Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота бакалавра

з теми: **«Автоматизована система «єРозклад»»**

Виконав: здобувач вищої освіти групи KNms1-B21
спеціальності 122 Комп’ютерні науки
Шевчук Олександр Володимирович

Керівник: Пилипюк Тетяна Михайлівна, кандидат фізико-математичних наук, доцент

Рецензент: Семенишина Ірина Віталіївна, кандидат фізико-математичних наук, доцент кафедри інформаційних технологій, фізико-математичних та безпекових дисциплін Закладу вищої освіти «Подільський державний університет»

м. Кам’янець-Подільський – 2024 р.

ЗМІСТ

АНОТАЦІЯ

У даній роботі представлено теоретичні й практичні аспекти розробки автоматизованої системи керування розкладом навчальних занять для закладів вищої освіти I-II рівня акредитації. Дана система складається з десктопного застосунку для адміністрації освітнього закладу, який відповідає за створення розкладу занять, управління розкладом та інформування про розклад, а також мобільного застосунку для здобувачів вищої освіти й викладачів, щоб переглядати та отримувати актуальний розклад занять.

Метою роботи є полегшення процесу керування розкладом навчальних занять в ЗВО I-II рівня акредитації за допомогою розробленої автоматизованої системи.

Під час створення системи використовувалися мови програмування C#, Java, PHP та системи керування базами даних SQLite і MySQL. Дані мови програмування використовували інтегровані середовища розробки Visual Studio Community, Android Studio та Visual Studio Code.

Крім того, було проаналізовано генетичний алгоритм та його застосування до формування розкладу на основі вхідних даних.

Ключові слова: розклад навчальних занять, управління розкладом, інформування про розклад занять, генетичний алгоритм, формування розкладу, автоматизована система.

ABSTRACT

The work is devoted to the theoretical and practical aspects of the automated system design for managing the class schedule for higher education institutions of the I-II level of accreditation. This system consists of a desktop application for educational institution administration, which is responsible for creating a class schedule, managing the schedule, informing about the schedule and a mobile application for higher education students and teachers to view and receive up-to-date class schedule.

The purpose of the work is to facilitate the process of managing the schedule in higher education institutions of the I-II level of accreditation using the developed automated system.

Programming languages C#, Java, PHP and database management systems SQLite and MySQL were used during the system creation. These programming languages used integrated development environments Visual Studio Community, Android Studio and Visual Studio Code.

In addition, the genetic algorithm and its application to the schedule formation based on the input data were analyzed.

Key words: class schedule, schedule management, informing about the class schedule, genetic algorithm, class schedule formation, automated system.

ВСТУП

Розклад навчальних занять є невід’ємною частиною освітнього процесу, оскільки він допомагає організувати процес навчання та забезпечувати раціональний розподіл навчального часу. Сам розклад визначає час та місце проведення занять орієнтованих на певні їх види, таких як лекції, семінарські, практичні та лабораторні заняття. Крім того, в ньому зазначаються назви навчальних дисциплін, інформація про викладачів, найменування академічних груп тощо. Розклад занять дозволяє студентам заздалегідь планувати свій час, враховуючи не тільки час на навчання, але й на інші зобов’язання та власні інтереси. Також він допомагає викладачам організувати власну роботу, розподіливши навчальні матеріали відповідно до виділеного часу. Це дозволяє їм ефективно планувати заняття та працювати зі студентами у найпродуктивніший спосіб. Ще розклад навчальних занять сприяє оптимальному використанню ресурсів освітнього закладу, таких як аудиторії, лабораторії та обладнання. Він запобігає перевантаженню занять і забезпечує рівномірне розподілення навчального навантаження протягом дня. Враховуючи викладене, слід зазначити, що управління розкладом навчальних занять є доволі клопітким та водночас важливим процесом в освітніх закладах.

В системі вищої освіти України встановлено чотири рівні акредитації навчальних закладів, при цьому найважча ситуація з управлінням розкладом навчальних занять насамперед, спостерігається в закладах, які знаходяться на I та II рівнях акредитації. У цих закладах прослідковується тенденція до простого та неефективного управління розкладом занять. Зазвичай розклад створюється автоматизовано за допомогою певного програмного продукту, але інформування та коригування відбуваються в електронних таблицях, які потім демонструються викладачам та студентам у графічному або електронному форматі. Така практика часто призводить до незручностей та недоліків у плануванні та змінах розкладу занять, оскільки викладачам і студентам при зміні розкладу потрібно фотографувати даний розклад на

інформаційних стендах закладу та/або завантажувати його в електронному форматі з сайту навчального закладу для користування.

Актуальність роботи. Автоматизація ефективного процесу управління розкладом навчальних занять в освітніх закладах є важливою та актуальною задачею дослідження. Дане дослідження зводиться до необхідності вирішення проблеми управління розкладом занять у закладах вищої освіти I та II рівня акредитації, які часто мають обмежений бюджет залученості до новітніх технологій та програмних продуктів. Для цих закладів важливо мати доступ до ефективного інструменту, який допоможе їм створювати, автоматизовано формувати, здійснювати інформування про розклад й керування розкладом занять, що призводить до оптимального розподілу ресурсів та забезпечує належну організацію навчального процесу без додаткових фінансових навантажень.

Об’єктом дослідження є автоматизація розкладу навчальних занять, а **предметом** – формування розкладу знать та інформування про розклад через автоматизовану систему для ЗВО I-II рівня акредитації.

Мета роботи полягає в полегшенні процесу керування розкладом навчальних занять в ЗВО I-II рівня акредитації за допомогою розробленої автоматизованої системи.

До основних **завдань роботи** можна віднести:

- аналіз існуючих рішень;
- визначення алгоритму формування розкладу;
- проєктування реляційної бази даних для зберігання та організації інформації пов’язаної з розкладом навчальних занять;
- розробка десктопного застосунку для створення, керування, друку розкладу;
- розробка мобільного застосунку для перегляду та інформування про зміни в розкладі для викладачів й здобувачів вищої освіти;
- реалізація серверної частини для обміну даними між застосунками.

Практичне значення. Кваліфікаційна робота бакалавра полягає у вирішенні проблеми з управління розкладом навчальних занять в ЗВО I-II рівня акредитації, забезпечуючи оптимізацію процесу формування розкладу та своєчасне інформування про зміни в розкладі, задовольняючи потреби всіх зацікавлених сторін.

Апробація результатів. Результати досліджень були оприлюднені на науковій конференції здобувачів вищої освіти за підсумками НДР у 2023-2024 навчальному році 9-10 квітня 2024 року м. Кам’янець-Подільський [17] та на ІІ міжнародній науково-практичній інтернет-конференції «Актуальні аспекти розвитку STEAM-освіти в умовах євроінтеграції» 26 квітня 2024 року м. Кропивницький [9].

Структура роботи. Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД РІШЕНЬ

1.1 Змістовна постановка задачі

Для спрощення керування розкладом навчальних занять необхідно розділити автоматизовану систему на два окремих програмних модулі. Перший модуль буде здійснювати безпосереднє управління розкладом, а другий – задовольняти потреби здобувачів освіти та викладачів щодо інформування про розклад занять. Варто зазначити, що кожен модуль даної системи є окремим програмним продуктом, який використовується на різних платформах. Отже, початковим кроком до процесу керування розкладом є додання базових даних до системи, а це:

- загальна інформація про заклад;
- галузі знань та спеціальності;
- предмети;
- викладачі;
- академічні групи;
- аудиторії.

Наступним кроком є створення поточного навчального процесу, тобто введення даних про тривалість навчального року, номер семестру, дати початку та завершення семестру. Крім того, необхідно вказати розклад дзвінків, який буде використовуватися в поточному семестрі. Далі потрібно створити журнал занять – поєднати предмети з викладачами та академічними групами, а також зазначити загальну кількість занять на поточний семестр і кількість занять визначених на тиждень.

Після виконання цих кроків, освітній заклад може перейти до створення розкладу навчальних занять. При цьому, перед процесом створення розкладу, адміністрація закладу повинна ще додати до системи графік навчального процесу для кожної академічної групи, вказуючи дати, протягом яких не буде проводитися навчання (через практику, екзамени, дипломне проєктування, державну атестацію тощо), даний графік вказується у скасованих заняттях.

Система забезпечує процес автоматизованого формування розкладу на основі введених даних, а також надає можливість створювати розклад власноруч та вносити корективи при формуванні. Після створення розкладу, його можна експортувати в Microsoft Excel для подальшого друку та розміщення на інформаційних стендах навчального закладу. Окрім друку, адміністрація закладу може опублікувати розклад занять, а також керувати ним, вносячи зміни, здійснюючи моніторинг аудиторного фонду та працю викладачів в певні дати з вказаними парами (номерами занять) тощо. Оскільки навчання здійснюється у 2 семестри, то розклад, журнал та ін. створюється окремо для кожного семестру.

На відміну від першого програмного продукту, який призначений для адміністрації освітнього закладу, другий – призначений для викладачів й студентів, щоб швидко та зручно здійснити перегляд поточного та майбутнього (запланованого) розкладу. Оскільки він розрахований на мобільні пристрої, користувач після проходження авторизації може в один клік отримувати розклад після відкриття застосунку. Після публікації розкладу занять, мобільний застосунок автоматично у фоновому режимі оновить розклад та сповістить про це користувача. Крім перегляду свого розкладу, користувач зможе переглянути наступну інформацію:

- інформацію про освітній заклад;
- виконані та заплановані заміни в розкладі;
- пошук академічної групи та/або викладача для перегляду його розкладу в обраний день (тиждень);
- перегляд вільних аудиторій в обраний день зі вказаним номером пари;
- поділитися розкладом та ін.

Важливо додати, що усі зазначені програмні продукти мають виконувати свою роботу у відносно автономному режимі, до поки не потрібно буде здійснити певні запити до сервера (наприклад, авторизація в системі, опублікування розкладу, оновлення певних даних). Тому основна інформація,

яка використовується у двох програмних модулях, буде зберігатися у локальній базі даних, що забезпечить не тільки швидкий доступ до необхідних даних, але й спростить процес резервного копіювання та відновлення даних, що значно зменшить навантаження на сервер.

1.2 Огляд існуючих рішень

Розглянемо популярні програмні рішення серед управління та формування розкладу занять для ЗВО різних рівнів акредитації.

Програмний комплекс «Автоматизована система управління навчальним закладом» (АСУ «МКР») [11] являє собою інтегровану сукупність програм, що забезпечують ефективне управління університетом в єдиному інформаційному просторі. До складу системи входять модулі, що працюють в середовищі Windows (наприклад, навчальний модуль, деканат, абітурієнт, методичний відділ, відділ кадрів тощо), а також вебпортал, що забезпечує доступ до розкладу занять, успішності, навчальних планів, нарахувань та контролю оплат за навчання та проживання в гуртожитку, тестування студентів, запис на дисципліни та ін. Вся інформація зберігається в єдиній спільній базі даних. Особливістю програмного комплексу є наявність інструментів для самостійного створення різних друкованих та статистичних форм, що робить систему майже незалежною від розробників. Крім того, комплекс дозволяє створювати та враховувати індивідуальні траєкторії навчання студентів, включаючи дистанційне навчання через Інтернет [11].

Автоматизована система «Деканат» [2] – інтегрований програмний комплекс, який є частиною АСУ «ВНЗ» та призначений для покращення управління навчальним процесом в освітніх закладах. Завдяки цій системі, методистам стає легше організовувати свою роботу та зменшується обсяг паперової документації. Серед можливостей, яких надає дана система в навчальному процесі є наступне: розробка навчальних та робочих планів на навчальний рік, закріплення навчальних груп за планами, генерація та відновлення робочих навчальних планів за навчальними планами, закріплення

контингенту для навантаження по робочих навчальних планах, навантаження кафедри (агрегація та розподілення викладачами), індивідуальний робочий план викладача кафедри, а також створення та використання web-розкладу. Web-розклад, який створюється за допомогою АС «Деканат», може бути вбудований на власні вебсторінки закладу. Сама система має оптимізацію для роботи на сайтах з високим трафіком (великим відвідуванням користувачів), забезпечуючи студентам та співробітникам закладу постійний доступ до своїх розкладів [2].

Пакет програм «Деканат» [10] – автоматизована система управління вищим навчальним закладом, яка призначена для організації та підтримки навчального процесу в вищих навчальних закладах України I-IV рівнів акредитації, розроблена ПП «Політек-СОФТ». Головною метою системи є зменшення часу, що витрачають працівники вищих навчальних закладів на виконання щоденних завдань та спрощення роботи з даними. Система базується на клієнт-серверній технології, що дозволяє встановлювати її на декілька комп’ютерів, об’єднаних в локальну мережу, для роботи з єдиною базою даних. Завдяки використанню додаткових вебсценаріїв, доступ до бази даних можливий з будь-якого місця через Інтернет. Для управління базами даних використовується FireBird. До роботи з системою можуть бути залучені як окремі працівники освітнього закладу (навчальна частина, секретарі деканатів та кафедр), так і всі учасники навчального процесу (викладачі та студенти). Також система включає в себе інтуїтивно зрозумілий інструмент для створення звітів, який підтримує мову HTML. Завдяки цьому інструменту, користувачі можуть створювати нові звіти або редагувати вже існуючі. Крім того, звіти, створені за допомогою системи, можуть бути переглянуті та відредаговані в популярних програмах, таких як MS Word та MS Excel, перед тим як бути надрукованими [10].

Висновок до розділу 1

Детально зазначено постановку задачі для досягнення мети кваліфікаційної робити. Вказано на важливість поділу автоматизованої системи на два окремих програмних модулі та зазначено функціональність даних модулів. Також оглянуто популярні програмні рішення для управління розкладом та формування розкладу навчальних занять для закладів вищої освіти.

РОЗДІЛ 2
КОМП’ЮТЕРНА СИСТЕМА

2.1 Технічні характеристики та системні вимоги до персонального комп’ютера

Для роботи з десктопним застосунком повинно використовуватись ПК з такими рекомендованими системними характеристиками:

- Операцій система (ОС): Windows 8.1;
- Тип ОС: 64 біт;
- Частота процесора: 2 ГГц;
- Кількість ядер процесора: 2;
- Обсяг оперативної пам’яті: 4 Гб;
- Обсяг накопичувача: 45 Гб;
- Дисплей: 1920x1080;
- Контролер: миша і клавіатура;
- Необхідне ПЗ: .NET Framework 4.8.

Для роботи з мобільним застосунком повинно використовуватись ПК з такими рекомендованими системними характеристиками:

- ОС: Android 8;
- Тип ОС: 64 біт (arm64-v8a);
- Частота процесора: 1,5 ГГц;
- Кількість ядер процесора: 4;
- Обсяг оперативної пам’яті: 4 Гб;
- Обсяг накопичувача: 16 Гб;
- Діагональ екрана: 6,3 дюйма.

Технічні характеристики ПК програміста, за яким розроблювалася автоматизована система:

- Ноутбук: Acer Nitro 5 AN515-54-58QC;
- ОС: Windows 10;
- Тип ОС: 64 біт;

- Процесор: Intel Core i5 9300H;
- Частота процесора: 2,4-4,1 ГГц;
- Кількість ядер процесора: 4;
- Дискретна відеокарта: Nvidia GeForce GTX 1650;
- Обсяг дискретної відеопам’яті: 4 Гб;
- Діагональ екрана: 15,6 дюйма;
- Роздільна здатність: 1920x1080;
- Обсяг оперативної пам’яті: 16 Гб;
- Обсяг SSD накопичувачів: 1024 Гб.

2.2 Вибір операційної системи та програмних засобів

Для того, щоб обрати правильну операційну систему, яка буде мати попитом серед користувачів, можна скористатися статистикою щодо розподілу частки ринку операційних систем, яку надає Statcounter Global Stats. Завдяки їй можемо дізнатися, які операційні системи найчастіше використовують користувачі в Україні, оскільки популярність операційної системи може вказувати на її якість, надійність і підтримку серед користувачів.

