Автоматизована система оцінки якості навчального процесу»

Виконав:
студент III курсу, групи ПГ-п81
Скідченко Олександр Анатолійович _____

Керівник:

Доцент кафедри ПСОН, к.т.н., доц. Цибульник С.О. _____

Рецензент:
Асистент Драчук О.О. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп’ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютерно-інтегровані технології та системи навігації і керування»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Н.І. Бурау

«___»_____2021 р.

ЗАВДАННЯ

на дипломну роботу студенту

Скідченку Олександру Анатолійовичу

1. Тема роботи «Автоматизована система оцінки якості навчального процесу», керівник роботи Цибульник Сергій Олексійович, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи 07.06.2021р.

3. Вихідні дані до роботи: реалізувати систему у вигляді мобільного, десктопного або веб-додатку, реалізувати можливість оцінювання викладачів студентами та/або студентів викладачами.

4. Зміст роботи:

1. Огляд стану проблеми.
2. Огляд та вибір технологій для реалізації автоматизованої системи.
3. Формування та формалізація вимог до системи.
4. Тестування розробленої системи.
5. Розгортання готового програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): таблиці, графіки, рисунки, тощо.

6. Дата видачі завдання 20.04.2021р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
	Огляд стану проблеми	27.04.2021	
	Огляд та вибір технологій для реалізації автоматизованої системи	04.05.2021	
	Формування та формалізація вимог до системи	14.05.2021	
	Тестування розробленої системи	21.05.2021	
	Розгортання готового програмного продукту	31.05.2021	
	Оформлення пояснювальної записки	07.06.2021	

Студент О.А. Скідченко

Керівник С.О. Цибульник

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломної роботи складається з трьох розділів, містить 9 рисунків, 32 таблиці, 23 джерела.

Мета. Дипломна робота присвячена розробці WEB-додатку для оцінювання якості навчального процесу шляхом оцінювання студентами роботи викладачів та навпаки, з метою покращення якості освітнього процесу навчального закладу, отримання актуальної інформації про здібності та успішність учасників освітнього процесу, зменшення грошових та трудових витрат на організацію опитувань.

Основними задачами є створення працездатної системи, що буде мати можливість до розширення, додавання нового функціоналу у разі потреби, достатньо дружній інтуїтивно зрозумілий користувацький інтерфейс.

У розділі аналізу вимог до програмного продукту були поставлені цілі та задачі, визначені основні функціональні та нефункціональні вимоги, означені призначення та ціль розробки додатку та актуальність проблеми, що вирішується.

У розділі конструювання програмного забезпечення була проведена декомпозиція предметної області, розглянуті основні ролі користувачів у додатку, описані технології для розробки WEB-додатків та їх можливі аналоги, розглянуті можливі підходи до розробки додатків та обраний найбільш підходящий для цілей роботи.

У розділі випробувань програмного забезпечення було виконано тестування додатку з допомогою мануального тестування, описані розповсюджені підходи до тестування WEB-додатків та технології, що для цього використовуються. Був описаний процес розгортування додатку.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, АВТОМАТИЗОВАНА СИСТЕМА, ОЦІНЮВАННЯ ЯКОСТІ НАВЧАЛЬНОГО ПРОЦЕСУ.

ABSTRACT

Structure and scope of work. The explanatory note of the work consists of three sections, contains 9 figures, 32 tables, 23 sources.

Goal. Work is devoted to the development of WEB-application for assessing the quality of the educational process by assessing the work of teachers and vice versa, in order to improve the quality of the educational process, obtain relevant information about the abilities and successability of participants in the educational process.

The main tasks are to create a workable system that will be able to expand, add new functionality if necessary, a fairly friendly intuitive user interface.

The section of the analysis of the requirements for the software product set goals and objectives, defined the main functional and non-functional requirements, defined the purpose and purpose of application development and the relevance of the problem to be solved.

In the software design section the subject area was decomposed, the main roles of users in the application were considered, technologies for WEB-applications development and their possible analogues were described, possible approaches to application development were considered and the most suitable one was selected.

In the software testing section, the application was tested using manual testing, and common approaches to testing WEB applications and the technologies used for this purpose were described. The application deployment process was described.

KEY WORDS: WEB APPLICATION, AUTOMATED SYSTEM, ASSESSMENT OF THE QUALITY OF THE LEARNING PROCESS.

Пояснювальна записка
до дипломної роботи

на тему: Веб-додаток «Автоматизована система оцінки якості навчального процесу»

Київ – 2021 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ,СКОРОЧЕНЬ І ТЕРМІНІВ

Фреймворк (framework з англ. – «каркас») – програмний каркас, який містить ряд уже реалізованих рішень для розповсюджених задач.

База даних (БД) – сховище для зберігання та операцій з структурованими (зазвичай у виді таблиць та відношень між ними) даними. Також використовується як синонім до Системи Керування Базою Даних (СКБД).

Залежність елементу – вид відношень між елементами системи, в якому один елемент використовує інший через його публічний інтерфейс для виклику логіки, що реалізована в цільовому елементі.

SOLID – акронім, що представляє собою 5 основних принципів об’єктно-орієнтованого програмування, описаних Робертом Мартіним.

SRP (Single Responsibility Principle) – перший з принципів SOLID, який каже що «кожен елемент повинен мати лише одну вісь зміни» (змінюватися лише за однієї причини), або «кожен елемент має виконувати лише одну функцію».

ORM фреймворк – фрефмворк що дозволяє проводити операції з сутностями в базі даних (видалення, створення, редагування, збереження), використовуючи їх представлення у виді об'єкту мови програмування. Дозволяє звести до мінімуму використання мови SQL або зовсім позбутися її.

JPA (Java Persistence API)– специфікація Java для ORM-фреймворків. Являє собою додатковий рівень абстракції між додатком та ORM-фреймворком.

Persistence provider – поняття що використовується для означення ORM-фреймворку, що використовується як реалізація JPA в конкретному проекті.

JDBC (Java DataBase Connectivity) – бібліотека Java, що дозволяє під’єднати додаток до баз даних та оперувати даними на найнижчому можливому рівні абстракції.

Hibernate – найперший та один з найбільш поширених ORM-фреймворків для мови Java.

Big O нотація – математична нотація, що використовується для означення часової та просторової складності алгоритмів.

FIRST – набір принципів, дотримання яких дозволяє писати ефективні модульні тести, що легко підтримуються.

JSON (JavaScript Object Notation) – спосіб представлення об'єктів у виді набору числових та строкових полів, що легко передавати з однієї системи в іншу.

IOC (Inversion Of Control) – принцип у програмуванні, при якому клієнтський код (бізнес-код) не керує процесом виконання програми, а викликається фреймворком у визначений момент.

SQL (Structured Query Language) – декларативна мова програмування, призначена для спілкування програм з реляційними базами даних..

DDL (Data Definition Language) – один з розділів SQL, що використовується для створення таблиць та відношень між ними.

Frontend – частина додатку, що відповідальна за відображення інформації в браузері користувача, прийом інформації від нього та відправлення її на серверну частину.

Backend – серверна частина додатку, що відповідальна за операції з даними, їх перетворенням, спілкування з базами даних, тощо.

ВСТУП

У час, коли діджиталізація відбувається буквально всюди [1], така велика і важлива сфера як освіта не може залишитись незмінною. З кожним роком кількість інформації, яка використовується в різноманітних алгоритмах росте експоненціально, оскільки сфера освіти, як і будь-яка інша сфера діяльності, прагне до покращення результату при мінімізації затрат та кількості людино-годин, необхідних для виконання деякої корисної роботи. Це саме та точка, в якій діджиталізація, або перехід всіх видів інформації, якою володіє система у цифрову форму, підвищує ефективність системи на порядки, при відносно невеликих затратах на перехід. При цьому діджиталізована система отримує всі переваги автоматичної системи, а саме [2]:

- мінімізація кількості персоналу, що обслуговує систему;
- використання сучасних технологій обчислення та обробки даних, що підвищує продуктивність системи на порядки;
- можливість системи працювати без втручання людини;
- перекладання деяких відповідальностей, які можна алгоритмізувати, з людського персоналу на комп’ютерну програму, що підвищує ефективність системи у всіх можливих площинах і т.д.

Однією з важливих частин освітнього процесу є перехресний збір інформації про студентів та викладачів, оскільки під час винесення деяких рішень (наприклад, продовження договору з викладачем, чи рішення про відрахування студента) важливо мати достовірну інформацію про якість виконання суб’єктом його обов’язків, щоб рішення було справедливим і базувалось на актуальній інформації.

Для цього можна використовувати автоматизовані системи, метою яких було б збирати інформацію від студентів та викладачів, які працюють один з одним для періодичного отримання актуальної інформації про їх успішність.

Мета створення даної роботи — автоматизація процесу перехресного оцінювання студентів та викладачів з метою фіксації їх успіхів у виконанні безпосередніх обов’язків та для подальшого використання цієї інформації.

Завданням даної роботи є створення клієнт-серверного додатку з WEB-інтерфейсом, що буде надавати можливість студентам оцінювати викладачів та навпаки по змінюваному ряду параметрів в режимі сесій (тобто голосування буде доступне в деякий чітко визначений час).

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ПРОДУКТУ

1.1.Загальні положення

Сесія оцінювання викладачів та студентів – деякий проміжок часу, під час якого відбувається оцінювання викладачів студентами та навпаки по ряду параметрів з метою централізованого збору інформації про їх ефективність, відповідальність, якість, швидкість роботи та інше.

Впровадження таких заходів з допомогою не-цифрових засобів, як то збір всіх учасників на кафедрах/факультетах з метою їх опитування тягне за собою велику кількість адміністративної роботи, потребу особистої присутності опитуваних, великі трати часу організовуючого персоналу, забезпечення анонімності та багато інших факторів, які сильно збільшують об’єм робіт для впровадження та постійного виконання такого роду заходів.

Використання ж автоматизованої програмної системи з веб-інтерфейсом здатне повністю прибрати такі етапи як збір групи для опитування, забезпечення анонімності, підрахунок та ведення статистики, організація нових сесій голосування, збереження даних минулих опитувань і т.д., що дозволяє суттєво зменшити витрати часу та грошей на регулярне проведення таких заходів, що в свою чергу позитивно сказується на якості навчального процесу, при доцільному використанні отриманої інформації.

Окрім цього сучасні мови програмування передбачають велику кількість готових рішень для розробки часто використовуваних елементів програмного забезпечення, як то клієнт-серверні додатки з веб-інтерфейсом, додатки з можливостями авторизації та автентифікації, модулі підключення до баз даних, інтеграції з іншими веб-сервісами та багато інших, що значно зменшує необхідну кількість зусиль для впровадження програмного продукту в експлуатацію з мінімальним необхідним функціоналом. Це також значить що програмний продукт швидше окупить себе.

Також важливим фактором для розробки програмного продукту є вибір виду кінцевого результату. На сьогодні є популярними 2 варіанти розробки програмного забезпечення. Кожен з підходів має свої недоліки та переваги [3,4]:

1. Desktop розробка, або створення комп'ютерного додатку що потребує встановлення на пристрій, на якому буде використовуватись. Переваги:
    - потреба передавати по мережі лише нову інформацію, або ту, що потребує оновлення;
    - можливість зберігати на пристрої значні об'єми даних, що позитивно сказується на витраті інтернет-трафіку та швидкості роботи програми;
    - швидкий доступ до збережених даних, адже читання та запис даних на локальний носій на порядки швидше ніж передача по мережі інтернет;
    - можливість використовувати обчислювальні потужності пристрою для перекладання деяких задач з сервера на пристрій користувача (так званий товстий клієнт), що зменшує навантаження на сервери та збільшує кількість одночасно оброблюваних запитів.

    Недоліки:

    - велика вартість написання програми для декількох платформ, оскільки для кожної нової знадобиться новий дистрибутив.
    - проблеми безпеки, оскільки велика кількість інформації зберігається на кожному екземплярі пристрою, що використовує програму;
    - можливі потреба в значних обчислювальних потужностях пристрою;
    - можливі випадки використання невалідних даних.
2. WEB-додаток, тобто написання однієї великої програми (серверного додатку), що буде обробляти всі запити від усіх користувачів. У більшості параметрів вона є протилежною десктоп-програмам.

    Переваги:

- надзвичайно низькі потреби до апаратного забезпечення користувача (тонкий клієнт), оскільки абсолютна більшість операцій та обчислень проводиться на стороні серверу, а пристрій користувача лише відправляє запити, отримує та відображає результат;
- практично відсутня потреба в пам’яті на пристрої, оскільки для всіх веб-запитів може використовуватись один браузер;
- відсутність потреби встановлювати дистрибутив додатку на пристрій користувача, оскільки для взаємодії з веб-додатком використовується HTTP-протокол. Цей протокол може використовуватись будь-яким сучасним браузером, що робить можливим роботу з додатком на будь-якій платформі, для якої написаний хоча б один браузер;
- більша у порівнянні з десктопними програмами безпека, оскільки на пристрої користувача зберігається лише інформація, необхідна для відображення уже отриманого запиту та відправлення наступного.

