МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2019 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Моделі та програмні засоби розробки клієнтської частини комп'ютерної системи моніторингу якості освіти

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 – "Комп'ютерна інженерія"

Науковий керівник роботи: _____ (підпис) Л.О. Шумова (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О. Критська (ініціали, прізвище)

Студент: _____ (підпис) Н.О. Височина (ініціали, прізвище)

Група: КІ-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Височиній Надії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Моделі та програмні засоби розробки клієнтської частини комп'ютерної системи моніторингу якості освіти

керівник проекту (роботи) Шумова Лариса Олександрівна, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "18" 10 2018 р. № 220/48
2. Строк подання студентом роботи 09.01.2019 р.
3. Вихідні дані до роботи дані, зібрані під час проходження науково-дослідної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз методів і засобів розробки комп'ютерної системи моніторингу якості освіти, розробка концепції комп'ютерної системи моніторингу, розробка клієнтської частини системи. Розгляд питань з охорони праці та безпеки в надзвичайних ситуаціях, екології.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях. Екологія	Критська Я.О., ст.викладач		

7. Дата видачі завдання 18.10.2018 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналітичний огляд літератури за темою роботи	1.09.18 – 23.09.18	
2	Аналіз методів розробки комп'ютерної системи моніторингу якості освіти	24.09.18 – 07.10.18	
3	Аналіз програмних та інструментальних засобів контролю знань	08.10.18 – 21.10.18	
4	Розробка концепції комп'ютерної системи	22.10.18 – 04.11.18	
5	Формування структурної моделі системи	05.11.18 – 18.11.18	
6	Розгляд питань з охорони праці та безпеки в надзвичайних ситуаціях, екології	19.11.18 – 30.11.18	
7	Оформлення пояснювальної записки	01.12.18 – 21.12.18	
8	Оформлення презентації роботи	24.12.18 – 6.01.19	

Студент _____ **Височина Н.О.**
(підпис) (прізвище та ініціали)

Науковий керівник _____ **Шумова Л.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Височина Н.О. Моделі та програмні засоби розробки клієнтської частини комп’ютерної системи моніторингу якості освіти.

Розглянуті актуальні завдання моніторингу та проектні рішення при розробці комп'ютерних систем для забезпечення ефективного моніторингу якості освіти. Представлена структурна модель комп’ютерної інформаційної системи, визначено її структурні елементи. Розроблено допоміжне програмне забезпечення. Розроблено інтерфейс та створено робочий прототип клієнтської частини системи.

Ключові слова: комп‘ютерна система, база даних, реплікація, якість освіти, моніторинг, програмне забезпечення.

АННОТАЦИЯ

Высочина Н.А. Модели и программные средства разработки клиентской части компьютерной системы мониторинга качества образования.

Рассмотрены актуальне задачи мониторинга и проектне решения при разработке компьютерных систем для обеспечения эффективного мониторинга качества образования. Представлена структурная модель компьютерной информационной системы, определены её структурне элементы. Разработано вспомогательное программное обеспечение. Разработан інтерфейс и создан рабочий прототип клиентской части системы.

Ключевые слова: компьютерная система, база данных, репликация, качество образования, моніторинг, программное обеспечение.

ABSTRACT

Vysochina N. Models and software development of the client part of the computer system for monitoring the quality of education.

Considered the current tasks of monitoring and design decisions in the development of computer systems to ensure effective monitoring of the quality of education. Determined a structural model of a computer information system is presented, its structural elements. Developed auxiliary software. Created an interface was developed and a working prototype of the client part of the system.

Keywords: computer system, database, replication, quality of education, monitoring, software.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЗЗСО – заклад загальної середньої освіти

ЄАРІС МЯО - єдина автоматизована регіональна інформаційна система моніторингу якості освіти

ІКТ - інформаційно-комп’ютерні технології

СКТ - системи комп'ютерного контролю знань

ЛОІППО - Луганський обласний інститут післядипломної педагогічної освіти

ІС – інформаційна система

АРМ - автоматизоване робоче місце

РЦМ - регіональний центр моніторингу

БД – база даних

ОС – операційна система

САПР – система автоматизованого проектування

ПЗ – програмне забезпечення

СУБД – систеа управління базами даних

ЕОМ – електронна обчислювальна машина

АІС – автоматизована інформаційна система

ЦБД – центральна база даних

СОПК – система обміну повідомленнями клієнт

СОПС - система обміну повідомленнями сервер

НДІ – нормативно-довідкова інформація

КВМ – контрольно-вимірювальні матеріали

ЗНО - зовнішнє незалежне оцінювання

ОЗ – оціночні заходи

ЛБД – локальна база даних

ПК – персональний комп’ютер

ВСТУП

Обґрунтування вибору теми дослідження. Розвиток освіти в Україні відбувається одночасно з сучасними європейськими культурно-освітніми процесами. Свідченням цього є оновлене освітнє законодавство України: Закон України «Про вищу освіту» та Національна стратегія розвитку освіти в Україні на період до 2021 року. Ці документи визначили подальший курс реформування освіти, мету і пріоритети її розвитку, національний характер освіти та інші важливі питання освітньої політики держави.

Якісна освіта розглядається сьогодні міжнародною спільнотою як одна з необхідних умов успішного розвитку країни. Зокрема, ЮНЕСКО акцентує питання якісної освіти на набуття молоддю життєвих компетенцій для успішного входження у сучасне суспільство задля досягнення соціальної злагоди. Європейський Союз розглядає якісну освіту передусім як інструмент економічного зростання на шляху до розбудови більш компетентно спроможної та динамічної спільноти.

Для ефективного управління освітою на кожній ланці освітнього простору важливим є виконання певних умов, серед яких дві є головними:

- принцип правди – рішення в освіті мають прийматися на повних, достовірних, надійних, актуальних даних про стан і динаміку розвитку освіти в доступній та зручній для управлінця формі;
- принцип відкритості – знеособлені дані про стан освіти мають бути доступними усім зацікавленим особам системи освіти (учителям, адміністрації навчальних закладів, керівникам методичних служб та органів управління освітою) в доступній та зручній для нього формі.

Дотримання вищеназваних принципів впливає на покращення якості освіти та прийняття ефективних управлінських рішень щодо функціонування системи освіти будь-якої країни.

Для України проблема якості освіти є національним пріоритетом, передумовою національної безпеки держави та гарантом забезпечення міжнародних норм і вимог законодавства щодо реалізації права громадян на освіту.

Найголовнішим завданням реформування системи освіти в Україні на основі принципів правди, прозорості й підзвітності є створення державної системи моніторингу освіти.

Моніторинг в освіті повинен створювати інформаційну систему, яка постійно поповнюється, що є показником безперервності відстеження, а також включати розробку апарату та технології вимірювання існуючого стану об'єкту. Він потребує систематичності та послідовності дослідження проблем, а якість результатів моніторингу (наприклад, рівень навчальних досягнень) залежить від якості технології та інструментарію для оцінювання.

Формування ефективної системи моніторингу якості освіти неможливе без функціонування високопродуктивних інформаційних систем, за допомогою яких генерується, обробляється й

аналізується інформація, отримана в результаті здійснення моніторингових досліджень, та інші дані, що можуть бути використані для подальшої аналітичної діяльності. Це свідчить про актуальність проведення моніторингових досліджень якості освіти та розробки інструментальних засобів, що забезпечують його ефективність. Незважаючи на те, що існує достатня кількість програмних засобів, які можуть використовуватися для зберігання, передачі, обробки даних моніторингу, актуальною залишається задача розробки комплексу взаємопов'язаних інструментальних засобів, об'єднаних в єдину інформаційну систему моніторингу якості освіти в ЗЗСО.

Аналіз наукової літератури свідчить, що питання управління освітою на основі отриманої інформації були предметом спеціальних досліджень ще в 20-ті роки XX століття. Суттєвий внесок у розв'язання цієї проблеми вніс М. Йорданський. Ним було обґрунтовано положення про організацію збору та обробки інформації з метою подальшого використання в управлінні навчальним процесом. У наш час моніторингові дослідження активно використовуються з метою оцінки якості освіти та здатності освітньої системи до розвитку.

Практика застосування різних технологій та систем автоматизації контролю навчальних досягнень учнів, відкриває можливості для створення ЄАРІС МЯО - системи збору, обробки та інтерпретації інформації про результати освітньої діяльності. Очевидною є необхідність застосування клієнт-серверної технології, та використання web-інтерфейсів для забезпечення роботи системи.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** розробки клієнтської частини комп'ютерної системи моніторингу якості освіти.

Об'єкт дослідження **–** процеси створення та експлуатації програмно-технічних комплексів для моніторингу якості освіти.

Предмет дослідження – програмне забезпечення комп'ютерної системи моніторингу якості освіти.

Мета і задачі дослідження. Метою дослідження є підвищення оперативності обробки інформації, отриманої під час проведення моніторингових заходів оцінювання якості освіти.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

- аналіз предметної області;
- формування основних напрямків розробки комп'ютерної системи;
- аналіз методів і засобів отримання інформації;
- розроблення моделей обробки інформації для подальшого аналізу;
- розроблення системи проведення моніторингових досліджень, обробки інформації та аналітичного аналізу;
- проектування загальної схеми роботи комп'ютерної інформаційної системи;
- розроблення програмних засобів і елементів інформаційної технології для проведення

моніторингових досліджень.

Методи дослідження. Проведені у роботі дослідження та розробка ґрунтуються на принципах системного аналізу, принципів створення програмного забезпечення інформаційних систем.

Наукова новизна отриманих результатів:

Удосконалено процеси проведення моніторингових досліджень шляхом створення єдиної комп'ютерої системи, що дозволяє підвищити якість та швидкість обробки інформації.

Особистий внесок здобувача полягає у розробленні нових інструментальних засобів, що дозволяють вирішити поставлені задачі. Усі основні результати отримані автором особисто.

Апробація матеріалів дисертації. Основні положення та ідеї магістерської роботи доповідалися та обговорювалися на форумі «ІТ-ідея – 2017» [1] та описані у статті «Комп'ютерні засоби розробки інформаційної системи моніторингу якості освіти», що опублікована у «Віснику Східноукраїнського національного університету ім. В. Даля» [2].

Практичне значення отриманих результатів полягає в тому, що основні наукові положення дисертації реалізовані у виді програмних засобів, які утворюють інформаційну систему для проведення моніторингових досліджень.

Публікації. За темою магістерської роботи з викладенням її основних результатів опубліковано 1 тези та 1 стаття в науковому фаховому виданні України.

Структура та обсяг дисертації. Дисертація складається із вступу, чотирьох розділів, висновків, списку використаних джерел і 2 додатків. Загальний обсяг дисертації складає 91 сторінок. Робота містить 2 таблиці, та 42 рисунки.

РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ОСВІТИ

1.1 Актуальність задач моніторингу якості освіти

Моніторингові дослідження є одним з найважливіших аспектів системи контролю якості освіти, яка відображає ступінь відповідності реальних освітніх результатів нормативним вимогам. На сьогоднішній день в якості найбільш сучасного інструменту для виявлення якісного результату освітньої діяльності застосовуються автоматизовані інформаційні системи моніторингу.

Організація моніторингу в сучасних умовах неможлива без використання комп'ютерних технологій. Їхнє застосування стає життєво важливою потребою для управління, оскільки прискореними темпами зростають потоки інформації і звичні паперові форми та звітність уже помітно гальмують ефективне управління.

Сучасний інформаційний ринок пропонує відносно невелику кількість програмних комплексів, розроблених різними фірмами, які дозволяють створити на їх основі єдиний інформаційний простір управління, як у окремому навчальному закладі, так і в територіальній системі освіти в цілому. Однак вони не в повній мірі задовольняють усім аспектам освітнього моніторингу, зокрема, можливостям ретельного аналізу даних і зручного відображення результатів. Потрібні нові принципові підходи щодо оцінювання навчального процесу як системного явища, відповідні технології оперативного збирання та обробки педагогічно значущої інформації.

Аналіз наукової літератури та останніх публікацій доводить, що проблема модернізації управління системою освіти – важлива соціальна задача, вирішення якої забезпечує необхідне покращення якості підготовки учнів [3]. Авторами останніх публікацій відмічено, що визначальною тенденцією розвитку національної системи освіти в Україні є її інформатизація, яка розглядається як невід'ємна складова інформатизації українського суспільства, відображує загальні тенденції глобалізації світових процесів розвитку цивілізації, виступає як визначальний інформаційний і комунікаційний базис розвитку особистості та соціально-економічних систем нашої держави [4]. Поняття інформатизації освіти пов'язується з широким впровадженням у систему освіти методів і засобів ІКТ, створенням на цій основі комп'ютерно орієнтованого інформаційно-комунікаційного середовища, з його подальшим наповненням науковими, освітніми та управлінськими інформаційними ресурсами, з наданням можливостей суб'єктам освітнього процесу здійснювати доступ до ресурсів середовища, використовувати його засоби і сервіси при розв'язуванні різних завдань [5]. У XXI столітті людина живе і діє в умовах постійних змін. Динамізм, мінливість у різних сферах сучасного життя стає його невіддільною ознакою. Завдання

ж освіти полягає в тому, щоб підготувати людину до адекватного продукування і сприйняття змін, до готовності вчасно відмовитися від старого досвіду і стереотипів. І це також є важливим складником якості сучасної освіти [6]. Використання комп'ютерних технологій дозволяє оптимізувати процес контролю якості освіти. Оптимізація процесу передбачає мінімальні затрати зусиль і часу для отримання об'єктивних відомостей про якість засвоєння навчального матеріалу [7]. Аналіз, проведений у роботі Пліш І.В. [8], підтвердив, що проблема використання ІКТ управління якістю освіти в закладах загальної середньої освіти України ще не стала об'єктом окремого дослідження, за результатами якого можна робити теоретичні висновки та провадження яких у практику освітнього менеджменту надасть можливість суттєво поліпшити якість освіти.

Проаналізувавши публікації на тему впровадження ІКТ у процес проведення моніторингових досліджень, можна зробити висновок про те, що на сьогодні не існує єдиної комп'ютерної системи, яка забезпечить проведення дистанційних моніторингових досліджень, але в усіх дослідженнях та публікаціях автори наголошують на тому, що розробка універсальної комп'ютерної системи є необхідною, тому що вона здатна скоротити затрати часу на обробку великої кількості інформації.

1.2 Огляд існуючих підходів до моніторингу якості освіти

Методи та технології моніторингу різноманітні, у якості основних слід виділити наступні методи моніторингових досліджень:

- метод порівняльних оцінок (порівняння з нормами якості освіти як базами оцінки);
- метод експертних оцінок;
- аналіз документальних джерел;
- аналіз результатів предметних тестів і даних успішності;
- збір та аналіз статистичних даних про різні види діяльності освітніх систем;
- аналіз баз та банків даних;
- імітаційне моделювання на ЕОМ;
- соціологічні методи (соціологічні та соціометричні опитування, факторний та кореляційний аналіз, соціальне прогнозування).

Всі застосовувані способи здійснення моніторингу можна розділити на наступні групи.

Поточне спостереження здійснюється з метою відстеження змін професійного розвитку під впливом освітнього процесу і визначення сенсу явищ, що відбуваються. Ефективність педагогічного спостереження залежить від психологічної компетентності педагога, його досвіду, ставлення до студентів, професійної позиції і т.д. Спостереження завжди характеризується суб'єктивністю, що може негативно позначитися на якості моніторингу.

Метод тестових ситуацій полягає в тому, що педагог створює спеціальні умови, в яких кожен із структурних компонентів навчально-професійної діяльності виявляється найвиразніше. Для цього використовуються прийоми переривання навчальних дій учнів, постановки уточнюючих питань, стимулювання рефлексії своїх пізнавальних дій, дозування допомоги в навчанні та ін.

Експлікація *(від лат. Explicatio - розгортання, роз'яснення)* має на увазі розгортання змісту навчально-професійної діяльності. Цей метод дозволяє не тільки діагностувати зміни, що відбуваються в розвитку учня, а й оперативно вносити корективи в процес освіти. Експлікація здійснюється шляхом постановки навідних запитань, надання допомоги у вигляді підказок і спільних дій, заохочення педагогом учнів. Реєстрація експлікаційних характеристик здійснюється в найпростішому випадку за допомогою використання методу спостереження, а фіксація даних - за допомогою опитувальників, у яких відображаються емпірично спостережувані навчально-професійні дії та якості учнів.

Опитувальний метод дозволяє отримати інформацію про розвиток суб'єктів освітнього процесу на основі аналізу письмових або усних відповідей на стандартні спеціально підібрані питання. Опитувальники дають можливість визначити рівень вираженості або сформованості основних компонентів навчально-професійної діяльності, особливості спрямованості учнів та педагогів, а також окремі навчально-пізнавальні властивості і якості.

Аналіз результатів навчально-професійної діяльності, при якому по заздалегідь наміченій схемі вивчаються письмові тексти, графічні матеріали, технічні вироби, творчі роботи учнів.

Тестування є одним із суб'єктивних методів збору даних про рівень розвитку педагогічних процесів і ступеня вираженості психічного розвитку суб'єктів освіти. Важливою перевагою тестування є орієнтація на норму, що дозволяє зіставляти, порівнювати оцінки, отримані за допомогою тесту. Для моніторингу застосовують інтелектуальні, особистісні, міжособистісні тести, практичні тестові завдання, процесуальні тести.

1.3 Аналіз програмних та інструментальних засобів контролю знань

СКТ - це системи тестування, що дозволяють проводити аналіз і оцінку знань учнів за допомогою сучасних інформаційних технологій. Одна з переваг автоматизованих систем контролю знань полягає в тому, що вони можуть використовувати складні методики подання завдань учням, звані стратегіями тестування.

Наразі існує велика кількість комп’ютерних систем контролю знань, їх класифікують наступним чином:

- по цілям, що досягаються в ході тестування (атестаційні, навчальні, інформаційні і т.і.);
- за процедурою створення (стандартизовані або не стандартизовані);
- за способом формування завдань: детерміновані, стохастичні та динамічні;
- за можливістю внесення змін: відкриті та закриті;
- за наявністю зворотного зв'язку: традиційні або адаптивні;
- за особливостями технічної реалізації: локальні, глобальні;
- за спеціалізацією: профільні або універсальні.

Основними модулями систем контролю знань є:

- модуль адміністрування бази тестів;
- модуль управління проведенням тестування;
- модуль тестованого;
- модуль адміністрування тестованих.

Системи комп'ютерного тестування мають передбачати: аналіз тестів з метою оцінки якості, легкість створення та модифікації тестових запитань, збір, збереження та представлення статистичної інформації щодо процесу тестування у зручній формі, можливість одночасного тестування великої кількості користувачів, безпеку, захищеність та стабільність, зручний та ефективний імпорт/експорт тестів, контроль часу тестування та резервне копіювання задля збереження інформації перерваного тестування, керування обліковими записами користувачів з можливістю розподілу прав доступу.

