МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ З УРАХУВАННЯМ МАТЕРІАЛЬНО-ТЕХНІЧНИХ ВИМОГ ТА РОЗТАШУВАННЯ АУДИТОРІЙ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Максим ПАСІЧЕНКО**

«__» _____ 20__ р.

Керівник роботи
канд. техн. наук,
доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступінь	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
« » _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну магістерську роботу здобувача вищої освіти
Пасіченко Максим

1. Тема кваліфікаційної роботи "Програмне забезпечення формування розкладу занять з урахуванням матеріально-технічних вимог та розташування аудиторій" затверджена наказом ректора ЧНУ ім. Петра Могили № 200 від «3» листопада 2024 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2024 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Система керування розкладом з можливістю автоматичної генерації розкладу

4. Перелік питань, що підлягають розробці:

- визначення цілей, завдань та вимог до системи;
- аналіз існуючих систем
- проєктування загальної архітектури інформаційної системи, включаючи систему зберігання та обробки даних;
- створення схеми бази даних, яку можна використовувати для зберігання та отримання інформації;
- розробка процесів і проміжних систем для імпорту інформації в інформаційну систему;
- впровадження алгоритмів для обробки розкладу

5. Перелік графічних матеріалів:

Презентація, рисунки, таблиці

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 20___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Програмне забезпечення формування розкладу занять з урахуванням матеріально-технічних вимог та розташування аудиторій

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	25.06.2024 р.	30.06.2024 р.	Виконано
2.	Огляд літератури за темою роботи	13.09.2024 р.	19.09.2024 р.	Виконано
3.	Аналіз предметної області	24.09.2024 р.	26.09.2024 р.	Виконано
4.	Розробка проєктних рішень	28.09.2024 р.	30.09.2024 р.	Виконано
5.	Моделювання та конструювання ПЗ	01.10.2024 р.	06.10.2024 р.	Виконано
6.	Кодування, тестування розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	11.10.2024 р.	07.11.2024 р.	Виконано
8.	Оформлення КМР та презентації	10.11.2024 р.	20.11.2024 р.	Виконано
9.	Попередній захист	28.11.2024 р.	28.11.2024 р.	Виконано
10.	Рецензування	05.12.2024 р.	09.12.2024 р.	Виконано
11.	Завершення оформлення КМР та презентації	09.12.2024 р.	14.12.2024 р.	Виконано
12.	Відгук керівника			
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Максим ПАСІЧЕНКО**

«__» _____ 20__ р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

“Програмне забезпечення формування розкладу занять з урахуванням матеріально-технічних вимог та розташування аудиторій”

Здобувач 608 гр.: Пасіченко Максим

Керівник: канд. техн. наук, доцент Давиденко Євген

Розклад в сучасному навчальному процесі – це не просто набір даних про подію, час та місце проведення занять, а ключовий інструмент, який формує структуру освітньої діяльності, особливо в сучасному світі, де цифровізація та швидкість змін стають нормою. Важливість розкладу сьогодні важко переоцінити, адже від правильної організації часу залежить не лише ефективність навчання, а й здатність адаптуватися до нових викликів: динаміки сучасного ринку праці, глобалізації та постійного оновлення знань.

Актуальність роботи викликана потребою структурування та оптимізації освітньої діяльності в контексті навчального розкладу.

Об’єкт роботи – процес формування розкладу занять навчального закладу.

Предмет роботи – інформаційна система для керування розкладом навчального закладу.

Мета роботи – надання платформи для створення, зберігання, обробки та відображення інформації навчального розкладу.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі визначається актуальність теми, мета та невеликий огляд поставленої задачі, предмет дослідження та об’єкт дослідження.

У першому розділі проведено аналіз предметної області, який включає огляд існуючих систем керування розкладом і навчанням. Здійснено порівняння їх функціональних можливостей, виявлено недоліки та обґрунтовано необхідність створення нової системи. Розглянуто специфікації вимог до програмного забезпечення, які враховують функціональність, продуктивність, зручність використання та безпеку.

Другий розділ присвячено моделюванню предмету та об’єкту дослідження. Описано методи складання розкладу, зокрема алгоритми цілочисельного програмування, імітації відпалу та генетичні алгоритми. Деталізовано основні етапи генерування розкладу, включаючи врахування обмежень і оптимізацію. Представлено діаграми прецедентів і класів, що відображають взаємозв’язки між компонентами системи.

У третьому розділі розглянуто архітектуру, моделювання та проєктування програмного забезпечення. Вибрано технології для розробки frontend та backend частин системи, описано процес створення бази даних та схему її розгортання. Розроблено структуру інформаційної системи, що забезпечує зручність її масштабування та інтеграцію з іншими платформами.

У четвертому розділі описано процес розробки та впровадження програмного забезпечення. Реалізовано основні модулі системи, включаючи функціонал автоматичного створення розкладу, а також механізми валідації та виправлення.

У висновках підсумовано основні результати дослідження, зазначено практичну цінність розробленого програмного забезпечення та можливості його впровадження у закладах освіти. Також окреслено перспективи подальшого розвитку системи, зокрема розширення її функціональності для інтеграції з іншими інформаційними платформами.

Кваліфікаційна робота викладена на 68 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 18 найменувань та 1 додатків. Праця містить 7 таблиць та 21 рисунків.

Ключові слова: автоматизація, інформаційна система, розклад занять, алгоритми оптимізації, керування навчальним процесом.

ABSTRACT

to the qualifying master's thesis

Software for generating class schedules taking into account material and technical requirements and classroom location

Student of 608 group: Pasichenko Maksym

Supervisor: Ph.D. tech. sciences, Associate Professor Davydenko Yevhen

The schedule in the modern educational process is not just a set of data about the event, time and place of classes, but a key tool that forms the structure of educational activities, especially in the modern world, where digitalization and the speed of change are becoming the norm. The importance of the schedule today is difficult to overestimate, because not only the effectiveness of training depends on the correct organization of time, but also the ability to adapt to new challenges: the dynamics of the modern labor market, globalization and constant updating of knowledge.

The relevance of the work is caused by the need to structure and optimize educational activities in the context of the educational schedule.

The object of the work is the process of forming the schedule of classes of an educational institution.

The subject of the work is an information system for managing the schedule of an educational institution.

The purpose of the work is to provide a platform for creating, storing, processing and displaying information on the educational schedule.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

The introduction defines the relevance of the topic, the goal and a brief overview of the task, the subject of the study and the object of the study.

The first section analyzes the subject area, which includes a review of existing scheduling and learning management systems. Their functional capabilities are compared, shortcomings are identified and the need for creating a new system is justified. The specifications of software requirements are considered, which take into account functionality, productivity, usability and security.

The second section is devoted to modeling the subject and object of the study. Methods for compiling a schedule are described, in particular integer programming algorithms, simulated annealing and genetic algorithms. The main stages of schedule generation are detailed, including taking into account constraints and optimization. Prerequisite and class diagrams are presented, reflecting the relationships between system components.

The third section considers the architecture, modeling and design of software. Technologies for developing the frontend and backend parts of the system are selected, the process of creating a database and its deployment scheme are described. The structure of the information system has been developed, ensuring its scalability and integration with other platforms.

The fourth section describes the process of software development and implementation. The main modules of the system have been implemented, including the functionality of automatic schedule creation, as well as validation and correction mechanisms.

The conclusions summarize the main results of the study, indicate the practical value of the developed software and the possibilities of its implementation in educational institutions. Prospects for further development of the system are also outlined, in particular, expanding its functionality for integration with other information platforms.

The qualification work is presented on 68 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of references with 18 titles and 1 appendices. The work contains 7 tables and 21 figures.

Keywords: automation, information system, class schedule, optimization algorithms, educational process management.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

LMS — Learning Management Systems

SRS — Scheduling Management Systems

SQL — Structured Query Language

NoSQL — Not Only SQL

СКБД — Система керування базами даних

ERD — Entity-Relationship Diagram

REST — Representational State Transfer

CRUD — Create, Read, Update, Delete

API — Application Programming Interface

JVM — Java Virtual Machine

JSON — JavaScript Object Notation

XML — eXtensible Markup Language

CLI — Command Line Interface

ВСТУП

Розклад в сучасному навчальному процесі – це не просто набір даних про подію, час та місце проведення занять, а ключовий інструмент, який формує структуру освітньої діяльності. Важливість розкладу важко переоцінити, адже від правильної організації часу залежить не лише ефективність навчання, а ефективність використання матеріально-технічної бази навчального закладу.

У сучасному освітньому просторі розклад має враховувати низку специфічних факторів, таких як стрімкий розвиток онлайн-навчання та гібридних форматів, які змінили підхід до планування часу. Правильно побудований розклад допомагає уникнути перенасичення студентів інформацією, зберігаючи баланс між живою комунікацією у навчальному закладі, віртуальними лекціями та часом для відпочинку.

Крім того, тенденція до індивідуалізації освіти підштовхує до створення гнучких розкладів, які враховують різні стилі навчання та особисті потреби студентів. У цьому контексті розклад перестає бути жорсткою структурою й перетворюється на інструмент персоналізації, що забезпечує доступ до навчальних матеріалів у зручний для кожного час.

Таким чином, розклад у сучасному навчальному процесі – це більше, ніж організаційний інструмент. Це дзеркало освітніх і суспільних трансформацій, які відображають зміну пріоритетів від простої передачі знань до розвитку гармонійної, стійкої та творчої особистості.

Формування розкладу є багатокритеріальним завданням з невизначеною множиною факторів, оскільки практично неможливо визначити всі критерії які враховують інтереси учасників академічного процесу. Добре структурований розклад може підвищити зацікавленість студентів відвідувати заняття, виконувати завдання та брати участь в інших навчальних заходах. З іншого боку, неефективний розклад може призвести до труднощів для здобувачів освіти та викладачів, таких як переповнені аудиторії та суперечливі розклади.

Враховуючи важливість створення розкладу, важливо знайти шляхи оптимізації процесу. Традиційні методи створення розкладу вручну схильні до помилок і неефективності, тому існує потреба досліджувати альтернативні рішення, які можуть покращити процес формування розкладу.

Створення системи управління розкладом університету є непростим завданням, але яке може значно підвищити ефективність і результативність академічного процесу. Така система може автоматизувати багато ручних процесів, пов’язаних із плануванням розкладу, завдань, іспитів та інших подій, звільняючи час для викладачів та адміністраторів, які можуть зосередитися на інших важливих завданнях. Система управління розкладом університету забезпечує централізовану платформу для керування розкладом та може допомогти кожному бути проінформованим про стан розкладу.

Завдяки автоматизації процесу та підвищенню його ефективності університети зможуть заощадити час і ресурси, а також нададуть студентам і викладачам добре структурований графік, який відповідає їхнім потребам. Платформа, розроблена в рамках цього дослідження, також надасть цінні дані та ідеї, які можна використовувати для подальшого вдосконалення процесу формування розкладу.

Підсумовуючи, формування університетського розкладу є критично важливим аспектом забезпечення успішного академічного середовища. Оптимізація процесу шляхом автоматизації та підвищення його ефективності суттєво вплине на ефективність та якість процесу формування розкладу.

Апробація результатів КМР відбулась під час Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія» (7-10 лютого 2023 р., ЧНУ ім. Петра Могили). (Додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Системи керування розкладом університетів є найбільш критичним компонентом діяльності та управління навчальним закладом. Ці системи розроблені для підтримки планування та керування будь-якими ресурсами академічної діяльності, включаючи лекції, лабораторні заняття та іспити. Система відіграє ключову роль у підвищенні продуктивності, ефективності та взаємодії університету з учасниками навчального процесу шляхом автоматизації багатьох ручних процесів, пов’язаних із плануванням і наданням доступу в режимі реального часу до розкладів та іншої інформації.

Область систем керування розкладом пов’язана з використанням технологій для підтримки різних процесів і операцій вищих навчальних закладів, включаючи планування та керування розкладом та матеріально-технічною базою.

Система керування зазвичай включає в себе керування розкладом навчальних занять, таким як розклад лекцій, практичних занять та розклад іспитів. Система має бути орієнтована на розроблена таким чином, щоб відповідати унікальним вимогам розкладу навчання в університеті, включаючи потребу в управлінні великими обсягами даних і адаптації до різних розкладів для різних відділів, програм і груп студентів.

Предметна область систем керування розкладом також охоплює розробку та впровадження алгоритмів і процедур для планування, наприклад щодо використання матеріально-технічної бази навчального закладу, а також керування складовими освітнього процесу. Це вимагає глибокого розуміння діяльності та потреб університету, а також розуміння передового досвіду планування та управління ресурсами.

Зрештою, мета університетських систем управління розкладом полягає в тому, щоб надати студентам, викладачам і адміністраторам ефективний інструмент

для управління розкладом, покращення зворотного зв’язку та максимального використання ресурсів. Автоматизуючи процес планування та надаючи доступ до розкладів та інформації в режимі реального часу, система може допомогти навчальним закладам працювати ефективніше, зменшити адміністративне навантаження та забезпечити кращий досвід для учасників освітнього процесу.

За класичним визначенням, розклад навчальних занять – один з основних організаційних документів, що регламентує освітній процес за денною, заочною формами навчання всіх освітніх ступенів, регулює навчальні заняття (лекції, лабораторні, практичні, семінарські заняття; консультації) по дням тижня, курсам, групам та місцям проведення.

Розклад навчальних занять забезпечує логічну послідовність у засвоєнні знань та практичних навичок здобувачів вищої освіти, ефективне використання науково-технічної, методичної бази, раціональне використання аудиторного фонду[1].

1.2 Аналіз існуючих засобів

Для повного розуміння поточного стану інформаційних систем керування розкладом необхідно ретельно проаналізувати наявні системи. Системи керування навчанням (LMS) та системи керування розкладом (SRS) стали невід'ємними інструментами в організації освітнього процесу. Пов'язаність цих систем дозволяє створити єдине інтегроване середовище, яке підвищує ефективність навчання, спрощує адміністративні процеси та покращує досвід користувачів.

Для визначення популярності різних систем керування навчанням вирішено було використати сервіс Google Trends.

Google Trends — сервіс від компанії Google, який дозволяє в режимі реального часу відслідковувати теми, за якими стежать люди. Також сервіс накопичує статистику по пошуковим запитам та пов’язаною з ними інформацію у сервісах Google Search, Google News або YouTube[2].

Для проведення аналізу було вирішено обрати декілька LMS: Moodle, Google Classroom, “Нові знання”, Canvas LMS та Blackboard Learn. Результати[3] проведеного аналізу наведено нижче.