Оскільки АС «єРозклад» складається з двох програмних продуктів, які розміщуються на різних ОС, скористаємося для обрання платформи під десктопний застосунок статистикою на рисунку 2.1. З даної статистики видно, що протягом січня 2023 – лютого 2024 року найбільш популярною ОС для настійного ринку була Windows (80,11%), отже розглянемо кілька переваг цієї системи:

1. Зручний інтерфейс користувача. Він відомий своїм інтуїтивно зрозумілим та легким у використанні інтерфейсом, який надає зручність та доступність для користувачів будь-якого рівня досвіду.

2. Широкий вибір програм. Windows має велику базу програм та ігор, які підтримуються даною операційну систему. Вона є стандартом для багатьох

розробників програмного забезпечення, що робить її сумісною з багатьма застосунками.

3. Підтримка та оновлення. Microsoft, розробник Windows, надає постійну підтримку та оновлення для своєї ОС. Це включає виправлення помилок, покращення безпеки та випуск нових функцій, що забезпечує актуальність та надійність системи.

4. Сумісність з обладнанням. Windows підтримує широкий спектр обладнання, що дозволяє використовувати різноманітні пристрої, від комп'ютерів до ноутбуків, планшетів та інших пристроїв.

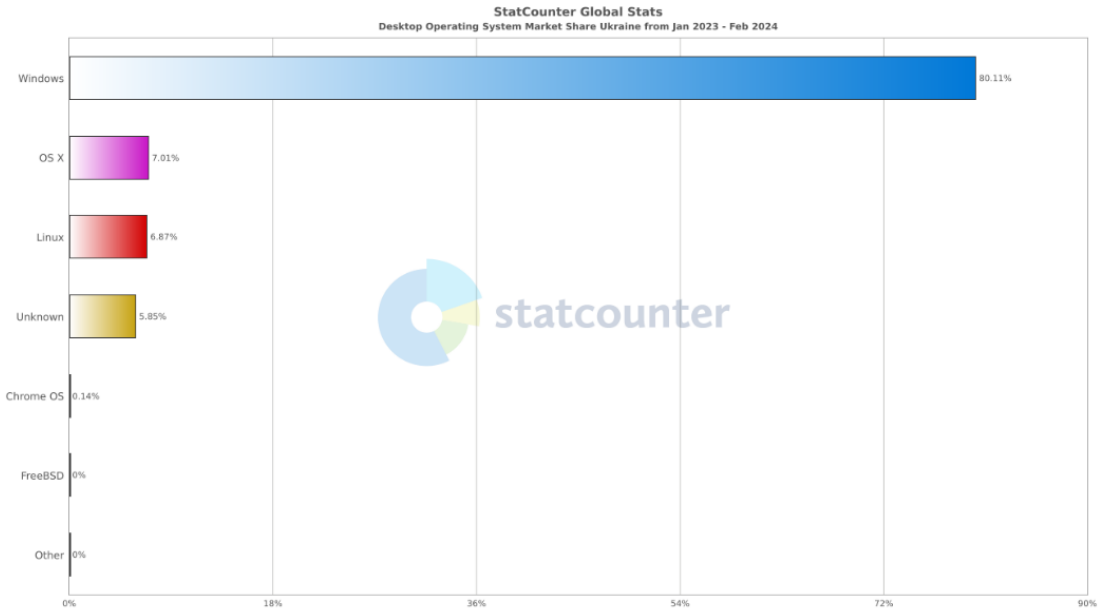


Рис. 2.1. Частка ринку настільних операційних систем в Україні [20]

Для обрання платформи під мобільний застосунок скористаємося статистикою на рисунку 2.2. Згідно з даної статистики, найпопулярнішою ОС для мобільного ринку протягом січня 2023 – лютого 2024 року був Android (69,77%), тож огляньмо декілька переваг цієї системи:

1. Відкритість та гнучкість. Android є відкритою платформою, що дозволяє розробникам використовувати різні інструменти та мови програмування для створення застосунків. Крім того, платформа підтримує широкий спектр пристроїв, від бюджетних до флагманських моделей, що дає можливість розробникам створювати застосунки для різних цільових аудиторій.

2. Багатозадачність. Смартфон має можливість виконувати багато завдань одночасно. Наприклад, можна легко переглядати вебсторінку та одночасно розділивши екран записувати до нотаток цитати з даної сторінки.

3. Легкість в налаштуванні. ОС дозволяє налаштувати різні аспекти пристрою, включаючи вигляд домашнього екрана, шрифти, клавіатуру та багато іншого. Користувач може персоналізувати свій пристрій так, щоб він відповідав його стилю та вподобанням.

4. Різні канали розповсюдження та продажів. Завдяки цій платформі розробники не обмежені лише Android Market (магазином застосунків від Google), вони можуть створювати власні магазини, використовувати сторонні маркетплейси для розповсюдження та продажу своїх застосунків або безпосередньо розмістити їх на своєму вебсайті для завантаження.

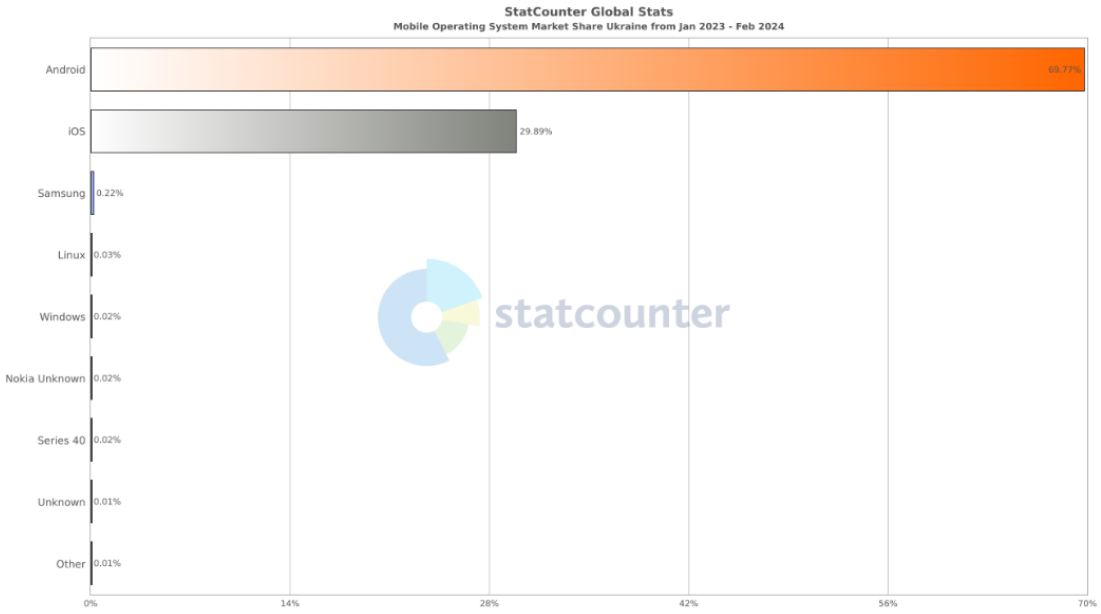


Рис. 2.2. Частка ринку мобільних операційних систем в Україні [24]

Для виконання поставлених завдань щодо розробки та функціювання автоматизованої системи, будуть використовуватися наступні набори програмних засобів.

Visual Studio Community – безкоштовне інтегроване середовище розробки від Microsoft, призначений для відкритих проєктів, малих команд і індивідуальних розробників. Дане середовище пропонує розробникам набір інструментів, які дозволяють створювати програми для платформ, таких як

Windows, Linux, macOS, iOS та Android. Багато мов програмування, включаючи C++, C#, Visual Basic, Python тощо, підтримуються в цьому потужному середовищі. Крім того, завдяки вбудованим інструментам для налагодження, тестування та контролю зміну коду, розробники можуть створювати високоякісні продукти.

Visual Studio Code – простий і універсальний редактор коду з відкритим вихідним кодом розроблений компанією Microsoft. Він має вбудовані функції, які включають навігацію по коду, інтелектуальне завершення коду та підсвічування синтаксису, а також підтримує багато мов програмування. Вбудований термінал дозволяє розробникам виконувати команди безпосередньо з редактора. Крім того, у редакторі є безліч розширень, які можна встановити, щоб додати додаткові функції та інструменти. Також VS Code підтримує Git та інші системи контролю версій, що полегшує співпрацю та управління кодом.

Android Studio є інтегрованим середовищем розробки розробленим компанією Google. Він призначений для створення програм для пристроїв на базі Android, використовуючи мови програмування Java та Kotlin. Android Studio базується на платформі IntelliJ IDEA, яка відома своїми потужними інструментами для аналізу коду та рефакторингу. Також середовище містить візуальний редактор інтерфейсу, емулятор різних пристроїв Android та інструменти для оптимізації продуктивності програм. Крім того, Android Studio має вбудовану підтримку Git та інтеграцію з Google Cloud Platform для розгортання застосунків у хмарі.

FontAwesome.Sharp є бібліотекою іконок з відкритим вихідним кодом розробленим для платформи .NET, що дозволяє розробникам легко додавати значки до своїх програм. Вона надає легкий спосіб інтеграції широкого діапазону іконок FontAwesome у ваші проєкти, забезпечуючи єдиний та узгоджений візуальний інтерфейс. Дозволяє використовувати власні шрифти, а також має вбудовану підтримку зміни кольору кожного значка, розміру та його стилю.

System.Data.SQLite.Core є бібліотекою ADO.NET для взаємодії з базами даних SQLite. Вона надає простий та ефективний спосіб роботи з базою даних в програмах .NET. Бібліотека повністю сумісна з ADO.NET 2.0 і підтримує такі функції, як транзакції, параметризовані запити, збережені процедури, тригери та індекси. Дана бібліотека має також вбудовану підтримку LINQ, що дозволяє використовувати мову запитів для роботи з базами даних. Крім того, бібліотека надає можливість працювати з базами даних SQLite в режимі «в пам'яті», що може бути корисним для тестування та тимчасового зберігання даних.

Newtonsoft.Json – популярна бібліотека для роботи з JSON у середовищі .NET. Вона надає прості та ефективні методи для читання, запису та маніпулювання даними JSON. Бібліотека підтримує конвертацію об'єктів .NET у формат JSON та навпаки, має розширені можливості для налаштування серіалізації та десеріалізації. Також підтримує широкі можливості налаштувань, включаючи форматування, обробку циклічних посилань та роботу LINQ для JSON.

ZXing Android Embedded є відкритою бібліотекою для читання штрих-кодів й QR-кодів в Android застосунках. Вона базується на популярній бібліотеці ZXing (Zebra Crossing) і дозволяє легко інтегрувати функціонал сканування без необхідності використовувати окремий сканер.

PHP QR Code – бібліотека з відкритим кодом (LGPL) для генерації QR-коду, а також двовимірного штрих-коду. Базується на бібліотеці libqrencode C, яка надає API для створення зображень штрих-кодів й QR-кодів формату png і jpeg. Бібліотека реалізована під PHP без зовнішніх залежностей.

Висновок до розділу 2

У розділі визначено системні вимоги до застосунків та характеристика ПК, за яким розроблялася АС «єРозклад». Зазначено ряд безкоштовних середовищ розробки та бібліотек, які імплементовані у застосунки для виконання поставлених задач кваліфікаційної робити.

РОЗДІЛ 3
РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1 Алгоритм формування розкладу

Кожен розклад формується випадковим чином дотримуючись певного алгоритму, враховуючи вхідні дані множини журналу (академічна група, викладач, предмет, загальна кількість занять на тиждень), аудиторії, номер заняття і дня тижня. Оскільки ЗВО I-II рівня акредитації використовують один сталий розклад, який використовується протягом семестру, алгоритм сформує розклад на тиждень, який з внесеними корективами (за потреби адміністрацією закладу) стане сталим на семестр. Усе це можна записати у вигляді такої моделі:

$$S_i = (W_i, L_i, C_i, J_i)$$

Тут W_i – день тижня, L_i – номер заняття, C_i – аудиторія, J_i – журнал (множина, яка об’єднує групу, предмет, викладача і загальну кількість занять на тиждень), S_i – розклад. На рисунку 3.1 зображено графічне подання розкладу даної моделі.

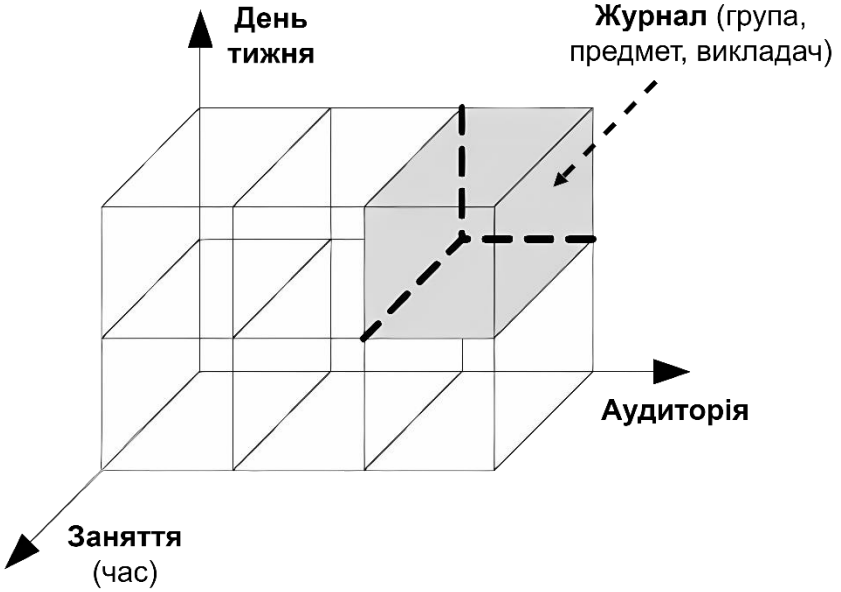


Рис. 3.1. Графічне подання розкладу

Кожна модель має свої властиві обмеження, які можна класифікувати на жорсткі та м’які. Жорсткі обмеження є незмінними та обов’язковими умовами, які повинні бути суворо дотримані в кожній моделі без винятків. М’які

обмеження є гнучкими та поступливими правилами, які можуть бути порушені в певних випадках, але лише за умови, що цей процес буде зведено до мінімуму. Тому сформулюємо жорсткі обмеження для формування розкладу:

- кожен викладач має проводити лише одне заняття для однієї конкретної групи в один проміжок часу;
- одна академічна група не може бути присутня на декількох різних заняттях, які проводяться одночасно;
- в одній аудиторії не дозволяється проводити більше ніж одне заняття в один проміжок часу;
- місткість аудиторії повинна бути достатньою для розміщення групи зі студентів, що відвідують заняття в ній.

Зазвичай для складання розкладу використовують еволюційні алгоритми. Серед них було обрано генетичний алгоритм. Він є методом оптимізації, який охоплює не лише випадковий перебір, а й оцінку та підбір найкращих рішень. Замість того, щоб генерувати випадкові набори даних, генетичний алгоритм створює кілька наборів даних (популяцій), з яких вибираються ті, що мають найкращі значення цільової функції. Далі вибрані набори даних (особини) використовуються для формування нових наборів даних шляхом схрещування та об’єднання частин даних (генів). Цей процес повторюється, допоки не буде досягнуто бажаного результату. Крім того, кожне нове покоління повинно мутувати, тобто вносити випадкові зміни до генів, щоб уникнути виродження популяції, коли наступне покоління може бути гіршим за попереднє. Для кращого розуміння на рисунку 3.2 подана візуалізація даного алгоритму у вигляді блок-схеми.

Варто додати, що алгоритм починає свою дію з формування першого покоління особин випадковим чином, після чого відбувається відбір декількох елітних особин. Існують декілька підходів відбору, серед них використовую турнірний відбір – вибираємо k-особин і проводимо турнір серед них шляхом обчислення та порівняння значень цільової функції для кожної особини, у підсумку лише найпридатніший кандидат серед цих відібраних особин

відбирається та передається наступному поколінню. Наступним кроком є схрещення, тобто створення нового покоління шляхом комбінування генів двох батьківських особин відібраних в попередньому етапі (підхід одноточкового схрещування). Наприкінці особини нового покоління з певною ймовірністю піддаються мутації, наприклад мутація взаємного обміну – випадково вибираємо два гени з особини та замінюємо їх місцями [15, с. 5; 16, с. 12]. Про необхідність мутації було описано вище. Далі популяція скорочується шляхом ініціалізації нової, в яку заносять лише найкращі популяції, розмір даної елітної популяції здамо у відсотковому еквіваленті від усіх наявних популяцій. Алгоритм закінчує своє виконання у двох випадках: досягнення максимальної кількості ітерацій поколінь або досягнення оптимального рішення (найкраще значення цільової функції, яке становить одиницю).

