Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення хмарної системи контролю успішності студентів”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Зобенко П.О.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *123 "*Комп'ютерна інженерія*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Зобенку Павлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення хмарної системи контролю успішності студентів*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення хмарної системи контролю успішності студентів*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Зобенко П.О.
(прізвище та ініціали)

АНОТАЦІЯ

Зобенко П.О. Програмне забезпечення хмарної системи контролю успішності студентів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для хмарної системи контролю успішності студентів.

Метою розробки є програмне забезпечення хмарної системи контролю успішності студентів.

Результат роботи – програмна реалізація хмарної системи контролю успішності студентів.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi.

Ключові слова: комп'ютерна інженерія, хмарна система, контроль успішності студентів

ABSTRACT

Zobenko P.O. Software of the cloud-based student performance monitoring system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for a cloud system for monitoring the success of students.

The purpose of the development is the software of the cloud system for monitoring students' progress.

The result of the work is the software implementation of a cloud system for monitoring student performance.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi environment.

Keywords: computer engineering, cloud system, monitoring of students' performance

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	бази даних
КСН	–	комп’ютерна система навчання
ПЗ	–	програмне забезпечення
РХ	–	робоча характеристика
СВЯ	–	середня вихідна якість
СУБД	–	системи управління базами даних

ВСТУП

Актуальність теми. Актуальність роботи обумовлена зростанням рівня комп'ютеризації та інформатизації громадського життя, збільшенням інформаційного обміну, інтенсивним розвитком ринку освітніх послуг. В умовах вступу України до системи європейської освіти існують проблеми, пов’язані з інтенсивним впровадженням нових інформаційних технологій навчання. Реалізація цих заходів вимагає суттєвих змін та реформ існуючої системи освіти. Нагальною необхідністю стає впровадження новітніх інформаційних технологій в навчально-пізнавальну діяльність студента, формування на їх основі нових стратегій, спрямованих на розширення форм самоосвіти та індивідуалізацію процесу навчання.

Існуючі сьогодні системи комп’ютерного навчання та контролю успішності студентів та контролю знань, в тому числі системи дистанційної освіти, не забезпечують в повному обсязі розв’язання такого актуальної наукової задачі, як удосконалення та інтенсифікації процесу самоосвіти за рахунок його автоматизації та урахування індивідуальних характеристик осіб, яких навчають. Тому тема бакалаврської роботи, яка спрямована на розробку програмного забезпечення, яке призначено для хмарної системи контролю успішності студентів, є актуальною.

Мета й завдання дослідження. Метою роботи є програмне забезпечення хмарної системи контролю успішності студентів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем контролю успішності студентів.
– Дослідження хмарної системи контролю успішності студентів.
– Програмна реалізація хмарної системи контролю успішності студентів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю успішності студентів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення хмарної системи контролю успішності студентів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації програмного забезпечення хмарного контролю успішності студентів. Також дану систему можливо використовувати при вступі до закладів вищої освіти України. У тому числі й при проведені ЗНО, НМТ, ЄДКІ, ЄВІ, ЄФВВ.

Сьогодні навчальні заклади використовують хмарні платформи тестування, щоб підвищити ефективність і точність процесів вступу. У цьому розділі буде розглянуто переваги та ключові переваги хмарного тестування під час вступу до університету, а також його вплив на зарахування студентів.

Традиційно вступ до університету передбачає виконання великих адміністративних завдань, включаючи розповсюдження та збір паперових заяв, їх обробку вручну та організацію екзаменаційних центрів. Цей процес може зайняти багато часу, бути схильним до помилок і складним для ефективного керування. Завдяки переходу на хмарне тестування університети можуть значно спростити процес вступу.

Використання хмарних платформ тестування дозволяє установам автоматизувати різні завдання, такі як подання заявок, планування іспитів і керування даними кандидатів. Ця автоматизація усуває ручну роботу з документами та зменшує адміністративні накладні витрати, дозволяючи офіцерам приймальної комісії більше зосередитися на оцінюванні заяв і прийнятті обґрунтованих рішень.

Основні переваги спрощення процесу вступу за допомогою хмарного тестування:

– Зменшення паперової роботи та адміністративного тягаря.

– Ефективне управління даними кандидатів.

– Автоматизовані системи планування та оповіщення.

– Покращена точність і мінімізація помилок.

Посилена безпека іспитів і заходи проти списування

Однією з головних турбот під час вступу до університету є збереження цілісності процесу іспиту. Традиційні паперові іспити сприйнятливі до шахрайства, видавання себе за іншу особу та витоку питань. Хмарні платформи тестування вирішують ці проблеми, запроваджуючи надійні заходи безпеки для сприяння справедливості та запобігання зловживанням.

Хмарні платформи тестування використовують передові технології для забезпечення безпеки іспиту, такі як біометрична автентифікація, безпечне блокування браузера та віддалений контроль. Ці заходи ідентифікують особу кандидатів, запобігають несанкціонованому доступу до зовнішніх ресурсів і забезпечують віддалений моніторинг під час іспиту. Крім того, використання рандомізованих банків запитань і адаптивних алгоритмів тестування робить майже неможливим для кандидатів ділитися запитаннями або шахраювати традиційними засобами.

Ключові переваги покращеної безпеки іспитів із хмарним тестуванням:

– Автентифікація за допомогою біометрії та безпечного доступу.

– Запобігання шахрайству та несанкціонованому доступу до ресурсів.

– Віддалений нагляд для моніторингу в реальному часі.

– Рандомізовані банки запитань, щоб мінімізувати обмін питаннями.

Ефективне оцінювання та швидші результати

Обробка та оцінка великої кількості екзаменаційних робіт вручну може бути тривалим і трудомістким завданням для приймальних комісій. Хмарні платформи тестування пропонують функції автоматизованого оцінювання, які спрощують цей процес і надають швидші результати як установам, так і заявникам.

Використання цифрових інструментів оцінювання дозволяє швидко й точно оцінювати сценарії відповідей. Ці інструменти можуть автоматично

оцінювати запитання об’єктивного типу, водночас надаючи оцінювачам інтуїтивно зрозумілі інтерфейси для оцінювання суб’єктивних відповідей. Крім того, хмарні системи генерують миттєві результати, дозволяючи кандидатам отримувати сповіщення, а університети – швидше приймати рішення про вступ.

Ключові переваги ефективної оцінки та швидших результатів за допомогою хмарного тестування:

– Автоматизоване виставлення оцінок для питань об’єктивного типу.

– Ефективний інтерфейс оцінювання суб’єктивних відповідей.

– Миттєва генерація результатів для своєчасного прийняття рішень щодо вступу.

– Зменшено навантаження на працівників приймальної комісії.

Покращена доступність і гнучкість для кандидатів

Хмарні платформи тестування пропонують кандидатам гнучкість для складання іспитів з будь-якого місця та в будь-який час, усуваючи географічні обмеження та забезпечуючи ширший доступ до освітніх можливостей. Така доступність особливо корисна для іноземних студентів або тих, хто не може долати великі відстані до екзаменаційних центрів.

Крім того, хмарні платформи тестування забезпечують підтримку різних пристроїв, включаючи ноутбуки, планшети та смартфони. Це гарантує, що кандидати можуть складати іспити на своїх улюблених пристроях, сприяючи інклюзивності та враховуючи різні стилі навчання.

Основні переваги покращеної доступності та гнучкості завдяки хмарному тестуванню:

– Усунення географічних обмежень.

– Більш широкий доступ до іспитів для іноземних студентів.

– Підтримка кількох пристроїв і стилів навчання.

– Зручний розклад іспитів і зниження витрат на відрядження.

Ключові висновки

Хмарне тестування революціонізувало процес вступу до університету, підвищивши ефективність, точність і доступність. Завдяки спрощенню адміністративних завдань, запровадженню суворих заходів безпеки, автоматизації оцінювання та забезпеченню гнучкості для кандидатів навчальні заклади можуть забезпечити справедливий та ефективний процес вступу.

Запровадження хмарних платформ тестування приносить користь не лише університетам, а й абітурієнтам, забезпечуючи швидші результати, спрощуючи логістику іспитів і розширюючи доступ до освітніх можливостей. Оскільки технології продовжують розвиватися, хмарне тестування має стати новим стандартом при вступі до університетів, створюючи більш інклюзивний, ефективний і точний процес оцінювання.

1.2 Область застосування

Областю застосування системи є навчальний процес у вищих навчальних закладах.

Оскільки тисячі абітурієнтів змагаються за обмежені місця, університети повинні забезпечити безпечний, прозорий і справедливий процес вступу. Саме тут хмарне тестування відіграє життєво важливу роль.

Розкриття хмарного тестування

Хмарне тестування, також відоме як онлайн-прокторінг, набуло значного поширення в останні роки. Це передбачає використання технології хмарних обчислень для віддаленого моніторингу та оцінки іспитів. Цей процес поєднує передові програмні рішення з надійними заходами безпеки, що дозволяє університетам підтримувати цілісність своїх вступних тестів.

Переваги хмарного тестування

Впровадження хмарного тестування пропонує численні переваги як для університетів, так і для абітурієнтів:

– **Покращена безпека:** Традиційні паперові іспити вразливі до шахрайства та підробки. Онлайн-прокторінг усуває такі ризики, використовуючи відео- та аудіо-моніторинг у реальному часі, розпізнавання обличчя та інструменти перевірки особи, забезпечуючи автентичність учасника тестування. Це значно зменшує випадки шахрайства та підтримує цілісність процесу експертизи.

– **Зручність і гнучкість:** хмарне тестування дозволяє кандидатам складати іспити у бажаний час і в бажаному місці, не вимагаючи від них фізичної поїздки до центрів тестування. Ця гнучкість розширює доступ до абітурієнтів із віддалених регіонів і створює рівні умови для всіх майбутніх студентів.

– **Економічна ефективність:** усуваючи потребу у фізичних тестових центрах, хмарне тестування зменшує операційні витрати для університетів. Крім того, це позбавляє кандидатів від витрат, пов'язаних з проїздом і проживанням. Це економічно ефективне рішення дозволяє університетам розподіляти ресурси на інші важливі сфери.

– **Повна інтеграція з технологіями.** Хмарне тестування легко інтегрується з різними технологічними досягненнями, такими як програмне забезпечення для дистанційного контролю, аналітика на основі штучного інтелекту та інструменти розпізнавання облич. Ці функції надають університетам комплексне рішення, яке забезпечує прозорість і справедливість під час процесу вступу.

Вплив хмарного тестування на вступ до університету

Запровадження хмарного тестування зробило революцію у вступі до університетів, пропонуючи кілька ключових висновків:

– **Безпека та конфіденційність даних:** Хмарне тестування забезпечує конфіденційність і безпеку даних заявника. Завдяки надійним протоколам шифрування та захищеним серверам університети можуть бути впевнені, що конфіденційна інформація залишається захищеною. Це підвищує довіру заявників і зміцнює загальну цілісність процесу вступу.

– **Покращена ефективність і масштабованість:** хмарне тестування спрощує процес вступу, дозволяючи університетам обробляти більшу кількість абітурієнтів без шкоди для якості оцінювання. Автоматизовані системи оцінювання та аналіз даних у режимі реального часу значно скорочують час, необхідний для обробки результатів і прийняття рішень щодо вступу.

– **Глобальне охоплення:** хмарне тестування долає географічні кордони, дозволяючи університетам залучати різноманітну групу абітурієнтів з усього світу. Підвищення доступності сприяє інклюзивності, різноманітності та мультикультуралізму у закладах вищої освіти.

– **Адаптивне оцінювання:** Завдяки можливостям хмарних обчислень університети можуть створювати адаптивні оцінювання, які адаптуються до здібностей кожного учасника. Це гарантує, що іспити точно оцінюють знання та здібності абітурієнта, надаючи справедливу оцінку всім кандидатам.

Статистика вказує на зростаюче значення хмарного тестування:

– Згідно зі звітом Grand View Research, очікується, що світовий ринок онлайн-прокторінгу досягне 1,57 мільярда доларів США до 2028 року, при середньорічному зростанні на 20,8%.

– Дослідження, проведене Міжнародним журналом інформаційних технологій та управління бізнесом, показало, що 80% студентів віддають перевагу онлайн-екзаменам над традиційним очним оцінюванням.

– Опитування, проведене Examity, показало, що 78% університетів планують прийняти онлайн-прокторінг як постійне рішення після пандемії, підкреслюючи довгостроковий потенціал хмарного тестування.

Роль хмарного тестування в забезпеченні цілісності даних під час вступу до університетів лише зростатиме. Технологічний прогрес у поєднанні зі зростаючим попитом на дистанційне навчання та оцінювання продовжують формувати ландшафт вищої освіти. Переваги хмарного тестування, такі як підвищена безпека, економічна ефективність і масштабованість, роблять його незамінним інструментом для університетів у всьому світі.

Оскільки ми приймаємо цифрову трансформацію, для університетів важливо адаптувати та використовувати хмарне тестування, щоб підтримувати цілісність і надійність своїх вступних процесів. Роблячи це, вони можуть забезпечити рівні можливості для всіх абітурієнтів, захистити конфіденційну інформацію та сформувати майбутнє безпечного вступу.

Традиційно університети покладаються на ручні методи тестування, які можуть займати багато часу та бути схильними до помилок. Однак із появою хмарних стратегій тестування університети тепер можуть революціонізувати свої процеси вступу та забезпечити безперебійний досвід для всіх зацікавлених сторін.

Хмарне тестування надає безліч переваг процесу вступу до університету. Давайте докладніше розглянемо деякі основні переваги:

– **Масштабованість:** хмарне тестування дозволяє університетам легко масштабувати свою інфраструктуру тестування, щоб прийняти велику кількість абітурієнтів. За допомогою традиційних методів може бути важко впоратися з раптовим збільшенням кількості тих, хто здає тест. Однак за допомогою хмарного тестування університети можуть швидко надати додаткові ресурси для задоволення попиту, мінімізуючи затримки та забезпечуючи зручну роботу для абітурієнтів.

– **Економія:** Традиційні методи тестування часто вимагають значних інвестицій у фізичну інфраструктуру та витрати на технічне обслуговування. Хмарне тестування усуває потребу в таких витратах, оскільки університети можуть використовувати хмарну інфраструктуру, надану постачальниками. Це не тільки зменшує витрати, але й дозволяє університетам перенаправляти свої ресурси в інші важливі сфери.

– **Гнучкість:** хмарне тестування дає університетам можливість проводити тести з будь-якого місця в будь-який час. Це особливо корисно для іноземних студентів, які можуть не мати змоги поїхати до кампусу для складання

іспитів. Тепер учні можуть зручно складати свої тести онлайн, мінімізуючи перешкоди та підвищуючи доступність.

– **Безпека даних.** Однією з основних проблем під час вступу є безпека конфіденційних даних студентів. Хмарні платформи тестування впроваджують надійні заходи безпеки, включаючи шифрування та контроль доступу, щоб забезпечити конфіденційність інформації студентів. Це вселяє довіру як до абітурієнтів, так і до університету, знаючи, що їхні дані захищені.

Інтеграція хмарного тестування в процес вступу

Впровадження хмарних стратегій тестування потребує системного підходу, щоб забезпечити повну інтеграцію в процес вступу до університету. Ось кілька ключових кроків, які слід враховувати:

1. **Оцініть платформи тестування:** університети повинні ретельно оцінити та вибрати хмарну платформу тестування, яка відповідає їхнім конкретним потребам. Фактори, які слід враховувати, включають масштабованість, функції безпеки, простоту використання та можливості інтеграції з існуючими системами прийому.

2. **Забезпечити навчання:** надзвичайно важливо навчати як викладачів, так і студентів тому, як ефективно використовувати хмарну платформу тестування. Це гарантує, що всі учасники почуваються комфортно та знають про нову систему, мінімізуючи можливі проблеми під час процесу тестування.

3. **Забезпечення сумісності.** Хмарна платформа тестування має бездоганно інтегруватися з іншими системами вступу, такими як система відстеження заявників та система інформації про студентів. Це забезпечить ефективний обмін даними та оптимізує загальний процес вступу.

4. **Відстежуйте продуктивність:** регулярно відстежуйте продуктивність хмарної платформи тестування, щоб виявити будь-які потенційні вузькі місця або проблеми. Це допоможе університетам завчасно вирішувати будь-які технічні проблеми, які можуть виникнути, і забезпечити безперебійне тестування для абітурієнтів.

Хмарні стратегії тестування мають потенціал для трансформації процесу вступу до університету, пропонуючи численні переваги, включаючи масштабованість, економію коштів, гнучкість і безпеку даних. Ретельно вибираючи та інтегруючи хмарну платформу тестування, університети можуть забезпечити безперебійний досвід для абітурієнтів і оптимізувати їхні робочі процеси вступу. Застосування хмарного тестування є кроком до оцифрування та модернізації процесу вступу, що зрештою покращить загальну ефективність і перевершить очікування студентів.

Пам’ятайте, впровадження хмарного тестування для вступу до університету – це не просто тренд, а необхідність. Будьте попереду конкурентів і підвищте якість вступу для свого закладу за допомогою цих інноваційних стратегій тестування.

Переваги хмарного тестування під час вступу до університетів щодо вартості та ресурсів

У цьому розділі розглядаються переваги впровадження хмарних систем тестування для вступу до університетів.

Розвиток хмарного тестування

Хмарне тестування дозволяє університетам проводити вступні тести в цифровому вигляді, усуваючи потребу в центрах фізичних іспитів і паперових анкетах. Натомість студенти можуть зручно складати іспити з будь-якого місця, де є доступ до Інтернету. Цей підхід пропонує численні переваги, включаючи зниження витрат і покращений розподіл ресурсів.

Вигоди вартості

– **Усунення витрат на фізичну інфраструктуру:** за допомогою хмарного тестування університети можуть значно скоротити витрати, пов’язані зі створенням і обслуговуванням центрів фізичних іспитів. Це включає витрати, пов’язані з приміщенням, меблями, охороною та комунальними послугами. Університет може перенаправити ці заощадження на інші важливі академічні потреби.

– **Зменшення витрат на друк і папір:** традиційні методи тестування вимагають значних ресурсів для друку анкет і бланків відповідей. Застосувавши хмарне тестування, університети можуть повністю усунути ці витрати, сприяючи економії коштів і зусиллям щодо сталого розвитку.

– **Підвищення ефективності тестування:** проведення іспитів онлайн усуває потребу в ручному нагляді, адміністративній документації та обробці результатів. Це, у свою чергу, зменшує адміністративні витрати, пов'язані з людськими ресурсами, зберіганням і керуванням даними.

Ресурсні переваги

– **Масштабованість і гнучкість.** Хмарні платформи тестування можуть працювати з великою кількістю одночасних тестувальників без проблем. Ця масштабованість дозволяє університетам приймати більшу кількість студентів і проводити іспити більш ефективно. Крім того, гнучкість онлайн-тестування дозволяє студентам вибрати зручний час для складання іспитів, зменшуючи матеріально-технічні проблеми.

– **Покращена безпека:** під час іспитів на паперовій основі ризик витоку запитань або обману викликає серйозне занепокоєння. Хмарні платформи тестування використовують розширені заходи безпеки, такі як рандомізовані запитання та відповіді, часові обмеження та віддалений контроль, щоб забезпечити цілісність іспитів. Це усуває потребу у фізичних заходах безпеки, таких як запечатані анкети та ручний контроль.

– **Легка обробка результатів:** хмарні системи автоматизують процес оцінювання та генерації результатів, значно скорочуючи зусилля та час, необхідні для обробки результатів. Університети можуть генерувати точні результати за короткий проміжок часу та безпечно ділитися ними зі студентами та іншими зацікавленими сторонами.

Ключові висновки

Впровадження хмарних систем тестування для вступу до університету дає кілька ключових переваг:

1. Значна економія за рахунок усунення витрат на фізичну інфраструктуру та зменшення витрат на друк і папір.

2. Покращено розподіл ресурсів шляхом спрощення адміністративних процесів і скорочення потреб у персоналі.

3. Масштабованість і гнучкість для більшої кількості студентів і забезпечення зручного тестування.

4. Посилені заходи безпеки для запобігання шахрайству та витоку питань.

5. Ефективна обробка результатів, що дозволяє швидше та точніше генерувати оцінки за іспит.

Підсумовуючи, хмарне тестування надає університетам рентабельне та ресурсоефективне рішення для проведення вступних іспитів. Використовуючи передові технології, навчальні заклади можуть оптимізувати процеси вступу, посилити безпеку та покращити загальний досвід як для студентів, так і для адміністраторів. Завдяки численним перевагам тестування в хмарі, безсумнівно, визначає майбутнє вступу до університетів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення хмарної системи контролю успішності студентів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Розглянемо стандарти та програмі продукти, які існують у області комп'ютерного навчання й систем контролю успішності студентів.

ISO/IEC JTC1

Об'єднаний технічний комітет №1 ISO/МЕК (англ. ISO/IEC Joint Technical Committee 1, ISO/IEC JTC 1) – підрозділ Міжнародної організації по стандартизації (англ. International Organization for Standardization, ISO) і Міжнародної електротехнічної комісії (МЕК, англ. International Electrotechnical Commission, IEC), що займається всіма питаннями зв'язаними зі стандартами в області інформаційних технологій.

Об'єднаний технічний комітет №1 був створений в 1987 році шляхом злиття Технічного Комітету 97 (Інформаційні технології) Міжнародної організації по стандартизації (англ. ISO/TC 97 Information Technology) і Технічного Комітету 83 Міжнародної електротехнічної комісії (англ. IEC/TC 83) з Підкомітетом 47B Міжнародної електротехнічної комісії (англ. IEC//SC 47B) якій під'єднався пізніше. Метою об'єднання були збори в один комітет з інформаційним технологіям діяльності двох родинних організацій.

Офіційна мета ОТК 1 складається в розробці, підтримці, просуванні й сприянні розвитку IT стандартів необхідних глобальному ринку для узгодження вимог виробників і споживачів, що включають:

– розробку й розвиток IT систем і засобів їхньої розробки;

– продуктивність і якість IT продуктів і систем;

– безпека IT систем і інформації;

– портативність прикладного програмного забезпечення;

– інтероперабельність IT продуктів і систем;

– уніфікацію інструментів і засобів розробки;

– гармонізацію IT словника;

– юзабіліті й ергономічність дизайну користувальницьких інтерфейсів.

Членство в ОТК 1 доступно для будь-яких делегатів країн (англ. National Body), так само як і членство в кожній із двох головних організаціях. Членом може бути будь-який учасник (У-члени, англ. P-members) або спостерігач (Н-члени, англ. O-members), залежно від здатності брати участь у голосуванні за пропонованими стандартами й іншими продуктами. Організації можуть висувати як представників контактних осіб (англ. Liaison Members), що як входять в ISO/МЕК так і зовнішніх. До представників країн не висувається ніяких вимог по участі в усіх (або одному) підкомітетах. У деяких випадках, підкомітети можуть бути створені для рішення нових питань (підкомітет 37 був створений по такому принципу в 2002 році), або розформовані якщо проблеми розв'язувані в підкомітеті більше не актуальні.

Состав ОТК 1 значно збільшився в серпні й вересні 2007 року, напередодні голосування по чернетці міжнародного стандарту Office Open XML.

Підкомітети

Більша частина роботи проходить у підкомітетах (ПК) які займаються певними питаннями. Більшість підкомітетів має кілька робочих груп (РГ). [1]:

1. ПК 02 (SC 02): Набори кодування символів.
2. ПК 06 (SC 06): Телекомунікації й інформаційний обмін між системами. Включає 4 робочі групи.
3. ПК 07 (SC 07): Програмне забезпечення й системотехніка. Включає 13 робочих груп.
4. ПК 17 (SC 17): Карти й ідентифікація особистості. Включає 8 робочих груп.

5. ПК 22 (SC 22): Мови програмування, засобу розробки інтерфейси системного програмного забезпечення. Включає 13 робочих груп.

6. ПК 23 (SC 23): Мультимедійна цифрова інформація для інформаційного обміну й зберігання. Включає 2 робочі групи.

7. ПК 24 (SC 24): Комп'ютерна графіка, обробка зображень і подання просторових даних. Включає 3 робочі групи.

8. ПК 25 (SC 25): Взаємодія IT устаткування. Включає 3 робочі групи.

9. ПК 27 (SC 27): Методи інформаційної безпеки. Включає 5 робочих груп.

10. ПК 28 (SC 28): Офісне встаткування. Включає 5 робочих груп.

11. ПК 29 (SC 29): Кодування аудіо, зображень, мультимедійної та гіпермедійної інформації. Включає 2 робочі групи:

– РГ 01 (WG 01): Кодування нерухливих зображень. Включає 2 підгрупи: Joint Photographic Experts Group; Joint Bi-level Image Experts Group.

– РГ 11 (WG 11): Кодування зображень, що рухаються, і аудио (англ. Moving Picture Experts Group – MPEG).

12. ПК 31 (SC 31): Автоматична ідентифікація й методи запису інформації. Включає 6 робочих груп.

13. ПК 32 (SC 32): Управління й обмін інформацією. Включає 4 робочі групи.

14. ПК 34 (SC 34): Опис і мова обробки документів. Включає 6 робочих груп:

– РГ 01 (WG 01): Опис інформації – мови розмітки(SGML, DSDL, і т.д.).

– РГ 02 (WG 02): Подання інформації.

– РГ 03 (WG 03): Асоціювання інформації.

– РГ 04 (WG 04): Office Open XML.

– РГ 05 (WG 05): Інтероперабельність документів.

– РГ 06 (WG 06): OpenDocument Format.

15. ПК 35 (SC 35): Користувальницькі інтерфейси. Включає 7 робочих груп.

16. ПК 36 (SC 36): Інформаційні технології для навчання, утворення й тренінгу. Включає 7 робочих груп:

– РГ 01 (WG 01): Словник.

– РГ 02 (WG 02): Коллаборативні технології.

– РГ 03 (WG 03): Інформація для що навчається.

– РГ 04 (WG 04): Управління й доставка навчання, утворення й тренінгу.

– РГ 05 (WG 05): Забезпечення якості й наочних основ.

– РГ 06 (WG 06): Міжнародні стандартизовані профілі.

– РГ 07 (WG 07): ITLET – культура, мова й індивідуальні потреби.

17. ПК 37 (SC 37): Біометрія. Включає 6 робочих груп.

18. ПК 38 (SC 38): Розподілені платформи додатків і сервіси.

Крім того в складі ОТК 1 виділена спеціальна робоча група (СРГ) і наступні робочі групи:

19. СРГ (SWG-A): Доступність.

20. РГ 06 (WG 06): Корпоративне управління в IT.

21. РГ 07 (WG 07): Сенсорні мережі.

SCORM

SCORM (англ. Sharable Content Object Reference Model, «зразкова модель об'єкта вмісту для спільного використання») – збірник специфікацій і стандартів, розроблений для систем дистанційного навчання.

В останній специфікація IMS CP була доповнена декількома спеціальними елементами, узятими з AICC CMI001 (значення цих елементів або передаються в навчальний об'єкт за допомогою Javascript API, або використовуються системою для управління навігацією по навчальних об'єктах, що входить у пакет).

Версії SCORM 1.0 і SCORM 1.1 минулого тестовими й поширювалися у вузьких колах для випробування й збору відкликань. У жовтні 2001 року вийшла версія SCORM 1.2 [4], що початку активно поширюватися. У той же час група

ADL продовжувала займатися доробкою SCORM, зокрема, поліпшенням можливостей навігації.

В 2002 році закінчилася спільна робота IMS Global (англ.), Ariadne і IEEE LTSC [5] по доробці специфікації IMS MD до рівня стандарту. Стандарт IEEE 1484.12.1 одержав назву LOM(англ. Learning Object Metadata), і завдяки зворотній сумісності з IMS MD, може використовуватися в SCORM-Пакетах для опису метаданих.

Також ADL вирішила оформити модель взаємодії як офіційний міжнародний стандарт, у зв'язку із чим звернулася в комітет зі стандартизації IEEE LTSC. Робоча група LTSC у контакті з AICC доробила специфікацію взаємодії, у результаті чого в 2003 році було випущено два офіційних стандарти:

– IEEE 1484.11.1 – Data Model For Content To Learning Management System Communication (Опис моделі даних, переданих між навчальним матеріалом і LMS);

– IEEE 1484.11.2 – ECMAScript API For Content To Runtime Services Communication (Опис способу взаємодії між навчальним матеріалом і сервісами періоду виконання за допомогою Javascript).

Тим часом консорціум IMS Global в 2003 році випустив специфікацію IMS Simple Sequencing [6] (IMS SS), що містить вимоги до опису послідовностей проходження навчального матеріалу. Ця специфікація лягла в основу розроблювальної ADL специфікації SCORM SN (англ. Sequencing and Navigation).

У січні 2004 року вийшла перша редакція SCORM 1.3 (яка отримала позначення SCORM 2004). У ній розділ SCORM RTE був представлений описом стандартів IEEE 1484.11 (із API що змінились, що стали називати SCORM API 2004), доповненим спеціальними елементами ADL, використовуваними для організації навігації, докладно описаної в новому розділі SCORM SN. У розділ SCORM CAM був внесений стандарт IEEE LOM замість IMS MD, а також додані вимоги до опису навігації по пакеті відповідно до IMS SS. У липні того ж року вийшла небагато змінена друга редакція SCORM 2004.

