Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для ведення та обліку розкладу занять в школі

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Леськів М.А.
(підпис) (прізвище та ініціали)

Керівник _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г. І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Мудрик І.Я.
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«___» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Леськіву Максиму Андрійовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка програмного забезпечення для ведення та обліку для розкладу занять в школі

Керівник роботи Небесний Руслан Михайлович, доктор філософії, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*
2. Термін подання студентом завершеної роботи 22 червня 2026 р.
3. Вихідні дані до роботи Літературні джерела
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ. 1.1 Проблематика та аналіз процесів складання розкладу. 1.2 Схема предметної області та автоматизація. 1.3 Порівняльний аналіз програмних аналогів. 1.4 Обґрунтування вибору технологічного стеку. 1.5 Висновки. РОЗДІЛ 2 РОЗРОБКА ЛОГІЧНОЇ СТРУКТУРИ БАЗИ ДАНИХ. 2.1 Інфологічне проектування предметної області. 2.2 Даталогічне проектування БД. 2.3 Логічна схема БД та нормалізація. 2.4 Висновки. РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОБҐРУНТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА. 3.1 Архітектура додатка та розробка інтерфейсу. 3.2 Ергономіка та модернізація графічних форм. 3.3 Реалізація модулів бізнес-логіки. 3.4 Інструкція користувача АРМ. 3.5 Системні вимоги та розгортання ПЗ. 3.6 Тестування системи та аналіз результатів. 3.7 Висновки. РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ. 4.1 Оптимізація діяльності людини-оператора. 4.2 Безпекові вимоги при експлуатації ПК. 4.3 Висновки. ВИСНОВКИ. ПЕРЕЛІК ДЖЕРЕЛ. ДОДАТКИ.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Актуальність теми, мета та завдання дослідження. 3. Аналіз предметної області: проблематика та обмеження. 4. Концептуальна архітектура інформаційної системи. 5. Логічна структура та інфологічна модель бази даних. 6. Обґрунтування технологічного стеку. 7. Практична реалізація клієнтського додатка. 8. Графічний інтерфейс та функціональні можливості АРМ. 9. Алгоритми обробки даних та пошуку. 10. Тестування системи: захист цілісності та валідація. 11. Ергономіка взаємодії та безпека життєдіяльності оператора. 12. Підсумкові результати роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Мариненко С.Ю., доцент		

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	виконано
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	виконано
3.	Виконання дослідження щодо методів автоматизації та управління процесом складання розкладу занять у загальноосвітніх навчальних закладах	17.02.2026-10.05.2026	виконано
	Розроблення локальної інформаційної системи та реляційної бази даних для автоматизованого ведення і обліку шкільного розкладу		виконано
4.	Оформлення розділу «АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ»	11.05.2026-17.05.2026	виконано
5.	Оформлення розділу «РОЗРОБКА ЛОГІЧНОЇ СТРУКТУРИ БАЗИ ДАНИХ»	18.05.2026-24.05.2026	виконано
6.	Оформлення розділу «ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОБҐРУНТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА»	25.05.2026-31.05.2026	виконано
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-08.06.2026	виконано
8.	Виконання завдання до підрозділу «Основи охорони	01.06.2026-08.06.2026	виконано
	праці»		виконано
9.	Оформлення кваліфікаційної роботи	09.06.2026-11.06.2026	виконано
10.	Нормоконтроль	18.06.2026	виконано
11.	Перевірка на плагіат	19.06.2026	виконано
12.	Попередній захист кваліфікаційної роботи	22.06.2026	виконано
13.	Захист кваліфікаційної роботи	23.06.2026	

Студент _____ Леськів М.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення для ведення та обліку розкладу занять в школі // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Леськів Максим Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2026 // С. 65, рис. – 16, табл. –10, кресл. –12, додат. –2, бібліогр. –30.

Ключові слова: втоматизація, розклад занять, база даних, реляційна структура, rad studio, інтерфейс користувача, оптимізація ресурсів, об'єктно-орієнтоване програмування.

Кваліфікаційна робота присвячена дослідженню методів та інструментів автоматизації процесу організації, ведення та оперативного обліку навчального розкладу в загальноосвітніх закладах. В першому розділі кваліфікаційної роботи описано специфіку бізнес-процесів складання шкільного розкладу з урахуванням жорстких та м'яких нормативних обмежень. Висвітлено проблематику традиційного паперового підходу до управління кадровим і аудиторним фондом та санітарно-гігієнічні вимоги до освітнього процесу. Розглянуто існуючі напрямки автоматизації та проведено порівняльний аналіз глобальних хмарних і локальних спеціалізованих систем. Проаналізовано системні та функціональні вимоги до майбутнього програмного забезпечення й обґрунтовано вибір технологічного стеку.

В другому розділі кваліфікаційної роботи розроблено логічну та фізичну структуру реляційної бази даних у середовищі СУБД Microsoft Access. Досліджено принципи нормалізації таблиць та необхідність застосування комбінованого підходу з частковою денормалізацією для забезпечення високої швидкодії клієнтського додатка. Подано детальну інфологічну модель «Сутність-Зв'язок» та даталогічні специфікації семи розроблених таблиць (базових довідників та щоденних реєстрів).

В третьому розділі кваліфікаційної роботи описано практичну реалізацію повноцінного клієнтського додатка з ергономічним графічним інтерфейсом у середовищі Delphi 12. Проаналізовано архітектуру взаємодії візуальних компонентів із локальним сховищем даних за допомогою технології Microsoft ADO та розроблено алгоритми бізнес-логіки (динамічна ініціалізація підключення, превентивна валідація полів). Проведено комплексне функціональне тестування інформаційної системи на реальних масивах даних, перевірено крос-версійну сумісність у Windows та сформовано інструкцію для користувача.

Об’єкт дослідження: процес організації, ведення та оперативного обліку навчального розкладу занять у закладах загальної середньої освіти.

Предмет дослідження: архітектурні принципи, алгоритмічні методи та програмні інструменти RAD Studio для автоматизації обліку та корекції шкільного розкладу.

ANNOTATION

Development of Software for Managing and Scheduling School Timetables // Qualification work of the educational level «Bachelor» // Leskiv Maksym Andriiovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2026 // P.65, fig. –16, tabl. –10, chair. –12, annexes. –2, references –30.

Keywords: automation, class schedule, database, relational structure, rad studio, user interface, resource optimization, object-oriented programming..

The qualification work is dedicated to the research of methods and tools for automating the process of organization, management, and operational accounting of the educational schedule in general secondary education institutions.

The goal of the work is to design and develop a specialized software application for automated formation, maintenance, and accounting of school class schedules to minimize resource conflicts and improve the efficiency of educational process administration.

The first section of the qualification paper considered the specifics of the business processes of compiling a school schedule, taking into account strict and soft regulatory constraints. The problems of the traditional paper-based approach to the management of human and classroom resources, as well as sanitary and hygienic requirements for the educational process, are highlighted. Existing automation trends are considered, and a comparative analysis of global cloud and local specialized systems is conducted.

In the second section of the qualification work, it is considered the development of the logical and physical structure of a relational database in the Microsoft Access DBMS environment. The principles of table normalization and the necessity of applying a combined approach with partial denormalization to ensure high

performance of the client application are investigated. A detailed Entity-Relationship infological model and datalogical specifications of the seven developed tables (basic directories and daily registers) are presented.

The third section of the qualification work describes the practical implementation of a fully-fledged client application with an ergonomic graphical user interface in the Delphi 12 environment. The architecture of the interaction between visual components and the local data repository using Microsoft ADO technology is analyzed. Business logic algorithms (dynamic connection initialization, preventive field validation) are developed. Comprehensive functional testing of the information system on real data sets was conducted, cross-version compatibility in Windows was verified, and a user manual was formed.

Object of research: the process of organization, management, and operational accounting of the educational class schedule in institutions of general secondary education.

Subject of research: architectural principles, algorithmic methods, and RAD Studio software tools for automating the accounting and correction of the school schedule.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADO (ActiveX Data Objects) – об'єктна модель та технологія компанії Microsoft для доступу до даних.

OLE DB (Object Linking and Embedding, Database) – набір системних інтерфейсів для доступу до даних різних типів

RAD (Rapid Application Development) – концепція та середовище швидкої розробки прикладного програмного забезпечення.

SQL (Structured Query Language) – мова структурованих запитів, призначена для управління даними в реляційних базах даних.

VCL (Visual Component Library) – об'єктно-орієнтована бібліотека візуальних компонентів для розробки графічного інтерфейсу в Delphi.

АРМ – автоматизоване робоче місце.

БД – база даних.

НФ (1НФ, 2НФ, 3НФ) – нормальна форма (перша, друга, третя нормальні форми бази даних).

ОС – операційна система.

ПЗ – програмне забезпечення.

ПІБ – прізвище, ім’я, по батькові.

ПК – персональний комп’ютер.

СанПіН – санітарні правила і норми.

СУБД – система управління базами даних.

ЗМІСТ

Вступ

Актуальність теми. Сучасний етап розвитку інформаційного суспільства характеризується тотальною цифровізацією та автоматизацією управлінських процесів у всіх сферах людської діяльності, серед яких освітня галузь посідає одне з ключових місць. Заклади загальної середньої освіти (школи, ліцеї, гімназії) щоденно стикаються з необхідністю координації великої кількості ресурсів: людських (вчителі, учні), просторових (навчальні кабінети, спортзали, лабораторії) та часових (сітка уроків, зміни, факультативи).

Організаційним ядром будь-якого навчального закладу є розклад занять. Він виступає головним інструментом тайм-менеджменту для учнів та викладачів, безпосередньо впливаючи на психоемоційне навантаження, втомлюваність та загальну ефективність засвоєння знань. Проте процес створення та оперативного обліку шкільного розкладу традиційними «ручними» методами є надзвичайно складною комбінаторною задачею. Адміністрація закладу змушена враховувати десятки жорстких обмежень: гранично допустиме денне навантаження учнів згідно з санітарними нормами (СанПіН), індивідуальні графіки роботи викладачів, обмеженість спеціалізованих кабінетів, поділ класів на групи (наприклад, при вивченні іноземних мов чи інформатики). Будь-які зміни – хвороба вчителя, ремонт кабінету чи незаплановані заходи – руйнують статичну структуру графіка, викликаючи «накладки» та організаційний хаос.

Вибір архітектури та інструментальних засобів розробки даної системи зумовлений передусім реальними фінансовими можливостями та поточним матеріально-технічним станом більшості вітчизняних загальноосвітніх шкіл, де комп'ютерне обладнання часто є застарілим і не має прогресивних характеристик. Створення автономного десктопного додатка в середовищі RAD Studio дозволяє отримати просту, швидку та повністю локальну альтернативу існуючим комерційним вебплатформам, які вимагають постійного підключення

до Інтернету, сторонніх хмарних сховищ та регулярної оплати коштовних підписок.

Розроблена програма функціонує безпосередньо на локальному ПК завуча, що забезпечує абсолютну приватність і безпеку збереження даних освітнього процесу без ризику витоку інформації у мережу. Завдяки високій оптимізації компілятора Object Pascal, програмний комплекс має мінімальні системні вимоги, демонструє максимальну швидкодію на техніці попередніх поколінь і є повністю безкоштовним у впровадженні. Тому проектування та практична реалізація такого бюджетного, незалежного та невимогливого до ресурсів програмного забезпечення є критично актуальним напрямком для реальної модернізації шкільного менеджменту.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього ступеня «Бакалавр» є проектування, розробка та практична реалізація ефективної автоматизованої системи обліку й підтримки інформаційної цілісності шкільного розкладу, яка дозволяє мінімізувати вплив людського фактора під час планування освітнього процесу.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати предметну область та специфіку нормативних вимог до формування шкільного розкладу;

– Провести порівняльний аналіз існуючих аналогів автоматизації планування занять;

– Спроектувати концептуальну, інфологічну та даталогічну моделі реляційної бази даних;

– Обґрунтувати вибір інструментальних засобів розробки (RAD Studio);

– Розробити клієнтський додаток із розгалуженим графічним інтерфейсом для адміністрування даних;

– Провести тестування системи на реальних масивах даних навчального закладу.

Практичне значення одержаних результатів. Створене програмне забезпечення може бути безпосередньо впроваджене в практику роботи адміністрації (завучів з навчальної роботи) середніх шкіл. Система дозволяє скоротити час на щотижневе обслуговування розкладу, ліквідувати ризик виникнення кадрових та аудиторних колізій, а також забезпечує швидке формування звітів та друкованих форм розкладу

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Проблематика та аналіз процесів складання розкладу

Процес управління розкладом у закладах загальної середньої освіти суттєво відрізняється від аналогічних процесів у вищих навчальних закладах, що зумовлено специфікою законодавства та фізіологічними особливостями учнів [1]. Створення розкладу – це багатофакторна оптимізаційна задача, яка включає координацію багатьох об'єктів предметної області.

До основних об'єктів планування відносяться учнівські класи, педагогічний колектив вчителів та матеріально-технічна база навчальних кабінетів. Кожен викладач має індивідуальну тарифікацію тижневого навантаження в годинах, методичні дні та індивідуальні обмеження щодо доступності. Кабінетний фонд включає загальні класні кімнати та спеціалізовані лабораторії, місткість яких повинна відповідати чисельності учнів.

Головна складність полягає в наявності великої кількості жорстких обмежень, порушення яких робить розклад фізично неможливим. Один вчитель не може одночасно проводити уроки у двох різних класах, одна аудиторія не може бути одночасно зайнята кількома групами, а учні одного класу не можуть мати два різних уроки одночасно.

Окрім жорстких критеріїв, існують м'які обмеження, які базуються на державних санітарних нормах. Складні предмети мають рівномірно розподілятися протягом тижня та ставитися на другі або треті уроки, коли працездатність учнів є найвищою. Для учнів наявність пустих уроків у вигляді вікон всередині навчального дня є неприпустимою з міркувань безпеки та дисципліни.

Традиційний підхід до ведення розкладу силами адміністрації за допомогою паперових носіїв супроводжується високим ризиком механічних помилок, низькою швидкістю адаптації до змін та відсутністю автоматичного

контролю за дотриманням лімітів робочого часу вчителів. Наявність зазначених чинників зумовлює необхідність переходу до використання спеціалізованих інформаційних систем розробки індивідуального програмного забезпечення [2, 3].

Процес оптимізації шкільного розкладу не може здійснюватися суб'єктивно або в довільному порядку, оскільки він регламентується суворими державними гігієнічними вимогами до організації освітнього процесу. Згідно з діючим Санітарним регламентом для закладів загальної середньої освіти [4], кожна навчальна дисципліна має чітко визначений бал складності, який відображає рівень інтелектуального та психоемоційного навантаження на учня.

Для формалізації комбінаторної задачі та закладання математичної логіки перевірки накладок у програмний комплекс, шкалу складності основних шкільних предметів систематизовано у таблиці 1.1.

Таблиця 1.1 – Шкала складності шкільних предметів за санітарно-гігієнічними нормами

Назва навчальної дисципліни	**Бал складності за СанПіН**	**Рекомендовані години для викладання**
1	**2**	**3**
Математика (Алгебра, Геометрія)	11	2–3 уроки (пік працездатності)
Іноземна мова (Англійська, Німецька)	10	2–4 уроки
Фізика, Хімія	9	2–3 уроки (можливе спарювання)
Українська мова, Світова література	8	1–4 уроки
Історія, Географія	7	1–5 уроки

Продовження таблиці 1.1

1	2	3
Біологія, Інформатика	6	1–5 уроки
Трудове навчання, Образотворче мистецтво	3	5–6 уроки
Фізична культура, Музичне мистецтво	1	5–6 уроки (розвантаження)

Врахування представлених балів складності є обов'язковим при проектуванні алгоритмів автоматизації. Завуч школи при формуванні сітки занять повинен розподіляти предмети так, щоб сумарний бал складності для конкретного класу досягав свого піку в середині тижня (вівторок, середа) та припадав на 2–3 уроки протягом навчального дня. Наявність такої нормативної бази зумовлює необхідність створення гнучких інформаційних каналів керування даними, які дозволяють оперативно відстежувати сумарні навантаження кадрового та аудиторного фонду.