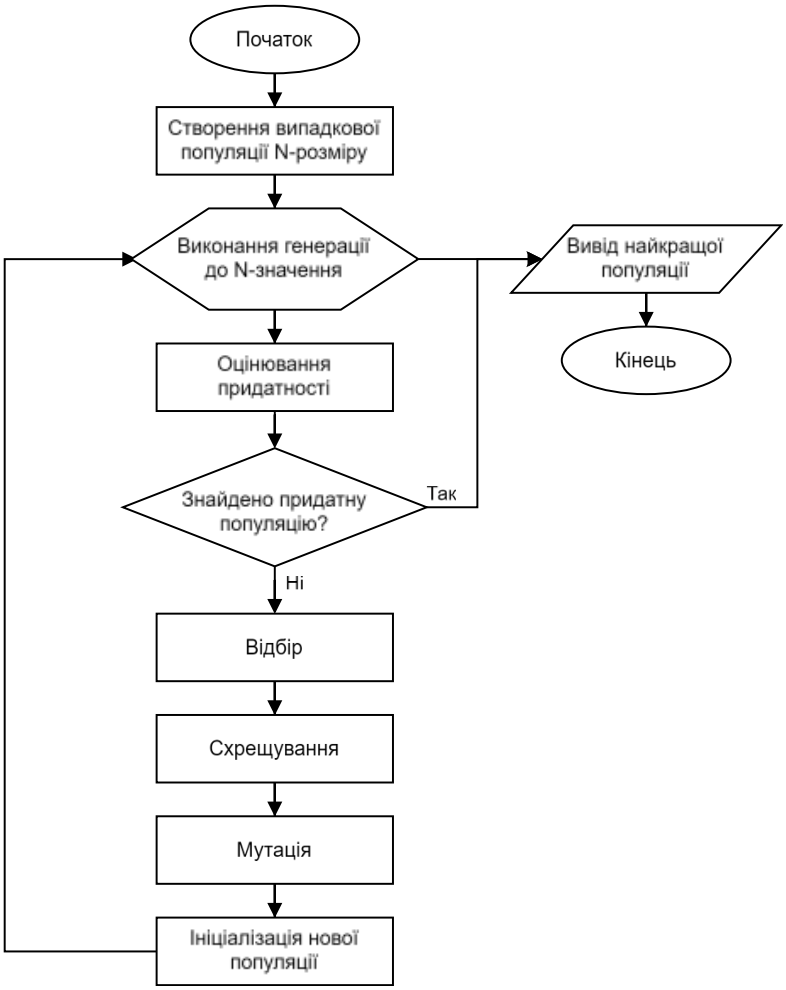


Рис. 3.2. Блок-схема генетичного алгоритму

3.2 Проєктування бази даних

Початковим кроком в проєктуванні бази даних є детальний аналіз предметної області, після чого здійснюється виділення сутностей та їхніх атрибутів, а також встановлення між ними відповідних зв’язків. Дані кроки визначають концептуальну модель, яку потрібно перетворити на логічну модель бази даних. При логічній моделі створюється реляційна схема, визначається структура таблиць, ключі, індекси та інші елементи, необхідні для зберігання та організації даних. Для перевірки коректності логічної моделі використовується нормалізація. Останнім етапом є безпосередня реалізація бази даних в конкретній системі управління базами даних [3; 5]. Головною базою даних в АС «єРозклад» є база даних модуля керування, оскільки вона зберігає всю необхідну інформацію для організації процесу керування розкладом навчальних занять. Загалом для даного модуля було спроєктовано 11 таблиць, кожна з яких містить відповідні ключі, атрибути та типи даних, адаптовані під цільову СУБД SQLite, в додатку А зазначено SQL-код, який створює ці таблиці з їх атрибутами й зв’язками між ними. На рисунку 3.3 подана реляційна схема бази даних, яка містить зв’язки між відношеннями.

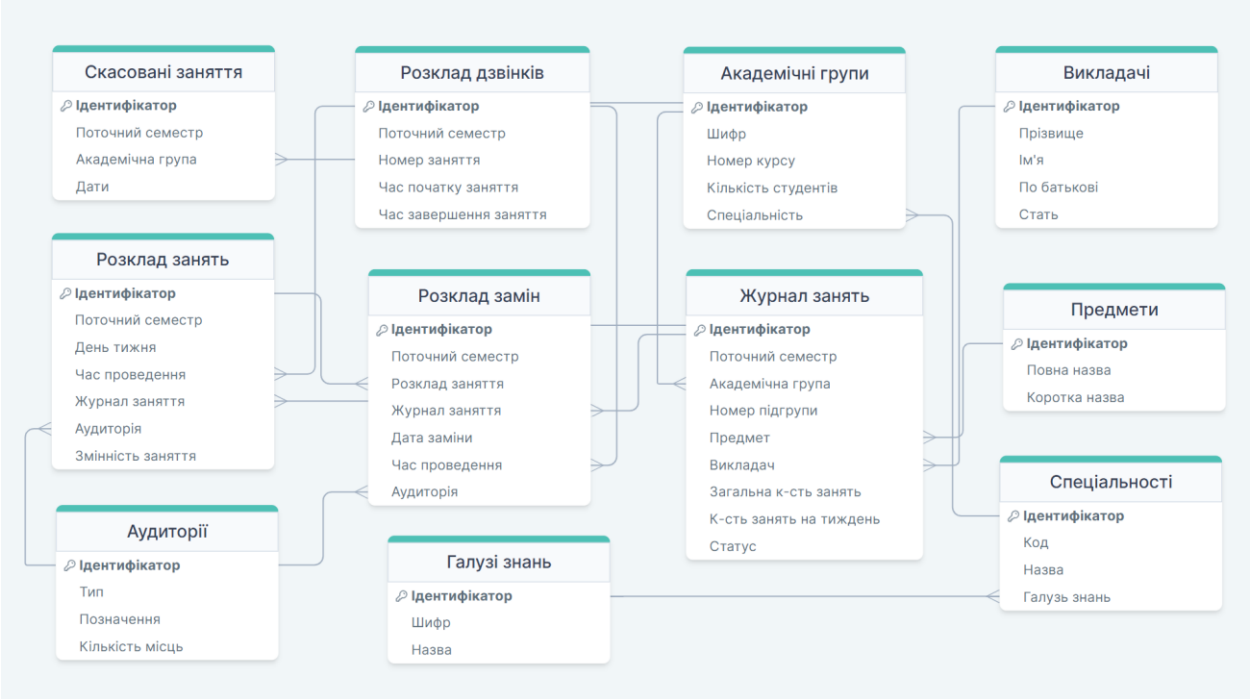


Рис. 3.3. Реляційна схема бази даних

3.3 Розробка десктопного застосунку

Оскільки десктопний застосунок розрахований на ОС Windows, для його реалізації цілком згодиться мова програмування C# та середовище розробки Visual Studio Community з Windows Forms. В даному середовищі розробки для створення віконних програм використовується форма (form), яка є основним елементом інтерфейсу користувача. Щоб додати будь-який елемент управління (кнопка, текстове поле, мітка тощо) на форму, потрібно перетягнути його з панелі елементів у область форми. Панель елементів містить бібліотеку вбудованих елементів управління, які можна використовувати для створення інтерфейсу користувача. Після того, як елемент управління перетягнуто на форму, його можна налаштувати, встановивши властивості (розмір, текст, колір тощо). Також можна додавати власний код для обробки подій, таких як натискання кнопки або зміна тексту в полі вводу. Завдяки використанню форм та візуального редактора, створення інтерфейсу користувача стає простим та інтуїтивно зрозумілим, оскільки не потрібно писати код для розміщення та налаштування елементів управління. На рисунку 3.4 зображено спроєктований інтерфейс десктопного застосунку. Користувацькі дані та налаштування у програмі будемо зберігати в тестовому файлі формату JSON, усі інші дані щодо навчального процесу будемо зберігати безпосередньо у локальну базу даних SQLite.

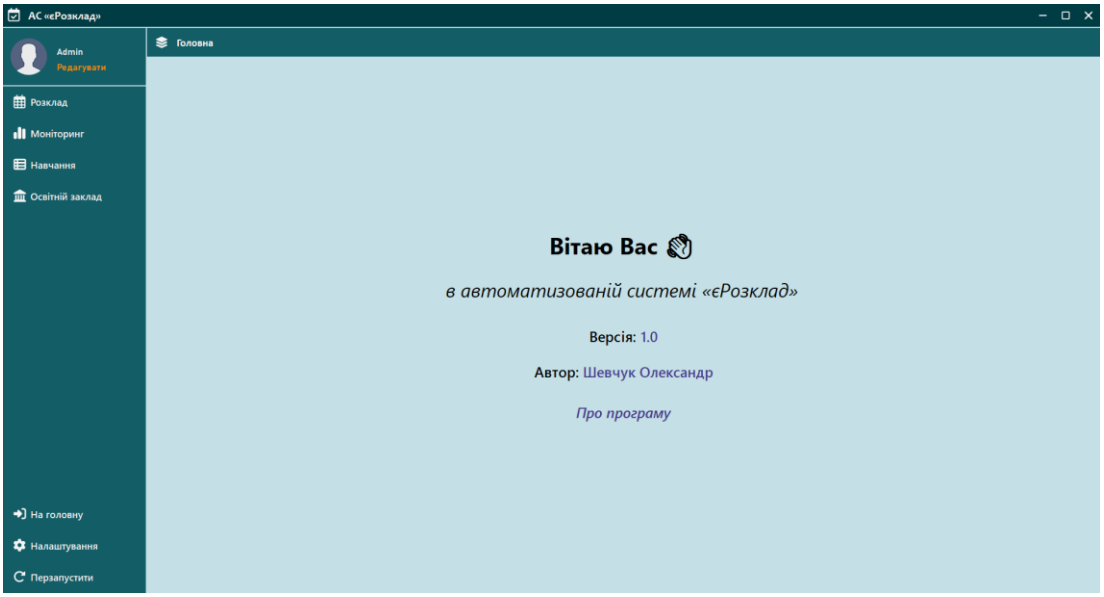


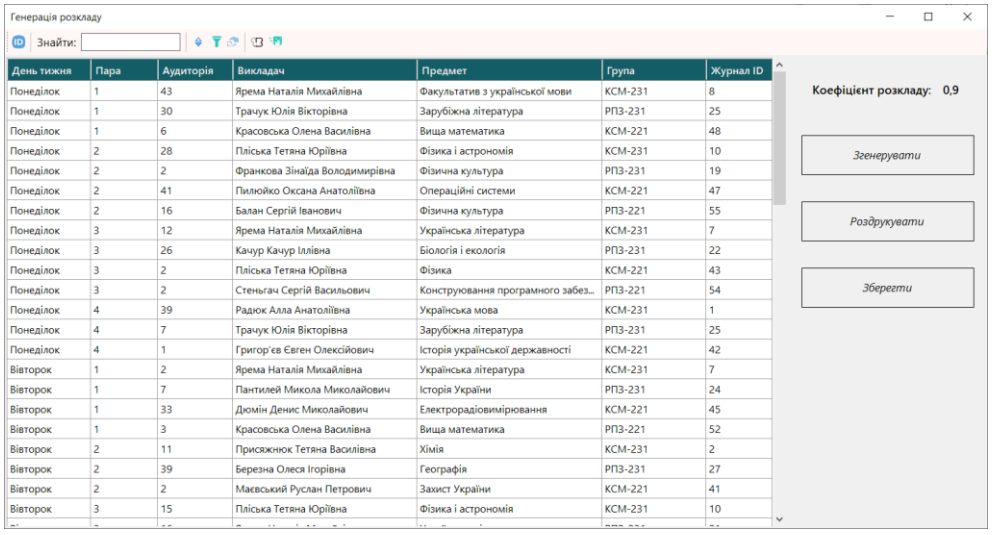
Рис. 3.4. Інтерфейс користувача десктопного застосунку

Десктопний застосунок містить такі основні частини меню: розклад, моніторинг, навчання та освітній заклад. Натиснувши на «Освітній заклад» користувач побачить випадаюче меню, яке містить: загальну інформацію, галузь знань, спеціальності, аудиторії, предмети, викладачів, академічні групи. Натискаючи на дані підменю користувачу буде представлено форму для внесення початкових даних. Важливо, щоб перейти до іншого меню потрібно задати поточний навчальний рік та його семестр. Перейшовши до «Навчання» користувач буде вносити такі дані як: журнал занять, розклад дзвінків, розклад занять, розклад замін та інформацію про скасовані заняття. У меню «Моніторинг» можна відстежувати викладачів та аудиторний фонд на основі розкладу занять.

Відкривши меню «Розклад» користувач має доступ до таких функцій:

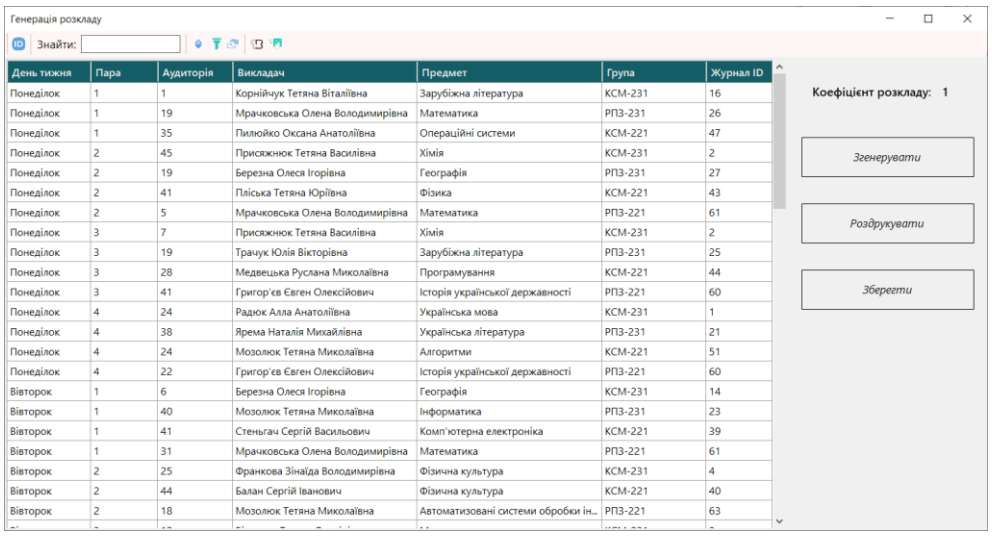
1. Вивід розкладу на основі обраної дати.
2. Генерація розкладу.
3. Друк розкладу (експорт даних в Excel).
4. Отримання QR-коду.
5. Опублікування розкладу.

Важливо, якщо користувач не вносив дані у журнал занять, вищеперелічені функції стануть недоступними, оскільки розклад автоматизовано формується на основі журналу занять. Формування здійснюється одразу після натискання на «Згенерувати розклад» (рис. 3.5-3.6).



