МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Веб-система для ведення Електронного журналу успішності учнів навчального закладу»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Тиркусова Н.В.**

Студента групи ІН – 64 — **Почкун А.В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 г.

ЗАВДАННЯ
до випускної роботи

Студентки четвертого курсу, групи ІН-64 спеціальності "Інформатика" денної форми навчання Почкун Анастасії Вікторівни.

Тема: "**Веб-система для ведення Електронного журналу успішності учнів навчального закладу**"

Затверджена наказом по СумДУ

№_____ от _____2020 г.

Зміст пояснювальної записки: 1) Інофмаційний огляд; 2) вибір методу рішення; 3) програмна реалізація

Дата видачі завдання "_____"_____2020 г.

Керівник випускної роботи _____ Тиркусова Н.В.

Завдання приняв до виконання _____ Почкун А.В.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	02.03.20 – 05.03.20	
2	Пошук аналогів та обгрунтування потреби	06.03.20 – 10.03.20	
3	Визначення вимог	11.03.20 – 11.03.20	
4	Визначення інструментарію	12.03.20 – 12.03.20	
5	Аналіз сервісів Google	13.03.20 – 13.03.20	
6	Складання календарного плану	16.03.20 – 17.03.20	
7	Визначення ризиків	18.03.20 – 19.03.20	
8	Розробка інтерфейсу користувача	20.03.20 – 26.03.20	
9	Проектування бази даних	27.03.20 – 03.04.20	
10	Розробка програмних модулів	06.04.20 – 30.04.20	
11	Компілювання програмного коду	01.05.20 – 11.05.20	
12	Тестування розробником	12.05.20 – 14.05.20	
13	Тестування незалежною особою	15.05.20 – 19.05.20	
14	Оформлення документації	11.03.20 – 21.05.20	
15	Архівація проекту	22.05.20 – 25.05.20	
16	Введення в експлуатацію	26.05.20 – 31.05.20	

Студент _____ Почкун А.В.
(підпис)

Керівник роботи _____ Тиркусова Н.В.
(підпис)

РЕФЕРАТ

Записка: 66 стор., 22 рис., 1 табл., 5 додатків, 7 джерел.

Об'єкт дослідження — навчальний процес зі складнощами обміну електронними ресурсами між викладачами та студентами.

Мета роботи — розробка веб-додатку для зручного перегляду студентами навчальних матеріалів.

Методи дослідження — в процесі виконання програмної реалізації було застосовано комбінацію технологій для побудови сайтів мовами Java, HTML, CSS, JavaScript, MySQL.

Результати — розроблено інформаційну систему, що здатна забезпечити зручне ведення журналу з оцінками. Також веб-додаток не обмежує у форматах файлів, тож публікувати можна буде не тільки таблиці журналу, а й текстові документи, відео, презентації, архіви та навіть вкладені каталоги.

НАВЧАЛЬНИЙ ПРОЦЕС, ОБМІН ЕЛЕКТОННИМИ РЕСУРСАМИ, ЖУРНАЛ З ОЦІНКАМИ, ЗРУЧНИЙ ДОСТУП ДО ФАЙЛІВ, МОВА ПРОГРАМУВАННЯ JAVA, ОБЛІК УСПІШНОСТІ.

ЗМІСТ

ВСТУП

Новим освітнім досвідом, який з'явився в багатьох країнах світу останнім часом, є дистанційне навчання. Людство має гостру необхідність у такому методі навчання, особливо в умовах карантину та самоізоляції. Для того, щоб зробити дистанційне навчання можливим, має бути якісне програмне забезпечення, що має широкий спектр функцій і доступні умови користування.

Дедалі більше закладів освіти впроваджує таку форму навчання у власну систему. Навчальний процес у вищих навчальних закладах може мати складнощі управління і користування електронними ресурсами, опублікованими на різних сайтах. Зазвичай, доступ до різних ресурсів потребує створення окремих облікових записів.

Студенту часто доводиться користуватись декількома сайтами лише для того, щоб на одному прочитати лекцію, а на іншому подивитись свої оцінки. Викладач публікує файли і користується додатковим функціоналом системи, який часто має недоліки та обмеження. Такими обмеженнями можуть бути вузькі права користувача відносно своїх публікацій, перевищення допустимого розміру файлу, некоректний формат файлу, замалий об’єм виділеної пам’яті для зберігання навчальних матеріалів.

Головною ідеєю проекту є створення програмного продукту, який частково спростить навчальний процес в аспектах користування електронними ресурсами.

Проект є актуальним тому, що електронний журнал – це базова потреба будь-якого освітнього закладу, в якому запроваджується електронний варіант обліку успішності та обміну навчальними матеріалами, практикується дистанційна форма навчання.

За своїм призначенням журнал може бути індивідуальною розробкою університету, безпечним і зручним інструментом у період навчання для студентів та викладачів.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Огляд існуючих рішень

Головне призначення інформаційної системи - зручний обмін електронними навчальними ресурсами, а зокрема, таблицями з оцінками. Найбільш зручними в цьому випадку є таблиці Excel. Нині існує можливість переглядати та редагувати їх у браузері, не встановлюючи програму MS Excel на комп’ютері.

Загалом, для студентів навчальними ресурсами є:

- Лекції;
- Таблиці з оцінками та пропусками занять;
- Завдання до практичних і лабораторних робіт;
- Методичні рекомендації для оформлення звітів;
- Архіви рекомендованих бібліотек для написання коду;
- Додаткова тематична література;
- Відео та аудіо-уроки.

Тому, дослідження проводилось виключно серед таких сайтів, які мають функціонал обміну файлами і схоже призначення.

Сервіс Google Classroom

Популярним рішенням для забезпечення дистанційного навчання є Google Classroom. Цей програмний продукт користується широким застосуванням в багатьох навчальних закладах різних рівнів акредитації. Відповідно до потреб навчального закладу, адміністратор може обрати певний тариф.

Google Classroom надає користувачу такі можливості:

- Створення класу / курсу;
- Запис учнів на курс та надання завдань;
- Публікація навчального матеріалу;
- Оцінювання розв’язків задач, облік успішності учнів;
- Інтерактивне спілкування учнів.

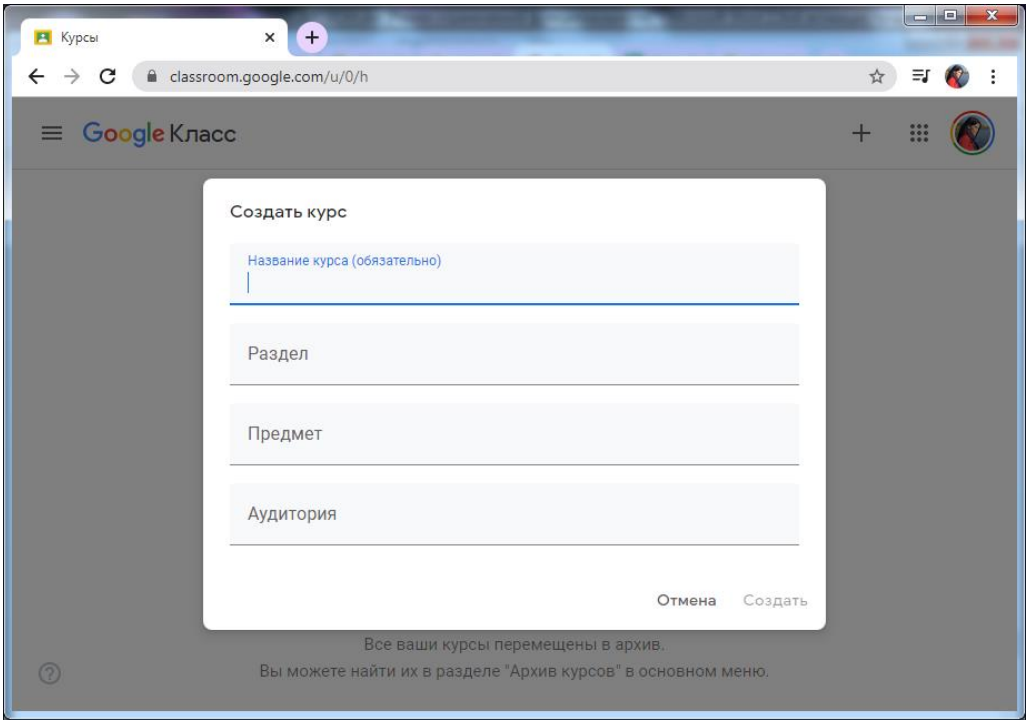

Рисунок 1.1 – Скріншот форми створення курсу в Google Classroom

Недоліки сервісу Google Classroom:

- Немає вебінарної кімнати. Але цю проблему вирішити нескладно: для цього можна користуватись YouTube або Google Hangouts, які дозволяють провести онлайн-зустріч.
- Безкоштовна версія не містить у собі електронний журнал для обліку успішності учнів. Така можливість відкрита корпоративним користувачам Google Classroom.
- Для авторів, які мають особисті акаунти, існують обмеження: загальна кількість учасників курсу може бути не більше, ніж 250, а приєднатися до курсу за один день можуть всього 100 користувачів.

Останній недолік є суттєвим для вищих навчальних закладів, які вже активно запроваджують дистанційний обмін навчальними ресурсами зі студентами. У випадку, коли необхідно поширити загальну методичну чи організаційну інформацію серед великої кількості студентів, необхідно буде дублювати її у кожному курсі, або взагалі обрати іншу платформу для публікації.

Огляд e-schools.info

Сервіс e-schools.info призначений для обробки і надання в зручному електронному вигляді інформації про успішність учнів і додаткових навчальних ресурсів. Облік успішності полягає у збереженні та аналізі оцінок і пропусків занять, фіксуванні коментарів до них, нотатків до уроків, домашніх завдань, зауважень, розкладів, занять і багато іншого.

У e-schools.info є 5 основних типів користувачів сайту: директор, адміністрація, вчитель, учень, батьки. Кожен з них має власні набір прав і рівень доступу до інформації.

Рисунок 1.2 – Скріншот головної сторінки e-schools.info

Проаналізовані системи мають широкий функціонал і можуть бути ефективним інструментом для дистанційного навчання. Проте, іноді, коли є чітко сформульована потреба, різноманіття додаткових функцій може заплутувати користувача. Часто, навіть, безкоштовні сервіси пропонують додаткові функції, але неповністю розкривають основне призначення, накладають обмеження на найпотрібніше.

Сервіс Google Drive

Ідея сервісу полягає у віддаленому зберіганні файлів різних форматів та розмірів. Доступ до файлів на каталогів на гугл-диску може бути як приватним, так і публічним. Більшість налаштувань можна встановлювати власноруч і, в разі відкриття публічного доступу, файл за посиланням можуть переглядати, коментувати або редагувати інші користувачі, список яких також можна регулювати.

Таким чином, файлами безлічі різних типів можна безпечно користуватись у режимі онлайн, не завантажуючи їх на свій пристрій.

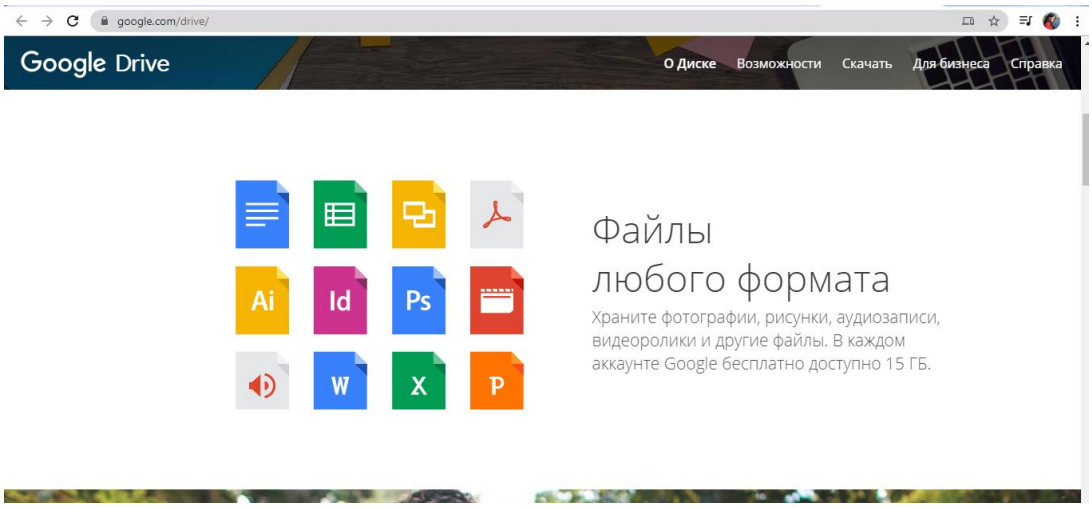

Рисунок 1.3 – Скріншот головної сторінки Google Drive

Проте, Google Drive не здатний реалізувати всі потреби системи обміну навчальними ресурсами між студентами та викладачами. Немає можливості групувати користувачів та надавати ролі «Студент» та «Викладач». Зручно було б надавати доступ до файлів не кожному студентові окремо, а одразу всій академічній групі. Така можливість також відсутня.

У результаті дослідження було підтверджено, що доречним буде створення самостійної системи, частково інтегрованої з деякими сервісами компанії Google. Новий продукт має бути зручним і простим у користуванні як для студентів, так і для викладачів.

1.2 Постановка задачі

Метою роботи є створення зручного варіанту обміну файлами між студентами і викладачами вищого навчального закладу.

Основні вимоги до проекту:

- Реалізація у вигляді сайту;
- Авторизація користувача за допомогою акаунту у Google;
- Безпечне зберігання файлів;
- Користувачі повинні мати, принаймні, одну роль з переліку: студент, викладач, адміністратор;

Під час авторизації необхідно робити додатковий запит до сервісу Google Account, щоб отримати базову інформацію, а саме: e-mail та повне ім’я користувача. Додаткова інформація (номер групи, якщо це студент, список доступних файлів та інше) буде зберігатись у базі даних.

У випадку, коли користувач ще незареєстрований, треба запропонувати йому залишити основні дані про себе і, за бажанням, коментар, які потім оброблятиме адміністратор сайту. Він повинен схвалити або видалити запит. Самостійно користувач не реєструється.

В особистому кабінеті студента необхідно розмістити список посилань на усі файли, доступ до яких має його група. Посилання повинні групуватися за автором, тобто викладачем.

Функціонал ролі «викладач»:

- Управління групами студентів: створення, зміна та видалення групи;
- Управління навчальними ресурсами: завантаження, створення, копіювання, редагування, видалення файлів і каталогів;
- Можливість групування файлів у каталоги;
- Перегляд списків і вмісту файлів у браузері,
- Управління доступом до файлів на рівні «каталог – група студентів»: відкриття доступу до каталогу має відбуватись для всіх студентів певної групи, видалення раніше відкритих доступів;

- У особистому кабінеті має бути список каталогів, доступом до яких він може керувати, а також посилання на місце зберігання файлів.

Роль адміністратора також має володіти цим функціоналом. Відмінність від викладача полягає у можливості управління користувачами та групами, обробці нових запитів на реєстрацію. Тільки адміністратор матиме право на створення кореневого каталогу викладача, у який той, надалі, буде завантажувати файли.

Основні операції за групами, користувачами і запитами, які мають бути реалізовані:

- Створення нового елементу;
- Зміна;
- Видалення;
- Отримання списку.

Адміністратор повинен бути власником проекту у Google Cloud Platform, мати змогу слідкувати за активністю користувачів та статистикою їхніх запитів, у панелі управління проектом.

2 ВИБІР МЕТОДУ РІШЕННЯ

Під час проектування інформаційної системи не варто ігнорувати існуючі інструменти для того, щоб реалізувати певну функцію. Іноді готове рішення краще, ніж власний аналог, створений з нуля. Звісно, не завжди це можливо, з огляду на: умови використання готового рішення, постановку задачі, побажань замовника щодо способів реалізації тощо.

