ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення:

«Система обліку студентів групи»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Владислав НОСУЛЬСЬКИЙ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД- 42
Владислав НОСУЛЬСЬКИЙ
Ім'я, ПРІЗВИЩЕ

Керівник: Дмитро БІБІКОВ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____Каміла СТОРЧАК
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Носульському Владиславу Анатолійовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Розробка програмного забезпечення: «Система обліку студентів групи

керівник кваліфікаційної роботи _____Дмитро БІБІКОВ_____
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література з теми бакалаврської роботи.
2. Дослідження досвіду навчальних закладів.
3. Розробка Програмного Забезпечення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд існуючих рішень, порівняння та аналіз
2. Проектування проекту
3. Тестування та експлуатація програмного забезпечення
5. Ілюстративний матеріал: *презентація*

6. Дата видачі завдання: «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-05.03.2024	
2	Обгрунтування актуальності роботи	06.03-11.03.2024	
3	Аналіз основних Існуючих рішень та досвіду	12.03-27.03.2024	
4	Інструменти та прийоми розробки прикладного програмного забезпечення	28.03-10.04.2024	
5	Розробка програмного забезпечення: «система обліку студентів групи»	11.04-15.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	16.05-22.05.2024	
8	Розробка демонстраційних матеріалів	23.05-24.05.2024	

Здобувач(ка) вищої освіти _____ Владислав НОСУЛЬСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Дмитро БІБІКОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: 53 стор., 4 табл., 17 рис., 30 джерела.

Мета роботи – покращення процесу обліку студентів та автоматизації рутинних завдань шляхом розробки програмного забезпечення для обліку студентів групи.

Об'єкт дослідження – облік студентів в навчальних закладах.

Предмет дослідження – програмне забезпечення для обліку студентів групи.

Короткий зміст роботи: У роботі проаналізовано існуючі системи обліку студентів, визначено їхні недоліки та переваги. Визначено ключові функціональні та нефункціональні вимоги до системи обліку студентів. Розроблено архітектуру програмного забезпечення, використовуючи UML діаграми, для забезпечення гнучкості та ефективності. Впроваджено систему обліку студентів, яка автоматизує рутинні завдання, такі як реєстрація, перегляд академічної інформації, керування курсами та оцінювання студентів. Система була розроблена на основі технологічного стеку, що включає Python, Django, JavaScript та React, з використанням PostgreSQL для зберігання даних. Проведено тестування системи, яке підтвердило її надійність та ефективність. Система була успішно впроваджена та протестована в реальних умовах, що підтвердило її практичну цінність.

КЛЮЧОВІ СЛОВА: ОБЛІК СТУДЕНТІВ, АВТОМАТИЗАЦІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, UML ДІАГРАМИ, ТЕСТУВАННЯ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

ABSTRACT

Text part of the bachelor level qualification work: 53 pages, 17 pictures, 4 table, 30 sources.

The purpose of the work – is to improve the student record management process and automate routine tasks by developing software for managing student group records.

Object of research – the management of student records in educational institutions.

Subject of research – the software for managing student group records.

Summary of the work: The work analyzes the existing student record management systems, identifying their shortcomings and advantages. It defines the key functional and non-functional requirements for the system. Using UML, a clear architecture was created that meets the latest software development requirements. The developed system automates routine tasks such as registration, viewing academic information, course management, and student evaluation. The system, built with Python, Django, JavaScript, React, and PostgreSQL, was thoroughly tested, confirming its reliability and effectiveness. The successful deployment and real-world testing of the system validate its practical value.

KEYWORDS: STUDENT RECORD MANAGEMENT, AUTOMATION, SOFTWARE, UML DIAGRAMS, TESTING, SOFTWARE DEVELOPMENT.

ЗМІСТ

ВСТУП

У сучасному освітньому контексті, де навчальні заклади стикаються з різноманітними управлінськими викликами, створення програмного забезпечення для обліку студентів стає критично важливим, які відкривають можливості для ефективного використання ресурсів, підвищення якості освіти та задоволення потреб всіх учасників навчального процесу.

Розробка програмного забезпечення "Система обліку студентів групи" є *актуальною* темою у зв'язку з потребою сучасного освітнього середовища у засобах управління, які допомагають оптимізувати навчальний процес, підвищувати якість навчання, забезпечувати комфорт для студентів та автоматизувати рутинні завдання. Інформаційні системи обліку студентів відповідають таким вимогам, забезпечуючи зручний доступ до даних, аналіз інформації та використання сучасних технологій у навчальному процесі.

Об’єкт дослідження цієї роботи є процес розробки та впровадження програмного забезпечення для обліку студентів. Включає в себе аналіз потреб користувачів, визначення вимог до системи, проєктування архітектури системи, розробку програмного забезпечення, тестування та впровадження системи. Також включає в себе вивчення та аналіз методів та технологій, які використовуються для розробки таких систем. Включає в себе процеси пошуку, сортування, визначення активності групи та соціального паспорту групи, а також функції додавання, видалення та редагування даних студентів.

Предметом дослідження дипломної роботи є методи та технології, що використовуються для розробки програмного забезпечення системи обліку студентів, впровадження та ефективне функціонування системи обліку студентів в освітньому середовищі. Включає в себе вивчення та аналіз програмних мов, фреймворків, баз даних, а також методів проєктування, тестування та впровадження систем, охоплює вивчення взаємодії користувачів з системою та їх вимог до функціональності та зручності використання.

Мета роботи полягає в розробці та реалізації програмного забезпечення системи обліку студентів з функцією пошуку, сортування та можливістю визначення активності групи, а також з соціальним паспортом групи.

Система обліку студентів групи, як програмне забезпечення, має включати всі необхідні функції для автоматизації процесів, що стосуються обліку студентів. Вона має бути:

- легкою для розуміння та використання користувачами;
- стабільною у випадку збоїв та гарантувати безпечне зберігання даних;
- здатною адаптуватися та розширюватися відповідно до зростаючих потреб навчального закладу.

Щодо розробки та впровадження системи обліку студентів, можна використовувати такі методи дослідження:

- Аналіз наукової літератури;
- Дослідження досвіду інших навчальних закладів;
- Проєктування та розробка ПЗСОС;
- Тестування ПЗСОС.

Наукова новизна роботи виявляється у створенні системи обліку студентів групи, яка інтегрує функції пошуку, сортування та аналізу даних студентів, а також можливість визначення активності групи та формування соціального паспорту групи. Дана робота спрямована на розв’язання важливої проблеми керування навчальним процесом, а також на впровадження передових методів у сфері освіти за допомогою сучасних інформаційних технологій.

В результаті виконання цієї дипломної роботи буде створено систему обліку студентів групи, яка відповідатиме потребам освітнього закладу. Система наддасть можливість:

- автоматизувати процеси, що стосуються обліку студентів;
- зробити процес навчання більш ефективним;
- покращити якість освітніх послуг.

Практична важливість цієї роботи полягає в тому, що інформаційна система обліку студентів дозволяє ефективно керувати навчальним процесом. За допомогою

цієї системи адміністрація навчального закладу може швидко та ефективно отримувати доступ до даних про студентів, використовувати ці дані для планування курсів та ресурсів, а також виявляти та вирішувати проблеми, що виникають під час навчання. Крім того, можливість визначення активності групи та формування соціального паспорту групи дозволяє аналізувати та покращувати групове навчання. Сприяє підвищенню якості освіти, задоволенню потреб учасників навчального процесу та оптимізації використання ресурсів навчального закладу. Таким чином, розробка та впровадження системи обліку студентів групи має велике практичне значення для підвищення ефективності та продуктивності навчального процесу.

1 ОГЛЯД СУЧАСНИХ МЕТОДІВ У ВІДПОВІДНІЙ ОБЛАСТІ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ ФУНКЦІОНАЛЬНОСТІ ІСНУЮЧИХ СИСТЕМ

1.1 Аналіз сучасних систем обліку студентів в освітніх установах

Бурхливий розвиток інформаційних технологій, впровадження у всі сфери людського життя останніми роками сприяли різкого розширенню поняття оброблюваної в комп'ютерних системах інформації. Сьогодні не мислиме уявити діяльність як підприємств так навчальних закладів, не залучаючи у ньому інформаційних технологій.

Студентський менеджмент для куратора групи стає однією з найважливіших чинників виживання навчальних закладів у умовах ринкових відносин. Інколи мінімальні вкладення і забезпечити максимальне використання "людських ресурсів" дозволяють начальному закладу виграти в конкурентної боротьби.

Система обліку студентів групи існує кожному навчальному закладі, а роль куратора групи зростає. Він стає одним із основних працівників сучасного навчального закладу.

Планування роботи групи як інструмент цілеспрямованою і ефективнішої роботи з групою є складовою стратегії і тактики виживання та розвитку навчального закладу при ринкових взаємовідносинах. [3].

На сьогоднішній день на вітчизняному та зарубіжному ринках існує достатня кількість програмних продуктів. Але ефективність деяких припадає під сумнів. Оскільки кожний навчальний заклад висуває ряд вимог, яким мала б відповідати бажана програма, дуже важко підібрати універсальний програмний продукт, який підходитиме більшості українським навчальними закладам.

Було проведено аналіз програмних продуктів, що існують на українському ринку інформаційних технологій, які забезпечують автоматизацію управління персоналом.

Пакет програм "Деканат" − це автоматизована система управління вищим навчальним закладом (АСУ ВНЗ), яка призначена для організації та підтримки навчального процесу в вищих навчальних закладах України I-IV рівнів акредитації. Ключові особливості пакету програм:

Організація та підтримка навчального процесу. Пакет дозволяє створити та підтримувати базу даних, в якій реєструється та формується інформація про структуру навчального процесу вищого навчального закладу.

Клієнт-серверна технологія. Пакет построєний за клієнт-серверною технологією, що дозволяє встановлювати його на множину комп'ютерів, які об'єднані в локальну мережу та працюють з єдиною базою даних.

Web-сценарії. Використання додаткових web-сценаріїв забезпечує можливість доступу до бази даних в межах окремих програм Пакету з всесвітньої павутини Інтернет.

Програма "ПС-Адміністратор". Входить до складу Пакету і призначена для щоденного тестування, резервного копіювання та, при необхідності, відновлення бази даних.

Інформаційна сумісність. Інформаційна сумісність з іншими продуктами ПП "Політек-СОФТ" забезпечує імпортування даних, які вже були внесені в бази даних інших продуктів.

Конструктор звітів. Пакет має зручний конструктор звітів, який дозволяє створювати та редагувати вже існуючі звітні документи, використовуючи HTML - мову розмітки гіпертексту.

Сумісність з операційними системами. Пакет працює в операційних системах Windows 95/98/ME/NT/2000/2003/XP/Vista/7.

Пакет програм "Деканат" для вищих навчальних закладів.

Пакет дозволяє створити та підтримувати базу даних, в якій реєструється та формується така інформація:

- структура навчального процесу вищого навчального закладу (факультети, кафедри, спеціальності, спеціалізації, навчальні плани, академічні та збірні групи, підгрупи, лекційні потоки);

- дані щодо навантаження кафедр (з генерацією звіту за Ф. У-4.01);
- результати обрахунку штатів кафедр;
- дані щодо всіх викладачів вищого навчального закладу та їх планового навантаження, розклад їх роботи;
- дані щодо всіх студентів вищого навчального закладу та подій, що відбуваються з ними під час навчання (оцінки, відвідування занять, рух студентів);
- аудиторний фонд вищого навчального закладу, його використання, розклад занять [4].

Підсистема "Електронний журнал", у якій передбачено шість рівнів доступу:

- "Студент" − має право на перегляд успішності своєї групи і на проходження тестів. Після першого входу, студенту рекомендується змінити пароль.
- "Викладач" − має право на зміну оцінок в журналі з предметів, які він викладає, на створення і редагування тестів зі своїх предметів, планування проходження тестів для окремих груп.
- "Староста" − має право на зміну оцінок та відвідуваність в журналі з предметів своєї групи.
- "Ректорат" − має право на перегляд успішності студентів всіх груп і факультетів.
- "Куратор" − має право на перегляд успішності студентів своєї групи.
- "Адміністратор" − має необмежений доступ до всіх описаних сервісів, і до додаткових модулів адміністрування бази даних.

Автоматизована система "Школа" − це програмно-технологічний комплекс для ефективного управління загальноосвітнім закладом.

АС "Школа" автоматизує всі ключові процеси діяльності установи освіти:

- навчально-виховна та адміністративна діяльність;
- планування, організацію та оперативне управління навчальним процесом;

– діловодство;
– формування, обмін та друк поточної, службової, статистичної інформації;
– обмін інформацією з органами управління освіти;
– питання господарського порядку;
– безпека навчального закладу (автоматизована система інформування клієнтів (АС) "Школа").

В системі реалізовано:

– автоматичне заповнення, перевірку, редагування звітів ЗНЗ-1, ЗНЗ-2, 76-РВК, 83-РВК та їх похідних форм, на основі інформації, внесеної до бази даних;
– автоматичне формування замовлень на виготовлення документів про освіту державного зразка та учнівських квитків;
– інтеграцію з інформаційним простором "Моя Освіта". АС "Школа" та "Моя Освіта";
– створюють єдиний інформаційно-освітній простір інтерактивної взаємодії всіх учасників;
– освітнього процесу: учнів, батьків, вчителів, адміністрації школи, управлінь освіти.

АС "Школа". Діє акційна пропозиція, за якою учні, чиї школи користуються інформаційним простором "Моя Освіта", отримують безкоштовний доступ до двох комплектів тестів на порталі "Абітурієнт". Учні обирають тести індивідуально за власним бажанням.

Детальний опис акції розміщено на порталі "Абітурієнт".

Директори шкіл отримують надійний інструмент для оперативного прийняття управлінських рішень та сучасну організацію управління навчальним закладом.

Методисти та заступники директора в системі АС "Школа" (рис.1.1):

– ведуть бази даних по вчителям, включаючи інформацію по атестації, перепідготовку та курси підвищення кваліфікації, посадові оклади;
– ефективно управляють інфраструктурою школи (навчальні класи, спеціалізовані кабінети і т.д.);

– складають і контролюють навчальні плани учнів, в т.ч. індивідуальні;
– складають розклад уроків з розбивкою на класи та підгрупи з закріпленням навчальних класів.

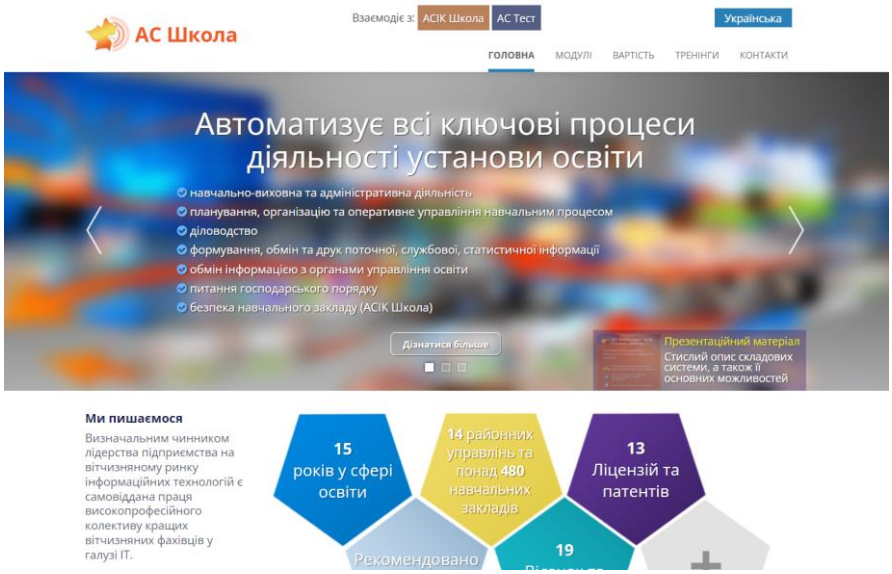


Рисунок 1.1 – Інтерфейс системі АС "Школа"

Вчителі:

– ведуть базу даних учнів;
– вносять підсумкові оцінки учнів у базу даних, працюють з табелями;
– ведуть звітність за результатами підсумкового контролю відповідно до навчальних планів;
– публікують рейтинги успішності учнів.

Переваги АС "Школа":

– унікальна технологія захисту інформації в системі;
– он-лайн інформування батьків про навчально-виховний процес учня;
– технологія "Редаговані статистичні звіти";
– управління і розподіл прав доступу системи;
– обмін даними з ІВС "Освіта";
– індивідуальне навчання, методичні посібники, контекстна довідка;
– регулярне оновлення і постійна сервісна підтримка [5].

1.2 Визначення та аналіз ключових вимог до системи обліку студентів

Аналіз вимог до системи обліку студентів включає в себе ідентифікацію та визначення ключових функціональних та нефункціональних вимог, що відповідають потребам користувачів. Розглянемо кожну категорію користувачів − студентів, викладачів та адміністрації − та їхні потреби:

1. Студенти:
   - Реєстрація в системі − можливість створення особистого профілю з введенням основних даних (ім'я, прізвище, контактна інформація).
   - Перегляд академічної інформації – доступ до розкладу занять, списку курсів, результатів іспитів та інших академічних даних.
   - Запис на курси – можливість вибору та запису на необхідні курси через систему.
   - Перегляд власних оцінок − можливість перегляду академічних досягнень та оцінок.
2. Викладачі:
   - Керування курсами − можливість створення та редагування інформації про курси, внесення змін до розкладу, оновлення списку студентів тощо.
   - Оцінювання студентів − можливість виставлення оцінок за завдання та іспити, перегляду заяв на переоцінку тощо.
   - Комунікація зі студентами − можливість надсилання повідомлень та оголошень студентам через систему.
3. Адміністрація:
   - Керування користувачами − можливість створення, блокування та видалення облікових записів користувачів.
   - Звітність − можливість генерації звітів про кількість зареєстрованих студентів, відвідуваність занять, успішність тощо.
   - Підтримка − можливість надання технічної підтримки користувачам системи.
4. Нефункціональні вимоги можуть включати:

– Безпека − захист персональних даних студентів та викладачів.
– Ефективність − швидка відповідь системи на запити користувачів.
– Надійність − запобігання втраті даних та збоїв системи.