День тижня	Пара	Аудиторія	Викладач	Предмет	Група	Журнал ID
Понеділок	1	43	Ярема Наталія Михайлівна	Факультатив з української мови	КСМ-231	8
Понеділок	1	30	Трачук Юлія Вікторівна	Зарубіжна література	РПЗ-231	25
Понеділок	1	6	Красовська Олена Василівна	Вища математика	КСМ-221	48
Понеділок	2	28	Пліська Тетяна Юріївна	Фізика і астрономія	КСМ-231	10
Понеділок	2	2	Франкова Зінаїда Володимирівна	Фізична культура	РПЗ-231	19
Понеділок	2	41	Пилюйко Оксана Анатоліївна	Операційні системи	КСМ-221	47
Понеділок	2	16	Балан Сергій Іванович	Фізична культура	РПЗ-221	55
Понеділок	3	12	Ярема Наталія Михайлівна	Українська література	КСМ-231	7
Понеділок	3	26	Качур Качур Іллівна	Біологія і екологія	РПЗ-231	22
Понеділок	3	2	Пліська Тетяна Юріївна	Фізика	КСМ-221	43
Понеділок	3	2	Стеньгач Сергій Васильович	Конструювання програмного забез...	РПЗ-221	54
Понеділок	4	39	Радюк Алла Анатоліївна	Українська мова	КСМ-231	1
Понеділок	4	7	Трачук Юлія Вікторівна	Зарубіжна література	РПЗ-231	25
Понеділок	4	1	Григор'єв Євген Олексійович	Історія української державності	КСМ-221	42
Вівторок	1	2	Ярема Наталія Михайлівна	Українська література	КСМ-231	7
Вівторок	1	7	Пантилей Микола Миколайович	Історія України	РПЗ-231	24
Вівторок	1	33	Дюмін Денис Миколайович	Електрорадіовимірювання	КСМ-221	45
Вівторок	1	3	Красовська Олена Василівна	Вища математика	РПЗ-221	52
Вівторок	2	11	Присяжнюк Тетяна Василівна	Хімія	КСМ-231	2
Вівторок	2	39	Березна Олеся Ігорівна	Географія	РПЗ-231	27
Вівторок	2	2	Маєвський Руслан Петрович	Захист України	КСМ-221	41
Вівторок	3	15	Пліська Тетяна Юріївна	Фізика і астрономія	КСМ-231	10

Рис. 3.5. Сформовано оптимальний розклад



День тижня	Пара	Аудиторія	Викладач	Предмет	Група	Журнал ID
Понеділок	1	1	Корнійчук Тетяна Віталіївна	Зарубіжна література	КСМ-231	16
Понеділок	1	19	Мрачковська Олена Володимирівна	Математика	РПЗ-231	26
Понеділок	1	35	Пилюйко Оксана Анатоліївна	Операційні системи	КСМ-221	47
Понеділок	2	45	Присяжнюк Тетяна Василівна	Хімія	КСМ-231	2
Понеділок	2	19	Березна Олеся Ігорівна	Географія	РПЗ-231	27
Понеділок	2	41	Пліська Тетяна Юріївна	Фізика	КСМ-221	43
Понеділок	2	5	Мрачковська Олена Володимирівна	Математика	РПЗ-221	61
Понеділок	3	7	Присяжнюк Тетяна Василівна	Хімія	КСМ-231	2
Понеділок	3	19	Трачук Юлія Вікторівна	Зарубіжна література	РПЗ-231	25
Понеділок	3	28	Медвецька Руслана Миколаївна	Програмування	КСМ-221	44
Понеділок	3	41	Григор'єв Євген Олексійович	Історія української державності	РПЗ-221	60
Понеділок	4	24	Радюк Алла Анатоліївна	Українська мова	КСМ-231	1
Понеділок	4	38	Ярема Наталія Михайлівна	Українська література	РПЗ-231	21
Понеділок	4	24	Мозолюк Тетяна Миколаївна	Алгоритми	КСМ-221	51
Понеділок	4	22	Григор'єв Євген Олексійович	Історія української державності	РПЗ-221	60
Вівторок	1	6	Березна Олеся Ігорівна	Географія	КСМ-231	14
Вівторок	1	40	Мозолюк Тетяна Миколаївна	Інформатика	РПЗ-231	23
Вівторок	1	41	Стеньгач Сергій Васильович	Комп'ютерна електроніка	КСМ-221	39
Вівторок	1	31	Мрачковська Олена Володимирівна	Математика	РПЗ-221	61
Вівторок	2	25	Франкова Зінаїда Володимирівна	Фізична культура	КСМ-231	4
Вівторок	2	44	Балан Сергій Іванович	Фізична культура	КСМ-221	40
Вівторок	2	18	Мозолюк Тетяна Миколаївна	Автоматизовані системи обробки ін...	РПЗ-221	63

Рис. 3.6. Сформовано найкращий розклад

Формування розкладу можна повторювати безліч разів. Найкращим сформованим розкладом вважається, якщо значення цільової функції дорівнює 1. Якщо значення в межах від 0,5 до 1, то розклад вважається оптимальним. Коли розклад створено, заклад може його опублікувати. Для цього потрібно натиснути на кнопку «Опублікувати». Після успішного опублікування відобразиться повідомлення, яке буде містити інформацію про успіх та версію даного розкладу. Щоб викладачі, а також здобувачі освіти отримали доступ до цього розкладу, користувачу потрібно натиснути на «Отримати QR-код» і система сформує даний код на сервері та завантажить його на робочий стіл, а також відкриє його для перегляду. Навчальний заклад може розмістити даний QR-код на інформаційному стенді, щоб користувачі могли авторизуватися у мобільному застосунку і переглядати розклад.

3.4 Розробка мобільного застосунку

Існує багато кросплатформних мов програмування, таких як React Native, Flutter, Xamarin тощо, які можуть підтримувати різні операційні системи. Однак, в даному випадку будемо використовувати мову програмування Java для розробки мобільного застосунку для ОС Android. В середовищі розробки Android Studio є дуже корисний елемент – візуальний редактор інтерфейсу. Він дозволяє просто створювати та редагувати інтерфейс користувача шляхом перетягування елементів управління безпосередньо на

екрані. Це значно полегшує процес розробки, оскільки розробники можуть бачити як буде виглядати їхній застосунок, не займаючись написанням дизайн-коду формату XML. Також, візуальний редактор має вбудовані інструменти для перевірки сумісності інтерфейсу з різними розмірами екранів та версіями Android. Це допомагає забезпечити оптимальний візуальний досвід користувачів на всіх пристроях, оскільки розробники можуть перевірити як буде виглядати їхній застосунок на різних екранах та пристроях, а також внести необхідні зміни для забезпечення максимальної сумісності. Існує два ключових файли конфігурації для Android – Gradle build та AndroidManifest. Перший є основним файлом конфігурації для всього проєкту. Він містить інформацію про версії бібліотек, які використовуються в проєкті, налаштування компіляції та конфігурації проєкту. Також відповідає за управління залежностями та налаштування середовища розробки й збірки проєкту [22, с. 21]. Другий – ключовий файл конфігурації самого застосунку. Він містить інформацію про дозволи, які потрібні застосунку для роботи, про його основні компоненти, такі як активності, сервіси, одержувачі та контент-провайдери, а також про інші важливі параметри, такі як мінімальна версія Android, яка підтримується застосунком [13, с. 46].

У процесі розробки мобільного застосунку будемо використовувати різні компоненти інтерфейсу. Одним з ключових компонентів для створення візуального інтерфейсу є активність (activity). Зазвичай, активність асоціюється з окремим екраном або вікном, причому перехід між вікнами відбувається як зміна від однієї активності до іншої [13, с. 229]. Проте, структурування застосунку на основі багатьох активностей не завжди є оптимальним рішенням. Для більш ефективної організації існує такий компонент як фрагмент (fragment). Фрагмент представляє шматочок візуального інтерфейсу, який може бути використаний повторно й багаторазово. Але фрагмент існує в контексті активності та має свій життєвий цикл, причому існувати окремо від активності він не може [13, с. 282; 22, с. 622]. Для зберігання даних застосунок буде використовувати файл JSON та

базу даних SQLite, яка буде містити інформацію пов’язану з розкладом занять. У файлі будемо зберігати такі дані, як роль, код та ім’я користувача, код доступу, а також інформацію про останнє оновлення розкладу. Варто відзначити, що авторизація користувача відбувається через сканування QR-коду наданим ЗВО від розробника для доступу до даних на сервері. Тому при успішній авторизації, застосунок буде містити код доступу, який буде використовуватися кожного разу при звертанні до сервера. Для звертання до сервера будемо використовувати клас HttpURLConnection, а для отримання інформації – BufferedReader разом з InputStreamReader. Крім того, для парсингу даних з потоку інформації у форматі JSON будемо використовувати клас JSONObject.

Для виведення інформації про розклад занять у вигляді власної конструкції необхідно виконати наступні кроки:

1. Створити макет списку з необхідними елементами, такими як TextView для виведення назви предмету, часу, аудиторії тощо (рисунок 3.7).
2. Створити клас, який буде описувати розклад та створити список об’єктів цього класу із наповненими даними розкладу, які будемо отримувати з бази даних.
3. Створити адаптер для списку, який буде з’єднувати дані з макетом списку. Для цього будемо використовувати клас ArrayAdapter, передавши в екземпляр адаптера контекст, макет списку та список даних.
4. Вивести список на екран, використавши віджет ListView.

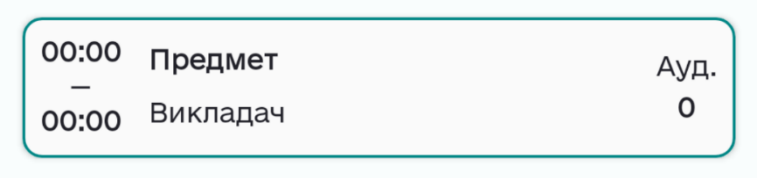


Рис. 3.7. Реалізований макет елемента списку в розкладі

Оскільки основною функцією мобільного застосунку є перегляд актуального розкладу, інтерфейс застосунку можна сконструювати таким чином, щоб при запуску застосунку, на головній сторінці відображався одразу розклад занять на поточний день. Отже, будемо використовувати три вкладки

розміщені внизу екрана, які будуть містити інформацію про денний розклад (на поточний та наступний день), тижневий розклад, а також вкладка перегляду корисної інформації – пошук розкладу за викладачем/групую, перегляд вільних аудиторій, інформація про застосунок та обраний заклад освіти користувачем тощо. Варто відзначити, що головна сторінка окрім розкладу ще відображає у реальному часі поточну дату і час відповідно до часового поясу України. Тому незалежно від обраного часового поясу на смартфоні, він буде відображатися завжди відносно України. На рисунках 3.8-3.9 подано функціональність даного застосунку.

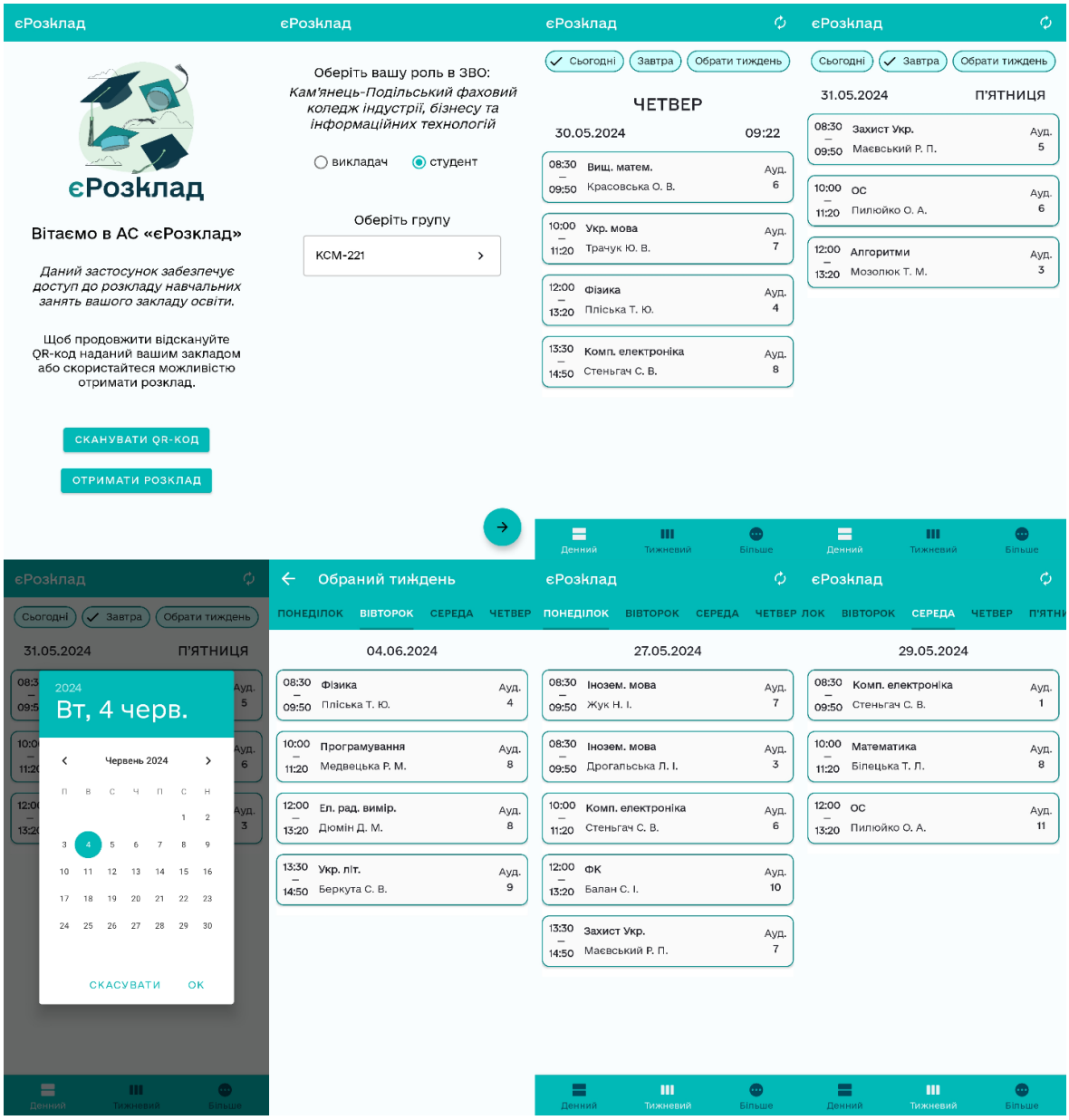


Рис. 3.8. Інтерфейс мобільного застосунку

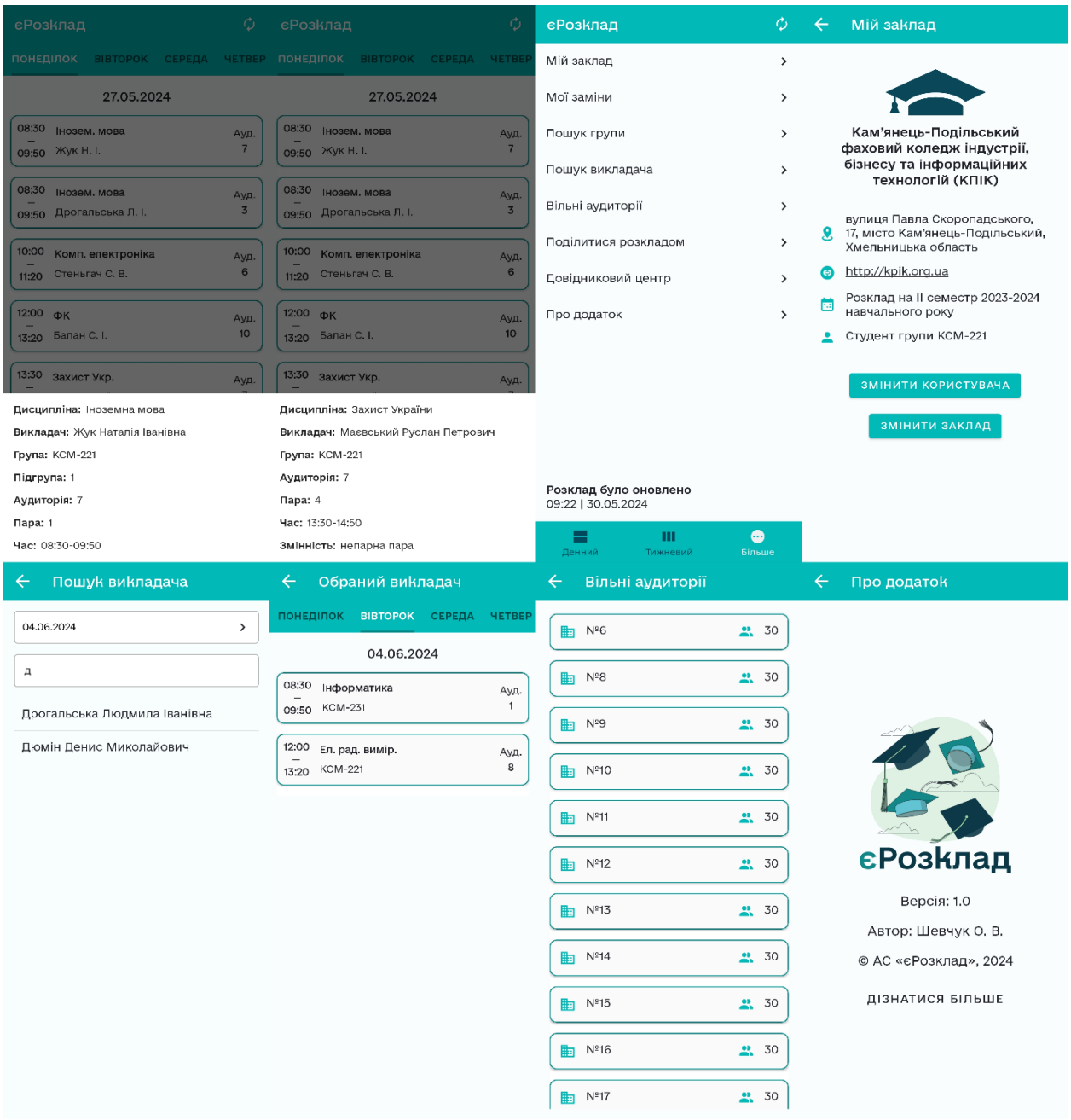


Рис. 3.9. Продовження інтерфейсу мобільного застосунку

З наведених ілюстрацій можна зробити висновок, що дизайн користувацького інтерфейсу мобільного застосунку відзначається легкістю та інтуїтивною зрозумілістю, що полегшує його використання. Крім того, мінімалістичний підхід у дизайні допомагає користувачам швидко знайти потрібну інформацію та здійснити необхідні дії, підвищуючи загальну ефективність взаємодії з мобільним застосунком.