Недоліки:

- необхідність купувати або арендувати великі сервери, оскільки всі обчислення проводяться саме на стороні серверу. Останнім часом [5] з поширенням хмарних технологій недолік почав нівелюватись, оскільки оренда віртуальних машини в великих компаніях-спеціалістах стає все більш і більш дешевою;
- потреба постійного підключення до інтернету;
- неможливість підлаштувати інтерфейс додатку під всі існуючі типи пристроїв (телефони, ноутбуки, планшети, телевізори), внаслідок чого деякі пристрої будуть некоректно відображати елементи інтерфейсу;

- великий ступінь централізації даних, через що потрібно постійно робити резервні копії баз даних та витрачати ресурси на їх підтримку.

Звертаючи увагу на специфіку програми, а саме відсутність надзвичайно великих об'ємів даних, відсутність об'ємних обчислень, які можна було б виконувати на пристрої користувача, необхідність централізованого зберігання свіжої інформації та надання її користувачам, для виконання даного проекту було обрано WEB-додаток.

1.2. Аналіз основних ролей та сутностей додатку

Система повинна представляти собою клієнт-серверний додаток, що здатний приймати запити на голосування від клієнта, перераховувати рейтинг об’єкта голосування та відображати рейтинг об’єктів голосування у вигляді діаграм.

Щонайперше, в системі повинно бути три ролі – по одній на кожну сторону, що взаємодіє:

1. Адміністратор, що відповідальний за адміністраційні та менеджерські дії як то додавання нових студентів, викладачів, запуск сесій голосування, створення, видалення та вибір критеріїв голосування, тощо.
2. Викладач, що має право голосувати за студентів, що закріплені за ним через групи та переглядати статистику цих студентів та свою статистику.
3. Студент, що має право голосувати за своїх викладачів, переглядати їх статистику та свою статистику.

Ще однією роллю є гість, тобто незареєстрований користувач, але він не має доступу нікуди, окрім як до сторінки входу в систему і тому в подальших розділах згадуватись не буде.

Перерахунок рейтингу відбувається після отримання HTTP-запиту від браузера клієнта, в якому знаходяться рейтинги для всіх критеріїв оцінювання, що наявні в поточній сесії голосування. Якщо сесія неактивна, додаток повинен повернути сторінку з помилкою.

Сесія голосування – це деякий період часу, під час якого можливе голосування студентів та викладачів один за одного по визначеному списку критеріїв. Список критеріїв вказується адміністратором під час створення та запуску сесії.

Рейтинг об’єкта оцінювання вираховується знаходженням середнього арифметичного між всіма голосами, що були віддані за нього. Формула для оновлення рейтингу об’єкта: $$R_N = \frac{(R_O \bullet n) + R_V}{n+1},$$

де R_N - рейтинг після зарахування голосу,

R_O - рейтинг до зарахування голосу,

n - кількість голосів до зарахування голосу,

R_V - рейтинг голосу, що зараховується.

Після перерахування рейтингу, перерахування також потребує діаграма, яка прив’язана до об’єкта голосування та до поточної сесії.

Також важливою частиною системи голосування є забезпечення анонімності всіх голосуючих (неможливість дізнатись, хто за кого і як проголосував) для забезпечення більш об’єктивних оцінок та відсутності упередженості викладачів до студентів після сесії голосування.

1.2.1. Функціональні вимоги

У системі передбачено наступні варіанти використання:

Таблиця 2.1 Варіант використання UC001

Назва	Аутентифікація в системі.
Опис	Аутентифікація користувача в системі.
Учасники	Користувач.
Передумови	Відкрита сторінка аутентифікації.
Постумови	Користувача аутентифіковано в системі. Перехід на сторінку вибору користувачів для оцінювання.
Основний сценарій	Користувач вводить дані для входу та натискає кнопку аутентифікації.

Таблиця 2.2 Варіант використання UC002

Назва	Вибір викладача для голосування.
Опис	Вибір студентом викладача для голосування із списку доступних.
Учасники	Студент.
Передумови	Відкрита сторінка списку викладачів.
Постумови	Перехід на сторінку голосування за викладача.
Основний сценарій	Студент обирає потрібного викладача із списку та натискає на нього.

Таблиця 2.3 Варіант використання UC003

Назва	Голосування за викладача.
Опис	Голосування студента за викладача.
Учасники	Студент.
Передумови	Відкрита сторінка голосування за викладача.
Постумови	Студент проголосував за викладача. Рейтинг викладача перераховано. Студент більше не здатний голосувати за викладача в поточній сесії. Перехід на сторінку вибору викладачів. Показ повідомлення про успішне зарахування голосу.
Основний сценарій	Студент натискає на обрані оцінки по кожному з критеріїв та натискає кнопку відправки даних на обробку.

Таблиця 2.4 Варіант використання UC004

Назва	Перегляд рейтингу викладача.

Опис	Студент переглядає рейтинг викладача по критеріям поточної сесії.
Учасники	Студент.
Передумови	Відкрита сторінка списку викладачів.
Постумови	Студент знаходиться на сторінці з рейтингом обраного викладача. На екрані знаходиться діаграма рейтингу.
Основний сценарій	Студент обирає потрібного викладача та натискає на кнопку «Рейтинг» біля нього.

Таблиця 2.5 Варіант використання UC005

Назва	Перегляд власного рейтингу користувача.
Опис	Користувач переглядає власну діаграму рейтингу по критеріям поточної сесії.
Учасники	Студент, викладач.
Передумови	Відкрита будь-яка сторінка.
Постумови	Користувач знаходиться на сторінці з власним рейтингом.
Основний сценарій	Користувач обирає в верхньому меню на сторінці кнопку «Мій рейтинг».

Таблиця 2.6 Варіант використання UC006

Назва	Пошук студента за критеріями.
Опис	Пошук студентів для голосування за критеріями.
Учасники	Викладач.

Передумови	Відкрита сторінка списку студентів.
Постумови	Оновлення списку студентів тими, що підпадають під обраний критерій.
Основний сценарій	Викладач обирає необхідний критерій та натискає кнопку «Оновити».

Таблиця 2.7 Варіант використання UC007

Назва	Вибір студента для голосування.
Опис	Вибір студента викладачем для голосування із списку доступних.
Учасники	Викладач.
Передумови	Відкрита сторінка списку студентів.
Постумови	Перехід на сторінку голосування за студента.
Основний сценарій	Викладач обирає потрібного студента із списку та натискає на нього.

Таблиця 2.8 Варіант використання UC008

Назва	Голосування за студента.
Опис	Голосування викладача за студента.
Учасники	Викладач.
Передумови	Відкрита сторінка голосування за студента.
Постумови	Викладач проголосував за студента. Рейтинг студента перераховано. Викладач більше не здатний голосувати за студента в поточній сесії. Перехід на сторінку вибору студентів. Показ повідомлення про успішне зарахування голосу.

Основний сценарій	Викладач натискає на обрані оцінки по кожному з критеріїв та натискає кнопку відправки даних на обробку.

Таблиця 2.9 Варіант використання UC009

Назва	Перегляд рейтингу інших користувачів.
Опис	Перегляд користувачем діаграм рейтингу інших користувачів.
Учасники	Користувач.
Передумови	Відкрита сторінка списку користувачів для голосування.
Постумови	Користувач переходить на сторінку перегляду рейтингу обраного користувача.
Основний сценарій	Користувач натискає на кнопку «Рейтинг» біля обраного користувача в списку доступних.

Таблиця 2.10 Варіант використання UC010

Назва	Перехід на сторінку створення нового користувача.
Опис	Перехід на сторінку створення адміністратором нового користувача.
Учасники	Адміністратор.
Передумови	Відкрита будь-яка сторінка.
Постумови	Адміністратор переходить на сторінку створення нового користувача.
Основний сценарій	Адміністратор натискає на кнопку «Новий користувач» на верхній панелі сторінки.

Таблиця 2.11 Варіант використання UC011

Назва	Створення нового користувача
Опис	Створення адміністратором нового користувача для надання йому доступу в систему.
Учасники	Адміністратор.
Передумови	Відкрита сторінка створення нового користувача.
Постумови	В системі створений новий користувач. Сторінка оновлюється, з’являється оповіщення про успішне створення нового користувача.
Основний сценарій	Адміністратор заповнює всі форми для створення нового користувача та натискає кнопку «Створити користувача».

Таблиця 2.12 Варіант використання UC012

Назва	Перехід на сторінку операцій з сесіями.
Опис	Адміністратор переходить на сторінку з операціями з сесіями голосування.
Учасники	Адміністратор.
Передумови	Відкрита будь-яка сторінка.
Постумови	Адміністратор переходить на сторінку операцій з сесіями.
Основний сценарій	Адміністратор натискає на кнопку «Сесії» на верхній панелі сторінки.

Таблиця 2.13 Варіант використання UC013

Назва	Створення нової сесії голосування.
Опис	Адміністратор створює нову сесію голосування.
Учасники	Адміністратор.
Передумови	Відкрита сторінка операцій з сесіями. Відсутня поточна сесія.
Постумови	Створена нова сесія голосування з заданими часовими рамками та заданими критеріями для оцінювання. Сторінка оновлюється.
Основний сценарій	Адміністратор обирає список необхідних критеріїв для оцінювання студентів та викладачів, задає часові рамки для сесії та натискає кнопку «Створити сесію».

Таблиця 2.14 Варіант використання UC014

Назва	Перехід на сторінку створення нового критерію.
Опис	Адміністратор переходить на сторінку нового критерію для голосування.
Учасники	Адміністратор.
Передумови	Відкрита сторінка операцій з сесіями. Відсутня поточна сесія.
Постумови	Перехід на сторінку створення нового критерію.

Основний сценарій	Адміністратор натискає на кнопку «Новий критерій» біля списку.

Таблиця 2.15 Варіант використання UC015

Назва	Створення нового критерію голосування.
Опис	Адміністратор вводить дані та створює новий критерій для голосування.
Учасники	Адміністратор.
Передумови	Відкрита сторінка створення нового критерію.
Постумови	Створений новий критерій для голосування для студента або викладача. Новий критерій доданий в список доступних для вибору під час створення сесії. Перехід на сторінку операцій з сесіями.
Основний сценарій	Адміністратор вводить назву критерію, обирає для кого цей критерій (студент чи викладач) та натискає кнопку «Створити критерій».

Функціональні вимоги:

Таблиця 2.16 Функціональна вимога REQ001

Номер	REQ001
Назва	Аутентифікація в системі
Опис	Система повинна надавати неаутентифікованому користувачу аутентифікуватись в системі з допомогою логіну і паролю.

Таблиця 2.17 Функціональна вимога REQ002

Номер	REQ002

Назва	Вихід із системи
Опис	Система повинна надавати аутентифікованому користувачу вийти із системи шляхом натискання кнопки «Вихід».

Таблиця 2.18 Функціональна вимога REQ003

Номер	REQ003
Назва	Голосування пошук студентів викладачем
Опис	Система повинна надавати користувачам з роллю «Викладач» шукати користувачів з роллю «Студент», що належать до груп, до яких приєднаний викладач.

Таблиця 2.19 Функціональна вимога REQ004

Номер	REQ004
Назва	Голосування пошук викладачів студентами
Опис	Система повинна надавати користувачам з роллю «Студент» шукати користувачів з роллю «Викладач», що приєднані до групи, в якій знаходиться студент.

Таблиця 2.20 Функціональна вимога REQ005

Номер	REQ005
Назва	Голосування студента за викладача
Опис	Система повинна надавати користувачам з роллю «Студент» голосувати за користувачів з роллю «Викладач», що приєднані до групи, в якій знаходиться студент.

Таблиця 2.21 Функціональна вимога REQ006

Номер	REQ006
Назва	Голосування викладача за студента
Опис	Система повинна надавати користувачам з роллю «Викладач» голосувати за користувачів з роллю «Студент», що належать до груп, до яких приєднаний викладач.

Таблиця 2.22 Функціональна вимога REQ007

Номер	REQ007
Назва	Перегляд власного рейтингу викладачами та студентами
Опис	Система повинна надавати користувачам з ролями «Викладач» та «Студент» переглядати свої діаграми рейтингів шляхом переходу на спеціальну сторінку.

