Проведений аналіз показав, що єдиного методу, кінцево-різницевої схеми, для різних моделей течії до сих пір не було: кожна з наведених чисельних схем розв'язувала лише певну задачу.

Показано, що чисельно-аналітичний метод здатен розв'язувати задачі генерації шуму гелікоптера для різних типів течії, як потенціальної, так і не потенціальної. Ця особливість методу дозволяє використовувати його для різних задач з моделювання шуму аеродинамічного походження.

***Ключові слова**: шум ротора гелікоптера, чисельні методи, чисельно-аналітичний метод.*

Одержано редакцією 20.01.2022 р.
Прийнято до публікації 23.06.2022 р.

УДК 519.688

DOI 10.31651/2076-5886-2022-1-18-30

PACS 02.60.-x, 02.60.Pn

ГУР'ЄВ Ярослав Ігорович
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького
e-mail: yaroslavhuriev@gmail.com

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені БогданаХмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

РОЗРОБКА ВЕБ-ОРІЄНТОВАНОГО ІНСТРУМЕНТАЛЬНОГО СЕРЕДОВИЩА ДЛЯ КЕРУВАННЯ РОЗКЛАДОМ ЗАНЯТЬ

У даній статті представлено розробку веб-орієнтованого інструментального середовища для ефективного керування розкладом занять в навчальних закладах. Зростання кількості учнів та викладачів у сучасних освітніх установах ставить перед керівництвом виклик в ефективному плануванні та уникненні конфліктів у графіку навчальних занять. Програмний інструмент, що пропонується, надає зручний інтерфейс для складання та оновлення розкладу, дозволяючи враховувати різноманітні обмеження, наявність різних груп студентів, викладачів та аудиторій. Призначений інструмент допомагає автоматизувати процес планування, забезпечує оптимальне розподілення ресурсів та допомагає уникнути помилок, що можуть виникати при складанні розкладу вручну. Результати експериментів підтверджують ефективність запропонованого інструменту та переваги використання веб-технологій для керування розкладом навчальних занять.

***Ключові слова:** розклад, автоматична генерація розкладу, генетичний алгоритм, веб-додаток.*

Вступ

Сучасні навчальні заклади стикаються з глобальною проблемою управління та автоматизованого створення розкладу навчальних занять. Зараз кількість учнів та викладачів швидко зростає, що призводить до збільшення зусиль, необхідних для створення розкладу, який задовольнить всіх учасників навчального процесу і уникне конфліктів. Нерідкі ситуації, коли одного викладача призначають на заняття в двох різних аудиторіях одночасно, призводять до поспіху та виправлень, що вже на етапі проведення занять може спричинити проблеми. Тому майбутнє управління розкладами належить веб-додаткам, які допоможуть укладачам розкладу зекономити час та уникнути грубих помилок, подібних згаданій. Такі веб-додатки будуть не тільки зручним інструментом для планування навчального процесу, але й допоможуть підвищити його ефективність та організованість, забезпечуючи зручний та оптимальний розклад для всіх учасників.

Мета статті – провівши аналіз аналогічних додатків розробити архітектуру та створити веб-орієнтоване інструментальне середовище для керування розкладом занять.

Практичне значення одержаних результатів. Додаток розроблений у ході написання статті можна використовувати для керування розкладом занять у навчальних закладах.

Виклад основного матеріалу

1. Концепція додатку та огляд предметної області

1.1 Огляд веб-додатків схожої тематики

У сфері генерації розкладів навчальних занять існує значна кількість програм, проте більшість з них є платними. У цьому розділі ми розглянемо безкоштовні альтернативи, зокрема FET, Timetable-Plus Spring Lite та College Schedule Maker.

FET є безкоштовним програмним забезпеченням, призначеним для генерації розкладів у школах та університетах. Він використовує швидкий та ефективний алгоритм для складання розкладів, і може вирішувати складні задачі за дуже короткий час - від 5 до 20 хвилин. У простіших випадках цей процес може займати менше 5 хвилин, а іноді навіть всього декілька секунд. Але для дуже складних задач, може знадобитись кілька годин.

Основні переваги FET включають:

- Широкий вибір локалізацій, включаючи українську.
- Повністю автоматизований алгоритм генерації розкладу, який при цьому дозволяє коригування результатів вручну.
- Кросплатформена реалізація програми, що означає, що вона доступна для різних операційних систем.
- Згенеровані розклади можна експортувати у формати CSV, XML або HTML.
- Підтримка великої кількості можливих часових та просторових обмежень, що дозволяє враховувати специфічні вимоги та обмеження для кожного розкладу.

Іншим інструментом для генерації розкладу занять є Timetable-Plus Spring Lite - це безкоштовне програмне забезпечення, яке дозволяє створювати розклади занять без конфліктів між ресурсами. Цей інструмент володіє гнучкістю вхідних даних та надає численні налаштування обмежень ресурсів, що дозволяє генерувати розклади як для звичайних шкіл, так і для коледжів і університетів.