Після аналізу цих вимог можна розробити детальний план реалізації системи обліку студентів, визначивши функціональні та технічні характеристики системи.

1.3 Технічний аналіз та оцінка існуючих систем обліку студентів

Для проведення детального аналізу технічних характеристик існуючих систем обліку студентів, важливо розглянути кілька основних аспектів, таких як функціональність, масштабованість, безпека, ефективність та інтеграційні можливості.

Функціональність. Система повинна мати можливість реєстрації студентів та викладачів, керування курсами та розкладом занять, виставлення оцінок та генерації звітів. Потрібна можливість комунікації між користувачами системи через повідомлення та оголошення.

Масштабованість. Система повинна бути готова до росту числа користувачів та обсягу даних з часом без втрати продуктивності.

Безпека. Необхідно забезпечити захист персональних даних користувачів, а також запобігти несанкціонованому доступу до системи та даних. Важливо мати систему резервного копіювання даних для запобігання втраті інформації у випадку аварійних ситуацій.

Ефективність. Система повинна мати швидку реакцію на запити користувачів та високу швидкість обробки даних, особливо в часи пікового навантаження.

Інтеграційні можливості. Означає здатність системи взаємодіяти з іншими існуючими системами університету, такими як системи електронного навчання, бібліотечні системи, тощо.

Ось кілька переваг та недоліків існуючих систем, які можна врахувати:

Переваги:

– Функціональність. Можливість вести повний облік студентів та курсів.

– Досвід. Існуючі системи можуть враховувати деякий досвід у розробці нової системи.
– Стабільність. Якщо існуюча система довго працює без проблем, це може свідчити про її стабільність.

Недоліки:

– Застарілі технології. Можливо, існуюча система побудована на застарілих технологіях, що обмежує її ефективність та масштабованість.
– Недостатня безпека. Деякі системи можуть мати проблеми з безпекою, такі як вразливості до кібератак або недостатні заходи захисту персональних даних.
– Неадекватна ефективність. Деякі системи можуть працювати повільно або неефективно під час великого навантаження.

Загалом, аналіз існуючих систем допоможе видокремити найкращі практики та уникнути недоліків при розробці нової системи обліку студентів.

1.4 Аналіз функціональних можливостей систем обліку студентів

При оцінці функціональності існуючих систем обліку студентів важливо розглянути, наскільки вони здатні відповідати потребам користувачів, таких як студенти, викладачі та адміністрація. Розглянемо ключових аспектів, які можуть бути враховані при оцінці:

Реєстрація та аутентифікація користувачів:

– Чи є в системі зручний процес реєстрації для студентів, викладачів та адміністрації?
– Наскільки ефективно забезпечена аутентифікація користувачів із застосуванням різних методів (наприклад, паролі, одноразові коди, біометрія)?

Модуль керування курсами та розкладом занять:

– Чи є в системі можливість створення та редагування курсів?
– Наскільки зручно відображається розклад занять для користувачів?

Модуль оцінювання студентів:

- Чи є в системі зручний інтерфейс для виставлення оцінок студентам?
- Наскільки ефективно система обробляє і зберігає результати оцінювання?

Модуль комунікації:

- Чи є в системі засоби для комунікації між користувачами (наприклад, повідомлення, чат)?
- Наскільки зручно користуватися цими інструментами та яка їхній функціонал?

Доступ до академічної інформації:

- Чи можуть студенти та викладачі з легкістю отримувати доступ до своєї академічної інформації (результати, розклад тощо)?
- Наскільки зручний та інтуїтивний інтерфейс для перегляду цієї інформації?

Система підтримки користувачів:

- Чи є в системі можливість звертатися за допомогою або технічною підтримкою?
- Наскільки швидко та ефективно вирішуються проблеми та запити користувачів?

Оцінка функціональності існуючих систем дозволить визначити їхні переваги та недоліки в контексті потреб користувачів і допоможе при розробці нової системи обліку студентів.

1.5 Обрання технологічного стеку для розробки системи обліку студентів

При виборі технологій для розробки нової системи обліку студентів важливо врахувати ряд факторів, таких як функціональність системи, потреби користувачів, масштаб проєкту, ефективність, безпека та зручність розробки та підтримки:

Мова програмування:

Python є популярним вибором для веброзробки завдяки своїй простоті, легкості використання та широкому спектру наявних бібліотек для обробки даних та розробки вебдодатків.

JavaScript (з фреймворком React або Angular) є ключовою мовою для фронтенд-розробки вебдодатків, а React (рис.1.2) або Angular (рис.1.3) допоможуть покращити продуктивність та організацію коду.

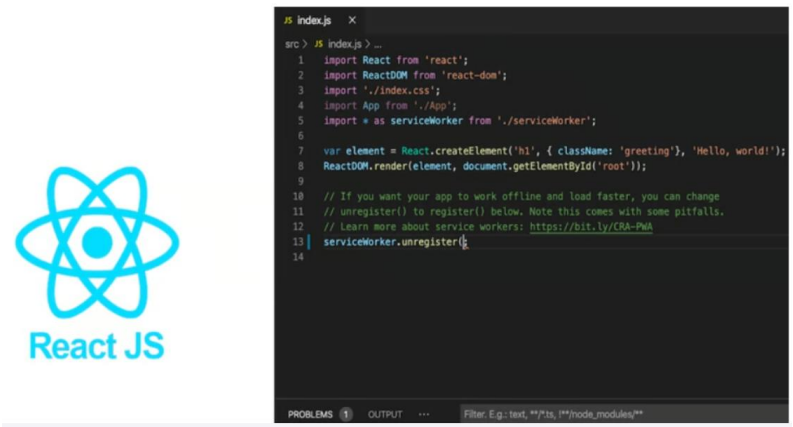


Рисунок 1.2 – Приклад написаний у фреймворці React

Рисунок 1. 3 − Приклад написаний у фреймворці Angular

SQL або NoSQL в залежності від потреб системи використовується SQL (наприклад, PostgreSQL або MySQL) для структурованих даних або NoSQL (наприклад, MongoDB) для більш гнучкої структури даних.

База даних:

PostgreSQL – потужна та надійна реляційна база даних, яка забезпечить стабільність та безпеку для системи обліку студентів (рис.1.4).

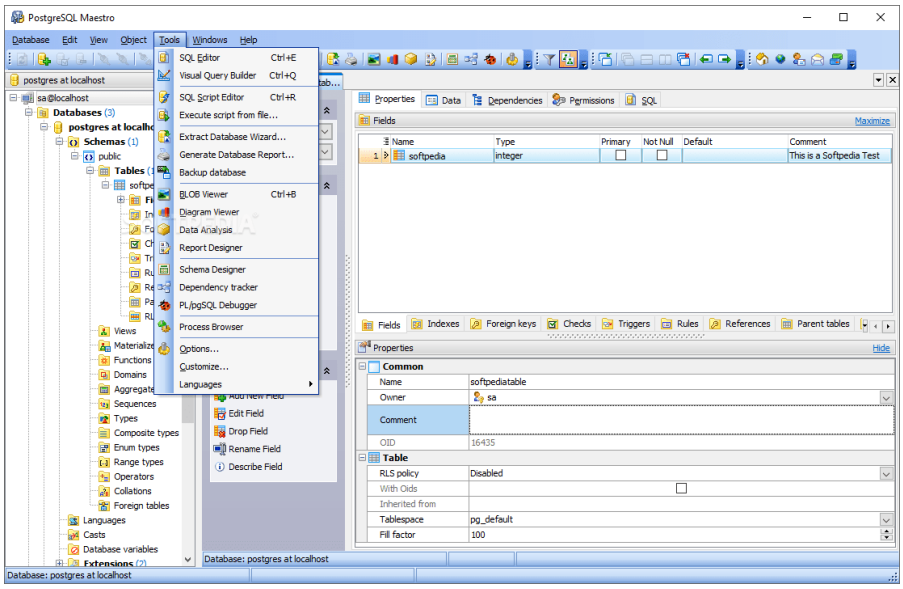


Рисунок 1.4 – Інтерфейс PostgreSQL

MongoDB – якщо потрібно працювати з більш гнучкою структурою даних або великим обсягом неструктурованих даних (рис.1.5).

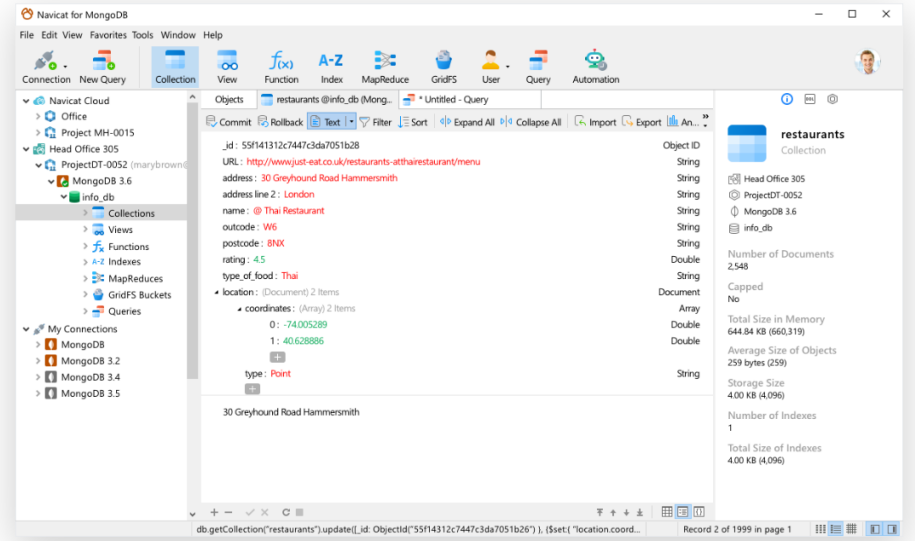


Рисунок 1.5 – Інтерфейс MongoDB

Фреймворк для розробки вебдодатків:

Django або Flask (Python) для бекенд-розробки вебдодатків Python фреймворки, такі як Django або Flask, можуть бути чудовим вибором, оскільки вони пропонують швидкий старт, гарну документацію та розширюваність.

Django − це високорівневий вебфреймворк Python, який сприяє швидкому розробуванню та чистому, прагматичному дизайну, створений досвідченими розробниками і допомагає усунути багато проблем веброзробки, тому єможливість зосередитись на написанні вашого додатку, не потрібно винаходити колесо.

Ключові особливостей Django:

Швидкість. Django був розроблений, щоб допомогти розробникам якомога швидше перевести додатки від концепції до завершення.

Безпека. Django серйозно ставиться до безпеки і допомагає розробникам уникнути багатьох поширених помилок безпеки.

Масштабованість. Деякі з найбільш завантажених сайтів в Інтернеті використовують можливість Django швидко та гнучко масштабуватися.

Django також має власну систему шаблонів, яка призначена для досягнення балансу між потужністю та легкістю (рис.1.6). Призначена для комфортної роботи з HTML для дизайнерів та розробників фронтенду. Але також гнучка та високо розширювана, що дозволяє розробникам розширювати мову шаблонів за потреби.

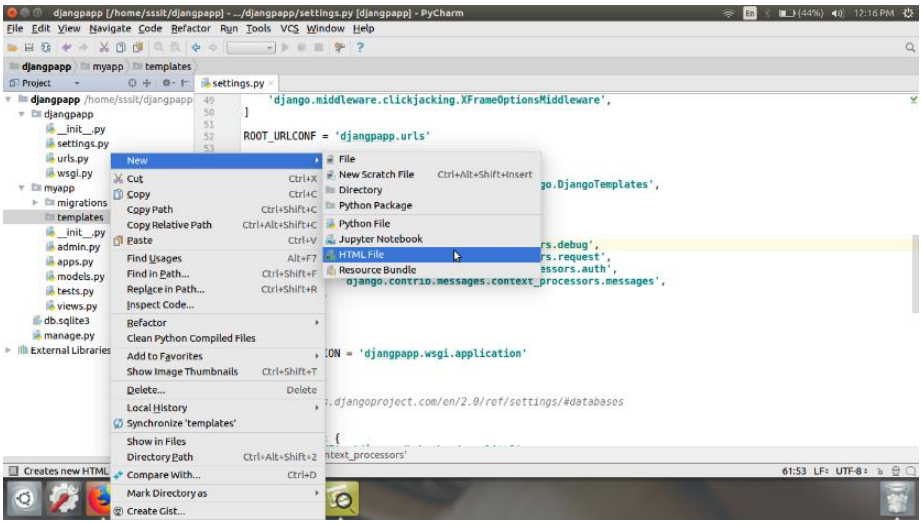


Рисунок 1.6 − Інтерфейс Django

Flask − це вебфреймворк Python, який дозволяє розробникам швидко та легко створювати вебдодатки (рис.1.7). Був розроблений Арміном Ронахером, лідером міжнародної групи ентузіастів Python (POCCO), і базується на інструментарії WSGI Werkzeug та двигуні шаблонів Jinja.

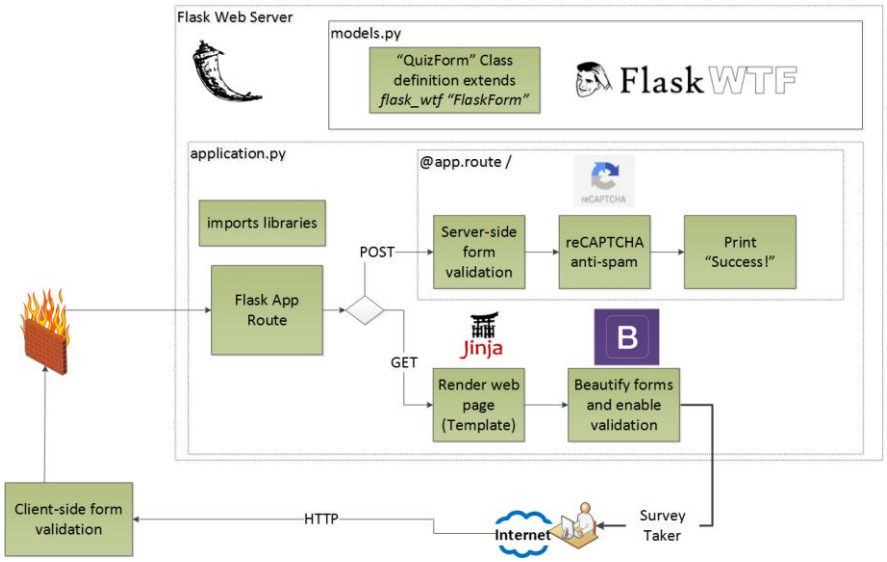


Рисунок 1.7 – Приклад в середовищі Flask

Ключові особливостей Flask:

Легкість використання. Flask розроблений таким чином, щоб допомогти розробникам якомога швидше перевести додатки від концепції до завершення.

Безпека. Flask серйозно ставиться до безпеки і допомагає розробникам уникнути багатьох поширених помилок безпеки.

Масштабованість. Деякі з найбільш завантажених сайтів в Інтернеті використовують можливість Flask швидко та гнучко масштабуватися.

Flask також має власну систему шаблонів, яка призначена для досягнення балансу між потужністю та легкістю, для комфортної роботи з HTML для дизайнерів та розробників фронтенду. Але вона також гнучка та високо розширювана, що дозволяє розробникам розширювати мову шаблонів за потреби.

React або Angular (JavaScript) для фронтенд-розробки JavaScript фреймворки, такі як React або Angular, допоможуть створити модульний та ефективний фронтенд.

Інші технології:

RESTful API для забезпечення взаємодії між фронтендом та бекендом можна використовувати RESTful API для передачі даних.

RESTful API (Representational State Transfer) − це архітектурний стиль для розробки вебсервісів, який використовує HTTP протокол для обміну даними. Він був представлений Роєм Філдінгом у 2000 році в його дисертації. З того часу він став одним з найбільш широко використовуваних підходів для створення вебAPI.

Ось декілька ключових принципів RESTful API:

Єдиний інтерфейс. RESTful API використовує єдиний інтерфейс для спілкування між клієнтом та сервером. Це означає, що всі ресурси мають унікальні ідентифікатори, а повідомлення містять достатньо інформації для обробки.

Клієнт-сервер. Принцип вимагає відокремлення клієнта та сервера, що дозволяє їм розвиватися незалежно.

Безстановість. Кожний запит від клієнта до сервера має всю необхідну інформацію для його обробки.

Кешування. Відповіді сервера можуть бути кешовані для покращення швидкості та ефективності.

Шарування системи. Клієнт не може точно визначити, чи він спілкується безпосередньо з сервером, чи з проміжним ресурсом.

Код за запитом (необов'язково). Сервер може надавати виконуваний код для підтримки розширення функціональності клієнта.

RESTful API використовується в різних областях, включаючи веброзробку, хмарні додатки, Інтернет речей та багато інших.

Docker та Kubernetes дозволяють зручно керувати розгортанням та масштабуванням додатків.

Kubernetes, також відомий як K8s, − це відкрита система для автоматизації розгортання, масштабування та управління контейнеризованими додатками. Групує контейнери, які складають додаток, в логічні одиниці для простого управління та виявлення.

Ключові особливості Kubernetes:

Автоматизовані розгортання та відкати. Kubernetes поступово вносить зміни до додатку або його конфігурації, в той час слідкуючи за здоров'ям додатку, щоб не вбивати всі екземпляри одночасно.

Відкриття сервісів та балансування навантаження. Немає потреби модифікувати додаток, щоб використовувати незнайомий механізм виявлення сервісів. Kubernetes надає Pods власні IP-адреси та єдине DNS-ім'я для набору Pods і може балансувати навантаження між ними.

Оркестрація зберігання. Автоматично монтує систему зберігання на вибір, незалежно від того, чи це локальне зберігання, провайдер публічної хмари, або мережева система зберігання, така як iSCSI або NFS.

Самозцілення. Перезапускає контейнери, які зазнали невдачі, замінює та перепланує контейнери, коли вузли помирають, вбиває контейнери, які не відповідають на користувацьку перевірку здоров'я, і не рекламує їх клієнтам, поки не будуть готові обслуговувати.

Docker − це платформа, що допомагає розробникам створювати, ділитися, запускати та перевіряти додатки будь-де за допомогою контейнерів. Ключові особливостей Docker:

Створення. Docker дозволяє розробникам створювати унікальні додатки за допомогою Docker-зображень та створювати багатоконтейнерні додатки за допомогою Docker Compose.