3.5 Реалізація серверної частини

Серверна частина є головною ланкою системи, оскільки вона реалізує взаємодію між різними програмними застосунками, забезпечуючи передачу та отримання інформації. Для реалізації серверної інфраструктури будемо використовувати хостинг. Існує безліч безкоштовних хостинг-провайдерів, але більшість з них мають обмеження, дозволяючи створювати лише вебсайти без можливості обміну даними між певними програмами та віддаленої передачі файлів на хостинг або їх завантаження з нього. Тому найкращим вирішенням даної проблеми є користування хостинг-провайдером 000webhost. Він є одним з найпопулярніших безкоштовних хостинг-провайдерів, який надає широкий спектр можливостей для розробки та розгортання вебсайтів з базами даних. Крім того, він має зручний інтерфейс для управління файлами на сервері, що спрощує процес їх завантаження, видалення та редагування. Також хостинг-провайдер має високу надійність та швидкість роботи, що забезпечує стабільну роботу вебсайтів та швидке завантаження сторінок й інформації з хостингу. Безкоштовний період послуг, які надає 000webhost складає один рік [21].

Процес реєстрації є доволі простим, вказуємо свою адресу електронної пошти та придумаємо пароль. Далі підтверджуємо запит на реєстрацію через надісланий лист на електронну пошту і все. Щоб розпочати користуватися можливостями хостингу натискаємо на кнопку «Strat now», заповнюємо поля для створення доменного імені (рисунок 3.10), натискаємо на «Create» та обираємо «Upload site» – для подальшого завантаження файлів на хостинг. Також потрібно спроєктувати структуру каталогів для даного проєкту. Отже, будемо використовувати таку структуру каталогів з основними файлами на хостингу для реалізації серверної інфраструктури:

```
.
└── public_html/
    ├── index.html
    ├── robots.txt
    ├── .htaccess
    └── api/
```

```
├── index.php
├── helper.php
├── db_connect.php
├── upload_schedule.php
├── .htaccess
├── ...
└── data/
    └── ...
```

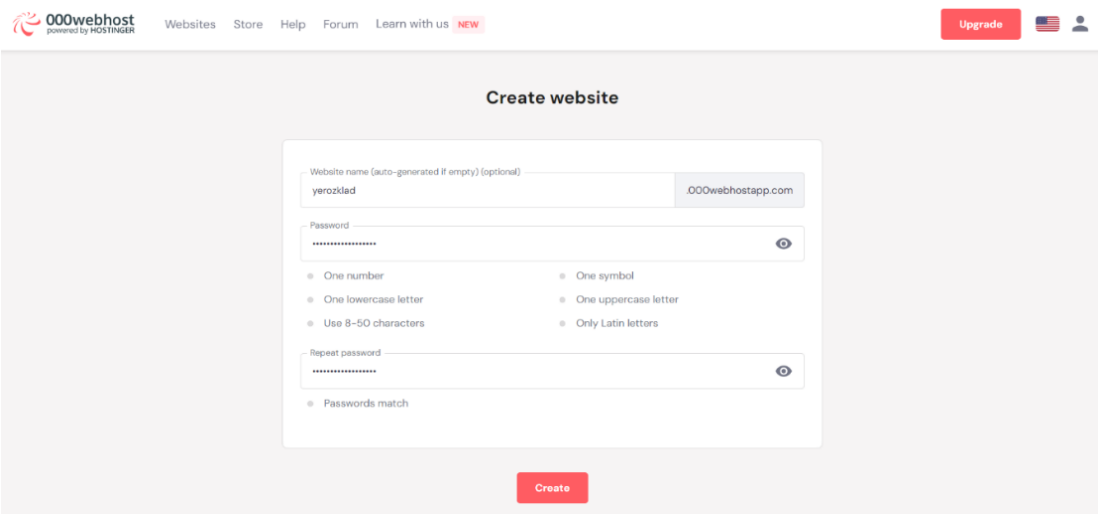


Рис. 3.10. Створення доменного імені в 000webhost

Основними функціями серверної частини в складовій АС «єРозклад є:

1. Автентифікація та авторизація: сервер забезпечує перевірку даних користувачів, що надходять від застосунків, підтверджуючи їхню справжність та надаючи доступ до відповідних функцій та даних, які знаходяться на сервері.

2. Обробка даних: сервер обробляє та аналізує дані, які надходять від застосунків, включаючи завантаження розкладу, додавання інформації від навчального закладу та її отримання, перевірку актуальності та оновлення розкладу в мобільному застосунку і т.д.

3. Зберігання даних: сервер відповідає за зберігання всіх даних про розклад, інформацію від навчального закладу та інших необхідних даних, забезпечуючи їхню надійність та доступність.

4. Синхронізація даних: сервер забезпечує синхронізацію даних між застосунками. Коли вносяться зміни в розклад або додається нова інформація

від закладу, сервер автоматично синхронізує дані між всіма пристроями, надсилаючи push-сповіщення користувачам.

5. Безпека: сервер забезпечує безпеку даних та взаємодії між застосунками, використовуючи певні методи шифрування та захисту даних, а також забезпечуючи захист від несанкціонованого доступу та можливих атак хакерів на інфраструктуру.

Отже, будемо використовувати мову програмування PHP та середовище розробки VS Code. Розпочнемо з файлу конфігурації (htaccess) вебсервера Apache. Його основне призначення полягає в наданні можливості адміністратору хостингу здійснювати URL-перенаправлення, URL-перезапис, контроль доступу до сторінок або всіх даних на хостингу, встановлення MIME-типів для різних типів файлів, керування кешуванням сторінок і файлів, налаштування компресії контенту для зменшення часу завантаження сторінок, а також впливу на SEO-параметри сайту, такі як встановлення канонічних URL-адрес, видалення або додавання www до URL-адрес тощо. Також є текстовий файл (robots), який забезпечує взаємодії вебсайту з пошуковими роботами (ботами). Він розміщується в кореневій директорії сайту і містить в собі вказівки для пошукових роботів про те, які сторінки та файли вони можуть, а які не можуть індексувати. Для початкової розробки, коли вебсторінка ще не готова до індексації пошуковими системами, рекомендується вказати в файлі robots.txt наступне:

```
User-agent: *
Disallow: /
```

Цей код забороняє всім пошуковим роботам індексувати всі файли та сторінки сайту. Також рекомендується використовувати стиснення GZIP для зменшення розміру файлів, які передаються з вебсервера на веббраузер користувача. Це дозволяє зменшити час завантаження сторінок сайту та економити трафік. Щоб ввімкнути стиснення на вебсервері Apache, необхідно додати наступний код у файл .htaccess:

```
<IfModule mod_gzip.c>
  mod_gzip_on Yes
```

```
    mod_gzip_dechunk Yes
    mod_gzip_item_include file \.(html?|txt|css|js|php|pl)$
    mod_gzip_item_include handler ^cgi-script$
    mod_gzip_item_include mime ^text/.*
    mod_gzip_item_include mime ^application/x-javascript.*
    mod_gzip_item_exclude mime ^image/.*
    mod_gzip_item_exclude rspheader ^Content-Encoding:.*gzip.*
</IfModule>
```

Для взаємодії хостингу з базою даних додамо наступний PHP-код у файл db_connect.php, проте спершу, створимо базу даних MySQL в 000webhost. Зайдемо в менеджер баз даних, натиснемо на «Create New Database», заповнимо потрібні дані та отримаємо інформацію, яку вкажемо у файл замість «***»:

```
class HelperDB {
    private static $server = "***";
    private static $port = "***";
    private static $db = "***";
    private static $user = "***";
    private static $password = "***";
    public static function getDB(string $query) {
        try {
            $conn = mysqli_connect(self::$server, self::$user,
self::$password, self::$db, self::$port);
            $conn->set_charset('utf8');
            $result = mysqli_query($conn, $query);
            mysqli_close($conn);
            return $result;
        } catch (mysqli_sql_exception $ex) {...}
    }
}
```

Щоб більш безпечно обмінюватися певною інформацією, будемо використовувати Base64 для кодування та розкодування переданих даних. Base64 – це схема кодування, яка дозволяє представляти двійкові дані у вигляді рядка символів, що складається з друкованих ASCII-символів. Наведемо приклад простої валідації даних за допомогою PHP-коду та бази даних MySQL:

```
<?php
```

```
    require_once('helper.php');
    require_once('db_connect.php');
    $ref = $_GET['ref'];
    if (Helper::isCorrect($ref)) {
        $params = Helper::getParams($ref);
        $login = $params['login'];
        $password = $params['password'];
        if (Helper::isCorrect($login) && Helper::isCorrect($password)) {
            if (isValidation($login, $password)) {
                Helper::push(json_encode(array("status" => "200", "message"
=> "authorized")));
            } else {
                Helper::push(json_encode(array("status" => "200", "message"
=> "NOT authorized")));
            }
        }
        Helper::showError(401);
    } else {
        Helper::showError(401);
    }
    function isValidation($login, $password) {
        $check = HelperDB::getDB("SELECT ...");
        if (mysqli_num_rows($check) > 0) {
            return true;
        }
        return false;
    }
    ?>
```

Загалом для реалізації серверної частини використовується мова програмування PHP, яка у більшості випадків виконує SQL-запити до бази даних і повертає результати у форматі JSON. Для інформативності наведемо алгоритм звернення, який зображено у вигляді блок-схеми на рисунку 3.11. Варто відзначити, що розклад занять зберігається у вигляді файлу локальної БД SQLite, який автоматизовано формується для спрощення структури даних і надсилається десктопним застосунком на хостинг у теку data використовуючи таке ім’я файлу: «id закладу»_«дата»_«час завантаження».db (наприклад, 2_2024-04-04_12-52-00.db).

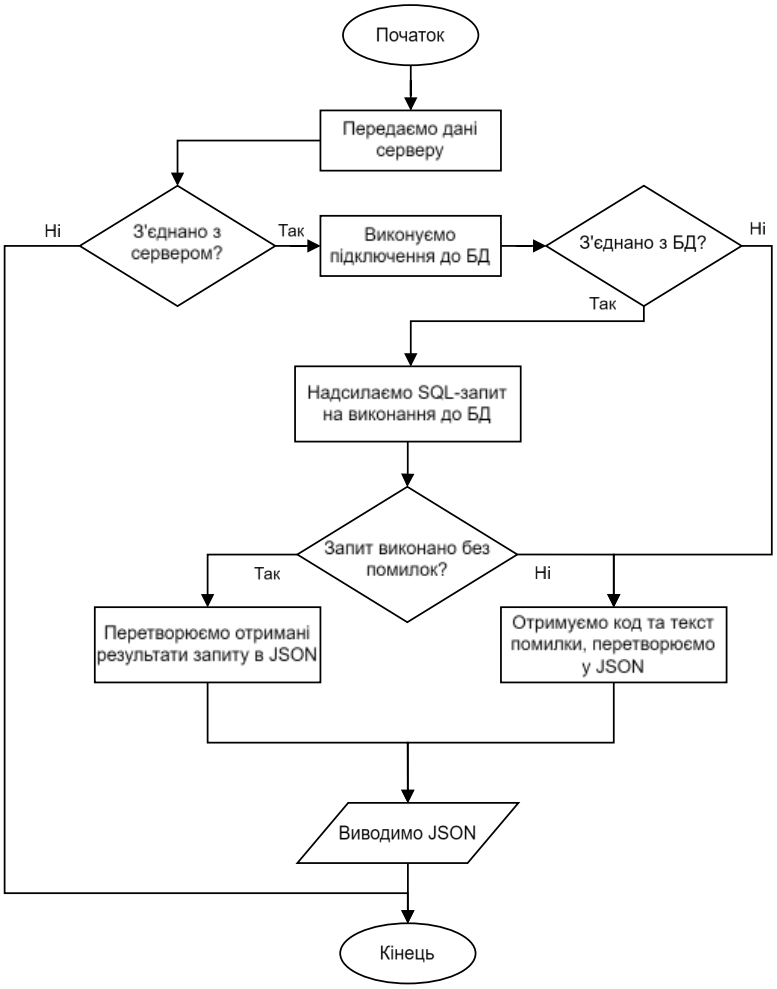


Рис. 3.11. Загальна блок-схема алгоритму звернення та отримання інформації від сервера

Слід зазначити, що для структурованого обміну даними використовується протокол HTTP. Він є основою для передачі даних у всесвітній павутині, визначає формати та порядки передачі повідомлень, щоб вебсервер та веббраузер могли розуміти один одного. Тому реалізована серверна інфраструктура використовує цей протокол та передає інформації використовуючи метод HTTP-запиту GET та інколи POST. GET – метод, який використовується для отримання даних із сервера. Коли клієнт надсилає GET-запит, сервер повертає запитані дані без їхньої зміни. POST – запит, який використовуються для надсилання даних на сервер. Він дозволяє клієнту створювати нові ресурси на сервері або надсилати дані для обробки [14, с. 9].

Висновок до розділу 3

У розділі детально описано застосування генетичного алгоритму для формування розкладу. Також наведено кроки проєктування головної бази даних модуля керування, використовуючи реляційну модель бази даних та вказано обрану систему керування базами даних – SQLite. Для виконання поставленої мети і завдань кваліфікаційної роботи використовуються різні мови програмування, такі як C#, Java та PHP. Наведено системні елементи застосунків та їх функціональні можливості, які відображені на рисунках. Вказано структуру файлів серверної частини та її фізичну реалізацію на обраному хостингу – 000webhost. Крім того, спроєктовано узагальнений алгоритм звернення та отримання інформації від сервера й акцентовано увагу на те, що для передачі інформації між застосунками використовуються певний протокол та методи HTTP-запиту – GET і POST та їх призначення для передачі інформації між застосунками.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ СИСТЕМИ

Процес тестування та налагодження є важливою складовою життєвого циклу розробки програмного забезпечення. Даний процес має на меті виявити й усунути помилки, невідповідності та дефекти у програмному забезпеченні, а також перевірити, чи відповідає система вимогам та очікуванням. Тестування починається з розробки тестових випадків, які базуються на вимогах до програмного забезпечення. Ці випадки описують конкретні сценарії використання системи, включаючи вхідні дані, дії користувача та очікувані результати. Потім тестові випадки виконуються на програмному забезпеченні, щоб перевірити, чи воно працює належним чином. Під час тестування можуть виникати помилки та дефекти, які повинні бути виправлені. Цей процес називається налагодженням. Налагодження може охоплювати аналіз коду програмного забезпечення, визначення причин помилок та внесення необхідних змін у код для їх виправлення. Після виправлення помилок програмне забезпечення знову тестується, щоб перевірити, чи дана програма працює відповідно до очікувань. Існують різні рівні тестування ПЗ, які використовуються для перевірки функціональності та виявлення дефектів у програмному забезпеченні. Ці рівні включають:

1. Компонентне (модульне) тестування – перевіряється функціональність та здійснюється пошук дефектів у частинах застосунків, які доступні й можуть бути протестовані окремо.

