

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ЗАПОРІЗЬКИЙ ЕЛЕКТРОТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Циклова комісія спеціальності
121 Інженерія програмного забезпечення
спеціалізація «Розробка програмного забезпечення»

АВТОМАТИЗОВАНА СИСТЕМА "ШКІЛЬНИЙ ЕЛЕКТРОННИЙ ЖУРНАЛ"

Пояснювальна записка до курсової роботи

121.47.30.01 ПЗ

Викладач

Члени комісії

Студент гр. РПЗ 19 2/9

Тетяна ПИЛИПЕНКО

Жанна ФЕДОРІНА

Олександр КОРОТКИЙ

Іван ЩЕДРОВСЬКИЙ

РЕФЕРАТ

Пояснювальна записка містить: 61 сторінку, 14 рисунків, 7 таблиць, 2 додатки, 11 джерел.

Мета роботи – розробка додатку для автоматизації шкільного журналу.

Метою створення програми є розробка програмного продукту призначеного для автоматизації шкільного журналу, пошуку та додавання або редагування даних у таблицях «довідниках».

Розділ «Опис предметної області» включає основні поняття, що стосуються теми роботи, опис алгоритму розробки програми, опис інформаційних потоків в організації, яка автоматизується.

Розділ «Постанова завдання» включає мету створення програми, функції програми, вимоги до проектованої системи, вимоги до надійності програмного продукту, умови роботи програми.

Розділ «Програмування» містить опис переваг вибраного інструментарію розробки програми, опис переваг вибраного операційного середовища, опис використовуваних моделей даних.

Розділ «Методика роботи користувача з системою» містить керівництво програміста та керівництво оператора.

ЖУРНАЛ, ВЧИТЕЛЬ, УЧЕНЬ, ГРУПИ, ПРЕДМЕТИ, РОЗКЛАД,
WINDOWS, SQL, BORLAND DELPHI, БАЗА ДАНИХ

ЗМІСТ

ВСТУП.....	4
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	5
1.1 Основні поняття.....	5
2 ПОСТАНОВА ЗАВДАННЯ	7
3 ПРОГРАМУВАННЯ	9
3.1 Обґрунтування вибору середовища розробки системи.....	9
3.2 Обґрунтування вибору середовища функціонування системи	13
3.3 Використані моделі даних	17
4 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ	23
4.1 Керівництво програміста.....	23
4.2 Керівництво оператора	24
ВИСНОВКИ.....	32
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	33
ДОДАТОК А.....	34
ДОДАТОК Б	60

ВСТУП

Збільшення обсягів знань та перебудова навчального процесу викликали значне збільшення обсягу інформації. Щороку зростає кількість оргтехніки в школах. Для вирішення проблем з обробки інформації потрібна правильна організація обробки документів, від якої залежить ефективність процесу управління в навчальних закладах. Значну частину часу забирає документаційне обслуговування: попередній розгляд, облік, зберігання, контроль тощо.

На сьогоднішній день за допомогою сучасної електронно-обчислювальної техніки здійснюється автоматизація розв'язання майже всіх задач, що виникають в процесі роботи навчального закладу. Автоматизація включає застосування керувальних пристроїв, які використовують електронні давачі і комп'ютерну техніку.

Задача автоматизації – підвищення якості роботи як окремих працівників, так і всього закладу в цілому. Комп'ютер – це тільки інструмент, який дозволяє максимально повністю використовувати кваліфікацію спеціаліста і максимально спростувати щоденну роботу.

Переваги від впровадження інформаційних технологій в навчальних закладах, та проблеми, які при цьому виникають розглянуті не повністю. Тому є необхідність поглибленого вивчення процесу впровадження спеціалізованого програмного забезпечення у сфері освіти, проблем та перспектив розвитку інформаційних систем.

Будь-який навчальний заклад має за мету збільшення відсотку знань учнів під час звичайної навчальної діяльності. Це досягається за рахунок підвищення продуктивності праці. Саме тому спостерігається інтенсивне впровадження у виробництво досягнень науково-технічного прогресу, застосування новітніх революційних технологій.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Електронний журнал - періодичне електронне видання, що є закінченим електронним ресурсом і містить групу електронних документів (статей), що пройшли редакційно-видавничу обробку та призначені для довготривалого зберігання, розповсюдження в комп'ютерних мережах у незмінному вигляді.

Необхідність автоматизації: в сучасному світі технології все більше впроваджуються в усі сфери нашого життя. Це не обійшло і навчальний процес. В сучасності діти та їх батьки хочуть дізнаватись про свої оцінки через інтернет і не телефонувати класному керівнику і відволікати його від роботи. Саме цю задачу вирішує мій додаток.

Цей зручний сервіс призначений для обробки і надання в зручному електронному вигляді інформації про успішність учнів, суміжній інформації, що доступна через Інтернет. Завдяки такому сервісу:

- батьки завжди можуть проаналізувати якість навчання та зкоректувати підготовку дитини з того чи іншого предмету;
- учні можуть бачити, які роботи їм треба перездати, а які вони ще не зробили;
- вчителі зможуть автоматично розраховувати оцінки їх учнів/студентів.

Журнал – центральний вузол всієї системи, пов'язує вчителів, учнів та уроки. В ньому виводиться інформація про оцінки учнів по групам. Також саме в ньому вчителі ставлять відмітки (оцінки або НБ).

Предмет або навчальна дисципліна – педагогічно адаптована система понять про явища, закономірності, закони, теорії, методи тощо будь-якої галузі діяльності (або сукупності різних галузей діяльності) із визначенням потрібного рівня сформованості у тих, хто навчається, певної сукупності умінь і навичок. Кожен предмет має свого закріпленого вчителя та аудиторію.

Розклад навчальних занять - один з основних організаційних документів, що регламентує освітній процес. Дає можливість вчителям не створювати заняття

окремо, а просто заповнювати їх в журналі. В розкладі вказаний предмет, день тижня в який він буде проведений та номер пари, а також група, з якою буде проведене заняття.

Вчитель – людина, яка навчає інших людей (своїх учнів), передає їм певні знання про життя. У вузькому розумінні — спеціаліст, що проводить навчальну та виховну роботу з учнями в загальноосвітніх школах різних типів.

Учень – особистість, яка бере участь в навчальному процесі та здобуває певні знання. Найчастіше ми використовуємо це поняття для означення дітей та підлітків, які відвідують загальноосвітні заклади – школи, гімназії ліцеї.

2 ПОСТАНОВА ЗАВДАННЯ

Метою створення програмного продукту є розробка додатку для автоматизації шкільного журналу.

Після вивчення предметної області і створення відповідної бази даних необхідно створити програмне забезпечення, яке матиме наступні функції:

- доступ до таблиць бази даних;
- редагування інформації в таблицях;
- оформлення груп учнів;
- пошук оцінок по даті;
- пошук уроку в розкладі;
- пошук потрібного учня за прізвищем або групою;
- виконання запитів з таблиць за різними критеріями;
- виведення звітів;
- друк інформації.

Вимоги до проектованої системи:

- обов'язкове виведення на екран вихідних форм;
- програма повинна надавати можливість переходу з однієї форми на іншу;
- передбачити можливість редагувати дані;
- програма повинна реалізовувати функції щодо зміни інформації (додавання, видалення);
- програма повинна мати зручний, максимально орієнтований на будь-якого користувача інтерфейс.

Вимоги до надійності програмного продукту наступні:

- програма повинна не давати можливість користувачеві вводити дані в комірки з вихідними даними;
- формувати звіти зазначених форм, виводити їх на екран та на пристрій друку;
- програма повинна конкретно інтерпретувати і зберігати інформацію;

– для виходу з форм довідників повинна бути кнопка повернення до головної форми.

Для нормальної роботи потрібний персональний комп'ютер з мікропроцесором Intel i3 3,30 GHz та вище, оперативна пам'ять 4Гб і вище, вільного простору на жорсткому диску 5Гб, SVGA-монітор, клавіатура та маніпулятор типу "миша". Для виводу на друк інформації потрібен принтер.

На всі пристрої, що використовуються в системі, потрібні драйвери цих пристроїв.

Програма нормально функціонує під керуванням операційної системи Microsoft Windows 10.

3 ПРОГРАМУВАННЯ

3.1 Обґрунтування вибору середовища розробки системи

Для розробки програмного продукту було обрано середовище розробки Borland Delphi.

Borland Delphi – це інтегроване середовище швидкої розробки програмного забезпечення для роботи під Microsoft Windows, що відноситься до класу RAD - (Rapid Application Development «Засіб швидкої розробки додатків») засобів CASE - технології. Delphi зробила розробку могутніх додатків Windows швидким процесом. Воно підтримує розробку Windows-додатків на мові програмування Delphi, яка є наступницею мови Object Pascal.

Спочатку середовище розробки була призначена виключно для розробки додатків Microsoft Windows, потім був реалізований також для платформ GNU / Linux, однак після випуску в 2002 році Kylix 3 його розробка була припинена, і, незабаром після цього, було оголошено про підтримку Microsoft . NET.

Мова Delphi - результат розвитку мови Turbo Pascal, який, у свою чергу, розвинувся з мови Pascal. Pascal був повністю процедурною мовою, Turbo Pascal, починаючи з версії 5.5, додав в Pascal об'єктно-орієнтовані властивості, а в Object Pascal динамічну ідентифікацію типу даних з можливістю доступу до метаданих класів (тобто до опису класів та їх членів) в компільованому коді.

Delphi в основному використовується для розробки настільних додатків та корпоративних СКБД, проте цей інструмент можна використовувати для розробки будь-якого загального програмного забезпечення. Не залишена осторонь і можливість побудови Веб-додатків, так потрібних у сучасному інформаційному світі.

Основний наголос в Borland Delphi робиться на те, щоб максимально продуктивно використовувати код. Це дозволяє дуже швидко розробляти додатки, оскільки вже існують заздалегідь підготовлені об'єкти. Також можна створювати свої власні об'єкти, без яких-небудь обмежень.

У стандартне постачання Delphi входять основні об'єкти з 270 базових класів. На цій мові дуже зручно писати як додатки до баз даних, так навіть і ігрові програми. Якщо взяти до уваги і зручний інтерфейс для створення графічних оболонок, то можна з упевненістю заявити, що мова Delphi - це дуже доступна для розуміння, але в той же час і дуже могутня мова програмування.

Delphi володіє широким набором можливостей, починаючи від проектувальника форм і закінчуючи підтримкою всіх форматів популярних баз даних.

Компілятор, вбудований в Delphi, забезпечує високу продуктивність, необхідну для побудови додатків в архітектурі "клієнт-сервер". Він пропонує легкість розробки та швидкий час перевірки готового програмного блоку, характерного для мов четвертого покоління (4GL) і в цей же час забезпечує якість коду, характерного для компілятору 3GL. Крім цього, Borland Delphi забезпечує швидку розробку без необхідності писати вставки на Сі або ручного написання коду (хоча це можливо).

Основний натиск в моделі Delphi робиться на максимальне ревикористання коду. Це дозволяє розробникам будувати додаток дуже швидко із раніше заготовлених об'єктів, а також дає їм змогу створити свої власні об'єкти для середовища Delphi. Ніяких обмежень по типам об'єктів, які можуть створювати розробники, не існує.

Перевагами Borland Delphi є:

- середовище усуває необхідність програмувати такі компоненти Windows загального призначення, як форми, піктограми і навіть діалогові панелі;
- заздалегідь є певні візуальні і невізуальні об'єкти, включаючи кнопки, об'єкти з даними, меню і вже побудовані діалогові панелі;
- час компіляції програм у будь-якої С-мови на порядок довше, ніж у Delphi, це знижує віддачу від програміста - не кожен може писати код без щонайменших помилок. Навіть на могутніх машинах С-продукти не можуть забезпечити швидкодії, властивій Delphi.

Однією з найбільш сильних сторін середовища програмування Delphi є її відкрита архітектура, завдяки якій Delphi допускає свого роду метапрограмування, дозволяючи “програмувати середовище програмування”. Такий підхід переводить Delphi на якісно новий рівень систем розробки додатків і дозволяє вбудовувати в цей продукт додаткові інструментальні засоби, що підтримують практично всі етапи створення прикладних систем. Такий широкий спектр можливостей відкривається завдяки реалізованій в Delphi концепції так званих відкритих інтерфейсів, що є сполучною ланкою між IDE (Integrated Development Environment) і зовнішніми інструментами.

Для розробки програмного продукту було використано Microsoft Access - реляційна СУБД корпорації Microsoft. Має широкий спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями і базами даних. Завдяки вбудованій мові VBA, в самому Access можна писати програми, що працюють з базами даних.

Основні компоненти MS Access: майстер таблиць; будівник екранних форм; будівник SQL-запитів (мова SQL в MS Access не відповідає стандарту ANSI); будівник звітів, що виводяться на друк. Вони можуть викликати скрипти на мові VBA, тому MS Access дозволяє розробляти програми БД практично «з нуля» або написати оболонку для зовнішньої БД. MS Access є файл-серверної СКБД і тому застосовується лише до маленьких додатків. Відсутній ряд механізмів необхідних у розрахованих на багато користувачів БД, таких, наприклад, як транзакції. Досвід показує, що навіть для проектів на 5-20 користувачів переважно використовувати клієнт-серверні рішення.

Бібліотека ADO (Microsoft ActiveX Data Object) служить для доступу до баз даних різних типів і надає об'єктний програмний інтерфейс до інтерфейсу OLE DB, який пропонується компанією Microsoft як альтернатива інтерфейсу ODBC. Об'єктна модель ADO реалізована на базі технології COM (Component Object Model). Бібліотека ADO може бути використана в будь-яких середовищах, які в змозі виступити в ролі OLE-клієнта, наприклад, в MS Office (VBA), 1С: Підприємство, адміністративних скриптах Windows (. Vbs і Js) і т.д. За допомогою

бібліотеки ADO можна звернутися до величезної кількості типів баз даних, наприклад, dBASE, Access, Excel, Oracle, Paradox, MS SQL Server, Sybase, текстові файли, FoxPro, Active Directory Service, Microsoft Jet, Interbase, Informix, PostgreSQL, MySQL і т.д., необхідно тільки наявність встановленого відповідного OLE-провайдера ("драйвера" відповідного типу бази даних, що встановлюється в систему як правило з дистрибутива цієї ж бази даних). Перелік властивостей і методів ADO не є вичерпним. Повний опис об'єктної моделі бібліотеки ADO можна отримати в MSDN або у файлі "ADO210.CHM", який входить в поставку MS Office.