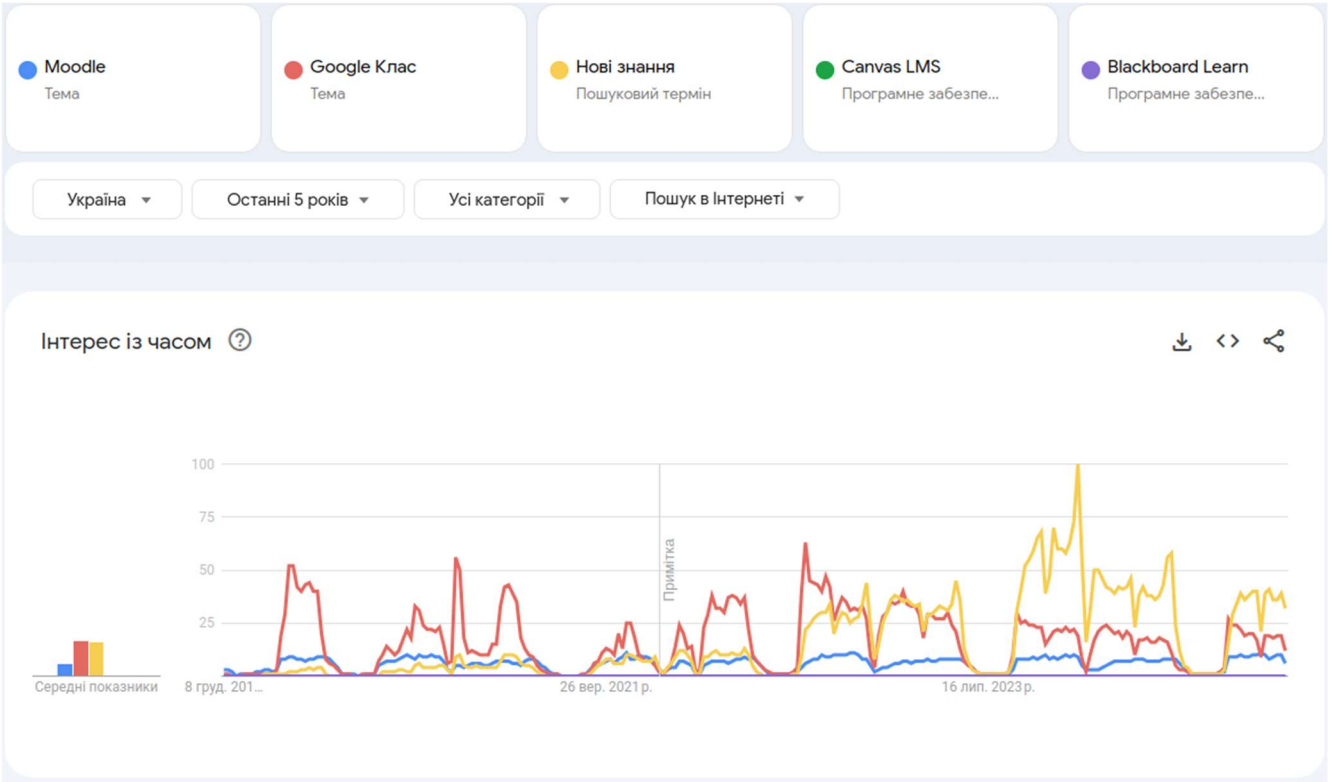

Рисунок 1.1 – Динаміка популярності LMS

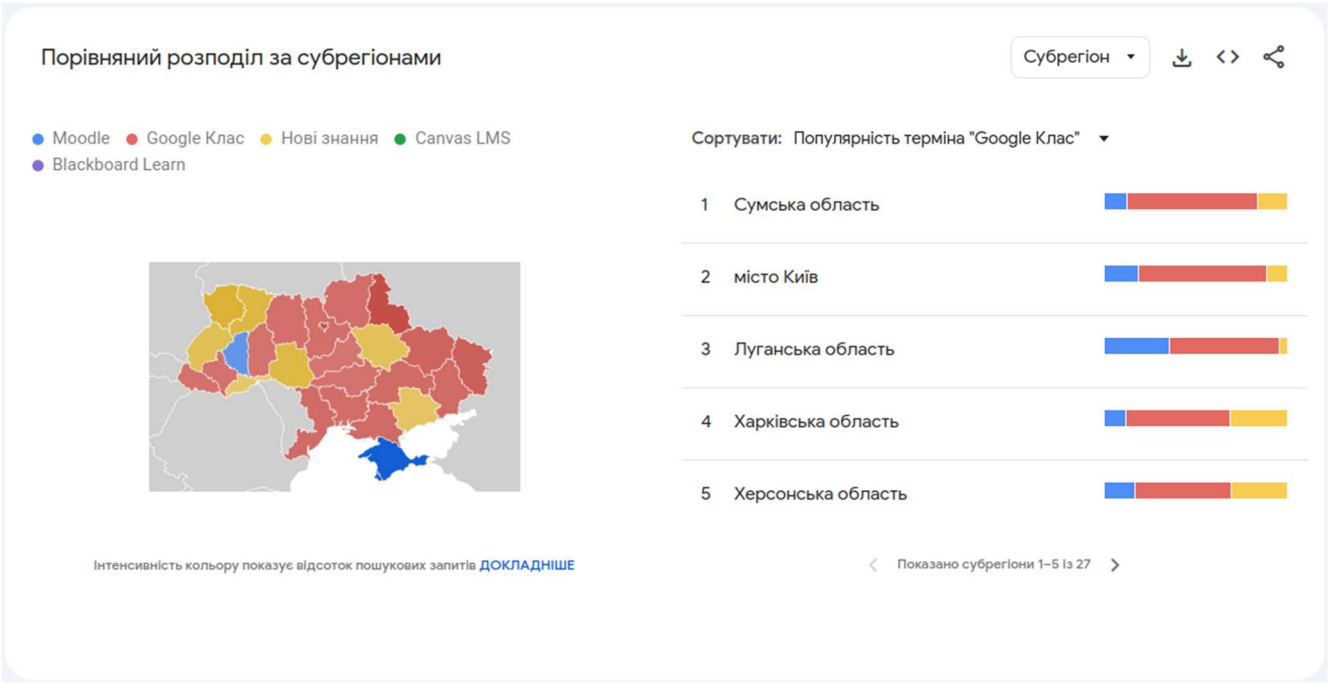

Рисунок 1.2 – Популярність LMS по регіонам

Цей аналіз надає корисну метрику для оцінки ринкового попиту та інтересу користувачів до різних систем керування навчанням.

Згідно цього дослідження можна визначити, що в період з кінця 2019 по кінець 2024 року найбільш популярними системами керування навчання є Moodle, Google Classroom та платформа “Нові знання”. З огляду та те, що інші системи керування навчанням є менш популярними або взагалі не використовуються та в подальших аналізах та дослідженнях розглядатися не будуть.

Щоб розробити систему керування навчанням важливо порівняти функціонал існуючих систем керування навчанням та їх особливості в розрізі питання керування розкладом. Це порівняння надасть розуміння переваг та недоліків систем, а також висвітлить найкращі практики та функції, які можна повторно використовувати або вдосконалити під час розробки нової або вдосконалення існуючої системи.

Moodle

Moodle — це система керування навчанням із відкритим вихідним кодом, розроблена для забезпечення викладачів, адміністраторів і учнів єдиною надійною, безпечною та інтегрованою системою для створення персоналізованого навчального середовища[4].

Особливостями Moodle LMS є[5]:

- курси: розділ, де викладачі додають навчальні матеріали та завдання для здобувачів освіти;
- dashboard: розділ, який містить таймлайн з строками здачі робіт та календар подій;
- оцінки: гнучке оцінювання робіт з можливістю відстеження прогресу виконання робіт;
- плагіни: дозволяє додавати додаткові особливості та функції до Moodle.

Google Classroom

Google Classroom — це сервіс від компанії Google, призначений для навчальних закладів, метою якого є допомогти викладачам створювати та поширювати завдання між учасниками учбового процесу[6].

Особливостями Google Classroom є [7]:

– курси: простір, в якому розміщуються завдання з певного предмету;

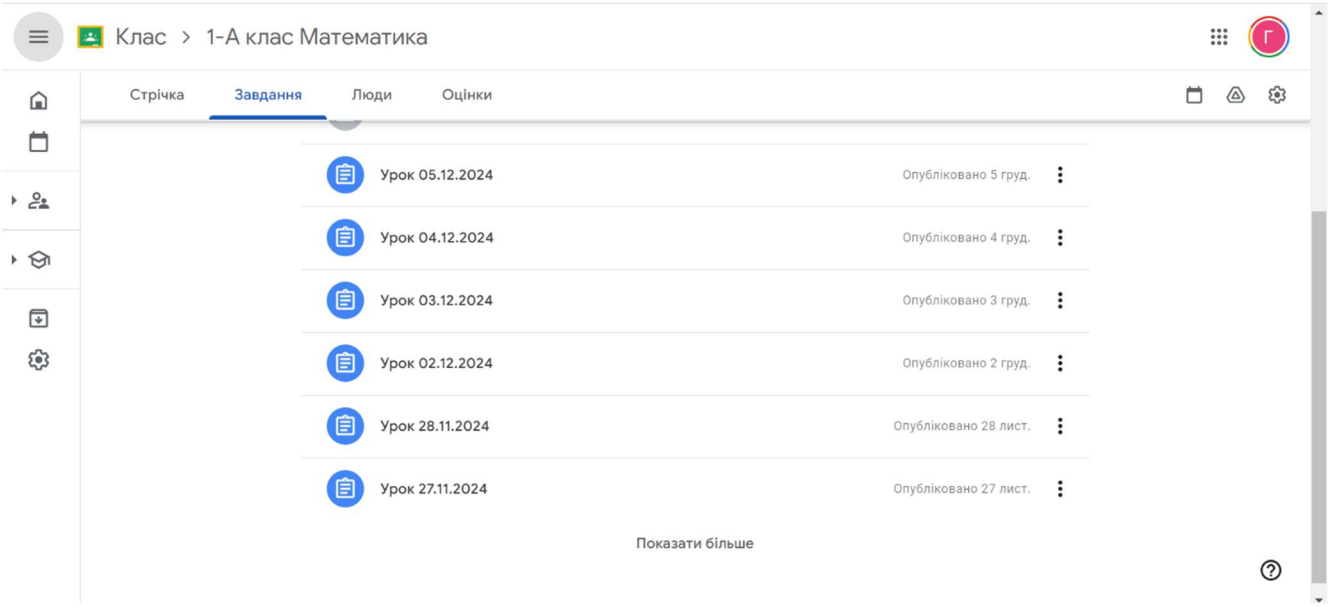

Рисунок 1.3 – Завдання простору ”Курс” платформи Google Classroom

– оцінки: оцінювання робіт та прямий зворотній зв’язок між викладачем та здобувачем освіти;

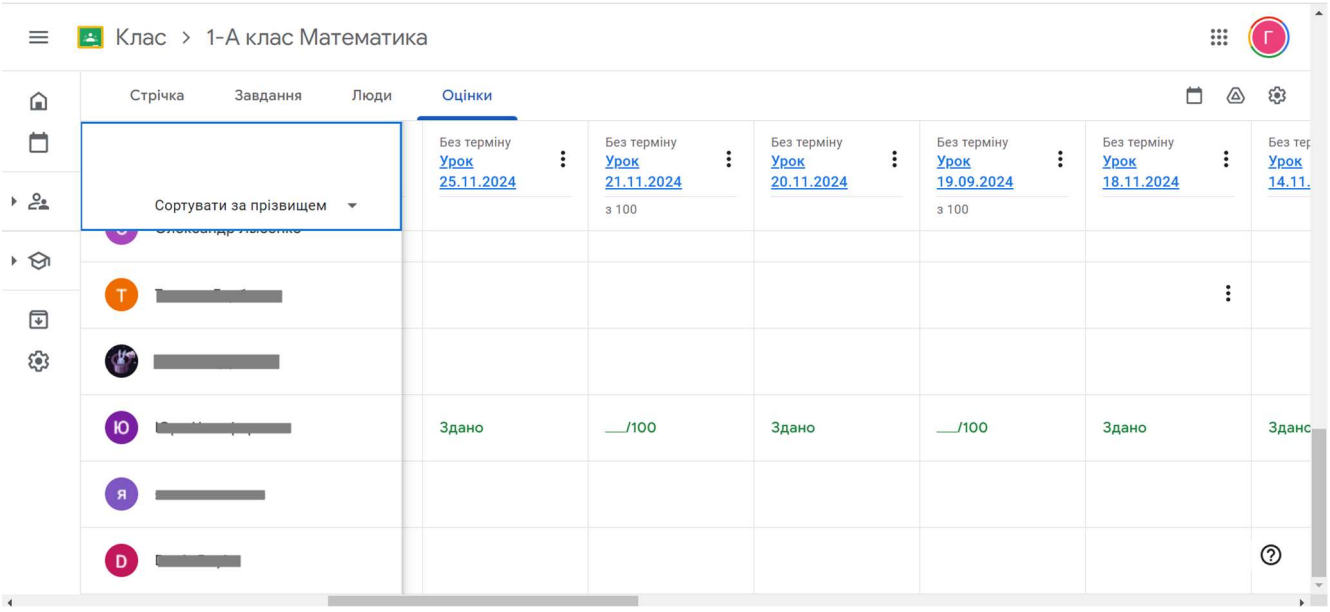

Рисунок 1.4 – оцінювання робіт здобувачів освіти на платформі Google Classroom

– сповіщення: інформування про зміни в курсах та стан оцінювання робіт;

– Google Workspace for Education: глибока інтеграція між іншими сервісами Google, такими як Google Forms, Google Calendar та іншими елементами з Google Drive.

Нові знання

Платформа “Нові знання” — це український онлайн-сервіс, що надає електронні щоденники та журнали з можливостями дистанційного навчання. Вона є складовою проєкту «КУРС: Освіта» і спрямована на спрощення та покращення освітнього процесу для учнів, батьків та вчителів[8].

Особливостями “Нові знання” є:

– журнали: можливість зручного створення уроків, виставлення оцінок та аналізу успішності учнів, класів, школи;

– щоденники: надають учасникам навчального процесу постійний доступ до всієї історії отриманих оцінок та домашніх завдань, а також надають можливість аналізу успішності за тривалим періодом;

– дистанційне навчання: можливість задавати, виконувати та перевіряти роботи онлайн;

– розклад: ручне створення навчального розкладу та його перегляд учасниками навчального процесу.

