МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра

ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАТФОРМИ ДЛЯ ОН-ЛАЙН НАВЧАННЯ

Здобувач освіти гр. ІНз-81с. Станіслав СТОВОЛОС

Науковий керівник Галина ОЛЕКСІЄНКО
кандидат ф.-м. наук, доцент

Завідувач кафедри Анатолій ДОВБИШ
Доктор технічних наук, професор

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. Кафедри Довбиш А.С.

"___"_____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи бакалавра

Студента четвертого курсу, групи ІНз-81с спеціальності 122 - Комп'ютерні науки, заочної форми навчання Стоволоса С. Ю.

Тема: Інформаційне та програмне забезпечення платформи для он-лайн навчання

Затверджена наказом по СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) Аналітичний огляд літератури; 2) Постановка завдання; 3) Вибір методів рішення завдання. 4) Практична реалізація поставленого завдання; 5) Висновки;

Дата видачі завдання "_____"_____2022р.

Керівник випускної роботи _____Олексієнко Г.А.

Завдання прийняв до виконання _____Стоволос С.Ю.

РЕФЕРАТ

Записка: 36 – сторінок, 19 - рисунків, 1 – додаток, 2 – таблиці, 17 – джерел.

Об’єкт дослідження – Інформаційна система платформи он-лайн навчання.

Мета роботи – розробка та програмна реалізація он-лайн платформи

Методи досліджень – методи розробки логіки для інформаційних систем, методи розробки веб-додатків.

Результати – Було проведено аналіз літератури, он-лайн систем та їх коду. Зроблено аналіз існуючих систем-аналогів, та способи вирішення проблем при їх реалізації. Розроблено логіку веб-додатку на мові Java, розроблена база даних та веб-сторінки. За використання стандартних технологій додаток має властивість до розвитку та підключення нових технологій. Використання технологій Data Access Object дозволяє самостійно розробляти необхідні запити, тому програма функціональна і її можна перероблювати як конструктор. Було розроблено html сторінки для відображення інформації, з’єднано їх з базою даних, розроблено три рівня доступу:Student, Professor та Admin – які можуть вносити зміни до бази даних, Student – найменше, Admin – найбільше.

JAVA, DAO, SERVLET, БАЗА ДАНИХ,
ВЕБ, JDBC, ОБ’ЄКТ, КЛАС, JSP

ЗМІСТ

ВСТУП

Наш час показав переваги та необхідність отримання знань в режимі онлайн в не залежності від їх рівня, будь то загальноосвітня програма навчання, курси або просто пізнавальні лекції. Особливою перевагою такого способу навчання є можливість отримання інформації з перших вуст не залежно від місця перебування, що дозволяє здобувати знання навіть поза кордоном, адже ні студенту ні викладачу не потрібно перетинати границі та сотні кілометрів для проведення заняття, для людей з обмеженими можливостями це взагалі один з небагатьох способів здобуття освіти. Пандемія коронавірусу зміцнила та дала сильний поштовх у напрямі он-лайн навчання, що ще раз підтвердило необхідність такого способу здобуття інформації. Багато ІТ-компаній саме через такий спосіб проводило навчання для програмістів, веб-дизайнерів, графічних-дизайнерів та інших професій ще до пандемії, бо хто як не люди, робота яких частіше зводиться до віддалених робіт на комп'ютері найкраще підходять для такого виду навчання. Все що потрібно зараз щоб отримати знання – це пристрій (комп'ютер, телефон, планшет) та інтернет, у наш час такий набір є навіть у віддалених селах, тому навіть будучи далеко від індустріалізації людина може отримувати якісні знання.

В даній роботі буде створено он-лайн платформу, програмний код до неї, та базу даних. Буде повністю розроблена логіка, запити та архітектура для подальшого оновлення та покращенню веб-дизайну.

1. АНАЛІЗ ПРОБЛЕМ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Принципи веб-додатку

В наш час мати власний бізнес чи зайнятість без виходу в інтернет майже нереально, якщо ти хочеш щоб люди знали про твою діяльність – найкращий спосіб це мати власний сайт, а якщо справа більш розвинена то і власний веб-додаток. З появою інтернету люди створювали свої інтернет сторінки, наразі знайти такі майже неможливо через велику прірву технологій що пішли у перед. Але мало зробити сторінку сайту, потрібно щоб вона виводила якусь інформацію та могла обробляти запити, адже кому в сучасному світі потрібна сторінка роль якої просто вивести текст на екран. Весь цей набір називають Інформаційною системою – це знаходження, пошук та обробка інформації ресурсів, що забезпечує функціонал системи.

Однозначно перевагою інформаційних систем можна назвати інтеграції – по суті це дозволяє нам використовувати дані не один раз для вирішення конкретного рішення, а багато, що збільшить число виконаних завдань.

Тому наразі всі провідні компанії мають свої веб-додатки, адже це суттєво спростовує надання та обробки інформації, а слідом і задовільнити кількість клієнтів стає в рази простіше.

Сам веб-додаток представляє набір інтернет сторінок які зв’язали до кучі, користувач може бачити та взаємодіяти з інформацією яку ми йому надаємо, а система в залежності від запиту буде її обробляти. Детально схему інформаційної системи зображено на рисунку 1.

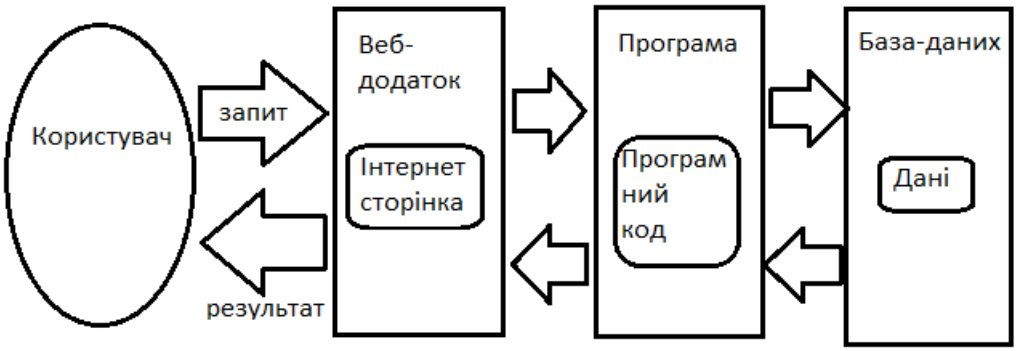

Рисунок 1. Робота інформаційної системи

1.2. Огляд існуючих рішень.