Інтеграція з існуючими інструментами. Docker працює з усіма інструментами розробки, такими як VS Code, CircleCI та GitHub.

Контейнеризація додатків для забезпечення узгодженості. Docker дозволяє запускати додатки в будь-якому середовищі узгоджено, від Kubernetes на власних серверах до AWS ECS, Azure ACI, Google GKE та інших.

Docker Desktop − безпечний та простий у використанні інструмент для створення, ділення та запуску додатків з контейнерами.

Спільнота Docker має активну спільноту розробників, яка допомагає у вирішенні проблем та обміні досвідом.

Docker допомагає розробникам прискорити налаштування та розгортання своїх розробницьких середовищ, а також підтримує розробку сучасних AI/ML-додатків. Docker також інтегрується з IDE, що дозволяє створювати та керувати контейнерами безпосередньо з IDE. Обираючи технології, важливо враховувати не тільки поточні потреби, а й майбутні перспективи розвитку проєкту, його можливості для розширення та підтримки.

1.6 Створення проєкту архітектури системи обліку студентів

При проєктуванні архітектури системи обліку студентів, важливо ретельно розробити структуру бази даних, класи, взаємодію між компонентами системи та інші аспекти. Розглянемо можливого підходу до розробки архітектури системи:

Структура бази даних:

- Використання реляційної бази даних для зберігання інформації про студентів, викладачів, курси, розклад занять та інші важливі дані.
- Таблиці можуть включати студентів (з основними даними), викладачів, курси, групи, розклад занять, оцінки тощо.
- Забезпечення належної нормалізації бази даних для забезпечення ефективності та надійності.

Діаграми класів:

- Розробка діаграм класів допоможе визначити основні класи системи та їх взаємозв'язки.
- Класи можуть включати класи для студентів, викладачів, курсів, розкладу занять, оцінок, користувачів тощо.
- Визначення атрибутів та методів кожного класу для виконання відповідних функцій системи.

Діаграми послідовностей:

- Розробка діаграм послідовностей допоможе визначити взаємодію між різними компонентами системи.

- Послідовності можуть включати процеси реєстрації студентів, запису на курси, виставлення оцінок тощо.
- Визначення послідовностей операцій та передачі даних між класами системи.

Архітектурні шаблони:

- Використання шаблонів архітектури, таких як MVC (Model-View-Controller) або MVP (Model-View-Presenter), допоможе забезпечити чітку структуру та розділення відповідальностей між компонентами системи.
- MVC може бути особливо корисним для вебдодатків, де модель відповідає за дані, представлення за відображення даних користувачу, а контролер за управління взаємодією між моделлю та представленням.

Після розробки архітектури системи, буде важливо провести тестування та валідацію, щоб переконатися в її ефективності та відповідності вимогам до системи.

1.7 Процес створення системи обліку студентів

Процес розробки системи включає кілька етапів, починаючи від аналізу вимог і закінчуючи випуском готового продукту. Розглянемо кожний етап:

Аналіз вимог:

- На цьому етапі визначаються функціональні та нефункціональні вимоги до системи на основі потреб користувачів.
- Проводиться узгодження замовником щодо обсягу робіт, функціональності та інших ключових аспектів проєкту.

Проєктування:

- Розробляється архітектура системи, включаючи структуру бази даних, діаграми класів, послідовностей та інші аспекти.
- Вибираються технології та інструменти, які будуть використовуватися для розробки.

Реалізація (написання коду):

- Розробляються різні модулі та компоненти системи згідно з визначеними вимогами та архітектурою.
- Кожен модуль підлягає кодуванню відповідно до обраної мови програмування та практик розробки.

Тестування:

- Проводиться функціональне та інтеграційне тестування для перевірки відповідності розроблених компонентів вимогам та їх взаємодії.
- Виявлені помилки та недоліки виправляються.

Виправлення помилок:

- Коригування виявлених помилок та допрацювання функціональності, яка не відповідає вимогам.
- Повторне тестування для перевірки ефективності внесених змін.

Реліз:

- Готовий продукт готується до релізу та випускається для використання користувачами.
- Враховуються можливості розгортання та підготовка до впровадження у виробниче середовище.

Підтримка та розвиток:

- Після релізу система підтримується, виявлені помилки виправляються, а користувачі отримують технічну підтримку.
- Враховуються нові потреби та вимоги користувачів для подальшого розвитку системи.

Даний цикл може повторюватися в залежності від вимог та потреб користувачів для постійного вдосконалення та підтримки системи.

1.8 Оцінка працездатності та готовності системи обліку студентів до впровадження

Тестування та впровадження системи є критичним етапом у життєвому циклі розробки програмного забезпечення. Ось детальний огляд процедур тестування та впровадження системи:

Тестування:

Функціональне тестування. Перевірка, чи працюють усі функції системи відповідно до вимог користувача. Включає тестування інтерфейсу, роботи з базою даних, роботи зовнішніх систем та інше.

Інтеграційне тестування. Перевірка взаємодії між різними компонентами системи. Це важливо, оскільки навіть працюючі окремо компоненти можуть несправно працювати разом.

Тестування безпеки. Перевірка системи на вразливість до атак та зловживань, а також дотримання стандартів безпеки даних.

Навантажувальне тестування. Визначення, як система реагує на велику кількість запитів або одночасних користувачів. Дозволяє виявити слабкі місця та оптимізувати систему.

Тестування відмовостійкості. Перевірка, як система поводиться в разі відмови окремих компонентів або сервісів. Важливо переконатися, що система може гідно працювати під час непередбачених обставин.

Оцінка працездатності та готовності до впровадження:

Перевірка виконання вимог. Перевірка, чи виконує система всі вимоги, визначені на етапі аналізу та проєктування.

Стійкість та ефективність. Перевірка, чи працює система стабільно та ефективно, а також чи вона відповідає критеріям продуктивності.

Документація та навчання. Переконання, що документація проєкту та інструкції з використання системи готові до використання. Підготовка навчальних матеріалів для користувачів системи.

Тестове впровадження. Запуск системи в обмеженому середовищі для перевірки її роботи в реальних умовах перед повним впровадженням.

План впровадження. Розроблення плану впровадження, включаючи розгортання, підготовку персоналу та впровадження системи в роботу.

Тільки після успішного завершення цих кроків система може бути готовою до повного впровадження і використання користувачами.

1.9 Аналіз та оцінка вартості впровадження та експлуатації систем обліку студентів

Оцінка вартості впровадження та експлуатації системи обліку студентів включає в себе ряд факторів, які потрібно врахувати для повного уявлення про загальну вартість проєкту. Розглянемо кілька ключових аспектів, які слід врахувати при оцінці вартості:

Вартість розробки − включає в себе оплату розробників або розробних команд, які будуть займатися створенням системи, а також витрати на необхідне програмне забезпечення та інструменти для розробки.

Вартість впровадження − включає в себе витрати на налаштування системи, інтеграцію з існуючими системами університету, підготовку персоналу та навчання користувачів.

Вартість обслуговування та підтримки − включає в себе витрати на технічну підтримку, оновлення системи, виправлення помилок та інші адміністративні витрати.

Вартість інфраструктури − включає в себе витрати на обладнання, хмарні послуги, мережеві ресурси та інші інфраструктурні витрати, необхідні для роботи системи.

Вартість ліцензування та сертифікації − якщо система використовує комерційне програмне забезпечення або потребує сертифікації, це також може становити значну частину вартості проєкту.

Вартість забезпечення безпеки та конфіденційності − включає в себе витрати на захист системи від кібератак, реалізацію заходів з конфіденційності та дотримання вимог щодо обробки персональних даних.

Вартість навчання та підтримки користувачів − включає в себе витрати на підготовку та навчання персоналу, а також на створення документації та матеріалів для користувачів системи.

Оцінка ризиків та резервів. Варто також врахувати ризики та несподівані витрати, які можуть виникнути під час реалізації проєкту, та відповідно включити резервні кошти у бюджет.

Після аналізу цих факторів можна скласти повний фінансовий план проєкту, в якому буде відображено загальну вартість впровадження та експлуатації системи обліку студентів.

1.10 Вибір найбільш підходящих або перспективних варіантів для подальшого дослідження та порівняння

Дипломна робота на тему "Розробка програмного забезпечення «Система обліку студентів групи»", яка буде розроблена на платформі RAD Studio 12, спрямована на створення комплексного програмного забезпечення для ефективного обліку та управління даними студентів навчального закладу. RAD Studio 12 − це потужне інтегроване середовище розробки, що дозволяє розробникам створювати додатки для різних операційних систем, включаючи Microsoft Windows, Mac OS, а також мобільні платформи Apple iOS та Android. Використання RAD Studio 12 (рис.1.8) спрощує процес розробки завдяки широкому спектру інструментів і компонентів, які включають в себе різноманітні функції та можливості, необхідні для створення інформаційної системи обліку студентів [1,2].

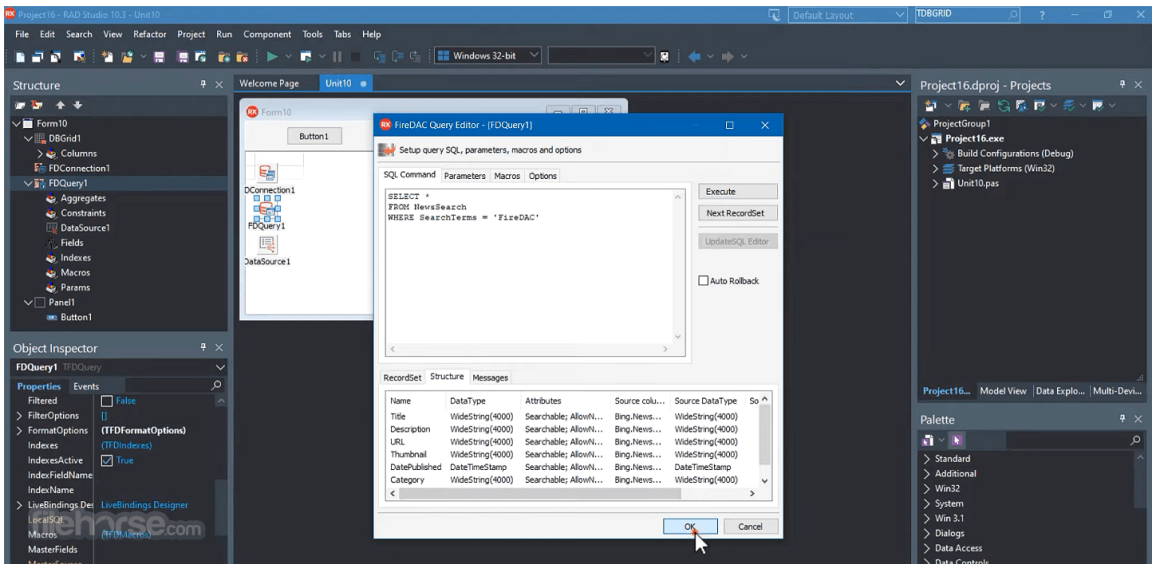


Рисунок 1.8 – Інтерфейс RAD Studio 12

RAD Studio 12 (також відоме як RAD Studio 12 Athens) − це середовище розробки програмного забезпечення (IDE) від Embarcadero, призначене для створення крос-платформних високопродуктивних нативних додатків. Воно підтримує розробку на мовах Delphi і C++.

Основні можливості RAD Studio 12:

– розробляються програми для Windows, macOS, Android, iOS, Linux і веб-платформ (HTML5, JavaScript) з єдиним кодом.
– використовуються потужні візуальні інструменти для швидкого створення інтерфейсів користувача:
  ~ VCL (Visual Component Library) − бібліотека візуальних компонентів для розробки Windows-додатків, містить різноманітні компоненти, такі як кнопки, текстові поля, списки, таблиці, меню та багато інших. Можна легко перетягувати дані компоненти на форму та налаштовувати їх властивості.
  ~ FMX (FireMonkey) − платформонезалежна бібліотека візуальних компонентів, яка дозволяє розробляти додатки для Windows, macOS, iOS, Android та Linux. FMX має сучасний вигляд, підтримує анімацію, векторну графіку та може працювати з різними роздільними здатностями екрану.

- ~ LiveBindings − механізм, який дозволяє зв’язувати дані між компонентами та джерелами даних без написання коду. Можна зв’язувати властивості компонентів з полями бази даних, об’єктами, файлами XML та іншими джерелами даних.
- ~ Styles - RAD Studio дозволяє використовувати стилі для зовнішнього вигляду вашого додатка. Можна вибрати один з вбудованих стилів або створити власний. Допомагає зробити додаток більш сучасним та привабливим.
- ~ Редактор форм − можна візуально розміщувати компоненти на формі, налаштовувати їх властивості та створювати події безпосередньо в редакторі форм. Спрощує роботу з інтерфейсом користувача.

– доступ до широкого спектра компонентів для різних цілей, таких як мережеві операції, робота з базами даних, мультимедіа тощо.

- ~ FireMonkey (FMX) − платформонезалежна бібліотека візуальних компонентів, яка дозволяє розробляти додатки для Windows, macOS, iOS, Android та Linux. FMX має сучасний вигляд, підтримує анімацію, векторну графіку та може працювати з різними роздільними здатностями екрану.
- ~ FireDAC − набір компонентів дозволяє нативний високошвидкісний прямий доступ з Delphi до баз даних, таких як InterBase, SQLite, MySQL, SQL Server, Oracle, PostgreSQL, IBM DB2, SQL Anywhere, Access, Firebird, Informix та багато інших.
- ~ LiveBindings − механізм, який дозволяє зв’язувати дані між компонентами та джерелами даних без написання коду. Можна зв’язувати властивості компонентів з полями бази даних, об’єктами, файлами XML та іншими джерелами даних.
- ~ VCL (Visual Component Library) − бібліотека візуальних компонентів для розробки Windows-додатків, містить різноманітні компоненти, такі як кнопки, текстові поля, списки, таблиці, меню та багато інших.

Можна легко перетягувати компоненти на форму та налаштовувати їх властивості.

– підключається до різних баз даних, таких як InterBase, SQLite, MySQL, SQL Server та Oracle:
  ~ FireDAC − набір компонентів дозволяє нативний високошвидкісний прямий доступ з Delphi до баз даних, таких як InterBase, SQLite, MySQL, SQL Server, Oracle, PostgreSQL, IBM DB2, SQL Anywhere, Access, Firebird, Informix та багато інших;
  ~ dbExpress − компоненти надають швидкий доступ до SQL-серверів баз даних.
  ~ ADO Components − набір містить компоненти dbGo, які дозволяють підключатися до ADO-сховища даних, виконувати команди та отримувати дані з таблиць у базах даних, використовуючи фреймворк ADO.
  ~ InterBase Components − компоненти дозволяють безпосередньо отримувати доступ до баз даних InterBase без проміжного рівня.
  ~ Data Access Components − на цій сторінці містяться компоненти, які можна використовувати з будь-яким механізмом доступу до даних. Наприклад, TClientDataset може працювати з даними, збереженими на диску, або використовувати компонент TDataSetProvider для співпраці з компонентами з інших груп [3,2].
– отримуються переваги від інтелектуального автодоповнення коду, відладки та профілювання.
  ~ Інтелектуальне автодоповнення коду:
    – RAD Studio 12 надає потужні інструменти для автоматичного доповнення коду, допомагає розробникам швидше писати код, зменшуючи кількість ручного введення.
    – за допомогою автодоповнення можна швидко знайти доступні методи, властивості та змінні, що спрощує роботу зі складними API та бібліотеками.

~ Відладка:
  - RAD Studio 12 має потужний відладчик, який допомагає виявляти та виправляти помилки в коді;
  - за допомогою точок зупину, перегляду змінних та трасування стеку можна аналізувати виконання програми та виявляти проблеми.
~ Профілювання дозволяє аналізувати продуктивність вашого додатка; можна визначити, які частини коду вимагають більше ресурсів (наприклад, часу виконання або пам'яті) та оптимізувати їх.

– розробляються та розгортаються програми для хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure та Google Cloud Platform.
~ Доступ до хмарних RESTful веб-сервісів. RAD Studio 12 має бібліотеку REST-клієнта, доступну на всіх платформах. Спрощує виклик REST-сервісів будь-яким постачальником сторонніх послуг. Бібліотека підтримує аутентифікацію та маніпуляцію відповідями JSON, а також маппінг даних набору даних та LiveBindings.
~ Підключення до Amazon та Azure Services, включаючи служби додатків, бази даних та зберігання, за допомогою RAD Studio Amazon API та Azure API. Хмарні сервіси визначені у зручних типах та класах, що спрощує аутентифікацію та використання ключових служб, таких як таблиці, сховище, черги та відра.
~ Інтеграція з постачальниками BaaS (Back-end as a Service):
  - можна легко інтегрувати хмарні сервіси від постачальників BaaS, таких як Kinvey, Parse та App;
  - отримати простий доступ до послуг, таких як сповіщення, аутентифікація та зберігання, без необхідності будувати їх самостійно або підтримувати;
~ RAD Server − це модульне рішення для розподілених додатків, яке надає легко розгортати сервер middleware, який містить модулі API та доступу до даних. Підтримує Push-сповіщення для iOS та Android,

інтегровану аутентифікацію, аналітику використання API в реальному часі та багато іншого [11,12].