Усі вище перелічені критерії зможуть здійснити проведення дистанційного моніторингу якості освіти серед учнів навчальних закладів без задіяння спеціальних експертів. Комп'ютерні системи контролю знань значно скоротять час на збір, обробку та аналіз інформації, порівняно з нині існуючою схемою проведення моніторингів у Луганській області, що проводить ЛОІППО.

Перевагами систем тестування є:

- автоматизація обробки результатів;
- забезпечення об'єктивності контролю знань;
- підвищення оперативності тестування;
- зменшення витрат на організацію та проведення моніторингового дослідження.

На сьогоднішній день комп'ютерні технології активно впроваджуються у сфері навчання не тільки у якості інтерактивного навчання, а також у якості засобів спрощення та вдосконалення існуючого освітнього процесу. Наступним етапом є створення єдиної універсальної моделі проведення дистанційного моніторингу серед учнів ЗЗСО. Нинішні можливості мережі Інтернет дозволяють створити таку модель, не витрачаючи на неї великі зусилля та кошти. Через те, що ЗЗСО територіально віддалені, створення локальних моделей систем тестування є неефективними, саме тому необхідно розглядати тільки ті системи, що організовані в мережі Інтернет.

Найпопулярнішими розробками у сфері комп'ютерних тестувань є Moodle, Google for Education, Open Test 2.0, INDIGO, ABBY FormReader, SQL-сервер, Statistical Package for the Social Sciences (SPSS).

Розглянемо систему управління навчанням Moodle. Це найбільш поширена в Україні Open Source (відкрита) система, що є безкоштовною. Moodle – це інструментальне середовище для розробки як окремих онлайн-курсів, так і освітніх веб-сайтів [9]. Цей програмний комплекс за своїми функціональними можливостями, простоті освоєння та зручності використання задовольняє більшості вимог, що пред'являються користувачами до систем електронного навчання. Moodle пропонує широкий спектр можливостей для повноцінної підтримки процесу навчання у дистанційному середовищі – різноманітні способи подання навчального матеріалу, перевірки знань та контролю успішності.

Перевагами системи Moodle є [10]:

- можливість налаштування шкали оцінок;
- можливість імпорту питань;
- можливість експорту таблиць з результатами;
- захист ключів до тестів та даних користувачів;
- налаштування розкладу часу проведення тестування.

Система дає можливість ознайомитись з результатами тестування безпосередньо після його проходження, автоматична обробка результатів тесту передбачає одержання балів за виконані завдання, а також представлення учаснику звіту з вказівкою правильних і неправильних відповідей, а також статистики по кількості правильно та неправильно виконаних завдань у тесті.

Основні переваги: повний набір необхідних функцій; відкритий вихідний код продукту (що дозволяє додати всі необхідні елементи); система Moodle універсальна в плані вимог (будь-яка ОС, встановлений модуль PHP і одна з СУБД); всі види тестів (включаючи написання есе).

Недоліки: система тестування є частиною великого програмного продукту; обслуговування надається за окрему плату.

Google for Education – звичний набір безкоштовних сервісів Google, що дають можливість створювати облікові записи для навчальних установ та освітніх організацій з розподілом прав доступу серед користувачів [11]. Ця хмарна технологія дозволяє організувати процес моніторингу, але не дає можливостей для автоматизації обробки результатів та не має конструктора для створення різного роду тестових завдань.

Система тестування Open Test 2.0 – це комп'ютерна система тестування знань створена для очного підсумкового контролю якості засвоєння теоретичного матеріалу, набутих знань і практичних навичок учнів у великих організаціях масштабу підприємства зі складною розподіленої структурою. Основною особливістю системи OpenTEST 2.0 є її спрямованість на забезпечення тестувань учнів з максимально суворою звітністю [12]. Областю застосування

можуть бути різноманітні підсумкові тестування, заліки, іспити, кваліфікаційні тести і будь-які інші види контролю знань учнів в яких головну роль відіграє максимально об'єктивна оцінка знань. При проектуванні модулів системи основна увага була приділена високій ергономічності системи, а також орієнтації на роботу з великим потоком користувачів [13]. Система дозволяє проводити тестування одночасно більше 1000 користувачів.

Система INDIGO - це професійний інструмент автоматизації процесу тестування і обробки результатів, який призначений для вирішення широкого спектра завдань: визначення рівня готовності учнів шкіл до ДПА та ЗНО, тестування та контроль знань студентів з різних дисциплін, визначення професійного рівня співробітників, проведення опитувань, автоматизація проведення вікторин та олімпіад [14]. Робота з INDIGO ділиться на дві частини: інтерфейс адміністратора і інтерфейс користувача (рис. 1.1).

Функціональні особливості:

– система тестування встановлюється на один комп'ютер-сервер за допомогою інсталяційного пакета;

– система може працювати як на ізольованому комп'ютері, так і в локальній мережі або через Інтернет;

– центр тестування можна розгорнути на Вашому комп'ютері або в хмарі на Інтернет-серверах;

– усі дані зберігаються централізовано в базі даних системи;

– одночасно можуть працювати скільки завгодно адміністраторів з різних комп'ютерів;

– користувачі працюють через web-браузери (Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Safari та інші). Є підтримка браузерів на мобільних пристроях.

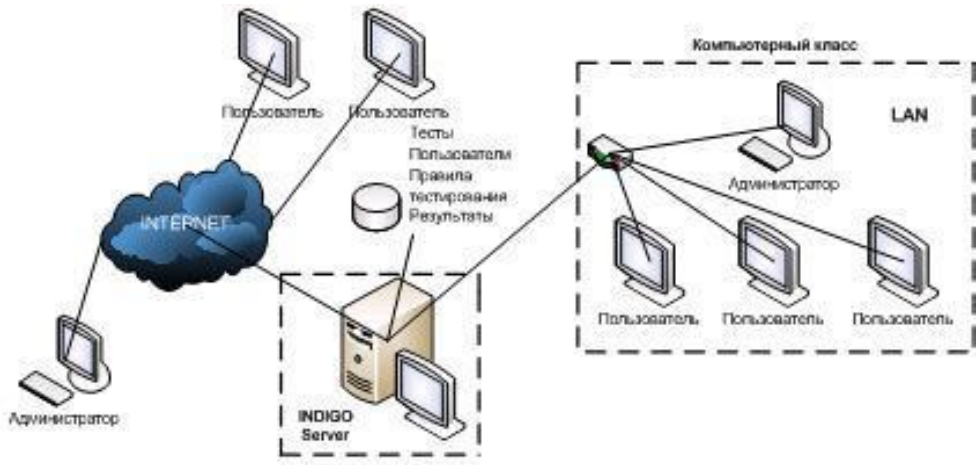


Рисунок 1.1 – Схема організації роботи системи INDIGO

Основні переваги: проста установка системи; доступний інтерфейс користувача; продукт

сумісний з усіма ОС сімейства Windows (XP / 2003 / Vista / 7/8); підтримка всіх поширених браузерів; централізоване зберігання даних і Web-інтерфейс користувачів; ієрархічна угруповання тестів і користувачів (правила тестування); широкі можливості конструктора тестів.

Недоліки: система є ліцензійною, відсутність поділу адміністратор-викладач (не завжди викладач має навички роботи з подібного роду системами); відносно великий обсяг споживаної пам'яті; високі вимоги до обладнання.

ABBYY FormReader - система введення форм, заповнених від руки або на принтері, із застосуванням технології ICR (Intelligent Character Recognition) [15]. ABBYY FormReader здатний:

– обробляти будь-які типи форм, відповідні простим вимогам набору тексту;

– розпізнавати текст, надрукований на машинці або принтері, для 172 мов і написаний друкованими літерами від руки для 90 мов, а також мітки (пункти) і штрих-коди;

– автоматично сортувати бланки, перевіряє комплектацію багатосторінкових форм

– не потребує втручання оператора на етапах сканування і розпізнавання;

– пропонує триступеневу технологію верифікації, що дозволяє підібрати оптимальний варіант перевірки для кожного типу даних;

– експортувати результати введення даних в файли форматів TXT, DBF, Microsoft Excel, CSV або бази даних, зберігати зображення у вигляді pdf-файлів.

Перевагами даного продукту є:

– ABBYY FormReader має зрозумілу структуру і зручний інтерфейс;

– програма забезпечена докладним електронним довідником і керівництвом користувача;

– для створення нового пакета і нового шаблону форми використовуються редактори, крок за кроком підказують користувачеві вірну послідовність дій;

– вивчення програми займає від кількох годин до кількох днів і не вимагає спеціальних знань і навичок;

– рівень коректного розпізнавання рукописних символів досягає 98%;

– якість даних при автоматизованому введенні форм виявляється на кілька порядків вище, ніж при ручному введенні;

– зводиться до нуля вплив людського фактора, абсолютно виключаються "позиційні" помилки, коли при ручному введенні оператор заносить, наприклад, ім'я в поле "по батькові";

– автоматичний контроль результатів розпізнавання за допомогою перехресних перевірок, правил контролю сум, звірок по словниках і баз даних;

– існує можливість створити власні правила перевірки, в тому числі з підключенням списків і словників, і використовувати їх для контролю якості даних, що вводяться.

MS SQL Server 2016 - система керування базами даних, розроблена корпорацією Microsoft. MS SQL Server - це платформа для вирішення критично важливих завдань в масштабі

підприємства, що володіє високою доступністю, підвищеною продуктивністю і безпекою [16]. Рішення являє собою добре масштабований, повністю реляційний, швидкодіючий сервер, здатний обробляти великі обсяги даних для клієнт-серверних додатків. Рекордна продуктивність MS SQL Server забезпечується новими технологіями роботи з пам'яттю, що допомагає підприємствам прискорити свій бізнес і реалізувати нові сценарії роботи. Крім того, SQL Server дозволяє використовувати нові гібридні хмарні рішення і користуватися новими перевагами хмарних обчислень. Розширені функції безпеки, у поєднанні з вбудованими, зручними для використання інструментами і керованим доступом до даних, дозволяють організації дотримуватися вимог строгих політик відповідності нормам. Перевагами платформи є:

– рекордна висока продуктивність;

– швидке отримання результатів аналізу як у локальному, так і у «хмарному» середовищі;

– розширені функції безпеки.

Можливості платформи представлені:

– єдиним пред’явленням даних: масштабна платформа для інтеграції корпоративної інформації включає в себе інструменти вилучення, перетворення і завантаження даних (ETL); MS SQL Server забезпечує доступність і можливість інтеграції будь-яких гетерогенних джерел даних, а також отримання даних з будь-якого стороннього джерела, в тому числі з SQL Server, Oracle, Teradata, DB2, ODBC, OLE DB, текстових файлів, Microsoft Entity Framework і хмарних систем. Платформа дозволяє інтегрувати дані з бізнес-процесів в Microsoft BizTalk Server для SAP, системи ERP і CRM, веб-служби та додатки для мейнфрейма, і фіксувати дані, що надходять в реальному часі від датчиків та інші складні потоки подій. ETL-інструменти MS SQL Server гарантують кращу в своєму класі швидкодію, а використання високопараллельних пакетів і оптимізованих пулів потоків дозволяють підвищити продуктивність обробки даних і забезпечують відмовостійкість критично важливих систем;

– бізнес-аналітикою: служби SQL Server допомагають побудувати комплексне рішення для аналізу даних корпоративного рівня. Користувачі отримують можливість попереджувального аналізу та інтерактивного перегляду агрегованих даних з різних точок зору; спростити процес створення складних рішень можна за допомогою узгодженої семантичної моделі бізнес-аналітики; платформа дозволяє створювати різні аналітичні рішення за допомогою зручних багатофункціональних засобів моделювання, бізнес-логіки і «тонкого» налаштування параметрів безпеки; рішення має збалансовану продуктивність та масштабованість, дозволяє використовувати обчислення в оперативній пам'яті та наскрізні запити до сервера й обробляти величезні обсяги даних з високою швидкістю;

– звітністю: MS SQL Server має широкі можливості створення оперативних звітів з підтримкою високоякісного друку, дозволяючи використовувати для перегляду та візуалізації веб-

браузери, а також інші спеціалізовані засоби; рішення дозволяє спростити механізми взаємодії і спільного доступу до даних, самостійно створювати звіти в різних форматах, візуалізувати дані. MS SQL Server гарантує високий рівень керованості та масштабованості звітів як локально, так і в хмарі, і дозволяє приймати рішення в режимі реального часу.

Statistical Package for the Social Sciences (SPSS) є найпоширенішою програмою для обробки статистичної інформації. Основною перевагою програмного комплексу SPSS, як одного з найбільш істотних досягнень в області комп'ютеризованого аналізу даних, є найширший охоплення існуючих статистичних методів, який вдало поєднується з великою кількістю зручних засобів візуалізації результатів обробки [17].

Можливості:

- введення і зберігання даних;
- можливість використання змінних різних типів;
- частотність ознак, таблиці, графіки, таблиці спряженості, діаграми;
- первинна описова статистика;
- маркетингові дослідження;
- аналіз даних маркетингових досліджень.

За всіма параметрами SPSS є складним і потужним статистичним пакетом. За допомогою пакета SPSS можна проводити практично будь-який аналіз даних, а останні версії програми знаходять застосування в самих різних наукових областях, в тому числі в педагогічних науках. За допомогою пакету, використовуючи таблиці, прості меню і діалогові вікна, можна виконувати, по-перше, аналіз величезних файлів даних з тисячами змінних, і, по-друге, робити все це без рядкової записи команд в мові програмування. За допомогою програми SPSS можна: керувати даними; впорядковувати дані; перетворювати дані, створювати нові змінні; аналізувати дані. Етапи аналітичного процесу, що реалізується в SPSS: планування; збір даних; забезпечення доступу до даних; підготовка даних до аналізу; виконання аналізу; формування звітів; уявлення і розповсюдження результатів. Поряд з використанням свого власного типу даних, пакет SPSS, може зчитувати дані практично з будь-яких типів файлів і використовувати їх для створення звітів у формі таблиць, графіків і діаграм, а також обчислювати описові статистики, робити складний статистичний аналіз і моделювання. Пакет має модульну структуру. Модулі пакету є інтегрованою сукупністю програмних продуктів, що забезпечують комплексне дослідження - від планування до управління даними, виконання аналізу та представлення результатів.

Дослідивши розробки у сфері комп’ютерного тестування, можна представити модель обміну інформацією (рис. 1.2), що можлива при впровадженні однієї з існуючих систем.

Рисунок 1.2 – Модель обміну інформацією

1.4 Аналіз вимог до розроблюваної комп'ютерної системи моніторингу якості освіти

Одним з напрямків роботи ЛОІППО є проведення моніторингових досліджень якості освіти. На сьогоднішній день робота проводиться в основному в паперовому вигляді з подальшою обробкою шляхом використання комп'ютерних технологій. Завдання в тестовій формі, які розробляють методисти інституту друкуються на папері. Це призводить до значних витрат часу на організацію проведення моніторингового дослідження та на подальший аналіз отриманої інформації. Для оптимізації процесу підготовки та проведення необхідно перенести всі дії в електронний вигляд, але з можливістю проведення тестування на місцях у паперовому вигляді через недостатню забезпеченість ЗЗСО. Пропонується створення комп'ютерної системи, яка буде зберігати завдання в тестовій формі у загальній базі даних. Комп'ютерна обробка бланків відповідей учнів після моніторингових досліджень, дозволяє прискорити процес обробки результатів та забезпечити конфіденційність зберігання отриманої інформації.

Співпрацюючи зі спеціалістами ЛОІППО, були виділені основні напрями розробки, а саме:

- створення єдиної регіональної бази даних результатів освітньої діяльності, що містить інформацію про всі оціночні заходи, що проводяться в ЗЗСО;
- забезпечення можливості легкої інтеграції системи з існуючими та проектованими системами;
- створення підсистеми формування звітів, що дозволяють оперативно отримувати

необхідну інформацію про результати освітньої діяльності в розрізі класу, педагога, предмета, освітнього закладу;

- створення набору інтерфейсів, що реалізують функції АРМ різних користувачів.

У перспективі ІС забезпечить повний інструментарій для моніторингу якості освіти у регіоні. При проектуванні підсистем, що забезпечують перетворення даних між вузлами ІС, можливо використання системи з деякими доробками, пов'язаними з адаптацією системи під особливості регіону. У зв'язку з цим перед архітекторами і розробниками ІС стоїть нетривіальне завдання по створенню універсальної системи для оцінки якості освіти, що врахує технічне оснащення ЗЗСО. Таким чином, затребуваність кінцевого продукту буде визначатися деяким мінімальним набором характеристик, заснованим на оцінці стану технічної бази ЗЗСО.

Підводячи підсумок всьому вищесказаному можна виділити найзагальніші характеристики розроблюваної системи, які в свою чергу визначають набір технологій, що необхідно використовувати, та початкове уявлення про архітектуру:

– система повинна мати централізоване сховище даних, яке містить загальні довідники, інформацію про всі проведені моніторингові дослідження, що проводяться в ЗЗСО регіону і т.і.;

– кожний ЗЗСО повинен мати власну базу даних, що містить специфічну інформацію про його навчальну діяльність;

– повинна бути реалізована гнучка система реплікації між центральним сховищем і локальними базами даних ЗЗСО, при цьому дані повинні передаватися не тільки в режимі online (через web), але і в режимі offline (всі оновлення переносяться за допомогою зашифрованих xml-файлів, які в свою чергу можна завантажити в систему з будь-якого носія інформації в будь-який час, що передбачає тимчасову відсутність доступу до мережі Інтернет);

– системні вимоги, що пред'являються клієнтською частиною, повинні бути мінімізовані, що має забезпечити прийнятну швидкість роботи навіть на досить слабких комп'ютерах;

– у рамках одного ЗЗСО можуть функціонувати кілька клієнтських програм, розташованих на різних робочих станціях локальної мережі, при цьому результати роботи кожного клієнта повинні зберігатися в загальну базу, розташовану в тій же локальній мережі, яка також є сховищем власних даних ЗЗСО;

– клієнтська частина програми повинна бути максимально простою в установці та розгортанні, а її графічний інтерфейс має бути інтуїтивно зрозумілим навіть для користувача, що володіє тільки початковими навичками роботи в середовищі Windows.

1.5 Постановка задачі та обґрунтування методики досліджень

Практика застосування різних технологій та систем автоматизації контролю навчальних досягнень учнів, відкриває можливості для створення ЄАРІС МЯО - системи збору, обробки та інтерпретації інформації про результати освітньої діяльності. Очевидною є необхідність застосування клієнт-серверної технології, та використання web-інтерфейсів для забезпечення роботи системи. Основними напрямками розробки є:

– створення єдиної регіональної бази даних результатів освітньої діяльності, що містить інформацію про всі оціночні заходи, які проводилися в ЗЗСО;

– створення підсистеми оперативної обробки даних моніторингу для формування звітів, що показують зведені дані про результати освітньої діяльності ЗЗСО, РЦМ;

– створення набору інтерфейсів, що реалізують функції АРМ різних користувачів.

Результатом розробки має стати система, яка в перспективі забезпечить повний інструментарій для проведення моніторингу якості освіти ЗЗСО регіону.