1.2 Схема предметної області та автоматизація

Для успішного проектування автоматизованого робочого місця адміністратора закладу освіти необхідно детально типізувати та структурувати інформаційні потоки, що циркулюють у процесі ведення розкладу. Інформаційна система розглядається як комплексний інтегрований механізм взаємодії користувача, програмного ядра та сховища даних. Загальна інформаційна схема предметної області, яка відображає архітектуру взаємодії всіх складових частин, наведена на рисунку 1.1.

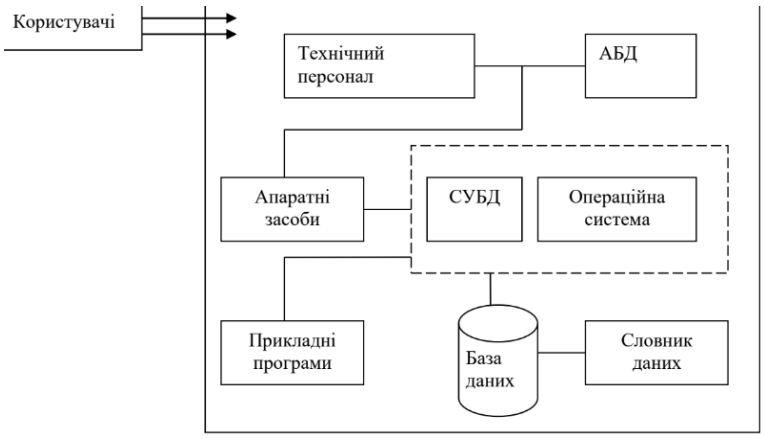

Рисунок 1.1 - Інформаційна схема взаємодії компонентів системи

Розглянемо функціональне призначення кожного базового компонента представленої інформаційної схеми в контексті розроблюваної системи. Користувачами системи виступають завучі школи, які взаємодіють із прикладним інтерфейсом для введення, редагування або перегляду розкладу уроків. Адміністратор баз даних відповідає за підтримання фізичної цілісності комп'ютерної техніки та резервне копіювання файлів сховища.

Прикладні програми є програмним ядром, написаним мовою Object Pascal у середовищі розробки, яке надає користувачу зручні екранні форми, таблиці та кнопки, а також виконує всі перевірки колізій у коді. Системний прошарок у вигляді СУБД приймає запити від нашої програми, шукає потрібні рядки у файлі бази даних та повертає їх назад у клієнтський додаток.

Безпосередньо база даних є файлом на жорсткому диску, де у вигляді реляційних таблиць фізично зберігається вся інформація про вчителів, кабінети та години занять. Словник даних зберігає метадані та опис структури реляційних таблиць. Впровадження даної схеми дозволяє повністю ізолювати кінцевого користувача від складних низькорівневих операцій із файлами

Автоматизоване робоче місце відповідального працівника повинно забезпечувати виконання наступного базового циклу бізнес-процесів:

1. Введення нормативно-довідкової інформації та формування списків вчителів, переліку дисциплін, кабінетного фонду.

2. Встановлення логічних зв'язків та закріплення конкретних предметів за викладачами для утворення навчального навантаження.

3. Оперативне ведення розкладу та безпосередня побудова сітки занять по днях тижня.

4. Валідація цілісності та автоматичний перевірочний моніторинг на предмет відсутності конфліктів кадрових чи аудиторних ресурсів.

Зазначені бізнес-процеси охоплюють усі сфери діяльності диспетчера розкладу та дозволяють повністю автоматизувати рутинні операції навчального закладу.

Для переходу від загальної концепції автоматизації до безпосереднього алгоритмічного проектування системи необхідно виконати декомпозицію базового бізнес-процесу ведення розкладу. Застосування методології системного аналізу дозволяє чітко структурувати інформаційні зв'язки, визначивши склад вхідних даних, регулюючих документів, механізмів виконання та кінцевих результатів.

Системна декомпозиція процесу оперативного обліку шкільного розкладу включає такі чотири взаємопов'язані категорії потоків:

– Вхідні інформаційні потоки (Inputs): базовий робочий навчальний план школи на поточний навчальний рік; тарифікація годин педагогічного колективу; затверджений списочний склад кабінетного фонду та лабораторій; специфікація поділу класів на підгрупи.

– Керуючі чинники та обмеження (Controls): Закон України «Про повну загальну середню освіту»; державні санітарно-гігієнічні правила та нормативи (СанПіН); внутрішні накази директора закладу щодо структури навчального року, тривалості змін та канікулярних періодів.

– Механізми реалізації процесу (Mechanisms): завуч з навчально-виховної роботи в ролі безпосереднього оператора системи; апаратне забезпечення автоматизованого робочого місця (персональний комп'ютер, пристрої друку); розроблюване спеціалізоване програмне забезпечення в середовищі RAD Studio Delphi 12.

– Вихідні результуючі потоки (Outputs): сформована та оптимізована електронна сітка розкладу уроків на весь навчальний тиждень; індивідуальні графіки завантаженості викладачів; звіти для контролюючих органів освіти; друковані форми розкладу для інформаційних стендів школи.

Проведена декомпозиція дозволяє виділити інформаційні ізольовані масиви даних, які у подальшому трансформуються у фізичні реляційні таблиці бази даних, а логічні обмеження та керуючі чинники лягають в основу програмного коду обробників подій.

1.3 Порівняльний аналіз програмних аналогів

На сьогоднішній день на ринку програмного забезпечення присутні декілька підходів до автоматизації обліку шкільного розкладу. Проведемо аналіз основних рішень, визначивши їх переваги та недоліки, щоб обґрунтувати доцільність розробки власного локального додатку.

Глобальні хмарні платформи освітнього менеджменту є комплексними системами, де розклад виступає лише одним із модулів поруч із електронними журналами. Перевагою таких систем є доступ через браузер, а недоліком – висока вартість підписки та абсолютна залежність від стабільності підключення до мережі Інтернет

Західні спеціалізовані десктопні пакети є потужними комерційними генераторами розкладу з математичними алгоритмами. Вони мають гарний автоматичний алгоритм розрахунку, але ліцензія є занадто дорогою, а інтерфейс потребує тривалого навчання користувача та не враховує специфіку вітчизняного обліку.

Локальні індивідуальні інформаційні системи представляють собою автономний десктопний додаток. Головними перевагами тут виступають повна незалежність від Інтернету, максимальна швидкодія, простота інтерфейсу та відсутність абонентської плати. Для наочності проведемо порівняльний аналіз за ключовими критеріями у таблиці 1.2.

Таблиця 1.2 - Порівняльний аналіз рішень для обліку шкільного розкладу

Критерій порівняння	**Хмарні платформи («Нові Знання»)**	**Західні пакети («aSc TimeTables»)**	**Проектована локальна ІС**
Тип архітектури	Web (Хмара)	Desktop (Локально)	Desktop (Локально)
Залежність від Інтернету	Абсолютна (не працює без мережі)	Відсутня	Відсутня (повна автономність)
Вартість впровадження	Постійна абонплата	Висока вартість ліцензії	Безкоштовно (весь код є)
Складність освоєння	Середня	Висока (потребує курсів)	Низька (інтуїтивний інтерфейс)
Автоматичний контроль колізій	Присутній	Присутній	Присутній (реалізується в коді)

На основі проведеного аналізу аналогів можна зробити висновок, що розробка локального, швидкого та незалежного від зовнішніх факторів додатку для оперативного обліку розкладу є виправданим кроком для більшості середніх шкіл.

1.4 Обґрунтування вибору технологічного стеку

Для реалізації локальної інформаційної системи обліку розкладу занять було обрано середовище розробки RAD Studio на базі мови програмування Object Pascal. Даний вибір обумовлений високою швидкістю візуальної розробки додатків. Наявність потужної бібліотеки візуальних компонентів дозволяє

створювати складні графічні інтерфейси, мінімізуючи рутинне написання коду для відображення елементів екранних форм.

RAD Studio компилює вихідний код безпосередньо в машинний код операційної системи Windows, що гарантує максимальну швидкодію додатку навіть на застарілих персональних комп'ютерах. Програма не потребує важких віртуальних машин або додаткових сторонніх фреймворків. Потужна підсистема роботи з базами даних містить універсальні компоненти, які забезпечують стабільне та швидке з'єднання з реляційними сховищами інформації.

В якості початкового сховища даних використовується реляційна структура Microsoft Access [5, 6, 7]. Вона є оптимальним вибором для локальних систем малого та середнього масштабу через відсутність необхідності розгортання та адміністрування важких серверних СУБД. Двигун баз даних повністю покриває потреби школи за обсягом даних, забезпечуючи при цьому підтримку повноцінного синтаксису SQL запитів та механізмів підтримання цілісності даних на рівні зв'язків між таблицями.

На основі проведеного системного аналізу предметної області сформуємо перелік функціональних та інформаційних вимог до проектованої системи, що виступає в ролі локального Технічного завдання.

1.4.1 Вимоги до бази даних

База даних системи повинна мати чітку реляційну структуру, зведену до простору нормальних форм для унеможливлення виникнення аномалій дублювання та модифікації даних. До структури висуваються такі вимоги:

– чітке виділення базових сутностей та наявність окремих таблиць-довідників для вчителів, предметів та щоденних реєстрів розкладу;

– підтримка цілісності відносин, де всі зв'язки між таблицями повинні базуватися на суворій відповідності первинних та зовнішніх ключів;

– унікальність та індексація полів для забезпечення високої швидкості пошуку інформації та виконання SQL-запитів.

Зазначені критерії дозволять створити надійний фундамент для збереження інформації освітнього процесу.

1.4.2 Функціональні вимоги до ПЗ

Програмний комплекс, що розробляється в середовищі RAD Studio, повинен забезпечувати повний цикл інтерактивного управління інформаційними масивами:

– наявність функцій введення та модифікації даних через зручні екранні інтерфейси для додавання нових викладачів та редагування дисциплін;

– інтеграція спеціальних засобів навігації, пошуку та фільтрації даних, що дозволяють оператору миттєво відсортувати розклад для конкретного класу;

– автоматичний контроль ресурсних колізій на рівні програмного коду для блокування ситуацій призначення вчителя на один час у два різних класи.

Реалізація цих функцій дозволить мінімізувати вплив людського фактора під час планування занять.

1.4.3 Вимоги до інтерфейсу та ергономіки

Ергономіка інтерфейсу клієнтського додатку є критичним фактором, оскільки кінцеві користувачі часто мають базовий рівень комп'ютерної грамотності. До інтерфейсу висуваються вимоги інтуїтивності візуальної схеми, де графічні елементи керування у вигляді таблиць та навігаторів мають бути згруповані логічно. Користувач повинен бачити розклад на тиждень в одному зручному візуальному блоці. Також необхідна валідація введення даних, яка захищає базу даних від випадкових помилок оператора за допомогою програмної перевірки повноти введення символів та аналізу часових меж безпосередньо перед фіксацією запису.

1.5 Висновки до першого розділу

У першому розділі кваліфікаційної роботи було проведено повний аналіз предметної області освітнього процесу загальноосвітньої школи. Визначено ключові обмеження та санітарні норми складання розкладу, сформовано загальну інформаційну схему автоматизованого робочого місця завуча. Шляхом порівняльного аналізу обґрунтовано доцільність створення локального додатку в середовищі RAD Studio з використанням реляційної бази даних, а також розроблено детальне технічне завдання з вимогами до інформаційної структури та інтерфейсу користувача.

РОЗДІЛ 2. РОЗРОБКА ЛОГІЧНОЇ СТРУКТУРИ БАЗИ ДАНИХ

2.1 Інфологічне проектування предметної області

Першим і фундаментальним етапом проектування бази даних автоматизованої системи обліку шкільного розкладу є побудова її інфологічної моделі. Інфологічна модель виступає як високорівневе відображення реальних об'єктів предметної області, їхніх властивостей та взаємозв'язків, яке повністю абстраговане від специфіки конкретної СУБД чи фізичних обмежень збереження файлів.

Головна мета цього етапу – формалізувати інформаційні потреби навчального закладу у вигляді чіткої структури даних, яка є зрозумілою як розробнику, так і майбутньому користувачу системи. Для побудови інфологічної моделі використовується класичний методологічний апарат – модель Сутність–Зв’язок (ER-модель) [8, 9, 10]. Базовими конструктивними елементами такої моделі виступають сутності, атрибути та логічні зв'язки між ними.

У процесі системного аналізу предметної області обліку навчального процесу в загальноосвітній школі було виділено три основні сутності: Вчителі, Предмети та Розклад. Сутність Вчителі відображає кадровий склад педагогічного колективу навчального закладу, який безпосередньо задіяний у проведенні уроків. Сутність Предмети містить перелік навчальних дисциплін, що затверджені навчальним планом школи для викладання. Сутність Розклад виступає як інтегруючий компонент освітнього процесу, який фіксує проведення конкретного предмету конкретним викладачем у визначений час.

Кожна з виділених сутностей володіє власним унікальним набором атрибутів, необхідних для повноцінного обліку. Сутність Вчителі містить цифровий ідентифікатор особи та текстове поле ПІБ викладача. Сутність Предмети складається з унікального коду дисципліни, її назви та посилання на ідентифікатор закріпленого вчителя. Інформаційні масиви сутності Розклад

структуровані за днями тижня, утворюючи відповідні щоденні реєстри, атрибутивний склад яких включає первинний ключ запису, текстовий діапазон часу проведення уроку, а також ідентифікатори та текстові назви відповідних дисциплін і викладачів.

Логічні зв'язки між виділеними сутностями концептуальної моделі мають реляційну природу типу один-до-багатьох (1:M). Один конкретний вчитель із довідника може вести багато різних предметів, тому зв'язок між ними класифікується як 1:M. Один предмет із довідника дисциплін може повторюватися у сітці розкладу багато разів протягом тижня на різних уроках, що утворює зв'язок типу 1:M між предметами та щоденними таблицями розкладу занять. Побудована модель повністю задовольняє базові інформаційні потреби, проте містить ознаки структурної надликовості, аналіз якої буде виконано у наступних підрозділах.

2.2 Даталогічне проектування БД

Після затвердження високорівневої інфологічної моделі наступним логічним етапом є даталогічне проектування. Даталогічне проектування – це процес відображення концептуальної схеми даних на мову конкретної моделі, яка підтримується обраною СУБД Microsoft Access [11, 12]. На цьому етапі відбувається фізична типізація полів: кожному атрибуту присвоюється конкретний системний тип даних, задаються обмеження на довжину полів та визначаються правила перевірки значень.

Для інформаційної системи розкладу занять у СУБД було створено 7 основних реляційних таблиць. До них відносяться базові довідники вчителів та предметів, а також п'ять щоденних реєстрів розкладу по днях тижня від понеділка до п'ятниці. Представимо структуру та фізичні параметри кожної проектованої таблиці у вигляді специфікацій.

2.2.1 Специфікація довідників

Таблиця Вчителі є базовим батьківським довідником системи. Фізична структура полів представленої таблиці наведена у таблиці 2.1.