Таблиця 2.23 Функціональна вимога REQ008

Номер	REQ008
Назва	Перегляд рейтингу студентів викладачем
Опис	Система повинна надавати користувачам з ролями «Викладач» переглядати діаграми рейтингів студентів з приєднаних до нього груп шляхом переходу на спеціальну сторінку.

Таблиця 2.24 Функціональна вимога REQ009

Номер	REQ009
Назва	Перегляд рейтингу викладачів студентами
Опис	Система повинна надавати користувачам з ролями «Студент» переглядати діаграми рейтингів викладачів, які

	приєднані до групи, в якій знаходиться студент шляхом переходу на спеціальну сторінку.

Таблиця 2.25 Функціональна вимога REQ010

Номер	REQ010
Назва	Створення нової сесії голосування
Опис	Система повинна надавати користувачам з роллю «Адміністратор» можливість створювати сесію голосування на визначений період часу з заданим списком критеріїв для викладачів та студентів.

Таблиця 2.26 Функціональна вимога REQ011

Номер	REQ011
Назва	Створення та видалення критеріїв голосування
Опис	Система повинна надавати користувачам з роллю «Адміністратор» можливість створювати нові критерії для голосування за викладачів та студентів та видаляти старі критерії.

Таблиця 2.27 Функціональна вимога REQ012

Номер	REQ012
Назва	Заборона голосування декілька разів одним користувачем за іншого в межах однієї сесії голосування
Опис	Система повинна запобігати реєстрації декількох голосів одного користувача на користь іншого в межах однієї сесії голосування.

Таблиця 2.28 Функціональна вимога REQ013

Номер	REQ013
Назва	Заборона голосування користувачам, які не мають відношення один з одним через посередництво групи (студент не може голосувати за викладача, який не викладає у нього жодного предмету)
Опис	Система повинна запобігати спробам користувача проголосувати за іншого користувача окрім випадків, коли ці користувачі знаходяться в відносинах студент – група – викладач або навпаки.

1.2.2. Нефункціональні вимоги

Нефункціональні вимоги, що ставляться до додатку [6]:

- додаток повинен бути доступним для користувачів щонайменше під час сесій голосування;
- повинен бути розширюваним, у разі потреби додавання нового функціоналу;
- повинен легко підтримуватися розробниками;
- повинен мати інтуїтивно зрозумілий користувацький інтерфейс;
- повинен буди достатньо дрібно поділений на невеликі підсистеми;
- додаток повинен розгортуватися на одному з безкоштовно доступних контейнерів сервлетів, наприклад Tomcat або Jetty;
- сторінки додатку повинні коректно відображатись у наступних браузерах: Mozilla Firefox, Opera, Google Chrome, Microsof Edge.

1.3.Постановка задачі

1.3.1. Призначення програмного продукту

Даний програмний продукт призначений для учасників освітнього процесу, а саме викладачів, студентів та керівництва кафедр або університетів (адміністраторів) для збору інформації про професійні успіхи студентів та викладачів, а саме для зменшення трудових, часових та грошових затрат на організацію та проведення заходів контролю якості освітнього процесу.

1.3.2. Цілі програмного продукту

Основною ціллю програмного продукту є зменшення затрат часу та ресурсів на організацію та проведення заходів з оцінювання якості освітнього процесу.

Для досягнення поставленої цілі необхідно спроектувати та реалізувати наступні програмні функції:

1. Вхід (аутентифікація) та вихід користувача з системи;
2. Систему реєстрації голосів користувачів та перерахунку даних рейтингу;
3. Систему пошуку користувачів;
4. Систему створення та налаштування сесій голосування;
5. Систему редагування критеріїв оцінювання студентів та викладачів;
6. Систему реєстрації адміністратором нових користувачів у системі.
7. Систему відображення чисельних даних рейтингів у виді діаграм;
8. Можливість додавання нових користувачів адміністратором.

1.4.Аналіз актуальності проблеми

Проблема існування рейтингової системи для об’єктивного та анонімного оцінювання викладачів студентами та навпаки має багато аспектів: педагогічний, соціальний, економічний, в рамках яких рейтингова система допомагає примати об’єктивні рішення по управлінню персоналом, зміцненню кадрового потенціалу, атестацією викладачів та студентів, тощо [7].

Також така система, будучи автоматизованою стає більш доступною для широкого спектру університетів, інститутів, шкіл і т.д., оскільки для її постійного використання не потрібно витрачати час та ресурси адміністрації навчального закладу, а з розвитком хмарних технологій відпала потреба навіть у покупці чи оренді власного серверу. Додаток можна просто завантажити на хмарний сервіс та користуватись, що зменшує затрати на підтримку такої системи до абсолютного мінімуму. Теоретично, наявність такої системи, доступної для кожної освітньої установи може покращити якість освіти у навчальних закладах, що її використовують і в країні в цілому.

Окрім цього, важливою перевагою даної системи є можливість для викладачів оцінювати студентів, чого в аналогах, наскільки показали пошуки, не реалізовано. Така можливість є безумовним плюсом, оскільки таким чином система буде забезпечувати керівництво університету додатковими даними про студентів, анонімними та актуальними, що допоможе приймати деякі рішення більш об’єктивно та не передвзято, як то рішення, чи відраховувати студента може бути переглянутим, якщо викладачі оцінюють його як хорошого студента.

2 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз технологій для побудови програмного забезпечення

Перш за все перед початком створення програмного забезпечення необхідно визначити основні вимоги до технологій, що будуть використані в подальшому. Для побудови WEB-додатків основними мовами програмування э Java та C# [9]. Обидві ці мови об'єктно-орієнтовані, строго типізовані, «із коробки» підтримують багатопотоковість (одночасна обробка великої кількості запитів від користувачів), що є дуже важливим для побудови клієнт-серверних додатків.

Також важливою частиною є наявність у мові програмування фреймворків для використання уже виконаних програмних рішень, як то доступ до бази даних для програми, серверів додатків для обробки HTTP-запитів від клієнтів та відправлення їм відповідей, реалізації валідації даних від користувача, реалізації патерну MVC (Model-View-Controller), тощо.

Для побудови додатку для даного дипломного проекту була обрана мова програмування Java версії 15, оскільки для неї уже написано більшість модулів, бібліотек та фреймворків, що використовуються сьогодні для побудови клієнт-серверних додатків, код написаний на цій мові дуже строго структурований, що дозволяє витрачати менше часу на розуміння коду та бізнес-логіки додатку програмістами команд підтримки та подальшої розробки, та, що також дуже важливо, на цій мові написана абсолютна більшість WEB-додатків великого масштабу, що дуже спрощує процес інтеграції розроблюваного додатку з вже готовими продуктами.

Основою для додатку обрано фреймворк Spring Boot 2.4, що включає в себе всі необхідні базові компоненти для створення додатків та дає можливість підключати додаткові модулі-фреймворки до готової основи.

Сервером для розгортування та тестування додатку обрано Embedded Apache Tomcat, що іде в комплекті зі Spring Boot 2.4, хоча додаток можна розгорнути і на будь якому іншому сервері, що підтримує .WAR формат файлів.

Базою даних обрана реляційна база даних MySQL, оскільки вона безкоштовна, підтримує всі розповсюджені операції з даними в таблицях і має непогану швидкодію [10].

Для реалізації шаблону MVC використовується єдине рішення для Spring-додатків – Spring MVC, призначенням якого є розділення процесу обробки запиту на 3 частини: модель (дані), контролер (сутність, що обробляє дані та модифікує їх) та відображення (заповнюється даними з моделі та відправляється користувачу в браузер у виді HTTP-відповіді), у результаті чого досягається менша зв'язаність коду та виконується принцип SRP.

Для забезпечення безпеки додатку (доступ до кінцевих точок по ролям і так далі) обрано Spring Security, що дозволяє використовувати найпоширеніші і перевірені практики в забезпеченні функцій авторизації та автентифікації [11].

Для генерації та наповнення динамічною інформацією статичних HTML-сторінок використовують так звані Template Engine, або рушії шаблонів. На зараз існує декілька доступних безкоштовних аналогів, наприклад FreeMarker Template, Thymleaf, Jade4j, вбудований в Java EE рушій JSP та багато інших. У даній роботі буде використовуватись рушій FreeMarker, оскільки всі вище описані рушії відрізняються в основному синтаксисом та наявністю деяких функцій, а арсеналу FreeMarker буде достатньо для потреб проекту.

Також, для спрощення комунікації додатку з БД обрано ORM-фреймворк Spring Data JPA з реалізацією Persistence-provider'a у виді Hibernate [12]. Spring Data JPA – це частина Spring Framework, що дозволяє працювати з сутностями моделі на максимально можливому рівні абстракції, що в свою чергу дає можливість без проблем змінювати реалізацію БД з одної на іншу, проводити базові операції с сутностями в БД без написання SQL-запитів взагалі, тощо. Мінусом цієї технології є зменшення швидкодії додатку в порівнянні з роботою з чистим JDBC або Hibernate, але цей недолік грає роль лише на високонавантажених системах або при недостачі серверних потужностей, а у всіх інших він переважується тим, що використання ORM-фреймворку дуже сильно спрощує розробку елементів, відповідальних за зв'язок додатку з БД, а відповідно і подальшу підтримку додатку

через відсутність коду, в якому потрібно розбиратися та меншої кількості помилок в коді. Hibernate в якості persistence-provider'а обраний через його швидкодію та безкоштовність. Хоча він є одним із найскладніших ORM-фреймворків для Java, його використання доцільне, оскільки при роботі зі Spring Data JPA безпосередньо розробниками Hibernate не використовується, а використовують лише інтерфейси-репозиторії, створені Spring Data JPA.

2.2.Декомпозиція предметної області

Для виконання поставлених функціональних вимог необхідно провести декомпозицію предметної області, тобто розклад функціональних вимог на сутності та операції з ними, що в сумі будуть виконувати поставлені задачі.

Першою сутністю в додатку буде сесія голосування (Voting Session), яка буде відповідальна за початок та кінець конкретного періоду голосування. Вона повинна бути керована адміністратором, з можливістю назначити початок сесії на визначену дату, що буде значити початок голосування, та автоматичного її закінчення після настання визначеної дати та часу. Також, оскільки для кожної сесії теми для голосування (наприклад пунктуальність викладача, якість подачі матеріалу, тощо) будуть змінюватись, необхідно передбачити можливість збереження списків цих тем для студентів та викладачів окремо. Таким чином, сутність Voting Session буде представляти собою об'єкт, що має дату початку, дату кінця та два посилання на списки тем для голосування: для студентів та викладачів.

Statistics Holder – сутність, відповідальна за зберігання інформації про голосування за одного конкретного користувача в межах однієї сесії. Містить посилання на користувача та до сесії голосування, до якої він належить, на список користувачів, що вже проголосували за власника holder'у в межах даної сесії (для запобігання багатократного голосування одного користувача за іншого), роль користувача, до якого належить (для пришвидшеного пошуку в базі даних) та список сутностей Statistics.

Statistics – сутність, що відповідальна за безпосереднє збереження результатів голосування за користувача на конкретну тему. Для кожного користувача таких об'єктів в межах сесії створюється стільки ж, скільки і тем для голосування в даній сесії для ролі користувача. Містить посилання на сутності StatisticsHolder та StatisticsTopic, totalVotes (загальна кількість голосів, на основі яких рахувалось середнє значення), та calculatedRating, що відповідає безпосередньо за рейтинг користувача по конкретній темі. Варто зазначити, що внесення поля totalVotes в цю сутність – свідома денормалізація бази даних, оскільки без цього для кожного перерахунку доводилось би звертатись до бази даних з зайвим запитом перерахувати всі голоси для конкретного користувача в даній сесії, що значно б сповільнило обробку запитів та збільшило навантаження на БД.

StatisticsTopic – відповідає за збереження теми для голосування за користувача (наприклад «Старанність» для студента або «Актуальність матеріалу» для викладача). Має всього два поля – name, що представляє саму тему та role, що позначає, для якої ролі призначена ця тема.

Сутність UserVote відповідальна за збереження інформації про голос користувача в межах однієї сесії. Має 3 поля: holder, що є посиланням на StatisticsHolder, за який було проведене голосування, timeOfVote, що є датою голосування, та строкове поле voterHash. Останнє поле використовується для визначення, чи голосував один користувач за іншого в межах поточної сесії, чи ні. Значення цього поля формується з допомогою алгоритму хешування, що також використовується для збереження паролів в БД для запобігання їх легкому прочитанню.

StudentGroup – відповідає за зв'язок між студентами та викладачами, що викладають у них, аналогічна навчальній групі. Ця сутність необхідна для забезпечення можливості пошуку користувачів для голосування, наприклад студенту для голосування показувати лише викладачів, що у нього викладають, а викладачу- лише студентів, що навчаються в групах, в яких він викладає.