У червні 2006 року Міністерство оборони США наказало, щоб всі розробки в області електронного навчання відповідали вимогам SCORM.

Пізніше в SCORM 1.3 минулого внесені ще деякі зміни: у жовтні 2006 року вийшла третя редакція, а в березні 2009 – четверта – SCORM 2004.

Розділи SCORM 2004

Вступна частина стандарту

Тут утримуються загальні положення й ідеї SCORM.

Модель нагромадження змісту (Content Aggregation Model, CAM)

Ця частина стандарту описує структуру навчальних блоків і пакетів навчального матеріалу. Пакет може містити курс, урок, тест, модуль і т.п. У пакет входять xml-файл (маніфест), де описаний структура пакета, і файли, що становлять навчальний блок. Цей файл повинен мати назву imsmanifest.xml і перебувати в кореневій папці пакета.

Маніфест пакета включає:

– метадані (властивості компонентів навчального матеріалу);

– організацію навчального матеріалу (у якому порядку розташовані компоненти);

– ресурси (посилання на файли, що втримуються в пакеті);

– під-маніфест (xml-файл може містити під-маніфести, що описують окремі частини пакета. Може мати сенс, якщо пакет дуже великої й має складну структуру, щоб не перевантажувати один файл більшим обсягом даних).

Блоки навчального матеріалу, що входять у пакет, можуть бути двох типів: ресурси й поділювані об'єкти вмісту (англ. Sharable Content Object (SCO)).

Ресурс (англ. asset) – елемент, що не взаємодіє із сервером системи управління навчанням (LMS-сервером). Це може бути html-сторінка, просто картинка, звуковий файл, flash-об'єкт і т. п. Ресурс може складатися з декількох файлів (наприклад, html-файл + css-файл с описом його стилів + js-файл із описом використовуваних у ньому функцій), але з погляду системи й учня ресурс буде розглядатися як єдиний неподільний об'єкт.

Поділюваний об'єкт умісту (SCO) – це елемент, що взаємодіє із системою управління навчанням: повідомляє про хід і результати навчання, одержує й передає додаткові дані й т.п. Як мінімум SCO повідомляє про свій запуск і завершення (шляхом виклику методів Initialize("") і Terminate("") об'єкта API_1484_11, використовуваного системою для взаємодії).

Приклад коду маніфесту SCORM-Пакета:

```
<?xml version="1.0" encoding=" UTF-8"?>
<manifest version="1.3" identifier="8EA33DC1"
xmlns="http://www.imsglobal.org/xsd/imscp_v1p1">
  <metadata>
    <schema>ADL SCORM</schema>
    <schemaversion>2004 4th Edition</schemaversion>
  </metadata>
  <organizations default="09B4C179">
    <organization identifier="09B4C179" structure="hierarchical">
      <title>Зміст</title>
      <item identifier="7D841A9D" isvisible="true"
identifierref="44D33973">
        <title>Приклад об'єкта SCO, взаємодіючого з LMS</title>
      </item>
    </organization>
  </organizations>
  <resources xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_rootv1p3">
    <resource identifier="44D33973" adlcp:scormType="sco" type="text/html"
href="sco.htm">
      <file href="sco.htm" />
    </resource>
  </resources>
</manifest>
```

Для передачі пакетів по мережі (наприклад, для завантаження в систему управління навчанням) специфікація SCORM CAM пропонує поміщати вміст пакета в zip-архів. Файл imsmanifest.xml повинен розташовуватися в корені архіву. Інші файли пакета повинні розташовуватися так, як зазначено їхнє розташування в елементах file у вмісті маніфесту. Наприклад, у випадку наведеного раніше коду маніфесту файл sco.htm повинен розташовуватися на тім же рівні, де й imsmanifest.xml, тобто в корені архіву. А якби в маніфесті було

прописано <file href="folder1\sco.htm" />, файл sco.htm повинен був би розташовуватися в папці folder1 в архіві.

Середовище виконання (Run-Time Environment, RTE)

Ця частина стандарту описує взаємодію SCO і системи управління навчанням (англ. Learning Management System, LMS) через програмний інтерфейс додатка (Application Program Interface,API). Вимоги SCORM RTE дозволяють забезпечити сумісність SCO і LMS, щоб кожна система дистанційного навчання могла взаємодіяти з SCO у такий же спосіб, як і будь-якому іншому, відповідному стандарту SCORM. LMS повинна забезпечувати доставку необхідних ресурсів користувачеві, запуск SCO, відстеження й обробку інформації про дії що вчиться, передачу SCO-об'єкту запитуваних даних і збереження одержуваних.

Взаємодія здійснюється через об'єкт API_1484_11, що розташовується в одному з батьківських вікон браузера стосовно вікна навчального об'єкта. Навчальний об'єкт повинен бути запущений або у фреймі (iframe) на сторінці системи управління навчанням, або в спливаючому вікні (за допомогою javascript-виклику window.open). На початку своєї роботи SCO-об'єкт повинен знайти об'єкт API_1484_11 в одному з батьківських вікон, застосувавши для цього алгоритм перебору батьківських вікон (SCORM API Discovery Algorithm), а потім викликати метод Initialize("") цього об'єкта.

Після успішної ініціалізації SCO може запитувати в системи дані за допомогою методу GetValue(«назва_елемента_даних») або посилати дані в систему за допомогою методу SetValue(«назва_елемента_даних», «значення»). Можливі елементи даних і їхніх припустимих значень перераховані в специфікації. Для примусового збереження даних, відправлених у систему, SCO-об'єкт повинен викликати метод Commit("").

Також API дозволяє відслідковувати SCO-об'єкту можливі помилки, що виникають у процесі взаємодії, за допомогою методів GetLastError, GetErrorString і GetDiagnostic.

Наприкінці своєї роботи SCO повинен викликати метод Terminate(""), тим самим повідомивши систему, що робота з ним закінчена і його можна закривати (або переходити до наступного об'єкта).

Приклад найпростішого SCO: (дана html-сторінка запитує в системи управління навчанням ім'я що вчиться, що її відкрив)

```
<html>
    <head>
        <script language="javascript">
        function findAPI(win) { //шукаємо в батьківських вікнах об'єкт із назвою API.
            var findAPITries=0; //будемо вважати кількість спроб, щоб пошук не був нескінченним.
            while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win)) {
                findAPITries++;
                if (findAPITries > 20) return null; //число 20 узяте умовно, теоретично його може й не вистачити.
                win = win.parent;
            }
            return win.API_1484_11;
        }
        function getAPI() { //одержуємо об'єкт API для поточного SCO.
            var theAPI = findAPI(window); //спочатку пробуємо шукати в батьках поточного вікна.
            if ((theAPI == null)) { //якщо не знайшли в батьках поточного вікна,
                if ((window.opener != null) && (typeof(window.opener) != "undefined"))
                    theAPI = findAPI(window.opener); //те спробуємо знайти в батьках вікна, що відкрило поточне.
            }
            return theAPI;
        }
        function start() { //ця функція спрацює в момент відкриття SCO.
            var api = getAPI();
            if (api!=null) {
                api.Initialize("");
                value=api.GetValue("cmi.learner_name");    //запитуємо    в системи ім'я що вчиться,
```

```
                    document.write("Ім'я учня: "+value); //і виводимо його на
екран.
                }
                else document.write("Не вдається підключитися до API
системи.");
            }
            function stop() { //ця функція спрацює в момент закриття SCO.
                var api = getAPI();
                if (api!=null) api.Terminate("");
            }
            </script>
            <title>Приклад об'єкта SCO, взаємодіючого з LMS</title>
        </head>
        <body onLoad="start()" onunload="stop()">
        </body>
    </html>
```

Упорядкування й навігація (Sequencing and Navigation, SN)

Ця частина стандарту описує, як повинна бути організована навігація й надання компонентів навчального матеріалу залежно від дій учня. Вимоги SCORM SN дозволяють підносити учневі матеріал відповідно до індивідуальних особливостей.

Вимоги відповідності (Conformance Requirements)

Ця частина містить повний список вимог на відповідність стандарту SCORM, що перевіряються ADL. Система управління навчанням або редактор навчальних матеріалів може одержати від ADL сертифікат про відповідність вимогам SCORM, якщо він функціонує відповідно до цих вказівок.

Системи управління навчанням, сумісні з SCORM:

– СДН "Доцент", сумісна з SCORM 1.2 і 2004 продукт компанії УНІАР.

– Moodle, вільно розповсюджувана система з відкритим вихідним кодом (php+MySql).

– ILIAS, система с відкритим кодом (php+mysql) під ліцензією GNU.

– iSpring Online, СДН, коммерчский продукт компанії iSpring Solutions, Inc.

– Sakai, вільно розповсюджувана система з відкритим кодом (java).

– SharePointLMS, сумісна з SCORM 1.2 і 2004.

– SABA, комерційний продукт компанії Saba Software Inc.

– WebTutor, сумісна з SCORM 1.2 і 2004.

– AcademLive, комерційна система дистанційного навчання, сумісна з SCORM 1.2 і 2004.

– OLAT.

– ShareKnowledge, сумісна з SCORM 1.2 і 2004.

Moodle

Moodle – система управління курсами (електронне навчання), також відома як система управління навчанням або віртуальним навчальним середовищем (англ.) [2]. Являє собою вільний (що поширюється по ліцензії GNU GPL) веб-додаток, що надає можливість створювати сайти для онлайн-навчання.

Рисунок 2.1 – Демонстраційний сайт Moodle

Система реалізує філософію «педагогіки соціального конструкціонізму» [3] і орієнтована насамперед на організацію взаємодії між викладачем і учнями, хоча підходить і для організації традиційних дистанційних курсів, а також підтримки очного навчання.

Moodle переведена на десятки мов [4], у тому числі й українську і використовується майже в 50 тисячах організацій з більш ніж 200 країн миру [5]. У Україні зареєстровано більше 1000 інсталяцій. Кількість користувачів Moodle у деяких інсталяціях досягає 500 тисяч чоловік [6].

Лідером і ідеологом системи є Martin Dougiamas з Австралії. Проект є відкритим і в ньому бере участь і безліч інших розроблювачів [7]. Русифікацію Moodle здійснює команда добровольців з Росії, Білорусії й України [8].

Фінансування проекту здійснюється в основному за рахунок мережі офіційних партнерів [9], які роблять послуги установки, технічної підтримки, хостингу, консультування, інтеграції, доробки й інші. Всі офіційні партнери виплачують членські внески й відсоток із продажів на користь MOODLE PTY LTD AS TRUSTEE FOR THE MOODLE TRUST [10], що керує Martin Dougiamas. Більша частина найбільш активних розроблювачів ядра Moodle є співробітниками Moodle Pty LTD. [11].

Moodle написана на PHP з використанням SQL-бази даних (MySQL, PostgreSQL, Microsoft SQL Server і ін. БД – використовується ADOdbXML). Moodle може працювати з об'єктами SCO і відповідає стандарту SCORM.

Завдяки розвиненій модульній архітектурі, можливості Moodle можуть легко розширюватися [12] сторонніми розроблювачами. Істотне розширення функціональних можливостей Moodle досягається за рахунок інтеграції підсистеми для організації вебінарів/веб-конференцій. Крім мовної підтримки й шаблонів оформлення, Moodle дозволяє підключати також наступні типи модулів:

– Елементи курсу.

– Звіти адміністратора.

– Типи завдань.

– Плагіни автентифікації.

– Блоки.

– Формати курсів.

– Звіти по курсам.

– Поля бази даних (для елемента курсу «База даних»).

– Плагин передплати на курси.

– Фільтри.

– Звіти по оцінках.

– Формати експорту оцінок.

– Формати імпорту оцінок.

– Портфоліо.

– Типи питань у тестах.

– Формати імпорту/експорту тестів.

– Звіти по тестах.

– Сховища файлів.

– Типи ресурсів.

– Плагини пошуку.

LIAS

ILIAS – вільна система управління навчанням (LMS) і підтримки навчального процесу.

Система поширена у ВНЗ. Базується на Apache, PHP, MySQL, XML. Відповідає стандарту SCORM, офіційна підтримка SCORM v1.2; v2004 RD3 (гарантується незалежність від платформи).

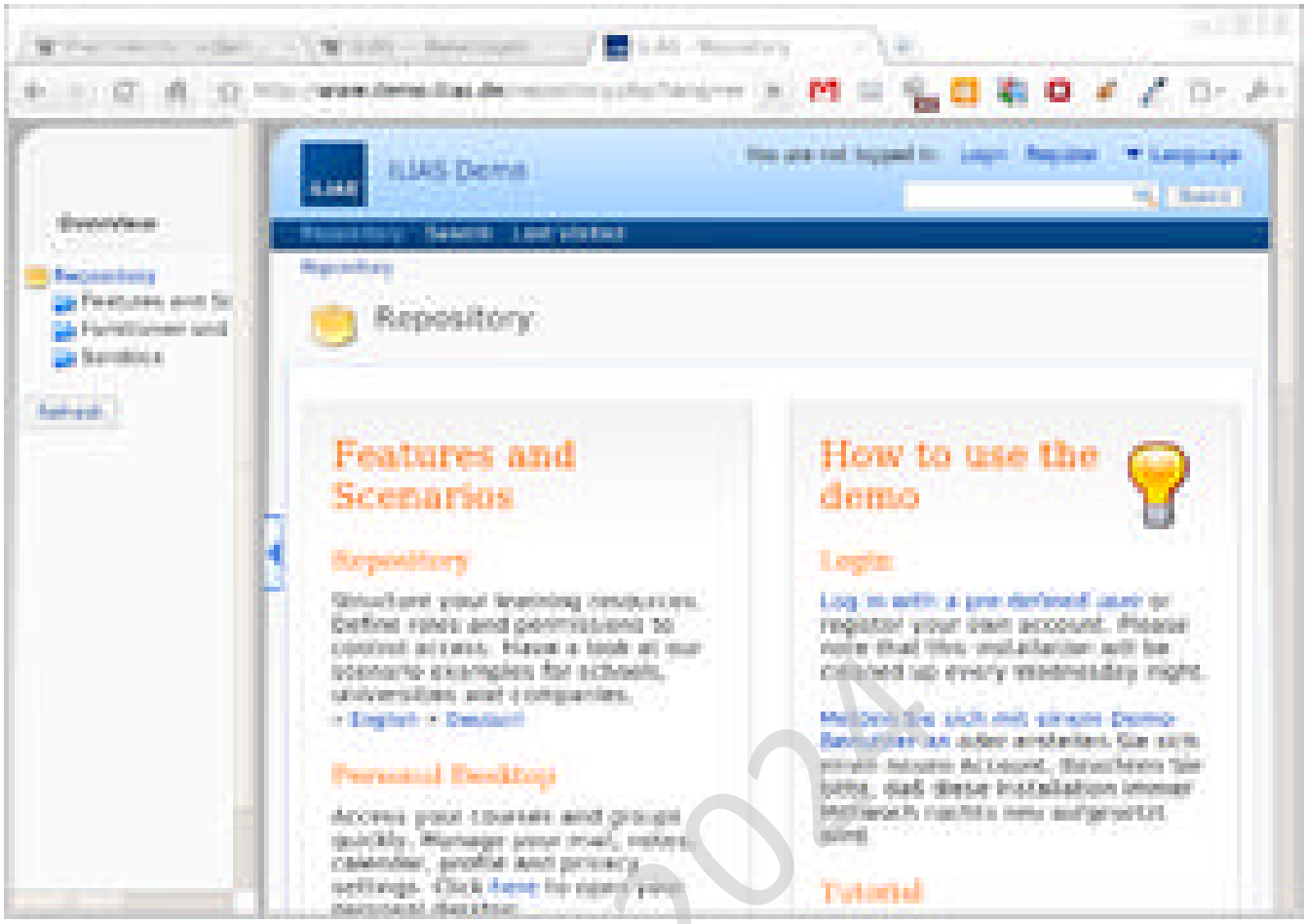

Рисунок 2.2 – Сайт ILIAS

Проект розвивається з 1998 року, одна з попередніх версій – 3.10.6 включає зокрема:

– Особистий робочий стіл з портлетами про однокурсників онлайн, новій пошті, нових повідомленнях на форумі й т.д.

– Контекстна довідкова система для студентів і авторів.

– Інтерфейс користувача й адміністратора.

– Інтерфейс SOAP для імпорту вмісту й користувачів.

– CAS, SOAP, RADIUS, LDAP, Shiboleth authentication/

– Багатомовність, підтримка російської мови.

– Нові версії (релізи безпеки й нові можливості) виходять кожні кілька місяців.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для хмарної системи контролю успішності студентів.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Сучасні інформаційні технології сьогодні досягли високого рівня. Їхнє застосування в різних сферах діяльності дозволяє значно полегшити працю людини, автоматизувати багато процесів, виконувані людиною, одержувати інформацію, що без автоматизації була не доступна.

Більшість із існуючих систем хмарного контролю успішності носять або універсальний характер і не враховують специфіку роботи конкретного навчального закладу, або є убудованими в загальну систему управління конкретного навчального закладу.

Метою роботи є розробка хмарної системи що дозволяє автоматизувати процес контролю успішності студентів на різних типах занять.

Цей процес досить трудомісткий і віднімає значний час викладача при проведенні навчальних занять. Більш складним є підведення підсумків, це важливо для викладача при виставлянні результуючої оцінки по досліджуваній дисципліні. Також навчальним процесом передбачається проміжне оцінювання успішності. Для автоматизації цих процесів доцільно розробити програмне забезпечення.

Сьогодні процес хмарного контролю успішності зводиться до використання кожним викладачем окремо, заздалегідь підготовлених бланків, у яких, найчастіше, у табличному виді представлені умовні позначки які показують успішність. Кожний викладач використовує власні системи й правила обліку. Поточної інформації про загальну успішність конкретного студента в конкретний момент часу при такому підході ні в кого немає.

Основним недоліком даної картки є те, що кожний викладач змушений щораз створювати для себе список навчальної групи. Така картка готується викладачем для всіх своїх груп.

Відсутність такої інформації не дозволяє на ранніх етапах виявляти потенційних боржників або прогульників і вживати заходи пов'язані з підвищенням навчальної дисципліни.

Призначенням розробки хмарної системи є надання програмного забезпечення, що має засоби контролю успішності студентів.

Програмний засіб реалізує функції ручного складання графіка навчального процесу, який реалізується конкретним викладачем для конкретної дисципліни, створення загального списку груп, складання списку досліджуваних дисциплін по кожній групі й ведення статистики успішності студентів.

Логічна модель бази даних

Програмне забезпечення або хмарна система хмарного контролю успішності може використовуватися багатьма викладачами однієї кафедри, факультету, університету, навчального закладу, тоді база даних такої системи зможе акумулювати зведену інформацію про відвідуваність і успішність і може бути використана для оцінки рейтингу студента.

Моделювання пов'язане з поданням семантики предметної області в моделі бази даних, тобто моделювання структур даних, опираючись на зміст цих даних. Найпоширеніша модель «сутність-зв'язок».

Модель «сутність-зв'язок» є концептуальною моделлю, тобто не враховує особливості конкретної СУБД. З її можуть бути отримані всі основні фактографічні моделі даних.

Основними поняттями моделі «сутність-зв'язок» є: сутність, зв'язок і атрибут.

Так для розглянутого завдання для розробки моделі необхідні наступні сутності:

– дисципліна – сутність необхідна для зберігання переліку назв предметів і короткого опису даного предмета;

– викладач – сутність необхідна для зберігання списку викладачів;

– студент – сутність необхідна для зберігання списку студентів;

– група – сутність необхідна для зберігання списків навчальних груп.

Між сутностями «Викладач» і «Дисципліна» існує зв'язок багато до багатьох, тому що теоретично будь-який викладач може вести будь-яку дисципліну (предмет), з іншої сторони ту саму дисципліну може вести будь-який викладач. Незважаючи на те, що запропонована модель досить повно описує поставлене завдання, у ній є деякі недоліки. Ці недоліки полягають у тому, що модель не враховує групування деяких видів даних. Так всі викладачі працюють на певних кафедрах, навчальні групи так само закріплені за певними кафедрами, крім кафедр присутня більші одиниці угруповання – факультети й університети.

З обліком вищесказаного й забезпечення можливості детального проектування бази даних використовується деталізована концептуальна модель предметної області. Ця модель ураховує можливість об'єднання навчальних груп по кафедрах, як адміністративним одиницям. Кафедри так само в себе включають викладачів, при цьому зв'язок між сутностями «Викладач» і «Кафедра» установлена багато до багатьох, тому що той самий викладач може працювати на одній і більше кафедрах. Крім цього сутність «Навчальний план» припускає наявність чітко сформульованого плану вивчення дисципліни. При цьому навчальний план досить сильно залежить від викладача, що викладає ту або іншу дисципліну.

Подальший розвиток інфологічної моделі на даному етапі не доцільний, тому що воно приведе до проектування логічної моделі конкретної СУБД.

Побудована модель дозволяє досить добре представити взаємодію між сутностями розглянутої предметної області й перейти до етапу функціонального проектування хмарної системи.

У рамках даної роботи розробляється програмне забезпечення для автоматизації процесу хмарного контролю успішності студентів.

Програмне забезпечення дозволить в електронному виді вести облік поточної успішності студентів.

До переваг програмного забезпечення можна віднести можливість використання її в режимі клієнт-сервер, що дозволяє забезпечити доступ до єдиної БД із будь-якої аудиторії кафедри. Застосування даної програми в навчальному процесі дозволяє контролювати відвідуваність студентів, формувати зведену оцінку про успішність студентам по всіх дисциплінах наявних у БД.

Її зручно використовувати на особистих ноутбуках викладачів, що мають можливість зв'язку з кафедральною мережею по каналах WiFi, а так само на лаборантських робочих місцях.

Система розроблювальна на основі web технологій буде доступна всім зареєстрованим користувачем у будь-якому місці.

Програмний засіб повинне представляти із себе web сайт, що представляє користувачеві дані про успішність у табличному виді. Хмарна система повинна бути універсальною, тобто не залежати від типу дисциплін і звичок викладача.

3.2 Розробка структурної схеми

Наведемо формалізований опис задачі бакалаврського дослідження. КН розглянуто у вигляді структурної схеми, наведеної на рисунку 3.1, та системи функціоналів вигляду:

$$F = \Phi(G^0, Q, C, G, t),$$

де G^0 – вектор, що описує стан підсистеми контролю початкового рівня знань особи, яку навчають $G^0(G^0_1, G^0_2, ..., G^0_n)$;

Q – вектор, що описує стан підсистеми психологічного тестування $Q(Q_1, Q_2, ...Q_n)$;

C – вектор, що описує стан навчальної системи $C(C_1, C_2, ..., C_n)$;

G – вектор, що описує стан підсистеми контролю отриманих знань $G(G_1, G_2, ..., G_n)$;

t – умовний параметр (час, витрачені кошти, апаратні ресурси та ін.), відносно якого визначається ефективність процесу (за умов, що $t > 0$).

Основні складові процесу моделювання визначені таким чином:

1. На етапі завдання початкових умов визначають:

– стан вектора $Q(Q_1, Q_2, ...Q_n)$, що дозволяє обґрунтувати адекватний вибір методу навчання з урахуванням думок експертів щодо аналізу особистісних характеристик і ментальних особливостей користувачів;

– стан вектора $G^0(G^0{}_1, G^0{}_2, ..., G^0{}_n)$ для вибору стартового рівня складності досліджуваного матеріалу.

2. Викладач виконує наповнення бази даних системи, визначає надійність і валідність тестових завдань. Залежно від стану векторів $Q(Q_1, Q_2, ...Q_n)$ і $G^0(G^0{}_1, G^0{}_2, ..., G^0{}_n)$ виконується настроювання навчальної підсистеми під кожного користувача індивідуально, тобто для кожної особи, яку навчають, формується свій вектор $C(C_1, C_2, ..., C_n)$.

3. Процес навчання полягає в послідовному дискретному переведенні підсистеми, яку навчають, з початкового стану β_1 у кінцевий β_k. При цьому вона проходить ряд проміжних станів $\beta_1, \beta_2, ..., \beta_k$, кожен з яких характеризується визначеним рівнем знань (ступенем навченості) $G_1, G_2, ..., G_k$.

4. Кожен *i-й* рівень знань описується n-мірним вектором $\overline{G}_i(g_1, g_2, ..., g_n)$, де $g_1, g_2, ..., g_n$ – компоненти вектора, кожен з яких кількісно визначає властивості навчальної підсистеми в даний момент часу. Перехід навчальної підсистеми зі стану β_i у стан β_{i+1} відбувається в результаті введення порції інформації C_i. При цьому стан β_{i+1} визначається не тільки порцією інформації C_i, але й попереднім станом підсистеми β_i, тобто $G_{i+1} = f(G_i, C_i)$.

5. Метою навчання є перехід підсистеми, яку навчають з початкового стану β_1, що характеризується рівнем знань G_i, у кінцевий β_k, якому відповідає рівень G_k.

6. Показник ефективності процесу навчання стосовно до окремого користувача визначено як значний приріст навченості ΔG_t, що дорівнює $\Delta G_t = \frac{G_k - G_1}{t}$, де G_1 – рівень знань на початку навчання; G_k – рівень знань наприкінці навчання.

7. Для аналізу якості навчання будемо використовувати коефіцієнт засвоєння інформації, який розраховується за формулою $G' = \frac{G}{C}$. У випадку неадекватної роботи системи викладач вносить необхідні коректування у вибір методу навчання та проводить повторне визначення початкового рівня навчання.

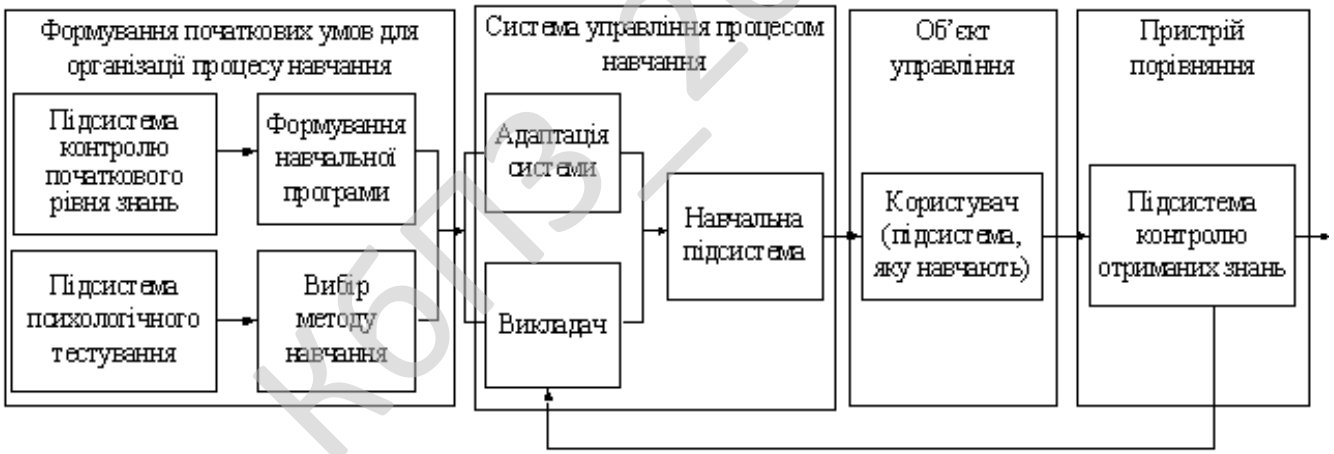

Рисунок 3.1 – Структурна схема системи

Для одержання комплексного показника індивідуальних характеристик того, кого навчають $Q(Q_1, Q_2, ...Q_n)$, запропонована методика обробки результатів психологічного тестування з використанням методів теорії нечітких множин і відносин з урахуванням думок експертів.

Основою для одержання такого показника є психологічний портрет користувача. Для приведення результатів тестування до єдиної шкали використано лінгвістичні змінні $Q_{x_j}^{T_i}$, які визначають оцінку даної характеристики залежно від здатності користувача до навчання:

$$\langle 'Характеристика', T(Q), X \rangle, \tag{3.1}$$

де '*Характеристика*' – назва лінгвістичної змінної, яка визначає деяку психологічну характеристику $x_j^{T_i}$ за результатами психологічного тестування за методикою T_i;

$T(Q) = \{погано' П', задовільно' З', добре' Д', відмінно' В'\}$ – терм-множина лінгвістичної змінної;

X – базова множина лінгвістичної змінної $'Характеристика'$, яка обумовлена шкалою психологічного тесту T_i в діапазоні $[1, n]$.

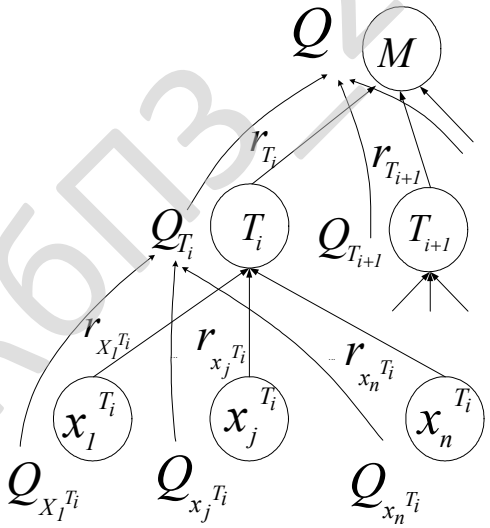

Рисунок 3.2 – Схема отримання інтегрального показника індивідуальних характеристик особи, яку навчають

Методика одержання інтегрального показника індивідуальних характеристик того, кого навчають, наведена у вигляді схеми на рисунку 3.2 та складається з таких етапів:

1. Визначають оцінки індивідуальних характеристик $Q_{x_j}^{T_i}$ з використанням функції належності, показаної на рисунку 3.3.

