

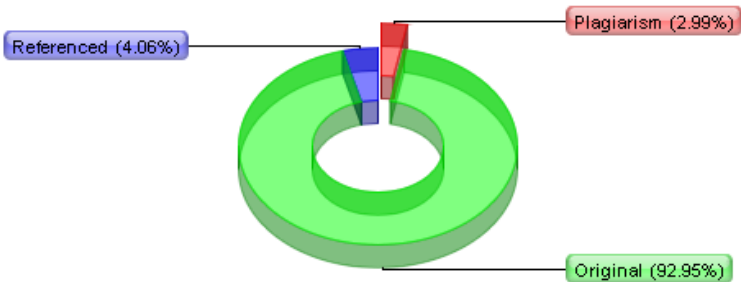
Детектор Плагиата v. 2215 - Отчёт оригинальности: 13.06.2024 10:25:44

Проанализированный документ: Селіверстов\_пз.docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ

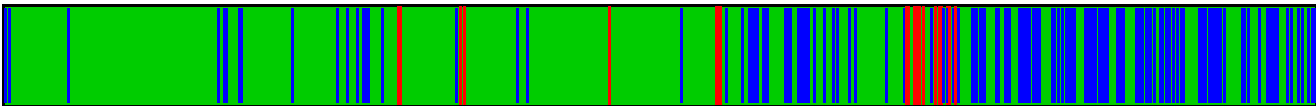
- ? Тип поиска: Поиск переписанного ? Язык: Uk
- ? Тип проверки: Интернет
- ТЭЕ и кодировка: DocX n/a

Детальный анализ тела документа:

? Диаграмма соотношения частей:



? Граф распределения зон:



? Источники плагиата: 21

	2%		248	1. <a href="https://psychologer.com.ua/manipuliatsiia/">https://psychologer.com.ua/manipuliatsiia/</a>
	2%		309	2. <a href="http://eprints.zu.edu.ua/33169/1/Інформ%20системи%20в%20менеджменті.pdf">http://eprints.zu.edu.ua/33169/1/Інформ системи в менеджменті.pdf</a>
	1%		206	3. <a href="https://victormochere.com/uk/how-to-fix-error-establishing-a-database-connection-on-wordpress">https://victormochere.com/uk/how-to-fix-error-establishing-a-database-connection-on-wordpress</a>

? Детали обработанных ресурсов: 172 - ОК / 3 - Ошибок

? Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
Обнаружена Wiki!	[не обнаружено]	[не обнаружено]	Обнаружено сокрытие!

? Античит-отчет UACE:

1. Статус: Анализатор <b>Включен</b> Нормализатор <b>Включен</b> сходство символов установлено на <b>100%</b>
2. Обнаруженный процент загрязнения UniCode: <b>25,8%</b> с лимитом: 4%
3. Процент нераспознанных символов после нормализации: <b>15%</b>
4. Все подозрительные символы будут отмечены фиолетовым цветом: <b>Abcd...</b>
5. Найдены невидимые символы: 0
Рекомендации по оценке: Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется

передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

Исключённые ресурсы:

URL не найдены

Включённые ресурсы:

URL не найдены

## ? Детальный анализ документа:

Міністерство освіти і науки України ДЗ

Цитування: 0,04%

id: 1

«луганський Національний університет імені тараса шевченка»

Навчально-науковий інститут математики та інформаційних технологій (назва факультету, інституту) Інформаційних технологій та систем (назва кафедри) Пояснювальна записка до дипломного проєкту (роботи) БАКАЛАВРА (освітньо-кваліфікаційний рівень) на тему: РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ ІЗ ЗАСТОСУВАННЯМ КЛІЄНТ-СЕРВЕРНИХ СУБД Виконав: студент 4 курсу напряму підготовки (спеціальності) 121

Цитування: 0,02%

id: 2

«Інженерія програмного забезпечення»\_

(шифр і назва напряму підготовки, спеціальності) Селіверстов В. Д. (прізвище та ініціали) Керівник \_\_\_\_ Смагіна О.О. (прізвище та ініціали) Рецензент \_\_\_\_ Козуб Ю. Г. (прізвище та ініціали) Полтава – 2024 року ЗМІСТ ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬЗ ВСТУП4 РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ ІНФОРМАЦІЙНО-ДОВІДКОВИХ СИСТЕМ6 1.1. Теоретичні основи інформаційно-довідкових систем6 1.2. Технологія проєктування та розробки інформаційно-довідкових систем10 1.3. Передпроєктний етап розробки інформаційно-довідкової системи для навчального закладу12 Висновки до розділу15 РОЗДІЛ 2. ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ БАЗИ ДАНИХ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ17 2.1. Клієнт – серверна архітектура СУБД17 2.2. Порівняльний аналіз СУБД клієнт – серверної архітектури21 2.3. Проєктування моделі бази даних інформаційно-довідкової системи24 Висновки до розділу34 РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ36 3.1. Аналіз засобів та технологій розробки додатку36 3.2. Розробка та реалізація програмного додатку42 3.3. Тестування додатку54 Висновки до розділу57 ВИСНОВКИ58 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ59 ДОДАТОК А62 ДОДАТОК Б63 ДОДАТОК В76 ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ АІДС – автоматизована інформаційно-довідкова система АІС – автоматизована інформаційна система БД – база даних ІДС – інформаційно-довідкова система СУБД – система управління базами даних ВСТУП Актуальність теми. У сучасних умовах, які характеризуються активізацією інформаційних процесів у всіх сферах суспільного життя, зокрема в освіті, набуває актуальності проблема інформаційного забезпечення освітнього закладу, від якого залежить ефективна організація навчального процесу. На відміну від роботи з довідковою інформацією на паперових носіях перехід до автоматизованих інформаційно-довідкових систем забезпечує ряд переваг. Електронні документи можуть одночасно використовуватися співробітниками в рамках однієї робочої групи, відділу або всього закладу. Доступ до них здійснюється дуже швидко. Прискорений доступ до стратегічної інформації разом із значною економією коштів може забезпечити і важливі організаційні переваги. Впровадження автоматизованих інформаційно-довідкових систем дозволяє вирішити завдання, що найбільш часто зустрічаються: забезпечення більш ефективного управління за рахунок автоматичного контролю виконання, прозорості діяльності організації на всіх рівнях; підтримку ефективного накопичення, керування і доступу до інформації; забезпечення кадрової гнучкості за рахунок більшої формалізації діяльності кожного співробітника і можливості збереження всієї передісторії його діяльності; виключення паперових документів з внутрішнього обороту підприємства і пов'язану з цим економію ресурсів за рахунок скорочення витрат на управління потоками електронної інформації в організації; істотне спрощення і здешевлення збереження паперових документів за рахунок наявності оперативного електронного архіву. Зазначене обумовлює актуальність обраної теми. Метою роботи є аналіз технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД. Об'єктом дослідження є технології розробки інформаційно- довідкових систем для навчальних закладів із застосуванням клієнт- серверних СУБД. Предметом дослідження є інформаційно-довідкова система для навчальних закладів із застосуванням клієнт-серверних СУБД. Відповідно до предмета, мети було визначено основні завдання дослідження: дослідити інформаційно-довідкову систему для навчальних закладів; проаналізувати етапи розробки інформаційно-довідкових систем; визначити завдання та вимоги до інформаційно-довідкової системи; розглянути можливості технології клієнт-

серверної архітектури; проаналізувати існуючі СУБД клієнт-серверної архітектури та обґрунтувати вибір СУБД для подальшої розробки бази даних; змодельовати базу даних інформаційно-довідкової системи; розробити клієнтський додаток інформаційно-довідкової системи для навчальних закладів із застосуванням клієнт-серверних СУБД. Для вирішення завдань дослідження використано такі методи дослідження: теоретичні: аналіз наукової літератури, узагальнення та систематизація теоретичних положень про інформаційно-довідкові системи для навчальних закладів, моделювання інформаційних процесів; емпіричні: порівняльний аналіз можливостей інструментів розробки програмних додатків; експериментальні: тестування розробленої системи. До складу роботи входять три розділи, які висвітлюють питання технології розробки інформаційно-довідкових систем для навчальних закладів, підходи до проєктування баз даних клієнт-серверної архітектури. Практичним результатом роботи є розроблена інформаційно-довідкова система навчального закладу:

Цитування: 0,06%

id: 3

"Дніпровська загальноосвітня школа I - III ступенів №5"

на базі клієнт- серверної архітектури. РОЗДІЛ 1.  
АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ ІНФОРМАЦІЙНО- ДОВІДКОВИХ СИСТЕМ 1.1. Теоретичні основи інформаційно-довідкових систем Зростання обсягів інформації, прискорення і ускладнення її обліку викликають необхідність автоматизації процесу зберігання, обробки та відображення інформації. Комплекс засобів, що дозволяє забезпечити безперервне протікання інформаційних процесів становлять автоматизовану інформаційну систему (АІС) - це [1, с. 9]. Різновидом АІС є автоматизована інформаційно-довідкова система. (АІДС). Інформаційно-довідкові системи орієнтовані більшою мірою на збір, зберігання і видачу за запитом користувача формалізованої інформації економічного, технічного або технологічного характеру. Можна вважати, що ІДС орієнтовані на роботу з конкретизованими даними цифрового або текстового типу. Залежно від предметної галузі АІДС можуть дуже сильно відрізнятися за своїми функціями та способами реалізації. Однак можна виділити ряд властивостей, які є загальними для всіх інформаційно-довідкових систем, а саме: вони призначені для збору, зберігання та обробки інформації. Тому в основі будь-якої з них лежить середовище зберігання та доступу до даних (база даних); вони орієнтуються на кінцевого користувача, які часто мають низький рівень інформаційної культури та володіють навичками роботи за комп'ютерами на початковому рівні. Тому клієнтські програми інформаційної системи повинні володіти простим та зручним графічним інтерфейсом, який надає кінцевому користувачеві всі необхідні для роботи функції та запобігає виконання зайвих дій [11, с. 15]. Щодо визначення завдань, які висуваються до інформаційно- довідкових систем, слід розглянути основні завдання інформаційних систем: Як і ІС, інформаційно-довідкові системи призначені для пошуку, обробки та збереження інформації за допомогою комп'ютерної техніки. Використання комп'ютерів дозволяє забезпечити більш швидку та надійну обробку інформації, зекономити людський час на роботу з даними та уникнути властивих людині випадкових помилок. Для збереження та обробки інформації ІДС повинні мати відповідні засоби для роботи з ними. У результаті розвитку більшості таких систем у них виділився окремий компонент, який являє собою різновид системи управління базами даних. ІДС повинні мати певні процедури, за допомогою яких можна автоматизувати процес формування звітної документації. Наявність можливостей виконання різноманітних інформаційних запитів користувачів до інформаційних сховищ та баз даних, а також підтримка спеціальних мов запитів для систем такого типу [10]. Слід зазначити, що завдання, які повинні вирішуватися конкретною інформаційно-довідковою системою, залежать від тієї прикладної галузі, для якої призначена ця система. Галузі застосування інформаційних додатків можуть бути доволі різноманітними: банківська справа, управління виробництвом, медицина, транспорт, освіта тощо. Використання АІДС в системах організаційного управління забезпечує: інформаційно-довідкове обслуговування; автоматизацію діловодства; розвинений діалог між користувачем і ЕОМ; формування і ведення локальних баз даних і використання централізованої бази даних при наявності обчислювальної мережі; надання різних сервісних послуг користувачам на робочому місці. В основу класифікації АІДС може бути покладено ряд класифікаційних ознак. Розрізняють автоматизовані інформаційно-довідкові системи з вигляду запиту і формою подання результату. Запит може бути стандартний і довільний. Результат може бути представлений або довідкою стандартної форми, або форма проєктується в довільному вигляді за бажанням користувача в момент обробки його

запиту. Залежно від характеру роботи з інформацією розрізняють наступні види АІДС: автоматизовані архіви (АА); автоматизовані системи діловодства (АСД); геоінформаційні системи (ГІС); автоматизовані довідники (АД) і каталоги (АК) та ін. За способом зберігання даних поділяють наступні АІДС: фактографічні (дані зберігаються в структурованому вигляді). У фактографічних АІС реєструються факти - конкретні значення даних (атрибутів) про об'єкти реального світу; документальні (в текстовому вигляді). Базу даних таких систем утворює сукупність неструктурованих текстових документів (статті, книги, реферати, тексти законів) і графічних об'єктів, забезпечена тим чи іншим формалізованим апаратом пошуку. У складі АІДС можна виділити три основних технологічних процеси, представлені на рис. 1.1. Рис. 1.1. Основні технологічні процеси АІДС

Організаційно-технологічна підсистема збору і введення інформації забезпечує відбір і накопичення даних в інформаційну систему і включає сукупність джерел інформації, організаційно-технологічні ланцюжки відбору інформації для накопичення в системі. Без правильно організованої, оперативно і ефективно діючої організаційно-технологічної підсистеми збору інформації неможлива ефективна організація функціонування всієї інформаційної системи в цілому. Підсистема подання та обробки інформації становить ядро інформаційно-довідкової системи і є відображенням уявлення розробниками і абонентами системи структури та картини предметної області, відомості про яку повинна відображати інформаційно-довідкова система. Підсистема подання та обробки інформації є одним з найбільш складних компонентів при розробці інформаційної системи. Інформаційним ядром цієї підсистеми, або, інакше кажучи, внутрішнім носієм знань про предметну галузь є база даних (БД). Нормативно-функціональна підсистема виведення інформації визначає користувачів, або інакше абонентів, системи, реалізує цільовий аспект призначення і виконання завдань інформаційної системи. Структуру автоматизованої інформаційно-довідкової системи поділяють на дві складові: функціональну частину, яка відображатиме цілі і завдання системи, і частина, яка забезпечує виконання цих завдань та містить засоби вирішення завдань. Частина, яка забезпечує АІДС, складається з інформаційного, програмного, технічного, організаційного забезпечення тощо. До інформаційного забезпечення належать масиви інформації, що зберігається в базах даних на дискових накопичувачах. Сюди ж відноситься і СУБД. До технічного забезпечення відноситься комплекс технічних засобів, які забезпечують вирішення поставлених завдань, а також засоби зв'язку з іншими АІС, які працюють в загальній мережі об'єкта, а також інші засоби зв'язку (телефон, факс та ін.). Програмне забезпечення включає операційні системи, сервісні програми, стандартні програми користувачів і пакети прикладних програм, виконані за модульним принципом і орієнтовані на вирішення певного класу задач, обумовленого призначенням АІДС. У міру необхідності в програмне забезпечення включаються також пакети програм для роботи з графічною інформацією. Організаційне забезпечення АІДС має на меті організацію їх функціонування, розвитку, підготовки кадрів і адміністрування (планування роботи, облік, контроль, аналіз, регулювання, документальне оформлення прав і обов'язків користувачів АІДС). Отже, інформаційно-довідкові системи є одним із видів інформаційних систем, які призначені для збереження, пошуку та обробки інформації. Створення АІДС забезпечує такі переваги розподіленої обробки даних, як більш низька вартість, висока надійність, поєднання автономного і розрахованого на багато користувачів режимів роботи і т. п.

