

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

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення мультимедійних та інформаційно-пошукових систем»**

спеціальності 121 Інженерія програмного забезпечення

**на тему: «Вебзастосунок для ведення шкільних журналів та учнівських
щоденників»**

Виконав:

студент IV курсу, групи КП-91

Кінаш Дарій Олегович _____

Керівник:

Доцент кафедри ПЗКС, к.т.н., доцент,

Люшенко Леся Анатоліївна _____

Консультант з нормоконтролю:

Доцент кафедри ПЗКС, к.т.н., доцент,

Онай Микола Володимирович _____

Рецензент:

Доцент кафедри СПСКС, к.т.н., доцент,

Тарасенко-Клятченко Оксана Володимирівна _____

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення
мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2022 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Кінашу Дарію Олеговичу

1. Тема проєкту «Вебзастосунок для ведення шкільних журналів та учнівських щоденників», керівник проєкту Люшенко Леся Анатоліївна, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету від «31» травня 2023 р. №2107-с
2. Термін подання студентом проєкту «16» червня 2023 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
 - аналіз наявних рішень поставленого завдання;
 - обґрунтування вибору засобів реалізації;
 - розроблення вебзастосунка “SDTT”;
 - аналіз розробленого застосунку.
5. Перелік обов’язкового графічного матеріалу:
 - діаграма варіантів використання застосунку для учня (креслення);
 - діаграма варіантів використання застосунку для вчителя (креслення);
 - діаграма варіантів використання застосунку для батьків (креслення);
 - діаграма варіантів використання застосунку для керівництва школи (креслення);
 - архітектура вебзастосунку (креслення);
 - логічна схема бази даних (креслення);
 - фізична схема бази даних (креслення).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2022	
2.	Розроблення та узгодження технічного завдання	25.11.2022	
3.	Розроблення структури вебзастосунку	15.12.2022	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2022	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2023	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2023	
7.	Програмна реалізація вебзастосунку	10.03.2023	
8.	Тестування вебзастосунку	17.03.2023	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2023	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2023	
11.	Підготовка графічної частини дипломного проєкту	21.04.2023	
12.	Оформлення документації дипломного проєкту	26.05.2023	

Студент

Дарій КІНАШ

Керівник проєкту

Леся ЛЮШЕНКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений створенню вебзастосунку для управління навчальним процесом онлайн в шкільному навчальному закладі, а саме: управління викладацькою діяльністю вчителів, освітнім процесом учнів та організацією комунікації між вчителями, батьками та учнями.

У роботі виконано порівняльний аналіз існуючих рішень для організації навчання онлайн, визначено вимоги до застосунку, обґрунтовано вибір технологій для реалізації, описано загальну структуру, архітектуру та особливості реалізації застосунку.

Вебзастосунок розроблений у формі вебсайту з динамічними сторінками, де передбачено реєстрацію користувачів, управління розкладом, ведення журналів та щоденників, моніторинг успішності учнів та забезпечення комунікації між батьками, вчителями та учнями. Інформаційна безпека вебзастосунку реалізована за рахунок розподілу прав доступу між ролями користувачів. Роль керівництва школи передбачає реєстрацію користувачів та керування діяльністю вчителів та навчанням учнів. Роль вчителя передбачає ведення журналів та щоденників, перегляду успішності учнів та комунікацію з учнями та батьками. Ролі учня та батька передбачають перегляд розкладу, завдань, оцінок та успішності учня.

У даному дипломному проєкті було розроблено архітектуру вебзастосунку, модулі для забезпечення функціональності, доступної користувачам, а також графічні елементи та дизайн вебсторінок.

ABSTRACT

This diploma project is dedicated to the development of a web application for management of school journals and student diaries.

The paper includes a comparative analysis of existing solutions for online learning organization, determination of application requirements, justification of technology selected for implementation, description of the overall structure, architecture, and implementation specifics of the application.

The web application is designed as a website with dynamic pages intended for organizing the online learning process, including registration and management of teachers, parents, and students, creating schedules, maintaining journals and diaries, viewing students' performance, and facilitating communication between parents and the homeroom teacher, as well as between students and the homeroom teacher. Information security of the web application is ensured through access control rights distribution among user roles. The role of school's executive involves registering and managing teachers, students, and parents, adding and managing subjects, classes, and creating schedules. The teacher role involves maintaining journals and diaries, viewing students' performance, and communicating with students and parents. The student and parent roles involve viewing schedules, assignments, grades, and student performance.

This diploma project includes the development of the web application architecture, modules to provide functionality accessible to users, as well as graphical elements and web page design.

ДП.045440-01-90 Вебзастосунок для ведення шкільних журналів та
учнівських щоденників. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	5	
	Технічне завдання		
ДП.045440-03-81	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	62	
	Пояснювальна записка		
ДП.045440-04-51	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	4	
	Програма та методика тестування		
ДП.045440-05-34	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	11	
	Керівництво користувача		
ДП.045440-06-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Варіанти використання застосунку для учня. Діаграма варіантів використання		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Варіанти використання застосунку для вчителя. Діаграма варіантів використання		
ДП.045440-08-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Варіанти використання застосунку для батьків. Діаграма варіантів використання		
ДП.045440-09-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Варіанти використання застосунку для керівництва школи. Діаграма варіантів використання		
ДП.045440-10-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Архітектура вебзастосунку. UML діаграма		
ДП.045440-11-99	Вебзастосунок для ведення шкільних журналів та учнівських щоденників.	1	
	Логічна схема бази даних. ER діаграма		

[illegible]

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ВЕДЕННЯ ШКІЛЬНИХ ЖУРНАЛІВ
ТА УЧНІВСЬКИХ ЩОДЕННИКІВ

Технічне завдання

ДП.045440-02-91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дарій КІНАШ

ЗМІСТ

1. Найменування та галузь застосування.....	3
2. Підстава для розроблення	3
3. Призначення розробки.....	3
4. Вимоги до програмного продукту	3
5. Вимоги до проєктної документації	4
6. Етапи проєктування	5
7. Порядок тестування розробки	5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для ведення шкільних журналів та учнівських щоденників.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для забезпечення закладів середньої освіти онлайн засобами для організації онлайн навчання, а саме засобом ведення шкільних журналів та учнівських щоденників та засобом комунікацій між батьками та вчителями і між учнями та вчителями.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

- Можливість керівництва школи реєструвати вчителів, учнів та батьків.
- Можливість керівництва школи створювати та редагувати розклад.
- Відмічання вчителем присутніх.
- Виставлення вчителем оцінок.
- Задання вчителем домашнього завдання.

- Перегляд учнем та батьками розкладу, завдань та оцінок учня.
- Перегляд вчителем свого розкладу.
- Можливість переглядати показники успішності учня.
- Можливість комунікації між учнями та вчителем та між батьками та вчителем за допомогою чату.

Додаткові вимоги:

- Реалізація клієнт-серверної архітектури.
- Реалізація REST API для комунікації клієнта та сервера.
- Безпека та конфіденційність даних.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- 1) пояснювальна записка;
- 2) програма та методика тестування;
- 3) керівництво користувача;
- 4) креслення:
 - діаграма варіантів використання застосунку для учня;
 - діаграма варіантів використання застосунку для вчителя;
 - діаграма варіантів використання застосунку для батьків;
 - діаграма варіантів використання застосунку для керівництва школи;
 - архітектура вебзастосунку;
 - логічна схема бази даних;
 - фізична схема бази даних.

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....	16.11.2022
Розроблення та узгодження технічного завдання	27.11.2022
Розроблення структури вебзастосунку	15.12.2022
Розроблення дизайну сторінок та графічних елементів	03.02.2023
Програмна реалізація вебзастосунку	15.03.2023
Тестування вебзастосунку.....	07.04.2023
Підготовка матеріалів текстової частини проєкту	28.04.2023
Підготовка матеріалів графічної частини проєкту	12.05.2023
Оформлення технічної документації проєкту.....	25.05.2023

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2023 р.

**ВЕБЗАСТОСУНОК ДЛЯ ВЕДЕННЯ ШКІЛЬНИХ ЖУРНАЛІВ
ТА УЧНІВСЬКИХ ЩОДЕННИКІВ**

Пояснювальна записка

ДП.045440-03-81

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дарій КІНАШ

2023

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	3
ВСТУП	5
1. АНАЛІЗ НАЯВНИХ РІШЕНЬ ПОСТАВЛЕНОГО ЗАВДАННЯ.....	6
1.1. Загальні положення та аналіз предметної області.....	6
1.2. Аналіз аналогічних застосунків.....	7
1.3. Актуальність розробки вебзастосунка “SDTT”	11
1.4. Загальні вимоги до застосунку	12
1.5. Вимоги до безпеки застосунку	15
1.6. Висновки до розділу	19
2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ.....	21
2.1. Обґрунтування вибору мови програмування	21
2.2. Обґрунтування вибору бази даних	26
2.3. Висновки до розділу	29
3. РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКА “SDTT”	31
3.1. Загальна структура застосунка	31
3.2. Архітектура застосунка	33
3.3. Особливості реалізації	43
3.4. Висновки до розділу	48
4. АНАЛІЗ РОЗРОБЛЕНОГО ЗАСТОСУНКУ	49
4.1. Тестування застосунку	49
4.2. Порівняння розробки з наявними аналогами.....	55
4.3. Рекомендації щодо подальшого вдосконалення та підтримки	56
4.4. Висновки до розділу	57
ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	60
ДОДАТКИ.....	62

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Кросплатформеність – здатність програмного забезпечення працювати на різних операційних системах, таких як Windows, macOS чи Linux, з використанням одного набору коду.

Кросбраузерність – здатність вебзастосунку або вебсайту відображатися і працювати на різних веббраузерах, таких як Google Chrome, Mozilla Firefox, Safari та інші, забезпечуючи однакову функціональність та вигляд незалежно від браузера.

ICVO (Інформаційна система управління освітою) – вебпортал (платформа) для обліку та керування українськими закладами освіти онлайн.

REST – архітектурний стиль для розробки мережових застосунків, який використовує протокол HTTP і URI для обміну даними між клієнтом і сервером.

HTTP – протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від HyperText Transfer Protocol, тобто протокол передачі гіпертекстових документів.

URI (Uniform Resource Identifier) – це компактний рядок символів, що ідентифікує унікальний ресурс або місцезнаходження в мережі Інтернет.

DAO (Data Access Object) – це шаблон проектування програмного забезпечення, який використовується для забезпечення доступу до даних в базі даних або іншому джерелі даних.

JWT (JSON Web Token) – це відкритий стандарт (RFC 7519) для безпечного обміну даними в форматі JSON між двома сторонами, що використовується для автентифікації та авторизації в розподілених системах.

API (Application Programming Interface) – набір правил та протоколів, що дозволяють різним програмним системам взаємодіяти між собою.

CORS (Cross-Origin Resource Sharing) – це механізм безпеки, що використовується в застосунках, що взаємодіють через мережу, для контролю доступу до ресурсів застосунку, до яких намагається доступитися застосунок з іншим доменним ім'ям.

SPA (Single-Page Application) – вебзастосунок, що складається з одної сторінки, що завантажується при першому запиті до ресурсу, та використовує динамічне оновлення вмісту сторінки за допомогою JavaScript коду.

СУБД – система управління базами даних, призначена для організації, збереження, управління та доступу до баз даних.

ACID (Atomicity, Consistency, Isolation, Durability) – набір властивостей, які визначають стандартизовану модель транзакцій в базах даних для забезпечення збереження цілісності даних під час виконання транзакцій.

ORM (Object-Relational Mapping) – підхід до взаємодії з реляційними базами даних, де класи та об'єкти програми відображаються на таблиці та рядки бази даних.

ODM (Object-Document Mapping) – підхід до взаємодії з документними базами даних, де класи та об'єкти програми відображаються на колекції документів чи документи бази даних.

MQL (MongoDB Query Language) – мова запитів, яка використовується для взаємодії з базою даних MongoDB.

Cypher – мова запитів, що використовується для маніпулювання даними в графовій базі даних Neo4j.

SQL (Structured Query Language) – стандартизована мова програмування, призначена для керування та взаємодії з реляційними базами даних.

ВСТУП

Завдяки розвитку ІТ все більше сфер людської діяльності переходить у онлайн формат, що забезпечує гнучкий графік роботи та відсутність прив'язаності до певного місцезнаходження. Як зазначається в документі МОН України “Деякі питання організації дистанційного навчання”, дистанційне навчання можна організовувати для осіб, що не мають змоги відвідувати заклад освіти чи потребують індивідуальний графік навчання [1], що уможливорює організацію онлайн освіти в Україні.

Пріоритетність даного напрямку розвитку освіти в першу чергу пов'язана з відсутністю можливості викладачів та учнів відвідувати очні заняття через російсько-українську війну, пандемії, природні катаклізми та надзвичайні ситуації, тощо. Онлайн освіта дозволяє не тільки не переривати навчальний процес, а й забезпечити умови доступу до якісного навчання, яке існує в Україні та у всьому світі. Тож впровадження онлайн навчання дозволить збільшити доступність до якісної освіти, покращить продуктивність навчання учнів шляхом забезпечення зручного часу навчання та вибору найкращих джерел знань. Окрім цього, онлайн навчання дозволяє зекономити державні ресурси на підтримку закладів освіти та час і кошти на переміщення здобувачів освіти та викладачів від дому до освітніх закладів.

Метою даного дипломного проєкту є створення вебзастосунку для управління навчальним процесом онлайн в шкільному навчальному закладі, а саме: управління викладацькою діяльністю вчителів, освітнім процесом учнів та організація комунікації між вчителями, батьками та учнями. Вебзастосунок розроблений у формі вебсайту з динамічними сторінками, де передбачено: реєстрацію користувачів, управління розкладом, ведення журналів та щоденників, моніторинг успішності учнів та забезпечення комунікації між батьками, вчителями та учнями.

1. АНАЛІЗ НАЯВНИХ РІШЕНЬ ПОСТАВЛЕНОГО ЗАВДАННЯ

1.1. Загальні положення та аналіз предметної області

Ринок застосунків для навчання онлайн розвивається дуже стрімко. Консалтингова компанія з дослідження ринків «Global Market Insights» стверджує, що глобальний ринок застосунків електронної освіти подолав відмітку в 315 мільярдів доларів собівартості у 2021 році, та прогнозує 20% сукупного річного темпу росту в період з 2022 по 2028 роки [2]. Так для організації навчального процесу онлайн в Україні до застосунку “Єдина школа” у період з 2018 по 2023 рік приєдналися більше 2600 навчальних закладів [3], а до застосунку “e school” – 4904 навчальні заклади [4].

На сьогодні в Україні для організації онлайн навчання найпопулярнішими є наступні вебзастосунки: “Нові знання”, “Electronic Journal”, “e schools”, “Єдина Школа”. Основними засобами цих застосунків є:

- Реєстрація та керування користувачами.
- Управління розкладом.
- Ведення журналів та щоденників.
- Ведення обліку успішності учня.
- Комунікативні чати.

Особливо важливими в навчальному процесі онлайн є комунікація між вчителем, учнем та батьками. Основними інструментами для цього є програмне забезпечення, яке відтворює шкільний журнал, учнівський щоденник та надає середовище онлайн спілкування у вигляді месенджера, чату, тощо. Основними завданнями такого програмного забезпечення є:

- Формування розкладу навчання.
- Контроль присутності учнів.
- Оцінка рівня знань та вмінь.
- Контроль виконання домашнього завдання.

- Організація комунікації між учнями, вчителями та батьками.
- Моніторинг успішності учнів.

Загалом наявні на ринку застосунки для організації онлайн навчання мають наступні проблеми:

- Відсутність засобів комунікації – відсутність можливості проводити онлайн навчання без зв'язку учнів та вчителів та батьків та вчителів.
- Відсутність засобів моніторингу успішності учнів – відсутність можливості переглядати середні бали учнів за певний період часу, як наслідок непроінформованість батьків про успішність учнів.
- Самостійна реєстрація чи відмова від реєстрації в системі застосунку – неможливість використання застосунку як єдиного електронного ресурсу.
- Невідповідність функціональності, доступної певним ролям, тим завданням, що вони виконують у навчальному процесі.

Актуальним питанням є створення вебзастосунку “SDTT” (School Diary and TimeTable) для ведення шкільних журналів, учнівських щоденників, формування розкладу навчання, створювання завдань та комунікації вчителів з учнями та батьками на єдиному електронному ресурсі при навчанні онлайн.

1.2. Аналіз аналогічних застосунків

Під час аналізу предметної області було проаналізовано вебзастосунки для організації та виконання навчального процесу: “Нові знання”, “*Electronic Journal*”, “*e schools*” та “Єдина Школа”.

Усі проаналізовані програмні продукти кросплатформені, виконані у формі вебзастосунку (кросбраузерні) та містять засоби для ведення шкільних журналів та учнівських щоденників онлайн. Розглянемо більш детально ці вебзастосунки.

Нові знання

Застосунок «Нові знання» має на меті забезпечити засоби керування закладами середньої освіти, зареєстрованих на платформі ІСУО (Інформаційна система управління освітою), а також забезпечити онлайн навчання цим закладам шляхом надання засобу ведення шкільних журналів, учнівських щоденників та засобу отримання статистичних звітів про успішність учнів.

Переваги:

- Наявні засоби для створення розкладу, заповнення шкільних журналів та учнівських щоденників.
- Базується на платформі ІСУО, що надає вчителям та керівництву закладу доступ до даних зареєстрованих закладів освіти та учнів.
- Можливість електронної реєстрації в будь-який заклад середньої освіти, підключений до платформи ІСУО.
- Можливість формувати робочі навчальні плани онлайн.
- Наявна база електронних підручників для вибору для програми закладу.

Недоліки:

- Застосунок базується на окремій платформі ІСУО, що спроектована для обслуговування всіх навчальних закладів України, та має на меті обслуговувати велику кількість закладів освіти, тривалість відгуку на запит до застосунку займатиме більше часу, ніж у застосунка, що має власну базу даних та обслуговує лише одну школу.
- Застарілий інтерфейс користувача.
- Застарілий модуль реєстрації користувачів (немає можливості при реєстрації користувача автоматично відправити йому дані для входу).

Electronic Journal

Застосунок «Electronic Journal» має на меті забезпечити організацію онлайн навчання за допомогою надання засобів ведення журналів та щоденників та засобу комунікації між вчителями, батьками та учнями.

Переваги:

- Застосунок розроблений на замовлення МОН України.
- Наявні засоби для створення розкладу та заповнення шкільних журналів та учнівських щоденників.
- Наявний засіб комунікації – чат між класним керівником та учнями та між класним керівником та батьками.

Недоліки:

- Задум проєкту не підходить для організації онлайн навчання закладів середньої освіти – користувач з роллю “вчитель” може самостійно додавати предмети та створювати для них розклад, що призводить до конфліктів з розкладом навчання.
- Для учнів та батьків немає можливості переглянути статистику успішності учня.

e schools

Застосунок «*e schools*» має на меті забезпечити організацію проведення навчання онлайн за допомогою створення окремого вебсайту для закладу освіти та надання засобів ведення журналів та щоденників та комунікації батьків, учнів та вчителів.

Переваги:

- Наявні засоби для створення розкладу та заповнення шкільних журналів та учнівських щоденників.
- Можливість створення вебсайту школи.
- Можливість збереження фотоальбомів та файлових архівів у окремому онлайн-сховищі.

- Наявний засіб комунікації для учнів та батьків.
- Наявний засіб перегляду успішності учня батьками.

Недоліки:

- Надміру складний інтерфейс
- Застосунок має на меті обслуговувати велику кількість закладів освіти, тому тривалість відгуку на запит до застосунку займатиме більше часу, ніж у застосунку, що обслуговує лише одну школу.

Єдина Школа

Застосунок «Єдина Школа» має на меті забезпечити заклади середньої освіти універсальним вебпорталом для комунікації вчителів та учнів, проведення онлайн дзвінків та конференцій, ведення журналів та щоденників та перегляду успішності учня.

Переваги:

- Наявні засоби для створення розкладу та заповнення шкільних журналів та учнівських щоденників.
- Наявний засіб комунікації для вчителів, учнів та батьків.
- Наявний засіб перегляду успішності учня батьками.
- Засіб для проведення онлайн конференцій та онлайн дзвінків.

Недоліки:

- Реєстрація батьків відбувається власноруч батьками, а не керівництвом школи, що може призвести до неправильно введених даних та появи небажаних користувачів.
- Задум проекту передбачає реєстрацією учнів батьками, а не керівництвом навчального закладу. Це може призвести до відсутності певних учнів у системі, а отже до виключення можливості користування електронним журналом як єдиним ресурсом, а також повної відсутності інформації про навчальний процес у незареєстрованих учнів під час онлайн навчання.

1.3. Актуальність розробки вебзастосунка “SDTT”

Проаналізувавши аналогічні застосунки для організації онлайн навчання, які включають ведення шкільних журналів та щоденників онлайн, можемо дійти висновку, що кожен із них має суттєві недоліки, а саме:

- Відсутність можливості перегляду успішності учня.
- Відсутність засобів комунікації між учнями та вчителями та між батьками та вчителями.
- Доступ вчителів до управління розкладом.
- Реєстрацію учнів не керівництвом навчального закладу.
- Некоректні (або ті, що вводять в оману) дані, які виникають під час самостійної реєстрації.
- Застарілий інтерфейс.

Враховуючи вище зазначене, актуальною є розробка програмного продукту у формі вебзастосунку “SDTT” з однозначним визначенням ролей, їх рівнів доступу з наступними функціональними можливостями:

- Можливість керівництва школи реєструвати вчителів, учнів та батьків.
- Можливість керівництва школи створювати та редагувати розклад.
- Відмічання вчителем присутніх.
- Виставлення вчителем оцінок.
- Задання вчителем домашнього завдання.
- Перегляд учнем та батьками розкладу, завдань та оцінок учня.
- Перегляд вчителем власного розкладу.
- Можливість переглядати показники успішності учня.
- Можливість комунікації між учнями та вчителем та між батьками та вчителем за допомогою чату.

1.4. Загальні вимоги до застосунку

Після проведення аналізу програмних продуктів та аналізу процесів ведення журналів та щоденників у закладах середньої освіти був створений загальний перелік вимог до вебзастосунку “SDTT”.