2.1 Застосування Google Account

Вдалим рішенням є застосування поширеного і безпечного вид авторизації користувача у системі – вхід через акаунт у Google.

Певною незручністю може бути те, що під час першого візиту користувачеві необхідно залишити заявку на підтвердження свого статусу студента чи викладача (на вибір). Оскільки сайт призначений для окремого навчального закладу, його аудиторія може бути чітко визначеною шляхом звіряння зі списками студентів. Цей етап є необхідним для того, щоб «познайомити» систему з користувачем, визначити його рівень доступу або відхилити його запит. Цю інформацію зазвичай обробляє адміністратор і, в разі схвалення, надалі користувачеві для входу потрібно буде просто обрати та підтвердити свою пошту.

2.2 Застосування Google Drive і Google Cloud Platform

Зручним і безпечним способом публікації навчальних ресурсів є розміщення їх на Google Drive. Для того, щоб отримати окремий диск, необхідно створити новий гугл-акаунт. Завдяки ньому буде відкрито доступ до більшості сервісів Google. Крім цього, акаунт буде мати повні права і, фактично, буде адміністратором усього проекту. Окремий сервіс Google, який має назву Cloud Platform, забезпечує зв’язок сайту з усіма іншими сервісами, робить можливою авторизацію за допомогою гугл-акаунту, забезпечує моніторинг активності користувачів сайту та надає розробнику доступ до документації.

Призначення Google Cloud Platform полягає у створенні віртуального проекту, що є прототипом розроблюваного сайту. У рамках цього проекту можна створити користувачів з різними рівнями доступу, отримувати API-ключі, тестувати генерацію тимчасових токенів, наприклад, для отримання списку файлів на гугл-диску.

Для того, щоб користувач бачив певний набір файлів, необхідно додатково створити проміжний інтерфейс, в якому дані будуть підібрані і відсортовані згідно правилам, встановленими викладачами. Таким інтерфейсом є сайт журналу.

Викладач матиме повні права відносно своїх публікацій. Для того, щоб студенти могли переглядати файли, викладач повинен спочатку встановити відкритий доступ до каталогу у налаштуваннях на диску. Після цього, на сайті журналу потрібно буде ще встановити додаткові налаштування, а саме відкрити доступ до каталогу певній групі студентів. Список груп зберігатиметься у базі. Ці налаштування будуть більш зручними для викладачів, ніж надання доступу кожному користувачеві окремо.

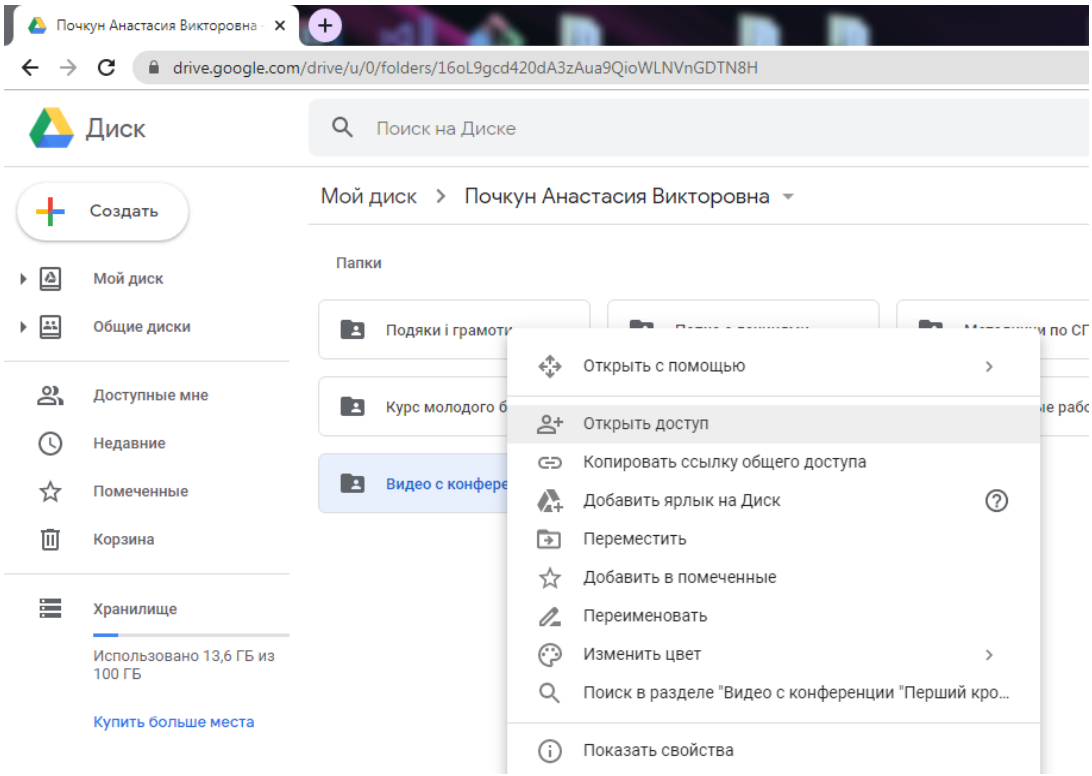

Рисунок 2.1 – Скріншот каталогів на гугл-диску

Відсутність обмеження доступу на рівні диску є об’єктивним рішенням, оскільки іноді студенти використовують посилання на лекційні матеріали у своїх практичних роботах, або просто користуються і діляться посиланнями між собою, з батьками, не заходячи до особистого кабінету.

2.3 Застосування Google Drive API

Доступ до розміщених публікацій можна отримувати за допомогою технології Google Drive API. Цей сервіс дозволяє різним додаткам отримувати, записувати і синхронізувати файли на гугл-диску.

Доступ до каталогів на диску буде зручніше відкривати не окремо кожному студентові, а всій академічній групі. Обмеження мають встановлювати на рівні інтерфейсу сайту, на гугл диску всі папки повинні бути відкриті всім користувачам.

2.4 Вибір середовища розробки

Враховуючи сучасні тенденції розвитку програмування і рівень можливостей інтеграції з сервісами Google, було вирішено обрати для розробки мову Java і застосувати технологію Java Servlet.

Java є об’єктно-орієнтованою мовою програмування, за допомогою якої програмісти створюють різноманітні прикладні додатки для комп’ютерів, смартфонів, планшетів тощо. На об’єктну природу мови вказує те, що всі дані і дії групуються в класи об'єктів. У Java всі об'єкти є похідними від головного об'єкта (він називається просто Object), з якого вони успадковують базову поведінку і властивості. Виключенням з повної об'єктності є примітивні типи (int, float тощо). Це було свідомим рішенням проектувальників мови задля збільшення швидкодії додатків. Тому Java не вважається повністю об'єктно-орієнтовною мовою.

Значною перевагою програм, написаних мовою Java, є те, що вони можуть запускатись на будь-яких комп’ютеризованих пристроях, працюють на базі різних операційних систем та, навіть, без повторної компіляції коду.

Синтаксис цієї мови програмування схожий на звичайну англійську мову. В ній є мінімальна кількість складних для запам'ятовування символів. Фактично, після встановлення JDK, настройки PATH і вивчення особливостей Classpath спеціаліст вже може створювати елементарні програми на Java.

Приклад простої програми, яка виводить «Hello, World!»:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Період становлення Java співпав з розквітом міжнародної інформаційної служби World Wide Web. Ця обставина зумовила оновлення напрямів розвитку Java, оскільки WWW вимагає використання крос-платформних додатків. В результаті були зміщені акценти з побутової електроніки на програмування для Інтернет. Нині продукти Java можна зустріти скрізь. Завдяки простоті та надійності цю мову використовують для розробки ПЗ в державній сфері, в науці, освіті, сфері охорони здоров'я, в приватному секторі при створенні програм для трейдингу, серверних додатків для банкінгу та для багатьох інших корпоративних і особистих цілей.

Технологія сервлетів використовується для створення веб-додатків. Спочатку на сервері встановлюються платформа Java та Apache Tomcat, який запускає програму і «публікує» її у якості сайту у мережі Інтернет. Після цього сайт реагує на запити користувачів, виконує обробку даних і надсилає у відповідь динамічні веб-сторінки.

Технологія сервлетів надійна і гнучка. Завдяки мові Java вона має багато інтерфейсів і класів, таких як Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse і т.д.

Двома найбільш поширеними типами запитів HTTP (їх також називають методами запитів) є get і post. Запит get отримує (або витягує) інформацію. Запит post поміщає (або відправляє) дані на сервер. Типове застосування

методу post - відправка на сервер інформації для аутентифікації, або даних з форми, в яку користувач ввів інформацію.

Сервлети веб-додатку зазвичай походять відкласу HttpServlet. У ньому визначені методи doGet і doPost для реакції на відповідні запити клієнта. Ці методи приймають в якості параметрів об'єкти HttpServletRequest і HttpServletResponse, які дають можливість здійснювати взаємодію між клієнтом і сервером. Методи інтерфейсу HttpServletRequest надають доступ до даних запиту. Методи інтерфейсу HttpServletResponse забезпечують повернення результатів.

3 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

Базовою рисою проекту є лаконічність, відповідність призначенню і відсутність зайвого функціоналу. При цьому, сайт розрахований, здебільшого, на молоду аудиторію. Тому його верстка має бути адаптивною, динамічною і сучасною.

Функціонал для студента має бути доступним одразу після авторизації, у його особистому кабінеті. Більш широкими є можливості викладача: окрім доступу до своїх каталогів, він буде мати змогу створювати, змінювати і видаляти групи, переглядати та редагувати шаблон списку групи у форматі таблиці Excel, щоб далі мати змогу швидше заповнювати таблиці з оцінками.

3.1 Модулі системи та взаємодія між ними

- Сайт;

Реагуючи на запити користувачів, певні java-сервлети або jsp- сторінки формують відповідь у форматі html. Наприклад, якщо у браузері ввести “http://myjournal.com.ua/” або “http://myjournal.com.ua/welcome.jsp”, результат буде однаковим: користувач побачить головну сторінку сайту. Це відбувається тому, що ці запити обробляє одна jsp-сторінка – welcome.jsp. Налаштування, які вказують на те, які запити обробляються певним сервлетом, встановлюються програмістом у файлі web.xml, який є «серцем» сайту. Лістинг коду web.xml наведено у Додатку А.

- База даних;

На схемі нижче (рис. 3.1), для прикладу, зображено один запит. Він є основним і таким, що підкреслює значення бази у проекті. Насправді, запитів від сайту до бази набагато більша кількість, наприклад: запит на отримання списку груп, запит на вибірку каталогів, до яких викладачі відкрили доступ певній групі тощо.

В адмін. панелі, реалізованій у особистому кабінеті викладача, можуть здійснюватися запити не тільки на вибірку, а ще й на додання, зміну або видалення даних у базі.

- Сервіси Google;

Саме до проекту на Google Cloud здійснює вхід користувач під час авторизації за допомогою облікового запису Google. У випадку успішної авторизації Google Account надає базову інформацію про обліковий запис користувача та токени. Цю інформацію отримує сайт і використовує для подальшої роботи з іншими сервісами. Токени необхідні для доступу до файлів на Google Drive, id облікового запису – для отримання інформації про користувача з бази.

Id облікового запису пов’язує сайт з сервісами, за допомогою нього можна дізнатися про користувача: студент це чи викладач, прізвище та ім’я, іншу додаткову інформацію. Якщо за цим параметром в базі не знайдено даних, замість особистого кабінету користувач бачить форму реєстрації.

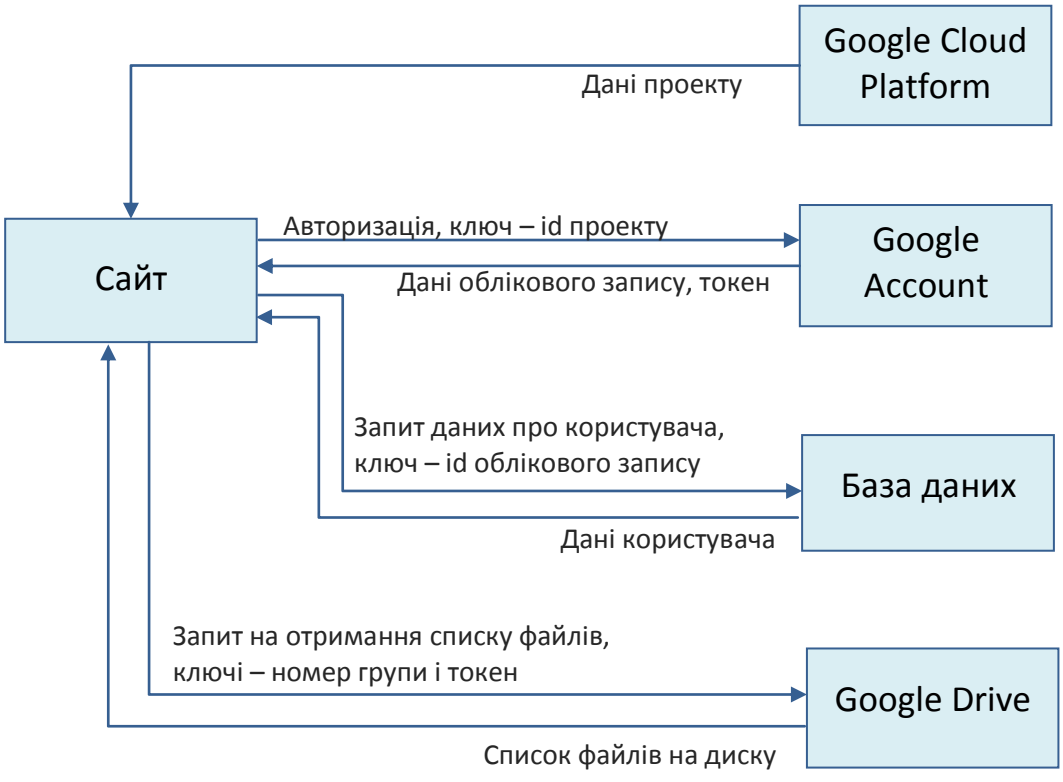

Рисунок 3.1 – Взаємодія модулів системи

3.1 Етапи виконання задачі

1) Створення нового акаунту journal.sumdu@gmail.com і проекту MyJournal у Google Cloud Platform, власником якого є новий акаунт.

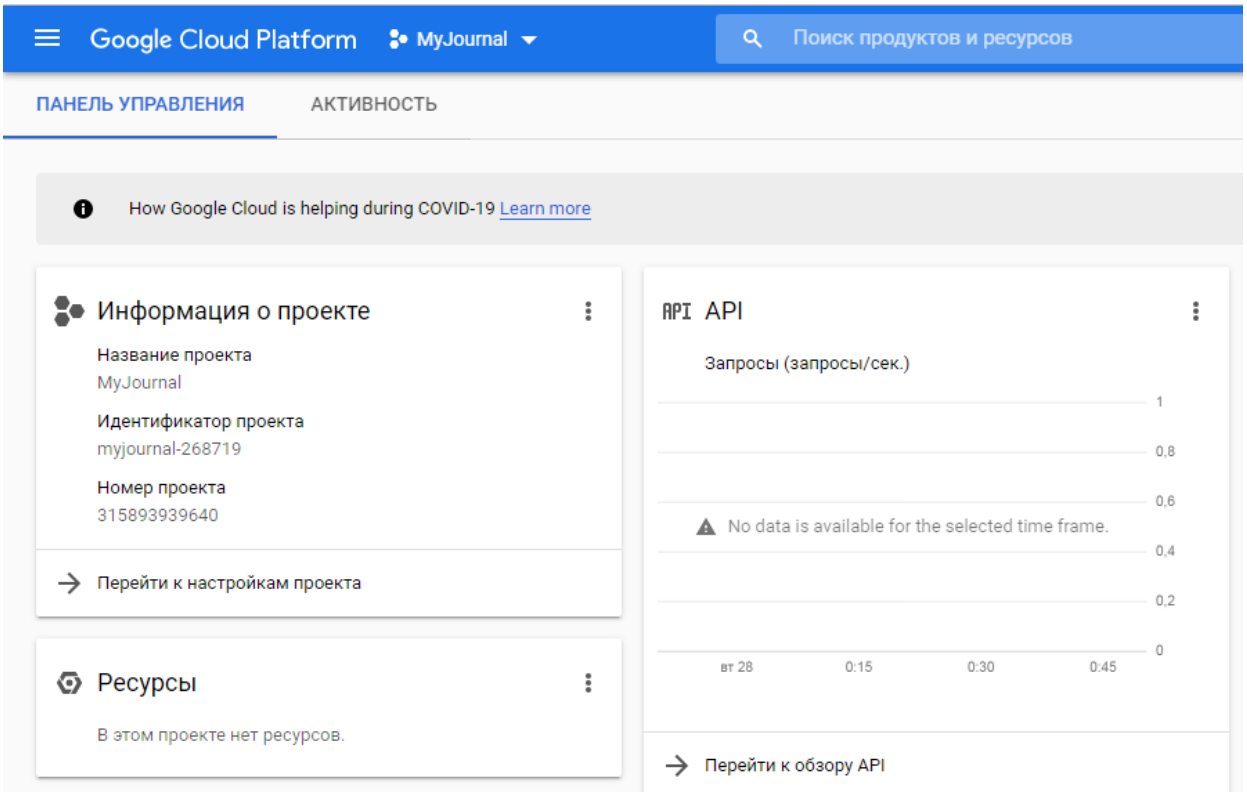

Рисунок 3.2 – Параметри проекту у Google Cloud Platform

2) Проектування структури бази даних.