Таблиця 2.1 – Фізична структура таблиці «Вчителі»

Ім'я поля	**Тип даних у СУБД**	**Розмір поля**	**Функціональне призначення**
ID_Вчителя	Автонумерація	Long Integer	Первинний ключ таблиці
Вчитель	Короткий текст	100 символів	ПІБ викладача

Візуальне представлення розробленої структури довідника в режимі проектування СУБД наведено на рисунку 2.1.

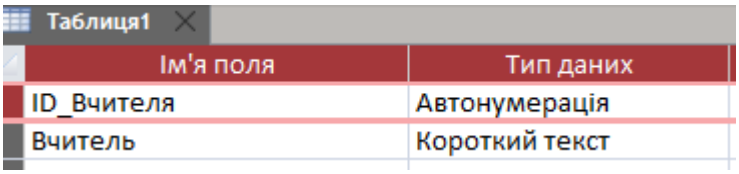

Рисунок 2.1 – Таблиця «Вчителі» у режимі конструктора MS Access

Таблиця Предмети містить посилання на довідник вчителів для забезпечення логічного зв'язку та закріплення дисциплін за конкретними авторами. Фізична структура полів представлена у таблиці 2.2.

Таблиця 2.2 – Фізична структура таблиці «Предмети»

Ім'я поля	**Тип даних у СУБД**	**Розмір поля**	**Функціональне призначення**
1	**2**	**3**	**4**
Предмет_ID	Автонумерація	Long Integer	Первинний ключ таблиці

Продовження таблиці 2.1

1	2	3	4
Назва_Предмету	Короткий текст	150 символів	Повна назва дисципліни
ID_Вчителя	Числовий	Long Integer	Зовнішній ключ (посилання на вчителя)

Фізичний вигляд проектованої структури довідника дисциплін у конструкторі СУБД відображено на рисунку 2.2.

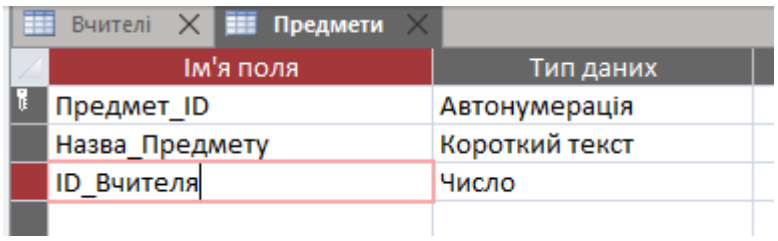

Рисунок 2.2 – Таблиця «Предмети» у режимі конструктора MS Access

Представлені таблиці-довідники складають інформаційну основу системи та заповнюються оператором у першу чергу перед початком формування розкладу.

Після проектування фізичної структури довідників у режимі конструктора, було проведено їх первинне тестове наповнення масивами інформації навчального закладу. Це необхідно для верифікації коректності визначених типів даних, перевірки працездатності індексації полів та забезпечення подальшого коректного імпорту значений у клієнтський додаток.

Зразки сформованих тестових масивів інформації для базових батьківських таблиць-довідників «Вчителі» та «Предмети» представлено у таблицях 2.3 та 2.4 відповідно.

Таблиця 2.3 – Тестове наповнення реляційної таблиці-довідника «Вчителі»

ID_Вчителя (Первинний ключ)	**Вчитель (ПІБ викладача)**
1	Шевченко Т.Г.
2	Коваль А.П.

Таблиця 2.4 – Тестове наповнення реляційної таблиці-довідника «Предмети»

Предмет_ID (Первинний ключ)	**Назва_Предмету (Дисципліна)**	**ID_Вчителя (Зовнішній ключ)**
1	Інформатика	1
2	Математика	2

Наповнення здійснювалося з суворим дотриманням умов унікальності первинних ключів. Наявність заповнених довідників дозволяє повністю уникнути дублювання текстової інформації при формуванні щоденних реєстрів, оскільки дочірні таблиці днів тижня оперують виключно числовими посиланнями на дані записи, що забезпечує логічну цілісність бази даних.

2.2.2 Специфікація таблиць розкладу

Реєстри розкладу по днях тижня мають повністю ідентичну структуру полів. Вони виступають як дочірні таблиці, що акумулюють інформацію з перших двох довідників. Узагальнена структура щоденної таблиці розкладу на прикладі таблиці Понеділок наведена у таблиці 2.5.

Таблиця 2.5 – Фізична структура щоденної таблиці розкладу «Понеділок»

Ім'я поля	**Тип даних у СУБД**	**Розмір поля**	**Функціональне призначення**
ID_Понеділка	Автонумерація	Long Integer	Первинний ключ запису уроку
Час	Короткий текст	30 символів	Часові межі проведення уроку
ID_Предмету	Числовий	Long Integer	Зовнішній ключ (посилання на предмет)
Предмет	Короткий текст	150 символів	Текстова назва дисципліни
ID_Викладач	Числовий	Long Integer	Зовнішній ключ (посилання на вчителя)
Викладач	Короткий текст	100 символів	Текстове ПІБ вчителя

Сформований повний перелік об'єктів реляційного ядра у бічній навігаційній панелі бази даних наведено на рисунку 2.3.

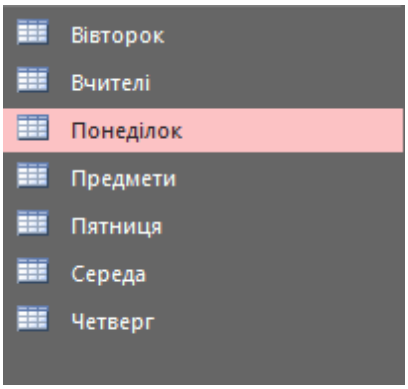

Рисунок 2.3 – Відображення структури таблиць бази даних у навігаційній панелі MS Access

Фізичне розгортання даної структури полів у середовищі Microsoft Access за допомогою конструктора таблиць дозволило створити стійке сховище даних з однозначною ідентифікацією кожного рядка.

2.3 Логічна схема БД та нормалізація

Фінальним етапом проектування інформаційного забезпечення системи є побудова логічної схеми бази даних та її верифікація за допомогою математичного апарату теорії нормалізації реляційних відносин. Логічна схема визначає остаточний набір таблиць, їхніх полів та зв'язків з урахуванням усіх обмежень цілісності СУБД.

Процес нормалізації полягає в послідовному розбитті великих або некоректно спроектованих таблиць на менші структури, що дозволяє повністю ліквідувати так звані аномалії бази даних та усунути дублювання інформації на жорсткому диску [13, 14, 15]. Проведемо аналіз та покроковий процес нормалізації розробленої бази даних шкільного розкладу занять.

2.3.1 Перша нормальна форма (1НФ)

Відношення знаходиться у Першій нормальній формі тоді і тільки тоді, коли кожен її атрибут є атомарним, а сама таблиця не містить повторюваних груп полів або масивів. Усі таблиці проектованої бази даних задовольняють вимогам 1НФ. Кожна клітинка містить лише одне значення, а всі рядки унікальні завдяки використанню первинних ключів автонумерації.

2.3.2 Друга нормальна форма (2НФ)

Відношення знаходиться у Другій нормальній формі, якщо воно задовольняє вимогам 1НФ і кожен його неключовий атрибут функціонально повною мірою залежить від усього первинного ключа, а не від його частини.

Оскільки у розробленій структурі всі первинні ключі є простими і складаються з одного поля автонумерації, то часткова залежність від ключа фізично неможлива. Відповідно, всі таблиці автоматично знаходяться у 2НФ.

2.3.3 Третя нормальна форма (3НФ)

Відношення знаходиться у Третій нормальній формі, якщо воно задовольняє вимогам 2НФ і жоден із його неключових атрибутів не перебуває в транзитивній функціональній залежності від первинного ключа. Це означає, що значення будького поля, яке не є ключем, не повинно залежати від іншого неключового поля.

Аналіз початкових щоденних таблиць розкладу виявив порушення вимог 3НФ. Поле текстової назви предмету прямо залежить від неключового поля цифрового ідентифікатора предмету, утворюючи транзитивну залежність. Аналогічно, поле викладача транзитивно залежить від поля його ідентифікатора. Така структура утворює надлишковість даних на жорсткому диску, що суперечить правилам третьої нормальної форми.

Для приведення логічної структури до вимог третьої нормальної форми та оптимізації транзакційного навантаження було застосовано комбінований підхід із частковою денормалізацією даних на рівні щоденних реєстрів розкладу. Текстові поля назв дисциплін та ПІБ викладачів були збережені безпосередньо у структурі щоденних таблиць спільно з відповідними числовими зовнішніми ключами (ID_Предмету, ID_Викладач).

Таке інженерне рішення обумовлене специфікою локальної архітектури СУБД Microsoft Access та OLE DB провайдера Jet. Наявність дубльованих текстових полів у дочірніх таблицях дозволяє клієнтському додатку RAD Studio Delphi 12 здійснювати миттєву візуалізацію даних у сітках DBGrid без необхідності виконання складних низькорівневих операцій об'єднання таблиць (JOIN) при кожному оновленні екрана. Логічна цілісність та виключення аномалій модифікації при цьому повністю забезпечуються на програмному рівні

за допомогою розроблених обробників подій бізнес-логики, які автоматично синхронізують ID з довідниками. Остаточна схема зв'язків між таблицями представлена на рисунку 2.4.

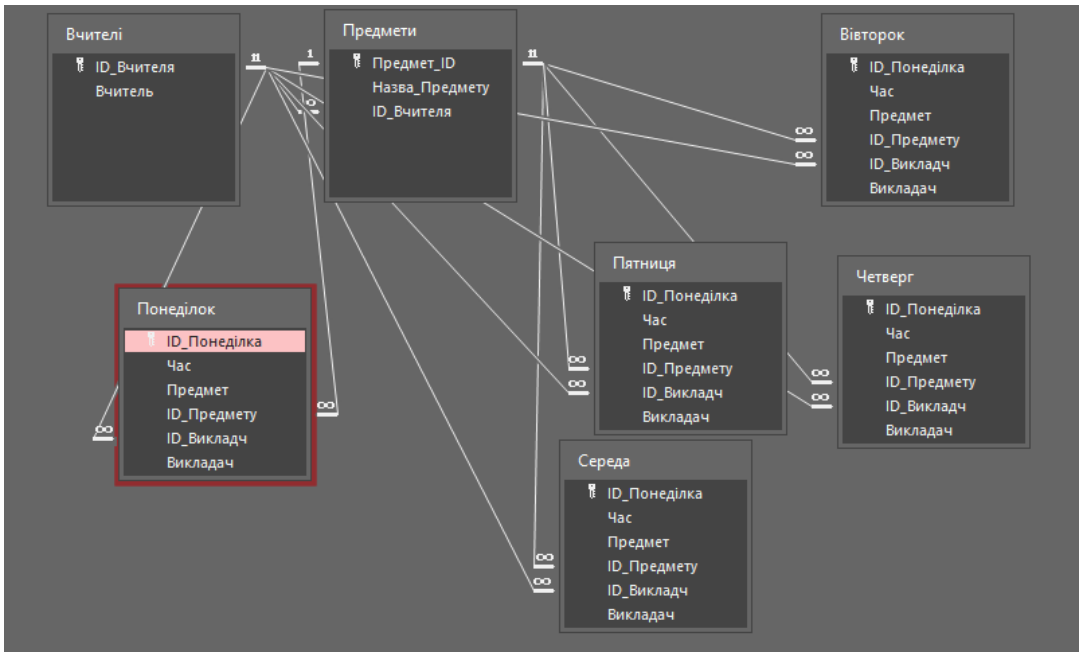

Рисунок 2.4 – Логічна схема зв'язків реляційної бази даних у СУБД MS Access

Побудована схема зв'язків забезпечує каскадне оновлення та видалення даних між усіма таблицями логічної моделі.

2.4 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було повністю розроблено інформаційне та логічне забезпечення системи. На основі аналізу предметної області побудовано інфологічну модель «Сутність–Зв’язок» та виділено ключові об'єкти обліку. Виконано даталогічне проектування з розробкою специфікацій полів для семи таблиць у СУБД Microsoft Access. Завдяки застосуванню математичного апарату теорії нормалізації проведено рефакторинг реляційних таблиць та приведено логічну схему бази даних до вимог третьої нормальної форми, що дозволило усунути надлишковість даних та гарантувати інформаційну цілісність системи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОБҐРУНТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

3.1 Архітектура додатка та розробка інтерфейсу

Практична реалізація інформаційної системи обліку та ведення шкільного розкладу занять виконана у сучасному об'єктно-орієнтованому середовищі розробки RAD Studio (Delphi 12) [16, 17, 18] з використанням бібліотеки візуальних компонентів VCL (Visual Component Library) [19].

Архітектура розробленого додатка базується на файл-серверній (настільній) технології, де роль локального автономного сховища виконує файл реляційної бази даних Microsoft Access, а клієнтська частина повністю бере на себе функції відображення графічного інтерфейсу та обробки бізнес-логіки. Специфіка роботи адміністрації загальноосвітніх навчальних закладів вимагає максимальної автономності та незалежності від зовнішніх факторів, тому розроблена локальна система здатна повноцінно функціонувати в умовах відсутності стабільного підключення до мережі Інтернет.

Взаємодія між графічними екранними формами користувача та таблицями Microsoft Access реалізована за допомогою технології Microsoft ADO (ActiveX Data Objects) [20]. Цей технологічний стек забезпечує високу швидкість доступу до реляційних масивів через універсальний системний провайдер Microsoft Jet OLE DB Provider. Безпосереднє створення програмного комплексу відбувалося у декілька етапів.

На першому етапі за допомогою інструментарію СУБД Microsoft Access було фізично створено сім реляційних таблиць: два довідники та п'ять щоденних реєстрів розкладу. Для забезпечення цілісності даних між створеними таблицями було налаштовано каскадні зв'язки типу «один-до-багатьох» (рис. 3.1).

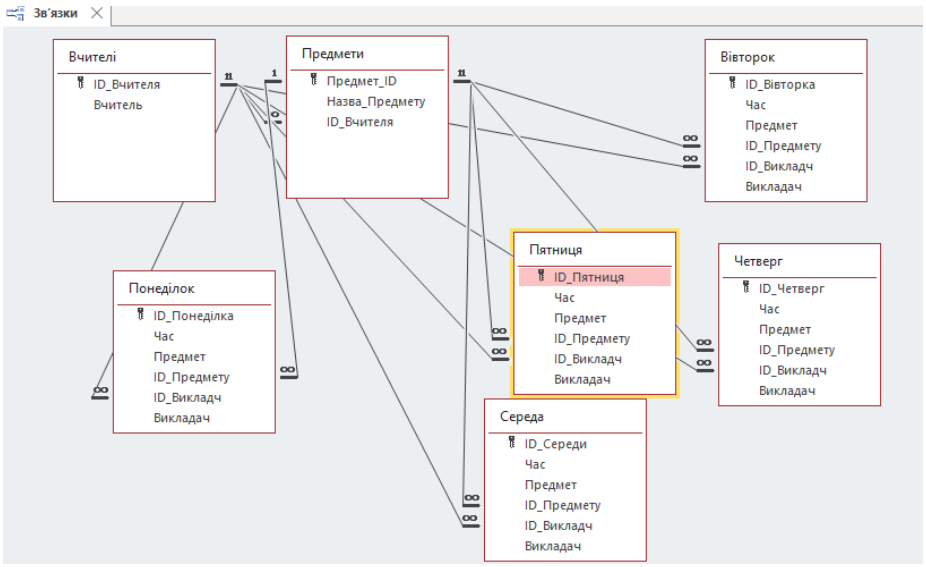

Рисунок 3.1 – Процес створення таблиць та налаштування реляційних зв'язків у СУБД MS Access

На другому етапі розпочалася розробка клієнтського додатка. Для забезпечення доступу програми до файлу бази даних на головну форму проекту було додано базовий невізуальний компонент TADOConnection. Через вбудований майстер підключення було обрано системний провайдер «Microsoft Jet 4.0 OLE DB Provider» та вказано шлях до локального файлу сховища. Властивість LoginPrompt була переведена у стан False для вимкнення системного запиту пароля при кожному запуску (рис. 3.2).