User, Student та Teacher – сутності що представляють собою користувачів додатку. Відповідно до реального життя, сутність студента має поле group, типу StudentGroup, а Викладач – список груп, в яких він викладає. Окірм цього, оскільки в проекті використовується Spring Security, то класи, що відповідальні за автентифікацію користувачів повинні імплементувати інтерфейс UserDetails, для забезпечення наявності всіх необхідних сервісних полів (наприклад, для позначення, чи заблокований користувач, чи не закінчився строк експлуатації акаунту, тощо), тому всі нащадки класу User також мають boolean-поля «accountNonExpired», «accountNonLocked», «credentialsNonExpired» та «enabled».

Окрім цього, всі класи сутностей мають числове поле класу Long id, що є первинним ключем сутності в БД (використовується для однозначної ідентифікації сутності в БД а також для швидкого пошуку потрібної сутності). На це поле в БД ORM-системою автоматично створюється індекс на основі бінарного дерева, що дозволяє знаходити в БД сутності за їх первинним ключем за логарифмічний час O(log(N)).

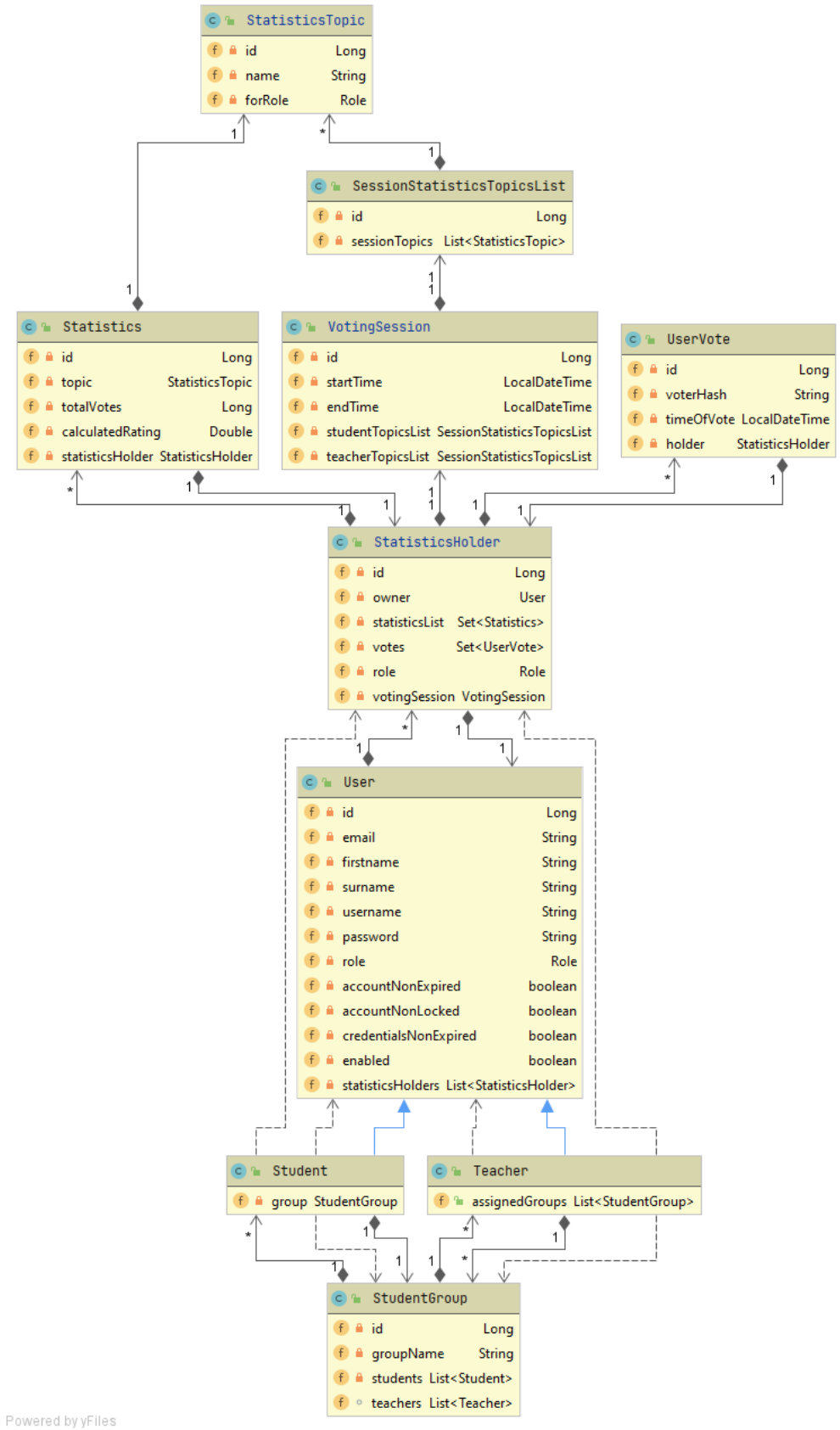

Рисунок 2.1 Клас-діаграма сутностей

2.3.Архітектура програмного забезпечення.

2.3.1. Загальні дані про архітектуру.

Архітектура майже всіх WEB-додатків будується на базі так званої трьохшарової архітектури [13]:

1. Шар відображення (Presentation layer), відповідальний за обробку даних, що прийшли від користувача, перетворення їх в вид, придатний для обробки бізнес-логікою (наприклад перетворення JSON -> Java Object) та формування відповіді користувачу (зворотне перетворення Java Object -> JSON, або формування HTML-сторінок). За визначенням, в цьому шарі не може бути ні бізнес-логіки, ні звернення до БД.
2. Шар бізнес-логіки, або шар сервісів (Service layer). У цьому шарі знаходиться вся бізнес-логіка додатку, що безпосередньо відповідальна за бізнес-процеси (створення та запуск нової сесії голосування, формування діаграми рейтингу користувача, тощо). Цей шар використовує шар доступу до даних та використовується шаром відображення.
3. Шар доступу до даних (Persistence layer). Відповідальний виключно за створення, збереження, оновлення, видалення та діставання об'єктів сутностей з БД. Також, як і шар представлення не повинен містити бізнес-логіки. Використовується шаром бізнес-логіки для маніпуляцій з даними в БД.

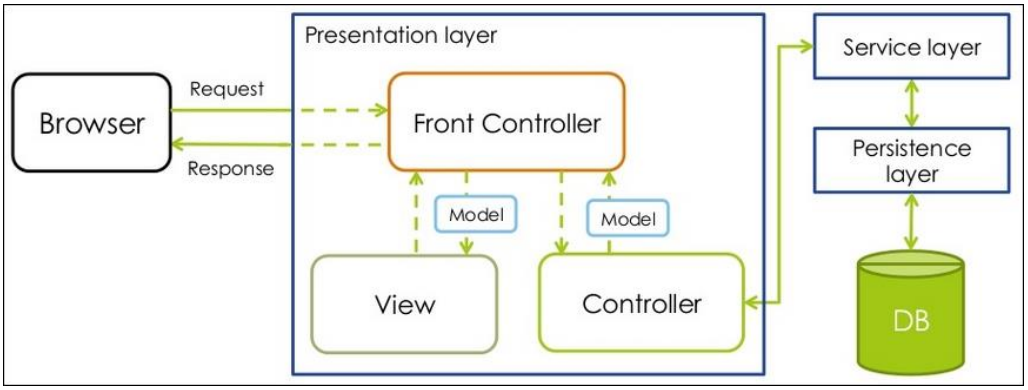

Рисунок 2.2 Загальний вид трьохшарової архітектури WEB-додатку

Presentation layer, в свою чергу, складається з трьох основних елементів:

1. Front Controller – це клас, що є реалізацією однойменного паттерну проектування, який знаходиться «спереду» всього додатку, обробляє всі користувацькі запити та перетворює ці запити та дані, що в них зберігаються у зручні для обробки Java-об'єкти. Після цього ці об'єкти передаються у контролер, який визначається по URL, на який прийшов запит де передаються безпосередньо в бізнес-логіку. У фреймворку Spring MVC таким контролером є DispatcherServlet,.
2. Controller – клас, який має чітко описану кінцеву точку (endpoint) у вигляді URL, при переході по якому запит користувача потрапляє в цей контролер і обробляється ним. Після завершення обробки контролер повинен переадресувати виконання на інший контролер або повернути деяке відображення (View).
3. View – елемент, що відповідає за відображення інформації, що передається користувачеві. Формується він з допомогою Template Engine (наприклад, FreeMarker Template), наповнюється даними, що передаються з контролера та відправляються користувачеві у виді HTML-сторінки.

Також у великих програмах, де один запит користувача може тягнути за собою велику кількість бізнес-логіки (наприклад виклик десятку сервісів), для спрощення використання сервісів використовуються так звані фасади (Facade) [14], що є патерном проектування. Вони призначені для «упакування» великої кількості викликів різних сервісів в один клас, що дозволяє не засмічувати класи контролера та винести цей код в окреме місце, відповідно до принципу SRP [15]. В цьому проекті фасади використовуватись не будуть, оскільки всі запити від користувача достатньо легковісні і не потребують великої кількості викликів сервісів.

У сучасних програмах, в яких читабельність та підтримуваність коду ставиться на перше місце (усі без виключення WEB-додатки в довгоживучих проектах) є надзвичайно важливим дотримуватися принципів SOLID [15], одним

із яких є принцип інверсії залежностей (Dependency Inversion або DI), який каже «мінімум залежностей на конкретику, максимум залежностей на абстракції». Тобто у випадку, коли клас використовує якийсь інший клас, то перший клас повинен використовувати другий не через його безпосередню реалізацію, а через деякий інтерфейс, що надає методи для реалізації. Такий підхід надзвичайно зменшує зв'язаність коду та дозволяє в будь-який момент змінити реалізацію другого класу на якусь іншу, без потреби змінювати код першого класу. Також є інший принцип, що зменшує зв'язаність елементів між собою: принцип інжекції залежностей (Dependency Injection), що каже «Клас не має створювати свої залежності, а повинен отримувати їх звідкись іще». Для реалізації цих принципів і була створена основа Spring Framework – Spring Core, що є так званим IoC-контейнером, тобто відповідальний за створення екземплярів потрібних об'єктів та підключення їх один до одного. Такий підхід дозволяє розробляти елементи ізольовано один від одного, а взаємодіяти вони будуть лише через інтерфейси своїх залежностей.

Оскільки даний проект розробляється на базі Spring Boot, усі класи для отримання своїх залежностей будуть використовувати анотацію @Autowired, що відповідальна за інжекцію одних елементів в інші.

Також такий підхід дозволяє значно спростити написання модульних тестів, оскільки підмінити залежності елементів на заглушки стає дуже простою задачею.

2.3.2. Шар контролерів.

Таблиця 2.1 Специфікація публічних методів контролерів

URL	Вхідні дані	Повертає	Призначення
AdminController			
/admin Method:GET	-	Головна сторінка адміністратора	Відображення основної інформації про стан сесії голосування та доступні теми для голосування.
/admin/new-session Method:POST	- LocalDateTime startTime	Переадресує на /admin.	Створює нову сесію голосування з тривалістю startTime – endTime та з

	- LocalDateTime endTime - List<String> studentTopics - List<String> teacherTopics		темами, отриманими в списках studentTopics та teacherTopics.
/admin/new-topic Method:GET	-	Сторінка вводу даних для нової теми голосування.	Відображення сторінки створення теми.
/admin/new-topic Method:POST	- StatisticsTopicDTO topicDTO	Головна сторінка адміністратора	Створює нову тему для голосування з даними, введеними користувачем.
/admin/new-student Method:GET	-	Сторінка вводу даних для реєстрації нового студента в системі..	Відображення сторінки реєстрації студента.
/admin/new-student Method:POST	- StudentDTO studentDTO	Головна сторінка адміністратора	Реєструє нового студента в системі з даними, введеними адміністратором..
/admin/new-teacher Method:GET	-	Сторінка вводу даних для реєстрації нового викладача в системі..	Відображення сторінки реєстрації викладача.
/admin/new-teacher Method:POST	- TeacherDTO teacherDTO	Головна сторінка адміністратора	Реєструє нового викладача в системі з даними, введеними адміністратором..
/admin/new-group Method:GET	-	Сторінка вводу даних для нової навчальної групи.	Відображення сторінки створення навчальної групи.
/admin/new- group Method:POST	- StudentGroupDTO studentGroupDTO	Головна сторінка адміністратора	Створює навчальну групу для голосування з даними, введеними користувачем.
UserRatingController			

/users-rating/{userName} Method:GET	- String userName	Сторінка з рейтингом обраного користувача.	Відображення рейтингу користувача, логін якого переданий як параметр.
/ users-rating/my-rating/ Method:GET	-	Сторінка з рейтингом поточного користувача.	Відображення сторінки рейтингу поточного користувача.
UsersController			
/users Method:GET	-	Сторінка з доступними для голосування користувачами.	Відображення сторінки з доступними для голосування користувачами. Для студентів повертає список доступних викладачів, для викладачів – сторінку пошуку студентів, заповнену всіма доступними студентами.
/users/criteria Method:POST	- StudentSearchCriteria studentSearchCriteria	Сторінка з результатами пошуку студентів.	Пошук студентів за критерієм та їх подальше відображення.
UserVotingController			
/user-vote-page/{username} Method:GET	- String username	Сторінка з статистикою вказаного користувача.	Відображення сторінки з рейтингом користувача, username якого вказаний в запиті.
/user-vote-page/vote Method:POST	- VotedStatisticsDTO statisticsDTO	Сторінка з доступними для голосування користувачами.	Реєстрація голосу за користувача. Дані про користувача, за якого голосують та результат голосування передаються в об’єкті statisticsDTO.