Нові функції в RAD Studio 12 (Athens):

- інтеграція Visual Assist для C++, завдяки автодоповненню коду, навігації та перейменуванню:
  - ~ Автодоповнення коду. Visual Assist дозволяє швидко знаходити доступні функції, змінні та класи, що спрощує роботу зі складними API та бібліотеками, використовуючи автодоповнення для швидкого введення коду та уникнення помилок.
  - ~ Навігація за допомогою Visual Assist дає швидко переходити між файлами, функціями та класами, та підтримує швидкий перехід до визначення, оголошення та використання змінних та функцій.
  - ~ Перейменування: якщо потрібно перейменувати змінну, функцію або клас, Visual Assist автоматично оновить всі посилання на цей елемент у вашому коді, що допомагає уникнути помилок при перейменуванні та зберегти консистентність коду.
- Попередня версія компілятора C++ на основі CLANG для Win64, тобто підтримка новіших функцій стандарту C++ та краща інтеграція з зовнішніми бібліотеками та кодом C:
  - ~ Компілятор на основі CLANG дозволяє використовувати сучасні функції та можливості, які входять до стандарту C++, використовуючи синтаксис C++11, C++14, C++17 та навіть C++20 для покращення продуктивності та зручності розробки.
  - ~ Компілятор CLANG дозволяє легше інтегрувати зовнішні бібліотеки, написані на C або C++, використовуючи функції та класи з існуючих бібліотек, що спрощує роботу зі стороннім кодом [5,6].
- доповнення до мови Delphi новими функціями, такі як багаторядкові літерали, для покращення зручності роботи розробників, наприклад:

```
var
  MyLongText: string;
begin
  MyLongText := 'Це багаторядковий літерал.' + sLineBreak +
                'Він дозволяє вам вводити' + sLineBreak +
                'текст на кількох рядках без зайвого коду.';
end;
```

– підтримка Skia у FireMonkey продуктивністю та якістю рендерингу графіки та елементів керування інтерфейсу користувача на всіх цільових платформах; Основні можливості Skia4Delphi включають:

  ~ Canvas 2D та розміщення тексту і дозволяє створювати складні графічні об’єкти та відображати текст на різних платформах, зручний та привабливий інтерфейс користувача.

  ~ Прискорений рендеринг за допомогою GPU. Skia4Delphi використовує апаратне прискорення для рендерингу графіки, що забезпечує високу продуктивність, і GPU для прискорення відображення зображень та графічних ефектів.

  ~ Підтримка різних форматів зображень *.bmp, *.gif, *.ico, *.jpg, *.png, *.wbmp, *.webp та *.raw. Дозволяє завантажувати та відображати зображення з різних джерел.

  ~ Заміна рендерингу в FireMonkey, використовучи Skia4Delphi, що дозволяє автоматично покращити якість рендерингу та продуктивність вашого додатка [8,14,19].

– вдосконалений MDI (Multiple Document Interface) та нова архітектура інтерфейсу користувача з вкладками для VCL, що допомагає модернізувати існуючі програми VCL з мінімальними зусиллями;

– новий компонент FireDAC "Запит за прикладом" (QBE) спрощує фільтрацію даних;

– помічник JSON для Delphi додає зіставлення даних у популярному форматі JSON з об'єктами, як це зроблено для XML;

багато покращень продуктивності та якості середовища IDE: збільшення швидкості роботи та зручності користування RAD Studio [13, 22, 26].

RAD Studio 12 має складові частини, які допомагають розробникам створювати програмне забезпечення:

- Дизайнер Форм (Form Designer) − дозволяє вам візуально створювати та налаштовувати форми вашого додатка, перетягувати та розміщувати компоненти на формі, налаштовувати їх властивості та створювати події.
- Вікно Редактора Вихідного Тексту (Editor Window), де пишеться код додатка, редагування та збереження.
- Палітра Компонент (Component Palette) − панель, на якій розташовані всі доступні компоненти, вибирати та перетягувати компоненти на форму для створення функціональності вашого додатка.
- Інспектор Об’єктів (Object Inspector) − вікно, де можна переглядати та редагувати властивості об’єктів на вашій формі, налаштовувати параметри компонентів та об’єктів.
- Довідник (On-line help) − ресурс, який надає вам інформацію про використання RAD Studio 12, де можна знайти відповіді на свої питання та дізнатися більше про функціональність та можливості [10, 21, 28].

1.11 Висновок до розділу

У першому розділі "Огляд сучасних методів у відповідній області та порівняльний аналіз функціональності існуючих систем" полягає в тому, що автоматизовані системи управління навчальними закладами, такі як "Деканат" для вищих навчальних закладів та "Школа" для загальноосвітніх закладів, є важливими інструментами для підтримки та оптимізації навчального процесу. Було виявлено, що існуючі системи мають різні технічні та функціональні характеристики, які впливають на їх ефективність та вартість впровадження та експлуатації.

Під час аналізу було виявлено, що система "Деканат" надає можливість ефективного керування структурою навчального процесу у вищих навчальних

закладах, включаючи облік навантаження кафедр, дані викладачів та студентів, а також розклад занять. У той час як система "Школа" забезпечує автоматизацію ключових процесів у загальноосвітніх закладах, таких як навчально-виховна та адміністративна діяльність, діловодство та формування звітності [8, 25, 29].

Обидві системи мають свої переваги, такі як унікальна технологія захисту інформації, онлайн інформування стейкхолдерів та інтеграція з іншими інформаційними сервісами.

Технічний аналіз показав, що вибір технологічного стеку відіграє важливу роль у розробці системи обліку студентів. Він впливає на швидкість розробки, здатність до масштабування та інтеграції з іншими системами.

Аналіз функціональних можливостей показав, що системи обліку студентів повинні бути гнучкими, щоб задовольняти різні потреби освітніх установ. Вони повинні надавати можливість відстежувати прогрес студентів, управляти навчальними планами та розкладами, а також забезпечувати зв’язок між студентами, викладачами та адміністрацією.

Процес створення системи обліку студентів включає створення проєкту архітектури, вибір технологічного стеку, розробку та тестування системи. Оцінка працездатності та готовності системи до впровадження є важливим етапом, який забезпечує її надійність та ефективність.

Аналіз та оцінка вартості впровадження та експлуатації систем обліку студентів допомагає визначити економічну ефективність впровадження таких систем. Вибір найбільш підходящих або перспективних варіантів для подальшого дослідження та порівняння допомагає визначити найкращі стратегії для розробки та впровадження систем обліку студентів.

Розглянуто функції RAD Studio 12, який є потужним інтегрованим середовищем розробки та надає розробникам широкі можливості для створення додатків для різних платформ, включаючи Windows, macOS, iOS, Android, Linux та інші. Підтримка різних хмарних платформ дозволяє розробляти та розгортати програми для відомих хмарних сервісів, таких як Amazon Web Services, Microsoft Azure та Google Cloud Platform. Покращена продуктивність та якість розробки

забезпечується за допомогою інструментів, таких як Visual Assist для C++ та SKIA для створення графічних ефектів. Зручність роботи з даними підтримується за допомогою FireDAC для роботи з базами даних та помічника JSON для зручної роботи з даними у форматі JSON [16, 27, 30].

Отже, обидва типи програмних рішень, як автоматизовані системи управління навчальними закладами, так і середовище розробки RAD Studio 12, є важливими інструментами, які сприяють підвищенню ефективності та якості освітнього процесу, а також полегшують роботу розробників у галузі програмування. А таж обрано для розробки програмного забезпечення до теми дипломної роботи "Розробка програмного забезпечення «Система обліку студентів групи»"[19, 21, 30]. В цілому, цей розділ підкреслює важливість глибокого розуміння сучасних методів у відповідній області та аналізу функціональності існуючих систем для розробки ефективних та економічно виправданих систем обліку студентів.

2 ПРОЄКТУВАННЯ ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Постановка задачі на розробку програмного продукту

Аналіз технічного завдання показує, що система обліку студентів повинно виконувати збір і угрупування інформації про даних студентів, в якій групі навчається, адреса, пільги, актив групи і т.д. У результаті повинен вийти база даних, яка б дозволяла зберігати, обробляти, автоматизувати і змінювати інформацію для вищеописаної довідкової системи. База даних повинна мати зручний, легкий і доступний для сприйняття інтерфейс користувача.

Згідно технічного завдання для представлення інформації про студентів групи навчального закладу у списку-результаті пошукового запиту повинна використовуватись наступна інформація (тільки істотна, а не вся можлива): прізвище.

Таким чином, кожний рядок із даними про студента повинні бути представлені у списку записом, який складається з чотирьох текстових та одного логічного полів. Детальна інформація про студента повинна представляти собою прізвище, ім'я, група, адреса, телефон і виводитись в окремий файл табличного процесора по відсортованому списку [8, 19, 29].

Детальна інформація про студента повинна представляти собою:

– Студенти (прізвище, ім'я, група, адреса, телефон);
– Актив групи (прізвище, ім'я, відповідальний за роботу, адреса, телефон);
– Соціальний паспорт групи (прізвище, ім'я, пільги, адреса, телефон);
– Облік користувачів (логін, пароль, доступ).

Аналіз існуючих систем для роботи у куратора групи та аналіз технічного завдання показують, що система обліку студентів, яка розробляється, матиме об’єм не менше 1 000 рядків програмного коду. При розробці даного продукту такого об’єму доцільно використовувати об'єктно-орієнтований підхід, як найпрогресивніший на сьогоднішній день. Також аналіз показує, що система

повинна легко розширюватись, бути легко масштабованим і виконуватись на різноманітних комп’ютерах.

У відповідності з технічним завданням (пункт 1.2.3), в якості мови програмування обрана мова Object Pascal, а розробка буде проводитись у середовищі RAD Studio 12 (DELPHI 12).

Ядром реалізованого в DELPHI механізму доступу до даних є процесор баз даних Borland Database Engine (BDE).

Архітектура BDE заснована на драйвера, які забезпечують обмін даний з коректним СУБД. Ядро процесора БД становить сукупність динамічних бібліотек, що містять механізми обміну даними, управління запитами, підтримки національних мов і т.д.

У складі RAD Studio 12 є ряд корисних допоміжних програм, які полегшують проєктування додатків для роботи з базами даних [5, 20, 24].

Таким чином, буде використовуватись ліцензійне, якісне, сумісне і повністю безкоштовне програмне забезпечення, що є суттєвим для навчального закладу.

Проєктування програмного ресурсу буде виконано з використанням реляційної бази даних системи, яка складається з чотирьох таблиць і зв'язку між таблицями - один до багатьох, а також - методу "Сутність-зв'язок".

2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Існує декілька етапів проведення нормалізації відношень за методом сутність-зв’язок [19, 17, 25].

Перший етап – визначення сутностей. Для вирішення завдання нормалізації відношень необхідно для початку визначити усі необхідні сутності (сутність – деякий об’єкт, що представляє інтерес для навчального закладу): студент, актив групи, соціальний паспорт. Кожна із вказаних сутностей повинна мати особливий ідентифікатор, не подібний із іншими сутностями.

Другий етап – визначення зв’язків між сутностями. Зв’язок – з’єднання між

двома або більше сутностями. Зазвичай – це дієслово. У даному випадку:

список_ містить _ студент;

список _ містить _ актив групи;

список _ містить _ соціальний паспорт;

список _ містить _ облік користувачів.

Третій етап – визначення атрибутів та ключів сутностей.

Атрибут – властивість сутності, мінімальний набір атрибутів. У нашому випадку атрибутами для вказаних сутностей будуть:

– Студенти (прізвище, ім'я, група, адреса, телефон);
– Актив групи (прізвище, ім'я, відповідальний за роботу, адреса, телефон);
– Соціальний паспорт групи (прізвище, ім'я, пільги, адреса, телефон);
– Облік користувачів (логін, пароль, доступ).

Наступним етапом в проєктуванні бази даних є визначення ступенів зв’язків між сутностями. Необхідно зазначити, що ступені зв’язків визначаються для бази даних під час всього періоду її існування. Визначення зв`язків представлено у вигляді ER-діаграм. Загальний підхід до побудови бази даних з використанням ER-методу полягає в побудові діаграми ER-типу, що включає в себе усі сутності та зв'язки важливі з точки зору інтересів організації.

Визначення ступеню зв`язку та класу належності сутностей «список» та «студент» показано на рисунку 2.1.

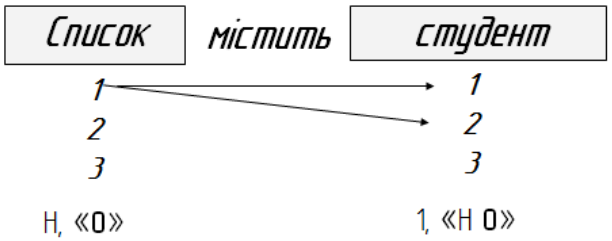


Рисунок 2.1– Визначення ступеню зв'язку та класу належності сутностей "список" та "студент"

Отже, ступінь зв’язку між сутностями 1:N, клас належності "О":"НО".

Проаналізувавши предметну область, ми виділили чотири досить рідко змінюючих сутності, три часто змінюючих та одну допоміжну, а також встановили зв’язки між ними та зображено більш детально у додатку 1.

2.3 Розробка логічної структури

Якісну програмну систему можна отримати за умови якісно виконаних аналізу і проєктування. Більшість існуючих методів об'єктно-орієнтованого аналізу і проєктування (ООАП) включають як мова моделювання, так і опис процесу моделювання. Мова моделювання − це нотація (в основному графічна), яка використовується методом для опису проєктів. Нотація являє собою сукупність графічних об'єктів, які використовуються в моделях; вона є синтаксисом мови моделювання. Наприклад, нотація діаграми класів визначає, яким чином подаються такі елементи і поняття, як клас, асоціація і множинність. Процес − це опис кроків, які необхідно виконати при розробці проєкту.

Уніфікована мова моделювання UML (Unified Modeling Language) − це наступник того покоління методів ООАП, які з'явилися в кінці 80-х і початку 90-х рр. UML є прямим об'єднанням і уніфікацією методів Буча, Рамбо і Якобсона, однак доповнює їх новими можливостями. Головними в розробці UML були наступні цілі:

- надати користувачам готовий до використання виразну мову візуального моделювання, що дозволяє розробляти осмислені моделі та обмінюватися ними;
- передбачити механізми розширюваності і спеціалізації для розширення базових концепцій;
- забезпечити незалежність від конкретних мов програмування і процесів розробки;
- забезпечити формальну основу для розуміння цієї мови моделювання;

(Мова повинна бути одночасно точним і доступним для розуміння, без зайвого формалізму);

Певний вплив одного об'єкта на інший з метою викликати відповідну реакцію називається операцією. В об'єктних і об'єктно-орієнтованих мовах операції, що виконуються над даним об'єктом, називаються методами і є складовою частиною визначення класу [4].

Почнемо проєктування з розробки моделі логіки предметної області, що розглядає процеси високого, загального рівня, без деталізації нюансів реалізації. Виділимо окремо дані, які використовуються в системі, процеси, кадрового складу та правила, які обумовлюють виконання процесів.

До даних відносимо:

- Студенти (прізвище, ім'я, група, адреса, телефон);
- Актив групи (прізвище, ім'я, відповідальний за роботу, адреса, телефон);
- Соціальний паспорт групи (прізвище, ім'я, пільги, адреса, телефон);
- Облік користувачів (логін, пароль, доступ).

Всі ці дані зберігаються у БД.

Процес функціонування програмного забезпечення формулюємо наступним чином:

1. працівник відділу кадрів вносить самостійно дані по працівникам навчального закладу відповідно до форм.
2. при потребі працівник відділу кадрів знаходить потрібну інформацію про працівника.
3. відділ кадрів перевіряє дійсність даних і при зміні даних працівника, наприклад, походження підвищення кваліфікації, проводиться оновлення даних.

З цього процесу видно, які саме функціональні вимоги має реалізувати система. Це:

- ведення реєстру студентів групи навчального закладу: включає в себе додавання осіб до списку, зміна параметрів студента, видалення студенту із

списку, сортування студента; при цьому пошук має здійснюватися за сортування повинно здійснюватися за групою від меншого числа до більшого, за введеним прізвищем, а потім є можливість експортувати в табличний процесор**;**

– ведення бази даних.

Розглянемо загальний принцип роботи програми. Програма складається з головного вікна SustemaProject та допоміжних діалогових вікон. Головне вікно містить заголовок, головне меню, панель інструментів, командні кнопки та елемент відображення таблиці бази даних з інформацією про студентів. Далі користувач, використовуючи головне меню, панель інструментів та командні кнопки, здійснює обробку бази даних SchoolDataBase. При цьому, при необхідності, відображаються допоміжні діалогові вікна:

– StudentForm – діалогове вікно додавання та редагування елемента списку студентів;
– DataModule1 – вікно відповідає за базу даних;
– PersonalForm – діалогове вікно відображення списку активу групи;
– PrepodForm – діалогове вікно відображення соціального паспорту студентів;
– UserForm – діалогове вікно вхід до системи;
– UsersForm – діалогове вікно обліку користувачів.

Структурна схема програмного продукту зображена на рисунку 2.2.

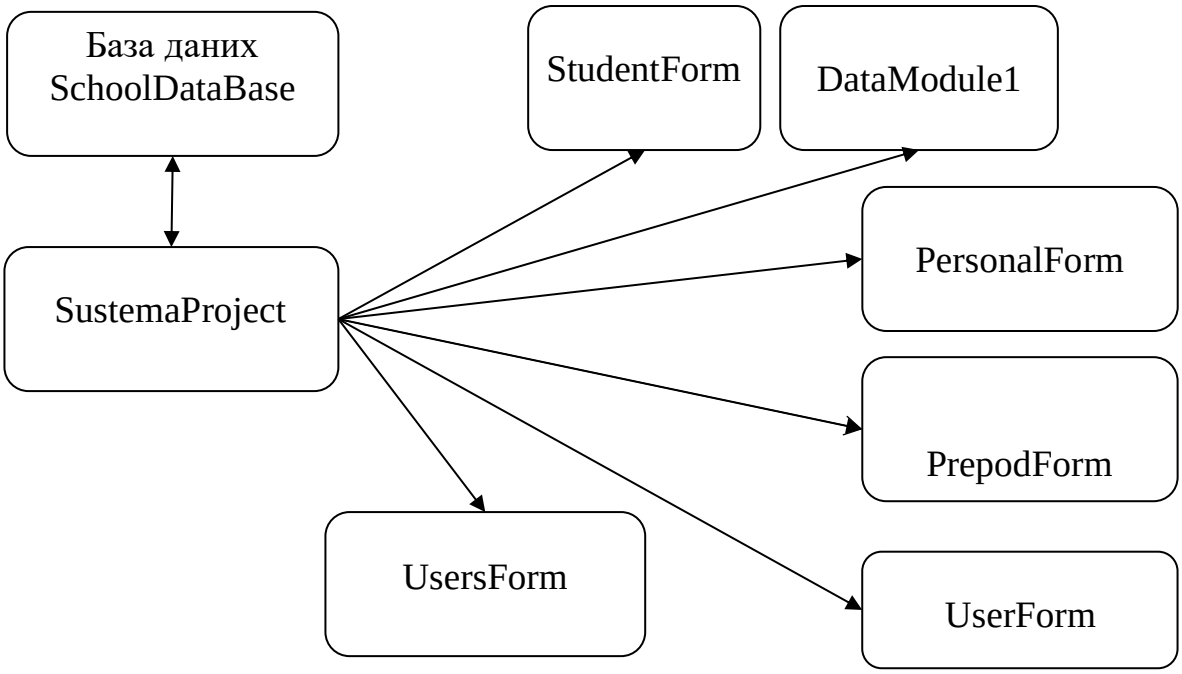


Рисунок 2.2 – Структурна схема програмної системи

2.4 Розробка фізичної структури

Фізична структура представляє собою розміщення файлів (фізичних одиниць) по піддиректоріях, в яких розташоване розроблювана програмна система [5].