Переглянемо додатки що вже є, підберемо їх по критеріям які ми становили для власної системи, а саме ті які пропонують нам навчальні матеріали у вигляді курсів. Під наші критерії було підібрано три системи:

1) CourSera – платформа що має велику базу знань, при цьому є співпраця з провідними компаніями.
2) edX – платформа що дозволяє вивчати курси з інструктором.
3) CodeAcademy – вузько направлена платформа для вивчення IT.

Для порівняння використовуємо наступні критерії:

Авторизація – можливість авторизації на сайт через додаткові додатки (наприклад пошта).

Курси – наявність платних та безплатних знань на сайті.

Система навчання – навчання з викладачем чи самостійно, наявність додаткового матеріалу (відео, фото).

Практичність – зручність користування додатком, зрозумілість інтерфейсу.

Доступність – кількість різномовних курсів, зміна мови інтерфейсу.

таблиця 1 – Порівняльна характеристика існуючих систем

	CourSera	edX	CodeAcademy
Авторизація через Google	+	-	+
Авторизація через соц. мережі	+	-	+
Наявність безплатних курсів	+	+	+
Наявність платних курсів	+	+	+
Викладач (аудиторія)	+	+	+
Викладач (особисто)	-	-	+
Самостійне вивчення матеріалу	-	-	+
Наявність допоміжних файлів (фото, відео)	-	-	+
Пошук по критеріям	+	+	+
Наявність англійської мови (сайту та курсів)	+	+	+

Наявність української мови (сайту та курсів)	-	+	-
Рейтинг курсів	+	-	-
Різноманітність галузей	+	+	-

Завдяки таблиці 1 ми бачимо що дані системи не є ідеальними, та при цьому з функцією надання навчальних матеріалів вони справляються. З цього при розробці власного додатку слід врахувати що при тесті даних додатків нам не вистачало як користувачу. Отже в нашу систему потрібна мати дві основні мови – англійську та українську, контакти викладача, якщо потрібно дізнатися про курс до того як ми записались на нього, можливість додавання відео, фото, аудіо – матеріалів та оцінки їх.

1.3. Постановка задачі.

Метою даної роботи є написання програмного коду та створення функціоналу онлайн платформи для навчання. Для того щоб виконати поставлені цілі, необхідно зробити:

1. Розробка бази даних
    - Проектування БД
    - Зв’язок таблиць через зовнішні та внутрішні ключи
    - Логіка поводження даних при їх редагуванні (зміна, видалення)
2. Розробка програмного коду на мові Java
    - Створення необхідних класів-об’єктів
    - Створення DAO-класів для роботи з базою даних
    - Створення Сервлетів для роботи з веб-сторінкою

3. Розробка веб сторінки
    - Створення архітектури
    - Виведення функціоналу для відправки запитів
    - Створення різних сторінок та переходу між ними (наприклад зі сторінки про всі курси – можна перейти на сторінку вибраного)
    - Створення сторінки авторизації з можливістю регістрації для нових користувачів
    - Відображення інформації про доступні курси та інформації про них

2. ВИБІР МЕТОДУ РІШЕННЯ

2.1. Вибір програмного середовища.

Для розробки даної системи була вибрана мова програмування Java та середовище IntellIj IDEA для неї, база даних реалізована на мові MySQL у середовищі Workbench. Такий вибір середовища зумовлений перевагами роботою з сервером та великим вибором драйверів та бібліотек.

Вигляд середовища IntellIj IDEA:

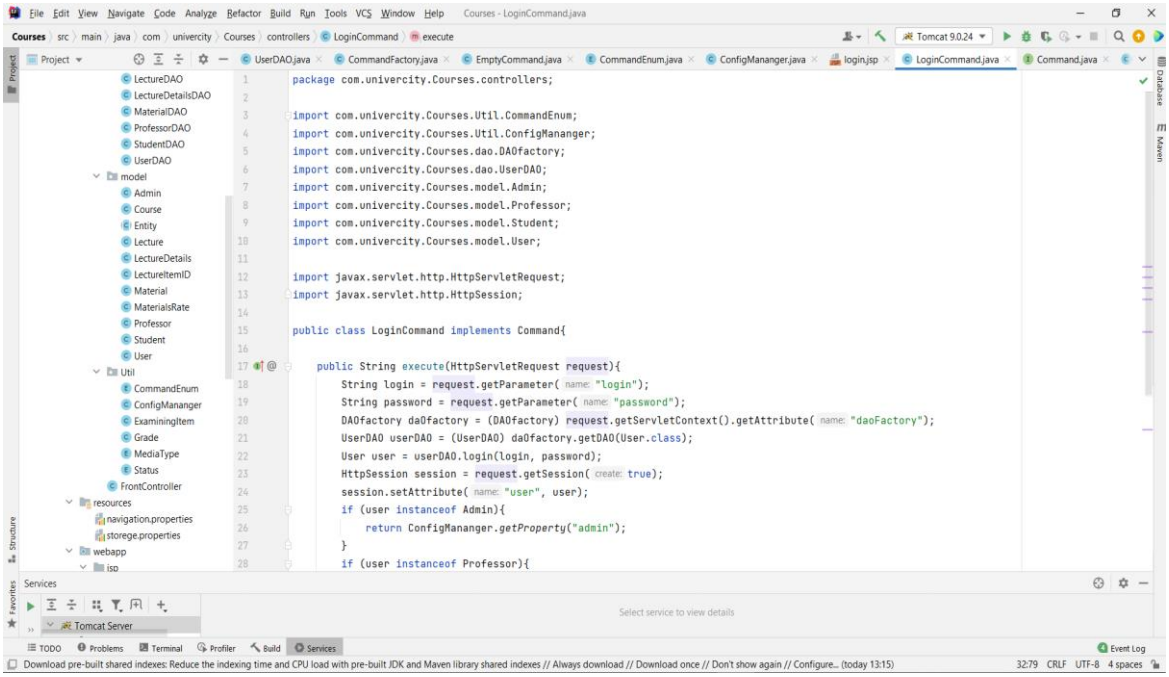

рисунок 2.1 IntellIj IDEA

Зліва маємо проект та його класи, посередині-справа відкритий в даний час клас, а знизу – вивід інформації при запуску та роботі програми.

