Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Мазниченка Іллі Вадимовича*
(ПІБ)

академічної групи *122-21ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему:

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц.Мартиненко А.А*			
економічний	*доц.Косьяненко Л.В*			
Рецензент				
Нормоконтролер	*доц. Гулина І.Г*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)
_____ М.О.Алексєєв
(підпис) (прізвище, ініціали)
« » _____ 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-21СК-1* Мазниченко І.В
(група) (прізвище та ініціали)
тема кваліфікаційної роботи Розробка веб-системи обліку роботи викладачів англійської мови
затверджена наказом ректора НТУ «ДП» від *23.04.2024р.* № 375-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки.***	***27.05.2024 р.***

Завдання видав _____ *Мартиненко А.А*
(підпис) (посада, прізвище, ініціали)
Завдання прийняв до виконання _____ *Мазниченко І.В*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *08.06.2024 р*

РЕФЕРАТ

Об'єкт, над яким працюємо – це веб-додаток для контролю вчителів англійської мови.

Мета кваліфікаційної роботи полягає в створенні цього веб-додатка, використовуючи Java та фреймворк Spring для забезпечення необхідного функціонального потенціалу.

У вступі ми проводимо аналіз проблеми та конкретизуємо мету нашої роботи, наводимо обґрунтування актуальності теми та уточнюємо постановку завдання.

В першому розділі ми аналізуємо предметну галузь, визначаємо актуальність завдання, формулюємо постановку завдання та вказуємо вимоги до програмної реалізації та технологій.

У другому розділі ми розглядаємо наявні рішення, обираємо платформу для розробки, розробляємо веб-орієнтовану інформаційну систему, описуємо її роботу, алгоритм та структуру функціонування.

В економічному розділі визначаємо трудомісткість розробленої інформаційної системи, проводимо підрахунок вартості роботи та розраховуємо час на її створення.

Практичне значення нашої роботи полягає у створенні веб-додатка, який дозволить слідкувати за продуктивністю вчителів англійської мови.

Актуальність розробленого веб-додатка обумовлена великим інтересом суспільства до сфери онлайн навчання, яка постійно зростає у популярності. Ключові слова: Веб-додаток, Postgres база даних, Java, Spring, Thymeleaf, HTML, CSS, Bootstrap.

ABSTRACT

The object we are working on is a web application for monitoring the productivity of English language teachers.

The aim of our work is to develop this web application, utilizing Java and the Spring framework to ensure the necessary functional potential. In the introduction, we analyze the problem and specify the purpose of our work, providing justification for the relevance of the topic and clarifying the task statement.

In the first chapter, we analyze the subject area, determine the relevance of the task, formulate the task statement, and specify the requirements for software implementation and technologies.

In the second chapter, we examine existing solutions, select a development platform, develop a web-oriented information system, describe its operation, algorithm, and structure of functioning.

In the economic section, we determine the complexity of the developed information system, calculate the cost of work, and estimate the time required for its creation.

The practical significance of our work lies in creating a web application that allows monitoring the productivity of English language teachers.

The relevance of the developed web application is driven by the society's great interest in the online learning sphere, which is constantly gaining popularity.

Keywords: web application, Postgres database, Java, Spring, Thymeleaf, HTML, CSS, Bootstrap.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – інтерфейс для програмування застосунків;

БД – база даних;

ООП – об’єктно-орієнтоване програмування;

ОС – операційна система;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології;

BE – Back-End;

FE – Front-End;

MVC – ModelViewController;

СКБД – система керування базами даних

ВСТУП

У сучасному світі, де використання інформаційних технологій стає все більш важливим у навчальній сфері, вирішення питань ефективного управління та контролю за роботою викладачів є ключовим завданням для навчальних закладів. Особливо це стосується викладачів англійської мови, які забезпечують процес навчання студентів. Тому розробка веб-системи для обліку роботи цих викладачів стає актуальним проектом, спрямованим на оптимізацію управлінських процесів та підвищення якості викладання.

Така система не лише відслідковуватиме робочий час викладачів, а й надаватиме корисну інформацію щодо їхньої активності та результативності. У своєму функціоналі вона може включати реєстрацію робочого часу, створення звіту про роботу викладача.

Головна мета проекту – створення ефективного інструменту для управління роботою викладачів англійської мови, що сприятиме покращенню якості введення обліку викладацької діяльності та ефективного контролю. Зважаючи на важливість володіння англійською мовою у сучасному світі, цей проект може значно підвищити ефективність викладачів при отриманні звітності про виконання своєї роботи.

Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані з напрямом підготовки «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 122 «Комп’ютерні науки» галузі знань «Інформаційні технології».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Інтернет стає все більш важливою частиною повсякденного життя людей по всьому світу.

Інтернет – це глобальна мережа з мільярдів комп'ютерів та інших електронних пристроїв. За допомогою Інтернету можна отримати доступ майже до будь-якої інформації, спілкуватися з будь-ким у світі та робити багато іншого.

Все це можна зробити, підключивши комп'ютер до Інтернету, що також називається «вийти в мережу». Коли хтось каже, що комп'ютер онлайн, це просто інший спосіб сказати, що він підключений до Інтернету. [1]

Всесвітня павутина – зазвичай скорочено «веб» – це сукупність різних веб-сайтів, до яких можна отримати доступ через Інтернет. Веб-сайт складається з пов'язаних між собою текстів, зображень та інших ресурсів. Веб-сайти можуть нагадувати інші форми засобів масової інформації, такі як газетні статті або телевізійні програми, або ж вони можуть бути інтерактивними, тобто існувати лише на комп'ютерах.

Метою веб-сайту може бути будь-що: платформа новин, реклама, онлайн-бібліотека, форум для обміну зображенням.

Це звісно не могло не торкнутися сферу викладання. Завдяки інтернету стала не обов'язкова фізична присутність в одному приміщені студентів та викладачів. З початком пандемії набрала обертів дистанційна освіта, завдяки якій можна здобувати знання знаходячись вдома, чим добре скористувались приватні школи англійського.

В процесі аналізу інших систем обліку не змогли знайти у вільному доступі приклади інших систем обліку подібного формату.

Це додало гнучкості для планування занять, але з’явилась проблема з тим щоб це все контролювати. Саме на це спрямований даний проект «Система обліку викладацької діяльності».

Розробка обліку викладацької діяльності вимагає стеження за численними аспектами роботи викладачів. Це складна задача, оскільки потрібно постійно оновлювати та контролювати дані про їхню активність та результативність. Така система має включати в себе зберігання інформації про робочий час, проведені заняття та інші аспекти їхньої роботи.

Це було б неможливо без використання БД, так як, вона зберігає у собі усі дані: дані викладача, дані робочого часу, дані про вартість занять кожної групи.

1.2. Призначення розробки та галузь її застосування

Призначенням розробки є створення веб-системи облікової роботи викладачів англійської мови.

Даний продукт має використовуватись внутрішніми працівниками шкіл англійської мови.

Призначення розробки полягає у наданні викладачам та їх керівникам наступних переваг автоматизованої системи:

– безперервна робота розроблюваної мережі;

– дає можливість викладачам зареєструватися та використовувати систему для обліку проведення занять;

– дає можливість їх керівникам відстежувати продуктивність кожного викладача, згідно з вказаним періодом;

– дає можливість керівникам встановити викладачу рівень доступу керівника.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- Освітня програма 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 375-с від 29.04.2024 р;

Завдання на кваліфікаційну роботу на тему «Створення веб-системи облікової роботи викладачів англійської мови».

1.4. Постановка завдання

Завдання є створення веб-системи облікової роботи викладачів англійської мови.

Кінцевий продукт передбачає наступний функціонал:

- сторінки входу та реєстрації аккаунту викладача;
- сторінка звітів викладача про кожен відпрацьований день;
- сторінка для додавання груп, які веде викладач;
- сторінка для розрахунку заробітної плати;
- сторінка для переглядів всіх викладачів;
- можливість доступу до сторінок в залежності від ролей користувача.

Для цього нам потрібно провести:

- аналіз предметної галузі;
- аналіз аналогів подібних систем;
- аналіз інструментів для виконання кваліфікаційної роботи;
- проектування систем і обліку;
- проектування алгоритмів роботи системи;

– провести програмну реалізацію;
– провести тестування;
– провести оформлення документації.

Для розробки було обрано мову програмування JAVA, стек технології: Spring Boot, Spring MVC, Hibernate, Spring Security, Spring Data, Thymeleaf та Flyway.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт має дотримуватися наступних функціональних вимог:

– якщо користувач не зареєстрований, то він не має доступ до веб-сервісу;
– користувач мусить мати змогу зареєструватись на сайті;
– реєстрація неможлива, якщо вже існує користувач з таким самим логіном;
– користувач повинен мати змогу створювати звіт про сьогоднішнє заняття, заносити дані про групи, які веде користувач;
– користувач має змогу редагувати свої зміни;
– керівник має доступ до всіх звітів всіх користувачів;
– програма не повинна бути вимогливою та підтримуватись у всіх сучасних браузерах.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно виконати наступні вимоги:

– перевірка введеної логіну: він повинен бути унікальним для реєстрації нового облікового запису.

– захист від несанкціонованого доступу до функціоналу за допомогою Spring Security;
– обробка виняткових ситуацій;
– виведення повідомлень у разі виникнення помилок;
– реалізувати відновлення даних за допомогою бекапу.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для коректного функціонування кінцевого продукту необхідними технічними умовами є:

– операційна система Windows, Linux, MacOS, Android або IOS;
– наявність будь-якого сучасного браузера, наприклад, Google Chrome, Opera, Microsoft Edge;
– підтримка високошвидкісного Інтернету;
– обсяг оперативної пам’яті (ОЗУ) не менш 2048 МБ;
– програма не повинна бути вимогливою та підтримуватись у всіх сучасних браузерах.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для розробки BE-частини веб-системи була обрана мова програмування Java. Код повинен бути написаний за всіма стандартами та згідно з договором про стиль коду мови Java. Для розробки програмного застосунку було використано середу розробки Intellij IDEA та систему для автоматизації збору проекту на основі опису їх структури у файлі pom.xml. Програма не повинна бути вимогливою та працювати у всіх найпоширеніших браузерах.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом виконання цієї кваліфікаційної роботи має бути програмний додаток, який у активній фазі дає доступ користувачам до сайту. Додаток має приймати запити на контролери та повертати необхідну відповідь на фронт-частину.

Кінцевий продукт має дотримуватися наступних функціональних вимог:

- якщо користувач не зареєстрований, то він не має доступ до веб-сервісу;
- користувач мусить мати змогу зареєструватись на сайті;
- реєстрація неможлива, якщо вже існує користувач з таким самим логіном;
- користувач повинен мати змогу створювати звіт про сьогоднішнє заняття, заносити дані про групи, які веде користувач;
- користувач має змогу редагувати свої зміни;
- керівник має доступ до всіх звітів всіх користувачів;
- програма не повинна бути вимогливою та підтримуватись у всіх сучасних браузерах.

2.2. Опис застосованих математичних методів

У розробленій системі використовуються стандартні математичні методи(складання) для розрахунку суми заробітної плати.

2.3 Опис використаної архітектури та шаблонів проектування

Spring MVC Framework слідує архітектурному шаблону Model-View-Controller (MVC), який працює навколо переднього контролера, тобто сервлету диспетчера. Сервлет диспетчера обробляє та відправляє всі вхідні HTTP-запити

до відповідного контролера. Він використовує @Controller і @RequestMapping як обробники запитів за замовчуванням. Анотація @Controller визначає, що певний клас є контролером. Анотація @RequestMapping відображає веб-запити на методи Spring Controller. (рис. 2.1) [2]

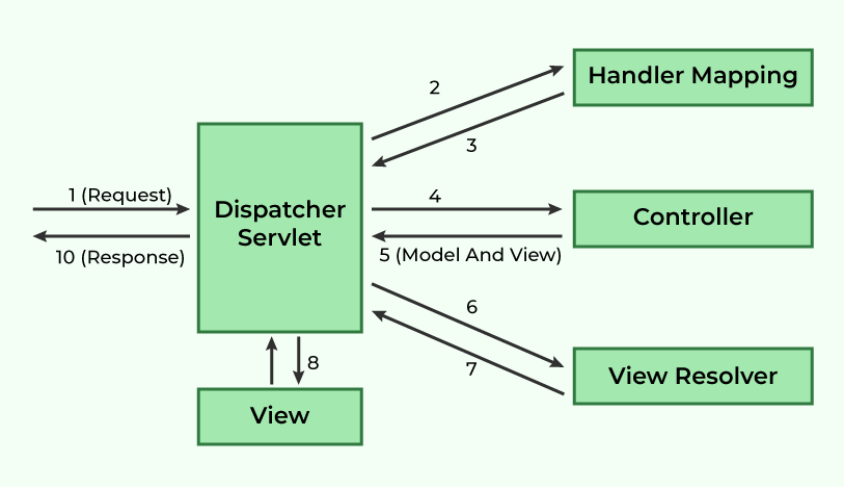

Рис. 2.1. Діаграма потоків пружинної моделі – подання – контролер

Spring MVC Framework працює наступним чином:

1. Всі вхідні запити перехоплюються сервлетом DispatcherServlet, який працює як передній контролер.
2. Потім DispatcherServlet отримує запис відображення обробника з XML-файлу і перенаправляє запит до контролера.
3. Контролер повертає об'єкт ModelAndView.
4. Сервлет DispatcherServlet перевіряє запис розв'язувача представлення в XML-файлі та викликає відповідний компонент представлення.