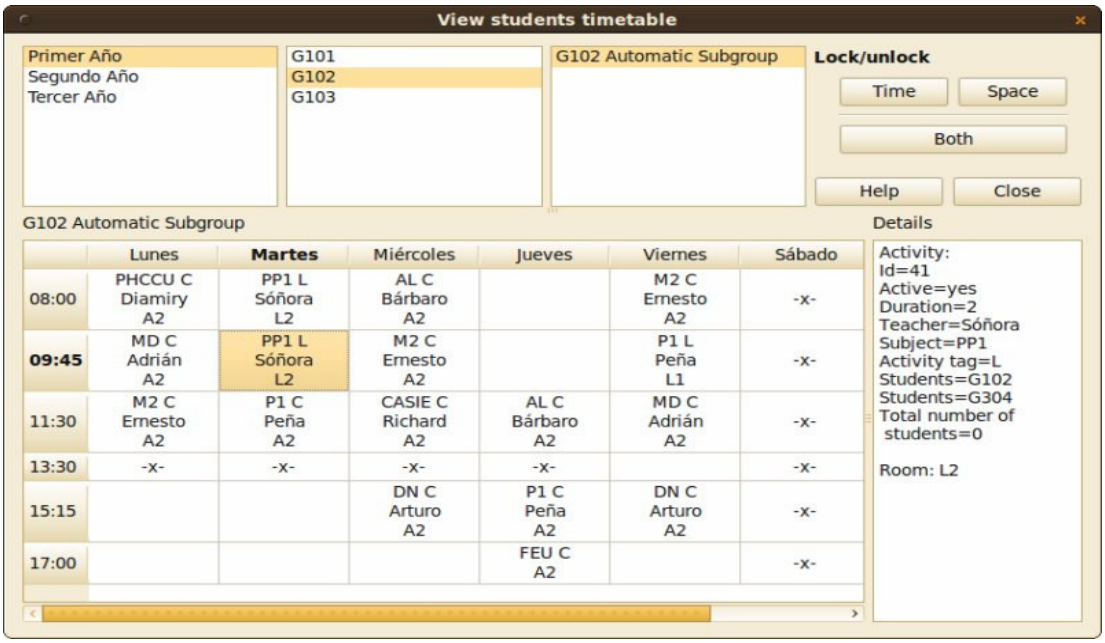

Рис 1. Загальний вигляд застосунку FET

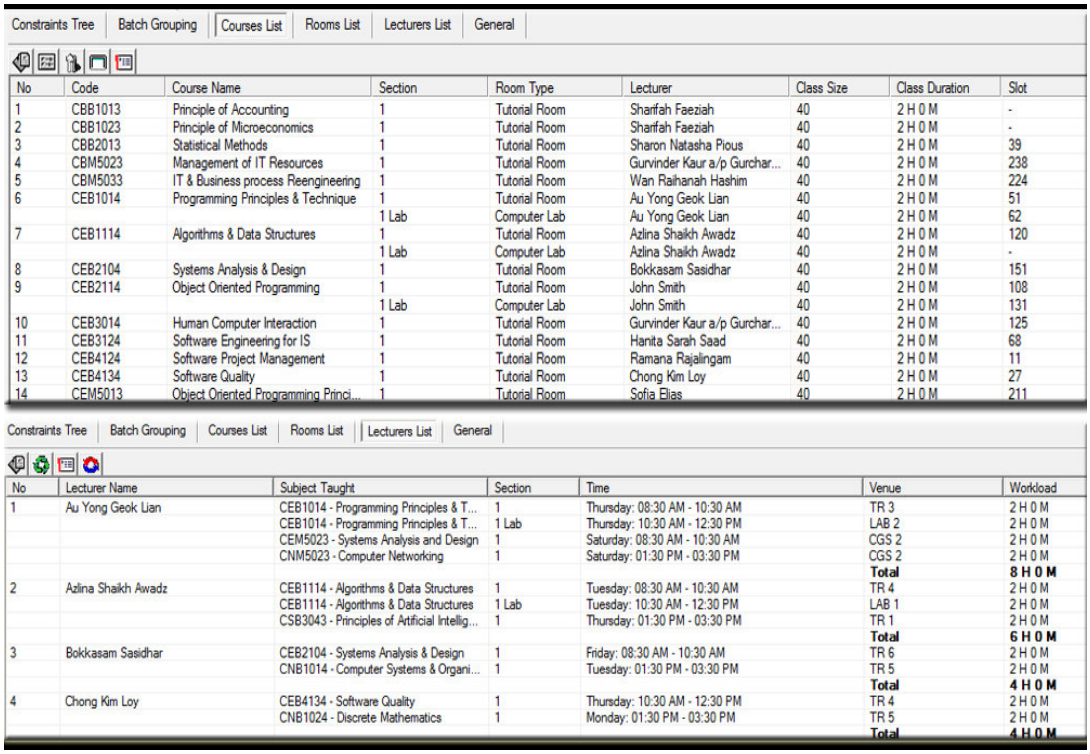

Рис 2. Загальний вигляд застосунку Timetable Plus Spring Lite

Основними особливостями Timetable-Plus Spring Lite є:

- Відсутність обмежень на розмір бази даних та вхідних даних, що робить його дуже гнучким і здатним працювати з великими обсягами інформації.
- Використання ієрархії при введенні даних та обмежень, що дозволяє структурувати інформацію за батьківськими та дочірніми елементами.
- Велика кількість налаштувань для обмежень по ресурсах, що дозволяє точно визначати параметри розкладу.

– Можливість ручного дозаповнення розкладу після генерації, що дозволяє виправити дрібні недоліки або врахувати специфічні потреби.
– Гнучкі налаштування для друку розкладів, що дозволяє вибрати оптимальний формат виведення інформації.
– Програма призначена для використання лише одним користувачем, що робить його зручним і ефективним інструментом для індивідуального використання.

Іншим інструментом для генерації розкладу є College Schedule Maker, який є веб-додатком. College Schedule Maker має простий функціонал, який дозволяє створювати розклад на тиждень для шкіл та коледжів. Відмінністю цього додатка від попередніх є його веб-орієнтованість та відсутність автоматичної генерації розкладу, користувач самостійно вносить заняття.