Вигляд середовища MySQL Workbench:

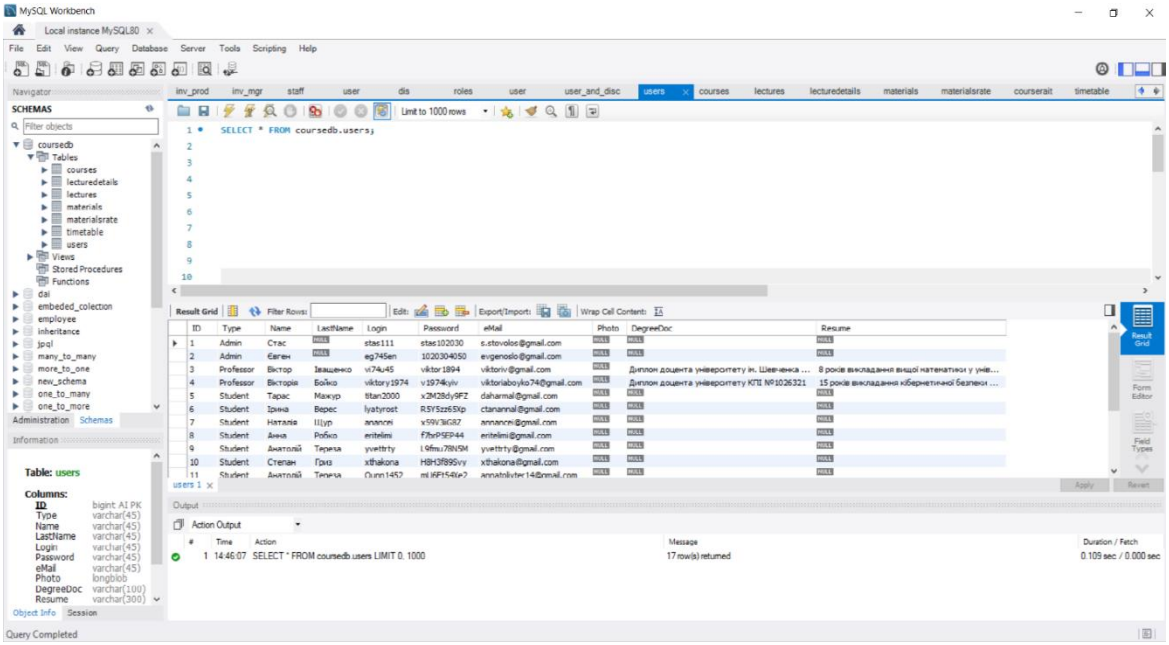

рисунок 2.2 MySQL

Навігація тут дещо схожа з IntellIj IDEA, зліва маємо базу даних, під нею бачимо вибрану таблицю та рядки які вона містить, посередині у нас відкрита робоча зона цієї таблиці, знизу – результати запиту, під результатами ми бачимо дії що були виконані.

2.2. Робота програмного коду з іншими системами.

Java є достатньо універсальною мовою програмування, велика кількість бібліотек, класів, інтерфейсів надає можливість розробляти різні типи проектів. Для створення веб-додатків є також велика кількість бібліотек, що дає гнучкість та практичність. Для веб додатку звісно необхідно мати не тільки один код, але і базу даних для збереження інформації про користувачів та для користувачів, також обов’язковим є створення серверу, щоб нашим додатком можна було користуватись. Основним завданням мови Java буде логіка та обробка запитів, а щоб це стало можливим – потрібно наладити зв’язок з базою даних та сервером.

Тобто роль програмного коду – це надсилання даних між двома берегами – базою даних та веб.

Для початку кращим рішенням буде створення візуальної схеми бази даних, так ми будемо бачити які класи нам необхідно створювати, та якими вони будуть і що будуть зберігати.

Далі – створення класів у програмному середовищі, розробка архітектури. І вже останнім етапом буде підключення до веб.

2.3. Програмний код Java у зв'язку з БД та ВЕБ.

1. Робота програмного коду з базою даних.

Для того щоб програма могла надсилати та отримувати якісь дані з бази даних використовують допоміжні драйвери та інтерфейси, які саме – залежить від мови програмування на якій пишеться код та бази даних. Наша програма розроблена на мові Java, а систему керування базами даних – вибрано MySQL. Для зв'язку з базою даних у мові Java є стандартний інтерфейс – JDBC (Java DataBase Connectivity), його драйвери і забезпечують реалізацію інтерфейсів для СУБД та протоколів. JDBC – надсилає запити тільки до реляційних баз даних, і тільки до тих у яких існують драйвери.

JDBC має чотири типи драйверів:

1 – Драйвер, який для взаємодії з СУБД використовує інший прикладний інтерфейс.

2 – Драйвер працюючий напряму з СУБД – мережевий.

3 – Драйвер, що для роботи з СУБД використовує зовнішні бібліотеки.

4 – Драйвер, який працює через протокол який є мережевим та незалежним від СУБД – для цього використовується проміжний java-сервер.

Саме тому для розробки програми ми вибрали такий метод роботи коду з базою даних, а саме через JDBC.

Для полегшення роботи програми з базою даних зазвичай використовують Framework-и – це вже створений іншими розробниками каркасний код який дозволяє нам адаптувати його під свої потреби. Це зменшує час та складність написання коду.

2. Робота програмного коду з веб-додатком.

Для нашої задачі необхідною умовою є доступність програми, тому вибір зупинився на веб версії. Перевага програми java у вигляді веб – це те, що для взаємодії з програмою користувачу не потрібно встановлювати додаткові додатки чи програми. Для того щоб програма отримувала та надавала запити в веб додаток ми використаємо сервлети.

Сервлети – інтерфейс java, який дає нам класи для обробки коду HTML, та дозволяє нам отримувати та надсилати запроси серверу.

При створені сервлети тримаються у віртуальній машині JVM – це дає нам функціонування garbage collection та захист від витоку інформації.

Для кожного користувача створюється свій сервлет, він реагує на запит, звертається до коду, та повертає вже готову відповідь у HTML-документі, при цьому кількість запитів не має обмежень.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1. Створення об’єктів Java.

Розділимо нашу java структуру на чотири папки: Model, DAO, Util та Controllers

У папці Model знаходяться основні об’єкти: User, Student, Lecture, Material та інші.

Для початку створюємо об’єкт Entity – це абстрактний клас, що зберігає поле ID класів-наслідників.

Далі створюємо необхідні нам класи наслідуючи Entity. Для користувачів, а вони у нас 3-х типів: Студенти, викладачі та адміністратори – ми створюємо спільний батьківський клас User – він зберігає спільні між ними поля, наприклад ім’я.