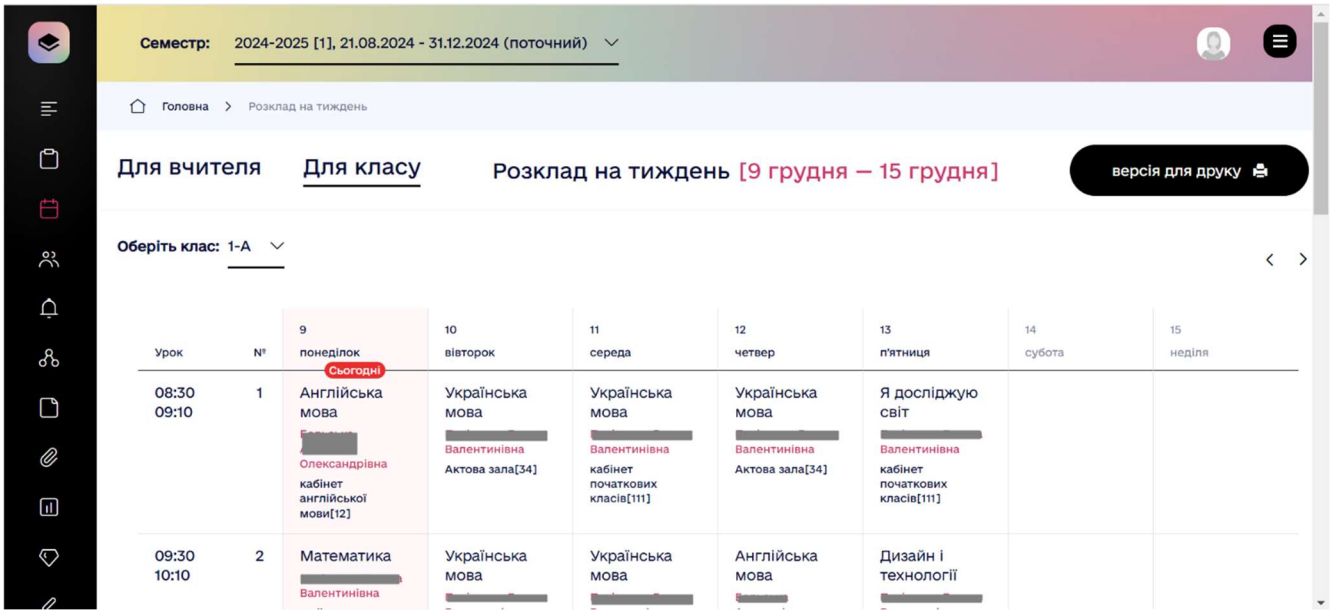

Рисунок 1.5 – розклад занять на платформі ”Нові знання”

Проаналізувавши вищенаведені особливості можна визначити, що кожна з систем керування навчанням містить інформацію про навчальний процес, яка необхідна для генерування розкладу. Але можна зазначити, що тільки система “Нові знання” має інтегроване рішення для відображення розкладу занять, проте у сервісі Google Classroom наявне рішення Google Calendar, яке можна використати для гнучкого відображення розкладу. Також система Moodle має у своєму складі календар та підтримку плагінів, яку можна використати для інтеграції з системою керування розкладом.

Підсумовуючи аналіз систем керування навчанням, можна зробити висновок, що сфера управління розкладом залишається малорозвиненою. Більшість систем пропонують лише базовий функціонал для ручного створення розкладу, але автоматична генерація, яка могла б значно спростити процес, поки що відсутня. Внаслідок цього існує потреба у створенні системи керування розкладом, в якій буде закладена можливість автоматичної генерації розкладу.

1.3 Специфікації вимог

Специфікацію вимог до системи управління розкладом навчального закладу можна розділити на кілька ключових областей, включаючи функціональні вимоги,

вимоги до продуктивності, вимоги до зручності використання та вимоги до безпеки. У цьому розділі наведено огляд кожної з цих областей та основні вимоги, які необхідно враховувати при розробці системи управління розкладом для університету.

Функціональні вимоги:

- система повинна мати можливість автоматично генерувати розклад на основі набору обмежень, таких як доступні аудиторії, наявність викладачів і розклад студентів;
- система повинна забезпечувати інтерфейс для додавання, зміни та видалення курсів і пов’язаної інформації, такої як описи курсів, попередні умови та викладачі;
- система повинна забезпечувати інтерфейс для додавання, зміни та видалення занять і пов’язаної інформації, такої як час занять, місця та можливості;
- система повинна забезпечувати інтерфейс для планування іспитів, включаючи випускні і проміжні іспити;
- система повинна забезпечувати інтерфейс для керування ресурсами, такими як класні кімнати та лабораторії, включаючи їх доступність, місткість і використання;
- система має надавати інтерфейс для керування розкладом студентів, включаючи додавання та видалення курсів, оновлення оцінок і відстеження відвідуваності;
- система повинна забезпечувати інтерфейс для керування розкладами викладачів, включаючи їхню доступність, курсові завдання та години роботи.

Вимоги до продуктивності:

- система повинна бути здатна вмістити зростаючу кількість користувачів, курсів і ресурсів;

- система повинна мати високий рівень надійності з мінімальними простоями та мінімальною втратою даних;
- система повинна мати швидкий час відгуку, навіть за умов високого навантаження.

Вимоги до зручності використання:

- система повинна мати зручний інтерфейс, простий у використанні як для адміністраторів, так і для користувачів;
- система повинна бути доступна з мобільних пристроїв, що дозволяє користувачам переглядати свої розклади та керувати ними на ходу.

Вимоги безпеки:

- система повинна забезпечувати автентифікацію користувача;
- система повинна забезпечувати контроль доступу з різними рівнями доступу на основі ролей і обов’язків користувачів.

Висновки до розділу 1

Було проведено дослідження предметної області систем керування навчанням.

Під час дослідження було проведено аналіз популярності систем керування навчанням на території України в період з кінця 2019 по кінець 2024 року. В результаті аналізу було виявлено дві найбільш популярні системи: Moodle та Google Classroom. Інші системи, такі як Blackboard та Canvas LMS є взагалі не популярними на території України.

Спираючись на аналіз популярності було досліджено переваги та недоліки двох найбільш популярних систем керування навчанням.

В результаті дослідження було виявлено, що жодна з систем керування навчанням не має функціоналу з автоматичного створення розкладу.

Підсумовуючи, специфікація вимог до системи управління розкладом навчального закладу повинна враховувати вимоги до функціональності,

продуктивності, зручності використання та безпеки системи. Система повинна мати можливість ефективно створювати розклади, керувати курсами, заняттями, ресурсами, студентами та викладачами, бути масштабованою та надійною, мати швидкий час відгуку, мати зручний інтерфейс, бути доступною з мобільних пристроїв, мати автентифікацію користувачів і контроль доступу. Ці вимоги мають бути чітко визначені та узгоджені між навчальним закладом і командою розробників, щоб забезпечити відповідність системи управління розкладом потребам і очікуванням навчального закладу.

2 МОДЕЛЮВАННЯ ПРЕДМЕТУ ТА ОБ'ЄКТУ ДОСЛІДЖЕННЯ

Основною метою кваліфікаційної роботи є створення інформаційної системи, ключовою особливістю якої є автоматична побудова зручного розкладу, який буде відповідати вимогам освітнього процесу, а також побажанням викладачів та здобувачів освіти.

2.1 Дослідження методів складання розкладу

Розв'язання задач складання розкладу належить до комбінаторних задач з високою розмірністю, які не можуть бути вирішені перебором через їхню складність. У наукових дослідженнях відзначаються ключові підходи, такі як комбінаторні методи та імітаційне моделювання, методи цілочисельного програмування, евристичні та еволюційні алгоритми.

Метод цілочисельного програмування

Метод цілочисельного програмування має на меті забезпечити оптимальне формування розкладу занять, враховуючи всі обмеження і цілі, що стоять перед навчальним закладом. Це дозволяє автоматизувати та пришвидшити процес складання розкладу, забезпечуючи його відповідність реальним потребам закладу. Основна ідея методу полягає у знаходженні найкращого (оптимального) рішення, яке відповідає певним умовам, з використанням цілочисельних змінних, що відображають реальні об'єкти, наприклад пари, аудиторії чи викладачі.

Реалізація алгоритму полягає у наступних кроках:

– виділення змінних – представлення реальних об'єктів у вигляді змінних;

– складання обмежень для змінних – формулювання обмежень у вигляді рівнянь або нерівностей, наприклад група не може бути присутньою одночасно на декількох парах;

– складання цільової функції – функція, в яку закладають цілі мінімізації «вікон» для забезпечення щільності розкладу, гарантування унікальності використання ресурсів та максимізація зручності;

– знаходження максимуму чи мінімуму цільової функції.

Основними недоліками методу цілочисельного програмування є:

– час розв’язання задачі зростає експоненційно зі збільшенням кількості змінних або обмежень;

– через велику складність моделі важко оцінити вплив різних чинників, що ускладнює її розв’язання.

Метод імітації відпалу

Методу імітації відпалу імітує фізичний процес охолодження металу, під час якого атоми стають менш рухливими, що дозволяє досягти стабільної структури з мінімальною енергією. В рамках задачі генерації навчального розкладу цільовою функцією є функція яка враховує штрафи за кожний конфліктний вибір, а низькоенергетичним станом – коректний розклад.

Ідею алгоритму викладено в наступних кроках:

– ініціалізація початкового розкладу випадковим чином з високою температурою T_0, для уникнення локальних мінімумів завдяки «гіршим» рішенням;

– зміна рішення – проведення мутації розкладу, а саме внесення випадкових змін в поточний розклад, змінюючи аудиторію чи часовий слот заняття або перестановка занять між собою;

– перевірка цільової функції розкладу, якщо новий розклад покращує цільову функцію – зміни нового розкладу приймаються. У випадку, якщо результат цільової функції погіршується – зміни можуть бути прийняті з певною ймовірністю, яка залежить від температури:

$$P = e^{-\Delta E/T} \quad (2.1)$$

де ΔE – різниця в значенні цільової функції між новим та старим розкладом,

T – поточна температура

– зниження температури за формулою 2.2:

$$T = T_0 \cdot \alpha^k \quad (2.2)$$

де α – коефіцієнт охолодження,

k – номер ітерації

– поступове зниження температури, доки не буде досягнуто критерій зупинки(задана кількості ітерацій або досягнення низької температури);

Недоліками алгоритму є:

– час виконання алгоритму залежить від кількості ітерацій;

– точність розв’язку залежить від коефіцієнту охолодження;

2.2 Опис алгоритму складання розкладу

Методи складання розкладу, розглянуті в попередньому розділі, базуються на ітераційній техніці покращення результатів: протягом ітерації шукається розв’язок і якщо цільова функція має кращий результат за попередню ітерацію - поточний розв’язок стає основним, далі алгоритми методів працюють, доки буде досягнуто критерій зупинки. Зазначені методи орієнтовані на пошук локальних оптимумів, результат розташування яких залежить від початкової точки роботи алгоритму.

Генетичні алгоритми є дійсно ефективним рішенням для подолання обмежень локальних оптимумів, характерних для традиційних методів, які залежать від вибору початкової точки, тому генетичний алгоритм є ефективним підходом до вирішення складних задач оптимізації, зокрема автоматизованого складання розкладів. Уперше цей метод був запропонований Дж. Холландом у 1975 році[9]. Принцип роботи генетичних алгоритмів базується на ідеях природного відбору та еволюції, що дозволяє адаптувати біологічні механізми до пошуку оптимальних рішень у складних задачах.

Основні етапи роботи генетичних алгоритмів включають:

– ініціалізація: генерація початкової популяції можливих рішень;

- оцінка: використання фітнес-функції для визначення ефективності кожного рішення;
- еволюційні операції: селекція, мутація та кросинговер для створення нових рішень;
- завершення: зупинка алгоритму за досягненням визначеного критерію;

У процесі складання розкладів генетичні алгоритми моделюють хромосому як набір генів. Кожен ген кодує елемент розкладу, наприклад, комбінацію "група-аудиторія-часовий_слот". Статичні дані (навчальний курс, тип заняття) залишаються незмінними, тоді як динамічні дані (викладачі, студенти) можуть змінюватися під час роботи алгоритму [10].

Алгоритм складання розкладу складається з таких етапів:

1. Збір вхідних даних, що включає в себе наступні дії:
   - імпорт даних з зовнішнього сховища;
   - ручне дозаповнення відсутніх даних;
2. Визначення пріоритетів: визначення пріоритетів складових алгоритму побудови розкладу, таких як пріоритет предметів або потреба в особливому обладнанні в аудиторії;
3. Врахування обмежень: при складанні розкладу необхідно враховувати такі обмеження:
   - жорсткі:
     - відповідність розміру аудиторії чисельності групи;
     - наявність необхідного обладнання;
     - доступність викладачів у визначені дні.
   - нежорсткі:
     - уникнення "вікон" у розкладі студентів;
     - рівномірне навантаження груп.

4. Генерація розкладу: створити початковий розклад, в якому заняття розподілені по часовим слотам, а групи студентів пов’язані з цими заняттями та в якому враховано жорсткі обмеження;

Алгоритм генерує розклад за принципом покрокового додаванням елементів розкладу, вибираючи складові розкладу з найбільшими обмеженнями та співставляючи їх з навчальними групами, отриманий результат перетворюється в елемент розкладу. Алгоритм повторюється поки є непроставлені часові слоти дисциплін для навчальних груп.

Для визначення значення цінності елементу розкладу в певний час в певній аудиторії можна застосувати наступну формулу:

$$Вел = (Свик * Пвик + Вспец + Ввік) * Кпар \quad (2.3)$$

Вел - вага елементу розкладу

Свик - можливість викладання викладачем в вказаний час

Пвик - побажання викладання викладача для даної аудиторії в даний час

Вспец - вага спеціалізації вибраної аудиторії

Ввік - вага наявності вікна у групи

Кпар - кількість пар підряд в елементі розкладу

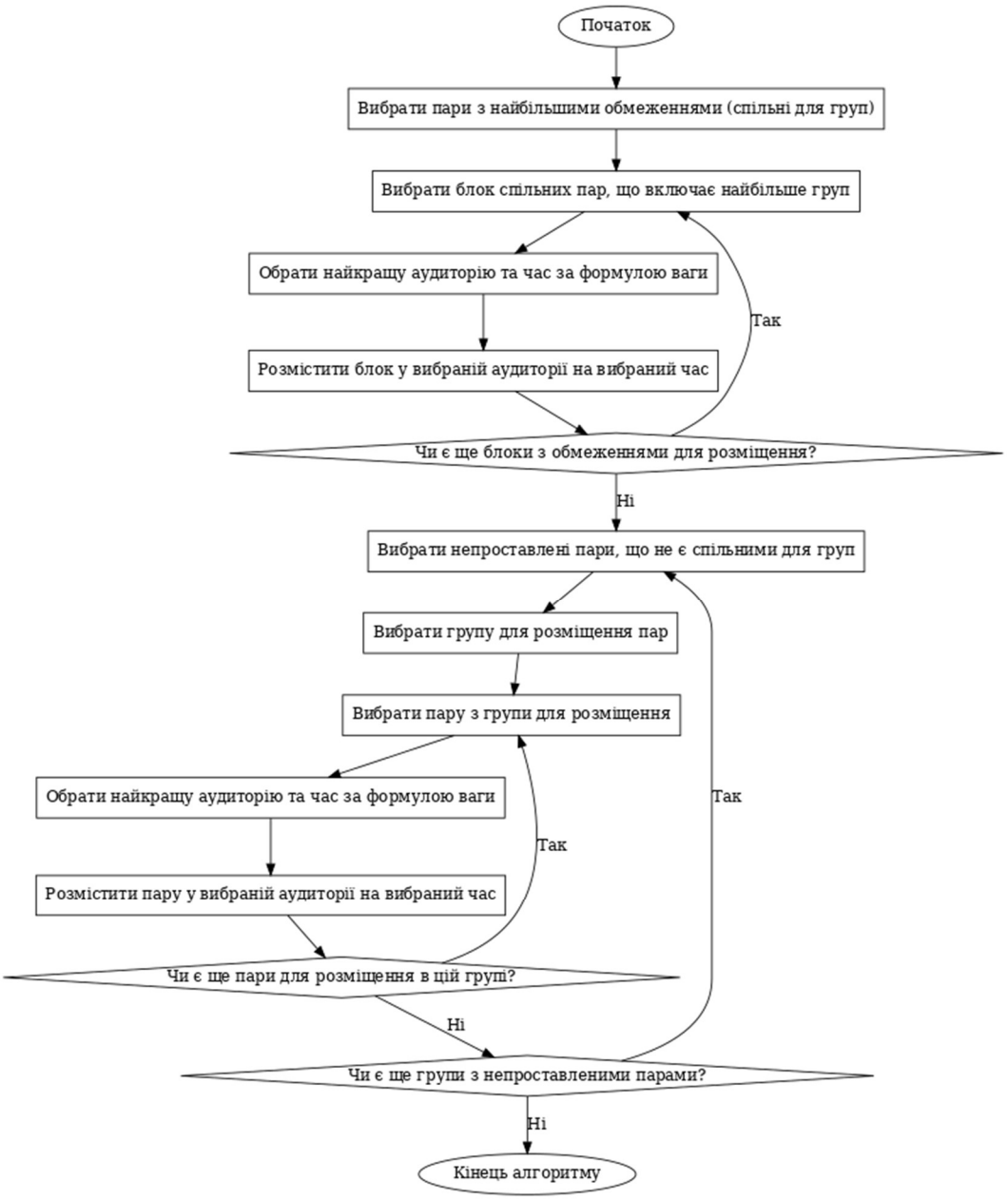

Рисунок 2.1 – Алгоритм побудови розкладу

5. Оптимізація розкладу: поліпшення розкладу, що включає модифікацію розкладу згідно нежорстких обмежень, а саме перестановки розкладу для виконання нежорстких обмежень з урахуванням вимог жорских обмежень