Вебзастосунок має складатися з клієнтської та серверної частини, тобто з вебсторінки доступної з браузера та REST сервера відповідно. У вебзастосунку передбачаються наступні ролі:

- Керівництво школи.
- Вчитель.
- Учень.
- Батьки.

Нижче визначені функціональні вимоги вебзастосунку для кожного модуля застосунку.

Модуль керівництва школи:

- Додавання та зміна даних вчителя, учня та батьків.
- Додавання класів.
- Додавання предметів.
- Прив'язка учня та вчителя до класу.
- Прив'язка батьків до учня.
- Прив'язка предметів до вчителя.
- Додавання та редагування уроку (пари) в розкладі для класу на вільну позицію (урок номер 1 – 8) буднього дня (понеділок – п'ятниця) парного чи непарного тижня та визначення для цього уроку предмету та вчителя (можливість зазначити класну кімнату, якщо навчання проводитиметься очно). Предмет, вчитель та класна кімната мають бути такими, що не конфліктують з розкладом інших класів у той самий час (тиждень, день та номер уроку).

Модуль вчителя:

- Перегляд свого розкладу роботи.
- Планування уроку на певне число (коли такий урок може відбутися за розкладом) з визначенням теми, типу уроку та заданням домашнього завдання.
- Контролювання та відмічання присутніх на уроці.
- Виставляння оцінки за домашнє завдання та роботу на уроці.
- Перегляд та надсилання повідомлень в учнівський та батьківський чати класу, керівником якого він є.
- Перегляд середніх балів учнів за розрізом: класний керівник, учитель з предмету за встановлені терміни.
- Перегляд розкладу уроків, призначених класів, типу та теми уроків та домашніх завдань за певне число поточного навчального року для класів, у яких він викладає предмет.
- Перегляд присутності учнів певного класу та їх оцінок за урок по своєму предмету за певне число поточного навчального року, якщо за таке число по розкладу є урок даного вчителя у даного класу.

Модуль батьків:

- Перегляд та надсилання повідомлень у батьківський чат класу.
- Перегляд розкладу своєї дитини (учня).
- Перегляд розкладу, домашніх завдань, оцінок та присутності своєї дитини (учня) за певне число поточного навчального року.
- Перегляд середніх балів своєї дитини (учня) з кожного предмету за встановлені терміни.

Модуль учня:

- Перегляд власного розкладу, типу та теми уроків, домашніх завдань, оцінок та присутності за певне число поточного навчального року.

- Перегляд та надсилання повідомлень в учнівський чат свого класу.
- Перегляд своїх середніх балів з кожного предмета за встановлені терміни.

Модуль авторизації та автентифікації:

- При запуску вебзастосунка у системі вже зареєстрований один користувач з роллю “керівництво школи”.
- При реєстрації на вказану адресу електронної пошти має приходити лист з кодом активації облікового запису користувача.
- Користувач може активувати обліковий запис, використавши код з попередньої вимоги.
- Користувач може відновити пароль від облікового запису, новий згенерований пароль має прийти на вказану при реєстрації електронну адресу.
- Користувач може увійти в систему за допомогою адреси електронної пошти та пароллю від облікового запису.
- Користувач може змінити свій пароль на новий.
- Користувач може вийти з облікового запису.

Нефункціональні вимоги вебзастосунка:

- Можливість безперебійного доступу до вебзастосунку 2000 користувачів, що одночасно знаходяться в системі, з часом відповіді на запити менше 10 секунд.
- Безпека та конфіденційність даних.

Вимоги до якості програмного забезпечення:

- Програмний продукт має пройти модульне та інтеграційне тестування.
- Серверна частина вебзастосунку має реалізовувати принцип REST.

- Перед збереженням у базі даних паролі користувачів повинні бути перетворені у рядок за допомогою хеш-функції без можливості зворотнього перетворення.

1.5. Вимоги до безпеки застосунку

Проаналізуємо ризики, загрози та сформулюємо вимоги безпеки вебзастосунку. Розглянемо наступні вразливості:

- відсутність валідації даних (на клієнтський та на серверній частинах);
- некоректна реалізація запитів до бази даних (конкатенація даних замість підстановки значень);
- збереження критичних даних без попереднього шифрування чи хешування (збереження паролів без хешування);
- неправильне налаштування CORS серверної частини (можливість надсилати запити з будь-якої адреси, а не тільки з клієнта);
- відсутність шифрування критичних даних при передачі їх між клієнтською та серверною частини (передача логіна та пароля при авторизації);
- не реалізовано принцип найменшого привілею (відсутність обмежень чи недостатні обмеження можливостей залежно від ролі користувача);
- виведення кодів помилок бази даних, сервера та клієнта користувачу;
- збереження конфіденційної інформації користувачів та інформації про внутрішню будову програми у файлах журналювання.

Вищезгадані вразливості, загрози та наслідки їх експлуатації для програмного забезпечення “SDTT”, що розроблюється, детально описані у табл. 1.

Ідентифікація ризиків та загроз безпеці вебзастосунка “SDTT”

№	Вразливість	Загроза	Наслідки
1	Відсутність валідації даних на клієнті.	Відправлення невалідних даних на сервер.	Непередбачувана робота сервера.
2	Відсутність валідації даних на сервері.	Спроба обробки невалідних даних на сервері.	Падіння сервера, запис некоректних даних у базу даних.
3	Некоректна реалізація запитів до бази даних (конкатинація даних замість підстановки значень).	Надання користувачем даних, що змінюють поведінку запиту до бази даних (SQL – ін’єкція у базу даних).	Видалення таблиць чи частини даних з таблиць, зміна даних чи витік конфіденційних даних.
4	Збереження критичних даних без попереднього шифрування чи хешування.	Доступ до незахищених конфіденційних даних.	Витік конфіденційних даних, отримання доступу до облікових записів.
5	Неправильне налаштування CORS серверної частини (можливість надсилати запити з будь-якої адреси, а не тільки з клієнта).	Отримання доступу до сервера та виконання запитів з будь-якого ресурсу (адреси).	Отримання зловмисниками даних для входу в обліковий запис та інших конфіденційних даних.
6	Відсутність шифрування конфіденційних даних при передачі їх між клієнтом та сервером.	Перехоплення логіна та пароля користувача при спробі входу в обліковий запис.	Отримання зловмисниками даних для входу в обліковий запис.

7	Не реалізовано принцип найменшого привілею (відсутність обмежень чи недостатні обмеження можливостей залежно від ролі користувача).	Використання певним типом користувачів того функціоналу, що за функціональними вимогами не мав би можливості бути використаним цим типом.	Використання функціоналу не за призначенням, нестабільність роботи застосунка, “нелегальні” операції над базою даних.
8	Виведення кодів помилок бази даних, сервера та клієнта користувачу.	Розкриття реалізації програмного продукту та його слабких місць.	Атаки на застосунок зі спробами вивести його з ладу чи викрасти дані.
9	Збереження конфіденційної інформації користувачів та інформації про внутрішню будову програми у файлах журналювання.	Доступ сторонніх осіб до файлів журналювання з конфіденційними даними.	Витік конфіденційної інформації та інформації про реалізацію програмного продукту.

Оскільки визначені наслідки неприпустимі, винесемо нефункціональні вимоги до безпеки застосунку у табл. 2, попередньо проаналізувавши вразливості та загрози та визначивши відповідні вимоги безпеки.

Таблиця 2

Обробка ризиків та загроз безпеці вебзастосунка “SDTT”

№	Вразливість	Загроза	Вимога безпеки
1	Відсутність валідації даних на клієнті.	Відправлення невалідних даних на сервер.	Валідація всіх полів вводу на клієнті.

2	Відсутність валідації даних на сервері.	Спроба обробки невалідних даних на сервері.	Валідація отриманих об'єктів у запитах на сервері.
3	Некоректна реалізація запитів до бази даних (конкатинація даних замість підстановки значень).	Надання користувачем даних, що змінюють поведінку запиту до бази даних (SQL – ін'єкція у базу даних).	Використання параметризованих запитів та валідація вхідних даних.
4	Збереження критичних даних без попереднього чи шифрування (збереження паролів без хешування).	Доступ до незахищених конфіденційних даних.	Збереження паролів у захешованому вигляді, шифрування даних.
5	Неправильне налаштування CORS серверної частини (можливість надсилати запити з будь-якої адреси, а не тільки з клієнта).	Отримання доступу до сервера та виконання запитів з будь-якого ресурсу (адреси).	Налаштування CORS таким чином, щоб лише довірений клієнт міг надсилати запити до сервера.
6	Відсутність шифрування конфіденційних даних при передачі їх між клієнтом та сервером.	Перехоплення логіна та пароля користувача при спробі входу в обліковий запис.	Шифрування конфіденційних даних при передачі по мережі.

7	Не реалізовано принцип найменшого привілею (відсутність обмежень чи недостатні обмеження можливостей залежно від ролі користувача).	Використання певним типом користувачів того функціоналу, що за функціональними вимогами не мав би можливості бути використаним цим типом.	Реалізація принципу найменшого привілею для всіх наявних ролей.
8	Виведення кодів помилок бази даних, сервера та клієнта користувачу.	Розкриття реалізації програмного продукту та його слабких місць.	Перехоплення повідомлень про помилки, їх обробка та виведення користувачу в узагальненому вигляді без розкриття деталей реалізації продукту.
9	Збереження конфіденційної інформації користувачів та інформації про внутрішню будову програми у файлах журналювання.	Доступ сторонніх осіб до файлів журналювання з конфіденційними даними.	Не записувати конфіденційну інформацію користувачів та інформації про внутрішню будову програми у файли журналювання.

Проаналізувавши ризики та загрози безпеці, сформовано вимоги до безпеки застосунку. Виконання цих вимог до безпеки застосунку гарантує: безперебійність роботи; збереження, безпеку даних; конфіденційність щодо інформації користувачів.

1.6. Висновки до розділу

З розвитком ІТ та появою чинників, що перешкоджають очному навчанню, виникла необхідність у розробці застосунку для організації онлайн

навчання. На ринку програмного забезпечення вже існують схожі застосунки, однак вони не пристосовані до реальних процесів у закладах середньої освіти, що може порушити процес навчання.

Для забезпечення конкурентноспроможності застосунку було проаналізовано предметну область та переваги і недоліки аналогічних застосунків. Серед основних проблем таких аналогів є: відсутність засобів комунікації, невідповідність функціональності, доступної певним ролям, тим завданням, що вони виконують у навчальному процесі та відсутність засобів перегляду успішності учнів.

Були визначені основні функціональні можливості застосунку, а також визначені функціональні вимоги застосунку, розбиті на модулі згідно з ролями користувачів: Керівництво школи, Вчитель, Учень та Батьки. Також було визначено нефункціональні вимоги та вимоги до безпеки застосунка шляхом аналізу ризиків і загроз безпеці програмного забезпечення.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Оскільки вебзастосунок буде використовувати клієнт-серверну архітектуру, для реалізації якої доступна велика кількість мов програмування та технологій. Проаналізуємо декілька поширених мов програмування та виберемо найбільш відповідну мову програмування та супутні технології (фреймворки, бібліотеки, тощо) для створення застосунку.

Враховуючи поставлені функціональні та нефункціональні вимоги до програмного продукту, визначимо характеристики мов програмування:

- кросплатформеність/кросбраузерність;
- довготривала підтримка мови програмування (версії мови програмування);
- підтримка об'єктно-орієнтованої парадигми програмування.

Розглянемо такі мови програмування: *C++*, *Java*, *TypeScript*.

C++

C++ – статично типізована високорівнева мова програмування загального призначення, що підтримує декілька парадигм програмування (процедурну, узагальнену та об'єктно-орієнтовану) [5]. Статична типізація дозволяє під час компіляції перевірити правильність типів даних та запобігти помилок під час виконання коду. *C++* є компільованою мовою, тобто код компілюється одразу в машинний код, що залежить від операційної системи, а отже скомпільована програма не є кросплатформною. Оскільки *C++* була створена у 1985 році, впродовж її розвитку до неї було додано безліч можливостей та написано велику кількість бібліотек та фреймворків. Мова відома своєю високою швидкодією, що дозволяє розробляти ефективні програми, тому *C++* має широке застосування у сферах розробки вбудованих систем, операційних систем, ігор, криптографічних програм, тощо.

Переваги мови C++:

- Висока швидкодія та ефективність.
- Підтримка об'єктно-орієнтованого та процедурного програмування.
- Можливість маніпулювати пам'яттю напряду, що дозволяє розробникам оптимізувати програми.
- Розширені можливості шаблонів та перевантаження операторів.
- Широкий вибір бібліотек та фреймворків, що розширюють функціональність мови.
- Можливість підключення коду на C.
- Підтримка багатопоточності.

Недоліки мови C++:

- Складність мови та велика кількість можливостей ускладнюють розробку та підтримку коду.
- Велика кількість стандартів та сумісність з попередніми версіями ускладнюють підтримку коду.
- Низький рівень абстракції та відсутність автоматичного звільнення пам'яті, що не використовується, можуть призвести до проблем з пам'яттю та безпекою.
- Велика кількість допустимих варіантів оголошень функцій можуть ускладнити читання та підтримку коду.
- Відсутність засобів для роботи з складними об'єктами може ускладнити розробку програми.
- Невелика кількість засобів для роботи з мережевим програмуванням порівняно з іншими мовами програмування.

Java

Java – це високорівнева об'єктно-орієнтована строго типізована мова програмування з автоматичним управлінням пам'яттю. Java компілюється в

байт-код, який може бути виконаний віртуальною машиною Java (JVM) на будь-якій операційній системі, що підтримує JVM, отже скомпільована програма є кросплатформною [6]. Java надзвичайно надійна за рахунок зворотної підтримки – кожна наступна версія мови підтримує попередню. Java широко використовується для розробки вебзастосунків, настільних застосунків, мобільних застосунків та програмного забезпечення для вбудованих систем. Оскільки програма компілюється та виконується віртуальною машиною, Java має незначні втрати в швидкодії порівняно з C++, але виграє у швидкодії порівняно з інтерпретованими мовами програмування. Здебільшого Java використовується для розробки вебзастосунків (особливо їх серверних частин, де використовується популярний фреймворк Spring), настільних застосунків, мобільних застосунків для ОС Android та програмного забезпечення для вбудованих систем.

Переваги мови Java:

- Кросплатформенність – можливість виконувати код на будь-якій платформі, яка підтримує віртуальну машину Java (JVM).
- Об'єктно-орієнтована парадигма – спрощує процес програмування і робить код більш структурованим.
- Автоматичне управління пам'яттю – розробнику не потрібно вручну виділяти та звільняти пам'ять.
- Багатопотоковість – дозволяє виконувати одночасно кілька операцій, що покращує швидкодію застосунків.
- Велика кількість стандартних бібліотек та фреймворків, що спрощує розробку застосунків.
- Надійність та стабільність завдяки зворотній сумісності версій.

Недоліки мови Java:

- Швидкість виконання: Java може бути повільнішою за мови нижчого рівня, такі як C++ і C.

- Використання пам'яті: Java потребує більше пам'яті для виконання програм, ніж мови нижчого рівня, як C++.
- Обмежені можливості для системного програмування: Java не дає безпосереднього доступу до різних операційних системних ресурсів.

TypeScript

TypeScript – компільована мова програмування, створена для розширення можливості мови JavaScript. TypeScript є зворотно сумісним з JavaScript, при компіляції транслюється в JavaScript, що є найпопулярнішою мовою програмування вебсторінок. Однак, на відміну від JavaScript, для запобігання недоліків пов'язаних з динамічною типізацією TypeScript має статичну типізацію, та як наслідок код компілюється перед виконанням замість інтерпретації під час виконання [7]. Як і JavaScript, TypeScript кросплатформна мова програмування. JavaScript широко використовується у мережевій розробці і є найбільш популярною мовою програмування для створення динамічних вебсторінок та для створення серверних застосунків. Найбільш популярні засоби розробки вебсторінок – фреймворки Angular та Vue.js та бібліотека React, тоді як для розробки вебсерверів використовується середовище Node.js.

Переваги TypeScript:

- Статична типізація дозволяє виявляти помилки на етапі компіляції, що сприяє більш високій надійності та стабільності коду.
- Підтримка класів, інтерфейсів, модулів та інших об'єктно-орієнтованих концепцій, що робить код більш структурованим та організованим.
- Підтримка `async/await` та інших механізмів, що дозволяють зручно працювати з асинхронним кодом.
- Розширена підтримка типів змінних та функцій, що дозволяє запобігати помилкам.

- Підтримка декількох парадигм програмування, включаючи об'єктно-орієнтовану та функціональну.
- Розширена підтримка ES6 та новішої версії стандарту JavaScript.

Недоліки TypeScript:

- Більші витрати часу на розробку коду порівняно з JavaScript, особливо на великих проектах.
- Проблеми зі сумісністю та інтеоперабельністю зі сторонніми бібліотеками та фреймворками, що не підтримують TypeScript.
- Відсутність можливості використовувати динамічну типізацію, що може бути більш зручною у певних випадках.
- Більш високий рівень абстракції може призвести до менш ефективного використання ресурсів порівняно зі звичайним JavaScript.

Обрані мови програмування

Враховуючи вимоги до мови програмування та необхідності розроблення двох частин застосунку – клієнтської та серверної частини, було обрано наступні мови програмування:

- Java версії 1.8 для написання серверної частини з використанням фреймворку Spring. Мова обрана завдяки кросплатформності, підтримки віртуальними машинами для вищих версій та об'єктно-орієнтованості. Фреймворк Spring був обраний як найпопулярніший засіб для розробки серверної частини на мові Java. TypeScript був відхилений через недостатню ефективність та невеликий вибір підтримуваних бібліотек та фреймворків. В свою чергу C++ був відхилений через складність та більші затрати під час розробки.
- Typescript для написання клієнтської частини з використанням фреймворку Angular. Мова обрана через те, що вона є надбудовою до найбільш популярної і найбільш пристосованої мови для розробки динамічних вебсторінок та розробки логіки вебсторінок в цілому –

JavaScript. Фреймворк Angular був обраний як найпопулярніший засіб розробки SPA на мовах JavaScript та TypeScript. C++ та Java були відкинуті через відповідно відсутність та недостатню кількість і застарілість функціоналу для розробки динамічних вебсторінок.

2.2. Обґрунтування вибору бази даних

Оскільки програмний продукт передбачає збереження даних користувачів та їх подальше використання, для цієї мети необхідно використати базу даних. Розглянемо декілька з найбільш поширених СУБД. Проаналізуємо їх переваги та недоліки, а також виберемо найбільш відповідну для створюваного застосунку.

Розглянемо наступні СУБД: *PostgreSQL*, *MongoDB* та *Neo4j*.

PostgreSQL

PostgreSQL – об’єктна реляційна база даних. Найкраще підходить для збереження структурованих даних у вигляді таблиць та гарантує цілісність даних та дотримання принципу ACID у транзакціях [8]. Як і більшість реляційних БД, PostgreSQL підтримує стандарт мови SQL. Оскільки PostgreSQL має відкритий код, є безкоштовною та має обширну документацію, у СУБД виросла велика користувацька база, що й призвело до популярності даної СУБД. У більшості популярних мов програмування, зокрема і у Java, є бібліотеки, що дозволяють взаємодіяти з СУБД як за допомогою звичайних SQL запитів, та і за допомогою механізмів ORM.

Переваги PostgreSQL:

- Ефективність збереження структурованих даних.
- Цілісність посилань (посилання foreign key не може посилатися на неіснуючий primary key).
- Відкритий код та безкоштовне використання.

- Підтримка стандартів SQL та ACID, що забезпечує передбачувану поведінку.
- Підтримка таких популярних форматів, як JSON.

Недоліки PostgreSQL:

- Складність проведення операцій з даними, приведеними до третьої нормальної форми і вище, у випадку необхідності отримувати або змінювати дані з багатьох таблиць.
- Обмежена підтримка порівняно з комерційними СУБД.
- Важкість горизонтального масштабування.
- Лише базовий текстовий пошук (в порівнянні з документними базами даних).

MongoDB

MongoDB – документно-орієнтована розподілена СУБД. На відміну від реляційних СУБД MongoDB використовує принцип NoSQL, зберігаючи дані у форматі BSON (формат похідний від JSON). Завдяки розподілу даних між кількома серверами MongoDB забезпечує горизонтальне масштабування шляхом створення розподілених кластерів даних [9]. Принцип NoSQL також надає можливість зберігати як структуровані, так і неструктуровані дані, що не є можливим у реляційних базах даних. У більшості популярних мов програмування є бібліотеки «клієнти», що дозволяють взаємодіяти з СУБД за допомогою механізмів ODM.

Переваги MongoDB:

- MongoDB дозволяє зберігати неструктуровані дані, що робить її гнучкою для розробки застосунків з невизначеними вимогами до зберігання даних.
- Зручна вертикальна та горизонтальна масштабованість.
- Збереження даних у похідному форматі від JSON значно спрощує розробку програмного забезпечення.

- Набагато більший функціонал текстового пошуку, ніж у інших СУБД.
- Недоліки MongoDB:
- Проблеми з цілісністю посилань (referential integrity). Проявляється у випадках, коли за певним ідентифікатором може не існувати відповідного об'єкта.
 - Відсутність транзакцій – MongoDB не підтримує транзакції в традиційному розумінні, хоча пропонує операції з блокуванням даних.
 - Не настільки ефективне використання дискового простору, як у реляційних СУБД.
 - Не найкраще рішення для збереження структурованих даних.

Neo4j

Neo4j – графова СУБД. Модель даних – граф – складається з вершин та ребер. Ця СУБД розроблена для зберігання нелінійних, зв'язаних даних, що можна описати у вигляді графа, тому не є найкращим рішенням для зберігання структурованих даних. Для забезпечення збереження цілісності даних використовуються транзакції, що реалізують принцип ACID. Для виконання операцій над графом розроблена декларативна мова Cypher. Архітектура СУБД підтримує розподілені системи, тому Neo4j легко масштабується горизонтально за допомогою шардингу [10].