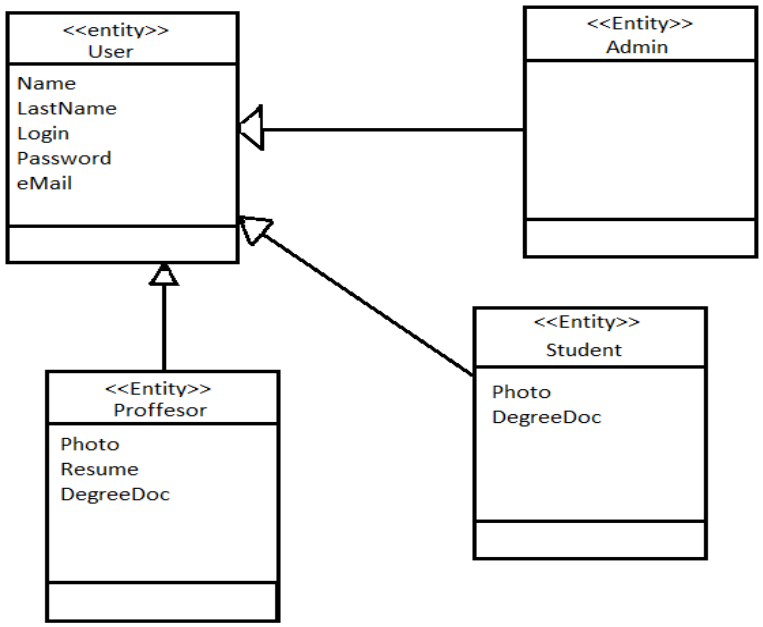

рисунок 3.1 Структура класів користувачів

Повна ієрархія паки Model виглядає так:

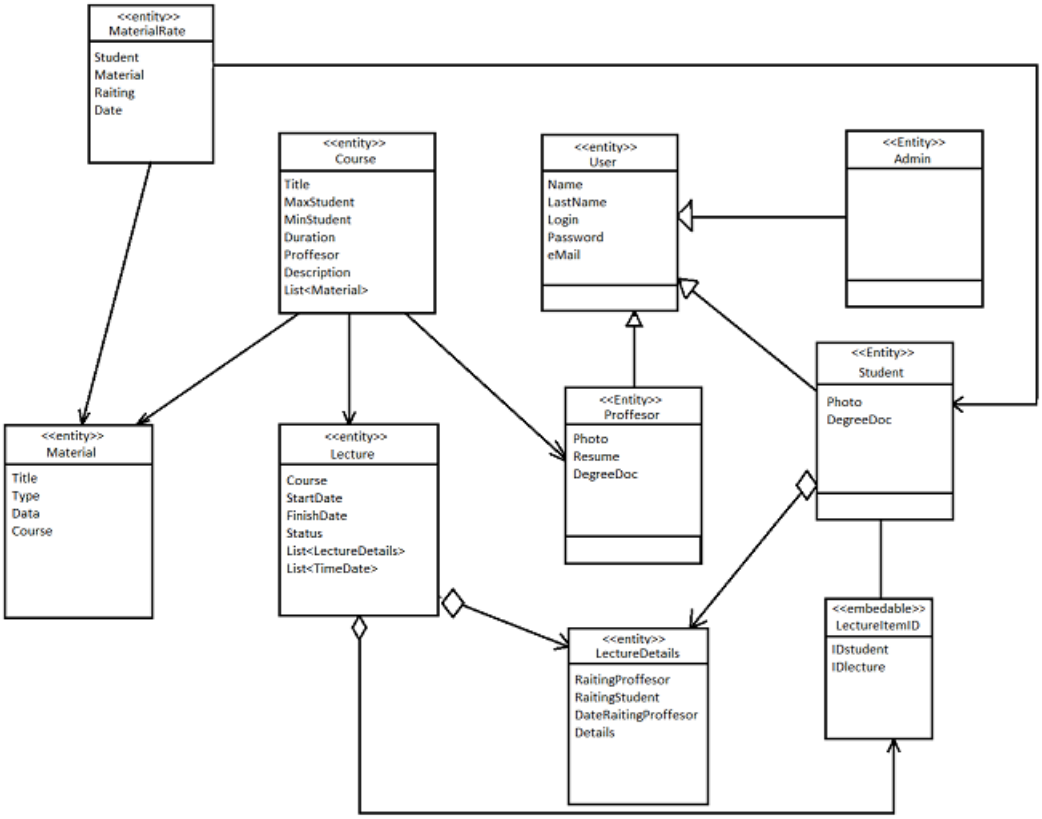

рисунок 3.1.1 Структура класів папки Model

Папка Util слугує нам збереженням допоміжних класів, в ній зберігаються Enum-и, класи-конструктори для збору полів і подальшого використання також класи для зчитування “.properties” файлів.

Enum:

Клас Enum **Status** – зберігає для класу Lecture з папки model три види статусу конкретного курсу: Finish – курс закінчився, Preparation – курс в стадії підготовки, на такий курс є можливість записатися студенту та Active – означає що даний курс вже розпочато.

Клас Enum **MediaType** – зберігає для класу Material папки model – тип матеріалу який завантажується, наприклад при завантажені фото тип матеріалу буде “PHOTO”.

Клас Enum **CommandEnum** – даний клас потрібен нам для переміщень між сторінками веб, у ньому створені назви команд які потрібно створити.

Наприклад enum “LOGIN” викликає command = new LoginCommand; LoginCommand – це клас що знаходиться у папці **Controllers**, в ньому розроблена команда що потрібно зробити.

Клас для зчитування файлу .properties:

ConfigMananger – має метод getProperties() що повертає зчитані ним дані, самі ж дані зчитуються при створені самого класу методом .load що викликається на properties. Має статичний конструктор та одне приватне поле типу Properties. Зчитує він посилання на .jsp класи сторінок, тому сам файл properties має назву navigation.properties.

Класи конструктори:

Такі класи необхідні для того щоб ми могли виймати різні дані з різних класів та збирати їх до кучі. Наприклад коли нам потрібно дістати курси які вже завершились, при цьому нам необхідні поля – це назва курсу, дата його початку, кінця та рейтинг цього курсу – ці поля зберігаються у трьох різних об’єктах і для того щоб не створювати їх та не виймати багато різної не потрібної інформації – у нас є клас ExamingItem у якого є безліч різних полів з безліч необхідними конструкторами.

Для цього також є клас Grade – він використовується для запису та виймання оцінок та рейтингу.

Папка DAO зберігає класи які необхідні для роботи з базою даних.