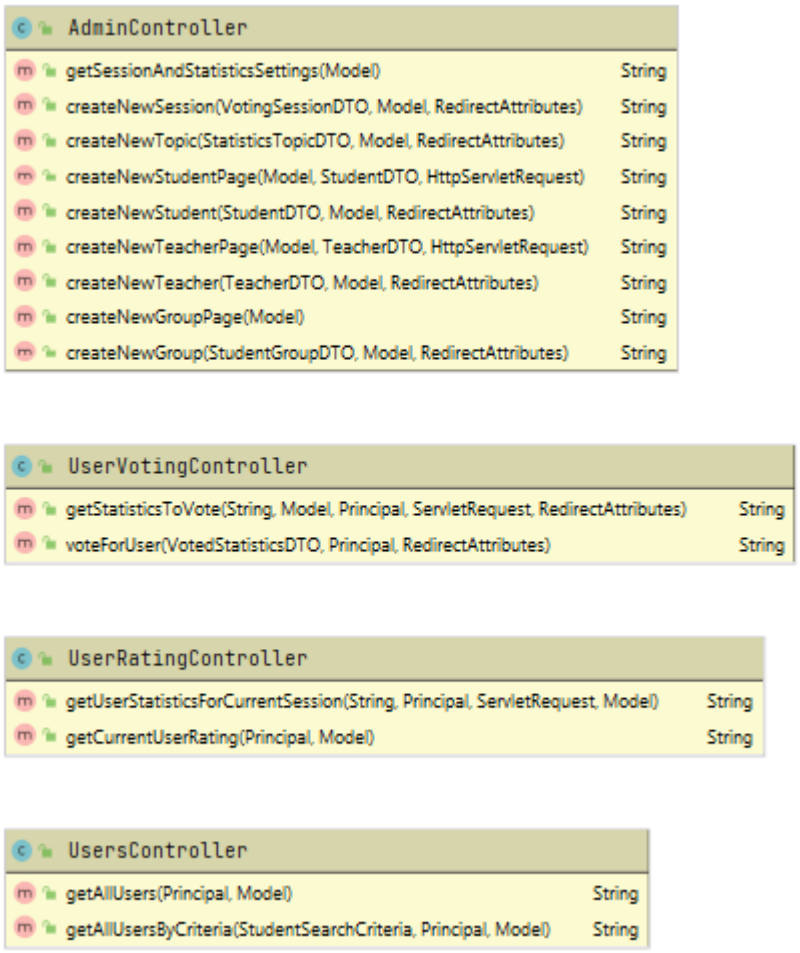

Рисунок 2.3 Класи контролерів та їх методи.

2.3.3. Шар сервісів

Класи сервісів містять у собі методи, що використовуються для виконання визначених функцій у межах домену програмного забезпечення, або якщо простіше – для виконання деяких частин бізнес-логіки. Задля спрощення подальшої підтримки та для покращення читабельності коду, дії, які повинен виконувати додаток для обробки запиту користувача розділяються по їх пов'язаності з бізнес-процесами та сутностями домену додатку, тобто за роботу з сутностями User буде використовуватись UserService, за операції з VotingSession – VotingSessionSerice, StatisticsDiagramService – для створення та управління діаграмами рейтингу користувачів, тощо.

Далі приведені специфікації класів та методів сервісів, що наявні в додатку.

Таблиця 2.1 Специфікація класів та методів шару сервісів.

Назва сервісу	
Назва методу	Опис методу
StatisticsDiagramService	
String getStatisticsImageFileName(String username)	Створює діаграму рейтингу для користувача, на основі даних про його рейтинг з бази даних. Приймає username користувача, для якого створюється діаграма. Повертає назву файлу з діаграмою.
StatisticsHolderProviderService	
StatisticsHolder getStatisticsHolderForUsername(String username)	Повертає об'єкт StatisticsHolder, який приєднаний до наявної сесії голосування та власником якого є користувач з заданим username.
StatisticsHolderService	
void recalculateStatisticsHolder(VotedStatisticsDTO voteDTO, User voter)	Перераховує показники рейтингу користувача, що вказаний в voteDTO, тобто того, за кого голосували, використовуючи дані про оцінки по темам, що також знаходяться в voteDTO. Також використовує параметр voter для перевірки наявності такого ж самого голосу в межах поточної сесії голосування.
StatisticsTopicService	
List<StatisticsTopic> findAvailableStudentStatisticsTopics()	Методи повертають список доступних для використання тем голосування для студента та викладача відповідно.
List<StatisticsTopic> findAvailableTeacherStatisticsTopics()	
void saveNewTopic(StatisticsTopicDTO topicDTO)	Метод зберігає нову тему для голосування, дані про яку знаходяться в аргументі topicDTO.
boolean checkIsTopicExists(StatisticsTopicDTO topicDTO)	Перевіряє, чи існує тема для голосування, та повертає відповідне булеве значення.
StudentGroupService	

void saveNewStudentGroup(StudentGroupDTO dto)	Створює нову навчальну групу.
StudentGroup findGroupByName(String name)	Повертає об'єкт StudentGroup, яка відповідає вказаному name. Якщо такої групи не існує, кидає ModelNotFoundException.
List<StudentGroup> findGroupsForTeacher(String teacherUsername)	Повертає список навчальних груп, до яких приєднаний викладач з заданим userName.
UserService	
User saveUser(UserDTO userDTO)	Зберігає нового користувача.
List<Student> getAllStudentsForCriteria(StudentSearchCriteria criteria, String searcher)	Повертає список студентів, що підпадають під задані в criteria параметри (частина прізвища та навчальна група студента) та знаходяться в групах, до яких приєднаний викладач з userName = searcher. Може повертуни пустий список.
List<User> getAllUsersForSearcher(String searcher)	Повертає список всіх доступних для голосування користувачів для користувача з username = searcher.
Student findStudentByUsername(String username)	Повертає користувача (студента, викладача чи користувача відповідно), з заданим значення поля username.
Teacher findTeacherByUsername(String username)	
User findUserByUsername(String username)	
UserVoteProviderService	
String generateUserVoteHash(User voter)	Повертає строку, яка є хешем унікальної суми полів об'єкту voter, а саме імені, прізвища, імені користувача та адресу електронної пошти.
boolean compareToHashed(String hashed, String notHashed)	Повертає булеве значення, що показує, чи є аргумент hashed хешем строки notHashed.
String getHashedUserData(User voter)	Повертає строку, що є унікальною сумою полів користувача (імені, прізвища, імені користувача та адресу електронної пошти).
VotingRegistrationService	
void registerNewVote(VotedStatisticsDTO votedStatisticsDTO, User voter)	Реєструє новий голос за користувача. Якщо голос для такої пари користувачів вже існує в межах

	поточної сесії – кидає виключення VoteAlreadyExistsException.
boolean checkIsAlreadyVoted(String username, User voter)	Повертає булеве значення, що відображає, чи існує голос від користувача voter за користувача з іменем користувача username в межах поточної сесії.
VotingSessionService	
boolean isCurrentSessionExists()	Повертає булеве значення, що відображає, чи запущена на зараз сесія голосування.
VotingSession getCurrentVotingSession()	Повертає поточну сесії голосування, якщо вона існує. Якщо сесія не існує, кидає виключення ModelNotFoundException
void saveNewVotingSession(VotingSessionDTO votingSessionDTO)	Створює новий екземпляр VotingSession, дані про яку знаходяться в аргументі votingSessionDTO.

Також однією із функцій шару сервісів є пошук студентів для викладачів, оскільки до одного викладача можуть бути приєднані декілька груп студентів, а в кожній групі може знаходитись по декілька десятків студентів. Для цього реалізована група класів, що реалізують єдиний інтерфейс StudentSearchStrategy та являються імплементацією патерну стратегія (Strategy). Інтерфейс має всього один метод, сигнатура якого виглядає так: List<Student> doSearch(StudentSearchCriteria criteria, String searcherUsername). Цей метод реалізований по-різному в кожному із чотирьох класів-реалізацій: SearchAllStudentsStrategy, SearchStudentsByGroup, SearchStudentsBySurnamePart та SearchStudentsByWholeCriteria. Кожен з цих класів повертає різні списки студентів, а вибір, який з класів буде виконувати пошук визначається за наступним правилом:

- Якщо StudentSearchCriteria.groupName та StudentSearchCriteria.surnamePart не вказані (null), то відбувається пошук всіх доступних для викладача студентів (SearchAllStudentsStrategy).

- Якщо StudentSearchCriteria.groupName не вказана, а StudentSearchCriteria.surnamePart вказана, відбувається пошук по частині прізвища студента (SearchStudentsBySurnamePart).
- Якщо StudentSearchCriteria.surnamePart не вказана, а StudentSearchCriteria.groupName вказана, відбувається пошук по навчальній групі студентів (SearchStudentsByGroup).
- Якщо обидва поля StudentSearchCriteria.groupName та StudentSearchCriteria.surnamePart вказані, то відбувається пошук по обом параметрам (SearchStudentsByWholeCriteria).

2.3.4. Шар репозиторіїв

Класи та методи шару репозиторіїв описуватися не будуть, оскільки весь доступ до даних виконувався за допомогою Spring Data JPA [16], і в такому випадку створення репозиторіїв складається лише з вибором назви для репозиторію, визначенням, які методи потрібні та переведенням потреб від методу в спеціальну форму, яку Spring Data JPA може розібрати та перетворити в код, що і буде виконуватись на етапі Runtime.

2.4.Розробка таблиць та відношень бази даних

Оскільки для зв’язку з базою даних використовується фреймворк Spring Data JPA в парі з Hibernate, то немає необхідності явно використовувати мову SQL та DDL-запити до бази даних для створення таблиць, задання первинних ключів, об’єднання таблиць з допомогою зовнішніх ключів, задання обмежень по типу UNIQUE, NOTNULL і т.д., оскільки все це фреймворк робить автоматично, на основі мета-інформації, вказаної за допомогою анотацій в класах об’єктів-сутностей [17]. Такими анотаціями є, наприклад анотація @Entity , що вказує фреймворку, що даний клас є класом сутності БД та повинен бути просканований на наявність додаткових анотацій, які будуть керувати створенням таблиць в БД та встановленням відношень з іншими таблицями. Або наприклад анотація @Id, що

вказує що наступне поле повинно бути використане як первинний ключ в таблиці, в якій будуть знаходитись екземпляри цієї сутності.