2. Визначають вплив показника важливості даної характеристики на кінцевий результат експерименту $r_{x_j}^{T_i}$ з терм-множиною *T(R)={'слабо', 'середньо', 'сильно'}*. Визначають правила згортки оцінок індивідуальних характеристик $Q_{x_j}^{T_i}$ з урахуванням впливу показника важливості цих характеристик $r_{x_j}^{T_i}$ на кінцевий результат залежно від вихідних цілей тестування:

$$Q_{T_i} = \bigcup_{j=1}^{m_{T_i}} (Q_{x_j{}^{T_i}} \cdot r_{x_j{}^{T_i}}). \qquad (3.2)$$

Для одержання результатів композиції $Q_{x_j{}^{T_i}} \cdot r_{x_j{}^{T_i}}$ використано таблиці лінгвістичних правил (ТЛП) (табл. 3.1).

Таблиця 3.1 – ТЛП композиції $Q_{x_j{}^{T_i}} \cdot r_{x_j{}^{T_i}}$

$r_{x_j{}^{T_i}}$	$Q_{x_j{}^{T_i}}$			
	«П»	«З»	«Д»	«В»
Сильно	«П»	«З»	«Д»	«В»
Середньо	«П»	«Д»	«В»	«В»
Слабо	«З»	«Д»	«В»	«В»

3. Визначають значення Q за формулою (3.2), замінюючи значення $Q_{x_j{}^{T_i}}$ на Q_{T_i} та $r_{x_j{}^{T_i}}$ на r_{T_i} .

4. У результаті одержуємо інтегральний показник індивідуальних характеристик Q, що визначається областю значень одного з термів лінгвістичної змінної *«Характеристика»*. За допомогою функції відображення (рисунок 3.3) визначають кращий метод навчання.

Таким чином у другому розділі наведена системна модель, яка дозволяє обирати метод навчання на підставі аналізу інтегрального показника індивідуальних характеристик користувачів при організації процесу самоосвіти. Використання такого підходу дозволяє значно підвищити рівень знань у системі освіти, а також скоротити час, витрачений на процес навчання.

Розглянемо організацію процесу навчання та контролю отриманих знань за двома напрямками: за результатами індивідуального та групового навчання. Для організації процесу навчання необхідно структурувати дисципліни, що вивчаються, відповідно до схеми, зображеної на рисунку 3.4: «Дисципліна» – «Модулі» – «Розділи» – «Питання».

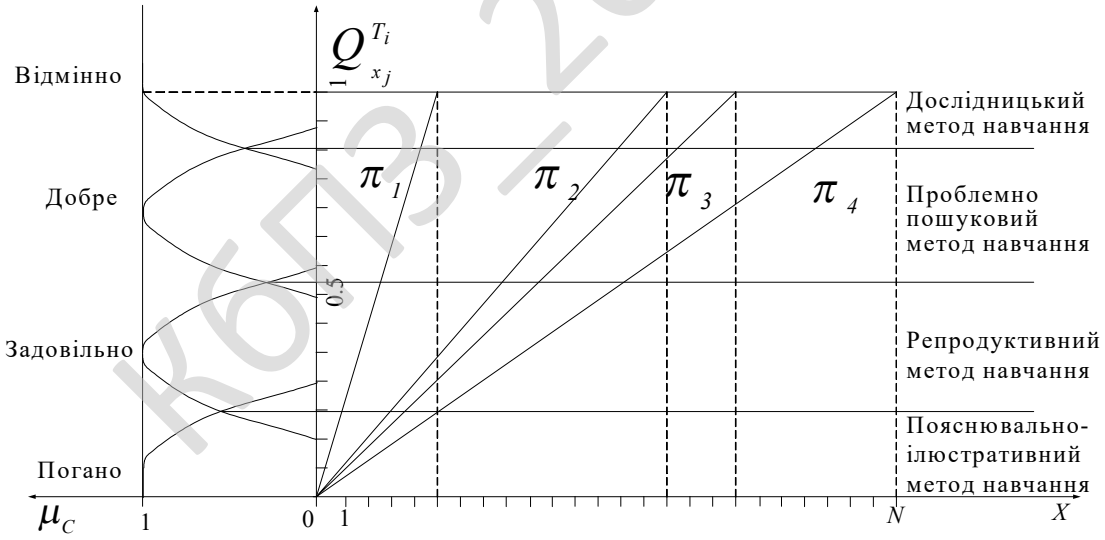

Рисунок 3.3 – Переклад функції приналежності з відносної шкали в універсальну

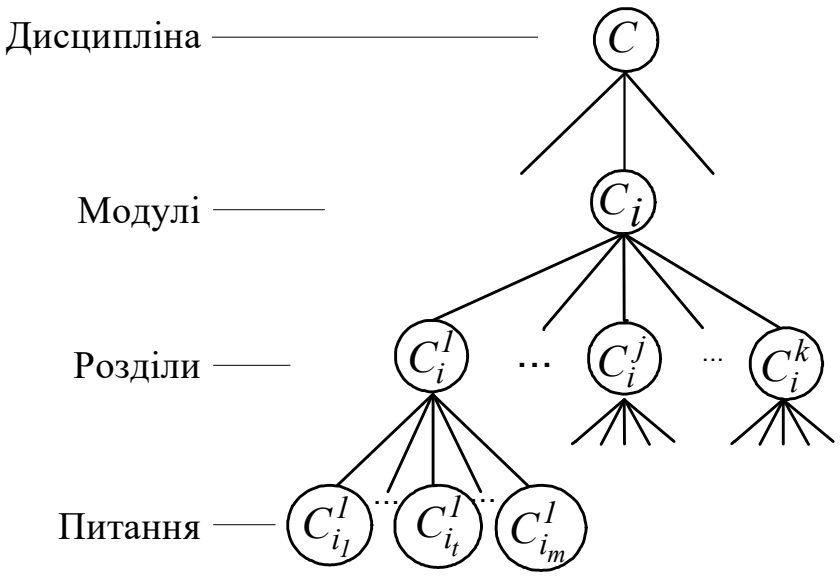

Рисунок 3.4 – Структура дисципліни, яка вивчається

На підставі такої структуризації кожна дисципліна наведена як тривимірна інформаційна матриця (рисунок 3.5, а), у якій кожен шар визначає один модуль дисципліни, що вивчається, а кількість стовпців – кількість питань, розташованих відповідно до вагових коефіцієнтів у порядку їхнього зростання. Для кожного питання ступінь вірогідності отриманого результату d_k визначається за чотирибальною шкалою. Ваговий коефіцієнт питання C_{ij} визначає інформаційну ємність кожного питання. Вагові коефіцієнти розподілені на три рівня за складністю в діапазоні $[0,1]$, що надає можливість формувати завдання рівноважної складності у тому випадку, коли номера питань вибираються випадково. Розподіл питань за їх складністю запропоновано виконувати згідно з чотирма рівнями засвоєння інформації:

– I рівень – це рівень відтворення засвоєної інформації;

– II рівень – це репродуктивний рівень діяльності;

– III рівень – це рівень умінь;

– IV рівень – це рівень навичок.

Таким чином, згідно з вимогами до побудови функціонально валідних та надійних тестів під час перевірки рівня засвоєння знань необхідно використовувати тестові послідовності, які складаються з питань із різних рівнів засвоєння інформації. Максимальна складність питань тесту визначається методом навчання, який було вибрано за методикою, наведеною вище.

Інформаційна матриця дозволяє кількісно оцінити:

– *інформаційну ємність (вагу) модуля* C_i як суму вагових коефіцієнтів питань, котрі його складають, що розраховується за формулою $C_i = \sum_{j=1}^{v} C_{ij}$;

– *інформаційну ємність (вагу) дисципліни* як суму ваг модулів, що її складають: $C = \sum_{i=1}^{M} C_i$

Для оцінки якості отриманих знань побудовано матрицю стану знань (рисунок 3.5, б). Для цього ті складові питання, на які отримані вірні відповіді, визначаються значенням d_k, а невірні – нулями.

Оцінку загальних знань за *і-м модулем* обчислюють як суму добутків вагових коефіцієнтів на показник вірогідності результату всіх питань у межах даного модуля: $G_i = \sum_{j=1}^{v} G_{ij} d_k$.

Оцінка загальних знань з дисципліни дорівнює сумі оцінок загальних знань усіх модулів, які її складають: $G = \sum_{i=1}^{M} G_i$.

Коефіцієнт засвоєння знань з дисципліни G_i' визначається як відношення загальних знань до інформаційної ємності дисципліні $G_i' = \frac{G_i}{C_i}$.

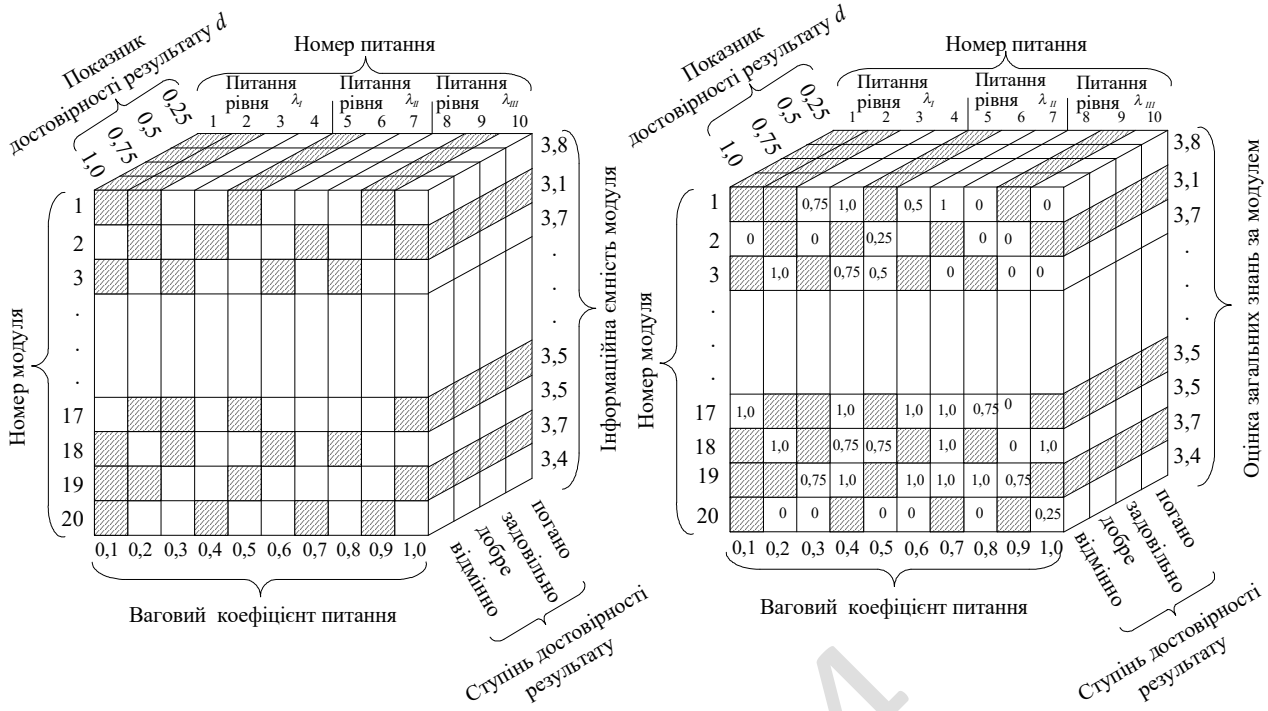

Рисунок 3.5 – Інформаційна матриця та матриця стану знань користувачів: а – інформаційна матриця; б – матриця стану знань

Для переведення отриманого коефіцієнта засвоєння знань з модуля G_i' у бальну оцінку q_λ використано узагальнену шкалу порядку (рисунок 3.6). Використання даної шкали припускає, що найвищий бал при тестуванні можливо одержати тільки в тому випадку, якщо відбулися всі рівні тестування з коефіцієнтом засвоєння знань $G_i' \geq 0{,}7$. Якщо розмірність узагальненої шкали порядку змінюється, то переведення коефіцієнта засвоєння у бальну оцінку здійснюється згідно з формулою:

$$q_{\lambda_s} = \sum_{i=1}^{k} \frac{Q_{\lambda_s}}{k} \cdot G'_{\lambda_i},$$

де q_{λ_s} – бальна оцінка за результатами тестування; k - кількість рівнів засвоєння інформації; Q_{λ_s} – максимальна розмірність шкали; G'_{λ_i} – коефіцієнт засвоєння інформації на рівні λ_i .

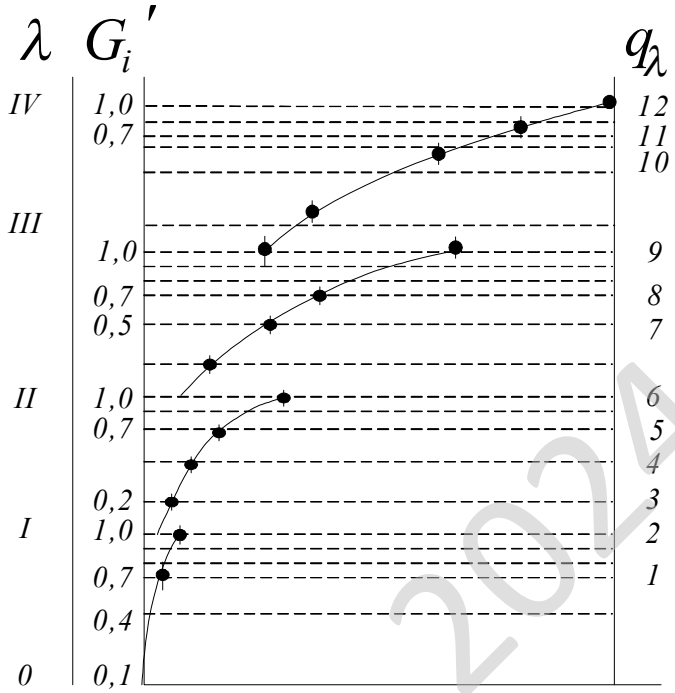

Рисунок 3.6 – Узагальнена шкала порядку

Використання такого підходу до оцінювання знань користувачів дозволяє отримувати структурну оцінку знань, одержаних під час навчання, та виділяти саме ті підрозділи навчального матеріалу, які були недостатньо засвоєні в процесі навчання, на відміну від загальної оцінки, яку отримують іншими методами тестового контролю знань.

Для обробки результатів тестового контролю знань після вивчення дисципліни групою користувачів запропоновано використовувати метод статистичного контролю якості продукції, який широко застосовується в

промисловості, а саме метод статистичного одиночного приймального контролю за якісними ознаками для невеликих партій виробів.

Оскільки обсяг вибірки *n* відносно обсягу партії *N* порівняно невеликий, то для вибору плану статистичного оцінювання вибрано гіпергеометричний закон розподілу, який характеризує кількість незадовільних результатів тестування *d* у вибірці з *n* питань, вибраних без повернення з партії результатів тестування *N*, із часткою невірних відповідей *P*. При цьому передбачається вилучення вибірок кінцевого обсягу:

$$P(d,n;D,N)=\frac{C_d^D C_h^H}{C_n^N}=\frac{C_{np}^{NP} C_{nf}^{NF}}{C_n^N}.$$

Основні позначення наведено у табл. 3.2.

Таблиця 3.2 – Основні позначення закону розподілу

Показники	Вибірка	Партія
Кількість результатів	*n*	*N*
Кількість незадовільних результатів	*d*	*D*
Кількість задовільних результатів	*h*	*H*
Частка незадовільних результатів	*p*	*P*
Частка задовільних результатів	*f*	*F*

При організації одиночного приймального контролю знань за якісними ознаками необхідно задати набір показників:

– *N* – обсяг партії;

– *n* – обсяг вибірки;

– P_1 – прийнятний рівень знань;

– *P* – частка незадовільних результатів тесту;

– P_2 – припустимий відсоток незадовільних результатів;

– α_1 – ризик студента або ймовірність відбраковування результатів тестування, якість яких P_1;

– G'_{min} – нижня межа припустимого значення коефіцієнта засвоєння знань, результати вважаються незадовільними, якщо $G' < G'_{min}$.

Методика проведення приймального контролю за якісними ознаками складається з таких етапів:

1. Визначають:

– кількість незадовільних результатів $D_1=NP_1$;

– приймальне число A шляхом накопичення ймовірностей гіпергеометричного розподілу доти, поки $P(d \le A) = 0,95$;

– бракувальне число $R=A+1$;

– ризик користувача $\alpha_1' = 1 - P(d \le A)$;

– ризик викладача $\beta_2 = P(d \le A)$.

2. Якщо $\alpha_1' \ne \alpha_1$, то шляхом зміни обсягу вибірки n забезпечується точне наближення до заданих показників якості. Таким чином, одержуємо необхідний обсяг вибірки n з партії результатів тестування N для забезпечення необхідних показників якості α_1 і β_2.

3. Одержують робочу характеристику (РХ). Функція робочої характеристики являє собою математичне сподівання ймовірності β прийняття партії залежно від частки незадовільних результатів P у партії.

4. Визначають середню вихідну якість (СВЯ) за такою формулою: $СВК = 0\alpha + P\beta = P\beta$.

3.3 Розробка функціональної схеми

Наведемо функціональну структуру комп'ютерної системи навчання (КСН) на основі розробленої системної моделі та методик організації процесу самоосвіти. КСН складається з наступних основних підсистем:

– підсистема вибору методу навчання на основі аналізу інтегрального показника індивідуальних характеристик користувачів;

– навчальна підсистема, що складається з електронного підручника й віртуального лабораторного практикуму;

– підсистема контролю отриманих знань.

Розроблена КСН може бути використана як інструментальна оболонка для створення дистанційних курсів з різних дисциплін. Реалізовано мережний режим роботи.

Функціональна схема хмарної системи контролю успішності студентів зображена на рисунку 3.7. Вимоги до функціональних характеристик. Система повинна виконувати функцію реєстрації й ідентифікації викладача або студента. Залежно від ролі користувача, можна виділити наступні функціональні вимоги:

– створення облікового запису викладача;

– додавання нових груп;

– створення списків груп;

– створення списку досліджуваних дисциплін;

– створення переліків лабораторних, контрольних і курсових робіт;

– ведення статистики успішності й відвідуваності;

– перегляд розкладу інших викладачів без можливості внесення змін;

– використання єдиної бази даних списків груп і студентів;

– формування статистики успішності;

– перегляд успішності студентами;

– розсилка SMS з результатами контролю успішності, та при необхідності з попередженнями щодо низької успішності та відсутності студентів на заняттях;

– наявність можливості доступу до бази даних по мережі.

Вимоги до надійності. У програмній системі необхідно передбачити захист даних від випадкового видалення й зміни. Тільки викладачі, наділені відповідними правами, які зареєстровані на сервері бази даних.

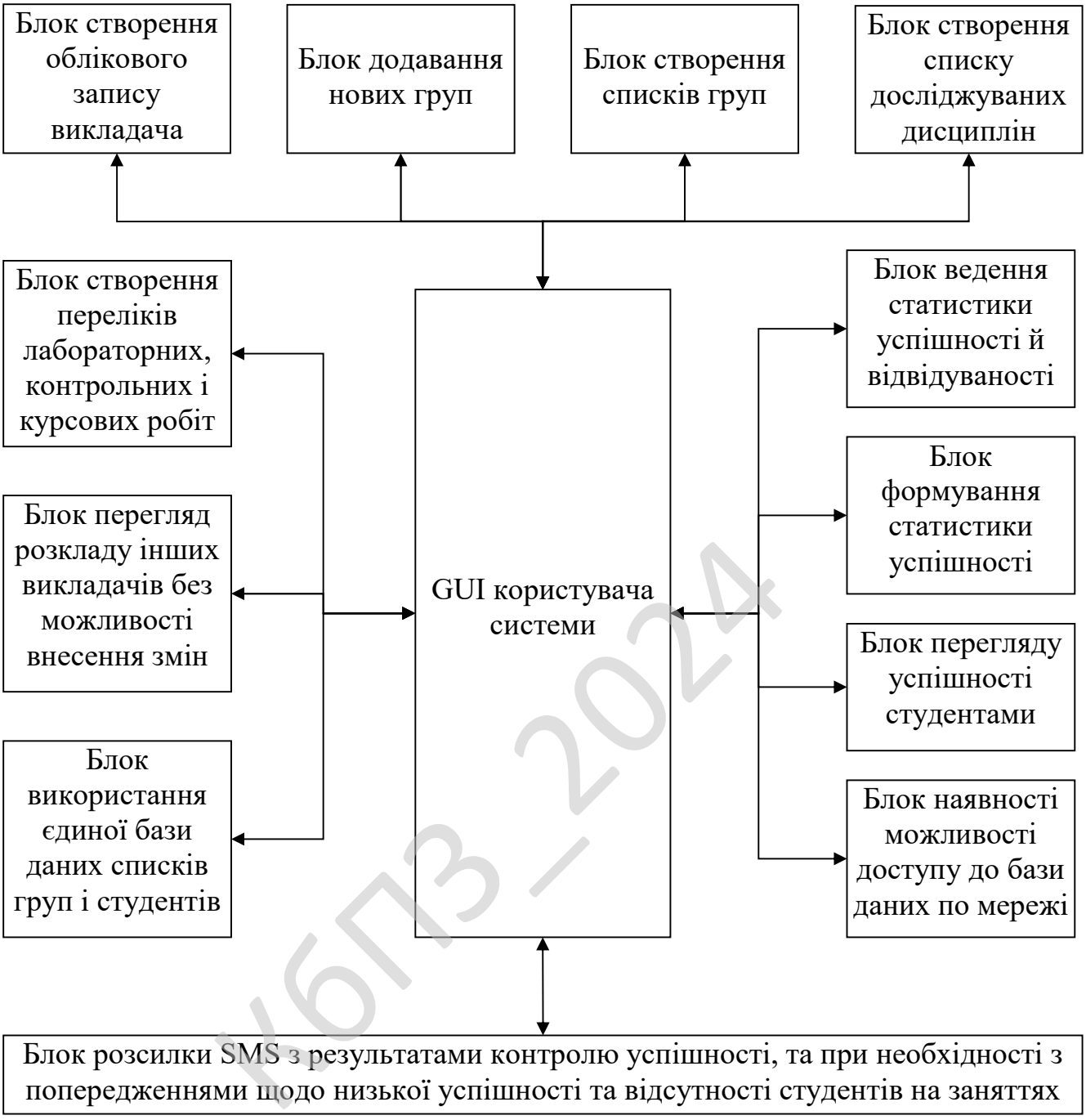

Рисунок 3.7 – Функціональна схема системи

З метою надійності програмного забезпечення вона повинна задовольняти наступним вимогам:

– розроблена програма повинна мати засоби захисту від помилкових дій користувачів;

– гарантувати схоронність даних при збоях у роботі зовнішніх пристроїв;

– виключити можливість доступу до файлів конфігурації користувачам.

Для підвищення надійності необхідно вжити наступних заходів:

– зконфігурувати апаратні й програмні засоби відповідно до технічних вимог;

– періодично здійснювати резервне копіювання інформації;

– регулярно перевіряти цілісність бази даних;

– підтримувати справність мережного встаткування.

Для роботи із системою кожний викладач повинен бути зареєстрований у БД. Вхід у систему здійснюється стандартним для web додатків способом, тобто уведенням логіну й пароля, що видає адміністратор. Однією з особливостей даного модуля буде є перехід на головну сторінку. При переміщенні попадаємо не в саму базу даних, а на сторінку з оригінальним дизайном, що містить багато графічної інформації. Далі всю роботу виконує меню, у якому й утримуються посилання для переходу до потрібних даних: факультети, кафедри й далі групи. Дизайну сторінки приділяється немаловажне значення. Хоча можна сказати, що для подібних систем це не так важливо, а важливий сам доступ до інформації й малий обсяг системи для коректної роботи. Однак розроблювачами зроблений великий акцент на оформлення, що, як передбачається, зробить перехід між сторінками сайту більше приємним і збільшить число відвідувань даного ресурсу. У даному вікні можна вказати тему заняття й тип заняття: лекція, практика, лабораторна. Список типів занять має можливість розширювати адміністратор.

Список предметів, по яких ведеться облік успішності заповнюється адміністратором через власний інтерфейс. Основна мета хмарної системи контролю успішності студентів є автоматизація роботи з даними, що супроводжують навчальний процес груп студентів і надання доступу, як посадовим особам, так і самим студентам до необхідній їм інформації з будь-якого комп'ютера, підключеного до мережі Інтернет.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.8. Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ далі можемо перейти до блоку створення облікового запису викладача з подальшим переходом до блоків додавання нових груп, створення списків груп, створення списку досліджуваних дисциплін. Крім цього через блок формування статистики успішності можемо перейти до блоків перегляду успішності студентами, блоку наявності можливості доступу до бази даних по мережі з подальшим переходом до блоку БД системи чи через блок ведення статистики успішності й відвідуваності перейти до блоків створення переліків лабораторних, контрольних і курсових робіт та перегляду розкладу інших викладачів без можливості внесення змін.

Рисунок 3.8 – Діаграма взаємодії процесів

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що після запуску програми спочатку відбувається виділення пам'яті ПЗ. Потім здійснюється:

– Ініціалізація підсистеми додавання бібліотек блоків.

– Сканування та підключення бібліотек блоків.

– Підключення до БД системи.

– Є доступ до таблиць та бібліотек (запит).

– Автентифікація користувача.

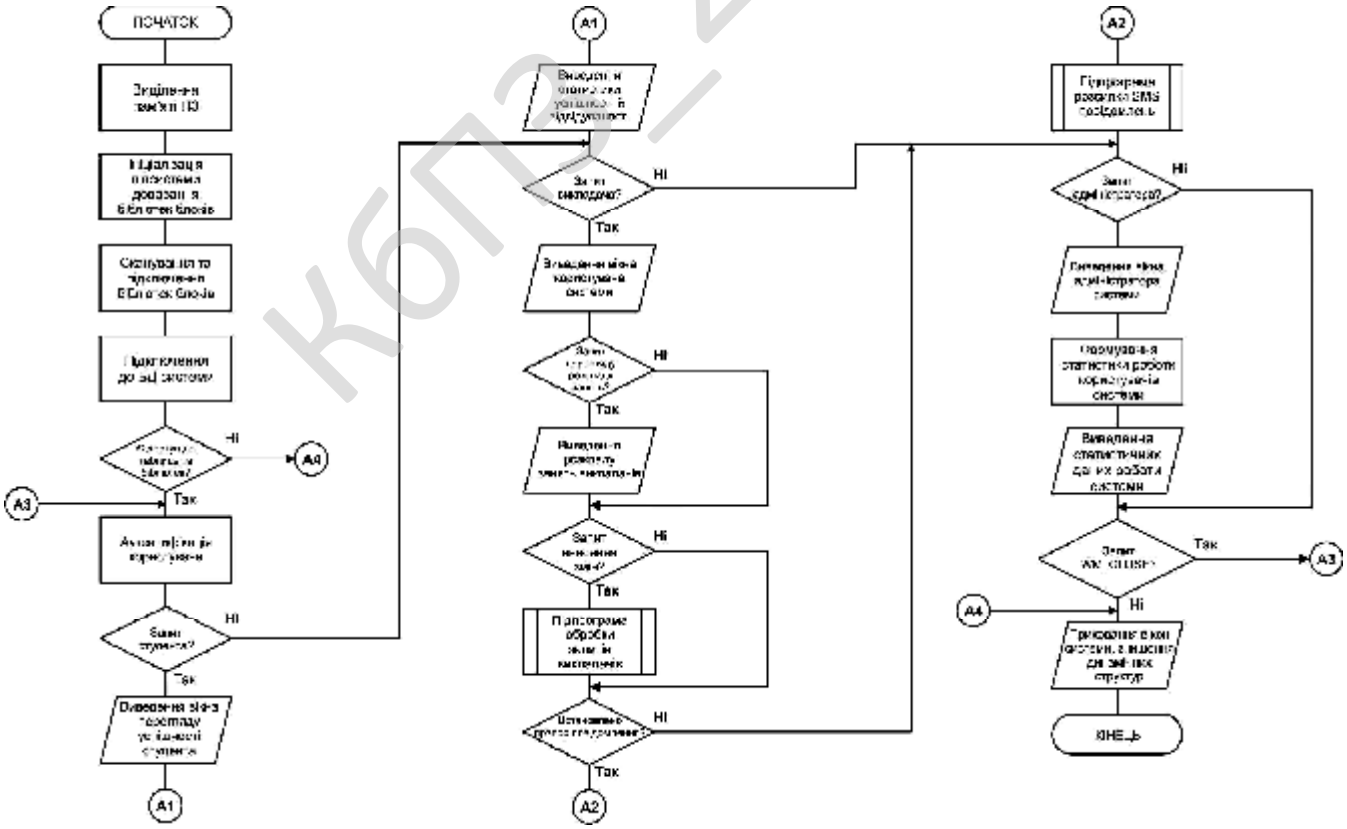

Рисунок 4.1 – Блок-схема основної програми

– Запит студента?

– Виведення вікна перегляду успішності студента.