Алгоритм оптимізації модифікує розклад за принципом Парето – перестановкою вибрати найкраще місце за формулою (2.3) та, якщо пара не зменшує свою цінність і не зменшується цінність інших пар, переставити її на це місце, перемістивши відповідно інші пари.

$$Вел = Свик * Пвик + Вспец + Ввік \quad (2.4)$$

Вел - вага елементу розкладу

Свик - можливість викладання викладачем в вказаний час

Пвик - побажання викладання викладача для даної аудиторії в даний час

Вспец - вага спеціалізації вибраної аудиторії

Ввік - вага наявності вікна у студентів

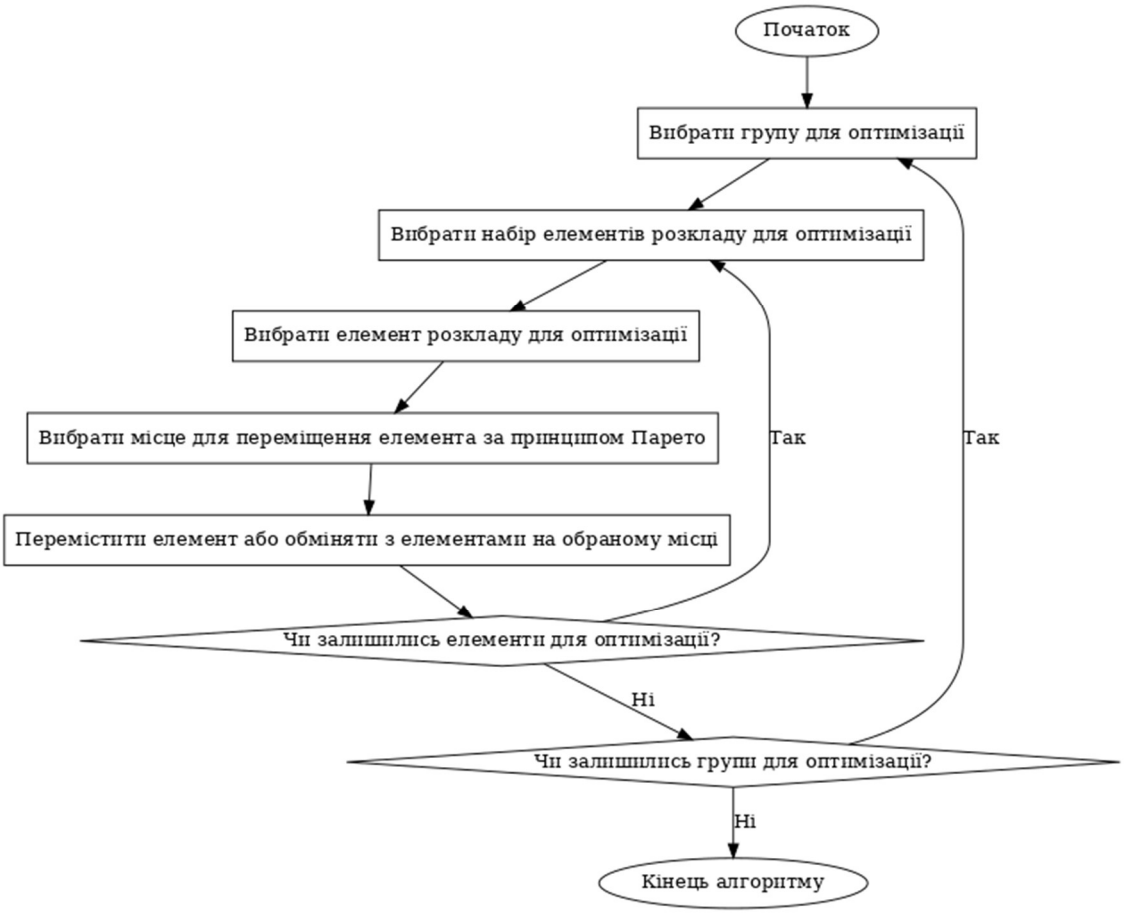

Рисунок 2.2 – Алгоритм оптимізації розкладу

6. Валідація та виправлення: після завершення внесення змін і оптимізації розкладу потрібно перевірити, чи відповідає він усім установленим обмеженням і вимогам, оцінити його якість та затвердити

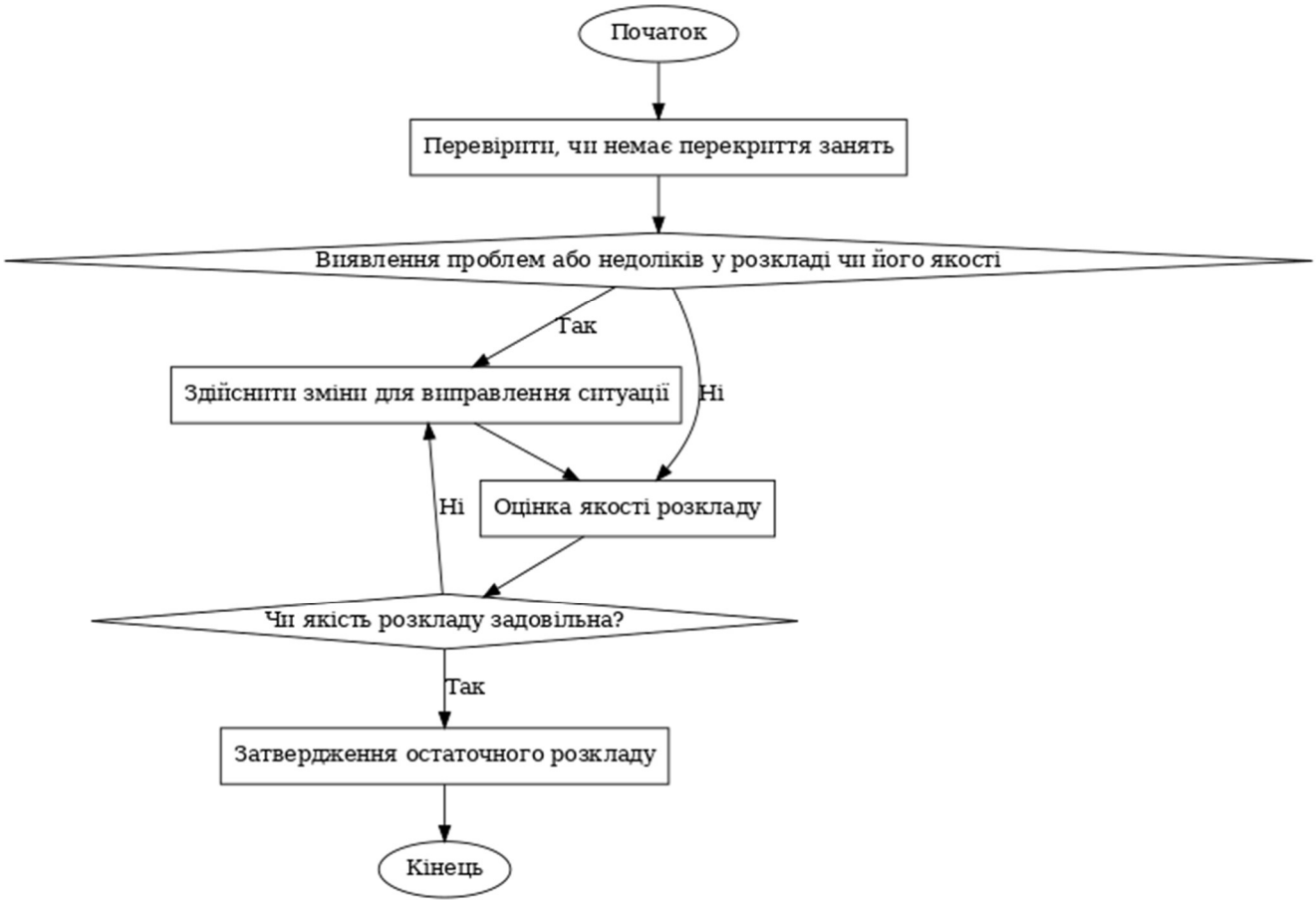

Рисунок 2.3 – Алгоритм перевірки розкладу встановленим вимогам

Використання генетичних алгоритмів дозволяє оптимізувати розподіл ресурсів і мінімізувати конфлікти. Водночас важливо враховувати, що випадкова ініціалізація популяції може створювати значні труднощі на подальших етапах. Тому перевага надається підходам, що забезпечують безконфліктність початкових рішень.

2.2 Діаграми прецедентів

Діаграма варіантів використання (або діаграма прецедентів, Use Case Diagram) - це графічне представлення взаємозв’язків між акторами та варіантами використання, і призначена для опису взаємодії між ними[11]. Її основні цілі включають:

1. Визначення загальних меж і контексту предметної області:

- дозволяє встановити межі системи та визначити, що саме буде входити в її функціональність, а що залишиться поза її межами;
- дає змогу зрозуміти взаємодію системи з зовнішніми акторами, такими як користувачі чи інші системи;

2. Формування загальних вимог до функціональної поведінки системи:
   - ілюструє основні функціональні можливості системи з точки зору користувача;
   - допомагає визначити, які завдання система повинна виконувати, і які результати очікуються від її використання;
3. Розробка вихідної концептуальної моделі системи:
   - є основою для створення більш детальних логічних та фізичних моделей;
   - дозволяє виділити ключові компоненти і функціональні модулі системи для подальшого проєктування;
4. Підготовка вихідної документації для взаємодії:
   - діаграма забезпечує зрозумілий інструмент для спілкування між розробниками, замовниками та кінцевими користувачами.
   - представляє функціональність системи у простій та зрозумілій формі.

Діаграма варіантів використання є основою для подальшого уточнення та деталізації вимог до системи, включаючи моделювання бізнес-процесів, проектування інтерфейсів користувача і визначення архітектури. Це робить її важливим кроком у процесі розробки.

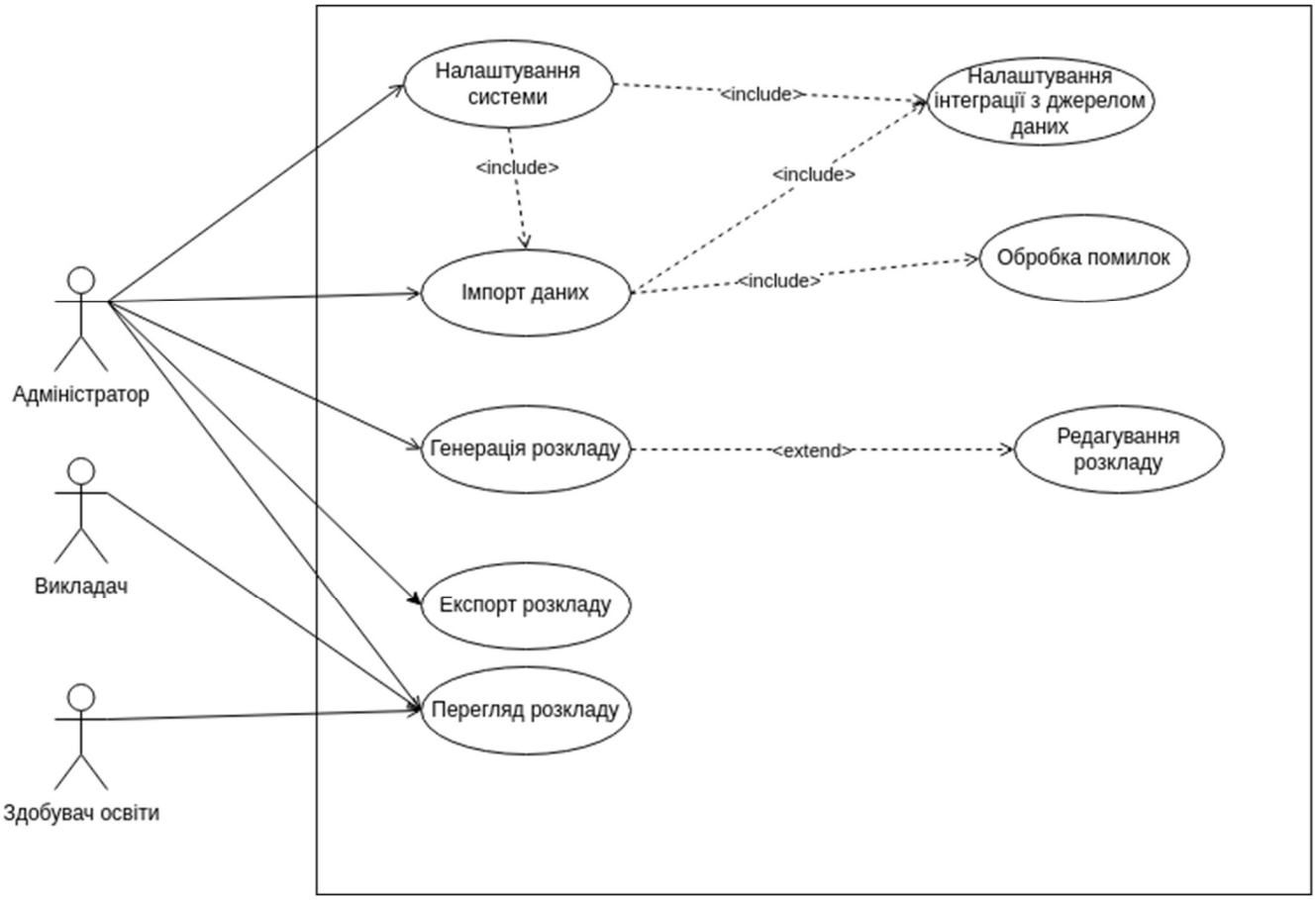

Рисунок 2.4 – Діаграма варіантів використання системи

Покладаючись на діаграму прецедентів(див. рис. #) можна описати наступних акторів системи:

– Адміністратор: представник навчального закладу, який налаштовує систему, тип джерела даних, інтегрує джерело даних в інформаційну систему;
– Викладач: особа, яка може вносити побажання щодо обмежень розкладу;
– Здобувач освіти: особа, яка отримує доступ до сформованого розкладу.