Основні можливості даного додатка включають:

– Друк розкладу, що дозволяє зручно матеріалізувати розклад та використовувати його офлайн.
– Можливість створення необмеженої кількості розкладів, що дозволяє використовувати додаток для різних навчальних закладів чи періодів.
– Збереження розкладу у якості зображення, що дозволяє зручно ділитися розкладом з іншими користувачами або друкувати його у зручному форматі.
– Імпорт та експорт розкладу, що дозволяє легко переміщувати розклад між різними пристроями або програмами, а також ділитися розкладом з іншими користувачами.

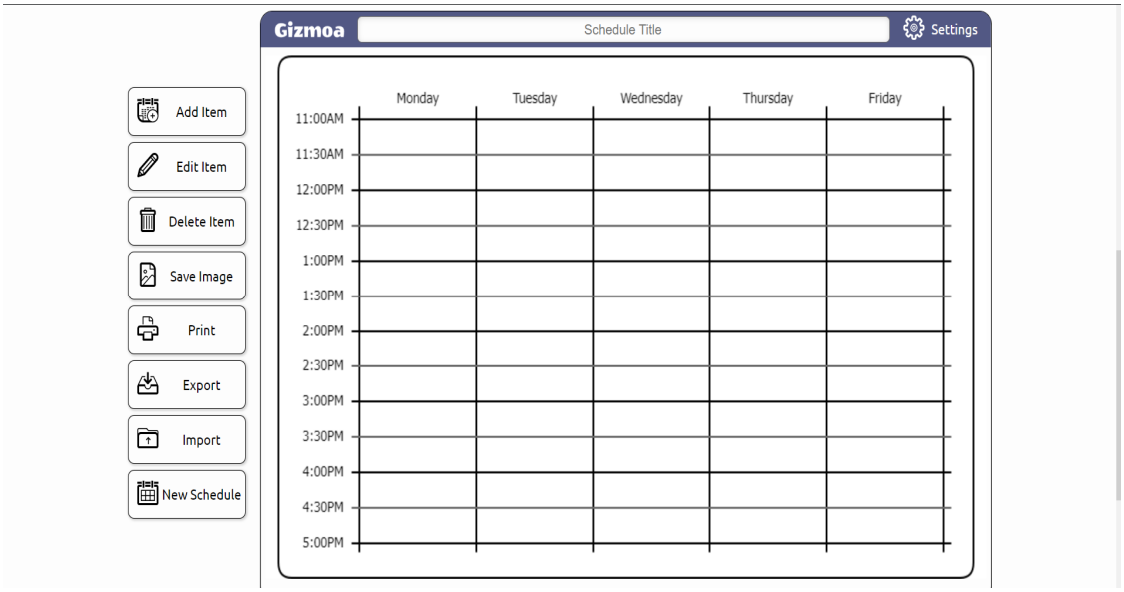

Рис 3. Загальний вигляд додатка College Schedule Maker

1.2 Опис концепції веб-додатку

Метою розробленого веб-додатка є максимальна спрощення процесу створення розкладу занять та економія часу для адміністраторів, які зазвичай створюють розклади вручну. Для досягнення цієї мети використовується генетичний алгоритм[1,2], що забезпечує автоматичну генерацію розкладів. Веб-орієнтованість додатка дозволяє адміністраторам легко поділитися створеним розкладом з користувачами, які просто мають авторизуватися в системі для доступу до актуального розкладу занять.

В системі виділяються два типи користувачів: адміністратори та звичайні користувачі. Адміністраторам доступний повний функціонал системи, зокрема, вони можуть редагувати та видаляти академічні групи та викладачів, створювати вхідні дані,

генерувати, переглядати та видаляти розклади, налаштовувати параметри генетичного алгоритму та обирати активний розклад.

Звичайні користувачі мають лише доступ до перегляду поточного розкладу, забезпечуючи синхронізацію між усіма користувачами системи. Надмірний пошук не потрібен, оскільки вся інформація про розклад знаходиться в одному місці, що робить його зручним у використанні.

Окрім того, велику увагу приділено можливості повторного використання вхідних даних, що дозволяє адміністраторам швидко створювати нові розклади, внесенням змін до існуючих даних, а не повторним вводом всієї інформації. Такий підхід мінімізує затрати часу на генерацію нових розкладів.

Останньою важливою особливістю додатка є його простота та інтуїтивність. Завдяки навігаційній панелі, яка містить всі можливі шляхи в додатку, користувачам легко орієнтуватися та не загубитися серед функцій програми.

2. Розробка веб-додатку та використані технології

У процесі створення серверної частини додатку було використано наступні технології: в якості фреймворку для web API частини був використаний .NET 6 мови програмування C#, за базу даних було обрано PostgreSQL. У клієнтській частині були використані бібліотеки React та Material UI мови JavaScript.

2.1 .NET

.NET є безкоштовною крос-платформеною програмною платформою з відкритим кодом, призначеною для розробки різноманітних застосунків. Вона побудована на основі високопродуктивного середовища виконання, яке успішно застосовується в багатьох великих масштабованих проектах [3].

За результатами опитування серед українських розробників у 2022 році, мова програмування C# займає друге місце за популярністю в комерційних проектах, поступаючись лише JavaScript. У розробці back-end додатків C# займає третє місце [4]. Головний конкурент C# - мова програмування Java.

Основні переваги C# та .NET, що сприяли їх вибору для розробки системи генерації та керування розкладом, включають:

- Використання Language Integrated Query (LINQ) - простої та високорівневої мови запитів до джерела даних, що інтегрована в C# [5].
- Використання auto-properties та зручного синтаксису, наприклад, null-conditional операторів.
- Проста та зрозуміла модель асинхронного програмування.
- Велика кількість сторонніх пакетів, зокрема, можливість використання менеджера пакетів NuGet.