SQL (англ. Structured query language — мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікація, система контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом. Не будучи мовою програмування в тому розумінні, як C або Pascal, SQL може формувати інтерактивні запити або, будучи вбудованою в прикладні програми, виступати в якості інструкцій для керування даними. Стандарт SQL, крім того, вміщує функції для визначення зміни, перевірки і захисту даних.

SQL — це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення, і видалення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

Перша версія SQL була розроблена на початку 1970-х років у IBM. Ця версія носила назву SEQUEL і була призначена для обробки і пошуку даних, що містилися в реляційній базі даних IBM, System R . Мова SQL пізніше була стандартизована Американськими Держстандартами (ANSI) в 1986. Спочатку SQL розроблялась як мова запитів і управління даними, пізніші модифікації SQL створено продавцями

системи управління базами даних, які додали процедурні конструкції, control-of-flow команд і розширення мов. З випуском стандарту SQL:1999 такі розширення були формально запозичені як частина мови SQL через Persistent Stored Modules (SQL/PSM).

Критики SQL включає відсутність крос-платформенності, невідповідною обробкою відсутніх даних (дивіться Null (SQL)), і іноді неоднозначна граматики і семантика мови.

Усі вище перераховані можливості дозволили зупинити свій вибір для рішення поставленої задачі у середовищі програмування Borland Delphi 10.3.

3.2 Обґрунтування вибору середовища функціонування системи

Розроблений програмний продукт може працювати в операційних системах: Windows 7 Professional, Windows 8, Windows 10, а також в операційній системі Windows 11.

Середовищем функціонування та розробки програми було обрано операційну систему Microsoft Windows 10 Professional. Їй було віддано перевагу перед іншими операційними системами тому що вона володіє усіма вимогами сучасних ОС: сумісність, переносимість, масштабованість, система безпеки, розподілена обробка, надійність і відмово-стійкість, локалізація, розширюваність.

– сумісність. Система може мати звичний інтерфейс ОС сімейства Windows, з деякими додаваннями і розширеннями, підтримку файлових систем NTFS5, NTFS4, FAT16 і FAT32. Більшість додатків, написаних під MSDOS,

W9x, NT4, а також деякі програми під OS/2 і POSIX запускаються і функціонують без проблем. При проектуванні NT враховувалася можливість роботи системи в різних мережних середовищах, тому в поставку входять засоби для роботи в Unix- і Novell-сітьях.

– переносимість (Portability). Система працює на різних процесорах сімейства x86, x64 виробництва Intel і AMD. Існує 64 бітова версія Windows 10.

Реалізація підтримки процесорів іншої архітектури можлива, але зажадає деякі зусилля.

- масштабованість (Scalability). В Windows 10 реалізована підтримка технології SMP, SMBv1, SMBv2 и SMBv3
- В Windows.NET Advanced Server і Datacenter Server окрім цього є підтримка COW (Cluster Workstations).
- система безпеки (Security). Реалізована звична для NT система безпеки на рівні користувачів.
- розподілена обробка (Distributed processing). Windows 10 має вбудовані в систему мережну нагоду, що забезпечує можливість зв'язку з різними типами комп'ютерів завдяки наявності різноманітних транспортних протоколів і технології «клієнт-сервер».
- надійність і відмовостійкість (Reliability and robustness). Архітектура Ос захищає додатки від пошкодження один одним і самою операційною системою. При цьому використовується відмовостійка структурована обробка особливих ситуацій на всіх архітектурних рівнях, яка включає відновлювану файлову систему NTFS і забезпечує захист за допомогою вбудованої системи безпеки і вдосконалених методів управління пам'яттю.
- локалізація (Localization). Система надає можливості для роботи в багатьох країнах світу на національних мовах, що досягається застосуванням стандарту ISO Unicode.
- розширюваність (Extensibility). Завдяки модульній побудові системи стає можливо додавання нових модулів на різні архітектурні рівні Ос.

Фінальний build - 2600. Зняти його можна або командою winver, або за версією ядра XP, наприклад файлу ntoskrnl.exe. Відрізнити піратський реліз від теперішнього часу можливо завдяки механізму активації (Windows Product activation) .В піратському релізі повинні бути включений засоби боротьби з нею, тому якщо Ви побачите теку з назвою crack, або ніж то на зразок цього, то можете бути уверенні, що реліз піратський. Якщо на Вашому диску немає нічого подібного, а сам диск має всі ознаки нелегального (немає голограми, ліцензійної

угоди на папері, немає наклейки, яка міняє колір якщо нагрівати її пальцем), то Ви ризикуєте тим, що Ваша версія перестане працювати в продовж місяця. Втім, є і «ламані піратські» релізи, які не вимагають реєстрації взагалі. Окрім цього, існують так звані корпоративні релізи, зроблені Microsoft для своїх найбільших OEM партнерів. Такі релізи не вимагають реєстрації спочатку. Microsoft затверджує, що для успішної інсталяції 10 Вам необхідний процесор не менше 1 гігагерц, 2 гігабайти оперативної пам'яті, і 20 гігабайт вільного місця на диску. Проте, для більш-менш комфортної роботи Вам знадобиться процесор не менше 1 гігагерц, і не менше 1 гігабайту оперативної пам'яті. Хоча, максимально полегшивши інтерфейс можна добитися того, що 10 вимагатиме менше пам'яті ніж W2k, тому якщо пожертвувати всякими "примочками", і візуальними ефектами, то цілком комфортно можна буде працювати і на менш могутніх системах. Взагалі, якщо на Вашій машині працює W2k, то працюватиме і 10, нітрохи не гірше, якщо не краще. На 16 Гб оперативної пам'яті система працює дуже швидко і дуже приємно, якщо не вантажити її дуже важкими додатками. Якщо включити всі візуальні ефекти, то 10 помітно пригальмовуватиме на відносно слабих процесорах, незалежно від об'єму оперативної пам'яті. Такий же ефект спостерігається при використуванні слабкої відеокарти. Проте, якщо система достатньо могутня, то нормально набудований 10 працюватиме набагато швидше, ніж будь-яка Ос від Microsoft випущена раніше.

Дана версія операційної системи Windows поєднує в собі переваги Windows 8 Professional (наприклад, засоби безпеки, керованість і надійність) із кращими якостями Windows 7 і Windows Vista(підтримка Plug and Play, простий користувацький інтерфейс і передові служби підтримки). Це робить Windows 10 Professional найбільш підходящою операційною системою для настільних комп'ютерів, застосовуваних у корпоративнім середовищі. Незалежно від того, де встановлюється Windows 10 Professional - на одному комп'ютері або в масштабі локальної мережі - ця система підвищує обчислювальні можливості підприємства, одночасно скорочуючи сукупну вартість програмного забезпечення всіх настільних комп'ютерів:

- новий рівень ефективності й надійності обчислювальних систем;
- швидкий доступ до передових інструментальних засобів цифрової ери;
- потужні засоби керування й підтримки, що полегшують вашу роботу.

Розглянемо переваги системи.

Існують сотні причин, які підтверджують необхідність відновлення встановленої комп'ютері операційної системи до рівня Windows 7 Professional.

Засіб Windows Messenger. Використання Windows Messenger є найпростішим способом встановлення зв'язку з користувачами мережі й проведення спільної роботи в режимі реального часу. За допомогою цього засобу можна бачити мережний статус користувачів, що входять у список контактів, і організувати спілкування з ними за допомогою текстових повідомлень, високоякісних голосових або відеозв'язків.

Вилучене керування робочим столом. За допомогою функції вилученого керування робочим столом можна використовувати свій настільний комп'ютер, одержуючи до нього доступ з іншого комп'ютера, що працює під керуванням операційної системи Windows 95 і більш пізніх версій Windows. Таким чином, користувач може одержувати доступ до всіх необхідних йому даним і додаткам, навіть перебуваючи поза своїм офісом.

Підтримка бездротових мереж стандарту 802.1x . Завдяки підтримці бездротових мереж стандарту 802.1x забезпечується безпечний доступ до системи, а також досягається значне підвищення продуктивності бездротових мереж.

Використання вилученого помічника. Використання вилученого помічника дозволяє вашому знайомому або фахівцеві з інформаційних технологій здійснювати вилучене керування вашим комп'ютером для усунення тієї або іншої проблеми або демонстрації роботи незнайомого додатка.

Керування комп'ютером. Використання технологій Intel mirror означає можливість застосування групових налаштувань і переміщуваних профілів користувачів, що значно полегшує роботу системних адміністраторів по керуванню окремими комп'ютерами.

Багатомовна підтримка. Дозволяє легко й просто створювати, читати й редагувати документи, підготовлені на різних мовах.

Технологія Dualview. Вміст робочого стола може відображатися на двох моніторах, підключених до однієї відеокарти; дана функція особливо зручна для користувачів переносних комп'ютерів.

Засіб User State Migration. За допомогою засобу User State Migration користувач може перенести свої дані й налаштування зі старого комп'ютера на новий.

Нове зовнішнє оформлення, що спрощує розв'язок завдань. Дозволяє більш швидко вирішувати типові завдання завдяки більш зрозумілому інтерфейсу й новим візуальним підказкам.

Саме ці переваги дозволили обрати операційну систему Windows 10 у якості середовища функціонування розробленого додатку.

3.3 Використані моделі даних

Для розробки даного програмного продукту використано реляційну модель даних. Реляційну модель даних було побудовано шляхом перетворення моделі «сутність-зв'язок» бази даних.

Одним з найбільш зручних способів представлення даних, незалежно від програмного забезпечення за допомогою якого буде реалізоване зберігання даних є модель «сутність-зв'язок» (entity-relationship model, ER-model), яка використовується для аналізу предметної області.

Модель «сутність-зв'язок» ґрунтується на деякій інформації про реальний світ і призначена для логічного представлення даних. Вона визначає значення даних їх взаємозв'язок з іншими даними, використовуваними в системі.

Модель «сутність-зв'язок» представляється в графічному вигляді і називається діаграмою «сутність-зв'язок». На ній представлені усі виявлені сутності предметної області і вказані взаємозв'язки між ними.

Реалізація моделі «сутність-зв'язок» наведена на рисунку 3.1.

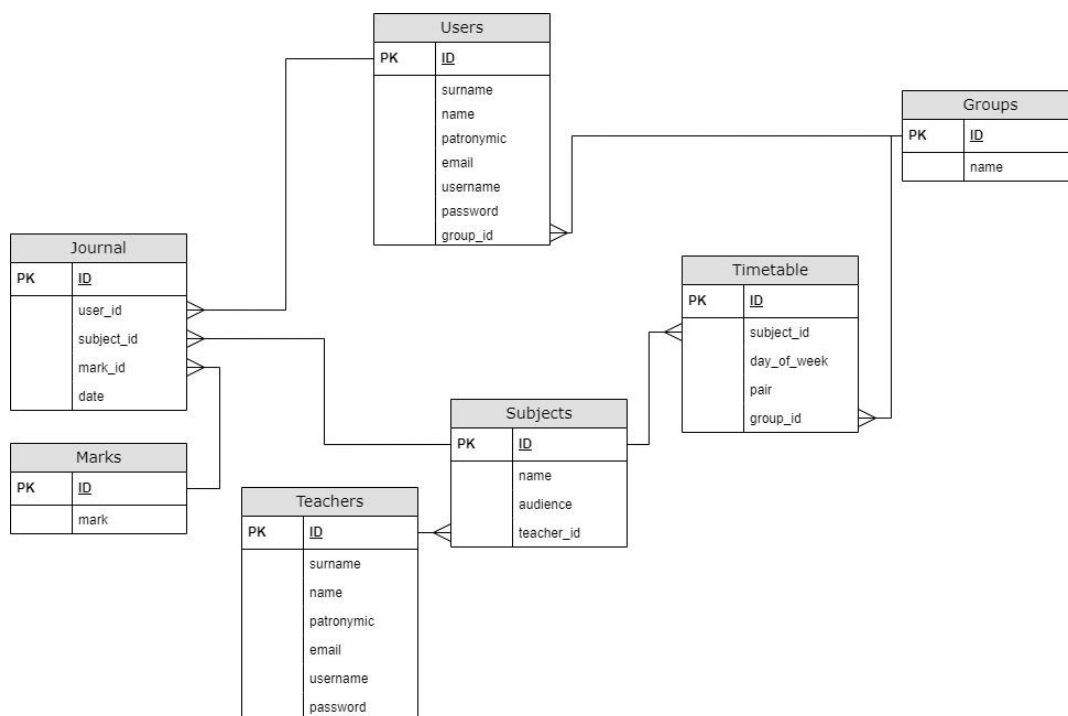


Рисунок 3.1 – ER - діаграма бази даних

В процесі проектування даного програмного продукту для побудови реляційної моделі було виділено наступні об’єктні множини: Journal, Marks, Users, Groups, Timetable, Subjects, Teachers.

Базу даних розроблено в форматі Microsoft Access. База даних вміщує сім таблиць.

Сутність таблиці «Marks» (ID, mark). Сутність таблиці призначена для зберігання даних про відмітки, які використовуються в таблиці «Journal». У реалізації бази даних сутність представлена таблицею «Marks» (табл. 3.1).

Таблиця 3.1 - Структура даних «Marks»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
mark	Короткий текст	20	-	Призначено для зберігання інформації про відмітку

Сутність таблиці «Journal» (ID, user_id, subject_id, mark_id, date). Сутність таблиці призначена для зберігання даних про відмітки учнів. У реалізації бази даних сутність представлена таблицею «Journal» (табл. 3.2).

Таблиця 3.2 - Структура даних «Journal»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
user_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID учня
subject_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID предмету
mark_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID відмітки
date	Дата та час	Короткий формат дати	-	Призначено для зберігання інформації про дату відмітки

Сутність таблиці «Groups» (ID, name). Сутність таблиці призначена для зберігання даних про групи учнів. У реалізації бази даних сутність представлена таблицею «Groups» (табл. 3.3).

Таблиця 3.3 - Структура даних «Groups»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
name	Короткий текст	20	-	Призначено для зберігання інформації про назву групи

Сутність таблиці «Users» (ID, surname, name, patronymic, email, username, password, group_id). Сутність таблиці призначена для зберігання даних про користувачів сервісу. У реалізації бази даних сутність представлена таблицею «Users» (табл. 3.4).