2. Інтеграційне тестування – призначене для перевірки зв’язку між компонентами, а також взаємодії з різними частинами системи.

3. Системне тестування – перевіряється інтегрована система на відповідність вимогам.

4. Приймальне тестування – формальний процес тестування, який перевіряє відповідність системи вимогам та проводиться з метою визначення чи задовольняє система приймальним критеріям та винесення рішення замовником або іншою уповноваженою особою чи приймається застосунок.

5. Регресійне тестування – спрямоване на перевірку змін, зроблених у застосунку або в оточуючому середовищі, для підтвердження того факту, що наявна функціональність працює як і раніше [1, с. 16].

Для відстежування помилок у програмному забезпечені використовують баг-трекінгові системи або системи відстеження помилок, які представляють собою програмні продукти, що дозволяють реєструвати й відслідковувати хід вирішення кожної помилки, виявленої тестувальником, до тих пір, поки проблема не буде вирішена.

Розглянемо найбільш популярні баг-трекінгові системи із відкритим кодом.

BugZilla – одна з найпопулярніших систем відстеження помилок, яка спочатку була розроблена для проєкту Mozilla, але згодом почала використовуватися в багатьох інших проєктах. BugZilla має багато можливостей, включаючи гнучку систему доступу на основі ролей, можливість створення запитів для пошуку помилок, підтримку електронної пошти та RSS-потоків, інтеграцію з системами керування версіями, такими як Git, CVS та SVN, багатомовний інтерфейс, підтримку СУБД MySQL, PostgreSQL та Oracle. Крім того, BugZilla має велику спільноту користувачів та розробників, що дозволяє знайти багато додаткових розширень та інтеграцій для цієї системи.

Mantis Bug Tracker – система, яка сприяє взаємодії між розробниками та користувачами (тестувальниками). Вона дозволяє користувачам створювати повідомлення про помилки та відстежувати наступні кроки щодо їх виправлення з боку розробників. Mantis Bug Tracking System має гнучкі можливості конфігурації, що дозволяє налаштовувати її не тільки для роботи над програмними продуктами, але і як систему обліку заявок для технічної підтримки.

Redmine – вебзастосунок для управління проєктами та відстеження помилок, який надає широкий спектр можливостей, включаючи ведення декількох проєктів, гнучку систему доступу, систему відстеження помилок,

діаграми Ганта, wiki та форуми для кожного проєкту, облік тимчасових витрат, легка інтеграція з репозиторіями (SVN, CVS, Git, Mercurial, Bazaar та Darcs), підтримка множинної аутентифікації LDAP, можливість самостійної реєстрації нових користувачів та підтримка різних СУБД.

Atlassian JIRA – комерційна система відстеження помилок, призначена для організації спілкування з користувачами, хоча в деяких випадках систему можна використовувати для управління проєктами. Розроблено компанією Atlassian Software Systems. JIRA створювалася для заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проєктами. Для кожного з проєктів створює і веде схеми безпеки та схеми оповіщення.

Для відстежування багів (помилок) будемо використовувати Mantis BT. Щоб зареєструвати виявлену помилку в системі, необхідно авторизуватися та натиснути на кнопку «Створити звіт про помилку» (рисунки 4.1-4.2). Процес створення звіту є легким та зручним, але водночас важливо знати правила оформлення багів. Також важливо спочатку перевірити, чи не було вже створено звіт про цю помилку раніше, щоб уникнути дублювання.

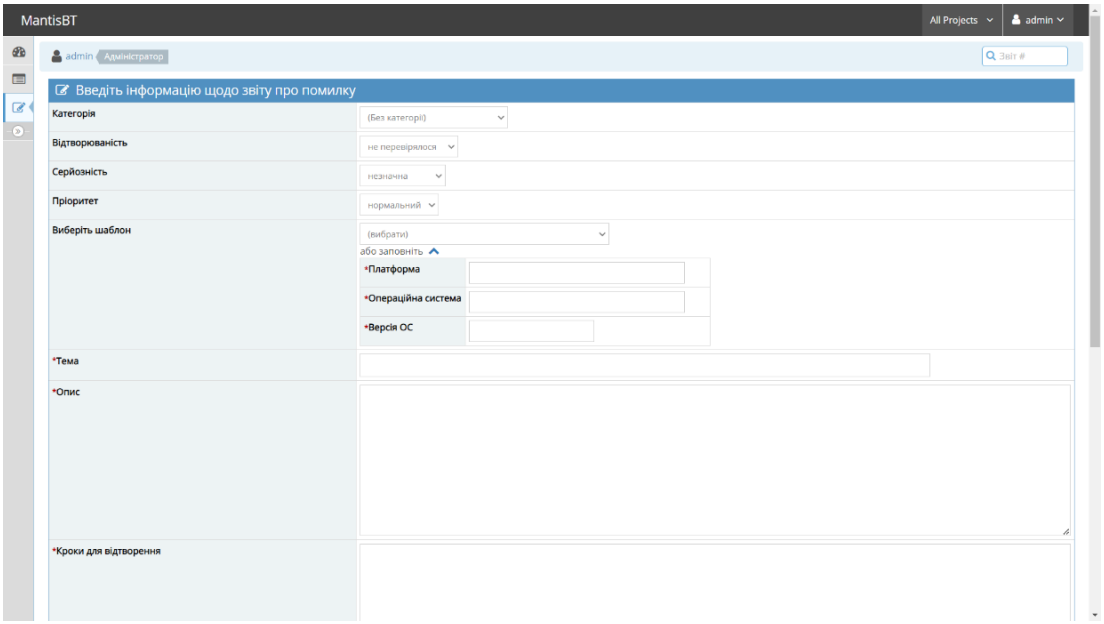


Рис. 4.1. Інтерфейс створення звіту про помилку в Mantis BT

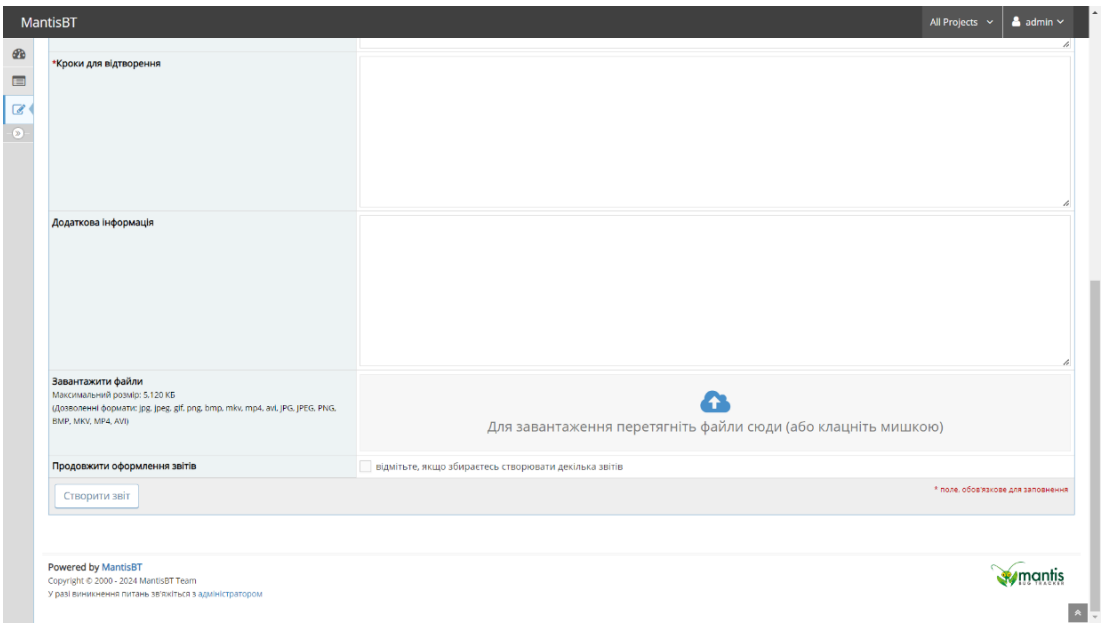


Рис. 4.2. Продовження інтерфейсу створення звіту про помилку

Розглянемо правила, яких необхідно дотримуватися при оформленні звіту про помилку:

- Поле «Тема» у звіті про помилку повинно містити дуже коротку, але водночас достатню для розуміння суті проблеми інформацію про баг.
- Баг повинен відповідати на такі три основні питання:
  - «Що?» – що відбувається або не відбувається згідно зі специфікаціями або уявленням тестувальника про нормальну роботу програмного продукту.
  - «Де?» – в якому місці інтерфейсу користувача або архітектури програмного продукту знаходиться проблема.
  - «Коли?» – в який момент роботи програмного продукту, після якої події або за яких умов проблема проявляється.
- Бути досить коротким, щоб повністю відображатися на екрані (у баг-трекінговій системі, де кінець речення в полі тема може бути обрізаним або вимагати скролінгу).
- Включати інформацію про середовище, в якому був виявлений баг.
- Поле «Опис» повинно містити опис проблеми в одному реченні, відповідно до принципів «що?», «де?» і «коли?», згідно з темою, та може містити кілька речень, якщо вказана інформація вказує на

важливі нюанси, які не могли бути описані в темі через обмеження кількості символів.

– Поле «Кроки відтворення» є найбільш цінною інформацією, оскільки воно являє собою керівництво до дії для тих, хто буде виправляти проблему, при цьому кількість кроків повинна бути достатньою для відтворення проблеми. Ще дане поле повинно містити короткий опис про фактичний та очікуваний результат помилки.

Також при оформленні звіту важливо додати статус серйозності та пріоритету помилки, щоб розробники могли визначити, наскільки швидко потрібно виправити її.

Серйозність – це атрибут, який відображає вплив дефекту на функціонування програми, ймовірність виникнення збою та наслідки цієї помилки, і може бути змінений лише у зв’язку з появою нової інформації про дефект.

Пріоритет – це атрибут, який визначає терміновість виправлення помилки та встановлюється менеджером продукту на основі загального розуміння системи, що тестується, та необхідності швидкого усунення помилки.

Ще важливо додати знімки екрана або відеозаписи, що демонструють помилку, оскільки це допоможе розробникам краще зрозуміти проблему та швидко її вирішити [8].

Висновок до розділу 4

У розділі описано процес тестування та налагодження системи, її важливість у забезпеченні належної роботи та зазначено основні рівні тестування, які використовуються для перевірки функціональності системи та виявлення дефектів. Вказано огляд популярних рішень для відстежування помилок в системі та зазначено обрану баг-трекінгову систему – Mantis BT. Також описано правила оформлення звіту про помилку, дотримання яких забезпечить максимальну інформативність повідомлень про виявлені дефекти в системі.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи проведено дослідження предметної області формування розкладів. Реалізовано процес автоматизованого формування розкладу навчальних занять для ЗВО І-ІІ рівня акредитації через застосування генетичного алгоритму. Особлива увага приділялася можливостям інформування про розклад занять, щоб здобувачі вищої освіти та викладачі могли своєчасно отримувати актуальний розклад. Крім того, було ретельно розглянуто проблематику в цій галузі та існуючі рішення, що підкреслює актуальність обраної теми та її практичне застосування в сучасних умовах. Також було проведено тестування та оптимізацію розробленої системи для забезпечення її стабільності та швидкодії. Враховано особливості роботи ЗВО І-ІІ рівня акредитації та специфіка навчального процесу, що дозволило створити ефективне рішення для автоматизації розкладу навчальних занять та керування ним.

При розробці автоматизованої системи використовувалися різноманітні мови програмування, адаптовані до конкретних платформ, таких як Windows, Android та Web. Ця багатоплатформність забезпечує гнучкість та широкий спектр можливостей для користувачів. Дані застосунків зберігаються у вигляді текстового файлу формату JSON, а дані навчального процесу зберігаються в полегшеній реляційній СУБД SQLite. Також серверна частина використовує СУБД MySQL для реалізації серверної інфраструктури та збереження персональних даних, ліцензійних ключів, кодів доступу та ін. інформації освітніх закладів. Також приділялася увага безпеці даних та контролю доступу до них. Для цього використовувались методи шифрування та кодування даних. Зокрема, для забезпечення безпеки при передачі даних на сервер та отриманні від нього інформації використовувалась схема кодування Base64. Для забезпечення безпеки зберігання даних в серверній базі даних використовувалась хеш-функція SHA-384 із сімейства алгоритмів SHA-2, яка дозволяє захистити дані від несанкціонованого доступу та модифікації.

Загалом система має зручний та інтуїтивно зрозумілий користувацький інтерфейс, що дозволяє швидко та легко опанувати роботу в програмних застосунках.

Важливо ще зазначити про перспективи та розширення автоматизованої системи. В першу чергу в десктопному застосунку можна додати контроль за навантаженням годин (вичитка навчальних дисциплін) та налаштувати систему під різні навчальні процеси, які можуть зберігати дані за всі навчальні роки, а не лише за поточний рік, який поділяється на два семестри. Також корисно було б реалізувати можливість додання тимчасового розкладу занять, який би демонструвався тільки викладачам в мобільному застосунку для можливого своєчасно коригування при змінах в розкладі.

Щодо мобільного застосунку, то тут також є що покращити. Зокрема, варто розглянути можливість інформування користувачів про різні події, які організовує освітній заклад. Така інформація могла б відображатися у вигляді банера на головній сторінці застосунку, що містить необхідні дані (наприклад, проходження анкетування, опитування). Загалом впровадження цих змін дозволило б значно покращити функціональність та користувацький досвід автоматизованої системи.

Розроблена автоматизована система задовольняє всім вимогам, поставленим на етапі постановки задачі, а це: створення системних модулів – один для управління розкладом, інший для інформування здобувачів освіти та викладачів про розклад занять, внесення базових даних освітнього закладу, а також внесення даних про поточний навчальний процес, який містить тривалість навчального року, номер семестру, дати початку та завершення семестру, а також розклад дзвінків. Створення журналу занять та створення розкладу навчальних занять, враховуючи графік навчального процесу для кожної академічної групи з можливістю автоматизованого формування розкладу, створювати розклад власноруч та вносити корективи при формуванні. Експортувати розклад в Microsoft Excel для друку та розміщення на інформаційних стендах закладу, а також опубліковувати розклад занять.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Авраменко А. С., Авраменко В. С., Косенюк Г. В. Тестування програмного забезпечення: навч. посіб. Черкаси: ЧНУ імені Богдана Хмельницького, 2017. 284 с.

2. АС «Деканат». *АСУ «ВНЗ»*. URL: https://vuz.osvita.net/as-dekanat (дата звернення: 04.06.2024).

3. Гайджарж В. І., Ізварін І. В. Бази даних в інформаційних системах. Київ: Університет «Україна», 2018. 418 с.

4. Дворецький М. Л., Нездолій Ю. О., Дворецька С. В., Кандиба І. О. Розробка мобільних застосунків для OS Android: навч. посіб. Миколаїв: ЧНУ ім. Петра Могили, 2021. 140 с.

5. Доценко С. І. Організація та системи керування базами даних: навч. посіб. Харків: УкрДУЗТ, 2023. 117 с.