Для виконання мети проекту використовувались фреймворки Spring такі як: Security, Web, Data, PostgreSql, Lombok, Flyway, Thymeleaf, Bootstrap.

Завдяки фреймворку Spring розробник може використовувати вже існуючий функціонал, таким чином – економиться час та сили на розробку більш унікальних функцій.

Flyway та PostgreSQL – це інструменти для роботи з базами даних. Наприклад, Flyway дозволяє використовувати скрипти для створення таблиць та здійснювати різні операції з ними. Використання СУБД PostgreSQL обумовлено тим, що саме ця база даних була використана в проекті.

Через необхідність демонстрації роботи застосунку було вирішено використовувати Thymeleaf - шаблонізатор Java XML/XHTML/HTML5. Він може працювати як у веб-середовищі, так і поза ним. Thymeleaf є більш підходящим для обслуговування XHTML/HTML5 на рівні представлення веб-додатків на основі MVC (Model-View-Controller). Завдяки ньому був написаний мінімальний FE, який відображає роботу BE частину системи обліку.

2.4 Опис використаних технологій та мов програмування

Дана інформаційна система була розроблена з використанням таких технологій та мов програмування:

- Java;
- Maven;
- Thymeleaf;
- Spring Boot;
- Lombok;
- Flyway;
- Hibernate;
- Bootstrap;
- PostgreSQL;
- HTML.

Java – це об'єктно-орієнтована мова програмування на основі класів, яка розроблена так, щоб мати якомога менше залежностей при реалізації. Вона призначена для того, щоб розробники додатків могли писати один раз і запускати будь-де (WORA), тобто скомпільований код на Java може працювати на всіх

платформах, які підтримують Java, без необхідності перекомпіляції. Вперше Java була випущена в 1995 році і широко використовується для розробки додатків для настільних комп'ютерів, Інтернету та мобільних пристроїв. Java відома своєю простотою, надійністю та функціями безпеки, що робить її популярним вибором для додатків корпоративного рівня.

Java була розроблена Джеймсом Гослінгом в Sun Microsystems Inc в 1995 році, а пізніше придбана корпорацією Oracle. Це проста мова програмування. Java полегшує написання, компіляцію та налагодження програм. Вона допомагає створювати багаторазовий код і модульні програми. Java - це об'єктно-орієнтована мова програмування, що базується на класах і має якомога менше залежностей при реалізації. Мова програмування загального призначення, створена для того, щоб розробники могли писати один раз і запускати будь-де, скомпільований код Java може працювати на всіх платформах, які підтримують Java. Програми на Java компілюються у байт-код, який можна запускати на будь-якій віртуальній машині Java. Синтаксис Java подібний до c/c++. [3]

Переваги:

- Незалежність від платформи:

  Програми на Java можуть працювати на будь-якому пристрої з JVM (віртуальною машиною Java), що робить її дуже універсальною мовою для використання програмістами. Це забезпечує більшу гнучкість і доступність додатків, розроблених на Java. [4]

- Об'єктно-орієнтована мова:

  Java дотримується парадигми ООП (об'єктно-орієнтованого програмування), що полегшує створення модульних програм та коду для багаторазового використання. Це сприяє розробці простіших, легших в обслуговуванні кодових баз для масштабних проєктів.

Недоліки:

- Продуктивність:

Хоча Java добре працює в багатьох ситуаціях, вона може бути повільнішою за деякі інші мови через використання віртуальної машини. Це є недоліком у випадках, коли висока продуктивність має вирішальне значення, наприклад, у системах реального часу або ресурсомістких додатках.

– Споживання пам'яті:

  Програми на Java часто критикують за те, що вони займають багато пам'яті, що призводить до підвищення вимог до апаратного забезпечення та збільшення витрат на хостинг. Цей недолік може викликати занепокоєння в розробників та компаній, які прагнуть оптимізувати використання ресурсів та мінімізувати операційні витрати.

– Синтаксис:

  Java відома своїм багатослівним синтаксисом, що призводить до довшого та складнішого коду, порівняно з деякими іншими мовами. Це може спричинити збільшення часу на розробку та збільшення витрат на обслуговування, особливо у великих проєктах.

Maven – це інструмент для управління та розуміння проектів, який надає розробникам повний фреймворк життєвого циклу збірки. Команда розробників може автоматизувати інфраструктуру збірки проекту майже в найкоротші терміни, оскільки Maven використовує стандартну структуру каталогів і життєвий цикл збірки за замовчуванням. [5]

Maven спрощує та стандартизує процес збірки проекту. Він легко справляється з компіляцією, розповсюдженням, документуванням, командною співпрацею та іншими завданнями. Maven збільшує можливість повторного використання і бере на себе більшість завдань, пов'язаних зі збіркою.

Структура проекту:

– src/main/java – директорія Java класів;
– src/main/resources – директорія конфігураційних файлів та templateів;
– src/test/java – директорія тестів.

Thymeleaf - це сучасний серверний движок шаблонів Java, який робить акцент на природні HTML-шаблони, які можна попередньо переглянути в браузері подвійним кліком, що дуже корисно для самостійної роботи над шаблонами інтерфейсу користувача (наприклад, дизайнером) без необхідності використання працюючого сервера. Якщо ви хочете замінити JSP, Thymeleaf пропонує один з найширших наборів функцій для полегшення такого переходу. Thymeleaf активно розвивається і підтримується. Для більш повного ознайомлення дивіться домашню сторінку проекту Thymeleaf. [6]

Інтеграцією Thymeleaf з Spring MVC керує проект Thymeleaf. Конфігурація включає кілька оголошень бобів, таких як ServletContextTemplateResolver, SpringTemplateEngine і ThymeleafViewResolver. Дивіться Thymeleaf+Spring для більш детальної інформації.

Java Spring Boot – це інструмент з відкритим вихідним кодом, який полегшує використання фреймворків на основі Java для створення мікросервісів та веб-додатків. Яким би не було визначення Spring Boot, розмова має починатися з Java – однієї з найпопулярніших і найпоширеніших мов програмування та обчислювальних платформ для розробки додатків. Розробники в усьому світі починають свій шлях кодування з вивчення Java. Гнучка та зручна у використанні, Java є улюбленою мовою розробників для різноманітних додатків від соціальних мереж, веб-додатків та ігор до мережевих та корпоративних додатків. [6]

Java – дуже популярна мова, але вона має декілька недоліків. Одним з найпопулярніших недоліків є те, що нам все ще потрібно писати шаблонний код, такий як гетери, сетери та метод toString() на Java, в той час як Kotlin та Scala, які також базуються на JVM, не потребують цього, і, отже, це є причиною їхньої більшої популярності у спільноті. Ось тут і з'являється Lombok, який долає цей недолік Java.

Project Lombok – це інструмент бібліотеки Java, який використовується для мінімізації/видалення шаблонного коду та економії дорогоцінного часу

розробників під час розробки за рахунок використання лише деяких анотацій. На додаток до цього, він також підвищує читабельність вихідного коду та економить місце. Але ви можете подумати, що в наш час всі використовують IDE, які надають можливість генерувати ці шаблони коду, тоді навіщо потрібен Lombok. Кожного разу, коли ми використовуємо IDE для генерації цих шаблонних кодів, ми просто рятуємо себе від написання всього цього коду, але насправді він присутній у нашому вихідному коді і збільшує LOC (рядки коду), а також зменшує зручність супроводу і читабельність. З іншого боку, Lombok додає всі ці шаблонні коди під час компіляції у файл «.class», а не в наш вихідний код. [7]

Flyway – це проект з відкритим вихідним кодом, розроблений на Java, який автоматизує міграцію схем баз даних для проектів на Java. Зазвичай скрипти міграції схем розміщуються у тому ж сховищі, що і код. Таким чином, проект досягає прямого керування версіями скриптів одночасно з керуванням версіями програми. [8]

Hibernate – це фреймворк, який забезпечує певний рівень абстракції, що означає, що програмісту не потрібно турбуватися про реалізацію, Hibernate робить все за вас, наприклад, встановлює з'єднання з базою даних, пише запити для виконання CRUD-операцій тощо.

Це java фреймворк, який використовується для розробки логіки персистентності. Логіка персистентності означає зберігання та обробку даних для тривалого використання. Точніше кажучи, Hibernate - це відкритий, неінвазивний, легкий java ORM (Object-relational mapping) фреймворк для розробки об'єктів, які не залежать від програмного забезпечення бази даних і створюють незалежну логіку персистентності у всіх JAVA, JEE. [9]

Фреймворк означає, що це спеціальне програмне забезпечення, що встановлюється, яке забезпечує рівень абстракції на одній або декількох технологіях, таких як JDBC, Servlet і т.д., щоб спростити або зменшити складність процесу розробки.

Bootstrap – це безкоштовний фронтенд-фреймворк для швидшої та простішої веб-розробки

Bootstrap включає шаблони дизайну на основі HTML та CSS для типографіки, форм, кнопок, таблиць, навігації, модалей, каруселей зображень та багато іншого, а також додаткові плагіни JavaScript [10]

Bootstrap також дає можливість легко створювати адаптивні дизайни.

PostgreSQL – це безкоштовна система баз даних з відкритим вихідним кодом, яка підтримує як реляційні (SQL), так і нереляційні (JSON) запити.

PostgreSQL – це внутрішня база даних для динамічних веб-сайтів і веб-додатків.

HTML розшифровується як Hypertext Markup Language (мова розмітки гіпертексту) і є широко використовуваною мовою програмування, що використовується для розробки веб-сторінок. У цьому підручнику з HTML ми дізнаємося, що таке HTML, які його функції, основні теги та елементи, що використовуються, і багато іншого. [11]

2.5. Опис структури програми та алгоритмів її функціонування.

Дана структура проекту є деревовидною та є стандартної структурою середовища розробки IntelliJ IDEA. Приклад приведено нижче на рисунку 2.2.

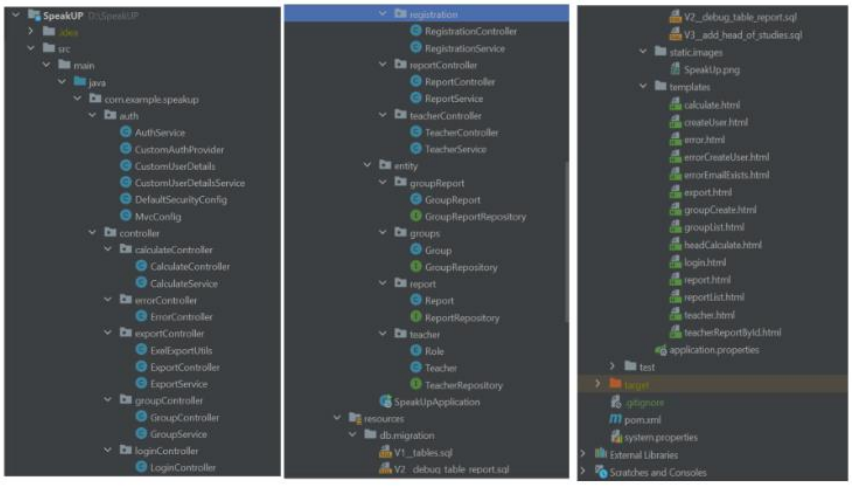

Рис. 2.2. Структура проекту

Дана програма побудована на основі фіча-модульної архітектури, що означає, що класи розподілені по папкам, згідно з їхнім функціоналом. Таким чином, у папці auth – логіка аутентифікації та авторизації, у папці controller є пакети з контролерами та сервісами, що використовувались, у папці entity – сутності(таблиці БД) та репозиторії. Клас, що запускає програму знаходиться у корені. Далі, у папці resources знаходяться усі ресурси, що необхідні ПЗ, такі як html сторінки, скрипти для БД.