Таблиця 3.4 - Структура даних «Users»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
surname	Короткий текст	20	-	Призначено для зберігання інформації про прізвище
name	Короткий текст	20	-	Призначено для зберігання інформації про ім'я
patronymic	Короткий текст	20	-	Призначено для зберігання інформації про по-батькові
email	Короткий текст	30	-	Призначено для зберігання інформації про пошту
username	Короткий текст	20	-	Призначено для зберігання інформації про логін
password	Короткий текст	20	-	Призначено для зберігання інформації про пароль
group_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID групи

Сутність таблиці «Subjects» (ID, name, audience, teacher_id). Сутність таблиці призначена для зберігання даних про предмети. У реалізації бази даних сутність представлена таблицею «Subjects» (табл. 3.5).

Таблиця 3.5 - Структура даних «Subjects»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
name	Короткий текст	30	-	Призначено для зберігання інформації про назву предмету
audience	Числовий	Довге ціле	-	Призначено для зберігання інформації про аудиторію
teacher_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про вчителя

Сутність таблиці «Teachers» (ID, surname, name, patronymic, email, username, password). Сутність таблиці призначена для зберігання даних про предмети. У реалізації бази даних сутність представлена таблицею «Teachers» (табл. 3.6).

Таблиця 3.6 - Структура даних «Teachers»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
surname	Короткий текст	20	-	Призначено для зберігання інформації про прізвище
name	Короткий текст	20	-	Призначено для зберігання інформації про ім'я
patronymic	Короткий текст	20	-	Призначено для зберігання інформації про по-батькові
email	Короткий текст	30		Призначено для зберігання інформації про пошту
username	Короткий текст	20		Призначено для зберігання інформації про логін
password	Короткий текст	20		Призначено для зберігання інформації про пароль

Сутність таблиці «Timetable» (ID, subject_id, day_of_week, pair, group_id). Сутність таблиці призначена для зберігання даних про предмети. У реалізації бази даних сутність представлена таблицею «Timetable» (табл. 3.7).

Таблиця 3.7 - Структура даних «Timetable»

Поле	Тип даних	Розмір поля	Ключ	Призначення
ID	Лічильник	Довге ціле	+	Унікальний ідентифікатор
subject_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID предмету

Продовження таблиці 3.7

Поле	Тип даних	Розмір поля	Ключ	Призначення
day_of_week	Числовий	Довге ціле	-	Призначено для зберігання інформації про номер дня тижня
pair	Числовий	Довге ціле	-	Призначено для зберігання інформації про номер пари
group_id	Числовий	Довге ціле	-	Призначено для зберігання інформації про ID групи

Структуру реляційної моделі бази даних представлено на рисунку 3.2.

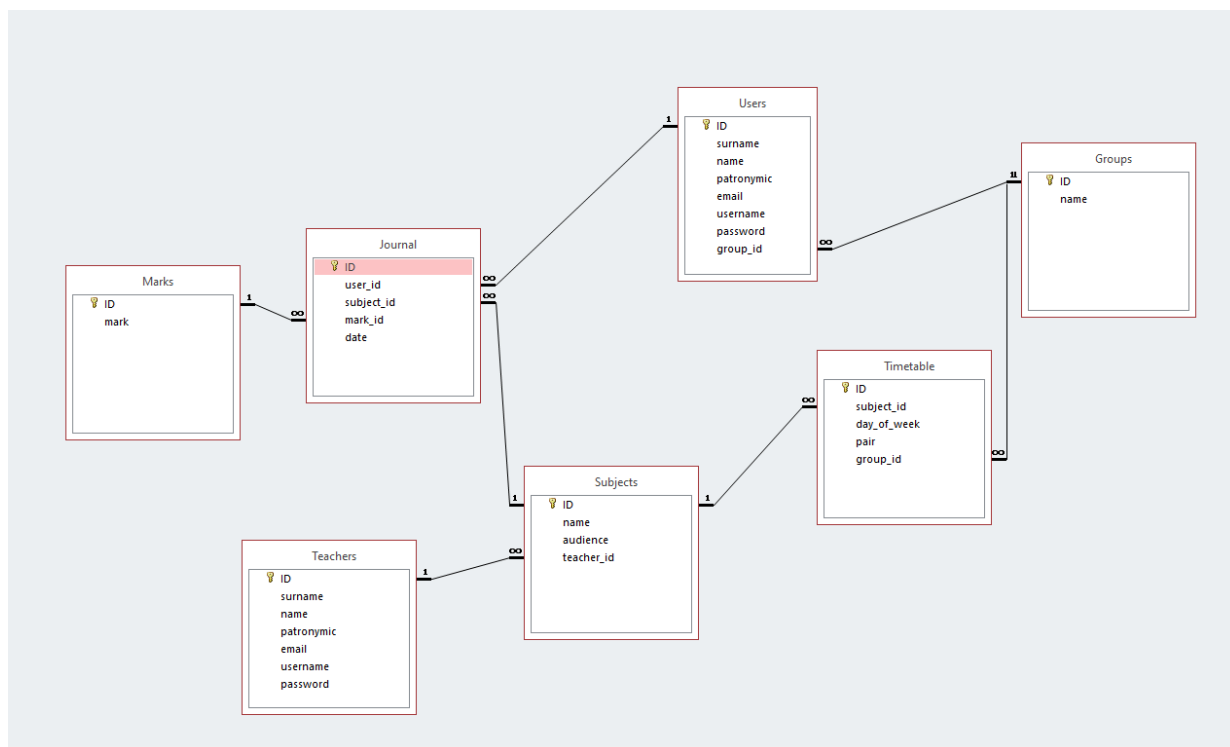


Рисунок 3.2 – Реляційна модель бази даних

4 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

4.1 Керівництво програміста

Програма призначена для автоматизації шкільного журналу. Додаток було створено за допомогою середовища Borland Delphi 10.3. База даних створена в СКБД Access.

Підключення таблиць бази даних на форми проекту виконано за допомогою Microsoft Jet 4.0 OLE DB Provider.

Контейнером даних являються компоненти ADO.

Через DataModule організоване з'єднання ADO з відповідними таблицями бази даних.

Умовою надійної експлуатації програми є наявність бази даних в папці з відповідним .exe- файлом.

Для створення запитів використовувався вбудований відладник SQL. Всі запити написані програмно, за виключенням тих, що використовуються при створенні звітів.

Для нормальної роботи потрібний персональний комп'ютер з мікропроцесором Intel i3 3,30 GHz та вище, оперативна пам'ять 4Гб і вище, вільного простору на жорсткому диску 5Гб, SVGA-монітор, клавіатура та маніпулятор типу "миша". Для виводу на друк інформації потрібен принтер.

Програма виконує наступні функції:

- доступ до таблиць бази даних;
- редагування інформації в таблицях;
- оформлення груп учнів;
- пошук оцінок по даті;
- пошук уроку в розкладі;
- пошук потрібного учня за прізвищем або групою;
- виконання запитів з таблиць за різними критеріями;
- виведення звітів;
- друк інформації.

Розроблений проект складається з 9 Units (юнітів):

- Main – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Journal»;
- DataModule1 – містить в собі компоненти, що не відображуються в вікні програми, виступає контейнером для зберігання цих компонентів, та полегшує доступ до них іншим юнітам;
- Subjects – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Subjects»;
- Timetable – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Timetable»;
- Groups – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Groups»;
- Users – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Users»;
- Teachers – модуль призначений для того, щоб робити зміни та пошуки в таблиці «Teachers»;
- TimeTableReport – цей модуль призначений для формування звіту таблиці «Розклад»;
- Authorization – цей модуль призначений для вибору типу користувача та вводу логіна та пароля

4.2 Керівництво оператора

Для запуску додатку потрібно відкрити файл ElectronicJournal.exe, після чого на екрані з'явиться вікно додатку, яке наведено на рисунку 4.1. На головному вікні існує меню, яке викликає інші вікна.

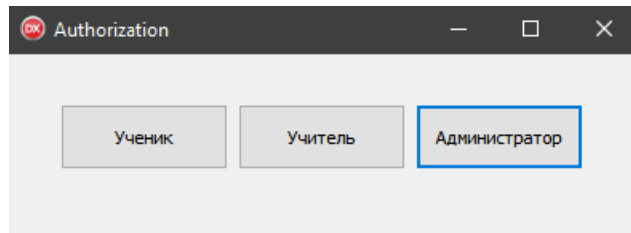


Рисунок 4.1 – Вигляд вікна авторизації

Після натискання на вибраний тип користувача з'являється поле для введення логіну (рисунок 4.2), а потім поле для введення паролю (рисунок 4.3)

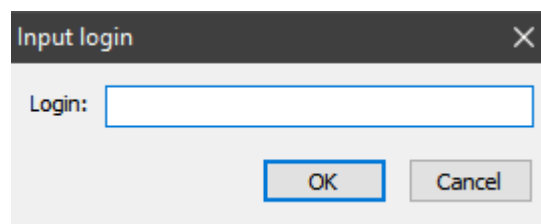


Рисунок 4.2 – Вигляд вікна для вводу логіну

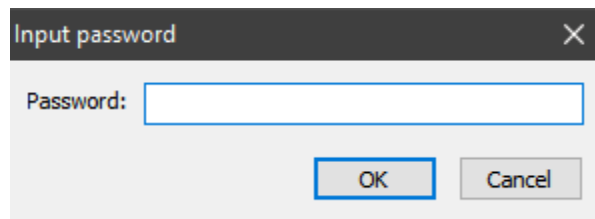


Рисунок 4.3 – Вигляд вікна для вводу паролю

При виникненні якоїсь помилки при вводі даних користувачем на формі з'являється напис, який вказує на цю помилку (рисунок 4.4)

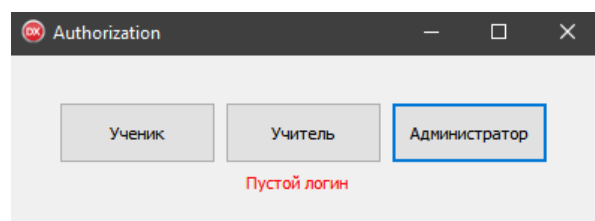


Рисунок 4.4 – Вигляд вікна авторизації при помилці

Після успішної авторизації користувача з'являється головна форма програми (рисунок 4.5). Після натискання на кнопку «Изменить», яка доступна тільки вчителю та адміністратору, з'являється панель в якій можна добавляти / змінювати / видаляти оцінки учнів. Всім користувачам доступна можливість змінити поточний місяць, оцінки за який відображаються. Вчителю доступна зміна групи, оцінки якої відображаються в таблиці. Учнію – змінна предмету.

Після натискання на кнопки «Добавить оценку», «Изменить оценку», «Удалить оценку» виконується перевірка даних, а точніше наявність вибраного учня, оцінки та перевірка дати (якщо в день тижня цієї дати немає уроку – видається помилка)

При закритті форми через хрестик, а не через кнопку меню «Exit» знову з'явиться форма авторизації

fullname	1	6	8	13	15	20	22	27	29
Баранова Алиса									
Белова Анастасия									
Беляева Дарья									
Васильев Андрей									
Власова Кристина									
Герасимов Алексей									
Давыдов Илья									
Давыдов Никита									
► Данилов Максим									
Ефимов Даниил									
Захарова Александра									
Игнатьева Дарья									
Казакова Тамара									
Козлова Вероника									
Копылов Егор									
Попов Андрей									
Прокофьев Дмитрий									
Прохорова Софья									
Сергеева Екатерина									
Филатов Максим									

Группа: 9-A Сентябрь Изменить

Предмет: Биология

Студент: Данилов Добавить оценку

Оценка: **Укажите ученика!** Изменить оценку

08.09.2021 Удалить оценку

Понедельник Среда

Рисунок 4.5 – Вигляд вікна програми «Main»

При виборі пункту меню «Subjects» відкривається форма Subjects (рисунок 4.6). На цій формі вчитель та учень можуть переглянути список предметів, а адміністратор може змінити дані предмету, додати новий або ж зовсім його видалити. Якщо адміністратор забуде вказати якийсь параметр при зміні, додаванні та видаленні – виникне помилка.

id	name	audience	teacher
1	Алгебра		1 Гусева Алиса
2	Геометрия		2 Белов Фёдор
3	Химия		3 Борисов Марк
4	Физика		4 Агеева Анна
5	Украинский язык		5 Чернышев Максим
6	Физкультура		6 Панина Кристина
7	Украинская литература		7 Савельева Елизавета
8	Зарубежная литература		8 Ефремов Святослав
9	Биология		9 Дроздова Ольга
10	Английский язык		10 Лукин Лукин

Изменить

Название:

Аудитория:

Учитель:

Укажите название!

Добавить предмет

Изменить предмет

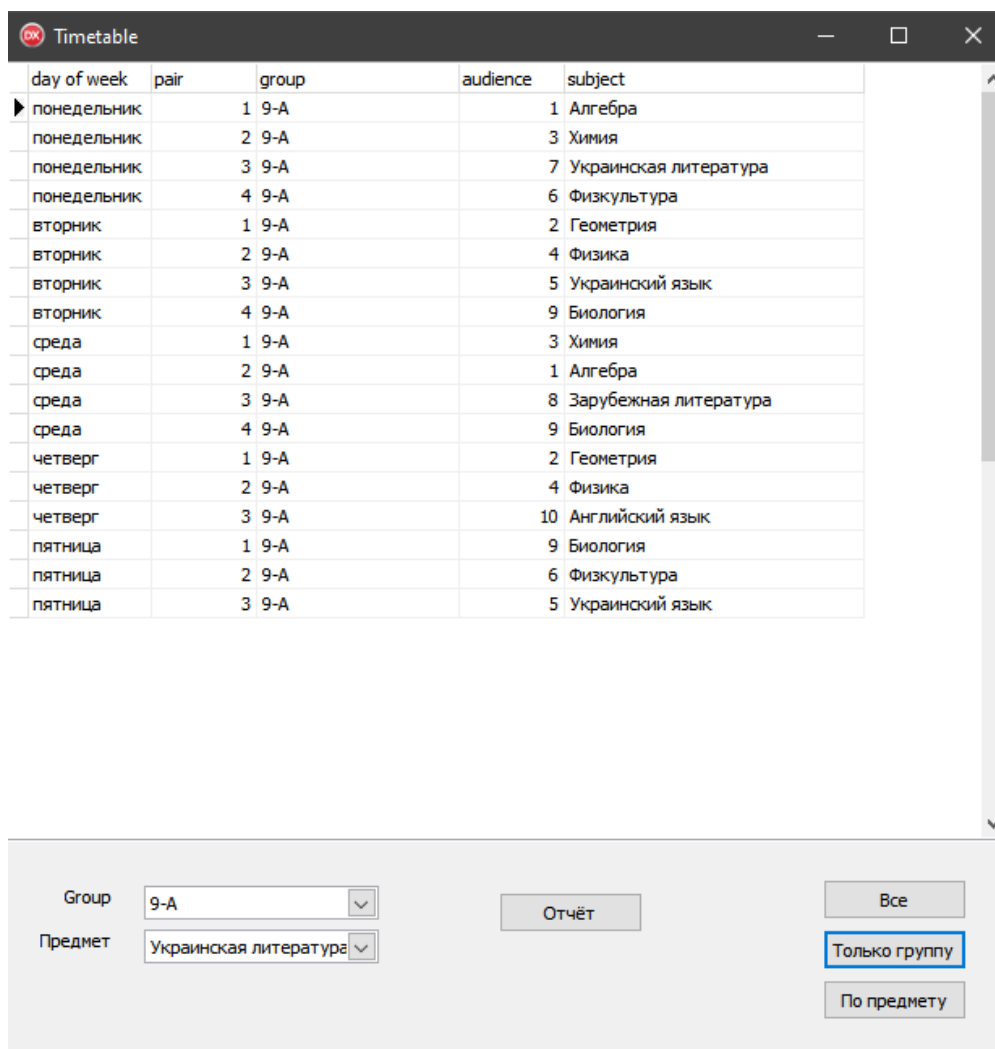
Удалить предмет

Очистить поля

Рисунок 4.6 – Видяд вікна програми «Subjects»

При натисненні на кнопку меню «Timetable» - відкриється форма Timetable (рисунок 4.7). Вчителю стане доступний для перегляду тільки розклад уроків за його предметом; учню доступний для перегляду тільки розклад уроків за його групою.