– Виведення статистики успішності й відвідуваності.

– Запит викладача?

– Виведення вікна користувача системи.

– Запит перегляду розкладу занять?

– Виведення розкладу занять викладачів .

– Запит внесення змін?

– Підпрограма обробки запитів викладачів.

– Встановлено прапор повідомлення (запит).

– Підпрограма розсилки SMS повідомлень.

– Запит адміністратора?

– Виведення вікна адміністратора системи.

– Формування статистики роботи користувачів системи.

– Виведення статистичних даних роботи системи.

– Запит WM_CLOSE?

– Приховання вікон системи.

На рисунку 4.2 зображено роботу підпрограми обробки запитів викладачів. Де відбуваються наступні дії:

– Запит редагування облікового запису викладача?

– Виведення вікна редагування облікового запису викладача.

– Введення даних облікового запису викладача.

– Запит додавання нових груп?

– Виведення вікна додавання нових груп.

– Введення даних групи.

– Запит додавання студентів у групу?

– Виведення вікна додавання студентів у групу.

– Введення даних студентів групи.

– Запит редагування дисциплін?

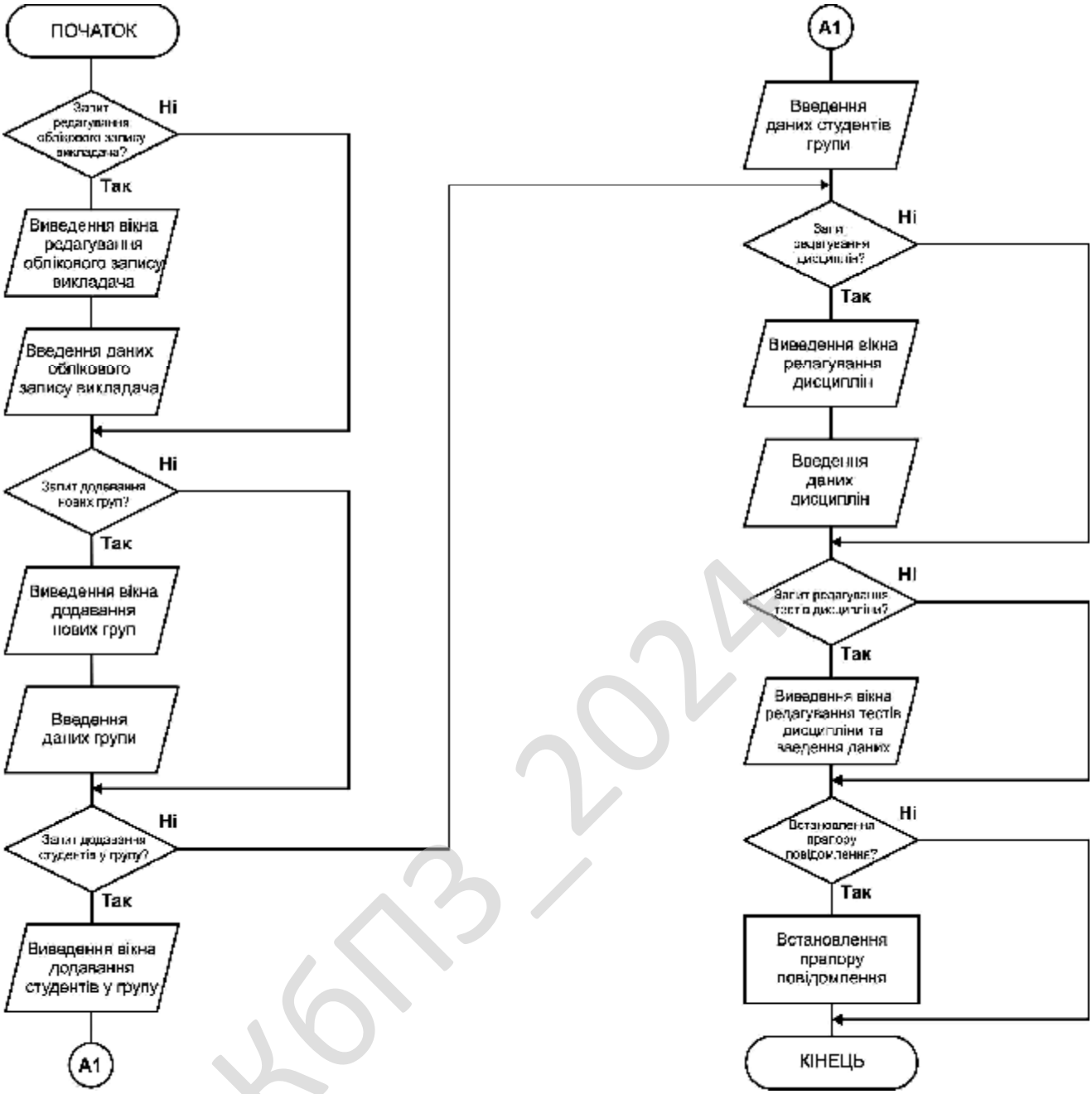

Рисунок 4.2 – Блок-схема підпрограми обробки запитів викладачів

– Виведення вікна редагування дисциплін.

– Введення даних дисциплін.

– Запит редагування тестів дисципліни?

– Виведення вікна редагування тестів дисципліни.

– Введення даних тестів дисципліни.

– Встановлення прапору повідомлення (запит).

– Встановлення прапору повідомлення.

На рисунку 4.3 зображено роботу підпрограми розсилки SMS повідомлень. Де відбуваються наступні дії:

– Завантаження модуля SOAP системи.

– Відправлення даних автентифікації шлюзу TurboSMS (Auth запит).

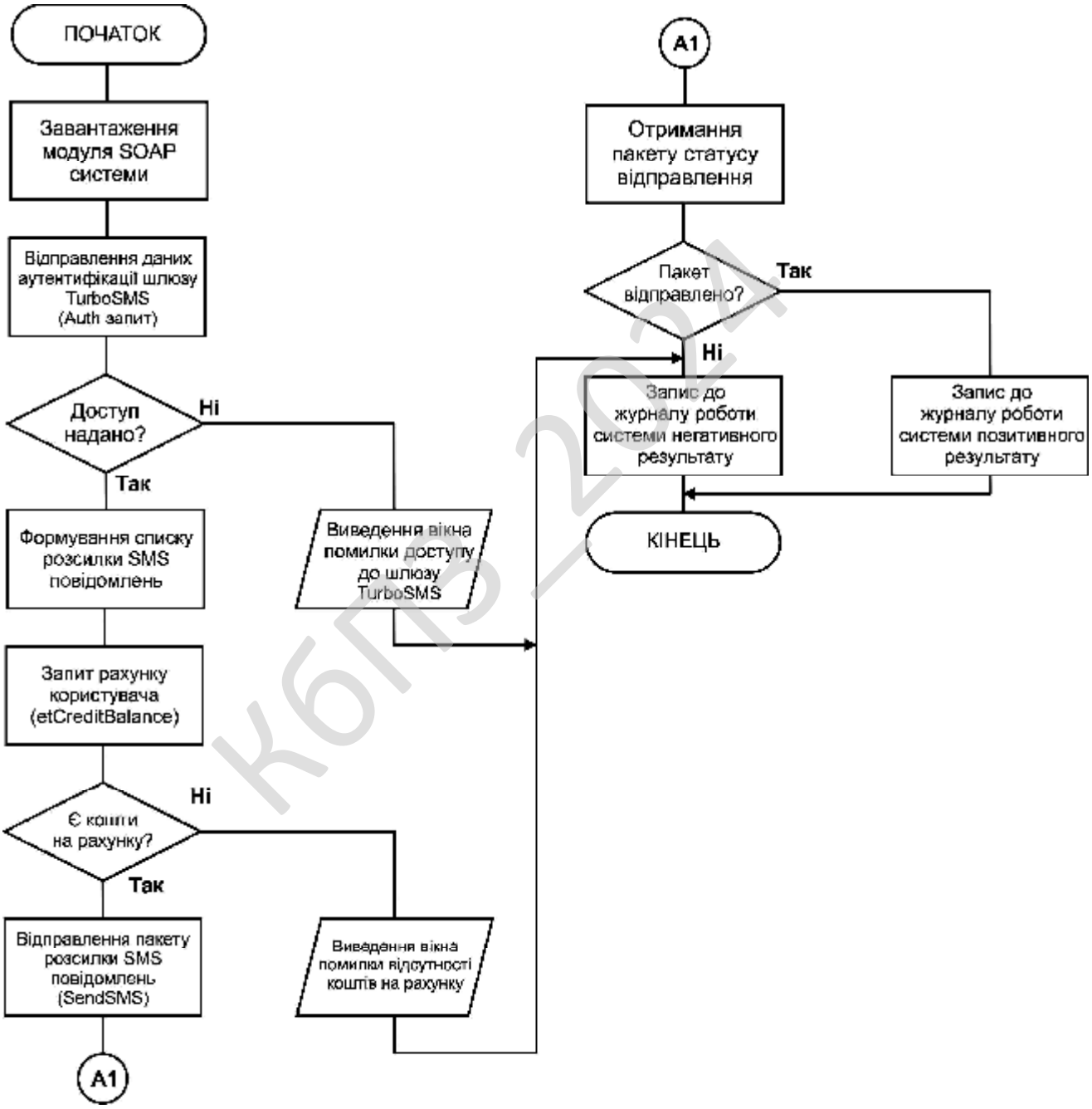

Рисунок 4.3 – Блок-схема підпрограми розсилки SMS повідомлень

– Доступ надано (запит).

– Формування списку розсилки SMS повідомлень.

– Запит рахунку користувача (etCreditBalance).

– Є кошти на рахунку (запит).

– Відправлення пакету розсилки SMS повідомлень (SendSMS).

– Отримання пакету статусу .

– Пакет відправлено (запит).

– Запис до журналу роботи системи позитивного результату.

Опис алгоритмів функціонування системи

Виклик віддалених процедур (Remote procedure call, RPC) – протокол, що дозволяє програмі, запущеній на одному комп'ютері бути викликаною на іншому комп'ютері без написання безпосередньо коду для цієї операції.

Концепція віддаленого виклику процедур. Ідея виклику віддалених процедур (Remote Procedure Call – RPC) полягає у розширенні механізму передачі управління і даних усередині програми, що виконується на одній машині, на передачу керування й даних через мережу. Засоби віддаленого виклику процедур призначені для полегшення організації розподілених обчислень.

Найбільша ефективність використання RPC досягається в тих додатках, в яких існує інтерактивний зв'язок між віддаленими компонентами з невеликим часом відповідей і відносно малою кількістю переданих даних.

Такі програми називаються RPC-орієнтованими.

Існує ряд технологій, що забезпечують RPC, зокрема:

– Sun RPC (бінарний протокол на базі TCP та UDP).

– Net Remoting (бінарний протокол на базі TCP, UDP, HTTP).

– XML-RPC (текстовий протокол на базі HTTP).

– SOAP – Simple Object Access Protocol (текстовий протокол на базі HTTP).

– Java RMI – Java Remote Method Invocation.

– JSON-RPC JavaScript Object Remote Procedure Calls (текстовий, на базі HTTP).

Характерними рисами виклику локальних процедур є:

– Асиметричність, тобто одна із сторін є ініціатором;

– Синхронність, тобто виконання процедури, що викликає віддалену процедуру, призупиняється з моменту видачі запиту і відновлюється тільки після повернення з викликаної процедур.

Реалізація віддалених викликів істотно складніше реалізації викликів локальних процедур. Почнемо з того, що оскільки викликаюча і викликана процедури виконуються на різних машинах, то вони мають різні адресні простори, і це створює проблеми при передачі параметрів і результатів, особливо якщо машини не ідентичні. Так як RPC не може розраховувати на розподілену пам'ять, то це означає, що параметри RPC не повинні містити вказівників на дані в нестековій пам'яті і що значення параметрів повинні копіюватися з одного комп'ютера на інший.

Але в реалізації RPC беруть участь щонайменше два процеси – по одному в кожній машині. У випадку, якщо один з них аварійно завершиться, можуть виникнути такі ситуації: при аварії викликаючої процедури віддалено викликані процедури стануть «осиротілими», а при аварійному завершенні віддалених процедур стануть «знедоленими батьками» викликаючи процедури, які будуть безрезультатно чекати відповіді від віддалених процедур.

Крім того, існує ряд проблем, пов'язаних з неоднорідністю мов програмування та операційних середовищ: структури даних і структури виклику процедур, підтримувані в будь-якому однією мовою програмування, не підтримуються точно так само у всіх інших мовах.

Ці та деякі інші проблеми вирішує широко поширена технологія RPC, що лежить в основі багатьох розподілених операційних систем, наприклад Plan 9.

Базові операції RPC

Щоб зрозуміти роботу RPC, розглянемо спочатку виконання виклику локальної процедури у звичайній машині, що працює автономно.

Нехай це, наприклад, буде системний виклик:

```
count = read (fd, buf, nbytes);
```

де fd – ціле число, buf – масив символів, nbytes – ціле число.

Щоб здійснити виклик, викликаючи процедура заштовхує параметри в стек у зворотному порядку. Після того, як виклик read виконаний, він поміщає значення, що повертається в регістр, переміщує адресу повернення і повертає управління викликаючій процедурі, яка вибирає параметри з стека, повертаючи його в початковий стан.

Зауважимо, що в мові С параметри можуть викликатися або по посиланню (by name), або за значенням (by value). По відношенню до викликаючої процедури параметри-значення є ініціалізованими локальними змінними. Викликана процедура може змінити їх, і це не вплине на значення оригіналів цих змінних в викликаючій процедурі.

Якщо в визвану процедуру передається покажчик на змінну, то зміна значення цієї змінної визваної процедури тягне зміну значення цієї змінної і для викликаючої процедури. Цей факт дуже істотний для RPC.

Існує також інший механізм передачі параметрів, який не використовується в мові С. Він називається call-by-copy/restore і полягає в необхідності копіювання викликаючою програмою змінних в стек у вигляді значень, а потім копіювання назад після виконання виклику поверх оригінальних значень викликаючої процедури.

Рішення про те, який механізм передачі параметрів використовувати, приймається розробниками мови. Іноді це залежить від типу переданих даних. У мові С, наприклад, цілі та інші скалярні дані завжди передаються за значенням, а масиви – по посиланню.

Ідея, покладена в основу RPC, полягає в тому, щоб зробити виклик віддаленої процедури по можливості схожим до виклику локальної процедури. Іншими словами – зробити RPC прозорим: викликаючій процедурі не потрібно знати, що викликається процедура знаходиться на іншій машині, і навпаки.

RPC досягає прозорості наступним шляхом. Коли викликана процедура дійсно є віддалена, в бібліотеку поміщається замість локальної процедури інша версія процедури, яка називається клієнтським стабом (stub – заглушка).

Подібно оригінальної процедурі, стаб викликається з використанням викликаючої послідовності, так само відбувається переривання при зверненні до ядра. Тільки на відміну від оригінальної процедури він не поміщає параметри в регістри і не запитує у ядра дані, замість цього він формує повідомлення для відправки ядру віддаленої машини.

Етапи виконання RPC

Після того, як клієнтський стаб був викликаний програмою-клієнтом, його першим завданням є заповнення буфера відправлені повідомленням. У деяких системах клієнтський стаб має єдиний буфер фіксованої довжини, що заповнюється щоразу з самого початку при вступі кожного нового запиту. В інших системах буфер повідомлення являє собою пул буферів для окремих полів повідомлення, причому деякі з цих буферів вже заповнені.

Цей метод особливо підходить для тих випадків, коли пакет має формат, що складається з великої кількості полів, але значення багатьох з цих полів не змінюються від виклику до виклику.

Потім параметри повинні бути перетворені у відповідний формат і вставлені в буфер повідомлення. До цього моменту повідомлення готове до передачі, тому виконується переривання за викликом ядра.

Коли ядро отримує управління, воно перемикає контексти, зберігає регістри процесора і карту пам'яті (дескриптори сторінок), встановлює нову карту пам'яті, яка буде використовуватися для роботи в режимі ядра.

Оскільки контексти ядра і користувача розрізняються, ядро має точно скопіювати повідомлення в свій власний адресний простір, так, щоб мати до нього доступ, запам'ятати адресу призначення (а, можливо, і інші поля заголовка), а також воно має передати його мережевому інтерфейсу. На цьому завершується робота на клієнтській стороні.

Включається таймер передачі, і ядро може або виконувати циклічне опитування наявності відповіді, або передати управління планувальником, який обере будь-який інший процес на виконання. У першому випадку прискорюється виконання запиту, але відсутнє мультипрограмування.

На стороні сервера біти, що надходять, поміщаються приймаючої апаратурою або у вбудований буфер, або в оперативну пам'ять. Коли вся інформація буде отримана, генерується переривання. Обробник переривання перевіряє правильність даних пакета і визначає, якому стабу слід їх передати.

Якщо жоден із стабів не очікує цей пакет, обробник повинен або помістити його в буфер, або взагалі відмовитися від нього. Якщо є очікуючий стаб, то повідомлення копіюється йому. Нарешті, виконується переключення контекстів, в результаті чого відновлюються регістри і карта пам'яті, приймаючи ті значення, які вони мали в момент, коли стаб зробив виклик receive.

Тепер починає роботу серверний стаб. Він розпаковує параметри і поміщає їх відповідним чином в стек. Коли все готово, виконується виклик сервера. Після виконання процедури сервер передає результати клієнту. Для цього виконуються всі описані вище етапи, тільки в зворотному порядку.

14 етапів виконання RPC: Клієнт 1. Виклик стабу 2. Підготувати буфер 3. Упакувати параметри 4. Заповнити поле заголовка 5. Обчислити контрольну суму в повідомленні 6. Переривання до ядра 7. Черга пакету на виконання 8. Передача повідомлення контролеру по шині QBUS 9. Час передачі по мережі Ethernet Сервер 10. Отримати пакет від контролера 11. Процедура обробки переривання 12. Обчислення контрольної суми 13. Переключення контексту в простір користувача 14.

Динамічне зв'язування

Розглянемо питання про те, як клієнт задає місце розташування сервера. Одним з методів вирішення цієї проблеми є безпосереднє використання адресу сервера в клієнтській програмі. Недолік такого підходу – його надзвичайна негнучкість: при переміщенні сервера, або при збільшенні числа серверів, або при

зміні інтерфейсу у всіх цих та багатьох інших випадках необхідно перекомпілювати всі програми, які використовували жорстке завдання адреси сервера.

Для того, щоб уникнути всіх цих проблем, в деяких розподілених системах використовується так зване динамічне зв'язування.

Початковим моментом для динамічного зв'язування є формальне визначення (специфікація) сервера. Специфікація містить ім'я файлу-сервера, номер версії і список процедур-послуг, що надаються даним сервером для клієнтів. Для кожної процедури дається опис її параметрів із зазначенням того, чи є даний параметр вхідним чи вихідним щодо сервера.

Деякі параметри можуть бути одночасно вхідними і вихідними – наприклад, деякий масив, який надсилається клієнтом на сервер, модифікується там, а потім повертається назад клієнтові.

Формальна специфікація сервера використовується в якості вихідних даних для програми-генератора стабів, яка створює як клієнтські, так і серверні стаби. Потім вони поміщаються у відповідні бібліотеки. Коли користувальницька (клієнтська) програма викликає якусь процедуру, визначену в специфікації сервера, відповідна стаб-процедура зв'язується з двійковим кодом програми.

При запуску сервера найпершою його дією є передача свого серверного інтерфейсу спеціальною програмою, так званим binder'ом. Цей процес, відомий як процес реєстрації сервера, включає передачу сервером свого імені, номера версії, унікального ідентифікатора і описувача місцезнаходження сервера. Описувач системно незалежний і може представляти собою IP, Ethernet, X.500 або ще яку-небудь адресу. Крім того, він може містити й іншу інформацію, наприклад дані, що стосуються автентифікації.

Коли клієнт викликає одну з віддалених процедур перший раз, наприклад, read, клієнтський стаб бачить, що він ще не підключений до сервера, і надсилає повідомлення binder-програмі з проханням про імпорт інтерфейсу потрібної версії

потрібного сервера. Якщо такий сервер існує, то binder передає описувач і унікальний ідентифікатор клієнтському стабу.

Клієнтський стаб при посилці повідомлення із запитом використовує в якості адреси описувач. У повідомленні містяться параметри і унікальний ідентифікатор, який ядро сервера використовує для того, щоб направити надійшло повідомлення в потрібний сервер у випадку, якщо їх декілька на цій машині.

Цей метод, що полягає в імпорті / експорті інтерфейсів, володіє високою гнучкістю. Наприклад, може бути кілька серверів, що підтримують один і той же інтерфейс, і клієнти розподіляються по серверах випадковим чином.

У рамках цього методу стає можливим періодичне опитування серверів, аналіз їх працездатності та, у разі відмови, автоматичне відключення, що підвищує загальну відмовостійкість системи. Цей метод може також підтримувати автентифікацію клієнта. Наприклад, сервер може визначити, що він може бути використаний тільки клієнтами з певного списку.

Однак у динамічного зв'язування є недоліки, наприклад, додаткові накладні витрати (тимчасові витрати) на експорт та імпорт інтерфейсів. Величина цих витрат може бути значною, тому що багато клієнтські процеси існують короткий час, а при кожному старті процесу процедура імпорту інтерфейсу повинна бути знову виконана.

Крім того, у великих розподілених системах може бути вузьким місцем програма binder, а створення декількох програм аналогічного призначення також збільшує накладні витрати на створення і синхронізацію процесів.

Протокол SOAP

SOAP – протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML.

Спочатку SOAP призначався, в основному, для реалізації віддаленого виклику процедур (RPC), а назва булла абревіатурою: Simple Object Access Protocol – простий протокол доступу до об'єктів.

Зараз протокол використовується для обміну повідомленнями в форматі XML, а не тільки для виклику процедур. SOAP є розширенням мови XML-RPC.

SOAP може використовуватись з будь-яким протоколом прикладного рівня: SMTP, FTP, HTTP та інш.

Проте, його взаємодія з кожним із цих протоколів має свої особливості, які потрібно відзначити окремо. Найчастіше SOAP використовується разом з HTTP. SOAP є одним із стандартів, на яких ґрунтується технологія веб-сервісів.

Для зв'язку з шлюзом TurboSMS по SOAP протоколу необхідно встановити модулі та бібліотеки для його підтримки в середовищі розробки.

Якщо необхідні модулі та бібліотеки відсутні, можна генерувати XML потрібної структури, який потрібно відправляти на сервер методом RAW POST.

Для того, щоб мати можливість підключатися до шлюзу по протоколу SOAP, необхідно в розділі налаштувань шлюзу поставити галочку "SOAP" в полі «Способи підключення».

Шлюз TurboSMS підтримує такі процедури:

– Auth авторизує користувача на сервері.

– GetCreditBalance повертає залишок на рахунку користувача.

– SendSMS відправляє повідомлення.

– GetMessageStatus повертає статус доставки повідомлення.

Крім транспортного протоколу для виклику методів сервера необхідний єдиний протокол, що описує формат повідомлень виклику методів сервера. В якості такого протоколу використовується SOAP – Simple Object Access Protocol.

Специфікація SOAP 1.1 можна знайти в Web за адресою www.w3.org/TR/SOAP. SOAP дозволяє використовувати виклик віддалених процедур (RPC) через HTTP. Запити кодуються у форматі XML. Це має свої переваги і недоліки.

Переваги:

– Легкість сприйняття тексту запиту людиною («читабельність»).

– Наявність парсерів XML, що дозволяють досить просто здійснювати аналіз вступників повідомлень.

З недоліків можна відзначити більший обсяг повідомлень в порівнянні з бінарними форматами представлення даних.

Нижче представлений приклад SOAP запиту:

```
       POST / examples HTTP/1.1
User-Agent: Radio UserLand/7.0 (WinNT)
Host: localhost: 81
Content-Type: text / xml; charset = utf-8
Content-length: 474
SOAPAction: "/ examples"
<? Xml version = "1.0"?>
<SOAP-ENV: Envelope
SO AP-ENV: encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
xmlns: SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
xmlns: SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
xmlns: xsd = "http://www.w3.org/1999/XMLSchema"
xmlns: xsi = "http://www.w3.org/1999/XMLSchema-instance">
   <SOAP-ENV:Body>
      <m:getStateName xmlns:m="http://www.soapware.org/">
         <statenum xsi:type="xsd:int"> 41 </ statenum>
         </ M: getStateName>
      </ SOAP-ENV: Body>
    </ SOAP-ENV: Envelope>
```

Розглянемо заголовок запиту. Формат URI в першому рядку запиту не специфіковані. Наприклад це може бути просто / або, як у нашому прикладі / examples.

User Agent і Host повинні бути вказані. Content-Type, тобто тип вмісту запиту, природно, text/xml. Content-Length – дліна запиту.

SoapAction – значення даного поля використовується для передачі повідомлення потрібного оброблювачу повідомлень сервера. Як правило, значення SoapAction збігається з URI в першому рядку запиту.

Тіло запиту являє собою документ у форматі XML. Кореневої тег SOAP-ENV: Envelop містить всередині себе тег SOAP-ENV: Body, що містить опис

викликається процедури. У нашому прикладі повідомлення описує запит на виклик процедури getStateName з параметром statenum рівним 41.

Відповідь сервера при успішному виклику виглядає так:

```
        HTTP/1.1 200 OK
Connection: close
Content-Length: 499
Content-Type: text / xml; charset = utf-8
Date: Wed, 28 Mar 2001 5:05:04 GMT
Server: UserLand Frontier/7.0-WinNT
<? Xml version = "1.0"?>
<SOAP-ENV: Envelope
SOAP-E NV: encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
xmlns: SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
xmlns: SOAP-ENV = http://schemas.xmlsoap.org/soap/envelope/
xmlns: xsd = http://www.w3.org/1999/XMLSchema
  xmlns: xsi = "http://www.w3.org/1999/XMLSchema-instance">
   <SOAP-ENV:Body>
      <m:getStateNameResponse xmlns:m="http://www.soapware.org/">
         <Result Xsi:type="xsd:string"> South Dakota </ Result>
         </ M: getStateNameResponse>
      </ SOAP-ENV: Body>
   </ SOAP-ENV: Envelope>
```

Відзначимо лише найбільш важливі моменти:

Простір імен в описі відповіді (тег <m: getStateNameResponse>) Повинен збігатися з простором імен в запиті. У нашому прикладі простір імен – m

Ім'я тега опису відповіді формується додаванням слова Response до імені викликається процедури. У нашому випадку це m: getStateNameResponse.

Delphi XE5 дозволяє створювати як сервера, так і клієнтів Web Services. Ми почнемо розгляд з створення сервера. Створення сервера Web Services в Delphi XE5 складається з наступних етапів:

– Опис інтерфейсу сервера, тобто методів, які будуть доступні для виклику клієнту.

– Реалізація методів сервера.

– Створення проекту Delphi XE5 і включення в нього результатів перших двох кроків.

Послідовно опишемо кожен з етапів.

Опис інтерфейсу сервера

У Delphi XE5 при створенні сервера Web Services методи доступні для виклику клієнту описуються у вигляді invokable інтерфейсів. Invokable інтерфейс – це інтерфейс для методів якого доступна RTTI (інформація про типи на етапі виконання).

Для того щоб із звичайного інтерфейсу зробити invokable досить вказати директиву компіляції {$M +}. Після цього всі нащадки і сам інтерфейс міститимуть RTTI.

В ієрархії VCL вже є такий інтерфейс IInvokable. Таким чином, при написанні сервера простіше всього успадкувати свій інтерфейс від Iinvokable. Крім того необхідно зареєструвати свій інтерфейс в invocation registry. Реєстрація дозволяє серверу визначити клас, який реалізує методи інтерфейсу, а клієнту отримати опис методів, підтримуваних сервером. Реєстрація здійснюється викликом методу InvRegistry.RegisterInterface в секції initialization модуля. Так як інтерфейс використовується не тільки сервером, але і клієнтом, то бажано визначити його в окремому модулі Delphi XE5. Для прикладу ми розробимо сервер, який здійснюватиме конвертування. У IDE Delphi XE5 виберемо пункт меню File/New/Unit. В отриманому порожньому модулі визначимо інтерфейс сервера:

```
unit u_Intrf;
interface
type
  IEncodeDecode = interface (IInvokable)
    ['{32B3312E-684C-444D-88DB-13DE6F535F6D}']
    function US (Value: Currency): Currency; stdcall;
    function RS (Value: Currency): Currency;
stdcall;
end;
implementation
uses   InvokeRegistry;
initialization
  InvRegistry.RegisterInterface (TypeInfo (IEncodeDecode));
```

```
end.
```

Зверніть увагу, що рядок ['{32B3312E-684C-444D-88DB-13DE6F535F6D}'] – це GUID інтерфейсу, для коректної роботи прикладу. Необхідно згенерувати його, а не вводити вручну або копіювати з наведеного тексту. Генерація GUID в IDE Delphi XE5 викликається натисканням Ctrl + Shift + G. У разі використання у функціях інтерфейсі скалярних типів даних генерація SOAP повідомлень відбувається автоматично без додаткових зусиль з боку програміста.

Якщо ж необхідно використовувати складні типи даних, такі як статичні масиви, набори і класи, то необхідно створити і зареєструвати клас-спадкоємець від TRemotableXS і перевизначити методи XSToNative і NativeToXS. Дані методи конвертують строкове і бінарне представлення даних один в одного.