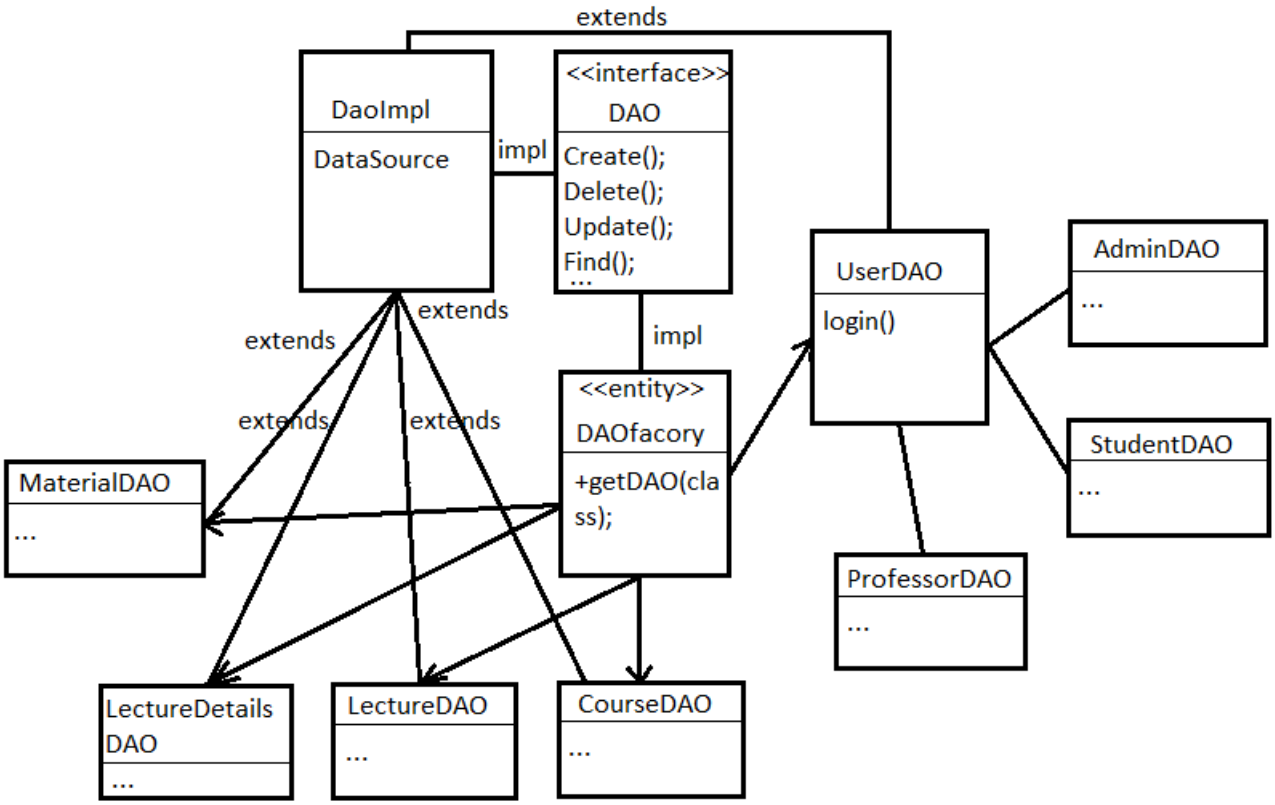

рисунок 3.1.2 Структура DAO

Інтерфейс DAO – має в собі нереалізовані стандартні методи.

Клас DAOImpl – імплементить інтерфейс DAO, зберігає в собі драйвер DataSource.

Клас DAOfactory – має в собі внутрішній статичний клас JDBCdaoFactory який зчитує дані з файлу .properties та створює connector, тобто зв’язок з базою даних. Реалізований метод getDAO – приймає тип об’єкта класу dao якого ми вокликаємо, та створює необхідний DAOclass. Наприклад при передачі User.class створиться UserDAO. Сам клас створюється у єдиному варіанті.

Класи: LectureDAO, UserDAO, CourseDAO… - екстендять клас DAO передаючи йому об’єкт класу, реалізують в собі логіку стандартних та розроблених для конкретного класу методів. В цих методах ми створюємо запит до бази даних та записуємо дані що повертаються.

Папка Controllers зберігає в собі класи команд необхідних для роботи html запитів. Розглянемо діаграму, щоб краще розуміти для чого нам потрібні команди, та взагалі логіку програми взаємодії з сервером:

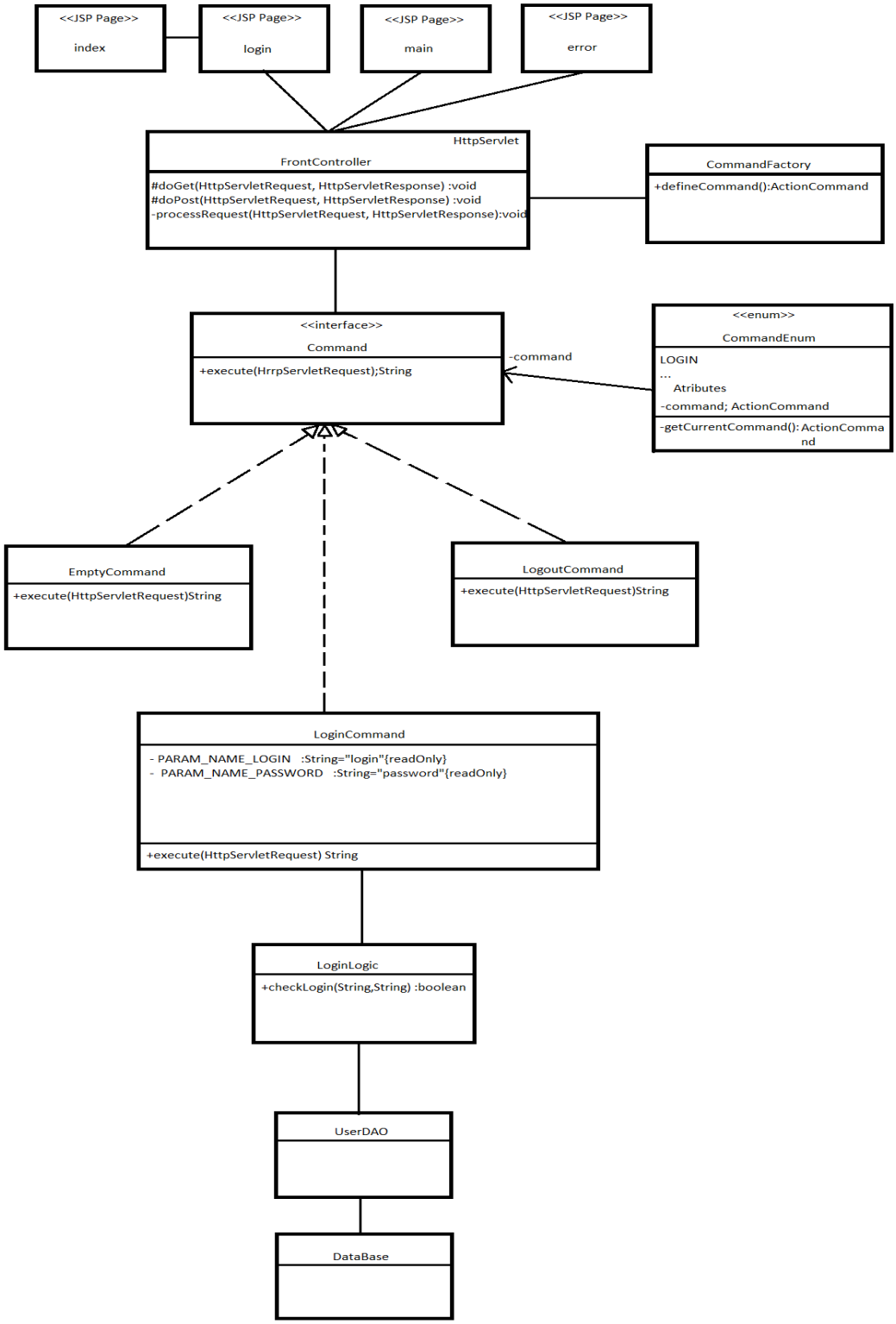

рисунок 3.1.3 Структура JSP

Така архітектура має назву MVC(Model/View/Controller):

Model – класи збереження.

View – .jsp сторінки.

Controller – сервлет.

Для виявлення типу та створення екземплярів команд використовується елемент реалізації шаблону Factory Method класу Command. По закінченню виконання команди, отриманні дані передаємо в request для подальшої передачі користувачу.

Клас LoginLogic – перевіряє правильність введених даних користувачем, він звіряє дані з даними бази даних, і видає команду в залежності від отриманого результату.

Команда EmptyCommand – виконується якщо при запиті до сервлету команда не передавалась.

3.2. Таблиці бази даних.

Для збереження необхідних нам даних, потрібно створити місце-таблиці для цих даних, також необхідно провести зв’язки між ними, для того щоб мати можливість об’єднувати їх в запитах.

Структура створеної бази даних:

3.1 Таблиця: Структура БД

Таблиця	Поле	Тип	Ключі	Обмеження
Users	ID	BigInt	PK	AI, Not null
	Type	Varchar(45)		Not null
	Name	Varchar(45)		Not null
	LastName	Varchar(45)		
	Login	Varchar(45)		Not null
	Password	Varchar(45)		Not null
	eMail	Varchar(45)		Not null
	Photo	LongBlob		
	DegreeDoc	Varchar(100)		
	Resume	Varchar(300)		
Courses	ID	Int	PK	AI, Not null
	Title	Varchar(100)		Not null
	maxStudent	Int		Not null
	minStudent	Int		Not null
	Duration	Int		Not null
	ProffesorID	BigInt	FK	Not null
	Description	Varchar(300)		Not null
Lecture	ID	Int	PK	AI, Not null
	ID_Course	Int	FK	Not null
	StartDate	Date		Not null
	FinishDate	Date		Not null
	Status	Varchar(45)		Not null
	ID	Int	PK	AI, Not null

Materials	ID_Course	Int	FK	Not null
	Title	Varchar(100)		Not null
	Type	Varchar(45)		Not null
	Data	LongBlob		
LectureDetails	ID_Lecture	Int	PK, FK	Not null
	ID_Student	BigInt	PK, FK	Not null
	RaitingProfessor	Double		
	DateRaitingProfessor	Date		
	Details	Varchar(100)		
	RaitingStudent	Double		
MaterialsRate	ID_Material	Int	FK	Not null
	ID_Student	BigInt	FK	Not null
	Rate	Double		Not null
	Date	Date		Not null
TimeTable	Lecture_ID	Int	FK	Not null
	Time	DateTime		Not null

3.3. Робота системи.

Для того щоб зрозуміти як система працює, розглянемо одну команду, та відслідкуємо поетапну роботу виконання запиту. Як приклад візьмемо команду входу вже зареєстрованого користувача.

При вході, користувач знаходиться на сторінці де потрібно ввести логін та пароль, або зареєструватись:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Login page</title>
</head>
<body>
    <form action="front" method="post">
        <input type="hidden" name="command" value="login">
        <table align="center">
            <tr>
                <td>Login</td> <td><input type="text" name="login"></td>
            </tr>
            <tr>
                <td>Password</td> <td><input type="password" name="password"></td>
            </tr>
            <tr>
                <td><label>     </label></td>
            </tr>
            <tr>
                <td><input type="submit" value="OK"></td>
                <td><input type="reset" value="Cansel"></td>
            </tr>
            <tr>
                <td><a href="front?command=registr">Registration</a></td>
            </tr>
        </table>
    </form>>
</body>
```

рисунок 3.3.1 login.jsp

- Це login.jsp сторінка для входу, коли користувач вводить необхідні дані вона записує ці дані та передає їх по команді далі.

FrontController – сервлет, отримує запит, створює команду – передаючи їй тип який потрібно виконати, метод doPost:

```java
@WebServlet(name = "frontController", value = "/front")
public class FrontController extends HttpServlet {


    public void init() {
        getServletContext().setAttribute( name: "daoFactory", DAOfactory.getDAOfactory());

    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        processRequest(request, response);
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        processRequest(request, response);
    }
    private void processRequest(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException{
        Command command = CommandFactory.getCommandFactory().getCommand(request);
        String page = command.execute(request);
        request.getRequestDispatcher(page).forward(request, response);
    }

    public void destroy() { ((DAOfactory)(getServletContext().getAttribute( name: "daoFactory"))).close(); }
    }
```

рисунок 3.3.2 frontController

В конструкторі видно, що при створенні FrontController-а, створюється і DAO – фабрика. В методі doPost викликається метод processRequest, він створює Command викликає CommandFactory та записує до ярлика отриманий результат.