На формі реалізований функціонал сортування по групі та по предмету, який доступний тільки для адміністратора. При натисненні кнопки «Отчёт» відображається форма з звітом «TimeTableReport» (рисунок 4.8)



The screenshot shows a window titled 'Timetable' with a table of weekly schedules. The table has columns: day of week, pair, group, audience, and subject. Below the table is a control panel with dropdown menus for 'Group' and 'Предмет', and buttons for 'Отчёт', 'Все', 'Только группу', and 'По предмету'.

day of week	pair	group	audience	subject
▶ понедельник		1 9-A		1 Алгебра
понедельник		2 9-A		3 Химия
понедельник		3 9-A		7 Украинская литература
понедельник		4 9-A		6 Физкультура
вторник		1 9-A		2 Геометрия
вторник		2 9-A		4 Физика
вторник		3 9-A		5 Украинский язык
вторник		4 9-A		9 Биология
среда		1 9-A		3 Химия
среда		2 9-A		1 Алгебра
среда		3 9-A		8 Зарубежная литература
среда		4 9-A		9 Биология
четверг		1 9-A		2 Геометрия
четверг		2 9-A		4 Физика
четверг		3 9-A		10 Английский язык
пятница		1 9-A		9 Биология
пятница		2 9-A		6 Физкультура
пятница		3 9-A		5 Украинский язык

Control Panel:

- Group: 9-A
- Предмет: Украинская литература
- Buttons: Отчёт, Все, **Только группу**, По предмету

Рисунок 4.7 – Вигляд вікна програми «Timetable»

При натисненні на кнопку меню «Groups» - відкриється форма Groups (рисунок 4.9).

Форма перегляду та редагування груп доступна тільки для адміністратора. При додаванні або змінні групи без указування назви – виникає помилка.

Рассписание				
День недели	Номер пары	Название группы	Аудитория	Название предмета
понедельник	1	9-А	1	Алгебра
понедельник	2	9-А	3	Химия
понедельник	3	9-А	7	Украинская литература
понедельник	4	9-А	6	Физкультура
вторник	1	9-А	2	Геометрия
вторник	2	9-А	4	Физика
вторник	3	9-А	5	Украинский язык
вторник	4	9-А	9	Биология
среда	1	9-А	3	Химия
среда	2	9-А	1	Алгебра
среда	3	9-А	8	Зарубежная литература
среда	4	9-А	9	Биология
четверг	1	9-А	2	Геометрия
четверг	2	9-А	4	Физика
четверг	3	9-А	10	Английский язык
пятница	1	9-А	9	Биология
пятница	2	9-А	6	Физкультура
пятница	3	9-А	5	Украинский язык

Рисунок 4.8 – Вигляд звіту «TimeTableReport»

Groups

ID	name
1	9-A
2	9-B
3	10-A
4	10-B
5	11

Изменить

Название:

Укажите название!

Добавить группу

Изменить группу

Удалить группу

Рисунок 4.9 – Вигляд вікна програми «Groups»

При натисненні на кнопку меню «Users» - відкриється форма Users (рисунк 4.10). Для учнів доступний перегляд інформації (не включаючи пароль та логін) без зміни групи Для вчителів доступний перегляд інформації (не включаючи пароль та логін) з змінною групи. Для адміністраторів – доступний перегляд інформації включаючи паролі, доступний переключач для видимості паролю і логіну, а також змінна всіх даних які є.

The screenshot shows a window titled 'Users' with a table of users and a form below it.

id	GN	surname	name	patronymic	email
21	9-Б	Нестерова	Софья	Владимировна	tiguan14@gmail.com
22	9-Б	Жилин	Максим	Кириллович	saagar.sinha@gmail.com
23	9-Б	Гаврилов	Никита	Георгиевич	umaralmohammed@gmail.com
24	9-Б	Смирнова	Татьяна	Михайловна	rohit.2241983@gmail.com
25	9-Б	Лебедева	Лилия	Артёмовна	bharath.kr@gmail.com
26	9-Б	Панина	Валерия	Марковна	julio.tan@gmail.com
27	9-Б	Костин	Михаил	Давидович	ivanbouchot@gmail.com
28	9-Б	Карпов	Владимир	Владиславович	dean.watkin@gmail.com
29	9-Б	Васильев	Владимир	Олегович	ria1@gmail.com
30	9-Б	Киселев	Константин	Егорович	revitn1@gmail.com
31	9-Б	Комиссаров	Али	Иванович	shengyin1377@gmail.com
32	9-Б	Борисова	Екатерина	Артёмовна	carcynsmommy2008@gmail.com
33	9-Б	Ларионов	Константин	Леонидович	maggiebit@gmail.com
34	9-Б	Цветкова	Екатерина	Михайловна	hardcorekicks@gmail.com
35	9-Б	Сафонов	Григорий	Александрович	riahgirl08@gmail.com
36	9-Б	Гордеев	Сергей	Антонович	robbynebel@gmail.com
37	9-Б	Бычкова	Виктория	Фёдоровна	brickpei@gmail.com
38	9-Б	Родионов	Марк	Данилович	ekayuliyanti@gmail.com
39	9-Б	Безрукова	Мария	Дмитриевна	allforanezka@gmail.com
40	9-Б	Петров	Виктор	Данилович	chirag.dekavadiya@gmail.com

Below the table, there is a form with the following elements:

- A dropdown menu for 'Группа' (Group) with '9-Б' selected.
- A checkbox for 'Password show'.
- An 'Изменить' (Edit) button.
- Input fields for 'ID:', 'Группа:', 'Фамилия:', 'Имя:', 'Отчество:', and 'Email:'.
- Buttons: 'Добавить нового ученика' (Add new student), 'Изменить данные ученика' (Edit student data), 'Удалить данные ученика' (Delete student data), and 'Очистить поля' (Clear fields).
- A red error message: 'Ошибка фамилии' (Surname error).

Рисунок 4.10 – Видяг вікна програми «Users»

При натисненні на кнопку меню «Teachers» - відкриється форма Teachers (рисунк 4.11). Для учнів та вчителів доступний перегляд інформації (не включаючи пароль та логін); для адміністраторів – доступний перегляд інформації

включаючи паролі, доступний переключач для видимості паролю і логіну, а також змінна всіх даних які є.

The screenshot shows a window titled 'Teachers' with a table of teachers and a form for editing a selected teacher's details.

ID	surname	name	patronymic	email	username	password
1	Гусева	Алиса	Львовна	crazyereka@gmail.com	archardo	mqLMuWGN
2	Белов	Фёдор	Дмитриевич	mslokobi@gmail.com	test	test
3	Борисов	Марк	Фёдорович	benberfield@gmail.com	areartic	BPcNeCgZ
4	Агеева	Анна	Михайловна	yotam6030@gmail.com	tuseamen	hVscMA3L
5	Чернышев	Максим	Маркович	pawel.babut@gmail.com	ationver	RAnzJdyn
6	Панина	Кристина	Александровна	ann.carpenter23@gmail.com	ndachuni	udZXAGZk
7	Савельева	Елизавета	Тимуровна	rizlack@gmail.com	repraegl	wKTKawG5
8	Ефремов	Святослав	Маркович	nguyenngocminh.169@gmail.com	oysurobs	bseFJULj
9	Дроздова	Ольга	Марковна	alexfincham@gmail.com	tronarbe	4EMnQaKQ
11	Лукин	Лукин	Иванович	dstehling@gmail.com	iandwrab	StXhcFZh

Below the table, there is a checkbox labeled 'Password show' and a button labeled 'Изменить'.

The form for editing a teacher's details includes the following fields and buttons:

- ID:
- Фамилия:
- Имя:
- Отчество:
- Email:
- Username:
- Password:
- Buttons: 'Добавить нового учителя', 'Изменить данные учителя', 'Удалить данные учителя', 'Очистить поля'

Рисунок 4.11 – Видяг вікна програми «Teachers»

ВИСНОВКИ

В результаті виконаної курсової роботи було створено автоматизовану систему керування шкільним електроним журналом.

Використання автоматизованої системи має підвищити зручність вчителям виставлення та учням перегляд оцінок. Розроблений програмний додаток легкий у користуванні та реалізує можливість бачити розклад уроків вчителів, доступ до їх emails, а також даних про учнів.

Програмний продукт було розроблено в середовищі програмування Borland Delphi 10.3

Програма працює на IBM-сумісних ЕОМ під керуванням операційних систем Microsoft Windows 7, 8, 10, 11.

Програма має зручний у використанні інтерфейс, не потребує багато часу на опанування принципів роботи з ним.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 Гофман В. Э., Хомоненко А. Д. Delphi. Быстрый старт. - СПб: БХВПетербург, 2003. - 288 с.
- 2 Шупрута В. В. Delphi 2005. Учимся программировать - М.: НТ Пресс, 2005. - 352 с.
- 3 Сухарев М. Основы Delphi. Профессиональный подход - СПб.: Наука и Техника, 2004 - 603с.
- 4 Віктор Пестриков, Артур Маслобоев. Delphi на примерах - СПб.: БХВПетербург, 2005 - 1156с.
- 5 Таненбаум Э. Современные операционные системы. 2-е изд. - СПб: Питер, 2002. - 1040 с.
- 6 Гэри Хансен, Джэймс Хансен. Базы данных: разработка и управление: Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 1999 - 704с.
- 7 Чен П. Модель "сущность-связь" - шаг к единому представлению о данных //СУБД. - СПб: Питер, 1995. - 203 с.
- 8 Кириллов В.В. Структуризованный язык запросов (SQL). - СПб.: ИТМО, 1994. - 80 с.
- 9 Кузнецов С.Д. Стандарты языка реляционных баз данных SQL: краткий обзор //СУБД. - СПб: Питер, 1996. - 242 с.
- 10 Дейт К. Руководство по реляционной СУБД DB2. - М.: Финансы и статистика, 1988. - 320 с.
- 11 Цикритизис Д., Лоховски Ф. Модели данных. - М.: Финансы и статистика, 1985. - 344 с.

ДОДАТОК А
(обов'язковий)
Текст програми

```

unit Unit1;
const CURRENT_SEMESTER = 1; CURRENT_YEAR = 2021; MOUNTH_COUNT = 5;
var Main: TMain; groupId, subjectId: Integer; arrayDaysOfWeek: array of Integer; currentMounth: Integer; mounthArr:
array[1..MOUNTH_COUNT] of String; selectDates: String;
uses Unit2, StrUtils, DateUtils, Unit3, Unit4, Unit5, Unit6, Unit7, Unit9;
procedure TMain.selectGroupClick(Sender: TObject); begin updateGrid(); end;
procedure TMain.selectSubjectClick(Sender: TObject); begin updateGrid(); end;
procedure TMain.showMainTable(groupId, subjectId: Integer);
var firstDate, endDate, text: String; i: Integer;
begin
    firstDate := '1.' + IntToStr(currentMounth) + '.' + IntToStr(CURRENT_YEAR);
    endDate := IntToStr(DaysInAMonth(CURRENT_YEAR, currentMounth)) + '.' + IntToStr(currentMounth) + '.' +
IntToStr(CURRENT_YEAR);
    text := 'TRANSFORM Max(m.mark) AS [Max-mark_id] SELECT t.fullname FROM (SELECT testing1.fullname,
IIf(testing1.date = testing3.firs, testing1.mark_id, Null) AS mark_id, testing3.firs FROM ( SELECT (u.surname & " " &
u.name) AS fullname, j.mark_id, j.date FROM ( SELECT * FROM Journal WHERE ( Journal.subject_id = ' +
IntToStr(subjectId) + ' AND Journal.date BETWEEN DateValue("'" + firstDate + "' ) AND DateValue("'" + endDate + "' ) ) )
AS j RIGHT JOIN ( SELECT Users.id AS id, Users.surname as surname, Users.name AS name, Users.patronymic AS
patronymic FROM Users INNER JOIN Groups ON Users.group_id = Groups.id WHERE Groups.id = ' +
IntToStr(groupId) + ' ORDER BY Users.id ) AS u ON j.user_id = u.id) AS testing1, ( SELECT dates.firs FROM (
SELECT Timetable.day_of_week FROM Timetable, Subjects, Groups WHERE ( Subjects.id = Timetable.subject_id AND
Groups.id = Timetable.group_id AND Subjects.id = ' + IntToStr(subjectId) + ' AND Groups.id = ' + IntToStr(groupId) + '
)) AS Timetable_get_group_subject, (' + selectDates + ') AS dates WHERE Timetable_get_group_subject.day_of_week = (
Weekday(dates.firs) - 1 ) ORDER BY dates.firs ) AS testing3 ) AS t LEFT JOIN Marks AS m ON t.mark_id = m.id
GROUP BY t.fullname PIVOT DAY(t.firs)';
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text := text;
    DataModule1.ADOQueryMain.Open;
    for i := 0 to DbGrid1.Columns.Count - 1 do
        DBGrid1.Columns[i].Width := 25;
    DBGrid1.Columns[0].Width := 200;
end;
procedure TMain.TeachersTabClick(Sender: TObject); begin Teachers.Show(); end;
function TMain.getGroupId(groupName: String): Integer;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
'SELECT * ' +
'FROM Groups ' +
'WHERE name LIKE "' + groupName + '"';
    DataModule1.ADOQueryMain.Open;
    getGroupId := DataModule1.DataSourceMain.DataSet.Fields[0].AsInteger;
end;
function TMain.getSubjectId(subjectName: String): Integer;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
'SELECT * ' +
'FROM Subjects ' +
'WHERE name LIKE "' + subjectName + '"';
    DataModule1.ADOQueryMain.Open;
    getSubjectId := DataModule1.DataSourceMain.DataSet.Fields[0].AsInteger;
end;
function TMain.getSubjectNameFromTeacherId(teacherId: Integer): String;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
'SELECT name ' +
'FROM Subjects ' +
'WHERE teacher_id = ' + IntToStr(teacherId) + ';';
    DataModule1.ADOQueryMain.Open;
    getSubjectNameFromTeacherId := DataModule1.DataSourceMain.DataSet.Fields[0].AsString;
end;