Основними об’єктами системи є студент, викладач, налаштування і файл. На основі цього твердження можна скласти DFD 0-го рівня, що зображує основний процес у системі і є першим кроком у розробці будь-якої бази. База повинна бути такою, що здатна реалізувати цей процес.

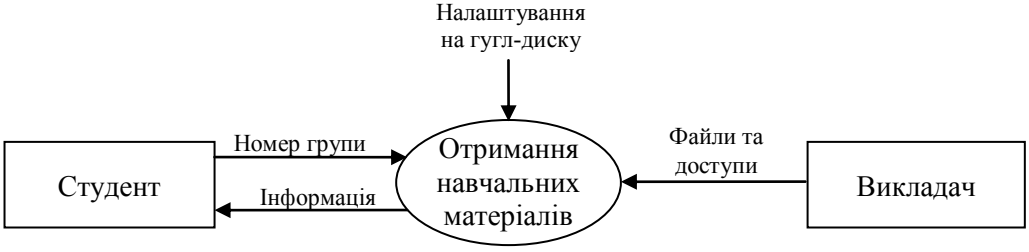

Рисунок 3.3 - DFD 0-го рівня

Більш інформативною є ER-діаграма – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. Дані представлені у вигляді компонентів (сутностей), пов'язаних між собою певними зв'язками, що виражають залежності між ними.

Сутності та зв'язки можуть мати свої атрибути. Наприклад, сутність «користувач» має атрибут «ім’я», а зв'язок має між сутностями «студент» та «група» володіє атрибутом «id групи».

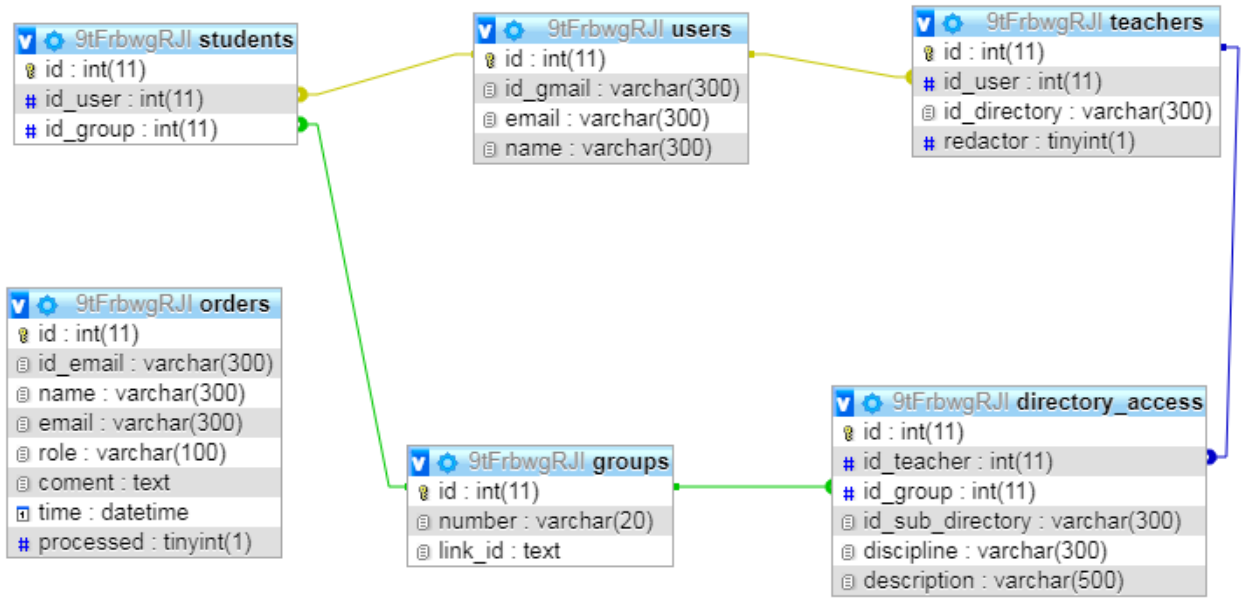

Рисунок 3.4 - ERD 0-го рівня

3) Створення проекту Java.

Найбільш значущі класи та бібліотеки, застосовані у проекті:

- Колекції з бібліотеки java.util;

Найчастіше застосовувались List та ArrayList. Перший є інтерфейсом, і містить методи для роботи зі списком даних. Другий є класом та однією з існуючих реалізацій інтерфейсу List. Для прикладу ArrayList використовується для формування списку файлів, доступних в особистому кабінеті студента.

java.util.Map - це структура даних, в якій об'єкти зберігаються не по одному, як у всіх інших, а в парі "ключ - значення".

Функція handleURLEncodedResponse виконує парсинг xml та формує у відповідь структуру даних типу Map:

```
public static Map handleURLEncodedResponse(HttpResponse response){
Map<String, Charset> map = Charset.availableCharsets();
Map<String,String> oauthResponse = new HashMap<String, String>();
// Структура oauthResponse є контейнером для результату

Set<Map.Entry<String, Charset>> set = map.entrySet();
Charset charset = null;
HttpEntity entity = response.getEntity();
for (Map.Entry<String, Charset> entry : set) {
   if (entry.getKey().equalsIgnoreCase(HTTP.UTF_8)) {
      charset = entry.getValue();
   }
}
try {
   List<NameValuePair> list = URLEncodedUtils.parse(EntityUtils
   .toString(entity), Charset.forName(HTTP.UTF_8));
   for (NameValuePair pair : list) {
      oauthResponse.put(pair.getName(), pair.getValue());
   }
}
catch (IOException e) {
  Error.logError(e, "Could not parse URLEncoded Response");
}
return oauthResponse;
}
```

- Класи бібліотек javax.servlet.http та org.apache.http.

Функціонал цих класів полягає у здійсненні та обробці запитів програми до сторонніх ресурсів та від клієнтів до програми. Маються на увазі саме ті запити, які надходять за http-протоколом.

Клас HttpServlet є головним інструментом, що дозволяє приймати запити і надсилати у відповідь довільну інформацію, зокрема, веб-сторінки. Завдяки цьому класу проект реалізовано у якості веб-додатку.

Сервлети веб-додатку здатні приймати запити типу get і post, мають для цього у своєму складі відповідні методи doGet і doPost, вхідними параметрами яких запит і майбутня відповідь. Запит можна читати, а відповідь записувати.

Наступний приклад коду демонструє оголошення сервлету для обробки заявок на реєстрацію користувачів та його функцію, що «реагує» на запити користувачів у браузері.

```
public class OrderManager extends javax.servlet.http.HttpServlet
```

// За сигнатурою можна зрозуміти, що клас наслідує HttpServlet і це робить його сервлетом. Анотація , застосована далі, вказує на наслідування методу від батьківського класу. Метод перезаписується у дочірньому класі.

// Функція, що приймає дані форми від незареєстрованого користувача:

```
@Override
public void doPost(HttpServletRequest request,
HttpServletResponse response) throws IOException {
    request.setCharacterEncoding("utf8");
    response.setContentType("text/html;charset=UTF-8");
    String action = request.getRequestURI();
    if(action.equals("/new_order")) {
      createNewOrder(request, response);
    }
    else if(action.equals("/new_order_accept")) {
      acceptOrder(request, response);
    }
  }…
}
```

Складовою частиною проекту є jsp-сторінки. Зазвичай саме їх бачить користувач при відвідуванні сайту, серверна частина якого написана мовою Java. Технологія JSP забезпечує динамічну генерацію HTML, XML на інших веб-сторінок. Деякі конструкції JSP дозволяють обробляти код, написаний на Java, тому безпосередньо на веб-сторінці можуть оброблятися об’єкти класів.

Для спрощення елементів скриптів, є доступ до декількох заздалегідь визначених змінним, таким, наприклад, як змінні HttpServletRequest request, HttpServletResponse response, PrintWriter out.

Приклад коду:

```
<html>
   <head>
      <meta charset="UTF-8" />
      <title>First JSP App</title>
   </head>
   <body>
      <% //початок скрипллету
         for(int i = 1; i < 5; i++){
           out.println("<br>Hello " + i);
         }
      %> //кінець скриплету
   </body>
</html>
```

4) Створення тестових користувачів і каталогів на гугл-диску.

У якості тестових користувачів було створено декілька облікових записів у Google, а також запрошено декількох реальних користувачів. Паролі до тестових акантів однакові – «myjournal_123».

Таблиця 3.1 – Користувачі веб-додатку

Обліковий запис	**Роль**
journal.sumdu@gmail.com	Викладач, адміністратор
n.tirkusova@cs.sumdu.edu.ua	Викладач
o.berest@cs.sumdu.edu.ua	Викладач
ivanovvanka266@gmail.com	Викладач, тестовий акаунт
nasty.pochkoon@gmail.com	Студент
darapugac9@gmail.com	Студент, тестовий акаунт
myskills.nasty.pochkoon@gmail.com	Студент
i.r.k.a.18lol@gmail.com	Студент

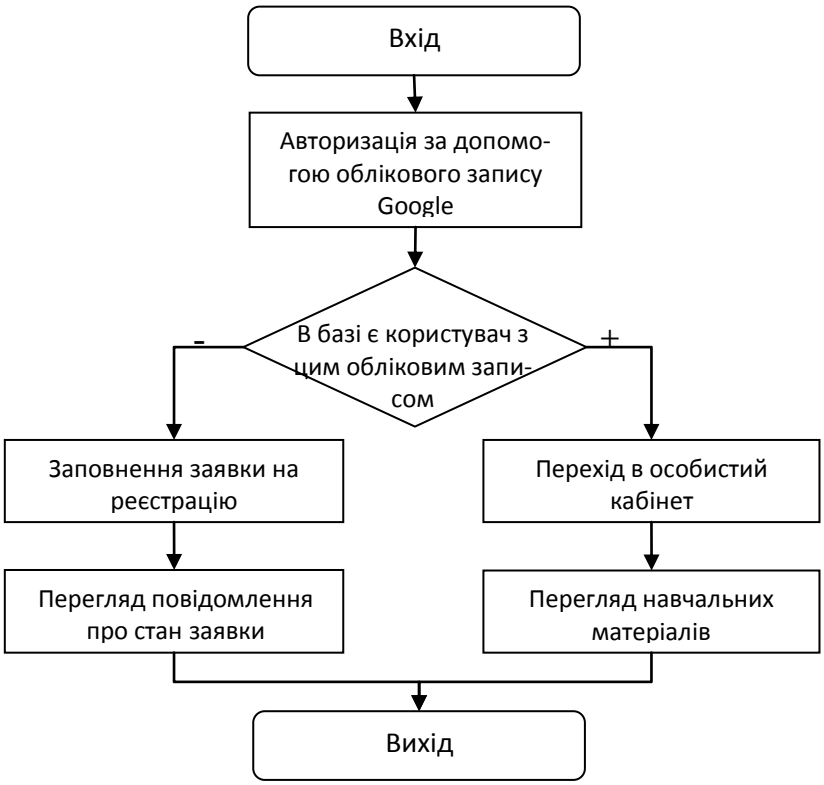

Рисунок 3.5 – Блок-схема, що зображує еталонну послідовність дій користувача

3.2 Технічні вимоги до серверу

Сайт myjournal.com.ua розміщено на хостину 1G. Вибір хостингу обумовлено складом та вартістю запропонованих послуг.

1) Можливість віддаленого доступу до серверу для того, щоб завантажити і контролювати роботу програми Apache Tomcat та платформи Java;

2) Операційна система Ubuntu або Centos, випущена не раніше, ніж три роки тому. Причиною є те, що старі версії ОС не здатні взаємодіяти з сучасними версіями веб-серверів та Java;

3) Не менше, ніж 50 Гб дискової пам’яті;

4) Хостинг для однієї бази даних.

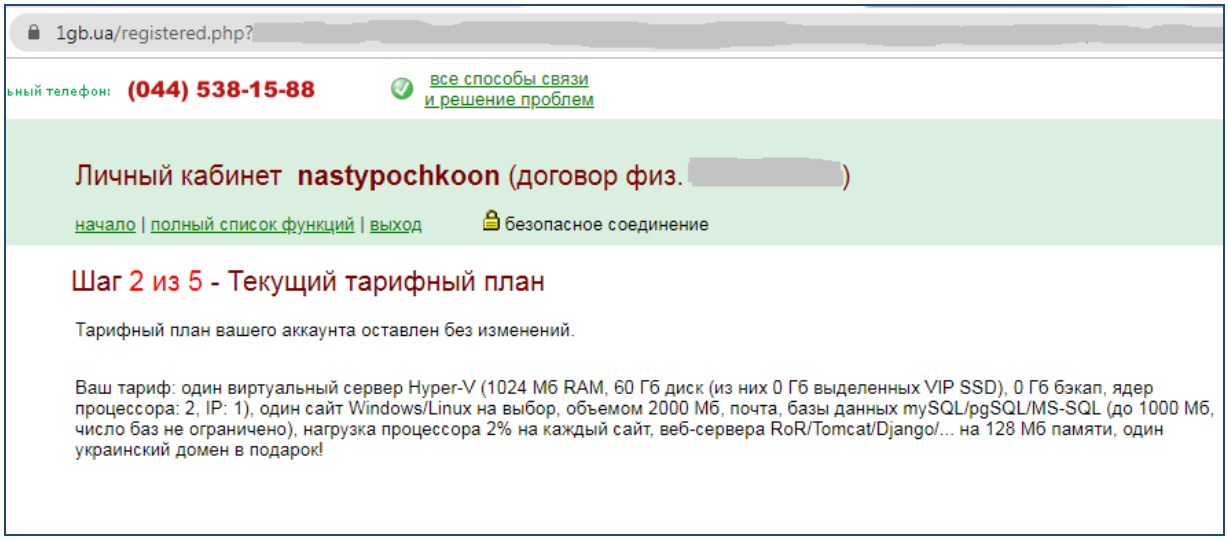

Рисунок 3.6 – Скріншот з особистого кабінету хостингу, параметри придбаного серверу

3.3 Результати розробки

Сайт доступний за посиланням: http://myjournal.com.ua/.