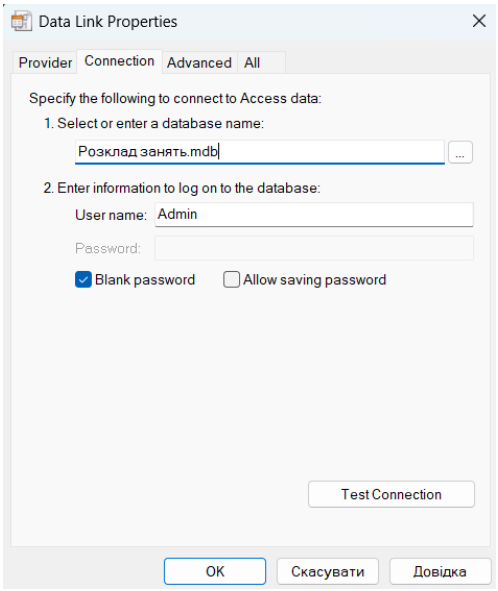

Рисунок 3.2 – Налаштування рядка підключення ConnectionString до бази даних

Для прямої взаємодії з кожною із семи створених таблиць на форму було додано компоненти TADOTable, які через властивість TableName були жорстко прив'язані до відповідних таблиць сховища. Передача масивів інформації до візуальних сіток TDBGrid реалізована через проміжні компоненти TDataSource, що дозволило повністю відділити бізнес-логіку програми від її візуального представлення (рис. 3.3).

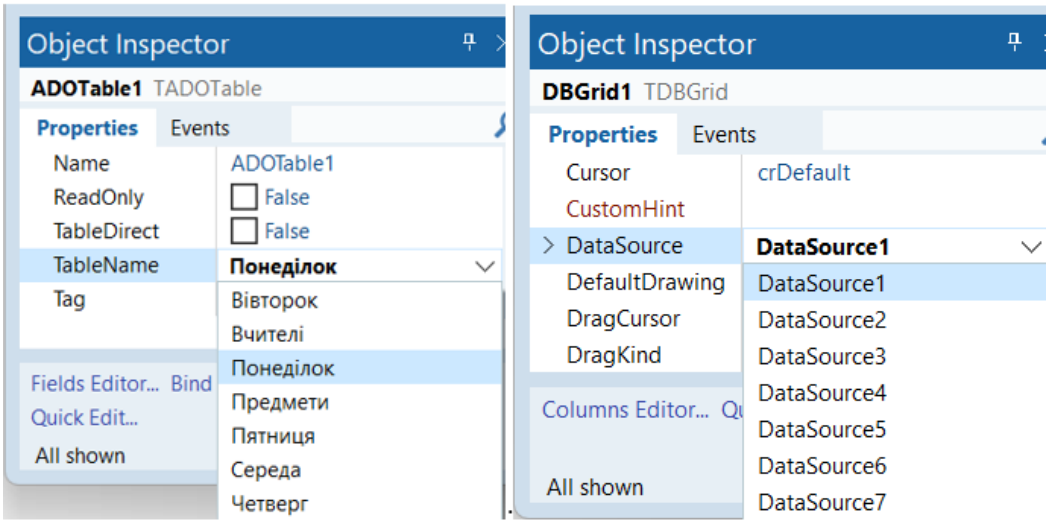

Рисунок 3.3 – Ініціалізація компонентів зв'язку та прив'язка таблиць до інтерфейсу

Для фінального налаштування ергономіки відображення даних було використано вбудований редактор колонок (Columns Editor). За допомогою функції «Add All Fields» до сіток було завантажено всі поля бази даних, після чого технічні поля ідентифікаторів були програмно приховані, а видимим стовпцям задано зрозумілі українські заголовки (рис. 3.4).

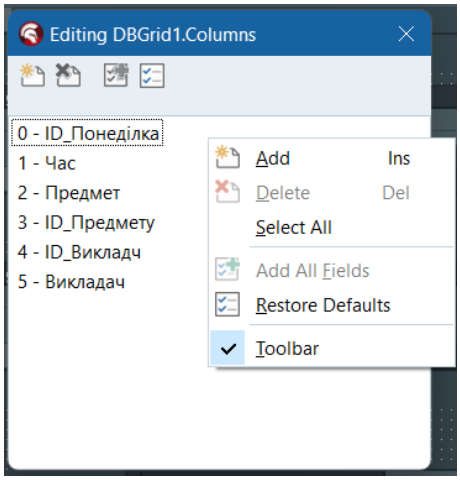

Рисунок 3.4 – Налаштування відображення стовпців розкладу через Columns Editor

Завдяки використанню компонентної архітектури інтегрованого пакета RAD Studio та послідовному налаштуванню ланцюжка об'єктів, програма відкриває лише одну системну сесію з файлом СУБД, що суттєво мінімізує навантаження на процесор та заощаджує оперативну пам'ять персонального комп'ютера.

Детальна карта логічного взаємозв'язку всіх семи налаштованих інформаційних каналів системи наведена у таблиці 3.1.

Таблиця 3.1 – Карта взаємозв'язку та налаштування компонентів інформаційного ядра системи

Назва каналу даних	**Компонент TADOTable**	**Компонент TDataSource**	**Компонент TDBGrid**	**Зв'язана таблиця в БД**
1	**2**	**3**	**4**	**5**
Розклад на понеділок	ADOTable1	DataSource1	DBGrid1	Понеділок
Розклад на вівторок	ADOTable2	DataSource2	DBGrid2	Вівторок
Розклад на середу	ADOTable3	DataSource3	DBGrid3	Середа

Продовження таблиці 3.1

1	2	3	4	5
Розклад на четвер	ADOTable4	DataSource4	DBGrid4	Четвер
Розклад на п’ятницю	ADOTable5	DataSource5	DBGrid5	П’ятниця
Довідник «Вчителі»	ADOTable6	DataSource6	DBGrid6	Вчителі
Довідник «Предмети»	ADOTable7	DataSource7	DBGrid7	Предмети

Завдяки використанню компонентної архітектури інтегрованого пакету RAD Studio, відображення та збереження первинних текстових даних здійснюється в автоматичному режимі. Оскільки всі компоненти TADOTable підключені до єдиного об'єкта з'єднання ADOConnection1, програма відкриває лише одну системну сесію з файлом СУБД Microsoft Access. Це дозволяє суттєво мінімізувати навантаження на процесор та заощаджувати оперативну пам'ять персонального комп'ютера, забезпечуючи високу стабільність функціонування автоматизованого робочого місця адміністратора.

3.2 Ергономіка та модернізація графічних форм

Стандартний інтерфейс компонентів VCL за замовчуванням має застарілий вигляд і створює зайве візуальне навантаження на оператора через монохромну палітру та відображення великої кількості службових даних. Для покращення ергономіки, зручності роботи та приведення додатка до сучасних стандартів UI/UX було проведено візуальну модернізацію графічної форми в середовищі Delphi 12.

Процес налаштування та оптимізації інтерфейсу користувача включав кілька кроків:

– Зміна теми оформлення. Через інженерні параметри проєкту (Project -> Options -> Application -> Appearance) стандартне сіре оформлення було замінено на сучасний темний графічний стиль. Графічне вікно конфігурації та вибору доступних візуальних стилів VCL Styles у середовищі розробки RAD Studio представлено на рисунку 3.5. Це дозволило автоматично стилізувати всі візуальні елементи (форми, кнопки, межі таблиць), зробивши інтерфейс комфортнішим для тривалої роботи оператора.

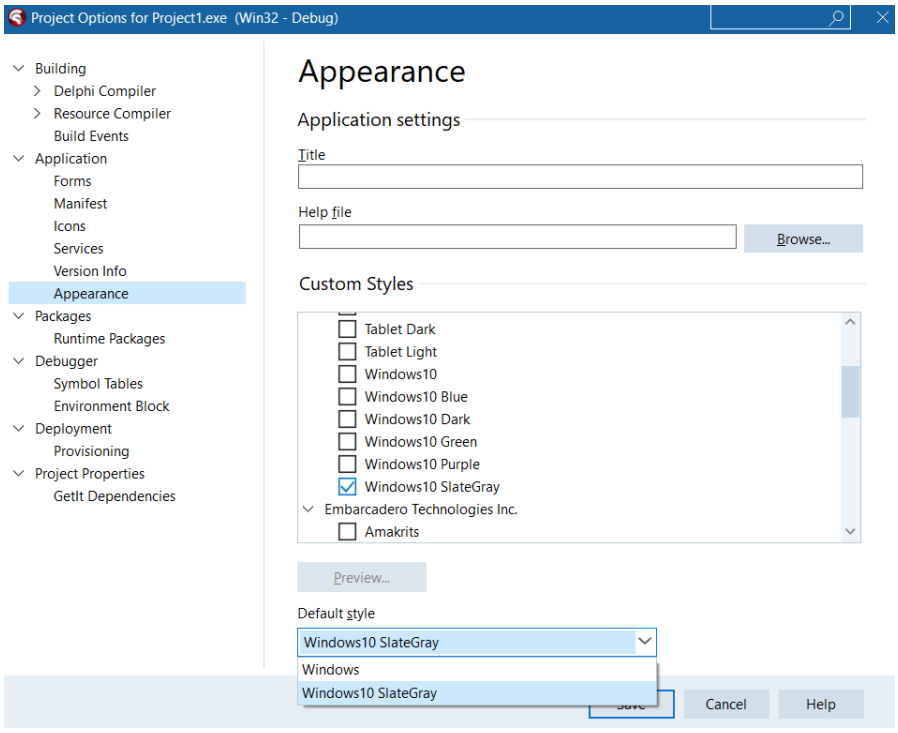

Рисунок 3.5 – Вікно конфігурації візуальних стилів VCL Styles у RAD Studio

– Рефакторинг відображення таблиць DBGrid. За допомогою вбудованого редактора колонок (Columns Editor) для кожної візуальної сітки було приховано технічні числові поля системних ідентифікаторів (ID_Понеділка, ID_Предмету, ID_Викладач), перевівши їхню властивість Visible у стан False. Загальний вигляд графічної форми Form1 на етапі проектування інтерфейсу (Design-time) із відображенням сітки розташування елементів, невізуальних компонентів зв'язку

та налаштованих колонок наведено на рисунку 3.6. Для інформаційних стовпців часу та назв було змінено властивість Title.Caption на зрозумілі українські заголовки: «Час уроку», «Навчальна дисципліна» та «ПІБ викладача». Також активовано параметр dgColumnResize для динамічного масштабування ширини стовпців під довжину тексту.

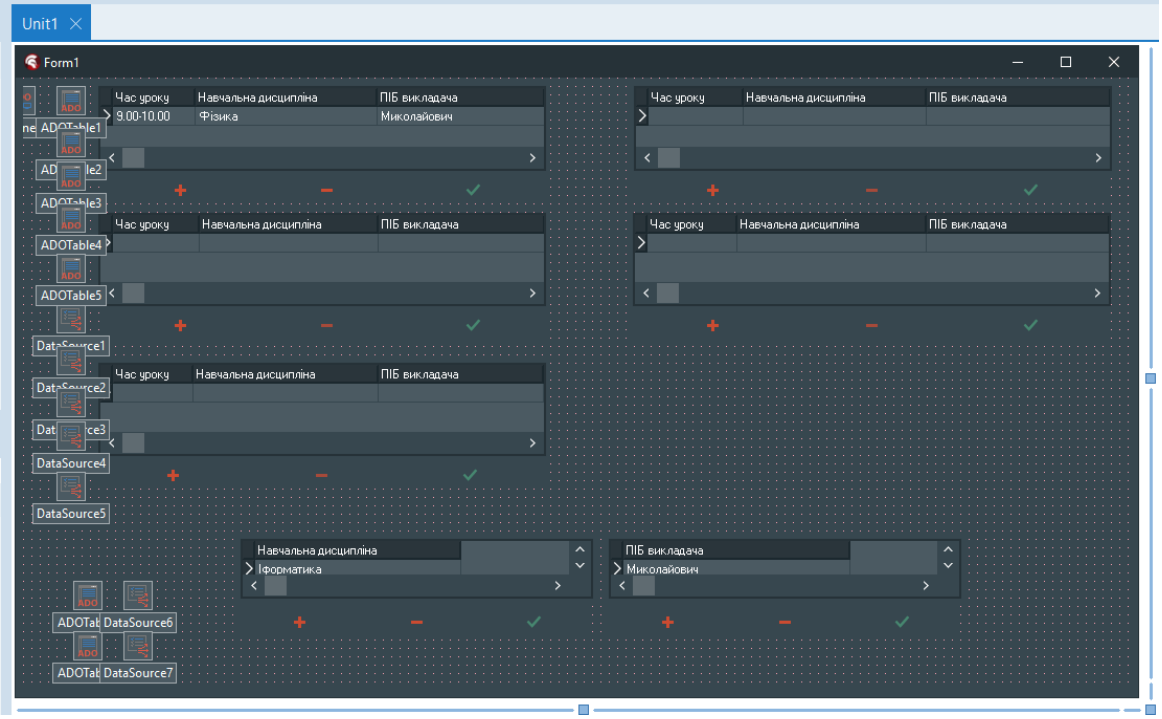

Рисунок 3.6 – Графічна форма проекту в режимі розробки інтерфейсу (Design-time)

– Оптимізація панелей керування. Компоненти TDBNavigator за замовчуванням містили десять кнопок, більшість з яких дублювали функції миші й захаращували екран. Через властивість VisibleButtons було приховано всі зайві елементи, окрім трьох основних інструментів для редагування: додавання нового рядка (nbInsert), видалення запису (nbDelete) та фізичного збереження змін у базу даних (nbPost).

Впровадження цих рішень дозволило очистити робочий простір від зайвої технічної інформації. Інтерфейс програми став інтуїтивно зрозумілим для

шкільного персоналу (завуча), що підвищує швидкість введення даних та знижує ризик виникнення механічних помилок при заповненні розкладу занять.

3.3 Реалізація модулів бізнес-логіки

Для забезпечення стабільного функціонування інформаційної системи та автоматизації операцій керування шкільним розкладом, базових візуальних налаштувань компонентів недостатньо. Логіка взаємодії елементів, автоматизація процесів та обробка дій оператора реалізована програмно мовою Object Pascal у модулі Unit1.pas.

Першою інженерною задачею, яка вирішується на рівні програмного коду, є динамічне підключення до реляційного сховища даних при запуску додатка. Оскільки початкові налаштування компонента ADOConnection1 містять абсолютний шлях до локального файлу на диску комп'ютера розробника, при перенесенні скомпільованої програми на інший персональний комп'ютер виникне критична помилка з'єднання.

Для усунення цього недоліку в обробник події створення головної форми OnCreate додано програмний код, який за допомогою функції ExtractFilePath автоматично визначає поточний робочий каталог розташування програми й самостійно формує коректну строку підключення ConnectionString до файлу «Розклад занять.mdb». Програмний код цієї процедури наведений у лістингу 3.1.