```java
import javax.servlet.http.HttpServletRequest;

public class CommandFactory {
    private static CommandFactory commandFactory = new CommandFactory();

    private CommandFactory() {
    }

    ;

    public static CommandFactory getCommandFactory() {
        return commandFactory;
    }

    public Command getCommand(HttpServletRequest request) {
        Command command = new EmptyCommand();
        String action = request.getParameter( name: "command");
        if (action == null || action.equals("")) {
            return command;
        }
        try {
            CommandEnum commandEnum = CommandEnum.valueOf(action.toUpperCase());
            return commandEnum.getCommand();
        } catch (IllegalArgumentException e){
            return command;
        }
```

рисунок 3.3.3 CommandFacrory

CommandFactory – методом getCommand отримує запит, та перевіряє його назву у Enum-а CommandEnum – який в свою чергу при наявності потрібного enum-у створить необхідну нам команду:

```java
public enum CommandEnum {
    LOGIN{
        {
            command = new LoginCommand();
        }
    },
    STUDENT_AVAILABLE_COURSES{
        {
            command = new StudentAvailableCoursesCommand();
        }
    },
    MYCOURSES{
```

рисунок 3.3.4 CommandEnum

*Команда яка викликається у прикладі виділена.

Enum створює клас LoginCommand та записує результат що повертається у ярлик command.

Сам клас LoginCommand збирає всю необхідну нам інформацію з бази даних.

```
10  import com.univercity.Courses.model.User;
11
12  import javax.servlet.http.HttpServletRequest;
13  import javax.servlet.http.HttpSession;
14
15  public class LoginCommand implements Command{
16
17      public String execute(HttpServletRequest request){
18          String login = request.getParameter( name: "login");
19          String password = request.getParameter( name: "password");
20          DAOfactory daOfactory = (DAOfactory) request.getServletContext().getAttribute( name: "daoFactory");
21          UserDAO userDAO = (UserDAO) daOfactory.getDAO(User.class);
22          User user = userDAO.login(login, password);
23          HttpSession session = request.getSession( create: true);
24          session.setAttribute( name: "user", user);
25          if (user instanceof Admin){
26              return ConfigMananger.getProperty("admin");
27          }
28          if (user instanceof Professor){
29              return ConfigMananger.getProperty("professor");
30          }
31          if (user instanceof Student){
32              CommandEnum.STUDENT_AVAILABLE_COURSES.getCommand().execute(request);
33              return ConfigMananger.getProperty("student");
34          }
35          return ConfigMananger.getProperty("login");
36      }
```

рисунок 3.3.5 LoginComamnd

Дістаємо з request(запиту) необхідні дані, під тими ярликами якими ми записали на login.jsp – сторінці.

Після отримання результатів, перевіряємо ярлик класу на тип користувача, і в залежності від того ми повертаємо посилання на сторінку через ConfigMananger

ConfigMananger – клас що зчитує з файлу navigation.properties посилання на сторінки, тобто слугує нам вказівником.

```java
package com.univercity.Courses.Util;

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

public class ConfigMananger {
    private static Properties properties;

    static {
        properties = new Properties();
        try {
            properties.load(ConfigMananger.class.getClassLoader().getResourceAsStream( name: "/navigation.properties"));
        }   catch (IOException e){
            e.printStackTrace();
        }
    }
    public static String getProperty(String name) { return properties.getProperty(name); }
}
```

рисунок 3.3.6 ConfigMananger

```
login=login.jsp
admin=jsp/admin/admin.jsp
professor=jsp/professor/professor.jsp
student=jsp/student/student.jsp
course=jsp/course/course.jsp
mycourses=jsp/student/mycourses.jsp
```

рисунок 3.3.7 vavigation.properties

Створюємо необхідну нам DAO, у нашому випадку – UserDAO, та викликаємо раніше розроблений метод .login куди ми передаємо змінні “login” та “password”

```java
private static class JDBCdaoFactory extends DAOfactory{
    private BasicDataSource dataSource;

        public JDBCdaoFactory() {
            try {

            Properties properties = new Properties();
            properties.load(DAOfactory.class.getClassLoader().getResourceAsStream( name: "storege.properties"));

                String url = properties.getProperty("database_url");
                String password = properties.getProperty("password");
                String user = properties.getProperty("user");
                String driver = properties.getProperty("driver_class");
                int poolSize = Integer.parseInt(properties.getProperty("initional_pool_size"));
                dataSource = new BasicDataSource();
                dataSource.setInitialSize(poolSize);
                dataSource.setUrl(url);
                dataSource.setPassword(password);
                dataSource.setUsername(user);
                dataSource.setDriverClassName(driver);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

рисунок 3.3.8 DAOfactory

DAO-фабрика зчитує з файлу storage.properties дані які необхідні для встановлення зв’язку з базою даних.

Далі метод getDAO() перевіряє тип класу який ми передаємо після чого створює необхідну DAO

```
public <E extends Entity> DAO<E> getDAO(Class<E> clazz){
    try {
        E entity = clazz.newInstance();
        if (entity instanceof Professor) {
            return (DAO<E>)new ProfessorDAO(dataSource);
        }
        if (entity instanceof Student) {
            return (DAO<E>)new StudentDAO(dataSource);
        }
        if (entity instanceof Admin) {
            return (DAO<E>)new AdminDAO(dataSource);
        }
        if (entity instanceof Course) {
            return (DAO<E>)new CourseDAO(dataSource);
        }
        if (entity instanceof Material){
            return (DAO<E>)new MaterialDAO(dataSource);
        }
        if (entity instanceof Lecture) {
            return (DAO<E>)new LectureDAO(dataSource);
        }
        if (entity instanceof LectureDetails){
            return (DAO<E>)new LectureDetailsDAO(dataSource);
        }
        if (entity instanceof User) {
            return (DAO<E>)new UserDAO(dataSource);
        }
    } catch (Exception e) {
```