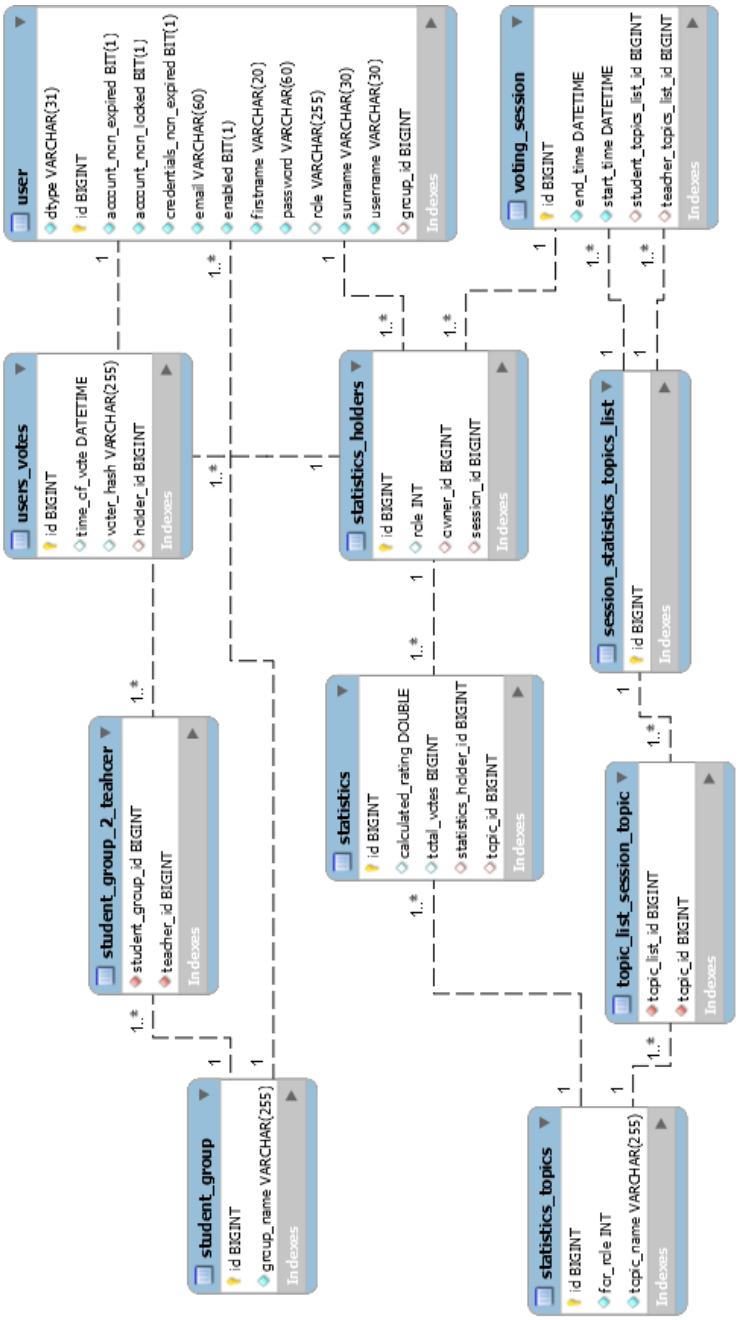

Рисунок 2.4. Таблиці сутностей бази даних та їх відношення.

2.5. Об’єкти передачі даних

Об’єкти передачі даних, також відомі як DTO (Data Transfer Object), Value Object є також одним з патернів проектування [18]. Під час передачі даних від серверної частини до коритувацького інтерфейсу з допомогою класів, що

використовуються в бізнес-логіці, можуть з’явитись небажані залежності між об’єктами домену (класами сутностей) та користувацьким інтерфейсом. Такі залежності ведуть до великої зв’язаності елементів серверної частини та користувацького інтерфейсу, що може привести до неочікуваних результатів в користувацькому інтерфейсі при зміні класів сутностей, чого бути не повинно. Також дуже часто виникає проблема, коли в сутності, що передається на відображення зберігається якась особлива структура даних (можливо, створена спеціально для потреб поточного проекту), для якої не існує обробників у рушії шаблонів, що використовується. І найважливіша та найскладніша проблема виникає, коли до проекту що розробляється потрібно приєднати зовнішню систему (наприклад інший проект, який буде відправляти дані про оновлення персональних даних користувачів системи), і при цьому цей проект може бути написаний навіть з допомогою іншої мови програмування. При цьому зовнішня система не може і не повинна вміти використовувати структури даних нашого проекту, а тим більше знати про їх внутрішню структуру, і навпаки. Для вирішення всіх вище описаних проблем у більшості мов програмування використовуються DTO. Такі об’єкти не мають поведінки, окрім get- та set- методів, використовують лише загальновживані типи даних (строки, цілі числа, числа з плаваючою точкою, boolean, тощо), внаслідок чого стає можливим використання їх для відображення на користувацькому інтерфейсі та їх передачу в інші системи, без розкриття внутрішньої структури нашого проекту, що підвищує інкапсуляцію системи. Такий клас повинен бути створений для кожного класу-сутності, що буде відображатися на користувацькому інтерфейсі, або створюватися на основі даних з нього.

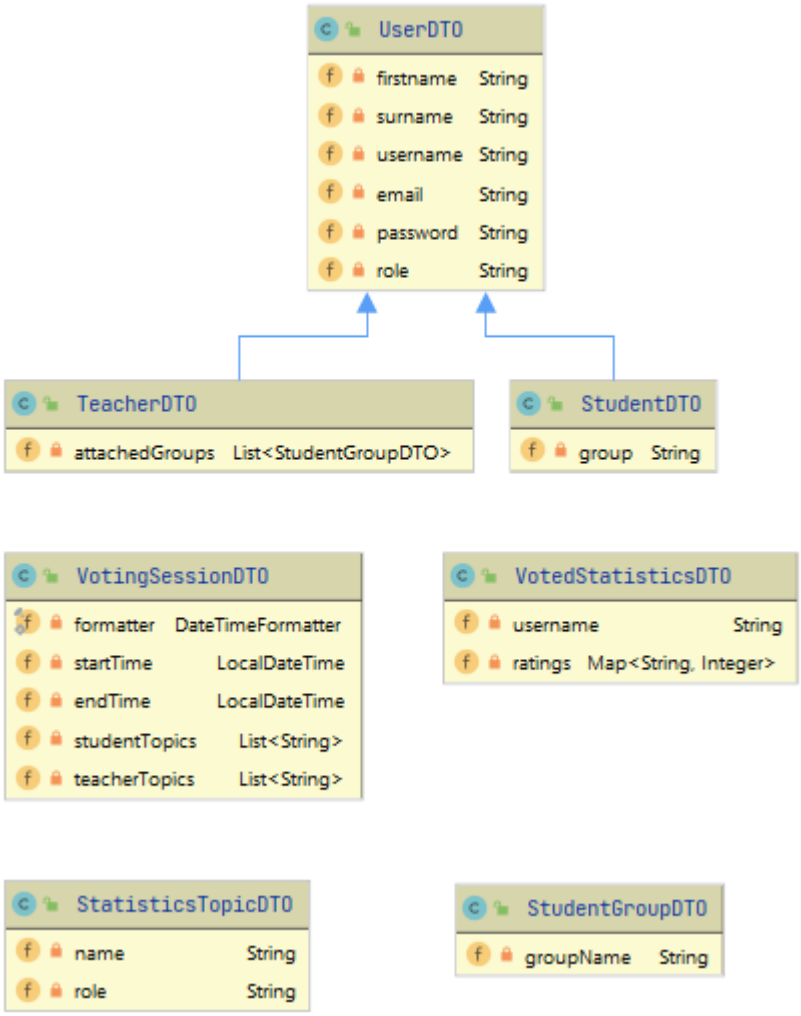

Рисунок 2.5 Ієрархія та поля класів-DTO.

2.6.Забезпечення валідації даних

Для валідації даних форм, що приходять від користувача буде використано пакет spring-boot-starter-validator, що коробки надає додатку підтримку можливості використання валідатору Hibernate Validator, що на даний момент є єдиним валідатором для мови Java, що підтримує стандарт JSR 380 [19]. Spring Framework має дуже зручну методику використання валідаторів даних: для перевірки на валідність форми (у нашому випадку DTO), що приходить в контролер достатньо перед оголошенням аргументу методу контролера поставили анотацію @Valid, що дає MVC фреймворку вказівку, що перед передаванням цього аргументу на опрацювання, його потрібно передати на валідацію валідатору, що

обраний за замовчуванням. Також в аргументах контролера обов’яково повинен бути вказаний об’єкт BindingResult, в який будуть передаватись результати валідації. Вказання параметрів для валідації проводиться з допомогою анотацій з пакету javax.validation, що забезпечує широкий вибір можливостей для валідації вхідних даних, як то перевірка з допомогою регулярного виразу (@Pattern), розмір строки, списку, колекції (@Size), перевірка на присутність об'єкту (@NotNull) та багато інших. Також присутня можливість писати власні валідатори, якщо потрібний функціонал не присутній в стандартній поставці.

Якщо валідація не пройшла успішно, результати відправляються на користувацький інтерфейс і відображають текст, який пов’язаний з обмеженням, що було порушене, як то недостатня довжина паролю чи використання непередбачуваних символів у полі для імені.

2.7.Розробка користувацького інтерфейсу

Для побудови користувацького інтерфейсу (UI) в сучасних великих додатках є дуже багато можливостей. Загалом, абсолютна більшість WEB-проектів насправді складаються з двох – бекенд (backend), тобто частина, яка відповідає за бізнес-логіку і роботу з даними, та фронтенд (frontend), що відповідає за відображення даних, прийом їх від користувача, відправку на бекенд та відображення відповідей. При цьому в багатьох випадках фронтенд-частина не є невеликим придатком до бекенду, а становить собою по суті окремий проект, який може розміщуватись на окремому сервері, потребує для своєї підтримки та розробки команду спеціалістів та може за розмірами та складністю наближатись до бекенд-частини, а то й перевершувати його. Так відбувається, оскільки протягом останніх років для великих компаній характерною рисою стає прикладання все більших зусиль для покращення враження клієнта [8] (customer experience). На ринку комерційної розробки нині все доволі просто – якщо користувачу незручно або не подобається користуватись програмним продуктом – він знайде інший, який йому сподобається.

Коли ж мова іде про розробку некомерційного проекту, як наприклад для освітнього закладу, немає сенсу витрачати ресурси та зусилля на створення складного та багаторівневого UI, оскільки аналогів даного додатку просто не існує, і достатньо зробити користувацький інтерфейс достатньо легким для сприйняття та інтуїтивно зрозумілим. Для такої задачі достатньо буде використання пари HTML + CSS, що достатньо прості в опануванні, мають в відкритому доступі безліч готових реалізацій компонентів та взагалі не навантажують браузер користувача та його інтернет-канал, на відміну від додатків з використанням JavaScript + JQuery.

Рисунок 2.6 Сторінка входу в систему.

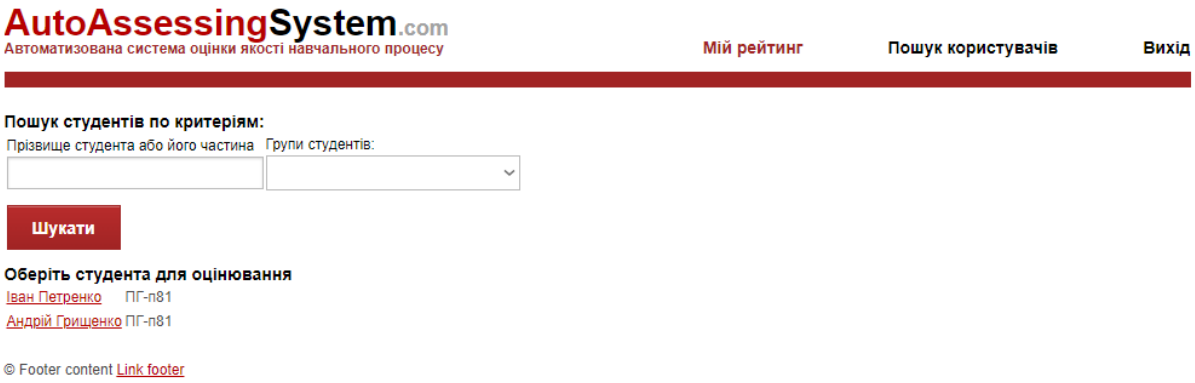

Рисунок 2.7 Сторінка вибору користувача для голосування.