Усі репозиторії успадковують інтерфейс JpaRepository, який включає в себе найнеобхідніші методи для роботи з базою даних. Приклад схеми таблиць та їх зв’язків у БД приведено нижче на рисунку 2.3.

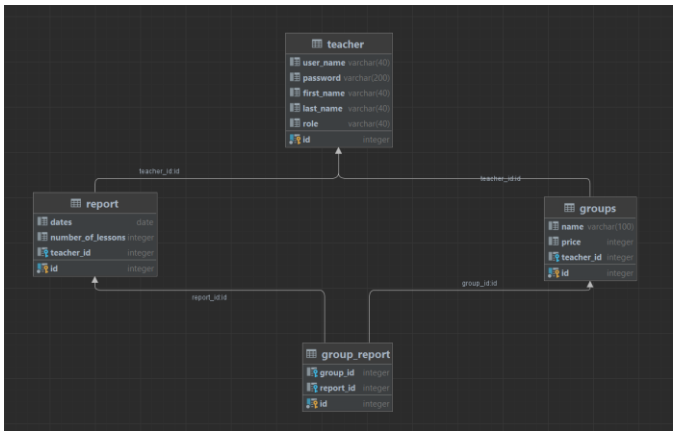

Рис. 2.3. Схеми таблиць та їх зв’язків у БД

Таблиця 2.1

Структура таблиці teacher

Назва колонки	Тип	Призначення
id	Integer	Унікальний ідентифікатор користувача
user_name	Varchar	Ім’я користувача
password	Varchar	Пароль користувача у зашифрованому вигляді

Закінч. табл. 2.1

first_name	Varchar	Ім’я користувача
last_name	Varchar	Прізвище користувача
role	Varchar	Роль користувача

Таблиця 2.2

Структура таблиці report

Назва колонки	Тип	Призначення
id	Integer	Унікальний ідентифікатор звіту
dates	Date	Дата звіту
number_of_lessons	Integer	Номер уроку
teacher_id	Integer	Унікальний ідентифікатор користувача. Зовнішній ключ

Таблиця 2.3

Структура таблиці groups

Назва колонки	Тип	Призначення
id	Integer	Унікальний ідентифікатор групи
name	Varchar	Ім’я групи
price	Integer	Ціна
teacher_id	Integer	Унікальний ідентифікатор користувача. Зовнішній ключ

Таблиця 2.4

Структура таблиці group_report

Назва колонки	Тип	Призначення
id	Integer	Унікальний ідентифікатор звіту
group_id	Integer	Ідентифікатор групи. Зовнішній ключ
report_id	Integer	Ідентифікатор звіту. Зовнішній ключ

Ця папка відповідає за роботу з базою даних. Вона містить файли з назвою changelog, в яких можна створювати таблиці, наповнювати їх даними, а також створювати різноманітні об'єкти бази даних, такі як індекси, тригери тощо. Формати запису можуть бути .xml або .sql. У моєму випадку, тут знаходяться скрипти для створення таблиць та sequences, додавання адміністратора з паролем "admin". Приклад файлів міграції приведено нижче на рисунку 2.4.

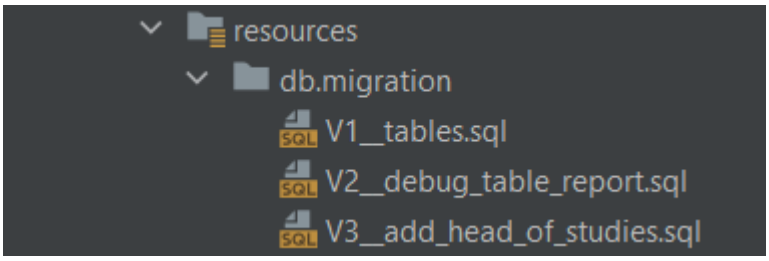

Рис. 2.4. Папка Migration

На разі час для контролерів, що безпосередньо оброблюють запит з фронт-частини й звертаються до БД через сервіси, а потім передають відповідь BE клієнтові.

CalculateController – контролер для обробки сумування кількості годин та вартості уроку за певний період часу. Основні api-методи:

- GET:
  - “/calculate” – повертає сторінку calculate;
  - “/calculate/teacher” – повертає сторінку headCalculate.

ErrorController – контролер для обробки помилок. Основні api-методи:

- GET:
  - “/error” – повертає сторінку error;
  - “/error/createUser” – повертає сторінку errorCreateUser;
  - “/error/emailExist” – повертає сторінку errorEmailExists.

GroupController – контролер для обробки створення або видалення груп. Основні api-методи:

- GET:
  - “/group” – повертає сторінку groupList;
  - “/group/create” – повертає сторінку groupCreate.
- POST:
  - “/group/create” – створює нову групу;
  - “/group/delete” – видаляє групу;
  - “/group/update” – оновлює групу.

LoginController – контролер для обробки авторизації. Основний api-метод:

– GET:
    - “/” – відповідає за авторизацію та аутентифікацію.

RegistrationController – контролер для обробки реєстрації. Основні api-методи:

– GET:
    - “/createUser” – повертає сторінку createUser;
– POST:
    - “/createUser” – оброблює реєстрацію користувача.

ReportController – контролер для обробки звітів. Основні api-методи:

– GET:
    - “/report” – повертає сторінку reportList;
    - “/report/date/*” – повертає сторінку report;
– POST:
    - “/report/create” – оброблює створення звіту;
    - “/report/delete” – оброблює видалення звіту;
    - “/report/create/group” – оброблює створення групи;
    - “/report/delete/group” – оброблює видалення групи.

TeacherController – контролер для обробки керуванням вчителями. Основні api-методи:

– GET:
    - “/teacher” – повертає список всіх вчителів;
    - “/teacher/date/*” – повертає дані про конкретного вчителя;
– POST:
    - “/teacher/delete” – оброблює видалення вчителя;
    - “/teacher/update” – оброблює оновлення вчителя.

2.6. Обґрунтування та організація вхідних та вихідних даних програми.

Системі надаються вхідні дані у форматі DTO-об’єкту, який формується на фронт-частині застосунку в залежності від дій користувача. Крім того, в залежності від необхідної інформації, фронт-частина відправляє відповідні запити до розроблюваної системи. Система обробляє ці запити та відповідає вихідними даними у форматі DTO-об’єкту.

2.7. Опис розробленого програмного продукту.

2.7.1. Використані технічні засоби.

В процесі тестування та розробки були використані наступні технічні засоби:

- оперативна пам’ять обсягом 8 ГБ;
- процесор AMD Ryzen 5 4500U (2.38 ГГц);
- накопичувач на жорсткому диску обсягом 1 ТБ;
- відеоадаптер Vega 7;
- клавіатура та мишка.

Вказані технічні засоби не мають бути обов’язково ідентичних характеристик для безперешкодної роботи системи.

2.7.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

- IntelliJ IDEA;
- Git, GitHub;
- PostgreSQL;
- Java;

– Maven;
– Thymeleaf;
– Spring Boot;
– Lombok;
– Flyway;
– Hibernate;
– Bootstrap;
– HTML.

IntelliJ IDEA – інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблене компанією JetBrains. Community версія середовища підтримує інструменти для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби складання Maven, Ant, Gradle, мови програмування Java, Scala, Clojure, Groovy, Kotlin і Dart. Підтримується розробка застосунків для мобільних платформ Android. До складу входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, редактор регулярних виразів, XML-редактор, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Ultimate Edition – комерційна версія, що відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML-діаграм, підрахунок покриття коду, можливістю роботи з фреймворками (Rails, Grails, Google Web Toolkit, Spring, Play Framework і Hibernate), засобами інтеграції з Perforce, Microsoft Team Foundation Server і Rational ClearCase.

Нижче приведено інтерфейс середовища розробки на рисунку 2.5.

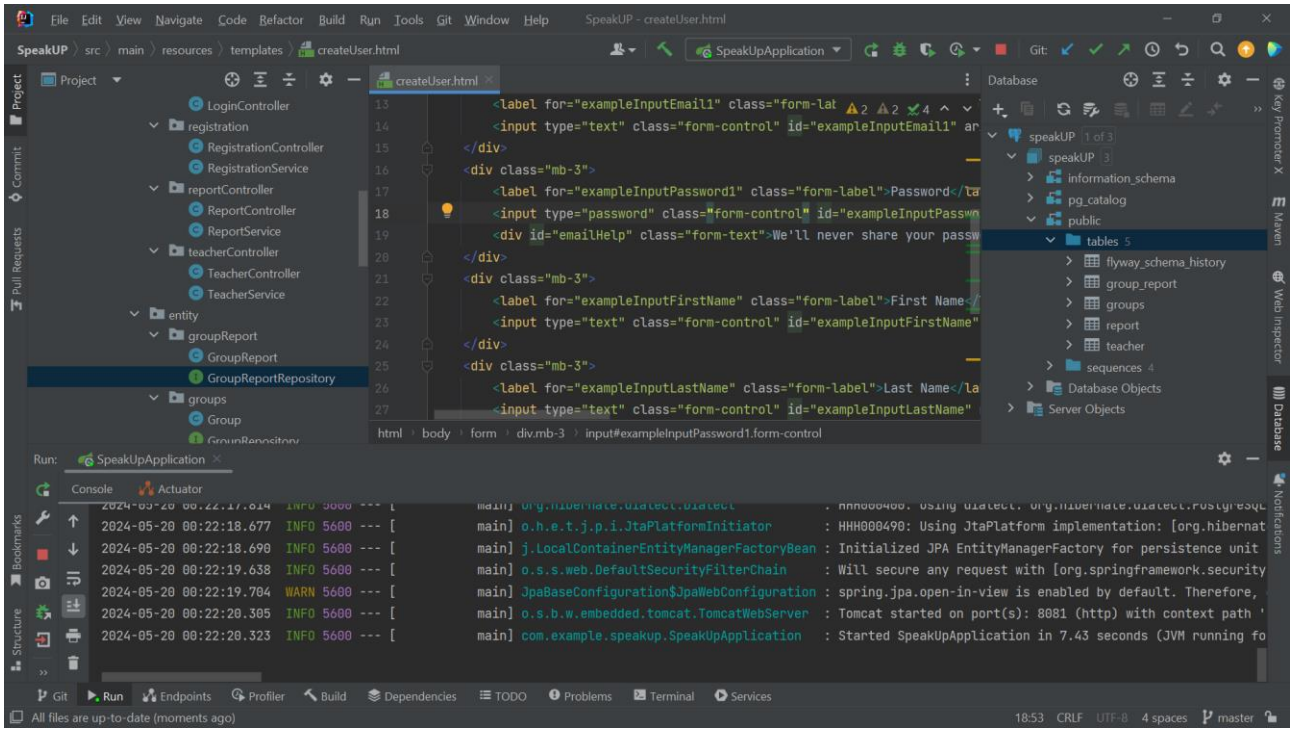

Рис. 2.5. Інтерфейс IntelliJ IDEA

Git – це розподілена система контролю версій, яка дозволяє командам розробників програмного забезпечення мати кілька локальних копій кодової бази проекту, незалежних одна від одної. Ці копії, або гілки, можна швидко створювати, об'єднувати та видаляти, що дає командам можливість експериментувати з невеликими обчислювальними витратами перед об'єднанням в основну гілку (іноді її називають майстер-гілкою). Git відомий своєю швидкістю, сумісністю з робочими процесами та відкритим вихідним кодом. [12]

GitHub – це платформа для розміщення коду для контролю версій та співпраці. Він дозволяє спільно працювати над проектами з будь-якого місця та зберігати версії проекту на віддаленому сервері.

PostgreSQL – це просунута реляційна СУБД корпоративного класу з відкритим вихідним кодом, яка підтримує як SQL (реляційні), так і JSON (нереляційні) запити. Це високостабільна система управління базами даних, яка підтримується більш ніж 20-річним розвитком спільноти. Цей ретельний і спільний підхід сприяв високому рівню її стійкості, цілісності та коректності.

PostgreSQL використовується як основне сховище даних для багатьох веб-, мобільних, геопросторових та аналітичних додатків. [13].

Нижче приведено інтерфейс PostgreSQL на рисунку 2.6.