Переваги Neo4j:

- Графова модель даних дозволяє зберігати та опрацьовувати складні зв'язки між даними більш ефективно ніж реляційні та документні СУБД.
- Швидкий доступ до даних через відсутність необхідності виконувати складні операції з'єднання таблиць.
- Дотримання принципу ACID при транзакціях.
- Легке горизонтальне масштабування.

Недоліки Neo4j:

- Графова модель не настільки ефективна для збереження структурованих даних, як реляційна модель.
- Обмежена кількість підтримуваних типів даних.
- Необхідність спеціальної оптимізації запитів для забезпечення ефективності та швидкодії.
- Необхідність використовувати спеціально розроблену мову запитів.

Обрана СУБД

Проаналізувавши сферу розробки програмного забезпечення та дані користувачів, що необхідно зберігати, можна зробити висновки, що ці дані структуровані і для їхнього збереження найкраще відповідає реляційна база даних PostgreSQL. Найбільший недолік реляційних баз даних – складність горизонтального масштабування – не є проблемою у даному випадку тому, що обсяг збережених даних не вимагатиме масштабування (програмний продукт призначений лише для одного закладу середньої освіти). Інші варіанти відхилені, оскільки графова модель даних не підходить для розроблюваного програмного забезпечення через збереження структурованих даних та складність написання запитів для предметної області, а документна, хоч і має свої переваги, може спричинити проблеми з цілісністю посилань.

2.3. Висновки до розділу

У даному розділі було розглянуто засоби збереження даних користувачів та реалізації клієнтської і серверної частин кросплатформного вебзастосунка. Після аналізу переваг та недоліків кожного засобу було обрано оптимальний варіант для кожної частини застосунку.

Для реалізації серверної частини було обрано мову програмування Java з використанням фреймворка Spring через швидкодію та кросплатформність

мови. TypeScript був відхилений через недостатню ефективність, а C++ був відхилений через складність та більші затрати під час розробки.

Для реалізації клієнтської частини було обрано мову програмування TypeScript через статичну типізацію, наявність безлічі спільних бібліотек та сумісність з мовою JavaScript та можливість реактивного програмування за допомогою фреймворка Angular. Мови C++ та Java були відкинуті через відповідно відсутність та недостатню кількість і застарілість функціоналу для розробки динамічних вебсторінок.

Для збереження даних було обрано реляційну СУБД PostgreSQL через необхідність збереження структурованих даних, важливість правильного проходження транзакцій та необхідність збереження цілісності посилань. Графова СУБД Neo4j була відхилена, оскільки графова модель даних не є найкращим рішенням для збереження структурованих даних, а документна СУБД MongoDB була відхилена через можливі проблеми з цілісністю посилань.

3. РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКА “SDTT”

3.1. Загальна структура застосунка

З точки зору ролевої моделі користувачів застосунок містить:

- Модуль керівництва школи – засіб реєстрації користувачів, засіб створення класів та предметів та засіб створення розкладу.
- Модуль вчителя – засіб перегляду розкладу, уроків за певне число, засіб для відмічання присутності та виставлення оцінок, засіб для комунікації та засіб перегляду успішності учнів.
- Модуль учня – засіб перегляду розкладу та уроків за певне число, засіб для комунікації вчитель – учень та засіб перегляду успішності.
- Модуль батьків – засіб перегляду розкладу та уроків за певне число, засіб для комунікації вчитель – батьки та засіб перегляду успішності.

На рис. 1-4 наведено діаграми варіантів використання застосунку за ролями користувачів.



Рис. 1. Діаграма варіантів використання застосунку для учня



Рис. 2. Діаграма варіантів використання застосунку для батьків



Рис. 3. Діаграма варіантів використання застосунку для вчителя



Рис. 4. Діаграма варіантів використання застосунку для керівництва школи

Застосунок складається з двох частин: клієнтської – для взаємодії з користувачем через браузер; серверної – для взаємодії з базою даних та клієнтською частиною.

3.2. Архітектура застосунка

Вебзастосунок реалізує клієнт-серверну архітектуру, де клієнт забезпечує взаємодію з користувачем, а сервер містить бізнес-логіку та зв'язує клієнта з базою даних [11]. Клієнт-серверна архітектура є найпопулярнішим способом реалізації вебзастосунків, оскільки вона має наступні переваги:

- Розділення функцій – можливість чітко розділити функції між клієнтом і сервером. Клієнт відповідає за представлення інтерфейсу користувачу та взаємодію з ним, тоді як сервер забезпечує обробку запитів і доступ до ресурсів, таких як база даних.

- Безпека – можливість контролювати доступ до ресурсів і забезпечувати безпеку вебзастосунка. Сервер може використовувати механізми аутентифікації та авторизації для перевірки прав доступу до даних та функціональності.
- Множинні платформи – можливість розробляти клієнтські застосунки для різних платформ, таких як веб, мобільні пристрої, настільні застосунки і тому подібне. Це дозволяє широкому колу користувачів отримати доступ до функціональності застосунка незалежно від їхньої платформи.

Архітектура серверної частини складається з таких компонентів:

- Controller – визначає ендпоінти та реалізує REST API (приймає і відповідає на HTTP запити) або WebSocket API (встановлення неперервного двостороннього з'єднання).
- Service – містить бізнес-логіку застосунка, проміжний шар між контролером та DAO.
- DAO – реалізує запити до бази даних.

Клієнтська частина реалізує шаблон MVVM (Model – View – ViewModel) [12] та складається з наступних компонентів:

- View – відповідає за графічний інтерфейс користувача.
- Model – відповідає за представлення даних (модель даних).
- ViewModel – відповідає за логіку взаємодії інтерфейсу та користувача.
- Client – відправляє та отримує відповіді на HTTP запити або ініціює WebSocket з'єднання.

Для взаємодії клієнта та сервера було використано підхід REST, що використовує стандартні HTTP методи GET, POST, PUT та DELETE для взаємодії з ресурсами (ендпоінтами), що мають унікальні ідентифікатори (URL адреси), та забезпечує безстановість сервера [13]. Окрім того для забезпечення безперервного обміну даними між клієнтом та сервером для реалізації засобу

комунікації було використано протокол WebSocket. Для передачі даних по використаних протоколах використовується JSON представлення моделей даних. Оскільки застосунок використовує реляційну СУБД PostgreSQL, взаємодія між сервером застосунку та СУБД відбувається за допомогою SQL запитів. Графічно архітектуру вебзастосунку зображено на рис. 5.

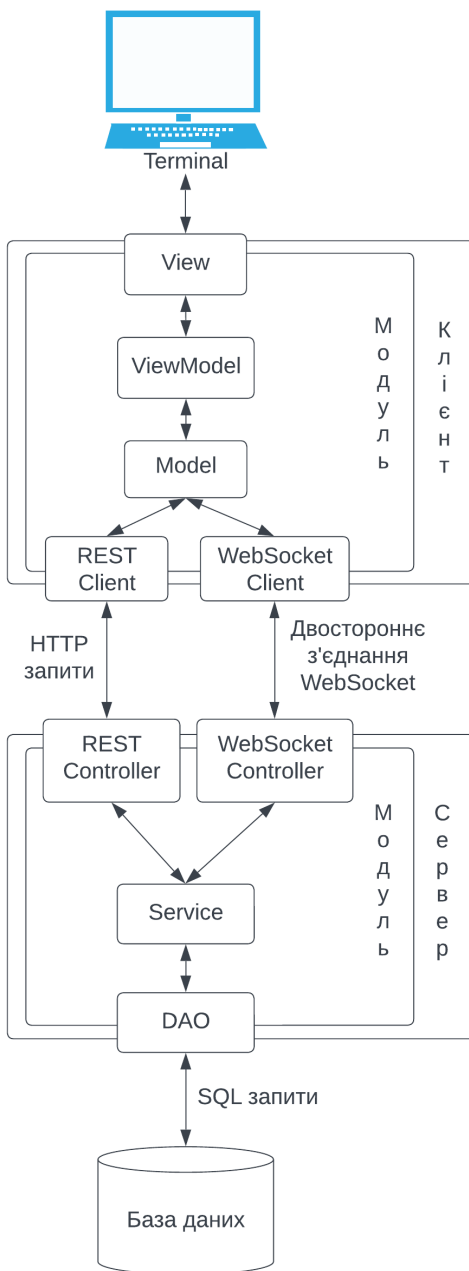


Рис. 5. Архітектура вебзастосунку

Розглянемо процес проєктування схеми бази даних для збереження даних користувачів застосунку. Враховуючи предметну область розроблюваного застосунку, було визначено наступні сутності:

- Користувач – для збереження даних користувача (дані для входу, роль, ім'я та прізвище, тощо).
- Код підтвердження – для збереження коду для активації облікового запису або коду для відновлення паролю та часу створення коду.
- Клас – для збереження даних класу (назва класу, дата створення, тощо).
- Предмет – для збереження даних предмету (назва предмету).
- Класна кімната – для збереження даних класної кімнати.
- Повідомлення – для збереження повідомлень чатів (текст повідомлення, приналежність до учнівського чи батьківського чату, дата й час повідомлення).
- Розклад – для збереження уроку в розкладі (парність тижня, день тижня, номер уроку, активність чи неактивність цього уроку).
- Журнал – для збереження відомостей про урок, що відбувся чи відбудеться певного числа (тип уроку, тема уроку, домашнє завдання, дата).
- Запис журналу – для збереження присутності та оцінок учня за урок та домашнє завдання.

Для сутностей було визначено атрибути та зв'язки, зв'язки N:M було замінено на зв'язки 1:N з використанням допоміжних сутностей та до сутностей було додано наступні атрибути-обмеження:

- обмеження первинного ключа (primary key constraint) – визначає значення колонки чи кількох колонок унікальним ідентифікатором рядка таблиці, позначається РК.

- Таким чином було сформовано логічну модель бази даних, зображену на рис. 6.



Після визначення логічної схеми у процесі проєктування фізичної схеми бази даних атрибутам сутностей було присвоєно типи даних загального використання, визначені в документації PostgreSQL [14], а до сутностей були додані наступні обмеження:

- обмеження унікальності (unique constraint) – забезпечує збереження унікальності значень стовпця або кількох стовпців серед рядків таблиці.
- обмеження на значення NULL (not null constraint) – забороняє записувати в стовпець таблиці значення NULL.

В табл. 3 наведено відображення атрибутів сутностей в атрибути фізичних таблиць бази даних та їх типи даних.

Таблиця 3

Відображення атрибутів сутностей логічної схеми в
атрибути таблиць фізичної схеми

Назва атрибута	Назва у фізичній моделі	Тип даних, обмеження
Користувач		
id користувача (PK)	id	serial4, PK, not null
id класу (FK)	class_id	int4, FK
адреса ел. пошти	email	varchar, not null
пароль	password	varchar, not null
ім'я	name	varchar, not null
прізвище	surname	varchar, not null
роль	role	varchar, not null
статус	status	varchar, not null

дата реєстрації	joined_on	date, not null
Код підтвердження		
id користувача (FK)	user_id	int4, FK, not null
код	code	varchar, not null
дата та час	timestamp	timestamp tz, not null
Приналежність батьків до учнів		
id учня (FK)	student_id	int4, FK, not null
id батька (FK)	parent_id	int4, FK, not null
Клас		
id класу (PK)	id	serial4, PK, not null
id вчителя (FK)	teacher_id	int4, FK
назва	name	varchar, not null
стан активності	is_active	bool, not null
дата створення	created_on	date, not null
Повідомлення		
id повідомлення (PK)	id	serial4, PK, not null
id користувача (FK)	user_id	int4, FK, not null
id класу (FK)	class_id	int4, FK, not null
текст повідомлення	message	varchar, not null
приналежність до чи батьківського учнівського чату	is_parent_chat	bool, not null
дата та час	timestamp	timestamp tz, not null

Предмет		
id предмета (PK)	id	serial4, PK, not null
назва	name	varchar, not null
Приналежність предметів до вчителів		
id предмета (FK)	subject_id	int4, FK, not null
id вчителя (FK)	teacher_id	int4, FK, not null
Класна кімната		
id кімнати (PK)	id	serial4, PK, not null
назва	name	varchar, not null
Розклад		
id розкладу (PK)	id	serial4, PK, not null
id класу (FK)	class_id	int4, FK, not null
id предмету (FK)	subject_id	int4, FK, not null
id вчителя (FK)	teacher_id	int4, FK, not null
id класної кімнати (FK)	classroom_id	int4, FK
парність тижня	is_odd_week	bool, not null
день	day	varchar, not null
номер уроку	lesson_no	int2, not null
стан активності	is_active	bool, not null
Журнал		
id журналу (PK)	id	serial4, PK, not null
id розкладу (FK)	timetable_id	int4, FK, not null

тип уроку	lesson_type	varchar, not null
тема уроку	lesson_theme	varchar, not null
домашнє завдання	homework	varchar
дата	date	date, not null
Запис журналу		
id учня (FK)	student_id	int4, FK, not null
id журналу (FK)	journal_id	int4, FK, not null
присутність	is_present	bool, not null
оцінка за домашнє завдання	homework_mark	int2
оцінка за урок	lesson_mark	int2

У процесі моделювання фізичної схеми бази даних були створені наступні таблиці:

- users;
- codes;
- student_parent;
- classes;
- messages;
- classrooms;
- subjects;
- teacher_subject;
- timetable;
- journal;
- journal_records.

На рис. 7 наведена фізична схема бази даних.

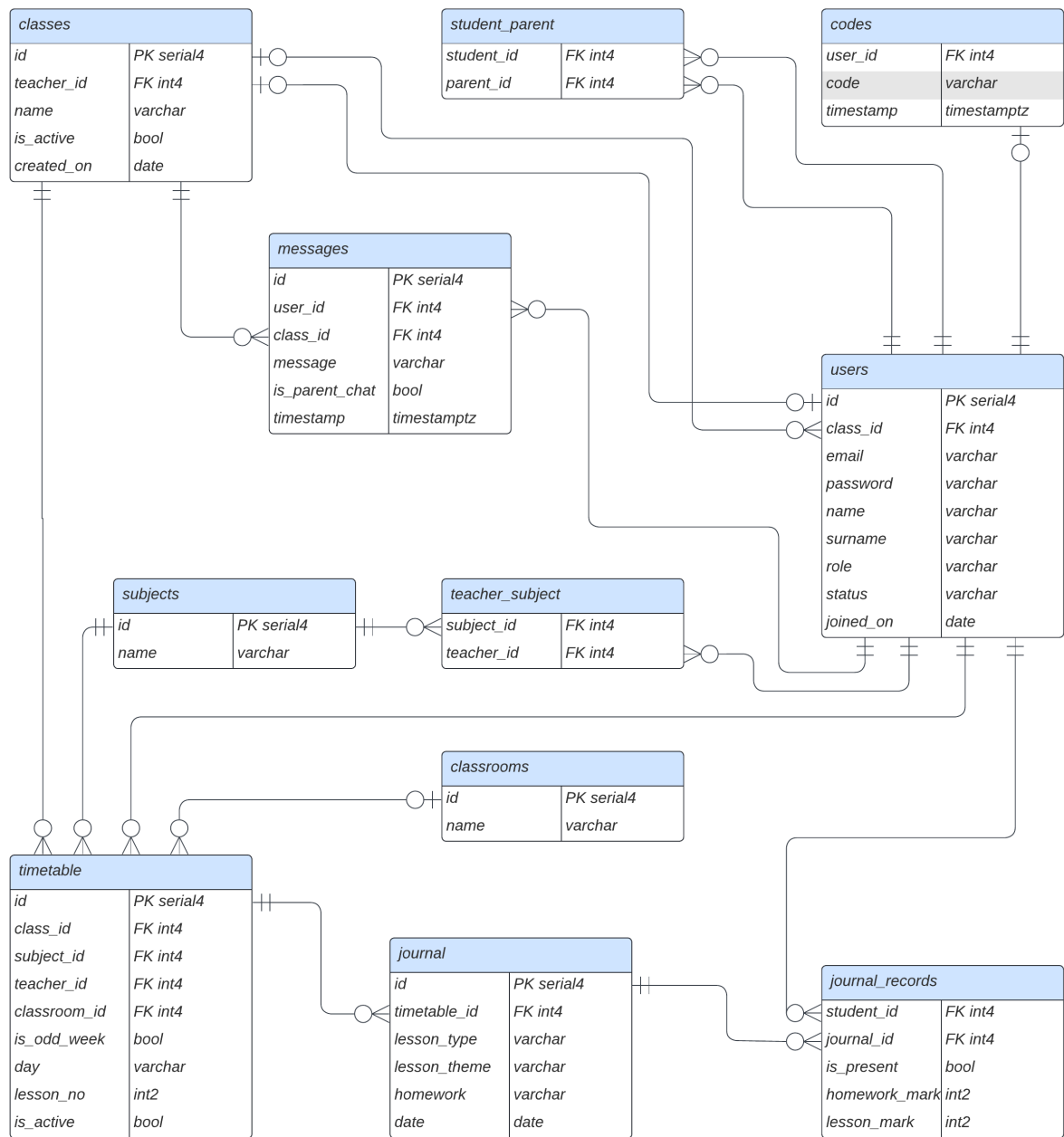


Рис. 7. Фізична модель бази даних

3.3. Особливості реалізації

Серверна частина реалізована за допомогою фреймворка Spring. Бібліотека Spring Web реалізує REST API та надає можливість визначати ендпоінти та приймати і відповідати на HTTP запити. Бібліотека Spring Boot забезпечує легке впровадження серверних застосунків без додаткових конфігурацій [15]. Бібліотека Spring Security разом з застосуванням JWT токена відповідають за автентифікацію, авторизацію та визначення ролі користувача при обробці кожного HTTP запиту. Серверна частина складається з модулів, описаних в функціональних вимогах підрозділу 1.4, що в свою чергу складаються з трьох компонентів, описаних в підрозділі 3.2: контролера, сервіса та DAO.

Клієнтська частина виконана за допомогою фреймворку Angular та складається з модулів, описаних в функціональних вимогах підрозділу 1.4, що в свою чергу складаються з наступних компонентів:

- Компонент Angular – складається з HTML файлу із розміткою сторінки (View в шаблоні MVVM), CSS файлу зі стилями сторінки та TS файлу із класом логіки поведінки сторінки (ViewModel в шаблоні MVVM) [16].
- Модель – TS файл, що визначає інтерфейс (поля та типи даних) моделі даних (Model в шаблоні MVVM)
- Сервіс (клієнт) – відповідає за надсилання HTTP запитів та отримання відповіді на них.
- Роутер – компонент, що визначає, який Angular компонент буде викликаний при переході за певним URL у межах клієнтської частини.

Для реалізації чату була застосована технологія WebSocket, що дозволяє клієнту та серверу встановити двонаправлене з'єднання через протокол TCP та проводити безперервний обмін даними [17].

Для реалізації запитів до бази даних було використано декларативну мову запитів SQL та механізм використання параметризованих запитів Prepared Statement для забезпечення ефективності та захисту запитів від SQL ін'єкцій [18].

Одне з основних завдань компоненту Service модуля керування журналами та щоденниками при отриманні запиту на перегляд чи заплановування уроку на певну дату – визначення парності тижня та дня тижня для вказаної дати для подальшого використання в SQL запиті до бази даних.

Алгоритм складається з наступних кроків:

- Визначення року початку навчання за вхідною датою. Якщо дата знаходиться між 1 січня та 1 червня відповідного року, то навчання почалося в попередньому році. Якщо дата знаходиться між 1 вересня та 1 січня, то навчання почалося в поточному році.
- Визначення парності тижня 1 вересня визначеного в попередньому кроці року. Якщо 1 вересня – вихідний, вважаємо його тиждень парним, інакше – непарним.
- Визначення парності тижня вхідної дати. Рахуємо кількість тижнів між тижнем 1 вересня та тижнем вхідної дати. Якщо кількість тижнів парна – парність тижня вхідної дати рівна парності тижня 1го вересня, в іншому випадку – протилежна за значенням.
- День тижня вхідної дати отримується за допомогою стандартних методів класу LocalDate.

Реалізація алгоритму наведена в лістингу 1.

Лістинг 1. Метод getDateInfo

```
private DateInfo getDateInfo(String dateString) {  
    LocalDate date = LocalDate.parse(  
        dateString,  
        java.time.format.DateTimeFormatter.ofPattern("dd-MM-yyyy")  
    );  
}
```

```

);
int year;
LocalDate firstJune = LocalDate.of(date.getYear(), 6, 1);
LocalDate firstSeptember = LocalDate.of(date.getYear(), 9, 1);

if (date.isBefore(firstJune)) {
    year = date.getYear() - 1;
} else if (date.isEqual(firstSeptember) || date.isAfter(firstSeptember)) {
    year = date.getYear();
} else {
    throw new IllegalArgumentException("Wrong date");
}

LocalDate firstSeptemberOfCurrentStudyYear = LocalDate.of(year, 9, 1);
boolean isStartWeekOdd;

if (firstSeptemberOfCurrentStudyYear.getDayOfWeek().getValue() >=
    DayOfWeek.MONDAY.getValue()
    &&
    firstSeptemberOfCurrentStudyYear.getDayOfWeek().getValue() <=
    DayOfWeek.FRIDAY.getValue())
{
    isStartWeekOdd = true;
} else {
    isStartWeekOdd = false;
}

LocalDate startDay = firstSeptemberOfCurrentStudyYear
    .with(DayOfWeek.MONDAY);
LocalDate endDay = date.with(DayOfWeek.MONDAY);
long weeksInBetween = ChronoUnit.WEEKS.between(startDay, endDay);
boolean isWeekOdd;

if (weeksInBetween % 2 == 0) {
    isWeekOdd = isStartWeekOdd;
} else {
    isWeekOdd = !isStartWeekOdd;
}

return new DateInfo(date.getDayOfWeek().toString(), isWeekOdd);
}

```

У компоненті «DAO» модуля керування журналами та щоденниками реалізовано SQL запити до бази даних для забезпечення отримання коректних даних. Розглянемо наступні запити:

- Запит для отримання доступних предметів при створенні розкладу наведений у лістингу 2. Під доступним предметом мається на увазі такий предмет, принаймні один викладач якого не має уроку по розкладу на шуканій позиції (парний/непарний тиждень, день тижня, номер уроку).