6. Жураковський Б. Ю., Зенів І. О. Розробка та реалізація мережних протоколів: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2020. 462 с.

7. Інформаційні технології: навч. посіб. / уклад. О. І. Зачек, В. В. Сеник, Т. В. Магеровська та ін. Львів: ЛьвДУВС, 2022. 432 с.

8. Матеріали з курсу від компанії QATestLab. *Тренінговий центр QATestLab*. URL: https://training.qatestlab.com/blog/course-materials (дата звернення: 05.06.2024).

9. Пилипюк Т., Шевчук О. Створення мобільного застосунку розкладу занять закладу освіти. *Актуальні аспекти розвитку STEAM-освіти в умовах євроінтеграції: збірник матеріалів II Міжнародної науково-практичної інтернет-конференції*. Кропивницький: ДонДУВС, 2024. С. 215-216.

10. ПП Деканат. *Політек-СОФТ*. URL: http://www.politek-soft.kiev.ua (дата звернення: 04.06.2024).

11. Програмний комплекс «Автоматизована система управління навчальним закладом». *АСУ «МКР»*. URL: https://mkr.org.ua (дата звернення: 04.06.2024).

12. Програмування – 1. Процедурне програмування: навч. посіб. / уклад. Ю. Є. Грудзинський, В. Б. Бобков. Київ: КПІ ім. Ігоря Сікорського, 2023. 317 с.

13. Радченко К. О. Розроблення мобільних застосунків: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2023. 546 с.

14. Серверні web-технології: навч. посіб. / уклад. О. С. Бунке. Київ: КПІ ім. Ігоря Сікорського, 2023. 109 с.

15. Троцько В. В. Методи штучного інтелекту: навч.-метод. посіб. Київ: Університет «КРОК», 2020. 86 с.

16. Фратавчан В. Г., Фратавчан Т. М., Лукашів Т. О., Літвінчук Ю. А. Методи та системи штучного інтелекту: навч. посіб. Чернівці: ЧНУ, 2023. 114 с.

17. Шевчук О. Система інформування розкладу занять в закладах вищої освіти. *Збірник матеріалів наукової конференції за підсумками НДР здобувачів вищої освіти фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка у 2023-2024 н. р.* Кам'янець-Подільський: КПНУ ім. Івана Огієнка, 2024. С. 108-113.

18. Albahari J. C# 12 in a nutshell. Sebastopol: O'Reilly Media, 2023. 1083 p.

19. Albahari J., Albahari B. C# 12 pocket reference. Sebastopol: O'Reilly Media, 2023. 284 p.

20. Desktop Operating System Market Share Ukraine. *StatCounter Global Stats*. URL: https://gs.statcounter.com/os-market-share/desktop/ukraine/#monthly-202301-202402-bar (дата звернення: 06.04.2024).

21. Free Web Hosting 000webhost. URL: https://www.000webhost.com (дата звернення: 06.06.2024).

22. Horton J. Android programming for beginners. 3rd ed. Birmingham: Packt Publishing, 2021. 742 p.

23. Kramer O. Genetic algorithm essentials. Berlin: Springer, 2017. 92 p.

24. Mobile Operating System Market Share Ukraine. *StatCounter Global Stats*. URL: https://gs.statcounter.com/os-market-share/mobile/ukraine/#monthly-202301-202402-bar (дата звернення: 06.06.2024).

ДОДАТКИ

Додаток А SQL-запит на створення бази даних із таблицями, ключами й зв'язками

```
CREATE TABLE field_knowledge (
    id INTEGER PRIMARY KEY,
    code TEXT NOT NULL,
    name TEXT NOT NULL
);
CREATE TABLE specialties (
    id INTEGER PRIMARY KEY,
    code TEXT NOT NULL,
    name TEXT NOT NULL,
    field_knowledge_id INTEGER NOT NULL REFERENCES field_knowledge(id) ON DELETE CASCADE
);
CREATE TABLE classrooms (
    id INTEGER PRIMARY KEY,
    type INTEGER NOT NULL,
    name TEXT NOT NULL UNIQUE,
    number_seats INTEGER NOT NULL
);
CREATE TABLE subjects (
    id INTEGER PRIMARY KEY,
    full_name TEXT NOT NULL,
    short_name TEXT NOT NULL
);
CREATE TABLE teachers (
    id INTEGER PRIMARY KEY,
    surname TEXT NOT NULL,
    name TEXT NOT NULL,
    patronymic TEXT NOT NULL,
    sex INTEGER NOT NULL
);
CREATE TABLE groups (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    course_number INTEGER NOT NULL,
    number_students INTEGER NOT NULL,
    specialty_id INTEGER NOT NULL REFERENCES specialties(id) ON DELETE CASCADE);
CREATE TABLE class_journal (
    id INTEGER PRIMARY KEY,
    semester INTEGER NOT NULL,
    group_id INTEGER NOT NULL REFERENCES groups(id) ON DELETE CASCADE,
    subgroup_number INTEGER NOT NULL,
    subject_id INTEGER NOT NULL REFERENCES subjects(id) ON DELETE CASCADE,
    teacher_id INTEGER NOT NULL REFERENCES teachers(id) ON DELETE CASCADE,
    total_classes INTEGER NOT NULL,
    weekly_classes INTEGER NOT NULL,
    is_completed INTEGER NOT NULL
);
CREATE TABLE bell_schedule (
    id INTEGER PRIMARY KEY,
    semester INTEGER NOT NULL,
    period_number INTEGER NOT NULL,
    period_start_time TEXT NOT NULL,
    period_end_time TEXT NOT NULL
);
CREATE TABLE class_schedule (
    id INTEGER PRIMARY KEY,
    semester INTEGER NOT NULL,
    day_week INTEGER NOT NULL,
    bell_schedule_id INTEGER NOT NULL REFERENCES bell_schedule(id) ON DELETE CASCADE,
    class_journal_id INTEGER NOT NULL REFERENCES class_journal(id) ON DELETE CASCADE,
```

```
    classroom_id INTEGER NOT NULL REFERENCES classrooms(id) ON DELETE CASCADE,
    variability INTEGER NOT NULL
);
CREATE TABLE substitution_schedule (
    id INTEGER PRIMARY KEY,
    semester INTEGER NOT NULL,
    class_schedule_id INTEGER NOT NULL REFERENCES class_schedule(id) ON DELETE CASCADE,
    class_journal_id INTEGER NOT NULL REFERENCES class_journal(id) ON DELETE CASCADE,
    date TEXT NOT NULL,
    bell_schedule_id INTEGER NOT NULL REFERENCES bell_schedule(id) ON DELETE CASCADE,
    classroom_id INTEGER NOT NULL REFERENCES classrooms(id) ON DELETE CASCADE
);
CREATE TABLE canceled_classes (
    id INTEGER PRIMARY KEY,
    semester INTEGER NOT NULL,
    group_id INTEGER NOT NULL REFERENCES groups(id) ON DELETE CASCADE,
    dates TEXT NOT NULL
);
```

Додаток Б Фрагмент програмного коду десктопного застосунку

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.Linq;
using System.Threading;
using System.Windows.Forms;
using yeRozklad.Source.Classes;
namespace yeRozklad.Source.Forms
{
    public partial class GenerationFrom : Form
    {
        public GenerationFrom()
        {
            InitializeComponent();
            button1.Click += (s, a) => Start();
            Start();
        }
        private void Start()
        {
            Animation(true);
            List<Journal> journals = new List<Journal>();
            List<Classrooms> classrooms = new List<Classrooms>();
            List<Period> periods = new List<Period>();
            string semester = General.getCurrentSemester;
            using (SQLiteConnection connection = DatabaseHelper.connectionDB)
            {
                connection.Open();
                using (SQLiteCommand cmd = new SQLiteCommand($"SELECT id, period_number FROM
bell_schedule WHERE semester ={semester}", connection))
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            periods.Add(new Period
                            {
                                ID = reader.GetInt32(0),
                                Number = reader.GetInt32(1)
                            });
                        }
                    }
```

```
                }
                using (SQLiteCommand cmd = new SQLiteCommand("SELECT id, name, number_seats
FROM classrooms WHERE type = 0", connection))
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            classrooms.Add(new Classrooms
                            {
                                ID = reader.GetInt32(0),
                                Name = reader.GetString(1),
                                Capacity = reader.GetInt32(2)
                            });
                        }
                    }
                }
                using (SQLiteCommand cmd = new SQLiteCommand($"SELECT cj.id, cj.teacher_id,
cj.subject_id, cj.group_id, cj.weekly_classes, ag.number_students FROM class_journal cj LEFT
JOIN groups ag ON cj.group_id = ag.id WHERE cj.subgroup_number = 0 AND cj.semester
={semester}", connection))
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            journals.Add(new Journal
                            {
                                ID = reader.GetInt32(0),
                                TeacherId = reader.GetInt32(1),
                                SubjectId = reader.GetInt32(2),
                                GroupId = reader.GetInt32(3),
                                LessonsPerWeek = reader.GetInt32(4),
                                NumberStudents = reader.GetInt32(5)
                            });
                        }
                    }
                }
            }
            label2.Text = null;
            dataGridView1.DataSource = null;
            dataGridView1.Rows.Clear();
            dataGridView1.Refresh();
            new Thread(() =>
            {
                ScheduleGeneration generation = new ScheduleGeneration(journals, classrooms,
periods);
                Schedule schedule = generation.Solver();
                this.Invoke((MethodInvoker)delegate
                {
                    Print(schedule, periods, classrooms, journals);
                });
            }).Start();
        }
        private void Print(Schedule schedule, List<Period> periods, List<Classrooms>
classrooms, List<Journal> journals)
        {
            List<(int ID, string Name)> teachers = new List<(int ID, string Name)>();
            List<(int ID, string Name)> subjects = new List<(int ID, string Name)>();
            List<(int ID, string Name)> groups = new List<(int ID, string Name)>();
            using (SQLiteConnection connection = DatabaseHelper.connectionDB)
            {
                connection.Open();
                using (SQLiteCommand cmd = new SQLiteCommand("SELECT id, surname || ' ' ||
name || patronymic FROM teachers", connection))
```

```
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            teachers.Add((reader.GetInt32(0), reader.GetString(1)));
                        }
                    }
                }
                using (SQLiteCommand cmd = new SQLiteCommand("SELECT id, full_name FROM
subjects", connection))
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            subjects.Add((reader.GetInt32(0), reader.GetString(1)));
                        }
                    }
                }
                using (SQLiteCommand cmd = new SQLiteCommand("SELECT id, name FROM groups",
connection))
                {
                    using (SQLiteDataReader reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            groups.Add((reader.GetInt32(0), reader.GetString(1)));
                        }
                    }
                }
            }
            CheckLessons(schedule.Lessons);
            List<Lesson> sortedLessons = schedule.Lessons.OrderBy(l => l.DayOfWeek).ThenBy(l
=> l.PeriodId).ToList();
            DataTable dt = new DataTable();
            dt.Columns.Add("День тижня", typeof(string));
            dt.Columns.Add("Пара", typeof(string));
            dt.Columns.Add("Аудиторія", typeof(string));
            dt.Columns.Add("Викладач", typeof(string));
            dt.Columns.Add("Предмет", typeof(string));
            dt.Columns.Add("Група", typeof(string));
            dt.Columns.Add("Журнал ID", typeof(int));
            var dayOfWeek = new[]
            {
                new[] { "Понеділок", "1" },
                new[] { "Вівторок", "2" },
                new[] { "Середа", "3" },
                new[] { "Четвер", "4" },
                new[] { "П'ятниця", "5" }
            };
            foreach (var lesson in sortedLessons)
            {
                var journal = journals.FirstOrDefault(j => j.ID == lesson.JournalId);
                if (journal != null)
                {
                    var classroom = classrooms.FirstOrDefault(c => c.ID ==
lesson.ClassroomId);
                    var period = periods.FirstOrDefault(p => p.ID == lesson.PeriodId);
                    var teacher = teachers.FirstOrDefault(t => t.ID == journal.TeacherId);
                    var subject = subjects.FirstOrDefault(s => s.ID == journal.SubjectId);
                    var group = groups.FirstOrDefault(a => a.ID == journal.GroupId);
                    dt.Rows.Add(
                        dayOfWeek[lesson.DayOfWeek - 1][0],
                        period.Number.ToString() ?? lesson.PeriodId.ToString(),
```

```
                        classroom.Name ?? lesson.ClassroomId.ToString(),
                        teacher.Name ?? journal.TeacherId.ToString(),
                        subject.Name ?? journal.SubjectId.ToString(),
                        group.Name ?? journal.GroupId.ToString(),
                        lesson.JournalId
                    );
                }
            }
            label2.Text = Math.Round(schedule.Weight, 2).ToString();
            dataGridView1.DataSource = dt;
            Animation(false);
        }
        private void CheckLessons(List<Lesson> lessons)
        {
            var uniqueLessons = new Dictionary<string, Lesson>();
            foreach (var lesson in lessons)
            {
                var key =
$"{lesson.DayOfWeek}_{lesson.PeriodId}_{lesson.ClassroomId}_{lesson.JournalId}";
                if (!uniqueLessons.ContainsKey(key))
                {
                    uniqueLessons.Add(key, lesson);
                }
                else
                {
                    uniqueLessons[key] = lesson;
                }
            }
            lessons.Clear();
            foreach (var lesson in uniqueLessons.Values)
            {
                lessons.Add(lesson);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
namespace yeRozklad.Source.Classes
{
    internal class Period
    {
        public int ID { get; set; }
        public int Number { get; set; }
    }
    internal class Classrooms
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public int Capacity { get; set; }
    }
    internal class Journal
    {
        public int ID { get; set; }
        public int TeacherId { get; set; }
        public int SubjectId { get; set; }
        public int GroupId { get; set; }
        public int NumberStudents { get; set; }
        public int LessonsPerWeek { get; set; }
    }
    internal class Lesson
    {
        public int DayOfWeek { get; set; }
        public int PeriodId { get; set; }
```

```
        public int ClassroomId { get; set; }
        public int JournalId { get; set; }
    }
    internal class Schedule
    {
        public List<Lesson> Lessons { get; set; }
        public double Weight { get; set; }
        public Schedule()
        {
            this.Lessons = new List<Lesson>();
        }
    }
    internal class ScheduleGeneration
    {
        private const int population = 200;
        private const int generations = 500;
        private const int tournaments = 6;
        private const double mutations = 0.1;
        private Random rand;
        private List<Journal> journals;
        private List<Classrooms> classrooms;
        private List<Period> periods;
        public ScheduleGeneration(List<Journal> journals, List<Classrooms> classrooms,
List<Period> periods)
        {
            this.journals = journals;
            this.classrooms = classrooms;
            this.periods = periods;
        }
        public Schedule Solver()
        {
            List<Schedule> p = new List<Schedule>();
            for (int i = 0; i < population; i++)
            {
                p.Add(RandomPopulation());
            }
            for (int i = 0; i < generations; i++)
            {
                foreach (var schedule in p)
                {
                    schedule.Weight = CalcWeight(schedule);
                }
                p = p.OrderByDescending(s => s.Weight).ToList();
                if (p[0].Weight == 1.0)
                {
                    break;
                }
                List<Schedule> temp = new List<Schedule>();
                int elite = (int)(population * 0.2);
                for (int j = 0; j < elite; j++)
                {
                    temp.Add(p[j]);
                }
                for (int j = elite; j < population; j++)
                {
                    Schedule s1 = Selection(p);
                    Schedule s2 = Selection(p);
                    Schedule s = Crossover(s1, s2);
                    Mutation(s);
                    temp.Add(s);
                }
                p = temp;
            }
            return p[0];
        }
```