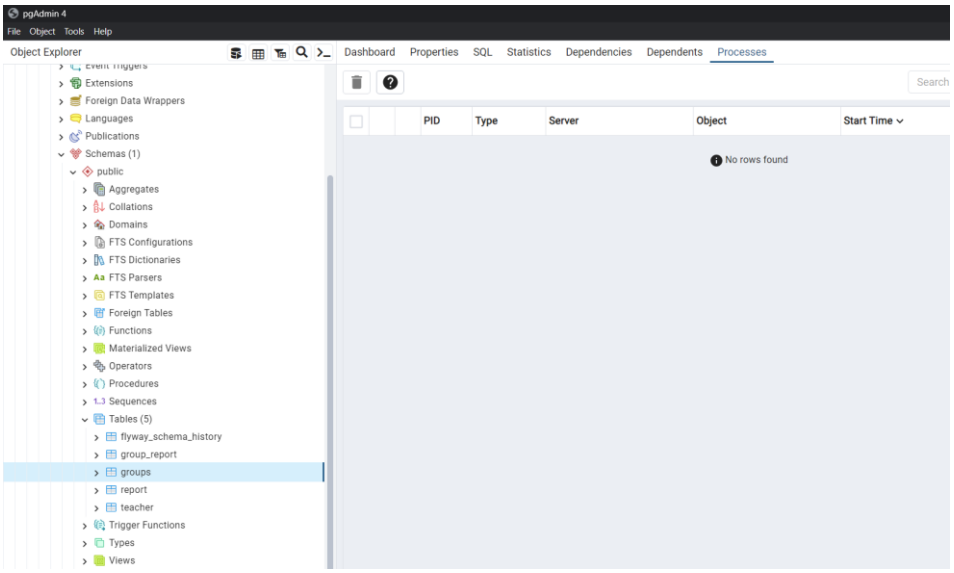

Рис. 2.6. Інтерфейс PostgreSQL

2.7.3. Виклик та завантаження програми

Цей продукт не буде розташований на сервері, тому його можна використовувати лише локально при запуску програми.

Для роботи з розробленим веб-додатком локально потрібно запустити клас SpeakUpApplication.java та перейти у браузері за адресою localhost:8081, на разі є тільки такий спосіб, далі це можливо реалізувати на віртуальному сервері, для цього буде потрібно зібрати проект завдяки команді mvn clean install та задеплоїти його на сервер, а також додати скрипт, що буде постійно тримати зібраний проект у запущеному стані. Авжеж, у такому випадку потрібно буде використовувати ip-адресу замість localhost, а також новий порт.

2.7.4. Опис інтерфейсу користувача

Цей застосунок є інтуїтивно зрозумілим та підтримується на усіх найпоширеніших браузерах.

Сторінка входу забезпечує обробку даних користувача та не дозволяє йому увійти, якщо введенні дані некоректні. Це зображено на рисунку 2.7.

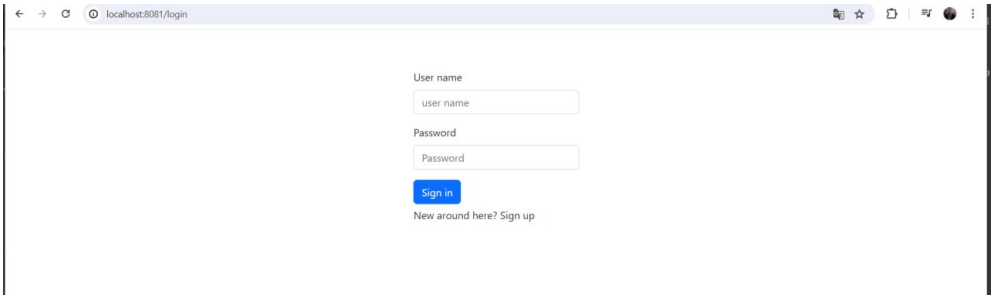

Рис. 2.7. Сторінка входу

Сторінка обробки демонструє логіку неправильно введених даних користувача. Це зображено на рисунку 2.8.

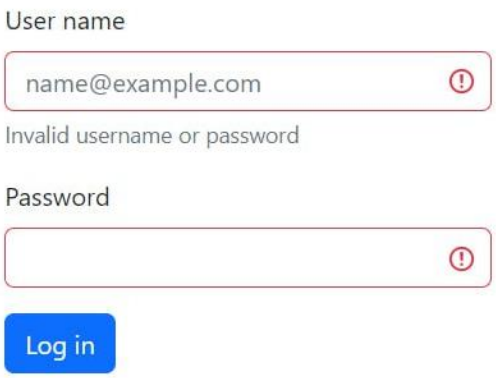

Рис. 2.8. Обробка виключених ситуацій

Ця сторінка забезпечую обробку даних при створенні нового користувача. Це зображено на рисунку 2.9.

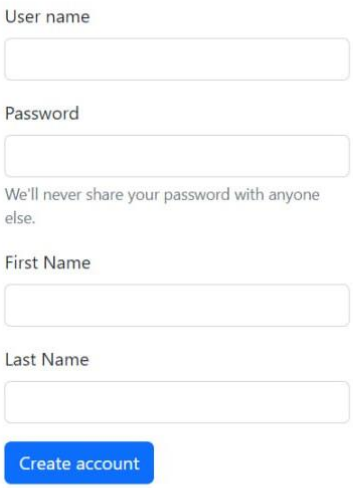

Рис. 2.9. Сторінка реєстрації користувача

Сторінка обробки демонструє логіку неправильно введених даних користувача. Це зображено на рисунку 2.10.

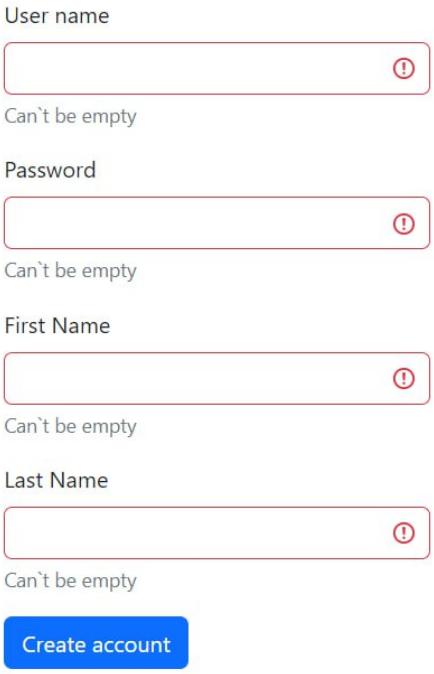

Рис. 2.10. Обробка виключених ситуацій

На цій сторінці Head of studies контролює акаунти вчителів та може видалити акаунт, якщо вчитель звільнився та редагувати його дані за запитом. Це зображено на рисунку 2.11.

Рис. 2.11. Головна сторінка Head of studies

На цій сторінці ми можемо редагувати дані вчителя. Це зображено на рисунку 2.12.

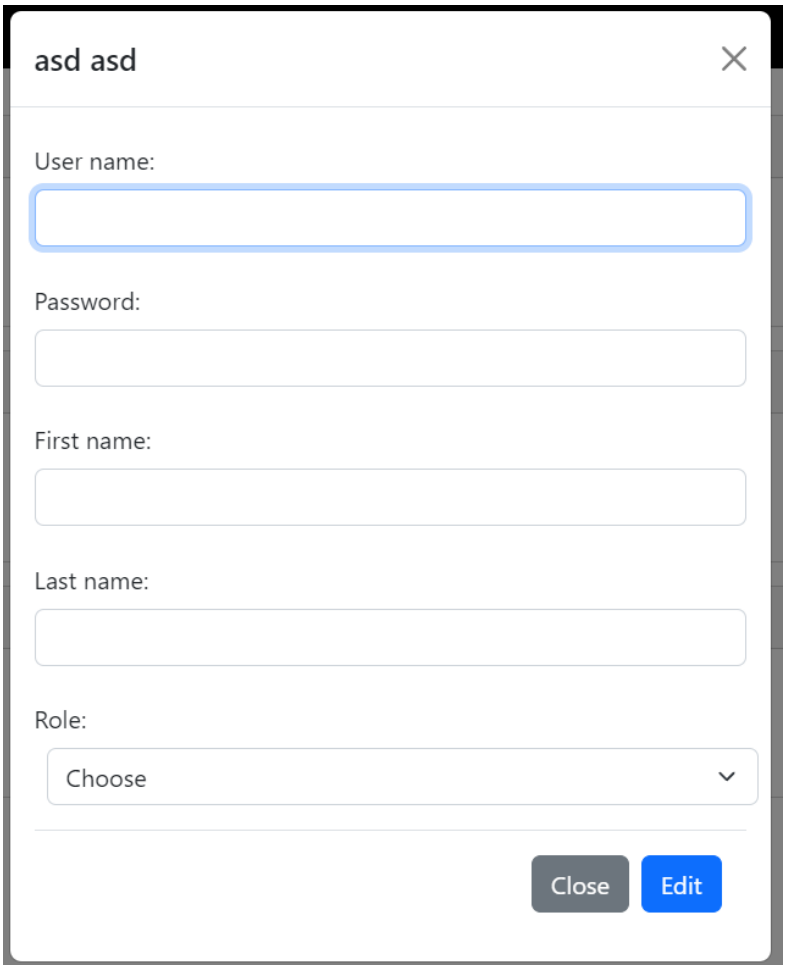

Рис. 2.12. Сторінка редагування даних вчителя

На цій сторінці ми можемо вибрати конкретного вчителя та розрахувати його продуктивність за певний період. Це зображено на рисунку 2.13.

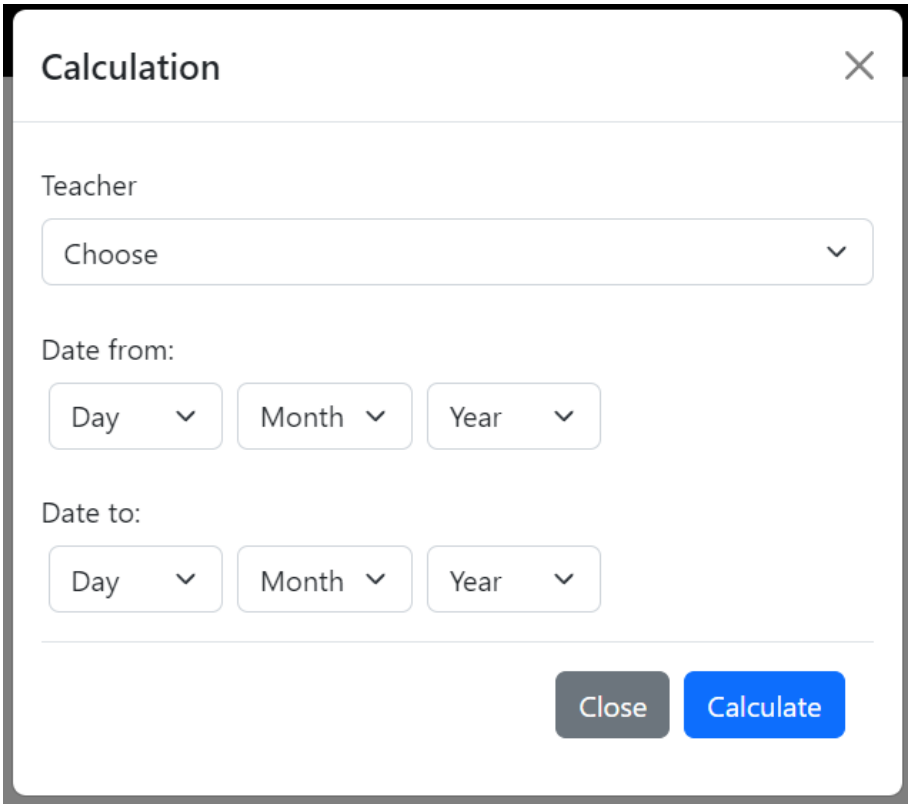

Рис. 2.13. Сторінка розрахунку продуктивності вчителя

На цій сторінці ми демонструємо розрахунок продуктивності вчителя. Це зображено на рисунку 2.14.