З метою об’єктивності варто також зазначити деякі недоліки .NET:

- JIT компіляція може призводити до "підвисань" при першому запуску додатку, оскільки вона відбувається під час роботи програми.
- Відносно проста можливість декомпіляції коду, через використання JIT компіляції.
- Складність структури .csproj файлів.

2.2 PostgreSQL

PostgreSQL є потужною системою керування об’єктно-реляційними базами даних з відкритим кодом. Вона розширює мову запитів SQL та пропонує багато можливостей, які дозволяють безпечно зберігати та масштабувати найбільш складні об’єми даних [6]. Початково при розробці системи для керування та генерації розкладу, планувалось використовувати нереляційну базу даних. Проте, згодом виявилося, що з урахуванням

архітектури програмного продукту, такий підхід буде складно реалізувати, оскільки більша частина об’єктів у базі даних мають між собою зв’язки. Тому було прийнято рішення замінити нереляційну базу даних на PostgreSQL.

Основні переваги PostgreSQL включають:

- Відкритий код.
- Висока сумісність із стандартом SQL.
- Можливість роботи зі складними типами даних.
- Гнучкий повнотекстовий пошук.
- Підтримка великої кількості мов програмування.
- Кросплатформеність.

Серед недоліків PostgreSQL можна виділити:

- Порівняно низька швидкість читання даних (порівняно з MySQL).
- Деякі застосунки з відкритим кодом можуть підтримувати лише MySQL, але не підтримувати PostgreSQL.
- Розширена документація доступна лише англійською мовою.

2.3 React

React - це безкоштовна фронтенд-бібліотека мови JavaScript з відкритим кодом, яка призначена для створення користувацьких інтерфейсів на основі UI компонентів [7].

Серед позитивних особливостей React можна відзначити наступне:

- React оновлює лише ті частини DOM браузера, стан яких змінився, завдяки використанню віртуального DOM.
- Використання JSX для написання React-компонентів, що спрощує код та надає синтаксичний цукор для швидшої розробки.
- Використання об'єкта props для передачі даних від батьківського до дочірнього компонента та подій для зворотної передачі даних.
- Можливість перевикористання React-компонентів.

Однак є також деякі недоліки React:

- React-компоненти містять як представлення, так і логіку, що може створювати проблеми при читанні та розумінні коду.
- Відсутність підтримки деякого ключового функціоналу для фронтенду, такого як роутинг та HTTP запити, що потребує підключення сторонніх бібліотек.
- Застосунки на основі React часто вимагають велику кількість сторонніх бібліотек, що може призводити до конфліктів між React та цими бібліотеками.

2.4 Material UI

Material UI - це бібліотека React компонентів з відкритим кодом, яка пропонує реалізацію дизайну Material Design від Google. Вона містить готовий набір компонентів, які можуть бути використані "з коробки" [8]. Використання цієї бібліотеки дозволяє уникнути ручної розмітки з використанням CSS та HTML, дозволяючи розробникам зосередитись на бізнес-логіці своїх застосунків. Завдяки великому розмаїттю компонентів, Material UI відповідає типовим потребам фронтенд розробників. Компоненти Material UI поділені на категорії, такі як введення даних, відображення даних, засоби зворотного зв'язку з користувачем, елементи поверхні, елементи навігації та макети. Кожну з цих категорій складають компоненти, які можна легко налаштовувати під вимоги конкретного застосунку.

3. Реалізація веб-додатку

3.1 Реалізація серверної частини

Для реалізації інтерфейсу застосунку було створено REST API [9]. Цей підхід дозволяє розробляти кілька клієнтських додатків, які використовують одне і те саме API, оскільки REST працює з ресурсами, а не з представленнями. В конкретному випадку ресурси, з якими взаємодіє API, включають Groups, Lessons, Login, Schedules, Schedule Inputs, Teachers та Users.

Генерацію розкладу здійснює метод POST ресурсу /schedules. При виклику цього методу, контролер отримує id вхідних даних, для яких потрібно згенерувати розклад. За допомогою цього id з бази даних отримуються відповідні вхідні дані та передаються до методу Solve в класі Solver. Саме цей метод містить в собі всю логіку генетичного алгоритму.

На початку алгоритму створюються розклади, в яких заняття випадковим чином розміщені у часі (день та номер пари). Потім алгоритм переходить у цикл, в якому спочатку обчислюються значення фітнес-функцій для кожного елементу популяції, з урахуванням штрафів. Штрафи включають такі категорії: штраф за вільні проміжки часу між парами викладачів або груп, штраф за пізні пари, які не відповідають вказаним параметрам, а також штраф за одноманітні дисципліни в групі (якщо в один день з'являється більше ніж n занять з однієї дисципліни, де n - параметр фітнес-функції). Значення штрафів та параметри, що визначають, коли штраф застосовується, можуть бути налаштовані окремо.

Далі популяція сортується в порядку спадання, і якщо елемент з індексом 0 має значення фітнес-функції 0, це означає, що знайдено ідеальний розв'язок, і його повертають. В іншому випадку відбувається відбір хромосом: з найкращим фітнесом обираються 1/4 популяції, і з кожної хромосоми створюються три нові мутовані хромосоми, переміщуючи два випадкові заняття в розкладі.

Цикл продовжується або до закінчення заданої кількості ітерацій, або поки не знайдено ідеальний розклад. Якщо цикл завершився, а ідеальний розклад досі не знайдений, то буде повернуто найкращий розклад популяції. Код алгоритму можна переглянути в додатку А.