Лістинг 2. Запит для отримання доступних предметів

```
select distinct(s.id) id, s.name
from teacher_subject ts inner join subjects s on ts.subject_id = s.id
where ts.teacher_id not in (
    select t.teacher_id from timetable t
    where t.odd_week = ? and t.day = ? and t.lesson_no = ?
    and t.id <> ? and is_active = true
) order by s.name
```

- Запит для отримання доступних вчителів для певного предмету при створенні розкладу наведений у лістингу 3. Під доступним вчителем мається на увазі такий вчитель, що веде шуканий предмет та не має уроку по розкладу на шуканій позиції (парний/непарний тиждень, день тижня, номер уроку).

Лістинг 3. Запит для отримання доступних вчителів

```
select u.id, u.surname || ' ' || u.name as name
from teacher_subject ts inner join users u on ts.teacher_id = u.id
where ts.subject_id = ? and u.id not in (
    select t.teacher_id from timetable t
    where t.odd_week = ? and t.day = ? and t.lesson_no = ?
    and t.id <> ? and is_active = true
) order by name
```

- Запит для отримання доступних класних кімнат при створенні розкладу наведений у лістингу 4. Під доступною класною кімнатою мається на увазі така класна кімната, яка не використана в розкладі на шуканій позиції (парний/непарний тиждень, день тижня, номер уроку).

Лістинг 4. Запит для отримання доступних класних кімнат

```
select c.id, c.name from classrooms c
where c.id not in (
    select distinct(t.classroom_id) from timetable t
    where t.classroom_id is not null and t.odd_week = ?
    and t.day = ? and t.lesson_no = ? and t.id <> ?
    and is_active = true
) order by c.name
```

- Запит для отримання розкладу вчителя на день за визначеною датою з додаванням відомостей про урок (тема, тип уроку та задане домашнє завдання) наведений у лістингу 5.

Лістинг 5. Запит для отримання розкладу вчителя на день

```
select t.id, t.lesson_no, c.name class_name,
s.name subject_name, cr.name classroom, j.id lesson_id,
j.lesson_type, j.lesson_theme, j.homework
from timetable t inner join classes c on t.class_id = c.id
inner join subjects s on t.subject_id = s.id
left join classrooms cr on t.classroom_id = cr.id
left join (
    select * from journal
    where date = to_date(?, 'DD-MM-YYYY')
) j on t.id = j.timetable_id
where t.odd_week = ? and upper(t.day) = upper(?)
and t.teacher_id = ? and t.is_active = true
```

3.4. Висновки до розділу

У даному розділі була визначена загальна структура вебзастосунку, визначені модулі та їх функціональні можливості. Також були визначені сценарії використання застосунку для керівництва школи, вчителя, учня і батьків та наведені діаграми даних сценаріїв для кожної ролі користувача.

Також були надані особливості архітектури вебзастосунку, а саме поділ на клієнтську та серверну частини та компоненти модулів цих частин: *controller*, *service* та *dao* для серверної частини та *view*, *viewmodel*, *model* та *client* для клієнтської частини. Визначені способи взаємодії клієнтської і серверної частини та серверної частини і бази даних та наведена діаграми архітектури вебзастосунку. Окрім того були визначені сутності для збереження даних, їх атрибути та зв'язки між ними та сформована логічна схема бази даних, після чого модель була пристосована до реалізації в контексті обраної СУБД PostgreSQL. Було визначено список таблиць, похідних від сутностей, та типи даних для атрибутів, що підтримуються СУБД. Також була наведена фізична схема бази даних.

У розділі також описані особливості реалізації застосунку, а саме компоненти клієнтської та серверної частин та використані для їхньої реалізації фреймворки, бібліотеки та технології. Також наведені розроблені SQL запити до бази даних та алгоритм визначення позиції дати в розкладі.

4. АНАЛІЗ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

4.1. Тестування застосунку

Для тестування застосунку було обрано наступні методики тестування: інтеграційне тестування, модульне тестування та функціональне тестування. Тестування було проведене як вручну, так і за допомогою програмних тестів. Графічний інтерфейс користувача клієнтської частини був протестований вручну, тоді як серверна частина була протестована програмними тестами, написаними за допомогою бібліотеки JUnit.

При тестуванні користувацького інтерфейсу та інтерфейсу серверної частини було перевірено роботу застосунку при введенні та надсиланні невалідних даних та вжито заходів з валідації даних при виявленні помилок. Також було перевірено відповідність застосунку функціональним та нефункціональним вимогам. Оскільки при надсиланні запитів до СУБД використовується механізм Prepared Statement, тестування на захист запитів від SQL ін'єкцій було опущене.

Тест кейси для модулів застосунку, описаних в функціональних вимогах підрозділу 1.4, детально описані в табл. 4.

Таблиця 4

Тест кейси для модулів застосунку

№	Протестований модуль	Опис дій	Очікувані результати
1	Модуль Керівництва школи: реєстрація та керування користувачами.	При додаванні та редагуванні користувача з будь-якою роллю вказати незареєстровану електронну адресу.	Користувач успішно доданий, відправлено лист з кодом активації облікового запису.

2	Модуль Керівництва школи: реєстрація та керування користувачами.	При додаванні та редагуванні користувача з будь-якою роллю вказати зареєстровану електронну адресу.	Виведене повідомлення про помилку – користувач з такою адресою вже існує.
3	Модуль Керівництва школи: реєстрація та керування користувачами.	При додаванні чи редагуванні користувача залишити обов'язкові поля пустими.	Виведене повідомлення про необхідність заповнити обов'язкові поля.
4	Модуль автентифікації та авторизації.	Перейти по посиланню з отриманого листа для активації облікового запису.	Виведення повідомлення про успіх активації облікового запису або невдачі у випадку неправильності посилання.
5	Модуль автентифікації та авторизації.	На сторінці логіну ввести: 1 – коректні логін та пароль для входу; 2 – некоректний логін або пароль для входу.	Результат: 1 – вхід в обліковий запис, перехід на сторінку налаштувань; 2 – вивід повідомлення про помилку входу.
6	Модуль автентифікації та авторизації.	На сторінці налаштувань при зміні старого пароля на новий вказати неправильний старий пароль.	Повідомлення про помилку.

7	Модуль автентифікації та авторизації.	На сторінці налаштувань при зміні пароля на новий: 1 – залишити поля пустими; 2 – при введенні підтвердження нового пароля ввести значення, що не збігається з новим паролем.	Має бути виведено: 1 – повідомлення про незаповнені поля; 2 – повідомлення, що паролі не збігаються.
8	Модуль Керівництва школи: створення та редагування розкладу.	При додаванні чи редагуванні уроку для певного класу вибрати предмет, викладача та класну кімнату з відповідних списків.	Доступні для вибору предмети мають бути такими, щоб хоча б один вчитель, що їх викладає, не був зайнятий на цей урок. Доступні для вибору вчителі мають бути такими, що ведуть обраний предмет та не є зайнятими на цьому уроці. Доступні для вибору класні кімнати мають бути такими, що не є зайнятими на цьому уроці.
9	Модуль Вчителя: перегляд розкладу за певне число.	На сторінці перегляду розкладу за певне число вибрати поточну дату чи дату, меншу за поточну.	В уроків, що були заплановані, має бути кнопка для переходу на сторінку перегляду та заповнення журналу.

10	Модуль Вчителя: перегляд розкладу за певне число.	На сторінці перегляду розкладу за певне число вибрати дату, більшу за поточну.	В уроків, що ще не були заплановані, має бути кнопка для відкриття вікна планування уроку.
11	Модуль Вчителя, Учня, Батьків: перегляд розкладу за певне число.	На сторінці перегляду розкладу за певне число відкрити вікно вибору дати.	Доступні для вибору дати мають належати до поточного навчального року.
12	Модуль Вчителя, Учня та Батьків: перегляд успішності учнів за вказаний термін.	На сторінці перегляду успішності відкрити вікно вибору дат.	Доступні для вибору дати мають належати до поточного навчального року.

Розглянемо приклади розроблених тестів, їх очікувані результати та результати виконання.

- Тест нефункціональної вимоги щодо можливості безперебійного доступу до вебзастосунку 2000 користувачів, що одночасно знаходяться в системі, з часом відповіді на запит менше 10 секунд. Тест полягає у паралельному надсиланні 2000 запитів до серверної частини для отримання записів щоденника учня за обрану дату. Даний запит був обраний через те, що його SQL запит один з найбільш ресурсно-затратних запитів до бази даних та очікується масове використання учнями цього запиту під час навчального дня. Результат виконання наведений на рис. 8.

```

✓ sendHttpRequest 9 sec 39 ms
✓ sendHttpRequest 9 sec 630 ms
✓ sendHttpRequest 9 sec 58 ms
✓ sendHttpRequest 9 sec 81 ms
✓ sendHttpRequest 9 sec 604 ms
✓ sendHttpRequest 9 sec 440 ms
✓ sendHttpRequest 9 sec 239 ms
✓ sendHttpRequest 9 sec 640 ms
✓ sendHttpRequest 9 sec 656 ms
✓ sendHttpRequest 9 sec 604 ms
✓ sendHttpRequest 9 sec 542 ms
✓ sendHttpRequest 9 sec 294 ms

12:13:06.152 [Finalizer] DEBUG org.apache.http.impl.conn.Default
12:13:06.152 [Finalizer] DEBUG org.apache.http.impl.conn.Default
Seconds executing: 9

=====
Default Suite
Total tests run: 2000, Passes: 2000, Failures: 0, Skips: 0
=====

```

Рис. 8. Результат тестування на навантаження сервера

- Тест для перевірки коректності реєстрації користувачів. Полягає у реєстрації кількох користувачів з однаковою адресою ел. пошти. Очікувані результати – pass при реєстрації унікальної адреси ел. пошти та fail при спробі реєстрації існуючої адреси ел. пошти. Використані вхідні дані наведені на рис. 9.

```

1 usage
private static String[] bodyProvider() {
    return new String[] {
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nukedjr@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Lana\", \"surname\": \"Newhouse\"}\"
    };
}

```

Рис. 9. Вхідні дані для тесту реєстрації користувачів

Результат тесту наведений на рис. 10.

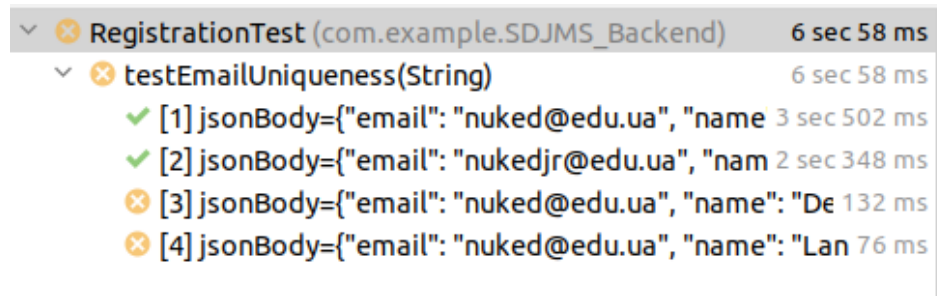


Рис. 10. Результат тесту реєстрації користувачів

- Тест для перевірки коректності роботи методу getDateInfo. Маємо дату у формі рядка та об'єкт DateInfo для порівняння з результатом роботи методу. При рівності результату функції та об'єкта DateInfo pass, при нерівності fail, при введенні некоректної дати або такої дати, що попадає на період літніх канікул error. Очікуваний результат: 1 – fail, 2 – pass, 3 – fail, 4 – pass, 5 – error, 6 – error. Вхідні дані для тесту наведені на рис. 11.

```
1 usage
private static Stream<DataProviderObj> dateProvider() {
    return Stream.of(
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "MONDAY", isWeekOdd: true)),
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "MONDAY", isWeekOdd: false)),
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "TUESDAY", isWeekOdd: false)),
        new DataProviderObj( date: "10-05-2023", new DateInfo( day: "WEDNESDAY", isWeekOdd: true)),
        new DataProviderObj( date: "15-08-2023", new DateInfo( day: "MONDAY", isWeekOdd: true)),
        new DataProviderObj( date: "1342343", new DateInfo( day: "MONDAY", isWeekOdd: true))
    );
}
```

Рис. 11. Вхідні дані для тесту методу getDateInfo

Результат тесту наведений на рис. 12.

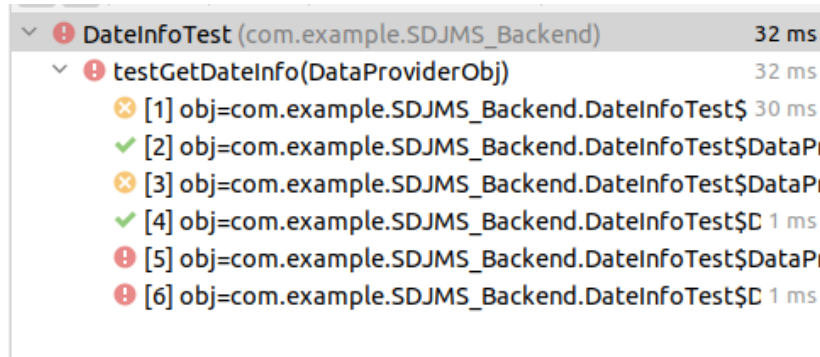


Рис. 12. Результат тесту методу getDateInfo

4.2. Порівняння розробки з наявними аналогами

При аналізі аналогічних застосунків у першому розділі було визначено, що кожен із них має суттєві недоліки. Розроблений вебзастосунок, що реалізує функціональні вимоги, визначені внаслідок аналізу переваг та недоків конкурентних продуктів, має наступні конкурентні переваги у порівнянні з аналогами:

- Можливість перегляду успішності учня – не кожен конкурентний застосунок надає можливість батькам чи вчителю переглядати середні бали учня, що унеможлиблює контроль якості навчального процесу та засвоєння знань учнем.
- Наявність засобів комунікації між учнями та класним керівником та між батьками та класним керівником – комунікація відіграє надзвичайно важливу роль під час онлайн навчання. Відсутність відповідного засобу може негативно вплинути на навчальний процес та проінформованість учнів та батьків.
- Повна відповідність функціональності ролей користувачів їх завданням у процесах функціонування закладу середньої освіти – деякі аналоги надають користувачам доступ до функціональності, яка не передбачена вищезгаданим процесом. Дотримання відповідності

забезпечить коректну роботу застосунку та дозволить запобігти проблемам у навчальному процесі.

- Можливість використовувати застосунок як єдиний інформаційний ресурс – відповідність застосунку процесам функціонування навчального закладу підрозуміває обов’язкову реєстрацію учнів, що робить застосунок єдиним інформаційним ресурсом та дозволяє не використовувати додаткові засоби на кшталт класичних паперових журналів.
- Використання сучасних засобів та технологій під час розробки – підтримка сучасних стандартів, висока надійність та ефективність застосунку.

Отже, у порівнянні з аналогами, що існують на ринку застосунків для організації онлайн навчання, розроблений застосунок має суттєві переваги, що роблять його конкурентноспроможним продуктом на ринку.

4.3. Рекомендації щодо подальшого вдосконалення та підтримки

Хоча основні засоби для забезпечення організації онлайн навчання було розроблено, у освітньому процесі можуть відбутися зміни, що вимагатимуть змін та доопрацювань програмного продукту. Окрім того, застосунок можна покращити, реалізувавши наступні засоби:

- збереження зведених оцінок учня по предметах за навчальний рік у документ, ідентичний таблицю, з можливістю експорту документа;
- видалення даних учнів та класів після вичерпання терміну зберігання цих даних;
- приватний чат учня з вчителем, який викладає в учня;
- сповіщення про нові повідомлення в чатах, задання домашнього завдання, тощо.

Під час розробки архітектури та бази даних застосунку було враховано перспективу масштабування та додавання нової функціональності, тому подальша розробка та розширення проєкту не викликатиме складнощів.

4.4. Висновки до розділу

У цьому розділі було проаналізовано розроблений застосунок, а саме описано тестування застосунку, проведено аналіз конкурентоспроможності застосунка та визначено рекомендації щодо розвитку продукту.

Методами модульного, інтеграційного та функціонального тестування була проведена перевірка застосунку на коректність функціонування, стійкість до введення некоректних даних, правильність взаємодії модулів одне з одним та відповідність функціональним та нефункціональним вимогам. Детальна інформація про тест кейси, а саме об'єкти тестування, дії та очікувані результати, винесена у окрему таблицю. Також наведено приклади розроблених тестів з очікуваними результатами та результатами виконання.

Окрім того, було порівняно розроблений застосунок з наявними аналогами на ринку та виокремлено конкурентні переваги застосунка над ними. Зважаючи на кількість та значимість переваг, було зроблено висновок про конкурентноспроможність застосунку.

Також було визначено рекомендації щодо подальшого вдосконалення для збереження актуальності програмного забезпечення та доповнення його новими можливостями.

ВИСНОВКИ

Метою даного дипломного проєкту була розробка вебзастосунка для ведення шкільних журналів та учнівських щоденників.

В процесі роботи над проєктом було проаналізовано предметну область та наявні на ринку застосунки-аналоги, такі як “Нові знання”, “*Electronic Journal*”, “*e schools*” та “*Єдина Школа*”. Було проаналізовано переваги та недоліки кожного аналога, після чого були визначені основні функціональні можливості для застосунку. На їх основі було сформовано функціональні та нефункціональні вимоги до застосунку, а також вимоги до якості застосунку. Окрім того було проведено аналіз ризиків і загроз безпеці програмного забезпечення, на основі якого були сформовані вимоги до безпеки застосунка.

Під час аналізу засобів реалізації було проаналізовано такі мови програмування: *Java*, *C++*, *TypeScript*. Для розробки серверної частини застосунку була обрана мова *Java* через її кросплатформеність та ефективність, тоді як для реалізації клієнтської частини – мова *TypeScript*, обрана через сумісність з більшістю сучасних браузерів та статичну типізацію. Також було проаналізовано наступні СУБД: *PostgreSQL*, *MongoDB*, *Neo4j*. Серед них була обрана реляційна СУБД *PostgreSQL*, оскільки її модель збереження даних найбільш підходяща для предметної області застосунку.

Застосунок реалізує клієнт-серверну архітектуру та використовує підходи REST та WebSocket для взаємодії клієнта та сервера. Клієнтська частина реалізує шаблон проєктування MVVM та розроблена за допомогою фреймворка Angular, в той час як серверна частина складається з контролера, сервіса та DAO і розроблена за допомогою фреймворка Spring. В процесі написання проєкту було визначено основні сутності та розроблено логічну схему бази даних, на основі якої було створено фізичну схему бази даних. В роботі описані розроблені алгоритми та запити до бази даних.

Для забезпечення якості програмного продукту застосунок був протестований модульним, ітераційним та функціональним тестуванням. Окрім того було порівняно розроблений застосунок з наявними на ринку аналогами, виокремлено конкурентні переваги застосунка над аналогами та зроблено висновок про конкурентноспроможність застосунку. Також було сформовано список рекомендацій щодо подальшого вдосконалення та підтримки застосунку.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Положення про дистанційну форму здобуття повної загальної середньої освіти [Електронний ресурс] — Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/z0941-20#n22>
2. E-Learning Market Size [Електронний ресурс] — Режим доступу до ресурсу: <https://www.gminsights.com/industry-analysis/elearning-market-size>
3. Єдина Школа [Електронний ресурс] — Режим доступу до ресурсу: <https://eschool-ua.com/>
4. e school [Електронний ресурс] — Режим доступу до ресурсу: <https://e-schools.info/>
5. B. Stroustrup. The C++ Programming Language Fourth Edition. Pearson Education, Inc., 2013. 11p.
6. H. Schildt. Java A Beginner's Guide Sixth Edition. McGraw-Hill Education, 2014. 6p.
7. B. Cherny. Programming TypeScript. O'Reilly Media, Inc., 2019. 6p.
8. About PostgreSQL [Електронний ресурс] — Режим доступу до ресурсу: <https://www.postgresql.org/about>
9. What is MongoDB? [Електронний ресурс] — Режим доступу до ресурсу: <https://www.mongodb.com/what-is-mongodb>
10. Scaling Neo4j Graph Database [Електронний ресурс] — Режим доступу до ресурсу: <https://neo4j.com/product/neo4j-graph-database/scalability>
11. Client-Server Model [Електронний ресурс] — Режим доступу до ресурсу: <https://www.geeksforgeeks.org/client-server-model/>
12. Introduction to Model View View Model [Електронний ресурс] — Режим доступу до ресурсу: <https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/>

13. What is REST? [Электронный ресурс] — Режим доступа до ресурсу:
<https://www.codecademy.com/article/what-is-rest>
14. Chapter 8. Data Types [Электронный ресурс] — Режим доступа до ресурсу:
<https://www.postgresql.org/docs/current/datatype.html>
15. What Is Spring Boot? [Электронный ресурс] — Режим доступа до ресурсу:
<https://stackify.com/what-is-spring-boot/>
16. A Practical Guide to Angular: Components & NgModules [Электронный ресурс] — Режим доступа до ресурсу: <https://www.telerik.com/blogs/a-practical-guide-to-angular-components-ngmodules>
17. What is web socket and how it is different from the HTTP? [Электронный ресурс] — Режим доступа до ресурсу: <https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>
18. How to Use PreparedStatement in Java? [Электронный ресурс] — Режим доступа до ресурсу: <https://www.geeksforgeeks.org/how-to-use-preparedstatement-in-java/>

ДОДАТКИ

Додаток 1
Копії графічних елементів



ДП.045440-06-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників. Варіанти використання застосунку для учня. Діаграма варіантів використання



ДП.045440-07-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників. Варіанти використання застосунку для вчителя. Діаграма варіантів використання