Реалізація методів сервера

Найбільш простим способом реалізації інтерфейсу на сервері є створення і реєстрація в invocation реєстрі класу-спадкоємця від TInvokableClass. Клас TInvokableClass має дві чудові особливості:

Invocation реєстр знає про те, як створити екземпляр цього класу і його спадкоємців при запиті клієнтом викликів методів інтерфейсу.

Так як клас TinvokableClass спадкоємець TInterfacedObject, то він уміє звільнити пам'ять у випадку, коли число посилань на нього дорівнює 0, що полегшує програмісту життя.

Текст модуля реалізації представлений нижче:

```
unit u_Impl;
interface
uses InvokeRegistry, u_Intrf;
type
  TEncodeDecode = class (TInvokableClass, IEncodeDecode)
  protected
    function US (Value: Currency): Currency; stdcall;
    function RS (Value: Currency): Currency; stdcall;
  end;
implementation
{TEncodeDecode}
```

```
function TEncodeDecode.RS (Value: Currency): Currency;
begin
  Result: = Value / 30;
end;
function TEncodeDecode.US (Value: Currency): Currency;
begin
  Result: = Value * 30;
end;
initialization
  InvRegistry.RegisterInvokableClass (TEncodeDecode);
end.
```

У разі, якщо необхідно наслідувати клас від TInvokableClass, необхідно створити і зареєструвати метод-фабрику класу, який зможе створювати екземпляри класу. Метод повинен бути типу TCreateInstanceProc = procedure (out obj: TObject);. При цьому примірник повинен вміти ліквідувати себе, якщо кількість посилань що використовують його клієнтів стане нульовим. При реєстрації такого класу методу InvRegistry.RegisterInvokableClass другим параметром необхідно передати ім'я методу-фабрики класу.

Розробка клієнта Web Services в Delphi XE5

Умовно розробку клієнта можна розбити на дві частини:

– Отримання опису інтерфейсу сервера.

– Написання коду виклику методів сервера.

У разі розробки сервера на Delphi XE5 існує модуль з описом інтерфейсу сервера на мові Object Pascal, тобто перший етап може бути пропущений. У разі якщо сервер був розроблений з використанням інших мов або модуль з описом інтерфейсу не доступний, необхідно одержати опис інтерфейсу у форматі WSDL або XML.

```
<? Xml version = "1.0"?>
<Definitions   xmlns =
"http://schemas.xmlsoap.org/wsdl/"  xmlns:xs="http://www.w3.org/2001/XMLSchema"nam
e="IEncodeDecodeservice"targetNamespace="http://www.borland.com/soapServices/"xmln
s:tns="http://www.borland.com/soapServices/" xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"xmlns: soapenc =
"http://schemas.xmlsoap.org/soap/encoding/">
<messagename="USRequest">
```

```
   <Part name = "Value" type = "xs:double"   />
   </ Message>
<messagename="USResponse">
   <Part name = "return" type = "xs:double"   />
   </ Message>
<messagename="RSRequest">
   <Part name = "Value" type = "xs:double"   />
   </ Message>
<messagename="RSResponse">
   <Part name = "return" type = "xs:double"   />
   </ Message>
<portTypename="IEncodeDecode">
<operationname="US">
   <Input   message = "tns: USRequest"   />
   <Output   message = "tns: USResponse"   />
   </ Operation>
<operationname="RS">
   <Input   message = "tns: RSRequest"   />
   <Output   message = "tns: RSResponse"   />
   </ Operation>
   </ PortType>
<bindingname="IEncodeDecodebinding"type="tns:IEncodeDecode">
   <Soap: binding   style = "rpc" transport =
"http://schemas.xmlsoap.org/soap/http" />
<operationname="US">
<Soap: operation soapAction="urn: u_Intrf-IEncodeDecode # US2RUS"   />
       <input>
       <Soap:body use="encoded"encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/"namespace="urn:u_Intrf-
IEncodeDecode"/>
          </Input>
<output>
<Soap: body use="encoded"encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/"namespace="urn:u_Intrf-
IEncodeDecode"/>
</ Output>
</ Operation>
<operationname="RS">
<Soap: operation   soapAction = "urn: u_Intrf-IEncodeDecode # RS"   />
<input>
       <Soap:body use="encoded"encodingStyle=
```

```
       "http://schemas.xmlsoap.org/soap/encoding/"namespace="urn:u_Intrf-
IEncodeDecode"   />
   </ Input>
<output>
<Soap:body use="encoded"encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"namespace="urn:u_Intrf-
IEncodeDecode"   />
   </ Output>
   </ Operation>
   </ Binding>
<servicename="IEncodeDecodeservice">
<portname="IEncodeDecodePort"binding="tns:IEncodeDecodebinding">
<Soap: address location =
       "http://localhost/cgi-bin/Server.exe/soap/IEncodeDecode"   />
   </ Port>
   </ Service>
   </ Definitions>
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Blowfish, який є симетричним алгоритмом шифрування, тобто таким, у якому ключ шифрування дорівнює ключу дешифрування. Він є мережею Фейштеля, у якій кількість ітерацій дорівнює 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину в межах 448 біт. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко.

Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автентифікація сторін і узгодження загальних параметрів і секретів. При реалізації на 32-бітних мікропроцесорах з більшим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа й шифрування даних. Розширення ключа перетворює ключ довжиною, принаймні, 448 біт у кілька масивів підключів загальною довжиною 4168 байт.

В основі алгоритму лежить мережа Фейштеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа й даних. Операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Елементи алгоритму:

1. P – масив, що складається з вісімнадцяти 32-бітних підключів:

$$P_1, P_2, ..., P_{18}.$$

2. Чотири 32-бітних S-boxes с 256 входами кожний. Перший індекс означає номер S-box, другий індекс – номер входу.

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$
$$S_{2,0}, S_{2,1}, \ldots S_{2,255};$$
$$S_{3,0}, S_{3,1}, \ldots S_{3,255};$$
$$S_{4,0}, S_{4,1}, \ldots S_{4,255};$$

Шифрування

Входом є 64-бітний елемент даних X, що ділиться на дві 32-бітні половини, X_l і X_r.

$$X_\text{l} = X_\text{l} \text{ XOR } P_\text{i}$$
$$X_\text{r} = F(X_\text{l}) \text{ XOR } X_\text{r}$$
$$\text{Swap } X_\text{l} \text{ and } X_\text{r}$$

Функція F

Розділити X_l на чотири 8-бітних елементи A, B, C, D.

$F(X_\text{l}) = ((S_{1,A} + S_{2,B} \bmod 2^{32}) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 2^{32}$

Дешифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерація підключів

Підключи обчислюються з використанням самого алгоритму Blowfish.

1. Ініціалізувати перший *P*-масив і чотири *S*-boxes фіксовані рядки.

2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа й т.д. Повторювати цикл доти, поки весь *P*-масив не буде побітово складний з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.

3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключи, описані в пунктах (1) і (2).

4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).

5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.

6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, заміняючи всі елементи *P*-масиву, а потім всі чотири *S*-boxes, виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1 На рисунку зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: БД системи; SMS шлюз; Налаштування; Довідка.

– Поля з БД: Дисципліна; Викладач; Студент; Група; Модулі; Примітки.

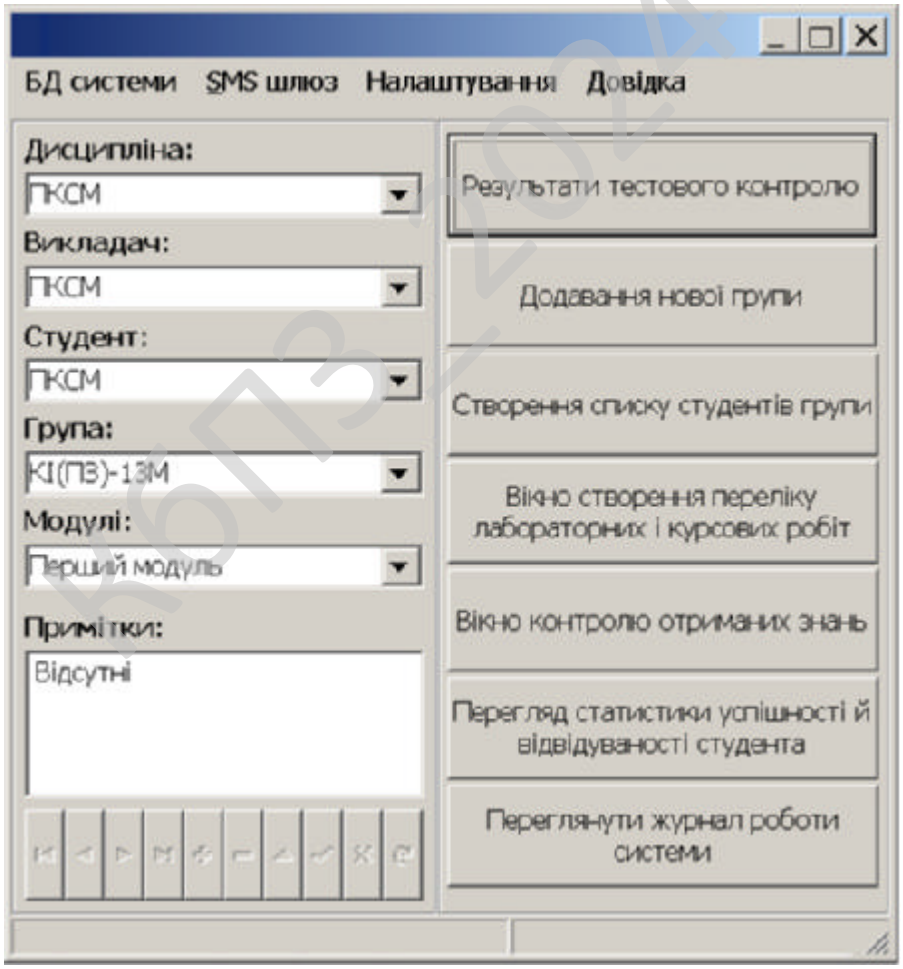

Рисунок 5.1 – Головне вікно програми

– Кнопки: Переглянути результати тестового контролю; Додавання нової групи; Створення списку студентів групи; Вікно створення переліку лабораторних і курсових робіт; Вікно контролю отриманих знань; Перегляд статистики успішності й відвідуваності студента; Переглянути журнал роботи системи.

На рисунку 5.2 зображено авторське право. Авторське право є ключовою галуззю права інтелектуальної власності; воно призначене захищати лише зовнішню форму вираження об'єкта, тобто їхнє матеріальне втілення. Авторське право не може використовуватись для захисту абстрактних ідей, концепцій, фактів, стилів та технік, що можуть бути використані у творі. Захист авторського права – одна з важливих категорій теорії цивільного та цивільно-процесуального права. Було обрано Shareware умову розповсюдження. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

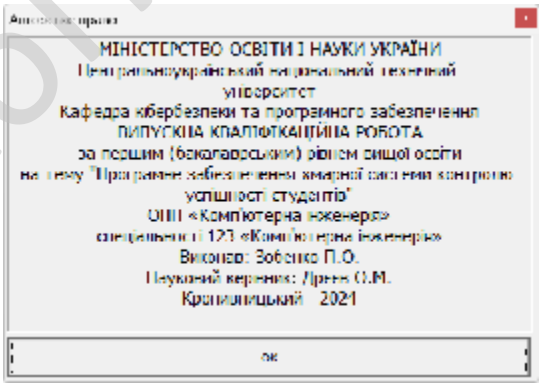

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для хмарної системи контролю успішності студентів.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контролю успішності студентів.

– Досліджена система контролю успішності студентів.

– На основі отриманих результатів досліджень створена програмна реалізація хмарної системи контролю успішності студентів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю успішності студентів.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для хмарної системи контролю успішності студентів. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Blowfish.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

2. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

3. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

4. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

5. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

6. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

7. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

8. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

9. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

10. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

11. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

12. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

13. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

14. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

15. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

16. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

17. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of

Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

18. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

19. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

20. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

21. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

22. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

23. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС,

важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

24. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

25. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

26. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

27. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

28. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

29. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

30. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019:

Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

31. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

32. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

33. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

34. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

35. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

36. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

37. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

38. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

39. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп'ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

40. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). - Полтава: ПолтНТУ. - 2016. - С. 98-103.

41. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). - Харків: ХУПС. - 2016. - С.96-100.

42. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". - Випуск 8 (145). - Х.: ХУПС - 2016. - С. 77-80.

43. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". - Випуск 6 (143). - Х.: ХУПС - 2016. - С. 216-220.

44. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). - Харків: ХУПС. - 2016. - С. 128-133.

45. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). - Харків: ХУПС. - 2016. - С. 150-158.

46. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 40-42.

47. Смірнов О.А., Коваленко А.С., Коваленко О.В., Доренський О.П. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи. Системи озброєння і військова техніка. – Випуск 2(46) – Х.: ХУПС – 2016. – С. 103-107.

48. Smirnov A.A., Kovalenko A.V. Kovalenko A.S. Dorensky A.P. Information model and its element for displaying information on technical condition of objects of integrated information system. International Journal of Computational Engineering Research (IJCER). – Volume 6, Issue 1. – India. Delhi. – 2016. – P. 21-27.

49. Смірнов О.А., Євсеєв С.П., Король О.Г., Коваленко О.В., Коваленко А.С., Смірнов С.А. Архітектура мікропроцесорів та компонентів ЕОМ. Навчальний посібник – Кіровоград: Вид. Лисенко В.Ф., 2015. – 550 с.

50. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення. Навчальний посібник. За ред. О.А. Смірнова. – Кіровоград: КНТУ 2013. – 409с.

51. Смірнов О.А., Осадчій С.І., Мелешко Є.В., Іванов С.Г., Павленко М.А., Усачов О.М. Основи технічної експлуатації АСУ. Навчальний посібник. – Кіровоград: КНТУ 2013. – 322с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку хмарної системи контролю успішності студентів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення хмарної системи контролю успішності студентів.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– хмарної системи контролю успішності студентів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 87 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення хмарної системи контролю успішності студентів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 60

Літера: РП

Кропивницький – 2024 року

Головний модуль розробленого ПЗ

```
unit Main;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дреєв О.М.
Кропивницький 2024
}
interface // інтерфейсна частина

uses
// Підключення бібліотек
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls;

type // створення типів даних
  TForm_Main = class(TForm)
    MainMenu1: TMainMenu;
    // Елементи меню
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N6: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N11: TMenuItem;
    N13: TMenuItem;
    SQL1: TMenuItem; // запити
    Button_Close: TButton;
    Bevel1: TBevel;
    N14: TMenuItem;
    N16: TMenuItem;
    N17: TMenuItem;
    Label1: TLabel;
    N18: TMenuItem;
    N19: TMenuItem;
    N5: TMenuItem;
    N7: TMenuItem;
    N10: TMenuItem;
    N12: TMenuItem;
    N20: TMenuItem;
// декларування процедур обробки  зовнішніх подій
// (натиснення клавіш на клавіатурі або мишці)
    procedure N2Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N9Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure SQL1Click(Sender: TObject);
    procedure Button_CloseClick(Sender: TObject);
    procedure N14Click(Sender: TObject);
    procedure N15Click(Sender: TObject);
    procedure N16Click(Sender: TObject);
    procedure N17Click(Sender: TObject);
    procedure N18Click(Sender: TObject);
    procedure N19Click(Sender: TObject);
    procedure N20Click(Sender: TObject);
  private
```

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

var
  Form_Main: TForm_Main;

implementation
// Підключення бібліотек та модулів
uses Rhouses, Presences, Students, Sales, In_Rhouse, In_ST, In_Presence,
  In_Sale, SQL_Query, QKnowledge, QSale_Cost, QLector, QGain, Form_Client,
  In_client, Unit4, In_Beg, Out, INy, Out_22;

{$R *.DFM}// ресурси

// процедури обробки головного меню
// Закриття програми
procedure TForm_Main.N2Click(Sender: TObject);
begin
     Close;
end;

// виклик форми таблиці
procedure TForm_Main.N4Click(Sender: TObject);
begin
     Form_Rhouses.Show;
end;

// виклик форми таблиці
procedure TForm_Main.N6Click(Sender: TObject);
begin
     Form_Students.Show;
end;

// виклик форми вводу даних
procedure TForm_Main.N7Click(Sender: TObject);
begin
     In_CBegin.Show;
end;

// виклик форми введення даних
procedure TForm_Main.N9Click(Sender: TObject);
begin
     Form_In_Rhouse.Show;
end;

procedure TForm_Main.N11Click(Sender: TObject);
begin
     Form_In_ST.Show;
end;

procedure TForm_Main.N10Click(Sender: TObject);
begin
     Form_Mov.Show;
end;

procedure TForm_Main.N12Click(Sender: TObject);
begin
     Out_2.Show;
end;

// виклик форми побудови SQL запитів
procedure TForm_Main.SQL1Click(Sender: TObject);
begin
     Form_SQL_Query.Show;
end;

procedure TForm_Main.Button_CloseClick(Sender: TObject);
```

```
begin
     close;
end;

// виклик форми категорій
procedure TForm_Main.N14Click(Sender: TObject);
begin
     Form_QKnowledge.Show;
end;

procedure TForm_Main.N15Click(Sender: TObject);
begin
     Form_QSale_Cost.Show;
end;

// виклик форми викладача
procedure TForm_Main.N16Click(Sender: TObject);
begin
     Form_QLector.Show;
end;

procedure TForm_Main.N17Click(Sender: TObject);
begin
     Form_QGain.Show;
end;

procedure TForm_Main.N18Click(Sender: TObject);
begin
 Form1.Show;
end;

procedure TForm_Main.N19Click(Sender: TObject);
begin
    Form_In_Client.Show;
end;

procedure TForm_Main.N20Click(Sender: TObject);
begin
    Form_In.Show;
end;

end.
```

Форма Students

```
unit Students;

interface // інтерфейсна частина
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

uses // Підключення бібліотек
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, DBCtrls, Grids, DBGrids, Db, DBTables;

type // створення типів даних
  TForm_Students = class(TForm)
    Table_Students: TTable; // таблиця
    DataSource_Students: TDataSource; // джерело даних
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Button_Close: TButton;
    Query_Delete: TQuery;
    procedure Button_CloseClick(Sender: TObject);
    procedure Table_StudentsBeforeDelete(DataSet: TDataSet);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
// локальні змінні
  Form_Students: TForm_Students;

implementation

uses Presences, Sales;

{$R *.DFM}// ресурси

// закриття форми
procedure TForm_Students.Button_CloseClick(Sender: TObject);
begin
     Close;
end;

procedure TForm_Students.Table_StudentsBeforeDelete(DataSet: TDataSet);
var // локальні змінні
   id: string;
begin
     if MessageDlg('Поточний запис буде видалено.'
        +#13+Продовжити?',
        mtConfirmation,[mbOK,mbCancel],0)=mrCancel then begin
        Abort;
     end;
     Query_Delete.Close;
     id:=Table_Students.FieldByName('id').AsString;
//видалення даних
     Query_Delete.SQL.Clear;
     Query_Delete.SQL.Add('DELETE FROM presences');
     Query_Delete.SQL.Add('WHERE presences.id_ST='+id);
     Query_Delete.ExecSQL;
```

```
      Query_Delete.SQL.Clear;
      Query_Delete.SQL.Add('DELETE FROM sales');
      Query_Delete.SQL.Add('WHERE sales.id_ST='+id);
      Query_Delete.ExecSQL;

      Form_Presences.Table_Presences.Close;
      Form_Presences.Table_Presences.Open;
      Form_Sales.Table_Sales.Close;
      Form_Sales.Table_Sales.Open;
end;

end.
```

Форма Form_Client

```
unit Form_Client;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, Classes, SysUtils, Graphics, Controls, StdCtrls, Forms,
  Dialogs, DBCtrls, DB, DBGrids, DBTables, Grids, ExtCtrls;

type // створення типів даних
  TForm1 = class(TForm)
    Table1ID: TFloatField;
    Table1FirstName: TStringField;
    Table1LastName: TStringField;
    Table1Adress: TStringField;
    Table1Category: TFloatField;
    Table1Description: TStringField;
    DBGrid1: TDBGrid;
    DBNavigator: TDBNavigator;
    Panel1: TPanel;
    DataSource1: TDataSource;
    Panel2: TPanel;
    Table1: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var // локальні змінні
  Form1: TForm1;

implementation

{$R *.DFM}// ресурси

procedure TForm1.FormCreate(Sender: TObject);
begin
  Table1.Open; //відкриття таблиці
end;

end.
```

Форма In_client

```
unit In_client;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, Classes, SysUtils, Graphics, Controls, StdCtrls, Forms,
  Dialogs, DBCtrls, DB, DBTables, Mask, ExtCtrls;

type // створення типів даних
  TForm_In_Client = class(TForm)
    Table1ID: TFloatField;
    Table1FirstName: TStringField;
    Table1LastName: TStringField;
    Table1Adress: TStringField;
    Table1Category: TFloatField;
    Table1Description: TStringField;
    ScrollBox: TScrollBox;
    Label1: TLabel;
    EditID: TDBEdit;
    Label2: TLabel;
    EditFirstName: TDBEdit;
    Label3: TLabel;
    EditLastName: TDBEdit;
    Label4: TLabel;
    EditAdress: TDBEdit;
    Label5: TLabel;
    EditCategory: TDBEdit;
    Label6: TLabel;
    EditDescription: TDBEdit;
    DBNavigator: TDBNavigator;
    Panel1: TPanel;
    DataSource1: TDataSource;
    Panel2: TPanel;
    Table1: TTable;
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var // локальні змінні
  Form_In_Client: TForm_In_Client;

implementation

{$R *.DFM}// ресурси

procedure TForm_In_Client.FormCreate(Sender: TObject);
begin
  Table1.Open;
end;

end.
```

Форма SQL запитів - SQL_Query

```
unit SQL_Query;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Grids, DBGrids, Db, DBTables;

type // створення типів даних
  TForm_SQL_Query = class(TForm)
    Query1: TQuery;
    DataSource1: TDataSource;
    Button_Execute: TButton;
    Button_Clear: TButton;
    Button_Close: TButton;
    GroupBox1: TGroupBox;
    Memo1: TMemo;
    GroupBox2: TGroupBox;
    DBGrid1: TDBGrid;
    Button_Save: TButton;
    Button_Load: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    procedure Button_ExecuteClick(Sender: TObject);
    procedure Button_ClearClick(Sender: TObject);
    procedure Button_CloseClick(Sender: TObject);
    procedure Button_LoadClick(Sender: TObject);
    procedure Button_SaveClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var // локальні змінні
  Form_SQL_Query: TForm_SQL_Query;

implementation

{$R *.DFM}// ресурси

procedure TForm_SQL_Query.Button_ExecuteClick(Sender: TObject);
begin
     // SQL-запит
     Form_SQL_Query.Query1.SQL:=Form_SQL_Query.Memo1.Lines;
     try
        // закриття запиту
        Form_SQL_Query.Query1.Active:=false;
        //відкриття запиту
        Form_SQL_Query.Query1.Active:=true;
     except
        //в випадку помилки …
        on error: EDatabaseError do begin
           MessageDlg('При виконанні запиту виникла помилка:'+#13+#13+
           error.Message ,mtError,[mbOK],0);
        end;
     end;
```

```
end;

procedure TForm_SQL_Query.Button_ClearClick(Sender: TObject);
begin
     Form_SQL_Query.Memo1.Lines.Clear;
end;

procedure TForm_SQL_Query.Button_CloseClick(Sender: TObject);
begin
     close;
end;

procedure TForm_SQL_Query.Button_LoadClick(Sender: TObject);
begin
     if OpenDialog1.Execute then begin
         Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
     end;
end;

procedure TForm_SQL_Query.Button_SaveClick(Sender: TObject);
begin
     if SaveDialog1.Execute then begin
         Memo1.Lines.SaveToFile(ChangeFileExt(SaveDialog1.FileName,'.sql'));
     end;
end;

end.
```

Форма In_Beg

```
unit In_Beg;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, Classes, SysUtils, Graphics, Controls, StdCtrls, Forms,
  Dialogs, DBCtrls, DB, DBTables, Mask, ExtCtrls;

type // створення типів даних
  TIn_CBegin = class(TForm)
    ScrollBox: TScrollBox;
    Panel2: TPanel;
    Button1: TButton;
    DataSource1: TDataSource;
    Table1: TTable;
    Table2: TTable;
    DataSource2: TDataSource;
    DBLookupComboBox1: TDBLookupComboBox;
    DataSource_ST1: TDataSource;
    Table_ST1: TTable;
    Table_ST2: TTable;
    DataSource_ST2: TDataSource;
    DBLookupComboBox_ST: TDBLookupComboBox;
    Label6: TLabel;
    Label4: TLabel;
    DBLookupComboBox2: TDBLookupComboBox;
    Label1: TLabel;
    DataSource3: TDataSource;
    DataSource4: TDataSource;
    Table3: TTable;
    Table4: TTable;
    Label2: TLabel;
    Label3: TLabel;
    Edit2: TEdit;
    Edit_Date: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var // локальні змінні
  In_CBegin: TIn_CBegin;

implementation

uses Unit4, Sales;

{$R *.DFM}// ресурси

procedure fill_result_data;
begin
     with In_CBegin do begin

     end;
```

```
end;

procedure fill_result_data1;
begin
     with In_CBegin do begin
         Edit2.Text:= '0';
         Edit_Date.Text:= '01.01.14';
       end;
end;

procedure TIn_CBegin.FormCreate(Sender: TObject);
begin
   fill_result_data1
end;

// Добавлення запису
procedure TIn_CBegin.Button1Click(Sender: TObject);
begin
    Form_Mov.Table1.Last;

   try
      Form_Mov.Table1.AppendRecord([
      Form_Mov.Table1.FieldByName('id').AsInteger+1,
      StrToDate(Edit_Date.Text),
      DBLookUpComboBox1.Field.Value,
      DBLookUpComboBox2.Field.Value,
      DBLookUpComboBox_ST.Field.Value,
      StrToInt(Edit2.Text)]);
   except
      MessageDlg('В поля форми введені неприпустимі дані',
         mtError,[mbOK],0);
      Exit;
   end;
end;

end.
```

Форма Out_22

```
unit Out_22;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина
// Зовнішні бібліотеки

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DBCtrls, DB, DBTables;

type // створення типів даних
  Tout_2 = class(TForm)
    Edit_Date: TEdit;
    DBLookupComboBox1: TDBLookupComboBox;
    DBLookupComboBox_ST: TDBLookupComboBox;
    DBLookupComboBox2: TDBLookupComboBox;
    Edit2: TEdit;
    Label2: TLabel;
    Label1: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    Label3: TLabel;
    Table2: TTable;
    DataSource2: TDataSource;
    DataSource1: TDataSource;
    Table1: TTable;
    Table_ST2: TTable;
    DataSource_ST2: TDataSource;
    DataSource_ST1: TDataSource;
    Table_ST1: TTable;
    Table3: TTable;
    DataSource3: TDataSource;
    Table4: TTable;
    DataSource4: TDataSource;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var // локальні змінні
  out_2: Tout_2;

implementation

uses Unit4;

{$R *.DFM}// ресурси

procedure Tout_2.FormCreate(Sender: TObject);
begin
     with Out_2 do begin
         Edit2.Text:= '0';
     end;
end;
```

```
procedure Tout_2.Button1Click(Sender: TObject);
begin
    Form_Mov.Table1.Last;

   try
      Form_Mov.Table1.AppendRecord([
      Form_Mov.Table1.FieldByName('id').AsInteger+1,
      StrToDate(Edit_Date.Text),
      DBLookUpComboBox1.Field.Value,
      DBLookUpComboBox2.Field.Value,
      DBLookUpComboBox_ST.Field.Value,
      StrToInt(Edit2.Text)]);
   except
      MessageDlg('В поля форми введені неприпустимі дані',
         mtError,[mbOK],0);
      Exit;
   end;

end;

end.
```

Файл форми Rhouses

```
unit Rhouses;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, DBCtrls, Grids, DBGrids, Db, DBTables;

type // створення типів даних
  TForm_Rhouses = class(TForm)
    Table_Rhouses: TTable;
    DataSource_Rhouses: TDataSource;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Button_Close: TButton;
    Query_Delete: TQuery;
    procedure Button_CloseClick(Sender: TObject);
    procedure Table_RhousesBeforeDelete(DataSet: TDataSet);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var // локальні змінні
  Form_Rhouses: TForm_Rhouses;

implementation

uses Presences, Sales;
{$R *.DFM}// ресурси
procedure TForm_Rhouses.Button_CloseClick(Sender: TObject);
begin
     Close;
end;

procedure TForm_Rhouses.Table_RhousesBeforeDelete(DataSet: TDataSet);
var // локальні змінні
   id: string;
begin
     //Підтверження видалення запису
     if MessageDlg('Поточний запис і всі пов'язані з нею записи будуть
видалені.'
        +#13+'Продовжити?',
        mtConfirmation,[mbOK,mbCancel],0)=mrCancel then begin
        Abort;
     end;

     //закриття запиту на видалення
     Query_Delete.Close;

     //отримання ключа видаляємого запису
     id:=Table_Rhouses.FieldByName('id').AsString;

     //очищение SQL-запиту
     Query_Delete.SQL.Clear;
     // текст SQL-запиту
```

```
      Query_Delete.SQL.Add('DELETE FROM presences');
      Query_Delete.SQL.Add('WHERE presences.id_rhouse='+id);

      // виконання запиту
      Query_Delete.ExecSQL;
      Query_Delete.SQL.Clear;
      Query_Delete.SQL.Add('DELETE FROM sales');
      Query_Delete.SQL.Add('WHERE sales.id_rhouse='+id);
      Query_Delete.ExecSQL;

      //відновлення даних в формах програми
      Form_Presences.Table_Presences.Close;
      Form_Presences.Table_Presences.Open;
      Form_Sales.Table_Sales.Close;
      Form_Sales.Table_Sales.Open;

end;
end.
```