Для більшості ресурсів (з винятком Login та POST методу Users) доступ вимагає авторизації з метою запобігання незаконному доступу користувачів ззовні умовної корпоративної мережі університету, який використовує цей застосунок для складання розкладу. Авторизація здійснюється на основі ролей, і в системі існують дві ролі: Admin та User. Операції, які користувачі можуть виконувати, розділено на ці ролі. Користувач з роллю Admin має доступ до всіх ресурсів у системі, тоді як користувачі з роллю User обмежені у можливості та мають лише право перегляду поточного розкладу. При реєстрації нового користувача в системі за замовчуванням встановлюється роль User, і в системі існує тільки один користувач з роллю Admin.

З програмної точки зору, авторизація реалізована за допомогою JWT (JSON Web Token). JWT - це відкритий стандарт, який забезпечує компактний і самодостатній метод безпечної передачі даних у форматі JSON між сторонами. Інформація у токені може бути перевірена та підтверджена завдяки підпису [10]. JWT складається з трьох частин: заголовка (header), вмісту (payload) та підпису (signature). Заголовок зазвичай містить тип токена та алгоритм підпису. Вміст містить інформацію, яку токен повинен передавати. Як видно з рис. 4, для токена з роллю Admin, який був згенерований у застосунку, вміст містить дані про ім'я користувача "admin", його роль "Admin", а також інформацію про те, коли токен стане недійсним та видавця токену.

```
{

"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
name": "admin",

"http://schemas.microsoft.com/ws/2008/06/identity/claim
s/role": "Admin",
  "exp": 1671115818,
  "iss": "ScheduleSystemIDP",
  "aud": "http://localhost:44485/"
}
```


Рис 4. Вміст JWT токена із роллю Admin

Токен у розробленому застосунку генерується за допомогою вбудованого сервісу TokenService. Згенерований токен дійсний протягом 2 годин, після чого користувач знову повинен буде авторизуватися для отримання нового токену.

Серверна частина програмного продукту побудована відповідно до принципів "Чистої архітектури" [11]. Основним принципом цієї архітектури є розділення застосунку на чотири окремі шари, як показано на рисунку 5.

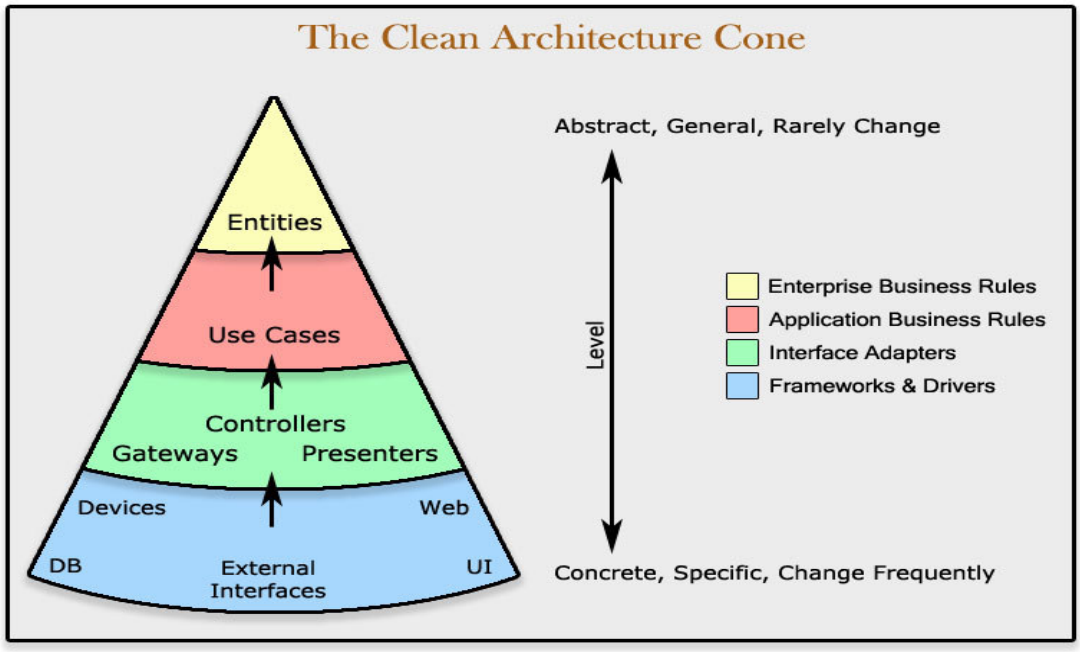

Рис 5. Схема шарів застосунку з чистою архітектурою

На рисунку 5 представлені чотири шари чистої архітектури. Синім позначено шар інфраструктури, який включає конкретні фреймворки, драйвери та інші технічні

компоненти. Зеленим позначений шар адаптерів інтерфейсів, до якого належать контролери (в даному застосунку). Червоним позначено шар "Use Cases", який містить специфічну для застосунку бізнес-логіку. Останній жовтий шар - це шар сутностей, які представляють загальні бізнес-правила, незалежні від конкретного застосунку і майже ніколи не змінюються, якщо не потрібно оновлення цих бізнес-правил. Чим вище розташований шар, тим більш абстрактним і загальним є його функціонал, тоді як чим нижче - тим більш специфічним і часто змінним.

Однією з численних переваг чистої архітектури є ізоляція шарів. Це означає, що зміни в одному шарі не впливають на інші шари. Наприклад, коли у випадку розробки використовувалась нереляційна база даних, яка стала недоліком, перехід на PostgreSQL відбувся без жодних змін у шарах контролерів, юз кейсів та сутностей, завдяки використанню чистої архітектури.

На рисунку 6 зображена структура проекту ScheduleSystem, де директорії Application, Controllers, Infrastructure та Domain відповідають різним шарам бізнес-логіки застосунку, адаптерів інтерфейсів, фреймворків і драйверів, а також загальній бізнес-логіці застосунку відповідно.