1.2. Технологія проектування та розробки інформаційно-довідкових систем

Проектування АІДС - процес створення і впровадження проєктів комплексного вирішення виробничих завдань за допомогою інформаційних технологій. Однією зі складових процесу проектування є детальна розробка окремих проєктних рішень, їх аналіз, апробація і впровадження. Методи проектування АІДС включають: індивідуальне (оригінальне), типове проектування, автоматизоване проектування. Індивідуальне проектування характеризується тим, що всі види робіт для різних об'єктів виконуються за індивідуальними проєктами. У процесі індивідуального проектування застосовуються свої оригінальні методики і засоби проведення робіт. Склад робіт на всіх етапах обстеження, проектування і впровадження створюються для конкретного об'єкта. Для цього методу проектування характерні висока трудомісткість, тривалі терміни проектування. Типове проектування - розбиття системи на безліч складових компонентів і створення для кожного з них закінченого проєктного рішення, яке при впровадженні прив'язується до конкретних умов об'єкта. Залежно від декомпозиції розрізняють: елементне проектування, підсистемне, об'єктне. Під час елементного методу проектування вся система розподіляється на кінцеве безліч елементів, кожен з яких є типовим. Елементами можуть бути проєктні рішення з інформаційного, технічного,

програмного видів забезпечення. Підсистемний метод проектування характеризується більш високим ступенем інтеграції елементів ІДС. Автоматизоване проектування здійснюється на основі CASE-засобів. Виділяються кілька етапів створення АІДС: I етап - Передпроектний (обстеження, складання звіту, техніко- економічного обґрунтування і технічного завдання, формування вимог користувача до АІДС, розробка документації на АІДС). Перш ніж починати проектування, необхідно виконати обстеження об'єкта, для якого створюється ІС. Це досить важливий етап, так як дозволяє виділити характерні особливості об'єкта, які слід врахувати в характеристиках розробляється ІС і які визначають подальшу роботу з проектування. Від якості передпроектного обстеження багато в чому залежить, чи доведеться в подальшому переглядати основні концепції створюваної ІДС і вносити в неї принципові зміни, що завжди є трудомістким завданням. Обстеження зводиться до аналізу існуючої системи і об'єкта автоматизації, для якого створюється система. На підставі вивчення об'єкта формується перелік завдань, які повинна вирішувати ІС. Результатом діяльності на цьому етапі буде сформоване технічне завдання, яке обґрунтування розробки та матеріали, що дають уявлення про склад і функціонування ІДС, і включає в себе: загальну характеристику об'єкта, для якого створюється ІДС; опис вимог до ІС; використовуваний комплекс технічних засобів; опис і постановку вирішення завдань, що входять в ІС; опис стандартного програмного забезпечення; опис організації інформаційної бази і т. д. II етап - Проектний (розробка проектних рішень системи і її частин, розробка або адаптація програм). Цей етап складається з двох підетапів: Етап технічного проектування - формуються проектні рішення із забезпечувальної і функціональної частин інформаційної системи, здійснюється постановка задачі і блок-схеми і їх рішення. Саме на цьому етапі відбувається проектування інформаційних моделей баз даних системи - побудова інфологічної та даталогічної моделей БД, які відображають всю концепцію даних системи з їх якостями та асоціаціями. Етап робочого проектування - здійснюється розробка фізичної бази даних, розробка програмного коду. Цей етап передбачає й доведення системи, коригування структури, створення різної документації: на поставку, на установку технічних засобів, інструкції по експлуатації, посадові інструкції тощо. III етап - Впровадження. Цей етап передбачає підготовку до впровадження, проведення дослідних випробувань, тестувань інформаційно- довідкової системи, здачу в промислову експлуатацію, підготовку персоналу, проведення навчання персоналу. IV етап - Супровід і аналіз функціонування. Цей етап складається з виявлення проблем, внесення змін в проектні рішення і існуючі ІС. Основними учасниками процесу створення автоматизованої інформаційно-довідкової системи є підприємство-замовник, для якого вона створюється і підприємство-розробник, що виконує роботи з проектування ІДС. Юридичні та організаційні взаємини конкретно замовників і розробників регулюються укладеними між ними договорами. 1.3. Передпроектний етап розробки інформаційно-довідкової системи для навчального закладу Відповідно до етапів розробки автоматизованої інформаційно- довідкової системи було проведено обстеження об'єкта автоматизації - інформаційно-довідкової системи навчального закладу:

” Цитування: 0,06%

id: 4

"Дніпровська загальноосвітня школа I - III ступенів №5"

На підставі вивчення об'єкта було сформовано цілі, завдання, вимоги до інформаційно-довідкової системи. Результатом цієї діяльності було розроблене технічне завдання. Технічне завдання на розробку інформаційно-довідкової системи для загальноосвітнього закладу матиме наступні розділи. Призначення, цілі та завдання: Інформаційно-довідкова система (ІДС) призначена для формування довідкової інформації за допомогою інформаційних технологій в загальноосвітньому закладі:

” Цитування: 0,09%

id: 5

"Лисичанська загальноосвітня школа I - III ступенів №5 Лисичанської міської ради Луганської області".

Мета ІДС: автоматизація процесів формування довідкової інформації в навчальному закладі:

” Цитування: 0,09%

id: 6

"Лисичанська загальноосвітня школа I - III ступенів №5 Лисичанської міської ради Луганської області".



Застосування ІДС сприятиме підвищенню ефективності організації навчального процесу та створенню єдиного інформаційного простору для введення, обробки, аналізу, збереження документів. Завдання, які вирішуються за допомогою ІДС : надання інформації про учнів, їх батьків, предмети та факультативи, що вивчають учні, результати навчальної діяльності кожного учня, викладацький персонал, навантаження вчителів; додавання, видалення, оновлення даних про учнів, батьків учнів, вчителів, навчальний процес, відвідування занять учнями тощо; ведення довідників, які використовуватимуться базою даних, а саме: національність, відомості про батьків, стан здоров'я учнів, кваліфікація вчителів та навчальний заклад, який вони закінчили; формування даних про учня, його успішність, інформація про відвідування занять за ознакою

Цитування: 0,01%

id: 7

«ПІБ»,

пошук даних про учнів за ознакою

Цитування: 0,01%

id: 8

«дата народження»

; формування даних про вчителя, його навчальне навантаження за ознакою ПІБ, пошук даних про викладачів за ознакою

Цитування: 0,01%

id: 9

«категорія»

; формування звітів про склад учнів класу, навантаження викладачів. Вимоги до ІДС  
Загальні вимоги Інформаційно-довідкова система повинна забезпечувати формування, збереження, редагування основних масивів довідкової інформації. ІДС повинна забезпечувати створення інформаційного документу як у електронному, так й у друкованому (паперовому) вигляді. ІДС повинна функціонувати в мережевому просторі та працювати в багатокористувацькому режимі. Вимоги до можливостей налаштування та захисту ІДС З метою захисту інформації у інформаційно-довідковій системі необхідно передбачити відповідні заходи. Доступ до системи повинен здійснюватися за допомогою введення логіна та пароля, що дасть можливість захистити інформацію від несанкціонованого доступу. Система повинна забезпечувати гнучкі можливості налаштування, а саме встановлення шляху доступу до бази даних, додавання нового користувача або видалення існуючого. Вимоги до програмного та технічного забезпечення ІДС В основі реалізації можливостей ІДС повинна знаходитися технологія реляційних баз даних, що реалізує систематизоване сховище даних певної предметної галузі. Реляційні бази даних дозволяють здійснювати контроль за надмірністю, несуперечливістю, цілісністю даних. База даних повинна мати клієнт-серверну архітектуру, що забезпечить централізоване сховище даних на серверній частині та паралельну багатокористувацьку обробку даних. Клієнт-серверна система складається з трьох основних компонент: програмне забезпечення сервера; програмне забезпечення кінцевого користувача (клієнта); проміжне програмне забезпечення. Клієнтська частина ІДС повинна функціонувати під управлінням операційної системи [Windows](#). Комплекс технічних засобів обробки даних для вирішення прикладних завдань повинен включати такі основні компоненти: серверні платформи в складі ЛОМ, які підтримують функціонування СУБД; робочі станції, які забезпечують роботу користувачів на відповідних робочих місцях; засоби, що забезпечують гарантоване безперебійне живлення; телекомунікаційне обладнання, яке підтримує інформаційний обмін між користувачами. Засоби безперебійного живлення повинні забезпечувати стабільне електроживлення в межах технічних характеристик обчислювальних засобів і периферійного обладнання. Електроживлення всіх складових частин сайту повинно виконуватися від електричної мережі змінного струму 220 В частотою 50 Гц. Склад і зміст робіт Створення ІДС відбувається за наступною черговістю робіт: аналіз предметної галузі; вибір СУБД та програмних засобів проектування БД; розробка інфологічної, даталогічної, фізичної моделей БД; розробка клієнтського додатку ІДС, що функціонує з серверною БД; тестування ІДС. Порядок контролю і приймання ІДС Для ІДС встановлюються такі види контролю: попередній контроль (1 етап); підсумковий контроль (2 етап). Попередній контроль повинен проводитися з метою визначення працездатності ІДС, кількісних та якісних характеристик ІДС, коригування подальших етапів розробки. Приймання ІДС (підсумковий контроль) повинна проводитися комісією на підставі результатів

попереднього контролю. Склад комісії, загальні вимоги з приймання робіт, порядок узгодження і затвердження приймальної документації повинні визначатися Замовником та узгоджуватися з виконавцем. Висновки до розділу В першому розділі дипломної роботи було розглянуто аналіз технології розробки інформаційно-довідкових систем для навчального закладу, а саме: досліджено теоретичні основи інформаційно-довідкових систем: розглянуто поняття ІДС, класифікація, структуру та складові підсистеми. Встановлено, що застосування ІДС дозволить ефективно здійснювати формування довідкової інформації й організацію роботи з нею; проаналізовано засоби проектування та етапи створення інформаційно-довідкових систем. Встановлено, що технологія розробки ІДС складається з наступних етапів: передпроектний, проектний (етапи технічного та робочого проектування), впровадження, супровід і аналіз функціонування; визначені завдання, вимоги та обґрунтування розробки та розроблено технічне завдання на розробку інформаційно- довідкової системи навчального закладу

Цитування: 0,06%

id: 10

"Дніпровська загальноосвітня школа I - III ступенів №5".

Під час дослідження встановлено, що для забезпечення централізованого сховища даних на серверній частині та паралельної багатокористувацької обробку даних база даних повинна мати клієнт- серверну архітектуру. РОЗДІЛ 2. ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ БАЗИ ДАНИХ ІНФОРМАЦІО-ДОВІДКОВОЇ СИСТЕМИ 2.1. Клієнт – серверна архітектура СУБД Застосування локальних СУБД мали певні обмеження, що не сприяли створенню ефективних додатків. Локальні СУБД не містять спеціальних додатків і сервісів, що можуть керувати даними, а використовують для цієї мети файлові сервіси операційної системи. Вся реальна обробка даних у таких СУБД здійснюється в клієнтському додатку, і будь які бібліотеки доступу до даних в цьому випадку також знаходяться в адресному просторі клієнтського додатку. Тому при виконанні запитів у таких СУБД, дані повинні бути доставлені в той же самий адресний простір клієнтського додатку. Це і призводить до перевантаження мережі при збільшенні числа користувачів і обсягу даних, а також загрожує іншими неприємними наслідками, наприклад, руйнуванням індексів і таблиць. Тому на сучасному етапі для створення додатків застосовуються клієнт-серверні СУБД [34, с. 268]. Архітектура клієнт-сервер є однією з домінуючих концепцій у створенні розподілених мережних додатків і передбачає взаємодію та обмін даними в всередині мережі. Технологія клієнт-сервер означає такий спосіб взаємодії програмних компонентів, при якому вони утворюють єдину систему. Як видно з назви, існує клієнтський процес, що вимагає певних ресурсів, а також серверний процес, що ці ресурси надає. Зовсім необов'язково, щоб вони перебували на одному комп'ютері. Звичайно прийнято розміщувати сервер на одному вузлу локальної мережі, а клієнтів - на інших вузлах [30, с.15]. У контексті бази даних клієнт управляє користувацьким інтерфейсом і логікою роботи, діючи як робоча станція. Клієнт приймає від користувача запит, перевіряє синтаксис і генерує запит до бази даних мовою SQL або іншою мовою бази даних, відповідно до логіки роботи програми-клієнта. Потім передає повідомлення серверу, очікує надходження відповіді й форматує отримані дані для подання їх користувачеві. Сервер приймає й опрацьовує запити до бази даних, після чого відправляє отримані результати назад клієнтові. Таке опрацювання включає перевірку повноважень клієнта, забезпечення вимог цілісності, а також виконання запиту й оновлення даних. Крім цього підтримується управління паралельністю й відновленням. Архітектура клієнт-сервер має ряд переваг [4, с.1]: забезпечується ширший доступ до існуючих баз даних; підвищується загальна продуктивність системи: оскільки клієнти та сервер перебувають на різних комп'ютерах, їхні процесори можуть виконувати різні завдання паралельно. Налаштування продуктивності комп'ютера із сервером спрощуються, якщо на ньому виконується тільки робота з базою даних; знижується вартість апаратного забезпечення; досить потужний комп'ютер з більшим пристроєм зберігання потрібний тільки серверу – для зберігання й управління базою даних; скорочуються комунікаційні витрати. Частина операцій виконують клієнтські комп'ютери і посилають через мережу тільки запити до баз даних, що дозволяє значно скоротити обсяг даних, що пересилають мережею; підвищується рівень несуперечливості даних, оскільки кожній клієнтській програмі не доведеться виконувати власну перевірку їх цілісності; архітектура клієнт-сервер природно відображається на архітектуру відкритих систем. У процесі розвитку технології клієнт-сервер змінювалися і способи її реалізації. Подальше розширення дворівневої архітектури клієнт-сервер припускає поділ функціональної



частини колишнього,

Цитування: 0,01%

id: 11

«ТОВСТОГО»

(інтелектуального) клієнта на дві частини. Способи організації доступу до даних за технологією клієнт-сервер та відмітимо деякі особливості кожного варіанту наведені нижче [34, с.320]. Дворівнева архітектура клієнт-сервер. СУБД поділяється на дві частини: клієнтську й серверну. У цій архітектурі на виділеному сервері, що працює, як правило, під управлінням серверної операційної системи, встановлюють спеціальне програмне забезпечення (ПЗ) - сервер БД. Основа роботи сервера БД - використання мови запитів (SQL). Запит мовою SQL, переданий клієнтом (робочою станцією) серверу БД, породжує пошук і вибір даних на сервері. Вибрані дані транспортуються мережею від сервера до клієнта (див. рис. 2.1). Рис. 2.1. Дворівнева архітектура

Цитування: 0,01%

id: 12

«клієнт-сервер»

Виділяють такі особливості використання дворівневої архітектури: можливість використання різноманітних засобів, що надає СУБД, завдяки безпосередньому з'єднанню з сервером БД; значно знижується навантаження на мережу. Завдяки відправленню запитів і опрацюванню їх на сервері клієнт обмінюється із сервером тільки необхідними даними; складніше піддається масштабуванню, ніж триврівнева архітектура, тому що доводиться використовувати різні клієнти для різних операційних систем; потребує додаткових затрат для облаштування (встановлення) клієнтської частини; при модифікаціях серверної частини (наприклад додаванні нових функцій, зміні базової СУБД тощо) як правило потребує заміни (поновлення) клієнтської частини. Триврівнева архітектура клієнт-сервер функціонує в [Intranet](#) та [Internet](#) мережах. Клієнтська частина (

Цитування: 0,01%

id: 13

«тонкий клієнт»

), з якою працює користувач, є [Web](#)-браузером або прикладною клієнтською програмою, що взаємодіє із [Web](#)-сервісами. Уся програмна логіка винесена на сервер застосувань, що забезпечує формування запитів до БД, які передаються серверу баз даних для їх виконання. Сервер застосувань може бути [Web](#)-сервером або спеціалізованою серверною програмою (наприклад, [Oracle Forms Server](#)). Схема такої структури подана на рис 2.2 Рис. 2.2. Схема триврівневої архітектури

Цитування: 0,01%

id: 14

«клієнт-сервер»

Саме триврівнева архітектура клієнт-сервер лежить в основі розробки сайтів, порталів. Наприклад, численні [Internet](#)-магазини, пошукові й довідкові сервери, системи для [Internet](#)-телефонії й обміну повідомленнями в реальному часі, системи передавання відео через [Internet](#) тощо. Технологія

Цитування: 0,01%

id: 15

"клієнт-сервер"

стосовно до СУБД зводиться до поділу системи на дві частини - додаток-клієнт ([Front-end](#)) і сервер бази даних ([back-end](#)). Архітектура

Цитування: 0,01%

id: 16

"клієнт-сервер "

значно спрощує і прискорює розробку додатків за рахунок того, що правила перевірки цілісності даних знаходяться на сервері. Неправильно працюючий клієнтський додаток не може призвести до втрати даних. Всі ці можливості, раніше властиві тільки складним і дорогим системам, зараз доступні навіть невеликим організаціям. Основа роботи сервера БД - використання мови запитів [SQL \(Structured Query Language\)](#). [SQL](#)-сервер забезпечує інтерпретацію запиту, його виконання в базі даних, формування результату виконання запиту і видачу його додатку-клієнту. При цьому ресурси клієнтського комп'ютера не беруть участь у фізичному виконанні запиту; клієнтський комп'ютер лише відсилає запит до серверної БД і отримує результат, після чого інтерпретує його необхідним чином і являє користувачу. Витягнуті дані транспортуються по мережі від сервера до клієнта. Тим самим,

кількість переданої по мережі інформації зменшується в багато разів. Отже, архітектуру клієнт-сервер, широко застосовують у сучасних додатках з застосуванням баз даних. 2.2. Порівняльний аналіз СУБД клієнт – серверної архітектури Клієнт-серверна СУБД – це система, що використовує технологію

Цитування: 0,01%

id: 17

«клієнт-сервер».