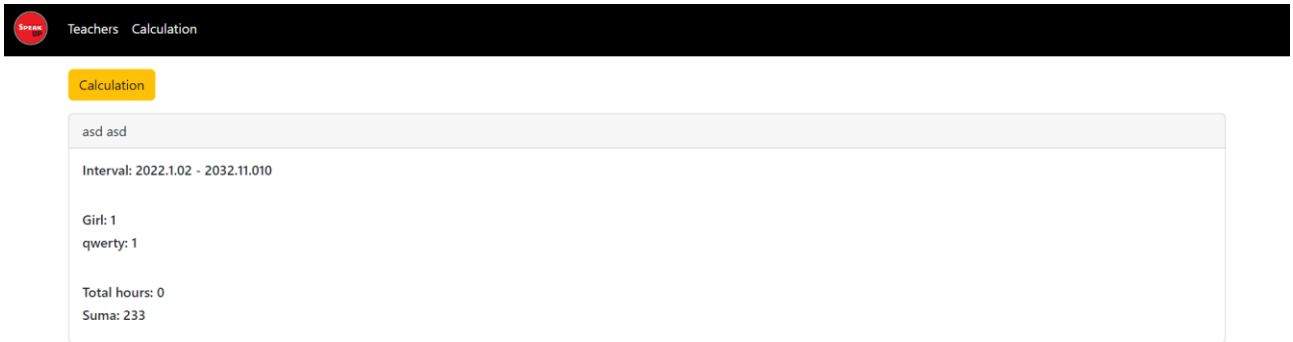

Рис. 2.14. Демонстрація розрахунку продуктивності вчителя

На цій сторінці ми можемо створювати новий звіт за сьогоднішній день. Це зображено на рисунку 2.15.

Рис. 2.15. Головна сторінка вчителя

На цій сторінці ми можемо створювати, видаляти та редагувати групи, які викладач веде або вів. Це зображено на рисунку 2.16.

Рис. 2.16. Головна сторінка груп які веде викладач

На цій сторінці ми створюємо нову групу та вказуємо ціну за урок в цій групі. Це зображено на рисунку 2.17.

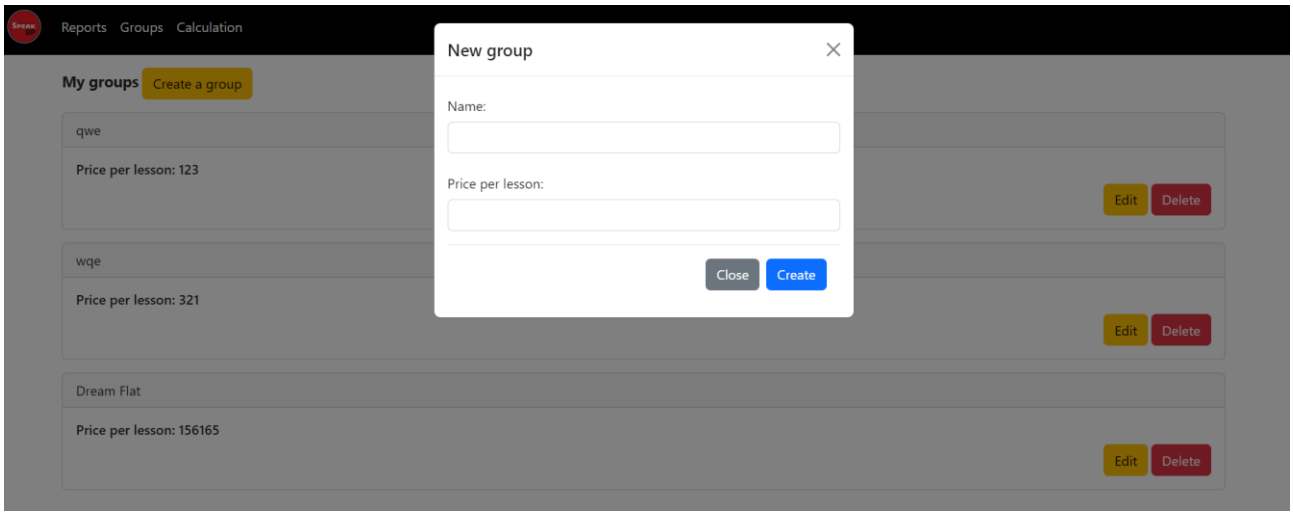

Рис. 2.17 Сторінка створення нової групи

На цій сторінці ми додаємо групи в яких в нас було сьогодні заняття. Це зображено на рисунку 2.18.

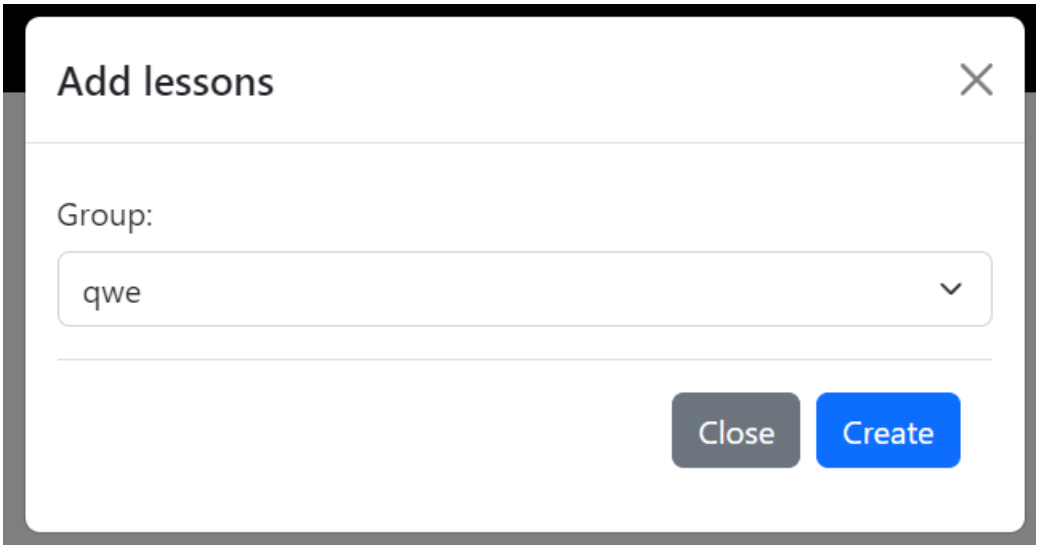

Рис. 2.18. Сторінка звітності викладача за сьогоднішній день

На рисунку 2.19. ми демонструємо додавання групи до звіту.

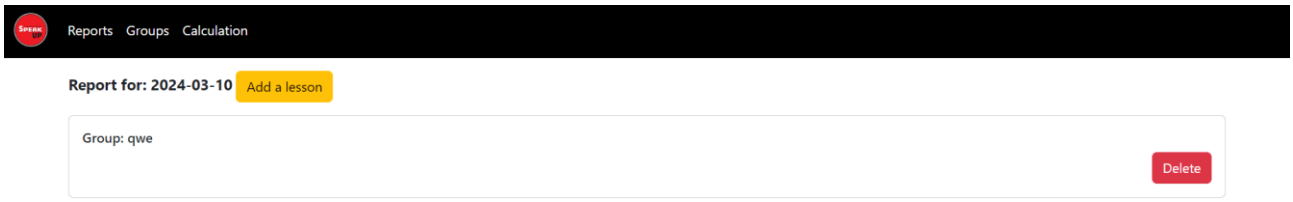

Рис. 2.19. Демонстрація додавання групи для звіту

На цій сторінці ми можемо розрахувати скільки годин викладач відпрацював та яку приблизну суму заробітної плати він заробив за певний період. Це зображено на рисунку 2.20.

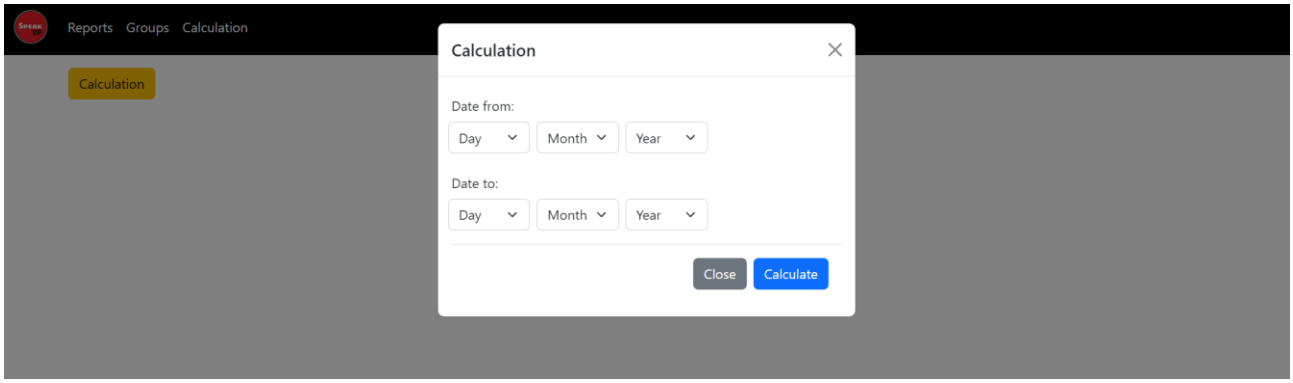

Рис. 2.20. Сторінка розрахунку роботи вчителя за певний період

На цій сторінці ми демонструємо розрахунок заробітної плати викладача за певний період. Це зображено на рисунку 2.21.

Рис. 2.21. Демонстрація розрахунку роботи викладача

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Прогнозована кількість операторів програми – 2050.
2. Коефіцієнт складності програми – 1,3.
3. Коефіцієнт корекції програми в ході її розробки – 0,1.
4. Годинна заробітна плата Java розробника – 215 грн/год.
5. Коефіцієнт збільшення витрати праці внаслідок недостатнього опису задачі – 1,2.
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8.
7. Вартість машино-години ЕОМ – 13 грн/год.

Годинну заробітну плату Java-розробника було розраховано за допомогою даних, отриманих з сайту "Української спільноти програмістів (DOU.ua)" [14]. Середня заробітна плата Java-розробника з досвідом роботи до одного року, за даними по Україні, становить приблизно 950 доларів США на місяць. На початок червня 2024 року, один долар США дорівнює 40 грн. Таким чином, середня заробітна плата в гривнях складає 38000 грн. При стандартному восьмигодинному робочому дні (приблизно 176 годин на місяць), середня годинна заробітна плата становить 215 грн/год.

Нормування праці у процесі розробки ПЗ значно ускладнюється через творчий характер роботи програміста. Тому складність розробки ПЗ може бути розрахована на основі системи моделей з різним ступенем точності оцінки. Трудомісткість розробки ПЗ можна обчислити за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{людино} - \text{годин} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ;

$t_{\text{д}}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = q \cdot C \cdot (1 + p) \tag{3.2}$$

де q – передбачуване число операторів (2050),

C – коефіцієнт складності програми (1,3),

p – коефіцієнт корекції програми в ході її розробки (0,1).

$$Q = 2050 \cdot 1{,}3 \cdot (1 + 0{,}1) = 3575$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{людино} - \text{годин} \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t_u = \frac{3575 \cdot 1{,}2}{80 \cdot 0{,}8} = 67{,}03 \text{ людино} - \text{годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25) \cdot k}, \text{людино} - \text{годин} \qquad (3.4)$$

де Q – умовне число операторів програми,

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$$t_a = \frac{2050}{23 \cdot 0{,}8} = 111 \text{ людино} - \text{годин}$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20..25) \cdot k}, \text{людино} - \text{годин} \qquad (3.5)$$

$$t_a = \frac{2050}{25 \cdot 0{,}8} = 102{,}5 \text{ людино} - \text{годин}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot k}, \text{людино} - \text{годин} \qquad (3.6)$$

$$t_{\text{отл}} = \frac{2050}{5 \cdot 0{,}8} = 512 \text{ людино} - \text{годин}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 \cdot t_{\text{отл}}, \text{людино} - \text{годин} \quad (3.7)$$

$$t_{\text{отл}}^{k} = 1{,}5 \cdot 512 = 768 \text{ людино} - \text{годин}$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{людино} - \text{годин} \quad (3.8)$$

де $\text{t}_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису.

$$\text{t}_{\text{др}} = \frac{Q}{(15..20) \cdot k}, \text{людино} - \text{годин} \quad (3.9)$$

$$\text{t}_{\text{др}} = \frac{2050}{15 \cdot 0{,}8} = 170 \text{ людино} - \text{годин}$$

$\text{t}_{\text{до}}$ – трудомісткість редагування, печатки й оформлення документації

$$\text{t}_{\text{до}} = 0{,}75 \cdot \text{t}_{\text{др}}, \text{людино} - \text{годин} \quad (3.10)$$

$$\text{t}_{\text{до}} = 0{,}75 \cdot 170 = 127 \text{ людино} - \text{годин}$$

$$\text{t}_{\text{д}} = 170 + 127 = 297 \text{ людино} - \text{годин}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 67,03 + 111 + 102,5 + 768 + 297 = 1395 людино-годин

У результаті ми розрахували, що в загальній складності необхідно 1395 людино-годин для розробки даного веб-додатку.