Як зазначено у розділі 2, для бази даних було використано PostgreSQL. Структура бази даних представлена на рисунку 7.

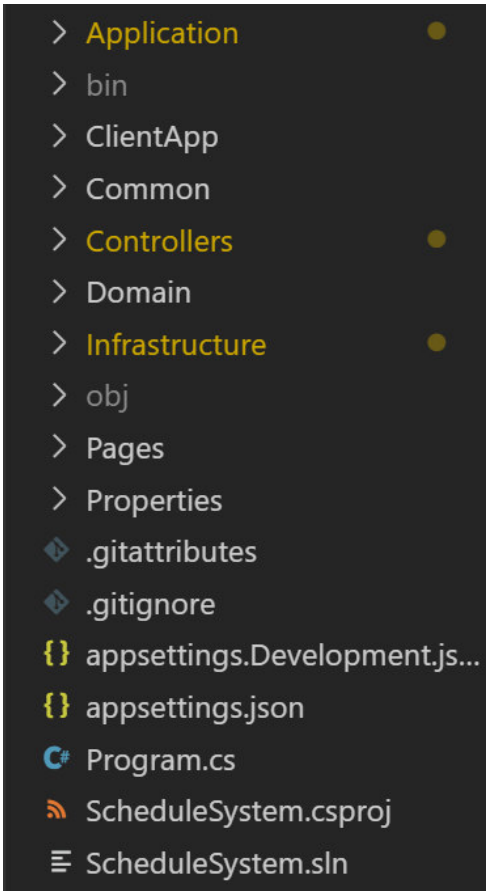

Рис 6. Структура папок в проекті застосунку ScheduleSystem

Таблиця InputData містить вхідні дані для розкладу, який потрібно згенерувати. Для зручності, InputData має поле з назвою (Name). Кожному запису в InputData відповідає один або декілька об'єктів з таблиці Lessons. Кожен об'єкт з Lessons представляє вхідні дані для одного заняття і містить інформацію про академічну групу (таблиця Group), викладача (таблиця Teacher), предмет та аудиторію. Після генерації

розкладу, буде створено новий запис у таблиці Schedule, а також для кожного об'єкту Lessons буде встановлено відповідний запис з таблиці LessonTime, який визначає день тижня та номер пари для проведення заняття. Таким чином, з одних і тих самих вхідних даних можна згенерувати декілька розкладів. Таблиці User та Role використовуються для автентифікації та авторизації користувачів.

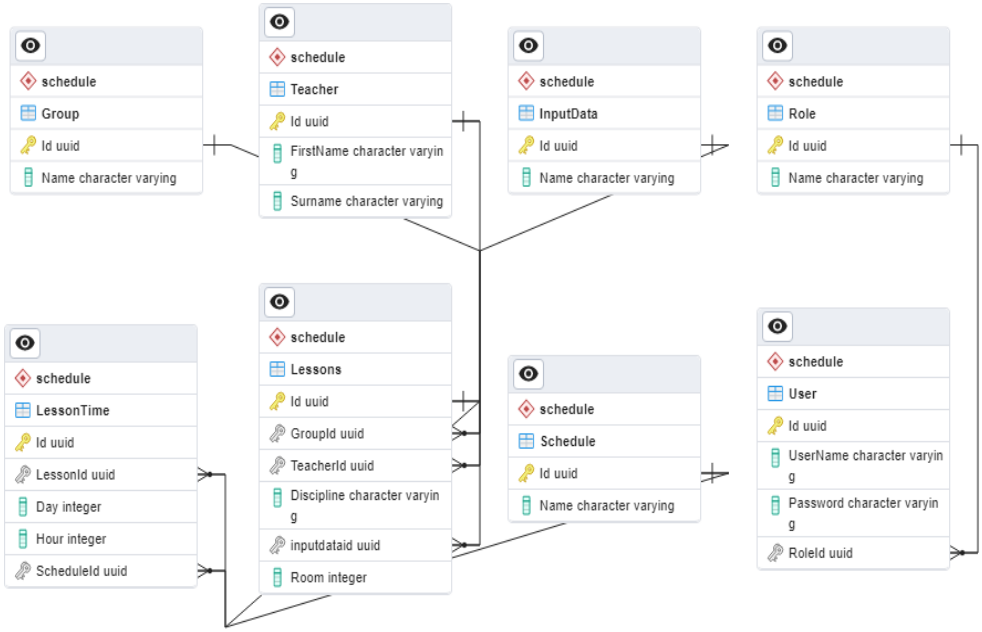

Рис 7. Діаграма відношень бази даних

У базі даних використано зовнішні ключі для зв'язку між батьківськими та дочірніми таблицями з метою підтримки каскадного видалення. Це означає, що якщо об'єкт у батьківській таблиці був видалений, то автоматично видаляться всі зв'язані з ним об'єкти у дочірніх таблицях.

3.2 Реалізація клієнтської частини

Клієнтська частина застосунку базується на односторінковому додатку (SPA) з використанням React. Застосунок має різні шляхи доступу. Шлях /scheduleinputdata (вкладка "Вхідні дані" на панелі навігації, як показано на рис. 8) містить таблицю для керування вхідними даними розкладів. При натисканні на один із рядків таблиці, користувач переходить на екран /scheduleinputdata/:id, де може змінювати об'єкти вхідних даних, такі як група, викладач, аудиторія та предмет.