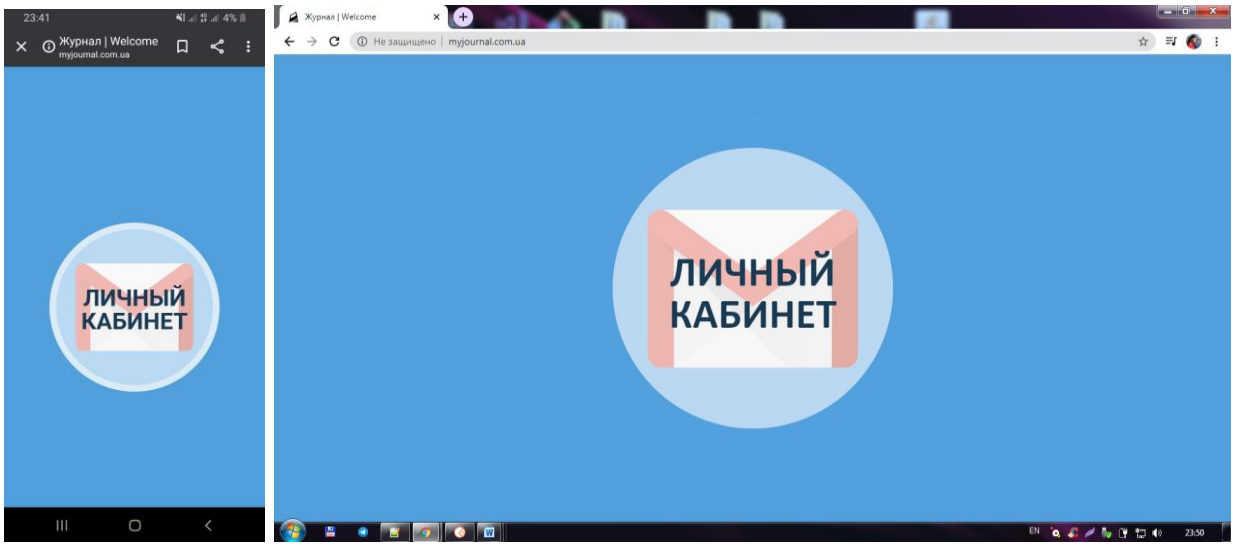

Рисунок 3.7 – Скріншоти головної сторінки у двох різних режимах: у смартфоні та на комп’ютері

Пульсуюча кнопка усередині екрану вказує на те, що при її натисканні відбудеться спроба авторизації облікового запису Google, а потім користувач буде направлений до особистого кабінету.

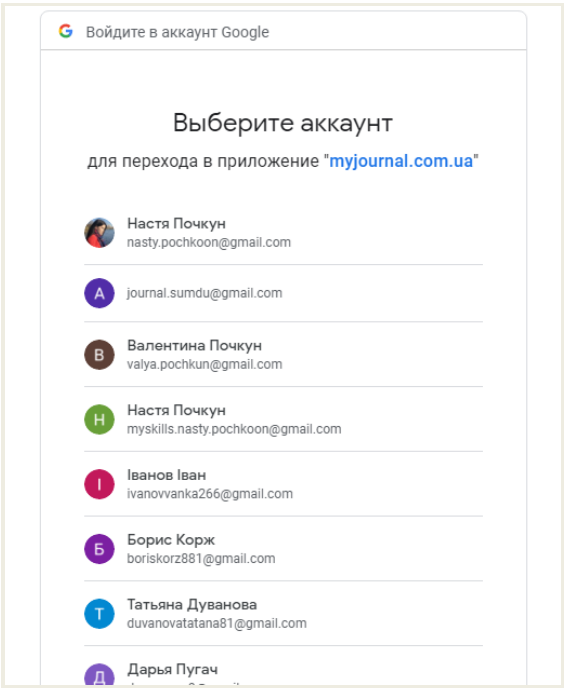

Рисунок 3.8 – Авторизація облікового запису Google

Під час першого візиту у особистому кабінеті користувач бачить форму для реєстрації облікового запису. Щоб мати можливість користуватись навчальними ресурсами, студентові необхідно на цій формі обрати «Студент» і в текстовому полі вказати додаткову інформацію про себе: ПІБ, групу тощо. Так адміністратор швидше обробить цей запит. Викладачеві для реєстрації потрібно обрати варіант «Преподаватель».

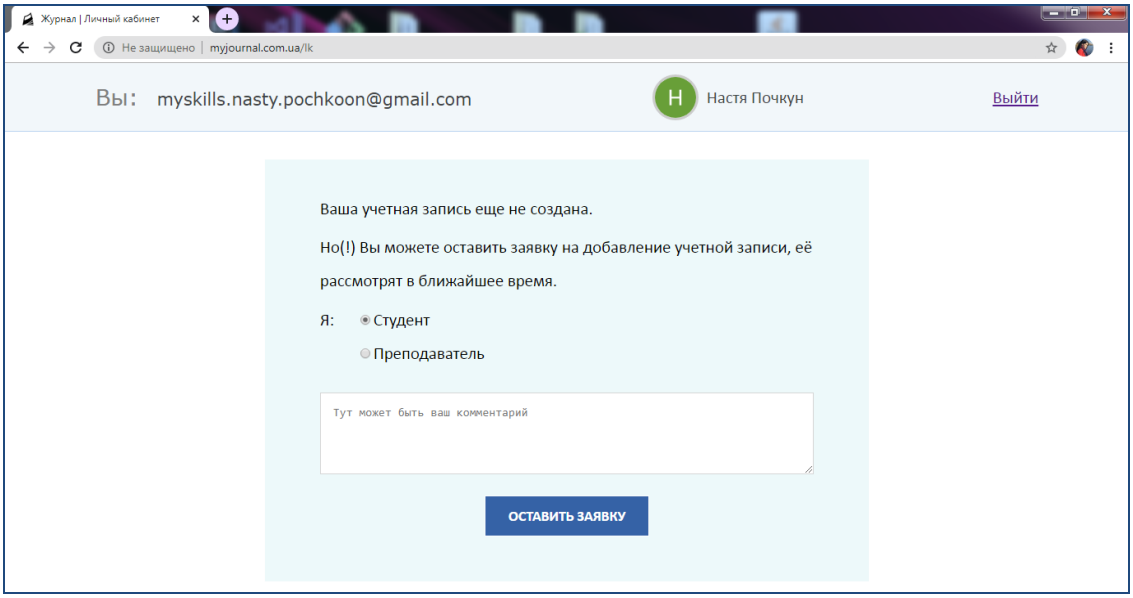

Рисунок 3.9 – Форма відправки запиту на реєстрацію облікового запиту

Після погодження запиту на реєстрацію адміністратором, авторизований студент бачить сторінку особистого кабінету.

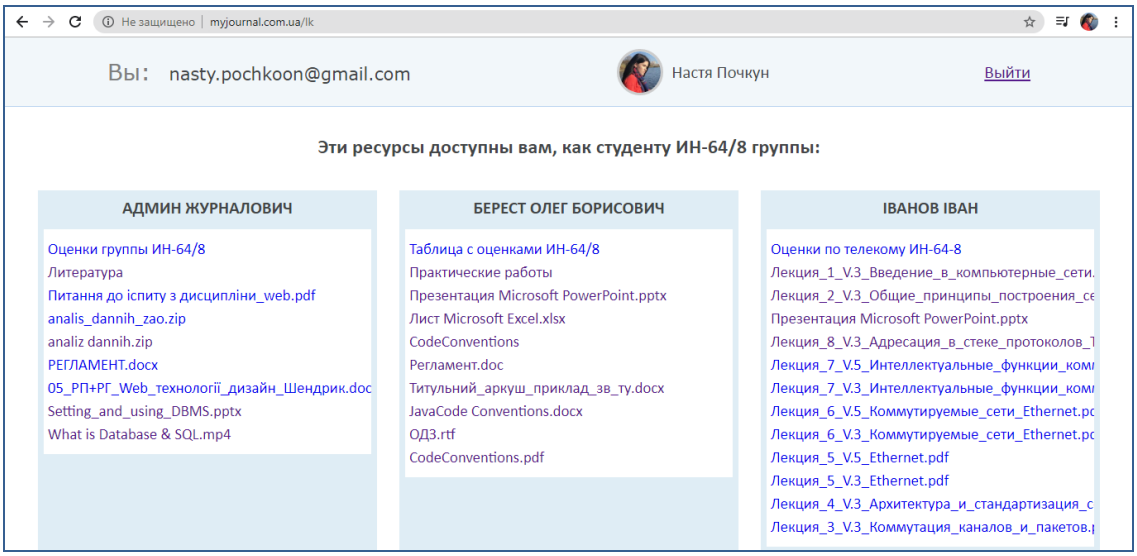

Рисунок 3.10 – Особистий кабінет студента

Авторизований викладач в особистому кабінеті бачить список каталогів, вкладених в його папку на гугл диску. До цих «підкаталогів» він може відкривати доступ певним групам студентів.

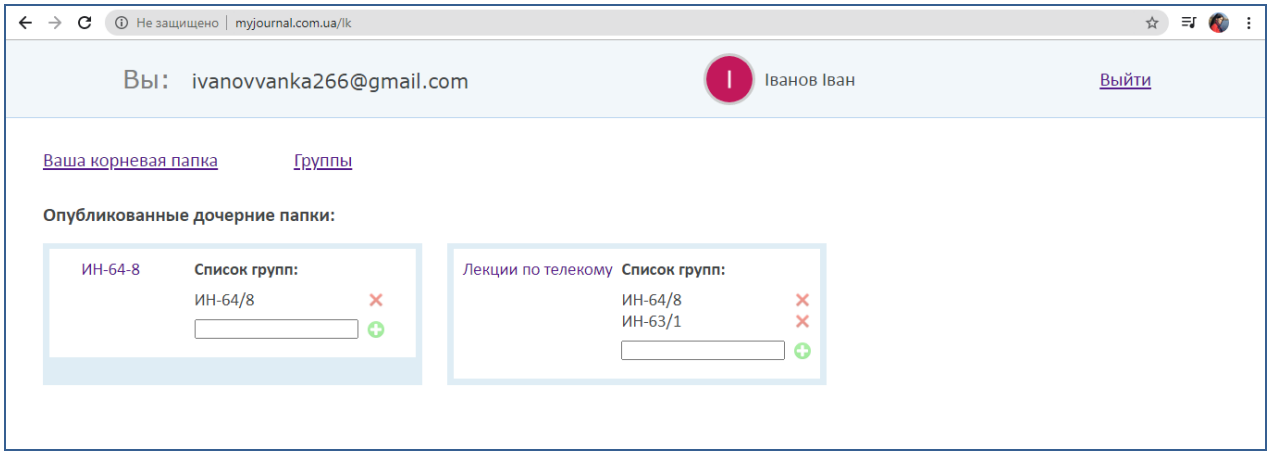

Рисунок 3.11 – Особистий кабінет викладача зі списком папок, розташованих у його каталозі на гугл-диску журналу

Майже весь функціонал адміністративної панелі доступний лише після отримання додаткових прав. Це можна зробити перейшовши на сторінку «Группы» і натиснувши на «Получить администские права». В цей момент спрацьовує перенаправлення на сторінку вибору облікового запису, це виглядає як повторна авторизація. Дійсно відбувається авторизація, але з більшим набором повноважень і доступів до ресурсів гугл-диску. Під час першої спроби користувач бачить повідомлення з проханням підтвердити надання права внесення змін до файлів на його гугл-диску. Звичайно, що не від усіх користувачів потрібно просити такий доступ, тому було вирішено винести посилання на цей «особливий вхід» до особистого кабінету викладача. Тож, щоб користуватись розширеним функціоналом, викладач має авторизуватись двічі: першого разу як звичайний викладач, а другого – як адміністратор.

На сторінці «Группы» викладач може переглядати списки груп та їх студентів, що вже зареєструвались у системі. Після вибору групи стає активним посилання на таблицю Excel, яка може бути шаблоном для таблиць з оцінками.

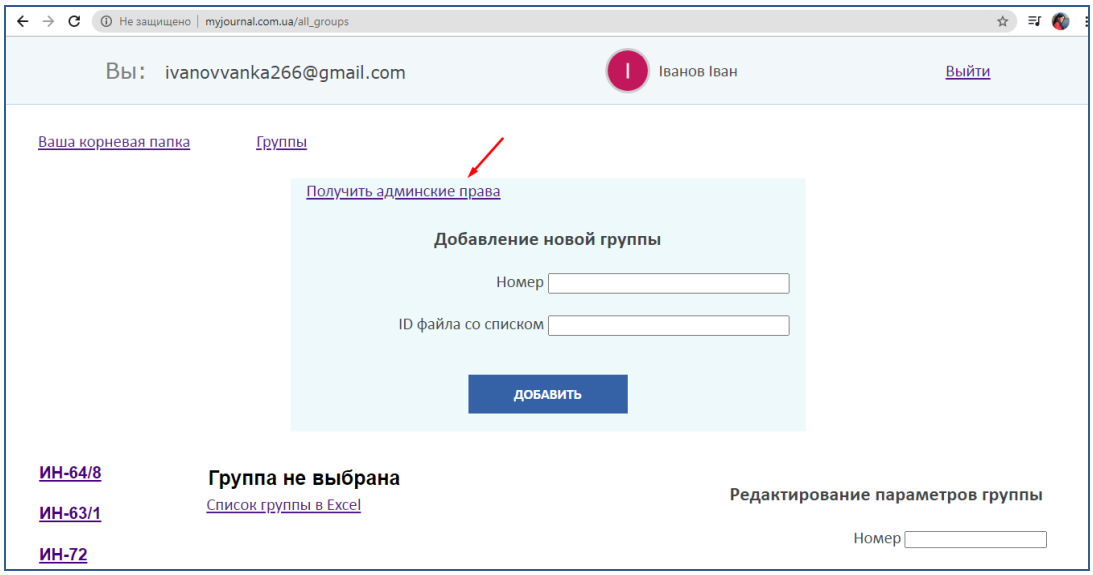

Рисунок 3.12 – Скріншот сторінки «Группы» в адмін.панелі

Без отримання додаткових прав, додання нової групи має обов’язково супроводжуватить вказанням id файлу, що є еталонним списком групи. Id файлу отпримати легко, поглянувши на посилання можна побачити окрему частину, відокремлену слешами. Вона виглядає так:

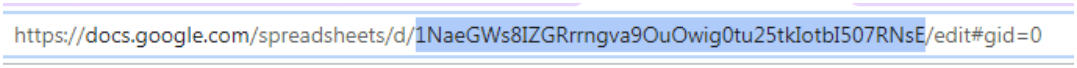

Рисунок 3.13 – Приклад посилання з виділеним Id файлу

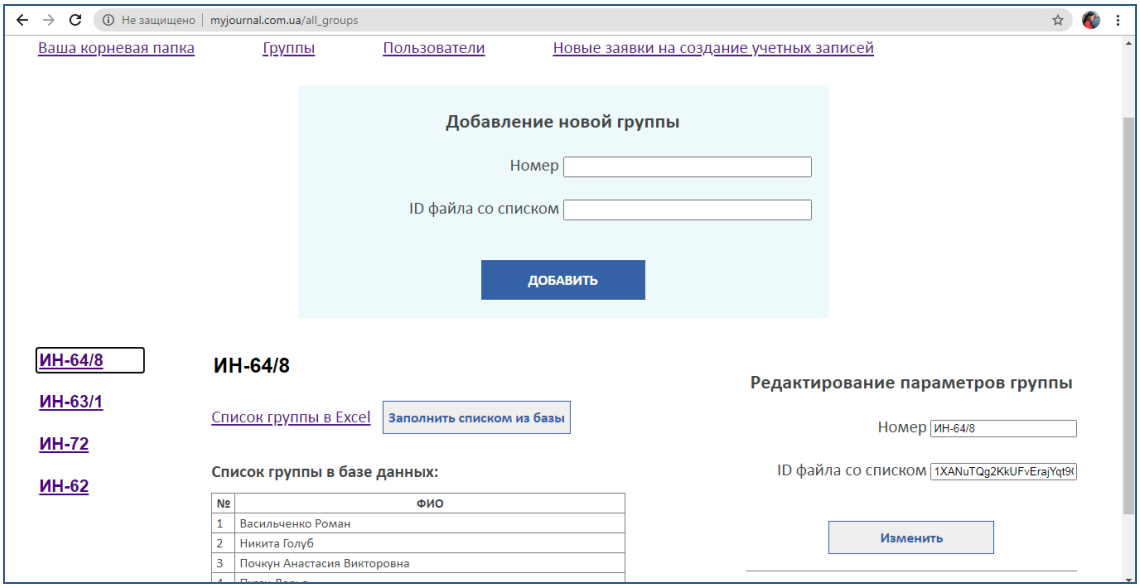

Рисунок 3.14 – Скріншот сторінки адмін.панелі для редагування даних про групи студентів викладачем з можливостями адміністратора

Після отримання прав адміністратора можна помітити, що з’явилась нова функція «Заполнить список миз бази». Це означає, що еталонна гугл-таблиця спочатку очиститься, а потім буде заповнена даними тих студентів, які вже є в системі. Тобто у файлі потрапить список, розмішений трохи нижче, під кнопкою. Також особливістю э те, що при створенні групи адміну не обов’язково вказувати id файлу, оскільки у нього достатньо прав на автоматичне створення таблиці на диску.