Такі СУБД складаються з клієнтської частини (яка входить до складу прикладної програми) і сервера. Клієнт-серверна СУБД дозволяє обмінюватися клієнту і сервера мінімально необхідними обсягами інформації. При цьому основна обчислювальна навантаження лягає на сервер. Клієнт може виконувати функції попередньої обробки перед передачею інформації сервера, але в основному його функції полягають в організації доступу користувача до сервера [36, с. 165]. На сучасному етапі існує багато СУБД, які мають клієнт-серверну архітектуру. Серед таких є [Firebird](#), [Interbase](#), [MS SQL Server](#), [Oracle](#), [PostgreSQL](#), [MySQL](#) та інші. Кожна з них має свої переваги та недоліки. Взагалі комерційні СУБД ([Oracle](#), [MS SQL Server](#) та ін. ) дорого коштують. Не кожне підприємство може придбати ліцензію на їх використання. Тому є доцільним застосовувати СУБД, які мають вільний код. Серед таких СУБД розглянемо найбільш популярні [Firebird](#), [PostgreSQL](#), [MySQL](#). [Firebird](#) – компактна, кроссплатформна, вільна система керування базами даних, що працює на [Linux](#), [Microsoft Windows](#) і різноманітних [Unix](#) платформах [6, с. 22]. Як перевагу [Firebird](#) можна відзначити архітектуру, що має багато версій (паралельна обробка

Обнаружен Плагиат: 0,38% <https://knowledge.allbest.ru/program...> + 2 ресурсов! id: 18

оперативних і аналітичних запитів: читають користувачі, що зчитують дані не блокують, тих хто записує), компактність (дистрибутив 10Mb), високу ефективність і потужну мовну підтримку для збережених процедур і тригерів. [Firebird](#) використовується в різних промислових системах (складські та господарські, фінансовий і державний сектори) з 2001 р. Це комерційно незалежний проєкт у вигляді вільної версії

[Interbase](#) 6.0. Серверною частиною є файл [fbserver.exe](#), який виконує функції суперсервера. Сервер [Firebird](#) - це програма, яка виконується на вузлу хоста в мережі, і має доступ до клієнтів з порту комунікації. Вона обслуговує запити безлічі клієнтів до безлічі баз даних, суперсервери ([Superserver](#)) є багатопоточним процесом, який запускає новий потік для кожного клієнта, що з'єднується [6, с. 36]. Робота сервера включає: управління зберіганням даних БД; управління транзакціями; підтримку блокування і статистики для БД; обробку запитів на додавання, зміна або видалення рядків, підтримку поточних і застарілих версій записів; підтримку метаданих БД; обслуговування клієнтських запитів на отримання результируючих даних та виконання збережених процедур; маршрутизацію повідомлень для клієнтів; підтримку кешованих даних. Клієнтську частину виконує бібліотека [fbclient.dll](#). Клієнтська бібліотека надає протокол з'єднання і транспортний рівень, які клієнтський додаток використовує для зв'язку з сервером. [MySQL](#) – вільна реляційна система управління базами даних. Розробку і підтримку [MySQL](#) здійснює корпорація [Oracle](#). [MySQL](#) є рішенням для малих і середніх додатків. Входить до складу серверів [WAMP](#), [AppServ](#), [LAMP](#) і в портативні збірки серверів [Denver](#), [XAMPP](#), [VertrigoServ](#). Зазвичай [MySQL](#) використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати [MySQL](#) в автономні програми. Гнучкість СУБД [MySQL](#) забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу [MyISAM](#), що підтримують повнотекстовий пошук, так і таблиці [InnoDB](#), що підтримують транзакції на рівні окремих записів. Більш того, СУБД [MySQL](#) поставляється із спеціальним типом таблиць [EXAMPLE](#), що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і [GPL](#)-ліцензуванню, в СУБД [MySQL](#) постійно з'являються нові типи таблиць. [MySQL](#) працює на великій кількості платформ: [AIX](#), [BSDi](#), [FreeBSD](#), [HP-UX](#), [Linux](#), [Mac OS X](#), [NetBSD](#), [OpenBSD](#), [OS / 2 Warp](#), [SGI IRIX](#), [Solaris](#), [SunOS](#), [SCO OpenServer](#), [UnixWare](#), [Tru64](#), [Windows 95](#), [Windows 98](#), [Windows NT](#), [Windows 2000](#), [Windows XP](#), [Windows Server 2003](#), [WinCE](#), [Windows Vista](#) і [Windows 7,8](#). [PostgreSQL](#) – вільна об'єктно-реляційна система управління базами даних. Існує в реалізаціях для безлічі [UNIX](#)-подібних платформ, включаючи [AIX](#), різні [BSD](#)-системи, [HP-UX](#), [IRIX](#), [Linux](#), [Mac OS X](#), [Solaris](#) / [OpenSolaris](#), [Tru64](#), [QNX](#), а також для [Microsoft Windows](#). [PostgreSQL](#) базується на мові [SQL](#) і підтримує багато з можливостей стандарту [SQL](#): 2003. У таблиці 2.1. наведено порівняльний аналіз СУБД клієнт-серверної архітектури, що вільно розповсюджуються. Таблиця 2.1 Порівняльний аналіз СУБД клієнт-серверної архітектури

Критерії, за якими здійснюється аналіз [Firebird MySQL PostgreSQL](#) Доступність – вільно розповсюджувана Так Так Так Багатоплатформна Так Так Так Максимальний розмір таблиці – 4 Гб ([ver.3.2](#)) 32 Тб Максимальний розмір бази даних Обмежено файловою системою Обмежено файловою системою відсутні Наявність багатOVERсійності ([MVCC](#)) Так частково Так Тест продуктивності движків (багатопоточний запит) 0,005с 14,5 ([MyISAM](#)) 1,385([InnoDB](#)) 3,1с Таким чином, аналіз СУБД довів, що всі розглянуті програмні продукти мають позитивні риси, а саме: вільно розповсюджуються, мають кросплатформну спрямованість. Але багатOVERсійність, як один з механізмів забезпечення одночасного конкурентного доступу до БД, СУБД [MySQL](#) підтримує частково: а саме тільки деякими засобами [MySQL – Maria](#), [InnoDB](#), [Falcon](#), [XtraDB](#). Багатопоточний тест продуктивності СУБД довів, що найбільш швидкісною є СУБД [Firebird](#) [5]. Тому саме цю СУБД беремо для розробки бази даних клієнт-серверної архітектури.

### 2.3. Проєктування моделі бази даних інформаційно-довідкової системи

Моделювання бази даних здійснюється на другому - проєктному етапі технології розробки інформаційно-довідкової системи. Розробка бази даних вимагає її ретельного проєктування, а саме розробки структури (схеми) бази даних. Проєктування БД передбачає наступні етапи [24, [с.7 – 11](#)]: Аналіз об'єкта автоматизації. Інфологічне проєктування. Даталогічне проєктування. Фізичне проєктування. Під час розробки технічного завдання до проєкту встановлено, що об'єктом автоматизації є навчального закладу

” Цитування: **0,09%**

id: **19**

"Лисичанська загальноосвітня школа I - III ступенів №5 Лисичанської міської ради Луганської області".

Інформаційно-довідкова система навчального закладу складається з інформації про навчальний процес: відомості про вчителя, учня, їх батьків, навчальний процес, відвідування занять учнями тощо. Інфологічне проєктування передбачає створення інформаційної моделі,

Обнаружен Плагиат: **0,12%** [http://ni.biz.ua/4/4\\_10/4\\_107317\\_eta...](http://ni.biz.ua/4/4_10/4_107317_eta...)

id: **20**

яка найбільш повно описує предметну галузь, без прив'язки до типу EOM і типу системних програмних засобів

[12, [с. 112](#)]. Однією з найбільш популярних засобів формалізованого представлення предметної галузі на етапі концептуального проєктування є [ER](#)- модель

” Цитування: **0,01%**

id: **21**

„сутність- зв'язок”

Обнаружен Плагиат: **0,14%** <https://ism.lpnu.ua/uk/content/system...>

id: **22**

([entity - relationship model](#)). Послідовність проведення [ER](#)-моделювання має наступне: визначення типів сутностей; визначення типів зв'язків; визначення атрибутів сутностей і зв'язків;

визначення потенційних і первинних ключів; перевірка моделі на відсутність надмірності. Під час аналізу було визначено наступні сутності та їх характеристики: Вчитель – містить дані про вчителів школи: ПІБ, дата народження, стать, паспортні дані, адреса проживання, ідентифікаційний код, документ про освіту, кваліфікація, категорія, адміністративна посада, дата прийому та звільнення з роботи. Предмет – дання про навчальні предмети, що викладаються учням: назва предмету, кількість годин, що відводиться на викладання, клас, де викладається цей предмет. Кваліфікація – відомості про кваліфікацію вчителів. Учень – містить наступну інформацію: номер особової справи учня, ПІБ, дата народження, стать, адреса проживання, національність, свідоцтво про народження, клас, де навчається, дата прийому та відрахування зі школи. Навчальний процес – відомості про учня, предмет та оцінки. Національність – відомості про національність. Стан здоров'я – дані про групу здоров'я, де учень займається фізичним виховання, довідка з медустанови про стан здоров'я, дата початку та закінчення дозволу на звільнення від занять з фізкультури. Батьки – дані про батьків учня: ПІБ батька, матері, опікуна (за наявності), кількість дітей у сім'ї, місце праці батька та матері. Факультатив – відомості про факультатив, учнів що вивчають цей факультатив, вчителів, що викладають. Навчальний заклад – відомості про вищий навчальний заклад, що закінчив вчитель. Клас – дані про клас, кількість учнів та класного керівника. Пропуски занять – інформація про кількість пропущених годин занять учнем за кожен місяць. Також було встановлено атрибути сутностей, що є набором певних

властивостей (табл. 2.2). Кожен атрибут був проаналізований на унікальність та можливість визначення у якості первинного або потенційного ключа, також визначені зовнішні ключі, що допоможуть встановити певні зв'язки між цими сутностями. Таблиця 2.2 Сутності та атрибути предметної галузі № Сутність Набір атрибутів та їх характеристика 1 Вчитель Ідент\_вчитель (первинний ключ), прізвище, ім'я, по\_батькові, дата\_народження, стать, серія\_паспорт, номер\_паспорт, ким\_виданий\_паспорт, адреса\_проживання, ідент\_налоговий\_код (потенційний ключ), категорія, серія\_диплом, номер\_диплом, ідент\_навчальний\_заклад (зовнішній ключ), ідент\_кваліфікація (зовнішній ключ), амін\_посада, дата\_прийом, дата\_звільнення. 2 Предмет Ідент\_предмет (первинний ключ), ідент\_вчитель (зовнішній ключ), ідент\_клас (зовнішній ключ), назва\_предмет, години\_предмет. 3 Кваліфікація Ідент\_кваліфікація (первинний ключ), назва\_кваліфікація. 4 Учень Іден\_учень (первинний ключ), номер\_особ\_справа (потенційний ключ), прізвище, ім'я, по\_батькові, дата\_народження, стать, адреса\_проживання, ідент\_національність (зовнішній ключ), документ\_особи, серія\_документ, номер\_документ, ідент\_сім'я (зовнішній ключ), ідент\_клас (зовнішній ключ), індив\_навчання, дата\_зарахування, дата\_відрахування. 5 Навчальний процес Ідент\_навчальний\_процес (первинний ключ), ідент\_учень (зовнішній ключ), ідент\_предмет (зовнішній ключ), оцінка\_1\_семестр, оцінка\_2\_семестр, оцінка\_рік. 6 Національність Ідент\_національність (первинний ключ), назва\_національність. 7 Стан здоров'я Ідент\_здоров'я (первинний ключ), ідент\_учень (зовнішній ключ), група\_здоров'я, довідка\_здоров'я, дата\_початок, дата\_закінчення. 8 Батьки Ідент\_сім'я (первинний ключ), ПІБ\_батька, ПІБ\_мати, опікун, кількість\_дітей, праця\_батько, праця\_мати. 9 Факультатив Ідент\_факультатив (первинний ключ), назва\_факультатив, ідент\_учень (зовнішній ключ), ідент\_вчитель (зовнішній ключ), кількість\_годин 10 Навчальний заклад Ідент\_навчальний\_заклад (первинний ключ), назва\_навчальний\_заклад, адреса\_навчальний\_заклад 11 Клас Ідент\_клас (первинний ключ), номер\_клас, буква\_клас, кількість\_учнів, ідент\_вчитель (зовнішній ключ) 12 Пропуски занять Ідент\_пропуски (первинний ключ), ідент\_учень (зовнішній ключ), місяць, кількість\_годин. За допомогою первинних та зовнішніх ключів було становлено зв'язки між сутностями. Ці зв'язки демонструють семантику взаємодії сутностей між собою. Встановлено 14 бінарних зав'язків між сутностями з показником кардинальності

Цитування: 0,02%

id: 23

«один до багатьох»

(1:M). Характеристика зв'язків наведена у табл. 2.3. Таблиця 2.3 Характеристика зв'язків між сутностями № Назва зв'язку Тип Ім'я батьківської таблиці Ім'я дочірньої таблиці 1 Національність 1:M Національність Учень 2 Пропуск 1:M Учень Пропуски занять 3 Зв'язки 1:M Батьки Учень 4 Стан здоров'я 1:M Учень Стан здоров'я 5 Результат 1:M Учень Навчальний процес 6 Навчається 1:M Клас Учень 7 Навчається 2 1:M Учень Факультатив 8 Викладається 1:M Клас Предмет 9 Оцінка 1:M Предмет Навчальний процес 10 Викладає 1 1:M Вчитель Предмет 11 Викладає 2 1:M Вчитель Факультатив 12 Керівник 1:M Вчитель Клас 13 ВНЗ 1:M Навчальний заклад Вчитель 14 Кваліфікація 1:M Кваліфікація Вчитель Наступним етапом було проведено аналіз на надмірність база даних. Встановлено, що зв'язки типу

Цитування: 0,02%

id: 24

«один до одного»,

які відповідають в моделі одному й тому ж концептуальному об'єкту, відсутні. Надлишкових зв'язків теж не встановлено. Також, аналіз довів, що наведена схема даних знаходиться у третій нормальній формі (3НФ), що дозволяє зробити висновок про відсутність надмірності й суперечності БД. Результатом інфологічного моделювання є побудова графічного зображення ER- моделі відповідно до нотацій Чена, яка подана у додатку А. Наступний етап проектування бази даних - створення даталогічної моделі, де проектується структура даних і будується даталогічна модель БД (схема БД), яка є описом логічної структури БД з урахуванням конкретної обраної СУБД (допустимі типи даних, найменування полів і таблиць, обмеження цілісності і т.п.) [24, с. 10]. У попередньому розділі було обґрунтовано вибір СУБД клієнт- серверної архітектури, тому для подальшого проектування буде застосовуватися FireBird ver 2.5.3 32 bit для операційної системи Windows. Аналіз атрибутів з точки зору даталогічного моделювання наведений у таблицях 2.4-2.15. Таблиця 2.4 Сутність Учень (Pupils) Назва атрибута Назва поля у FireBird Тип Первинний й/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Іден\_учень