Після введення необхідних вхідних даних на цьому ж екрані, користувач може згенерувати розклад, натиснувши кнопку "Згенерувати розклад". Доступні також шляхи /group (вкладка "Групи" на рис. 8) та /teacher (вкладка "Викладачі" на рис. 8) для керування академічними групами та викладачами відповідно. Шлях /settings (вкладка "Налаштування" на рис. 8) дозволяє користувачу налаштувати параметри генетичного алгоритму для генерації розкладу. Перейшовши на екран /schedule (вкладка "Згенеровані розклади" на рис. 8), користувач може переглянути список раніше згенерованих розкладів і вибрати конкретний розклад для перегляду, натиснувши на нього у списку. Після цього відбудеться перехід на екран /schedule/:id, де користувач може переглянути розклад занять на конкретний день.

Всі вищезазначені шляхи доступні користувачам з роллю Admin. Користувач із роллю User має доступ лише до екрану /schedule, де він може переглядати поточний розклад. Адміністратор (користувач із роллю Admin) може встановлювати, який із згенерованих розкладів є поточним.

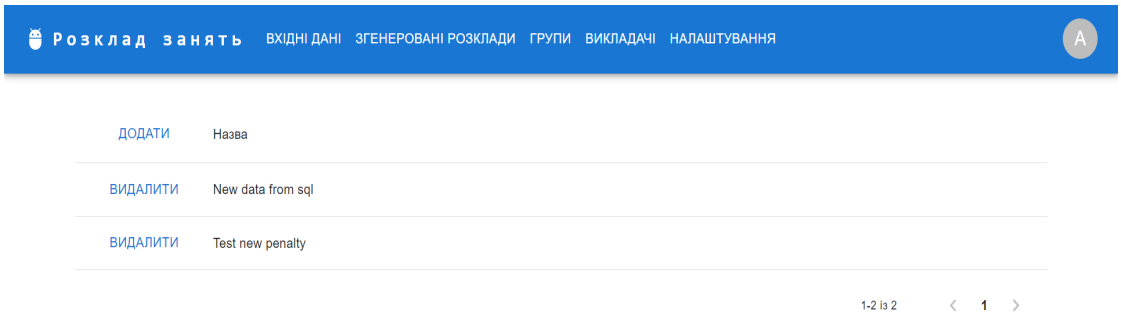

Рис 8. Загальний вигляд клієнтської частини

Для здійснення HTTP-запитів з клієнта на сервер використовується бібліотека Axios. Axios є HTTP клієнтом для node.js та браузера і має корисні можливості, такі як перехоплення запитів за допомогою перехоплювачів. В даному застосунку було застосовано перехоплювач запиту для підстановки заголовку авторизації, а також перехоплювач відповіді для перенаправлення користувача на сторінку логіну, якщо сервер поверне статус-код 401.

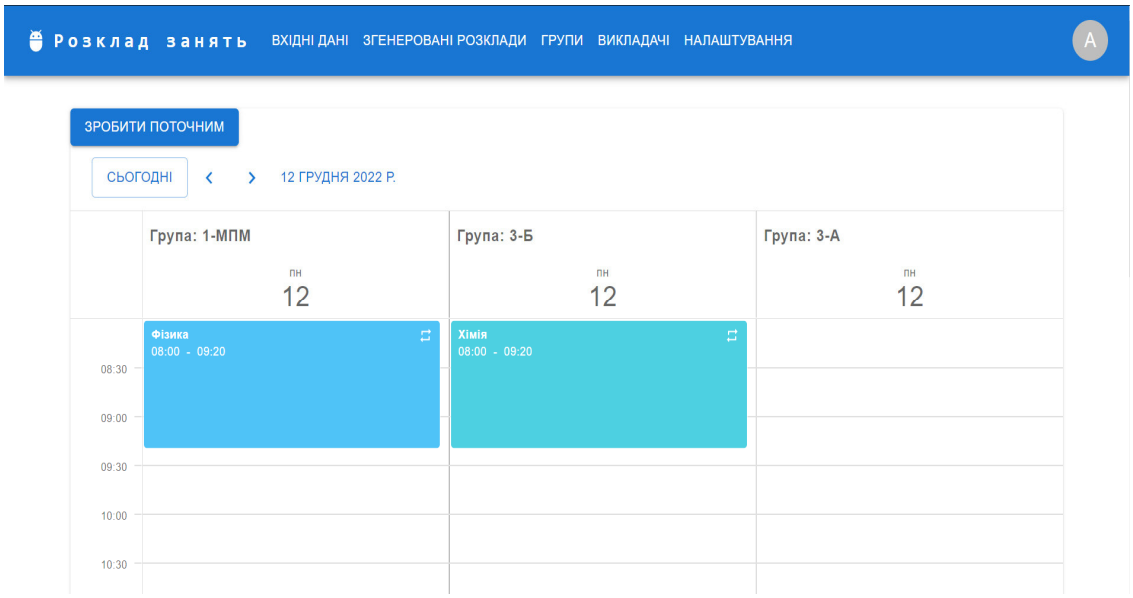

Рис 9. Загальний вигляд екрану /schedule/:id, для користувача із роллю Admin

Для відображення згенерованого розкладу використовувався готовий компонент DevExtreme React Scheduler. Цей компонент дозволяє відображати дані планувальника

та забезпечує можливість керування розкладом користувачем. Залежно від потреб користувача, компонент може відображати розклад поденно, потижнево або помісячно [12]. Загальний вигляд екрану з розкладом можна побачити на рис. 9.

На рисунку 9 видно, що розклад на день включає час початку та час закінчення заняття, а також академічну групу, для якої призначено дане заняття. Користувач може вибирати дату, на якій потрібно переглянути розклад. При натисканні на картку заняття з'являється додаткова інформація про аудиторію та викладача, який проводить це заняття.

Цей готовий компонент спрощує відображення розкладу та забезпечує зручні можливості для його перегляду та інтерактивної роботи з ним користувачем.