1.6 Висновки до розділу 1

1. Для ефективного проектування комп’ютерної системи моніторингу якості освіти проведено:
   - огляд існуючих підходів до моніторингу якості освіти;
   - детальний аналіз призначення вирішуваних завдань і характеристик програмних та інструментальних засобів контролю знань.
2. Вивчена методика проведення процедур оцінки якості освіти.
3. У результаті аналізу вимог до розроблюваної системі, виділені основні її характеристики, які визначають набір використовуваних технологій та дозволяють сформувати уявлення про архітектуру системи.
4. Визначено основні напрямки досліджень дисертаційної роботи, метою якої є підвищення оперативності моніторингу якості освіти в ЗЗСО.
5. Поставлено завдання та обгрунтована методика досліджень.

РОЗДІЛ 2 РОЗРОБКА КОНЦЕПЦІЇ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ОСВІТИ

2.1 Методологічні аспекти проектування системи

Методи проектування ІС відображають різні способи їх створення, що підтримуються відповідними засобами проектування.

Усі методи проектування ІС класифікують:

- за виконанням технологічного (виробничого) процесу проектування - методи аналізу, синтезу, декомпозищї, формалізації та моделювання;

- за ступенем автоматизації проектних робіт (оригінальне, типове й автоматизоване проектування);

- за організацією процесів проектування — різні організаційні методи.

У науковому сенсі процес проектування є важливим об'єктом дослідження. Серед методів наукових досліджень широко використовують аналіз і синтез, особливо на передпроектній стадії, для вивчення ІС та системи управління підприємства, пізнання сутності функціональних задач і структури управління.

У процесі проектування ІС на всіх стадіях та етапах застосовується метод декомпозиції за двома напрямами:

- декомпозиція даних, тобто розчленування їх на прості компоненти з виявленням взаємозв'язків між ними (вхідні й вихідні дані, а також дані, що зберігаються в БД);

- декомпозиція процесів (оскільки процес є логічно завершеною послідовністю дій, яка виконується у предметній сфері з групою даних, його декомпозиція передбачає підбиття підсумків, вид контролю, модифікацію, генерацію звітів). Декомпозиція процесів дає змогу розробити профіль транзакції — графічне подання всіх процесів оброблення певної сукупності даних (наприклад, вхідного, або головного, файла). Транзакція розробляється під час проектування системи.

Застосування методів формалізації та моделювання пов'язане з використанням економіко-математичних моделей, а також обчислювальних алгоритмів.

Оригінальне (індивідуальне) проектування передбачає, що всі види проектних робіт орієнтовані на створення індивідуальних проектів для конкретних підприємств з урахуванням їхніх специфічних особливостей. Проте в його процесі теж використовують стандартні засоби ОС, процедури типових процесів обробки даних, окремі інструментальні засоби проектування.

Типове проектування залежно від рівня декомпозиції проектованої ІС на окремі компоненти передбачає застосування елементного, підсистемного, об'єктного методів проектування.

За **елементного методу проектування** декомпозиція здійснюється на рівні задач й окремих проектних рішень на основі інформаційного, програмного, математичного і технічного забезпечення. Для кожного компонента (елемента) створюються ТПР, наприклад ТПР-задача, ТПР-техніка, ТПР-персонал.

Під час застосування **підсистемного методу проектування** декомпозиція виконується на рівні підсистем, що виступають типовими елементами. При цьому досягаються функціональна повнота підсистеми, мінімізація зовнішніх інформаційних зв'язків, параметричне налаштування розв'язання задач підсистеми, альтернативність схем у межах значень вхідних параметрів.

Для кожної підсистеми створюється проектне рішення. Засобами проектування є ППП.

Об'єктне проектування передбачає створення типового проекту ІС для узагальненого об'єкта, виділеного з групи об'єктів як еталон. При цьому група однотипних об'єктів може бути невеликою (наприклад, для годинникових заводів).

При **типовому проектуванні** застосовуються: стандартні засоби ОС; типові компоненти — ТПР, ППП, типові АСУ (наприклад, АСУ «СИГМА», АСУ «ЛЬВІВ» та ін.); конкретні інструментальні засоби.

Автоматизоване проектування — це створення проектів ІС на основі САПР, що ґрунтуються на глобальній інформаційній моделі ОУ (модельне проектування). Модель має містити формалізований опис інформаційних компонентів та відношень між ними, включаючи їхні зв'язки й алгоритмічну взаємодію.

При такому проектуванні використовують стандартні засоби ОС, САПР, взаємозв'язаний комплекс інструментальних засобів проектування; засоби модернізації функціонуючої ІС.

Організаційні методи проектування. Ці методи охоплюють питання, які стосуються послідовності створення проекту, добору спеціалістів на кожному етапі, забезпечення якісного документування проекту, контролю проектування, організації колективів розробників ІС, інформування учасників проектування про стан розробки проекту, забезпечення виконання програмних та інформаційних інтерфейсів.

До цієї групи належить метод «зверху вниз» (спадне проектування), де формалізація процесу проектування здійснюється у вигляді графа-дерева, а проектування можна розпочинати з будь-якої задачі та вести паралельно для кількох.

Проектну документацію можна створювати одночасно з прийняттям проектних рішень (наприклад, із розробленням програм), при цьому зберігається їх повна відповідність.

Модульний метод проектування пов'язаний зі створенням програмного та інформаційного забезпечення з множини відносно незалежних модулів. Модулі мають інформаційні взаємозв'язки, які визначаються у такий спосіб, що кожний модуль не має інформації про внутрішній зміст інших модулів, крім тієї, яка міститься у специфікаціях інтерфейсу. Цей метод дає змогу звести проектування до оптимізувального синтезу функціонально незалежних окремих частин (модулів), які разом виконують задані функції системи з потрібною ефективністю.

Оптимальний модульний синтез має такі переваги:

- спрощуються розроблення і налагодження ПЗ;

- спрощується подальша модифікація системи (модульні програми можна поліпшити простою заміною окремих модулів, які функціонально є еквівалентними, але мають кращі системні характеристики);

- поліпшуються керуючі програми;

- забезпечується можливість застосування технічних засобів;

- поліпшується використання можливостей програмістів.

Однак застосування оптимального модульного синтезу пов'язане зі зміною традиційної методології проектування, збільшенням трудомісткості аналізу зібраного матеріалу на етапі передпроектного обстеження, з появою додаткової роботи та аналізу великої кількості альтернатив, розбиттям існуючої системи на підсистеми (задачі). Зростає трудомісткість розроблення інтерфейсу та погодження модулів.

Розбиття програмного й інформаційного забезпечення ІС на окремі модулі та їх подальше спряження є найважчим і слабко формалізованим процесом, тому що розподіл та спряження пов'язані з плануванням й організацією роботи програмістів та аналітиків (постачальників задач).

Структурний метод передбачає наявність програм, що динамічно налагоджуються на структури масивів інформаційного фонду системи. При цьому опис масивів слід формалізувати, а їх збереження та підтримка в адекватному стані мають бути організовані в системі інформаційного фонду. Цей метод використовують під час створення БД, він спрямований на забезпечення логічної та фізичної незалежності даних.

Метод «на основі математичної моделі» передбачає для розв'язання задачі вибір та розробку економіко-математичної моделі, що включає створення алгоритму розв'язання та складання прикладної програми.

Метод неперервності розвитку системи полягає в тому, що після створення ІС у процесі її функціонування з'являються нові, змінюються діючі задачі управління, виникає необхідність внести зміни у систему. Цей процес часто є інерційнішим, ніж процес ручного оброблення даних. Тому під час проектування ІС у логіку прикладних програм мають бути закладені як

організація даних у вторинній пам'яті ЕОМ, так і методи доступу до них, що забезпечує фізичну незалежність задач та дає змогу автоматизувати внесення змін.

Методи проектування ІС сприяють підвищенню якості створюваних проектів, зростанню продуктивності праці всіх спеціалістів-розробників проекту, зниженню вартісних і трудових витрат на проектування, скороченню термінів виконання проектних робіт, спрощенню впровадження, супроводу та модернізації функціонуючої ІС.

2.2 Узагальнена процесна модель ЗЗСО

У [18] запропоновано моделі ЗЗСО, що відображають основні принципи TQM-ідеології та є базою для організації ефективного управління якістю освіти у ЗЗСО. В узагальненій процесній моделі (рис. 2.1) враховано те, що освітній процес є основним, але його здійснення можливе тільки за певних умов, які створюються допоміжним процесом. З точки зору кібернетичного підходу, освітній і допоміжний процеси є об‘єктами управління, а процес управління – суб‘єктом управління. При цьому зворотний зв‘язок між об‘єктами управління та суб‘єктом управління забезпечується моніторингом.

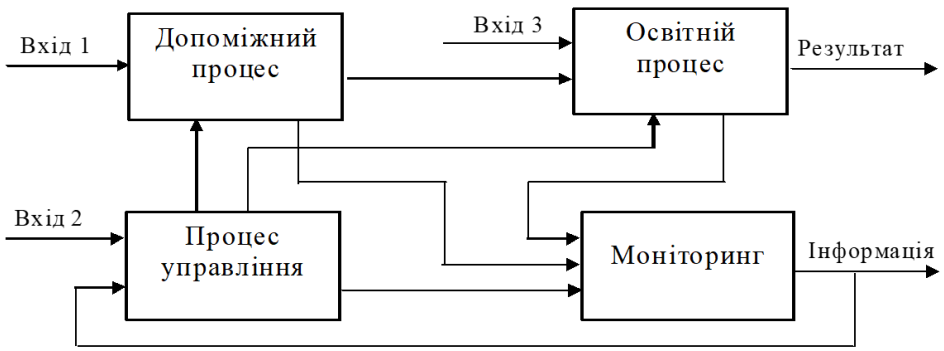


Рисунок 2.1 - Узагальнена процесна модель ЗЗСО

Особливість такого зворотного зв‘язку полягає в тому, що здійснюється моніторинг якості не тільки освітнього і допоміжного процесів, але й якості самого процесу управління .

Узагальнена ресурсна модель ЗЗСО відображає ресурси кожного з процесів та їх узгодженість. Особливість цієї моделі полягає у тому, що ресурси ЗЗСО мають три складові. Перша – це локальні ресурси, які використовуються тільки даним процесом. Друга складова є частиною вхідних ресурсів. Третя складова це спільні ресурси, які використовуються всіма процесами.

Згідно з узагальненою РПР-моделлю ЗЗСО (рис. 2.2), побудованою за схемою «Ресурси-Процеси-Результати», моніторинг якості має здійснюватися для множини ресурсів, множини процесів і множини результатів.

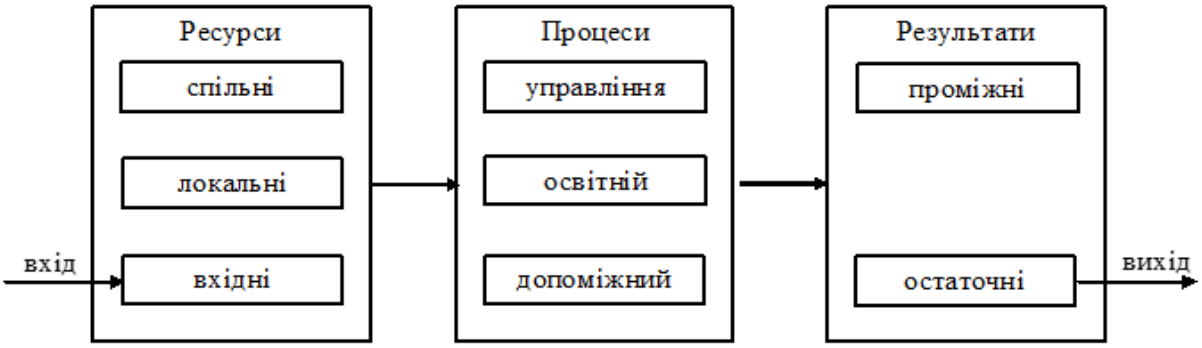


Рисунок 2.2 - Узагальнена РПР-модель ЗЗСО

З принципу ієрархічності показників якості випливає, що наочною моделлю якості об‘єкта є дерево якостей (рис. 2.3).

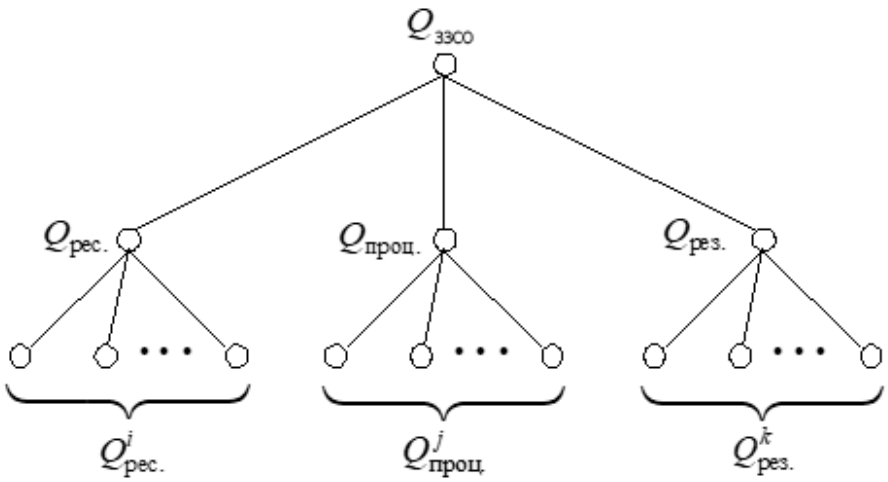


Рисунок 2.3 - Узагальнене дерево якості ЗЗСО за РПР-моделлю

Якість ЗЗСО $Q_{\text{ззсо}}$ має три складові:

– якість ресурсів $Q_{\text{рес.}}$;

– якість процесів $Q_{\text{проц.}}$;

– якість результатів $Q_{\text{рез.}}$,

які, у свою чергу, мають певні складові.

Якість може розглядатися як сукупність складових (2.1).

$$Q_{\text{ззсо}} = \langle Q_{\text{рес.}}; Q_{\text{проц.}}; Q_{\text{рез.}} \rangle, \tag{2.1}$$

Представлення якості ЗЗСО у вигляді (2.1) надає інформацію окремо про стан ресурсів, якість процесів та рівень результатів, що забезпечує можливість спрямувати зусилля ЗЗСО на

покращення конкретної складової якості.

2.3 Проектування поведінки системи

Згідно сучасної методології, процес створення ІС являє собою процес побудови і послідовного перетворення ряду узгоджених моделей на всіх етапах життєвого циклу (ЖЦ) ІС. На кожному етапі ЖЦ створюються специфічні для нього моделі - організації, вимоги до ІС, проекту ІС, вимог до додатків і т.і.

Для успішної реалізації проекту об'єкт автоматизації повинен бути насамперед адекватно описаний, повинні бути побудовані повні й несуперечливі функціональні та інформаційні моделі системи. Повна модель повинна відображати не тільки бізнес-процеси, а й алгоритми роботи окремих інформаційних функцій, екранні інтерфейси, структури даних і т.і. Подібна модель є достатньою не тільки для формування технічного завдання, але і для створення ескізного проекту, тобто розробки всіх необхідних попередніх рішень щодо загальносистемних та організаційних питань.

Після визначення функцій системи і розробки інформаційної основи наступним кроком є проектування поведінки системи. Поведінкова модель системи показує, за рахунок чого досягається необхідна функціональність і які дані використовуються для її забезпечення. Таким чином, поведінкова модель безпосередньо базується на функціональній та інформаційній моделях системи.

Поведінкові моделі, як правило, будують для функцій (процесів), які відображаються на останніх рівнях діаграм декомпозиції IDEF0 та DFD. Для DFD поведінкова модель у вигляді діаграми або схеми є більш вдалим рішенням, ніж опис елементарних процесів у вигляді мініспеціфікації.

Найбільш відомими і популярними способами та методологіями розробки поведінкових моделей є:

- блок-схеми алгоритмів;
- EPC-діаграми;
- методологія BPMN.

Блок-схеми алгоритмів і BPMN-діаграми найкраще підходять для моделювання традиційних систем, заснованих на принципах структурного підходу та характеризуються чітким поділом функцій на елементарні кроки (процедури, оператори, дії і т.д.), що виконуються в певній послідовності (за алгоритмом). EPC-діаграми та BPMN-діаграми є хорошим засобом моделювання подієво-орієнтованих систем, у яких виконання дій або їх набору залежить від подій, що відбуваються в системі. Крім цього, зазначені способи і методології можуть використовуватися не тільки в цілях поведінкового, але і функціонального моделювання. Наявність у них елементів, що

дозволяють відображати умови виконання тих чи інших дій (логічних символів), дозволяє краще зрозуміти логіку і послідовність виконання функцій системи.

Object Management Group – консорціум (некомерційне об’єднання), що займається розробкою та просуванням об'єктно-орієнтованих технологій і стандартів. З консорціумом співпрацює близько 800 організацій — найбільші виробники ПЗ. 1997р. OMG затвердила UML як стандартну мову об’єктно-орієнтованого програмування. Версія мови 1.1 була прийнята на озброєння практично всіма найбільшими компаніями — виробниками ПЗ (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase і ін.). Крім того, практично всі світові виробники CASE-засобів підтримали Rational Software та заявили про реалізацію підтримки UML у своїх продуктах.

Проектування враховує вимоги, які протирічать одна одній. Продуктом проектування є моделі, які дозволяють нам:

– зрозуміти структуру майбутньої системи (класифікувати побажання);
– збалансувати вимоги (усвідомити обмеження, домовитись щодо пріорітетів);
– намітити схему реалізації (знайти компроміси).

Моделювання (проектування) ІС включає наступні процеси і одночасно корисні результати:

– візуалізація;
– специфікація, тобто точне визначення вимог до системи та її поведінки;
– конструювання, що включає:
  1) пряме проектування з генерацією коду;
  2) зворотне проектування;
– документування.

Послідовність дій при розробці ІС полягає у побудові наступних моделей (рис. 2.4).

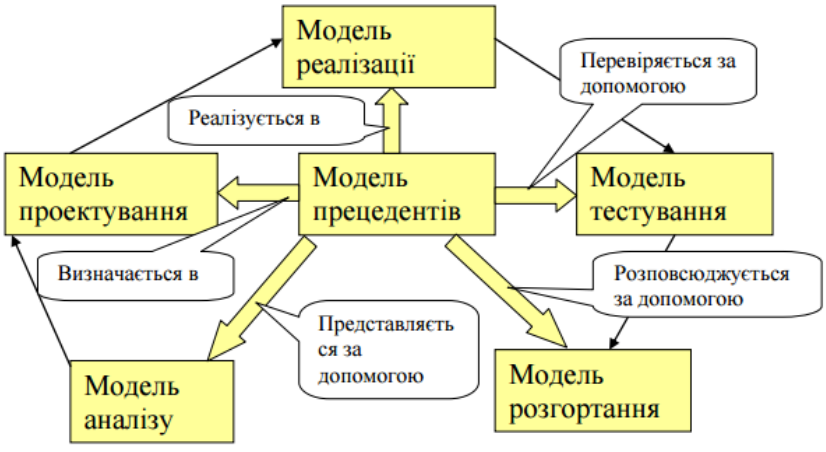


Рисунок 2.4 - Шість основних моделей уніфікованого процесу

Згідно рисунку, модель прецедентів суттєво впливає на інші п’ять моделей. Модель

прецедентів є основою розробки. Вона містить функціональні вимоги до системи, що моделюється, це вимоги до системи на мові замовника.