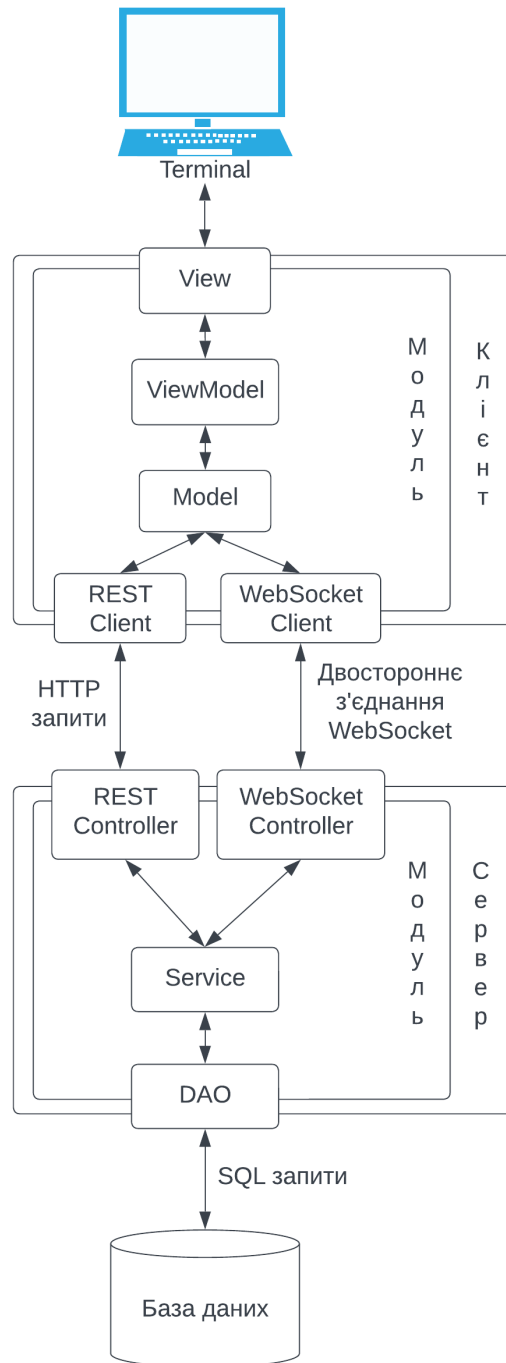
ДП.045440-08-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників. Варіанти використання застосунку для батьків. Діаграма варіантів використання



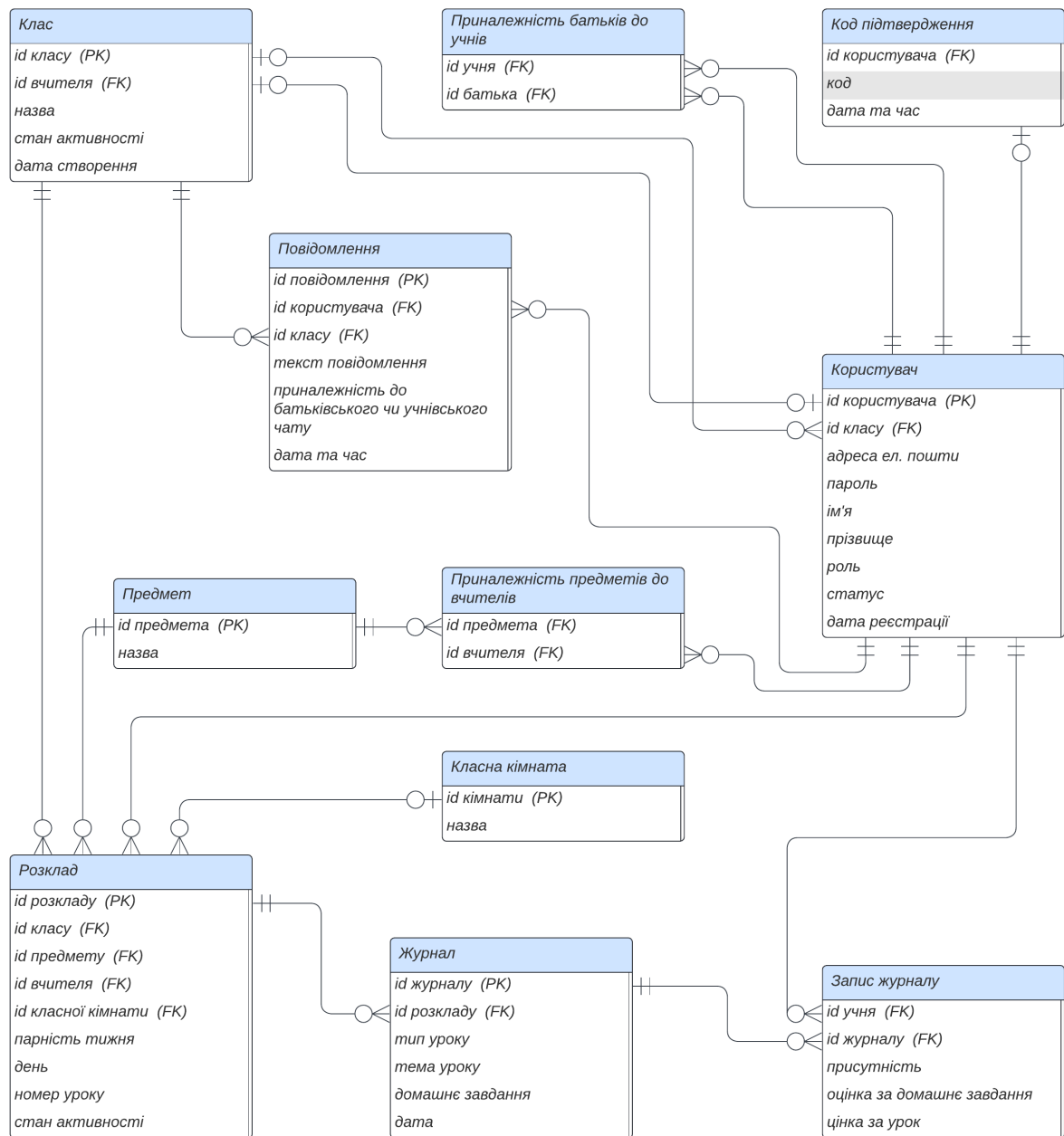
ДП.045440-09-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників. Варіанти використання застосунку для керівництва школи. Діаграма варіантів використання



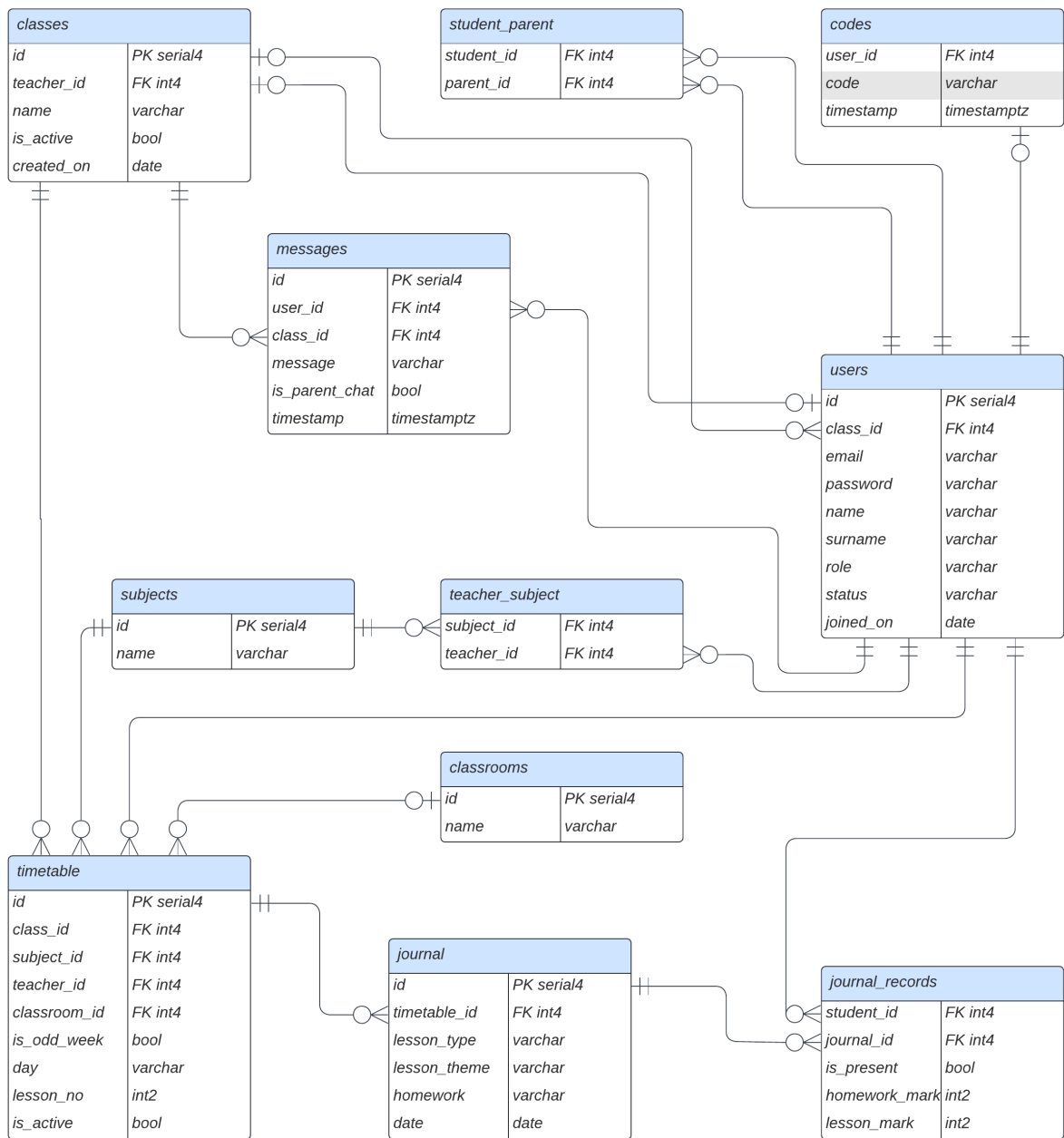
ДП.045440-10-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників.
Архітектура вебзастосунку. UML діаграма



ДП.045440-11-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників.
 Логічна схема бази даних. ER діаграма



ДП.045440-12-99

Вебзастосунок для ведення шкільних журналів та учнівських щоденників.
Фізична схема бази даних. ER діаграма

Додаток 2
Лістинг програми

Фрагмент коду компонента DAO серверної частини

```
package com.example.SDJMS_Backend.dao.impl;

import com.example.SDJMS_Backend.dao.TimetableDao;
import com.example.SDJMS_Backend.models.entity.*;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
import org.springframework.stereotype.Repository;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.List;

@Repository
@AllArgsConstructor
@Slf4j
public class TimetableDaoImpl implements TimetableDao {

    private final JdbcTemplate jdbcTemplate;

    private static JournalRecord mapJournalRecord(ResultSet rs, int rowNum)
throws SQLException {
        return new JournalRecord(
            rs.getInt("student_id"),
            rs.getString("name"),
            rs.getBoolean("is_present"),
            rs.getObject("homework_mark", Integer.class),
            rs.getObject("lesson_mark", Integer.class)
        );
    }

    private static RawTimetableDayResponse mapRawTimetableDay(ResultSet rs,
int rowNum) throws SQLException {
        return new RawTimetableDayResponse(
```

```

        rs.getInt("id"),
        rs.getInt("lesson_no"),
        rs.getString("subject_name"),
        rs.getString("teacher_name"),
        rs.getString("teacher_surname"),
        rs.getString("classroom")
    );
}

private static SimpleDescription mapSimpleRelatedEntity(ResultSet rs, int
rowNum) throws SQLException {
    return new SimpleDescription(
        rs.getInt("id"),
        rs.getString("name")
    );
}

private static TimetableDayResponseTeacher
mapTimetableDayTeacher(ResultSet rs, int rowNum) throws SQLException {
    return new TimetableDayResponseTeacher(
        rs.getInt("id"),
        rs.getInt("lesson_no"),
        rs.getString("subject_name"),
        rs.getString("class_name"),
        rs.getString("classroom")
    );
}

private static TimetableDayResponseTeacherExt
mapTimetableDayExtTeacher(ResultSet rs, int rowNum) throws SQLException {
    return new TimetableDayResponseTeacherExt(
        rs.getInt("id"),
        rs.getInt("lesson_no"),
        rs.getString("subject_name"),
        rs.getString("class_name"),
        rs.getString("classroom"),
        rs.getInt("lesson_id"),
        rs.getString("lesson_type"),
        rs.getString("lesson_theme"),
        rs.getString("homework")
    );
}

```

```

        );
    }

    private static TimetableDayResponseStudentExt
    mapTimetableDayExtStudent(ResultSet rs, int rowNum) throws SQLException {
        return new TimetableDayResponseStudentExt(
            rs.getInt("id"),
            rs.getInt("lesson_no"),
            rs.getString("subject_name"),
            rs.getString("teacher_name"),
            rs.getString("classroom"),
            rs.getString("lesson_type"),
            rs.getString("lesson_theme"),
            rs.getString("homework"),
            rs.getObject("is_present", Boolean.class),
            rs.getObject("lesson_mark", Integer.class),
            rs.getObject("homework_mark", Integer.class)
        );
    }

    @Override
    public List<RawTimetableDayResponse>
    getTimetableDay(RawTimetableDayRequest req) {
        return jdbcTemplate.query(
            "select tt.id, tt.lesson_no, s.name subject_name, t.name
teacher_name, t.surname teacher_surname, c.name classroom\n" +
            "from timetable tt inner join subjects s on tt.subject_id = s.id
inner join users t on tt.teacher_id = t.id left join classrooms c on
tt.classroom_id = c.id\n" +
            "where tt.is_active = true and tt.class_id = ? and tt.odd_week = ?
and tt.day = ?",
            TimetableDaoImpl::mapRawTimetableDay, req.getClassId(),
            req.getIsWeekOdd(), req.getDay());
    }

    @Override
    public List<SimpleDescription>
    getSuitableSubjects(SuitableLessonEntityRequest req) {
        return jdbcTemplate.query(
            "select distinct(s.id) id, s.name from tchr_sbjet ts inner join

```

```

subjects s on ts.subject_id = s.id where ts.teacher_id not in (select
t.teacher_id from timetable t where t.odd_week = ? and t.day = ? and
t.lesson_no = ? and t.id <> ? and is_active = true) order by s.name",
        TimetableDaoImpl::mapSimpleRelatedEntity, req.getIsWeekOdd(),
req.getDay(), req.getLessonNo(), req.getId()
    );
}

@Override
public List<SimpleDescription>
getSuitableTeachers(SuitableLessonEntityRequest req) {
    return jdbcTemplate.query(
        "select u.id, u.surname || ' ' || u.name as name from tchr_sbjct
ts inner join users u on ts.teacher_id = u.id where ts.subject_id = ? and u.id
not in (select t.teacher_id from timetable t where t.odd_week = ? and t.day =
? and t.lesson_no = ? and t.id <> ? and is_active = true) order by name",
        TimetableDaoImpl::mapSimpleRelatedEntity, req.getSubjectId(),
req.getIsWeekOdd(), req.getDay(), req.getLessonNo(), req.getId()
    );
}

@Override
public List<SimpleDescription>
getSuitableClassrooms(SuitableLessonEntityRequest req) {
    return jdbcTemplate.query(
        "select c.id, c.name from classrooms c where c.id not in (select
distinct(t.classroom_id) from timetable t where t.classroom_id is not null and
t.odd_week = ? and t.day = ? and t.lesson_no = ? and t.id <> ? and is_active =
true) order by c.name",
        TimetableDaoImpl::mapSimpleRelatedEntity, req.getIsWeekOdd(),
req.getDay(), req.getLessonNo(), req.getId()
    );
}

@Override
public void createLesson(LessonCreateData data) {
    jdbcTemplate.update(
        "insert into timetable (class_id, odd_week, day, lesson_no,
subject_id, teacher_id, classroom_id) values (?, ?, ?, ?, ?, ?, ?)",
        data.getClassId(), data.getIsWeekOdd(), data.getDay(),

```

```

data.getLessonNo(), data.getSubjectId(), data.getTeacherId(),
data.getClassroomId()
    );
}

@Override
public void updateLesson(LessonUpdateData data) {
    jdbcTemplate.update("update timetable set subject_id = ?, teacher_id =
?, classroom_id = ? where id = ?",
        data.getSubjectId(), data.getTeacherId(), data.getClassroomId(),
data.getId());
}

@Override
public void deleteLesson(int id) {
    jdbcTemplate.update("delete from timetable where id = ?", id);
}

@Override
public List<TimetableDayResponseTeacher>
getTimetableDayForTeacher(TimetableDayRequestTeacher req) {
    return jdbcTemplate.query("select tt.id, tt.lesson_no, s.name
subject_name, cls.name class_name, c.name classroom\n" +
        "from timetable tt inner join subjects s on tt.subject_id = s.id
inner join classes cls on tt.class_id = cls.id left join classrooms c on
tt.classroom_id = c.id\n" +
        "where tt.is_active = true and tt.teacher_id = ? and tt.odd_week =
? and tt.day = ?",
        TimetableDaoImpl::mapTimetableDayTeacher, req.getTeacherId(),
req.getIsWeekOdd(), req.getDay());
}

@Override
public List<TimetableDayResponseTeacherExt>
getTimetableDayExtForTeacher(DateSearchInfo dsi, DateInfo di) {
    return jdbcTemplate.query(
        "select t.id, t.lesson_no, c.name class_name, s.name subject_name,
cr.name classroom, j.id lesson_id, j.lesson_type, j.lesson_theme,
j.homework\n" +
        "from timetable t inner join classes c on t.class_id = c.id

```

```

inner join subjects s on t.subject_id = s.id left join classrooms cr on
t.classroom_id = cr.id\n" +
        "left join (select * from journal where date = to_date(?, 'DD-
MM-YYYY')) j on t.id = j.timetable_id\n" +
        "where t.odd_week = ? and upper(t.day) = upper(?) and
t.teacher_id = ? and t.is_active = true",
        TimetableDaoImpl::mapTimetableDayExtTeacher, dsi.getDate(),
di.isWeekOdd(), di.getDay(), dsi.getId());
    }

    @Override
    public void planLesson(LessonProperties lp, DateInfo di) {
        int isAllowed = jdbcTemplate.queryForObject(
            "select count(id) count from timetable where teacher_id = ? and id
= ? and odd_week = ? and upper(day) = upper(?)",
            (ResultSet rs, int rowNum) -> rs.getInt("count"),
            lp.getTeacherId(), lp.getTimetableId(), di.isWeekOdd(), di.getDay());
        if (isAllowed == 1) {
            KeyHolder keyHolder = new GeneratedKeyHolder();
            jdbcTemplate.update(
                connection -> {
                    PreparedStatement ps = connection.prepareStatement(
                        "insert into journal (timetable_id, lesson_type,
lesson_theme, homework, date) values (?, ?, ?, ?, to_date(?, 'DD-MM-YYYY'))
returning id",
                        Statement.RETURN_GENERATED_KEYS);
                    ps.setInt(1, lp.getTimetableId());
                    ps.setString(2, lp.getLessonType());
                    ps.setString(3, lp.getLessonTheme());
                    ps.setString(4, lp.getHomework());
                    ps.setString(5, lp.getDate());
                    return ps;
                }, keyHolder);
            int journalId = keyHolder.getKey().intValue();

            jdbcTemplate.update("insert into journal_list (journal_id,
student_id) select ? journal_id, id student_id from users where role =
'student' and class_id = (select class_id from timetable where id = ?)",
                journalId, lp.getTimetableId());
        }
    }

```

```

    }

    @Override
    public List<JournalRecord> getLessonJournal(int journalId, int teacherId)
    {
        return jdbcTemplate.query("select j.student_id, u.surname || ' ' ||
u.name name, j.is_present, j.homework_mark, j.lesson_mark from journal_list j
inner join users u on j.student_id = u.id where j.journal_id = ? order by
u.surname, u.name",
        TimetableDaoImpl::mapJournalRecord, journalId);
    }

    @Override
    public void setPresence(PresenceInfo data) {
        jdbcTemplate.update("update journal_list set is_present = ? where
journal_id = ? and student_id = ?",
        data.getPresence(), data.getJournalId(), data.getStudentId());
    }

    @Override
    public void setHomeworkMark(MarkInfo data) {
        jdbcTemplate.update("update journal_list set homework_mark = ? where
journal_id = ? and student_id = ?",
        data.getMark(), data.getJournalId(), data.getStudentId());
    }

    @Override
    public void setLessonMark(MarkInfo data) {
        jdbcTemplate.update("update journal_list set lesson_mark = ? where
journal_id = ? and student_id = ? and is_present = true",
        data.getMark(), data.getJournalId(), data.getStudentId());
    }

    @Override
    public List<RawTimetableDayResponse>
getTimetableDayForStudent(TimetableDayRequestStudent req) {
        return jdbcTemplate.query("select tt.id, tt.lesson_no, s.name
subject_name, t.name teacher_name, t.surname teacher_surname, c.name
classroom\n" +
        "from timetable tt inner join subjects s on tt.subject_id = s.id

```

```

        inner join users t on tt.teacher_id = t.id left join classrooms c on
        tt.classroom_id = c.id\n" +
            "where tt.is_active = true and tt.class_id = (select class_id from
            users where id = ?) and tt.odd_week = ? and tt.day = ?",
            TimetableDaoImpl::mapRawTimetableDay, req.getStudentId(),
            req.getIsWeekOdd(), req.getDay());
    }

    @Override
    public List<TimetableDayResponseStudentExt>
    getTimetableDayExtForStudent(DateSearchInfo dsi, DateInfo di) {
        return jdbcTemplate.query(
            "select t.id, t.lesson_no, u.surname || ' ' || u.name
            teacher_name, s.name subject_name, cr.name classroom, j.lesson_type,
            j.lesson_theme, j.homework,\n" +
            "jr.is_present, jr.lesson_mark, jr.homework_mark\n" +
            "from timetable t inner join users u on t.teacher_id = u.id inner
            join subjects s on t.subject_id = s.id left join classrooms cr on
            t.classroom_id = cr.id\n" +
            "left join (select * from journal where date = to_date(?, 'DD-MM-
            YYYY')) j on t.id = j.timetable_id\n" +
            "left join (select * from journal_list where student_id = ?) jr on
            j.id = jr.journal_id\n" +
            "where t.odd_week = ? and upper(t.day) = upper(?) and t.class_id =
            (select class_id from users where id = ?) and t.is_active = true",
            TimetableDaoImpl::mapTimetableDayExtStudent, dsi.getDate(),
            dsi.getId(), di.isWeekOdd(), di.getDay(), dsi.getId());
    }
}