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ $К_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{по} = З_{зп} + З_{МВ}, грн \quad (3.11)$$

де $З_{зп}$ заробітна плата виконавців, яка визначається за формулою:

$$З_{зп} = t \cdot С_{пр}, грн \quad (3.12)$$

де t – загальна трудомісткість людино-годин,

$С_{пр}$ – середня годинна заробітна плата програміста, грн/год.

$$З_{зп} = 1395 \cdot 215 = 299925 \text{ грн}$$

$З_{МВ}$ – вартість машинного часу, необхідного для налагодження програми на ЕОМ.

$$З_{МВ} = t_{отл} \cdot С_{Мч}, грн \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год,

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 768 \cdot 13 = 9984 \text{ грн}$$

$$К_{по} = 299925 + 9984 = 309909 \text{ грн}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{міс} \quad (3.14)$$

де B_k– число виконавців,

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин.)

$$T = \frac{1395}{1 \cdot 176} = 7{,}92 \text{ міс}$$

Висновок: На розробку даного веб-додатку піде 1395 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 7,92 місяців при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення веб-додатку складатимуть 309909 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи було створення веб-додатку, який може бути використаний користувачами як на комп'ютерних пристроях, матиме зручний, інтуїтивно зрозумілий інтерфейс та необхідні функціональні можливості.

В результаті виконання кваліфікаційної роботи був розроблений вебдодаток для школи англійської мови "SpeakUp". Додаток є дуже зручним в своєму використані та має певні функції автоматизації процесів.

Під час виконання кваліфікаційної роботи були виконанні наступні задачі:

1. Розробка алгоритму для вирішення задачі.

2. Написання програмного коду для веб-додатку.

3. Розробка моделі бази даних.

Створений веб-додаток надає наступні функціональні можливості:

1. Можливість реєстрації для нових користувачів та авторизації для вже існуючих.

2. Можливість перегляду та редагування профілю користувача.

3. Користувач має список звітів за кожен відпрацьований день та власний список груп які він веде.

4. Розподілення користувачів за ролями.

5. Можливість перегляду кількості відпрацьованих годин та приблизної заробітньої плати за вказаний період.

Веб-додаток розроблений з використанням мови програмування Java та WEB-орієнтованого фреймворку Spring.

В ході кваліфікаційної роботи було вирахувано трудомісткість та порахована приблизна вартість розробленого додатку. Загальна вартість програмного продукту становить 309909 грн, а час створення – 1395 годин або 7,92 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Java (Programing language) / URL: https:// en.wikipedia.org/ wiki/Java_ (programming_language)
2. Overview of Spring Framework / URL: https://docs.spring.io/spring framework/docs/3.2.x/spring-framework-reference/html/overview.html
3. Web MVC Framework / URL: https://docs.spring.io/springframework /docs/3.2.x/spring-framework-reference/html/mvc.html
4. Spring Security / URL: https://spring.io/projects/spring-security
5. Spring Data JPA / URL: https://spring.io/projects/spring-data-jpa
6. Spring Boot Introduction / URL: https://www.tutorialspoint.com/spring_boot /spring_boot_introduction.htm
7. Project Lombok / URL: https://projectlombok.org/
8. HTML / URL: https://uk.wikipedia.org/wiki/HTML
9. CSS / URL: https://uk.wikipedia.org/wiki/CSS
10. Thymeleaf / URL: https://www.thymeleaf.org/
11. Hibernate Framework / URL: https://www.javatpoint.com/hibernatetutorial #:~:text=Hibernate%20is%20a%20Java%20framework,Persistence%20API) %20for%20data%20persistence
12. dou.ua зарплати / URL: https://jobs.dou.ua/salaries/?period=2022-12&position=Junior%20SE&technology=Java&experience=0-1&english=1-3
13. Apache Maven / URL: https://maven.apache.org/
14. Рейтинг мов програмування 2023 / URL: https://dou.ua/lenta/articles/language-rating-2023/
15. What is Bootstrap? / URL: https://www.techtarget.com/whatis/definition/bootstrap
16. Java Script / URL: https://uk.wikipedia.org/wiki/JavaScript
17. Мартін Фаулер. Рефакторинг коду на JavaScript: покращення проекту існуючого коду. ,2020. 464 с

18. The Java Tutorials. URl: https://docs.oracle.com/javase/tutorial/ (дата звернення: 10.11.2021).
19. . Alex Rodriguez. RESTful Web services: The basics. URL: https://developer.ibm.com/articles/ws-restful/
20. Document for the GPM Data. URL: https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf

ДОДАТОК А

КОД ПРОГРАМИ

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>SpeakUp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpeakUp</name>
  <description>SpeakUp</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
```

```
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>

<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jasper</artifactId>
    <version>9.0.38</version>
</dependency>
<dependency>
    <groupId>com.example</groupId>
    <artifactId>SpeakUp</artifactId>
```

```
        <version>0.0.1-SNAPSHOT</version>
      </dependency>
    </dependencies>

    <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
          <configuration>
            <excludes>
              <exclude>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
              </exclude>
            </excludes>
          </configuration>
        </plugin>
      </plugins>
    </build>

</project>
```

SpeakUpApplication.java

```
@SpringBootApplication
public class SpeakUpApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpeakUpApplication.class, args);
    }

}
```

AuthService.java

```
@Service
public class AuthService {

    public boolean hasAuthority(String name){
        for (GrantedAuthority authority : getUser().getAuthorities()) {
            if(name.equals(authority.getAuthority())){
                return true;
```

```
            }
        }
        return false;
    }

    public String getUserName(){
        return getUser().getUsername();
    }

    private CustomUserDetails getUser(){
        return (CustomUserDetails) getAuthentication().getPrincipal();
    }

    private Authentication getAuthentication(){
        SecurityContext context = SecurityContextHolder.getContext();

        return context.getAuthentication();
    }
}
```

CustomAuthProvider.java

```
@RequiredArgsConstructor
@Service
public class CustomAuthProvider implements AuthenticationProvider {
    private final CustomUserDetailsService customUserDetailsService;

    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        String username = authentication.getName();
        String password = authentication.getCredentials().toString();

        CustomUserDetails user = customUserDetailsService.loadUserByUsername(username);

        return checkPassword(user, password);
    }

    @Override
    public boolean supports(Class<?> authentication) {
        return UsernamePasswordAuthenticationToken.class.isAssignableFrom(authentication);
    }

    private Authentication checkPassword(CustomUserDetails rawUser, String rawPassword) {
```

```
            if (Objects.equals(rawPassword, rawUser.getPassword())) {


                return new UsernamePasswordAuthenticationToken(rawUser, rawUser.getPassword(),
rawUser.getAuthorities());
            } else {
                throw new BadCredentialsException("Bad credentials");
            }
        }
    }
```

CustomUserDetails.java

```
    public class CustomUserDetails implements UserDetails {

        Teacher teacher;
        public CustomUserDetails(Teacher teacher){
            this.teacher=teacher;

        }
        @Override
        public Collection<? extends GrantedAuthority> getAuthorities() {
            return Collections.singleton((GrantedAuthority) () -> teacher.getRole().toString());
        }
        public Integer getUserId(){
            return teacher.getId();
        }

        @Override
        public String getPassword() {
            return teacher.getPassword();
        }

        @Override
        public String getUsername() {
            return teacher.getUserName();
        }

        @Override
        public boolean isAccountNonExpired() {
            return true;
        }
```

```
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

CustomUserDetailService.java

```
@RequiredArgsConstructor
@Service
public class CustomUserDetailsService implements UserDetailsService {
    private final TeacherRepository teacherRepository;

    @Override
    public CustomUserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Optional<Teacher> byUserName = teacherRepository.findByUserName(username);
        if (byUserName.isEmpty()){
            throw new UsernameNotFoundException("User not found");
        }

        Teacher teacher = byUserName.get();
        String role = teacher.getRole().toString();
        String password = teacher.getPassword();

        return new CustomUserDetails(teacher);
    }

}
```

DefaultSecurityConfig.java

```
@RequiredArgsConstructor
@EnableWebSecurity
public class DefaultSecurityConfig {
```

```
    private final CustomAuthProvider authenticationProvider;

    @Bean
    PasswordEncoder passwordEncoder() {
        return PasswordEncoderFactories.createDelegatingPasswordEncoder();
    }

    @Bean
    SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                .authorizeRequests()
                .antMatchers("/login").permitAll()
                .antMatchers("/createUser").permitAll()
                .antMatchers("/error/*").permitAll()
                .anyRequest()
                .authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .defaultSuccessUrl("/")
                .usernameParameter("userName")
                .passwordParameter("password")
                .failureForwardUrl("/error")
        ;
        return http.build();
    }

    @Autowired
    public void injectCustomAuthProvider(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(authenticationProvider);
    }
}
```

MvcConfig.java

```
@EnableWebSecurity
@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
```

```
        registry.addViewController("/login").setViewName("login");

    }
}
```

CalculateController.java

```
@RequiredArgsConstructor
@RequestMapping("/calculate")
@Controller
public class CalculateController {
    private final CalculateService calculateService;
    private final TeacherService teacherService;

    @GetMapping
    public ModelAndView get(@RequestParam Map<String,String> map, Authentication authentication){
        ModelAndView modelAndView = new ModelAndView("calculate");
        modelAndView.addObject("years", calculateService.getYears());
        modelAndView.addObject("months", calculateService.getMonth());
        modelAndView.addObject("days", calculateService.getDay());
        if (!map.isEmpty()){
            modelAndView.addObject("dates",calculateService.date(map));
            modelAndView.addObject("hours",calculateService.sum(map,authentication));
            modelAndView.addObject("sumPrice",calculateService.sumPrice(map,authentication));
            modelAndView.addObject("exist", "true");
        } else {
            modelAndView.addObject("exist", "false");
        }
        return modelAndView;
    }

    @GetMapping("/teacher")
    public ModelAndView getCalculateTeacher(@RequestParam Map<String,String> map){
        ModelAndView modelAndView = new ModelAndView("headCalculate");
        modelAndView.addObject("teachers",teacherService.getAllTeacher());
        modelAndView.addObject("years",calculateService.getYears());
        modelAndView.addObject("months",calculateService.getMonth());
        modelAndView.addObject("days",calculateService.getDay());
        if (!map.isEmpty()){
            modelAndView.addObject("dates",calculateService.date(map));
            modelAndView.addObject("hours",calculateService.sumHead(map));

modelAndView.addObject("teacher",teacherService.getTeacherById(Integer.valueOf(map.get("teacherId"))));
```

```
            modelAndView.addObject("counts", calculateService.mapGroupCount(map));
            modelAndView.addObject("sumPrice",calculateService.sumPriceForHeadOfStudies(map));
            modelAndView.addObject("exist", "true");
        } else {
            modelAndView.addObject("exist", "false");
        }
        return modelAndView;
    }
}
```