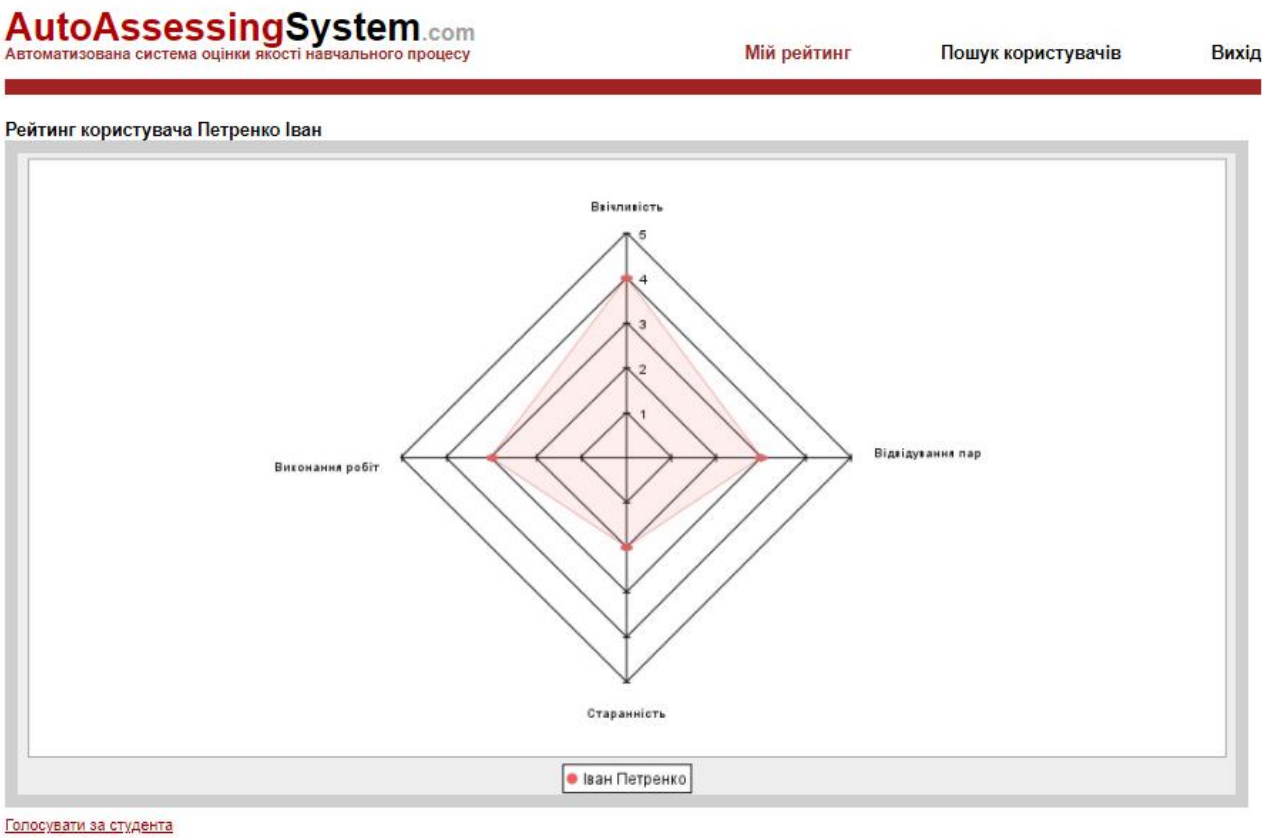

Рисунок 2.8 Сторінка рейтингу користувача.

Рисунок 2.9 Сторінка голосування по критеріям.

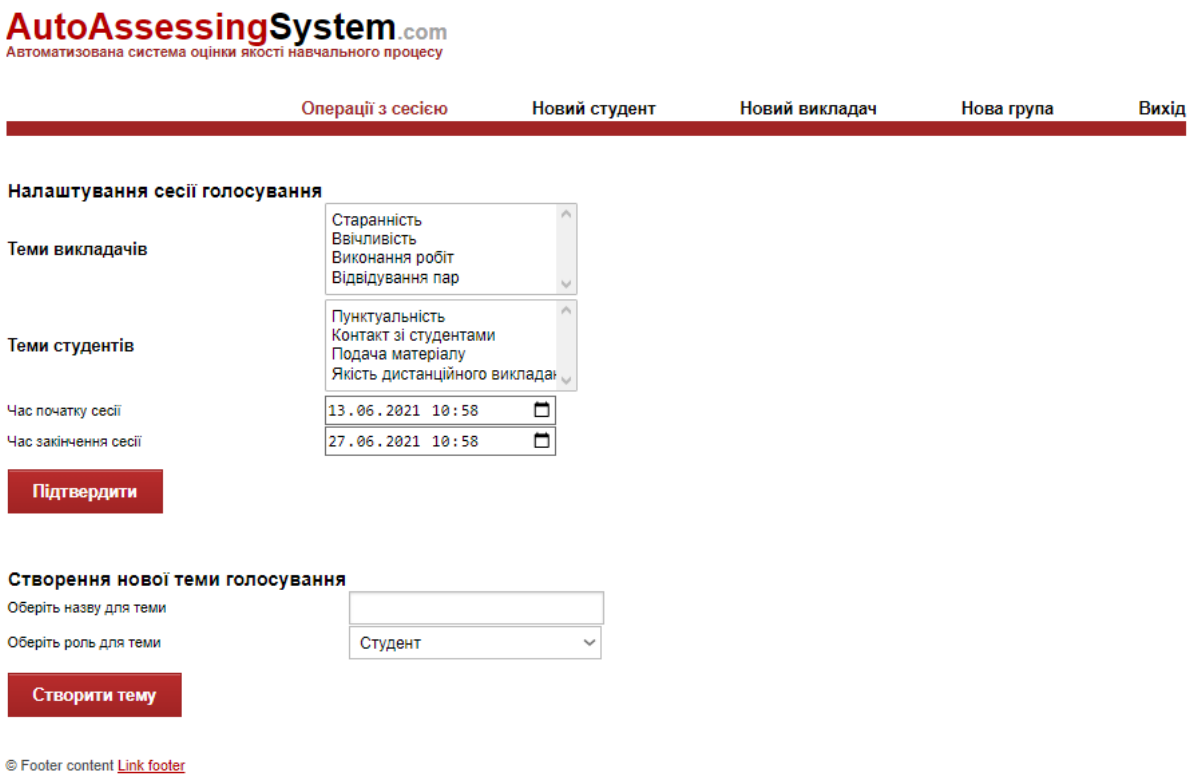

Рисунок 2.10 Сторінка створення сесії та критеріїв для голосування.

Висновки до розділу 2

У даному розділі було обрано необхідні для виконання проекту технології, такі як MVC- та ORM-фреймворк, вибір бази даних для збереження інформації, розглянуті сильні і слабкі сторони аналогів та аргументовано, чому були обрані саме такі технології. Було проведено декомпозицію предметної області додатку, що дозволило перетворити бізнес-сутності на Java-класи, побудувати між ними однозначно визначені відносини та правильно розкласти частини бізнес-логіки по спеціально створеним для цього сервісам, дотримуючись принципу SRP. Описано загальну концепцію трьохшарової архітектури WEB-додатку, її основні частини та сутності, описано деякі патерни проектування, що використовуються в даному проекті. Описано призначення, значення що повертаються та сигнатури методів та класів двох архітектурних шарів, а саме контролерів та сервісів. Приведено основні проблеми, які вирішують DTO в проектах та їх призначення. Описано технологію,що використовується для валідації даних форм. Приведено схему

таблиць та відношень в БД, описано спосіб створення на настройки відображення сутностей бізнес-логіки в БД.

3 ВИПРОБУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1.Опис процесів тестування

Тестування програмного забезпечення переслідує декілька цілей [20]. По-перше, це спосіб перевірити, чи відповідає отриманий додаток технічному завданню. Таке тестування зазвичай проводиться у виді «приймального тестування» (acceptance testing), під час якого на готовому екземплярі програми проводяться випробування по правильності відпрацювання всіх сценаріїв, що повинна забезпечувати система. Також одним із типів тестування є так звані модульні тести (unit-tests), що пишуться для кожного нетривіального публічного методу кожного класу, для перевірки відповідності методів цього класу специфікації. Такий підхід спрощує розробку та підтримку середніх та великих додатків через те, що в додатках такого типу кількість зв'язків між елементами може бути надзвичайно великою, і під час зміни чи доповнення одного елементу може зламатися логіка іншого елемента, взагалі не зв'язаного з основною зміною, що зазвичай викликає появу багів, які дуже важко виявити.

3.1.1. Мета випробувань

Метою тестування в загальному випадку є **Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**[20]:

- підтвердити відповідність розроблених елементів (класів та методів) поставленим до них вимогам (модульне тестування);
- перевірити правильність взаємодії елементів між собою (інтеграційне тестування);
- перевірити правильність відпрацювання всіх основних сценаріїв, передбачених для системи (приймальне тестування);

- перевірити здатність системи зберігати працездатність під час великих навантажень, наприклад, при тисячі запитів в секунду (тестування продуктивності).

Окрім вище зазначеного, тестування проводиться для того, щоб на етапі планування та розробки виявити баги та непередбачувану поведінку елементів, яка може бути обумовлена особливостями мови програмування або особливостями зв’язку між використаними елементами/бібліотеками. Виявлення таких проблем на ранніх етапах розробки програми дозволяє значно (на порядки) зменшити вартість їх вилучення.

3.1.2. Технології для тестування.

Для тестування Java-додатків, незалежно WEB- або десктоп, використовуються дві технології [21]:

JUnit – бібліотека, що має всі необхідні класи та анотації для модульного тестування та перевірки умов (так звані assert'и).

Mockito – фреймворк, що розширює можливості Junit, використовуючи так звані моки або заглушки (mock) та стаби (stab), що дозволяють імітувати поведінку залежностей, необхідних для елементу, що тестується. Ці елементи (моки) можна програмувати як завгодно, вони також не мають внутрішньої логіки, але можуть повертати задані значення при виклику потрібних методів з заданими параметрами. Ці можливості дозволили вивести модульне тестування на якісно новий рівень, оскільки з допомогою Mockito класи стало можливо тестувати в абсолютній ізоляції від внутрішньої реалізації їхніх залежностей.

3.1.3. Випробування серверної частини

Модульне тестування реалізовується з допомогою написання модульних тестів для кожного нетривіального методу (містить у собі бізнес-логіку, або має

декілька сценаріїв роботи), що перевіряють логіку, процес та результат відпрацювання методу [22]. Такі тести, відповідно принципам FIRST (Fast, Independent, Repeatable, Self-Validating, Timely) повинні бути легковісними (виконуватися швидко), незалежними (повинні виконуватись в довільному порядку без впливу на результат), повторюваними (незалежно від кількості повторень виконання тестів, результат не повинен змінюватись), очевидність (тест повинен або успішно виконатись, або провалитись, без проміжних значень) та своєчасними (тести повинні писатись відразу після написання методу або перед його написанням). Виконання цих умов дозволяє писати легковісні та прості для сприймання та підтримки тести. Кількість таких тестів на проект може бути дуже великою (десятки тестів на один метод, за умови що складність логіки насправді цього потребує). У даному проекті такі тести будуть написані майже для всіх методів класів сервісів, оскільки репозиторії створюються фреймворком та не потребують тестування. Окрім основних плюсів у модульних тестів є ще один плюс, який дуже важливий для розробників команди підтримки – дивлячись не модульні тести, написані для класу, розробник може зрозуміти, для чого призначений клас навіть не дивлячись у документацію, що дуже сильно зменшує затрати часу на зрозуміння бізнес-логіки, запрограмованої в класі.

Інтеграційне тестування в даному проекті виконуватись не буде, тому що головна проблема, яку вирішують інтеграційні тести, а саме перевірка результатів сумісної роботи багатьох великих елементів у програмі відсутня, оскільки всі елементи невеликі та не мають побічних ефектів, які могли б вплинути на інші елементи.

Тестування на продуктивність виконуватись не буде також, оскільки система не розробляється для надвеликої кількості користувачів та не буде терпіти великі навантаження, які б могли змусити «вузькі місця» гальмувати систему і, окрім цього, переробка додатку після тестування на навантаження або її початкове планування для високого навантаження – надзвичайно затратний процес, який потребує роботи висококласних спеціалістів, а для даного проекту це не потрібно.

3.1.4. Випробування клієнтської та серверної частин разом

Для таких випробувань зазвичай використовують один з двох підходів, або їх комбінацію [23]:

- мануальне, або ручне тестування (тестувальник з допомогою клавіатури та миші через веб-браузер самостійно перевіряє працездатність всіх елементів та сценаріїв системи). Такий підхід характерний для проектів, що лише почались розвиватись а також для тих, інтерфейси та сценарії яких постійно змінюються;
- автоматизоване тестування – тип тестування, коли тестувальник замість ручної перевірки сценаріїв та компонентів створює на одній з доступних мов програмування (зазвичай на тій самій мові, на якій написаний проект) або з допомогою спеціального фреймворку (наприклад, Selenium) спеціальні скрипти, які самостійно виконують необхідні сценарії та відправляють результат тестувальнику. Такий підхід дуже сильно зменшує затрати часу на тестування інтерфейсу, але потребує значно більше часу та знань у порівнянні з мануальним тестуванням, при тому що після зміни інтерфейсу або бізнес-логіки, що пов’язана з процесом що перевіряється, може з’явитись потреба написання нового скрипту, оскільки старий став неактуальним. Саме через це таким підходом найчастіше користуються великі та довгоживучі проекти, для яких зміна інтерфейсу або значні зміни в бізнес-логіці – дуже рідке явище.

У даному проекті тестування буде проведено мануальним методом, оскільки сценаріїв в системі небагато, а написання автоматичних тестів навіть для такої їх кількості може зайняти дуже значний час, що не є доцільним.

3.2.Результати тестування

3.2.1. Модульні тести

Запуск модульних тестів проекту показав наступні результати:

Всього тестів: 24.

Успішно завершено: 24.

Провалено: 0.

3.2.2. Мануальне тестування.

Таблиця 3.1 Перевірка можливості входу в систему.