Лістинг 3.1 – Програмний код динамічної ініціалізації з'єднання з базою даних

```
procedure TForm1.FormCreate(Sender: TObject);
var
  DbPath: string;
begin
  DbPath   :=   ExtractFilePath(Application.ExeName)   +
'Розклад занять.mdb';
  ADOConnection1.Connected := False;
  ADOConnection1.ConnectionString :=
    'Provider=Microsoft.Jet.OLEDB.4.0;' +
    'Data Source=' + DbPath + ';' +
```

```
        'Persist Security Info=False';
      try
        ADOConnection1.Connected := True;
        ADOTable1.Active := True;
        ADOTable2.Active := True;
        ADOTable3.Active := True;
        ADOTable4.Active := True;
        ADOTable5.Active := True;
        ADOTable6.Active := True;
        ADOTable7.Active := True;
      except
        on E: Exception do
          ShowMessage('Помилка ініціалізації бази даних: ' +
E.Message);
      end;
end;
```

Зазначений алгоритм використовує захищений блок try ... except. Якщо файл бази даних буде відсутній у папці з програмою, система не завершить роботу аварійно, а виведе інформаційне вікно з попередженням для оператора. Після успішного встановлення зв'язку код програмно активує всі сім таблиць TADOTable, що забезпечує завантаження та відображення даних в інтерфейсі користувача.

Детальний покроковий аналіз логіки роботи алгоритму динамічної ініціалізації з'єднання (лістинг 3.1) включає такі послідовні етапи:

1. Оголошення локальної змінної DbPath рядкового типу (string), яка призначена для тимчасового збереження результуючого фізичного шляху до файлу бази даних.

2. Виклик стандартної системної функції ExtractFilePath(Application.ExeName), яка динамічно обчислює та повертає повний шлях до каталогу жорсткого диска, з якого було запущено поточний виконуваний файл додатка. Це дозволяє позбутися жорсткої адресації (абсолютних шляхів).

3. Конкатенація (об'єднання) отриманого базового шляху з константним рядком назви реляційного файлу сховища 'Розклад занять.mdb'.

4. Примусове переведення властивості ADOConnection1.Connected у стан False для коректного скидання та деактивації можливих попередніх незавершених сесій зв'язку.

5. Динамічне формування керуючого рядка ConnectionString із чітким зазначенням типу системного провайдера (Provider=Microsoft.Jet.OLEDB.4.0) та обчисленого джерела даних (Data Source).

6. Входження у захищений блок обробки виключних ситуацій try ... except. У разі виникнення помилок (наприклад, за відсутності файлу) керування передається в блок except, який перехоплює клас виключення Exception і виводить модальне вікно ShowMessage з точним описом проблеми, захищаючи додаток від аварійного крашу.

7. При штатному виконанні операції з'єднання, код послідовно переводить властивість Active у стан True для всіх семи об'єктів класу TADOTable (від ADOTable1 до ADOTable7), що ініціює автоматичне завантаження даних у буфер оперативної пам'яті клієнтського додатка.

Другою задачею є організація контролю за введенням даних для захисту реляційної структури від появи некоректних записів. При ручному заповненні сітки розкладу існує ризик того, що оператор випадково спробує зберегти рядок із пустими ключовими полями, що призведе до збоїв у логіці відображення занять.

Для запобігання виникненню подібних помилок у програмний модуль було інтегровано алгоритм валідації полів, який автоматично спрацьовує під час виклику події BeforePost безпосередньо перед фізичним записом сформованого рядка у файл бази даних. Програмний код модуля валідації наведено у лістингу 3.2.

Лістинг 3.2 – Програмний код перевірки заповнення полів розкладу перед збереженням

```
procedure TForm1.ADOTable1BeforePost(DataSet: TDataSet);
var
  FField: TField;
```

```
     begin
         if  Trim(DataSet.FieldByName('Час').AsString)  =  ''
then
       begin
         ShowMessage('Помилка:  Поле  "Час"  не  може  бути
порожнім!');
         Abort;
       end;

       FField := DataSet.FindField('ID_Предмету');
       if FField <> nil then
       begin
         if (FField.IsNull) or (FField.AsInteger = 0) then
           FField.AsInteger := 1;
       end;

       FField := DataSet.FindField('ID_Викладач');

       if FField <> nil then
       begin
         if (FField.IsNull) or (FField.AsInteger = 0) then
           FField.AsInteger := 1;
       end;
end;
```

Якщо під час перевірки виявляється, що текстове поле «Час» залишилося порожнім, програма за допомогою команди Abort скасовує операцію відправки даних до файлу Microsoft Access. При цьому транзакція переривається, на екран виводиться попереджувальне модальне вікно, а фокус введення повертається оператору для виправлення помилки.

Логічна структура та етапи виконання алгоритму превентивної валідації полів (лістинг 3.2) базуються на таких конструкціях мови Object Pascal:

1. Перехоплення події BeforePost на рівні набору даних DataSet, яка автоматично генерується ядром середовища розробки безпосередньо перед фізичним записом сформованого транзакційного рядка на диск.

2. Звернення до цільового індексованого поля за допомогою методу DataSet.FieldByName('Час') та його приведення до рядкового типу через властивість AsString.

3. Перевірка умовного оператора if ... then на предмет наявності порожнього значення ''.

4. У разі істинності умови (поле не заповнене), програма активує модальне діалогове вікно попередження оператора ShowMessage.

5. Виклик стандартної інженерної команди переривання трансляції даних Abort. Ця команда генерує безшумне виключення на рівні VCL, повністю зупиняє операцію відправки пакету даних до СУБД, скасовує дію користувача та примусово повертає курсор у фокус редагування комірки.

Третьою важливою задачею є реалізація механізму оперативного пошуку та фільтрації розкладу. Оскільки загальні сітки відображають усі уроки тижня разом, диспетчеру необхідний інструмент для миттєвого відбору занять за конкретною дисципліною.

Для реалізації цієї функції прикладний модуль використовує вбудовану властивість Filter об'єктів наборів даних. Програмний код, який забезпечує динамічну фільтрацію занять за назвою предмета, наведений у лістингу 3.3.

Лістинг 3.3 – Програмний код алгоритму фільтрації та пошуку інформації у розкладі

```
procedure TForm1.EditSearchChange(Sender: TObject);
begin
  if EditSearch.Text <> '' then
  begin
    ADOTable1.Filter := 'Предмет LIKE ' + QuotedStr('%' +
EditSearch.Text + '%');
    ADOTable1.Filtered := True;
  end
  else
  begin
    ADOTable1.Filtered := False;
  end;
end;
```

Робота цього алгоритму побудована на застосуванні спеціального оператора порівняння логічних рядків LIKE спільно зі службовими символами відсотків, що дозволяє оператору здійснювати гнучкий нечіткий пошук. При введенні перших літер назви дисципліни у текстове поле EditSearch, система автоматично відбирає всі відповідні записи, приховуючи інші рядки розкладу.

Написаний комплекс програмних обробників подій повністю реалізує закладену бізнес-логіку обробки інформаційних потоків навчального закладу.

Покроковий аналіз механізму роботи алгоритму динамічної фільтрації (лістинг 3.3) відображає логіку взаємодії з властивостями компонентів:

1. Перехоплення події зміни текстового субстрату в полі введення EditSearch за допомогою системного обробника OnChange.

2. Якщо оператор ввів пошукові символи, програма динамічно формує рядок властивості Filter для об'єкта ADOTable1. Формування строки використовує функцію QuotedStr, яка автоматично екранує лапки, унеможливлюючи виникнення синтаксичних помилок SQL-запиту.

3. Використання символів відсотків '%...%' спільно з оператором порівняння рядків LIKE вказує реляційному ядру СУБД на необхідність проведення гнучкого нечіткого пошуку за підрядком (входження в будь-якій частині назви дисципліни).

4. Примусова активація логічного параметра ADOTable1.Filtered := True, що змушує компонент миттєво перебудувати індекси відображення сітки DBGrid, залишаючи на екрані лише записи, які задовольняють умову фільтру.

5. У разі очищення поля пошуку оператором, гілка else переводить параметр Filtered у стан False, що миттєво повертає таблицю у вихідний стан повного відображення інформації.

3.4 Інструкція користувача АРМ

Розроблене інструктивно-методичне керівництво призначене для швидкого освоєння інтерфейсу інформаційної системи та забезпечення безпомилкової роботи оператора (завуча навчального закладу). Запуск автоматизованого робочого місця здійснюється за допомогою виконуваного файлу Project1.exe.

Після запуску програми на екрані з'являється головне вікно додатка, виконане в інтегрованій темній графічній темі, що відображено на рисунку 3.7.

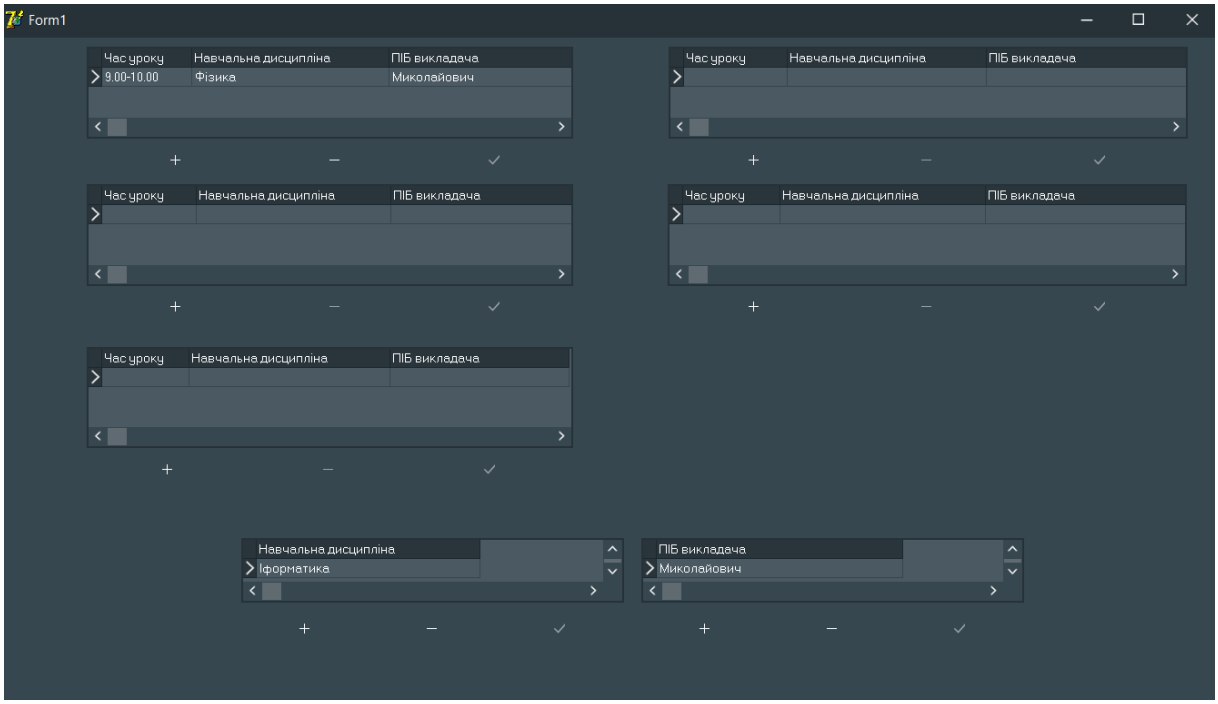

Рисунок 3.7 – Головне вікно інформаційної системи обліку шкільного розкладу

Робочий простір розділений на дві функціональні зони: верхня частина форми призначена для керування розкладом по днях тижня, а нижня містить базові довідники викладачів та навчальних дисциплін. Взаємодія з базою даних Microsoft Access автоматизована через локальні панелі навігації під кожною таблицею.

Порядок заповнення та коригування довідкової інформації:

1. Для додавання нового викладача або навчального предмета оператору необхідно перейти до відповідної нижньої таблиці форми та натиснути кнопку «+» (Додати) на панелі навігатора. У створеному порожньому рядку з клавіатури вводиться відповідне текстове значення.

2. Для видалення застарілого чи помилкового запису з довідника оператор виділяє мишкою потрібний рядок у таблиці й натискає кнопку «-» (Видалити).

3. Для остаточного збереження внесених змін або нових назв у реляційному файлі сховища обов'язково натискається кнопка з зеленою галочкою «✓» (Фіксація запису).

Порядок формування та редагування розкладу занять по днях тижня:

1. У таблиці відповідного дня тижня (наприклад, «Понеділок» чи «Вівторок») оператор ініціює створення нового рядка за допомогою кнопки «+» на потрібному навігаторі.

2. У полі «Час уроку» вказуються часові межі проведення навчального заняття.

3. У стовпцях «Навчальна дисципліна» та «ПІБ викладача» вносяться відповідні відомості про предмет і вчителя, що веде цей урок.

4. Після завершення введення даних рядок обов'язково фіксується у базі даних натисканням кнопки «✓» під цією таблицею.

Для швидкого відбору уроків за конкретним предметом оператор вводить назву дисципліни у текстове поле пошуку та активує функцію фільтрації, після чого система автоматично приховує інші записи розкладу. Очищення поля пошуку повертає таблиці у вихідний стан повного відображення даних.

3.5 Системні вимоги та розгортання ПЗ

Для забезпечення стабільного, безвідмовного та високопродуктивного функціонування розробленого автоматизованого робочого місця адміністратора шкільного розкладу, було визначено комплекс мінімальних та рекомендованих апаратних і програмних вимог до обчислювальної техніки навчального закладу.

Оскільки архітектура створеного додатка базується на ефективному використанні локальних OLE DB провайдерів та динамічному завантаженні реляційних індексів у оперативну пам'ять, програма демонструє низький рівень споживання системних ресурсів. Це дозволяє успішно експлуатувати її навіть на застарілих комп'ютерних робочих станціях, що є критично важливим для бюджетного сектору середньої освіти.

Повний перелік параметрів апаратного забезпечення та вимог до операційної системи комп'ютера завуча зведено в інженерну таблицю 3.2.

Таблиця 3.2 – Технічні вимоги до апаратно-програмного комплексу розгортання системи

Назва компонента системи	**Мінімальні технічні параметри**	**Рекомендовані технічні параметри**
1	**2**	**3**
Центральний процесор (CPU)	x86/x64 сумісний, з тактовою частотою від 1.6 ГГц	Intel Core i3 / AMD Ryzen 3 або вище (від 2.4 ГГц)
Оперативна пам'ять (RAM)	2 ГБ	4 ГБ або більше (для миттєвого кешування таблиць)
Дисковий простір (HDD/SSD)	50 МБ вільного місця (враховуючи річний ріст бази даних)	100 МБ вільного місця на твердотільному накопичувачі SSD
Графічний адаптер та екран	Роздільна здатність 1024x768, підтримка 16-бітного кольору	Роздільна здатність 1920x1080 (Full HD) для масштабних таблиць
Периферійні пристрої	Клавіатура, маніпулятор «миша»	Клавіатура, миша, лазерний принтер (для друку розкладу)
Операційна система (ОС)	Microsoft Windows 7 (з пакетом оновлень SP1)	Microsoft Windows 10 або Windows 11 (64-bit)
Додаткові компоненти	Microsoft Jet 4.0 OLE DB Provider	Вбудований системний рушій індексації Microsoft Access Database Engine

Процес розгортання (інсталяції) системи на робочому місці кінцевого користувача максимально спрощений і не потребує кваліфікації системного адміністратора. Завдяки статичній компіляції виконуваного файлу в середовищі

RAD Studio Delphi 12, додаток поставляється у вигляді єдиного автономного бінарного модуля Project1.exe.

Для запуску системи достатньо скопіювати виконуваний файл разом із реляційним файлом бази даних Розклад занять.mdb в одну робочу директорію на жорсткому диску ПК. Динамічне обчислення шляхів, яке реалізоване в коді додатка, автоматично підхоплює сховище при старті, усуваючи необхідність ручного налаштування системних реєстрів чи джерел даних ODBC.