Розробка діаграми роботи фізичної структури програмної системи здійснюється на основі аналізу логічної структури дипломної роботи зображена у додатку 3.

Спираючись на отримані результати, перейдемо до складання фізичної моделі бази даних програмної системи. Для цього деталізуємо інформацію, яка буде зберігатися у таблицях бази даних, визначивши точний перелік полів та їх типи (див. таблиці 2.1− 2.7).

Діалогове вікно системи обліку студентів групи складається з таких елементів:

Label1 – об’єкт, який описує назву поля "Прізвище";

FilterFamEdit – об’єкт, який вписується назва даних фільтрації;

StudentNavigator – об’єкт, який відповідає за навігацію по списку;

MainMenu1 – об’єкт, який служить для програми як меню;

DBGrid1 – об’єкт, який має параметри наведені в таблиці 2.1.

Таблиця 2.1 – Інформація про студента "Список" (DBGrid1)

Назва поля	Тип даних	Пояснення
Прізвище	Var char(250)	зміна/введення текстових даних про прізвище, первісний ключ
Ім’я	Var char(250)	зміна/введення текстових даних про ім’я
Група	Var char(250)	зміна/введення текстових даних про групу
Адреса	Var char(250)	зміна/введення текстових даних про адресу
Телефон	int, pk	введення цифрових даних про номер телефону

Діалогове вікно "Соціальний паспорт" складається з таких елементів:

Label1 – об’єкт, який описує назву поля "Прізвище";

FilterFamEdit – об’єкт, який вписується назва даних фільтрації;

StudentNavigator – об’єкт, який відповідає за навігацію по списку;

MainMenu1 – об’єкт, який служить для програми як меню;

DBGrid1 – об’єкт, який має параметри наведені в таблиці 2.2:

Таблиця 2.2 – Інформація "Соціальний паспорт" (DBGrid1)

Назва поля	Тип даних	Пояснення
Прізвище	Var char(250)	зміна/введення текстових даних про прізвище, первісний ключ
Ім’я	Var char(250)	зміна/введення текстових даних про ім’я
Пільги	Var char(250)	зміна/введення текстових даних про пільги
Адреса	Var char(250)	зміна/введення текстових даних про адресу
Телефон	int, pk	введення цифрових даних про номер телефону

Діалогове вікно "Актив групи" складається з таких елементів:

Label1 – об’єкт, який описує назву поля "Прізвище";

FilterFamEdit – об’єкт, який вписується назва даних фільтрації;

StudentNavigator – об’єкт, який відповідає за навігацію по списку;

MainMenu1 – об’єкт, який служить для програми як меню;

DBGrid1 – об’єкт, який має параметри наведені в таблиці 2.3.

Таблиця 2.3 – Інформація "Актив групи" (DBGrid1)

Назва поля	Тип даних	Пояснення
Прізвище	Var char(250)	зміна/введення текстових даних про прізвище, первісний ключ
Відповідає за роботу	Var char(250)	зміна/введення текстових даних про відповідальність за роботу
Група	Var char(250)	зміна/введення текстових даних про групу
Адреса	Var char(250)	зміна/введення текстових даних про адресу
Телефон	int, pk	введення цифрових даних про номер телефону

Діалогове вікно "Вхід в систему обліку студентів групи" складається з таких елементів:

Label1 – об’єкт, в якому записано назву поля "Логін";

Label2 – об’єкт, в якому записано назву поля "Пароль";

LoginEdit – об’єкт, в якому вводиться логін;

PasswordEdit – об’єкт, в якому вводиться пароль;

Button3 – об’єкт, в якому записано назву поля "?";

Button1 – об’єкт, в якому записано назву поля "Вхід";

Button2 – об’єкт, в якому записано назву поля "Відмінити".

Діалогове вікно "Облік користувачів" складається з таких елементів:

Label1 – об’єкт, який описує назву поля "Логін";

LoginEdit – об’єкт, в якому вводиться логін, який вписується назва даних фільтрації;

DBNavigator1 – об’єкт, який відповідає за навігацію по списку;

DBGrid1 – об’єкт, який має параметри наведені в таблиці 2.4.

Таблиця 2.4 – Інформація "Облік користувачів" (DBGrid1)

Назва поля	Тип даних	Пояснення
Логін	Var char(250)	зміна/введення текстових даних про логін, первісний ключ
Пароль	Var char(250)	введення цифрових даних
Доступ	Var char(250)	зміна/введення текстових даних про доступ

Виходячи з цієї структури інформації, за допомогою програмного середовища RAD Studio 12 буде згенеровано лістинг бази даних.

Вхідні дані, які не зберігаються в даній системі і призначені для формування певних вибірок вихідних даних, є пошуковими запитами користувачів.

Користувачі можуть здійснювати пошук за наступними даними:

пошук студентів за прізвищем або групою. Пошук виконується адміністратором або куратором. Вихідними даними є перелік знайдених студентів (або відсутність даних у випадку, коли в БД немає відповідних студентів), сформований у вигляді таблиці з полями "Прізвище", "Ім’я", "Група", "Адреса", "Телефон";

пошук або сортування студента за прізвищем або групою;

експорт у електронну таблицю.

При цьому реалізація перерахованого функціоналу потребує лише розширення рівнів застосувань, без необхідності будь-яких змін на рівні бази даних.

2.5 Ефективність та оптимізація програмного коду

Блок коду виводить інформацію про кількість записів у таблиці PrepodTable на екран користувача. Блок коду є обробником події Click для об'єкта, що має ім'я

N7 (лістинг 2.1), який, ймовірно, є об'єктом кнопки або іншого елемента інтерфейсу. У цьому блоку коду виконується наступне:

Змінні ***s*** та і ініціалізуються як цілі числа.

Змінна ***s*** встановлюється в 0.

Запускається цикл ***for***, який ітерується від 0 до кількості записів у таблиці PrepodTable, що знаходиться у DataModule1, з кожним проходженням циклу збільшуючи значення ***s*** на 1.

Після циклу виводиться повідомлення (за допомогою ShowMessage), яке містить назву таблиці (Datamodule1.PrepodTable.TableName) та кількість записів у ній (s).

Лістинг 2.1 – Блок коду виведення інформації про кількість записів у таблиці

```
procedure TPrepodForm.N7Click(Sender: TObject);
var
s:integer;
i:integer;
begin
s:=0;
for i:=0 to DataModule1.PrepodTable.RecordCount-1 do
 s:=s+1;
ShowMessage('Назва таблиці: '+Datamodule1.PrepodTable.TableName+#13+
'Кількість записів: '+IntToStr(s));
end;
```

Блок коду є процедурою FormCreate, яка викликається при створенні форми TPrepodForm (лістинг 2.2), та включає такі дії:

- Перевіряє, чи змінна ***login*** має значення 'admin'. Якщо так, то зроблено кнопку або елемент інтерфейсу N9 видимим.
- Якщо змінна ***login*** не має значення 'admin', то кнопка або елемент інтерфейсу N9 стає невидимим.
- Перевіряє, чи змінна Dostup має значення false. Якщо так, то виконується блок коду, який вимикає можливість редагування DBGrid1 (імовірно, це елемент інтерфейсу для відображення даних у табличному вигляді).
- Встановлюється стиль шрифту для DBGrid1 на жирний (fsBold).

Отже, цей блок коду визначає видимість елементів інтерфейсу на формі TPrepodForm залежно від значень змінних login і Dostup та змінює стиль шрифту для DBGrid1.

Лістинг 2.2 – Блок перевірки логіну і паролю.

```
procedure TPrepodForm.FormCreate(Sender: TObject);
begin
if login='admin' then N9.Visible:=true
  else n9.Visible:=false;
   if Dostup=false then begin
DBGrid1.Enabled:=false;
PrepodNavigator.VisibleButtons:=[nbFirst,nbNext,nbPrior,nbLast,nbRefresh];
DbGrid1.Font.Style:=[fsBold];
end;
end;
```

У фрагменті лістингу виконується робота з об'єктом Excel, який, ймовірно, був створений раніше як XLApp, що тут відбувається:

1. Вибирається робоча книга (робочий документ) з першого індексу (Workbooks[1]), що ймовірно є першою відкритою книгою.
2. З робочого документа вибирається робочий аркуш, що має назву 'Група-студенти'.
3. Здійснюється запис тексту у конкретні клітинки аркуша за допомогою властивості Cells.
   - На першому рядку (рядок 1) та другому стовпці (стовпець 2) встановлюється значення "Коледж".
   - На другому рядку (рядок 2) та першому, другому, третьому, четвертому та п'ятому стовпці встановлюються відповідно значення "Прізвище", "Ім'я", "Група", "Адреса" і "Телефон".

Отже, цей код додає заголовки стовпців у таблицю Excel на робочому аркуші 'Група-студенти'.

Лістинг 2.3 – Додавання заголовків стовпців у таблицю

```
Sheet:=XLApp.Workbooks[1].WorkSheets['Група-студенти'];
Sheet.Cells[1,2]:='Коледж';
Sheet.Cells[2,1]:='Прізвище';
Sheet.Cells[2,2]:='Ім"я';
Sheet.Cells[2,3]:='Група';
```

```
Sheet.Cells[2,4]:='Адреса';
Sheet.Cells[2,5]:='Телефон';
```

2.6 Проєктування інтерфейсу

Структура відповідності логічної і фізичної структури зображена в додатку 3 дипломної роботи.

Після запуску системи на екрані з’являється вікно входу в систему (рис. 2.3).

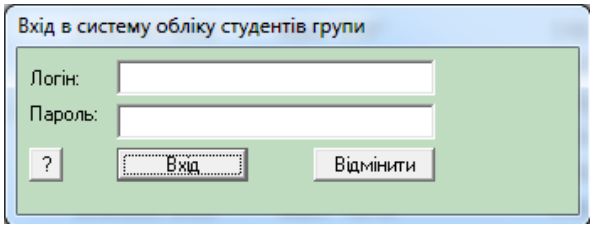


Рисунок 2.3 – Вікно входу в систему обліку студентів групи

Після завантаження системи обліку студентів групи з’являється діалогове вікно, в якому необхідно ввести пароль і логін користувачів "Адміністратора" або "Куратора".

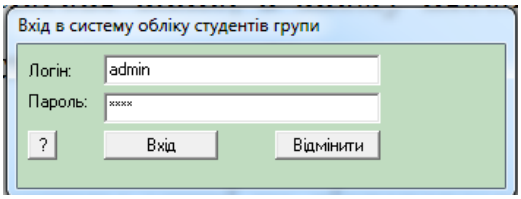


Рисунок 2.4 – Вікно входу в систему обліку студентів групи» під користувачем "admin"

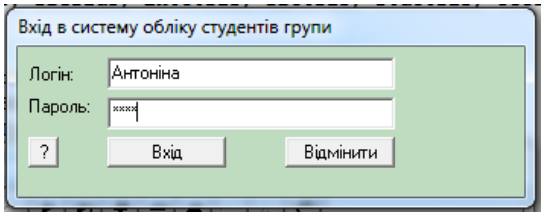


Рисунок 2.5 – Вікно входу в систему обліку студентів групи під користувачем "Антоніна"

Після введені паролю і логіну у діалоговому вікні, яке завантажилось і має назву "Система обліку студентів групи", на екрані з'являється головне вікно програми (рис. 2.6).

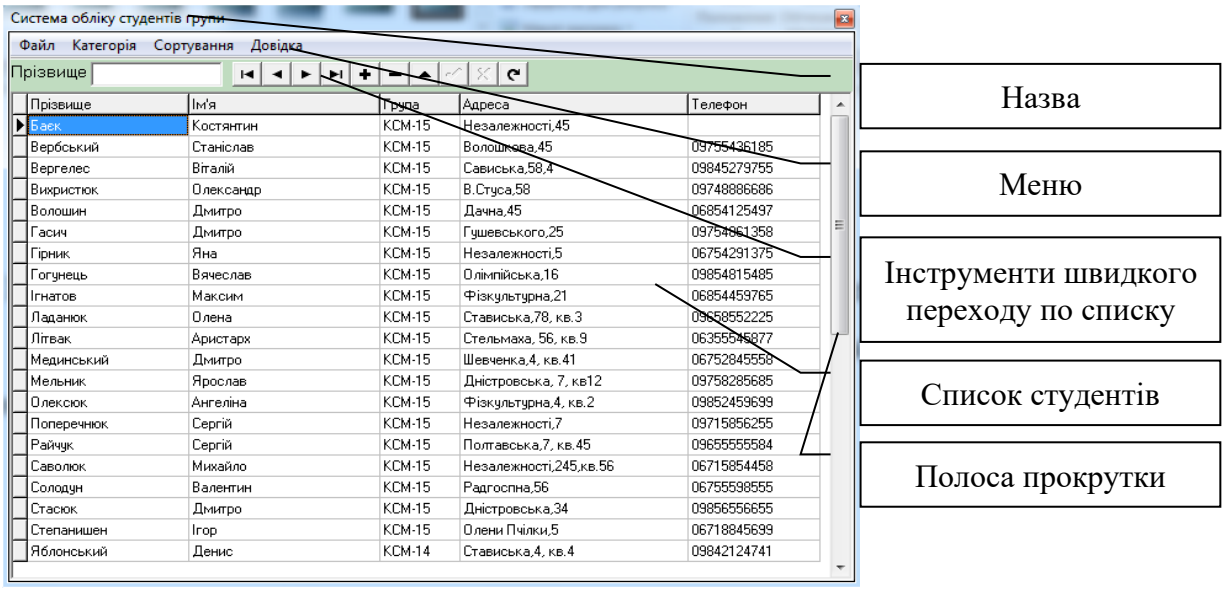


Рисунок 2.6 – Інтерфейс системи обліку студентів групи

У верхній частині вікна знаходиться головне меню.

Нижче знаходиться панель інструментів, на які необхідно для швидкого доступу переходу до наступних осіб у списку (рис. 2.6). Всю іншу частину вікна займає область, у якій відображається інформація про студентів групи навчального закладу у вигляді таблиці.

2.7 Висновок до розділу

У другому розділі «Проєктування технічного та робочого проєкту» було детально розглянуто процес проєктування технічного та робочого проєкту програмного продукту:

Визначення основних цілей та вимог до продукту, включаючи функціональність, продуктивність, надійність та інші ключові характеристики.

Вибір оптимальної структури даних та методу їх організації для забезпечення ефективної обробки та зберігання даних.

Створення моделі системи, яка відображає основні компоненти та їх взаємодію, включаючи процеси, функції та потоки даних.

Детальна реалізація компонентів системи, включаючи алгоритми, структури даних, модулі та інтерфейси.

Проведення аналізу та оптимізації програмного коду для підвищення його продуктивності та ефективності.

Створення зручного та інтуїтивно зрозумілого інтерфейсу користувача, який забезпечує ефективну взаємодію користувача з системою.

Таким чином, у розділі було виконано важливу роботу по проєктуванню програмного продукту, що створює основу для подальшої його реалізації. Кожен з цих етапів відіграє важливу роль у процесі розробки програмного продукту і впливає на його якість та ефективність.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Встановлення програмного продукту

Програмний продукт знаходиться на диску у вигляді файлу запуску. Тому для його встановлення потрібно, щоб комп'ютер був обладнаний приводом для зчитування оптичних дисків або USB-порти.

Спочатку необхідно вставити диск у привод або копіювання через USB. На диску знаходиться виконуваний файл SustemaProject.exe, який необхідно запустити двічі клацнувши по ньому.

Якщо в процесі запуску програми на екран не було виведено повідомлення про помилку, то процес встановлення програми пройшов успішно.

3.2 Інструкція з експлуатації програмного продукту

Після запуску завантажувального файлу на екрані з'являється вікно входу в програмну систему, зображене на рисунку 3.1.

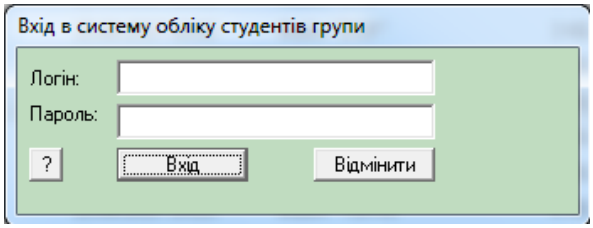
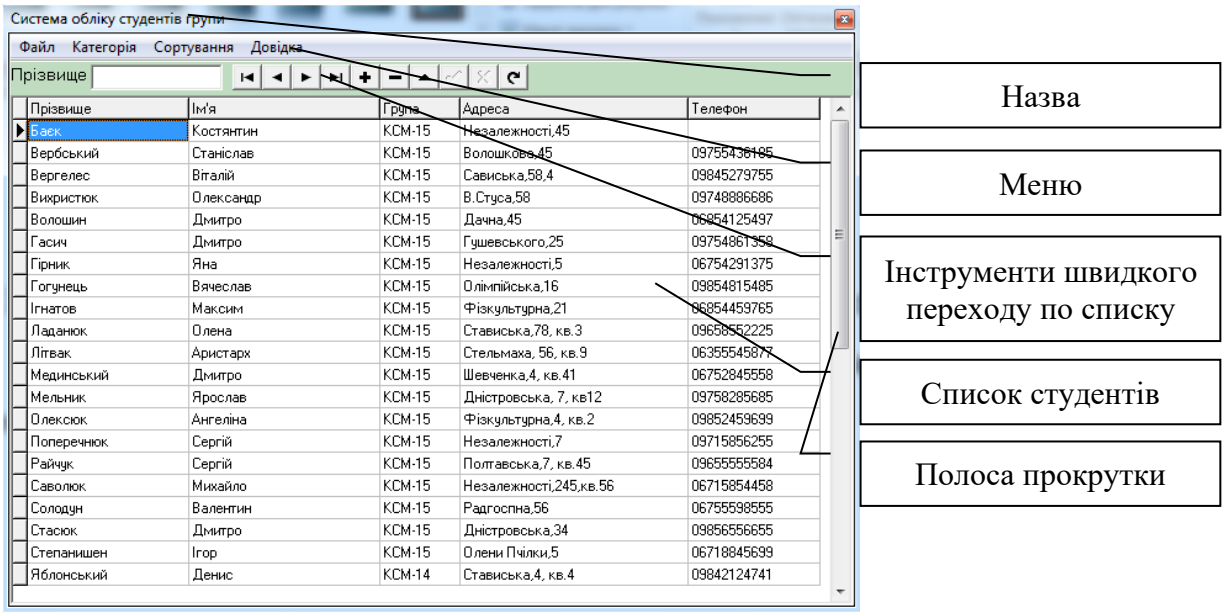


Рисунок 3.1 – Вікно входу в систему обліку студентів групи

Після завантаження систему обліку студентів групи з'являється діалогове вікно, в якому необхідно ввести пароль і логін користувачів "Адміністратора" або "Куратора".