Таблиця 2.1 – Опис прецеденту “SettingsSystem”

Характеристика	Опис
Назва прецеденту	SettingsSystem
Короткий опис	Адміністратор налаштовує систему
Актори	Адміністратор

Передумови	Обліковий запис з розширеними правами доступу
Основний потік подій	– виконати автентифікацію та авторизацію; – налаштувати інтеграцію з джерелом даних; – виконати імпорт даних
Стан системи після закінчення прецеденту	Відображення завантажених даних.
Альтернативні потоки подій	AccessDenied DatasourceFail

Таблиця 2.2 – Опис прецеденту “SettingsSystem:AccessDenied”

Характеристика	Опис
Назва прецеденту	SettingsSystem:AccessDenied
Короткий опис	Інформування користувачу про недостатній рівень прав доступу до системи
Актори	Адміністратор
Передумови	Обліковому запису користувача не надано розширені права доступу
Основний потік подій	– виконати автентифікацію; – виконати авторизацію; – отримати повідомлення про відсутність прав доступу
Стан системи після закінчення прецеденту	Система інформує користувача про відсутність розширених привілеїв доступу
Альтернативні потоки подій	-

Таблиця 2.3 – Опис прецеденту “SettingsSystem:DatasourceFail”

Характеристика	Опис
Назва прецеденту	SettingsSystem: DatasourceFail

Короткий опис	Помилка взаємодії з інтеграційним модулем джерела даних
Актори	Адміністратор
Передумови	Автентифікований та авторизований користувач з адміністративними ролями доступу
Основний потік подій	– виконати вхід у систему; – отримати повідомлення про відсутність прав доступу
Стан системи після закінчення прецеденту	Користувачу надано повідомлення про помилку взаємодії з інтеграційним модулем джерела даних
Альтернативні потоки подій	-

Таблиця 2.4 – Опис прецеденту “GenerateSchedule”

Характеристика	Опис
Назва прецеденту	GenerateSchedule
Короткий опис	Генерація розкладу
Актори	Адміністратор
Передумови	Дані необхідні для генерації розкладу наявні в системі
Основний потік подій	– виконати вхід у систему; – ініціювати генерацію розкладу;
Стан системи після закінчення прецеденту	Адміністратору надано розклад
Альтернативні потоки подій	ScheduleNeedCorrection ScheduleSuccessfullyGenerated

Таблиця 2.5 – Опис прецеденту “GenerateSchedule:ScheduleNeedCorrection”

Характеристика	Опис

Назва прецеденту	GenerateSchedule:ScheduleNeedCorrection
Короткий опис	Уточнення розкладу
Актори	Адміністратор
Передумови	Згенеровано розклад
Основний потік подій	– виконати вхід у систему; – перейти в розділ згенерованого розкладу;
Стан системи після закінчення прецеденту	Модифікований адміністратором розклад
Альтернативні потоки подій	ScheduleSuccessfullyGenerated

Таблиця 2.6 – Опис прецеденту “GenerateSchedule:ScheduleSuccessfullyGenerated”

Характеристика	Опис
Назва прецеденту	GenerateSchedule:ScheduleSuccessfullyGenerated
Короткий опис	Уточнення розкладу
Актори	Адміністратор
Передумови	Згенеровано розклад
Основний потік подій	– виконати вхід у систему; – перейти в розділ згенерованого розкладу;
Стан системи після закінчення прецеденту	Модифікований адміністратором розклад
Альтернативні потоки подій	-

2.3 Діаграма класів

Діаграма класів є одним із ключових елементів мови моделювання UML (Unified Modeling Language), який використовується для візуального представлення структури системи. Вона описує класи, їхні властивості (атрибути),

методи (операції), а також взаємозв'язки між класами, що є основою об’єктно-орієнтованого підходу до розробки програмного забезпечення.

Діаграма класів має важливе значення в контексті моделювання програмних систем, виконуючи кілька ключових функцій. Вона слугує інструментом для моделювання структури системи, відображаючи організацію даних та взаємодію між класами. Завдяки цьому забезпечується наочне представлення зв’язків між об'єктами, що дозволяє краще зрозуміти архітектуру програмного забезпечення.

Крім того, діаграма класів сприяє аналізу вимог до системи, допомагаючи візуалізувати властивості об'єктів та їхні взаємозв'язки. Це дозволяє чітко визначити вимоги на початкових етапах розробки, зменшуючи ризики неправильного розуміння завдань. Її також використовують у процесі проєктування системи, оскільки вона є основою для формування архітектури та створення вихідного коду, що відображає закладену структуру.

Окрім технічного застосування, діаграма класів відіграє роль у спрощенні комунікації між учасниками проєкту, зокрема між розробниками, архітекторами, замовниками та користувачами. Її зрозуміле графічне подання полегшує обговорення технічних деталей та забезпечує узгодженість бачення функціоналу системи.

Серед переваг діаграми класів варто зазначити, що вона сприяє формуванню чіткого уявлення про структуру системи, підтримує принципи об'єктно-орієнтованого проєктування та дозволяє виявити можливі структурні проблеми на ранніх етапах розробки. Таким чином, діаграма класів є невід'ємним інструментом у створенні якісних, масштабованих та ефективних програмних рішень, забезпечуючи базу для подальшого деталізованого проєктування і реалізації системи.

Основні елементи діаграми класів:

1. Клас: Представляє набір об’єктів із подібними властивостями та поведінкою, складається з трьох частин:

- Назва класу.
- Атрибути (властивості класу).
- Методи (функції або операції, які клас може виконувати).

2. Зв’язки між класами:
   - Асоціація: Відображає статичні зв’язки між об'єктами класів. Наприклад, "студент навчається в університеті".
   - Агрегація: Відносини "частина-ціле", де частини можуть існувати незалежно від цілого. Наприклад, "курс містить модулі".
   - Композиція: Тісніший зв’язок, ніж агрегація, де частини не можуть існувати без цілого. Наприклад, "бібліотека складається з книжок".
   - Успадкування: Відображає ієрархічні відносини між класами, де підклас успадковує властивості і методи батьківського класу.
   - Залежність: Показує, що один клас використовує інший, але це не є постійним зв'язком.
3. Модифікатори доступу, вказують на рівень доступу до атрибутів і методів:
   - + (public) – доступний для всіх.
   - - (private) – доступний лише всередині класу.
   - # (protected) – доступний для класу та його підкласів.

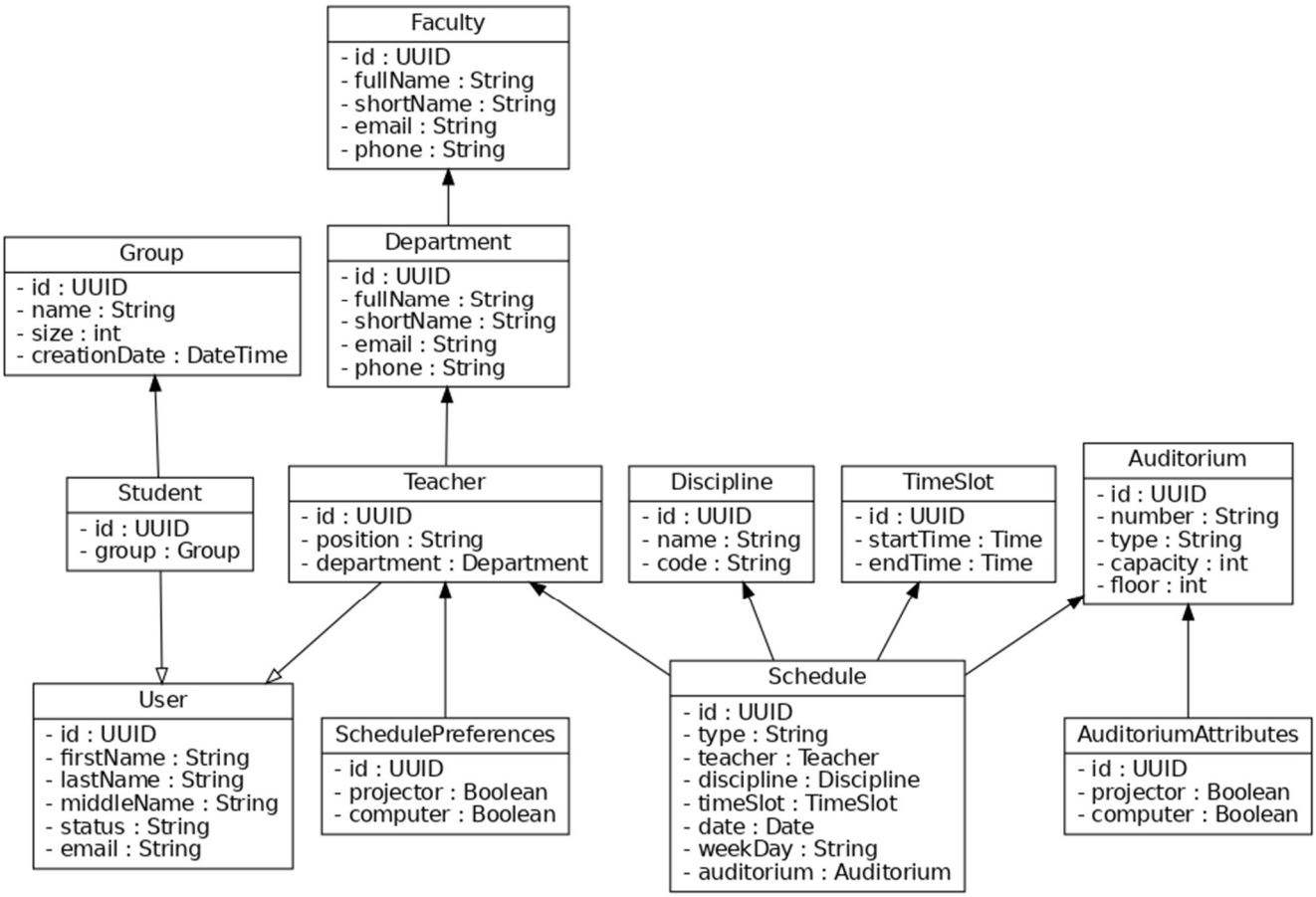

Рисунок 2.5 – Діаграма класів системи

2.4 Концептуальна модель бази даних

Одним із ключових етапів проєктування інформаційної системи є створення концептуальної моделі бази даних. Концептуальна модель визначає основні сутності системи, їхні властивості (атрибути) та взаємозв'язки. Вона слугує основою для розробки логічної та фізичної моделі бази даних і забезпечує чітке розуміння структури даних.

Мета концептуальної моделі полягає у відображенні предметної області системи без прив'язки до конкретної СКБД. Це дозволяє визначити:

– Основні дані, які необхідно зберігати.

– Зв’язки між елементами даних.

– Атрибути, які характеризують кожну сутність.

Для створення концептуальної моделі використовується діаграма «сутність-зв'язок» (Entity-Relationship Diagram, ERD), яка є основним інструментом моделювання даних. ERD допомагає представити структуру даних, визначаючи сутності, їх атрибути та взаємозв'язки. У процесі проєктування баз даних найчастіше використовуються дві основні нотації: Пітера Чена та «Вороняча лапка».

Для обґрунтування вибору нотації для створення ER-діаграми було здійснено дослідження нотацій, результати якого наведено в таблиці.

Таблиця 2.7 – Порівняння нотацій

Параметр	Нотація Пітера Чена	Нотація «Вороняча лапка»
Деталізація	Висока	Середня
Зручність читання	Помірна	Висока
Популярність	Використовується в освіті	Популярна у промисловості
Графічна простота	Складніша	Простіша

Після проведеного аналізу видно, що нотація «Вороняча лапка» є простішою та читабельною, але в той же час нотація Пітера Чена є більш деталізованою, що дозволяє уникнути помилок і забезпечити чітке розуміння моделі на всіх етапах розробки.

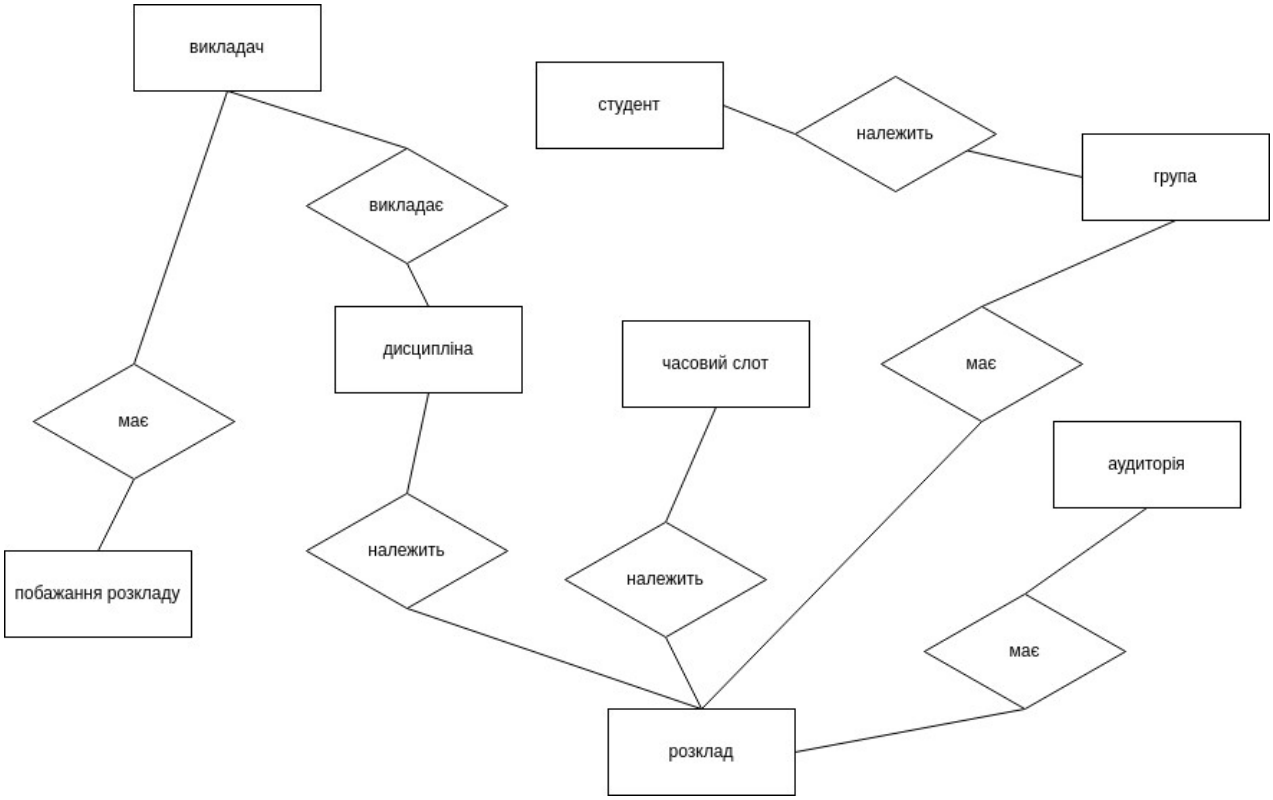

Рисунок 2.6 – концептуальна модель БД

Висновки до розділу 2

Розділ присвячений детальному опису процесу створення інформаційної системи для автоматичного складання розкладу з використанням генетичних алгоритмів. Генетичний підхід дозволяє ефективно вирішувати складні задачі оптимізації, зокрема адаптацію розкладу до жорстких і м'яких обмежень, таких як доступність ресурсів, побажання викладачів та рівномірне навантаження студентів.