[P\\_id\\_pupil Integer](#) первинний Так номер\_особ\_справа [P\\_case\\_number Varchar](#) 16 Так прізвище [P\\_last\\_name varchar](#) 32 Так ім'я [P\\_first\\_name Varchar](#) 32 Так по\_батькові [P\\_parent\\_name Varchar](#) 32 Так дата\_народження [P\\_birth\\_date Date](#) Ні стать [P\\_sex Varchar](#) 10 Ні чол адреса\_проживання [P\\_address Varchar](#) 128 Ні ідент\_національність [P\\_id\\_nationality integer](#) зовнішній Так документ\_особи [P\\_document\\_type Varchar](#) 16 Ні свідоцтво серія\_документ [P\\_doc\\_series Varchar](#) 16 Ні номер\_документ [P\\_doc\\_number Varchar](#) 16 Ні ідент\_сім'я [P\\_id\\_family Integer](#) зовнішній Так ідент\_клас [P\\_id\\_class Integer](#) зовнішній Так індив\_навчання [p\\_individual Varchar](#) 4 Так ні дата\_зарахування [P\\_beginning\\_date Date](#) Ні дата\_відрахування [P\\_ending\\_date date](#) Ні 01.01.1900 Таблиця 2.5 Сутність Вчитель ([Teachers](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язкове Так\ні За замовчуванням Ідент\_вчитель [T\\_id\\_teacher integer](#) первинний Так прізвище [T\\_last\\_name varchar](#) 32 Так ім'я [T\\_first\\_name varchar](#) 32 Так по\_батькові [T\\_parent\\_name varchar](#) 32 Так дата\_народження [T\\_birth\\_date date](#) Так стать [T\\_sex varchar](#) 10 Так чол серія\_паспорт [T\\_doc\\_series varchar](#) 16 Так номер\_паспорт [T\\_doc\\_number varchar](#) 16 Так кім\_виданий\_паспорт [T\\_doc\\_issue varchar](#) 64 Так адреса\_проживання [T\\_address varchar](#) 128 ні ідент\_налоговий\_код [T\\_identification\\_code varchar](#) 16 ні категорія [T\\_category varchar](#) 16 Так спеціаліст серія\_диплом [T\\_degree\\_series varchar](#) 16 ні номер\_диплом [T\\_degree\\_number varchar](#) 16 ні ідент\_навчальний\_заклад [T\\_id\\_education integer](#) зовнішній Так ідент\_кваліфікація [T\\_id\\_qualification integer](#) зовнішній Так амін\_посада [T\\_position varchar](#) 32 ні 01.01.1900 дата\_прийому [T\\_start\\_of\\_work Date](#) ні дата\_звільнення [T\\_end\\_of\\_work date](#) ні Таблиця 2.6 Сутність Кваліфікація ([Qualification](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_кваліфікація [Q\\_id\\_qualification integer](#) первинний Так назва\_кваліфікація [Q\\_name varchar](#) 50 Так Таблиця 2.7 Сутність Предмет ([Subject](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_предмет [S\\_id\\_subject Integer](#) первинний Так ідент\_вчитель [S\\_id\\_teacher integer](#) зовнішній Так ідент\_клас [S\\_id\\_class Integer](#) зовнішній Так назва\_предмет [S\\_name Varchar](#) 32 Так години\_предмет [S\\_hours integer](#) Так 0 Таблиця 2.8 Сутність Навчальний процес ([Progress](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_навчальний\_процес [P\\_id\\_progress integer](#) первинний Так ідент\_учень [P\\_id\\_pupil Integer](#) зовнішній Так ідент\_предмет [P\\_id\\_subject Integer](#) зовнішній Так оцінка\_1\_семестр [P\\_first\\_semester\\_mark Integer](#) Так 0 оцінка\_2\_семестр [P\\_second\\_semester\\_mark Integer](#) Так 0 оцінка\_рік [P\\_year\\_mark float](#) ні Таблиця 2.9 Сутність Національність ([Nationality](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_національність [N\\_id\\_nationality integer](#) первинний Так назва\_національність [N\\_nation varchar](#) зовнішній 32 Так Таблиця 2.10 Сутність Стан здоров'я ([Health](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве За замовчуванням Ідент\_здоров'я [H\\_id\\_health Integer](#) первинний так ідент\_учень [H\\_id\\_pupil Integer](#) зовнішній Так група\_здоров'я [H\\_group Varchar](#) 64 Так довідка\_здоров'я [H\\_doc\\_health Varchar](#) 128 Так дата\_початок [H\\_begin\\_date liberation date](#) Так дата\_закінчення [H\\_end\\_date liberation date](#) ні Таблиця 2.11 Сутність Батьки ([Family](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язкове За замовчуванням Ідент\_сім'я [F\\_id\\_family Integer](#) первинний так ПІБ\_батька [F\\_father Varchar](#) 128 Ні ПІБ\_мати [F\\_mather Varchar](#) 128 Ні опікун [F\\_guardian Varchar](#) 128 Ні кількість\_дітей [F\\_child\\_count Smallint](#) так 1 праця\_батько [F\\_father\\_work Varchar](#) 128 Ні праця\_мати [F\\_mather\\_work Varchar](#) 128 ні Таблиця 2.12 Сутність Факультатив ([Elective](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_факультатив [E\\_id\\_elective Integer](#) Первинний Так назва\_факультатив [E\\_name varchar](#) 25 так ідент\_учень [E\\_id\\_pupil Integer](#) Зовнішній Так ідент\_вчитель [E\\_id\\_teacher Integer](#) зовнішній Так кількість\_годин [E\\_hours Integer](#) Так Таблиця 2.13 Сутність Навчальний заклад ([Education](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_навчальний\_заклад [E\\_id\\_education Integer](#) первинний Так назва\_навчальний\_заклад [E\\_name Varchar](#) 64 Так адреса\_навчальний\_заклад [E\\_address varchar](#) 128 так Таблиця 2.14 Сутність Клас ([Class](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний/ зовнішній ключ Розмір Обов'язко ве Так\ ні За замовчуванням Ідент\_клас [C\\_id\\_class Integer](#) первинний Так номер\_клас [C\\_level Integer](#) Так буква\_клас [C\\_name Char](#) 2 Так кількість\_учнів [C\\_class\\_count integer](#) Так ідент\_вчитель [C\\_id\\_teacher Integer](#) зовнішній так Таблиця 2.15 Сутність Пропуски занять ([Absence](#)) Назва атрибута Назва поля у [FireBird](#) Тип Первинний й/ зовнішній ключ Розмір Обов'язкове Так\ ні За замовчуванням Ідент\_пропуски [A\\_id\\_absence Integer](#)



первинний Так ідент\_учень [A\\_id\\_pupil Integer](#) зовнішній Так місяць [A\\_manth Varchar](#) 16 Так кількість\_годин [A\\_hours integer](#) Так Даталогічна модель наведена на рис. 2.3 Рис. 2.3. Даталогічна модель БД Після проєктування бази даних здійснюється фізична реалізація моделі БД, яка складається зі



**Обнаружен Плагиат: 0,22%** [http://ni.biz.ua/4/4\\_10/4\\_107317\\_eta...](http://ni.biz.ua/4/4_10/4_107317_eta...)

id: 25

створення та завантаження даних в БД, розробку і налагодження прикладних програм для роботи з базою даних, оформлення документації. На цьому етапі будується фізична модель БД, яка описує способи фізичної організації даних.

Для зручного проєктування скористуємося програмним засобом [IBExpert](#) - GUI-оболонкою, призначеною для розробки та адміністрування баз даних [Firebird](#), а також для вибору та зміни даних, що зберігаються в базах. Для роботи з цим програмним засобом, в-першу чергу, потрібно встановити сервер бази даних [FireBird](#) та запустити його на виконання. Після встановлення [IBExpert](#) потрібно зареєструвати нову базу даних: користувач [SYSDBA](#), пароль – [masterkey](#). Потім – під'єднатися до цієї бази. Приклад створеної таблиці наведено на рис. 2.4 Рис. 2.4. Приклад таблиці [Teachers](#) Аналогічно створюються всі дванадцять таблиць. Потім за допомогою встановлення зовнішніх ключів визначаються зв'язки між таблицями. Таким чином, на цьому етапі було розроблено інфологічну, даталогічну модель даних на підставі яких було розроблено фізичну модель бази даних [school.gdb](#). Висновки до розділу У другому розділі було спроєктовано та розроблено базу даних системи інформаційно-довідкової системи навчального закладу. Під час цієї діяльності було досягнуто наступних результатів: розглянуто теоретичні підходи до технології клієнт-серверної архітектури. Встановлено, що використання клієнт-серверних СУБД є більш ефективними в порівнянні з локальними базами даних, завдяки забезпеченню широкого доступу до існуючих БД, підвищенню модульності програмного продукту; проаналізовано сучасні клієнт-серверні СУБД, які вільно розповсюджуються. Під час аналізу було визначено СУБД для розробки інформаційно-довідкової системи навчального закладу – [FireBird](#), яка дозволяє працювати з реляційними базами даних й вирішувати встановлені в технічному завданні задачі; під час методологічного проєктування визначено дванадцять сутностей, їх атрибути та зв'язки між ними, також розроблено інфологічну та даталогічну модель бази даних інформаційно- довідкової системи навчального закладу; за допомогою засобу адміністрування баз даних [IBExpert](#) розроблено фізичну модель бази даних [school.gdb](#). РОЗДІЛ 3.

## РОЗРОБКА ДОДАТКУ ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ ДЛЯ НАВЧАЛЬНИХ ЗАКЛАДІВ

3.1. Аналіз засобів та технологій розробки додатку Наступним шагом технології розробки ІДС є розробка програмного забезпечення (ПЗ) — це процес, направлений на створення і підтримку працездатності, якості і надійності програмного забезпечення, використовуючи технології, методології і практики з інформатики, управління проєктами, математики, інженерії і інших областей знання. Тому дуже важливо підійти до вибору інструментів та технологій розробки ПО. При виборі мови програмування були розглянуті дві мови – C++ і C#. У наведених нижче таблицях 3.1–3.9. відмічена наявність або відсутність тих чи інших можливостей в розглянутих мовах програмування C++ і C# [39]. В таблицях було зроблено порівняння з таких характеристик: парадигма, типізація, компілятор/інтерпретатор, управління пам'яттю, управління потоком обчислень, типи та структури даних, об'єктно-орієнтовані можливості та інші можливості. Умовні позначення в таблицях: + можливість існує - можливість відсутня +/- можливість підтримується не повністю -/+ можливість підтримується дуже обмежено Таблиця 3.1 Парадигми Мови програмування

Мови	Можливості
Імперативна	Об'єктно- орієнтована
Функціональна	Рефлексивна
Узагальнене програмування	Логічна Декларативна Розподілена
C++	+ + +/- - + - - +/- C#
+ + +/- -/+ + - -/+ -/+	Таблиця 3.2 Типізація
Можливості C++ C#	Статична типізація + + Динамічна типізація - +
Явна типізація + +	Неявна типізація -/+ -/+
Неявне приведення типів без втрати даних + +	Неявне приведення типів з втратою даних + -
Неявне приведення типів в неоднозначних ситуаціях + +	Аліаси типів + +
Вивід типів змінних з ініціалізатора +/- +	Вивід типів змінних з використання +/- +
Вивід типів - аргументів при виклику методу + +	Вивід сигнатури для локальних функцій +/- -

Таблиця 3.3 Керування пам'яттю Таблиця 3.4 Компілятор/інтерпретатор Таблиця 3.5 Керування потоком обчислень Таблиця 3.6 Типи та структури даних Таблиця 3.7 Об'єктно-орієнтовані можливості Таблиця 3.8 Функціональні можливості Таблиця 3.9 Різні можливості Мови програмування

Можливості Макроси Шаблони / [Generics](#) Підтримка [Unicode](#) в ідентифікаторах Перевантаження функцій Іменовані параметри Значення параметрів за замовчуванням Локальні функції Зіставлення

із зразком Контрактне програмування Наявність бібліотек для роботи з графікою і мультимедіа ([OpenGL](#) / [WebGL](#) / [OpenML](#) , [OpenAL](#) , [DirectX](#)) C++ + + + + - + + - + C# - + + + + + / - + / - + + Проведений аналіз мов програмування свідчить, що мови програмування C++ та C# є дуже потужними засобами створення програмних додатків. Можливості цих мов дуже схожі, але є ряд відмінностей. Фундаментальні основи переваг C++ в можливості писати код, який буде виконуватися безпосередньо процесором, і можливості прямої роботи з пам'яттю. Але в нашому проєкті це не принципово. Перевагою C# є те, що під [Windows](#), C#-проєкти швидше розроблювати та помітно зручніше налагоджувати. Крім того, одною з найбільш актуальних мов програмування, на якій ведеться розробка клієнтських додатків для баз даних з використанням технології [ADO.NET](#), є C# [37, с. 55]. Ця мова найбільш оптимізована розробником [Microsoft](#) для роботи з цією технологією. Саме це обумовлює вибір C# як інструменту розробки інформаційно- довідкової системи навчального закладу, де однією з основним частин є клієнт-серверна база даних. В технології [Microsoft.NET](#) створена достатня гнучка і ефективна модель доступу до даних - [ADO.NET](#), що надає розробникам набір об'єктів, на основі яких можна створювати додатки будь-якого масштабу (від локальних до глобальних). В цій технології закладена можливість роботи програми в стані

” Цитування: 0,01%

id: 26

"розриву"

з'єднання з базою даних [37, с. 32]. Додатки підключаються до бази даних тільки на невеликий проміжок часу. З'єднання встановлюється тільки тоді, коли клієнт з віддаленого комп'ютера запрошує на сервері дані. В об'єктній моделі [ADO.NET](#) можна виділити декілька класів (рис. 3.1) та кілька рівнів. Рис. 3.1. Ієрархія класів [ADO .Net](#) Рівень даних. Це базовий рівень, на якому розташовуються самі дані (наприклад, таблиці бази даних [Firebird](#)). На даному рівні забезпечується фізичне зберігання інформації на магнітних носіях і маніпуляція з даними на рівні вихідних таблиць (вибірка, сортування, додавання, видалення, оновлення). Рівень бізнес-логіки. Це набір об'єктів, що визначають, з якою базою даних належить встановити зв'язок і які дії необхідно буде виконати з даними, що містяться в неї. Для встановлення зв'язку з базами даних використовується об'єкт [FbDataConnection](#). Для зберігання команд, що виконують будь-які дії над даними, використовується об'єкт [FbDataAdapter](#). Для зберігання результатів вибірки використовується об'єкт [DataSet](#). Рівень додатку. Це набір об'єктів, що дозволяють зберігати і відображати дані на комп'ютері кінцевого користувача. Для зберігання інформації використовується об'єкт [DataSet](#), а для відображення даних є досить великий набір елементів управління ([DataGrid](#), [TextBox](#), [ComboBox](#), [Label](#) і т. Д.). Таким чином, з урахуванням вибору мови програмування C# та технології доступу до даних [ADO.NET](#), виберемо середовище [Visual Studio](#). 3.2. Розробка та реалізація програмного додатку Під час розробки C#-додатків під [FireBird](#), особливу увагу потрібно приділити підключенню до бази даних. Це можна зробити за допомогою компоненту [ADO.NET Data Provider](#). Компонент [ADO.NET Data Provider](#) володіє всіма необхідними функціональними можливостями, які можуть бути затребувані при підключенні до [FireBird](#). Компонент дозволяє змінювати параметри авторизації, підключення – дозволяє вибрати локальний або віддалений сервер, здійснювати налаштування порту і кодування, в якій буде здійснюватися робота з даними. Завантажити компонент можна на офіційному сайті [FireBird](#). У проєкт додатку [Visual Studio](#) досить підключити файл [FirebirdSql.Data.FirebirdClient.dll](#). Також в додаток необхідно додати рядок з вказаним простором імен: [using FirebirdSql.Data.FirebirdClient](#); В додатку, що розроблюється, підключення до бази даних здійснюється за допомогою розробленого класу [FirebirdInterface](#) (додаток В). Цей клас розроблено з метою створення додаткового рівня абстракції над об'єктами [ADO.NET](#) і [ADO.NET Data Provider](#). У таблиці 3.10 наведено основні методи, що реалізовані у цьому класі. Таблиця 3.10 Основні методи, що реалізовані у класі [FirebirdInterface](#) № Ім'я методу Призначення методу 1 [InitConnectString](#) Повертає сформований




Обнаружен Плагиат: 0,18% <https://psychologer.com.ua/manipuliat...> + 5 ресурсів!

id: 27

рядок з'єднання з базою даних на підставі переданих у функцію параметрів 2 [openDataBase](#) Відкриває з'єднання з базою даних, у випадку успішного з'єднання функція повертає значення