```

Фрагмент коду компонента Service серверної частини

```
package com.example.SDJMS_Backend.service.impl;

import com.example.SDJMS_Backend.dao.TimetableDao;
import com.example.SDJMS_Backend.models.entity.*;
import com.example.SDJMS_Backend.service.TimetableService;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.time.LocalDate;
import java.time.DayOfWeek;
import java.time.temporal.ChronoUnit;

@Service
@Transactional
@Slf4j
@AllArgsConstructor
public class TimetableServiceImpl implements TimetableService {

    private final TimetableDao timetableDao;

    private DateInfo getDateInfo(String dateString) {
        LocalDate date = LocalDate.parse(dateString,
            java.time.format.DateTimeFormatter.ofPattern("dd-MM-yyyy"));

        int year;
        LocalDate firstJuly = LocalDate.of(date.getYear(), 7, 1); //changed to
july to demonstrate project, must be june
        LocalDate firstSeptember = LocalDate.of(date.getYear(), 9, 1);

        if (date.isBefore(firstJuly)) {
            year = date.getYear() - 1;
        } else if (date.isEqual(firstSeptember) ||
date.isAfter(firstSeptember)) {
            year = date.getYear();
        } else {
```

```

        throw new IllegalArgumentException("Wrong date");
    }

    LocalDate firstSeptemberOfCSY = LocalDate.of(year, 9, 1);
    boolean isStartWeekOdd;

    if (firstSeptemberOfCSY.getDayOfWeek().getValue() >=
        DayOfWeek.MONDAY.getValue() &&
        firstSeptemberOfCSY.getDayOfWeek().getValue() <=
        DayOfWeek.FRIDAY.getValue())
    {
        isStartWeekOdd = true;
    } else {
        isStartWeekOdd = false;
    }

    LocalDate startDay = firstSeptemberOfCSY.with(DayOfWeek.MONDAY);
    LocalDate endDay = date.with(DayOfWeek.MONDAY);

    long weeksInBetween = ChronoUnit.WEEKS.between(startDay, endDay);
    boolean isWeekOdd;

    if (weeksInBetween % 2 == 0) {
        isWeekOdd = isStartWeekOdd;
    } else {
        isWeekOdd = !isStartWeekOdd;
    }

    return new DateInfo(date.getDayOfWeek().toString(), isWeekOdd);
}

@Override
public List<RawTimetableDayResponse>
getRawTimetableForDay(RawTimetableDayRequest req) {
    return timetableDao.getTimetableDay(req);
}

@Override
public List<SimpleDescription>
getSuitableSubjects(SuitableLessonEntityRequest req) {

```

```

        if (req.getId() == null) {
            req.setId(-1);
        }
        return timetableDao.getSuitableSubjects(req);
    }

    @Override
    public List<SimpleDescription>
getSuitableTeachers(SuitableLessonEntityRequest req) {
        if (req.getId() == null) {
            req.setId(-1);
        }
        return timetableDao.getSuitableTeachers(req);
    }

    @Override
    public List<SimpleDescription>
getSuitableClassrooms(SuitableLessonEntityRequest req) {
        if (req.getId() == null) {
            req.setId(-1);
        }
        return timetableDao.getSuitableClassrooms(req);
    }

    @Override
    public void createLesson(LessonCreateData data) {
        timetableDao.createLesson(data);
    }

    @Override
    public void updateLesson(LessonUpdateData data) {
        timetableDao.updateLesson(data);
    }

    @Override
    public void deleteLesson(int id) {
        timetableDao.deleteLesson(id);
    }

    @Override

```

```

        public List<TimetableDayResponseTeacher>
getTimetableDayForTeacher(TimetableDayRequestTeacher req) {
            return timetableDao.getTimetableDayForTeacher(req);
        }

@Override
        public List<TimetableDayResponseTeacherExt>
getTimetableDayExtForTeacher(DateSearchInfo dsi) {
            DateInfo di = getDateInfo(dsi.getDate());
            return timetableDao.getTimetableDayExtForTeacher(dsi, di);
        }

@Override
        public void planLesson(LessonProperties lp) {
            LocalDate currentDate = LocalDate.now();
            if (LocalDate.parse(lp.getDate(),
java.time.format.DateTimeFormatter.ofPattern("dd-MM-
yyyy")).isAfter(currentDate)) {
                timetableDao.planLesson(lp, getDateInfo(lp.getDate()));
            }
        }

@Override
        public List<JournalRecord> getLessonJournal(int journalId, int teacherId)
{
            return timetableDao.getLessonJournal(journalId, teacherId);
        }

@Override
        public void setPresence(PresenceInfo data) {
            timetableDao.setPresence(data);
        }

@Override
        public void setHomeworkMark(MarkInfo data) {
            timetableDao.setHomeworkMark(data);
        }

@Override
        public void setLessonMark(MarkInfo data) {

```

```
        timetableDao.setLessonMark(data);
    }

    @Override
    public List<RawTimetableDayResponse>
getTimetableDayForStudent(TimetableDayRequestStudent req) {
        return timetableDao.getTimetableDayForStudent(req);
    }

    @Override
    public List<TimetableDayResponseStudentExt>
getTimetableDayExtForStudent(DateSearchInfo dsi) {
        DateInfo di = getDateInfo(dsi.getDate());
        return timetableDao.getTimetableDayExtForStudent(dsi, di);
    }
}
```

Фрагмент коду компонента Client клієнтської частини

```
import {Injectable} from "@angular/core";
import {HttpClient, HttpParams} from "@angular/common/http";
import {Observable} from "rxjs";
import {appLinks} from "../../app.links";
import {LessonClassWise} from "../../admin/models/lessonClassWise";
import {LessonRequest} from "../../admin/models/lessonRequest";
import {LessonDef} from "../../admin/models/lessonDef";
import {SimpleDescr} from "../../admin/models/simpleDescr";
import {SuitableLessonEntityRequest} from
"../../admin/models/suitableLessonEntityRequest";
import {LessonCreateData} from "../../admin/models/lessonCreateData";
import {LessonUpdateData} from "../../admin/models/lessonUpdateData";
import {LessonTeacherWise} from "../../teacher/models/lessonTeacherWise";
import {LessonTeacherWiseExt} from
"../../teacher/models/lessonTeacherWiseExt";
import {JournalRecord} from "../../teacher/models/journalRecord";
import {LessonStudentWise} from "../../student/models/lessonStudentWise";

@Injectable({
  providedIn: 'root'
})
export class TimetableService {

  constructor(private http: HttpClient) {}

  public getTimetableForDay(req: LessonRequest): Observable<LessonClassWise[]>
{
  return this.http.put<LessonClassWise[]>(appLinks.getTimetableDay, req);
}

  public getSuitableSubjects(req: SuitableLessonEntityRequest):
Observable<SimpleDescr[]> {
  return this.http.post<SimpleDescr[]>(appLinks.getSuitableSubjects, req);
}

  public getSuitableTeachers(req: SuitableLessonEntityRequest):
Observable<SimpleDescr[]> {
  return this.http.post<SimpleDescr[]>(appLinks.getSuitableTeachers, req);
}
```

```

    }

    public getSuitableClassrooms(req: SuitableLessonEntityRequest):
Observable<SimpleDescr[]> {
        return this.http.post<SimpleDescr[]>(appLinks.getSuitableClassrooms, req);
    }

    public createLesson(data: LessonCreateData): Observable<void> {
        return this.http.post<void>(appLinks.changeLesson, data);
    }

    public updateLesson(data: LessonUpdateData): Observable<void> {
        return this.http.put<void>(appLinks.changeLesson, data);
    }

    public deleteLesson(id: number): Observable<void> {
        return this.http.delete<void>(appLinks.changeLesson + `/${id}`);
    }

    public getTeacherTimetableForDay(body: any): Observable<LessonTeacherWise[]>
{
        return this.http.put<LessonTeacherWise[]>(appLinks.getTimetableDayTeacher,
body);
    }

    public getTeachersLessonsForDate(body: any):
Observable<LessonTeacherWiseExt[]> {
        return
this.http.put<LessonTeacherWiseExt[]>(appLinks.getTimetableDayExtTeacher,
body);
    }

    public planLesson(body: any): Observable<void> {
        return this.http.post<void>(appLinks.planLesson, body);
    }

    public getJournalForLesson(id: number): Observable<JournalRecord[]> {
        return this.http.get<JournalRecord[]>(appLinks.getLessonJournal + '/' +
id);
    }

```

```

public setPresence(body: any): Observable<void> {
    return this.http.put<void>(appLinks.setPresence, body);
}

public setHomeworkMark(body: any): Observable<void> {
    return this.http.put<void>(appLinks.setHomeworkMark, body);
}

public setLessonMark(body: any): Observable<void> {
    return this.http.put<void>(appLinks.setLessonMark, body);
}

public getStudentsTimetableForDay(body: any): Observable<LessonClassWise[]>
{
    return this.http.put<LessonClassWise[]>(appLinks.getTimetableDayStudent,
body);
}

public getStudentsLessonsForDate(body: any): Observable<LessonStudentWise[]>
{
    return
this.http.put<LessonStudentWise[]>(appLinks.getTimetableDayExtStudent, body);
}

public getStudentsLessonsForDateForParent(body: any):
Observable<LessonStudentWise[]> {
    return
this.http.put<LessonStudentWise[]>(appLinks.getTimetableDayExtParent, body);
}
}

```

Фрагмент коду компонента View клієнтської частини

```
<app-user-skeleton></app-user-skeleton>
<div class="insert-area">
  <div class="scrollable">
    <p class="ia-title">Diary</p>
    <div class="date-input">
      <mat-form-field>
        <input matInput [matDatepicker]="picker"
[value]="dateDisplayForm.value" [min]="minDate" [max]="maxDate"
(dateChange)="onDateChange($event)" ">
        <mat-datepicker-toggle matIconSuffix [for]="picker"></mat-datepicker-
toggle>
        <mat-datepicker #picker></mat-datepicker>
      </mat-form-field>
    </div>
    <div class="tt-div">
      <table>
        <tbody>
          <tr *ngFor="let number of [1, 2, 3, 4, 5, 6, 7, 8]">
            <td class="l-num">{{number}}</td>
            <td class="l-info">
              <span
*ngIf="!lessonsMap?.get(number)">&nbsp;</span>{{lessonsMap?.get(number)?.subje
ctName}}<br>

              <span *ngIf="!lessonsMap?.get(number)">&nbsp;</span>
              <span *ngIf="lessonsMap?.get(number)">Teacher:
</span>{{lessonsMap?.get(number)?.teacherName}}<br>

              <span *ngIf="!lessonsMap?.get(number)">&nbsp;</span>
              <span *ngIf="lessonsMap?.get(number)">Classroom: </span>
              <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.classroom">(not set)</span>
              {{lessonsMap?.get(number)?.classroom}}
            </td>
            <td class="l-descr">
              <span *ngIf="lessonsMap?.get(number)">Lesson type: </span>
              <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.lessonType">(not set)</span>
```

```

        {{lessonsMap?.get(number)?.lessonType}}<br>

        <span *ngIf="lessonsMap?.get(number)">Lesson theme: </span>
        <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.lessonTheme">(not set)</span>
        {{lessonsMap?.get(number)?.lessonTheme}}<br>

        <span *ngIf="lessonsMap?.get(number)">Homework: </span>
        <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.homework">(not set)</span>
        {{lessonsMap?.get(number)?.homework}}<br>
    </td>
    <td class="l-marks">
        <span *ngIf="lessonsMap?.get(number)">Presence: </span>
        <span *ngIf="lessonsMap?.get(number) &&
(lessonsMap?.get(number)?.isPresent === null || chosenDate >
currentDate)">(not set)</span>
        <span *ngIf="chosenDate <= currentDate &&
lessonsMap?.get(number)?.isPresent === true">present</span>
        <span *ngIf="chosenDate <= currentDate &&
lessonsMap?.get(number)?.isPresent === false">absent</span><br>

        <span *ngIf="lessonsMap?.get(number)">Lesson mark: </span>
        <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.lessonMark">(not set)</span>
        {{lessonsMap?.get(number)?.lessonMark}}<br>
        <span *ngIf="lessonsMap?.get(number)">Homework mark: </span>
        <span *ngIf="lessonsMap?.get(number) &&
!lessonsMap?.get(number)?.homeworkMark">(not set)</span>
        {{lessonsMap?.get(number)?.homeworkMark}}<br>
    </td>
    <td *ngIf="!lessonsMap?.get(number)">
        <span>&nbsp;</span>
    </td>
</tr>
</tbody>
</table>
</div>
</div>
</div>

```

Фрагмент коду компонента ViewModel клієнтської частини

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormControl } from "@angular/forms";
import { NzNotificationService } from "ng-zorro-antd/notification";
import { TimetableService } from "../../core/services/timetable.service";
import { MatDatepickerInputEvent } from "@angular/material/datepicker";
import { LessonStudentWise } from "../../models/lessonStudentWise";

@Component({
  selector: 'app-timetable-day-ext-student',
  templateUrl: './timetable-day-ext-student.component.html',
  styleUrls: ['./timetable-day-ext-student.component.scss']
})
export class TimetableDayExtStudentComponent implements OnInit {

  lessonsMap: Map<number, LessonStudentWise> = new Map();
  dateDisplayForm: FormControl = new FormControl(new Date());
  chosenDate: Date = new Date();
  currentDate: Date = new Date();
  minDate: Date;
  maxDate: Date;

  constructor(
    private notification: NzNotificationService,
    private timetableService: TimetableService
  ) {
    const date: Date = new Date();
    if (date >= new Date(date.getFullYear(), 8, 1)) {
      this.minDate = new Date(date.getFullYear(), 8, 1);
      this.maxDate = new Date(date.getFullYear() + 1, 6, 0);
    } else {
      this.minDate = new Date(date.getFullYear() - 1, 8, 1);
      this.maxDate = new Date(date.getFullYear() ,6, 0);
    }
  }

  ngOnInit(): void { this.loadLessonList(); }

  onChange(event: MatDatepickerInputEvent<Date>) {
```

```

        if (event.value === null) {
            this.notify("Date can't be null. Please select a valid date.")
        } else {
            this.chosenDate = event.value;
            this.loadLessonList();
        }
    }

    loadLessonList() {
        this.timetableService.getStudentsLessonsForDate({date:
this.formatDate(this.chosenDate)}).subscribe(
            (list) => {
                this.lessonsMap = new Map();
                list.forEach((lesson) => {
                    this.lessonsMap?.set(lesson.lessonNo, lesson);
                });
            },
            () => {
                this.notify("Oh snap. Something crashed. Please try again later.");
            }
        );
    }

    private formatDate(date: Date) {
        let day: number = date.getDate();
        let month: number = date.getMonth() + 1;
        let year: number = date.getFullYear();

        let dayStr: string = day < 10 ? '0' + day : day.toString();
        let monthStr: string = month < 10 ? '0' + month : month.toString();