Назва сценарію	Вхід користувача в систему
Передумови	Незареєстрований користувач на сторінці логіну.
Кроки для виконання	1. Перейти на сторінку логіну. 2. Ввести коректі дані користувача. Натиснути «Увійти»
Очікуваний результат	Перехід користувача на базову сторінку для його ролі: - для викладача або студента – сторінка перегляду доступних для голосування користувачів. - для адміністратора – сторінка операцій з сесією.

Таблиця 3.2 Перевірка можливості перегляду статистики іншого користувача під час сесії.

Назва сценарію	Перехід на сторінку перегляду статистики іншого користувача під час сесії голосування.
Передумови	Зареєстрований користувач з роллю студент або викладач.

Кроки для виконання	1. Перейти на сторінку списку доступних користувачів. 2. Натиснути на ім’я одного з доступних користувачів.
Очікуваний результат	Перехід на сторінку з графіком рейтингу обраного користувача та з посиланням на сторінку голосування за цього користувача.

Таблиця 3.3 Перевірка можливості перегляду рейтингу іншого користувача без сесії.

Назва сценарію	Перехід на сторінку перегляду статистики іншого користувача в час, коли сесія не активна
Передумови	Зареєстрований користувач з роллю студент або викладач.
Кроки для виконання	1. Перейти на сторінку списку доступних користувачів. 2. Натиснути на ім’я одного з доступних користувачів.
Очікуваний результат	Перехід на сторінку з інформацією про те, що сесія голосування на зараз не активна.

Таблиця 3.4 Перевірка можливості переходу на сторінку голосування за іншого користувача.

Назва сценарію	Перехід на сторінку голосування за іншого користувача.
Передумови	Зареєстрований користувач з роллю студент або викладач на сторінці рейтингу іншого користувача.

Кроки для виконання	Натиснути на кнопку «Голосувати за викладача/студента»
Очікуваний результат	Перехід на сторінку на сторінці з темами для голосування за іншого користувача.

Таблиця 3.5 Перевірка можливості голосування за іншого користувача.

Назва сценарію	Голосування за користувача.
Передумови	Зареєстрований користувач з роллю студент або викладач на сторінці з темами для голосування за іншого користувача.
Кроки для виконання	1. Вибрати оцінку за кожну тему на сторінці. 2. Натиснути кнопку «Проголосувати».
Очікуваний результат	Зарахування голосу за користувача, перерахунок рейтингу користувача, за якого проголосували, перехід на сторінку вибору користувачів для голосування.

Таблиця 3.6 Перевірка можливості пошуку студентів за критеріями для викладача.

Назва сценарію	Пошук студентів за критеріями.
Передумови	Зареєстрований користувач з роллю викладач на сторінці з доступними користувачами.
Кроки для виконання	1. Вибрати навчальну групу для пошуку студента. 2. Ввести частину або повністю прізвище студента. 3. Натиснути кнопку «Шукати».

Очікуваний результат	Оновлення сторінки списку доступних користувачів студентами, які підпадають під умови пошуку.

Таблиця 3.7 Перевірка можливості переходу на сторінку та перегляду власного рейтингу.

Назва сценарію	Перехід на сторінку власного рейтингу.
Передумови	Зареєстрований користувач з роллю студент або викладач на сторінці доступних для голосування користувачів.
Кроки для виконання	1. Натиснути на кнопку «Мій рейтинг» в заголовку сторінки
Очікуваний результат	Перехід на сторінку з діаграмою власного рейтингу.

Таблиця 3.8 Перевірка можливості реєстрації нового студента в системі.

Назва сценарію	Реєстрація нового студента в системі.
Передумови	Зареєстрований користувач з роллю адміністратор.
Кроки для виконання	1. Натиснути на кнопку «Новий студент» в заголовку сторінки. 2. Ввести коректні дані про студента. 3. Обрати навчальну групу студента. 4. Натиснути «Створити студента».
Очікуваний результат	Створення нового студента, перехід на базову сторінку адміністратора, відображення на сторінці напису «Користувач успішно створений».

Таблиця 3.9 Перевірка можливості реєстрації нового викладача в системі.

Назва сценарію	Реєстрація нового викладача в системі.
Передумови	Зареєстрований користувач з роллю адміністратор.
Кроки для виконання	1. Натиснути на кнопку «Новий викладач» в заголовку сторінки. 2. Ввести коректні дані про викладача. 3. Обрати навчальні групи для викладача. 4. Натиснути «Створити викладача».
Очікуваний результат	Створення нового викладача, перехід на базову сторінку адміністратора, відображення на сторінці напису «Користувач успішно створений».

Таблиця 3.10 Перевірка можливості створення нової навчальної групи.

Назва сценарію	Створення нової навчальної групи.
Передумови	Зареєстрований користувач з роллю адміністратор.
Кроки для виконання	1. Натиснути на кнопку «Нова група» в заголовку сторінки. 2. Ввести коректні дані навчальної групи. 3. Натиснути «Створити групу».
Очікуваний результат	Створення нової навчальної групи, перехід на базову сторінку адміністратора, відображення на сторінці напису «Група успішно створена».

Таблиця 3.11 Перевірка можливості запуску нової сесії голосування.

Назва сценарію	Запуск нової сесії голосування.

Передумови	Зареєстрований користувач з роллю адміністратор.
Кроки для виконання	1. Натиснути на кнопку «Операції з сесією» в заголовку сторінки. 2. Обрати теми для голосування для студентів та викладачів. 3. Обрати час початку та закінчення сесії. 4. Натиснути «Створити сесію».
Очікуваний результат	Запуск сесії голосування, перехід на базову сторінку адміністратора, відображення на сторінці напису «Сесія на зараз почата» та відображення часу початку та закінчення сесії.

Після мануальної перевірки всіх описаних вище тест-кейсів, очікуваний результат був досягнутий, непередбачуваної поведінки не виявлено.

3.3. Розгортування додатку

Оскільки даний програмний продукт розроблений на мові Java, то для його використання немає ніяких архітектурних обмежень, тобто він буде запускатись і працювати на будь-якій машині, для якої уже розроблено JVM (MAC, Windows, Linux, тощо).

Також, оскільки проект виконаний на Spring Boot, то додаток можливо запускати і на вбудованому контейнері сервлетів EmbeddedTomcat, що запаковується в .jar-файл разом з усіма залежностями при створенні артефакту додатку (частіше всього використовується для потреб тестування, коли не треба піднімати окремий сервер на ком’ютері), або ж є можливість запакувати проект в

.war-файл та завантажити на будь-який контейнер сервлетів, що підтримує даний формат (Tomcat, Jetty, WebLogic, WebSphere, тощо).

3.3.1. Розгортування додатку на машині.

Розгортання складається з наступних кроків:

1. Завантажити та встановити безкоштовну базу даних MySQL (параметри бази даних такі як порт, ім’я користувача та пароль можна вибирати довільно).Надалі знадобляться ім’я користувача, пароль та порт бази даних.
2. Встановити будь-який дистрибутив Java (open-source, Orcale, тощо) версії 15 або вище. Задати необхідні параметри оточення (envorinment variables) на машині.
3. Завантажити на машину дистрибутив програми.
4. Відкрити файл config/application.properties, в якому знаходяться налаштування бази даних та початкових даних.
5. Задати бажані логін та пароль для користувача адміністратора у властивості «admin.username» та «admin.password» відповідно.
6. Вписати порт, на якому буде працювати додаток (стандартно 8088) у властиість «server.port».
7. Вписати у властивість «spring.datasource.url» на місце < database_port > порт, вказаний під час встановлення БД в пункті 1.
8. Вписати у властивість «spring.datasource.url» < database_name> бажану назву екземпляру бази даних, або назву уже існуючої.
9. Вписати у властивості «spring.datasource.username» та «spring.datasource.password» ім’я користувача та пароль від бази даних, що були вказані під час встановлення БД в пункті 1.
10. Записати у властивість «diagram.image.path» шлях до папки «images», що знаходиться біля .jar-файлу.

11. Запустити файл «startApplication.bat», дочекатись поки додаток запуститься.
12. Перейти у веб-браузері на адресу https://localhost:<порт з пункту 6> та ввести логін та пароль адміністратора.

Висновки до розділу 3

В даному розділі було проведено аналіз якості розробки програмного продукту. Було описано основні підходи до тестування WEB-додатків, їх основні недоліки та переваги для великих і малих проектів. Було зроблено огляд найпоширеніших інструментів для тестування програмного забезпечення для мови програмування Java та описані базові принципи їх роботи. Було проведено тестування яке виявило:

- специфікація нетривіальних методів розроблених класів відповідає задуманій, оскільки всі модульні тести виконані успішно;
- система в загальному відповідає описаним в розділі 1 вимогам, оскільки всі тест-кейси під час мануального тестування були пройдені успішно та система під час їх проходження не показала непередбачуваної або помилкової поведінки.

На основі цих даних можна зробити висновок, що програмний продукт виконує поставлені до нього вимоги та є цілком працездатним.

ПЕРЕЛІК ПОСИЛАНЬ

1. Michael Rachinger Romana Rauter, Christiana Müller, Wolfgang Vorraber, Eva Schirg Digitalization and its influence on business model innovation. Journal of Manufacturing Technology Management. 2018 р. №3.

2. 8 Benefits of Digital Transformation. Virtru. URL: https://www.virtru.com/blog/8-benefits-digital-transformation/ (дата звернення: 07.06.2021).

3. Desktop App vs Web App: Comparative Analysis. Digital Skynet. URL: https://digitalskynet.com/blog/Desktop-App-vs-Web-App-Comparative-Analysis (дата звернення: 07.06.2021).

4. Kuldeep R. Difference between Web and Desktop Applications. ArtOfTesting. URL: https://artoftesting.com/difference-between-web-application-and-desktop-application (дата звернення: 07.06.2021).

5. How Cloud Computing Is Changing Business Dynamics. Advantage Services. URL: https://www.advantageservices.net/blog/58/How-Cloud-Computing-Is-Changing-Business-Dynamics (дата звернення: 07.06.2021).

6. Определение нефункциональных требований. Devprom ALM. URL: https://myalm.ru/news/Определение-нефункциональных-требований (дата звернення: 07.06.2021).

7. Васильева Е.Ю, О. А. Граничина, С. Ю. Трапицын Рейтинг преподавателей, факультетов и кафедр в вузе. Санкт-Петербург : Вид-во РГПУ ім. А.И.Герцена. 2007 р. 146 с.

8. The Value of Customer Experience. URL: https://hbr.org/2014/08/the-value-of-customer-experience-quantified (дата звернення: 07.06.2021)

9. TIOBE Index for June 2021. TIOBE - The Software Quality Company. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 07.06.2021).

10. 8 Advantages of Using MySQL. DevOps.com. URL: https://devops.com/8-advantages-using-mysql/ (дата звернення: 07.06.2021).

11. Spring Security Features. www.javatpoint.com. URL: https://www.javatpoint.com/spring-security-features (дата звернення: 07.06.2021).

12. Hibernate ORM. Hibernate. URL: https://hibernate.org/orm/ (дата звернення: 07.06.2021).

13. Fowler M. Patterns of Enterprise Application Architecture. 7-ме вид. Addison–Wesley. 2011. 431 с.

14. Фасад. Refactoring and Design Patterns. URL: https://refactoring.guru/uk/design-patterns/facade (дата звернення: 07.06.2021).

15. A Solid Guide to SOLID Principles. Baeldung. URL: https://www.baeldung.com/solid-principles (дата звернення: 07.06.2021).

16. Walls C. Spring in Action, Fifth Edition. Manning Publications Co. 2019. 521 с.

17. Bauer C., King G., Gregory G. Java Persistence with Hibernate, Second Edition. Manning Publications Co, 2007. 876 с.

18. Data Transfer Object. Developer tools, technical documentation and coding examples. URL: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN (дата звернення: 03.06.2021).

19. Contributors to Wikimedia projects. Bean Validation. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Bean_Validation (дата звернення: 03.06.2021).

20. Abhijit A. Sawant Pranit H. Bari, P. M. Chawan Software Testing Techniques and Strategies. International Journal of Engineering Research and Applications. - Травень-Червень 2012. - с. 980-986.

21. Tawde S. Testing Frameworks for Java. EDUCBA. URL: https://www.educba.com/testing-frameworks-for-java/ (дата звернення: 07.06.2021).

22. Unit Testing in Software Testing: Definition and Benefits. TestFort Testing & QA Company. URL: https://testfort.com/blog/software-unit-testing-what-is-that-why-is-it-important (дата звернення: 07.06.2021).

23. Manual Testing vs. Automated Testing. Perfecto by Perforce. URL: https://www.perfecto.io/blog/automated-testing-vs-manual-testing-vs-continuous-testing (дата звернення: 07.06.2021).