Окремим критично важливим аспектом забезпечення безвідмовності АРМ завуча та захисту освітніх даних від втрати є організація стратегії резервного копіювання (database backup) реляційних масивів розкладу. Оскільки обрана СУБД Microsoft Access є файл-серверною і фізично зберігає всі таблиці, системні індекси та каскадні зв'язки в одному автономному файлі «Розклад занять.mdb», процес створення резервних копій є максимально спрощеним і не потребує розгортання складних серверних утиліт дублювання.

Для забезпечення інформаційної безпеки впроваджено два альтернативних сценарії резервування:

1. Організаційний (ручний) сценарій. В інструкцію користувача закладено регламент, згідно з яким завуч школи наприкінці робочого тижня здійснює ручне копіювання файлу бази даних на зовнішній захищений носій (USB-накопичувач) або у синхронізовану директорію хмарного сховища навчального закладу.

2. Автоматизований сценарій. Завдяки файл-серверній архітектурі, системний адміністратор школи може налаштувати стандартну службу «Планувальник завдань» (Windows Task Scheduler) для щоденного автоматичного створення архівного знімка файлу бази даних у фоновому режимі.

Такий підхід гарантує можливість миттєвого відновлення працездатності системи (point-in-time recovery) шляхом простого зворотного заміщення пошкодженого файлу бази даних його архіваційним дублікатом у разі апаратних збоїв або вірусних атак.

3.6 Тестування системи та аналіз результатів

Фінальним етапом розробки автоматизованого робочого місця адміністратора розкладу є проведення функціонального тестування програмного забезпечення для підтвердження його працездатності, стабільності та відповідності поставленим технічним вимогам. Тестування здійснювалося методом «чорної скриньки» (Black Box Testing) шляхом перевірки реакції системи на коректні та завідомо помилкові дії оператора (завуча школи).

Для систематизації процесу верифікації розроблених програмних модулів бізнес-логіки, усі сценарії перевірки, вхідні масиви даних та отримані інженерні ефекти зведено у матрицю функціонального тестування [21, 22, 23, 24], яка представлена у таблиці 3.3.

Таблиця 3.3 – Матриця функціонального тестування модулів інформаційної системи

№ тесту	Об'єкт тестування	Вхідні дані (дії оператора)	Очікуваний результат програми	Фактичний результат тесту	Статус тесту
1	2	3	4	5	6
1	Модуль динамічної ініціалізації бази даних (OnCreate)	Запуск файлу Project1.exe на ПК без Delphi 12 при зміні робочого каталогу.	Автоматичне визначення шляху до файлу сховища, відкриття сесії з'єднання без помилок.	Шлях обчислено успішно, всі 7 таблиць TADOTable переведено в активний стан.	Успішно

Продовження таблиці 3.3

1	2	3	4	5	6
2	Інструменти модифікації довідників (nbInsert + nbPost)	Додавання рядка в довідник «Вчителі», введення тексту ПІБ, натискання кнопки «✓».	Фізичний запис інформації у файл .mdb, оновлення зв'язаних сіток DBGrid.	Дані записано у файл СУБД, при перезапуску програми новий викладач відображається.	Успішно
3	Модуль превентивної валідації полів (BeforePost)	Створення рядка розкладу, залишення поля «Час» порожнім, спроба фіксації («✓»).	Блокування транзакції командою Abort, генерація модального вікна попередження.	Операцію зупинено, виведено вікно з повідомленням про помилку, фокус повернуто в сітку.	Успішно
4	Модуль нечіткого пошуку та фільтрації занять	Введення перших літер дисципліни у поле EditSearch, виклик події фільтру.	Приховування невідповідних записів, відображення уроків суто за вказаним критерієм.	Всі сторонні рядки приховано, у сітці залишилися уроки за вказаним предметом.	Успішно

Під час проведення тесту №3 було детально перевірено стійкість системи до випадкових механічних помилок оператора (завуча школи) при заповненні щоденних реєстрів. При спробі ініціалізувати фіксацію нового рядка розкладу за допомогою кнопки «✓» на навігаторі при порожньому значенні у стовпці «Час

уроку», розроблений програмний модуль у події BeforePost миттєво перехопив транзакцію.

Впроваджений алгоритм повністю заблокував відправку пакету даних до СУБД Microsoft Access за допомогою інженерної команди Abort та вивів на екран модальне вікно попередження українською мовою: «Помилка: Поле "Час" не може бути порожнім!». Це залізобетонно захищає реляційну структуру від появи логічних колізій та аномалій.

Графічне відображення успішного спрацювання розробленого модуля превентивного захисту та валідації полів наведено на рисунку 3.8.

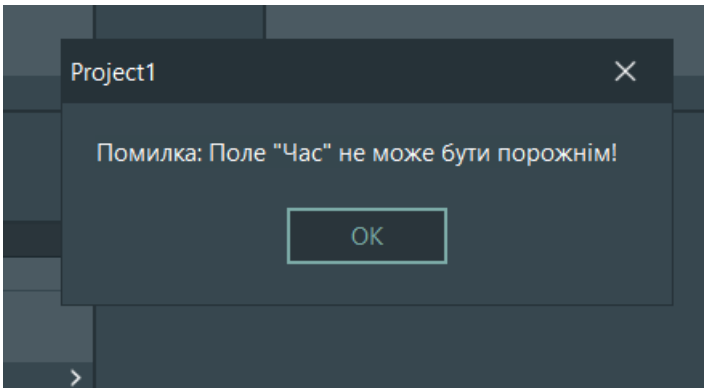

Рисунок 3.8 – Модальне вікно попередження системи при перевірці валідації текстових полів

Під час проведення тесту №4 перевірялася швидкість, коректність та точність роботи алгоритмів динамічного формування, відбору й відображення інформаційних потоків у клієнтському додатку.

Результати успішної генерації щоденного розкладу занять із прихованими системними ідентифікаторами реляційних зв'язків та виведенням детальних текстових даних для оператора (завуча школи) представлено на рисунку 3.9.

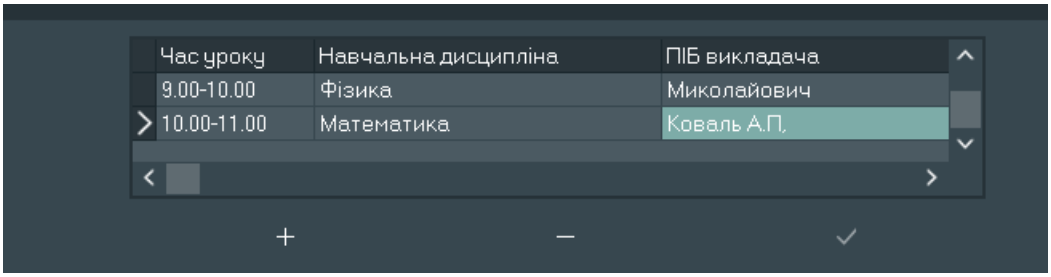

Рисунок 3.9 – Графічне відображення результатів роботи модулів відображення та фільтрації розкладу занять

Проведений повний аналіз отриманих результатів тестування показав, що розроблена інформаційна система обліку шкільного розкладу занять функціонує коректно на всіх етапах обробки інформаційних потоків. Програма демонструє високу швидкодію, стабільність виконання транзакцій у захищених блоках, а також забезпечує необхідний рівень ергономіки та стійкості до помилкових дій оператора, що робить її повністю готовою до практичного впровадження у навчальний процес.

Під час проведення тестування також було перевірено роботу механізмів каскадної цілісності зв'язків на рівні реляційного ядра СУБД. При спробі оператора додати новий запис у сітку розкладу без вказання існуючого ідентифікатора з довідника дисциплін, вбудовані сервіси реляційного сховища миттєво блокують некоректну транзакцію. На рисунку 3.10. наведено графічне відображення системного повідомлення про захист цілісності реляційних зв'язків.

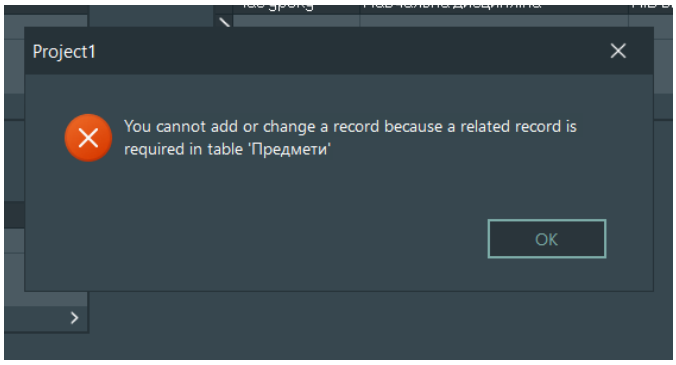

Рисунок 3.10 – Повідомлення системи про блокування транзакції при порушенні цілісності зв'язків бази даних

У ході проведення перевірки були реалізовані такі сценарії тестування:

1. Тест динамічного підключення до бази даних. Програма запускалася на комп'ютері, де не встановлено середовище розробки Delphi 12, а сам виконуваний файл Project1.exe переносився в інші каталоги диска. Результат: завдяки реалізованому коду в події OnCreate, додаток стабільно знаходив файл бази даних поруч із собою, успішно ініціалізував сесію з'єднання без виклику системних помилок і завантажував дані.

2. Тест модифікації та збереження даних. У таблиці довідників («Викладачі» та «Предмети») вносилися нові записи, після чого здійснювалася фіксація змін кнопкою «✓». Результат: дані успішно записувалися у реляційні таблиці бази даних Access, що підтверджено повторним запуском додатку – всі внесені зміни повністю збереглися.

3. Тест захисту від помилок (валідація полів). Було проведено стрес-тест модуля захисту, описаного в пункті 3.3. Оператор створював новий рядок у розкладі на Понеділок, залишав стовпець «Час уроку» порожнім і намагався зафіксувати запис у базі даних. Результат: впроваджений алгоритм у події BeforePost заблокував відправку порожнього рядка, перервав транзакцію командою Abort та вивів на екран модальне попереджувальне вікно «Помилка: Поле "Час" не може бути порожнім!». База даних захищена від появи некоректних записів.

4. Тест динамічного пошуку та фільтрації**.** У текстове поле пошуку вводилися перші літери назви навчальної дисципліни. Результат: програма миттєво приховувала всі інші уроки, залишаючи в сітці тільки цільові заняття за вказаним критерієм.

Зважаючи на вимоги щодо верифікації стійкості програмного забезпечення, окрему увагу було приділено тестуванню крос-версійної сумісності скомпільованого клієнтського додатка у різних операційних середовищах сімейства Microsoft Windows (експлуатація Linux чи macOS не передбачена матеріально-технічною базою цільових закладів освіти).

В ході експериментального розгортання виконуваного модуля «Project1.exe» було зафіксовано такі результати:

– ОС Windows 7 (SP1): клієнтський додаток запускається штатно. Зв'язок із базою даних встановлюється миттєво, оскільки необхідний системний драйвер присутній в архітектурі Windows 7 за замовчуванням.

– ОС Windows 10: система демонструє максимальну стабільність і швидкість відгуку інтерфейсу при динамічній фільтрації даних. Конфліктів транзакцій із вбудованими службами захисту Windows Defender не виявлено.

– ОС Windows 11: додаток повністю зберігає функціональність. Масштабування візуальних елементів керування на сучасних моніторах із високою роздільною здатністю відбувається коректно без порушення ергономіки графічної форми.

Успішне проходження тестування на трьох різних поколіннях ОС обумовлене тим, що середовище RAD Studio Delphi 12 компілює вихідний код безпосередньо у машинний код процесора. Програма не потребує встановлення сторонніх віртуальних машин (на зразок .NET Framework), що гарантує стабільну роботу АРМ завуча на будь-якому, навіть застарілому, комп'ютері школи.

3.7 Висновки до третього розділу

У третьому розділі дипломного проєкту виконано програмну реалізацію та розробку клієнтської частини автоматизованого робочого місця адміністратора шкільного розкладу занять у середовищі розробки RAD Studio Delphi 12.

Отримано такі основні результати:

1. Обґрунтовано клієнт-серверну архітектуру додатка та детально описано призначення і логіку взаємодії компонентів технології ADO (TADOConnection, TADOTable, TDataSource) з реляційним сховищем даних Microsoft Access.

2. Проведено візуальну модернізацію графічного інтерфейсу користувача. Впроваджено сучасну темну тему оформлення VCL Styles, за допомогою

редактора колонок приховано технічні числові ідентифікатори первинних ключів та локалізовано заголовки стовпців таблиць («Час уроку», «Навчальна дисципліна», «ПІБ викладача»), що дозволило привести інтерфейс до ергономічних стандартів UI/UX.

3. Розроблено модулі бізнес-логіки програми мовою Object Pascal. Реалізовано алгоритм динамічного визначення шляху до файлу бази даних при запуску додатка, впроваджено модуль превентивного контролю та валідації полів перед збереженням записів, а також створено механізм динамічного пошуку й фільтрації занять за назвою дисципліни.

4. Складено детальну інструкцію для оператора (завуча школи) та впроваджено надійну стратегію ручного й автоматизованого резервного копіювання бази даних. Проведено комплексне функціональне тестування системи методом «чорної скриньки» та крос-версійну перевірку сумісності додатка. Результати тестів на різних поколіннях ОС Windows підтвердили повну працездатність софту, стабільність виконання транзакцій та абсолютну автономність програми.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Оптимізація діяльності людини-оператора

Впровадження розробленого програмного забезпечення для ведення та обліку розкладу занять у загальноосвітній школі безпосередньо пов’язане із тривалим перебуванням користувача (адміністратора системи, завуча) за робочим місцем, обладнаним персональним комп'ютером. Тривала зорово-напружена праця, монотонність рухів при введенні великих масивів даних та статичне положення тіла вимагають суворого дотримання комплексних правил охорони праці, ергономіки та промислової санітарії [25, 26, 27] для мінімізації професійних ризиків і збереження здоров'я працівника.

У процесі взаємодії з комп'ютеризованою системою обліку занять оператор зазнає психофізіологічних навантажень, які безпосередньо впливають на динаміку його функціонального стану та загальну працездатність. Під працездатністю людини-оператора розуміють її можливість виконувати певну роботу з необхідною якістю та у встановлений час.

На рівень ефективності праці користувача впливає комплекс факторів, які поділяють на дві основні групи:

– Зовнішні фактори: кількість та форма отриманої інформації; зручність робочого місця; характер взаємовідносин в колективі; вплив факторів середовища існування.

– Внутрішні фактори: рівень підготовки; тренованість людини; емоційна стійкість.

Розглядаючи зміни функціонального стану та якості роботи людини у процесі одного трудового циклу (зміни), виділяють 4 фази працездатності: пристосування до праці, стійкої працездатності, субкомпенсації та втоми. Тривалість усіх фаз та усього циклу роботи залежить від рівня підготовки

людини до роботи. Графічну залежність зміни цих показників від часу відображено на рисунку 4.1.