Модель аналізу допомагає розробникам уточнити та структурувати функціональні вимоги, виявлені за допомогою моделі прецедентів, це вимоги до системи на мові розробника.

Модель проектування описує фізичну реалізацію прецедентів.

Модель реалізації визначає, як елементи моделі проектування будуть організовані у програмні компоненти, наприклад файли з вхідним кодом, динамічно завантажувальні бібліотеки (dll) і т.і.

Модель тестування містить опис виконання системного та інтеграційного тестування компонентів моделі реалізації, включаючи сценарії тестування, які, як правило, будуються на основі прецедентів (тестування «чорного ящика»).

Модель розгортання визначає фізичну організацію системи в термінах вузлів обробки.

2.4 Діаграми варіантів використання

Діаграми варіантів використання показують взаємодії між варіантами використання і діючими особами, відображаючи функціональні вимоги до системи з точки зору користувача.

Мета побудови - документування функціональних вимог в загальному вигляді.

Варіант використання - послідовність дій (транзакцій), виконуваних системою у відповідь на подію, що ініціюється деяким зовнішнім об'єктом (дійовою особою). Варіант використання описує типову взаємодію між користувачем та системою і відображає уявлення про поведінку системи з точки зору користувача.

Найпростіший випадок: варіант використання визначається в процесі обговорення з користувачем тих функцій, які він хотів би реалізувати, або цілей, які він переслідує по відношенню до розроблюваної системи.

Переваги моделі варіантів використання:

- визначає користувачів і кордони системи;
- визначає системний інтерфейс;
- зручна для спілкування користувачів з розробниками;
- використовується для написання тестів; є основою для написання документації користувача;
- добре вписується в будь-які методи проектування.

2.5 Моделі клієнт-серверної взаємодії

Для кожної розподіленої компонентної програмної системи важлива схема її внутрішньої організації. Найбільш поширена модель – **Клієнт-сервер**. Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатка на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатка можуть виконуватися на різних комп'ютерах, виконуючи серверні або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків та мережі в цілому.

Архітектура «клієнт-сервер» визначає загальні принципи організації взаємодії в мережі, де є сервери, вузли-постачальники деяких специфічних функцій (сервісів) і клієнти, споживачі цих функцій. Практичні реалізації такої архітектури називаються клієнт-серверними технологіями. Кожна технологія визначає власні або використовує наявні правила взаємодії між клієнтом та сервером, які називаються протоколом обміну (протоколом взаємодії).

Розташування компонентів на стороні клієнта або сервера визначає наступні основні моделі їх взаємодії в рамках дволанкової архітектури:

– сервер терміналів - розподілене представлення даних;
– файл-сервер - доступ до віддаленої базі даних і файлових ресурсів;
– сервер БД - віддалене уявлення даних;
– сервер додатків - віддалений додаток.

Перераховані моделі представлені на рисунку 2.5.

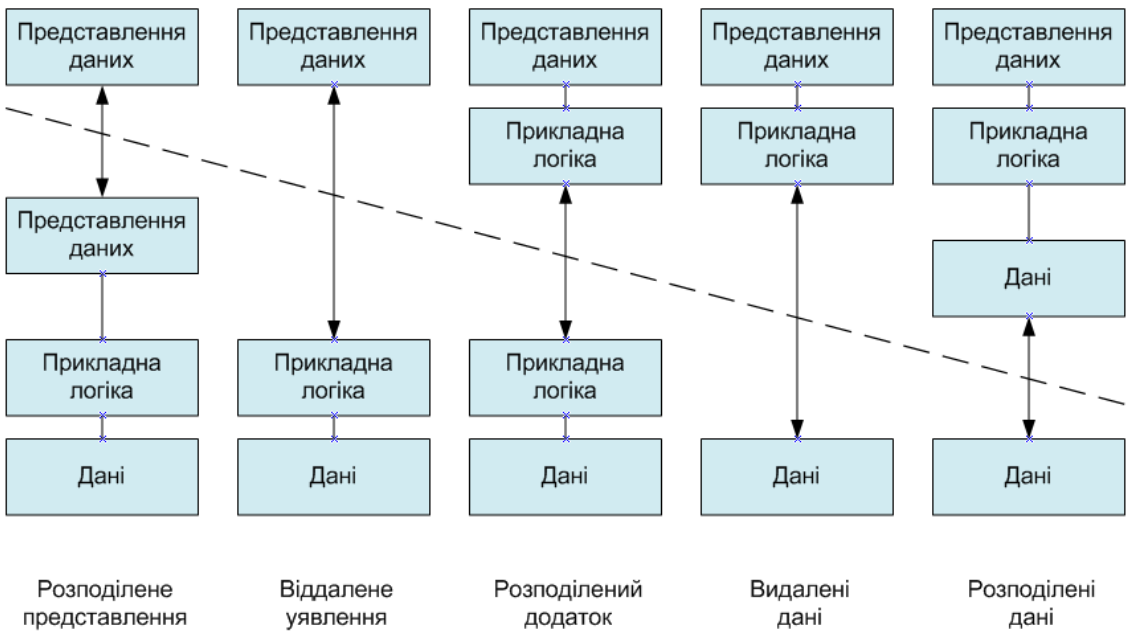


Рисунок 2.5 - Моделі клієнт-серверної взаємодії

Історично першою з'явилася модель розподіленого представлення даних (модель сервер терміналів). Вона реалізовувалася на універсальній ЕОМ (мейнфрейми), що виступала в ролі сервера, з підключеними до неї алфавітно-цифровими терміналами.

З появою ПК і локальних мереж, була реалізована модель файлового сервера, який представляв доступ файлових ресурсів, у т.ч і до віддаленої БД. У цьому випадку виділений вузол мережі є файловим сервером, на якому розміщені файли БД. На клієнтах виконуються додатки, у яких поєднані компонент уявлення та прикладний компонент (СУБД і прикладна программа), що використовують підключену віддалену базу як локальний файл. Протоколи обміну при цьому представляють собою набір низькорівневих викликів операцій файлової системи. Така модель показала свою неефективність з огляду на те, що при активній роботі з таблицями БД виникає велике навантаження на мережу. Частковим вирішенням цієї проблеми є підтримка тиражування (реплікації) таблиць і запитів. У цьому випадку, наприклад при зміні даних, оновлюється не вся таблиця, а тільки модифікована її частина.

З появою спеціалізованих СУБД з'явилася можливість реалізації іншої моделі доступу до віддаленої БД - моделі сервера баз даних. У цьому випадку ядро СУБД функціонує на сервері, прикладна програма на клієнті, а протокол обміну забезпечується за допомогою мови SQL. Такий підхід у порівнянні з файловим сервером веде до зменшення завантаження мережі й уніфікації інтерфейсу «клієнт-сервер». Однак, мережевий трафік залишається досить високим, крім того, як і раніше неможливо задовільне адміністрування додатків, оскільки в одній програмі поєднуються різні функції.

З розробкою і впровадженням на рівні серверів БД механізму збережених процедур з'явилася концепція активного сервера БД. У цьому випадку частина функцій прикладного компонента реалізовані у вигляді збережених процедур, що виконуються на стороні сервера. Інша прикладна логіка виконується на стороні клієнта. Протокол взаємодії - відповідний діалект мови SQL.

Переваги такого підходу:

– можливе централізоване адміністрування прикладних функцій;

– зниження вартості володіння системою (TOC, total cost of ownership) за рахунок оренди сервера, а не його покупки;

– значне зниження мережевого трафіку (тому що передаються не SQL-запити, а виклики збережених процедур).

Основний недолік - обмеженість можливостей розробки збережених процедур у порівнянні з мовами високого рівня.

Реалізація прикладного компонента на стороні сервера надає наступну модель - сервер додатків. Перенесення функцій прикладного компонента на сервер знижує вимоги до конфігурації клієнтів та спрощує адміністрування, але представляє підвищені вимоги до продуктивності,

безпеки і надійності сервера.

Нині намічається тенденція повернення до того, з чого починалася клієнт-серверна архітектура - до централізації обчислень на основі моделі термінал-сервера. У сучасній реінкарнації термінали відрізняються від своїх алфавітно-цифрових предків тим, що маючи мінімум програмних і апаратних засобів, вони представляють мультимедійні можливості. Роботу терміналів забезпечує високопродуктивний сервер, куди винесено все, аж до віртуальних драйверів пристроїв, включаючи драйвери відеопідсистеми.

При проектуванні архітектури веб-орієнтованої ІС основним критерієм виділення підсистем та їх функціональності є їх просторова розподіленість. Підсистеми взаємодіють, обмінюючись повідомленнями за допомогою граничних (інтерфейсних) об'єктів особливого типу. У свою чергу, паралельні завдання, що формують підсистему, представляються як активні об'єкти, які можуть взаємодіяти один з одним у межах однієї підсистеми або, долаючи кордони, у різних підсистемах різними способами: асинхронно, синхронно та за допомогою групових комунікацій.

2.6 Класи клієнт-серверних додатків

У рамках загальної структури додатків типу клієнт-сервер, виконувана робота може бути розділена між клієнтом та сервером по-різному. Точна частка виконуваних операцій і обсяг переданих по мережі даних залежать від природи інформації, що міститься в БД, підтримуваних типів додатків, доступності обладнання, яке може працювати спільно, а також від характеру використання даних. Схеми деяких основних класів додатків наведено на рисунках 2.6, 2.7, 2.8, 2.9. Можливі також інші варіанти розподілу задач між сервером і клієнтом. На рисунках зображено чотири класи додатків з різними варіантами розподілу задач між сервером і клієнтом.

2.6.1 Обробка даних на базі хоста

Дана схема не є справжнім додатком клієнт-сервер, а відноситься до традиційного оточення мейнфрейма, коли вся або майже вся обробка даних здійснюється на головній обчислювальної машині. Найчастіше у подібному обчислювальному середовищі інтерфейс користувача має вигляд примітивного терміналу.

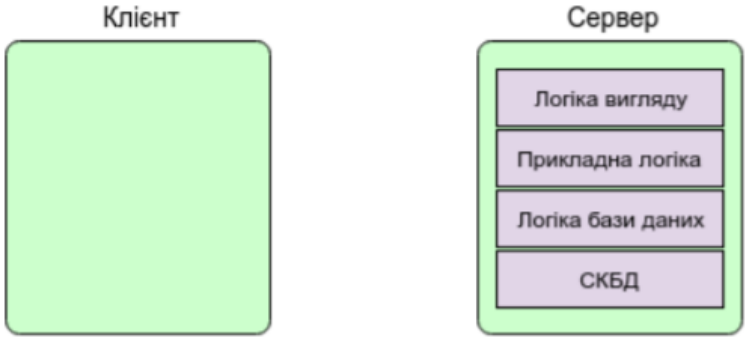


Рисунок 2.6 – Обробка даних на базі хоста

Але навіть якщо користувач використовує персональний комп'ютер, його роль у випадку обробки даних на базі хоста обмежується емуляцією терміналу. Така схема є доцільною, коли основна обчислювальна потужність концентрується на стороні сервера. Як правило, клієнти в цьому випадку мають потужність значно меншу, ніж потрібна для виконання задач, для яких створений сервер.

2.6.2 Обробка даних на базі сервера

Найпростішим класом конфігурації клієнт-сервер є схема, у якій клієнт відповідає лише за надання графічного інтерфейсу користувача, тоді як практично вся обробка даних здійснюється на сервері. Це дозволяє знизити навантаження на серверну частину, особливо у випадках, коли логіка вигляду складна і потребує багато системних ресурсів.

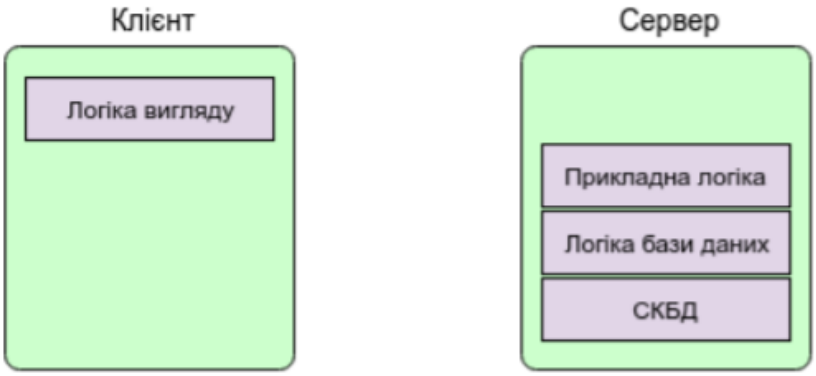


Рисунок 2.7 – Обробка даних на базі сервера

2.6.3 Обробка даних на базі клієнта

Дана схема демонструє зворотній підхід: практично вся обробка даних здійснюється на стороні клієнта, за винятком процедур перевірки цілісності даних та іншої логіки, що відноситься

до обслуговування БД, які краще виконувати на сервері. Як правило, складні функції для роботи з БД розташовуються на стороні клієнта. На сьогоднішній день реалізації даної схеми є найбільш поширеними реалізаціями архітектури клієнт-сервер, вона дозволяє користувачеві працювати з додатками, що відповідають його локальним потребам.

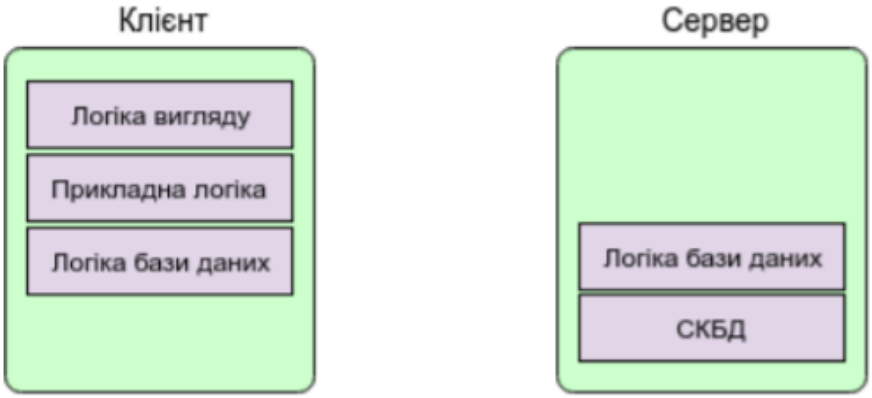


Рисунок 2.8 – Обробка даних на базі клієнта

2.6.4 Спільна обробка даних

У даній конфігурації обробка даних оптимізована таким чином, щоб використовувати сильні сторони як клієнта, так і сервера, а також самого факту розподілу даних.

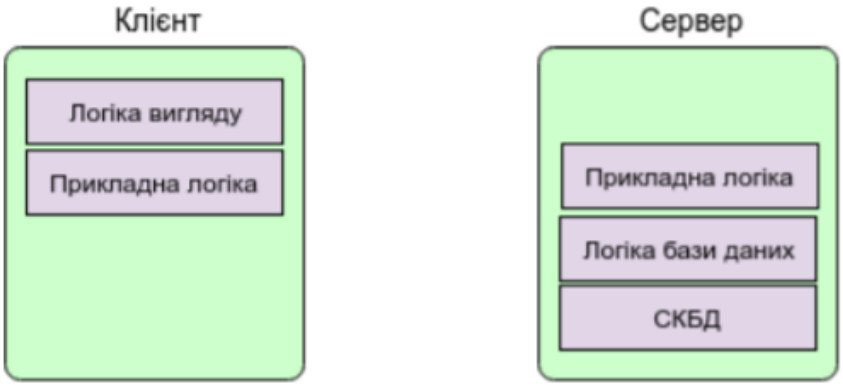


Рисунок 2.9 – Спільна обробка даних

Такі зміни набагато складніші в установці та обслуговуванні, але в довгостроковій перспективі вони дозволяють забезпечити кращі показники продуктивності та ефективності використання мережевих ресурсів, ніж інші методи реалізації архітектури клієнт-сервер.

2.7 Структурна модель системи

У результаті аналізу вимог до розроблюваної системі, виділені основні її характеристики, які визначають набір використовуваних технологій та дозволяють сформувати уявлення про архітектуру.

Комп'ютерну інформаційну систему пропонується реалізувати на трьох відокремлених логічних рівнях. Рівень клієнтів - забезпечує функціональність різних АРМ ЗЗСО; експерта; РЦМ для відповідних категорій користувачів. Рівень управління - призначений для синхронізації і диспетчеризації потоків даних між віддаленими АРМ рівня клієнтів і сервісними підсистемами рівня сервісів. Рівень сервісів призначений для взаємодії з різного роду зовнішніми АІС, що забезпечують процес контролю знань, наповнення інформаційних сховищ, ведення єдиної НДІ. Структурна модель пропонованого рішення наведена на рис. 2.10.