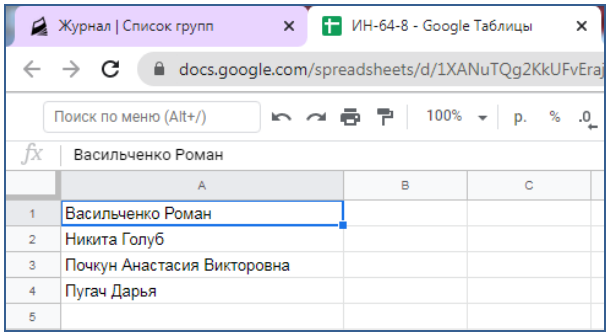

Рисунок 3.15 – Скріншот таблиці зі списком студентів

З набуттям адміністративних прав відкривається доступ до редагування списку користувачів. Можна перейти до ще двох сторінок: перша з існуючими користувачами, друга – з новими запитами. Прийняття запиту приховує користувача у списку нових і додає до списку користувачів. У разі його видалення зі списку, запит користувача знову стає активним і таким, який можна прийняти або відхилити.

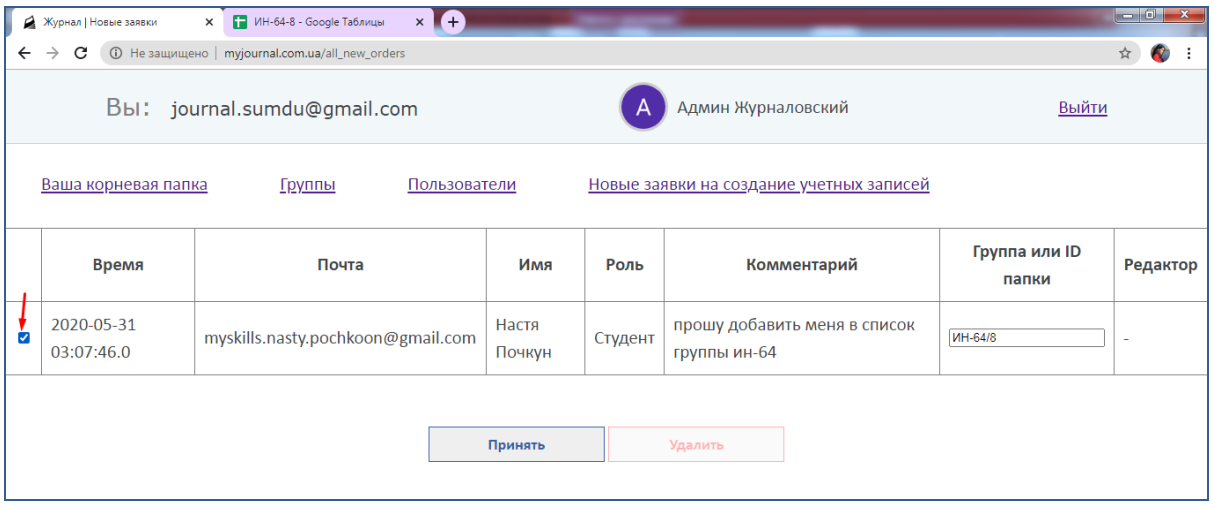

Рисунок 3.16 – Скрішнот сторінки для обробки нових запитів на реєстрацію

Як було описано попередньо, адміністратор бачить коментар студента, завдяки якому може витратити менше часу на обробку заявки.

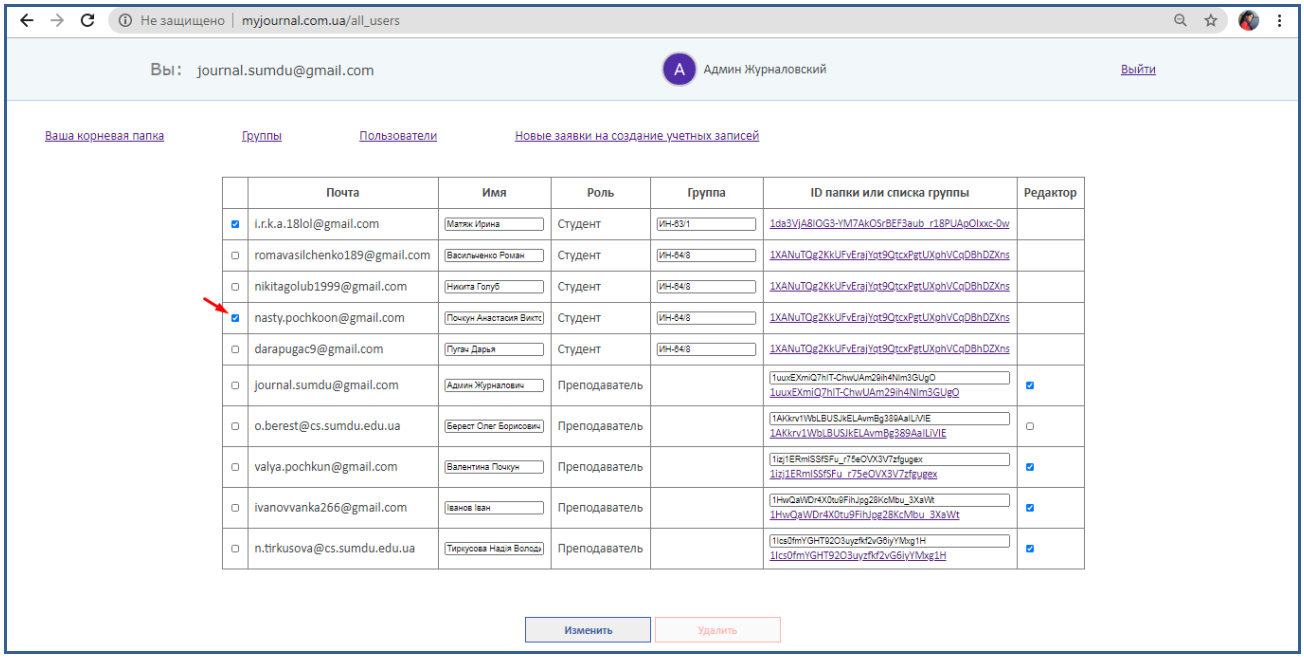

Рисунок 3.17 – Скрішнот сторінки для редагування даних користувачів

Для того, щоб обрати позицію, в якій необхідно зберегти зміни, потрібно відмітити її галочкою. Для студента обо’язковим є заповнення групи. У випадку з викладачем обо’язковим є шлях до його папки на гугл-диску.

У випадку успішного внесення змін, адміністратор бачить відповідне повідомлення.

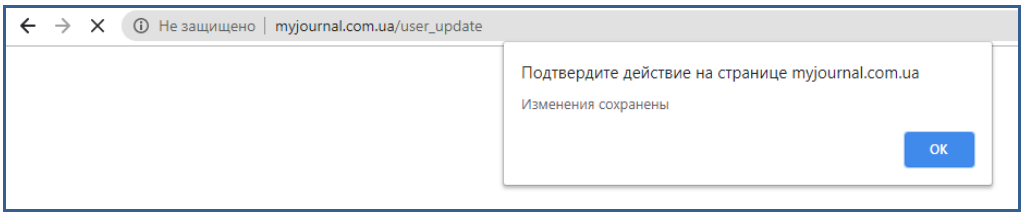

Рисунок 3.18 – Повідомлення про успішне внесення змін

ВИСНОВОК

У результаті виконання завдання спроектовано та реалізовано сайт “myjournal.com.ua”. Сайт розміщено на хостингу, знаходиться у публічному доступі. Зручним у ньому є те, що для входу користувачеві не потрібен пароль, авторизація виконується за допомогою облікового запису Google.

Спочатку цей проект мав ідею електронного журналу, але потім виявилося, що він також може бути інструментом отримання доступу до різних файлів, не тільки таблиць з оцінками. Спрощення процесу пошуку навчальних ресурсів і безпека їх зберігання – головні переваги сайту. Враховуючи переважний досвід викладачів у роботі з сервісами Google, такими як пошта або диск, додаток має бути простим у освоєнні та користуванні.

На сучасному етапі розвитку значно підвищуються вимоги до рівня освіти та підготовки спеціалістів, які повинні володіти запасом теоретичних знань, практичних навичок, вміти орієнтуватись у складній ситуації, бути готовими приймати нестандартні рішення.

До навчальних закладів приходить нове покоління, яке пристосувалось до сучасних технологій. Тому вже зараз час задуматись про вдосконалення існуючих заходів, групування інформації, орієнтацію на потреби сучасного студента і спрощення ведення навчальних процесів для викладачів.

Навіть подібний проект з простою ідею, в певній мірі, підтримує тенденцію інформатизації освітніх процесів.

СПИСОК ДЖЕРЕЛ

1. Дистанційна освіта в Україні: навчання у стилі digital nomads [Електронний ресурс]. – Режим доступу: https://studway.com.ua/distanciyna-osvita/

2. Психолого-педагогічні проблеми вищої і середньої освіти в умовах сучасних викликів: теорія і практика: матеріали ІІІ Міжнародної науково-практичної конференції.. - С. 88-90.

3. Сервлеты | Руководство по сервлету [Електронний ресурс]. – Режим доступу: https://www.javatpoint.com/servlet-tutorial

4. ДІДЖИТАЛІЗАЦІЯ ОСВІТИ – КОМПЕТЕНЦІЇ XXI СТОЛІТТЯ [Електронний ресурс]. – Режим доступу: https://vseosvita.ua/library/didzitalizacia-osviti-kompetencii-hhi-stolitta-172970.html

5. Как входить на сайты и в приложения, используя аккаунт Google [Електронний ресурс]. – Режим доступу: https://support.google.com/accounts/answer/112802?co=GENIE.Platform%3DDesktop&hl=ru

6. Електронні щоденники та журнали [Електронний ресурс]. – Режим доступу: https://e-schools.info/

7. Google Classroom [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom

8. Брюс Еккель. Філософія Java / Брюс Еккель. - Четверте видання. – Місце видання: Київ, 2016 рік

9. Роберт Седжвік. Алгоритми на Java / Роберт Седжвік, Кевін Уейн - Четверте видання. – Місце видання: Харків, 2016 рік

10. Джош Лонг. Java в хмарі. Spring Boot, Spring Cloud, Cloud Foundry / Джош Лонг, Кеннет Бастані - Місце видання: Санкт-Петербург, 2019 рік

ДОДАТОК А

КОНФІГУРАЦІЯ ДОДАТКУ У ФАЙЛІ WEB.XML

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                  http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0">

<servlet>
    <servlet-name>default</servlet-name>
    <servlet-
class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>0</param-value>
    </init-param>
    <init-param>
        <param-name>listings</param-name>
        <param-value>false</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>Welcome</servlet-name>
    <jsp-file>/welcome.jsp</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>Welcome</servlet-name>
    <url-pattern>/welcome</url-pattern>
    <url-pattern>/welcome/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AuthServlet</servlet-name>
    <servlet-class>com.journal.servlets.AuthServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AuthServlet</servlet-name>
    <url-pattern>/auth</url-pattern>
    <url-pattern>/auth/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>GoogleAuthServlet</servlet-name>
    <servlet-class>com.journal.servlets.GoogleAuthServlet</servlet-
class>
</servlet>
<servlet-mapping>
    <servlet-name>GoogleAuthServlet</servlet-name>
    <url-pattern>/google-auth</url-pattern>
```

```
    <url-pattern>/google-auth/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>LkServlet</servlet-name>
    <jsp-file>/lk.jsp</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>LkServlet</servlet-name>
    <url-pattern>/lk</url-pattern>
    <url-pattern>/lk/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AdminAllGroups</servlet-name>
    <jsp-file>/all_groups.jsp</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>AdminAllGroups</servlet-name>
    <url-pattern>/all_groups</url-pattern>
    <url-pattern>/all_groups/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AdminAllUsers</servlet-name>
    <jsp-file>/all_users.jsp</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>AdminAllUsers</servlet-name>
    <url-pattern>/all_users</url-pattern>
    <url-pattern>/all_users/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AdminAllNewOrders</servlet-name>
    <jsp-file>/all_new_orders.jsp</jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>AdminAllNewOrders</servlet-name>
    <url-pattern>/all_new_orders</url-pattern>
    <url-pattern>/all_new_orders/</url-pattern>
     <url-pattern>/new_order_delete</url-pattern>
    <url-pattern>/new_order_delete/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>OrderManager</servlet-name>
    <servlet-class>com.journal.servlets.OrderManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>OrderManager</servlet-name>
    <url-pattern>/new_order</url-pattern>
    <url-pattern>/new_order/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>OrderManager</servlet-name>
```

```
    <servlet-class>com.journal.servlets.OrderManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>OrderManager</servlet-name>
    <url-pattern>/new_order_accept</url-pattern>
    <url-pattern>/new_order_accept/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>NewGroup</servlet-name>
    <servlet-class>com.journal.servlets.GroupManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>NewGroup</servlet-name>
    <url-pattern>/new_group</url-pattern>
    <url-pattern>/new_group/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>GetStudentList</servlet-name>
    <servlet-class>com.journal.servlets.GroupManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>GetStudentList</servlet-name>
    <url-pattern>/get_student_list</url-pattern>
    <url-pattern>/get_student_list/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>EditGroup</servlet-name>
    <servlet-class>com.journal.servlets.GroupManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>EditGroup</servlet-name>
    <url-pattern>/edit_group</url-pattern>
    <url-pattern>/edit_group/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>DeleteGroup</servlet-name>
    <servlet-class>com.journal.servlets.GroupManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DeleteGroup</servlet-name>
    <url-pattern>/delete_group</url-pattern>
    <url-pattern>/delete_group/</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>ResetGroupList</servlet-name>
    <servlet-class>com.journal.servlets.GroupManager</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ResetGroupList</servlet-name>
    <url-pattern>/reset_group_list</url-pattern>
    <url-pattern>/reset_group_list/</url-pattern>
</servlet-mapping>
```

```
<servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-
class>
    <init-param>
        <param-name>fork</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>xpoweredBy</param-name>
        <param-value>false</param-value>
    </init-param>
    <load-on-startup>3</load-on-startup>
</servlet>

<!-- The mapping for the default servlet -->
<servlet-mapping>
    <servlet-name>default</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- The mappings for the JSP servlet -->
<servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.jsp</url-pattern>
    <url-pattern>*.jspx</url-pattern>
</servlet-mapping>

<session-config>
    <session-timeout>30</session-timeout>
</session-config>
…
<welcome-file-list>
    <welcome-file>welcome.html</welcome-file>
    <welcome-file>welcome.htm</welcome-file>
    <welcome-file>welcome.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