Описані основні етапи алгоритму, від ініціалізації популяції до оптимізації та валідації розкладу, демонструють системний підхід до розробки. Представлені діаграми прецедентів і класів забезпечують графічне уявлення функціональності та структури системи, що спрощує її розробку та подальшу реалізацію.

Розглянуті методи, формули оцінки елементів розкладу та принципи роботи системи свідчать про високу адаптивність та інтерактивність розробленого рішення, що спрямоване на автоматизацію та покращення якості управління навчальними процесами.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Перед початком реалізації інформаційної системи необхідно вибрати перелік технологій, які будуть використані при створенні інформаційної системи.

При виборі технологій розробки системи формування розкладу занять для вищих навчальних закладів увага приділялась технологіям, які можна використовувати в комерційних цілях без сплати розробнику за користування.

Перед впровадженням необхідно вибрати платформу, на якій буде розгорнуто інформаційну систему, розробити архітектуру сервісів, визначити структуру бази даних і вибрати мову програмування. Важливо оцінити вибрані технології, щоб переконатися, що вони належним чином ліцензовані та відповідають потребам інформаційної системи. Далі наведено огляд технологій, які відповідають цим критеріям і задовольняють вимоги інформаційної системи, що будується.

3.1 Вибір мов програмування

Java є однією з найпоширеніших мов програмування у світі, яка надає кілька ключових переваг, які роблять її ідеальним вибором для створення складних, масштабованих і безпечних інформаційних систем.

Однією з ключових переваг використання Java є те, що це об’єктно-орієнтована мова програмування. Це означає, що вона забезпечує чітку та зрозумілу структуру для організації коду, що полегшує створення складних і масштабних інформаційних систем. Крім того, Java надає багатий набір бібліотек і фреймворків, які можна використовувати для розробки різних програм.

Ще однією перевагою Java є її незалежність від платформи. Код Java можна запускати на будь-якій платформі, на якій встановлено віртуальну машину Java (JVM), що спрощує розгортання та підтримку інформаційної системи у

різноманітних системах. Це робить мову програмування Java чудовим вибором для створення інформаційної системи.

Java також надає надійні функції безпеки, які є вирішальними при роботі з конфіденційними даними, такими як особиста інформація студентів. Java надає такі функції, як перевірка типів і обробка винятків, які допомагають забезпечити безпеку та стабільність інформаційної системи.

Також є плюсом те, що Java має велику й активну спільноту розробників і користувачів. Це полегшує пошук рішень проблем, отримання допомоги та використання досвіду та знань інших учасників спільноти.

3.2 Вибір фреймворків

Spring Framework є гнучким фреймворком для створення застосунків на мові програмування Java, і є найбільш популярним в області розробки серверних застосунків[12], що робить його чудовим вибором для розробки інформаційної системи керування розкладом. Spring Framework надає кілька ключових переваг, які роблять його ідеальним вибором для створення складних і масштабованих інформаційних систем.

Однією з ключових переваг Spring Framework є його модульна конструкція. Spring Framework складається з набору модульних компонентів, кожен з яких можна використовувати окремо або в комбінації для створення складних програм. Це спрощує додавання нових функцій, інтеграцій і функціональних можливостей інформаційної системи.

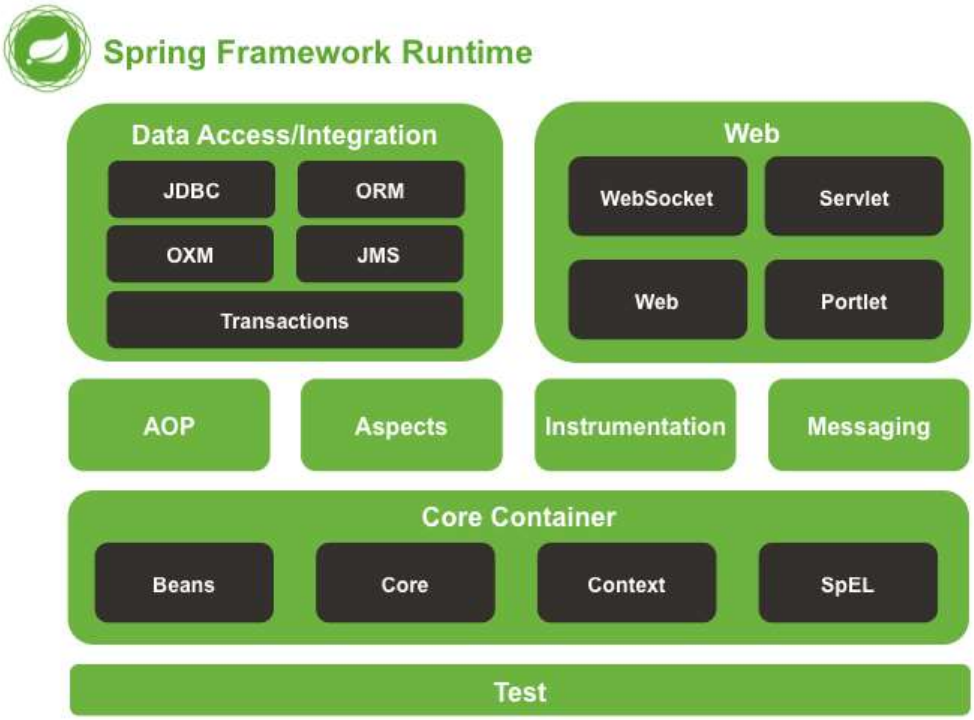

Рисунок 3.1 – структура Spring Framework [13]

Ще однією перевагою Spring Framework є наявність модуля Spring MVC у якому є підтримка REST API. За допомогою Spring Framework можна легко створювати та розгортати REST API, який необхідний для інтеграції інформаційної системи з іншими системами.

Spring Framework також має модуль Spring Security, який надає надійні функції безпеки, які мають вирішальне значення при роботі з конфіденційними даними, такими як особиста інформація користувачів. За допомогою Spring Framework можна легко реалізувати такі функції безпеки, як автентифікація та авторизація, щоб забезпечити безпеку даних користувачів.

Ще однією ключовою перевагою Spring Framework є те, що він має велику й активну спільноту розробників і користувачів, яка надає велику кількість ресурсів, навчальних посібників та інструментів. Це полегшує пошук вирішення проблем, отримання допомоги та використання досвіду та знань інших.

3.3 Вибір СКБД

Система керування базами даних (СКБД) – це комплекс програмного забезпечення, яке керує зберіганням, пошуком і маніпулюванням даними в базі даних. Метою СКБД є забезпечення централізованого, організованого та ефективного способу керування даними, і вона пропонує широкий спектр функціональних можливостей, таких як безпека даних, резервне копіювання та відновлення, контроль паралельності, інтеграція даних та узгодженість.

Structured Query Language (SQL, мова структурованих запитів) і NoSQL — це дві великі категорії СУБД, які відрізняються способом зберігання та отримання даних.

Бази даних SQL є реляційними базами даних, що означає, що дані організовані в таблиці з визначеними зв’язками між ними. Бази даних SQL добре підходять для структурованих даних, де дані можна легко представити в табличному форматі. Бази даних SQL використовують мову SQL для запитів і обробки даних, яка є стандартною мовою для реляційних баз даних. Прикладами баз даних SQL є PostgreSQL, MySQL, Oracle.

З іншого боку, бази даних NoSQL не використовують фіксовану схему і призначені для обробки неструктурованих або напівструктурованих даних, таких як зображення, відео або публікації в соціальних мережах. Бази даних NoSQL можна масштабувати горизонтально, що означає, що до системи можна додавати додаткові сервери, щоб збільшити її пропускну здатність, що робить їх добре придатними для великих даних і веб-додатків у реальному часі. Прикладами баз даних NoSQL є MongoDB, Cassandra та CouchDB.

PostgreSQL — це система керування реляційними базами даних із відкритим вихідним кодом, яка має багато розширених функцій, таких як розширене індексування, розширений захист і підтримку типів даних JSON і JSONB (двійковий JSON). PostgreSQL дозволяє користувачам визначати власні типи даних, а також підтримує кілька мов програмування, включаючи C/C++, C#, Java, Python.

PostgreSQL добре підходить для складних програм і часто використовується для сховищ даних, геопросторового аналізу та бізнес-аналітики. Надійний набір функцій PostgreSQL і дотримання стандартів SQL роблять його популярним вибором для веб-додатків і корпоративних програм, що керуються даними. Він також пропонує розширену аналітику даних і можливості звітування, що робить його ідеальним вибором для застосунків, що інтенсивно обробляють дані.

Підсумовуючи, вибір між СУБД SQL і NoSQL залежить від конкретних вимог інформаційної системи. Бази даних SQL найкраще підходять для структурованих даних, тоді як бази даних NoSQL краще підходять для неструктурованих даних. PostgreSQL — це потужна система керування реляційною базою даних, яка добре підходить для складних інформаційних систем. Її розширені функції, розширюваність і дотримання стандартів SQL роблять її чудовим вибором для застосунків, які інтенсивно обробляють дані.

3.4 Вибір інструментів розгортання змін схеми бази даних

Розгортання змін схеми бази даних — це важлива складова життєвого циклу розробки програмного забезпечення. Цей процес передбачає внесення змін до структури бази даних: створення таблиць, зміну їх структури у контрольований і повторюваний спосіб. Інструмент розгортання схеми бази джаних вирішує наступні питання:

– Контроль версій схеми бази даних: у проєктах зі складною структурою баз даних важливо точно знати, яка версія схеми використовується в різних середовищах (dev, release, production) - зміни мають бути документованими легковідтворюваними;

– Автоматизація процесів: ручне оновлення бази даних збільшує ризик помилок через людський фактор, тоді як інструмент розгортання схеми баз даних зменшує ризик виникнення помилок та забезпечує послідовність внесення змін в схему БД;

– Управління середовищами: забезпечення однакових умов для всіх середовищ (від тестового до бойового) для уникнення помилок, які виникають через невідповідність схем.

– Резервування і відкат: зміни мають бути зворотними у випадку помилок.

Liquibase — це популярний інструмент для управління змінами баз даних (Database Change Management). Він дозволяє створювати, відслідковувати, версіонувати і розгортати зміни в структурі баз даних[14].

Liquibase підтримує опис змін (change sets) у форматах YAML, XML, JSON або SQL. Інструмент підтримує популярні СКБД, таких як PostgreSQL, MySQL, Oracle, SQL Server, SQLite та інші. Liquibase реєструє зміни в спеціальній таблиці DATABASECHANGELOG, що дозволяє відслідковувати, які зміни вже застосовані до БД. Liquibase підтримує відкат змін (rollback) до певної версії або стану.

3.5 Вибір технологій розробки frontend частини інформаційної системи

Вибір технологій для розробки фронтенд-частини інформаційної системи є одним із ключових аспектів створення сучасного та ефективного продукту. Це рішення залежить від кількох факторів, таких як специфіка проекту, вимоги до продуктивності, досвід команди та доступний бюджет. Тому для вибору фреймфорку було вирішено врахувати його вік, популярність, криву навчання.

Згідно досліджень[15] є три найбільш популярні frontend фреймворки: React, Angular та Vue.js (рис.).

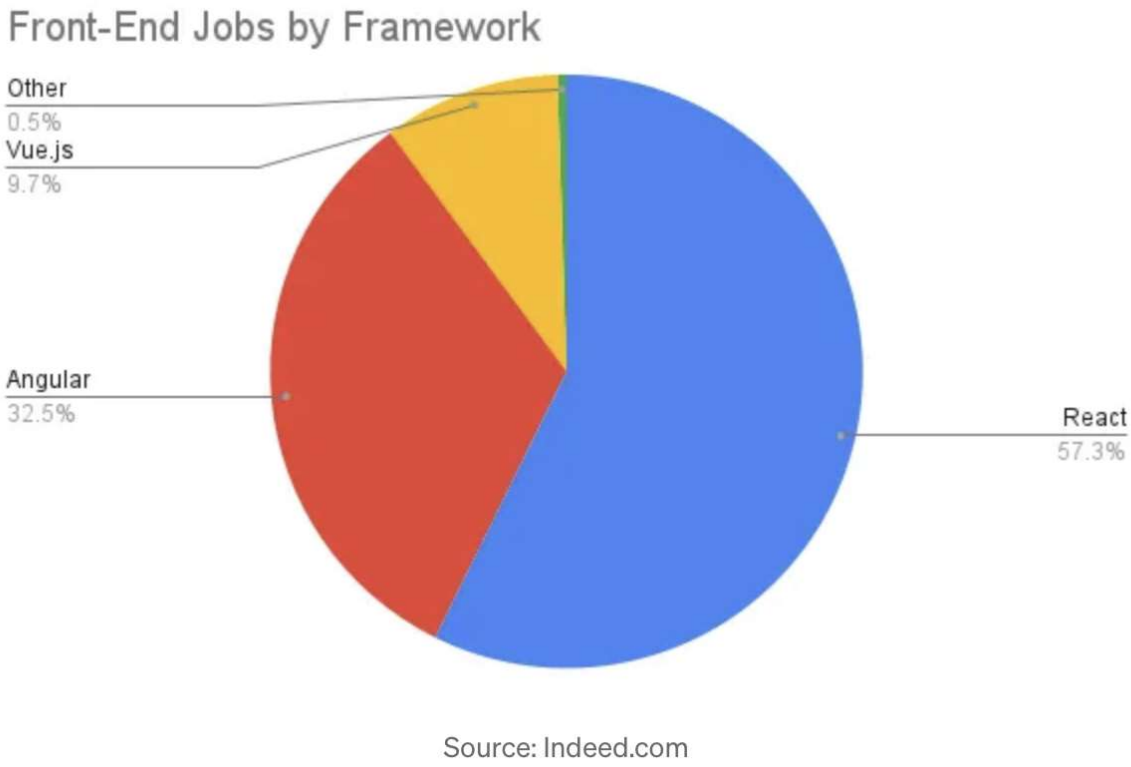

Рисунок 3.2 – популярність frontend фреймворків в розрізі вакансій

React (ReactJS) - бібліотека для спрощення побудови й маніпулювання DOM елементами, розроблена компанією Facebook. Спочатку розробка та використання бібліотеки відбувалось лише для внутрішніх потреб, й лише згодом зробили публічною.