```

```

function TMain.getGroupNameFromStudentId(studentId: Integer): String;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
        'SELECT g.name ' +
        'FROM Users AS u ' +
        'INNER JOIN Groups AS g ON g.id = u.group_id ' +
        'WHERE u.id = ' + IntToStr(studentId) + ' ';
    DataModule1.ADOQueryMain.Open;
    getGroupNameFromStudentId := DataModule1.DataSourceMain.DataSet.Fields[0].AsString;
end;

procedure TMain.Groups1Click(Sender: TObject);
begin
    Groups.Show();
end;

procedure TMain.timetable1Click(Sender: TObject);
begin
    Timetable.Show();
end;

procedure TMain.doQueryDaysOfWeek(groupId, subjectId: Integer);
begin
    DataModule1.ADOQueryTimetableGet.Close;
    DataModule1.ADOQueryTimetableGet.SQL.Text :=
        'SELECT day_of_week '
        + 'FROM Timetable '
        + 'WHERE (group_id = ' + IntToStr(groupId) + ') AND '
        + '(subject_id = ' + IntToStr(subjectId) + ')';
    DataModule1.ADOQueryTimetableGet.Open;
end;

procedure TMain.updateArrayDaysOfWeek(groupId, subjectId: Integer);
var
    len, i: Integer;
begin
    doQueryDaysOfWeek(groupId, subjectId);
    len := DataModule1.DataSourceTimetableGet.DataSet.RecordCount;
    SetLength(arrayDaysOfWeek, len);
    DataModule1.DataSourceTimetableGet.DataSet.First();
    while not DataModule1.DataSourceTimetableGet.DataSet.Eof do begin
        for i := 0 to len - 1 do
            begin
                arrayDaysOfWeek[i] :=
                    DataModule1.DataSourceTimetableGet.DataSet.FieldByName('day_of_week').AsInteger;
                DataModule1.DataSourceTimetableGet.DataSet.Next();
            end;
        end;
    end;

    procedure TMain.updateStringDaysOfWeek(groupId, subjectId: Integer);
    var
        len, i: Integer;
        result: String;
    begin
        updateArrayDaysOfWeek(groupId, subjectId);
        len := Length(arrayDaysOfWeek);
        result := '';
        for i := 0 to len - 1 do
            begin
                case(arrayDaysOfWeek[i]) of
                    1: result := result + ' Понедельник';
                    2: result := result + ' Вторник';
                    3: result := result + ' Среда';
                    4: result := result + ' Четверг';
                    5: result := result + ' Пятница';
                end;
            end;
        end;
    end;
end;

```

```

end;
daysOfWeekLabel.Caption := result;
end;
procedure TMain.Users1Click(Sender: TObject);
begin
    Users.Show();
end;
procedure TMain.doQueryStudentsFromGroup(groupId: Integer);
begin
    DataModule1.ADOQueryStudentsFromGroup.Close;
    DataModule1.ADOQueryStudentsFromGroup.SQL.Text :=
'SELECT '
+ 'Users.id AS id, '
+ 'Users.surname as surname, '
+ 'Users.name AS name, '
+ 'Users.patronymic AS patronymic '
+ 'FROM Users '
+ 'INNER JOIN Groups ON Users.group_id = Groups.id '
+ 'WHERE Groups.id = ' + IntToStr(groupId) + ' '
+ 'ORDER BY Users.id ';
    DataModule1.ADOQueryStudentsFromGroup.Open;
end;
procedure TMain.ExitTabClick(Sender: TObject);
begin
    Authorization.Show();
    Authorization.Close();
end;
procedure TMain.updateGrid();
var
    group, subject: String;
begin
    group := selectGroup.KeyValue;
    subject := selectSubject.KeyValue;
    if NOT (group = "") then
        groupId := getGroupId(selectGroup.KeyValue)
    else
        selectGroup.KeyValue := DataModule1.ADOTableGroups.FieldByName('name').AsString;
    if NOT (subject = "") then
        subjectId := getSubjectId(subject)
    else
        selectSubject.KeyValue := DataModule1.ADOTableSubjects.FieldByName('name').AsString;
    showMainTable(groupId, subjectId);
    updateStringDaysOfWeek(groupId, subjectId);
    doQueryStudentsFromGroup(groupId);
end;
procedure TMain.openPanelClick(Sender: TObject);
begin
    errorLabel.Visible := False;
    Panel1.Visible := NOT Panel1.Visible;
    selectStudent.KeyValue := "";
end;
function TMain.getMarkId(markName: String): Integer;
begin
    DataModule1.ADOQueryAddMarks.Close;
    DataModule1.ADOQueryAddMarks.SQL.Text :=
'SELECT * ' +
'FROM Marks ' +
'WHERE mark LIKE "' + markName + '"';
    DataModule1.ADOQueryAddMarks.Open;
    getMarkId := DataModule1.DataSourceAddMarks.DataSet.Fields[0].AsInteger;
end;
function TMain.getStudentId(studentSurname: String): Integer;
begin

```

```

DataModule1.ADOQueryAddMarks.Close;
DataModule1.ADOQueryAddMarks.SQL.Text :=
'SELECT id ' +
'FROM Users ' +
'WHERE surname = "' + studentSurname + '"';
DataModule1.ADOQueryAddMarks.Open;
getStudentID := DataModule1.DataSourceAddMarks.DataSet.Fields[0].AsInteger;
end;
procedure TMain.addRecordInJournal(studentID, subjectID, markID: Integer; date: TDateTime);
begin
DataModule1.ADOQueryAddMarks.Close;
DataModule1.ADOQueryAddMarks.SQL.Text :=
'INSERT INTO Journal (user_id, subject_id, mark_id, [date]) ' +
'VALUES (' + IntToStr(studentID) + ', ' + IntToStr(subjectID) +
', ' + IntToStr(markID)
+ ', DateValue("' + DateTimeToStr(date) + '"))';
DataModule1.ADOQueryAddMarks.ExecSQL;
end;
procedure TMain.updateRecordInJournal(studentID, subjectID, markID: Integer; date: TDateTime);
begin
DataModule1.ADOQueryAddMarks.Close;
DataModule1.ADOQueryAddMarks.SQL.Text :=
'UPDATE Journal ' +
'SET ' +
'mark_id = ' + IntToStr(markID) + ' ' +
'WHERE ' +
'user_id = ' + IntToStr(studentID) + ' AND ' +
'subject_id = ' + IntToStr(subjectID) + ' AND ' +
'date = DateValue("' + DateTimeToStr(date) + '")';
DataModule1.ADOQueryAddMarks.ExecSQL;
end;
procedure TMain.deleteRecordFromJournal(studentID, subjectID: Integer; date: TDateTime);
begin
DataModule1.ADOQueryAddMarks.Close;
DataModule1.ADOQueryAddMarks.SQL.Text :=
'DELETE ' +
'FROM Journal ' +
'WHERE ' +
'user_id = ' + IntToStr(studentID) + ' AND ' +
'subject_id = ' + IntToStr(subjectID) + ' AND ' +
'date = DateValue("' + DateTimeToStr(date) + '")';
DataModule1.ADOQueryAddMarks.ExecSQL;
end;
function TMain.checkValueInArray(arr: array of Integer; value: Integer): Boolean;
var
len, i: Integer;
begin
len := Length(arr);
for i := 0 to len - 1 do begin
if (arr[i] = value) then begin
checkValueInArray := true;
Exit;
end;
end;
checkValueInArray := false;
end;
function TMain.journalActionControllerCheckOnError(action, student, mark: String; date: TDateTime; errorString:
TLabel): Boolean;
var
selectedDayNumber: Integer;
begin
if (student = "") then begin
errorString.Visible := True;

```

```

    errorString.Caption := 'Укажите ученика!';
    journalActionControllerCheckOnError := True;
    Exit;
end;
if ((mark = '') AND NOT (action = 'delete')) then begin
    errorString.Visible := True;
    errorString.Caption := 'Укажите оценку!';
    journalActionControllerCheckOnError := True;
    Exit;
end;
selectedDayNumber := dayofweek(date) - 1;
if (NOT checkValueInArray(arrayDaysOfWeek, selectedDayNumber)) then begin
    errorString.Visible := True;
    errorString.Caption := 'Неверная дата!';
    journalActionControllerCheckOnError := True;
    Exit;
end;
journalActionControllerCheckOnError := False;
end;
function TMain.getSurname(fullName: String): String;
var
    position: Integer;
begin
    position := Pos(' ', fullName);
    getSurname := Copy(fullName, 0, position);
end;
procedure TMain.journalOnClick(Column: TColumn);
var
    colName, selectedDateText, str, surname: String;
    position: Integer;
begin
    selectStudent.KeyValue := getSurname(DBGrid1.Fields[0].AsString);
    selectMark.KeyValue := DBGrid1.Fields[Column.Index].AsString;
    colName := Column.DisplayName;
    selectedDateText := colName + ' ' + IntToStr(currentMounth) + ' ' + IntToStr(CURRENT_YEAR);
    if NOT (colName = 'fullname') then begin
        selectDate.Date := StrToDate(selectedDateText);
    end;
end;
procedure TMain.Main2Click(Sender: TObject);
begin
    Subjects.Show();
end;
function TMain.checkRecordInDataBase(studentID, subjectID: Integer; date: TDateTime): Integer;
begin
    DataModule1.ADOQueryAddMarks.Close;
    DataModule1.ADOQueryAddMarks.SQL.Text :=
        'SELECT * ' +
        'FROM Journal ' +
        'WHERE ' +
        'user_id = ' + IntToStr(studentID) + ' AND ' +
        'subject_id = ' + IntToStr(subjectID) + ' AND ' +
        'date = DateValue('' + DateTimeToStr(date) + '');';
    DataModule1.ADOQueryAddMarks.Open;
    checkRecordInDataBase := DataModule1.DataSourceAddMarks.DataSet.Fields[0].AsInteger;
end;
procedure TMain.journalActionController(action: String; subjectId, groupId: Integer; studentComboBox, markComboBox:
TDBLookupComboBox);
var
    markId, studentID, len, selectedDayNumber, i: Integer;
    dateIsCorrect: Boolean;
    student, mark: String;
    date: TDateTime;

```

```

begin
    date := selectDate.Date;
    student := studentComboBox.KeyValue;
    mark := markComboBox.KeyValue;
    if (journalActionController.CheckOnError(action, student, mark, date, errorLabel) = true) then
        Exit;

    studentId := getStudentId(student);
    markId := getMarkId(mark);
    if ((action = 'add') AND (checkRecordInDataBase(studentID, subjectId, date) <> 0)) then
        action := 'update';

    case StrUtils.IndexStr(action, ['add', 'update', 'delete']) of
        0: addRecordInJournal(studentID, subjectId, markID, date);
        1: updateRecordInJournal(studentID, subjectId, markID, date);
        2: deleteRecordFromJournal(studentID, subjectId, date);
    end;
    showMainTable(groupId, subjectId);
    studentComboBox.KeyValue := '';
    markComboBox.KeyValue := '';
end;
procedure TMain.buttonAddClick(Sender: TObject); begin journalActionController('add', subjectId, groupId,
selectStudent, selectMark) end;
procedure TMain.buttonChangeClick(Sender: TObject); begin journalActionController('update', subjectId, groupId,
selectStudent, selectMark) end;
procedure TMain.buttonDeleteClick(Sender: TObject); begin journalActionController('delete', subjectId, groupId,
selectStudent, selectMark) end;
procedure TMain.DateTimePicker1OnChange(Sender: TObject); begin errorLabel.Visible := False; end;
procedure TMain.FormActivate(Sender: TObject);
begin
    errorLabel.Visible := False;
    ComboBox1.Clear;
    currentMounth := -1;
    Groups1.Visible := True;
    Panel1.Visible := False;
    if (Authorization.userType = 'teacher') then begin
        selectSubject.KeyValue := getSubjectNameFromTeacherId(Authorization.userID);
        selectSubject.Enabled := False;
        Groups1.Visible := False;
    end;
    if (Authorization.userType = 'student') then begin
        selectGroup.KeyValue := getGroupNameFromStudentId(Authorization.userID);
        selectGroup.Enabled := False;
        openPanel.Enabled := False;
        Groups1.Visible := False;
    end;
    fillDatesComboBox();
    updateGrid();
end;
procedure TMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Authorization.Show();
    selectSubject.Enabled := True;
    selectGroup.Enabled := True;
    openPanel.Enabled := True;
end;
procedure TMain.FormCreate(Sender: TObject);
begin
    NullStrictConvert := False;
    groupId := 1;
    subjectId := 1;
end;
procedure TMain.FormMouseWheel(Sender: TObject; Shift: TShiftState;