```
        private Schedule RandomPopulation()
        {
            rand = new Random();
            Schedule s = new Schedule();
            foreach (var journal in journals)
            {
                for (int i = 0; i < journal.LessonsPerWeek; i++)
                {
                    Lesson lesson = new Lesson();
                    lesson.JournalId = journal.ID;
                    var _classrooms = classrooms.Where(c => c.Capacity >=
journal.NumberStudents).ToList();
                    lesson.ClassroomId = _classrooms[rand.Next(_classrooms.Count)].ID;
                    lesson.PeriodId = periods[rand.Next(periods.Count)].ID;
                    lesson.DayOfWeek = rand.Next(1, 6);
                    s.Lessons.Add(lesson);
                }
            }
            return s;
        }
        private double CalcWeight(Schedule schedule)
        {
            int constraints = 4;
            double weight = 1.0;
            foreach (var lesson in schedule.Lessons)
            {
                var journal = journals.FirstOrDefault(j => j.ID == lesson.JournalId);
                bool teacherConflict = schedule.Lessons.Any(l => l != lesson && l.DayOfWeek
== lesson.DayOfWeek && l.PeriodId == lesson.PeriodId && journals.FirstOrDefault(j => j.ID ==
l.JournalId)?.TeacherId == journal.TeacherId);
                if (teacherConflict)
                {
                    weight -= 1.0 / constraints;
                }
                bool groupConflict = schedule.Lessons.Any(l => l != lesson && l.DayOfWeek ==
lesson.DayOfWeek && l.PeriodId == lesson.PeriodId && journals.FirstOrDefault(j => j.ID ==
l.JournalId)?.GroupId == journal.GroupId);
                if (groupConflict)
                {
                    weight -= 1.0 / constraints;
                }
                bool classroomConflict = schedule.Lessons.Any(l => l != lesson && l.DayOfWeek
== lesson.DayOfWeek && l.PeriodId == lesson.PeriodId && l.ClassroomId == lesson.ClassroomId
&& l.JournalId == lesson.JournalId);
                if (classroomConflict)
                {
                    weight -= 1.0 / constraints;
                }
                bool capacityConflict = classrooms.FirstOrDefault(c => c.ID ==
lesson.ClassroomId).Capacity < journal.NumberStudents;
                if (capacityConflict)
                {
                    weight -= 1.0 / constraints;
                }
            }
            return weight;
        }
        private Schedule Selection(List<Schedule> p)
        {
            rand = new Random();
            List<Schedule> t = new List<Schedule>();
            for (int i = 0; i < tournaments; i++)
            {
                int indx = rand.Next(p.Count);
                t.Add(p[indx]);
```

```
            }
            Schedule s = t[0];
            double w = CalcWeight(s);
            for (int i = 1; i < t.Count; i++)
            {
                double c = CalcWeight(t[i]);
                if (c > w)
                {
                    s = t[i];
                    w = c;
                }
            }
            return s;
        }
        private Schedule Crossover(Schedule s1, Schedule s2)
        {
            rand = new Random();
            Schedule s = new Schedule();
            int point = rand.Next(s1.Lessons.Count);
            for (int i = 0; i < point; i++)
            {
                s.Lessons.Add(s1.Lessons[i]);
            }
            for (int i = point; i < s2.Lessons.Count; i++)
            {
                s.Lessons.Add(s2.Lessons[i]);
            }
            return s;
        }
        private void Mutation(Schedule s)
        {
            rand = new Random();
            for (int i = 0; i < s.Lessons.Count; i++)
            {
                if (rand.NextDouble() < mutations)
                {
                    int indx = rand.Next(s.Lessons.Count);
                    (s.Lessons[i], s.Lessons[indx]) = (s.Lessons[indx], s.Lessons[i]);
                }
            }
        }
    }
}
```

Додаток В Фрагмент програмного коду мобільного застосунку

```
package ua.shevchuk.yerozklad.home;
import android.app.DatePickerDialog;
import android.os.Bundle;
import android.os.Handler;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListView;
import android.widget.TextView;
import androidx.fragment.app.Fragment;
import com.google.android.material.chip.Chip;
import com.google.android.material.chip.ChipGroup;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import ua.shevchuk.yerozklad.R;
import ua.shevchuk.yerozklad.classes.AppData;
import ua.shevchuk.yerozklad.classes.General;
```

```
public class TodayFragment extends Fragment {
    private View rootView;
    private ChipGroup chipGroup;
    private Chip chipOpen;
    private TextView textView1, textView2, textView3, textView4;
    private ListView listView;
    private Handler handler;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        rootView = inflater.inflate(R.layout.fragment_today, container, false);
        chipGroup = rootView.findViewById(R.id.chipGroup);
        chipOpen = rootView.findViewById(R.id.chip3);
        textView1 = rootView.findViewById(R.id.textView1);
        textView2 = rootView.findViewById(R.id.textView2);
        textView3 = rootView.findViewById(R.id.textView3);
        textView4 = rootView.findViewById(R.id.textView4);
        listView = rootView.findViewById(R.id.listView);
        chipOpen.setOnClickListener(v -> {
            DatePickerDialog dpd = new DatePickerDialog(getContext(), (view, year, month,
day) -> {
                Calendar c = new GregorianCalendar(year, month, day);
                String date = General.getDateFormat("dd.MM.yyyy", c.getTime());
                General.openItemsActivity(getActivity(),  new String[][] {{"key",
"selectedWeek1"}, {"value", date}, {"value2", AppData.readData("role")}, {"value3",
AppData.readData("user_code")}});
            }, General.getCalendar().get(Calendar.YEAR),
General.getCalendar().get(Calendar.MONTH), General.getCalendar().get(Calendar.DAY_OF_MONTH));
            dpd.show();
        });
        chipGroup.check(chipGroup.getChildAt(0).getId());
        chipGroup.setOnCheckedChangeListener((group, checkedId) -> {
            Chip selectedChip = rootView.findViewById(checkedId);
            if (selectedChip.getId() == R.id.chip1) {
                updateCurrentTime();
                General.hideView(textView1, false);
                General.updateSchedule(General.currentDay(),
Integer.parseInt(General.getDateFormat("dd")), null, null,
General.getDateFormat("dd.MM.yyyy"), listView, textView4, getContext());
            } else if (selectedChip.getId() == R.id.chip2) {
                handler.removeCallbacksAndMessages(null);
                General.hideView(textView1, true);
                Calendar c = General.getCalendar();
                c.add(Calendar.DAY_OF_YEAR, 1);
                Date nextDay = c.getTime();
                int number = Integer.parseInt(General.getDateFormat("dd", nextDay));
                String day = General.getDateFormat("EEEE", nextDay);
                String date = General.getDateFormat("dd.MM.yyyy", nextDay);
                textView2.setText(date);
                textView3.setText(day);
                General.updateSchedule(General.getIndexDay(nextDay), number, null, null,
date, listView, textView4, getContext());
            }
        });
        handler = new Handler();
        updateCurrentTime();
        return rootView;
    }
    private void updateCurrentTime() {
        String currentDay = General.getDateFormat("EEEE");
        String currentDate = General.getDateFormat("dd.MM.yyyy");
        String currentTime = General.getDateFormat("HH:mm");
        if (!currentDay.equals(textView1.getText())) {
            textView1.setText(currentDay);
```

```
            General.updateSchedule(General.currentDay(),
Integer.parseInt(General.getDateFormat("dd")), null, null, currentDate, listView, textView4,
getContext());
        }
        if (!currentDay.equals(textView2.getText())) {
            textView2.setText(currentDate);
        }
        if (!currentTime.equals(textView3.getText())) {
            textView3.setText(currentTime);
        }
        handler.postDelayed(this::updateCurrentTime, 1000);
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        handler.removeCallbacksAndMessages(null);
    }
}
public class ScheduleItem {
    private String startTime;
    private String endTime;
    private String shortDiscipline;
    private String shortTeacher;
    private String classroom;
    private String lessonNumber;
    private String group;
    private String subgroup;
    private String fullDiscipline;
    private String fullTeacher;
    private String variability;
    public ScheduleItem(String startTime, String endTime, String shortDiscipline, String
shortTeacher, String classroom, String lessonNumber, String group, String subgroup, String
fullDiscipline, String fullTeacher, String variability) {
        this.startTime = startTime;
        this.endTime = endTime;
        this.shortDiscipline = shortDiscipline;
        this.shortTeacher = shortTeacher;
        this.classroom = classroom;
        this.lessonNumber = lessonNumber;
        this.group = group;
        this.subgroup = subgroup;
        this.fullDiscipline = fullDiscipline;
        this.fullTeacher = fullTeacher;
        this.variability = variability;
    }
    public String getStartTime() {
        return startTime;
    }
    public String getEndTime() {
        return endTime;
    }
    public String getShortDiscipline() {
        return shortDiscipline;
    }
    public String getShortTeacher() {
        return shortTeacher;
    }
    public String getClassroom() {
        return classroom;
    }
    public String getLessonNumber() {
        return lessonNumber;
    }
    public String getGroup() {
        return group;
```

```
    }
    public String getSubgroup() {
        return subgroup;
    }
    public String getFullDiscipline() {
        return fullDiscipline;
    }
    public String getFullTeacher() {
        return fullTeacher;
    }
    public String getVariability() {
        return variability;
    }
}
public class General {
    private static TimeZone getTimeZone() {
        return TimeZone.getTimeZone("Europe/Kiev");
    }
    public static Locale getLocale() {
        return new Locale("uk", "UA");
    }
    public static Calendar getCalendar() {
        return GregorianCalendar.getInstance(getTimeZone(), getLocale());
    }
    public static String getDateFormat(String format) {
        SimpleDateFormat sdf = new SimpleDateFormat(format, getLocale());
        sdf.setTimeZone(getTimeZone());
        return sdf.format(new Date());
    }
    public static String getDateFormat(String format, Date date) {
        SimpleDateFormat sdf = new SimpleDateFormat(format, getLocale());
        sdf.setTimeZone(getTimeZone());
        return sdf.format(date);
    }
    public static boolean isNullOrEmpty(String string) {
        return string == null || string.trim().isEmpty();
    }
    public static void message(String t, View v) {
        Snackbar.make(v, t, Snackbar.LENGTH_SHORT).show();
    }
    public static void message(String t, String buttonText, View v, View.OnClickListener
listener) {
        Snackbar snackbar = Snackbar.make(v, t, Snackbar.LENGTH_LONG);
        snackbar.setAction(buttonText, listener);
        snackbar.show();
    }
    public static int getIndexDay(Date date) {
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);
        int day = calendar.get(Calendar.DAY_OF_WEEK);
        switch (day) {
            case Calendar.MONDAY: return 1;
            case Calendar.TUESDAY: return 2;
            case Calendar.WEDNESDAY: return 3;
            case Calendar.THURSDAY: return 4;
            case Calendar.FRIDAY: return 5;
            case Calendar.SATURDAY: return 6;
            case Calendar.SUNDAY: return 7;
        } return 0;
    }
    public static void hideView(View view, boolean flag) {
        if (flag) {
            if (view.getVisibility() == View.VISIBLE) {
                view.setVisibility(View.GONE);
            }
```

```
        } else {
            if (view.getVisibility() == View.GONE) {
                view.setVisibility(View.VISIBLE);
            }
        }
    }
    public static void hideSchedule(boolean flag, ListView listView, TextView textView) {
        if (flag) {
            General.hideView(listView, true);
            General.hideView(textView, false);
        } else {
            General.hideView(listView, false);
            General.hideView(textView, true);
        }
    }
    public static boolean isAllowedDate(String selectDate) {
        try {
            List<String> list =
DatabaseHelper.getListData(DatabaseHelper.GET_EDUCATION_DATES);
            String start_date = list.get(0);
            String end_date = list.get(1);
            SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
            Date currentDate = dateFormat.parse(selectDate);
            Date startDate = dateFormat.parse(start_date);
            Date endDate = dateFormat.parse(end_date);
            if (currentDate.compareTo(startDate) >= 0 && currentDate.compareTo(endDate) <= 0)
{               return true;
            } else { return false; }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }
    public static void updateSchedule(int day, int number, String role, String user, String
selectDate, ListView listView, TextView textView, Context context) {
        int parity = (number % 2 == 0) ? 2 : 1;
        String userRole = role;
        String userCode = user;
        if (General.isNullOrEmpty(userRole) && General.isNullOrEmpty(userCode) ||
General.isNullOrEmpty(userRole) || General.isNullOrEmpty(userCode)) {
            userCode = AppData.readData("user_code");
            userRole = AppData.readData("role");
        }
        if (isAllowedDate(selectDate)) {
            List<ScheduleItem> scheduleItems = new ArrayList<>();
            List<List<String>> list =
DatabaseHelper.getList2Data(DatabaseHelper.queryToSchedule(userRole, day, userCode, parity));
            if (list.size() > 0) {
                for (List<String> row : list) {
                    scheduleItems.add(new ScheduleItem(row.get(0), row.get(1), row.get(6),
row.get(4), row.get(9), row.get(2), row.get(3), row.get(8), row.get(7), row.get(5),
row.get(10)));
                }
            }
            if (scheduleItems.size() > 0) {
                hideSchedule(false, listView, textView);
                General.scheduleList(context, listView, userRole, scheduleItems);
            } else {
                hideSchedule(true, listView, textView);
            }
        } else {
            hideSchedule(true, listView, textView);
        }
    }
```

```
    public static void scheduleList(Context context, ListView listView, String role,
List<ScheduleItem> scheduleItems) {
        ArrayAdapter<ScheduleItem> adapter = new ArrayAdapter<ScheduleItem>(context,
R.layout.schedule_item, R.id.textView_startTime, scheduleItems) {
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView textViewStartTime = view.findViewById(R.id.textView_startTime);
                TextView textViewEndTime = view.findViewById(R.id.textView_endTime);
                TextView textViewDiscipline = view.findViewById(R.id.textView_discipline);
                TextView textViewTeacher = view.findViewById(R.id.textView_teacher);
                TextView textViewClassroom = view.findViewById(R.id.textView_classroom);
                ScheduleItem item = getItem(position);
                String startTime = item.getStartTime();
                String endTime = item.getEndTime();
                String shortDiscipline = item.getShortDiscipline();
                String shortTeacher = item.getShortTeacher();
                String classroom = item.getClassroom();
                int lesson = Integer.parseInt(item.getLessonNumber());
                String group = item.getGroup();
                int subgroup = Integer.parseInt(item.getSubgroup());
                String fullDiscipline = item.getFullDiscipline();
                String fullTeacher = item.getFullTeacher();
                int variability = Integer.parseInt(item.getVariability());
                textViewStartTime.setText(startTime);
                textViewEndTime.setText(endTime);
                textViewDiscipline.setText(shortDiscipline);
                textViewTeacher.setText(role.equals("1") ? group : shortTeacher);
                textViewClassroom.setText(classroom);
                view.setOnClickListener(v -> {
                    View customLayout =
LayoutInflater.from(context).inflate(R.layout.selected_schedule_dialog, null);
                    LinearLayout linearLayout1, linearLayout2;
                    linearLayout1 = customLayout.findViewById(R.id.about_subgroup);
                    linearLayout2 = customLayout.findViewById(R.id.about_variability);
                    TextView textView1, textView2, textView3, textView4, textView5,
textView6, textView7, textView8, textView9;
                    textView1 = customLayout.findViewById(R.id.discipline);
                    textView2 = customLayout.findViewById(R.id.teacher);
                    textView3 = customLayout.findViewById(R.id.group);
                    textView4 = customLayout.findViewById(R.id.subgroup);
                    textView5 = customLayout.findViewById(R.id.classroom);
                    textView6 = customLayout.findViewById(R.id.lesson);
                    textView7 = customLayout.findViewById(R.id.time);
                    textView8 = customLayout.findViewById(R.id.variability);
                    General.hideView(linearLayout1, subgroup == 0);
                    General.hideView(linearLayout2, variability == 0);
                    textView1.setText(fullDiscipline);
                    textView2.setText(fullTeacher);
                    textView3.setText(group);
                    textView4.setText(String.valueOf(subgroup));
                    textView5.setText(classroom);
                    textView6.setText(String.valueOf(lesson));
                    textView7.setText(startTime + "-" + endTime);
                    textView8.setText(variability == 1 ? "непарна пара" : "парна пара");
                    final BottomSheetDialog bottomSheerDialog = new
BottomSheetDialog(context);
                    bottomSheerDialog.setContentView(customLayout);
                    bottomSheerDialog.show();
                });
                return view;
            }
        };
        listView.setAdapter(adapter);
    }
}
```