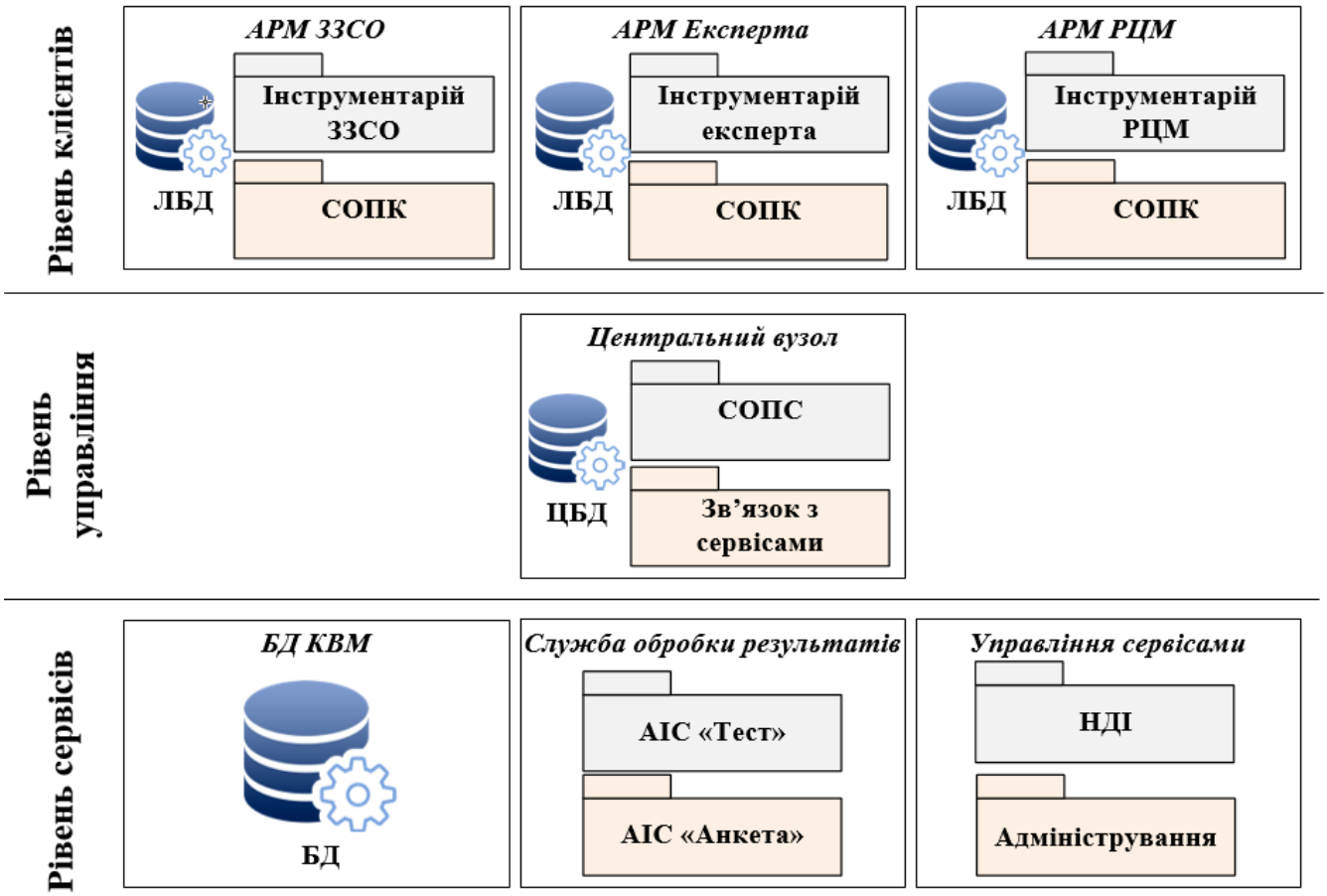


Рисунок. 2.10 – Структурна модель комп'ютерної інформаційної системи

2.7.1 Підсистеми загального призначення

Підсистема обміну повідомленнями логічно реалізується у вигляді двох частин: СОПК та СОПС.

СОПК у рамках пропонованого рішення є обов'язковою частиною будь-якого АРМ рівня клієнтів. Забезпечує можливість АРМ експортувати та імпортувати необхідні дані. Наприклад, необхідність оновлення клієнта, оновлення кодифікатора, зміна плану і т.д.

СОПС – призначений для управління потоком вхідних та вихідних повідомлень, диспетчеризації потоку повідомлень між елементами ІС.

Підсистема зв'язку з сервісами забезпечує взаємодію АРМ клієнтського рівня з загальними сервісами, наданими системою. На вході отримує повідомлення з СОПС, перетворює дані цього повідомлення в формат, що підтримується сервісною АІС. Отримує результати роботи АІС, перетворює дані в формат повідомлень і передає їх СОПС.

2.7.2 Спеціалізована підсистема рівня управління

Підсистема ЦБД - регіональна база даних результатів освітньої діяльності - забезпечує централізоване сховище даних про учнів, педагогів, освітніх досягнень. За рахунок підсистеми служба обробки результатів забезпечується інтеграцією з іншими сервісними підсистемами.

2.7.3 Підсистема рівня сервісів

Служба обробки результатів - виконує наступні завдання: експорту - імпорту в / з різні системи автоматизації оцінки освітніх досягнень. По суті, є надбудовою над існуючими АІС, яка надає зручний єдиний інтерфейс для рівня управління. Наявність такого інтерфейсу дає потенційну можливість розширювати список використовуваних систем оцінки освітніх досягнень.

Підсистема НДІ забезпечує ведення єдиного кодифікатора розділів предметів, критеріїв атестації та іншої умовно-стійкої інформації.

Підсистема адміністрування реалізує стандартну функціональність по адмініструванню елементів ІС.

БД КВМ- сховище контрольно-вимірювальних матеріалів, розроблених експертами, білетів ЗНО минулих років і т.д. Забезпечує єдине джерело для оцінки якості підготовки учнів у регіоні. Підсистема зв'язку з сервісами реалізує функції імпорту білетів з БД КВМ, створених за попереднім запитом ЗЗСО.

2.7.4 АРМ ЗЗСО (рівень клієнтів)

Підсистема інструментарій ЗЗСО реалізує бізнес-логіку характерну для цього типу клієнта:

– планування ОЗ. Планування оціночних заходів здійснюється оператором ЗЗСО (функції якого як правило виконує завуч). Існує кілька видів ОЗ: внутрішні (ініціюються у ЗЗСО в рамках плану проведення оціночних заходів) - поточний, рубіжний контроль успішності, анкетування по позанавчальних компетенцій, психологічне тестування та інші; зовнішні - ЗНО, підсумкова атестація;

– запити незалежних КВМ з центрального сховища (не передбачений в ході реалізації першого етапу). За сформованим планом оцінювання відправляється запит на рівень управління, який передається в БД КВМ, у результаті обробки запиту відповідальна особа отримує набір КВМ, по заданому предмету та заданій темі;

– дозволяє формувати зміст КВМ, забезпечує підтримку локального кодифікатора з можливістю складання специфікації екзаменаційної роботи;

– автоматизація обробки результатів оцінювання - ведення результатів ОЗ. Первинна обробка результатів оцінювання передбачена ПЗ практично всіх використовуваних АІС. Передбачається, що в ЛБД ЗЗСО надходять матриці відповідей учнів з накладеними на них ключами відповідей. Основна функція даного етапу - зв'язок матриці відповідей з елементами локального кодифікатора;

– синхронізація НДІ. До НДІ відносяться: дані учнів, дані педагогів, довідники (предметів, видів контролю, періодичності контролю, локальний кодифікатор);

– механізм формування запитів на зміни. Викладач або відповідальна особа можуть сформувати запит на зміну КВМ, НДІ і т.д. Запит надходить на вхід СОПК і передається на центральний вузол. СОПС в рамках центрального вузла аналізує структуру запиту і передає його на розгляд експертам. У разі позитивної обробки запиту система формує відповідь і розсилає прийняту зміну по іншим вузлам. У разі негативної - система інформує про це відправника;

– надає звіти про стан та результати різних ОЗ, динаміці освітніх досягнень. Аналіз результатів ОЗ реалізується в двох варіантах. За поточним заходом - встановлюється межа оцінювання в рамках, для кожного учня розраховується сумарний бал, відсоток виконання завдань, дозволяє визначити проблемні теми, оцінити якість ОЗ. Порівняльний аналіз - порівнюються результати поточного заходу з історією даного учня. Оцінюється загальна динаміка класу, динаміка конкретного індивідуума;

– дозволяє аналізувати інформацію необхідну для атестації ЗЗСО, окремих педагогів;

– реалізує процедури імпорту / експорту в інші системи на рівні ЗЗСО. Передбачається,

що формати даних в рамках ЛБД ЗЗСО і деякої зовнішньої системи, з якої необхідно здійснювати інтеграцію, відомі. Процедури імпорту / експорту з зовнішніх підсистем реалізуються за рахунок конвертації даних в ці відомі формати. Або за рахунок додаткової обробки підготовлених в рамках зовнішніх підсистем файлів, що мають деяку фіксовану структуру, наприклад, MS Excel, у необхідний xml.

ЛБД – локальна база даних ЗЗСО. Містить всю необхідну інформацію для роботи АРМ ЗЗСО.

2.7.5 АРМ РЦМ (рівень клієнтів)

Підсистема інструментарій РЦМ реалізує бізнес-логіку характерну для цього типу клієнта:

- планування ОЗ для ЗЗСО;
- синхронізація НДІ;
- механізм формування запитів на зміни;
- надає звіти про стан, результати і плануванні різних ОЗ, динаміці освітніх досягнень;
- дозволяє аналізувати інформацію необхідну для атестації ОЗ, окремих педагогів.

ЛБД - локальна база даних ЗЗСО. Містить всю необхідну інформацію для роботи АРМ РЦМ.

2.7.6 АРМ Експерта (рівень клієнтів)

Підсистема інструментарій Експерта реалізує бізнес-логіку характерну для цього типу клієнта:

- дозволяє формувати зміст для КВМ;
- відстежує стан кодифікаторів за рівнем компетенції експерта;
- дозволяє віддалено приймати / відхиляти пропозиції щодо наповнення БД КВМ і кодифікатор, що надійшли від інших експертів / педагогів.

ЛБД – локальна база даних ЗЗСО містить всю необхідну інформацію для роботи АРМ Експерта.

2.8 Аналіз інструментів, що використовуються при розробці системи

Виходячи з вимог наведених вище, виникла необхідність використання широкого спектру сучасних технологій та інструментів, що забезпечують високу швидкість і надійність роботи всієї системи.

Для реалізації центральної бази даних був використаний MS SQL Server 2016 - система керування базами даних, розроблена корпорацією Microsoft. Вибір саме цієї СУБД був обумовлений більшою мірою вимогами замовника, що має ліцензію на даний продукт і навчених фахівців для його адміністрування.

З використанням ще однієї технології компанії Microsoft - веб-сервісів - було створено ядро підсистеми ЦБД. Веб-сервіс - програмна система, що ідентифікується уніфікованим рядком абстрактних чи фізичних ресурсів (URI), чиї загальнодоступні інтерфейси визначені мовою XML. Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису за допомогою повідомлень, заснованих на XML, і переданих за допомогою інтернет-протоколів. По суті це означає, що за допомогою веб-сервісів функції будь-якої програми можуть стати доступними через Інтернет. Саме ця технологія дозволила організувати універсальну систему реплікації клієнтів з центральною базою даних, за допомогою виклику функцій ядра підсистеми ЦБД через web.

Інше застосування веб-сервісів у загальній схемі рішення - надання інформації в репрезентативному вигляді для використання її при побудові інтерфейсу користувача серверної частини (рівень управління). Іншими словами, функції, які повертають різні зрізи даних серверної БД, можуть бути викликані будь-якою програмою за допомогою мережі Інтернет, що і використовується web-додатком (ASP.NET), що реалізує інтерфейс користувача підсистеми ЦБД.

Для реалізації клієнтської частини системи було використане інтегроване середовище розробки програмного забезпечення Microsoft Visual Studio 2014 і мову C#. Цей вибір обумовлений тим, що, маючи великі функціональні можливості, клієнтський додаток повинен залишатися легко розширюваним і підтримуваним, що в свою чергу неможливо без застосування об'єктно-орієнтованого підходу. Обраний інструмент розробки є визнаним лідером в області написання захищених і багатофункціональних об'єктно-орієнтованих програм. При цьому такий інструментарій істотно скорочує час реалізації проекту, що важливо.

До недоліків можна віднести необхідність використання на кожній клієнтській машині системи .NET framework, але дана проблема частково вирішується включенням всіх необхідних бібліотек в інсталятор продукту.

Для зберігання локальних даних освітнього закладу був використаний SQL Server Express Edition, який є безкоштовно розповсюджуваною версією SQL Server. По-перше, даний продукт задовольняє всім початковим вимогам. По-друге, має ідентичні типи даних з базою, яка використовується як центральне сховище, що дещо спрощує реплікацію. По-третє, має прийнятну швидкість роботи, для систем такого рівня. Ці три чинники, переважують обмеження та зумовили вибір MS SQL Server Express в якості ЛБД. До основних обмежень можна віднести максимальний розмір пам'яті в 1 Гб і максимальний розмір бази в 4 Гб, що, з огляду на специфіку предметної області, не є суттєвим.

Необхідність подання аналітичних даних в зручному для користувача вигляді, породила ідею використання будь-якого генератора звітів з розширеними функціями побудови діаграм і графіків. Через деякі особливості схеми бази даних, стандартний генератор - Crystal Reports, вбудований в MS Visual Studio, не впорався з цим завданням. У зв'язку з цим був використаний повнофункціональний, швидкий і компактний генератор звітів Fast Report, який має досить низьку вартість для систем такого класу.

2.9 Висновки до розділу 2

1. Розглянуті методологічні аспекти проектування системи і основні класи клієнт-серверних архітектур, типи архітектур клієнт-серверних додатків, їх переваги та недоліки. Велика частина уваги приділена опису способів спілкування комп’ютерів у клієнт-серверній архітектурі. Проаналізувавши переваги та недоліки, можна зробити висновок, що для системи збору, обробки та інтерпретації інформації про результати освітньої діяльності очевидною є необхідність застосування клієнт-серверної технології, та використання web-інтерфейсів для забезпечення роботи системи.

2. Представлена узагальнена процесна модель і наочна модель якості освіти у ЗЗСО, що забезпечує можливість спрямувати зусилля ЗЗСО на покращення конкретної складової якості.

3. Визначено структурні елементи комп'ютерної ІС, яку пропонується реалізувати на трьох відокремлених логічних рівнях: рівень клієнтів; рівень управління; рівень сервісів.

4. Проведено аналіз інструментів, що забезпечують високу швидкість і надійність роботи всієї системи; обгрунтовано вибір:

– MS SQL Server 2016 для реалізації центральної бази даних;

– інтегрованого середовища розробки програмного забезпечення Microsoft Visual Studio 2014 і мову C # для реалізації клієнтської частини системи;

– SQL Server Express Edition для зберігання локальних даних освітнього закладу.

РОЗДІЛ 3 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ОСВІТИ

3.1 Функціонал клієнстької частини системи

3.1.1 Елементи інтерфейсу

Будь-яка ІС повинна підтримувати деякі стандарти в інтерфейсі, які покликані зробити її більш передбачуваною для користувача. Такий підхід не тільки спрощує навчання фахівців для роботи з програмою-клієнтом, але і дозволяє розробнику застосовувати деякі шаблонні рішення при створенні призначених для користувача форм. При проектуванні клієнта ІС був прийнятий ряд стандартів по відображенню зрізів даних різних типів.

Доступ до всіх функціональних частин програми можна отримати з головної форми (рис. 3.1). Також тут можна задати деякі налаштування, пов'язані з роботою клієнта (наприклад, робочу дату).

Рисунок 3.1 – Головна форма

Всі інші форми додатка відносяться до одного з наступних типів, що наведені нижче.

Прості спискові форми – призначені для відображення допоміжної НДІ, яка не може бути змінена на клієнті. Розраховані на малу кількість записів і не підтримують розширені засоби для пошуку і фільтрації даних. Містять, як правило, всього два нередактованих поля - Код і Назва. Приклад форми даного типу наведено на рис. 3.2.

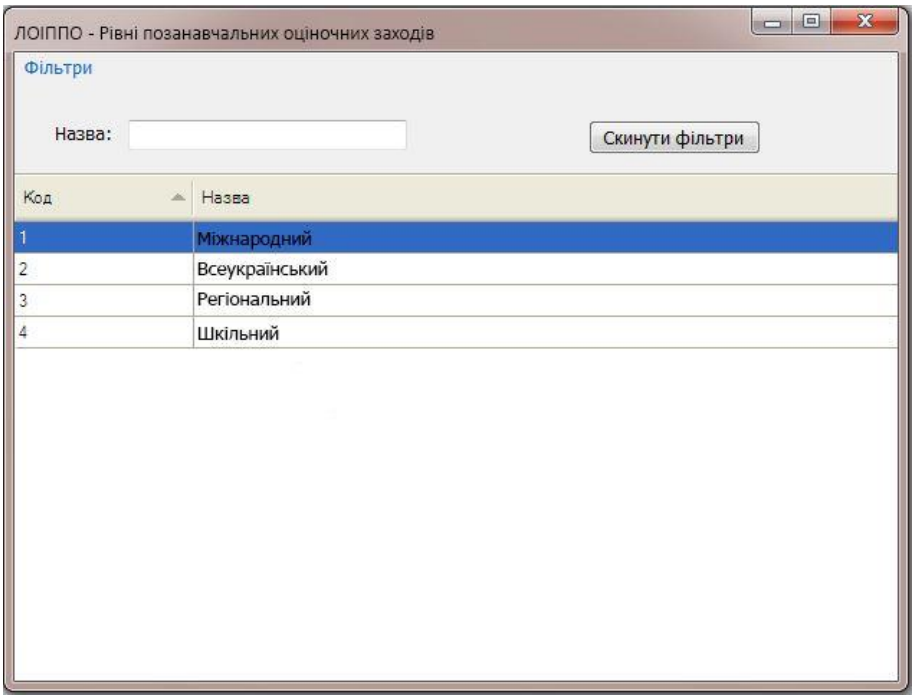


Рисунок 3.2 – Проста спискова форма

Спискові форми з розширеним функціоналом (рис. 3.3) - служать для відображення довідників з великою кількістю записів і полів, а також пов'язаних даних з декількох таблиць. Дозволяють виробляти пошук та фільтрацію даних, але не дозволяють редагувати записи. Для додавання, видалення і редагування записів використовується контекстне меню, яке також може містити додаткові опції.

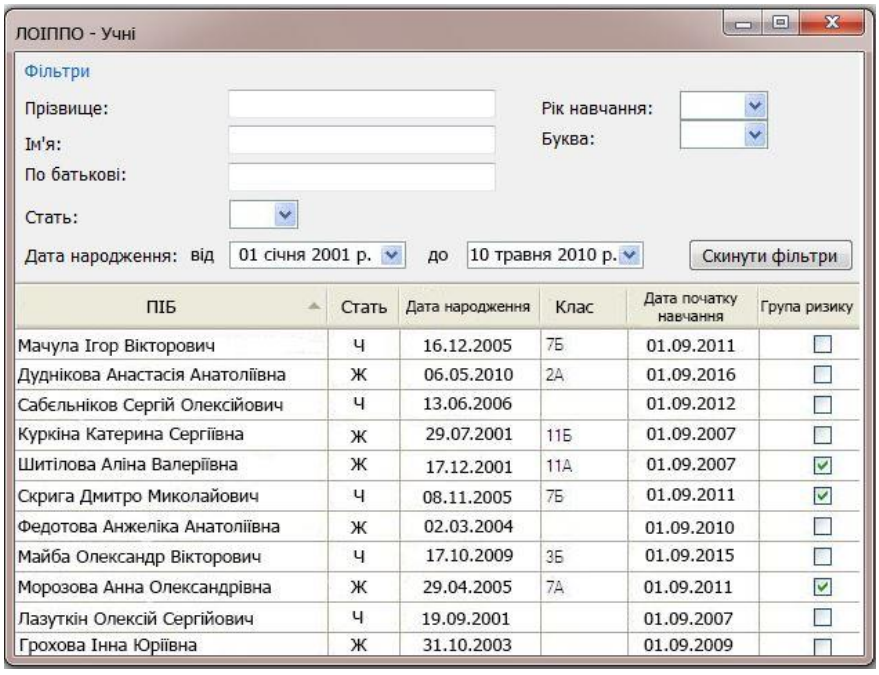


Рисунок 3.3 – Спискова форма з розширеним функціоналом

Ієрархічні форми (рис. 3.4) - реалізують той же функціонал, що і спискові форми, але

відображають дані з ієрархічною структурою у вигляді дерева. Також, підтримують прив'язку різних контекстних меню для вузлів певного типу. Деревовидна форма має засоби фільтрації даних і швидкого вилучення вузлів.