```



```

    WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
    If DBGrid1.Focused then If (WheelDelta < 0) then DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0) else
    DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
    If selectGroup.Focused then If (WheelDelta < 0) then selectGroup.Perform(WM_KEYDOWN, VK_DOWN, 0) else
    selectGroup.Perform(WM_KEYDOWN, VK_UP, 0);
    If selectSubject.Focused then If (WheelDelta < 0) then begin selectSubject.Perform(WM_KEYDOWN, VK_DOWN, 0)
    else selectSubject.Perform(WM_KEYDOWN, VK_UP, 0);
    If selectStudent.Focused then If (WheelDelta < 0) then selectStudent.Perform(WM_KEYDOWN, VK_DOWN, 0) else
    selectStudent.Perform(WM_KEYDOWN, VK_UP, 0);
    If selectMark.Focused then If (WheelDelta < 0) then selectMark.Perform(WM_KEYDOWN, VK_DOWN, 0) else
    selectMark.Perform(WM_KEYDOWN, VK_UP, 0);
    Handled := True;
end;
function TMain.getMounthNameFromId(id: Integer): String;
const months: Array [1..12] of String = ('Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь',
'Октябрь', 'Ноябрь', 'Декабрь');
begin
    if ((id < 1) OR (id > 12)) then begin
        ShowMessage('getMounthNameFromId ID error');
        Exit;
    end;
    getMounthNameFromId := months[id];
end;
function TMain.getMounthIdFromName(name: String): Integer;
const months: Array [1..12] of String = ('Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь', 'Июль', 'Август', 'Сентябрь',
'Октябрь', 'Ноябрь', 'Декабрь');
begin getMounthIdFromName := StrUtils.IndexStr(name, months) + 1; end;
procedure TMain.createDataSelectRequest();
var
    len, i: Integer;
    resultStr: String;
begin
    len := DaysInAMonth(CURRENT_YEAR, currentMounth);
    resultStr := '';
    for i := 1 to len do begin
        resultStr := resultStr + 'SELECT DateValue("'" + IntToStr(i) + '.' + IntToStr(currentMounth) + '.' +
IntToStr(CURRENT_YEAR) + "') AS firs FROM Teachers';
        if (i <> len) then begin
            resultStr := resultStr + ' UNION ';
        end;
    end;
    selectDates := resultStr;
end;
procedure TMain.fillDatesComboBox();
var
    i: Integer;
begin
    if (CURRENT_SEMESTER = 1) then begin
        currentMounth := 9;
        ComboBox1.Items.Add(getMounthNameFromId(currentMounth));
        ComboBox1.Items.Add(getMounthNameFromId(10));
        ComboBox1.Items.Add(getMounthNameFromId(11));
        ComboBox1.Items.Add(getMounthNameFromId(12));
        ComboBox1.Items.Add(getMounthNameFromId(1));
    end
    else
    if (CURRENT_SEMESTER = 2) then begin
        currentMounth := 2;
        ComboBox1.Items.Add(getMounthNameFromId(currentMounth));
        ComboBox1.Items.Add(getMounthNameFromId(3));
        ComboBox1.Items.Add(getMounthNameFromId(4));
        ComboBox1.Items.Add(getMounthNameFromId(5));
    end;
end;

```

```

    ComboBox1.Items.Add(getMounthNameFromId(6));
end;
updateDatesComboBox();
end;
procedure TMain.updateDatesComboBox();
begin
    ComboBox1.Text := getMounthNameFromId(currentMounth);
    createDataSelectRequest();
end;
procedure TMain.ComboBox1Click(Sender: TObject);
begin
    currentMounth := getMounthIdFromName(ComboBox1.Text);
    createDataSelectRequest();
    showMainTable(groupId, subjectId);
end;
end.
UNIT 3
unit Unit3;
var
    Subjects: TSubjects;
    recordId: Integer;
uses Unit2, StrUtils, Unit9;
function TSubjects.getSurname(fullName: String): String;
var position: Integer;
begin
    position := Pos(' ', fullName);
    getSurname := Copy(fullName, 0, position);
end;
procedure TSubjects.showSubjectsTable();
begin
    DataModule1.ADOQuerySubjectsShow.Close;
    DataModule1.ADOQuerySubjectsShow.SQL.Text :=
        'SELECT s.id, s.name, s.audience, (t.surname & " " & t.name) AS teacher ' +
        'FROM Subjects AS s ' +
        'INNER JOIN Teachers AS t ON (t.ID = s.teacher_id) ' +
        'ORDER BY s.id';
    DataModule1.ADOQuerySubjectsShow.Open;
end;
procedure TSubjects.DBGrid1CellClick(Column: TColumn);
var
    colName: String;
    fmt: TFormatSettings;
begin
    recordId := DBGrid1.Fields[0].AsInteger;
    nameEdit.Text := DBGrid1.Fields[1].AsString;
    audienceEdit.Text := DBGrid1.Fields[2].AsString;
    teacherComboBox.KeyValue := getSurname(DBGrid1.Fields[3].AsString);
end;
function TSubjects.getTeacherId(surname: String): Integer;
begin
    DataModule1.ADOQuerySubjects.Close;
    DataModule1.ADOQuerySubjects.SQL.Text :=
        'SELECT id ' +
        'FROM Teachers ' +
        'WHERE surname = "' + surname + '"';
    DataModule1.ADOQuerySubjects.Open;
    getTeacherId := DataModule1.DataSourceSubjects.DataSet.Fields[0].AsInteger;
end;
function TSubjects.setRecordId(name: String; audienceNumber, teacherId: Integer): Integer;
begin
    DataModule1.ADOQuerySubjects.Close;
    DataModule1.ADOQuerySubjects.SQL.Text :=
        'SELECT id ' +

```

```

'FROM Subjects' +
'WHERE (' +
'  name = ' + name + ', ' +
'  audience = ' + IntToStr(audienceNumber) + ', ' +
'  teacher_id = ' + IntToStr(teacherId) + ');
DataModule1.ADOQuerySubjects.Open;
setRecordId := DataModule1.DataSourceSubjects.DataSet.Fields[0].AsInteger;
end;
procedure TSubjects.addRecordInSubjects(name, audience: String; teacherId: Integer);
begin
  DataModule1.ADOQuerySubjects.Close;
  DataModule1.ADOQuerySubjects.SQL.Text :=
'INSERT INTO Subjects (name, audience, teacher_id) ' +
'VALUES (' + name + ', ' +
'  ' + audience + ', ' +
'  ' + IntToStr(teacherId) +
'  )';
  DataModule1.ADOQuerySubjects.ExecSQL;
end;
procedure TSubjects.updateRecordInSubjects(name, audience: String; teacherId, recordId: Integer);
begin
  DataModule1.ADOQuerySubjects.Close;
  DataModule1.ADOQuerySubjects.SQL.Text :=
'UPDATE Subjects ' +
'SET ' +
'  name = ' + name + ', ' +
'  audience = ' + audience + ', ' +
'  teacher_id = ' + IntToStr(teacherId) + ' ' +
'WHERE ' +
'  id = ' + IntToStr(recordId) + ';';
  DataModule1.ADOQuerySubjects.ExecSQL;
end;
procedure TSubjects.deleteRecordFromSubjects(recordId: Integer);
begin
  DataModule1.ADOQuerySubjects.Close;
  DataModule1.ADOQuerySubjects.SQL.Text :=
'DELETE ' +
'FROM Subjects ' +
'WHERE ' +
'  id = ' + IntToStr(recordId);
  DataModule1.ADOQuerySubjects.ExecSQL;
end;
function TSubjects.subjectsActionControllerCheckOnError(action, name, audience, teacher: String; errorString: TLabel):
Boolean;
var
  selectedDayNumber: Integer;
begin
  if (name = '') then begin
    errorString.Visible := True;
    errorString.Caption := 'Укажите название!';
    subjectsActionControllerCheckOnError := True;
    Exit;
  end;
  if (audience = '') then begin
    errorString.Visible := True;
    errorString.Caption := 'Укажите номер аудитории!';
    subjectsActionControllerCheckOnError := True;
    Exit;
  end;
  if (teacher = '') then begin
    errorString.Visible := True;
    errorString.Caption := 'Укажите учителя!';
    subjectsActionControllerCheckOnError := True;
  end;
end;

```

```

    Exit;
end;
subjectsActionControllerCheckOnError := False;
end;
procedure TSubjects.subjectsActionController(action: String; nameEdit, audienceEdit: TEdit;
    teacherComboBox: TDBLookupComboBox; errorString: TLabel; recordId: Integer);
var
    name, audience, teacher: String;
    teacherId: Integer;
begin
    name := nameEdit.Text;
    audience := audienceEdit.Text;
    teacher := teacherComboBox.KeyValue;
    if (subjectsActionControllerCheckOnError(action, name, audience, teacher, errorString) = true) then begin
        Exit;
    end;
    teacherId := getTeacherId(teacher);
    case StrUtils.IndexStr(action, ['add', 'update', 'delete']) of
        0: addRecordInSubjects(name, audience, teacherId);
        1: updateRecordInSubjects(name, audience, teacherId, recordId);
        2: deleteRecordFromSubjects(recordId);
    end;
    showSubjectsTable();
    clearClick(clear);
end;
procedure TSubjects.clearClick(Sender: TObject);
begin
    recordId := -1;
    nameEdit.Text := '';
    audienceEdit.Text := '';
    teacherComboBox.KeyValue := '';
end;
procedure TSubjects.buttonAddClick(Sender: TObject); begin subjectsActionController('add', nameEdit, audienceEdit,
teacherComboBox, errorLabel, recordId); end;
procedure TSubjects.buttonChangeClick(Sender: TObject); begin subjectsActionController('update', nameEdit,
audienceEdit, teacherComboBox, errorLabel, recordId); end;
procedure TSubjects.buttonDeleteClick(Sender: TObject); begin subjectsActionController('delete', nameEdit, audienceEdit,
teacherComboBox, errorLabel, recordId); end;
procedure TSubjects.openPanelClick(Sender: TObject);
begin Panel1.Visible := NOT Panel1.Visible; end;
procedure TSubjects.FormActivate(Sender: TObject);
begin
    openPanel.Enabled := True;
    Panel1.Visible := False;
    if ((Authorization.userType = 'student') OR (Authorization.userType = 'teacher')) then begin
        openPanel.Enabled := false;
    end;
    recordId := DBGrid1.Fields[0].AsInteger;
    nameEdit.Text := DBGrid1.Fields[1].AsString;
    audienceEdit.Text := DBGrid1.Fields[2].AsString;
    teacherComboBox.KeyValue := getSurname(DBGrid1.Fields[3].AsString);
    showSubjectsTable();
end;
procedure TSubjects.FormCreate(Sender: TObject); begin NullStrictConvert := False; end;
procedure TSubjects.FormMouseWheel(Sender: TObject; Shift: TShiftState;
    WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
    If DBGrid1.Focused then begin
        If (WheelDelta < 0) then begin
            DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0)
        end
        else begin
            DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
        end
    end
end;

```

```

    end;
end;
If teacherComboBox.Focused then begin
    If (WheelDelta < 0) then begin
        teacherComboBox.Perform(WM_KEYDOWN, VK_DOWN, 0)
    end
    else begin
        teacherComboBox.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
end;
Handled := True;
end;
end.
UNIT 4
unit Unit4;
var
    Timetable: TTimetable;
    groupName, subjectName, currentType: String;
USES Unit2, StrUtils, Unit8, Unit9;
function TTimetable.getGroupId(groupName: String): Integer;
begin
    DataModule1.ADOQueryTimetable.Close;
    DataModule1.ADOQueryTimetable.SQL.Text :=
        'SELECT * ' +
        'FROM Groups ' +
        'WHERE name LIKE "' + groupName + '"';
    DataModule1.ADOQueryTimetable.Open;
    getGroupId := DataModule1.DataSourceTimetable.DataSet.Fields[0].AsInteger;
end;
function TTimetable.getSubjectId(subjectName: String): Integer;
begin
    DataModule1.ADOQueryTimetable.Close;
    DataModule1.ADOQueryTimetable.SQL.Text :=
        'SELECT * ' +
        'FROM Subjects ' +
        'WHERE name LIKE "' + subjectName + '"';
    DataModule1.ADOQueryTimetable.Open;
    getSubjectId := DataModule1.DataSourceTimetable.DataSet.Fields[0].AsInteger;
end;
procedure TTimetable.showTable(commitType: String);
var
    group, subject: String;
begin
    group := 'group_id';
    subject := 'subject_id';
    case StrUtils.IndexStr(commitType, ['all', 'group', 'subject']) of
        1: group := IntToStr(getGroupId(groupName));
        2: subject := IntToStr(getSubjectId(subjectName));
    end;
    DataModule1.ADOQueryTimetableShow.Close;
    DataModule1.ADOQueryTimetableShow.SQL.Text :=
        'SELECT ' +
        'WeekdayName(t.day_of_week) AS [day of week], ' +
        't.pair AS [pair], ' +
        'g.name AS [group], ' +
        's.audience AS [audience], ' +
        's.name AS [subject] ' +
        'FROM ' +
        'Timetable AS t, ' +
        'Subjects AS s, ' +
        'Groups AS g ' +
        'WHERE ' +
        '(' +

```

```

        '(t.subject_id = s.id) AND ' +
        '(g.id = t.group_id) AND ' +
        '(t.subject_id = ' + subject + ') AND ' +
        '(t.group_id = ' + group + ') ' +
    ') ' +
    'ORDER BY ' +
    't.day_of_week, ' +
    'g.name, ' +
    't.pair;';
    DataModule1.ADOQueryTimetableShow.Open;
end;
procedure TTimetable.Button1Click(Sender: TObject);
begin
    showTable('all');
    currentType := 'all';
end;
procedure TTimetable.Button2Click(Sender: TObject);
begin
    showTable('group');
    currentType := 'group';
end;
procedure TTimetable.Button3Click(Sender: TObject);
begin
    showTable('subject');
    currentType := 'subject';
end;
procedure TTimetable.Button4Click(Sender: TObject);
begin
    TimeTableReport.QuickRep1.PreviewModal();
end;
function TTimetable.getSubjectNameFromTeacherId(teacherId: Integer): String;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
    'SELECT name ' +
    'FROM Subjects ' +
    'WHERE teacher_id = ' + IntToStr(teacherId) + ';';
    DataModule1.ADOQueryMain.Open;
    getSubjectNameFromTeacherId := DataModule1.DataSourceMain.DataSet.Fields[0].AsString;
end;
function TTimetable.getGroupNameFromStudentId(studentId: Integer): String;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
    'SELECT g.name ' +
    'FROM Users AS u ' +
    'INNER JOIN Groups AS g ON g.id = u.group_id ' +
    'WHERE u.id = ' + IntToStr(studentId) + ';';
    DataModule1.ADOQueryMain.Open;
    getGroupNameFromStudentId := DataModule1.DataSourceMain.DataSet.Fields[0].AsString;
end;
procedure TTimetable.FormActivate(Sender: TObject);
begin
    Button1.Visible := True;
    Button2.Visible := True;
    Button3.Visible := True;
    currentType := 'all';
    selectGroup.KeyValue := DataModule1.ADOTableGroups.FieldByName('name').AsString;
    selectSubject.KeyValue := DataModule1.ADOQuerySubjectsShow.FieldByName('name').AsString;
    groupName := selectGroup.KeyValue;
    subjectName := selectSubject.KeyValue;
    if (Authorization.userType = 'teacher') then begin
        selectSubject.KeyValue := getSubjectNameFromTeacherId(Authorization.userID);
    end;
end;