Після введені паролю і логіну у діалоговому вікні, яке завантажилось і має назву "Система обліку студентів групи", на екрані з'являється головне вікно

програмної системи (рис. 3.2).



Прізвище	Ім'я	Група	Адреса	Телефон
Баєк	Костянтин	КСМ-15	Незалежності,45	
Вербський	Станіслав	КСМ-15	Волошкова,45	09755436185
Вергелес	Віталій	КСМ-15	Сависька,58,4	09845279755
Вихристюк	Олександр	КСМ-15	В.Стуса,58	09748886686
Волошин	Дмитро	КСМ-15	Дачна,45	06854125497
Гасич	Дмитро	КСМ-15	Гушевського,25	09754861358
Гірник	Яна	КСМ-15	Незалежності,5	06754291375
Гогунець	Вячеслав	КСМ-15	Олімпійська,16	09854815485
Ігнатов	Максим	КСМ-15	Фізкультурна,21	06854459765
Ладанюк	Олена	КСМ-15	Стависька,78, кв.3	09658552225
Літвак	Аристарх	КСМ-15	Стельмаха, 56, кв.9	06355545877
Мединський	Дмитро	КСМ-15	Шевченка,4, кв.41	06752845558
Мельник	Ярослав	КСМ-15	Дністровська, 7, кв12	09758285685
Олексюк	Ангеліна	КСМ-15	Фізкультурна,4, кв.2	09852459699
Поперечнюк	Сергій	КСМ-15	Незалежності,7	09715856255
Райчук	Сергій	КСМ-15	Полтавська,7, кв.45	09655555584
Саволюк	Михайло	КСМ-15	Незалежності,245,кв.56	06715854458
Солодун	Валентин	КСМ-15	Радгоспна,56	06755598555
Стасюк	Дмитро	КСМ-15	Дністровська,34	09856556655
Степанишен	Ігор	КСМ-15	Олени Пчілки,5	06718845699
Яблонський	Денис	КСМ-14	Стависька,4, кв.4	09842124741

Рисунок 3.2 – Інтерфейс системи обліку студентів групи

У верхній частині вікна знаходиться головне меню:

команда "Файл":

- Відкрити в Excel;
- Вихід;

команда "Категорія":

- Соціальний паспорт;
- Актив групи;

команда "Сортування":

- по прізвищу;
- по групі;

команда "Сортування":

- Довідка.

Після нажаті на кнопку "Відкрити в Excel" відобразиться вся інформація з діалогового вікна "Система обліку студентів групи" у табличному редакторі (рис.3.3).

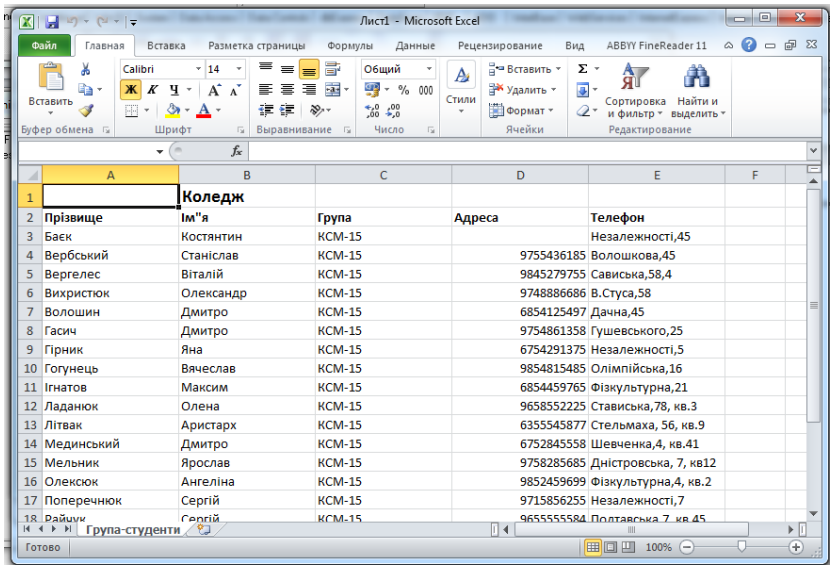


Рисунок. 3.3 – Відображення інформації з діалогового вікна системи обліку студентів групи

Нижче знаходиться панель інструментів, на які необхідно для швидкого доступу переходу до наступних осіб у списку (рис. 3.3). Всю іншу частину вікна займає область, у якій відображається інформація про студентів групи навчального закладу у вигляді таблиці. Лістинг головної сторінки програмного забезпечення відображено у додатку 4.

Діалогове вікно "Соціальний паспорт" дає можливість відобразити або змінити інформації про особливі дані студентів групи навчального закладу: прізвище, ім’я, пільги, адресу, телефон (рис. 3.4). Всі дані даної таблиці експортуються у табличний процесор. Лістинг діалогового вікна "Соціальний паспорт" відображено у додатку 5.

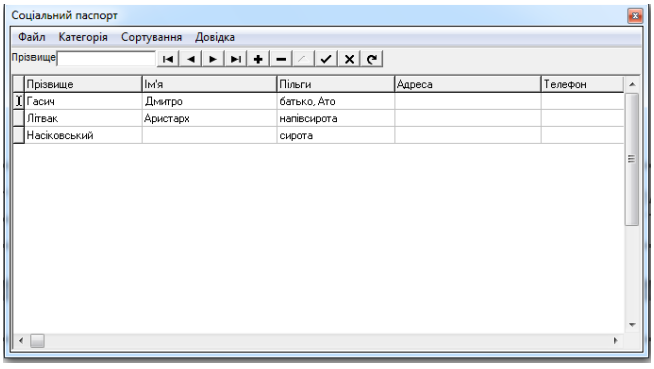


Рисунок. 3.4 – Інтерфейс діалогового вікна "Соціальний паспорт"

Діалогове вікно "Актив групи" дає можливість відобразити або змінити інформації про особливі дані студентів групи навчального закладу: прізвище, ім’я, відповідальність за роботу, адресу, телефон (рис. 3.5). Всі дані даної таблиці експортуються у табличний процесор. Лістинг діалогового вікна "Актив групи" відображено у додатку 6.

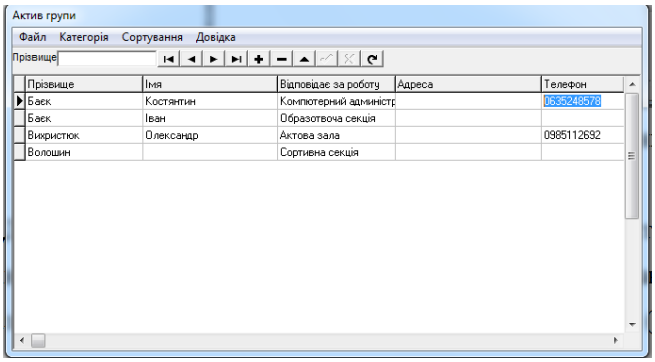


Рисунок 3.5 – Інтерфейс діалогового вікна "Актив групи"

3.3 Висновок до розділу

У третьому розділі "Тестування програмного забезпечення" було розглянуто важливий етап розробки програмного продукту системи обліку студентів – тестування.

Інструкції зі встановлення програмного продукту. Було розроблено детальні інструкції зі встановлення програмного продукту, що включають в себе кроки для коректного встановлення та налаштування програмного продукту. Важливий елемент, який забезпечує гладке впровадження програмного продукту та його правильну роботу.

Інструкція з експлуатації програмного продукту. Було створено інструкції з експлуатації, які допомагають користувачам ефективно використовувати програмний продукт. Інструкції включають в себе інформацію про основні функції та можливості продукту, а також поради щодо його використання.

Тестування є критично важливим етапом у процесі розробки програмного

продукту. Воно допомагає забезпечити, що продукт відповідає всім вимогам та працює належним чином. Інструкції зі встановлення та експлуатації є важливими компонентами, які допомагають забезпечити гладке впровадження та ефективне використання програмного продукту. Завдяки цьому розділу, користувачі зможуть легко встановити та використовувати програмний продукт, що, в свою чергу, підвищує його цінність та ефективність.

ВИСНОВКИ

У наш час існує безліч програмних додатків дозволяють забезпечувати якісне збереження і обробку інформації. Так для зберігання великого обсягу інформації, що стосується певної області дуже зручно користуватися системами управління базами даних (СКБД), або іншим програмним забезпеченням, яке дозволяє:

- надійно зберігати інформацію;
- змінювати (додавати, видаляти, оновлювати) інформацію;
- зменшити час доступу до необхідної інформації;
- реалізувати різні рівні доступу до інформації, розраховані на різних користувачів.

Останнім часом бази даних знаходять все більш широке застосування в нашому житті. Практично у всіх галузях економіки, промисловості, ринкових відносин використовуються бази даних, що дозволяють зберігати й обробляти інформацію.

Тому складність сучасної технології баз даних з'явилась результатом розвитку протягом декількох десятиліть способів обробки даних і керування інформацією. Обробка даних розвивалась від примітивних методів п’ятидесятих років до складних інтегрованих систем сьогоднішнього дня [3].

Основною функцією системи обліку студенті навчальних закладів оптимізація умов роботи кураторів груп та економії часу, який витрачався при ручній праці. І надає можливість отримання потрібної інформації про студентів групи, має функцію пошуку потрібного студента, сортування, визначити актив групи та мати соціальний паспорт групи, надає можливість додавання, видалення та редагування записів, що стосуються студентів.

В даному розділі спроєктовано базу даних за допомогою використанням з використанням реляційної бази даних системи, яка складається з чотирьох таблиць і зв'язку між таблицями - один до багатьох, а також - методу "Сутність-зв'язок".

Спроєктовано інтерфейс програмного забезпечення для кращого та ефективнішого сприйняття інформації, а також для можливості узагальнення,

систематизації даних по студентах групи.

Розробка системи обліку студенті навчальних закладів проводилась у середовищі RAD Studio 12.

Розглянуто функції RAD Studio 12, який є потужним інтегрованим середовищем розробки та надає розробникам широкі можливості для створення додатків для різних платформ, включаючи Windows, macOS, iOS, Android, Linux та інші. Підтримка різних хмарних платформ дозволяє розробляти та розгортати програми для відомих хмарних сервісів, таких як Amazon Web Services, Microsoft Azure та Google Cloud Platform. Покращена продуктивність та якість розробки забезпечується за допомогою інструментів, таких як Visual Assist для C++ та SKIA для створення графічних ефектів. Зручність роботи з даними підтримується за допомогою FireDAC для роботи з базами даних та помічника JSON для зручної роботи з даними у форматі JSON.

Результати дипломної роботи можуть бути представлені на розгляд керівництва навчальних закладів для впровадження розробленого програмного продукту в роботу куратора групи.

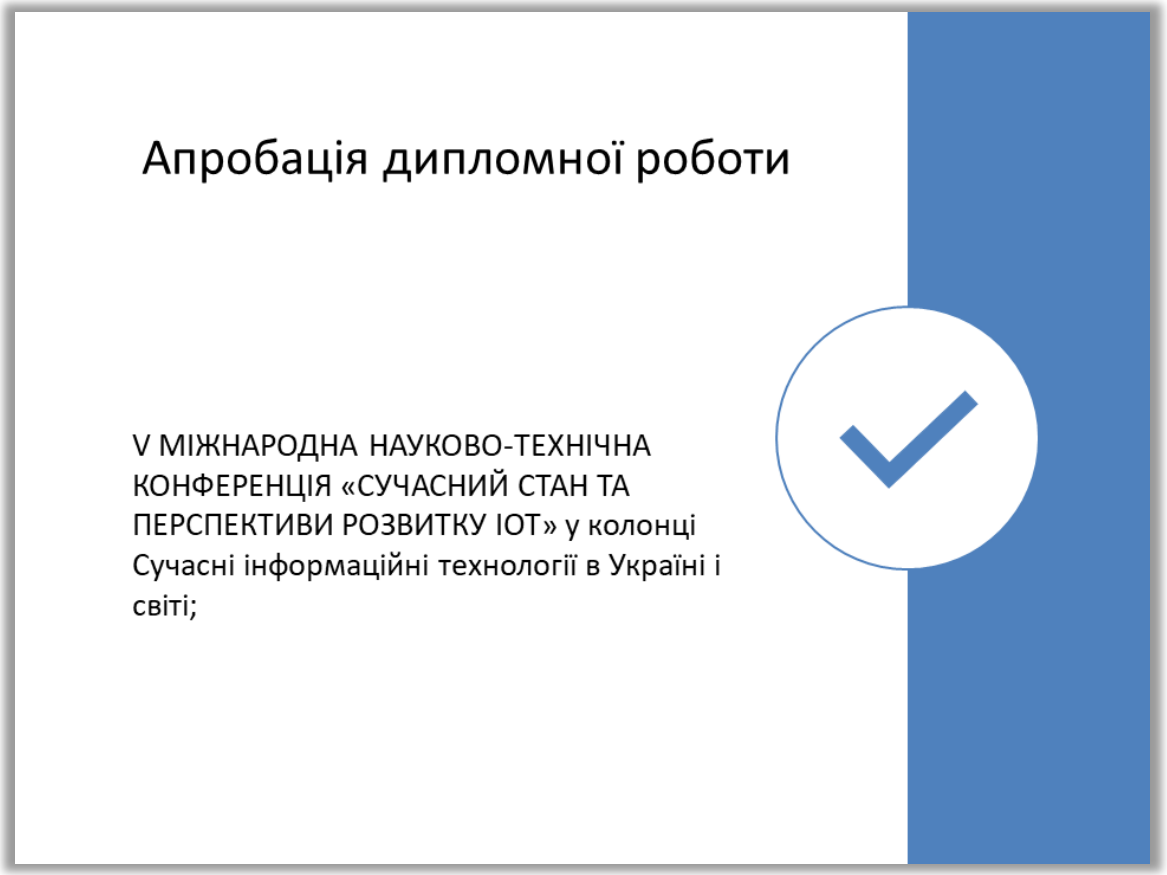
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Developing Cloud Applications. URL: https://docwiki.embarcadero.com/RADStudio/Athens/en/Developing_Cloud_Applications
2. RAD Studio Features - RAD Studio: Cross-Platform Database Solutions – Embarcadero Creating an iOS App Creating an iOS App - RAD Studio. URL: https://www.embarcadero.com/
3. RAD Studio Installation links (ISO and Web installers). URL: https://www.barnsten.com/rad-studio-installation-links-iso-and-webinstallers/
4. Автоматизовані системи управління навчальними закладами: Деканат. URL: http://www.politek-soft.kiev.ua/ru/index.php?do=products&product=deanery
5. Автоматизовані системи управління навчальними закладами: Школа. URL: https://school.osvita.net/
6. Актуальні проблеми та перспективи розвитку інформаційних систем в освіті / за ред. О. М. Гейченко. – К.: Інститут педагогіки НАПН України, 2018. – 248 с.
7. Аналіз сучасних інформаційних систем в освіті. URL: https://tech.vernadskyjournals.in.ua/journals/2020/1_2020/part_1/14.pdf
8. Анохін О. М. Інформаційні технології в управлінні навчальним закладом: теорія і практика / О. М. Анохін. – К.: Педагогічна думка, 2012. – 256 с.
9. Бачинський О. І. Інформаційні системи та технології в освіті: навч. посіб. / О. І. Бачинський, В. П. Коваленко. – К.: Видавничий центр «Академія», 2014. – 240 с.
10. Білоусов О. В. Інформаційні системи та технології в управлінні навчальним закладом: навч. посіб. / О. В. Білоусов. – К.: Літера ЛТД, 2016. – 320 с.
11. Блог Embarcadero. URL: https://blogs.embarcadero.com/es/
12. Використання RAD Studio для розробки програмного забезпечення для освітніх закладів. URL: https://lib.iitta.gov.ua/704093/1/statfree.pdf
13. Гейченко О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Гейченко. – К.: Інститут педагогіки НАПН України, 2014. – 352 с.