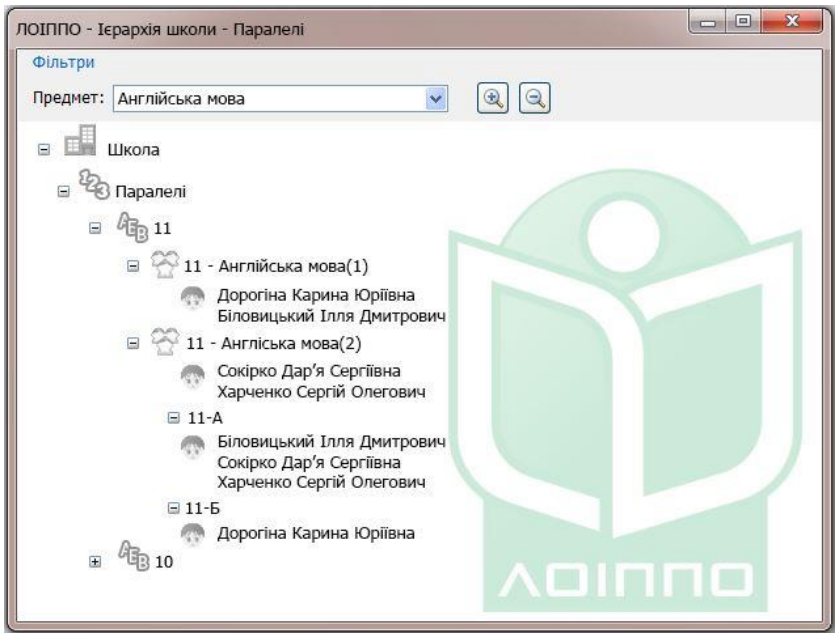


Рисунок 3.4 – Ієрархічна форма

Карткові форми (рис. 3.5) - призначені для редагування і додавання даних. Викликаються, як правило, з облікових форм для конкретного запису та можуть містити додаткову інформацію про цей запис (те, чого з яких-небудь причин немає в обліковій формі). Також, більшість критичних форм містить кнопку «Функції», яка здійснює доступ до додаткового функціоналу.

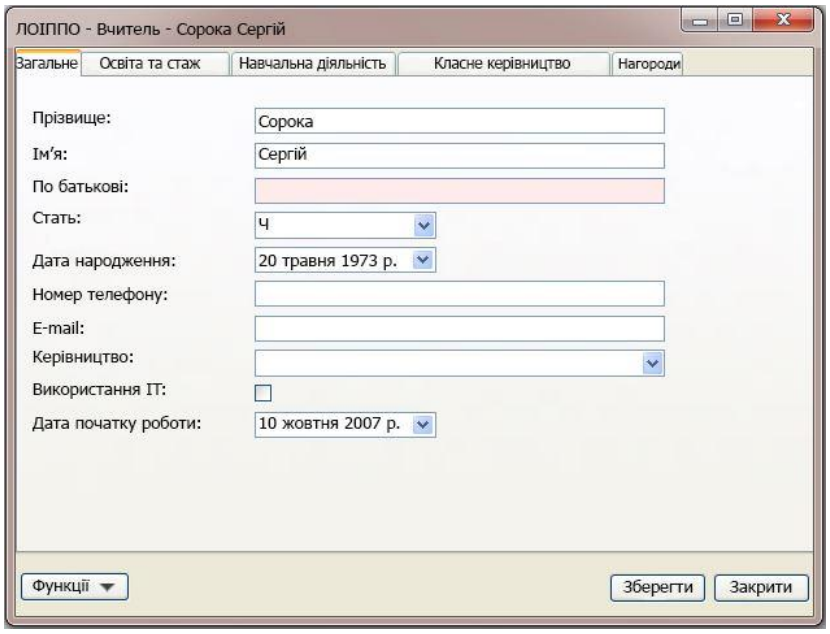


Рисунок 3.5 – Карткова форма

Форми інших видів - використовуються в обмеженій кількості, для реалізації діалогів з користувачем (наприклад, під час налаштування клієнта), відображення даних зі складною структурою та залежностями. Прикладом такої форми може служити форма налаштування шкали оцінювання для ОЗ (рис. 3.6).

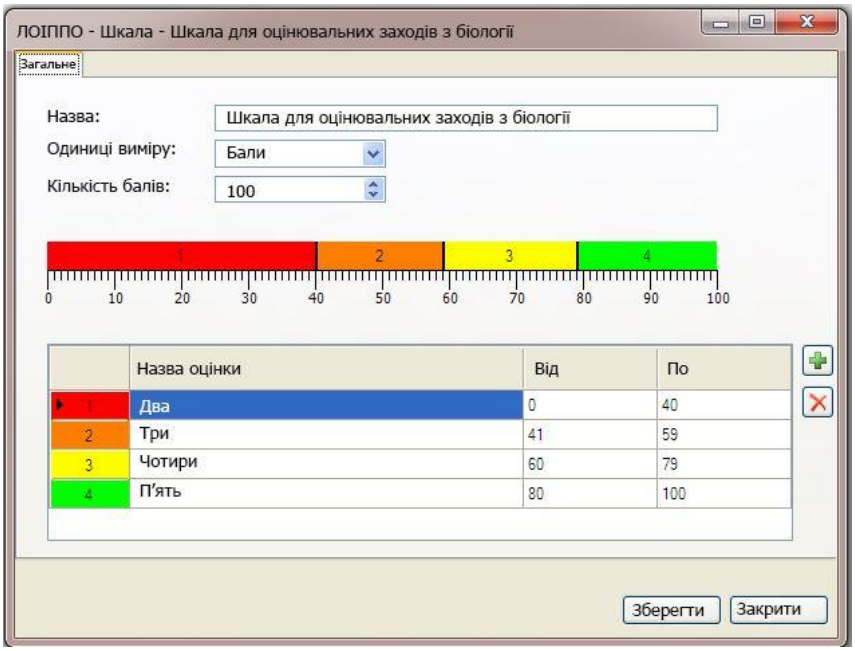


Рисунок 3.6 – Форма специфічного виду

Форми предналаштування звітів - необхідні для налаштування фільтрів та інших параметрів перед друкуванням звітів (рис. 3.7).

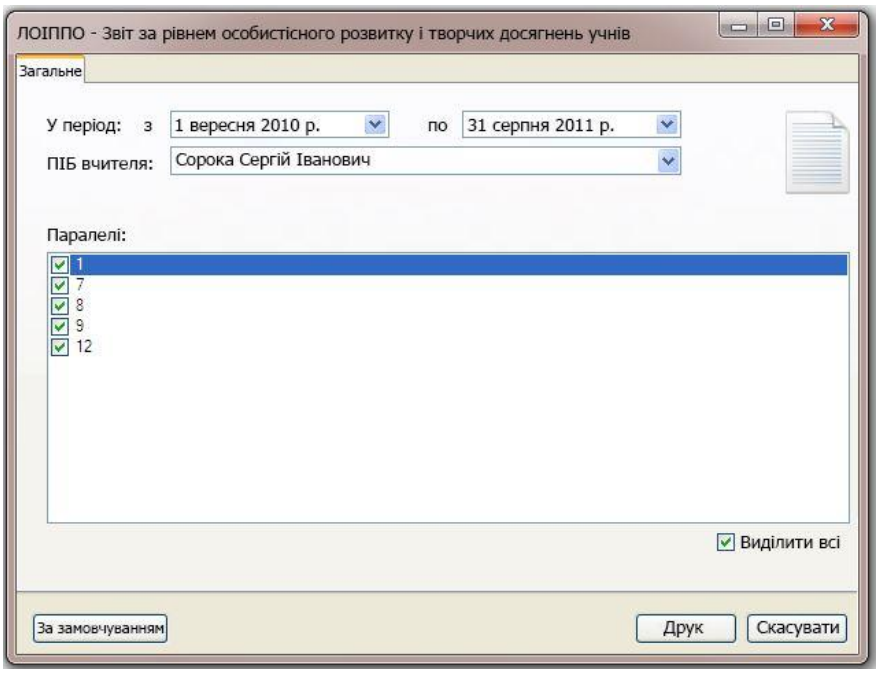


Рисунок 3.7 – Форма предналаштування звіту

Друковані форми (рис. 3.8) - форми, створені за допомогою генератора звітів FastReport. Мають можливість виведення на друк, а також експорту в різні формати. Можуть містити як звичайні табличні дані, так і графіки, діаграми та ін.

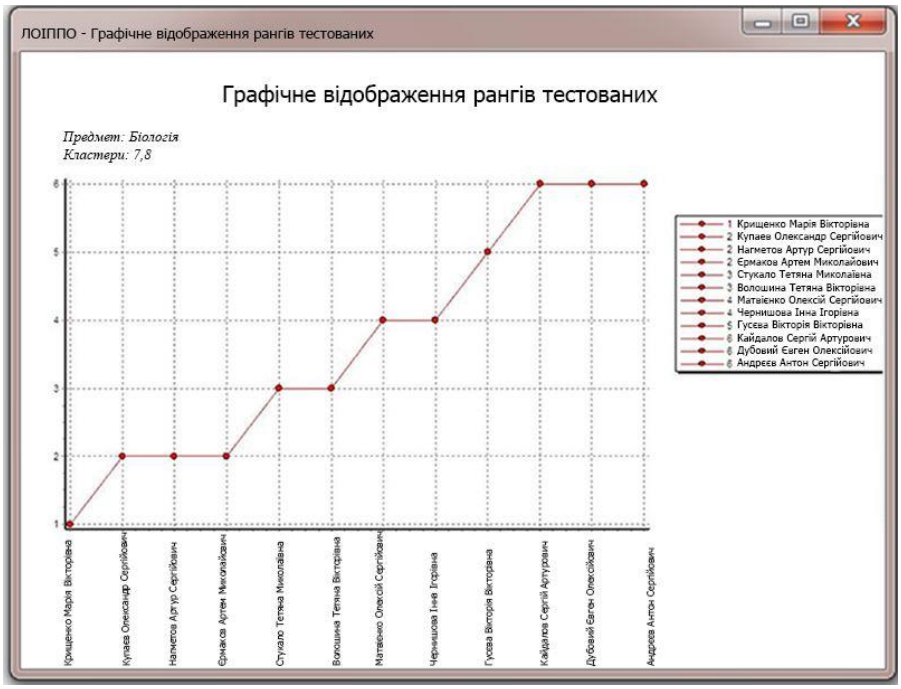


Рисунок 3.8 – Друкована форма

Форми майстрів (рис. 3.9) - форми, що вимагають від користувача виконання ряду кроків, що використовуються для послідовного збору інформації.

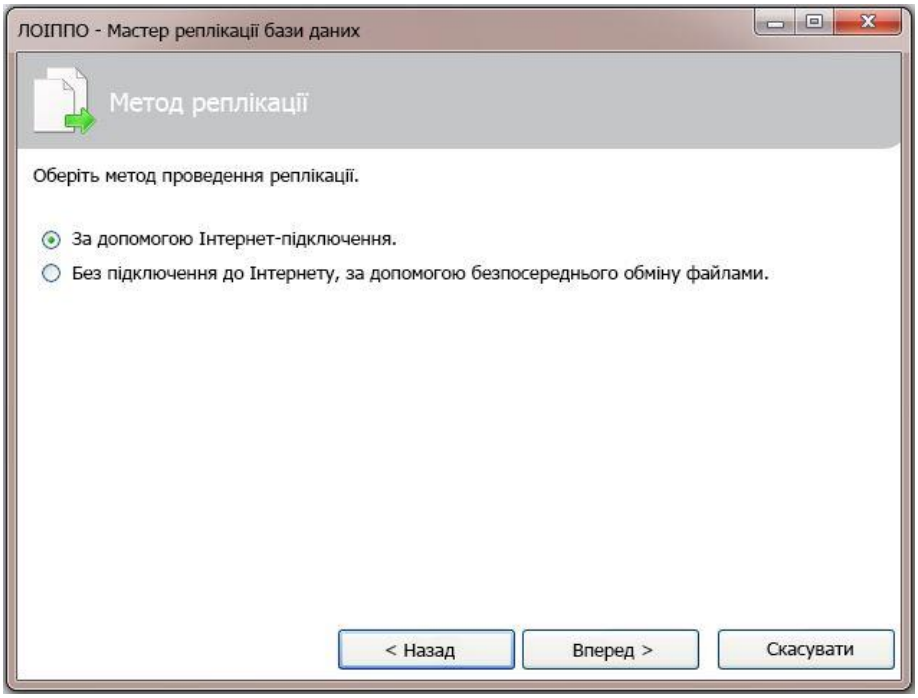


Рисунок 3.9 – Форма-майстер

3.1.2 Ролі користувачів

На першому етапі розробки було прийнято рішення про створення двох ролей для користувачів клієнтської програми, що фактично відповідає двом АРМ. Поділ здійснюється як на рівні інтерфейсу, так і на рівні доступу до даних, але в рамках однієї програми. Для кожного користувача в системі заводиться обліковий запис, а в момент авторизації (введення логіна і пароля), здійснюється прив'язка користувача та його ролі, а також прив'язка користувача до його робочих даних. Така система дозволяє бачити педагогу, наприклад, створені тільки ним ОЗ.

Далі, найпростіше описати функціональні частини системи і ролі користувачів за допомогою діаграми варіантів використання, яка приведена на рис. 3.10.

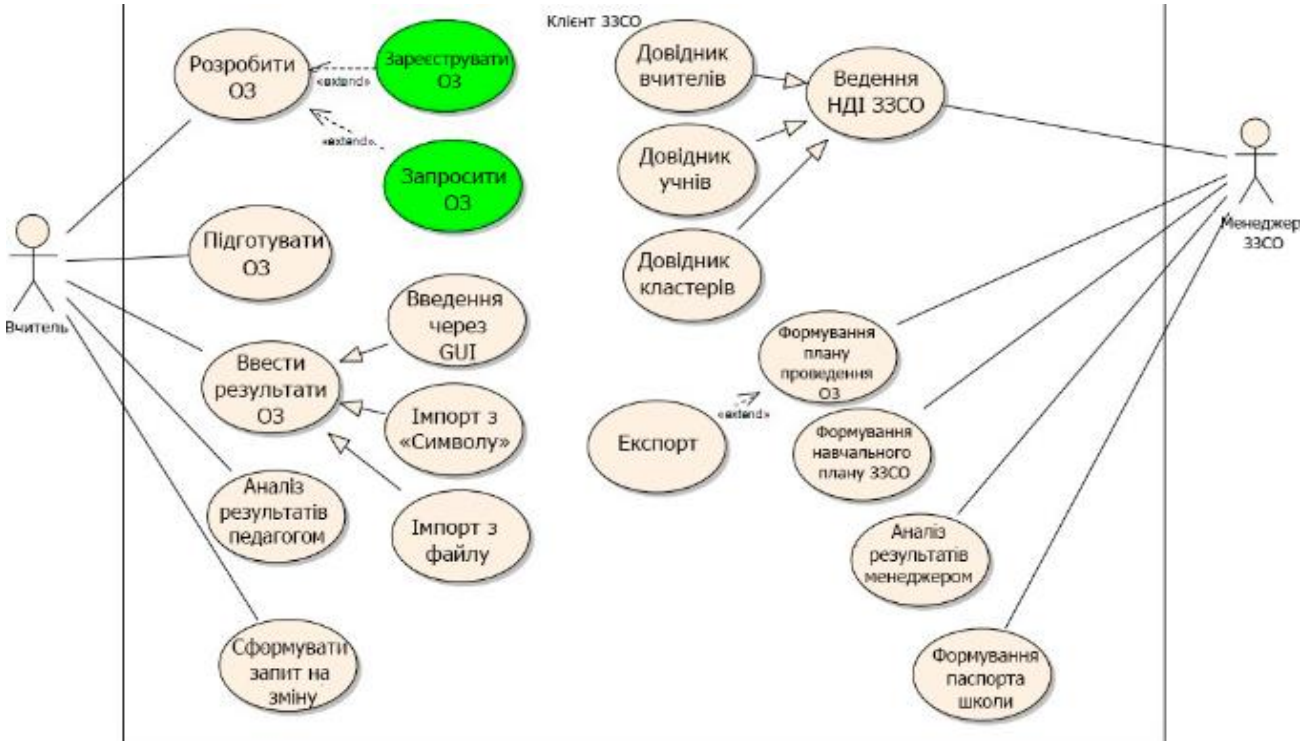


Рисунок 3.10 – Діаграма варіантів використання клієнта ЗЗСО

Як видно з діаграми, існує два види користувачів системи (акторів) - це вчитель і так званий менеджер ЗЗСО, яким може бути завуч або директор.

3.1.3 АРМ вчителя

Центральним поняттям у предметній області, пов'язаним з АРМ вчителя, є поняття ОЗ, яке представляє собою розширену форму тестування. Під розширеною формою тестування слід розуміти можливість складання різних типів питань для оцінювання. Сюди входять не тільки звичайні тестові завдання з єдиним правильним варіантом відповіді, але і питання, які потребують

множинного вибору, самостійної відповіді, зіставлення різних понять або складання послідовності дій, що веде до вирішення завдання. Завдяки тому, що текст кожного питання зберігається в форматі rtf, в нього можна включати графіки, зображення, наочні посібники, що значною мірою знижує формулювання завдання. До того ж такий підхід дозволяє практично без змін внести вже наявні у вчителя тести в систему. У доповненні до цього, з сервера можуть бути отримані вже готові ОЗ (або питання з яких вони можуть бути складені), що дозволяє проводити централізовані тестування з мінімальними витратами часу і коштів. Усі наведені вище можливості реалізують варіант використання «Розробити ОЗ».

Наступний етапом життєвого циклу ОЗ є його підготовка до проведення. Тут мається на увазі вибір дати проведення оцінювання, клас (кластер) у якому воно буде проводитися, особа, що контролює захід (найчастіше - вчитель).

Після внесення всіх необхідних даних педагог зможе роздрукувати оцінювання натисненням однієї кнопки. При цьому роздатковий матеріал буде надрукований для кожного учня, що входить у зазначений кластер, з усіма необхідними атрибутами, такими як ПІБ, варіант, код АСК «Символ» (мова про нього піде далі). Також, на етапі підготовки оцінювання, вчителю буде запропоновано скласти (або вибрати з шаблону) шкалу, за якою будуть оцінюватися результати. Оцінка може здаватися у будь-який баловій системі, а кожен виставлений бал може мати свій опис. Підставою для виставляння тієї чи іншої оцінки служить кількість (або процентне співвідношення) набраних умовних одиниць, якими оцінюється кожне питання оцінювання у залежності від його складності. Таким чином, якщо викладач поставить шкалу в період підготовки ОЗ, то при введенні результатів в систему, кожному учневі буде автоматично виставлена оцінка адекватна виявленим знань.