ДОДАТОК В

СЕРВЛЕТИ, ЩО ЗДІЙСНЮЮТЬ ОБРОБКУ ДАНИХ ПІД ЧАС АВТОРИЗАЦІЇ

// Сервлет AuthServlet, який формує запит до Google з метою здійснення авторизації через обліковий запис

```
package com.journal.servlets;
import com.Error;
import com.api.CreateCode;
import com.journal.actions.Logout;
import com.journal.model.User;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class AuthServlet extends javax.servlet.http.HttpServlet {

  @Override
  public void init() throws ServletException {
    super.init();
  }

  @Override
  public void doGet(
   HttpServletRequest request,
   HttpServletResponse response) throws ServletException, IOException {

    Object obj = request.getSession().getAttribute("user");
    String role = "" + request.getParameter("role");
    if(obj != null &&
    obj instanceof User && !(role.equals("prepod"))) {
      response.sendRedirect("http://myjournal.com.ua/lk");
    }
    else {
      Logout.handle(request);
      try {
        CreateCode action = new CreateCode();
        action.handle(request, response);
      }
      catch (Exception e) {
        request.getSession().invalidate();
        Error.logError(e);
      }
    }

  }
}
```

// Сервлет GoogleAuthServlet приймає зворотній запит від Google, що містить у собі ключову інформацію

```
package com.journal.servlets;
import com.Error;
import com.api.CreateToken;
import com.api.Userinfo;
import com.inside.DAO;
import com.journal.model.Teacher;
import com.journal.model.User;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

public class GoogleAuthServlet extends javax.servlet.http.HttpServlet
{
@Override
public void init() throws ServletException { super.init(); }

@Override
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {

  try {
    HttpSession session = request.getSession(true);
    String code   = "" + request.getParameter("code");
    Object user   = session.getAttribute("user");
    session.setAttribute("code", code);
    if((!code.isEmpty()) && !(user instanceof User)) {
      CreateToken action = new CreateToken(code);
      boolean isAdmin = (boolean)request.getSession() .getAttribute(
      "isAdmin");
      action.handle(isAdmin);
      session.setAttribute("access_token",  action.getAccessToken());
      session.setAttribute("id_token",    action.getIdToken());
      Userinfo userinfo = new Userinfo();
      userinfo.setClient(action.getClient());
      userinfo.handle(request, response);
      User person = DAO.getPerson(request);
      if((!isAdmin) && person instanceof Teacher){
        ((Teacher) person).setRedactor(false);
      }
      if(person != null) {
        request.getSession().setAttribute("person", person);
  }}}
  catch (Exception e){
    request.getSession().invalidate();
    Error.logError(e);
  }
  response.sendRedirect("http://myjournal.com.ua/lk");
 }
}
```

ДОДАТОК С

СЕРВЛЕТИ, ЩО ЗДІЙСНЮЮТЬ ОБРОБКУ ДАНИХ ОКРЕМИХ СУТНОСТЕЙ: ГРУПА, КОРИСТУВАЧ, ЗАПИТ

// Сервлет GroupManager для здійснення операцій з групами студентів: додання, зміна, видалення, отримання списку груп

```
package com.journal.servlets;

import com.Error;
import com.api.SpreadsheetsManager;
import com.google.gson.Gson;
import com.inside.DAO;
import com.journal.model.Group;
import com.journal.model.Teacher;
import com.journal.model.User;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

public class GroupManager extends javax.servlet.http.HttpServlet {

@Override
public void init() throws ServletException { super.init(); }


@Override
public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
  String action = request.getRequestURI();
  User person = (User)request.getSession().getAttribute("person");
  if(action.equals("/get_student_list") && person instanceof Teacher){
    getStudentList(request, response);
    return;
  }
  boolean ok = false;
  request.setCharacterEncoding("utf8");
  response.setContentType("text/html;charset=UTF-8");
  PrintWriter out = response.getWriter();
  String str_redirect = "location.href = `http://myjournal.com.ua/
all_groups`";
  String id_group = request.getParameter("id_group");
  if(id_group.isEmpty() || !(person != null && person instanceof
Teacher)){
    processResponse(out, str_redirect, false);
    return;
  }
```

```
  if(action.equals("/reset_group_list") && ((Teacher) person)
.isRedactor()) {
    String id_file = request.getParameter("id_file");
    String access_token = (String) request.getSession()
.getAttribute("access_token");
    ok = SpreadsheetsManager.checkAndClear(access_token, id_file);
    if (ok) {
    ArrayList list = DAO.getStudentList(id_group);
    ok = SpreadsheetsManager.appendRows(access_token, id_file, list);
    }
  }
  else if (action.equals("/delete_group")) {
    ok = DAO.deleteGroup(id_group);
  }
  if (ok) {
    List<Group> all_groups = DAO.getGroupList();
    request.getSession().setAttribute("all_groups", all_groups);
  }
  processResponse(out, str_redirect, ok);
}


public static void getStudentList(HttpServletRequest request,
HttpServletResponse response){
  try {
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/json");
    String id_group = request.getParameter("id_group");
    User person = (User) request.getSession().getAttribute("person");
    if (person != null && id_group != null) {
      ArrayList list = DAO.getStudentList(id_group);
      PrintWriter out = response.getWriter();
      String JsonString = new Gson().toJson(list);
      out.print(JsonString);
      out.flush();
    }
  }
  catch (Exception e){ Error.logError(e);}
}


@Override
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
  boolean ok = false;
  request.setCharacterEncoding("utf8");
  response.setContentType("text/html;charset=UTF-8");
  PrintWriter out = response.getWriter();
  String str_redirect = "location.href = `http://myjournal.com.ua
/all_groups`";
  User person = (User)request.getSession().getAttribute("person");
  if(!(person != null && person instanceof Teacher)){
    processResponse(out, str_redirect, false);
    return;
  }
  try {
    String action = request.getParameter("action");
```

```
    if(action.equals("add")) {
      ok = DAO.createNewGroup(request);
    }
    else if(action.equals("edit")) {
      ok = DAO.editGroup(request);
    }
    List<Group> all_groups = DAO.getGroupList();
    request.getSession().setAttribute("all_groups", all_groups);
  }
  catch (Exception e) {Error.logError(e);}
  processResponse(out, str_redirect, ok);
}


public void processResponse(PrintWriter out, String str_redirect,
boolean ok){
  if(ok) {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Изменения сохранены');" + str_redirect);
    out.println("</script>");
  }
  else {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Произошла ошибка');" + str_redirect);
    out.println("</script>");
  }
}}
```

// Сервлет UserManager для здійснення операцій з користувачами: додання, зміна, видалення, отримання списку

```
package com.journal.servlets;

import com.Error;
import com.inside.DAO;
import com.journal.model.Teacher;
import com.journal.model.User;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;


public class UserManager extends javax.servlet.http.HttpServlet {

@Override
public void init() throws ServletException { super.init(); }


@Override
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
  request.setCharacterEncoding("utf8");
  response.setContentType("text/html;charset=UTF-8");
```

```
  String action = request.getRequestURI();
  if (action.equals("/user_update") || action.equals("/user_update/"))
{
    updateUsers(request, response);
  } else if (action.equals("/user_delete") || action.equals(
"/user_delete/")) {
    deleteUsers(request, response);
  }
}

public void updateUsers(HttpServletRequest request,
HttpServletResponse response) throws IOException {
  PrintWriter out = response.getWriter();
  String str_redirect = "location.href = `http://myjournal.com.ua
/all_users`";
  boolean ok = false;
  User person = (User)request.getSession().getAttribute("person");
  if(!(person != null && person instanceof Teacher && ((Teacher)
person).isRedactor())){
    processResponse(out, str_redirect, false);
    return;
  }
  try { ok = DAO.updateUsers(request);}
  catch (Exception e) { Error.logError(e);}
  processResponse(out, str_redirect, ok);
}

public void deleteUsers(HttpServletRequest request,
HttpServletResponse response) throws IOException {
  PrintWriter out = response.getWriter();
  String str_redirect = "location.href = `http://myjournal.com.ua
/all_users`";
  boolean ok = false;
  User person = (User)request.getSession().getAttribute("person");
  if(!(person != null && person instanceof Teacher && ((Teacher)
person).isRedactor())){
    processResponse(out, str_redirect, false);
    return;
  }
  try { ok = DAO.deleteUsers(request);}
  catch (Exception e) { Error.logError(e);}
  processResponse(out, str_redirect, ok);
}


public void processResponse(PrintWriter out, String str_redirect,
boolean ok){
  if(ok) {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Изменения сохранены');" + str_redirect);
    out.println("</script>");
  }
  else {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Произошла ошибка');" + str_redirect);
    out.println("</script>");
 }}}
```

// Сервлет для обробки даних запиту користувача на реєстрацію: додання, прийняття, видалення, отримання списку запитів

```
package com.journal.servlets;

import com.Error;
import com.inside.DAO;
import com.journal.model.Teacher;
import com.journal.model.User;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class OrderManager extends javax.servlet.http.HttpServlet {

@Override
public void init() throws ServletException { super.init(); }


@Override
public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
  request.setCharacterEncoding("utf8");
  response.setContentType("text/html;charset=UTF-8");
  String action = request.getRequestURI();
  if (action.equals("/new_order") || action.equals("/new_order/")) {
    createNewOrder(request, response);
  } else if (action.equals("/new_order_accept") || action.equals(
"/new_order_accept/")) {
    acceptOrder(request, response);
  } else if (action.equals("/new_order_delete") || action.equals(
"/new_order_delete/")) {
    deleteOrder(request, response);
  }
}


public void createNewOrder(HttpServletRequest request,
HttpServletResponse response) throws IOException {
  boolean orderIsCreated = false;
  Object obj = request.getSession().getAttribute("orderIsCreated");
  PrintWriter out = response.getWriter();
  if (obj != null) {
    orderIsCreated = (boolean) obj;
  }
  String str_redirect = "location.href=`http://myjournal.com.ua/lk`";
  User user = (User) request.getSession().getAttribute("user");
  if(user == null){
    processResponse(out, str_redirect, false);
    return;
  }
  if (orderIsCreated) {
    out.println("<script type=\"text/javascript\">");
```

```
    out.println("alert('Запрос уже принят ранее. Ожидайте.');" +
str_redirect);
    out.println("</script>");
  }
  else {
    try {
      orderIsCreated = DAO.checkOldOrder(user);
      request.getSession().setAttribute("orderIsCreated",
orderIsCreated);
      if (!orderIsCreated) {
        orderIsCreated = DAO.createNewOrder(request);
        processResponse(out, str_redirect, orderIsCreated);
      }
      else {
        out.println("<script type=\"text/javascript\">");
        out.println("alert('Запрос уже принят ранее. Ожидайте.');" +
str_redirect);
        out.println("</script>");
      }
    } catch (Exception e) { Error.logError(e);}
  }
}


public void acceptOrder(HttpServletRequest request,
HttpServletResponse response) throws IOException {
  PrintWriter out = response.getWriter();
  String str_redirect =
"location.href=`http://myjournal.com.ua/all_new_orders`";
  boolean ok = false;
  User person = (User)request.getSession().getAttribute("person");
  if(!(person != null && person instanceof Teacher && ((Teacher)
person).isRedactor())){
    processResponse(out, str_redirect, false);
    return;
  }
  try { ok = DAO.acceptOrders(request);}
  catch (Exception e) { Error.logError(e);}
  processResponse(out, str_redirect, ok);
}

public void deleteOrder(HttpServletRequest request,
HttpServletResponse response) throws IOException {
  PrintWriter out = response.getWriter();
  String str_redirect =
"location.href=`http://myjournal.com.ua/all_new_orders`";
  boolean ok = false;
  User person = (User)request.getSession().getAttribute("person");
  if(!(person != null && person instanceof Teacher && ((Teacher)
person).isRedactor())){
    processResponse(out, str_redirect, false);
    return;
  }
  try { ok = DAO.deleteOrders(request);}
  catch (Exception e) { Error.logError(e);}
  processResponse(out, str_redirect, ok);
}
```

```
public void processResponse(PrintWriter out, String str_redirect,
boolean ok){
  if(ok) {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Изменения сохранены');" + str_redirect);
    out.println("</script>");
  }
  else {
    out.println("<script type=\"text/javascript\">");
    out.println("alert('Произошла ошибка');" + str_redirect);
    out.println("</script>");
  }
}}
```