” Цитування: 0,01%

id: 28

«істина»	
 <b>Обнаружен Плагиат: 0,15%</b> <a href="https://psychologer.com.ua/manipuliat...">https://psychologer.com.ua/manipuliat...</a> + 5 ресурсів!	id: 29
3 <a href="#">closeDataBase</a> Закриває з'єднання з базою даних 4 <a href="#">isOpen()</a> Перевірка з'єднання з базою даних, у випадку відкритого з'єднання функція повертає значення	
 Цитування: <b>0,01%</b>	id: 30
«істина»	
 <b>Обнаружен Плагиат: 0,4%</b> <a href="https://psychologer.com.ua/manipuliat...">https://psychologer.com.ua/manipuliat...</a> + 5 ресурсів!	id: 31
5 <a href="#">fillSchema</a> Функція ініціалізує властивості <a href="#">SelectCommand</a> об'єкта <a href="#">FbDataAdapter</a> 6 <a href="#">fill</a> Здійснює заливку даних з фізичної бази даних в об'єкт <a href="#">DataTable</a> 7 <a href="#">save</a> Здійснює збереження змін у БД В розробленому клієнтському додатку з'єднання з базою даних здійснюється у діалоговому вікні <a href="#">LoginDialog</a> , що наведено у рис. 3.2. Рис. 3.2. Діалогове вікно <a href="#">LoginDialog</a> Підключення до бази даних	
здійснюється у методі <a href="#">Accept_Click</a> кнопки	
 Цитування: <b>0,01%</b>	id: 32
«Увійти».	
Код методу наведено нижче: <a href="#">private void Accept_Click(object sender, EventArgs e) {</a> <a href="#">fbObject.InitConnectionString(ConnectItems.library, ConnectItems.database,</a> <a href="#">ConnectItems.users[userList.SelectedIndex], password.Text, ConnectItems.role,</a> <a href="#">ConnectItems.charset, ConnectItems.port); if (!fbObject.openDataBase()) result =</a> <a href="#">DialogResult.Retry; else result = DialogResult.OK; }</a> В цьому методі формується рядок підключення до БД за допомогою виклику функції <a href="#">InitConnectionString</a> : <a href="#">public string</a> <a href="#">InitConnectionString(string clientLibrary, string database, string userID, string password, string role,</a> <a href="#">string charset, int port) { FbConnectionStringBuilder connectString = new</a> <a href="#">FbConnectionStringBuilder(); connectString.ClientLibrary = clientLibrary; connectString.Database</a> <a href="#">= database; connectString.UserID = userID; connectString.Password = password;</a> <a href="#">connectString.Charset = charset; connectString.Role = role; connectString.Port = port;</a> <a href="#">connect.ConnectionString = connectString.ConnectionString; connect.InfoMessage += new</a> <a href="#">FbInfoMessageEventHandler(OnInfoMessage); return connect.ConnectionString; }</a> Функція <a href="#">openDataBase</a> відкриває з'єднання з БД: <a href="#">public bool openDataBase() { try { if</a> <a href="#">(connect.ConnectionString.Length 0) connect.Open(); else throw new System.Exception(</a>	
 Цитування: <b>0,04%</b>	id: 33
"Строка соединения с БД не корректна."	
<a href="#">); } catch (System.Exception Except) { MessageBox.Show(Except.Message,</a>	
 Цитування: <b>0,01%</b>	id: 34
"Ошибка",	
<a href="#">MessageBoxButtons.OK, MessageBoxIcon.Error); connect.Close(); return false; } return</a> <a href="#">connect.State == System.Data.ConnectionState.Open ? true : false; }</a> Інтерфейс клієнтського додатку складається з головної форми, яка подана у рис 3.3. Рис 3.3. Головна форма клієнтського додатку системи До складу форми входить головне меню та панель інструментів, з лівого боку розташовано ієрархічне деревовидне меню, з правого – головний елемент відображення змісту таблиць бази даних. Головне меню складається з пунктів:	
 Цитування: <b>0,01%</b>	id: 35
«База даних»	
– налаштування, від'єднання від БД, вихід з системи;	
 Цитування: <b>0,01%</b>	id: 36
«Довідники»	
– інформація довідникового характеру, що викликає форми: гілка	
 Цитування: <b>0,01%</b>	id: 37
«Учень»	
–	

Цитування: <b>0,01%</b> «Національність»,	id: <b>38</b>
Цитування: <b>0,02%</b> «Відомості про батьків»,	id: <b>39</b>
Цитування: <b>0,01%</b> «Стан здоров'я»	id: <b>40</b>
; гілка	
Цитування: <b>0,01%</b> «Вчитель»	id: <b>41</b>
–	
Цитування: <b>0,02%</b> «Вищий навчальний заклад»,	id: <b>42</b>
Цитування: <b>0,01%</b> «Кваліфікація»	id: <b>43</b>
;	
Цитування: <b>0,01%</b> «Запити»	id: <b>44</b>
– вибіркова інформація: гілка	
Цитування: <b>0,01%</b> «Учень»	id: <b>45</b>
–	
Цитування: <b>0,02%</b> «Запит за ПІБ»,	id: <b>46</b>
Цитування: <b>0,01%</b> «Успішність учня»,	id: <b>47</b>
Цитування: <b>0,02%</b> «Відвідування занять учнем»,	id: <b>48</b>
Цитування: <b>0,03%</b> «Запит за датою народження»	id: <b>49</b>
; гілка	
Цитування: <b>0,01%</b> «Вчитель»	id: <b>50</b>
–	
Цитування: <b>0,02%</b> «Запит за ПІБ»,	id: <b>51</b>
Цитування: <b>0,02%</b> «Запит за категорією»,	id: <b>52</b>
Цитування: <b>0,01%</b> «Навантаження викладача»	id: <b>53</b>
;	
Цитування: <b>0,01%</b> «Звіти»	id: <b>54</b>
– інформація звітнього характеру:	
Цитування: <b>0,04%</b> «Відомість про склад учнів класу»,	id: <b>55</b>

Цитування: 0,02% id: 56

«Картка навантаження вчителя»,

яку можна роздрукувати. В головній формі відбиваються дані таблиць [CLASS](#) (класи), [PUPILS](#) (учні), [SUBJECT](#) (предмети), [PROGRESS](#) (успішність), [ABSENCE](#) (пропуски занять), [ELECTIVE](#) (факультативи). При виборі назви класу в правій частині форми відображаються скороченні дані про учнів з таблиці [PUPILS](#) з ознакою приналежності до певного класу (за полем [P\\_ID\\_CLASS](#)). Повну інформацію про окремого учня можна отримати при натисканні мишею на ПІБ цього учня. При цьому відкривається форма

Цитування: 0,01% id: 57

«Учень»,

приклад якої подано у рис. 3.4. Ця форма має можливості не тільки відображення інформації про учня, а й додавання, видалення, оновлення даних. Додавання нового учня здійснюється за допомогою кнопки

Цитування: 0,01% id: 58

«Додати запис»,

що розташована на панелі інструментів головної форми. При цьому відкривається аналогічна форма

Цитування: 0,01% id: 59

«Учень»,

але з порожніми полями. Аналогічні принципи закладено у роботі з гілкою

Цитування: 0,01% id: 60

«Вчителі».

Форма яка активується –

Цитування: 0,02% id: 61

«Особова картка вчителя».

Рис. 3.4. Приклад форми

Цитування: 0,02% id: 62

«Особова картка вчителя»

Особливістю побудови головної форми є те, що об'єкт управління [DataGridView](#) має пристосовувати відображення даних в сітці в залежності від структури таблиць, що зв'язується з цим об'єктом. В об'єкті управління [DataGridView](#) повинні розміщуватися дані з різних зв'язаних таблиць, ховатися різні стовбці, бути різними заголовки стовбців. Для розв'язання цього завдання було застосовано поліморфізм, а саме створено базовий клас [PlymorphicWidget](#), від якого створено похідні класу [PupilsTable](#), [SubjectTable](#), [ProgressTable](#), [AbsenceTable](#), [TeacherTable](#), [ElecNiveTable](#), в яких перевизначається віртуальний метод [adapt](#) базового класу, що формує структуру відображуваної таблиці. Приклад коду базового класу [PlymorphicWidget](#) й похідного від нього класу [SubjectTable](#) наведено нижче:

```
public class PlymorphicWidget { protected TreeNode nodeData; protected MainWindow widget;
public PlymorphicWidget(MainWindow window) { widget = window; } protected virtual void
view(){} public virtual void adaptFilterA() { widget.mainGrid.DataSource = null; } public virtual
void adaptFilterB() { widget.mainGrid.DataSource = null; } public virtual void adapt() {
widget.mainGrid.DataSource = null; } public virtual void showEditDialog() { } public virtual void
insertDialog() { } public void deleteRow() { if (widget.mainGrid.DataSource != null) { if
(DialogResult.Yes == MessageBox.Show(
```

Цитування: 0,01% id: 63

"Видалити запис?",

Цитування: 0,01% id: 64

"Увага",

```
MessageBoxButtons.YesNo, MessageBoxIcon.Question)) { DataRow row =
widget.findCurrentRow(widget.mainGrid); if (row != null) row.Delete(); else MessageBox.Show(
```

Цитування: 0,02% id: 65



"Рядок не знайдено",	
” Цитування: 0,01%	id: 66
"Помилка",	
MessageBoxButtons.OK, MessageBoxIcon.Error); } } else MessageBox.Show(	
” Цитування: 0,02%	id: 67
"Таблиця не визначена!",	
” Цитування: 0,01%	id: 68
"Увага",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); } public virtual void dataSave() { } public virtual void dataRefresh() { } public TreeNode Node { get { return nodeData; } set { nodeData = value; } } } Далі наведено перевизначений метод <code>adapt</code> похідного класу <code>SubjectTable</code> : <code>public class SubjectTable : PlymorphicWidget { public override void adapt(MainWindow widget) { widget.mainGrid.DataSource = 0; DataView view = widget.subjectView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; widget.initViewRelation(</code>	
” Цитування: 0,01%	id: 69
" <code>relation</code> 11",	
” Цитування: 0,01%	id: 70
"Вчитель",	
1, 5); <code>widget.mainGrid.Columns[0].Visible = false; widget.mainGrid.Columns[1].Visible = false; widget.mainGrid.Columns[2].Visible = false; widget.mainGrid.Columns[3].HeaderText =</code>	
” Цитування: 0,01%	id: 71
"Назва предмету"	
; <code>widget.mainGrid.Columns[3].Width = 230; widget.mainGrid.Columns[4].HeaderText =</code>	
” Цитування: 0,01%	id: 72
"Кількість годин"	
; <code>widget.mainGrid.Columns[4].Width = 230; } public override void showDialog(MainWindow widget) { MessageBox.Show(</code>	
” Цитування: 0,01%	id: 73
" <code>SubjectTable</code> "	
); } } Приклад вікна довідника	
” Цитування: 0,01%	id: 74
«Начальний заклад»	
наведено у рис. 3.5: Рис. 3.5. Приклад вікна довідника	
” Цитування: 0,01%	id: 75
«Начальний заклад»	
Навігація у вікні реалізована за допомогою <code>bindingNavigator</code> , до стандартних компонентів якого було додано кнопку	
” Цитування: 0,01%	id: 76
«Зберегти».	
Код методу при натисканні на кнопку наведено нижче: <code>private void saveToolStripButton_Click(object sender, EventArgs e) { this.Validate(); bindingSource_ed.EndEdit(); if (!fb.save(</code>	
” Цитування: 0,01%	id: 77
" <code>EDUCATION</code> "	
)) { <code>MessageBox.Show(</code>	
”	

"Збереження не виконано"	id: 78
); } } В цьому методі викликається метод <code>save</code> об'єкта <code>fbObject</code> класу <code>FirebirdInterface</code> : <code>public bool save(string tableName) { DataTable currentTable = dataTable(tableName); if (currentTable.GetChanges() == null) return false; try { tableAdapter(tableName).Update(currentTable.GetChanges()); currentTable.AcceptChanges(); return true; } catch (Exception) { currentTable.RejectChanges(); return false; } }</code> У меню	
Цитирования: 0,02%	id: 79
«Запити»	
здійснюється формування вибірки даних за параметром. Приклад вікна параметру для вибірки даних навантаження викладача за вказаним ПІБ наведено у рис. 3.6. Рис. 3.6. Приклад вікна параметру ПІБ викладача Результат формування вибірки даних за вказаним параметром наведено у рис. 3.7. Рис. 3.7. Вибірка даних за параметром Вибірка даних за параметром здійснюється за допомогою віртуальних методів, що перевизначаються: <code>adaptFilterA()</code> та <code>adaptFilterB()</code> . Приклад коду наведено нижче: <code>public override void adaptFilterA() { widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if (widget.pupilsParamDlg.ShowDialog() == DialogResult.OK) {string filterParemetr =</code>	
Цитирования: 0,01%	id: 80
"P_NAME = "	
+	
Цитирования: 0,01%	id: 81
""	
+ <code>widget.pupilsParamDlg.cbPupilsName.Text</code> +	
Цитирования: 0,01%	id: 82
""	
; <code>DataGridView view = new DataGridView(widget.fbObject.dataTable(</code>	
Цитирования: 0,01%	id: 83
"PUPILS"	
), <code>filterParemetr</code> ,	
Цитирования: 0,01%	id: 84
"P_ID_PUPIL",	
<code>DataGridViewRowState.CurrentRowState); if (view.Count == 0) MessageBox.Show(</code>	
Цитирования: 0,03%	id: 85
"Дані відсутні в базі",	
Цитирования: 0,01%	id: 86
"Увага!",	
<code>MessageBoxButtons.OK, MessageBoxIcon.Warning); else { widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view(); }</code> } } У змінній <code>filterParemetr</code> формується параметр, за яким фільтруються дані, і який застосовується у <code>DataGridView</code> . Цей елемент застосовувався для фільтрації даних. Меню	
Цитирования: 0,01%	id: 87
«Звіти»	
формує документи на друк. На рис. 3.8 наведено приклад друку документу. Рис. 3.8. Приклад форми друку Після розробки клієнтського додатку було створено інсталяційний пакет, до складу якого входить сервер <code>FireBird</code> , скомпільований додаток, файл бази даних <code>school.gdb</code> . Таким чином, було розроблено інформаційно-довідкову систему навчального закладу	
Цитирования: 0,06%	id: 88
"Дніпровська загальноосвітня школа I - III ступенів №5"	
на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та	

завданням. 3.3. Тестування додатка Об'єкт і цілі тестування Розроблюваний програмний комплекс повинен формувати інформаційну базу інформаційно-довідкової системи навчального закладу. Програмний проєкт повинен забезпечувати: Введення (додавання) і відображення інформації з бази даних. Пошук даних по введеному параметру. Видалення, редагування даних; Ведення довідників, які будуть використовуватися базою даних. Формування звітних форм. Методи тестування В ході тестування будемо використовувати ручний метод. Ручне тестування ([manual testing](#)) - частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно проводиться тестувальниками або звичайними користувачі шляхом моделювання можливих сценаріїв дії користувача. Етапи тестування Тестування включало п'ять етапів, які відповідають цілям тестування. Введення (додавання) і відображення інформації з бази даних. Завдання: додати запис про предмети. В ході тестування заповнена форма

Цитування: 0,04%

id: 89

«Відомості про предмети, що викладаються».