        return dayStr + '-' + monthStr + '-' + year;
    }

    notify(text: string) {
        this.notification
            .blank("", text)
    }
}

```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ
ІГОРЯ СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

**ВЕБЗАСТОСУНОК ДЛЯ ВЕДЕННЯ ШКІЛЬНИХ ЖУРНАЛІВ
ТА УЧНІВСЬКИХ ЩОДЕННИКІВ**

Виконав: студент групи КП-91 Кінаш Дарій Олегович

Керівник: доцент кафедри ПЗКС, к.т.н., доцент Люшенко Леся Анатоліївна

Київ - 2023

ПОСТАНОВКА ЗАДАЧІ

Метою даного дипломного проєкту є створення вебзастосунку для управління навчальним процесом онлайн в шкільному навчальному закладі, а саме: управління викладацькою діяльністю вчителів, освітнім процесом учнів та організація комунікації між вчителями, батьками та учнями. Серед засобів управління передбачено: реєстрацію користувачів, управління розкладом, ведення журналів та щоденників, моніторинг успішності учнів та забезпечення комунікації між батьками, вчителями та учнями.

Аналіз:

- аналіз предметної області та аналогічних застосунків
- визначення вимог до застосунку

Розробка:

- обрання засобів реалізації
- проєктування
- розробка

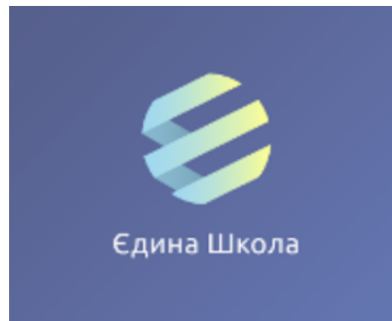
Результати:

- порівняння з наявними аналогами
- рекомендації з подальшого вдосконалення та підтримки

АКТУАЛЬНІСТЬ

- Перехід більшості сфер людської життєдіяльності у онлайн формат.
- Стрімкий розвиток ринку застосунків для онлайн освіти.
- Фактори, що перешкоджають очному навчанню.
- Суттєві недоліки наявних рішень для організації онлайн навчання.

НАЯВНІ АНАЛОГИ



ЗАГАЛЬНІ ВИМОГИ ДО ЗАСТОСУНКУ

- Можливість керівництва школи реєструвати вчителів, учнів та батьків.
- Можливість керівництва школи створювати та редагувати розклад.
- Відмічання вчителем присутніх.
- Виставлення вчителем оцінок.
- Задання вчителем домашнього завдання.
- Перегляд учнем та батьками розкладу, завдань та оцінок учня.
- Перегляд вчителем власного розкладу.
- Можливість переглядати показники успішності учня.
- Можливість комунікації між учнями та вчителем та між батьками та вчителем за допомогою чату.

ЗАСОБИ РЕАЛІЗАЦІЇ

Тип ПЗ – вебзастосунок

Мови програмування:

серверна частина – **Java**,
фреймворк **Spring**

клієнтська частина – **TypeScript**,
фреймворк **Angular**

СУБД - **PostgreSQL**



АРХІТЕКТУРА СИСТЕМИ

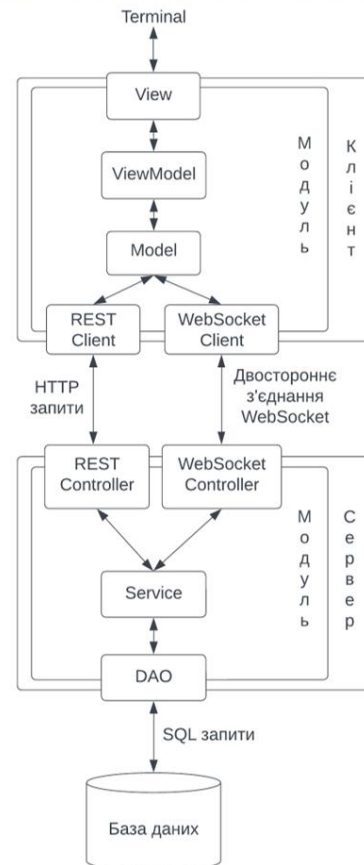
Вебзастосунок реалізує клієнт-серверну архітектуру.

Компоненти серверної частини:

1. Controller
2. Service.
3. DAO.

Компоненти клієнтської частини:

1. View.
2. Model.
3. ViewModel.
4. Client.



ФУНКЦІОНАЛЬНІСТЬ УЧНЯ



ФУНКЦІОНАЛЬНІСТЬ БАТЬКІВ



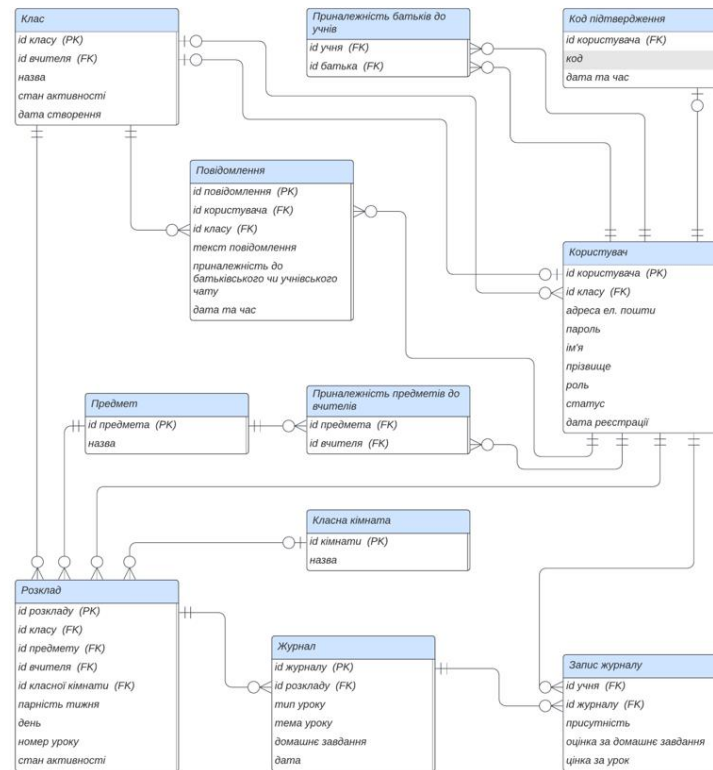
ФУНКЦІОНАЛЬНІСТЬ ВЧИТЕЛЯ



ФУНКЦІОНАЛЬНІСТЬ КЕРІВНИЦТВА ШКОЛИ



ЛОГІЧНА МОДЕЛЬ БАЗИ ДАНИХ



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Olha
Koval

Log out

Navigation

- Users
- Timetable
- Subjects
- Classes
- Settings

User list

Add student

Add teacher

Add parent

Search for users

Student

Enter name or surname

Search

Email	Name	Surname	Class	Edit
mykgorb@swd.com	Mykyta	Gorbach	17-B	Edit
j.doe@wfr.com	John	Doe	17-B	Edit
opchik@edu.ua	Olena	Pchilka	17-A	Edit
g.d@srv.com	Gustav	Donuoghue	17-A	Edit
mstg@ukredu.com	Mariia	Stugna	17-A	Edit
i.ost@srv.com	Iosif	Ostapenko	17-B	Edit
nosok@edu.ua	Nataliya	Sokolova	17-B	Edit
j.d@gmail.com	Ostap	Vyshnia	17-B	Edit
d.v@service.org.ua	Dmytro	Vovk	17-B	Edit
i.gushchyk@cyan.to	Inna	Gushchyk	17-B	Edit

< 1 of 2 >

ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Olha
Koval

Log out

Navigation

- Users
- Timetable
- Subjects
- Classes
- Settings

Timetable

Choose class: 17-B

First week

Monday	Tuesday	Wednesday	Thursday	Friday
Foreign literature 1 Palamarchuk Tetiana Classroom 4	Geography 1 Koval Johann Classroom 4	English 1 Grabar Olga Classroom 6	World history 1 Kirk John Classroom 4	Geometry 1 Opaliy Ivan Classroom 3
Geometry 2 Opaliy Ivan Classroom 3	Geometry 2 Opaliy Ivan Classroom 4	First aid 2 Brodna Mariya Classroom 2		
Algebra 3 Opaliy Ivan Classroom 5	Chemistry 3 Khrushch Ivan Classroom 4			
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7

Headquarters: Kyiv, Ukraine
Contacts: sdjms.recovery@gmail.com

SDJMS - Copyright 2023 - All Rights Reserved



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Ivan
Opalyi [Log out](#)

Navigation

- [📅 Timetable](#)
- [📖 Journal](#)
- [⚙️ Settings](#)

Timetable

First week

Monday	Tuesday	Wednesday	Thursday	Friday
1	1	1	Geometry 1 Class 17-A Classroom 5	Geometry 1 Class 17-B Classroom 3
Geometry 2 Class 17-B Classroom 3	Geometry 2 Class 17-B Classroom 4	2	2	2
Algebra 3 Class 17-B Classroom 5	Geometry 3 Class 17-A Classroom 3	3	Algebra 3 Class 17-A Classroom 5	3
Algebra 4 Class 17-A Classroom 4	4	Algebra 4 Class 17-A Classroom 6	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8

Headquarters: Kyiv, Ukraine
Contacts: sdjms.recovery@gmail.com

SDJMS - Copyright 2023 - All Rights Reserved



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Ivan
Opalyi

Log out

Navigation

- 📅 Timetable
- 📅 Journal
- ⚙️ Settings

Journal

30/05/2023



Geometry	Lesson type: regular	Open lesson
1 Class 17-B	Lesson theme: Triangle similarity theoremes	
Classroom 5	Homework: Textbook pp. 37 - 38 ex. 1, 2, 4 (b), 6, 8, 9	
Algebra	Lesson type: test	Open lesson
2 Class 17-B	Lesson theme: Derivatives. Introduction	
Classroom 5	Homework: read textbook pp. 123-128, do ex. 1, 2	
3		
4		
5		
6		
7		
8		

ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Ivan
Opaliy

Log out

Navigation

- 📅 Timetable
- 📅 Journal
- ⚙️ Settings

Journal

Back

Student	Presence	Homework mark	Lesson mark
Dirk Johnatan	☒	10	12
Doe John	☒	11	12
Gorbach Mykyta	☒	2	Not set
Gushchuk Inna	☐	Not set	Not set
Ostapenko Iosif	☒	10	Not set
Sokolova Nataliya	☒	12	11
Stark Joanna	☒	12	Not set

Headquarters: Kyiv, Ukraine
Contacts: sdjms.recovery@gmail.com

SDJMS - Copyright 2023 - All Rights Reserved



ПРИКЛАД РОБОТИ ЗАСТОСУНКУ

SDTT

Nataliya
Sokolova

Log out 

Navigation

-  Timetable
-  Diary
-  Settings

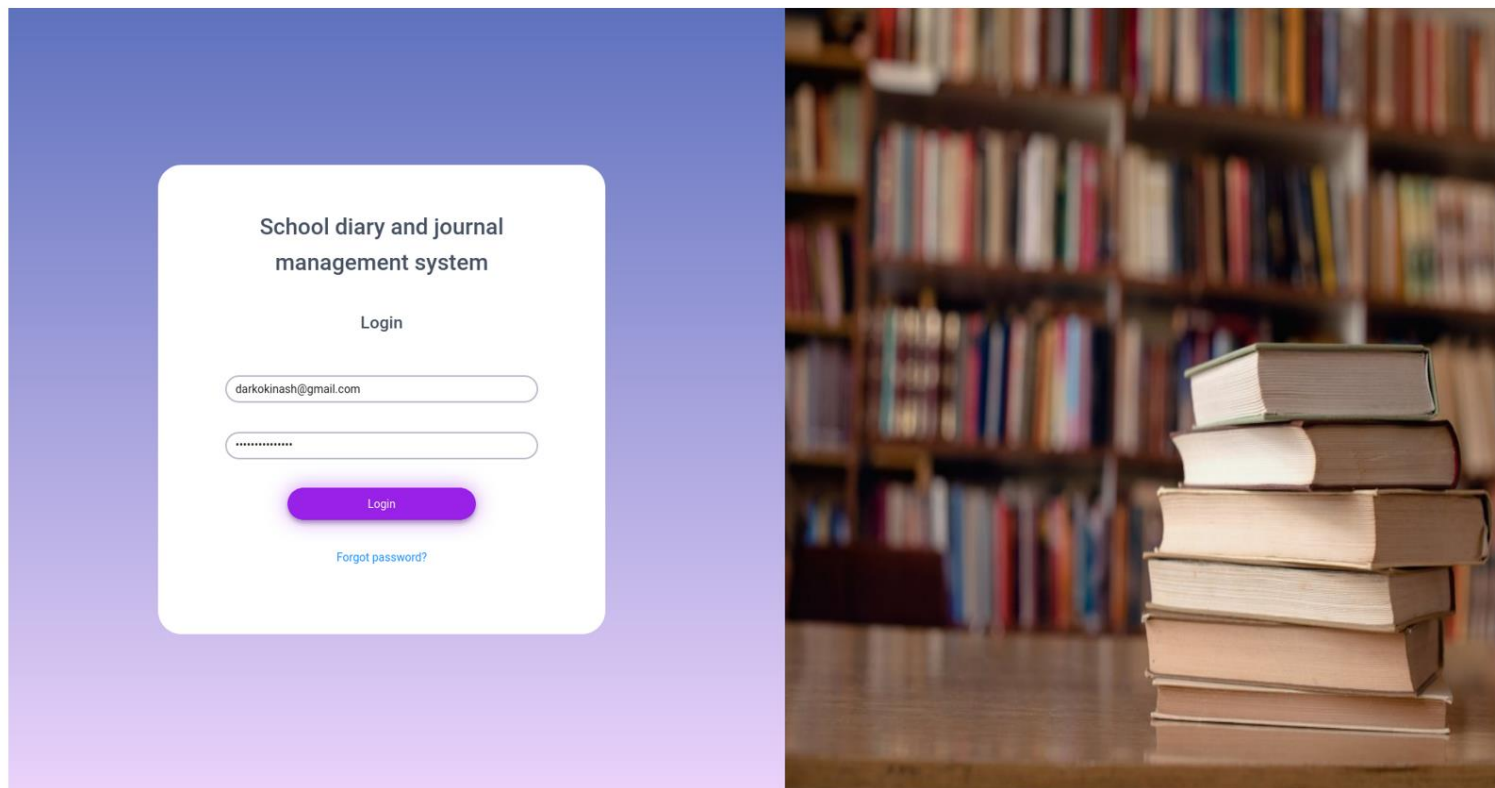
Diary

30/05/2023



Geometry	Lesson type: regular	Presence: present
1 Teacher: Opalyi Ivan	Lesson theme: Triangle similarity theoremes	Lesson mark: 11
Classroom: 5	Homework: Textbook pp. 37 - 38 ex. 1, 2, 4 (b), 6, 8, 9	Homework mark: 12
Algebra	Lesson type: test	Presence: present
2 Teacher: Opalyi Ivan	Lesson theme: Derivatives. Introduction	Lesson mark: 10
Classroom: 5	Homework: read textbook pp. 123-128, do ex. 1, 2	Homework mark: 10
3		
4		
5		
6		
7		
8		

ПРИКЛАД РОБОТИ ЗАСТОСУНКУ



ТЕСТУВАННЯ ЗАСТОСУНКУ

Приклад тестування на навантаження сервера.

Надсилення 2000 паралельних запитів для отримання розкладу, завдань та оцінок учня за певне число.

```
✓ sendHttpRequest 9 sec 39 ms
✓ sendHttpRequest 9 sec 630 ms
✓ sendHttpRequest 9 sec 58 ms
✓ sendHttpRequest 9 sec 81 ms
✓ sendHttpRequest 9 sec 604 ms
✓ sendHttpRequest 9 sec 440 ms
✓ sendHttpRequest 9 sec 239 ms
✓ sendHttpRequest 9 sec 640 ms
✓ sendHttpRequest 9 sec 656 ms
✓ sendHttpRequest 9 sec 604 ms
✓ sendHttpRequest 9 sec 542 ms
✓ sendHttpRequest 9 sec 294 ms

12:13:06.152 [Finalizer] DEBUG org.apache.http.impl.conn.Default
12:13:06.152 [Finalizer] DEBUG org.apache.http.impl.conn.Default
Seconds executing: 9

=====
Default Suite
Total tests run: 2000, Passes: 2000, Failures: 0, Skips: 0
=====
```

ТЕСТУВАННЯ ЗАСТОСУНКУ

Приклад тестування реєстрації користувачів.

Користувач не може бути зареєстрований з адресою ел. пошти, вже наявною у базі даних.

```
1 usage
private static String[] bodyProvider() {
    return new String[] {
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nukedjr@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Derek\", \"surname\": \"Newhouse\"}\",
        "{\"email\": \"nuked@edu.ua\", \"name\": \"Lana\", \"surname\": \"Newhouse\"}\"
    };
}
```

RegistrationTest (com.example.SDJMS_Backend)	6 sec 58 ms
testEmailUniqueness(String)	6 sec 58 ms
✓ [1] jsonBody={"email": "nuked@edu.ua", "name": "Derek", "surname": "Newhouse"}	3 sec 502 ms
✓ [2] jsonBody={"email": "nukedjr@edu.ua", "name": "Derek", "surname": "Newhouse"}	2 sec 348 ms
✗ [3] jsonBody={"email": "nuked@edu.ua", "name": "Derek", "surname": "Newhouse"}	132 ms
✗ [4] jsonBody={"email": "nuked@edu.ua", "name": "Lana", "surname": "Newhouse"}	76 ms

ТЕСТУВАННЯ ЗАСТОСУНКУ

Приклад тестування методу getDateInfo.

Перевірка методу на коректність визначення властивостей дати