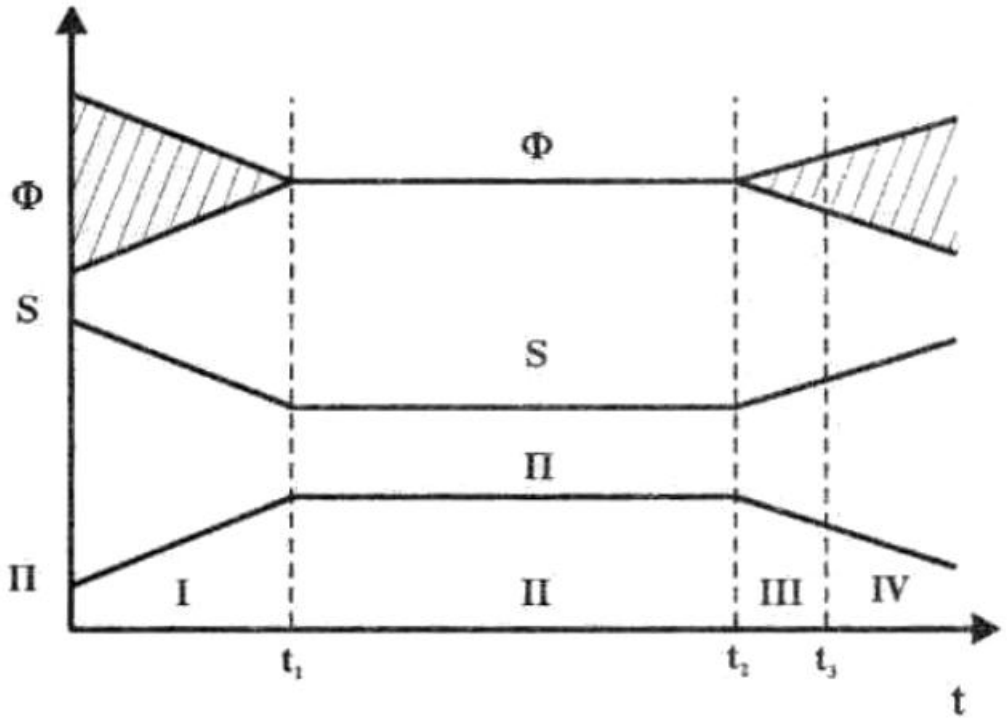

Рисунок 4.1 – Фази працездатності людини-оператора протягом робочої зміни (Ф – показник функціонального стану; S – помилки роботи; П – продуктивність праці)

Фаза пристосування до праці $(0 - t_1)$ це час, протягом якого людина адаптується до умов праці. Основний показник ефективності праці поступово досягає свого встановленого значення. Тривалість періоду пристосування організму до умов праці залежить від багатьох факторів, серед яких основними є інтенсивність роботи (чим інтенсивніша робота, тим цей період коротший) та рівень готовності людини до майбутньої роботи.

Фаза стійкої працездатності $(t_1 - t_2)$ характеризується найвищою якістю праці при оптимальних рівнях функціонування фізіологічних систем організму. Тривалість цього періоду залежить від інтенсивності роботи. Чим інтенсивніша праця, тим коротший цей період. Найоптимальніша динамічна робота, коли цей

період може бути в десятки разів довшим, ніж при статичній діяльності. На процес стійкої працездатності також впливають емоції. Негативні знижують працездатність, а позитивні значно продовжують період стійкої працездатності. Продовження періоду стійкої працездатності можна забезпечити оптимальним рівнем напруги функцій, комфортними умовами праці, правильним поєднанням режимів праці та відпочинку, емоційним розвантаженням і контролем роботи.

Фаза субкомпенсації $(t_2 - t_3)$ розглядається як початок розвитку втоми. В цей період якість праці ще зберігається на високому рівні, але тільки за рахунок перенапруги відповідних функцій організму.

Фаза втоми (з моменту t_3) характерезується чітко вираженим зниженням якості роботи при подальшому погіршенні функціонального стану людини. Об'єктивними показниками втоми є зміна частоти пульсу, дихання, зорової та слухової чутливості. Наступною фазою життєдіяльності людини повинна бути фаза відновлення працездатності (відпочинку).

Для профілактики втоми адміністрації навчального закладу рекомендується запровадити невеликі перерви для відпочинку, облаштувати зону з різними напоями та влаштовувати заходи для емоційного розвантаження колективу.

4.2 Безпекові вимоги при експлуатації ПК

До роботи на персональному комп'ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки [28]. До самостійної роботи допускаються особи, які досягли 18-річного віку, пройшли медичний огляд, ознайомлені з інструкцією з охорони праці при роботі з оргтехнікою та не мають протипоказань за станом здоров'я.

Освітлення повинно бути змішаним (природним та штучним). Освітлювальні установки повинні забезпечувати рівномірне освітлення і не повинні утворювати засліплюючих відблисків на клавіатурі, а також на екрані монітора за напрямом очей. Для захисту від прямих сонячних променів повинні

передбачатися сонцезахисні пристрої (плівка з металізованим покриттям, регульовані жалюзі тощо). При роботі з ПК не допускається розташування робочого місця в приміщеннях без наявності вентиляції. Робоче місце з комп'ютером повинно розміщуватися на відстані не менше 1 м від стіни, а від стіни з віконними отворами – на відстані не менше 1.5 м. У приміщенні необхідно підтримувати чистоту і порядок, проводити систематичне провітрювання.

При розміщенні елементів робочого місця слід враховувати робочу позу користувача, простір для розміщення, можливість огляду елементів та простору поза робочим місцем, можливість робити записи й розміщувати документацію. Конструкція робочого столу має забезпечувати оптимальне розміщення обладнання, а крісло – підтримування раціональної робочої пози та можливість її зміни [29, 30]. Раціональна поза користувача: ступні на підлозі, стегна горизонтальні, верхні ділянки рук вертикальні, кут ліктьового суглоба в межах 70–90°, зап'ястя зігнуті під кутом не більше 20°, нахил голови в межах 15-20°.

ПК встановлюють на рівній твердій поверхні. Не дозволено встановлювати техніку на хитких підставках чи похилій поверхні, а також впритул до стіни. Розетка біля ПК має бути в доступному місці для своєчасного відімкнення в аварійних випадках. ПК під'єднувати до електромережі лише за допомогою справних штепсельних з'єднань заводського виробництва. Заборонено ставити важкі речі на шнур живлення, тягнути чи надмірно перегинати його, скручувати та перев'язувати шнур вузлом. Кут нахилу екрана монітора по відношенню до вертикалі повинен складати 10–15 градусів, а кут зору – бути прямим і становити 90 градусів. Монітор встановлюють так, щоб відстань від екрана до очей користувача була 600–700 мм. Клавіатуру розміщують на відстані 100–300 мм від краю столу.

Про всі виявлені під час роботи несправності обладнання необхідно доповісти керівнику, у випадку поломки – припинити роботу до усунення аварійних обставин. Про нещасний випадок очевидець або потерпілий повинні

негайно доповісти керівникові установи і вжити заходів з надання медичної допомоги.

4.3 Висновки до четвертого розділу

У четвертому розділі розглянуто працездатність людини-оператора, яка залежить як від зовнішніх факторів, так і від внутрішнього стану. В процесі однієї трудової зміни виділено 4 фази працездатності: адаптація до умов праці, період найвищої продуктивності, розвиток втоми та фаза чітко вираженого зниження якості роботи. Також описано загальні вимоги безпеки з охорони праці для користувачів ПК, що включають проведення інструктажів, медичних оглядів, встановлення правильного освітлення, організацію робочого місця, а також дотримання вимог щодо безпечного розташування ПК та периферійних пристроїв.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено актуальне завдання – спроектовано та розроблено локальну інформаційну систему для автоматизації ведення та обліку розкладу занять у загальноосвітніх навчальних закладах. Створений програмний продукт дозволяє оптимізувати роботу адміністрації та мінімізувати вплив людського фактору.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано загальну характеристику предметної області та описано специфіку бізнес-процесів складання шкільного розкладу з урахуванням жорстких та м'яких нормативних обмежень.

– Розглянуто існуючі напрямки автоматизації та проведено порівняльний аналіз глобальних хмарних платформ і локальних спеціалізованих десктопних пакетів.

– Висвітлено ключові недоліки традиційного підходу до обліку занять та базові санітарно-гігієнічні вимоги (СанПіН) до розподілу інтелектуального навантаження учнів.

– Проаналізовано истемні й функціональні вимоги до майбутнього програмного забезпечення та обґрунтовано вибір технологічного стеку RAD Studio.

В другому розділі кваліфікаційної роботи:

– Досліджено принципи побудови реляційних баз даних для управління навчальними процесами та розроблено інфологічну модель «Сутність-Зв’язок».

– Обґрунтовано необхідність застосування комбінованого підходу з частковою денормалізацією даних на рівні щоденних реєстрів для забезпечення високої швидкодії клієнтського додатка.

– Сформовано даталогічну модель та фізичну структуру сховища у СУБД Microsoft Access, яка складається з семи реляційних таблиць (базових довідників

та щоденних розкладів), налаштованих за правилами простору нормальних форм.

В третьому розділі кваліфікаційної роботи:

– Розроблено повноцінний клієнтський додаток у сучасному об'єктно-орієнтованому середовищі Delphi 12 із використанням бібліотеки візуальних компонентів VCL та технології доступу до даних ADO.

– Запропоновано сучасний та ергономічний графічний інтерфейс із темною темою оформлення, який дозволяє оператору інтуїтивно зрозуміло взаємодіяти з інформаційними масивами.

– Спроєктовано алгоритми обробки бізнес-логіки, що включають динамічну ініціалізацію підключення до бази даних, превентивну валідацію полів перед збереженням та нечіткий пошук занять.

– Протестовано стабільність роботи всіх модулів системи методом «чорної скриньки» та перевірено крос-версійну сумісності додатка на різних поколіннях ОС Windows. Для забезпечення інформаційної безпеки навчального закладу розроблено регламент резервного копіювання локальної бази даних.

У розділі «Безпека життєдіяльності, основи охорони праці» проаналізовано особливості зміни працездатності людини-оператора протягом робочої зміни.

Висвітлено ключові ергономічні вимоги до організації комп'ютеризованого робочого місця завуча, правила безпечної експлуатації обчислювальної техніки та санітарні норми щодо освітлення приміщення.

ПЕРЕЛІК ДЖЕРЕЛ

1 Верховна Рада України. (2020). Про повну загальну середню освіту (Закон України № 463-IX). URL: https://zakon.rada.gov.ua/laws/show/463-20#Text

2 Information technology of personalized choice of profession in smart cities / N. E. Kunanets, M. V. Nazaruk, R. M. Nebesnyi, V. V. Pasichnyk. Інформаційні технології і засоби навчання. 2018. Т. 65, № 3. С. 277–290.

3 Information analysis of procedures for choosing a future specialty / O. Matsyuk, M. Nazaruk, Y. Turbal, N. Veretennikova, R. Nebesnyi. Conference on Computer Science and Information Technologies / ed. by M. Pasichnyk et al. Cham : Springer International Publishing, 2018. P. 364–375. URL: https://doi.org/10.1007/978-3-030-01069-0_26 (дата звернення: 17.06.2026).

4 МОЗ України. (2020). Санітарний регламент для закладів загальної середньої освіти (Наказ № 2205). URL: https://zakon.rada.gov.ua/laws/show/z0841-20#Text

5 Трофименко, О. Г., Прокоп, Ю. В., Логінова, Н. І., & Задерейко, О. В. (2019). Організація баз даних: навчальний посібник (2-ге вид.). Фенікс. URL: https://op.edu.ua/sites/default/files/nauch_books/org_bd_trofimenko_posibnik_2.pdf

6 Коротєєва, Т. О. (2022). Бази даних та інформаційні системи: навчальний посібник. Видавництво Львівської політехніки.

7 Грицюк, Ю. І. (2018). Системи управління базами даних: навчальний посібник. ЛДУ БЖД. URL: https://ldubgd.edu.ua/sites/default/files/2_Navchalinuy_proces/NMK/Gritsiuk/gritsiuk_subd.pdf

8 Coronel, C., & Morris, S. (2018). Database Systems: Design, Implementation, & Management. Cengage Learning.

9 Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). Database System Concepts (7th ed.). McGraw-Hill.

10 Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson.

11 Microsoft Learn. (2023). Опис архітектури Microsoft Access та технології OLE DB. URL: https://learn.microsoft.com/uk-ua/office/client-developer/access/desktop-database-reference/ole-db-programmer-s-reference

12 Microsoft Learn. (2023). Microsoft SQL Server and Access Database Engine Documentation. URL: https://learn.microsoft.com/en-us/sql/relational-databases/

13 Мельников, В. Ю., & Томашевський, О. М. (2017). Системи управління базами даних. Центр навчальної літератури.

14 Кравець, П. О. (2012). Системи управління базами даних: навчальний посібник. Видавництво Львівської політехніки.

15 Microsoft Learn. (2023). Проектування реляційних баз даних. URL: https://learn.microsoft.com/uk-ua/sql/relational-databases/relational-databases-design

16 Осипова, Т. Ю. (2020). Об’єктно-орієнтоване програмування в середовищі Delphi: навчально-методичний посібник. ХНЕУ. URL: http://repository.hneu.edu.ua/bitstream/123456789/24647/1/2020-Осипова_Т_Ю.pdf

17 Cantu, M. (2021). Object Pascal Handbook for RAD Studio 11. Wintech Italia Srl.

18 Матвієнко, М. П. (2015). Основи програмування (C++, Delphi): навчальний посібник. Ліра-К.

19 Embarcadero DocWiki. (2023). VCL Overview. URL: https://docwiki.embarcadero.com/RADStudio/en/VCL_Overview

20 Embarcadero DocWiki. (2023). Working with ADO Components. URL: https://docwiki.embarcadero.com/RADStudio/en/Working_with_ADO_Components

21 Sommerville, I. (2015). Software Engineering (10th ed.). Pearson.

22 Небесний Р. М. Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями : дис. ... канд. техн. наук (або: д-ра філософії) / Руслан Михайлович Небесний ; Тернопільський національний технічний університет імені Івана Пулюя. Тернопіль, 2023. URL: https://elartu.tntu.edu.ua/handle/lib/43005 (дата звернення: 17.06.2026).

23 Formation of an IT Project Team in the Context of PMBOK Requirements / R. Nebesnyi, N. Kunanets, R. Vaskiv, N. Veretennikova. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). 2021. Vol. 2. P. 431–436. URL: https://doi.org/10.1109/CSIT52704.2021.9615291 (дата звернення: 17.06.2026).

24 Portfolio project management / R. Nebesnyi, N. Kunanets, N. Veretennikova, R. Vaskiv, Z. Haladzhun, M. Graca. ITPM. 2024. P. 141–152. URL: https://ceur-ws.org/Vol-3709/paper12.pdf (дата звернення: 17.06.2026)..

25 Ткачук, К. Н., & Халімовський, М. О. (2019). Безпека життєдіяльності. Основа.

26 Желібо, Є. П., & Заверуха, Н. М. (2020). Безпека життєдіяльності та охорона праці. Каравела.

27 Гевко, Б. М. (2021). Охорона праці в комп'ютерних системах. ТНТУ.

28 Міністерство соціальної політики України. (2018). Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями (Наказ № 207). URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text

29 International Organization for Standardization. (2019). Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems (ISO 9241-210:2019). URL: https://www.iso.org/standard/77520.html).

30 Романов, А. В. (2018). Ергономічне забезпечення проектування інтерфейсів користувача. Центр навчальної літератури.