Перегляд нового запису поданий з права (рис. 3.9). Рис. 3.9. Результати тестування введення (додавання) і відображення інформації в БД Аналогічно проведено тестування на додавання та перегляд даних в інших таблицях. Висновок: операції введення і відображення даних в програмному проєкті відповідають вимогам. Пошук даних за введеним параметром. Завдання: відфільтрувати таблицю Вчителя за параметром

Цитування: 0,02%

id: 90

«Кушнір Іван Васильович»,

результат пошуку подана рис. 3.10. Рис. 3.10. Результати тестування пошуку даних за введеним параметром Аналогічно протестований інші пошуки за параметром. Висновок: операції пошуку даних по введеному параметру в проєкті Програми відповідають вимогам. 3.Видалення, редагування даних. Завдання: видалити і оновити дані про вчителя. В ході тестування була виділена запис на видалення з прізвищем Кушнір Іван Васильович. Після натискання на панелі інструментів кнопки видалення з'являється діалогове вікно - запит на видалення запису (рис. 3.11). Рис. 3.11. Результати тестування видалення, редагування даних В результаті підтвердження запис видаляється з бази даних. Аналогічно протестовано видалення і редагування записів в формі Учні, Предмети, Факультативи. Висновок: операції видалення і редагування даних в програмному проєкті відповідають вимогам. 4.Ведення довідників, які будуть використовуватися базою даних. Як довідників виступають таблиці: кваліфікація вчителя, освіта, національність учня, батьки, стан здоров'я. Завдання: ввести дані в довідник кваліфікація вчителя. Виберемо в меню Довідники: вчителя- кваліфікація. При натисканні кнопки додавання запису внизу списку з'являється порожній запис для введення даних (рис. 3. 12). Рис. 3.12. Результати тестування ведення довідників Аналогічно протестовано додавання даних в інші довідники. Висновок: операції додавання даних в довідники в проєкті Програми відповідають вимогам. 5.Формування звітних форм. Завдання: сформувати звіти: склад учнів за списком, картка навантаження вчителя. В ході тестування за допомогою меню Звіти були обрані зазначені форми. Приклад звіту відомості успішності учнів за семестр показаний на рис.3.13. Рис. 3.13. Результати тестування звітів Висновок: операції формування звітних форм в проєкті Програми відповідають вимогам. Висновки до розділу Під час розробки клієнтського додатку інформаційно-довідкової системи навчального закладу було проаналізовано можливості технології [ADO.NET](#), що має набір об'єктів, на основі яких можна створювати додатки будь-якого масштабу. Встановлено, що найбільш оптимізованою мовою для роботи з цією технологією є [C #](#), тому для розробки додатку було обрано середовище [Visual Studio](#). За допомогою [ADO.NET Data Provider](#) в додатку здійснено підключення до бази даних [FireBird](#). Згідно встановлених етапів технології проєктування інформаційних систем було розроблено інформаційно-довідкову систему навчального закладу:

Цитування: 0,06%

id: 91

"Дніпровська загальноосвітня школа I - III ступенів № 5 "

на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням. ВИСНОВКИ Під час виконання дипломної роботи було проведено аналіз технології розробки інформаційно-довідкових систем для навчальних закладів із застосуванням клієнт-серверних СУБД. Під час дослідження отримані наступні висновки:

досліджено теоретичні основи інформаційно-довідкових систем: розглянуто поняття ІДС, класифікація, структуру та складові підсистеми; проаналізовано засоби проектування та етапи створення інформаційно-довідкових систем. Встановлено, що технологія розробки ІДС складається з наступних етапів: передпроектний, проектний (етапи технічного та робочого проектування), впровадження, супровід і аналіз функціонування; визначені завдання, вимоги та обґрунтування розробки та розроблено технічне завдання на розробку ІДС; розглянуто теоретичні підходи до технології клієнт-серверної архітектури, проаналізовано сучасні клієнт-серверні СУБД, які вільно розповсюджуються. Для розробки ІДС обрано [FireBird](#); під час методологічного проектування розроблено інфологічну та даталогічну модель бази даних системи та розроблено фізичну модель бази даних [school.gdb](#); під час розробки клієнтського додатку ІДС були обґрунтовано застосовані технологія [ADO.NET](#), мова програмування [C #](#), середовище розробки [Visual Studio](#); розроблено інформаційно-довідкову систему навчального закладу:

**Цитування: 0,06%** id: 92

"Дніпровська загальноосвітня школа I - III ступенів №5"

на базі клієнт-серверної архітектури, яка повністю відповідає висунутим вимогам та завданням. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ Анісімов А.В.

**Обнаружен Плагиат: 0,2%** <https://asnk.kpi.ua/docs/syllabus/Syl...> + 2 ресурсов! id: 93

Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. — Київ. – 2017. – 110 с.

**Обнаружен Плагиат: 0,17%** <https://elib.chdtu.edu.ua/e-books/374...> + 2 ресурсов! id: 94

Антоненко В. М. Сучасні інформаційні системи і технології: управління знаннями : навч. посібник / В. М. Антоненко, С. Д. Мамченко, Ю. В. Рогушина.

– Ірпінь : Нац. університет ДПС України, 2016. – 212 с. Воронін А. М. Інформаційні системи прийняття рішень: навчальний посібник. / Воронін А. М., Зіатдінов Ю. К., Климова А. С. — К. : НАУ-друк, 2009. — 136с. Галузинський Г. П. Інформаційні системи у бізнесі. Практикум для індивідуальної роботи: навч.- метод. посіб. для самост. вивч. Дисципліни. / Галузинський Г. П., Денісова О. О., Писаревська Т. А. — К. : КНЕУ, 2008. — 524с. Годун

**Обнаружен Плагиат: 0,09%** <http://eprints.zu.edu.ua/33169/1/Инф...> + 2 ресурсов! id: 95

В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М.

Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.

**Обнаружен Плагиат: 0,1%** <https://uk.wikipedia.org/wiki/Информа...> id: 96

Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки

**Цитування: 0,01%** id: 97

«Транспортні технології»

**Обнаружен Плагиат: 0,12%** <http://eprints.zu.edu.ua/33169/1/Инф...> + 2 ресурсов! id: 98

/ О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2010. – 222 с.

Інформаційні системи в економіці : навч. посібник / Пономаренко В. С., Золотарьова І. О., Бутова Р. К. та ін. – Х. : Вид. ХНЕУ, 2011. – 176 с. Інформаційні системи в промисловості : навчальний посібник / Л. О. Добровольська, О. О. Черевко. – Маріуполь : ПДТУ, 2014. – 238 с. Інформаційні системи в сучасному бізнесі : навчальний

**Обнаружен Плагиат: 0,22%** <http://eprints.zu.edu.ua/33169/1/Инф...> id: 99

посібник / В. С. Пономаренко, І. О. Золотарьова, Р. К. Бутова та ін. – Х. : Вид. ХНЕУ, 2011. – 484 с. Калінеску Т.В. Інформаційні системи і технології в

оподаткуванні: навч. посіб. / Т.В. Калінеску, Г.С. Ліхоносова, О.М. Антіпов. – Луганськ: вид-во СНУ ім. В. Даля, 2011. – 407 с. Клімушин П. С. Інформаційні системи та технології в економіці : навч. посіб. / П. С.Клімушин, О.В. Орлов, А.О. Серенок. — Х. : Вид-во ХарPI НАДУ

**Цитування: 0,01%** id: 100


«Магістр»,

2011. – 448 с. Морзе Н.В. Інформаційні системи. Навч. посібн. /за наук. ред. Н. В. Морзе; Морзе Н.В., Піх О.З. – Івано-Франківськ,

» Цитування: <b>0,01%</b>	id: <b>101</b>
«ЛілеяНВ»,	
– 2015. – 384 с. Павлиш В. А. Основи інформаційних технологій і систем: Навчальний посібник. / Павлиш В. А., Гліненко Л. К. - Львів: Видавництво Львівської політехніки, 2013. – 500 с. Пістунов І. М. Інформаційні системи в фінансово-кредитних установах [текст] навчальний посібник / І. М. Пістунов, Т. В. Борщ. – К.:	
» Цитування: <b>0,02%</b>	id: <b>102</b>
«Центр учбової літератури»,	
2013. – 234 с. Сендзюк М.	
Ⓢ Обнаружен Плагиат: <b>0,17%</b> <a href="http://eprints.zu.edu.ua/33169/1/Инф...">http://eprints.zu.edu.ua/33169/1/Инф...</a> + 3 ресурсов!	id: <b>103</b>
А. Інформаційні системи і технології в економіці: навч.-метод. посіб. для самост. вивч. дисципліни / М.А. Сендзюк; М-во освіти і науки України, ДВНЗ	
» Цитування: <b>0,05%</b>	id: <b>104</b>
“Київ. нац. екон. ун-т ім. В. Гетьмана”.	
Ⓢ Обнаружен Плагиат: <b>0,33%</b> <a href="http://eprints.zu.edu.ua/33169/1/Инф...">http://eprints.zu.edu.ua/33169/1/Инф...</a> + 3 ресурсов!	id: <b>105</b>
– К. : КНЕУ, 2010. – 68 с. Сікірда Ю. В. Інформаційні системи і технології в управлінні зовнішньоекономічною діяльністю : конспект лекцій / Ю. В. Сікірда, А. В. Залевський. – Кіровоград : Видавництво КЛА НАУ, 2013. – 177 с. Соколов В.Ю. Інформаційні системи і технології	
: Навч. посіб. / Соколов В.Ю. – К. : ДУКТ, 2010. – 138 с.	
Ⓢ Обнаружен Плагиат: <b>0,12%</b> <a href="https://www.researchgate.net/publica...">https://www.researchgate.net/publica...</a>	id: <b>106</b>
Федотова Е.Л. Информационные технологии и системы: учеб. пособие / Е.Л. Федотова. – М.: ИД	
» Цитування: <b>0,01%</b>	id: <b>107</b>
“ФОРУМ”:	
ИНФРА-М, 2014. – 352 с. Шило С. Г. Інформаційні системи та технології : навчальний посібник / С. Г. Шило, Г. В. Щербак, К. В. Огурцова. – Х. : Вид. ХНЕУ, 2013. – 220 с. Юринець В. Є. Інформаційні системи управління персоналом, діловодства і документообігу: навч. посіб. / Юринець В. Є., Юринець Р. В. – Л. : Тріада плюс, 2008. – 628 с.	
Ⓢ Обнаружен Плагиат: <b>0,11%</b> <a href="http://www.repository.hneu.edu.ua/bi...">http://www.repository.hneu.edu.ua/bi...</a> + 3 ресурсов!	id: <b>108</b>
Пасічник В.В., Резніченко В. А. Організація баз даних та знань.–К. : Видавнича група	
» Цитування: <b>0,01%</b>	id: <b>109</b>
«ВНУ»,	
2006. –384 с.	
Ⓢ Обнаружен Плагиат: <b>0,17%</b> <a href="https://asn.kpi.ua/docs/syllabus/Syl...">https://asn.kpi.ua/docs/syllabus/Syl...</a> + 2 ресурсов!	id: <b>110</b>
Бази даних в інформаційних системах : підруч. / В. І. Гайдаржи, І. В. Изварін. - К. : Ун-т Україна, 2018. - 418 с.	
Шпеник Т.Б. Організація баз даних. Логічне проектування та робота з віддаленими базами даних. Методичні вказівки і завдання до лабораторних робіт для студентів 2-го курсу інженерно-технічного факультету спеціальності 123 –	
» Цитування: <b>0,01%</b>	id: <b>111</b>
«Комп’ютерна інженерія».	
– Ужгород:	
» Цитування: <b>0,01%</b>	id: <b>112</b>
«АУТДОРШАРК»,	
2021. – 79 с. Балик Н.Р., Мандзюк В.І. Бази даних <b>MySQL</b> : Навчальний посібник. — Тернопіль:	
» Цитування: <b>0,03%</b>	id: <b>113</b>
«Навчальна книга – Богдан»,	



2010.— 160 с.

 **Обнаружен Плагиат: 0,16%** <http://www.repository.hneu.edu.ua/bi...> + 5 ресурсов! id: 114

Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 1. Організація баз даних та знань. –

 **Цитування: 0,01%** id: 115

«Комп'ютинг»,

 **Обнаружен Плагиат: 0,18%** <http://www.repository.hneu.edu.ua/bi...> + 5 ресурсов! id: 116

2006. – 460с. 7. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Організація баз даних та знань.

–

 **Цитування: 0,01%** id: 117

«Комп'ютинг»,

2006. – 590с. ДОДАТОК А Інфологічна модель бази даних інформаційно-довідкової системи  
ДОДАТОК Б Програмний код файлу `PymorphicWidget using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Data; using System.Windows.Forms; namespace UIClient { public enum TableItems { School = 0, Classes = 1, Pupils = 2, Subject = 3, Progress = 4, Absence = 5, Teachers = 6, Elective = 7 } public class PymorphicWidget { protected TreeNode nodeData; protected MainWindow widget; public PymorphicWidget(MainWindow window) { widget = window; } protected virtual void view(){} public virtual void adaptFilterA() { widget.mainGrid.DataSource = null; } public virtual void adaptFilterB() { widget.mainGrid.DataSource = null; } public virtual void adapt() { widget.mainGrid.DataSource = null; } public virtual void showEditDialog() { } public virtual void insertDialog() { } public void deleteRow() { if (widget.mainGrid.DataSource != null) { if (DialogResult.Yes == MessageBox.Show(`

 **Цитування: 0,01%** id: 118

"Видалити запис?",

 **Цитування: 0,01%** id: 119

"Увага",

`MessageBoxButtons.YesNo, MessageBoxIcon.Question)) { DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null) row.Delete(); else MessageBox.Show(`

 **Цитування: 0,02%** id: 120

"Рядок не знайдено",

 **Цитування: 0,01%** id: 121

"Помилка",

`MessageBoxButtons.OK, MessageBoxIcon.Error); } } else MessageBox.Show(`

 **Цитування: 0,02%** id: 122

"Таблиця не визначена!",

 **Цитування: 0,01%** id: 123

"Увага",

`MessageBoxButtons.OK, MessageBoxIcon.Warning); } public virtual void dataSave() { } public virtual void dataRefresh() { } public TreeNode Node { get { return nodeData; } set { nodeData = value; } } } public class PupilsTable : PymorphicWidget { public PupilsTable(MainWindow window): base(window) { } protected override void view() { widget.initViewRelation(`

 **Цитування: 0,01%** id: 124

"relation"1",

 **Цитування: 0,01%** id: 125

"Національність",

1, 15); `widget.initViewRelation(`

 **Цитування: 0,01%** id: 126

```

"relation2",
” Цитування: 0,01% id: 127
"Клас",
1, 16); widget.initViewRelation(
” Цитування: 0,01% id: 128
"relation3",
” Цитування: 0,01% id: 129
"Батьки",
1, 17); widget.mainGrid.Columns[0].Visible = false; widget.mainGrid.Columns[1].Visible = false;
widget.mainGrid.Columns[2].HeaderText =
” Цитування: 0,01% id: 130
"ПІБ"
; widget.mainGrid.Columns[2].Width = 230; widget.mainGrid.Columns[3].Visible = false;
widget.mainGrid.Columns[4].HeaderText =
” Цитування: 0,01% id: 131
"Дата народження"
; widget.mainGrid.Columns[4].Width = 230; widget.mainGrid.Columns[5].Visible = false;
widget.mainGrid.Columns[6].Visible = false; widget.mainGrid.Columns[7].Visible = false;
widget.mainGrid.Columns[8].Visible = false; widget.mainGrid.Columns[9].Visible = false;
widget.mainGrid.Columns[10].Visible = false; widget.mainGrid.Columns[11].Visible = false;
widget.mainGrid.Columns[12].Visible = false; widget.mainGrid.Columns[13].Visible = false;
widget.mainGrid.Columns[13].Visible = false; widget.mainGrid.Columns[14].Visible = false;
widget.mainGrid.Columns[15].Visible = false; widget.mainGrid.Columns[16].Visible = false;
widget.mainGrid.Columns[17].Visible = false; } public override void adaptFilterA() {
widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if
(widget.pupilsParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr =
” Цитування: 0,01% id: 132
"P_NAME = "
+
” Цитування: 0,01% id: 133
""
+ widget.pupilsParamDlg.cbPupilsName.Text +
” Цитування: 0,01% id: 134
""
; DataView view = new DataView(widget.fbObject.dataTable(
” Цитування: 0,01% id: 135
"PUPILS"
), filterParemetr,
” Цитування: 0,01% id: 136
"P_ID_PUPIL",
DataViewRowState.CurrentRows); if (view.Count == 0) MessageBox.Show(
” Цитування: 0,03% id: 137
"Дані відсутні в базі",
” Цитування: 0,01% id: 138
"Увага!",
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { widget.mainGrid.DataSource = 0;
widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();
} } } public override void adaptFilterB() { widget.pupilsBirthdayParam = new