14. Єрьоменко О. М. Інформаційні системи та технології в управлінні освітою: навч. посіб. / О. М. Єрьоменко. – К.: Літера ЛТД, 2015. – 240 с.
15. Інформаційні системи в освіті. URL: https://core.ac.uk/download/pdf/147037367.pdf
16. Інформаційні системи в освіті: автоматизовані навчальні системи. URL: https://core.ac.uk/download/pdf/147037367.pdf
17. Інформаційні системи та технології в освіті: навч. посіб. / за ред. О. В. Бондар. – К.: Видавничий дім «Академія», 2017. – 448 с.
18. Капшук О. В. Інформаційні системи та технології в освіті: навч. посіб. / О. В. Капшук. – К.: Видавничий центр «Академія», 2013. – 208 с.
19. Коваленко В. П. Інформаційні системи та технології в освіті: навч. посіб. / В. П. Коваленко. – К.: Видавничий центр «Академія», 2012. – 224 с.
20. Кузьмінська О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Кузьмінська. – К.: Видавничий центр «Академія», 2015. – 288 с.
21. Мазур О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Мазур. – К.: Видавничий центр «Академія», 2014. – 256 с.
22. Пометун О. І. Інформаційні системи та технології в освіті: навч. посіб. / О. І. Пометун. – К.: Видавничий центр «Академія», 2013. – 240 с.
23. Програмне забезпечення для освітніх закладів. URL: https://elizlabs.com.ua/tekhn%D1%96chne-obladnannia-ua/programne-zabezpechennya
24. Розробка інформаційної системи обліку студентів. URL: https://eir.nuos.edu.ua/items/32df3771-e337-4dc2-82ad-dc3bab10d218
25. Розробка інформаційної системи обліку учнів позашкільних навчальних закладів. URL: https://eir.nuos.edu.ua/items/32df3771-e337-4dc2-82ad-dc3bab10d218
26. Сучасні підходи до розробки програмного забезпечення. URL: https://foxminded.ua/pidkhody-do-rozrobky-prohramnoho-zabezpechennia/
27. Сущенко О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Сущенко. – К.: Видавничий центр «Академія», 2015. – 240 с.

28. Управління навчальним закладом https://www.academia.edu/39903687/Управління_навчальним_закладом
29. Фурман О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Фурман. – К.: Видавничий центр «Академія», 2014. – 224 с.
30. Шевченко О. М. Інформаційні системи та технології в освіті: навч. посіб. / О. М. Шевченко. – К.: Видавничий центр «Академія», 2013. – 208 с.

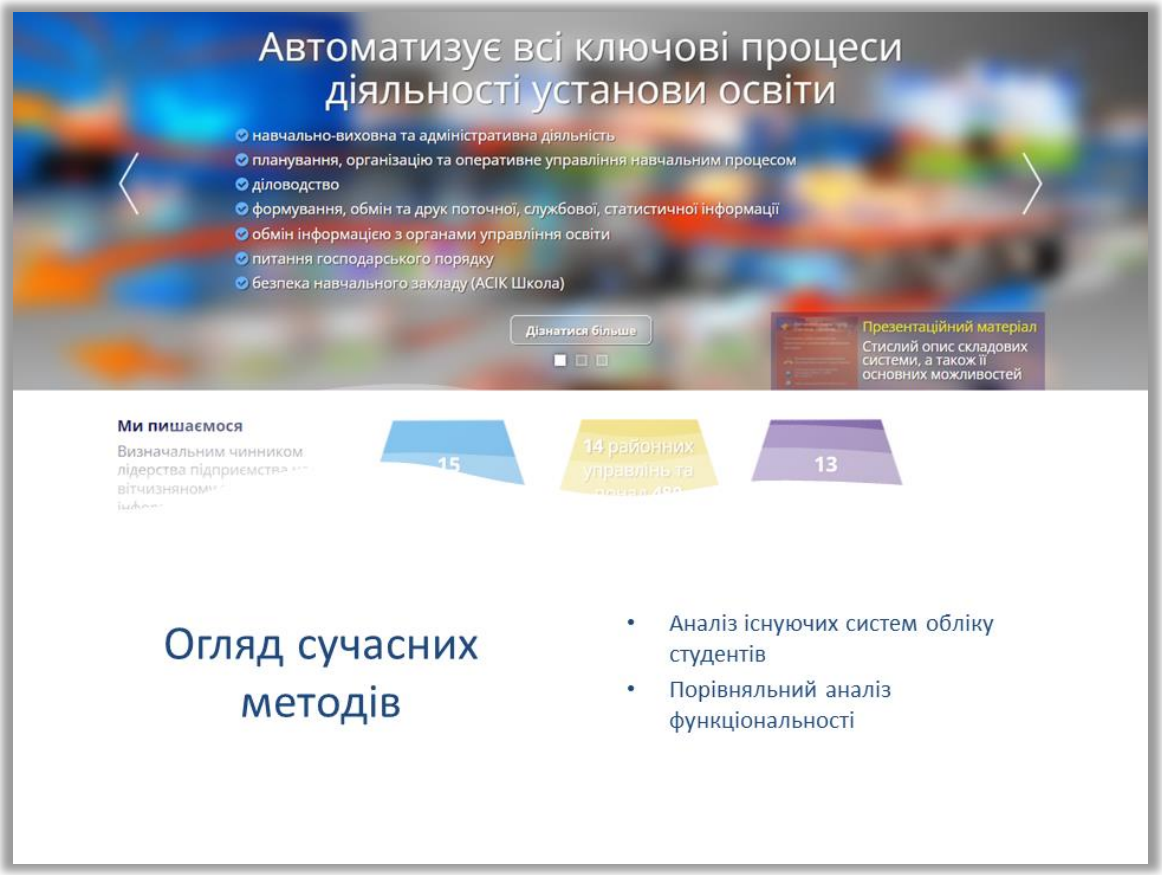
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ:



Вступ

Актуальність теми

Мета і завдання роботи



Огляд сучасних методів

- Аналіз існуючих систем обліку студентів
- Порівняльний аналіз функціональності

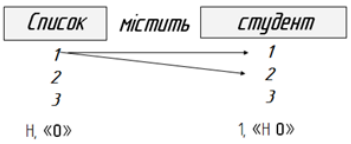
Визначення вимог до системи

- Ключові функціональні та нефункціональні вимоги
- Потреби користувачів: студенти, викладачі, адміністрація

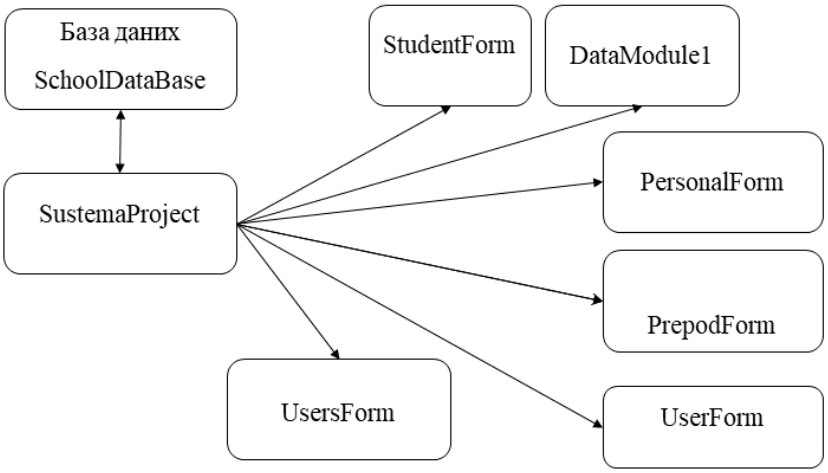
Вибір технологічного стеку

- Огляд мов програмування та фреймворків
- Вибір бази даних
- Інструменти для розробки веб-додатків

Проєктування архітектури системи

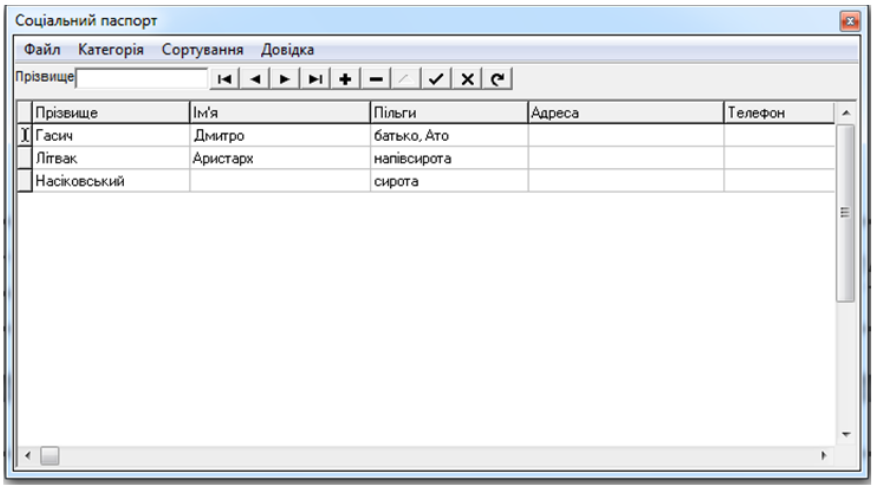


- Структура бази даних
- Діаграми класів і послідовностей
- Архітектурні шаблони



Розробка програмного забезпечення

- Етапи розробки: аналіз вимог, проєктування, реалізація, тестування
- Інтеграція компонентів системи



Тестування та впровадження

- Методи тестування: функціональне, інтеграційне, безпекове, навантажувальне
- Оцінка працездатності та готовності до впровадження

Оцінка вартості впровадження та експлуатації

- Витрати на розробку та впровадження
- Витрати на обслуговування та підтримку
- Інфраструктурні витрати

Висновки та рекомендації

Досягнуті результати

Перспективи подальших досліджень

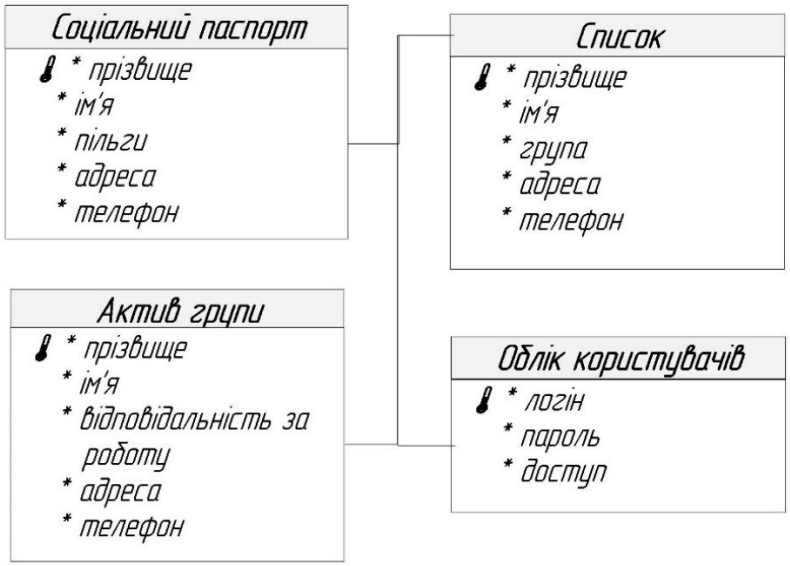
Дякую за увагу!

- Готовий відповісти на ваші питання.

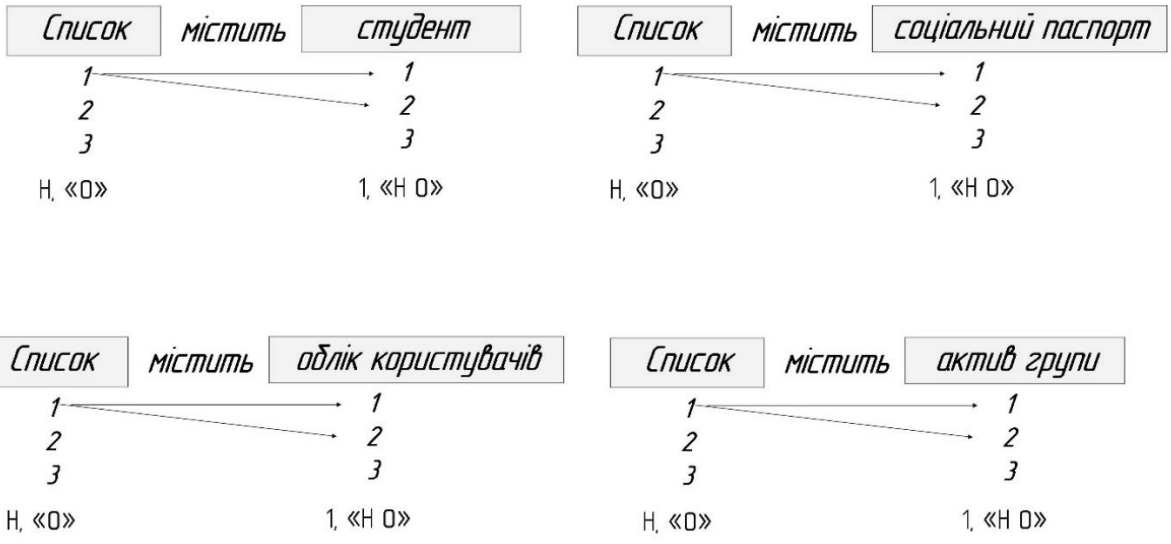
ДОДАТКИ

Додаток 1

Структура бази даних



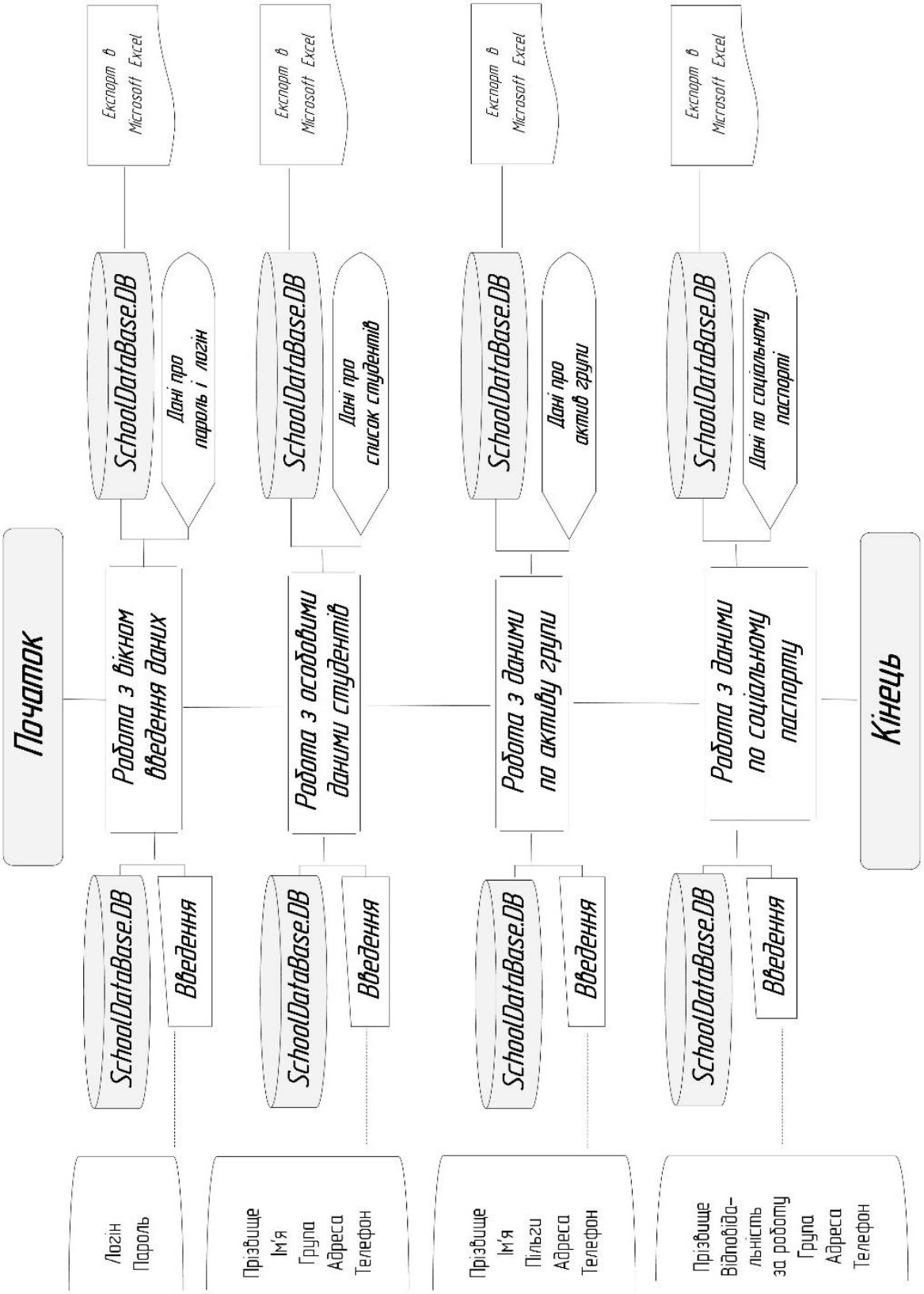
Реляційний тип бази даних



Метод «Сутність–зв'язок»

Додаток 2

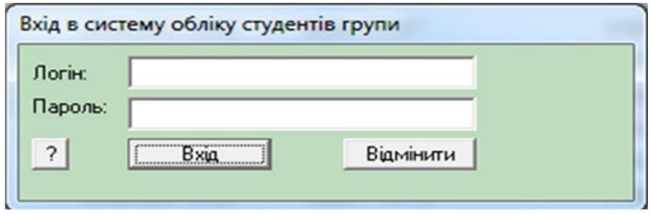
Діаграма роботи фізичної структури інформаційної системи



Додаток 3

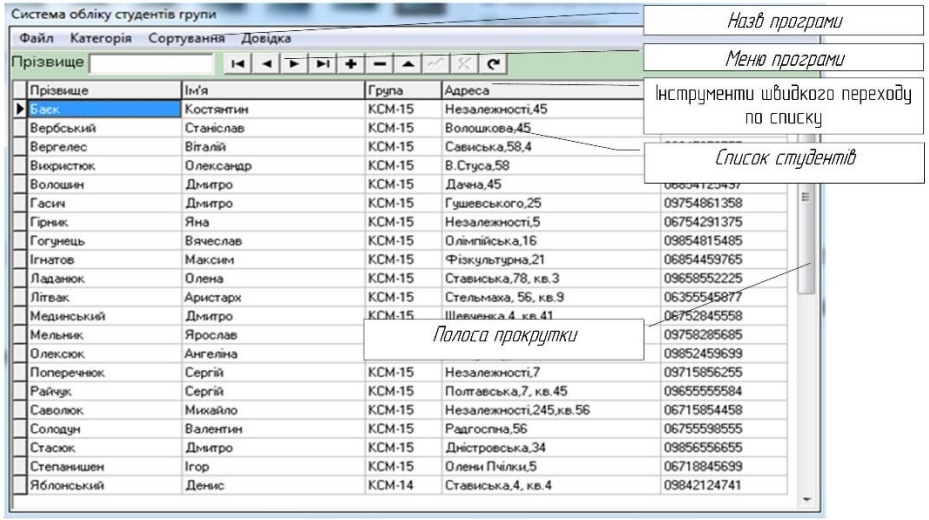
Таблиця відповідності логічної та фізичної структури