Наступна частина структури проведення оцінювання, а значить і наступний варіант використання - пункт, названий «Ввести результати оцінювання». Введення результатів може здійснюватися кількома способами, найпростіший з яких - використання АСК «Символ». АСК «Символ» - пристрій, розроблений Томським університетом систем управління і радіоелектроніки, призначене для перевірки знань учнів. Не так давно воно було запущено в масове виробництво і тепер планується впровадження даного пристрою в усіх школах Томської області. При тестуванні за допомогою «Символ», учень отримує екзаменаційний лист, на якому роздруковані його ПІБ, варіант і спеціальний код ідентифікації в «Символ». Код вводиться в пристрій, а потім залишається занести туди номери обраних варіантів відповідей. Після закінчення тестування, за допомогою data-кабелю, «Символ» з'єднується з комп'ютером, на якому встановлена ІС і «віддає» внесені в нього дані системі. Здійснюється вся операція натисканням на одну кнопку, а якщо при підготовці оцінювання була введена шкала, то система «проставить» кожному учню відповідну оцінку. Єдиний недолік такого підходу полягає в тому, що з використанням «Символ», можна проводити тільки класичні тести, з єдиним правильним варіантом відповіді. Ще один шлях

внесення даних, через xml-файл певної структури, створений для інтеграції програми з будь-якими іншими зовнішніми системами.

Як і будь-яка інша ІС, дана система повинна надавати великі можливості для аналізу введених даних, в тому числі і ОЗ. Багато, або навіть більшість таких можливостей будуть реалізовані в наступних ЖЦ системи. Але навіть зараз, в першому наближенні, викладач може отримати основні аналітичні дані, що дозволяють йому модернізувати навчальний процес і контрольні завдання. Така інформація видається системою у вигляді звітів та дозволяє оцінити, які питання або розділи були найважчими для тестованих, а на які відповісти було найпростіше, також зробити висновки про успішність класу в цілому і кожного учня окремо.

3.1.4 АРМ менеджера ЗЗСО

Так як все, описане вище відносилося до ролі користувача, названої «Вчитель», то варто сказати кілька слів про роль «менеджер ЗЗСО». У його «володінні» знаходяться не тільки ОЗ і аналітика пов'язана з ними, але і вся НДІ ЗЗСО, така як списки учнів по класах, підгрупами і т.д., списки вчителів, паспорт школи, навчальні плани.

Ще одна частина подібної інформації виходить з сервера і не може бути відредагована користувачами клієнта, але при цьому повинна використовуватися, наприклад, при складанні навчальних планів менеджером ЗЗСО. Завдяки механізму зміни робочої дати менеджер може не тільки відстежувати актуальний стан навчального процесу, а й оцінювати його в динаміці, переміщаючись на певні проміжки часу назад і порівнюючи результати. Використовуючи цей же підхід, можна переглянути історію викладання предметів в класах, успіхів учнів і т.і.

3.2 Опис архітектури системи

3.2.1 Загальна схема залежностей між елементами системи

Для розуміння архітектури клієнтського додатка слід коротко розглянути архітектуру всієї системи вцілому і зрозуміти залежності між її елементами. Отже, виходячи з вимог, що пред'являються до системи, вона повинна підтримувати централізований збір даних від ЗЗСО регіону та дозволяти зберігати загальну НДІ. Таким чином, для цих цілей необхідна загальна база даних, яка і була реалізована під керуванням MS SQL Server 2016. У проектній документації цей вузол одержав назву – ЦБД, на відміну від ЛБД, що зберігають дані на рівні ЗЗСО.

Далі, потрібно було створити універсальний засіб доступу до інформації, що зберігається в ЦБД, причому дозволяє працювати з нею по мережі. Такими можливостями володіє web-сервіс, який отримує доступ до ЦБД та реалізує функції, які повертають різні зрізи даних. Очевидно, що

фізично СУБД і web-сервіс розташовані на одній машині. Разом, обидва ці вузла отримали назву підсистеми ЦБД, як реалізують засоби зберігання і доступу до даних.

Для роботи з ЦБД на рівні муніципалітетів, тобто для управління НДІ, аналізу даних, отриманих від ЗЗСО і т.і., було розроблено web-додаток (ASP.NET), який фактично реалізує інтерфейс користувача підсистеми ЦБД. Так як працює він виключно з web-сервісом по мережі, то може бути розташований на будь-якому сервері, що надає послуги хостингу.

Останньою, але не менш важливою підсистемою, є клієнтська програма рівня ЗЗСО, функціонал якої вже був описаний вище. Варто сказати, що для кожного ЗЗСО може бути організовано кілька робочих місць (тобто встановлено кілька клієнтських додатків), але працювати вони повинні з єдиною ЛБД даної установи.

Більш детально архітектуру системи можна розглянути на рис. 3.11

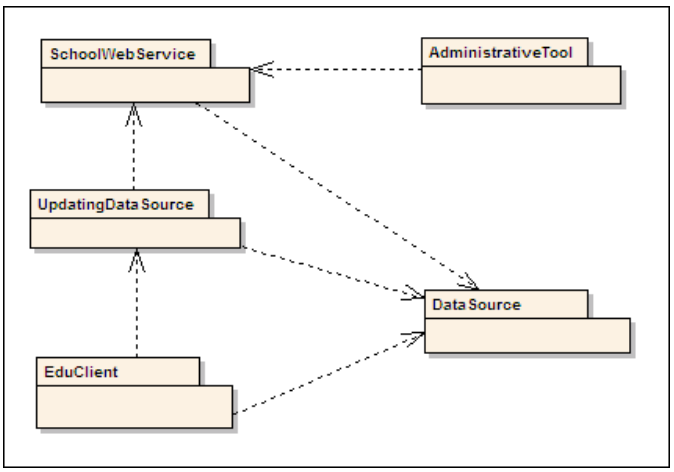


Рисунок 3.11 – Архітектура системи на рівні модулів

Фізично, блоки, зображені на схемі, є окремими програми, або бібліотеками (dll), які використовуються в різних частинах системи. Далі йде короткий опис представлених модулів:

- SchoolWebService фактично, являє собою web-сервіс доступу до даних, є ядром підсистеми ЦБД, описує практично всю логіку роботи серверної частини системи;
- AdministrativeTool web-додаток реалізує інтерфейс користувача підсистеми ЦБД;
- EduClient є частиною клієнта, реалізує інтерфейс роботи з користувачем і менеджмент всього програми;
- DataSource є частиною клієнта, реалізує доступ до даних ЛБД, виконаний у вигляді окремої бібліотеки, як видно зі схеми, даний модуль використовується як на клієнті ЗЗСО, так і для роботи web-сервісу, так як схеми локальних баз ЗЗСО і ЦБД відрізняються незначно, то для доступу до даних використовується одна і та ж бібліотека;
- UpdatingDataSource забезпечує зв'язок з web-сервісом, необхідний, перш за все, для отримання обміну даними з підсистемою ЦБД, дана бібліотека реалізує всю систему відправки

повідомлень, як online, так і offline.

Далі буде дано більш докладний опис модулів, використовуваних клієнтським додатком, а також схеми БД рівня ЗЗСО.

3.2.2 Організації доступу до даних на клієнті

Велика частина можливостей, що реалізуються клієнтським додатком, пов'язана з роботою з базою даних ЗЗСО. Отже, одним з основних пріоритетів, при розробці клієнта, було створення надійного і швидкого способу доступу до інформації, що зберігається в ЛБД. Для подання інформації в репрезентативному вигляді була розроблена досить проста, але ефективна об'єктна модель, яка увійшла до складу модуля DataSource. Основою для побудови доступу до таблиць БД став абстрактний клас, названий BasicEntity. Логіка роботи даного класу ґрунтується на наступних моментах:

- клас описує рядок деякої абстрактної таблиці і методи роботи з нею, а також методи для навігації по таблиці і позиціонування на різних її записах;
- позиціонувати на конкретному рядку таблиці, можна отримати або змінити для неї значення полів;
- за допомогою механізму фільтрів, можна обмежити кількість записів, які обирають з таблиці;
- використовуючи механізм сортування, можна визначити порядок, в якому буде здійснюватися прохід по таблиці;
- крім навігаційних операцій, що дозволяють отримувати за один запит тільки один запис таблиці (що позитивно позначається на використанні пам'яті, але негативно впливає на продуктивність), описані методи для отримання одразу кількох рядків з бази за один запит; використання такого підходу вимагає більше пам'яті (так як для кожного отриманого рядка таблиці створюється відповідний об'єкт), але дозволяє зменшити кількість запитів до БД, у кожному конкретному випадку розробник може вибрати золоту середину;
- для кожного класу, що представляє таблицю, описані методи для доступу до пов'язаних даними.

У результаті, схема роботи з будь-таблицею в БД здійснюється наступним чином:

а) створюємо новий клас, для роботи з конкретною таблицею, успадкованих від BasicEntity;

б) дописуємо public властивості для доступу до полів таблиці зовні;

в) якщо необхідно, дописуємо додаткові методи для отримання даних зі зв'язаних таблиць.

Далі будуть наведені конкретні приклади доступу до даних з різних сфер використання клієнта ЗЗСО.

Як вже було сказано, одним з центральних понять в аналізованої предметної області є ОЗ. На рис. 3.12 наведено фрагмент схеми БД, що описує таблиці для зберігання даних про оцінювання.

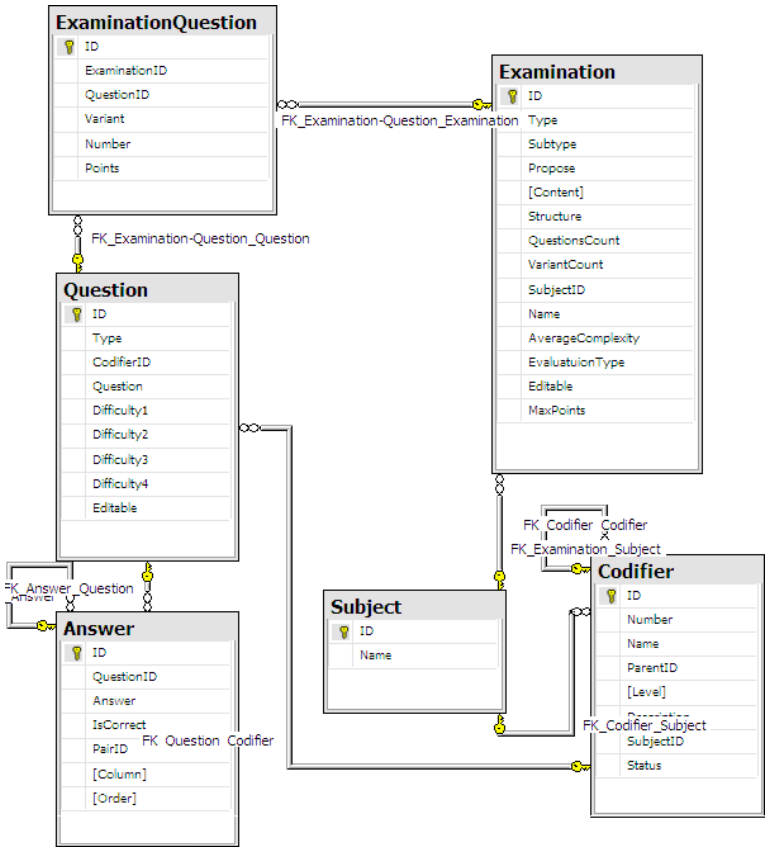


Рисунок 3.12 – Фрагмент схеми БД, що описує ОЗ

Наведемо короткий опис і розкриємо призначення таблиць, відображених на схемі.

Examination - служить для зберігання даних про ОЗ, таких, як наприклад, його призначення, мета, структура, максимальна кількість балів, яку можна отримати при виконанні і т.д.

Question - описує питання, що належать ОЗ.

ExaminationQuestion - забезпечує зв'язок типу «багато-до-багатьох» між ОЗ та питаннями і таким чином дозволяє прив'язувати одне питання до кількох оціночних заходів із зазначенням його складності в конкретному контексті.

Answer - відповіді на питання ОЗ.

Subject - предмет до якого прив'язаний конкретний оціночний захід.

Codifier - кодифікатор (або підрозділ по предмету), який вказується для кожного питання. Якщо питання розглядається в прив'язці до ОЗ, то для нього може бути зазначений тільки кодифікатор, пов'язаний з предметом, обраним для оцінювання. Самі кодифікатори представляють собою трирівневу ієрархічну структуру, прив'язану до одного предмета.

Нижче, на рис. 3.13, показана діаграма класів, пов'язаних з оцінюванням, з якої видно, що

всі класи, призначені для роботи з таблицями, успадковані від BasicEntity і структура зв'язків між ними багато в чому нагадує зв'язку між відповідними таблицями в БД.

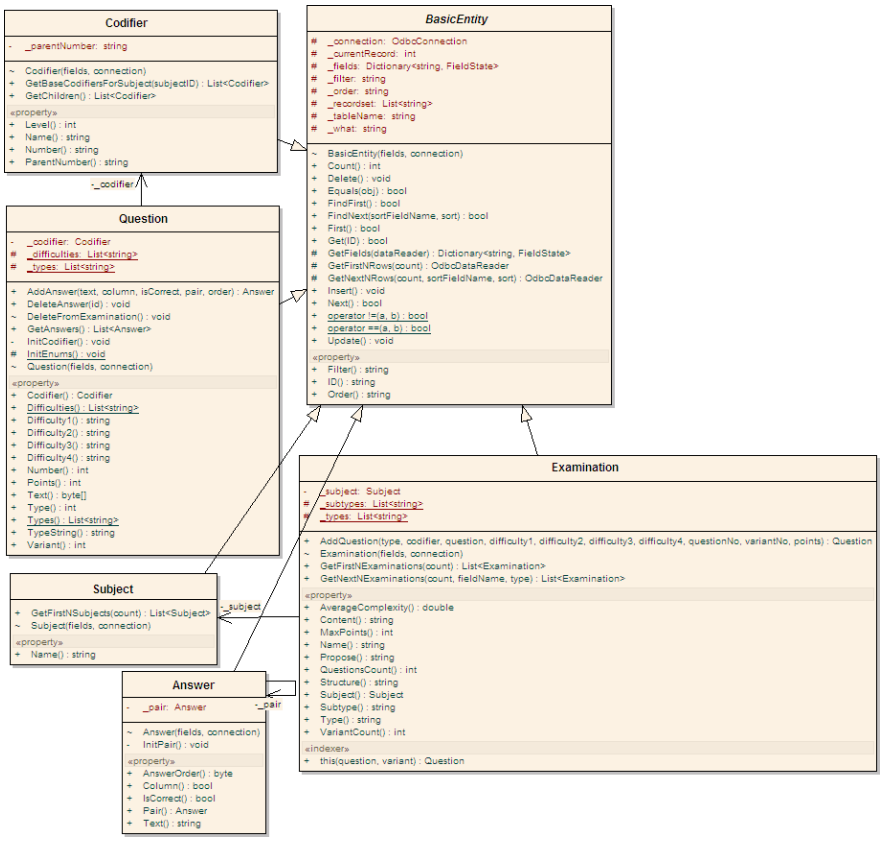


Рисунок 3.13 – Діаграма класів, призначених для роботи з ОЗ

Також, можна помітити, що для зручного доступу до пов'язаних даними в кожному класі описані відповідні методи. Наприклад, позиціонувати на конкретному ОЗ, можна отримати всі його запитання, а для кожного питання - всі відповіді.

3.2.3 Механізм реплікації даних

Одним з найважливіших вимог до системи, є підтримка реплікації даних. Перед описом особливостей реалізації даного блоку варто згадати, чому не були використані стандартні засоби реплікації MS SQL Server, існує кілька вагомих причин. По-перше, у версії Express, яка була використана в якості СУБД для ЗЗСО, досить сильно урізані можливості реплікації. По-друге, замовником був вироблений ряд правил перетворення та перевірки даних, що передаються перед їх «застосуванням» у базі. Наприклад, при отриманні нового пункту кодифікатора з сервера слід перевірити, чи не створений пункт з ідентичним номером на клієнті, і якщо збіг номерів має місце, то перебудувати дерево кодифікатора таким чином, щоб уникнути накладання. І, по-третє, повинен бути розроблений механізм «файлової» реплікації, який не використовує передачу даних

по мережі, що було б важко реалізувати стандартними засобами.

Як було описано вище, всю функціональність, пов'язану з обміном даними з сервером, на клієнті виконує модуль UpdatingDataSource. На рис. 3.14 показана діаграма класів даного модуля, куди також включений класи web-сервісу і його «заголовків».

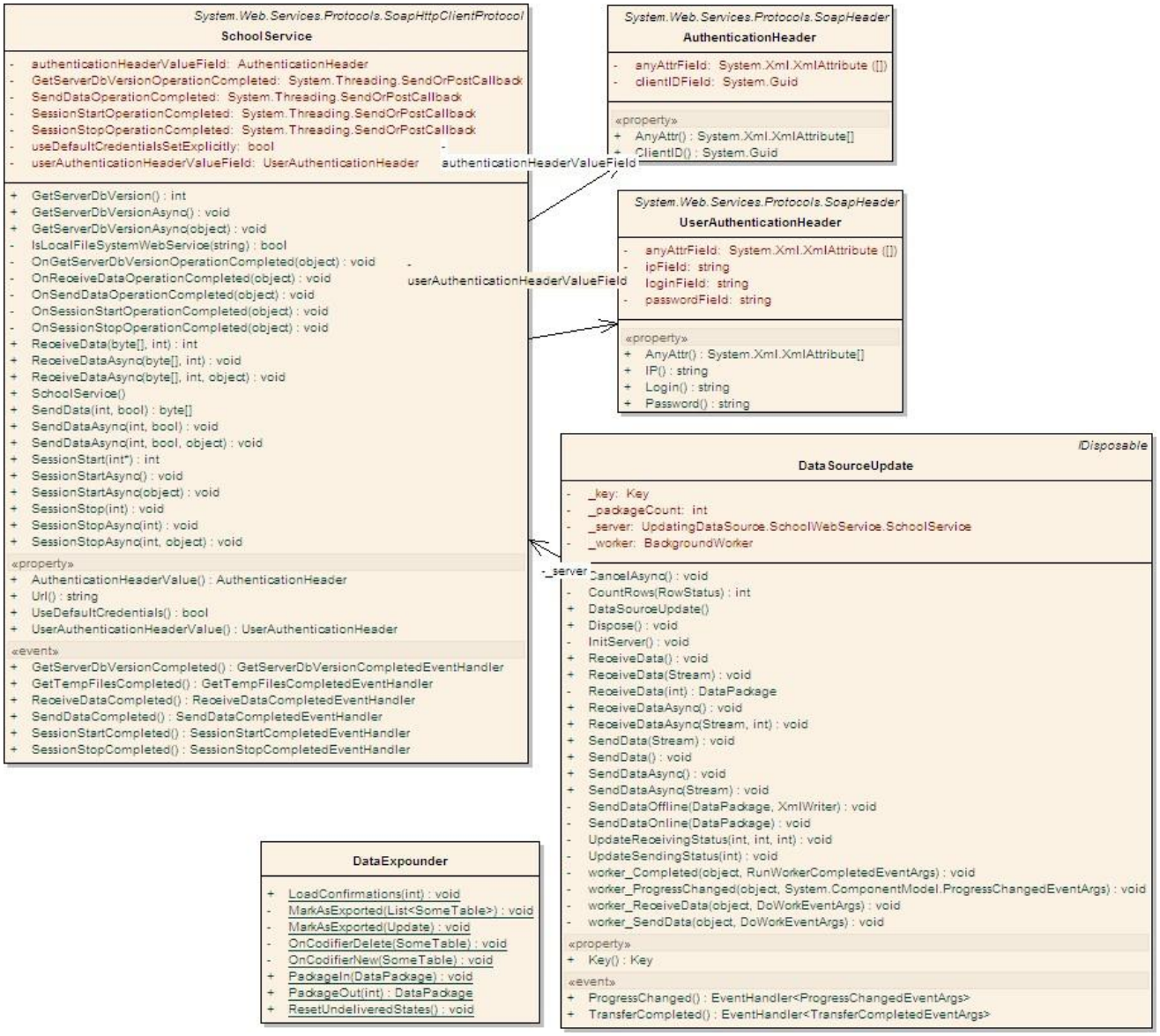


Рисунок 3.14 - Діаграма класів модуля UpdatingDataSource

Основою для реплікації є службова таблиця Update, у якій фіксуються всі зміни даних, пов'язані з роботою користувача. Механізм реєстрації змін реалізований у базовому класі об'єктної моделі - BasicEntity. За замовчуванням, кожен об'єкт фіксує зміни в даних «підзвітною» йому таблиці. При необхідності, логірування для будь-якої з таблиць можна відключити (це актуально, наприклад, при створенні і відновленні резервної копії шкільної бази за допомогою клієнта). Звичайно, що в таблицю Update потрапляють не змінені дані, а факт їх зміни.