```

```

PupilsBirthdayParam(); if (widget.pupilsBirthdayParam.ShowDialog() == DialogResult.OK) {
string filterParemetr =
"Цитирования: 0,01%
"P_BIRTH_DATE = "
+
"Цитирования: 0,01%
""
+ widget.pupilsBirthdayParam.date1.Value.ToShortDateString() + "" + " AND " + "P_BIRTH_DATE
= " +
"Цитирования: 0,01%
""
+ widget.pupilsBirthdayParam.date2.Value.ToShortDateString() +
"Цитирования: 0,01%
""
; DataView view = new DataView(widget.fbObject.dataTable(
"Цитирования: 0,01%
"PUPILS"
), filterParemetr,
"Цитирования: 0,01%
"P_ID_PUPIL",
DataRowState.CurrentRows); if (view.Count == 0) MessageBox.Show(
"Цитирования: 0,03%
"Дані відсутні в базі",
"Цитирования: 0,01%
"Увага!",
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { widget.mainGrid.DataSource = 0;
widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();
} } public override void adapt() { widget.mainGrid.DataSource = 0; DataView view =
widget.pupilsView[widget.treePosition[nodeData.Text]]; widget.currentTable =
view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();
} public override void showEditDialog() { widget.ppIDlg = new PupilsDialog(widget.fbObject);
DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null)
widget.ppIDlg.ShowDialog(row); else MessageBox.Show(
"Цитирования: 0,02%
"Рядок не знайдено",
"Цитирования: 0,01%
"Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() {
widget.ppIDlg = new PupilsDialog(widget.fbObject); widget.ppIDlg.ShowDialog(); } public
override void dataSave() { widget.fbObject.save(
"Цитирования: 0,01%
"PUPILS"
); } public override void dataRefresh() { widget.fbObject.refill(
"Цитирования: 0,01%
"PUPILS"
); } } public class SubjectTable : PolymorphicWidget { public SubjectTable(MainWindow window) :
base(window) { } protected override void view() { widget.initViewRelation(

```

” Цитирования: 0,01%	id: 151
"relation11",	
” Цитирования: 0,01%	id: 152
"Вчитель",	
1, 0); widget.initViewRelation(	
” Цитирования: 0,01%	id: 153
"relation14",	
” Цитирования: 0,01%	id: 154
"Клас",	
1, 1); widget.mainGrid.Columns[2].Visible = false; widget.mainGrid.Columns[3].Visible = false;	
widget.mainGrid.Columns[4].Visible = false; widget.mainGrid.Columns[5].HeaderText =	
” Цитирования: 0,01%	id: 155
"Назва предмета"	
; widget.mainGrid.Columns[5].Width = 230; widget.mainGrid.Columns[6].HeaderText =	
” Цитирования: 0,01%	id: 156
"Кількість годин"	
; widget.mainGrid.Columns[6].Width = 230; } public override void adaptFilterA() {	
widget.teacherParamDlg = new TeacherNameParam(widget.fbObject); if	
(widget.teacherParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr1 =	
” Цитирования: 0,01%	id: 157
"I_NAME = "	
+	
” Цитирования: 0,01%	id: 158
""	
+ widget.teacherParamDlg.cbTeacher.Text +	
” Цитирования: 0,01%	id: 159
""	
; DataView view1 = new DataView(widget.fbObject.dataTable(	
” Цитирования: 0,01%	id: 160
"TEACHERS"	
), filterParemetr1,	
” Цитирования: 0,01%	id: 161
"I_ID_TEACHER",	
DataViewRowState.CurrentRows); if (view1.Count == 0) MessageBox.Show(	
” Цитирования: 0,03%	id: 162
"Дані відсутні в базі",	
” Цитирования: 0,01%	id: 163
"Увага!",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { string filterParemetr2 =	
” Цитирования: 0,05%	id: 164
"S_ID_TEACHER = " + "" + view1[0].Row[0]	
.ToString() +	
” Цитирования: 0,01%	id: 165
""	
; DataView view2 = new DataView(widget.fbObject.dataTable(	
” Цитирования: 0,01%	id: 166

```

"SUBJECT"
), filterParemetr2,
” Цитирования: 0,01% id: 167
"S_ID_SUBJECT",
DataRowState.CurrentRows); if (view2.Count == 0) { MessageBox.Show(
” Цитирования: 0,03% id: 168
"Дані відсутні в базі",
” Цитирования: 0,01% id: 169
"Увага!",
MessageBoxButtons.OK, MessageBoxIcon.Warning); return; } widget.mainGrid.DataSource = 0;
widget.currentTable = view2.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view2;
this.view(); } } } public override void adaptFilterB() { } public override void adapt() {
widget.mainGrid.DataSource = 0; DataView view =
widget.subjectView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =
view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();
} public override void showEditDialog() { widget.sbjDlg = new SubjectDialog(widget.fbObject);
DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null)
widget.sbjDlg.ShowDialog(row); else MessageBox.Show(
” Цитирования: 0,02% id: 170
"Рядок не знайдено",
” Цитирования: 0,01% id: 171
"Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() {
widget.sbjDlg = new SubjectDialog(widget.fbObject); widget.sbjDlg.ShowDialog(); } public
override void dataSave() { widget.fbObject.save(
” Цитирования: 0,01% id: 172
"SUBJECT"
); } public override void dataRefresh() { widget.fbObject.refill(
” Цитирования: 0,01% id: 173
"SUBJECT"
); } } public class ProgressTable : PolymorphicWidget { public ProgressTable(MainWindow window)
: base(window) { } protected override void view() { widget.initViewRelation(
” Цитирования: 0,01% id: 174
"relation13",
” Цитирования: 0,01% id: 175
"Учень",
3, 0); widget.initViewRelation(
” Цитирования: 0,01% id: 176
"relation5",
” Цитирования: 0,01% id: 177
"Предмет",
2, 0); widget.mainGrid.Columns[2].Visible = false; widget.mainGrid.Columns[3].Visible = false;
widget.mainGrid.Columns[4].Visible = false; widget.mainGrid.Columns[8].Visible = false;
widget.mainGrid.Columns[5].HeaderText =
” Цитирования: 0,01% id: 178
"1-й семестр"
; widget.mainGrid.Columns[5].Width = 140; widget.mainGrid.Columns[6].HeaderText =

```



” Цитування: 0,01%	id: 179
"2-й семестр"	
; widget.mainGrid.Columns[6].Width = 140; widget.mainGrid.Columns[7].HeaderText =	
” Цитування: 0,02%	id: 180
"Оцінка за рік"	
; widget.mainGrid.Columns[7].Width = 140; } public override void adaptFilterA() { widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if (widget.pupilsParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr1 =	
” Цитування: 0,01%	id: 181
"P_NAME = "	
+	
” Цитування: 0,01%	id: 182
""	
+ widget.pupilsParamDlg.cbPupilsName.Text +	
” Цитування: 0,01%	id: 183
""	
; DataView view1 = new DataView(widget.fbObject.dataTable( ” Цитування: 0,01%	id: 184
"PUPILS"	
), filterParemetr1,	
” Цитування: 0,01%	id: 185
"P_ID_PUPIL",	
DataViewRowState.CurrentRows); if (view1.Count == 0) MessageBox.Show( ” Цитування: 0,03%	id: 186
"Дані відсутні в базі",	
” Цитування: 0,01%	id: 187
"Увага!",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { string filterParemetr2 =	
” Цитування: 0,05%	id: 188
"P_ID_PUPIL = " + "" + view1[0].Row[0]	
.ToString() +	
” Цитування: 0,01%	id: 189
""	
; DataView view2 = new DataView(widget.fbObject.dataTable( ” Цитування: 0,01%	id: 190
"PROGRESS"	
), filterParemetr2,	
” Цитування: 0,01%	id: 191
"P_ID_PROGRESS",	
DataViewRowState.CurrentRows); if (view2.Count == 0) { MessageBox.Show( ” Цитування: 0,03%	id: 192
"Дані відсутні в базі",	
” Цитування: 0,01%	id: 193
"Увага!",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); return; } widget.mainGrid.DataSource = 0; widget.currentTable = view2.Table.TableName; widget.bindingSource1.DataSource =	

```

widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view2;
this.view(); } } } public override void adaptFilterB() { } public override void adapt() {
widget.mainGrid.DataSource = 0; DataView view =
widget.progresView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =
view.Table.TableName; widget.bindingSource1.DataSource =
widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();
} public override void showEditDialog() { widget.prgDlg = new ProgressDialog(widget.fbObject);
DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null)
widget.prgDlg.ShowDialog(row); else MessageBox.Show(

```

” Цитирования: **0,02%** id: 194

"Рядок не найдено",

” Цитирования: **0,01%** id: 195

"Помилка",

```

MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() {
widget.prgDlg = new ProgressDialog(widget.fbObject); widget.prgDlg.ShowDialog(); } public
override void dataSave() { widget.fbObject.save(

```

” Цитирования: **0,01%** id: 196

"PROGRESS"

```

); } public override void dataRefresh() { widget.fbObject.refill(

```

” Цитирования: **0,01%** id: 197

"PROGRESS"

```

); } } public class AbsenceTable : PolymorphicWidget { public AbsenceTable(MainWindow window)
: base(window) { } protected override void view() { widget.initViewRelation(

```

” Цитирования: **0,01%** id: 198

"relation16",

” Цитирования: **0,01%** id: 199

"Клас",

```

1, 0); widget.initViewRelation(

```

” Цитирования: **0,01%** id: 200

"relation6",

” Цитирования: **0,01%** id: 201

"Учень",

```

2, 0); widget.mainGrid.Columns[1].Visible = false; widget.mainGrid.Columns[2].Visible = false;
widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[6].Visible = false;
widget.mainGrid.Columns[4].HeaderText =

```

” Цитирования: **0,01%** id: 202

"Місяць"

```

; widget.mainGrid.Columns[4].Width = 140; widget.mainGrid.Columns[5].HeaderText =

```

” Цитирования: **0,01%** id: 203

"Кількість годин"

```

; widget.mainGrid.Columns[5].Width = 170; } public override void adaptFilterA() {
widget.pupilsParamDlg = new PupilsNameParam(widget.fbObject); if
(widget.pupilsParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr1 =

```

” Цитирования: **0,01%** id: 204

"P\_NAME = "

+

” Цитирования: **0,01%** id: 205

""

```

+ widget.pupilsParamDlg.cbPupilsName.Text +

```

Цитування: 0,01%	id: 206
; <code>DataGridView view1 = new DataGridView(widget.fbObject.dataTable(</code>	
Цитування: 0,01%	id: 207
<code>"PUPILS"</code>	
<code>), filterParemetr1,</code>	
Цитування: 0,01%	id: 208
<code>"P_ID_PUPIL",</code>	
<code>DataGridViewState.CurrentRow); if (view1.Count == 0) MessageBox.Show(</code>	
Цитування: 0,03%	id: 209
<code>"Дані відсутні в базі",</code>	
Цитування: 0,01%	id: 210
<code>"Увага!",</code>	
<code>MessageBoxButtons.OK, MessageBoxIcon.Warning); else { string filterParemetr2 =</code>	
Цитування: 0,05%	id: 211
<code>"A_ID_PUPIL = " + "" + view1[0].Row[0]</code>	
<code>.ToString() +</code>	
Цитування: 0,01%	id: 212
""	
; <code>DataGridView view2 = new DataGridView(widget.fbObject.dataTable(</code>	
Цитування: 0,01%	id: 213
<code>"ABSENCE"</code>	
<code>), filterParemetr2,</code>	
Цитування: 0,01%	id: 214
<code>"A_ID_ABSENCE",</code>	
<code>DataGridViewState.CurrentRow); if (view2.Count == 0) { MessageBox.Show(</code>	
Цитування: 0,03%	id: 215
<code>"Дані відсутні в базі",</code>	
Цитування: 0,01%	id: 216
<code>"Увага!",</code>	
<code>MessageBoxButtons.OK, MessageBoxIcon.Warning); return; } widget.mainGrid.DataSource = 0;</code>	
<code>widget.currentTable = view2.Table.TableName; widget.bindingSource1.DataSource =</code>	
<code>widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view2;</code>	
<code>this.view(); } } } public override void adaptFilterB() { } public override void adapt() {</code>	
<code>widget.mainGrid.DataSource = 0; DataGridView view =</code>	
<code>widget.absenceView[widget.treePosition[nodeData.Parent.Text]]; widget.currentTable =</code>	
<code>view.Table.TableName; widget.bindingSource1.DataSource =</code>	
<code>widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view();</code>	
<code>} public override void showEditDialog() { widget.absDlg = new AbsenceDialog(widget.fbObject);</code>	
<code>DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null)</code>	
<code>widget.absDlg.ShowDialog(row); else MessageBox.Show(</code>	
Цитування: 0,02%	id: 217
<code>"Рядок не знайдено",</code>	
Цитування: 0,01%	id: 218
<code>"Помилка",</code>	
<code>MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() {</code>	
<code>widget.absDlg = new AbsenceDialog(widget.fbObject); widget.absDlg.ShowDialog(); } public</code>	
<code>override void dataSave() { widget.fbObject.save(</code>	

” Цитування: 0,01%	id: 219
"ABSENCE"	
); } public override void dataRefresh() { widget.fbObject.refill(	
” Цитування: 0,01%	id: 220
"ABSENCE"	
); } } public class TeacherTable : PolymorphicWidget { public TeacherTable(MainWindow window) : base(window) { } protected override void view() { widget.initViewRelation(	
” Цитування: 0,01%	id: 221
"relation8",	
” Цитування: 0,01%	id: 222
"Кваліфікація",	
1, 17); widget.initViewRelation(	
” Цитування: 0,01%	id: 223
"relation9",	
” Цитування: 0,01%	id: 224
"Освіта",	
1, 18); widget.mainGrid.Columns[0].Visible = false; //widget.mainGrid.Columns[1].Visible = false; widget.mainGrid.Columns[2].Visible = false; widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[4].Visible = false; widget.mainGrid.Columns[5].Visible = false; widget.mainGrid.Columns[6].Visible = false; widget.mainGrid.Columns[7].Visible = false; widget.mainGrid.Columns[8].Visible = false; widget.mainGrid.Columns[9].Visible = false; widget.mainGrid.Columns[10].Visible = false; widget.mainGrid.Columns[11].Visible = false; widget.mainGrid.Columns[12].Visible = false; widget.mainGrid.Columns[13].Visible = false; widget.mainGrid.Columns[14].Visible = false; widget.mainGrid.Columns[15].Visible = false; widget.mainGrid.Columns[16].Visible = false; widget.mainGrid.Columns[18].Visible = false; widget.mainGrid.Columns[1].HeaderText =	
” Цитування: 0,01%	id: 225
"ПІБ"	
; widget.mainGrid.Columns[1].Width = 230; } public override void adaptFilterA() { widget.teacherParamDlg = new TeacherNameParam(widget.fbObject); if (widget.teacherParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr =	
” Цитування: 0,01%	id: 226
"I_NAME = "	
+	
” Цитування: 0,01%	id: 227
""	
+ widget.teacherParamDlg.cbTeacher.Text +	
” Цитування: 0,01%	id: 228
""	
; DataView view = new DataView(widget.fbObject.dataTable(	
” Цитування: 0,01%	id: 229
"TEACHERS"	
), filterParemetr,	
” Цитування: 0,01%	id: 230
"I_ID_TEACHER",	
DataViewRowState.CurrentRows); if (view.Count == 0) MessageBox.Show(	
” Цитування: 0,03%	id: 231
"Дані відсутні в базі",	
”	