У своєму чистому вигляді React є нічим більшим, аніж звичайнісінькою бібліотекою для написання простих frontend застосунків , оскільки не містить ані Routing, ані інших додаткових можливостей.

До плюсів можна віднести:

– максимально зрозумілий синтаксис, якщо вам знайомі Javascript і HTML;

– крива входу у користування React є максимально пологою;

– функціональний підхід до написання компонентів;

– стан DOM зберігається й обчислюється у віртуальному DOM, що робить рендерінг React дуже швидким;

Оскільки за замовчуванням React не включає практично жодних додаткових можливостей, окрім роботи з DOM, необхідно шукати бібліотеки для тих чи інших завдань.

На відміну від React, Angular є не просто бібліотекою для роботи з DOM, а цілим повноцінним фреймворком, розробленим компанією Google. Фреймвокр керується принципами Model-View-ViewModel побудови застосунків, й «з коробки» містить зокрема такі можливості, як:

– CLI для генерування структури файлів різних Angular-конструкцій, структури npm бібліотек, тестування, лінтування і т.д.

– Ряд бібліотек для самих різноманітних завдань — router, бібліотеки для роботи з формами, анімацією й інші.

– При ініціалізації проєкту можна доволі тонко налаштовувати те, що з доступних бібліотек буде додано до нього.

Історично, Angular хоча й виник трішки пізніше за React, фактично є переписаною з «нуля» версією фреймворку AngularJS, котрий існував раніше за нього.

До переваг фреймворку можна віднести:

– модульна архітектура дозволяє масштабувати застосунок та надає можливість чітко групувати елементи.

– повноцінний фреймворк, що має все необхідне для написання сучасних вебзастосунків;

– нативна підтримка typescript;

– використання Model-View-ViewModel принципів побудови застосунків, що дає змогу чітко розмежовувати реалізацію бізнес-логіки від її представлення.

– актуальна і вичерпна документація;

– велике ком’юніті, багато доступної в інтернеті інформації щодо будь-якого аспекту роботи фреймворку.

Мінуси фреймворку:

– крива входу є доволі крутою, оскільки структура фреймворку передбачає доволі великий обсяг знань, яким необхідно володіти для початку розробки;

– нестабільна швидкість роботи;

VueJS є наймолодшим з трійки фреймворків. Vue є свого роду сумішшю Angular і React. З одного боку, у чистому вигляді — це бібліотека для роботи з DOM, та з іншого боку — це фреймворк, у якого є офіційний Vue CLI й певна екосистема бібліотек навколо нього.

Vue широко використовує й наполягає на підході Single File Component, щодо написання компонентів, як основному. Його суть полягає в тому, що стилі, розмітка і логіка компонента мають знаходитись в одному файлі. Фреймворк підтримується зусиллями команди розробника бібліотеки та ком’юніті.

Плюси:

– висока швидкість збірки й роботи застосунку.

– кількість інструментів і бібліотек доволі обмежена, але є всі необхідні бібліотеки для роутингу та керування станом.

– малий розмір результатів збірки.

– підхід Single File Component дозволяє досить швидко розробити мінімально життєздатний продукт.

Мінуси:

– мінімалістична документація, але всі необхідні знання для початку розробки в документації наявні

В результаті дослідження було вирішено використати фреймворк VueJS через його простоту створення компонентів та молодий вік, що означає врахування недоліків фреймворків Angular та React.

3.6 Архітектура інформаційної системи

Інформаційна система формування розкладу занять для навчального закладу є багатофункціональною та складною платформою, яка включає модулі для генерації розкладу, управління користувачами, інтеграції з іншими системами, а також зберігання та обробки великих обсягів даних. У цьому контексті вибір модульної архітектури видається найраціональнішим рішенням, з огляду на її гнучкість, масштабованість і зручність в обслуговуванні.

Завдяки модульній архітектурі можна легко досягти гнучкості у керуванні модулями - легко додавати або змінювати модулі без впливу на інші частини системи. Додавання нових функцій не буде мати суттєвого впливу на продуктивність всієї системи через наявність гнучкого горизонтального та вертикального масштабування - ресурси системи виділяються відповідно потребам модуля, а не всієї системи.

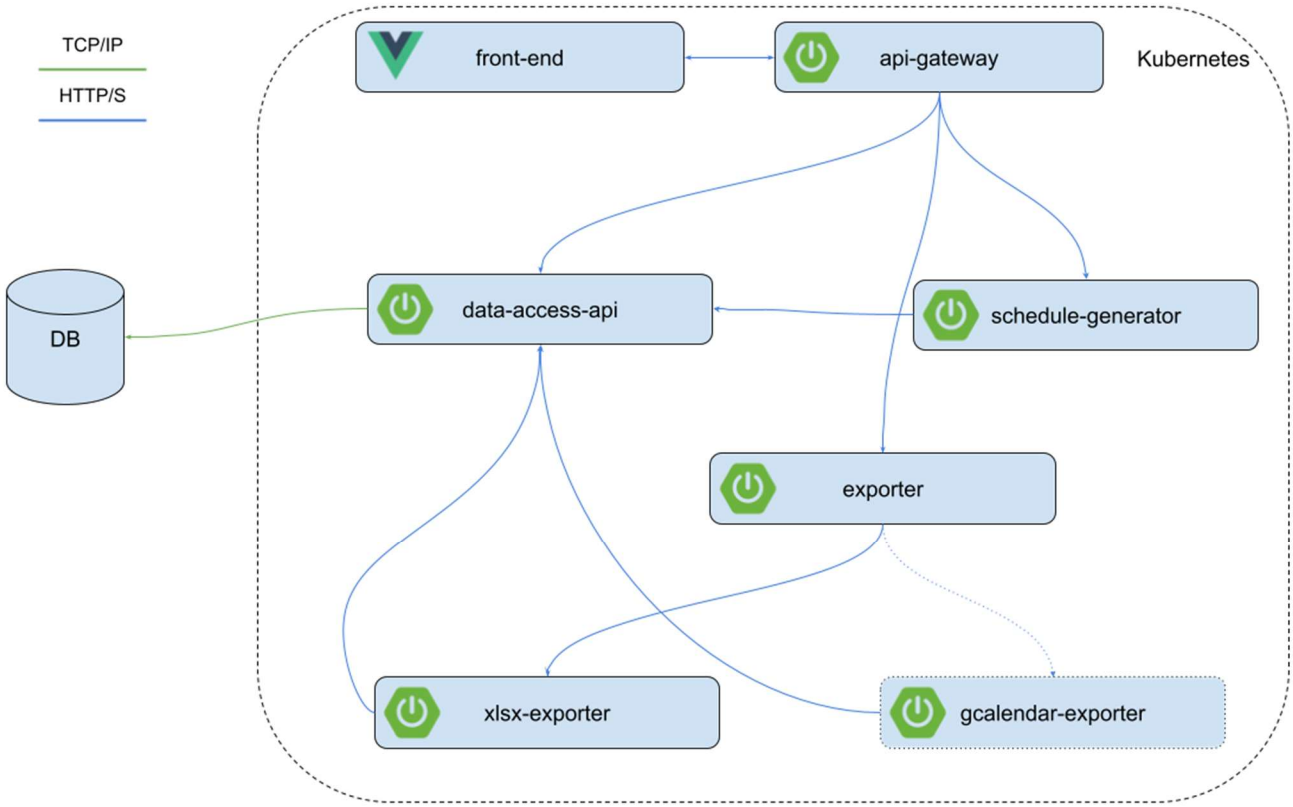

Рисунок 3.3 – архітектура інформаційної системи керування розкладом

3.7 Вибір середовища розгортання

Docker, Kubernetes та REST API є одними з найбільш часто використовуваних інструментів у сучасній розробці програмного забезпечення. Вони є популярним вибором для створення та розгортання інформаційних систем, особливо в контексті мікросервісів і хмарних систем.

Docker — це платформа для упаковки та розгортання застосунків в контейнерах[16]. Контейнери забезпечують упаковку застосунку та його залежностей в єдиний блок, який можна легко переміщувати між різними обчислювальними середовищами. Це полегшує керування та розгортання програм, а також їх тестування та налагодження.

Перевага використання Docker полягає в тому, що він дозволяє створити послідовне та відтворюване середовище для інформаційної системи. Це важливо, тому що інформаційна система повинна мати відтворюване середовище виконання, на кількох серверах, як на тестових, так і на робочих оточеннях. За допомогою Docker можна створити контейнер, який містить усі необхідні залежності та конфігурації, а потім розгорнути цей контейнер на будь-якому сервері, на якому встановлено Docker. Це зменшує ризик помилок конфігурації та гарантує, що програма завжди працюватиме належним чином.

Ще одна перевага використання Docker полягає в тому, що він полегшує масштабування програми. За допомогою Docker можна створити кілька екземплярів серверного застосунку інформаційної системи та запускати їх на різних серверах. Це дозволяє легко виконувати горизонтальне масштабування - додавати більше обчислювальних ресурсів до інформаційної системи по мірі зростання навантаження, за допомогою додавання нових серверів, що особливо важливо для інформаційної системи розкладу університету, якій може знадобитися працювати з великою кількістю користувачів і великими обсягами даних.

Kubernetes — це платформа оркестрації з відкритим кодом для автоматизації розгортання, масштабування та керування контейнерними застосунками[17].

Kubernetes забезпечує централізований спосіб керування контейнерами, що полегшує розгортання та керування великою кількістю контейнерів у масштабований і надійний спосіб на великій кількості серверів.

Основна перевага використання Kubernetes для інформаційної системи полягає в тому, що він забезпечує надійну та масштабовану платформу для розгортання контейнерів і керування ними. За допомогою Kubernetes можна легко керувати розгортанням, доступністю і масштабуванням інформаційної системи, що важливо, при використанні інформаційної системи.

Kubernetes також надає можливість керувати конфігурацією та безпекою інформаційної системи. За допомогою Kubernetes можна керувати конфігурацією своїх контейнерів, гарантуючи, що вони налаштовані правильно та безпечно. Це важливо для забезпечення відповідності інформаційної системи вимогам захисту конфіденційних даних.

Крім того, Kubernetes надає спосіб моніторингу та керування продуктивністю інформаційної системи. За допомогою Kubernetes ви можливо відстежувати продуктивність контейнерів і вносити необхідні зміни, щоб забезпечити безперебійну та ефективну роботу системи. Це важливо для підтримки надійності та доступності системи, а також для того, щоб користувачі могли отримати доступ до потрібної інформації, коли вона їм потрібна.

Ще одна перевага використання Kubernetes полягає в тому, що це полегшує керування життєвим циклом контейнерів - за допомогою Kubernetes ви можливо автоматизувати розгортання та масштабування контейнерів, а також керувати оновленнями та поверненням інформаційної системи до попередньої версії. Це спрощує обслуговування інформаційної системи та гарантує, що система завжди оновлюється та працює безперебійно.

Representational State Transfer (REST) — це архітектурний стиль для створення мережевих програмних систем. Цей архітектурний стиль базується на наборі обмежень і призначений для забезпечення єдиного інтерфейсу між

компонентами мережевої системи. REST характеризується використанням методів HTTP для виконання операцій над ресурсами, які ідентифікуються унікальними URI, і обміну представленнями цих ресурсів у стандартизованому форматі, наприклад JSON або XML. REST є популярним вибором для створення масштабованих і підтримуваних веб-служб і API[18]. REST є популярним вибором для створення API, оскільки він простий, гнучкий і масштабований.

Використання REST API для розробки інформаційної системи має кілька переваг. По-перше, REST API дозволяють легко надавати функціональні можливості інформаційної системи іншим системам, наприклад мобільним програмам або зовнішнім веб-сайтам. Це полегшує інтеграцію з іншими інструментами, які можуть бути корисними для LMS.

По-друге, REST API забезпечують гнучкий і масштабований спосіб доступу до даних і керування ними. За допомогою REST API легко версіонувати API та підтримувати зворотну сумісність, що важливо, оскільки інформаційна система розвивається та змінюється з часом.

Висновки до розділу 3

У розділі описано вибір технологій, архітектури та інструментів для створення інформаційної системи керування розкладом. Для реалізації backend обрано мову програмування Java та фреймворк Spring, який забезпечує модульність, підтримку REST API.

Як основу для зберігання даних обрано PostgreSQL — реляційну базу даних із відкритим вихідним кодом, яка підтримує складні запити та забезпечує високу продуктивність. Liquibase використовується як інструмент для управління змінами схеми бази даних, що дозволяє автоматизувати процес оновлення структури даних.

Для frontend проаналізовано три популярні фреймворки: React, Angular та Vue.js. Обрано React через його гнучкість, простоту, активну спільноту та низький поріг входження для розробників. Згідно з архітектурою, система має клієнт-

серверну структуру з поділом на модулі, що дозволяє масштабувати її відповідно до потреб університету.

Висновок підсумовує: обрані технології та архітектурні рішення забезпечують гнучкість, безпеку й адаптивність системи. Сформована архітектура дозволяє легко інтегрувати додаткові функціональні модулі, а використані інструменти гарантують надійність і ефективність роботи системи.

4 РОЗРОБКА ТА РОЗГОРТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Середовище розробки

Розробка програмного забезпечення для автоматичного формування розкладу занять вимагає використання сучасних інструментів, які забезпечують зручну роботу, високу продуктивність та інтеграцію з необхідними інструментами. Основною мовою програмування, на якій заплановано реалізацію інформаційної системи, є Java.

Середовище розробки IntelliJ IDEA було обрано як основний інструмент для розробки даної інформаційної системи. Відповідно до дослідження JetBrains [12], понад 75% розробників на Java використовують IntelliJ IDEA, що підтверджує її статус лідера серед інтегрованих середовищ розробки (IDE) для роботи з даною мовою програмування. Це рішення обумовлено низкою ключових переваг IntelliJ IDEA:

– Підтримка розробки на Java: IntelliJ IDEA надає широкий спектр інструментів для роботи з Java, що включає автодоповнення коду, інтеграцію з системами збірки (Gradle, Maven) та підтримку популярних фреймворків, таких як Spring.

– Розширення функціональності через плагіни: IntelliJ IDEA Ultimate версія забезпечує інтеграцію з іншими технологіями, які використовуються в проєкті.