ДОДАТОК D

КЛАС DAO, ЯКИЙ МІСТИТЬ ПЕРЕВАЖНУ КІЛЬКІСТЬ ФУНКЦІЙ ДЛЯ ВИКОНАННЯ ДІЙ З ОБ'ЄКТАМИ

```
package com.inside;
import com.Error;
import com.api.Client;
import com.api.ResponseParser;
import com.api.SpreadsheetsManager;
import com.journal.actions.Logout;
import com.journal.model.*;
import org.json.simple.JSONObject;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.json.simple.JSONArray;

import javax.servlet.http.HttpServletRequest;
import java.sql.*;
import java.time.LocalDateTime;
import java.util.*;

public class DAO {
private static Connection conn;
public static void setConn() throws Exception {
  Class.forName("com.mysql.jdbc.Driver");
  String username = "***";
  String url = "jdbc:mysql://mysql316.1gb.ua:3306/gbua_myjournal";
  String password = "***";
  Properties props = new Properties();
  props.setProperty("user", username);
  props.setProperty("useSSL", "false");
  props.setProperty("password", password);
  conn = DriverManager.getConnection(url, props);
}


public static Connection getConn() {
  try {
    if (!(conn != null && !(conn.isClosed()))) {setConn();}
  }
  catch (Exception e) {Error.logError(e);}
  return conn;
}


public static void closeConn() {
  try {
    if (conn != null && !(conn.isClosed())) {conn.close();}
  }
  catch (Exception e) {System.out.println(e);}
}
```

```
public static User getFileList(HttpServletRequest request) throws
Exception {
  User person = getPerson(request);
  List filelist = new ArrayList();
  try {
    if (person instanceof Teacher) {
      filelist = getFileListForTeacher(person);
    } else if (person instanceof Student) {
      filelist = getFileListForStudent(person);
    }
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  person.setFiles(filelist);
  return person;
}


public static User getPerson(HttpServletRequest request) throws
Exception {
  User user = (User) request.getSession().getAttribute("user");
  try {
    conn = getConn();

    String queryCheckUser = "SELECT u.id as id_user, MAX(t.id) AS
id_teacher, " +
        "MAX(t.redactor) AS redactor, " +
        "MAX(s.id) AS id_student, MAX(t.id_directory) AS id_directory,
" +
        "MAX(s.id) AS id_student FROM `users` u " +
        "LEFT JOIN `teachers` t ON(u.id = t.id_user) " +
        "LEFT JOIN `students` s ON(u.id = s.id_user) " +
        "WHERE u.id_gmail = ? " +
        "GROUP BY u.id";

    PreparedStatement statementCheckUser =
conn.prepareStatement(queryCheckUser);
    statementCheckUser.setString(1, user.getIdEmail());
    ResultSet resultSet = statementCheckUser.executeQuery();
    boolean userIsValid = resultSet.next();
    String id_person = "";
    String id_user = resultSet.getString("id_user");
    if (userIsValid) {
      id_person = resultSet.getString("id_teacher");
      if (id_person != null && (!id_person.isEmpty())) {
        user.setId(id_user);
        Teacher teacher = new Teacher();
        teacher.setId(id_user);
        teacher.setIdPerson(id_person);
        teacher.setEmail(user.getEmail());
        boolean redactor = false;
```

```
        String string_redactor = resultSet.getString("redactor");
        if (string_redactor.equals("1")) {
          redactor = true;
        }
        teacher.setRedactor(redactor);
        teacher.setIdDirectory(resultSet.getString("id_directory"));
        return teacher;
      } else {
        id_person = resultSet.getString("id_student");
        if (id_person != null && (!id_person.isEmpty())) {
          user.setId(id_user);
          Student student = getStudentById(conn, user);
          student.setIdPerson(id_person);
          student.setEmail(user.getEmail());
          return student;
  }}}}
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  closeConn();
  return user;
}


public static boolean checkOldOrder(User user) {
  boolean result = false;
  try {
    conn = getConn();
    String queryCheckOrder = "SELECT * FROM `orders` WHERE id_email =
?";
    PreparedStatement statementCheckOrder =
conn.prepareStatement(queryCheckOrder);
    statementCheckOrder.setString(1, user.getIdEmail());
    ResultSet resultSet = statementCheckOrder.executeQuery();
    result = resultSet.next();
  } catch (Exception e) {Error.logError(e);}
  closeConn();
  return result;
}


public static boolean createNewOrder(HttpServletRequest request) {
  try {
    conn = getConn();
    User user = (User) request.getSession().getAttribute("user");
    String role = request.getParameter("role");
    String coment = request.getParameter("coment");
    LocalDateTime time = LocalDateTime.now();
    String queryNewOrder = "INSERT INTO `orders`(id_email, name,
email, role, coment, time, processed) " +
        "VALUES(?, ?, ?, ?, ?, ?, false)";
    System.out.println(queryNewOrder);
    PreparedStatement statementCheckOrder =
conn.prepareStatement(queryNewOrder);
```

```
    statementCheckOrder.setString(1, user.getIdEmail());
    statementCheckOrder.setString(2, user.getName());
    statementCheckOrder.setString(3, user.getEmail());
    statementCheckOrder.setString(4, role);
    statementCheckOrder.setString(5, coment);
    statementCheckOrder.setString(6, ""+time);
    int i = statementCheckOrder.executeUpdate();
    return i > 0;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  closeConn();
  return false;
}


public static boolean acceptOrders(HttpServletRequest request) {
  try {
    conn = getConn();
    String queryGetMaxId = "SELECT MAX(u.id) AS user_id, MAX(s.id) AS
student_id, MAX(t.id) AS teacher_id " +
        "FROM `users` u, `students` s, `teachers` t";

    PreparedStatement statementGetMaxId =
conn.prepareStatement(queryGetMaxId);
    ResultSet resultSet = statementGetMaxId.executeQuery();
    int user_id;
    int student_id;
    int teacher_id;
    if(resultSet.next()) {
      user_id = resultSet.getInt("user_id");
      student_id = resultSet.getInt("student_id");
      teacher_id = resultSet.getInt("teacher_id");
    }
    else {return false;}
    List<Group> all_groups =
(List<Group>)request.getSession().getAttribute("all_groups");
    if(all_groups == null) {
      all_groups = DAO.getGroupList();
      request.getSession().setAttribute("all_groups", all_groups);
    }
    ArrayList<Order> accepted = new ArrayList();
    ArrayList<Order> orderList = (ArrayList<Order>)
request.getSession().getAttribute("orderList");
    Map<String, String[]> map = request.getParameterMap();
    for (Object entry : map.entrySet()) {
      Map.Entry<String, String[]> entry1 = (Map.Entry<String,
String[]>) entry;
      String name = entry1.getKey();
      String[] val = entry1.getValue();
      if(val.length < 1){
        continue;
      }
```

```
      if (val[0].equals("on")) {
        for (Order order : orderList) {
          if (order.getIdEmail().equals(name)) {
            if (order.getRole().equals("Студент")) {
              String[] str_group = map.get("group_" + name);
              for (Group group : all_groups) {
                if (group.getNumber().equals(str_group[0])) {
                  order.setGroup(Integer.parseInt(group.getId()));
                  break;
                }
              }
            }
            if (order.getRole().equals("Преподаватель")) {
              String[] id_directory = map.get("id_directory_" + name);
              String[] redactor = map.get("redactor_" + name);
              order.setIdDirectory(id_directory[0]);
              order.setRedactor(redactor[0].equals("on"));
            }
            accepted.add(order);
            break;
    }}}}
    if(accepted.size() == 0){ return false; }
    for(Order order:accepted) {
      user_id++;
      String queryNewUser = "INSERT INTO `users`(id, id_gmail, name,
email) VALUES(?, ?, ?, ?);";
      PreparedStatement statementNewUser =
conn.prepareStatement(queryNewUser);
      statementNewUser.setInt(1, user_id);
      statementNewUser.setString(2, order.getIdEmail());
      statementNewUser.setString(3, order.getName());
      statementNewUser.setString(4, order.getEmail());
      statementNewUser.executeUpdate();
      String queryNewPerson = "";
      if(order.getRole().equals("Студент")){
        student_id++;
        queryNewPerson = "INSERT INTO `students`(id, id_user,
id_group) VALUES(?, ?, ?);";
        PreparedStatement statementNewPerson =
conn.prepareStatement(queryNewPerson);
        statementNewPerson.setInt(1, student_id);
        statementNewPerson.setInt(2, user_id);
        statementNewPerson.setInt(3, order.getGroup());
        statementNewPerson.executeUpdate();
      }
      if(order.getRole().equals("Преподаватель")){
        teacher_id++;
        queryNewPerson = "INSERT INTO `teachers`(id, id_user,
id_directory, redactor) VALUES(?, ?, ?, ?);";
        PreparedStatement statementNewPerson =
conn.prepareStatement(queryNewPerson);
        statementNewPerson.setInt(1, teacher_id);
        statementNewPerson.setInt(2, user_id);
        statementNewPerson.setString(3, order.getIdDirectory());
        statementNewPerson.setBoolean(4, order.isRedactor());
        statementNewPerson.executeUpdate();
      }
```

```
      String queryUpdateOrder = "UPDATE `orders` SET processed = true
WHERE id = ?";
      PreparedStatement statementUpdateOrder =
conn.prepareStatement(queryUpdateOrder);
      statementUpdateOrder.setInt(1, order.getId());
      statementUpdateOrder.executeUpdate();
    }
    return true;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
    }
  }
  closeConn();
  return false;
}

public static boolean deleteOrders(HttpServletRequest request) {
  try {
    conn = getConn();
    ArrayList<Order> accepted = new ArrayList();
    ArrayList<Order> orderList = (ArrayList<Order>)
request.getSession().getAttribute("orderList");
    Map<String, String[]> map = request.getParameterMap();

    String allId = "";
    for (Object entry : map.entrySet()) {
      Map.Entry<String, String[]> entry1 = (Map.Entry<String,
String[]>) entry;
      String name = entry1.getKey();
      String[] val = entry1.getValue();
      if(val.length < 1){
        continue;
      }
      if (val[0].equals("on")) {
        for (Order order : orderList) {
          if (order.getIdEmail().equals(name)) {
            if(allId != ""){ allId = allId + ","; }
            allId = allId + order.getId();
            break;
    }}}}
    if(allId == ""){ return false; }
    String queryDeleteOrder = "DELETE FROM `orders` WHERE id in (" +
allId + ")";
    PreparedStatement statementDeleteOrder =
conn.prepareStatement(queryDeleteOrder);
    statementDeleteOrder.executeUpdate();
    return true;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
```

```
  }}
  closeConn();
  return false;
}

public static boolean updateUsers(HttpServletRequest request) {
  try {
    conn = getConn();

    List<Group> all_groups =
(List<Group>)request.getSession().getAttribute("all_groups");
    if(all_groups == null) {
      all_groups = DAO.getGroupList();
      request.getSession().setAttribute("all_groups", all_groups);
    }
    ArrayList<User> updated = new ArrayList();
    ArrayList<User> userList = (ArrayList<User>)
request.getSession().getAttribute("userList");
    Map<String, String[]> map = request.getParameterMap();
    for (Object entry : map.entrySet()) {
      Map.Entry<String, String[]> entry1 = (Map.Entry<String,
String[]>) entry;
      String name = entry1.getKey();
      String[] val = entry1.getValue();
      if(!(val != null && val.length > 0)){
        continue;
      }
      if (val[0].equals("on")) {
        for (User user : userList) {
          if (user.getIdEmail().equals(name)) {
            if (user instanceof Student) {
              String[] str_group = map.get("group_" + name);
              for (Group group : all_groups) {
                if (group.getNumber().equals(str_group[0])) {
                  ((Student)user).setIdGroup(group.getId());
                  break;
            }}}
            if (user instanceof Teacher) {
              String[] id_directory = map.get("id_directory_" + name);
              String[] redactor = map.get("redactor_" + name);
              ((Teacher)user).setIdDirectory(id_directory[0]);
              ((Teacher) user).setRedactor(false);
              if(redactor != null && redactor.length > 0) {
                ((Teacher)
user).setRedactor(redactor[0].equals("on"));
              }
            }
            String[] user_name = map.get("name_" + name);
            user.setName(user_name[0]);
            updated.add(user);
            break;
    }}}}
    if(updated.size() == 0){ return false; }
    for(User user:updated) {
      String queryUpdateUser = "UPDATE `users` SET name = ? WHERE id =
?;";
      PreparedStatement statementUpdateUser =
```

```
conn.prepareStatement(queryUpdateUser);
      statementUpdateUser.setString(1, user.getName());
      statementUpdateUser.setString(2, user.getId());
      statementUpdateUser.executeUpdate();

      if(user instanceof Student){
        String id_person = ((Student) user).getIdPerson();
        String id_group = ((Student)user).getIdGroup();
        String queryUpdatePerson = "UPDATE `students` SET id_group = ?
WHERE id = ?;";
        PreparedStatement statementUpdatePerson =
conn.prepareStatement(queryUpdatePerson);
        statementUpdatePerson.setInt(1, Integer.parseInt(id_group));
        statementUpdatePerson.setInt(2, Integer.parseInt(id_person));
        statementUpdatePerson.executeUpdate();
      }
      if(user instanceof Teacher){
        String id_person = ((Teacher) user).getIdPerson();
        String queryUpdatePerson = "UPDATE `teachers` SET id_directory
= ?, redactor = ? WHERE id = ?;";
        PreparedStatement statementUpdatePerson =
conn.prepareStatement(queryUpdatePerson);
        statementUpdatePerson.setString(1, ((Teacher)
user).getIdDirectory());
        statementUpdatePerson.setBoolean(2, ((Teacher)
user).isRedactor());
        statementUpdatePerson.setInt(3, Integer.parseInt(id_person));
        statementUpdatePerson.executeUpdate();
    }}
    return true;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  closeConn();
  return false;
}

public static boolean deleteUsers(HttpServletRequest request) {
  try {
    conn = getConn();
    ArrayList<User> accepted = new ArrayList();
    ArrayList<User> userList = (ArrayList<User>)
request.getSession().getAttribute("userList");
    Map<String, String[]> map = request.getParameterMap();

    String allId = "";
    String allIdEmail = "";
    for (Object entry : map.entrySet()) {
      Map.Entry<String, String[]> entry1 = (Map.Entry<String,
String[]>) entry;
      String name = entry1.getKey();
      String[] val = entry1.getValue();
      if(val.length < 1){
```

```
        continue;
      }
      if (val[0].equals("on")) {
        for (User user : userList) {
          if (user.getIdEmail().equals(name)) {
            if(allId != ""){ allId = allId + ","; }
            if(allIdEmail != ""){ allIdEmail = allIdEmail + ","; }
            allId = allId + user.getId();
            allIdEmail = allIdEmail + user.getIdEmail();
            break;
    }}}}
    if(allId == ""){ return false; }
    String queryDeleteUsers = "DELETE FROM `users` WHERE id in (" +
allId + ")";
    PreparedStatement statementDeleteUsers =
conn.prepareStatement(queryDeleteUsers);
    statementDeleteUsers.executeUpdate();

    String queryUpdateOrder = "UPDATE `orders` SET processed = false
WHERE id_email in (" + allIdEmail + ")";
    PreparedStatement statementUpdateOrder =
conn.prepareStatement(queryUpdateOrder);
    statementUpdateOrder.executeUpdate();

    return true;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  closeConn();
  return false;
}


public static boolean createNewGroup(HttpServletRequest request) {
  try {
    conn = getConn();
    User person = (Teacher)
request.getSession().getAttribute("person");
    String number = request.getParameter("number_new_group");
    String id_file = request.getParameter("id_file_new_group");
    String access_token =
(String)request.getSession().getAttribute("access_token");
    if(id_file.isEmpty() || id_file == ""){
      if(((Teacher) person).isRedactor()) {
        id_file = SpreadsheetsManager.create(access_token, number);
      }
      else { return false; }
    }
    String queryNewGroup = "INSERT INTO `groups`(number, link_id)
VALUES(?, ?)";
    System.out.println(queryNewGroup);
    PreparedStatement statementNewGroup =
conn.prepareStatement(queryNewGroup);
```

```
    statementNewGroup.setString(1, number);
    statementNewGroup.setString(2, id_file);
    int i = statementNewGroup.executeUpdate();
    return i > 0;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
  }}
  closeConn();
  return false;
}

public static boolean editGroup(HttpServletRequest request) {
  try {
    conn = getConn();
    User user = (User) request.getSession().getAttribute("user");
    String number = request.getParameter("number_edit_group");
    String id_group = request.getParameter("id_group");
    String id_file = request.getParameter("id_file_edit_group");

    String queryNewGroup = "UPDATE `groups` SET number = ?, link_id =
? WHERE id = ?";
    System.out.println(queryNewGroup);
    PreparedStatement statementNewGroup =
conn.prepareStatement(queryNewGroup);
    statementNewGroup.setString(1, number);
    statementNewGroup.setString(2, id_file);
    statementNewGroup.setInt(3, Integer.parseInt(id_group));
    int i = statementNewGroup.executeUpdate();
    return i > 0;
  }
  catch (Exception e) {
    Error.logError(e);
    if(e instanceof
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException){
      Logout.handle(request);
    }
  }
  closeConn();
  return false;
}


public static Student getStudentById(Connection conn, User user)
throws Exception {
  String queryCheckStudent = "SELECT s.id_group, g.number, da.*,
u.name AS author_name " +
      "FROM `students` s " +
      "LEFT JOIN `groups` g ON(s.id_group = g.id) " +
      "LEFT JOIN `directory_access` da ON(da.id_group = s.id_group) "
+
      "LEFT JOIN `teachers` t ON(da.id_teacher = t.id) " +
      "LEFT JOIN `users` u ON(u.id = t.id_user) " +
      "WHERE s.id_user = ? ORDER BY u.name";
```

```
  PreparedStatement statementCheckStudent =
conn.prepareStatement(queryCheckStudent);
  statementCheckStudent.setString(1, user.getId());

  ResultSet resultSet = statementCheckStudent.executeQuery();
  Student student = new Student();
  if (resultSet.next()) {
    String id_group = resultSet.getString("id_group");
    String number = resultSet.getString("number");
    student.setGroup(number);
    student.setIdGroup(id_group);
  }
  resultSet.beforeFirst();
  List<File> directory_list = new ArrayList();
  while (resultSet.next()) {
    String id_sub_directory = resultSet.getString("id_sub_directory");
    String author_name = resultSet.getString("author_name");
    File file = new File(id_sub_directory, author_name);
    directory_list.add(file);
  }
  student.setDirectoryList(directory_list);
  return student;
}


public static List<FileList> getFileListForStudent(User person) throws
Exception {
  List<FileList> big_file_list = new ArrayList();
  if (person instanceof Student) {
    Student student = (Student) person;
    FileList filelist = new FileList();
    filelist.setList(new ArrayList<File>());
    for (File dir : student.getDirectoryList()) {
      int j = big_file_list.size() - 1;
      boolean isNewAuthor = false;
      if (j >= 0) {
        FileList tmplist = big_file_list.get(j);
        isNewAuthor = (tmplist.getAuthorName() != null &&
!tmplist.getAuthorName().equals(dir.getAuthorName()));
      } else {
        j = 0;
        filelist.setAuthorName(dir.getAuthorName());
        big_file_list.add(filelist);
      }
      if (isNewAuthor) {
        filelist = new FileList();
        filelist.setAuthorName(dir.getAuthorName());
        filelist.setList(new ArrayList<File>());
        big_file_list.add(filelist);
        j++;
      }
      Map<String, String> map = getAccessFilelist(dir.getId());
      Object objArray = map.get("files");
      JSONArray array = (JSONArray) objArray;
      int size = array.size();
      int i = 0;
```

```
      while (i < size) {
        Object objEl = array.get(i);
        JSONObject jsonObject = (JSONObject) objEl;
        File file = new File();
        file.setId(jsonObject.get("id").toString());
        file.setMimeType(jsonObject.get("mimeType").toString());
        //System.out.println(file.getMimeType());
        file.genLink();
        file.setName(jsonObject.get("name").toString());
        file.setAuthorName(dir.getAuthorName());
        if(file.getLink() != null) {
          big_file_list.get(j).getList().add(file);
        }
        else{
          int l = 1;
        }
        i++;
  }}}
  return big_file_list;
}


public static List<File> getFileListForTeacher(User person) throws
Exception {
  List<File> filelist = new ArrayList();
  if (person instanceof Teacher) {
    Teacher teacher = (Teacher) person;
    Map<String, String> map =
getAccessFilelist(teacher.getIdDirectory());
    Object objArray = map.get("files");
    JSONArray array = (JSONArray) objArray;
    int size = array.size();
    int i = 0;
    String all_id = "";
    while (i < size) {
      Object objEl = array.get(i);
      JSONObject jsonObject = (JSONObject) objEl;
      File file = new File();
      file.setGroupList(new ArrayList<Group>());
      String id = jsonObject.get("id").toString();
      file.setId(id);
      if (all_id != "") {
        all_id = all_id + ",";
      }
      all_id = all_id + "'" + id + "'";
      file.setMimeType(jsonObject.get("mimeType").toString());
      //System.out.println(file.getMimeType());
      file.genLink();
      file.setName(jsonObject.get("name").toString());
      filelist.add(file);
      i++;
    }

    if(all_id != "") {
      String queryCheckGroups = "SELECT g.id, g.number, g.link_id,
da.id_sub_directory FROM `directory_access` da" +
```

```
          " JOIN `groups` g ON (g.id = da.id_group) WHERE
da.id_sub_directory IN (" + all_id + ")";

      conn = getConn();
      PreparedStatement statementCheckGroups =
conn.prepareStatement(queryCheckGroups);
      ResultSet resultSet = statementCheckGroups.executeQuery();
      ArrayList<Group> list_group = new ArrayList();
      while (resultSet.next()) {
        String id_sub_directory =
resultSet.getString("id_sub_directory");
        String number = resultSet.getString("number");
        String id = resultSet.getString("id");
        String link_id = resultSet.getString("link_id");
        Group group = null;
        try {
          if (Integer.parseInt(id) <= list_group.size() - 1) {
            group = list_group.get(Integer.parseInt(id));
          }
        }
        catch (Exception e) {
          Error.logError(e);
        }
        if (group == null) {
          group = new Group();
        }
        group.setId(id);
        group.setNumber(number);
        group.setLinkId(link_id);
        for (File file : filelist) {
          if (file.getId().equals(id_sub_directory)) {
            file.getGroupList().add(group);
            break;
          }
      }}
      closeConn();
  }}
  return filelist;
}


public static Map<String, String> getAccessFilelist(String id_dir) {
  Map<String, String> map = null;
  try {
    Client client = new Client();
    HttpGet get = new HttpGet(client.getUrlForGetFileList(id_dir));
    CloseableHttpClient httpclient = HttpClients.createDefault();
    CloseableHttpResponse httpResponse = httpclient.execute(get);
    map = ResponseParser.handleResponse(httpResponse);
  }
  catch (Exception e) {Error.logError(e); }
  return map;
}


public static void deleteOneDirectoryAccess(String id_dir, String
id_group) {
```

```
  try {
    conn = getConn();
    String queryCheckStudent = "DELETE FROM `directory_access` WHERE
id_group = ? AND id_sub_directory = ?";
    PreparedStatement statementCheckStudent =
conn.prepareStatement(queryCheckStudent);
    statementCheckStudent.setInt(1, Integer.parseInt(id_group));
    statementCheckStudent.setString(2, id_dir);
    statementCheckStudent.execute();
  }
  catch (Exception e) {Error.logError(e); }
  closeConn();
}


public static void addOneDirectoryAccess(String id_dir, String
id_group, String id_teacher) {
  try {
    conn = getConn();
    String queryCheckStudent = "INSERT INTO
`directory_access`(id_group, id_sub_directory, id_teacher) VALUES(?,
?, ?)";
    PreparedStatement statementCheckStudent =
conn.prepareStatement(queryCheckStudent);
    statementCheckStudent.setInt(   1, Integer.parseInt(id_group));
    statementCheckStudent.setString(2, id_dir);
    statementCheckStudent.setInt(   3, Integer.parseInt(id_teacher));
    statementCheckStudent.execute();
  }
  catch (Exception e) {Error.logError(e); }
  closeConn();
}


public static List<Group> getGroupList() {
  List<Group> list = new ArrayList();
  try {
    conn = getConn();
    String queryCheckStudent = "SELECT * FROM `groups`";
    PreparedStatement statementCheckStudent =
conn.prepareStatement(queryCheckStudent);
    ResultSet resultSet = statementCheckStudent.executeQuery();
    while (resultSet.next()) {
      Group group = new Group();
      group.setId(resultSet.getString("id"));
      group.setNumber(resultSet.getString("number"));
      group.setLinkId(resultSet.getString("link_id"));
      list.add(group);
  }}
  catch (Exception e) {Error.logError(e); }
  closeConn();
  return list;
}


public static ArrayList<Order> getOrderList(){
  ArrayList<Order> list = new ArrayList();
```

```
  try {
    conn = getConn();
    String queryCheckOrders = "SELECT * FROM `orders` WHERE NOT
processed ORDER BY time desc";
    PreparedStatement statementCheckOrders =
conn.prepareStatement(queryCheckOrders);
    ResultSet resultSet = statementCheckOrders.executeQuery();
    while (resultSet.next()){
      Order order = new Order(resultSet);
      list.add(order);
  }}
  catch (Exception e) {Error.logError(e); }
  closeConn();
  return list;
}


public static ArrayList<User> getUserList(){
  ArrayList<User> list = new ArrayList();
  try {
    conn = getConn();
    String queryCheckOrders = "SELECT u.id as id_user, u.name,
u.email, u.id_gmail, COALESCE(s.id,t.id) as id_person," +
        "COALESCE(t.redactor,false) as redactor, \n" +
        "case when s.id is null then 'Преподаватель' else 'Студент'
end as role, \n" +
        "g.number as group_number,\n" +
        "case when s.id is null then t.id_directory else g.link_id end
as id_link\n" +
        "FROM `users` u left join `students` s on(u.id = s.id_user)
left join `teachers` t on(u.id = t.id_user) \n" +
        "left join `groups` g\n" +
        "on(s.id_group = g.id)\n" +
        "ORDER BY role desc, g.number, u.name";
    PreparedStatement statementCheckOrders =
conn.prepareStatement(queryCheckOrders);
    ResultSet resultSet = statementCheckOrders.executeQuery();
    while (resultSet.next()){
      User user;
      if(resultSet.getString("role").equals("Студент")){ user = new
Student(resultSet); }
      else if(resultSet.getString("role").equals("Преподаватель")){
user = new Teacher(resultSet); }
      else{ user = new User(resultSet); }
      list.add(user);
  }}
  catch (Exception e) {Error.logError(e);}
  closeConn();
  return list;
}


public static ArrayList getStudentList(String id_group){
  ArrayList list = new ArrayList();
  try {
    conn = getConn();
    String queryCheckOrders = "SELECT u.name FROM `students` s JOIN
```

```
`users` u ON (s.id_user = u.id) " +
        "WHERE s.id_group = ? Order by u.name";
    PreparedStatement statementCheckOrders =
conn.prepareStatement(queryCheckOrders);
    statementCheckOrders.setInt(1, Integer.parseInt(id_group));
    ResultSet resultSet = statementCheckOrders.executeQuery();
    while (resultSet.next()){
      list.add(resultSet.getString("name"));
    }
  }
  catch (Exception e) {Error.logError(e);}
  closeConn();
  return list;
}



public static boolean deleteGroup(String id_group){
  try {
    conn = getConn();
    String queryDeleteGroup = "DELETE FROM `groups` WHERE id = ?";
    PreparedStatement statementDeleteGroup =
conn.prepareStatement(queryDeleteGroup);
    statementDeleteGroup.setInt(1, Integer.parseInt(id_group));
    statementDeleteGroup.execute();
    closeConn();
    return true;
  }
  catch (Exception e) {
    Error.logError(e);
    closeConn();
    return false;
}}}
```