CalculateService.java

```
@Getter
@RequiredArgsConstructor
@Service
public class CalculateService {
    private ReportRepository reportRepository;
    private GroupReportRepository groupReportRepository;
    private TeacherRepository teacherRepository;
    private List<String> day = new
ArrayList<>(Arrays.asList("1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16",
            "17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"));
    private Map<String,String> month = new HashMap<>();
    private List<String> years = new
ArrayList<>(Arrays.asList("2022","2023","2024","2025","2026","2027","2028","2029","2030","2031",
            "2032","2033","2034","2035","2036","2037","2038","2039","2040"));

    @Autowired
    public CalculateService(
            ReportRepository reportRepository,
            GroupReportRepository groupReportRepository,
            TeacherRepository teacherRepository
    ){
        this.reportRepository = reportRepository;
        this.groupReportRepository=groupReportRepository;
        this.teacherRepository=teacherRepository;

        month.put("1","Январь");
        month.put("2","Февраль");
        month.put("3","Март");
        month.put("4","Апрель");
        month.put("5","Май");
        month.put("6","Июнь");
```

```
        month.put("7","Июль");
        month.put("8","Август");
        month.put("9","Сентябрь");
        month.put("10","Октябрь");
        month.put("11","Ноябрь");
        month.put("12","Декабрь");
    }

    public Integer sum(Map<String,String> map, Authentication authentication){

        Date from = genFromDate(map);
        Date on = genOnDate(map);

        Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
        Integer sum = reportRepository.sum(userId, from, on);

        if (sum == null){
            return 0;
        }else {
            return sum;
        }
    }

    public String date(Map<String,String> map){
        String dayFrom = map.get("dayFrom");
        String monthFrom = map.get("monthFrom");
        String yearFrom = map.get("yearFrom");
        String dayOn = map.get("dayOn");
        String monthOn = map.get("monthOn");
        String yearOn = map.get("yearOn");

        String result = yearFrom + "." + monthFrom + "." + dayFrom + " - " + yearOn + "." + monthOn + "." +
dayOn;
        return result;
    }

    public Integer sumPrice(Map<String,String> map,Authentication authentication){
        Date from = genFromDate(map);
        Date on = genOnDate(map);

        Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
```

```
    Integer sum = groupReportRepository.sumBetweenDate(userId,from,on);

    if (sum == null){
        return 0;
    }
    return sum;
}

public Integer sumHead(Map<String,String> map){
    Date from = genFromDate(map);
    Date on = genOnDate(map);

    Integer userId = Integer.valueOf(map.get("teacherId"));
    Integer sum = reportRepository.sum(userId, from, on);

    if (sum == null){
        return 0;
    }else {
        return sum;
    }
}

public List<Map<String,Integer>> mapGroupCount(Map<String,String> map){
    Date from = genFromDate(map);
    Date on = genOnDate(map);
    int teacherId = Integer.parseInt(map.get("teacherId"));

    return groupReportRepository.mapGroupCount(teacherId,from,on);
}

private Date genFromDate(Map<String,String> map){
    String from = map.get("yearFrom") + "-" + map.get("monthFrom") + "-" + map.get("dayFrom");
    SimpleDateFormat format = new SimpleDateFormat();
    format.applyPattern("yyyy-MM-dd");

    try {
        Date fromDate = format.parse(from);
        return fromDate;
    } catch (ParseException e) {
        e.printStackTrace();
    }
```

```
        return null;
    }

    private Date genOnDate(Map<String,String> map){
        String on = map.get("yearOn") + "-" + map.get("monthOn") + "-" + map.get("dayOn");
        SimpleDateFormat format = new SimpleDateFormat();
        format.applyPattern("yyyy-MM-dd");

        try {
            Date onDate = format.parse(on);
            return onDate;
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return null;
    }

    public Integer sumPriceForHeadOfStudies(Map<String,String> map){
        Date from = genFromDate(map);
        Date on = genOnDate(map);
        int teacherId = Integer.parseInt(map.get("teacherId"));

        return groupReportRepository.sumBetweenDate(teacherId,from,on);
    }
}
```

ErrorController.java

```
@RequiredArgsConstructor
@RequestMapping("/error")
@Controller
public class ErrorController {

    @GetMapping
    public ModelAndView get(){
        ModelAndView modelAndView = new ModelAndView("error");
        return modelAndView;
    }

    @GetMapping("/createUser")
    public ModelAndView getErrorCreateUser(){
        ModelAndView modelAndView = new ModelAndView("errorCreateUser");
        return modelAndView;
```

```
    }

    @GetMapping("/emailExists")
    public ModelAndView getErrorEmailExists(){
        ModelAndView modelAndView = new ModelAndView("errorEmailExists");
        return modelAndView;
    }

}
```

ExelExportUtils.java

```
public class ExelExportUtils {

    private XSSFWorkbook workbook;
    private XSSFSheet sheet;
    private List<Report> listReport;

    public ExelExportUtils(List<Report> listReport) {
        this.listReport = listReport;
        workbook = new XSSFWorkbook();
    }

    private void createCells(Row row, int columnCount, Object value, CellStyle style) {
        sheet.autoSizeColumn(columnCount);
        Cell cell = row.createCell(columnCount);
        if( value instanceof Integer) {
            cell.setCellValue((Integer) value);
        } else if( value instanceof Double) {
            cell.setCellValue((Double) value);
        } else if( value instanceof LocalDate) {
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd LLLL yyyy");
            cell.setCellValue(((LocalDate) value).format(formatter));
        } else if( value instanceof Long) {
            cell.setCellValue((Long) value);
        } else {
            cell.setCellValue((String) value);
        }
        cell.setCellStyle(style);
    }

    private void createHeaderRow() {
        sheet = workbook.createSheet("Report");
```

```
        Row row = sheet.createRow(0);
        CellStyle style = workbook.createCellStyle();
        createCells(row, 0, "Report", style);
        sheet.addMergedRegion(new CellRangeAddress(0,0,0,3));

        row = sheet.createRow(1);
        createCells(row, 0, "id", style);
        createCells(row, 1, "date", style);
        createCells(row, 2, "numberLessons", style);
        createCells(row, 3, "teacher", style);
    }

    private void writeCustomerData() {
        int rowCount = 2;
        CellStyle style = workbook.createCellStyle();
        XSSFFont font = workbook.createFont();
        font.setFontHeight((short) 14);

        for(Report report : listReport) {
            Row row = sheet.createRow(rowCount++);
            int columnCount = 0;
            createCells(row, columnCount++, report.getId(), style);
            createCells(row, columnCount++, report.getDates(), style);
            createCells(row, columnCount++, report.getNumberOfLessons(), style);
            createCells(row, columnCount++, report.getTeacher().getFirstName(), style);
        }
    }

    public void exportDataToExel(HttpServletResponse response) throws IOException {
        createHeaderRow();
        writeCustomerData();
        ServletOutputStream outputStream = response.getOutputStream();
        workbook.write(outputStream);
        workbook.close();
        outputStream.close();
    }


}
```

ExportController.java

```
@RequestMapping("/export")
```

```
@AllArgsConstructor
@Controller
public class ExportController {

    private ExportService exportService;

    @GetMapping
    public void exportToExel(HttpServletResponse response) throws IOException {
        response.setContentType("application/octet-stream");
        response.setHeader("Content-Disposition", "attachment; filename=Report.xlsx");
        exportService.exportReportToExel(response);
    }


}
```

ExportService.java

```
@AllArgsConstructor
@Service
public class ExportService {

    private ReportRepository reportRepository;

    public List<Report> exportReportToExel(HttpServletResponse response) throws IOException {
        List<Report> listReport = reportRepository.findAll();
        ExelExportUtils exportUtils = new ExelExportUtils(listReport);
        exportUtils.exportDataToExel(response);
        return listReport;
    }
}
```

GroupController.java

```
@RequestMapping("/group")
@RequiredArgsConstructor
@Controller
public class GroupController {
    private final GroupService groupService;

    @GetMapping
    public ModelAndView getGroupPage(Authentication authentication){
        ModelAndView modelAndView = new ModelAndView("groupList");
        List<Group> groupByTeacher = groupService.getGroupByTeacher(authentication);
```

```
        modelAndView.addObject("groups",groupByTeacher);
        return modelAndView;
    }

    @GetMapping("/create")
    public ModelAndView getGroupCreatePage(){
        ModelAndView modelAndView = new ModelAndView("groupCreate");
        return modelAndView;
    }

    @PostMapping("/create")
    public void postCreateGroup(@RequestParam Map<String,String> map, HttpServletResponse resp, Authentication authentication){
        groupService.createGroup(map,resp,authentication);
    }

    @PostMapping("/delete")
    public void postDeleteGroup(@RequestParam Map<String,String> map, HttpServletResponse resp){
        groupService.deleteGroup(map,resp);
    }

    @PostMapping("/update")
    public void updateGroup(@RequestParam Map<String,String> map, HttpServletResponse resp){
        groupService.updateGroup(map,resp);
    }
}
```

GroupService.java

```
@RequiredArgsConstructor
@Service
public class GroupService {
    private final GroupRepository groupRepository;
    private final TeacherRepository teacherRepository;
    private final GroupReportRepository groupReportRepository;
    private final ReportRepository reportRepository;

    public List<Group> getGroupByTeacher(Authentication authentication){
        Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
        Teacher teacher = teacherRepository.findById(userId).get();
        List<Group> groups = groupRepository.findAllByTeacher(teacher).get();
        return groups;
    }
```

```
public void createGroup(Map<String,String> map, HttpServletResponse resp, Authentication authentication){
    String name = map.get("name");
    Integer price = Integer.parseInt(map.get("price"));
    Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
    Teacher byId = teacherRepository.findById(userId).get();

    Group group = new Group();
    group.setName(name);
    group.setPrice(price);
    group.setTeacher(byId);

    groupRepository.save(group);

    try {
        resp.sendRedirect("/group");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void deleteGroup(Map<String,String> map, HttpServletResponse resp){
    int id = Integer.parseInt(map.get("id"));
    Group group = groupRepository.findById(id).get();
    groupRepository.delete(group);

    try {
        resp.sendRedirect("/group");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void updateGroup(Map<String,String> map, HttpServletResponse resp){
    String name = map.get("name");
    String price = map.get("price");
    int id = Integer.parseInt(map.get("id"));

    Group group = groupRepository.findById(id).get();

    if (!name.equals("")){
```

```
            group.setName(name);
        }

        if (!price.equals("")){
            group.setPrice(Integer.valueOf(price));
        }

        groupRepository.save(group);

        try {
            resp.sendRedirect("/group");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

LoginController.java

```
@RequestMapping("/")
@RequiredArgsConstructor
@Controller
public class LoginController {
    private final AuthService authService;

    @GetMapping
    public void get(HttpServletResponse resp){

        if (authService.hasAuthority(Role.TEACHER.toString())){
            try {
                resp.sendRedirect("/report");
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else if (authService.hasAuthority(Role.HEAD_OF_STUDIES.toString())){
            try {
                resp.sendRedirect("/teacher");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
```

```
}
```

RegistrationController.java

```
@RequiredArgsConstructor
@RequestMapping("/createUser")
@Controller
public class RegistrationController {
    private final RegistrationService registrationService;

    @GetMapping
    public ModelAndView get(){
        ModelAndView modelAndView = new ModelAndView("createUser");
        return modelAndView;
    }

    @PostMapping
    public void creteUser(@RequestParam Map<String,String> map, HttpServletResponse resp){
        registrationService.createUser(map,resp);
    }
}
```

RegistrationService.java

```
@RequiredArgsConstructor
@Service
public class RegistrationService {
    private final TeacherRepository teacherRepository;

    public void createUser(Map<String,String> req, HttpServletResponse resp){
        String userName = req.get("userName");
        String password = req.get("password");
        String firstName = req.get("firstName");
        String lastName = req.get("lastName");
        Optional<Teacher> byUserName = teacherRepository.findByUserName(userName);
        if (userName.equals("") || password.equals("") || firstName.equals("") || lastName.equals("")){
            try {
                resp.sendRedirect("/error/createUser");
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else if (!byUserName.isEmpty()){
            try {
                resp.sendRedirect("/error/emailExists");
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
        }else {

            Teacher teacher = new Teacher();
            teacher.setUserName(userName);
            teacher.setPassword(password);
            teacher.setFirstName(firstName);
            teacher.setLastName(lastName);
            teacher.setRole(Role.TEACHER);

            teacherRepository.save(teacher);

            try {
                resp.sendRedirect("/login");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }
}
```

ReportController.java

```
@RequiredArgsConstructor
@RequestMapping("/report")
@Controller
public class ReportController {
    @Autowired
    private ReportService reportService;

    @GetMapping
    public ModelAndView getReportPage(Authentication authentication){
        ModelAndView modelAndView = new ModelAndView("reportList");
        modelAndView.addObject("reports",reportService.getReportByTeacher(authentication));
        modelAndView.addObject("teacher",1);
        modelAndView.addObject("sumForMonth",reportService.getSumForMont(authentication));
        return modelAndView;
    }

    @GetMapping("/date/*")
```

```
public ModelAndView getReportByIdPage(HttpServletRequest req,Authentication authentication){
    String[] split = req.getRequestURI().split("/");
    Integer id = Integer.parseInt(split[3]);
    ModelAndView modelAndView = new ModelAndView("report");
    modelAndView.addObject("report",reportService.getById(id));
    modelAndView.addObject("groups",reportService.getAllGroupByReport(id));
    modelAndView.addObject("groupList",reportService.getAllGroupByTeacher(authentication));
    return modelAndView;
}

@PostMapping("/create")
public void createReport(HttpServletResponse resp, Authentication authentication){
    reportService.createReport(resp,authentication);

}

@PostMapping("/delete")
public void deleteReport(@RequestParam Map<String,String> map,HttpServletResponse resp){
    reportService.deleteReport(map,resp);
}