Файл форми uST

```
unit uST;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, DB, ADODB, StdCtrls, Buttons, Grids, DBGrids, ExtCtrls,
  DBCtrls, Menus, uRichEditWithLinks, Printers, DAO2000, ComObj, ImgList;

type // створення типів даних
  TForm1Tree = class(TForm)
    ADOConnection1: TADOConnection;
    qTreeCompanies: TADOQuery;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Splitter1: TSplitter;
    Panel4: TPanel;
    TreeCompanies: TTreeView;
    Panel5: TPanel;
    DataSource1: TDataSource;
    PopupMenu1: TPopupMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    MainMenu1: TMainMenu;
    Connect1: TMenuItem;
    Exit1: TMenuItem;
    ools1: TMenuItem;
    About1: TMenuItem;
    RichEditWithLinks1: TRichEditWithLinks;
    Panel6: TPanel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    SaveDialog1: TSaveDialog;
    OpenDialog1: TOpenDialog;
    PopupMenu2: TPopupMenu;
    MenuItem1: TMenuItem;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    N4: TMenuItem;
    CompactBase1: TMenuItem;
    Edit1: TEdit;
    ImageList1: TImageList;
    Procedure ExpandLevel( Node : TTreeNode);
    procedure TreeCompaniesExpanding(Sender: TObject; Node: TTreeNode;
      var AllowExpansion: Boolean);
    procedure TreeCompaniesChange(Sender: TObject; Node: TTreeNode);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure MyConnect(Sender: TObject);
    procedure TreeCompaniesClick(Sender: TObject);
    procedure POPupMenuAdd(Sender: TObject);
```

```
    procedure POPupMenuAddIN(Sender: TObject);
    procedure POPupMenuDelete(Sender: TObject);
    procedure TreeCompaniesEdited(Sender: TObject; Node: TTreeNode;
      var S: String);
    procedure About1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure MenuItem1Click(Sender: TObject);
    procedure MenuItem2Click(Sender: TObject);
    procedure MenuItem3Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure CompactBase1Click(Sender: TObject);
    procedure Edit1Locate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var // локальні змінні
  Form1Tree: TForm1Tree;
  RedoDone:Boolean;
  DataBaseName:string;

implementation

uses about;

{$R *.DFM}// ресурси

Procedure DeclarePar;
begin
   with Form1Tree.qTreeCompanies.Parameters.AddParameter do begin
        Name := 'ParentID';
        DataType := ftInteger;
        Direction := pdInput;
        Value := 0;
   end;
   with Form1Tree.qTreeCompanies.Parameters.AddParameter do begin
        Name := 'ID';
        DataType := ftInteger;
        Direction := pdInput;
        Value := 0;
   end;
end;

Procedure TForm1Tree.ExpandLevel( Node : TTreeNode);
Var ID , i   : Integer;
    TreeNode : TTreeNode;
Begin
// Для самого верхнього рівня вибрати лише тіх які на мають батька,
    IF Node = nil Then ID:=0
    Else ID:=Integer(Node.Data);
    qTreeCompanies.Close;
    qTreeCompanies.SQL.Clear;
    declarepar;
    qTreeCompanies.Parameters.ParamByName('ParentID').Value :=ID;
    qTreeCompanies.SQL.Add('Select * From KNTU Where ParentID=:ParentID');
    qTreeCompanies.Open;
    TreeCompanies.Items.BeginUpdate;



// Для кожного рядка з полученого набору даних
```

```
// формуємо гілкі в TreeView, як  дочірні  до тої,
//  яку ми тількишо відкрили
    For i:=1 To qTreeCompanies.RecordCount Do
    Begin
// Запишемо в поле Data гілки її індифікаційний номер(ID) в таблиці
 TreeNode:=TreeCompanies.Items.AddChildObject(Node,
qTreeCompanies.FieldByName('Name').AsString
,Pointer(qTreeCompanies.FieldByName('ID').AsInteger));
 TreeNode.ImageIndex:=1;
 TreeNode.SelectedIndex:=2;

// Добавимо фіктивну (пусту) дочірню гілку  тільки  для того,
// щоб відобразити [+] на гілці і  мати можливвість її розкрити

 TreeCompanies.Items.AddChildObject(TreeNode ,'' , nil);
 qTreeCompanies.Next;
    End;
    TreeCompanies.Items.EndUpdate;
End;

procedure TForm1Tree.TreeCompaniesExpanding(Sender: TObject;
  Node: TTreeNode; var AllowExpansion: Boolean);
begin
  IF Node.getFirstChild.Data = nil Then
  Begin
          Node.DeleteChildren;
          ExpandLevel(Node);
  End;


end;

procedure TForm1Tree.TreeCompaniesChange(Sender: TObject; Node: TTreeNode);
var id:integer;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin

//ID батьківської гілки , Для неї шукаємо всі дочірні
    ID:=Integer(TreeCompanies.Selected.Data);
    qTreeCompanies.Close;
    qTreeCompanies.SQL.Clear;
    DeclarePar;
    showmessage(inttostr(qTreeCompanies.Parameters.Count));
    qTreeCompanies.Parameters.ParamByName('ParentID').Value :=ID;
    qTreeCompanies.SQL.Add('Select * From KNTU Where ParentID=:ParentID');
    qTreeCompanies.Open;
    End;
end;

procedure TForm1Tree.DBGrid1DblClick(Sender: TObject);
var ID:integer;
begin
 ID:=qTreeCompanies.FieldByName('ID').AsInteger;
 If ID <> 0 then
 begin
    Redodone:=False;
    RichEditWithLinks1.Clear;
    RichEditWithLinks1.Lines.Add(qTreeCompanies.FieldByName('TEXT').AsString);
 end;

Allow:=True;
    ID:=qTreeCompanies.FieldByName('ID').AsInteger;
TreeCompanies.OnExpanding(TreeCompanies ,TreeCompanies.Selected, Allow);

// Перебираємоє  всі  отриміні  дочірні гілки  та шукаємо ту, ID якої
// співпадає з ID строки в правій таблиці.
   FOR i:=0 To TreeCompanies.Selected.Count-1 Do
   IF Integer(TreeCompanies.Selected.Item[i].Data) = ID Then
```

```
     Begin
          TreeCompanies.Selected.Item[i].Expand(False);
          TreeCompanies.Selected.Item[i].Selected:=True;
          TreeCompanies.Repaint;
          Exit;
     End;

end;

procedure TForm1Tree.MyConnect(Sender: TObject);
var mainPath, filebase, rconnect:string;
begin
 MainPath:=ExtractFilePath(Application.ExeName);
 filebase:= MainPath+'ST.mdb';
 DatabaseName:=FileBase;
 Form1Tree.Caption:='База: '+ filebase;
 rconnect:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='+
 filebase+  ';Persist Security Info=False';
 aconnect:='Microsoft.Jet.OLEDB.4.0;
 DataSource='+filebase+';Mode=ReadWrite;Persist Security Info=False';
 ADOConnection1.close;
 AdoConnection1.ConnectionString:=Rconnect;
 ADOConnection1.Open;
 TreeCompaniesClick(Sender);
end;

procedure TForm1Tree.TreeCompaniesClick(Sender: TObject);
var id:integer;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin

//ID гілки
   ID:=Integer(TreeCompanies.Selected.Data);
   qTreeCompanies.Close;
   qTreeCompanies.SQL.Clear;
   DeclarePar;
   qTreeCompanies.SQL.Add('Select * From KNTU Where ID=:ID');
   qTreeCompanies.SQL.Add('Select * From KNTU Where ParentID=:ParentID');
   Select * From KNTU Where ParentID=:ParentID
   qTreeCompanies.Parameters.ParamByName( 'ID').Value:=ID;
   qTreeCompanies.Open;
   if id = 0 then
   RichEditWithLinks1.Lines.Clear;
   DBGrid1DblClick(sender);
   End;
end;

procedure TForm1Tree.POPupMenuAdd(Sender: TObject);
var id,RParentID:integer;
Label Fin;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin
   ID:=Integer(TreeCompanies.Selected.Data);
   qTreeCompanies.Close;
   qTreeCompanies.SQL.Clear;
   DeclarePar;
   qTreeCompanies.SQL.Add('Select * From KNTU Where ID=:ID');
   qTreeCompanies.Parameters.ParamByName( 'ID').Value:=ID;
   qTreeCompanies.Open;
   RParentID:= qTreeCompanies.FieldByName('ParentID').AsInteger;
    if RParentID = 0 then
    begin
      showmessage('До нульового не можна!');
      goto fin;
    end;
   qTreeCompanies.Append;
   qTreeCompanies.FieldByName('ParentID').Value:=RParentID;
```

```
   qTreeCompanies.FieldByName('NAME').AsString:='NEW_IN_'+INTTOSTR(RParentID);
   qTreeCompanies.Post;
   qTreeCompanies.Edit;

qTreeCompanies.FieldByName('NAME').AsString:=qTreeCompanies.FieldByName('ID').As
String+'_NEW_IN_'+INTTOSTR(RParentID);
   qTreeCompanies.Post;
 TreeCompanies.Items.AddObject(TreeCompanies.Selected,
//AddChildObject
 qTreeCompanies.FieldByName('Name').AsString
 ,Pointer(qTreeCompanies.FieldByName('ID').AsInteger));
   TreeCompanies.Repaint;
///FullExpand;
   End;
  fin:
end;

procedure TForm1Tree.POPupMenuAddIN(Sender: TObject);
var id,RID:integer;
Label Fin;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin
//ID гілка , для неї  шукаємо всі з ID
   ID:=Integer(TreeCompanies.Selected.Data);
   qTreeCompanies.Close;
   qTreeCompanies.SQL.Clear;
   DeclarePar;
   qTreeCompanies.SQL.Add('Select * From KNTU Where ID=:ID');
   qTreeCompanies.Parameters.ParamByName( 'ID').Value:=ID;
   qTreeCompanies.Open;
   RID:= qTreeCompanies.FieldByName('ID').AsInteger;
   qTreeCompanies.Append;
   qTreeCompanies.FieldByName('ParentID').Value:=RID;
   qTreeCompanies.FieldByName('NAME').AsString:='NEW_INTO '+IntToStr(RID);
   qTreeCompanies.Post;
   qTreeCompanies.Edit;

qTreeCompanies.FieldByName('NAME').AsString:=qTreeCompanies.FieldByName('ID').As
String+'NEW_INTO '+IntToStr(RID);
   qTreeCompanies.Post;
 TreeCompanies.Items.AddChildObject(TreeCompanies.Selected,
//AddChildObject
 qTreeCompanies.FieldByName('Name').AsString
 ,Pointer(qTreeCompanies.FieldByName('ID').AsInteger));
   TreeCompanies.Repaint;
   End;
  fin:
end;

procedure TForm1Tree.POPupMenuDelete(Sender: TObject);
// Видалити
var ii,id:integer;
Label Fin;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin
   if TreeCompanies.Selected.HasChildren then
   begin
     showmessage('Неможливо видалити! Є вложення');
     goto fin;
   end;
   ID:=Integer(TreeCompanies.Selected.Data);
   qTreeCompanies.Close;
   qTreeCompanies.SQL.Clear;
   DeclarePar;
   qTreeCompanies.SQL.Add('Select * From KNTU Where ID=:ID');
   qTreeCompanies.Parameters.ParamByName( 'ID').Value:=ID;
   qTreeCompanies.Open;
```

```
   RParentID:= qTreeCompanies.FieldByName('ParentID').AsInteger;
    if ID = 0 then
    begin
      showmessage('Нулевой нельзя!');
      goto fin;
    end;
   for ii := 0  to qTreeCcompanies.RecordCount-1  do
   begin
        qTreeCompanies.Delete;
   end;
    TreeCompanies.Items.Delete(TreeCompanies.Selected);
   End;
  fin:
end;

procedure TForm1Tree.TreeCompaniesEdited(Sender: TObject; Node: TTreeNode;
  var S: String);
var id:integer;
Label Fin;
begin
   IF TreeCompanies.Selected <> nil Then
   Begin
   ID:=Integer(TreeCompanies.Selected.Data);
   if ID = 0 then
   begin
     showmessage('Головну не правити!');
     goto fin;
   end;
   qTreeCompanies.Close;
   qTreeCompanies.SQL.Clear;
   DeclarePar;
   qTreeCompanies.SQL.Add('Select * From KNTU Where ID=:ID');
   qTreeCompanies.Parameters.ParamByName( 'ID').Value:=ID;
   qTreeCompanies.Open;
   qTreeCompanies.Edit;
   qTreeCompanies.FieldByName('NAME').AsString:=s;
   qTreeCompanies.Post;
   End;
  fin:
end;

procedure TForm1Tree.About1Click(Sender: TObject);
begin
 AboutBox.showmodal;
end;

procedure TForm1Tree.Exit1Click(Sender: TObject);
begin
  ADOConnection1.Close;
  Close;
end;

procedure TForm1Tree.MenuItem1Click(Sender: TObject);
begin
// копіювати в буфер
    RichEditWithLinks1.CopyToClipboard;
end;

procedure TForm1Tree.MenuItem2Click(Sender: TObject);
begin
// вставити з  буфера
   RichEditWithLinks1.PasteFromClipboard;
end;

procedure TForm1Tree.MenuItem3Click(Sender: TObject);
begin
// вирізати
   RichEditWithLinks1.CutToClipboard;
end;
```

```
procedure TForm1Tree.Button1Click(Sender: TObject);
begin
// в БД
   RedoDone:=True;
   if ( qTreeCompanies.RecordCount >= 0 ) and (
qTreeCompanies.FieldByName('ID').value <> 0 ) then
   begin
     qTreeCompanies.edit;

qTreeCompanies.FieldByName('TEXT').asstring:=TrimRight(RichEditWithLinks1.Lines.
Text);
     qTreeCompanies.post;
   end;
   FRedak.Close;
end;

procedure TForm1Tree.Button2Click(Sender: TObject);
begin
 // на диск
  if Savedialog1.Execute then
  begin
   richeditwithlinks1.Lines.SaveToFile(Savedialog1.FileName);
  end;
end;

procedure TForm1Tree.Button3Click(Sender: TObject);
begin
 // з диску
 if Opendialog1.Execute then
  begin
   richeditwithlinks1.Lines.clear;
   richeditwithlinks1.Lines.LoadFromFile(Opendialog1.FileName);
  end;
end;

procedure TForm1Tree.Button4Click(Sender: TObject);
var P:TextFile;
 ii: integer;
begin
// друкувати
   if RichEditWithLinks1.Lines.Count > -1 then
   begin
     AssignPrn(P);
     Rewrite(P);
       for ii :=0  to RichEditWithLinks1.Lines.Count-1 do
       begin
           Writeln(P,RichEditWithLinks1.Lines[ii]);
       end;
     System.CloseFile(P);
   end;
end;

procedure TForm1Tree.BitBtn1Click(Sender: TObject);
begin
 // bold
RichEditWithLinks1.SetFocus;
if fsBold in RichEditWithLinks1.SelAttributes.Style then
RichEditWithLinks1.SelAttributes.Style :=RichEditWithLinks1.SelAttributes.Style
- [fsBold] else
RichEditWithLinks1.SelAttributes.Style :=RichEditWithLinks1.SelAttributes.Style
+ [fsBold];
RichEditWithLinks1.SetFocus;
redodone:=True;
end;

procedure TForm1Tree.BitBtn2Click(Sender: TObject);
begin
// курсив
```

```
        RichEditWithLinks1.SetFocus;
        if fsItalic in RichEditWithLinks1.SelAttributes.Style then
RichEditWithLinks1.SelAttributes.Style :=RichEditWithLinks1.SelAttributes.Style
- [fsItalic] else
RichEditWithLinks1.SelAttributes.Style :=RichEditWithLinks1.SelAttributes.Style
+ [fsItalic];
        RichEditWithLinks1.SetFocus;
end;

procedure TForm1Tree.BitBtn3Click(Sender: TObject);
begin
// підкреслений
      RichEditWithLinks1.SetFocus;
      if fsUnderline in RichEditWithLinks1.SelAttributes.Style then
      RichEditWithLinks1.SelAttributes.Style
:=RichEditWithLinks1.SelAttributes.Style-[fsUnderline] else
      RichEditWithLinks1.SelAttributes.Style
:=RichEditWithLinks1.SelAttributes.Style+[fsUnderline];
      RichEditWithLinks1.SetFocus;
end;

procedure TForm1Tree.CompactBase1Click(Sender: TObject);
//зжимання бази
Var  password:string;
  TempName : Array[0..MAX_PATH] of Char;

// ім'я тимчасового файлу
  TempPath : String;

// шлях
  Name : String;
  tmpDAO : _DBEngine;
  ClassID : TGUID;

  //V35,
  V36 : String;

// версія DAO
Begin
  Password:='';
  TreeCcompanies.Enabled :=False;
  qTreeCompanies.Close;
  ADOConnection1.Close;
  V35:='DAO.DBEngine.35';
  V36:='DAO.DBEngine.36';
  try
// отримаємо ClassID
    try
      ClassID := ProgIDToClassID(v36);
    except
      try
        ClassID := ProgIDToClassID(v36);
      except
        raise
      end;
    end;

// Шлях до тимчасового файлу
    TempPath:=ExtractFilePath(DatabaseName);
    if TempPath='' Then TempPath:=GetCurrentDir;

//отримаємо ім.'я тимчасового файлу
    GetTempFileName(PChar(TempPath),'mdb',0,TempName);
    Name:=StrPas(TempName);
    DeleteFile(PChar(Name));
    if Password <> '' Then Password:=';pwd='+Password;
    tmpDAO := CreateComObject(ClassID) as _DBEngine;
//    tmpDAO.FreeLocks;
    tmpDAO.CompactDatabase(DatabaseName,Name,0,0,Password);
```

```
    DeleteFile(PChar(DatabaseName));
// видаляємо упаковану базу
    RenameFile(Name,DatabaseName);
    ShowMessage(' Упаковка бази завершена ');
  except
        on E: Exception do ShowMessage(e.message);
  end;
  MyConnect(Sender);
  qTreeCompanies.Open;
  TreeCompanies.Enabled :=True;
end;

procedure TForm1Tree.Edit1Locate(Sender: TObject);
begin

  if length(Edit1.Text) > 0 then
  begin
  qTreeCompanies.Filtered:=true;
  qTreeCompanies.Filter:='( Name '+' LIKE ''%'+Edit1.Text+'%'' ) OR ( Text '+'
LIKE ''%'+Edit1.Text+'%'' )';
  end
  else
  begin
  qTreeCompanies.Filtered:=False;
  qTreeCompanies.Filter:='';
  end;
end;

end.
```

Файл модулю Unit1

```
unit Unit1;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  DB, DBTables, StdCtrls, Mask, Buttons, ComCtrls,Printers, ExtCtrls,
  DBCtrls, Grids, DBGrids,ClipBrd, Menus,DBITYPES, DBIPROCS,DBIERRS,OleServer;

type // створення типів даних
  TForm1 = class(TForm)
    MaskEdit1: TMaskEdit;
    DataSource1: TDataSource;
    Table1: TTable;
    SpeedButton1: TSpeedButton;
    OpenDialog1: TOpenDialog;
    ProgressBar1: TProgressBar;
    StatusBar1: TStatusBar;
    SpeedButton3: TSpeedButton;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    ListBox1: TListBox;
    TabSheet2: TTabSheet;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    ComboBox1: TComboBox;
    Edit1: TEdit;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    Mpack: TMenuItem;
    MdelAll: TMenuItem;
    DosWin: TMenuItem;
    N7: TMenuItem;
    BatchMove1: TBatchMove;
    DataSource2: TDataSource;
    Table2: TTable;
    MdellInd: TMenuItem;
    Button1: TButton;
    WINDOS: TMenuItem;
    Mzap: TMenuItem;
    N9: TMenuItem;
    Edit2: TEdit;
    ComboBox2: TComboBox;
    Button2: TButton;
    Label1: TLabel;
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ComboBox1Change(Sender: TObject);
    procedure ListBox1DblClick(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure MpackClick(Sender: TObject);
    procedure MdelAllClick(Sender: TObject);
```

```
    procedure FormCreate(Sender: TObject);
    procedure DosWinClick(Sender: TObject);
    procedure MdellIndClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure WINDOSClick(Sender: TObject);
    procedure MzapClick(Sender: TObject);
    procedure N9Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    MyEsc: Boolean;
  end;

var // локальні змінні
  Form1: TForm1;

implementation

uses About;

{$R *.DFM}// ресурси

{
пошук  ключа  в колонці таблиці Функция потребує на вході аргумент
типу TTable (наприклад, Table1)
}
function Isk( oTable: TTable;var Istr:String; var Pstr:String): Integer;
var RRecno: TSTmark;
    Napr: Integer;
label ZYes,ZNo,ZNo2,Nach;
begin
//SetWindowsHookEx()
Napr:=0;
Result := 0;
Form1.MyEsc:=False;
   RRecno:=oTable.GetSTmark;
   oTable.Next;
  Nach:
   while not oTable.Eof do
   begin
         Result :=AnsiPos(Istr,oTable.FieldByName(Pstr).AsString);
//MyBrowse.SetFocus;
         if Form1.MyEsc then goto ZNo;
         if Result >0 then goto ZYes;
         Application.ProcessMessages();
         oTable.next;
         If (Rrecno = TSTmark(oTable.Recno)) then goto Zno;
   end;
ZNo:
   oTable.GotoSTmark(Rrecno);
//   ShowMessage('Не знайдено ' + Istr);
  if (Napr = 0) and
    (MessageDlg('Не знайдено ' + Istr+'. Продолжить пошук з початку ?',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes)
   then
   begin
     Napr:=1;
     oTable.First;
     goto Nach;
   end else showmessage('Не знайдено' + Istr);
   goto ZNo2;
ZYes:
  oTable.CursorPosChanged;
  MessageBeep(MB_OK);
  Napr:=0;
ZNo2:
  Form1.MyEsc:=False;
```

```
  oTable.FreeSTmark(Rrecno);
end;


procedure MyDisplay();
var i,ii: Integer;
    F: TFieldDef;
    Ln,D: String;
begin
Form1.Table1.Active := True;
//-------------------------------
   Form1.ComboBox2.Items.Clear;
   for ii := 0 to Form1.table1.Fields.Count-1 do
   begin
     Form1.ComboBox2.Items.Add(Form1.Table1.Fields[ii].DisplayLabel);
   end;
   Form1.ComboBox2.ItemIndex:=0;
//-------------------------------
Form1.Table1.GetIndexNames(Form1.ComboBox1.Items);
Form1.ComboBox1.Items.Add('No Index');
Form1.ListBox1.Items.Add('            ФАЙЛ '+ Trim(Form1.MaskEdit1.Text));
   if (Form1.ComboBox1.Items.Count <> -1)and(Form1.ComboBox1.Items[0] <> 'No
Index') then
 begin
  for  i:=0  to Form1.ComboBox1.Items.Count-2 do
  begin
        Form1.ListBox1.Items.Add('     Індекс:'+Form1.ComboBox1.Items[i]);
        Form1.ListBox1.Items.Add('
Інд.вираз:'+Form1.Table1.IndexDefs.Items[i].Expression);
  end;
 end;
with Form1.Table1 do begin
for i := 0 to FieldDefs.Count - 1 do begin
F := FieldDefs.Items[i];
case F.DataType of
ftUnknown: D := 'Unknown';
ftString: D := 'String';
ftSmallint: D := 'SmallInt';
ftInteger: D := 'Integer';
ftWord: D := 'Word';
ftBoolean: D := 'Boolean';
ftFloat: D := 'Float';
ftCurrency: D := 'Currency';
ftBCD: D := 'BCD';
ftDate: D := 'Date';
ftTime: D := 'Time';
ftDateTime: D := 'DateTime';
ftBytes: D := 'Bytes';
ftVarBytes: D := '';
ftBlob: D := 'BLOB';
ftMemo: D := 'Memo';
ftGraphic: D := 'Graphic';
else
D := '';
Ln:=''
end;
 Ln := 'FieldDefs.Items[i].Size='+ IntToStr(FieldDefs.Items[i].Size);
Ln := Ln+' Fields[i].DataSize='+IntToStr(Fields[i].DataSize);
Form1.ListBox1.Items.Add('        '+IntToStr(i)+'. '+F.Name + ', ' + D+', '+Ln);
end;
end;
Form1.Table1.Active := False;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
 Form1.ListBox1.Items.Clear;
 Form1.ComboBox1.Items.Clear;
 Form1.ComboBox1.Items.Add('No Index');
```

```
 If Length(Table1.TableName)<>0 then
 begin
 Table1.Active:=False;
 end;
   If OpenDialog1.Execute then
   begin
     if Length(Opendialog1.Files[0])<>0 then
     begin
      Table1.TableName:=Opendialog1.FileName;
      MaskEdit1.Text:= Table1.TableName;
      MyDisplay();
     end;
   end;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
var P:TextFile;
 ii: integer;
begin
// Виклик
//PrinterSetupDialog1.Execute;
//PrintDialog1.Execute;
 if Length(Table1.TableName) <> 0
 then
  begin
   AssignPrn(P);
   Rewrite(P);
   Writeln(P,'            Файл '+ Trim(Form1.MaskEdit1.Text));
   if (ComboBox1.Items.Count <> -1)and(ComboBox1.Items[0] <> 'No Index') then
   begin
   for  ii:=0  to ComboBox1.Items.Count-2 do
   begin
   Writeln(P,'     Індекс:'+ComboBox1.Items[ii]);
   Writeln(P,'     інд.вираз:'+Table1.IndexDefs.Items[ii].Expression);
  end;
end;
   for ii:=0  to Listbox1.Items.Count-1 do
    Writeln(P,Listbox1.Items[ii]);
   System.CloseFile(P);
  end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
If Length(Table1.TableName)<>0 then
 begin
 Table1.Active:=False;
 end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
If Length(Table1.TableName)<>0 then
 begin
 Table1.Active:=False;
 end;
end;

procedure TForm1.ComboBox1Change(Sender: TObject);
begin
if (ComboBox1.ItemIndex <> -1) and (ComboBox1.Items[ComboBox1.ItemIndex] <> 'No
Index') then
 begin
  Form1.Table1.IndexName:=ComboBox1.Items[ComboBox1.ItemIndex];
  Edit1.Text:=Table1.IndexDefs.Items[ComboBox1.ItemIndex].Expression;
 end
 else
 Form1.Table1.IndexName:='';
end;
```

```
procedure TForm1.ListBox1DblClick(Sender: TObject);
var L: string;
 ii: integer;
begin
L:='';
for ii:=0  to Listbox1.Items.count-1 do
begin
 L:=L+Listbox1.Items[ii];
 if ii <> (Listbox1.Items.count-1) then L:=L+' '#13' ';
end;
clipboard.AsText := L;
//Дані - в буфер!!!
//clipboard.GetTextBuf(PChar,Size);
ShowMessage('Дані в буфері');
end;

procedure TForm1.DBGrid1DblClick(Sender: TObject);
var // локальні змінні
bm : TSTMark;
pch,pch1:  PChar;
s,s2 : string;
i,j : integer;
begin

s := '';
for j := 0 to Form1.Dbgrid1.Columns.Count-1 do
s := s + Form1.Dbgrid1.Columns.Items[j].Title.Caption +#9 ;
s := s + #13+#10;
if not Form1.Dbgrid1.DataSource.DataSet.active then
begin
ShowMessage('Нема активної!');
Exit;
end;
try
Form1.Dbgrid1.Visible := False;

bm := Form1.Dbgrid1.DataSource.DataSet.GetSTmark

Form1.Dbgrid1.DataSource.DataSet.First;
while not Form1.Dbgrid1.DataSource.DataSet.EOF do
begin
s2 := '';
for j := 0 to Form1.Dbgrid1.Columns.Count-1 do
begin
s2 := s2 + Form1.Dbgrid1.Columns.Items[j].Field.AsString +#9;
end;
s := s + s2 + #13+#10;
Form1.Dbgrid1.DataSource.DataSet.Next;
end;

GetMem(pch,100);
GetMem(pch1,100);
GetKeyboardLayoutName(pch);
StrCopy(pch1,pch);
while pch <> '00000419' do
begin
ActivateKeyboardLayout(HKL_NEXT,0);
GetKeyboardLayoutName(pch);
if strComp(pch, pch1) = 0 then

StrCopy(pch,'00000419');
end;

clipboard.AsText := s;
//Данні - в буфер!!!

while strComp(pch, pch1)<>0 do
begin
```

```
ActivateKeyboardLayout(HKL_NEXT,0);
GetKeyboardLayoutName(pch);
end;
FreeMem(pch);
FreeMem(pch1);
Form1.Dbgrid1.DataSource.DataSet.GotoSTmark( bm );
finally
Form1.Dbgrid1.Visible := True;
end;
end;

procedure TForm1.N1Click(Sender: TObject);
var // локальні змінні
S : String;
begin
AboutBox.ShowModal;
s:=ExpandFileName(Application.ExeName);
s:=DateTimeToStr(FileDateToDateTime(FileAge(s)));
s:=s+' '#13#10' #'+CurrentFileInfo(Application.ExeName);
Application.CreateForm(TAboutBox, AboutBox);
AboutBox.ShowModal;
end;

procedure TForm1.N7Click(Sender: TObject);
begin
// Вихід
 Table1.Active:=False;
 Form1.Close;
end;

procedure TForm1.MpackClick(Sender: TObject);
var rdir,newname:string;
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin
  ShowMessage(' Спочатку виберіть таблицю..');
  exit;
 end;
// Пакувати packtable
   getdir(0,rdir);
if FileExists('Temp.dbf') then
if MessageDlg('Do Видалити ' + ExtractFileName(FileName) + '?'), []) = IDYes
then
    DeleteFile('Temp.dbf');
    Table1.active:=False;
      with BatchMove1 do
       begin
          Source := Table1;
          Table2.Tablename:='Temp.dbf';
          Table2.DatabaseName:=Table1.DatabaseName;
          Destination := Table2;
          Mode := batCopy;
          Execute;
       end;
       Table1.active:=False;
       Table2.active:=False;
       Session.Close; // CloseDataBase;
       newname:=ChangeFileExt(Table1.Tablename, '.BAK');
       DeleteFile(newname);
if not RenameFile(UpperCase(OpenDialog1.FileName),UpperCase(newname)) then
       showmessage(' Не вдалося перейменувати '+OpenDialog1.FileName+' у
'+newname);
if not RenameFile(UpperCase(Table2.Tablename),UpperCase(OpenDialog1.FileName))
then
       showmessage(' Не вдалося перейменувати '+Table2.Tablename+' у
'+OpenDialog1.FileName);
       //       RenameFile
       Table1.Tablename:= OpenDialog1.FileName;
       Table1.active:=True;
```

```
end;

procedure TForm1.MdelAllClick(Sender: TObject);
var rdir,newname:string;
var usego: integer;
begin
// Очистити
 if Form1.ListBox1.Items.Count = 0 then
 begin
  ShowMessage('Спочатку виберіть таблицю...');
  exit;
 end;
if FileExists('Temp.dbf') then
if MessageDlg('Do you really want to delete ' + ExtractFileName(FileName) +
'?'), []) = IDYes then
    DeleteFile('Temp.dbf');
   usego:= Table1.RecordCount;
    Table1.active:=False;
      with BatchMove1 do
       begin
          Source := Table1;
          Table2.Tablename:='Temp.dbf';
          Table2.DatabaseName:=Table1.DatabaseName;
          if usego > 0 then RecordCount:=1;
          Destination := Table2;
          Mode := batCopy;
          Execute;
       end;
       if usego > 0 then
       begin
          Table1.active:=False;
          Table2.active:=True;
          Table2.Delete;
          Table2.active:=False;
       end;
      with BatchMove1 do
       begin
          Source := Table2;
          Destination := Table1;
          Mode := batCopy;
          Execute;
       end;
       Session.Close; // CloseDataBase;
       Table1.active:=True;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
 Form1.ListBox1.Items.Clear;
 Form1.ComboBox1.Items.Clear;
end;

procedure TForm1.DosWinClick(Sender: TObject);
// dos- win
var stroka,dstroka,tek:string;
var   F: TFieldDef;
var ii:integer;
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin
  exit;
 end;
                Table1.First;
                 while not Table1.Eof do
                begin
                   Table1.Edit;
                   for ii := 0 to Table1.FieldCount - 1 do
                   begin
```

```
                        F := Table1.FieldDefs.Items[ii];
                        If F.DataType = ftString Then
                        begin
                          Tek:=Table1.Fields[ii].FieldName;
                           Stroka:= Trim(Table1.FieldByName(Tek).AsString);
                           if length(stroka) <> 0 then
                           begin
                              Dstroka:=Stroka;
                              OemToChar(PChar(Stroka),PChar(DStroka));
                              Table1.Fields[ii].Value:=DStroka;
                           end;
                        end;
                    end;
                    Table1.Post;
                    Table1.Next;
                end;
                Table1.First;
end;

procedure TForm1.MdellIndClick(Sender: TObject);
// Видалити індексний файл
var // локальні змінні
 FileHandler : Integer;
 Buf : Byte;
 newname: string;
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin
  exit;
 end;
 Table1.active:=false;
 Table1.IndexName:='';
 Session.Close;
 Buf := 3;
 FileHandler:=FileOpen(Table1.TableName,fmOpenWrite);
 try
  FileSeek(FileHandler,0,0);
  FileWrite(FileHandler,Buf,1);

// обнуляємо признак існування CDX-файла
  Buf :=0;
  FileSeek(FileHandler,28,0);
  FileWrite(FileHandler,Buf,1);

// обнуляємо признак існування CDX-файла
 finally
  FileClose(FileHandler);
 end;
  newname:=ChangeFileExt(Table1.Tablename, '.CDX');
  DeleteFile(newname);
  Form1.ComboBox1.Items.Clear;
  Form1.ComboBox1.Items.Add('No Index');
  Table1.active:=True;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
видалити групу
 if table1.active then
 begin
     if MessageDlg('Видалити виделені записи',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
    dbgrid1.SelectedRows.Delete;
 end;
end;
procedure TForm1.WINDOSClick(Sender: TObject);
// WinToDos
var stroka,dstroka,tek:string;
var   F: TFieldDef;
```

```
var ii:integer;
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin
  ShowMessage('спочатку виберіть таблицю...');
  exit;
 end;
                Table1.First;
                 while not Table1.Eof do
                begin
                   Table1.Edit;
                   for ii := 0 to Table1.FieldCount - 1 do
                   begin
                       F := Table1.FieldDefs.Items[ii];
                       If F.DataType = ftString Then
                       begin
                         Tek:=Table1.Fields[ii].FieldName;
                          Stroka:= Trim(Table1.FieldByName(Tek).AsString);
                          if length(stroka) <> 0 then
                          begin
                             Dstroka:=Stroka;
                             CharToOem(PChar(Stroka),PChar(DStroka));
                              //OemToChar DOS->WIN CharToOem WinToDos
                             Table1.Fields[ii].Value:=DStroka;
                          end;
                       end;
                   end;
                   Table1.Post;
                   Table1.Next;
                 end;
                 Table1.First;

end;
procedure TForm1.MzapClick(Sender: TObject);
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin
  exit;
 end;
     MdelAllClick(Sender);
     MpackClick(Sender);
end;

procedure TForm1.N9Click(Sender: TObject);
begin
 if Form1.ListBox1.Items.Count = 0 then
 begin

  exit;
 end;
     MdelAllClick(Sender);
     MpackClick(Sender);
     MdellIndClick(Sender);
end;

procedure TForm1.Button2Click(Sender: TObject);
var rtext,rpole:string;
begin
 // пошук по полю
 Rtext:=Edit2.Text;
 Rpole:=Table1.Fields[ComboBox2.ItemIndex].FieldName;
 Dbgrid1.visible:=False;
 Isk(Table1,Rtext, Rpole);
 Dbgrid1.visible:=True;
end;
end.
```