Цитирования: 0,01%	id: 232
"Увага!",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view(); } } } public override void adaptFilterB() { widget.teacherCategoryParamDlg = new TeacherCategoryParam(); if (widget.teacherCategoryParamDlg.ShowDialog() == DialogResult.OK) { string filterParemetr =	
Цитирования: 0,01%	id: 233
"T_CATEGORY = "	
+	
Цитирования: 0,01%	id: 234
""	
+ widget.teacherCategoryParamDlg.cbTextParam.Text +	
Цитирования: 0,01%	id: 235
""	
; DataView view = new DataView(widget.fbObject.dataTable(	
Цитирования: 0,01%	id: 236
"TEACHERS"	
), filterParemetr,	
Цитирования: 0,01%	id: 237
"T_ID_TEACHER",	
DataViewRowState.CurrentRows); if (view.Count == 0) MessageBox.Show(	
Цитирования: 0,03%	id: 238
"Дані відсутні в базі",	
Цитирования: 0,01%	id: 239
"Увага!",	
MessageBoxButtons.OK, MessageBoxIcon.Warning); else { widget.mainGrid.DataSource = 0; widget.currentTable = view.Table.TableName; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = view; this.view(); } } } public override void adapt() { widget.mainGrid.DataSource = 0; widget.currentTable =	
Цитирования: 0,01%	id: 240
"TEACHERS"	
; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable); widget.mainGrid.DataSource = widget.bindingSource1; this.view(); } public override void showEditDialog() { widget.tchDlg = new TeachersDialog(widget.fbObject); DataRow row = widget.findCurrentRow(widget.mainGrid); if (row != null) widget.tchDlg.ShowDialog(row); else MessageBox.Show(	
Цитирования: 0,02%	id: 241
"Рядок не знайдено",	
Цитирования: 0,01%	id: 242
"Помилка",	
MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() { widget.tchDlg = new TeachersDialog(widget.fbObject); widget.tchDlg.ShowDialog(); } public override void dataSave() { widget.fbObject.save(	
Цитирования: 0,01%	id: 243
"TEACHERS"	
); } public override void dataRefresh() { widget.fbObject.refill(	
Цитирования: 0,01%	id: 244



```

"TEACHERS"
); } } public class ElectiveTable : PolymorphicWidget { public ElectiveTable(MainWindow window) :
base(window) { } protected override void view() { widget.initViewRelation(
" Цитування: 0,01% id: 245
"relation7",
" Цитування: 0,01% id: 246
"Учень",
2, 4); widget.initViewRelation(
" Цитування: 0,01% id: 247
"relation12",
" Цитування: 0,01% id: 248
"Вчитель",
1, 5); widget.mainGrid.Columns[0].Visible = false; widget.mainGrid.Columns[2].Visible = false;
widget.mainGrid.Columns[3].Visible = false; widget.mainGrid.Columns[1].HeaderText =
" Цитування: 0,01% id: 249
"Факультатив"
; widget.mainGrid.Columns[1].Width = 230; widget.mainGrid.Columns[6].HeaderText =
" Цитування: 0,01% id: 250
"Кількість годин"
; widget.mainGrid.Columns[6].Width = 230; } public override void adaptFilterA() { } public
override void adaptFilterB() { } public override void adapt() { widget.mainGrid.DataSource = 0;
widget.currentTable =
" Цитування: 0,01% id: 251
"ELECTIVE"
; widget.bindingSource1.DataSource = widget.fbObject.dataTable(widget.currentTable);
widget.mainGrid.DataSource = widget.bindingSource1; this.view(); } public override void
showEditDialog() { widget.elcDlg = new ElectiveDialog(widget.fbObject); DataRow row =
widget.findCurrentRow(widget.mainGrid); if (row != null) widget.elcDlg.ShowDialog(row); else
MessageBox.Show(
" Цитування: 0,02% id: 252
"Рядок не знайдено",
" Цитування: 0,01% id: 253
"Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); } public override void insertDialog() {
widget.elcDlg = new ElectiveDialog(widget.fbObject); widget.elcDlg.ShowDialog(); } public
override void dataSave() { widget.fbObject.save(
" Цитування: 0,01% id: 254
"ELECTIVE"
); } public override void dataRefresh() { widget.fbObject.refill(
" Цитування: 0,01% id: 255
"ELECTIVE"
); } } public class WidgetAgregat { private PolymorphicWidget[] polymorphWidget; private const int
size = 7; public WidgetAgregat(MainWindow window) { polymorphWidget = new
PolymorphicWidget[size]; polymorphWidget[0] = new PolymorphicWidget(window);
polymorphWidget[1] = new PupilsTable(window); polymorphWidget[2] = new
SubjectTable(window); polymorphWidget[3] = new ProgressTable(window); polymorphWidget[4] =
new AbsenceTable(window); polymorphWidget[5] = new TeacherTable(window);
polymorphWidget[6] = new ElectiveTable(window); } public PolymorphicWidget this[int index] {get
{ if (index < 0 || index == size) return polymorphWidget[0]; return polymorphWidget[index]; } } }

```

```
ДОДАТОК В Програмний код файлу FirebirdInterface using System; using System.Collections;
using System.Collections.Generic; using System.Data; using System.Text; using System.Linq;
using System.Windows.Forms; using FirebirdSql.Data.FirebirdClient; using
System.Text.RegularExpressions; namespace UIClient { public class FirebirdInterface { private
List FbDataAdapter adapters; private FbConnection connect; private DataSet memory_db; public
FbConnection ConnectionObject { get { return connect; } } public FirebirdInterface() { connect =
new FbConnection(); adapters = new List FbDataAdapter (); memory_db = new DataSet(); }
private string queryGetTables() { return
```

Цитирования: 0,11%

id: 256

```
"SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE (RDB$SYSTEM_FLAG = 0) AND
(RDB$VIEW_SOURCE IS NULL) ORDER BY RDB$RELATION_NAME"
```

```
; } private string queryGetFields(string tableName) { return
```

Цитирования: 0,09%

id: 257

```
"SELECT R.RDB$FIELD_NAME, R.RDB$NULL_FLAG, T.RDB$TYPE_NAME, F.RDB$FIELD_LENGTH,
F.RDB$FIELD_SCALE, F.RDB$FIELD_SUB_TYPE "
```

+

Цитирования: 0,04%

id: 258

```
"FROM RDB$TYPES T, RDB$RELATION_FIELDS R "
```

+

Цитирования: 0,11%

id: 259

```
"INNER JOIN RDB$FIELDS F ON F.RDB$FIELD_TYPE = T.RDB$TYPE AND T.RDB$FIELD_NAME =
'RDB$FIELD_TYPE' "
```

+

Цитирования: 0,11%

id: 260

```
"WHERE F.RDB$FIELD_NAME = R.RDB$FIELD_SOURCE AND R.RDB$SYSTEM_FLAG = 0 AND
RDB$RELATION_NAME = '"
```

```
+ tableName +
```

Цитирования: 0,04%

id: 261

```
" ORDER BY R.RDB$FIELD_POSITION ASC"
```

```
; } private DataColumn createDataColumn(string columnName, string typeName, int subType,
bool allowDBNull, int lenght) { DataColumn column = new DataColumn(columnName);
column.AllowDBNull = allowDBNull; switch (typeName) { case (
```

Цитирования: 0,01%

id: 262

```
"SHORT"
```

```
): { column.DataType = typeof(Int16); break; } case (
```

Цитирования: 0,01%

id: 263

```
"LONG"
```

```
): { column.DataType = typeof(Int32); break; } case (
```

Цитирования: 0,01%

id: 264

```
"INT64"
```

```
): { column.DataType = subType == 0 ? typeof(Int64) : typeof(Decimal); break; } case (
```

Цитирования: 0,01%

id: 265

```
"FLOAT"
```

```
): { column.DataType = typeof(Single); break; } case (
```

Цитирования: 0,01%

id: 266

```
"DOUBLE"
```

```
): { column.DataType = typeof(Double); break; } case (
```

Цитирования: 0,01%

id: 267

```
"DATE"
```

```
); { column.DataType = typeof(DateTime); column.DateTimeMode = DataSetDateTime.Local;
break; } case (
```

Цитирования: 0,01%

id: 268

"TIMESTAMP"

```
); { column.DataType = typeof(DateTime); column.DateTimeMode = DataSetDateTime.Local;
break; } case (
```

Цитирования: 0,01%

id: 269

"TIME"

```
); { column.DataType = typeof(TimeSpan); break; } case (
```

Цитирования: 0,01%

id: 270

"TEXT"

```
); { column.DataType = typeof(String); column.MaxLength = lenght; break; } case (
```

Цитирования: 0,01%

id: 271

"VARYING"

```
); { column.DataType = typeof(String); column.MaxLength = lenght; break; } case (
```

Цитирования: 0,01%

id: 272

"CSTRING"

```
); { column.DataType = typeof(String); column.MaxLength = lenght; break; } case (
```

Цитирования: 0,01%

id: 273

"BLOB"

```
); { column.DataType = typeof(byte[]); column.MaxLength = lenght; break; } } return column; }
private void loadTableColumn(string tableName) { int index =
memory_db.Tables.IndexOf(tableName); if (index == 0) { if (!isOpen()) openDataBase(); DataTable
table = new DataTable(
```

Цитирования: 0,01%

id: 274

"FIELDSNAME"

```
); FbDataAdapter adapter = new FbDataAdapter(queryGetFields(tableName), connect);
adapter.Fill(table); int tableCount = table.Rows.Count; for (int i = 0; i < tableCount; ++i) { string
columnName = table.Rows[i][0].ToString().TrimEnd(); bool allowDBNull = table.Rows[i][1] !=
DBNull.Value ? false : true; string typeName = table.Rows[i][2].ToString().TrimEnd(); int lenght =
table.Rows[i][3] != DBNull.Value ? Convert.ToInt32(table.Rows[i][3]) : 0; int subTypeIndex =
table.Rows[i][5] != DBNull.Value ? Convert.ToInt32(table.Rows[i][5]) : 0; DataColumn column =
createDataColumn(columnName, typeName, subTypeIndex, allowDBNull, lenght);
memory_db.Tables[index].Columns.Add(column); } memory_db.Tables[tableName].PrimaryKey
= new DataColumn[] { memory_db.Tables[tableName].Columns[0] };
memory_db.Tables[tableName].Columns[0].AutoIncrement = true; fillSchema(tableName); if
(isOpen()) closeDataBase(); } } public string initConnectionString(string clientLibrary, string
database, string userID, string password, string role, string charset, int port) {
FbConnectionStringBuilder connectionString = new FbConnectionStringBuilder();
connectionString.ClientLibrary = clientLibrary; connectionString.Database = database;
connectionString.UserID = userID; connectionString.Password = password; connectionString.Charset =
charset; connectionString.Role = role; connectionString.Port = port; connect.ConnectionString =
connectionString.ConnectionString; return connect.ConnectionString; } public bool openDataBase() {
try { if (connect.ConnectionString.Length == 0) connect.Open(); else throw new System.Exception(
```

Цитирования: 0,04%

id: 275

"Рядок з'єднання з БД не коректний"

```
); } catch (System.Exception Except) { MessageBox.Show(Except.Message,
```

Цитирования: 0,01%

id: 276

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); connect.Close(); return false; } return
connect.State == System.Data.ConnectionState.Open ? true : false; } public bool closeDataBase()
{ try { connect.Close(); } catch (System.Exception Except) { MessageBox.Show(Except.Message,
```

Цитування: 0,01%	id: 277
"Помилка",	
<pre> MessageBoxButtons.OK, MessageBoxIcon.Error); return false; false; } } return connect.State == System.Data.ConnectionState.Closed ? true : public bool isOpen() { return connect.State == System.Data.ConnectionState.Open ? true : false; } public List string listTables() { List string list = new List string (); for (int i = 0; i memory_db.Tables.Count; ++i) list.Add(memory_db.Tables[i].TableName); return list; } public List string listColumns(int indexTable) { List string list = new List string (); if (tableName(indexTable) != string.Empty) { int count = memory_db.Tables[indexTable].Columns.Count; for (int i = 0; i count; ++i) list.Add(memory_db.Tables[indexTable].Columns[i].ColumnName); } return list; } public void loadTables() { if (!isOpen()) openDataBase(); this.clear(); connect); Regex rgx = new Regex(@ </pre>	
Цитування: 0,01%	id: 278
"\"\$"	
<pre> ); DataTable table = new DataTable( </pre>	
Цитування: 0,01%	id: 279
"TABLESNAME"	
<pre> ); FbDataAdapter adapter = new FbDataAdapter(queryGetTables(), adapter.Fill(table); int tableCount = table.Rows.Count; for (int i = 0, j = 0; i tableCount; ++i) { string item = table.Rows[i][j].ToString(); item = item.TrimEnd(); if (!rgx.IsMatch(item)) { adapters.Add(new FbDataAdapter()); FbCommandBuilder commandBuilder = new FbCommandBuilder(adapters.Last()); commandBuilder.ConflictOption = ConflictOption.OverwriteChanges; memory_db.Tables.Add(new DataTable(item)); } } if (isOpen()) closeDataBase(); } public void fillSchema(string tableName) { try { int index = memory_db.Tables.IndexOf(tableName); if (index == 0 &amp;&amp; connect.ConnectionString.Length 0) { adapters[index].SelectCommand = new FbCommand( </pre>	
Цитування: 0,02%	id: 280
"SELECT * FROM "	
<pre> + tableName, connect); adapters[index].FillSchema(memory_db, System.Data.SchemaType.Mapped); } else throw new Exception( </pre>	
Цитування: 0,06%	id: 281
"Помилка! Не вірне ім'я таблиці або рядок з'єднання"	
<pre> ); } catch (Exception e) { MessageBox.Show(e.Message, </pre>	
Цитування: 0,01%	id: 282
"Помилка",	
<pre> MessageBoxButtons.OK, MessageBoxIcon.Error); } } public int tableIndex(string name) { try { int index = memory_db.Tables.IndexOf(name); if (index == -1) throw new Exception( </pre>	
Цитування: 0,04%	id: 283
"Таблиці з цим ім'ям не існує"	
<pre> ); return index; } catch (Exception e) { MessageBox.Show(e.Message, </pre>	
Цитування: 0,01%	id: 284
"Помилка",	
<pre> MessageBoxButtons.OK, MessageBoxIcon.Error); return -1; } } public string tableName(int index) { try { if (index == 0 &amp;&amp; index == memory_db.Tables.Count - 1) return memory_db.Tables[index].TableName; else throw new Exception( </pre>	
Цитування: 0,04%	id: 285
"Таблиці з цим індексом не існує"	
<pre> ); } catch (Exception e) { MessageBox.Show(e.Message, </pre>	
Цитування: 0,01%	id: 286
"Помилка",	
<pre> MessageBoxButtons.OK, MessageBoxIcon.Error); return string.Empty; } } public DataTable dataTable(string tableName) { int index = tableIndex(tableName); if (index != -1) return </pre>	

```
memory_db.Tables[index]; else return null; } public FbDataAdapter tableAdapter(string
tableName) { int index = tableIndex(tableName); if (index != -1) return adapters[index]; else
return null; } public DataSet dataSet() { return memory_db; } public bool refill(string tableName)
{ try { int index = memory_db.Tables.IndexOf(tableName); if (index = 0 && index =
memory_db.Tables.Count - 1) { adapters[index].Fill(memory_db, tableName); return true; } else
return false; } catch (Exception e) { MessageBox.Show(e.Message,
```

” Цитирования: **0,01%**

id: 287

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); return false; } } public bool fill(String
tableName) { try { int index = memory_db.Tables.IndexOf(tableName); if (index = 0 && index =
memory_db.Tables.Count - 1) { if (memory_db.Tables.Contains(tableName))
memory_db.Tables[index].Clear(); this.loadTableColumn(tableName);
adapters[index].Fill(memory_db, tableName); return true; } else return false; } catch (Exception
e) { MessageBox.Show(e.Message,
```

” Цитирования: **0,01%**

id: 288

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); return false; } } public bool save(string
tableName) { DataTable currentTable = dataTable(tableName); if (currentTable.GetChanges() ==
null) return false; try { tableAdapter(tableName).Update(currentTable.GetChanges());
currentTable.AcceptChanges(); return true; } catch (Exception) { MessageBox.Show(
```

” Цитирования: **0,03%**

id: 289

"Не вдалося зберегти запис",

” Цитирования: **0,01%**

id: 290

"Помилка!",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); currentTable.RejectChanges(); return false; } }
public void clear() { adapters.Clear(); memory_db.Relations.Clear(); memory_db.Tables.Clear();
memory_db.Clear(); } public void clear(int index) { try { string item = tableName(index); if (item
== string.Empty) { adapters.Remove(adapters[index]); memory_db.Tables[index].Clear(); } }
catch (Exception e) { MessageBox.Show(e.Message,
```

” Цитирования: **0,01%**

id: 291

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); } } public void clear(string tableName) { try {
int index = tableIndex(tableName); if (index != -1) { adapters.Remove(adapters[index]);
memory_db.Tables[index].Clear(); } } catch (Exception e) { MessageBox.Show(e.Message,
```

” Цитирования: **0,01%**

id: 292

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); } } public void setRelation(string relationName,
string parentTable, string primaryKey, string childrenTable, string forigenKey) { try {
memory_db.Relations.Add(new DataRelation(relationName,
dataTable(parentTable).Columns[primaryKey], dataTable(childrenTable).Columns[forigenKey],
true)); } catch (Exception e) { MessageBox.Show(e.Message,
```

” Цитирования: **0,01%**

id: 293

"Помилка",

```
MessageBoxButtons.OK, MessageBoxIcon.Error); } } public DataTable executeQuery(string
strQuery) { DataTable table = new DataTable(
```

” Цитирования: **0,01%**

id: 294

"QUERY"

```
); FbDataAdapter adapter = new FbDataAdapter(strQuery, connect); adapter.Fill(table); return
table; } } }
```

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!



Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу.  
Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)

Детектор Плагиата - Ваше право на оригинальность! ☐ SkyLine LLC