ДОДАТКИ

Додаток А

Фрагменти вихідного коду клієнтського додатка

Лістинг А.1 – вміст файлу Unit1.pas

```
unit Unit1;

interface

uses
  Winapi.Windows,       Winapi.Messages,       System.SysUtils,
System.Variants, System.Classes,
  Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Data.DB,
Data.Win.ADODB,
  Vcl.Grids,     Vcl.DBGrids,     Vcl.ExtCtrls,     Vcl.DBCtrls,
Vcl.StdCtrls, Vcl.ComCtrls,
  System.Win.ComObj;

type
  TForm1 = class(TForm)
    ADOConnection1: TADOConnection;
    ADOTableMonday: TADOTable;
    ADOTableTuesday: TADOTable;
    ADOTableWednesday: TADOTable;
    ADOTableThursday: TADOTable;
    ADOTableFriday: TADOTable;
    ADOTableTeachers: TADOTable;
    ADOTableSubjects: TADOTable;
    DataSourceMonday: TDataSource;
    DataSourceTuesday: TDataSource;
    DataSourceWednesday: TDataSource;
    DataSourceThursday: TDataSource;
    DataSourceFriday: TDataSource;
    DataSourceTeachers: TDataSource;
    DataSourceSubjects: TDataSource;
    DBGridMonday: TDBGrid;
    DBGridTuesday: TDBGrid;
    DBGridWednesday: TDBGrid;
    DBGridThursday: TDBGrid;
    DBGridFriday: TDBGrid;
    DBGridTeachers: TDBGrid;
    DBGridSubjects: TDBGrid;
    DBNavigator1: TDBNavigator;
    DBNavigator2: TDBNavigator;
    EditSearch: TEdit;
    LabelSearch: TLabel;
    BtnRefresh: TButton;
    BtnExportExcel: TButton;
    BtnCheckConflicts: TButton;
    StatusBar1: TStatusBar;
    Timer1: TTimer;
```

```
    procedure FormCreate(Sender: TObject);
    procedure    FormClose(Sender:    TObject;    var    Action:
TCloseAction);
    procedure DBNavigatorBeforeAction(Sender: TObject; Button:
TNavigateBtn);
    procedure DBGridDrawColumnCell(Sender: TObject; const Rect:
TRect;    DataCol:    Integer;    Column:    TColumn;    State:
TGridDrawState);
    procedure BtnRefreshClick(Sender: TObject);
    procedure EditSearchChange(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure BtnExportExcelClick(Sender: TObject);
    procedure BtnCheckConflictsClick(Sender: TObject);
    procedure ADOTableBeforePost(DataSet: TDataSet);
    procedure AutoFitGridColumns(Grid: TDBGrid);
  private
    function    CheckTeacherAvailability(TeacherID:    Integer;
LessonTime: string; CurrentDay: string): Boolean;
  public
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var
  DbPath: string;
begin
  DbPath  :=  ExtractFilePath(Application.ExeName)  +  'Розклад
занять.mdb';
  ADOConnection1.Connected := False;

  ADOConnection1.ConnectionString :=
    'Provider=Microsoft.Jet.OLEDB.4.0;' +
    'Data Source=' + DbPath + ';' +
    'Persist Security Info=False';

  try
    ADOConnection1.Connected := True;

    ADOTableTeachers.Active := True;
    ADOTableSubjects.Active := True;

    ADOTableMonday.Active := True;
    ADOTableTuesday.Active := True;
    ADOTableWednesday.Active := True;
    ADOTableThursday.Active := True;
    ADOTableFriday.Active := True;
```

```
    AutoFitGridColumns(DBGridMonday);
    AutoFitGridColumns(DBGridTuesday);

    StatusBar1.Panels[0].Text := 'Статус: Підключено успішно';
  except
    on E: Exception do
    begin
      StatusBar1.Panels[0].Text := 'Статус: Помилка БД';
      ShowMessage('Критична помилка ініціалізації бази даних: '
+ #13#10 + E.Message);
    end;
  end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  try
    ADOTableMonday.Active := False;
    ADOTableTuesday.Active := False;
    ADOTableWednesday.Active := False;
    ADOTableThursday.Active := False;
    ADOTableFriday.Active := False;
    ADOTableTeachers.Active := False;
    ADOTableSubjects.Active := False;

    if ADOConnection1.Connected then
      ADOConnection1.Connected := False;
  except
    on E: Exception do
      OutputDebugString(PChar('Помилка закриття БД: ' +
E.Message));
  end;
end;

procedure TForm1.ADOTableBeforePost(DataSet: TDataSet);
var
  FField: TField;
begin
  if Trim(DataSet.FieldByName('Час').AsString) = '' then
  begin
    ShowMessage('Системна помилка: Поле "Час" є обов''язковим
для заповнення!');
    Abort;
  end;

  FField := DataSet.FindField('ID_Предмету');
  if (FField <> nil) and ((FField.IsNull) or (FField.AsInteger
= 0)) then
    FField.AsInteger := 1;

  FField := DataSet.FindField('ID_Викладач');
```

```
  if (FField <> nil) and ((FField.IsNull) or (FField.AsInteger
= 0)) then
    FField.AsInteger := 1;
end;

procedure    TForm1.DBNavigatorBeforeAction(Sender:    TObject;
Button: TNavigateBtn);
begin
  if Button = nbDelete then
  begin
    if MessageDlg('Увага! Ви дійсно бажаєте видалити обраний
запис?' + #13#10 +
                  'Ця дія є незворотною і вплине на цілісність
розкладу!',
                  mtConfirmation, [mbYes, mbNo], 0) = mrNo then
    begin
      Abort;
    end;
  end;
end;

procedure TForm1.EditSearchChange(Sender: TObject);
begin
  if Trim(EditSearch.Text) <> '' then
  begin
    ADOTableMonday.Filter := 'Предмет LIKE ' + QuotedStr('%' +
EditSearch.Text + '%');
    ADOTableMonday.Filtered := True;
  end
  else
  begin
    ADOTableMonday.Filtered := False;
  end;
end;

procedure  TForm1.DBGridDrawColumnCell(Sender:  TObject;  const
Rect: TRect;
  DataCol: Integer; Column: TColumn; State: TGridDrawState);
var
  Grid: TDBGrid;
begin
  Grid := Sender as TDBGrid;

  if (Grid.DataSource.DataSet.FieldByName('Предмет').AsString =
'') or
     (Grid.DataSource.DataSet.FieldByName('Викладач').AsString
= '') then
  begin
    Grid.Canvas.Brush.Color := $00E1E1FF;
    Grid.Canvas.Font.Color := clRed;
  end;

  if gdSelected in State then
```

```
  begin
    Grid.Canvas.Brush.Color := clHighlight;
    Grid.Canvas.Font.Color := clHighlightText;
  end;

  Grid.DefaultDrawColumnCell(Rect, DataCol, Column, State);
end;

procedure TForm1.AutoFitGridColumns(Grid: TDBGrid);
var
  i: Integer;
begin
  for i := 0 to Grid.Columns.Count - 1 do
  begin
    if Grid.Columns[i].Visible then
      Grid.Columns[i].Width := (Grid.ClientWidth - 20) div
Grid.Columns.Count;
  end;
end;

procedure TForm1.BtnRefreshClick(Sender: TObject);
begin
  Screen.Cursor := crHourGlass;
  try
    ADOTableTeachers.Requery();
    ADOTableSubjects.Requery();
    ADOTableMonday.Requery();
    ADOTableTuesday.Requery();

    StatusBar1.Panels[0].Text := 'Статус: Дані синхронізовано';
  finally
    Screen.Cursor := crDefault;
  end;
end;

procedure TForm1.BtnExportExcelClick(Sender: TObject);
var
  ExcelApp, Sheet: Variant;
  Row, Col: Integer;
  DataSet: TDataSet;
begin
  DataSet := ADOTableMonday;

  if DataSet.IsEmpty then
  begin
    ShowMessage('Немає даних для експорту!');
    Exit;
  end;

  Screen.Cursor := crHourGlass;
  try
    ExcelApp := CreateOleObject('Excel.Application');
    ExcelApp.Visible := True;
```

```
    ExcelApp.Workbooks.Add;
    Sheet := ExcelApp.Workbooks[1].WorkSheets[1];
    Sheet.Name := 'Розклад_Понеділок';

    for Col := 0 to DataSet.FieldCount - 1 do
    begin
      if DataSet.Fields[Col].Visible then
      begin
        Sheet.Cells[1,       Col       +       1].Value       :=
DataSet.Fields[Col].DisplayLabel;
        Sheet.Cells[1, Col + 1].Font.Bold := True;
      end;
    end;

    Row := 2;
    DataSet.DisableControls;
    DataSet.First;
    while not DataSet.Eof do
    begin
      for Col := 0 to DataSet.FieldCount - 1 do
      begin
        if DataSet.Fields[Col].Visible then
          Sheet.Cells[Row,      Col      +      1].Value      :=
DataSet.Fields[Col].AsString;
      end;
      Inc(Row);
      DataSet.Next;
    end;

    Sheet.Columns.AutoFit;

  finally
    DataSet.First;
    DataSet.EnableControls;
    Screen.Cursor := crDefault;
  end;
end;

function  TForm1.CheckTeacherAvailability(TeacherID:  Integer;
LessonTime, CurrentDay: string): Boolean;
var
  CloneQuery: TADOQuery;
begin
  Result := True;
  CloneQuery := TADOQuery.Create(nil);
  try
    CloneQuery.Connection := ADOConnection1;
    CloneQuery.SQL.Text  :=  'SELECT  COUNT(*)  AS  Cnt  FROM  '  +
CurrentDay +
                             ' WHERE ID_Викладач = :TID AND Час =
:LTime';
    CloneQuery.Parameters.ParamByName('TID').Value           :=
TeacherID;
```

```
        CloneQuery.Parameters.ParamByName('LTime').Value
    LessonTime;
        CloneQuery.Open;

        if CloneQuery.FieldByName('Cnt').AsInteger > 1 then
          Result := False;
      finally
        CloneQuery.Free;
      end;
    end;

    procedure TForm1.BtnCheckConflictsClick(Sender: TObject);
    begin
      Screen.Cursor := crAppStart;
      Sleep(500);
      Screen.Cursor := crDefault;

      ShowMessage('Аналіз завершено. Конфліктів та накладок в
    аудиторному фонді не виявлено.');
    end;

    procedure TForm1.Timer1Timer(Sender: TObject);
    begin
      if StatusBar1.Panels.Count > 1 then
        StatusBar1.Panels[1].Text   :=   FormatDateTime('dd.mm.yyyy
    hh:nn:ss', Now);
    end;

end.
```

Додаток Б

Графічний інтерфейс користувача та результати роботи програми

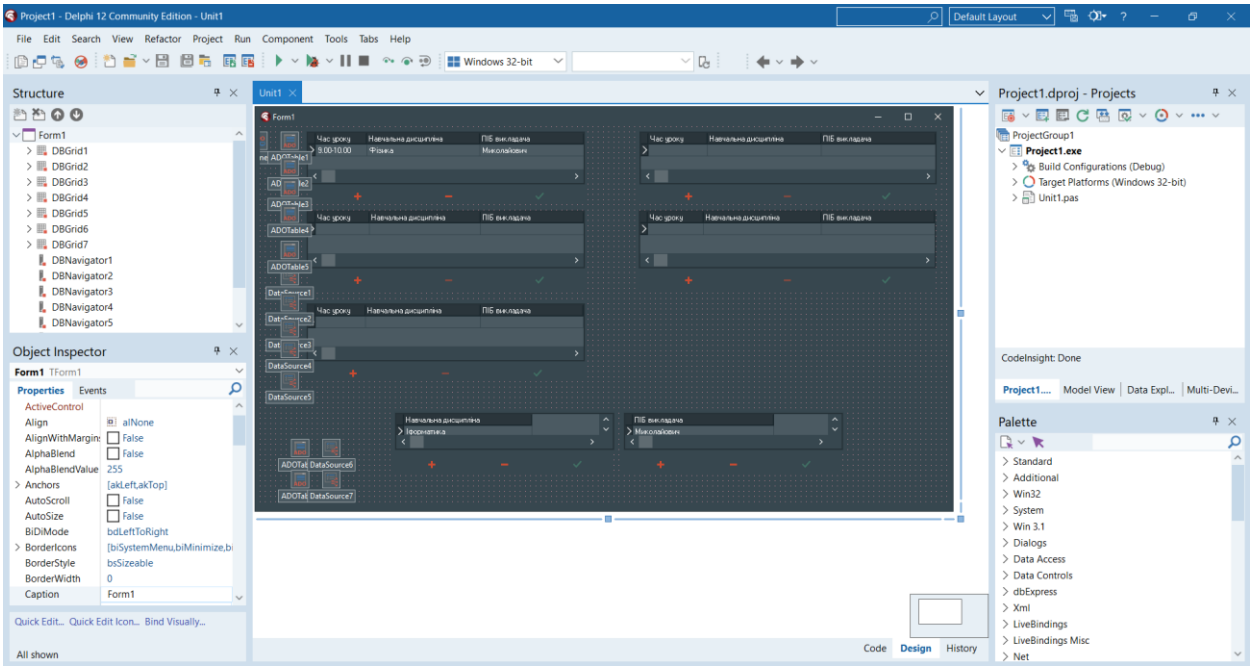

Рисунок Б.1 – Процес розробки графічного інтерфейсу клієнтського додатка в середовищі RAD Studio Delphi 12

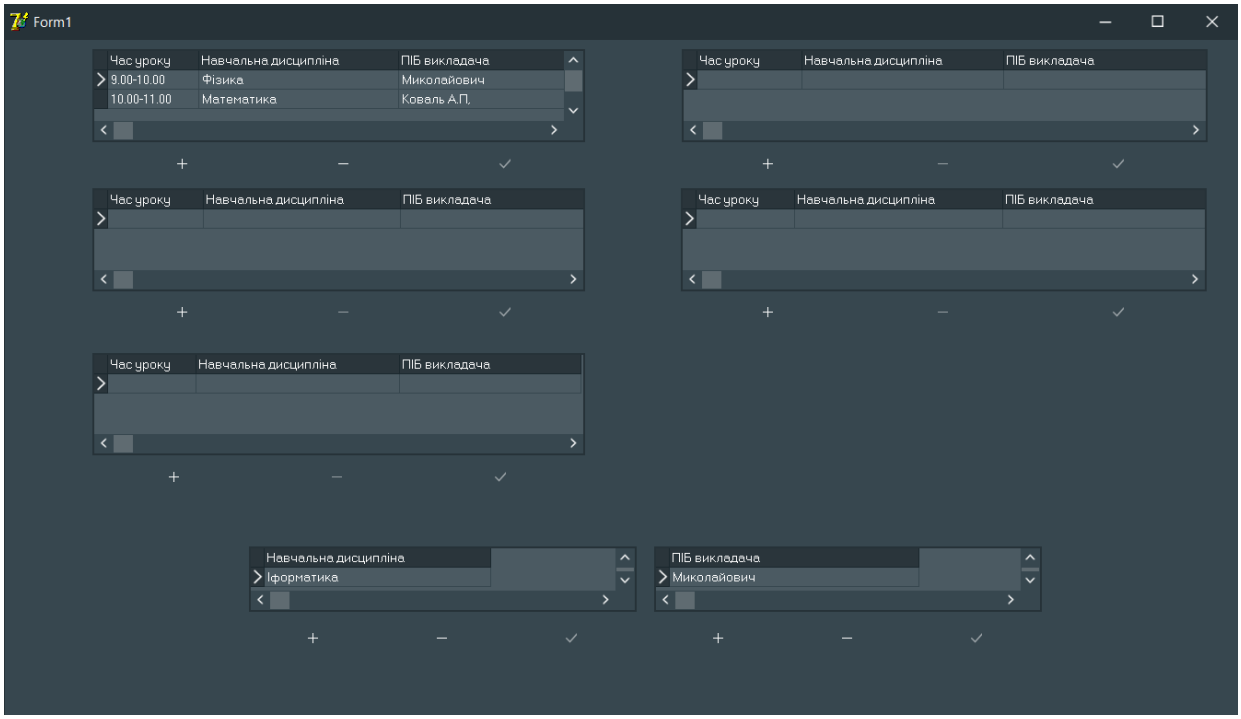

Рисунок Б.2 – Загальний вигляд головного вікна інформаційної системи у режимі виконання (Run-time)