- Vue.js: Вбудована підтримка плагіна для Vue.js дозволяє працювати із сучасними frontend технологіями. Інструмент надає можливість автодоповнення коду, відображення ієрархії компонентів, підтримує інтеграцію з TypeScript і пропонує зручності для роботи з шаблонами, стилями та скриптами.
- Database Tools: У складі Ultimate версії доступний потужний клієнт для роботи з базами даних. Він дозволяє підключатися до популярних

СКБД, таких як PostgreSQL, виконувати SQL-запити, переглядати структуру бази даних, редагувати записи та здійснювати міграції. Це забезпечує ефективну інтеграцію між серверною частиною проєкту та базою даних, а також полегшує аналіз і налагодження даних.

– Підтримка систем контролю версій: Вбудована інтеграція з Git дозволяє розробникам легко керувати змінами в коді, вести командну розробку та забезпечувати контроль версій програмного забезпечення.

– Інтуїтивно зрозумілий інтерфейс: Інтерфейс IntelliJ IDEA орієнтований на розробників, що дозволяє швидко опановувати інструмент навіть новачкам, забезпечуючи при цьому високу ефективність роботи з великими проєктами.

– Можливості для тестування та налагодження: IntelliJ IDEA підтримує різні підходи до тестування, включаючи JUnit і TestNG, що дозволяє забезпечити високу якість програмного забезпечення.

– Інтеграція з такими інструментами, як плагін Vue.js і Database Tools, які доступні в Ultimate версії, робить IntelliJ IDEA оптимальним вибором для створення як серверної, так і клієнтської частин програмного забезпечення. Це забезпечує комплексний підхід до розробки інформаційної системи та полегшує управління проєктом.

Таким чином, вибір IntelliJ IDEA Ultimate як основного середовища розробки забезпечує необхідні умови для створення масштабованого, надійного та інтегрованого програмного забезпечення. Завдяки потужному інструментарію, гнучкості налаштувань і високій популярності серед розробників, це середовище ідеально підходить для реалізації завдань проєкту.

4.2 Конфігурація оточення для впровадження інформаційної системи

Ефективне впровадження інформаційних систем складних і багатокомпонентних, зокрема таких, як системи автоматичної побудови розкладу, вимагає ретельного підходу до конфігурації оточення. Головними аспектами, що

забезпечують відтворюваність, масштабованість і стабільність окремих модулів системи, є використання контейнеризації за допомогою Docker.

Використовуючи docker можна не тільки створювати фінальні образи з застосунками інформаційної системи, але й виконувати компіляцію вихідного коду, під час збірки контейнера, що гарантує відтворюваність процесу збірки вихідного коду в незалежності від стану хостової системи. А використовуючи docker multistage build можна зменшити розмір фінального образу, завдяки “ігноруванню” проміжних артефактів компіляції вихідного коду. Для серверних застосунків створено файл Dockerfile(рис. 4.1), який складається з двох етапів:

– перший етап - компіляція вихідного коду застосунку в excutable jar
– другий етап - копіювання скомпільованого excutable jar з першого етапу безпосередньо в другий етап, в якому виконується створення образу, який буде запускатись на кінцевих серверах.

```
1  FROM maven:3.9.9-amazoncorretto-21-alpine as build-stage
2  WORKDIR /app
3  COPY . .
4  RUN mvn clean package
5
6  FROM amazoncorretto:21-alpine as production-stage
7  COPY --from=build-stage /app/Api/target/api.jar /app.jar
8  ENV TZ=Europe/Kiev
9  RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
10
11 EXPOSE 8080
12 ENTRYPOINT exec java $JAVA_OPTS -Dfile.encoding=UTF-8 -Djava.security.egd=file:/dev/./urandom -jar /app.jar
```

Рисунок 4.1 - вміст файлу Dockerfile для backend застосунків

Збірка образу для фронтальної частини інформаційної системи відбувається за тим самим принципом (рис. 4.2)

```
1  FROM node:20.15.0-alpine as build-stage
2  WORKDIR /app
3  COPY package*.json ./
4  RUN npm install
5  COPY . .
6  RUN mvn run build
7
8  FROM nginxinc/nginx-unprivileged:stable-alpine as production-stage
9  COPY --from=build-stage /app/dist /usr/share/nginx/html
10 RUN rm /etc/nginx/conf.d/default.conf
11 COPY nginx.cong /etc/nginx/conf.d/default.conf
12
13 EXPOSE 8080
14 CMD ["nginx", "-g", "daemon off;"]
```

Рисунок 4.2 - вміст файлу Dockerfile для frontend застосунку

Складність інформаційної системи автоматичної побудови розкладу спонукає на оптимізацію трудовитрат на розгортання та керування контейнерів інформаційної системи. Kubernetes є провідною платформою оркестрації контейнеризованих додатків, яка вирішує наявну проблему.

Для розгортання системи в Kubernetes використовуються такі основні компоненти:

– Pod — мінімальна одиниця обчислення в Kubernetes, яка містить один або кілька контейнерів;

– Deployment — компонент для керування оновленнями та масштабуванням застосунків;

– Service — забезпечує доступність додатка через IP-адресу або DNS;

– ConfigMap та Secret — використовуються для управління конфігурацією додатків;

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: data-access-api
  labels:
    app: data-access-api
spec:
  replicas: 3
  selector:
    matchLabels:
      app: data-access-api
  template:
    metadata:
      labels:
        app: data-access-api
    spec:
      containers:
        - name: data-access-api-container
          image: data-access-api:latest
          ports:
            - containerPort: 8080
          envFrom:
            - configMapRef:
                name: data-access-config
```

Рисунок 4.3 - конфігурація ресурсу Deployment для backend застосунків

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: data-access-api-config
data:
  DATABASE_URL: "postgresql://user:password@db-host:5432/schedule"
```

Рисунок 4.4 - конфігурація ресурсу ConfigMap для backend застосунків

```yaml
apiVersion: v1
kind: Service
metadata:
  name: data-access-api-service
spec:
  selector:
    app: data-access-api
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Рисунок 4.4 - конфігурація ресурсу Service для backend застосунків

4.2 Розгортання міграції схеми бази даних

```xml
<?xml version="1.1" encoding="UTF-8" standalone="no"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
                   xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
                   xmlns:pro="http://www.liquibase.org/xml/ns/pro" xmlns:xsi="http://www.w3.org/2001
                   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.li
    <changeSet id="1734326803510-1">
        <createTable tableName="auditorium">
            <column name="id" type="UUID">
                <constraints nullable="false" primaryKey="true" primaryKeyName="auditorium_pk"/>
            </column>
            <column name="number" type="TEXT">
                <constraints nullable="false"/>
            </column>
            <column name="type" type="TEXT">
                <constraints nullable="false"/>
            </column>
            <column name="capacity" type="INTEGER">
                <constraints nullable="false"/>
            </column>
            <column name="floor" type="INTEGER">
                <constraints nullable="false"/>
            </column>
        </createTable>
    </changeSet>
</databaseChangeLog>
```

Рисунок 4.5 - міграція схеми БД для сутності auditorium

```xml
<?xml version="1.1" encoding="UTF-8" standalone="no"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
                   xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
                   xmlns:pro="http://www.liquibase.org/xml/ns/pro" xmlns:xsi="http://www.w3.org/200
                   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.li
    <changeSet id="1734326803510-6">
        <createTable tableName="timeslot">
            <column name="id" type="UUID">
                <constraints nullable="false" primaryKey="true" primaryKeyName="timeslot_pk"/>
            </column>
            <column name="start_time" type="time(6) WITHOUT TIME ZONE">
                <constraints nullable="false"/>
            </column>
            <column name="end_time" type="time(6) WITHOUT TIME ZONE">
                <constraints nullable="false"/>
            </column>
        </createTable>
    </changeSet>
</databaseChangeLog>
```

Рисунок 4.6 - міграція схеми БД для сутності timeslot

Висновки до розділу 4

У цьому розділі описано практичну реалізацію інформаційної системи для автоматизованого складання навчальних розкладів, зокрема процеси її розробки, налаштування оточення.

Одним із центральних аспектів стало середовище розробки, для якого обрано сучасні й продуктивні інструменти. backend системи реалізовано з використанням мови програмування Java та фреймворку Spring, що забезпечує модульність. В якості СКБД використано PostgreSQL, яка гарантує ефективне управління даними завдяки розширеним функціям індексації та підтримці JSON-типів.

Для розгортання оточення для впровадження системи обрано Docker-контейнери, що дозволяють спростити розгортання та підтримку проєкту. Це рішення забезпечує незалежність від середовища розгортання та масштабованість платформи. Liquibase використано для управління змінами в базі даних, що дозволяє автоматизувати процес оновлення її структури.

Таким чином, розроблена інформаційна система є сучасним інструментом, що автоматизує процес створення розкладу. Вона забезпечує гнучкість, інтеграцію з іншими платформами та високу продуктивність, що робить її придатною для впровадження в навчальних закладах різного рівня. Подальший розвиток системи може включати інтеграцію з LMS-платформами та розширення функціоналу для управління ресурсами.

ВИСНОВКИ

Виконано комплексне дослідження та розроблено програмне забезпечення для автоматизованого формування розкладу занять, що враховує матеріально-технічні вимоги та доступність ресурсів. Основна увага була зосереджена на вирішенні багатокритеріальної задачі оптимізації, яка є ключовою у створенні ефективного та збалансованого розкладу для навчальних закладів.

На першому етапі роботи було проведено глибокий аналіз предметної області, включаючи вивчення існуючих систем керування розкладом та їх функціональних можливостей. Виявлено недоліки в сучасних рішеннях, зокрема обмеженість у автоматизації процесів і відсутність інтеграції з навчальними потребами. Це дозволило сформувати перелік вимог до нової системи, яка мала враховувати як жорсткі обмеження (наявність ресурсів, доступність аудиторій та викладачів), так і м'які критерії, такі як рівномірність навантаження та мінімізація "вікон" у розкладі.

Розробка алгоритмів була однією з основних складових роботи. Було використано генетичний підхід, який зарекомендував себе як ефективний інструмент для розв’язання задач оптимізації.

На архітектурному рівні система була побудована з використанням сучасних технологій, таких як Spring Framework для серверної частини та PostgreSQL для зберігання даних. Це забезпечує модульність, масштабованість і гнучкість програмного забезпечення, що спростить модифікацію системи в майбутньому та її інтеграцію з іншими платформами, зокрема системами управління навчанням.

Практична реалізація включала впровадження розробленого програмного забезпечення на реальних даних. Отримані результати підтвердили його відповідність вимогам та ефективність у вирішенні поставлених завдань.

Автоматизація процесу формування розкладу значно зменшила адміністративне навантаження, забезпечила високу точність та зручність роботи із системою.

Таким чином, виконана робота не лише запропонувала інноваційне рішення для автоматизації складання розкладу, але й заклала основу для подальшого розвитку інформаційної системи. Перспективи вдосконалення включають впровадження технологій штучного інтелекту, інтеграцію з хмарними платформами та розширення функціоналу для підтримки різних форматів навчання, зокрема дистанційного. Робота підтвердила актуальність дослідження і практичну цінність розробленого програмного забезпечення для закладів освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про розклад навчальних занять. *Харківська гуманітарно-педагогічна академія*. URL: http://www.hgpa.kharkov.com/wp-content/uploads/2019/normatyvni/pologennya_rozklad.pdf (дата звернення: 16.12.2024).

2. Google Trends: See what's trending across Google Search, Google News and YouTube. - Google News Initiative. *Google News Initiative - Home*. URL: https://newsinitiative.withgoogle.com/resources/trainings/google-trends-lesson/ (дата звернення: 16.12.2024).

3. Compare LMS. *Google Trends*. URL: https://trends.google.com/trends/explore?date=today%205-y&geo=UA&q=/m/021x7z,/m/010pkp62,Нові%20знання,/g/11cmtv4s6p,/m/03by31t&hl=uk (дата звернення: 16.12.2024).

4. About Moodle - MoodleDocs. *MoodleDocs*. URL: https://docs.moodle.org/405/en/About_Moodle#What_is_Moodle? (дата звернення: 16.12.2024).

5. Features - MoodleDocs. *MoodleDocs*. URL: https://docs.moodle.org/405/en/Features (дата звернення: 16.12.2024).

6. Classroom Management Tools & Resources. *Google for Education*. URL: https://edu.google.com/workspace-for-education/classroom (дата звернення: 16.12.2024).

7. About Classroom - Classroom Help. *Google Help*. URL: https://support.google.com/edu/classroom/answer/6020279?hl=en&sjid=18180740729701049036-EU (дата звернення: 16.12.2024).

8. Допомога | Нові знання. *Нові знання*. URL: https://nz.ua/page/support (дата звернення: 16.12.2024).

9. Кононюк А. Ю. Нейроні мережі і генетичні алгоритми. Київ : Корнійчук, 2008. 446 с. URL: https://ecat.ust.edu.ua/ft/NN_GA.pdf (дата звернення: 16.12.2024).

10. Yurchak I., Moskovych T. Applying of genetic algorithms in the automated workload distribution system for teachers and students. *Computer systems and network*. 2018. Т. 1, № 905. С. 149–157. URL: https://doi.org/10.23939/csn2018.905.149 (дата звернення: 16.12.2024).

11. Maxym Z. Варіанти використання та сценарії (Use Cases and Scenarios). *Maxym Zosym*. URL: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення: 16.12.2024).

12. Java programming - the state of developer ecosystem in 2023 infographic. *JetBrains: Developer Tools for Professionals and Teams*. URL: https://www.jetbrains.com/lp/devecosystem-2023/java (дата звернення: 16.12.2024).

13. 2. Introduction to the Spring Framework. *Spring | Home*. URL: https://docs.spring.io/spring-framework/docs/4.3.15.RELEASE/spring-framework-reference/html/overview.html (дата звернення: 16.12.2024).

14. Introduction to Liquibase. *Liquibase Documentation*. URL: https://docs.liquibase.com/concepts/introduction-to-liquibase.html (дата звернення: 16.12.2024).

15. Порівнюємо React, Angular і Vue – найпопулярніші бібліотеки й фреймворки у 2022 році. *DOU*. URL: https://dou.ua/forums/topic/39933 (дата звернення: 16.12.2024)

16. Docker: Accelerated Container Application Development. *Docker*. URL: https://docker.com (дата звернення: 16.12.2024).

17. Production-Grade Container Orchestration. *Kubernetes*. URL: https://kubernetes.io (дата звернення: 16.12.2024).

18. Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST). *Home - UC Irvine Donald Bren School of Information & Computer Sciences*. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення: 16.12.2024).

ДОДАТОК А

Апробація кваліфікаційної роботи

Міністерство освіти і науки України
Чорноморський національний університет
імені Петра Могили

«Інформаційні технології та інженерія»

Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів

ТЕЗИ

7–10 лютого 2023 року

Пасіченко М. Г., Давиденко Є. О. Алгоритми побудови розкладу занять університету 132