Висновки

У статті було успішно реалізовано кілька важливих аспектів, спрямованих на вдосконалення генетичного алгоритму для автоматизованої генерації розкладу навчальних занять. Окрім цього, було розроблено веб-орієнтований додаток, який дозволяє зручно керувати та адмініструвати розклад занять. Нарешті, були проаналізовані існуючі рішення для подібних проблем.

Основні результати, отримані під час проведення дослідження, можна узагальнити таким чином:

1. Було розроблено веб-додаток, що забезпечує зручне керування та автоматизовану генерацію розкладу навчальних занять.

2. Генетичний алгоритм генерації розкладу було удосконалено, що дозволило здійснювати більш точну та ефективну оптимізацію розкладу навчальних занять.

Ці досягнення представляють значний прогрес у вирішенні проблеми генерації розкладу та розширюють можливості для ефективного управління навчальним процесом.

Список використаної літератури:

1. Whitley, D. (1994). A genetic algorithm tutorial. Statistics and computing, 4(2), 65-85.
2. Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary Algorithms and Neural Networks: Theory and Applications, 43-55.
3. Troelsen, A., & Japikse, P. (2017). Pro C# 7: With. net and. net Core (Vol. 1328). Apress.
4. Рейтинг мов програмування 2022 [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/language-rating-2022/.
5. Language Integrated Query (LINQ) (C#) [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq.
6. Obe, R. O., & Hsu, L. S. (2017). PostgreSQL: up and running: a practical guide to the advanced open source database. " O'Reilly Media, Inc.".
7. Boduch, A., & Derks, R. (2020). React and React Native: A complete hands-on guide to modern web and mobile development with React. js. Packt Publishing Ltd.
8. Boduch, A. (2019). React Material-UI Cookbook: Build captivating user experiences using React and Material-UI. Packt Publishing Ltd.
9. Masse, M. (2011). REST API design rulebook: designing consistent RESTful web service interfaces. " O'Reilly Media, Inc.".
10. Jones, M., Bradley, J., & Sakimura, N. (2015). Json web token (jwt) (No. rfc7519).
11. Martin, R. C., Grenning, J., Brown, S., Henney, K., & Gorman, J. (2018). Clean architecture: a craftsman's guide to software structure and design (No. s 31, pp. 57-91). Prentice Hall.
12. React Scheduler - Getting Started. [Електронний ресурс]. – Режим доступу: https://devexpress.github.io/devextreme-reactive/react/scheduler/docs/guides/getting-started/.

References:

1. Whitley, D. (1994). A genetic algorithm tutorial. Statistics and computing, 4(2), 65-85.
2. Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary Algorithms and Neural Networks: Theory and Applications, 43-55.
3. Troelsen, A., & Japikse, P. (2017). Pro C# 7: With. net and. net Core (Vol. 1328). Apress.

4. Reitynh mov prohramuvannia 2022 [Programming Language Rankings 2022]. (2022). dou.ua. Retrieved from https://dou.ua/lenta/articles/language-rating-2022/ [in Ukrainian]
5. Language Integrated Query (LINQ) (C#). learn.microsoft.com Retrieved from https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq.
6. Obe, R. O., & Hsu, L. S. (2017). PostgreSQL: up and running: a practical guide to the advanced open source database. " O'Reilly Media, Inc.".
7. Boduch, A., & Derks, R. (2020). React and React Native: A complete hands-on guide to modern web and mobile development with React. js. Packt Publishing Ltd.
8. Boduch, A. (2019). React Material-UI Cookbook: Build captivating user experiences using React and Material-UI. Packt Publishing Ltd.
9. Masse, M. (2011). REST API design rulebook: designing consistent RESTful web service interfaces. " O'Reilly Media, Inc.".
10. Jones, M., Bradley, J., & Sakimura, N. (2015). Json web token (jwt) (No. rfc7519).
11. Martin, R. C., Grenning, J., Brown, S., Henney, K., & Gorman, J. (2018). Clean architecture: a craftsman's guide to software structure and design (No. s 31, pp. 57-91). Prentice Hall.
12. React Scheduler - Getting Started. devexpress.github.io Retrieved from https://devexpress.github.io/devextreme-reactive/react/scheduler/docs/guides/getting-started/.

HURIEV Yaroslav,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF A WEB-BASED SCHEDULING MANAGEMENT TOOL

Summary. Introduction. *Modern educational institutions face a global problem of managing and automating the creation of class schedules. Currently, the number of students and teachers is rapidly increasing, which leads to an escalation in efforts required to create a schedule that satisfies all participants in the educational process and avoids conflicts. Situations where one teacher is assigned to classes in two different classrooms simultaneously are not uncommon, resulting in haste and adjustments that can cause issues during the actual classes. Therefore, the future of schedule management lies in web applications that help schedule creators save time and avoid critical errors such as the one mentioned. Such web applications will not only be a convenient tool for planning the educational process but also contribute to its efficiency and organization, providing a user-friendly and optimal schedule for all participants..*

Purpose. *The aim of the article is to analyze similar application, develop an architecture and create a web-based tool for managing class schedules.*

Results. *A web application has been developed, providing convenient management and automated generation of class schedules. Also, the genetic algorithm for schedule generation has been improved, enabling more accurate and efficient optimization of class schedules. These achievements represent significant progress in solving the scheduling problem and expand the possibilities for effective management of the educational process.*

Conclusion. *The article successfully implemented several important aspects aimed at improving the genetic algorithm for automated generation of class schedules. In addition, a web-based application was developed, which allows convenient management and administration of class schedules. Finally, existing solutions for similar problems were analyzed.*

Keywords: *schedule, automated schedule generation, genetic algorithm, web application.*

Одержано редакцією 17.12.2021 р.
Прийнято до публікації 23.06.2022 р.