рисунок 3.3.9 DAOfactory

На отриманій UserDAO – викликаємо метод .login, який містить в собі запит до бази даних:

```java
public User login(String login, String password){
    String quary = "select * from users where Login  = ? and Password = ?";
    try (Connection connection = dataSource.getConnection();
            PreparedStatement preparedStatement = connection.prepareStatement(quary)){
        preparedStatement.setString( parameterIndex: 1, login);
        preparedStatement.setString( parameterIndex: 2, password);
        ResultSet resultSet = preparedStatement.executeQuery();
        User user = null;
        if (resultSet.next()) {
            long id = resultSet.getLong( columnIndex: 1);
            String type = resultSet.getString( columnIndex: 2);
            String name = resultSet.getString( columnIndex: 3);
            String lastName = resultSet.getString( columnIndex: 4);
            String eMail = resultSet.getString( columnIndex: 7);
            if (type.equalsIgnoreCase( anotherString: "Student")) {
                byte[] photo = resultSet.getBytes( columnIndex: 8);
                String degreeDoc = resultSet.getString( columnIndex: 9);
                user = new Student(photo, degreeDoc, name, lastName, login, password, eMail, id);
            }
            else if (type.equalsIgnoreCase( anotherString: "Professor")) {
                byte[] photo = resultSet.getBytes( columnIndex: 8);
                String degreeDoc = resultSet.getString( columnIndex: 9);
                String resume = resultSet.getString( columnIndex: 10);
                user = new Professor(photo, resume, degreeDoc, name, lastName, login, password, eMail, id);
            }
            else {
                user = new Admin(name, lastName, login, password, eMail, id);
            }
            return user;
```

рисунок 3.3.10 UserDAO

*Синім виділено звернення до БД.

Додаткові файли .jar:

Для того щоб мати зв’язок з базою даних та працювати з .jsp класами стандартного набору бібліотек Java нам не вистачить, тому для написання коду було завантажено наступні файли:

Сам сервер TomCat версії 9.0.24

Commons-pool – для створення декількох зв’язок з БД

Mysql-connector – для створення зв’язку з БД

Javax.servet-api – для зв’язку з веб сторінками

Commons-dbcp – для створення запитів до БД

Повний список завантажених файлів можна переглянути у додатку 1.

3.4. Результат роботи програми.

Для користувача, описані попередньо дії виглядають так:

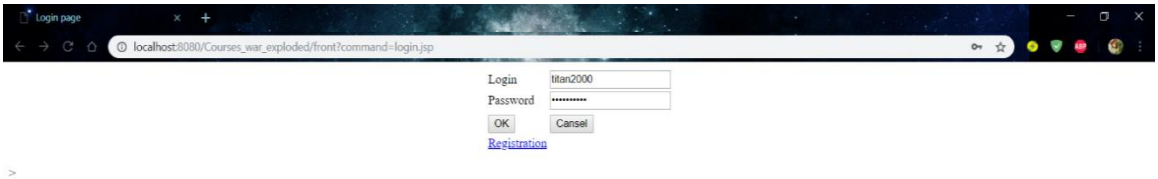

рисунок 3.4.1 Сторінка входу

Сторінка входу, де є поля авторизації: Login та Password, кнопка підтвердження та відміни, а також гіперпосилання на сторінку реєстрації.

Після вводу даних користувача, який в системі відмічений як студент, ми переходимо на головну сторінку студента – student.jsp.

рисунок 3.4.2 Головна сторінка

Головна сторінка звідки нам виводиться основна інформація, та є доступ до інших вкладок.

Зверху зліва ми бачимо три блока: 1 – Active – показує курси які зараз проходять. 2 – Preparation – показує курси які формуються, тут виводяться тільки ті курси, на які студент не записаний. 3 – logout – кнопка яка дозволяє вийти з системи, відкриває сторінку логіна.

Зверху справа у нас чотири гіперпосилання: 1- Profile – відкриває профіль користувача, відображає інформацію про нього. 2 – My courses – відкриває активні та в стадії підготовки курси на які користувач вже записався. 3 – History – дозволяє переглянути користувачу його курси що вже закінчились. 4 – Time table – дозволяє переглянути розклад з xx/xx/xx по xx/xx/xx даті.

Відображення курсів – спочатку йде назва, з гіперпосиланням на сторінку цього курсу, потім дата початку та кінця та оцінка цього курсу, яка вираховується середньою між оцінками які ставили попередні студенти цього курсу.

ВИСНОВКИ

В процесі виконання даної роботи було розроблено архітектуру веб-додатку, написано логіку на мові Java, створено веб-сторінки для відображення та взаємодії користувача з додатком, спроектовано та розроблено базу даних на мові MySQL. В результаті ми отримали готову програму яка може виводити потрібні дані з БД та обробляти їх. Сам додаток працює на сервері TomCat – під нього розроблені сервлети. Має базовий необхідний функціонал, та може легко оновлюватись.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Java. методы программирования: уч.-мет. Пособие / И.Н. Блинов, В.С. Романчик – Минск: Издательство «Четыре четверти».2013 - 896с.
2. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих: Бхаргава А. 2016-288с.
3. Java Programming Forums [Електронний ресурс] - Режим доступу: The Java Community - https://www.javaprogrammingforums.com/
4. StackOverflow [Електронний ресурс] - Режим доступу: https://stackoverflow.com/
5. Java T point [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/
6. Apache Maven [Електронний ресурс] – Режим доступу: https://maven.apache.org/
7. SQL CookBook: Anthony Molinaro: publishing “O`REILLY”. 2009-672с.
8. Apache software foundation [Електронний ресурс] – Режим доступу: https://www.apache.org/
9. Betacode [Електронний ресурс] – Режим доступу: https://betacode.net/10429/java-jsp-standard-tag-library
10. TED [Електронний ресурс] – Режим доступу: https://www.ted.com/
11. OldCode [Електронний ресурс] – Режим доступу: http://old.code.mu/sql
12. GitHub [Електронний ресурс] – Режим доступу: https://github.com/
13. GuruWeba [Електронний ресурс] – Режим доступу: https://guruweba.com/html/
14. CourSera [Електронний ресурс] – Режим доступу: https://www.coursera.org/
15. edX [Електронний ресурс] – Режим доступу: https://www.edx.org/course/introduction-to-project-management

16. CodeAcademy [Електронний ресурс] – Режим доступу: https://www.codecademy.com/learn/paths/master-statistics-with-python
17. Java Cookbook: Problems and Solutions for Java Developers 4th Edition: Ian F. Darwin: Publisher “O'Reilly Media”. 2020-638c.

ДОДАТОК 1. Файл завантажень 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>

    <groupId>com.univercity</groupId>
    <artifactId>Courses</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>Courses</name>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.source>1.8</maven.compiler.source>
        <junit.version>5.7.1</junit.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.21</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.2.2</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/commons-pool/commons-pool -->
        <dependency>
            <groupId>commons-pool</groupId>
            <artifactId>commons-pool</artifactId>
            <version>1.4</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
```

```
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
            <scope>provided</scope>
        </dependency>



    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
            </plugin>
        </plugins>
    </build>
</project>
```