ДОДАТОК Е

КОД СТОРІНКИ ОСОБИСТОГО КАБІНЕТУ

```
<%@ page language="java" contentType="text/html;charset=UTF-8" %>
<html>
<head>
<title>Журнал | Личный кабинет</title>
<link rel="stylesheet" href="design/lk.css">
<link rel="icon" href="design/logo.png">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<%@ page import = "java.util.List"     %>
<%@ page import = "java.util.ArrayList"   %>
<%@ page import = "com.inside.DAO"     %>
<%@ page import = "com.journal.model.*"   %>
<%
Object obj = request.getSession().getAttribute("user");
if (obj == null || !(obj instanceof User)) {
 response.sendRedirect("http://myjournal.com.ua/auth");
}
else{
 User user = (User)obj;
%>
   <script type='text/javascript'>
      var loc = location.href;
      if(loc.indexOf('#')>0){
         location.href = 'http://myjournal.com.ua/lk';
      }
   </script>
   <div class = "container header">
    <div id = "userEmail">
     <span class = "titleText">Вы:</span>
     <%= user.getEmail() %>
    </div>
    <div class = "container">
     <a href = "http://myjournal.com.ua/lk"><img id = "userPicture" src
= <%= user.getPicture() %> ></a>
       <span style='margin:0px 10px'><%= user.getName() %></span>
    </div>
    <div><a href =
"http://myjournal.com.ua?action=logout">Выйти</a></div>
   </div>
   <div class = "container_lk">
   <%
   User person = (User)request.getSession().getAttribute("person");
   if(person instanceof Teacher) {
%>
  <script type='text/javascript'>
   function add_directory_access(id_dir){
    var group = document.getElementById("new_group_for_" +
id_dir).value;
    if(group != null && group != ""){
     location.href = "http://myjournal.com.ua/lk?action
```

```
=add_directory_access&id_dir="+id_dir+"&group=" + group;
     }
    }
  </script>
<%
    List<Group> all_groups =
(List<Group>)request.getSession().getAttribute("all_groups");
    if(all_groups == null) {
     all_groups = DAO.getGroupList();
     request.getSession().setAttribute("all_groups", all_groups);
    }
    Teacher teacher = (Teacher)person;
    String action = request.getParameter("action");
   if(action != null){
   String id_dir   = request.getParameter("id_dir");
   String id_group = request.getParameter("id_group");
   String number = request.getParameter("group");

   if(action.equals("delete_directory_access") && id_group != null &&
id_dir != null){
    DAO.deleteOneDirectoryAccess(id_dir, id_group);
   }
   else {
    if(action.equals("add_directory_access") && number != null &&
id_dir != null){
     id_group = "";
     for (Group group : all_groups) {
      if(group.getNumber().equals(number)){
       id_group = group.getId();
      }
     }
     if(id_group != ""){
      DAO.addOneDirectoryAccess(id_dir, id_group,
teacher.getIdPerson());
   }}}
   response.sendRedirect("http://myjournal.com.ua/lk");
  }
  File root_dir = new File();
  root_dir.setId(teacher.getIdDirectory());
  root_dir.setMimeType("application/vnd.google-apps.folder");
  root_dir.genLink();

  out.println("<div class = 'container_titles'>");
  out.println("<div class='title_block'><a href = '" +
root_dir.getLink() + "' target='_blank'>Ваша корневая пап-
ка</a></div>");
  out.println("<div class='title_block'><a href =
'http://myjournal.com.ua/all_groups'>Группы</a></div>");
  if(teacher.isRedactor()){
   out.println("<div class='title_block'><a href =
'http://myjournal.com.ua/all_users'>Пользователи</a></div>");
   out.println("<div class='title_block'><a href =
'http://myjournal.com.ua/all_new_orders'>Новые заявки на создание уче-
тных записей</a></div>");
    }
    out.println("</div>");
    out.println("<div class = 'container_titles'>");
```

```
  out.println("<div class='title_block'><b>Опубликованные дочерние па-
пки:</b></div>");
  out.println("</div></div>");
    out.println("<div class='container_blocks'>");
  List<File> files = DAO.getFileListForTeacher(person);
  for (File file : files) {
   if(!(file.getMimeType().equals("application/vnd.google-
apps.folder"))) {
     continue;
   }
   out.println("<div class='one_author_block'>");
   out.println("<div class='dir_container container_a'>");
   out.println("<a href = '" + file.getLink() + "' target='_blank'>" +
file.getName() + "</a>");
   List<Group> group_list = file.getGroupList();
   out.println("<div class = 'group_list'><b>Список групп:</b>");
   String all_dir_id = "";
   for (Group group : group_list) {
    out.println("<div class = 'group_row'>" + group.getNumber() + " "
+
    "<a href='http://myjournal.com.ua/lk?action
=delete_directory_access&" + "id_dir=" + file.getId() + "&id_group=" +
group.getId() + "'>" + "<img src = 'design/delete.png' class =
'img_18'></a></div>");
    all_dir_id = all_dir_id + " " + group.getId() + " ";
   }
   out.println("<div class = 'group_row' style = 'margin: 10px
auto'><input type = 'text' id = 'new_group_for_" + file.getId() + "'
list = 'all_groups_for_" + file.getId() + "'>");
   out.println("<datalist id = 'all_groups_for_" + file.getId() +
"'>");
   for (Group group : all_groups) {
    if(all_dir_id.indexOf(group.getId()) == -1) {
     out.println("<option value='" + group.getNumber() + "' />");
    }
   }
   out.println("</datalist>");
   out.println("<img src = 'design/add.png' class = 'img_18' onclick =
'add_directory_access(`" + file.getId() + "`)'></div>");
   out.println("</div></div></div>");
  }
  out.println("</div>");
   }
 else {
  if(person instanceof Student){
   Student student = (Student)person;
   out.println("<h3>Эти ресурсы доступны вам, как студенту " +
student.getGroup() + " группы:</h3>");
   out.println("<div class='container_blocks'>");
   List<FileList> all_file_lists = DAO.getFileListForStudent(person);
   for (FileList file_list : all_file_lists) {
    out.println("<div class='one_author_block'>");
    out.println("<h4>" + file_list.getAuthorName()+"</h4>");
    List<File> files = file_list.getList();
    out.println("<div class='container_a'>");
    for (File file : files) {
     out.println("<a href = '" + file.getLink() + "' target='_blank'>"
```

```
+ file.getName() + "</a><br>");
    }
    out.println("</div>");
    out.println("</div>");
   }
   out.println("</div>");
  }
  else {
%>
   <div id = "new_order">
    <p class = "new_order_p">Ваша учетная запись еще не создана.</p>
    <p class = "new_order_p">Но(!) Вы можете оставить заявку на добав-
ление учетной записи, её рассмотрят в ближайшее время.</p>
    <form method = "post" action =
"http://myjournal.com.ua/new_order">
     <div class = "new_order_row_block">
      <span >Я:</span>
      <span>
       <p><input type = "radio" name = "role" value = "Студент"
checked>Студент</p>
       <p><input type = "radio" name = "role" value = "Преподава-
тель">Преподаватель</p>
      <span>
     </div>
     <textarea name = "coment" placeholder = "Тут может быть ваш ком-
ментарий" class = "new_order_coment"></textarea>
     <input type="submit" name = "submit_coment" value = "Оставить за-
явку" class = "new_order_submit">
    </form>
   </div>
<%
}}}
%>
</body>
</html>
```