Файл бібліотеки ABS

```
unit ABS;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

{$I s_.INC}

interface // інтерфейсна частина

uses
  SysUtils, Messages, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Grids, SpeedBar,
  Menus, Placemnt, s_Const, s_Ctrls, VCLUtils,
DesignIntf, DesignWindows, DesignEditors

type // створення типів даних

{ TSpeedbarEditor }

  TSelectData = record
    bRowCount: Integer;
    bRow: Integer;
    sRowCount: Integer;
    sRow: Integer;
  end;

  TSpeedbarEditor = class(TDesignWindow)
    SectionsBox: TGroupBox;
    NewSection: TButton;
    DelSection: TButton;
    ButtonsBox: TGroupBox;
    UpBtn: TSpeedButton;
    DownBtn: TSpeedButton;
    AddButton: TButton;
    RemoveButton: TButton;
    CloseBtn: TButton;
    SectionName: TEdit;
    SectionNameLabel: TLabel;
    SectionList: TDrawGrid;
    ButtonsList: TDrawGrid;
    LabelHint: TLabel;
    PopupMenu: TPopupMenu;
    CopyMenu: TMenuItem;
    PasteMenu: TMenuItem;
    CutMenu: TMenuItem;
    FormPlacement1: TFormPlacement;
    procedure DelSectionClick(Sender: TObject);
    procedure AddButtonClick(Sender: TObject);
    procedure RemoveButtonClick(Sender: TObject);
    procedure CloseBtnClick(Sender: TObject);
    procedure UpBtnClick(Sender: TObject);
    procedure DownBtnClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure SectionNameExit(Sender: TObject);
    procedure SectionListSelectCell(Sender: TObject; Col, Row: Longint;
      var CanSelect: Boolean);
    procedure SectionListDrawCell(Sender: TObject; Col, Row: Longint;
      Rect: TRect; State: TGridDrawState);
    procedure ButtonsListDblClick(Sender: TObject);
    procedure ButtonsListKeyDown(Sender: TObject; var Key: Word;
```

```
      Shift: TShiftState);
    procedure ButtonsListMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ButtonsListMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure ButtonsListMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ButtonsListSelectCell(Sender: TObject; Col, Row: Longint;
      var CanSelect: Boolean);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure NewSectionClick(Sender: TObject);
    procedure SectionNameKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ButtonsListDrawCell(Sender: TObject; Col, Row: Longint;
      Rect: TRect; State: TGridDrawState);
    procedure SectionListMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure SectionListDragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure SectionListDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure SectionListKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure CopyMenuClick(Sender: TObject);
    procedure PasteMenuClick(Sender: TObject);
    procedure CutMenuClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    FButton: TBtnControl;
    FImage: TButtonImage;
    FBar: TSpeedBar;
    FDrag: Boolean;
    FDragItem: TSpeedItem;
    FLocked: Integer;
    FSelectData: TSelectData;
    procedure Copy;
    procedure Cut;
    procedure Paste;
    procedure OnPasteItem(Item: TObject);
    procedure SaveSelection;
    procedure RestoreSelection;
    procedure SelectButton(Section: Integer; Item: TSpeedItem; SelectBar:
Boolean);
    procedure UpdateEnabled(BtnRow, Section: Integer);
    function CheckSpeedBar: Boolean;
    function ConfirmDelete: Boolean;
    function CurrentSection: Integer;
    function GetForm: TCustomForm;
    procedure SetSection(Section: Integer);
    procedure UpdateData;
    procedure UpdateListHeight;
    procedure SetSpeedBar(Value: TSpeedBar);
    function ItemByRow(Row: Integer): TSpeedItem;
    function SectionByRow(Row: Integer): TSpeedbarSection;
    function ItemBySectionRow(Section, Row: Integer): TSpeedItem;
    procedure CMSpeedBarChanged(var Message: TMessage); message
CM_SPEEDBARCHANGED;
  protected
    procedure Activated; override;
    function UniqueName(Component: TComponent): string; override;
  public
    procedure FormClosed(Form: TForm); override;
    function EditAction(Action: TEditAction): boolean; override;
    property SpeedBar: TSpeedBar read FBar write SetSpeedBar;
    property OwnerForm: TCustomForm read GetForm;
  end;

{ TSpeedbarCompEditor }
```

```
  TSpeedbarCompEditor = class(TComponentEditor)
    procedure ExecuteVerb(Index: Integer); override;
    function GetVerb(Index: Integer): string; override;
    function GetVerbCount: Integer; override;
  end;

implementation

uses TypInfo, MaxMin, s_LConst, s_Props, s_Dsgn;

{$R *.DFM}// ресурси

type // створення типів даних
  TDesigner = IDesigner;
  TFormDesigner = IFormDesigner;


function FindEditor(Speedbar: TSpeedbar): TSpeedbarEditor;
var // локальні змінні
  I: Integer;
begin
  Result := nil;
  for I := 0 to Screen.FormCount - 1 do begin
    if Screen.Forms[I] is TSpeedbarEditor then begin
      if TSpeedbarEditor(Screen.Forms[I]).SpeedBar = SpeedBar then
      begin
        Result := TSpeedbarEditor(Screen.Forms[I]);
        Break;
      end;
    end;
  end;
end;

procedure ShowSpeedbarDesigner(Designer: TDesigner; Speedbar: TSpeedbar);
var // локальні змінні
  Editor: TSpeedbarEditor;
begin
  if Speedbar = nil then Exit;
  Editor := FindEditor(Speedbar);
  if Editor <> nil then begin
    Editor.Show;
    if Editor.WindowState = wsMinimized then Editor.WindowState := wsNormal;
  end
  else begin
    Editor := TSpeedbarEditor.Create(Application);
    try
      Editor.Designer := TFormDesigner(Designer);
      Editor.Speedbar := Speedbar;
      Editor.Show;
    except
      Editor.Free;
      raise;
    end;
  end;
end;

{ Опис TSpeedbarCompEditor }

procedure TSpeedbarCompEditor.ExecuteVerb(Index: Integer);
begin
  case Index of
    0: ShowSpeedbarDesigner(Designer, TSpeedbar(Component));
  end;
end;

function TSpeedbarCompEditor.GetVerb(Index: Integer): string;
begin
  case Index of
```

```
    0: Result := LoadStr(srSpeedbarDesigner);
  end;
end;

function TSpeedbarCompEditor.GetVerbCount: Integer;
begin
  Result := 1;
end;

{ Опис TSpeedbarEditor }

const
  MaxBtnListHeight = 158;

function TSpeedbarEditor.UniqueName(Component: TComponent): string;
var // локальні змінні
  Temp: string;

  Comp: TComponent;
  I: Integer;
begin
  Result := '';
  if (Component <> nil) then Temp := Component.ClassName
  else Temp := TSpeedItem.ClassName;
  if (UpCase(Temp[1]) = 'T') and (Length(Temp) > 1) then
    System.Delete(Temp, 1, 1);
  Result := Designer.UniqueName(Temp);
  I := 1;
  repeat
    Result := Temp + IntToStr(I);
    Comp := OwnerForm.FindComponent(Result);
    Inc(I);
  until (Comp = nil) or (Comp = Component);

end;

function TSpeedbarEditor.GetEditState: TEditState;
begin
  Result := [];
  if RemoveButton.Enabled then begin
    Result := [esCanDelete, esCanCut, esCanCopy];
  end;
  if AddButton.Enabled and ClipboardComponents then
    Include(Result, esCanPaste);
end;

procedure TSpeedbarEditor.EditAction(Action: TEditAction);
begin
  case Action of
    eaCut: Cut;
    eaCopy: Copy;
    eaPaste: Paste;
    eaDelete: RemoveButtonClick(Self);
  end;
  Result := True;

end;

procedure TSpeedbarEditor.SelectButton(Section: Integer; Item: TSpeedItem;
  SelectBar: Boolean);
var // локальні змінні
  FCompList: IDesignerSelections;

  FCompList: TDesignerSelectionList;
  Sect: TSpeedbarSection;
begin
  if CheckSpeedBar and Active then begin
    FCompList := CreateSelectionlist;
    FCompList := TDesignerSelectionList.Create;
```

```
    if not SelectBar then begin
      if (ActiveControl = SectionList) or (ActiveControl = SectionName) then
      begin
        Sect := SectionByRow(Section);
        if Sect <> nil then FCompList.Add(Sect);
      end;
      if (FCompList.Count = 0) and (Item <> nil) then FCompList.Add(Item);
    end;
    if (FBar <> nil) and (FCompList.Count = 0) then FCompList.Add(FBar);
    SetSelection(FCompList);
  end;
end;

procedure TSpeedbarEditor.FormClosed(Form: TCustomForm);

procedure TSpeedbarEditor.FormClosed(Form: TForm);

begin
  if Form = OwnerForm then Free;
end;

procedure TSpeedbarEditor.FormModified;
begin
  if not (csDestroying in ComponentState) then UpdateData;
end;


procedure TSpeedbarEditor.Activated;
begin
  SelectButton(CurrentSection, ItemByRow(ButtonsList.Row), False);
  PasteMenu.Enabled := CheckSpeedBar and (FBar.SectionCount > 0) and
    ClipboardComponents;
end;

function TSpeedbarEditor.ConfirmDelete: Boolean;
begin
  Result := MessageDlg(LoadStr(srConfirmSBDelete), mtWarning, mbYesNoCancel, 0)
= mrYes;
end;

procedure TSpeedbarEditor.SaveSelection;
begin
  with FSelectData do begin
    bRowCount := ButtonsList.RowCount;
    bRow := ButtonsList.Row;
    sRowCount := SectionList.RowCount;
    sRow := SectionList.Row;
  end;
end;

procedure TSpeedbarEditor.RestoreSelection;
var // локальні змінні
  NewSRow, NewBRow: Integer;
begin
  NewSRow := FSelectData.sRow;
  if (SectionList.RowCount > FSelectData.sRowCount) or
    (NewSRow > SectionList.RowCount - 1) then
    NewSRow := SectionList.RowCount - 1;
  if NewSRow < 0 then NewSRow := 0;
  SectionList.Row := NewSRow;
  SetSection(SectionList.Row); { set ButtonsList to current section }
  NewBRow := FSelectData.bRow;
  if (ButtonsList.RowCount > FSelectData.bRowCount) or
    (NewBRow > ButtonsList.RowCount - 1) then
    NewBRow := ButtonsList.RowCount - 1;
  if NewBRow < 0 then NewBRow := 0;
  ButtonsList.Row := NewBRow;
end;
```

```
procedure TSpeedbarEditor.UpdateEnabled(BtnRow, Section: Integer);
var // локальні змінні
  EnableSect, EnableBtn: Boolean;
begin
  EnableSect := CheckSpeedBar and (FBar.SectionCount > 0);
  EnableBtn := EnableSect and (BtnRow >= 0) and (ItemBySectionRow(Section,
    BtnRow) <> nil);
  DelSection.Enabled := EnableSect;
  SectionName.Enabled := EnableSect;
  AddButton.Enabled := EnableSect;
  RemoveButton.Enabled := EnableBtn;
  CopyMenu.Enabled := EnableBtn;
  CutMenu.Enabled := EnableBtn;
  PasteMenu.Enabled := EnableSect and ClipboardComponents;
  UpBtn.Enabled := EnableBtn and (BtnRow > 0);
  DownBtn.Enabled := EnableBtn and (BtnRow < ButtonsList.RowCount - 1);
end;

function TSpeedbarEditor.CheckSpeedBar: Boolean;
begin
  Result := (FBar <> nil) and (FBar.Owner <> nil) and (FBar.Parent <> nil)
    and (Designer.{$IFDEF s__D6}Root Form <> nil);
end;

function TSpeedbarEditor.CurrentSection: Integer;
begin
  if CheckSpeedBar and (FBar.SectionCount > 0) then
    Result := SectionList.Row
  else Result := -1;
end;

procedure TSpeedbarEditor.SetSection(Section: Integer);
var // локальні змінні
  I: Integer;
begin
  if CheckSpeedBar then begin
    I := Section;
    if (I >= 0) and (I < FBar.SectionCount) then begin
      SectionName.Text := TSpeedbarSection(FBar.Sections[I]).Caption;
      ButtonsList.RowCount := FBar.ItemsCount(I);
    end
    else begin
      SectionName.Text := '';
      ButtonsList.RowCount := 0;
    end;
    SectionList.DefaultColWidth := SectionList.ClientWidth;
    ButtonsList.DefaultColWidth := ButtonsList.ClientWidth;
  end;
end;

procedure TSpeedbarEditor.UpdateData;
begin
  Inc(FLocked);
  try
    SaveSelection;
    if CheckSpeedBar then SectionList.RowCount := FBar.SectionCount
    else SectionList.RowCount := 0;
    RestoreSelection; { set section }
  finally
    Dec(FLocked);
  end;
  UpdateEnabled(ButtonsList.Row, SectionList.Row);
  SelectButton(CurrentSection, ItemByRow(ButtonsList.Row), False);
end;

function TSpeedbarEditor.GetForm: TCustomForm;
begin
  Result := TCustomForm(Designer.Root);
```

```
  Result := Designer.Form; { GetParentForm(FBar) }
end;

procedure TSpeedbarEditor.UpdateListHeight;
var // локальні змінні
  Cnt: Integer;
  MaxHeight: Integer;
begin
  Canvas.Font := Font;
  MaxHeight := MulDiv(MaxBtnListHeight, Screen.PixelsPerInch, 96);
  ButtonsList.DefaultRowHeight := FBar.BtnHeight + 2;
  Cnt := Max(1, Max(ButtonsList.ClientHeight, MaxHeight) div
    (FBar.BtnHeight + 2));
  ButtonsList.ClientHeight := Min(ButtonsList.DefaultRowHeight * Cnt,
    MaxHeight);
  SectionList.DefaultRowHeight := Canvas.TextHeight('Wg') + 2;
end;

procedure TSpeedbarEditor.SetSpeedBar(Value: TSpeedBar);
var // локальні змінні
  I: Integer;
begin
  if FBar <> Value then begin
    if FBar <> nil then FBar.SetEditing(0);
    FBar := Value;
    if FBar <> nil then FBar.SetEditing(Handle);
    Inc(FLocked);
    try
      if FBar <> nil then UpdateListHeight;
      if FBar.SectionCount = 0 then NewSectionClick(Self)
      else
        for I := 0 to FBar.SectionCount - 1 do begin
          if FBar.Sections[I].Name = '' then begin
            FBar.Sections[I].Name := UniqueName(FBar.Sections[I]);
            Designer.Modified;
          end;
        end;
      if ButtonsList.RowCount > 0 then ActiveControl := ButtonsList
      else ActiveControl := SectionList;
      UpdateData;
      ButtonsList.Row := 0;
    finally
      Dec(FLocked);
    end;
    SectionList.Row := 0;
  end;
end;

procedure TSpeedbarEditor.CMSpeedBarChanged(var Message: TMessage);
begin
  if Pointer(Message.LParam) = FBar then begin
    case Message.WParam of
      SBR_CHANGED: Designer.Modified;
      SBR_DESTROYED: Close;
      SBR_BTNSIZECHANGED: if FBar <> nil then UpdateListHeight;
    end;
  end
  else if (Message.WParam = SBR_BTNSELECT) and CheckSpeedBar then begin
    SelectButton(-1, nil, True);
    Designer.Modified;
  end;
end;

function TSpeedbarEditor.ItemBySectionRow(Section, Row: Integer): TSpeedItem;
begin
  if CheckSpeedBar then Result := FBar.Items(Section, Row)
  else Result := nil;
end;
function TSpeedbarEditor.SectionByRow(Row: Integer): TSpeedbarSection;
```

```
begin
  if CheckSpeedBar and (Row >= 0) and (Row < FBar.SectionCount) then
    Result := FBar.Sections[Row]
  else Result := nil;
end;

function TSpeedbarEditor.ItemByRow(Row: Integer): TSpeedItem;
begin
  Result := ItemBySectionRow(CurrentSection, Row);
end;

procedure TSpeedbarEditor.NewSectionClick(Sender: TObject);
var // локальні змінні
  S: string;
  I: Integer;
begin
  if CheckSpeedBar then begin
    I := 0;
    repeat
      S := Format(LoadStr(srNewSectionName), [I]);
      Inc(I);
    until FBar.SearchSection(S) < 0;
    I := NewSpeedSection(FBar, S);
    if I >= 0 then FBar.Sections[I].Name := UniqueName(FBar.Sections[I]);
    ActiveControl := SectionName;
    Designer.Modified;
  end;
end;

procedure TSpeedbarEditor.DelSectionClick(Sender: TObject);
var // локальні змінні
  Sect: Integer;
  Item: TSpeedItem;
begin
  if CheckSpeedBar and ConfirmDelete then begin
    Sect := SectionList.Row;
    if (Sect >= 0) and (Sect < FBar.SectionCount) then begin
      Self.ValidateRename(FBar.Sections[Sect],
        FBar.Sections[Sect].Name, '');
      Designer.ValidateRename(FBar.Sections[Sect],
        FBar.Sections[Sect].Name, '');
      try
        while FBar.ItemsCount(Sect) > 0 do begin
          Item := FBar.Items(Sect, 0);
          if Item <> nil then begin
            OwnerForm.RemoveComponent(Item);
            Item.Free;
          end;
        end;
        FBar.RemoveSection(Sect);
      finally
        Designer.Modified;
      end;
    end;
  end;
end;

procedure TSpeedbarEditor.Copy;
var // локальні змінні
  CompList: IDesignerSelections;
  CompList: TDesignerSelectionList;
  Item: TSpeedItem;
begin
  CompList := CreateSelectionlist;

  CompList := TDesignerSelectionList.Create;

  try
    Item := ItemByRow(ButtonsList.Row);
```

```
    if Item <> nil then begin
      Item.InvalidateItem;
      CompList.Add(Item);
      CopyComponents(OwnerForm, CompList);
      Item.UpdateSection;
    end;
  finally
    CompList.Free;
  end;
end;

procedure TSpeedbarEditor.Paste;
var // локальні змінні
  CompList: IDesignerSelections;

  CompList: TDesignerSelectionList;

begin
  if CheckSpeedBar then begin
    CompList := CreateSelectionlist;

    CompList := TDesignerSelectionList.Create;

    try
      FBar.OnAddItem := OnPasteItem;
      try
        PasteComponents(OwnerForm, FBar, CompList);
      finally
        FBar.OnAddItem := nil;
      end;
      UpdateData;
    finally
      CompList.Free;
    end;
  end;
end;

procedure TSpeedbarEditor.Cut;
begin
  Copy;
  RemoveButtonClick(Self);
end;

procedure TSpeedbarEditor.OnPasteItem(Item: TObject);
begin
  if (Item <> nil) then begin
    if CheckSpeedBar and (Item is TSpeedItem) then begin
      TSpeedItem(Item).ASection := CurrentSection;
      TSpeedItem(Item).Visible := False;
    end
  end;
end;

procedure TSpeedbarEditor.AddButtonClick(Sender: TObject);
var // локальні змінні
  I: Integer;
  Item: TSpeedItem;
begin
  I := CurrentSection;
  if I < 0 then Exit;
  Item := TSpeedItem.Create(OwnerForm);
  if Item <> nil then
    try
      FBar.AddItem(I, Item);
      Item.Name := UniqueName(Item);
      Designer.Modified;
      if (Sender <> nil) then ActivateInspector(#0);
    except
      Item.Free;
```

```
      raise;
    end
  else raise ESpeedbarError.CreateRes(srSBItemNotCreate);
end;

procedure TSpeedbarEditor.RemoveButtonClick(Sender: TObject);
var // локальні змінні
  Item: TSpeedItem;
begin
  Item := ItemByRow(ButtonsList.Row);
  if Item <> nil then begin
    Self.ValidateRename(Item, Item.Name, '');
    Designer.ValidateRename(Item, Item.Name, '');
    OwnerForm.RemoveComponent(Item);
    Item.Free;
    Designer.Modified;
  end;
end;

procedure TSpeedbarEditor.CloseBtnClick(Sender: TObject);
begin
  Close;
end;

procedure TSpeedbarEditor.UpBtnClick(Sender: TObject);
var // локальні змінні
  I, Sect: Integer;
begin
  if CheckSpeedBar and FBar.FindItem(ItemByRow(ButtonsList.Row), Sect, I) then
  begin
    if I > 0 then begin
      FBar.Sections[Sect].List.Move(I, I - 1);
      Designer.Modified;
      ButtonsList.Invalidate;
      ButtonsList.Row := ButtonsList.Row - 1;
    end;
  end;
end;

procedure TSpeedbarEditor.DownBtnClick(Sender: TObject);
var // локальні змінні
  I, Sect: Integer;
begin
  if CheckSpeedBar and FBar.FindItem(ItemByRow(ButtonsList.Row), Sect, I) then
  begin
    if I < FBar.ItemsCount(Sect) - 1 then begin
      FBar.Sections[Sect].List.Move(I, I + 1);
      Designer.Modified;
      ButtonsList.Invalidate;
      ButtonsList.Row := ButtonsList.Row + 1;
    end;
  end;
end;

procedure TSpeedbarEditor.CopyMenuClick(Sender: TObject);
begin
  Copy;
end;

procedure TSpeedbarEditor.PasteMenuClick(Sender: TObject);
begin
  Paste;
end;

procedure TSpeedbarEditor.CutMenuClick(Sender: TObject);
begin
  Cut;
end;
procedure TSpeedbarEditor.SectionNameExit(Sender: TObject);
```

```
var // локальні змінні
  I: Integer;
begin
  if CheckSpeedBar and (FBar.SectionCount > 0) then begin
    I := CurrentSection;
    if I >= 0 then begin
      FBar.Sections[I].Caption := SectionName.Text;
      Designer.Modified;
    end;
  end;
end;

procedure TSpeedbarEditor.SectionListSelectCell(Sender: TObject; Col,
  Row: Longint; var CanSelect: Boolean);
begin
  CanSelect := False;
  if CheckSpeedBar and (Row < FBar.SectionCount) and (Row >= 0) then begin
    if FLocked = 0 then begin
      SetSection(Row);
      UpdateEnabled(ButtonsList.Row, Row);
      SelectButton(Row, ItemBySectionRow(Row, ButtonsList.Row), False);
    end;
    CanSelect := True;
  end;
end;

procedure TSpeedbarEditor.SectionListDrawCell(Sender: TObject; Col,
  Row: Longint; Rect: TRect; State: TGridDrawState);
begin
  if CheckSpeedBar then begin
    if (Row < FBar.SectionCount) and (Row >= 0) then begin
      DrawCellText(Sender as TDrawGrid, Col, Row,
        FBar.Sections[Row].Caption, Rect, taLeftJustify, vaCenter
    end;
  end;
end;

procedure TSpeedbarEditor.SectionListKeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
begin
  case Key of
    VK_RETURN: if SectionByRow(SectionList.Row) <> nil then
ActivateInspector(#0);
    VK_DELETE: DelSectionClick(Self);
    VK_INSERT, VK_ADD: NewSectionClick(Self);
    else Exit;
  end;
  Key := 0;
end;

procedure TSpeedbarEditor.ButtonsListKeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
begin
  case Key of
    VK_RETURN: if ItemByRow(ButtonsList.Row) <> nil then ActivateInspector(#0);
    VK_DELETE: RemoveButtonClick(Self);
    VK_INSERT, VK_ADD: AddButtonClick(Self);
    else Exit;
  end;
  Key := 0;
end;

procedure TSpeedbarEditor.ButtonsListDblClick(Sender: TObject);
type // створення типів даних
  PParamData = ^TParamData;
  TParamData = record
    Flags: TParamFlags;
    ParamNameAndType: array[0..100] of Char;
  end;
```

```
const

  sSender: string[7] = '*Sender';
  sSender: string[6] = 'Sender';

  sObject: string[7] = 'TObject';
var // локальні змінні
  Btn: TSpeedItem;
  I, Num: Integer;
  MethodName: string;
  Method: TMethod;
  TypeData: PTypeData;
  ParamData: PParamData;
  PropInfo: PPropInfo;
  Candidates: TPropInfoList;
begin
  Btn := ItemByRow(ButtonsList.Row);
  if Btn = nil then Exit;
  Candidates := TPropInfoList.Create(Btn, [tkMethod]);
  try
    for I := Candidates.Count - 1 downto 0 do begin
      PropInfo := Candidates[I];
      if CompareText(PropInfo^.Name, 'OnClick') = 0 then begin
        Method := GetMethodProp(Btn, PropInfo);
        MethodName := TFormDesigner(Designer).GetMethodName(Method);
        if MethodName = '' then begin
          MethodName := Btn.Name + 'Click';
          Num := 0;
          while TFormDesigner(Designer).MethodExists(MethodName) do begin
            MethodName := Btn.Name + 'Click' + IntToStr(Num);
            Inc(Num);
          end;
          TypeData := AllocMem(SizeOf(TTypeData));
          try
            TypeData^.MethodKind := mkProcedure;
            TypeData^.ParamCount := 1;
            ParamData := PParamData(@TypeData^.ParamList);
            with ParamData^ do begin
              Flags := [];
              ParamNameAndType[0] := Char(Length(sSender));
              Move(sSender[1], ParamNameAndType[1], Length(sSender));
              ParamNameAndType[Length(sSender) + 1] := char(Length(sObject));
              Move(sObject[1], ParamNameAndType[Length(sSender) + 2],
                Length(sObject));
            end;
Method := TFormDesigner(Designer).CreateMethod(MethodName, TypeData);
            Method.Data := OwnerForm;
          finally
            FreeMem(TypeData, SizeOf(TTypeData));
          end;
          Btn.OnClick := TNotifyEvent(Method);
          Designer.Modified;
        end;
if (MethodName <> '') and TFormDesigner(Designer).MethodExists(MethodName) then
          TFormDesigner(Designer).ShowMethod(MethodName);
        Break;
      end;
    end;
  finally
    Candidates.Free;
  end;
end;

procedure TSpeedbarEditor.ButtonsListMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var // локальні змінні
  Item: TSpeedItem;
begin
  if (X < FBar.BtnWidth + 2) and (Button = mbLeft) then
```

```
  begin
    Item := ItemByRow(ButtonsList.Row);
    if Item <> nil then begin
      FDrag := True;
      if Item.Visible then FDragItem := nil
      else begin
        FDragItem := Item;
        if FButton = nil then begin
          FButton := TBtnControl.Create(Self);
          TBtnControl(FButton).AssignSpeedItem(Item);
        end;
      end;
    end;
  end;
end;

procedure TSpeedbarEditor.ButtonsListMouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var // локальні змінні
  P: TPoint;
begin
  if FDrag and (FButton <> nil) and (FDragItem <> nil) then begin
    P := (Sender as TControl).ClientToScreen(Point(X, Y));
    X := P.X - (FButton.Width {div 2});
    Y := P.Y - (FButton.Height {div 2});
    FButton.Activate(Bounds(X, Y, FBar.BtnWidth, FBar.BtnHeight));
  end
  else if FDrag then SetCursor(Screen.Cursors[crNoDrop]);
end;

procedure TSpeedbarEditor.ButtonsListMouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var // локальні змінні
  P: TPoint;
begin
  if FDrag and (Button = mbLeft) then
  try
    if (FDragItem <> nil) and (FButton <> nil) then begin
      Dec(X, FButton.Width {div 2});
      Dec(Y, FButton.Height {div 2});
      P := (Sender as TControl).ClientToScreen(Point(X, Y));
      FButton.Free;
      FButton := nil;
      if CheckSpeedBar and (FBar = FindSpeedBar(P)) then begin
        P := FBar.ScreenToClient(P);
        if FBar.AcceptDropItem(FDragItem, P.X, P.Y) then begin
          Designer.Modified;
        end;
      end;
    end
    else SetCursor(Screen.Cursors[ButtonsList.Cursor]);
  finally
    FDrag := False;
    FDragItem := nil;
  end;
end;

procedure TSpeedbarEditor.ButtonsListSelectCell(Sender: TObject; Col,
  Row: Longint; var CanSelect: Boolean);
var // локальні змінні
  Item: TSpeedItem;
begin
  Item := ItemByRow(Row);
  CanSelect := not FDrag and (Item <> nil);
  if FLocked = 0 then begin
    if CanSelect then begin
      UpdateEnabled(Row, SectionList.Row);
      SelectButton(CurrentSection, Item, False);
    end
```

```
    else if not FDrag then begin
      UpdateEnabled(-1, SectionList.Row);
      SelectButton(-1, nil, True);
    end;
  end;
end;

procedure TSpeedbarEditor.FormCreate(Sender: TObject);
begin
  FImage := TButtonImage.Create;
  FButton := nil;
  FBar := nil;
  FDrag := False;
  if NewStyleControls then Font.Style := [];

  with FormPlacement1 do begin
    UseRegistry := True;
    IniFileName := SDelphiKey;
  end;

end;

procedure TSpeedbarEditor.FormDestroy(Sender: TObject);
begin
  FImage.Free;
end;

procedure TSpeedbarEditor.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
  FButton.Free;
  FButton := nil;
  if FBar <> nil then begin
    FBar.SetEditing(0);
    SelectButton(-1, nil, True);
    FBar.Invalidate;
  end;
  FBar := nil;
end;

procedure TSpeedbarEditor.SectionNameKeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
begin
  if Key = (VK_RETURN) then begin
    SectionNameExit(SectionName);
    Key := 0;
    ActiveControl := SectionList;
  end;
end;

procedure TSpeedbarEditor.ButtonsListDrawCell(Sender: TObject; Col,
  Row: Longint; Rect: TRect; State: TGridDrawState);
var // локальні змінні
  I: Integer;
begin
  I := CurrentSection;
  if (I >= 0) and (Row < FBar.ItemsCount(I)) then
    DrawCellButton(Sender as TDrawGrid, Rect, ItemByRow(Row), FImage,
TDrawGrid(Sender).IsRightToLeft );
end;

procedure TSpeedbarEditor.SectionListMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var // локальні змінні
  ACol, ARow: Longint;
begin
  if (Button = mbLeft) then
    with (Sender as TDrawGrid) do begin
      MouseToCell(X, Y, ACol, ARow);
```

```
      Tag := Row;
      BeginDrag(False);
    end;
end;

procedure TSpeedbarEditor.SectionListDragDrop(Sender, Source: TObject; X,
  Y: Integer);
var // локальні змінні
  Col, Row: Longint;
begin
  try
    (Sender as TDrawGrid).MouseToCell(X, Y, Col, Row);
    FBar.Sections[(Sender as TDrawGrid).Tag].Index := Row;
    Designer.Modified;
    UpdateData;
    SectionList.Row := Row;
  finally
    (Sender as TDrawGrid).Tag := 0;
  end;
end;

procedure TSpeedbarEditor.SectionListDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
var // локальні змінні
  Col, Row: Longint;
begin
  (Sender as TDrawGrid).MouseToCell(X, Y, Col, Row);
  Accept := (Row >= 0) and (Row <> (Sender as TDrawGrid).Tag);
end;

procedure TSpeedbarEditor.FormShow(Sender: TObject);
begin
  if FBar <> nil then UpdateListHeight;
  SectionList.DefaultColWidth := SectionList.ClientWidth;
  ButtonsList.DefaultColWidth := ButtonsList.ClientWidth;
end;

end.
```