```

```

    subjectName := selectSubject.KeyValue;
    selectSubject.Enabled := False;
    Button1.Visible := False;
    Button2.Visible := False;
    Button3.Visible := False;
    currentType := 'subject';
    showTable('subject');
end;
if (Authorization.userType = 'student') then begin
    selectGroup.KeyValue := getGroupNameFromStudentId(Authorization.userID);
    groupName := selectGroup.KeyValue;
    selectGroup.Enabled := False;
    Button1.Visible := False;
    Button2.Visible := False;
    Button3.Visible := False;
    currentType := 'group';
    showTable('group');
end;
end;
procedure TTimetable.FormMouseWheel(Sender: TObject; Shift: TShiftState;
    WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
    If DBGrid1.Focused then begin
        If (WheelDelta < 0) then
            DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0)
        else
            DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
    If selectGroup.Focused then begin
        If (WheelDelta < 0) then
            selectGroup.Perform(WM_KEYDOWN, VK_DOWN, 0)
        else
            selectGroup.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
    If selectSubject.Focused then begin
        If (WheelDelta < 0) then
            selectSubject.Perform(WM_KEYDOWN, VK_DOWN, 0)
        else
            selectSubject.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
    Handled := True;
end;
procedure TTimetable.selectGroupClick(Sender: TObject); begin groupName := selectGroup.KeyValue;
showTable(currentType); end;
procedure TTimetable.selectSubjectClick(Sender: TObject); begin subjectName := selectSubject.KeyValue;
showTable(currentType); end;
end.
UNIT 5
unit Unit5;
var
    Groups: TGroups;
    groupId: Integer;
uses Unit2, StrUtils, Unit9;
procedure TGroups.showGroupsTable();
begin
    DataModule1.ADOQueryGroupsShow.Close;
    DataModule1.ADOQueryGroupsShow.SQL.Text :=
        'SELECT *' +
        'FROM Groups';
    DataModule1.ADOQueryGroupsShow.Open;
end;
procedure TGroups.DBGrid1CellClick(Column: TColumn);
begin

```

```

    groupId := DBGrid1.Fields[0].AsInteger;
    nameEdit.Text := DBGrid1.Fields[1].AsString;
end;
procedure TGroups.addRecordInGroups(name: String);
begin
    DataModule1.ADOQueryGroups.Close;
    DataModule1.ADOQueryGroups.SQL.Text :=
    'INSERT INTO Groups (name) ' +
    'VALUES (' + name + ')';
    DataModule1.ADOQueryGroups.ExecSQL;
end;
procedure TGroups.updateRecordInGroups(groupId: Integer; name: String);
begin
    DataModule1.ADOQueryGroups.Close;
    DataModule1.ADOQueryGroups.SQL.Text :=
    'UPDATE Groups ' +
    'SET ' +
    'name = ' + name + ' ' +
    'WHERE ' +
    'id = ' + IntToStr(groupId) + ';';
    DataModule1.ADOQueryGroups.ExecSQL;
end;
procedure TGroups.deleteRecordFromGroups(groupId: Integer);
begin
    DataModule1.ADOQueryGroups.Close;
    DataModule1.ADOQueryGroups.SQL.Text :=
    'DELETE ' +
    'FROM Groups ' +
    'WHERE ' +
    'id = ' + IntToStr(groupId);
    DataModule1.ADOQueryGroups.ExecSQL;
end;
function TGroups.groupsActionControllerCheckOnError(action, name: String; errorString: TLabel): Boolean;
var
    selectedDayNumber: Integer;
begin
    if ((name = '') AND NOT (action = 'delete')) then begin
        errorString.Visible := True;
        errorString.Caption := 'Неправильное название!';
        groupsActionControllerCheckOnError := True;
        Exit;
    end;
    groupsActionControllerCheckOnError := False;
end;
procedure TGroups.groupsActionController(action: String; nameEdit: TEdit; groupId: Integer; errorString: TLabel);
var
    name: String;
begin
    name := nameEdit.Text;
    if (groupsActionControllerCheckOnError(action, name, errorString) = true) then begin
        Exit;
    end;
    case StrUtils.IndexStr(action, ['add', 'update', 'delete']) of
        0: addRecordInGroups(name);
        1: updateRecordInGroups(groupId, name);
        2: deleteRecordFromGroups(groupId);
    end;
    showGroupsTable();
    nameEdit.Text := '';
end;
procedure TGroups.buttonAddClick(Sender: TObject); begin groupsActionController('add', nameEdit, groupId,
errorLabel); end;

```



```

procedure TGroups.buttonChangeClick(Sender: TObject); begin groupsActionController('update', nameEdit, groupId,
errorLabel); end;
procedure TGroups.buttonDeleteClick(Sender: TObject); begin groupsActionController('delete', nameEdit, groupId,
errorLabel); end;
procedure TGroups.openPanelClick(Sender: TObject); begin Panel1.Visible := NOT Panel1.Visible; end;
procedure TGroups.FormActivate(Sender: TObject);
begin
    Panel1.Visible := False;
    openPanel.Enabled := True;
    if ((Authorization.userType = 'teacher') OR (Authorization.userType = 'student')) then begin
        openPanel.Enabled := False;
    end;
    groupId := DBGrid1.Fields[0].AsInteger;
    nameEdit.Text := DBGrid1.Fields[1].AsString;
end;
procedure TGroups.FormCreate(Sender: TObject);
begin
    NullStrictConvert := False;
end;
procedure TGroups.FormMouseWheel(Sender: TObject; Shift: TShiftState;
    WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
    If DBGrid1.Focused then begin
        If (WheelDelta < 0) then begin
            DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0)
        end
        else begin
            DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
        end;
    end;
end;
end.
UNIT 6
unit Unit6;
var
    Users: TUsers;
    recordId: Integer;
    loginAndPassword: Boolean;
Uses Unit2, StrUtils, Unit9;
procedure TUsers.Button1Click(Sender: TObject);
begin
    resetPanelFields();
end;
procedure TUsers.CheckBox1Click(Sender: TObject);
begin
    loginAndPassword := CheckBox1.Checked;
    DBGrid1.Columns[6].Visible := loginAndPassword;
    DBGrid1.Columns[7].Visible := loginAndPassword;
    usernameEdit.Visible := loginAndPassword;
    passwordEdit.Visible := loginAndPassword;
    Label7.Visible := loginAndPassword;
    Label8.Visible := loginAndPassword;
end;
procedure TUsers.DBGrid1CellClick(Column: TColumn);
begin
    updatePanelFields();
end;
function TUsers.getGroupNameFromStudentId(studentId: Integer): String;
begin
    DataModule1.ADOQueryMain.Close;
    DataModule1.ADOQueryMain.SQL.Text :=
'SELECT g.name ' +
'FROM Users AS u ' +

```

```

'INNER JOIN Groups AS g ON g.id = u.group_id ' +
'WHERE u.id = ' + IntToStr(studentId) + ';';
DataModule1.ADOQueryMain.Open;
getGroupNameFromStudentId := DataModule1.DataSourceMain.DataSet.Fields[0].AsString;
end;
procedure TUsers.FormActivate(Sender: TObject);
begin
  CheckBox1.Visible := True;
  openPanel.Visible := True;
  selectGroup.Enabled := True;
  Panel1.Visible := False;
  if (Authorization.userType = 'student') then begin
    CheckBox1.Visible := False;
    openPanel.Visible := False;
    selectGroup.KeyValue := getGroupNameFromStudentId(Authorization.userID);
    selectGroup.Enabled := False;
  end;
  if (Authorization.userType = 'teacher') then begin CheckBox1.Visible := False; openPanel.Visible := False; end;
  updateGrid();
  CheckBox1Click(CheckBox1);
end;
procedure TUsers.FormMouseWheel(Sender: TObject; Shift: TShiftState;
  WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
  If DBGrid1.Focused then begin
    If (WheelDelta < 0) then begin
      DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0)
    end
    else begin
      DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
  end;
  If selectGroup.Focused then begin
    If (WheelDelta < 0) then begin
      selectGroup.Perform(WM_KEYDOWN, VK_DOWN, 0)
    end
    else begin
      selectGroup.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
  end;
  If selectGroupPanel.Focused then begin
    If (WheelDelta < 0) then begin
      selectGroupPanel.Perform(WM_KEYDOWN, VK_DOWN, 0)
    end
    else begin
      selectGroupPanel.Perform(WM_KEYDOWN, VK_UP, 0);
    end;
  end;
  Handled := True;
end;
function TUsers.getGroupId(groupName: String): Integer;
begin
  DataModule1.ADOQueryUsers.Close;
  DataModule1.ADOQueryUsers.SQL.Text :=
'SELECT * ' +
'FROM Groups ' +
'WHERE name LIKE ''' + groupName + ''';
  DataModule1.ADOQueryUsers.Open;
  getGroupId := DataModule1.DataSourceUsers.DataSet.Fields[0].AsInteger;
end;
procedure TUsers.openPanelClick(Sender: TObject);
begin
  Panel1.Visible := NOT Panel1.Visible;

```

```

    updatePanelFields();
end;
procedure TUsers.updatePanelFields();
begin
    selectId.KeyValue := DBGrid1.Fields[0].AsString;
    recordId := selectId.KeyValue;
    selectGroupPanel.KeyValue := DBGrid1.Fields[1].AsString;
    surnameEdit.Text := DBGrid1.Fields[2].AsString;
    nameEdit.Text := DBGrid1.Fields[3].AsString;
    patronymicEdit.Text := DBGrid1.Fields[4].AsString;
    emailEdit.Text := DBGrid1.Fields[5].AsString;
    usernameEdit.Text := DBGrid1.Fields[6].AsString;
    passwordEdit.Text := DBGrid1.Fields[7].AsString;
end;
procedure TUsers.resetPanelFields();
begin
    selectId.KeyValue := "";
    recordId := 0;

    selectGroupPanel.KeyValue := "";
    surnameEdit.Text := "";
    nameEdit.Text := "";
    patronymicEdit.Text := "";
    emailEdit.Text := "";
    usernameEdit.Text := "";
    passwordEdit.Text := "";
end;
procedure TUsers.showMainTable(groupId: Integer);
begin
    DataModule1.ADOQueryUsersShow.Close;
    DataModule1.ADOQueryUsersShow.SQL.Text :=
        'SELECT u.id, g.name AS GN, surname, u.name, patronymic, email, username, password ' +
        'FROM Users AS u ' +
        'INNER JOIN Groups AS g ON ((g.id = u.group_id) AND (g.id = ' + IntToStr(groupId) + ')) ' +
        'ORDER BY u.id';
    DataModule1.ADOQueryUsersShow.Open;
end;
procedure TUsers.selectGroupClick(Sender: TObject);
begin
    updateGrid();
    resetPanelFields();
end;
procedure TUsers.updateGrid();
var
    group: String;
    groupId: Integer;
begin
    group := selectGroup.KeyValue;
    if NOT (group = "") then begin
        groupId := getGroupId(group)
    end
    else
        begin
            selectGroup.KeyValue := DataModule1.ADOQueryGroupsShow.FieldByName('name').AsString;
            groupId := getGroupId(selectGroup.KeyValue)
        end;
    showMainTable(groupId);
end;
procedure TUsers.showErrorLabel(text: String);
begin
    errorLabel.Visible := True;
    errorLabel.Caption := text;
end;

```

```

function TUsers.usersActionControllerCheckOnError(action: String; id, groupId: Integer; surname, name, patronymic,
email, username, password: String): Boolean;
begin
  if ((String(id) = "") AND (action <> 'add')) then begin showErrorLabel('Ошибка ИД');
usersActionControllerCheckOnError := True; Exit; end;
  if action = 'delete' then begin usersActionControllerCheckOnError := False; Exit; end;
  if String(groupId) = "" then begin showErrorLabel('Ошибка ИД группы'); usersActionControllerCheckOnError := True;
Exit; end;
  if surname = "" then begin showErrorLabel('Ошибка фамилии'); usersActionControllerCheckOnError := True; Exit; end;
  if name = "" then begin showErrorLabel('Ошибка имени'); usersActionControllerCheckOnError := True; Exit; end;
  if patronymic = "" then begin showErrorLabel('Ошибка отчества'); usersActionControllerCheckOnError := True; Exit;
end;
  if email = "" then begin showErrorLabel('Ошибка email'); usersActionControllerCheckOnError := True; Exit; end;
  if (username = "") AND (loginAndPassword = true) then begin showErrorLabel('Ошибка username');
usersActionControllerCheckOnError := True; Exit; end;
  if (password = "") AND (loginAndPassword = true) then begin showErrorLabel('Ошибка password');
usersActionControllerCheckOnError := True; Exit; end;
  usersActionControllerCheckOnError := False;
end;

procedure TUsers.addRecordInUsers(groupId: Integer; surname, name, patronymic, email, username, password: String);
begin
  DataModule1.ADOQueryUsers.Close;
  DataModule1.ADOQueryUsers.SQL.Text :=
'INSERT INTO Users (surname, name, patronymic, email, username, [password], group_id) ' +
'VALUES (' +
  "" + surname + "", ' +
  "" + name + "", ' +
  "" + patronymic + "", ' +
  "" + email + "", ' +
  "" + username + "", ' +
  "" + password + "", ' +
  IntToStr(groupId)+
  ');';
  DataModule1.ADOQueryUsers.ExecSQL;
end;

procedure TUsers.updateRecordInUsers(id, groupId: Integer; surname, name, patronymic, email, username, password:
String);
begin
  DataModule1.ADOQueryUsers.Close;
  DataModule1.ADOQueryUsers.SQL.Text :=
'UPDATE Users ' +
'SET ' +
'surname = ' + surname + ", ' +
'[name] = ' + name + ", ' +
'patronymic= ' + patronymic + ", ' +
'email = ' + email + ", ' +
'username = ' + username + ", ' +
'[password] = ' + password + ", ' +
'group_id = ' + IntToStr(groupId) + ' ' +
'WHERE ' +
'id = ' + IntToStr(id) + ' ';
  DataModule1.ADOQueryUsers.ExecSQL;
end;

procedure TUsers.deleteRecordFromUsers(id: Integer);
begin
  DataModule1.ADOQueryUsers.Close;
  DataModule1.ADOQueryUsers.SQL.Text :=
'DELETE ' +
'FROM Users ' +
'WHERE id = ' + IntToStr(id) + ' ';
  DataModule1.ADOQueryUsers.ExecSQL;
end;