```
1 usage
private static Stream<DataProviderObj> dateProvider() {
    return Stream.of(
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "MONDAY", isWeekOdd: true)),
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "MONDAY", isWeekOdd: false)),
        new DataProviderObj( date: "15-05-2023", new DateInfo( day: "TUESDAY", isWeekOdd: false)),
        new DataProviderObj( date: "10-05-2023", new DateInfo( day: "WEDNESDAY", isWeekOdd: true)),
        new DataProviderObj( date: "15-08-2023", new DateInfo( day: "MONDAY", isWeekOdd: true)),
        new DataProviderObj( date: "1342343", new DateInfo( day: "MONDAY", isWeekOdd: true))
    );
}
```

▼	! DateInfoTest (com.example.SDJMS_Backend)	32 ms
▼	! testGetDateInfo(DataProviderObj)	32 ms
✗	[1] obj=com.example.SDJMS_Backend.DateInfoTest\$	30 ms
✓	[2] obj=com.example.SDJMS_Backend.DateInfoTest\$DataPr	
✗	[3] obj=com.example.SDJMS_Backend.DateInfoTest\$DataPr	
✓	[4] obj=com.example.SDJMS_Backend.DateInfoTest\$C	1 ms
!	[5] obj=com.example.SDJMS_Backend.DateInfoTest\$DataPr	
!	[6] obj=com.example.SDJMS_Backend.DateInfoTest\$C	1 ms

КОНКУРЕНТНІ ПЕРЕВАГИ

- Можливість перегляду успішності учня.
- Наявність засобів комунікації між учнями та класним керівником та між батьками та класним керівником.
- Повна відповідність функціональності ролей користувачів їх завданням у процесах функціонування закладу середньої освіти.
- Можливість використовувати застосунок як єдиний інформаційний ресурс.
- Використання сучасних засобів та технологій під час розробки.

НАПРЯМКИ РОЗВИТКУ

Застосунок можна поліпшити, реалізуювши наступні засоби:

- збереження зведених оцінок учня по предметах за навчальний рік у документ, ідентичний табелю, з можливістю експорту документа;
- видалення даних учнів та класів після вичерпання терміну зберігання цих даних;
- приватний чат учня з вчителем, який викладає в учня;
- сповіщення про нові повідомлення в чатах, задання домашнього завдання, тощо.

ВИСНОВКИ

1. Виконано аналіз предметної області та наявних аналогів.
2. Сформовано вимоги до програмного продукту.
3. Поаналізовано та обрано засоби для розробки застосунку.
4. Спроектовано архітектуру застосунку, логічну та фізичну схеми БД.
5. Реалізовано та протестовано функціональність застосунку.
6. Визначено напрями подальшого розвитку застосунку.

Розробка виконана у повному обсязі згідно з положенням Технічного завдання, тестування продукту проведено відповідно до затвердженої програми та методики тестування.

РЕЗУЛЬТАТ ПЕРЕВІРКИ НА УНІКАЛЬНІСТЬ



Ім'я користувача:
Люшенко Леся Анатоліївна gmail

Дата перевірки:
11.06.2023 15:09:59 EEST

Дата звіту:
11.06.2023 19:47:49 EEST

ID перевірки:
1015547631

Тип перевірки:
Doc vs Internet + Library

ID користувача:
91603

Назва документа: КП91_Кінаш

Кількість сторінок: 55 Кількість слів: 8424 Кількість символів: 64070 Розмір файлу: 965.79 KB ID файлу: 1015200033

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.59%
Схожість

Найбільша схожість: 0.34% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/35651/1/Nechyporenko_bakala).

0.94% Джерела з Інтернету 15 Сторінка 57

1.37% Джерела з Бібліотеки 158 Сторінка 57

ДЯКУЮ ЗА УВАГУ!

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2022 р.

ВЕБЗАСТОСУНОК ДЛЯ ВЕДЕННЯ ШКІЛЬНИХ ЖУРНАЛІВ
ТА УЧНІВСЬКИХ ЩОДЕННИКІВ

Програма та методика тестування

ДП.045440-04-51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дарій КІНАШ

2022

ЗМІСТ

1. Об'єкт випробувань	3
2. Мета тестування	3
3. Методи тестування.....	3
4. Засоби та порядок тестування.....	4

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для ведення шкільних журналів та учнівських щоденників “SDTT”, що реалізує клієнт-серверну архітектуру та написаний на мовах TypeScript та Java відповідно.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Функціональна працездатність елементів сторінок вебзастосунку.
2. Забезпечення належного рівня безпеки даних.
3. Зручність роботи з вебзастосунком.
4. Відповідність реалізованого вебзастосунку вимогам технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом White Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. При тестуванні використовуються наступні методи:

1. Функціональне тестування.
2. Модульне тестування.
3. Інтеграційне тестування.
4. Тестування продуктивності програмного забезпечення, зокрема навантажувальне тестування.
5. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується вручну та за допомогою написання програмних тестів на мові Java засобами фреймворку JUnit.

Працездатність вебзастосунку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Програмного модульного тестування окремих модулів.
4. Ручного інтеграційного тестування взаємодії модулів.
5. Функціонального тестування коду.
6. Тестування при максимальному навантаженні.
7. Тестування зручності використання інтерфейсу.

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2023 р.

ВЕБЗАСТОСУНОК ДЛЯ ВЕДЕННЯ ШКІЛЬНИХ ЖУРНАЛІВ
ТА УЧНІВСЬКИХ ЩОДЕННИКІВ

Керівництво користувача

ДП.045440-05-34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Леся ЛЮШЕНКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дарій КІНАШ

2023

ЗМІСТ

1. Опис вебсторінок, доступних всім типам користувачів	3
2. Опис вебсторінок, доступних учню	5
3. Опис вебсторінок, доступних батькам.....	6
4. Опис вебсторінок, доступних вчителю.....	7
5. Опис вебсторінок, доступних керівництву школи	10

1. Опис вебсторінок, доступних всім типам користувачів

- Сторінка входу в обліковий запис. На сторінці доступна форма для входу (введення логіну та паролю) та посилання на сторінку відновлення паролю. Після успішного входу (проходження авторизації та автентифікації) користувач потрапляє на сторінку налаштувань облікового запису. Сторінка зображена на рис. 1.

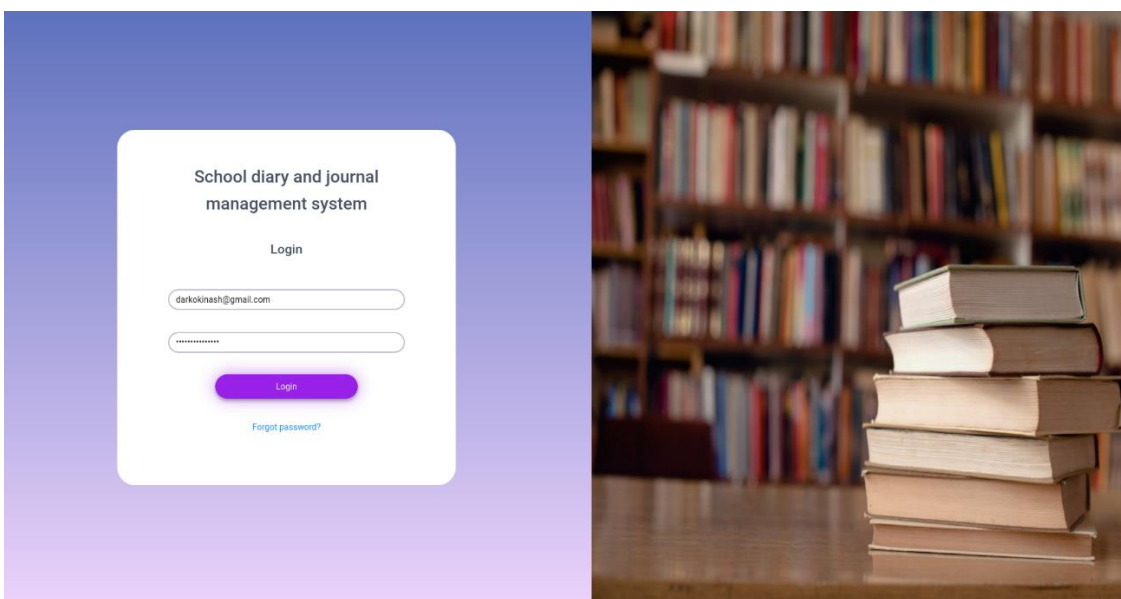


Рис. 1. Сторінка входу в обліковий запис

- Сторінка відновлення паролю. На сторінці доступна форма для введення адреси ел. пошти. Якщо існує користувач з такою адресою, на дану адресу буде відправлено лист з посиланням на відновлення паролю. Сторінка зображена на рис. 2.

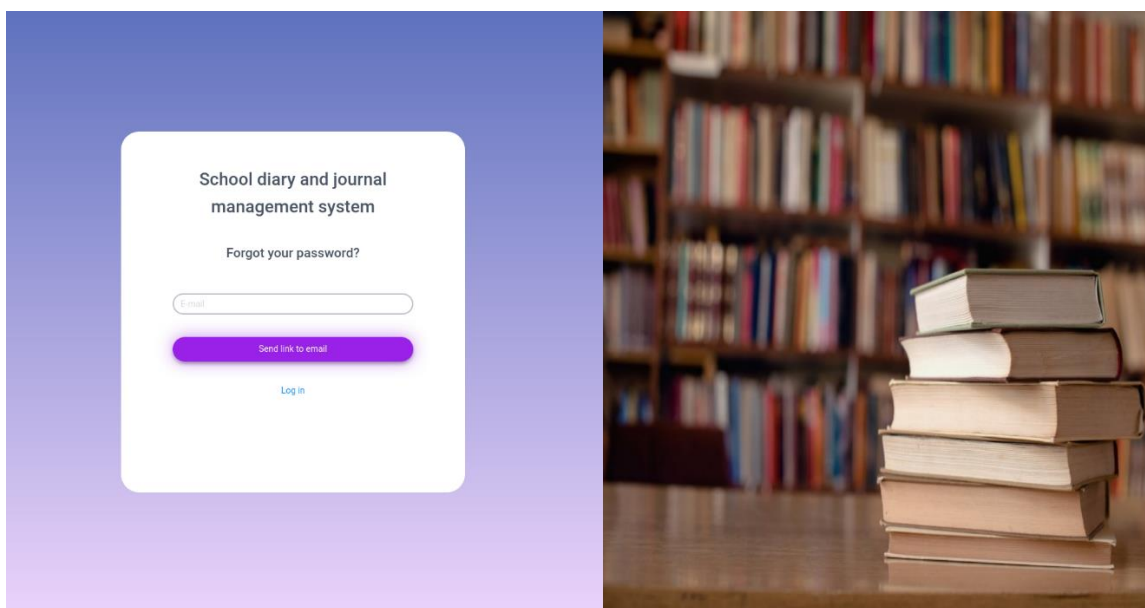


Рис. 2. Сторінка відновлення паролю

- Сторінка налаштувань облікового запису. На сторінці доступна форма для зміни паролю. Необхідні для заповнення поля – поле для старого пароля, поле для нового пароля та поле підтвердження нового пароля. Сторінка зображена на рис. 3.

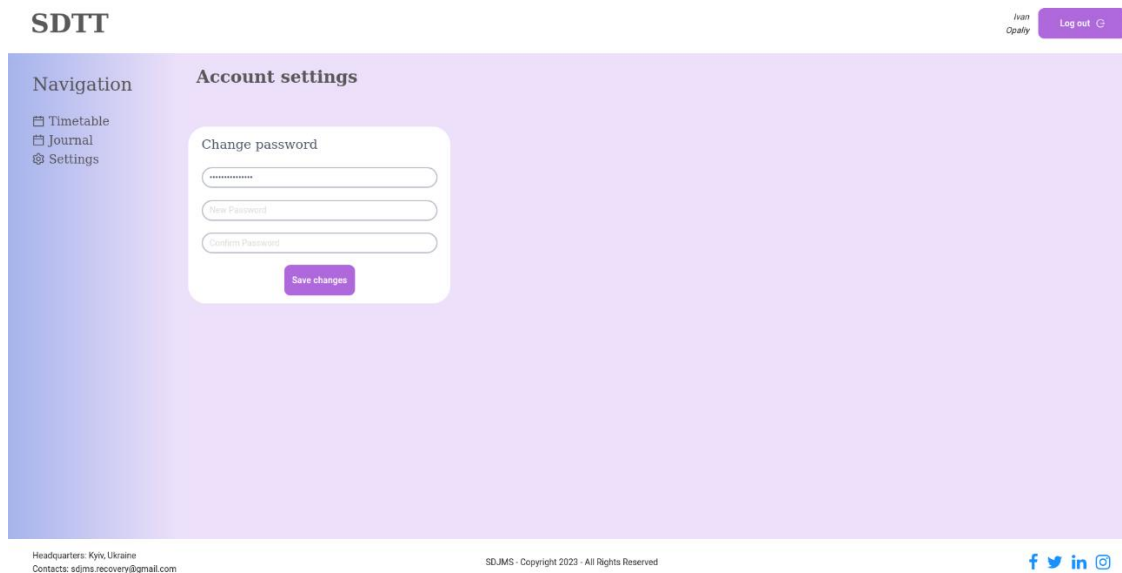


Рис. 3. Сторінка налаштувань облікового запису

2. Опис вебсторінок, доступних учню

- Сторінка розкладу класу учня – розписані будні дні парного та непарного тижнів з назвами уроків, прізвищем та ім'ям вчителя та номером (назвою) кабінету, якщо такий номер було вказано при створенні розкладу. Сторінка зображена на рис. 4.

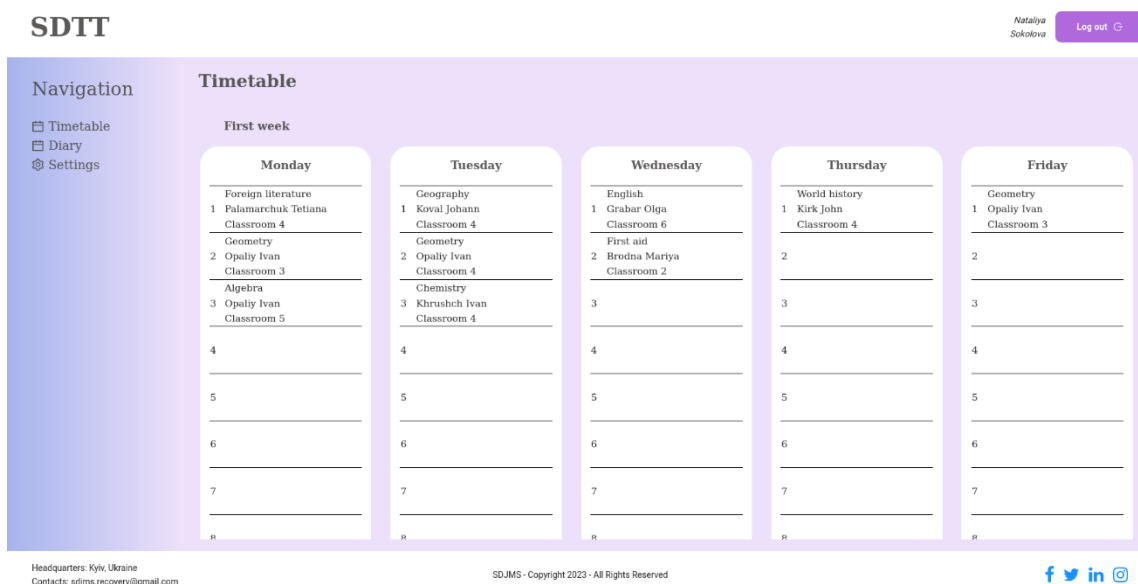


Рис. 4. Сторінка розкладу класу учня

- Сторінка з можливістю переглянути детальну інформацію про розклад класу учня за певне число поточного навчального року. Вгорі сторінки розміщене поле з можливістю вибору дати. Під ним знаходиться таблиця з уроками, вчителями, що їх ведуть, присутністю учня, домашнім завданням та оцінками за уроки за обрану дату. Сторінка зображена на рис. 5.

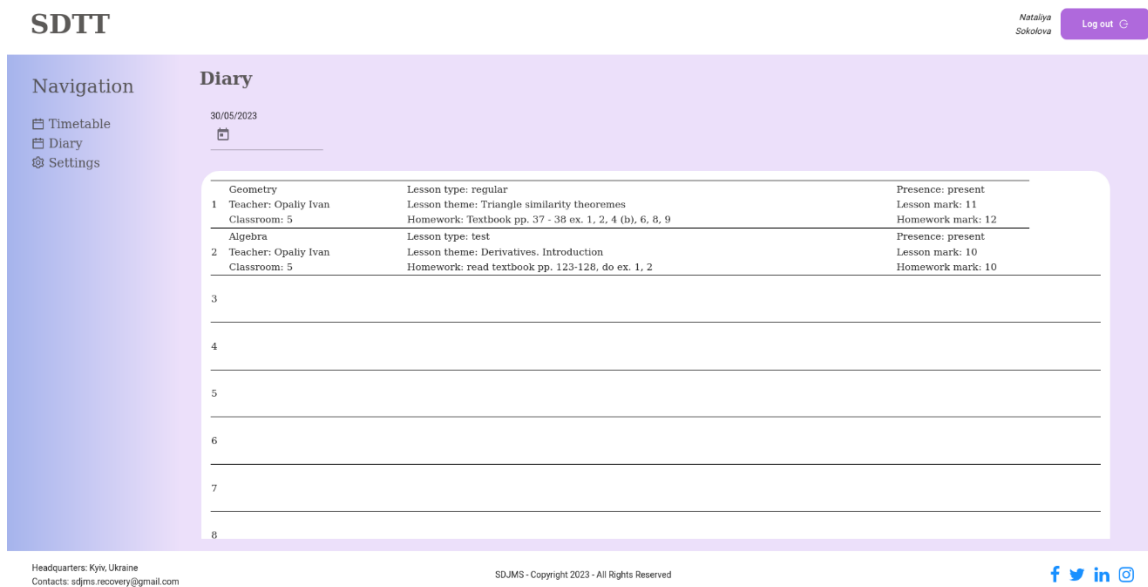


Рис. 5. Сторінка детальної інформації про розклад учня

- Сторінка перегляду успішності за певний період поточного навчального року. Вгорі сторінки розміщене поле з можливістю вибору діапазону дат. Під ним знаходиться список предметів, які учень вивчає у поточному навчальному році та середні бали з предметів за обраний період.
- Сторінка учнівського чату – для комунікації з однокласниками та класним керівником – можливість переглядати та надсилати повідомлення в режимі реального часу.

3. Опис вебсторінок, доступних батькам

- Сторінка розкладу класу учня, що прив'язаний до батьків – розписані будні дні парного та непарного тижнів з назвами уроків, прізвищем та ім'ям вчителя та номером (назвою) кабінету, якщо такий номер було вказано при створенні розкладу.

- Сторінка з можливістю переглянути детальну інформацію про розклад класу учня, що прив'язаний до батьків, за певне число поточного навчального року. Вгорі сторінки розміщене поле з можливістю вибору дати та список з можливістю вибрати учня, якщо прив'язаних учнів більше одного. Під ним знаходиться таблиця з уроками, вчителями, що їх ведуть, присутністю учня, домашнім завданням та оцінками за уроки за обрану дату.
- Сторінка перегляду успішності учня, що прив'язаний до батьків, за певний період поточного навчального року. Вгорі сторінки розміщене поле з можливістю вибору діапазону дат та список з можливістю вибрати учня, якщо прив'язаних учнів більше одного. Під ним знаходиться список предметів, які учень вивчає у поточному навчальному році та середні бали з предметів за обраний період.
- Сторінка батьківського чату – для комунікації з батьками та класним керівником класу учня, що прив'язаний до батька – можливість переглядати та надсилати повідомлення.

4. Опис вебсторінок, доступних вчителю

- Сторінка розкладу вчителя – розписані будні дні парного та непарного тижнів з назвами уроків, назвами класів та номерами (назвами) кабінетів, якщо такі номери було вказано при створенні розкладу.
- Сторінка з можливістю переглянути детальну інформацію про розклад вчителя за певне число поточного навчального року. Сторінка зображена на рис. 6. Вгорі сторінки розміщене поле з можливістю вибору дати. Під ним знаходиться таблиця з уроками, класами та домашнім завданням за обрану дату. Якщо обрана дата більша за поточну, то при натисканні на кнопку “Plan lesson” з’являється форма

для визначення теми, типу уроку та задання домашнього завдання на майбутній урок, зображена на рис. 7. Якщо обрана дата менша або рівна поточній даті, то при натисканні на кнопку “Open lesson” буде відкрита сторінка журналу класу на даний урок за обрану дату з можливістю відмічати присутніх та виставляти оцінки, зображена на рис. 8.

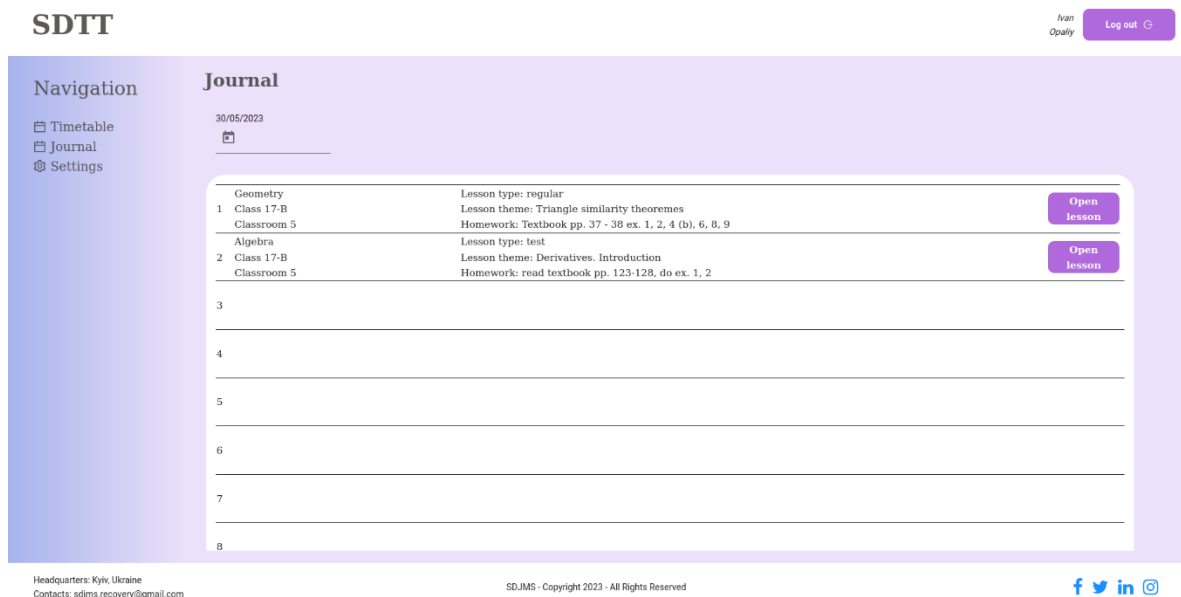


Рис. 6. Сторінка детальної інформації про розклад вчителя

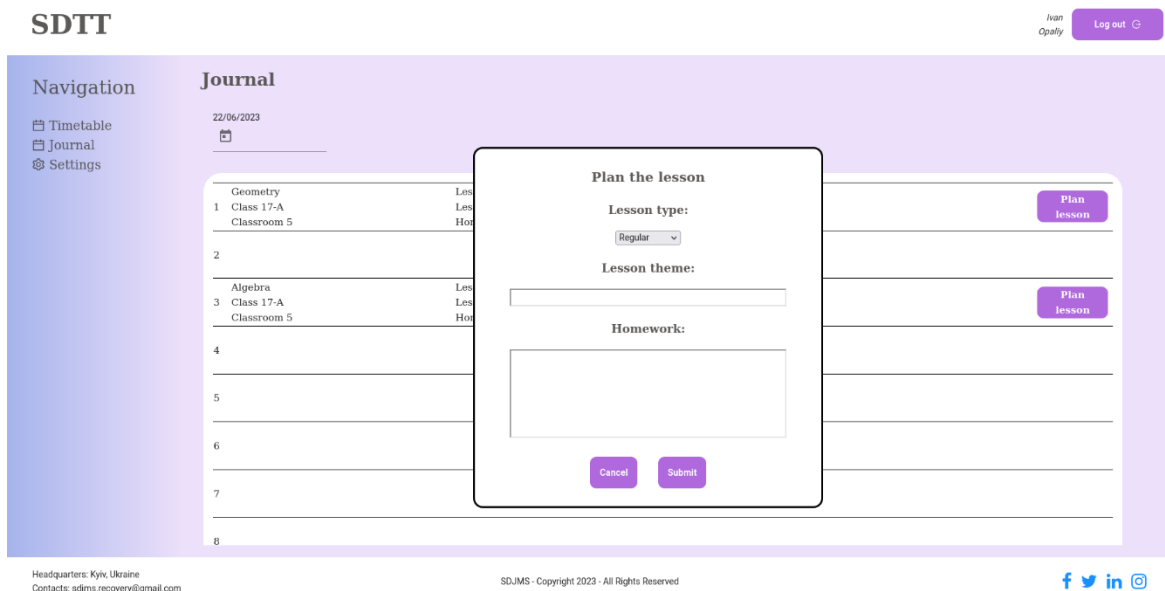


Рис. 7. Форма планування уроку

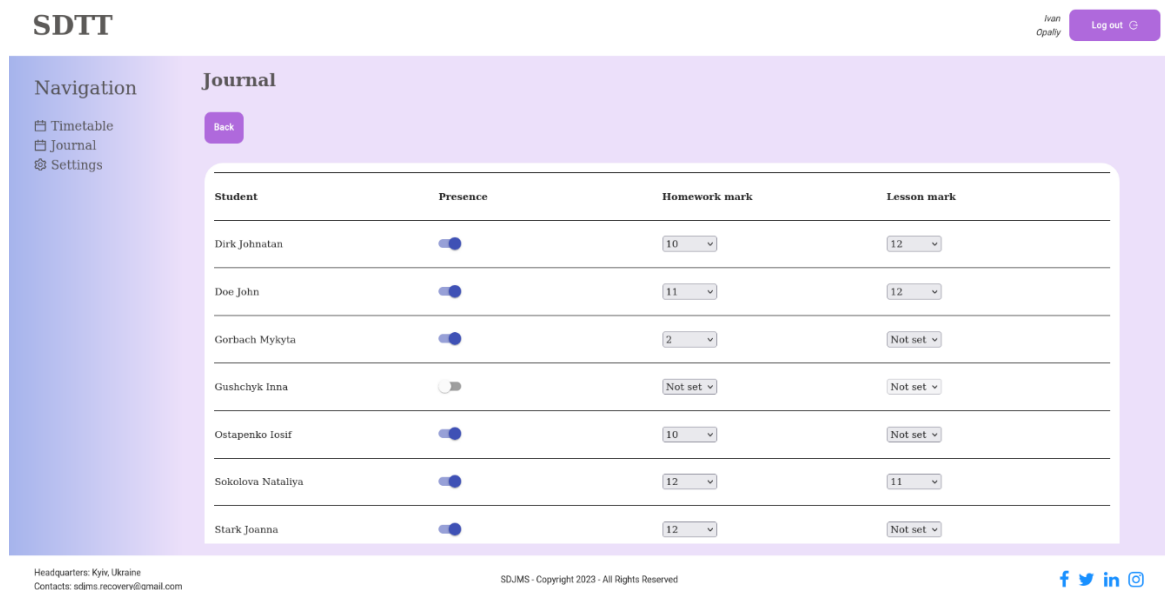


Рис. 8. Сторінка журналу класу за певний урок

- Сторінка перегляду успішності – середній бал учнів певного класу з предмету, який веде викладач, за вибраний період. Вгорі сторінки розміщені списки для вибору класу та предмету і поле з можливістю

вибору діапазону дат. Нижче знаходиться список учнів та їх середні бали з обраного предмету за обраний період.

- Сторінка учнівського чату класу, класним керівником якого він є, з можливістю переглядати та надсилати повідомлення.
- Сторінка батьківського чату класу, класним керівником якого він є, з можливістю переглядати та надсилати повідомлення.

5. Опис вебсторінок, доступних керівництву школи

- Сторінка реєстрації та керування користувачами — наявний пагінований список користувачів, який можна фільтрувати по ролі користувача та по імені користувача. Для учнів та вчителів наявні посилання для переходу на сторінки прив'язки батьків та предметів відповідно. Для кожного користувача є кнопка для редагування його електронної адреси, імені та прізвища. Також наявні кнопки для додавання учнів, вчителів та батьків. Сторінка зображена на рис. 9.

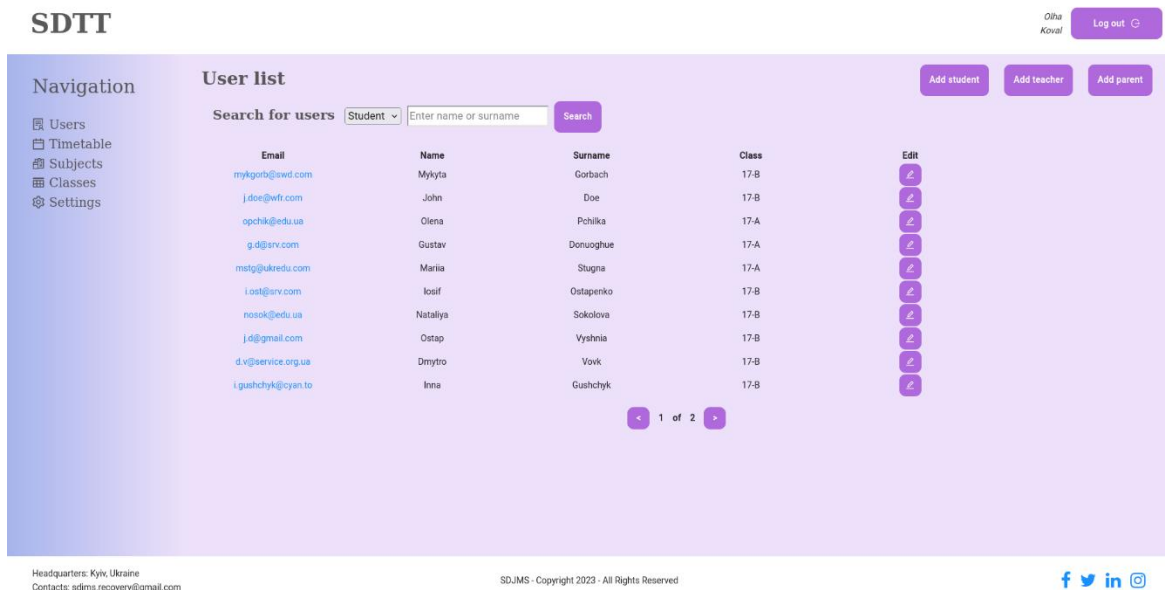


Рис. 9. Сторінка реєстрації та керування користувачами

- Сторінка додавання предметів – відображення списку наявних предметів та форма додавання нового предмету.
- Сторінка додавання класів – відображення списку наявних класів та форма додавання нового класу.
- Сторінка прив'язки учнів до класів та до батьків – форми для прив'язки учня до класу та для прив'язки батьків до учня.
- Сторінка прив'язки вчителів до класу та до предметів – форми для прив'язки вчителя до класу (отримання статусу класного керівника) та для прив'язки предметів до вчителя.
- Сторінка створення та редагування розкладу. На сторінці відображено робочі дні парного та непарного тижнів. Для обраного класу зі списку для уроку певного дня можна задати предмет, вчителя та класну кімнату. Сторінка зображена на рис. 10.

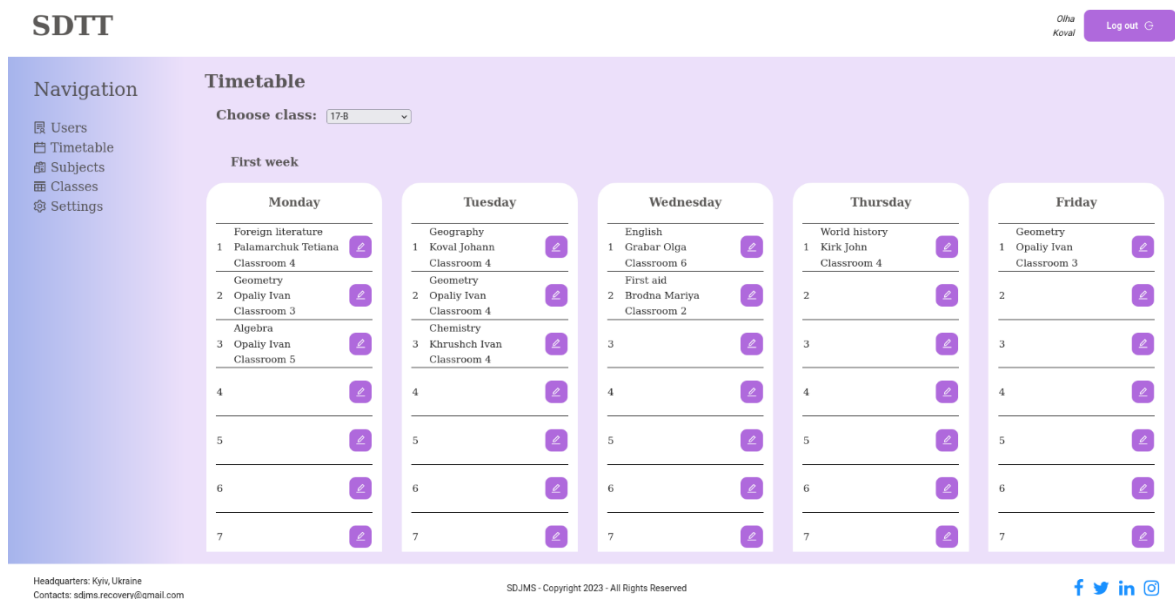


Рис. 10. Сторінка створення та редагування розкладу