@PostMapping("/create/group")
public void createGroupForReport(@RequestParam Map<String,String> map,HttpServletResponse resp){
    reportService.createGroupReport(map);

    try {
        resp.sendRedirect("/report/date/" + map.get("reportId"));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@PostMapping("/delete/group")
public void deleteGroupByReport(@RequestParam Map<String,String> map, HttpServletResponse resp){
    reportService.deleteGroupByReport(map);

    try {
        resp.sendRedirect("/report/date/" + map.get("reportId"));
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
    }
}
```

ReportService.java

```
@RequiredArgsConstructor
@Service
public class ReportService {
    private final ReportRepository reportRepository;
    private final TeacherRepository teacherRepository;
    private final GroupReportRepository groupReportRepository;
    private final GroupRepository groupRepository;

    public List<Report> getReportByTeacher(Authentication authentication){
        Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
        Teacher teacher = teacherRepository.findById(userId).get();
        List<Report> reports = reportRepository.findAllReportByTeacher(teacher).get();
        return reports;
    }

    public Report getById(Integer id){
        return reportRepository.findById(id).get();
    }

    public void createReport(HttpServletResponse resp, Authentication authentication){
        Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();

        Teacher teacher = teacherRepository.findById(userId).get();

        Report report = new Report();
        report.setNumberOfLessons(0);
        report.setTeacher(teacher);
        report.setDates(LocalDate.now());

        reportRepository.save(report);

        try {
            resp.sendRedirect("/report");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
public void deleteReport(Map<String,String> map, HttpServletResponse resp){
    int id = Integer.parseInt(map.get("id"));
    Optional<Report> byId = reportRepository.findById(id);
    reportRepository.delete(byId.get());

    try {
        resp.sendRedirect("/report");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public List<Group> getAllGroupByReport(Integer idReport){
    Report report = reportRepository.findById(idReport).get();
    List<GroupReport> list = groupReportRepository.findAllByReport(report).get();
    List<Group> groupList = new ArrayList<>();
    for (GroupReport groupReport:list) {
        groupList.add(groupReport.getGroup());
    }
    return groupList;
}

public List<Group> getAllGroupByTeacher(Authentication authentication){
    Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
    Teacher teacher = teacherRepository.findById(userId).get();
    return groupRepository.findAllByTeacher(teacher).get();
}

public void createGroupReport(Map<String,String> map){
    int reportId = Integer.parseInt(map.get("reportId"));
    int groupId = Integer.parseInt(map.get("groupId"));
    Report report = reportRepository.findById(reportId).get();
    Group group = groupRepository.findById(groupId).get();

    report.setNumberOfLessons(report.getNumberOfLessons() + 1);
    reportRepository.save(report);

    GroupReport groupReport = new GroupReport();
    groupReport.setGroup(group);
    groupReport.setReport(report);
```

```
    groupReportRepository.save(groupReport);
}

public void deleteGroupByReport(Map<String,String> map){
    int reportId = Integer.parseInt(map.get("reportId"));
    int groupId = Integer.parseInt(map.get("groupId"));

    Report report = reportRepository.findById(reportId).get();
    Group group = groupRepository.findById(groupId).get();

    report.setNumberOfLessons(report.getNumberOfLessons() - 1);
    reportRepository.save(report);

    GroupReport groupReport = groupReportRepository.findByGroupAndAndReport(group,report).get();
    groupReportRepository.delete(groupReport);
}

public Integer getSumForMont(Authentication authentication){
    Integer userId = ((CustomUserDetails) authentication.getPrincipal()).getUserId();
    LocalDate now = LocalDate.now();
    String[] arr = String.valueOf(now).split("-");
    Integer day = Integer.valueOf(arr[2]);
    Integer month = Integer.valueOf(arr[1]);
    Integer year = Integer.valueOf(arr[0]);
    String from;
    String on;
    if (day < 21){
        month--;
        from = year + "-" + month + "-21" ;
        month++;
        on = year + "-" + month + "-20";
    } else {
        from = year + "-" + month + "-21" ;
        month++;
        if (month == 13){
            month=1;
            year++;
        }
        on = year + "-" + month + "-20";
    }
```

```
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat();
        simpleDateFormat.applyPattern("yyyy-MM-dd");

        try {
            Date fromDate = simpleDateFormat.parse(from);
            Date onDate = simpleDateFormat.parse(on);

            Integer sumBetweenDate = groupReportRepository.sumBetweenDate(userId, fromDate, onDate);
            if (sumBetweenDate == null){
                return 0;
            }
            return sumBetweenDate;

        } catch (ParseException e) {
            e.printStackTrace();
        }
        return 0;
    }

}
```

TeacherController.java

```
@RequestMapping("/teacher")
@RequiredArgsConstructor
@Controller
public class TeacherController {
    private final TeacherService teacherService;

    @GetMapping
    public ModelAndView get(){
        ModelAndView modelAndView = new ModelAndView("teacher");
        modelAndView.addObject("teachers",teacherService.getAllTeacher());
        modelAndView.addObject("roles", new
ArrayList<Role>(Arrays.asList(Role.TEACHER,Role.HEAD_OF_STUDIES)));
        return modelAndView;
    }

    @GetMapping("/date/*")
    public ModelAndView getTeacherReports(HttpServletRequest req){
        ModelAndView modelAndView = new ModelAndView("teacherReportById");
        String[] split = req.getRequestURI().split("/");
        Integer id = Integer.valueOf(split[3]);
```

```
        modelAndView.addObject("reports",teacherService.getReportsByTeacher(id));
        modelAndView.addObject("teacher",teacherService.getTeacherById(id));
        return modelAndView;
    }

    @PostMapping("/delete")
    public void deleteTeacher(@RequestParam Map<String,String> map, HttpServletResponse resp){
        teacherService.deleteTeacher(map);

        try {
            resp.sendRedirect("/teacher");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @PostMapping("/update")
    public void updateTeacher(@RequestParam Map<String,String> map,HttpServletResponse
resp,HttpServletRequest req){
        teacherService.updateTeacher(map);
        try {
            resp.sendRedirect("/teacher");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

TeacherService.java

```
@RequiredArgsConstructor
@Service
public class TeacherService {
    private final TeacherRepository teacherRepository;
    private final ReportRepository reportRepository;

    public Teacher getTeacherById(Integer id){
        return teacherRepository.findById(id).get();
    }

    public List<Teacher> getAllTeacher(){
        return teacherRepository.findAll();
    }
```

```
public List<Report> getReportsByTeacher(Integer id){
    Teacher teacher = teacherRepository.findById(id).get();
    return reportRepository.findAllReportByTeacher(teacher).get();
}

public void deleteTeacher(Map<String,String> map){
    int id = Integer.parseInt(map.get("id"));
    Teacher teacher = teacherRepository.findById(id).get();
    teacherRepository.delete(teacher);
}

public void updateTeacher(Map<String,String> map){
    int id = Integer.parseInt(map.get("id"));
    String userName = map.get("userName");
    String password = map.get("password");
    String firstName = map.get("firstName");
    String lastName = map.get("lastName");
    String role = map.get("role");

    Teacher teacher = teacherRepository.findById(id).get();


    if (!userName.equals("")){
        teacher.setUserName(userName);
    }

    if (!password.equals("")){
        teacher.setPassword(password);
    }

    if (!firstName.equals("")){
        teacher.setFirstName(firstName);
    }

    if (!lastName.equals("")){
        teacher.setLastName(lastName);
    }

    if (!role.equals("Выбрать") && !role.equals("")){
        teacher.setRole(Role.valueOf(role));
```

```
        }

        teacherRepository.save(teacher);
    }

}
```

GroupReport.java

```
@Data
@Entity
public class GroupReport {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @OneToOne
    @JoinColumn(name = "group_id")
    private Group group;

    @OneToOne
    @JoinColumn(name = "report_id")
    private Report report;
}
```

GroupReportRepository.java

```
@Repository
public interface GroupReportRepository extends JpaRepository<GroupReport,Integer> {

    Optional<List<GroupReport>> findAllByReport(Report report);

    Optional<GroupReport> findByGroupAndAndReport(Group group, Report report);

    @Query(value = "select sum(g.price)\n" +
            "from group_report gr\n" +
            "inner join report r on r.id=gr.report_id\n" +
            "inner join groups g on g.id=gr.group_id\n" +
            "where r.teacher_id = :id and r.dates between :startDate and :endDate", nativeQuery = true)
    Integer sumBetweenDate(@Param("id") Integer id, @Param("startDate") Date startDate, @Param("endDate")
Date endDate);

    @Query(value = "select g.name, count(g.id)\n" +
            "from group_report gr\n" +
```

```
            "inner join report r on r.id=gr.report_id\n" +
            "inner join groups g on g.id=gr.group_id\n" +
            "where r.teacher_id = :id and r.dates between :startDate and :endDate\n" +
            "group by g.id", nativeQuery = true)
    List<Map<String, Integer>> mapGroupCount(@Param("id") Integer id, @Param("startDate") Date start, @Param("endDate") Date end);


}
```

Group.java

```
@Data
@Entity
@Table(name = "groups")
public class Group {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Integer id;

   @Column
   private Integer price;

   @Column
   private String name;

   @ManyToOne
   @JoinColumn(name = "teacher_id")
   private Teacher teacher;
}
```

GroupRepository.java

```
@Repository
public interface GroupRepository extends JpaRepository<Group,Integer> {
    Optional<List<Group>> findAllByTeacher(Teacher teacher);
}
```

Report.java

```
@Data
@Entity(name = "report")
public class Report {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Integer id;
```

```
    @Column
    private LocalDate dates;

    @Column
    private Integer numberOfLessons;

    @ManyToOne
    @JoinColumn(name = "teacher_id")
    private Teacher teacher;
}
```

ReportRepository.java

```
@Repository
public interface ReportRepository extends JpaRepository<Report,Integer> {
    Optional<List<Report>> findAllReportByTeacher(Teacher teacher);

    @Query(value = "select sum(r.number_of_lessons)\n" +
            "from report r\n" +
            "where r.teacher_id = :id and r.dates between :startDate and :endDate",nativeQuery = true)
    Integer sum(@Param("id") Integer id,@Param("startDate") Date startDate, @Param("endDate") Date endDate);
}
```

Role.java

```
@Getter
public enum Role {
    TEACHER,
    HEAD_OF_STUDIES
}
```

Teacher.java

```
@Data
@Entity
@Table(name = "teacher")
public class Teacher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column
    private String userName;
```

```
    @Column
    private String password;

    @Column
    private String firstName;

    @Column
    private String lastName;

    @Enumerated(EnumType.STRING)
    private Role role;
}
```

TeacherRepository.java

```
@Repository
public interface TeacherRepository extends JpaRepository<Teacher,Integer> {

    Optional<Teacher> findByUserName(String userName);

}
```

Application.properties

```
server.port=8081
spring.datasource.url=jdbc:postgresql://localhost:5432/speakUP
spring.datasource.username=postgres
spring.datasource.password=VisaGold1234
```

V1_tables.sql

```
create table teacher(
    id serial primary key,
    user_name varchar(40),
    password varchar(200),
    first_name varchar(40),
    last_name varchar(40),
    role varchar(40)
);

create table groups(
    id serial primary key,
    name varchar(100),
    price int,
    teacher_id int,
    foreign key (teacher_id) references teacher(id) on delete cascade
```

```
);

create table report(
    id serial primary key,
    dates date,
    total_price int,
    number_of_lessons int,
    teacher_id int,
    foreign key (teacher_id) references teacher(id) on delete cascade
);

create table group_report(
    id serial primary key,
    group_id int,
    report_id int,
    foreign key (group_id) references groups(id) on delete cascade,
    foreign key (report_id) references report(id) on delete cascade
)
```

V2_debug_table_report.sql

```
alter table report
drop column total_price;
```

V3_add_head_of_studies.sql

```
insert into teacher(id,user_name,password,first_name,last_name,role)
values (4,'admin','admin','admin','admin','HEAD_OF_STUDIES');
```

ДОДАТОК Б

Відгук

керівника економічного розділу

на кваліфікаційну робот бакалавра

на тему:

«Розробка веб-системи обліку роботи викладачів

англійської мови»

Студента групи 122-21СК-1 Мазніченко Іллі Вадимовича

Керівник економічного розділу доц. Л.В. Касьяненко

Каф. ПЕП та ПУ, к.е.н

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім'я файлу	**Опис**
Пояснювальні документи	
Кваліфікаційна робота Мазніченко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Мазніченко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Мазніченко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Мазніченко.ppt	Презентація кваліфікаційної роботи