```

```

procedure TUsers.usersActionController(action: String; id: Integer; group, surname, name, patronymic, email, username,
password: String);
var
  groupId: Integer;
begin
  groupId := getGroupId(group);
  if (usersActionControllerCheckOnError(action, id, groupId, surname, name, patronymic, email, username, password) =
true) then begin
    Exit;
  end;
  case StrUtils.IndexStr(action, ['add', 'update', 'delete']) of
    0: addRecordInUsers(groupId, surname, name, patronymic, email, username, password);
    1: updateRecordInUsers(id, groupId, surname, name, patronymic, email, username, password);
    2: deleteRecordFromUsers(id);
  end;
  showMainTable(groupId);
end;
procedure TUsers.buttonAddClick(Sender: TObject);
begin
  usersActionController('add', recordId, selectGroupPanel.KeyValue,
  surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
  emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
procedure TUsers.buttonChangeClick(Sender: TObject);
begin
  usersActionController('update', recordId, selectGroupPanel.KeyValue,
  surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
  emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
procedure TUsers.buttonDeleteClick(Sender: TObject);
begin
  usersActionController('delete', recordId, selectGroupPanel.KeyValue,
  surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
  emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
end.
UNIT 7
unit Unit7;
var
  Teachers: TTeachers;
  recordId: Integer;
  loginAndPassword: Boolean;
uses Unit2, StrUtils, Unit9;
procedure TTeachers.CheckBox1Click(Sender: TObject);
begin
  loginAndPassword := CheckBox1.Checked;
  DBGrid1.Columns[5].Visible := loginAndPassword;
  DBGrid1.Columns[6].Visible := loginAndPassword;
  usernameEdit.Visible := loginAndPassword;
  passwordEdit.Visible := loginAndPassword;
  Label7.Visible := loginAndPassword;
  Label8.Visible := loginAndPassword;
end;
procedure TTeachers.FormActivate(Sender: TObject);
begin
  CheckBox1.Visible := True;
  openPanel.Visible := True;
  Panel1.Visible := False;
  if (Authorization.userType = 'student') then begin
    CheckBox1.Visible := False;
    openPanel.Visible := False;
  end;
  if (Authorization.userType = 'teacher') then begin

```

```

    CheckBox1.Visible := False;
    openPanel.Visible := False;
end;
CheckBox1Click(CheckBox1);
end;
procedure TTeachers.FormMouseWheel(Sender: TObject; Shift: TShiftState;
    WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
    If DBGrid1.Focused then begin
        If (WheelDelta < 0) then begin
            DBGrid1.Perform(WM_KEYDOWN, VK_DOWN, 0)
        end
        else begin
            DBGrid1.Perform(WM_KEYDOWN, VK_UP, 0);
        end;
    end;
end;
procedure TTeachers.openPanelClick(Sender: TObject);
begin
    Panel1.Visible := NOT Panel1.Visible;
    updatePanelFields();
end;
procedure TTeachers.updatePanelFields();
begin
    selectId.KeyValue := DBGrid1.Fields[0].AsString;
    recordId := selectId.KeyValue;
    surnameEdit.Text := DBGrid1.Fields[1].AsString;
    nameEdit.Text := DBGrid1.Fields[2].AsString;
    patronymicEdit.Text := DBGrid1.Fields[3].AsString;
    emailEdit.Text := DBGrid1.Fields[4].AsString;
    usernameEdit.Text := DBGrid1.Fields[5].AsString;
    passwordEdit.Text := DBGrid1.Fields[6].AsString;
end;
procedure TTeachers.resetPanelFields();
begin
    selectId.KeyValue := '';
    recordId := 0;
    surnameEdit.Text := '';
    nameEdit.Text := '';
    patronymicEdit.Text := '';
    emailEdit.Text := '';
    usernameEdit.Text := '';
    passwordEdit.Text := '';
end;
procedure TTeachers.showMainTable();
begin
    DataModule1.ADOQueryTeachersShow.Close;
    DataModule1.ADOQueryTeachersShow.SQL.Text :=
        'SELECT * ' +
        'FROM Teachers ' +
        'ORDER BY ID';
    DataModule1.ADOQueryTeachersShow.Open;
end;
procedure TTeachers.Button1Click(Sender: TObject);
begin
    resetPanelFields();
end;
procedure TTeachers.DBGrid1CellClick(Column: TColumn);
begin
    updatePanelFields();
end;
procedure TTeachers.selectGroupClick(Sender: TObject);
begin

```

```

    resetPanelFields();
end;
procedure TTeachers.showErrorLabel(text: String);
begin
    errorLabel.Visible := True;
    errorLabel.Caption := text;
end;
function TTeachers.teachersActionControllerCheckOnError(action: String; id: Integer; surname, name, patronymic, email,
username, password: String): Boolean;
begin
    if ((String(id) = '') AND (action <> 'add')) then begin
        showErrorLabel('Ошибка ИД');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if action = 'delete' then begin
        teachersActionControllerCheckOnError := False;
        Exit;
    end;
    if surname = '' then begin
        showErrorLabel('Ошибка фамилии');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if name = '' then begin
        showErrorLabel('Ошибка имени');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if patronymic = '' then begin
        showErrorLabel('Ошибка отчества');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if email = '' then begin
        showErrorLabel('Ошибка email');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if (username = '') AND (loginAndPassword = true) then begin
        showErrorLabel('Ошибка username');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    if (password = '') AND (loginAndPassword = true) then begin
        showErrorLabel('Ошибка password');
        teachersActionControllerCheckOnError := True;
        Exit;
    end;
    teachersActionControllerCheckOnError := False;
end;
procedure TTeachers.addRecordInTeachers(surname, name, patronymic, email, username, password: String);
begin
    DataModule1.ADOQueryTeachers.Close;
    DataModule1.ADOQueryTeachers.SQL.Text :=
    'INSERT INTO Teachers (surname, name, patronymic, email, username, [password]) ' +
    'VALUES (' +
        ''' + surname + ''', ' +
        ''' + name + ''', ' +
        ''' + patronymic + ''', ' +
        ''' + email + ''', ' +
        ''' + username + ''', ' +
        ''' + password + ''' ' +

```

```

');';
DataModule1.ADOQueryTeachers.ExecSQL;
end;
procedure TTeachers.updateRecordInTeachers(id: Integer; surname, name, patronymic, email, username, password:
String);
begin
DataModule1.ADOQueryTeachers.Close;
DataModule1.ADOQueryTeachers.SQL.Text :=
'UPDATE Teachers ' +
'SET ' +
'surname = ' + surname + ', ' +
'name = ' + name + ', ' +
'patronymic = ' + patronymic + ', ' +
'email = ' + email + ', ' +
'username = ' + username + ', ' +
'[password] = ' + password + ' ' +
'WHERE ' +
'id = ' + IntToStr(id) + ';';
DataModule1.ADOQueryTeachers.ExecSQL;
end;
procedure TTeachers.deleteRecordFromTeachers(id: Integer);
begin
DataModule1.ADOQueryTeachers.Close;
DataModule1.ADOQueryTeachers.SQL.Text :=
'DELETE ' +
'FROM Teachers ' +
'WHERE id = ' + IntToStr(id) + ';';
DataModule1.ADOQueryTeachers.ExecSQL;
end;
procedure TTeachers.teachersActionController(action: String; id: Integer; surname, name, patronymic, email, username,
password: String);
begin
if (teachersActionControllerCheckOnError(action, id, surname, name, patronymic, email, username, password) = true)
then begin
Exit;
end;
case StrUtils.IndexStr(action, ['add', 'update', 'delete']) of
0: addRecordInTeachers(surname, name, patronymic, email, username, password);
1: updateRecordInTeachers(id, surname, name, patronymic, email, username, password);
2: deleteRecordFromTeachers(id);
end;
showMainTable();
end;
procedure TTeachers.buttonAddClick(Sender: TObject);
begin
teachersActionController('add', recordId,
surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
procedure TTeachers.buttonChangeClick(Sender: TObject);
begin
teachersActionController('update', recordId,
surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
procedure TTeachers.buttonDeleteClick(Sender: TObject);
begin
teachersActionController('delete', recordId,
surnameEdit.Text, nameEdit.Text, patronymicEdit.Text,
emailEdit.Text, usernameEdit.Text, passwordEdit.Text);
end;
end.
UNIT 8

```



```

unit Unit8;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, QuickRpt, QRCtrl;
type
  TTimeTableReport = class(TForm)
    QuickRep1: TQuickRep;
    TitleBand1: TQRBand;
    ColumnHeaderBand1: TQRBand;
    DetailBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    QRLabel4: TQRLabel;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    QRDBText4: TQRDBText;
    QRLabel5: TQRLabel;
    QRDBText3: TQRDBText;
    QRDBText5: TQRDBText;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  TimeTableReport: TTimeTableReport;
implementation
{$R *.dfm}
uses Unit4, Unit2;
end.

UNIT 9
unit Unit9;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Data.DB, Vcl.Grids,
  Vcl.DBGrids;
type
  TAuthorization = class(TForm)
    admin: TButton;
    student: TButton;
    teacher: TButton;
    errorLabel: TLabel;
    procedure authorizationController(action: String);
    procedure studentClick(Sender: TObject);
    procedure teacherClick(Sender: TObject);
    procedure adminClick(Sender: TObject);
    function authorizationControllerCheckOnError(action, login, password: String): Boolean;
    procedure showErrorLabel(text: String);
    function teacherAuthentication(login, password: String): Integer;
    procedure showMainForm();
    function studentAuthentication(login, password: String): Integer;
    procedure hideErrorLabel();
  private
    { Private declarations }
  public
    userType: String;
    userID: Integer;
  end;
const
  ADMIN_LOGIN = 'admin';

```

```

ADMIN_PASSWORD = 'admin';
var
  Authorization: TAuthorization;
implementation
{$R *.dfm}
uses Unit1, Unit2, StrUtils;
procedure TAuthorization.showErrorLabel(text: String);
begin
  errorLabel.Visible := True;
  errorLabel.Caption := text;
end;
procedure TAuthorization.hideErrorLabel();
begin
  errorLabel.Visible := False;
  errorLabel.Caption := '';
end;
function TAuthorization.authorizationControllerCheckOnError(action, login, password: String): Boolean;
begin
  if (login = '') then begin
    showErrorLabel('Пустой логин');
    authorizationControllerCheckOnError := true;
    Exit;
  end;
  if (password = '') then begin
    showErrorLabel('Пустой пароль');
    authorizationControllerCheckOnError := true;
    Exit;
  end;

  authorizationControllerCheckOnError := false;
end;
function TAuthorization.teacherAuthentication(login, password: String): Integer;
var value: String;
begin
  DataModule1.ADOQueryAuthorization.Close;
  DataModule1.ADOQueryAuthorization.SQL.Text :=
'SELECT id ' +
'FROM Teachers ' +
'WHERE (username = ''' + login + ''') AND ' +
'(password = ''' + password + ''')';
  DataModule1.ADOQueryAuthorization.Open;
  value := DataModule1.DataSourceAuthorization.DataSet.Fields[0].AsString;
  if (value = '') then begin teacherAuthentication := -1; Exit; end;
  teacherAuthentication := StrToInt(value);
end;
function TAuthorization.studentAuthentication(login, password: String): Integer;
var value: String;
begin
  DataModule1.ADOQueryAuthorization.Close;
  DataModule1.ADOQueryAuthorization.SQL.Text :=
'SELECT id ' +
'FROM Users ' +
'WHERE (username = ''' + login + ''') AND ' +
'(password = ''' + password + ''')';
  DataModule1.ADOQueryAuthorization.Open;
  value := DataModule1.DataSourceAuthorization.DataSet.Fields[0].AsString;
  if (value = '') then begin studentAuthentication := -1; Exit; end;
  studentAuthentication := StrToInt(value);
end;
procedure TAuthorization.showMainForm();
begin
  Main.Show();
  Authorization.Hide();
end;

```

```

end;
procedure TAuthorization.authorizationController(action: String);
var
  currentLogin, currentPassword: String;
  value: Integer;
begin
  hideErrorLabel();
  InputQuery('Input login', 'Login: ', currentLogin);
  InputQuery('Input password', 'Password: ', currentPassword);
  if (authorizationControllerCheckOnError(action, currentLogin, currentPassword) = true) then
    Exit;
  case StrUtils.IndexStr(action, ['admin', 'teacher', 'student']) of
    0: begin
      if (currentLogin = ADMIN_LOGIN) AND (currentPassword = ADMIN_PASSWORD) then begin
        userType := 'admin';
        showMainForm();
        Exit;
      end;
      showErrorLabel('Вы ввели неверный логин или пароль');
      Exit;
    end;
    1: begin
      value := teacherAuthentication(currentLogin, currentPassword);
      if (value = -1) then begin
        showErrorLabel('Вы ввели неверный логин или пароль');
        Exit;
      end;
      userType := 'teacher';
      userId := value;
      showMainForm();
    end;
    2: begin
      value := studentAuthentication(currentLogin, currentPassword);
      if (value = -1) then begin
        showErrorLabel('Вы ввели неверный логин или пароль');
        Exit;
      end;
      userType := 'student';
      userId := value;
      showMainForm();
    end;
  end;
end;
procedure TAuthorization.adminClick(Sender: TObject); begin authorizationController('admin'); end;
procedure TAuthorization.studentClick(Sender: TObject); begin authorizationController('student'); end;
procedure TAuthorization.teacherClick(Sender: TObject); begin authorizationController('teacher'); end;
end.

```

ДОДАТОК Б
(обов'язковий)
Приклади звітів

Рассписание

День недели	Номер пары	Название группы	Аудитория	Название предмета
понедельник	1	9-А	1	Алгебра
понедельник	2	9-А	3	Химия
понедельник	3	9-А	7	Украинская литература
понедельник	4	9-А	6	Физкультура
вторник	1	9-А	2	Геометрия
вторник	2	9-А	4	Физика
вторник	3	9-А	5	Украинский язык
вторник	4	9-А	9	Биология
среда	1	9-А	3	Химия
среда	2	9-А	1	Алгебра
среда	3	9-А	8	Зарубежная литература
среда	4	9-А	9	Биология
четверг	1	9-А	2	Геометрия
четверг	2	9-А	4	Физика
четверг	3	9-А	10	Английский язык
пятница	1	9-А	9	Биология
пятница	2	9-А	6	Физкультура
пятница	3	9-А	5	Украинский язык

Рисунок Б.1 – Звіт для таблиці «TimeTableReport»