Файл бібліотеки Utils

```
unit Utils;
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Факультет механіко-технологічний
Кафедра кібербезпки та програмного забезпечення
Вихідний код до бакалаврської роботи
на тему: Програмна реалізація хмарної системи контролю успішності студентів
Виконав: Завірюха Євгеній Олександрович
Керівник: Дрєєв О.М.
Кропивницький 2024
}

interface // інтерфейсна частина

uses SysUtils;

type // створення типів даних

  TSysCharSet = set of Char;

  TCharSet = TSysCharSet;

{ ** Common string handling routines ** }


function StrToOem(const AnsiStr: string): string;
function OemToAnsiStr(const OemStr: string): string;
function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
function ReplaceStr(const S, Srch, Replace: string): string;
function DelSpace(const S: string): string;
function DelChars(const S: string; Chr: Char): string;
function DelBSpace(const S: string): string;
function DelESpace(const S: string): string;
function DelRSpace(const S: string): string;
function DelSpace1(const S: string): string;
function Tab2Space(const S: string; Numb: Byte): string;
function NPos(const C: string; S: string; N: Integer): Integer;
function MakeStr(C: Char; N: Integer): string;
function MS(C: Char; N: Integer): string;
function AddChar(C: Char; const S: string; N: Integer): string;
function AddCharR(C: Char; const S: string; N: Integer): string;
function LeftStr(const S: string; N: Integer): string;
function RightStr(const S: string; N: Integer): string;
function CenterStr(const S: string; Len: Integer): string;
function CompStr(const S1, S2: string): Integer;
function CompText(const S1, S2: string): Integer;
function Copy2Symb(const S: string; Symb: Char): string;
function Copy2SymbDel(var S: string; Symb: Char): string;
function Copy2Space(const S: string): string;
function Copy2SpaceDel(var S: string): string;
function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
function WordCount(const S: string; const WordDelims: TCharSet): Integer;
function WordPosition(const N: Integer; const S: string;
function ExtractWord(N: Integer; const S: string;
  const WordDelims: TCharSet): string;
function ExtractWordPos(N: Integer; const S: string;
  const WordDelims: TCharSet; var Pos: Integer): string;
function ExtractDelimited(N: Integer; const S: string;
  const Delims: TCharSet): string;
function ExtractSubstr(const S: string; var Pos: Integer;
  const Delims: TCharSet): string;
function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
function QuotedString(const S: string; Quote: Char): string;
function ExtractQuotedString(const S: string; Quote: Char): string;
function FindPart(const HelpWilds, InputStr: string): Integer;
function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
function XorString(const Key, Src: ShortString): ShortString;
```

```
function XorEncode(const Key, Source: string): string;
function XorDecode(const Key, Source: string): string;
function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;
  IgnoreCase: Boolean): Boolean;
function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
function Numb2USA(const S: string): string;
function Dec2Hex(N: Longint; A: Byte): string;
function D2H(N: Longint; A: Byte): string;
function Hex2Dec(const S: string): Longint;
function H2D(const S: string): Longint;
function Dec2Numb(N: Longint; A, B: Byte): string;

const
  CRLF = #13#10;
  DigitChars = ['0'..'9'];

  Brackets = ['(',')','[',']','{','}'];
  StdWordDelims = [#0..' ',',','.',';','/','\',':','''','"','`'] + Brackets;

implementation


uses  Windows  WinTypes, WinProcs ;

function StrToOem(const AnsiStr: string): string;
begin
  SetLength(Result, Length(AnsiStr));
  if Length(Result) > 0 then
    CharToOemBuff(PChar(AnsiStr), PChar(Result), Length(Result));
    AnsiToOemBuff(@AnsiStr[1], @Result[1], Length(Result));
end;

function OemToAnsiStr(const OemStr: string): string;
begin
  SetLength(Result, Length(OemStr));
  if Length(Result) > 0 then
    OemToCharBuff(PChar(OemStr), PChar(Result), Length(Result));
    OemToAnsiBuff(@OemStr[1], @Result[1], Length(Result));
end;

function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
var // локальні змінні
  I, SLen: Integer;
begin
  SLen := Length(S);
  I := 1;
  while I <= SLen do begin
    if not (S[I] in EmptyChars) then begin
      Result := False;
      Exit;
    end
    else Inc(I);
  end;
  Result := True;
end;

function ReplaceStr(const S, Srch, Replace: string): string;
var // локальні змінні
  I: Integer;
  Source: string;
begin
  Source := S;
  Result := '';
  repeat
    I := Pos(Srch, Source);
    if I > 0 then begin
      Result := Result + Copy(Source, 1, I - 1) + Replace;
      Source := Copy(Source, I + Length(Srch), MaxInt);
    end
```

```
    else Result := Result + Source;
  until I <= 0;
end;

function DelSpace(const S: String): string;
begin
  Result := DelChars(S, ' ');
end;

function DelChars(const S: string; Chr: Char): string;
var // локальні змінні
  I: Integer;
begin
  Result := S;
  for I := Length(Result) downto 1 do begin
    if Result[I] = Chr then Delete(Result, I, 1);
  end;
end;

function DelBSpace(const S: string): string;
var // локальні змінні
  I, L: Integer;
begin
  L := Length(S);
  I := 1;
  while (I <= L) and (S[I] = ' ') do Inc(I);
  Result := Copy(S, I, MaxInt);
end;

function DelESpace(const S: string): string;
var // локальні змінні
  I: Integer;
begin
  I := Length(S);
  while (I > 0) and (S[I] = ' ') do Dec(I);
  Result := Copy(S, 1, I);
end;

function DelRSpace(const S: string): string;
begin
  Result := DelBSpace(DelESpace(S));
end;

function DelSpace1(const S: string): string;
var // локальні змінні
  I: Integer;
begin
  Result := S;
  for I := Length(Result) downto 2 do begin
    if (Result[I] = ' ') and (Result[I - 1] = ' ') then
      Delete(Result, I, 1);
  end;
end;

function Tab2Space(const S: string; Numb: Byte): string;
var // локальні змінні
  I: Integer;
begin
  I := 1;
  Result := S;
  while I <= Length(Result) do begin
    if Result[I] = Chr(9) then begin
      Delete(Result, I, 1);
      Insert(MakeStr(' ', Numb), Result, I);
      Inc(I, Numb);
    end
    else Inc(I);
  end;
end;
```

```
function MakeStr(C: Char; N: Integer): string;
begin
  if N < 1 then Result := ''
  else begin

    if N > 255 then N := 255;
    SetLength(Result, N);
    FillChar(Result[1], Length(Result), C);
  end;
end;

function MS(C: Char; N: Integer): string;
begin
  Result := MakeStr(C, N);
end;

function NPos(const C: string; S: string; N: Integer): Integer;
var // локальні змінні
  I, P, K: Integer;
begin
  Result := 0;
  K := 0;
  for I := 1 to N do begin
    P := Pos(C, S);
    Inc(K, P);
    if (I = N) and (P > 0) then begin
      Result := K;
      Exit;
    end;
    if P > 0 then Delete(S, 1, P)
    else Exit;
  end;
end;

function AddChar(C: Char; const S: string; N: Integer): string;
begin
  if Length(S) < N then
    Result := MakeStr(C, N - Length(S)) + S
  else Result := S;
end;

function AddCharR(C: Char; const S: string; N: Integer): string;
begin
  if Length(S) < N then
    Result := S + MakeStr(C, N - Length(S))
  else Result := S;
end;

function LeftStr(const S: string; N: Integer): string;
begin
  Result := AddCharR(' ', S, N);
end;

function RightStr(const S: string; N: Integer): string;
begin
  Result := AddChar(' ', S, N);
end;

function CompStr(const S1, S2: string): Integer;
begin

  Result := CompareString(GetThreadLocale, SORT_STRINGSORT, PChar(S1),
    Length(S1), PChar(S2), Length(S2)) - 2;

  Result := CompareStr(S1, S2);

end;
```

```
function CompText(const S1, S2: string): Integer;
begin

  Result := CompareString(GetThreadLocale, SORT_STRINGSORT or NORM_IGNORECASE,
    PChar(S1), Length(S1), PChar(S2), Length(S2)) - 2;

  Result := CompareText(S1, S2);

end;

function Copy2Symb(const S: string; Symb: Char): string;
var // локальні змінні
  P: Integer;
begin
  P := Pos(Symb, S);
  if P = 0 then P := Length(S) + 1;
  Result := Copy(S, 1, P - 1);
end;

function Copy2SymbDel(var S: string; Symb: Char): string;
begin
  Result := Copy2Symb(S, Symb);
  S := DelBSpace(Copy(S, Length(Result) + 1, Length(S)));
end;

function Copy2Space(const S: string): string;
begin
  Result := Copy2Symb(S, ' ');
end;

function Copy2SpaceDel(var S: string): string;
begin
  Result := Copy2SymbDel(S, ' ');
end;

function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
var // локальні змінні
  SLen, I: Cardinal;
begin
  Result := AnsiLowerCase(S);
  I := 1;
  SLen := Length(Result);
  while I <= SLen do begin
    while (I <= SLen) and (Result[I] in WordDelims) do Inc(I);
    if I <= SLen then Result[I] := AnsiUpperCase(Result[I])[1];
    while (I <= SLen) and not (Result[I] in WordDelims) do Inc(I);
  end;
end;

function WordCount(const S: string; const WordDelims: TCharSet): Integer;
var // локальні змінні
  SLen, I: Cardinal;
begin
  Result := 0;
  I := 1;
  SLen := Length(S);
  while I <= SLen do begin
    while (I <= SLen) and (S[I] in WordDelims) do Inc(I);
    if I <= SLen then Inc(Result);
    while (I <= SLen) and not(S[I] in WordDelims) do Inc(I);
  end;
end;

function WordPosition(const N: Integer; const S: string;
  const WordDelims: TCharSet): Integer;
var // локальні змінні
  Count, I: Integer;
begin
  Count := 0;
```

```
  I := 1;
  Result := 0;
  while (I <= Length(S)) and (Count <> N) do begin
    { skip over delimiters }
    while (I <= Length(S)) and (S[I] in WordDelims) do Inc(I);
    if I <= Length(S) then Inc(Count);
    { if not finished, find the end of the current word }
    if Count <> N then
      while (I <= Length(S)) and not (S[I] in WordDelims) do Inc(I)
    else Result := I;
  end;
end;

function ExtractWord(N: Integer; const S: string;
  const WordDelims: TCharSet): string;
var // локальні змінні
  I: Integer;
  Len: Integer;
begin
  Len := 0;
  I := WordPosition(N, S, WordDelims);
  if I <> 0 then
    while (I <= Length(S)) and not(S[I] in WordDelims) do begin
      Inc(Len);
      SetLength(Result, Len);
      Result[Len] := S[I];
      Inc(I);
    end;
  SetLength(Result, Len);
end;

function ExtractWordPos(N: Integer; const S: string;
  const WordDelims: TCharSet; var Pos: Integer): string;
var // локальні змінні
  I, Len: Integer;
begin
  Len := 0;
  I := WordPosition(N, S, WordDelims);
  Pos := I;
  if I <> 0 then
    while (I <= Length(S)) and not(S[I] in WordDelims) do begin
      Inc(Len);
      SetLength(Result, Len);
      Result[Len] := S[I];
      Inc(I);
    end;
  SetLength(Result, Len);
end;

function ExtractDelimited(N: Integer; const S: string;
  const Delims: TCharSet): string;
var // локальні змінні
  CurWord: Integer;
  I, Len, SLen: Integer;
begin
  CurWord := 0;
  I := 1;
  Len := 0;
  SLen := Length(S);
  SetLength(Result, 0);
  while (I <= SLen) and (CurWord <> N) do begin
    if S[I] in Delims then Inc(CurWord)
    else begin
      if CurWord = N - 1 then begin
        Inc(Len);
        SetLength(Result, Len);
        Result[Len] := S[I];
      end;
    end;
```

```
    Inc(I);
  end;
end;

function ExtractSubstr(const S: string; var Pos: Integer;
  const Delims: TCharSet): string;
var // локальні змінні
  I: Integer;
begin
  I := Pos;
  while (I <= Length(S)) and not (S[I] in Delims) do Inc(I);
  Result := Copy(S, Pos, I - Pos);
  if (I <= Length(S)) and (S[I] in Delims) then Inc(I);
  Pos := I;
end;

function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
var // локальні змінні
  Count, I: Integer;
begin
  Result := False;
  Count := WordCount(S, WordDelims);
  for I := 1 to Count do
    if ExtractWord(I, S, WordDelims) = W then begin
      Result := True;
      Exit;
    end;
end;

function QuotedString(const S: string; Quote: Char): string;
begin
  Result := AnsiQuotedStr(S, Quote);

var // локальні змінні
  I: Integer;
begin
  Result := S;
  for I := Length(Result) downto 1 do
    if Result[I] = Quote then Insert(Quote, Result, I);
  Result := Quote + Result + Quote;
end;

function ExtractQuotedString(const S: string; Quote: Char): string;
var // локальні змінні
{$IFDEF MBCS}
  P: PChar;
begin
  P := PChar(S);
  if P^ = Quote then Result := AnsiExtractQuotedStr(P, Quote)
  else Result := S;

  I: Integer;
begin
  Result := S;
  I := Length(Result);
  if (I > 0) and (Result[1] = Quote) and
    (Result[I] = Quote) then
  begin
    Delete(Result, I, 1);
    Delete(Result, 1, 1);
    for I := Length(Result) downto 2 do begin
      if (Result[I] = Quote) and (Result[I - 1] = Quote) then
        Delete(Result, I, 1);
    end;
  end;
end;

function Numb2USA(const S: string): string;
var // локальні змінні
```

```
  I, NA: Integer;
begin
  I := Length(S);
  Result := S;
  NA := 0;
  while (I > 0) do begin
    if ((Length(Result) - I + 1 - NA) mod 3 = 0) and (I <> 1) then
    begin
      Insert(',', Result, I);
      Inc(NA);
    end;
    Dec(I);
  end;
end;

function CenterStr(const S: string; Len: Integer): string;
begin
  if Length(S) < Len then begin
    Result := MakeStr(' ', (Len div 2) - (Length(S) div 2)) + S;
    Result := Result + MakeStr(' ', Len - Length(Result));
  end
  else Result := S;
end;

function Dec2Hex(N: LongInt; A: Byte): string;
begin
  Result := IntToHex(N, A);
end;

function D2H(N: LongInt; A: Byte): string;
begin
  Result := IntToHex(N, A);
end;

function Hex2Dec(const S: string): Longint;
var // локальні змінні
  HexStr: string;
begin
  if Pos('$', S) = 0 then HexStr := '$' + S
  else HexStr := S;
  Result := StrToIntDef(HexStr, 0);
end;

function H2D(const S: string): Longint;
begin
  Result := Hex2Dec(S);
end;

function Dec2Numb(N: Longint; A, B: Byte): string;
var // локальні змінні
  C: Integer;
  Number: Cardinal;

  Number: Longint;

begin
  if N = 0 then Result := '0'
  else begin
    Number := Cardinal(N);
    Number := N;
    Result := '';
    while Number > 0 do begin
      C := Number mod B;
      if C > 9 then C := C + 55
      else C := C + 48;
      Result := Chr(C) + Result;
      Number := Number div B;
    end;
  end;
```

```
  if Result <> '' then Result := AddChar('0', Result, A);
end;

function Numb2Dec(S: string; B: Byte): Longint;
var // локальні змінні
  I, P: Longint;
begin
  I := Length(S);
  Result := 0;
  S := UpperCase(S);
  P := 1;
  while (I >= 1) do begin
    if S[I] > '@' then Result := Result + (Ord(S[I]) - 55) * P
    else Result := Result + (Ord(S[I]) - 48) * P;
    Dec(I);
    P := P * B;
  end;
end;

function RomanToInt(const S: string): Longint;
const
  RomanChars = ['C','D','I','L','M','V','X'];
  RomanValues: array['C'..'X'] of Word =
    (100,500,0,0,0,0,1,0,0,50,1000,0,0,0,0,0,0,0,0,5,0,10);
var // локальні змінні
  Index, Next: Char;
  I: Integer;
  Negative: Boolean;
begin
  Result := 0;
  I := 0;
  Negative := (Length(S) > 0) and (S[1] = '-');
  if Negative then Inc(I);
  while (I < Length(S)) do begin
    Inc(I);
    Index := UpCase(S[I]);
    if Index in RomanChars then begin
      if Succ(I) <= Length(S) then Next := UpCase(S[I + 1])
      else Next := #0;
      if (Next in RomanChars) and (RomanValues[Index] < RomanValues[Next]) then
      begin
        Inc(Result, RomanValues[Next]);
        Dec(Result, RomanValues[Index]);
        Inc(I);
      end
      else Inc(Result, RomanValues[Index]);
    end
    else begin
      Result := 0;
      Exit;
    end;
  end;
  if Negative then Result := -Result;
end;

function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
begin
  Result := '';
  if Digits > 32 then Digits := 32;
  while Digits > 0 do begin
    if (Digits mod Spaces) = 0 then Result := Result + ' ';
    Dec(Digits);
    Result := Result + IntToStr((Value shr Digits) and 1);
  end;
end;

function FindPart(const HelpWilds, InputStr: string): Integer;
var // локальні змінні
  I, J: Integer;
```

```
  Diff: Integer;
begin
  I := Pos('?', HelpWilds);
  if I = 0 then begin
    { if no '?' in HelpWilds }
    Result := Pos(HelpWilds, InputStr);
    Exit;
  end;
  Diff := Length(InputStr) - Length(HelpWilds);
  if Diff < 0 then begin
    Result := 0;
    Exit;
  end;
  { now move HelpWilds over InputStr }
  for I := 0 to Diff do begin
    for J := 1 to Length(HelpWilds) do begin
      if (InputStr[I + J] = HelpWilds[J]) or
        (HelpWilds[J] = '?') then
      begin
        if J = Length(HelpWilds) then begin
          Result := I + 1;
          Exit;
        end;
      end
      else Break;
    end;
  end;
  Result := 0;
end;

function XorString(const Key, Src: ShortString): ShortString;
var // локальні змінні
  I: Integer;
begin
  Result := Src;
  if Length(Key) > 0 then
    for I := 1 to Length(Src) do
      Result[I] := Chr(Byte(Key[1 + ((I - 1) mod Length(Key))]) xor
Ord(Src[I]));
end;

function XorEncode(const Key, Source: string): string;
var // локальні змінні
  I: Integer;
  C: Byte;
begin
  Result := '';
  for I := 1 to Length(Source) do begin
    if Length(Key) > 0 then
      C := Byte(Key[1 + ((I - 1) mod Length(Key))]) xor Byte(Source[I])
    else
      C := Byte(Source[I]);
    Result := Result + AnsiLowerCase(IntToHex(C, 2));
  end;
end;

function XorDecode(const Key, Source: string): string;
var // локальні змінні
  I: Integer;
  C: Char;
begin
  Result := '';
  for I := 0 to Length(Source) div 2 - 1 do begin
    C := Chr(StrToIntDef('$' + Copy(Source, (I * 2) + 1, 2), Ord(' ')));
    if Length(Key) > 0 then
      C := Chr(Byte(Key[1 + (I mod Length(Key))]) xor Byte(C));
    Result := Result + C;
  end;
end;
```

```
function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;
  IgnoreCase: Boolean): Boolean;
var // локальні змінні
  I: Integer;
  S: string;
begin
  for I := 1 to ParamCount do begin
    S := ParamStr(I);
    if (SwitchChars = []) or ((S[1] in SwitchChars) and (Length(S) > 1)) then
    begin
      S := Copy(S, 2, MaxInt);
      if IgnoreCase then begin
        if (AnsiCompareText(S, Switch) = 0) then begin
          Result := True;
          Exit;
        end;
      end
      else begin
        if (AnsiCompareStr(S, Switch) = 0) then begin
          Result := True;
          Exit;
        end;
      end;
    end;
  end;
  Result := False;
end;

function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
var // локальні змінні
  I: Integer;
  S: string;
begin
  I := 1;
  while I <= ParamCount do begin
    S := ParamStr(I);
    if (SwitchChars = []) or ((S[1] in SwitchChars) and (Length(S) > 1)) then
    begin
      if (AnsiCompareText(Copy(S, 2, MaxInt), Switch) = 0) then begin
        Inc(I);
        if I <= ParamCount then begin
          Result := ParamStr(I);
          Exit;
        end;
      end;
    end;
    Inc(I);
  end;
  Result := '';
end;end.
```