Назва поля	Тип даних	Пояснення
Прізвище	Var char(250)	зміна/введення текстових даних про прізвище, первісний ключ
Ім'я	Var char(250)	зміна/введення текстових даних про ім'я
Група	Var char(250)	зміна/введення текстових даних про групу
Адреса	Var char(250)	зміна/введення текстових даних про адресу
Телефон	int, pk	введення цифрових даних про номер телефону
Пільги	Var char(250)	зміна/введення текстових даних про пільги
Відповідає за роботу	Var char(250)	зміна/введення текстових даних про відповідальність за роботу

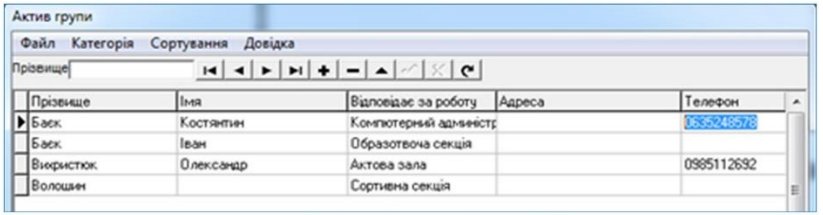


Вікно входу в програму «Система обліку студентів групи»

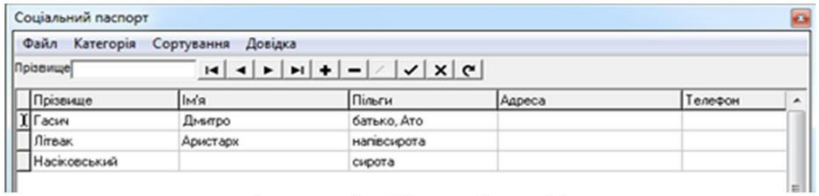
Структура програмного забезпечення



Головна сторінка



Інтерфейс діалогового вікна «Актив групи»



Інтерфейс діалогового вікна «Соціальний паспорт»

Додаток 4

Лістинг головної сторінки програмного забезпечення

```
unit StudentUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, ADODB, Grids, DBGrids, Menus, ToolWin, ComCtrls, StdCtrls,
  ExtCtrls, DBCtrls,ComObj;

type
  TStudentForm = class(TForm)
    DBGrid1: TDBGrid;
    MainMenu1: TMainMenu;
    FileItem: TMenuItem;
    ExitItem: TMenuItem;
    CategoryItem: TMenuItem;
    SortItem: TMenuItem;
    ByFamItem: TMenuItem;
    ByClassItem: TMenuItem;
    ToolBar1: TToolBar;
    Label1: TLabel;
    FilterFamEdit: TEdit;
    StudentNavigator: TDBNavigator;
    HelpItem: TMenuItem;
    ProgrammInfo: TMenuItem;
    CategoryPrepod: TMenuItem;
    ExelItem: TMenuItem;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    ToolButton1: TToolButton;
    procedure ByFamItemClick(Sender: TObject);
    procedure ByClassItemClick(Sender: TObject);
    procedure ExitItemClick(Sender: TObject);
    procedure FilterFamEditChange(Sender: TObject);
    procedure ProgrammInfoClick(Sender: TObject);
    procedure CategoryPrepodClick(Sender: TObject);
    procedure HelpMeItemClick(Sender: TObject);
    procedure ExelItemClick(Sender: TObject);
    procedure TableInfoItemClick(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  StudentForm: TStudentForm;

implementation

uses DataModuleUnit, PrepodUnit, UserUnit, Unit1, Unit2;

{$R *.dfm}
```

```
procedure TStudentForm.ByFamItemClick(Sender: TObject);
begin
DataModule1.StudentTable.Sort:='Фамилия';
end;

procedure TStudentForm.ByClassItemClick(Sender: TObject);
begin
DataModule1.StudentTable.Sort:='Класс';
end;

procedure TStudentForm.ExitItemClick(Sender: TObject);
begin
Close;
end;

procedure TStudentForm.FilterFamEditChange(Sender: TObject);
begin
If Length(FilterFamEdit.Text)>0 then
   DataModule1.StudentTable.Filtered:=True
else DataModule1.StudentTable.Filtered:=False;
DataModule1.StudentTable.Filter:='Фамилия>'''+FilterFamEdit.Text+'''';
end;

procedure TStudentForm.ProgrammInfoClick(Sender: TObject);
begin
MessageDlg("Система обліку студентів групи" дозволяє вести облік студентів групи,
список активу групи та соціальний паспорт, забергати у файлі формату програми
Microsoft Excel',
mtInformation,[mbok],0);
end;

procedure TStudentForm.CategoryPrepodClick(Sender: TObject);
begin
StudentForm.Hide;
PrepodForm.Show;
end;

procedure TStudentForm.HelpMeItemClick(Sender: TObject);
begin
//winhelp(StudentForm.Handle,'к.hlp',HELP_CONTEXT,0);
end;


procedure TStudentForm.ExelItemClick(Sender: TObject);
var
 XLApp,Sheet,Colum:Variant;
 index,i:Integer;
begin
 XLApp:= CreateOleObject('Excel.Application');
 XLApp.Visible:=true;
 XLApp.Workbooks.Add(-4167);
 XLApp.Workbooks[1].WorkSheets[1].Name:='Група-студенти';
 Colum:=XLApp.Workbooks[1].WorkSheets['Група-студенти'].Columns;
 Colum.Columns[1].ColumnWidth:=20;
 Colum.Columns[2].ColumnWidth:=20;
 Colum.Columns[3].ColumnWidth:=20;
 Colum.Columns[4].ColumnWidth:=20;
 Colum.Columns[5].ColumnWidth:=20;

 Colum:=XLApp.Workbooks[1].WorkSheets['група-студенти'].Rows;
 Colum.Rows[2].Font.Bold:=true;
 Colum.Rows[1].Font.Bold:=true;
 Colum.Rows[1].Font.Color:=clBlack;
 Colum.Rows[1].Font.Size:=14;
```

```
 Sheet:=XLApp.Workbooks[1].WorkSheets['Група-студенти'];
 Sheet.Cells[1,2]:='Коледж';
 Sheet.Cells[2,1]:='Прізвище';
 Sheet.Cells[2,2]:='Ім"я';
 Sheet.Cells[2,3]:='Група';
 Sheet.Cells[2,4]:='Адреса';
 Sheet.Cells[2,5]:='Телефон';

 index:=3;
 DataModule1.StudentTable.First;
for i:=0 to DataModule1.StudentTable.RecordCount-1 do
 begin
   Sheet.Cells[index,1]:=DataModule1.StudentTable.Fields.Fields[1].AsString;
   Sheet.Cells[index,2]:=DataModule1.StudentTable.Fields.Fields[2].AsString;
   Sheet.Cells[index,3]:=DataModule1.StudentTable.Fields.Fields[3].AsString;
   Sheet.Cells[index,4]:=DataModule1.StudentTable.Fields.Fields[5].AsString;
   Sheet.Cells[index,5]:=DataModule1.StudentTable.Fields.Fields[4].AsString;
   Inc(index);
   DataModule1.StudentTable.Next;
 end;
 end;
procedure TStudentForm.TableInfoItemClick(Sender: TObject);
var
s:integer;
i:integer;
begin
s:=0;
for i:=0 to DataModule1.StudentTable.RecordCount-1 do
 s:=s+1;
ShowMessage('Назва таблиці: '+Datamodule1.StudentTable.TableName+#13+'Кількість
записів: '+IntToStr(s));
end;

procedure TStudentForm.N1Click(Sender: TObject);
begin
PersonalForm.Show;

end;

procedure TStudentForm.N2Click(Sender: TObject);
begin
UsersForm.ShowModal;
end;

procedure TStudentForm.FormActivate(Sender: TObject);
begin
If login='admin' then N2.Visible:=true
  else n2.Visible:=false;
end;

procedure TStudentForm.FormCreate(Sender: TObject);
begin
if Dostup=false then begin
DBGrid1.Enabled:=false;
StudentNavigator.VisibleButtons:=[nbFirst,nbNext,nbPrior,nbLast,nbRefresh];
DbGrid1.Font.Style:=[fsBold];
end;
end;




end.
```

Додаток 5

Лістинг діалогового вікна «Соціальний паспорт»

```
unit PrepodUnit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, ToolWin, ComCtrls, Menus, StdCtrls, ExtCtrls,
  DBCtrls,ComObj;

type
  TPrepodForm = class(TForm)
    DBGrid1: TDBGrid;
    ToolBar1: TToolBar;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    Label1: TLabel;
    FilterFamEdit: TEdit;
    Sort: TMenuItem;
    N5: TMenuItem;
    PrepodNavigator: TDBNavigator;
    ExelItem: TMenuItem;
    Cghfdrf1: TMenuItem;
    N6: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N4Click(Sender: TObject);
    procedure FilterFamEditChange(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure ExelItemClick(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N8Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  PrepodForm: TPrepodForm;

implementation

uses DataModuleUnit, StudentUnit, UserUnit, Unit1;

{$R *.dfm}

procedure TPrepodForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
StudentForm.Close;
```

```
end;

procedure TPrepodForm.N4Click(Sender: TObject);
begin
PrepodForm.Hide;
StudentForm.Show;
end;

procedure TPrepodForm.FilterFamEditChange(Sender: TObject);
begin
If Length(FilterFamEdit.Text)>0 then
   DataModule1.PrepodTable.Filtered:=true
 else DataModule1.PrepodTable.Filtered:=false;
DataModule1.PrepodTable.Filter:='Фамилия>'''+FilterFamEdit.Text+'''';
end;

procedure TPrepodForm.N5Click(Sender: TObject);
begin
DataModule1.PrepodTable.IndexFieldNames:='Фамилия';
end;

procedure TPrepodForm.ExelItemClick(Sender: TObject);
var
 XLApp,Sheet,Colum:Variant;
 index,i:Integer;
begin
 XLApp:= CreateOleObject('Excel.Application');
 XLApp.Visible:=true;
 XLApp.Workbooks.Add(-4167);
 XLApp.Workbooks[1].WorkSheets[1].Name:='Соціальний паспорт';
 Colum:=XLApp.Workbooks[1].WorkSheets['Соціальний паспорт'].Columns;
 Colum.Columns[1].ColumnWidth:=20;
 Colum.Columns[2].ColumnWidth:=20;
 Colum.Columns[3].ColumnWidth:=20;
 Colum.Columns[4].ColumnWidth:=20;
 Colum.Columns[5].ColumnWidth:=20;

 Colum:=XLApp.Workbooks[1].WorkSheets['Соціальний паспорт'].Rows;
 Colum.Rows[2].Font.Bold:=true;
 Colum.Rows[1].Font.Bold:=true;
 Colum.Rows[1].Font.Color:=clBlack;
 Colum.Rows[1].Font.Size:=14;

 Sheet:=XLApp.Workbooks[1].WorkSheets['Соціальний паспорт'];
 Sheet.Cells[1,2]:='Коледж';
 Sheet.Cells[2,1]:='Прізвище';
 Sheet.Cells[2,2]:='Ім"я';
 Sheet.Cells[2,3]:='Пільги';
 Sheet.Cells[2,4]:='Адреса';
 Sheet.Cells[2,5]:='Телефон';

 index:=3;
 DataModule1.PrepodTable.First;
 for i:=0 to DataModule1.PrepodTable.RecordCount-1 do
  begin
   Sheet.Cells[index,1]:=DataModule1.PrepodTable.Fields.Fields[1].AsString;
   Sheet.Cells[index,2]:=DataModule1.PrepodTable.Fields.Fields[2].AsString;
   Sheet.Cells[index,3]:=DataModule1.PrepodTable.Fields.Fields[3].AsString;
   Sheet.Cells[index,4]:=DataModule1.PrepodTable.Fields.Fields[5].AsString;
   Sheet.Cells[index,5]:=DataModule1.PrepodTable.Fields.Fields[4].AsString;
   Inc(index);
   DataModule1.PrepodTable.Next;
  end;
 end;
```

```
procedure TPrepodForm.N6Click(Sender: TObject);
begin
MessageDlg('Система обліку студентів групи',
mtInformation,[mbok],0);
end;

procedure TPrepodForm.N2Click(Sender: TObject);
begin
Close;
end;

procedure TPrepodForm.N7Click(Sender: TObject);
var
s:integer;
i:integer;
begin
s:=0;
for i:=0 to DataModule1.PrepodTable.RecordCount-1 do
 s:=s+1;
ShowMessage('Назва таблиці: '+Datamodule1.PrepodTable.TableName+#13+'Кількість
записів: '+IntToStr(s));
end;


procedure TPrepodForm.N8Click(Sender: TObject);
begin
PersonalForm.Show;
PrepodForm.hide;
end;

procedure TPrepodForm.FormCreate(Sender: TObject);
begin
if login='admin' then N9.Visible:=true
  else n9.Visible:=false;
   if Dostup=false then begin
DBGrid1.Enabled:=false;
PrepodNavigator.VisibleButtons:=[nbFirst,nbNext,nbPrior,nbLast,nbRefresh];
DbGrid1.Font.Style:=[fsBold];
end;
end;

end.
```

Додаток 6

Лістинг діалогового вікна «Актив групи»

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, DBCtrls, StdCtrls, Menus, ToolWin, ComCtrls, Grids,
  DBGrids,ComObj;

type
  TPersonalForm = class(TForm)
    DBGrid1: TDBGrid;
    ToolBar1: TToolBar;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    Excel1: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N11: TMenuItem;
    Label1: TLabel;
    FilterFamEdit: TEdit;
    DBNavigator1: TDBNavigator;
    N12: TMenuItem;
    procedure FilterFamEditKeyPress(Sender: TObject; var Key: Char);
    procedure N8Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N6Click(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure Excel1Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  PersonalForm: TPersonalForm;

implementation

uses DataModuleUnit, PrepodUnit, StudentUnit, Unit2, UserUnit;

{$R *.dfm}

procedure TPersonalForm.FilterFamEditKeyPress(Sender: TObject;
  var Key: Char);
begin
If Length(FilterFamEdit.Text)>0 then
   DataModule1.PersonalTable.Filtered:=True
```

```
else DataModule1.PersonalTable.Filtered:=False;
DataModule1.PersonalTable.Filter:='Фамилия>'''+FilterFamEdit.Text+'''';
end;

procedure TPersonalForm.N8Click(Sender: TObject);
begin
DataModule1.PersonalTable.Sort:='Фамилия';
end;

procedure TPersonalForm.N3Click(Sender: TObject);
begin
Close;
end;

procedure TPersonalForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
StudentForm.Close;
end;

procedure TPersonalForm.N6Click(Sender: TObject);
begin
StudentForm.Show;
PersonalForm.Hide;
end;

procedure TPersonalForm.N5Click(Sender: TObject);
begin
PrepodForm.Show;
PersonalForm.Hide;
end;

procedure TPersonalForm.Excel1Click(Sender: TObject);
var
 XLApp,Sheet,Colum:Variant;
 index,i:Integer;
begin
 XLApp:= CreateOleObject('Excel.Application');
 XLApp.Visible:=true;
 XLApp.Workbooks.Add(-4167);
 XLApp.Workbooks[1].WorkSheets[1].Name:='Актив групи';
 Colum:=XLApp.Workbooks[1].WorkSheets['Актив групи'].Columns;
 Colum.Columns[1].ColumnWidth:=20;
 Colum.Columns[2].ColumnWidth:=20;
 Colum.Columns[3].ColumnWidth:=20;
 Colum.Columns[4].ColumnWidth:=20;
 Colum.Columns[5].ColumnWidth:=20;

 Colum:=XLApp.Workbooks[1].WorkSheets['Актив групи'].Rows;
 Colum.Rows[2].Font.Bold:=true;
 Colum.Rows[1].Font.Bold:=true;
 Colum.Rows[1].Font.Color:=clBlack;
 Colum.Rows[1].Font.Size:=14;

 Sheet:=XLApp.Workbooks[1].WorkSheets['Актив групи'];
 Sheet.Cells[1,2]:='Коледж';
 Sheet.Cells[2,1]:='Прізвище';
 Sheet.Cells[2,2]:='Ім"я';
 Sheet.Cells[2,3]:='Відповідає за роботу';
 Sheet.Cells[2,4]:='Адреса';
 Sheet.Cells[2,5]:='Телефон';

 index:=3;
 DataModule1.PersonalTable.First;
```

```
for i:=0 to DataModule1.PersonalTable.RecordCount-1 do
 begin
   Sheet.Cells[index,1]:=DataModule1.PersonalTable.Fields.Fields[1].AsString;
   Sheet.Cells[index,2]:=DataModule1.PersonalTable.Fields.Fields[2].AsString;
   Sheet.Cells[index,3]:=DataModule1.PersonalTable.Fields.Fields[3].AsString;
   Sheet.Cells[index,4]:=DataModule1.PersonalTable.Fields.Fields[5].AsString;
   Sheet.Cells[index,5]:=DataModule1.PersonalTable.Fields.Fields[4].AsString;
   Inc(index);
   DataModule1.PersonalTable.Next;
 end;
 end;

procedure TPersonalForm.N2Click(Sender: TObject);
 var
s:integer;
i:integer;
begin
s:=0;
for i:=0 to DataModule1.PersonalTable.RecordCount-1 do
 s:=s+1;
ShowMessage('Назва таблиці: '+Datamodule1.PersonalTable.TableName+#13+'Кількість
записів: '+IntToStr(s));
end;

procedure TPersonalForm.N11Click(Sender: TObject);
begin
MessageDlg('Система обліку студентів групи',
mtInformation,[mbok],0);
end;

procedure TPersonalForm.FormCreate(Sender: TObject);
begin
if login='admin' then n12.Visible:=true
  else N12.Visible:=false;
     if Dostup=false then begin
DBGrid1.Enabled:=false;
DbNavigator1.VisibleButtons:=[nbFirst,nbNext,nbPrior,nbLast,nbRefresh];
DbGrid1.Font.Style:=[fsBold];
end;
end;

end.
```