У будь-який момент користувач клієнтського додатка може ініціювати відправку

актуальних даних на сервер, а якщо необхідно, то і отримання даних з сервера - ці процеси незалежні. Тому, принцип реплікації даних в системі полягає фактично в тому, що в міру необхідності сервер і клієнт відправляють один одному пакети оновлень, які і завантажуються в базу. Для формування пакету використовується прохід по записах логу, відсортованих у порядку надходження. На кожному кроці алгоритму вибирається запис зі статусом відмінним від «доставлено» і додається в пакет відправки - об'єкт клас DataPackage, мова про який піде далі. Щоб уникнути конфліктів при вставці відправляються записи на серверній стороні, викликані перевіркою зовнішніх ключів, у пакет додаються всі необхідні пов'язані дані. По суті, ми додаємо в пакет рядок таблиці, а разом з нею ті рядки з пов'язаних таблиць, які можуть знадобитися для її вставки в ЦБД. Розмір пакета даних обмежується спеціальної налаштуванням - максимальною кількістю самостійних рядків (без урахування пов'язаних), яке він може містити.

Далі, варто сказати, що собою являє пакет даних, і яким чином він може бути відправлений на сервер. Пакет даних - об'єкт класу DataPackage, включає в себе, перш за все, колекцію об'єктів типу BasicEntity, а також деякі службові поля для відправки підтверджень доставки, роль яких незначна. Кожен такий пакет може бути перетворений в формат XML за допомогою механізму серіалізації, а потім збережений в файл або відправлений по мережі, за допомогою web-сервісу. Таким чином, механізм формування пакета не відрізняється для випадку відправки даних online і offline. Також, для обох цих випадків, розроблена система оповіщень про доставку. Відповіддю на порцію даних, отриманих клієнтської або серверної стороною, є значення syncID, яке представляє собою порядковий номер (взятий з таблиці Update) останнього запису завантаженого в базу. Таким чином, клієнт завжди «знає» яка частина його даних відправлена на сервер. Відповідно, записи логу, на які отримано підтвердження про доставку, більше не розглядаються при реплікації та можуть бути видалені. На рис. 3.15 приведена діаграма послідовності, що описує процедуру відправки даних на сервер online.

Звичайно, що механізм відправки даних з сервера будується на тих же принципах, з тією лише відмінністю, що для нього існує безліч адресатів у вигляді ЗЗСО. Для клієнта процес отримання зводиться до завантаження пакетів у базу та відправці підтверджень. Однак, завантаження всіх пакетів, у рамках одного оновлення, для клієнта ведеться в одній транзакції. Тобто, якщо станеться розрив з'єднання на будь-якому етапі, то піде відкат транзакції, незалежно від того скільки пакетів вже завантажено. Для сервера, навпаки, транзакція закривається після отримання кожного пакету.

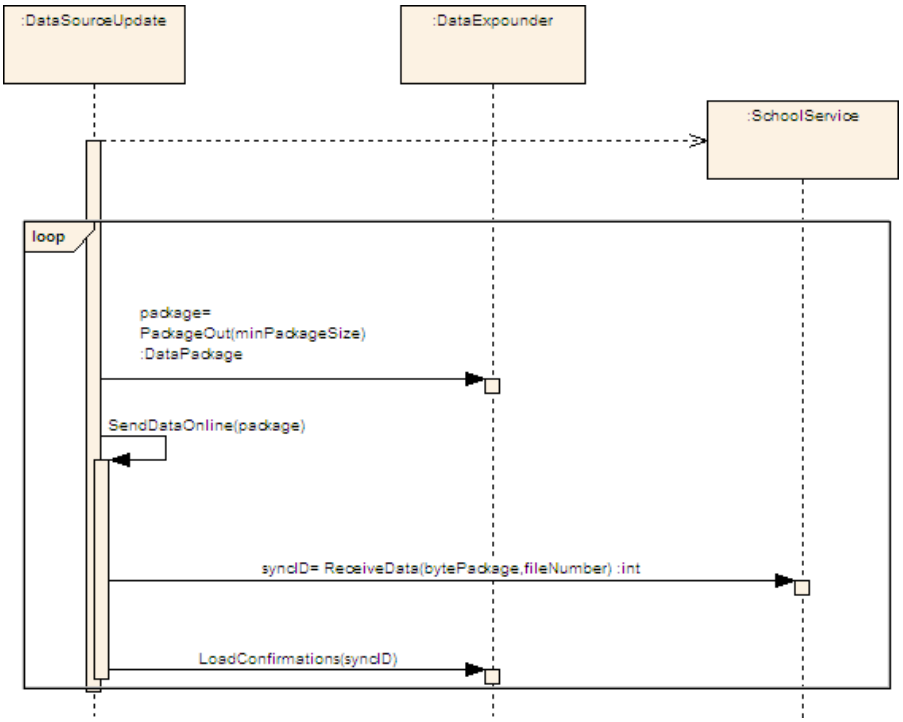


Рисунок 3.15 - Діаграма послідовності, що описує процедуру відправки даних на сервер online

Нижче, приведена діаграма послідовності, що описує механізм отримання даних клієнтом online (рис. 3.16).

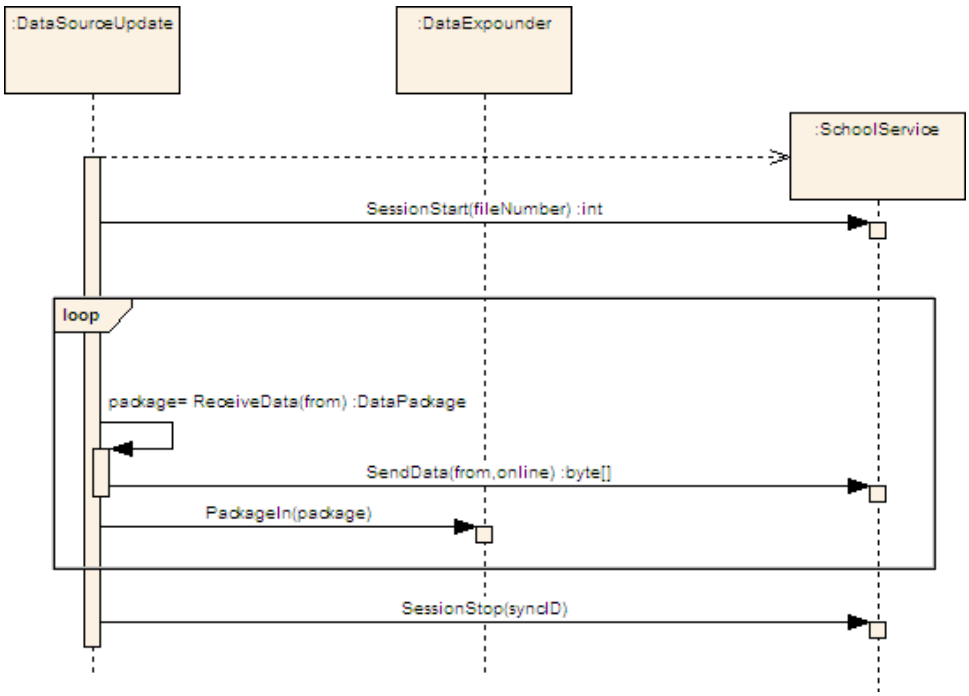


Рисунок 3.16 - Діаграма послідовності, що описує процедуру отримання даних клієнтом online

Як вже було сказано вище, схеми отримання і відправки даних через файли незначно відрізняються від режиму online, тому немає сенсу детально їх описувати. Варто лише сказати, що підтвердження про доставку не відправляються окремими файлами, а включаються в заголовок основного повідомлення, що містить пакети з даними. Таким чином, відправляючи актуальні дані на сервер, клієнтська програма в той же файл вивантажує інформацію про своє останнє оновлення з сервера, неважливо online або offline.

Передаючи дані по мережі, необхідно бути впевненим, що їх конфіденційність буде збережена. Тому, не останнє місце в розробці модуля реплікації даних займає блок шифрування. Клієнт і сервер ведуть обмін тільки зашифрованими даними, як у варіанті online, так і offline. Основою є стандартна схема - повідомлення шифрується симетричним ключем, який, в свою чергу, шифрується алгоритмом RSA і дописується в заголовок повідомлення. У якості алгоритму симетричного шифрування обраний шифр Rijndael, переможець конкурсу AES (Advanced Encryption Standard), який на сьогоднішній день використовується в великому числі криптографічних додатків. Таким чином, для роботи з сервером, кожному клієнтському запиту генерується пара ключів RSA, які забезпечують не тільки безпеку передачі даних, а й унікальність визначення школи. Тобто, кожен клієнт, під яким би ID він не ідентифікувався на сервері, не зможе правильно інтерпретувати передані йому дані, не маючи закритого ключа. Для збільшення надійності алгоритму шифрування і зменшення трафіку, при передачі оновлень по мережі, кожне повідомлення стискається алгоритмом zip, безпосередньо перед шифруванням. Використання поточних алгоритмів стиснення і шифрування дозволяє уникнути зайвих витрат пам'яті і підвищує загальну ефективність процесу реплікації. Для вирішення поставлених завдань були використані стандартні модулі стиснення і криптографії, що входять до складу бібліотек .NET Framework 2.0.

3.2.4 Особливості схеми бази даних

Як говорилося вище, для реалізації клієнтської бази даних була використана СУБД MS SQL Server Express 2016, що дозволила в повній мірі здійснити поставлені завдання. Варто сказати, що за своєю структурою, серверна БД, мало чим відрізняється від клієнтської, і відмінність складають лише кілька службових таблиць. Тому обмежимося описом схеми тільки БД клієнтської частини програми. Так як система спочатку розроблялася як розподілена, з можливістю асинхронної реплікації, то в якості сурогатних первинних ключів для всіх таблиць використовуються значення типу uniqueidentifier (guid). Такий підхід дозволяє значно спростити генерацію первинних ключів для вставляються записів, так як не доводиться дбати про їхню унікальність в межах всіх реплікованих БД. Дана особливість схеми врахована при розробці об'єктної моделі: так кожен об'єкт будь-якого класу, успадкованих від BasicEntity, вже має властивість ID, для доступу до первинного ключа. А це в свою чергу, дозволило перевантажити оператори порівняння для

об'єктів, що працюють з таблицями БД.

Далі, для прикладу, наведемо деякі фрагменти схеми БД, що описують ті чи інші частини предметної області. Найпростіша і в той же час необхідна складова - це надання інформації про працівників ЗЗСО, що навчаються у самій школі, а також їх діяльності. Сюди входять, паспорт школи, картки вчителів і учнів, предмети навчальні плани, класи і т.д. На рис. 3.17 зображений фрагмент схеми БД, що описує організацію викладачів і учнів в рамках ЗЗСО.

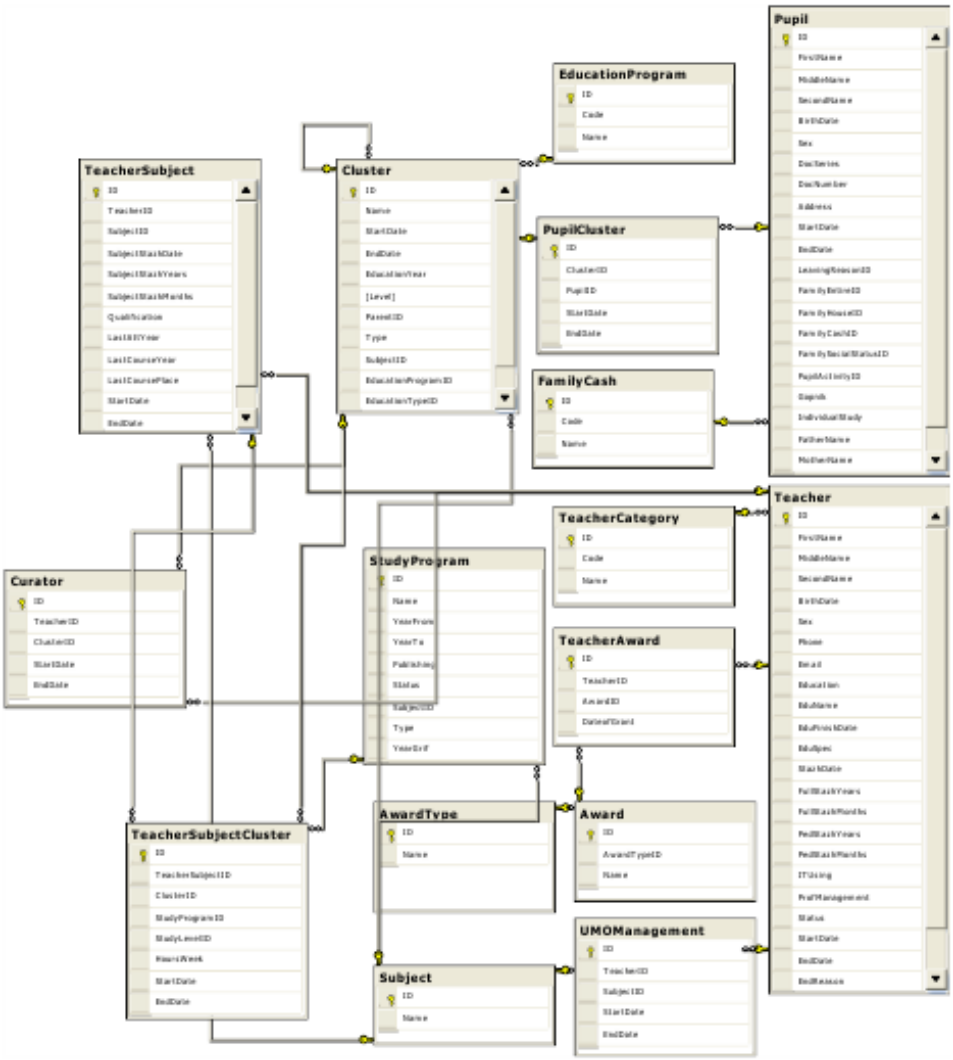


Рисунок 3.17 - Фрагмент схеми БД, що описує організацію учнів та викладачів

В цілому, призначення таблиць зрозуміло з їх назви, проте, слід більш детально зупинитися на понятті, кластера, яке не так очевидно. Кластер - будь-яке об'єднання учнів, тимчасове або постійне. Сукупність кластерів являє собою ієрархічну структуру, причому учні можуть бути прив'язані до будь-якого кластеру, незалежно від його положення в дереві. Наприклад, кластерами є навчальні паралелі, класи, підгрупи будь-яких рівнів і т.д. Вся інформація про кластери

поміщена в таблицю Cluster, а прив'язка до них учнів здійснюється через таблицю PupilCluster (зв'язок «багато-до-багатьох»). Оскільки будь-який кластер в ієрархії пов'язаний з учнями, які в ньому навчаються (а не тільки листя), у будь-який момент можна з однаковою легкістю отримати всіх учнів першої паралелі, учнів 1А класу або учасників шахової секції.

Іншою особливістю таких сутностей, як кластер, вчитель, учень і багатьох інших, які представлені на фрагменті схеми, є наявність у них полів StartDate і EndDate. Ці поля необхідні для завдання періодів навчання (для учнів), викладання (для вчителів) та існування (для інших сутностей і зв'язків). Справа в тому, що при видаленні, наприклад, учня, через інтерфейс клієнтської програми, це не призводить до його повного видалення з бази. Відповідним записом, а якщо необхідно і пов'язаним з ним, проставляється дата закінчення існування (EndDate). Усі записи, періоди існування, які не узгоджуються з робочою датою, не відображаються користувачу. Таким чином, кожний ЗЗСО зберігає історію зміни даних про його викладачів, учнів, кластерів та т.і. Скориставшись механізмом зміни робочої дати, можна відстежити ці зміни.

Іншою важливою задачею клієнтської частини програми, є проведення ОЗ та збір статистики по ним. У розділі, що описує можливості об'єктної моделі, вже був приведений невеликий фрагмент схеми, що відображає дану функціональність. На рис. 3.18 приведена докладніша схема.

Дана схема дозволяє реалізувати стандартний функціонал для такого роду систем: автоматичне виставлення оцінок з використанням процентних та бальних шкал, створення і проведення ОЗ з різнотипними питаннями і т.і. Окремо варто сказати про кодифікатори, які представляють собою ієрархічно вибудувані підрозділи предметів. Кодифікатори можуть бути двох типів - кодифікатори знань (таблиця Codifier, рис. 3.18) і умінь (таблиця SkillsCodifier, рис. 3.18). Основне дерево кодифікатор для кожного предмета потрапляє в базу з сервера, при реплікації, проте, самі користувачі клієнтського додатка можуть розширювати його і доповнювати. Пункти кодифікатор обох типів прив'язуються до кожного питання зі шкільної бази (це обов'язкові поля). Такий підхід дозволяє проводити більш детальний аналіз самих питань і відповідей учнів на них.

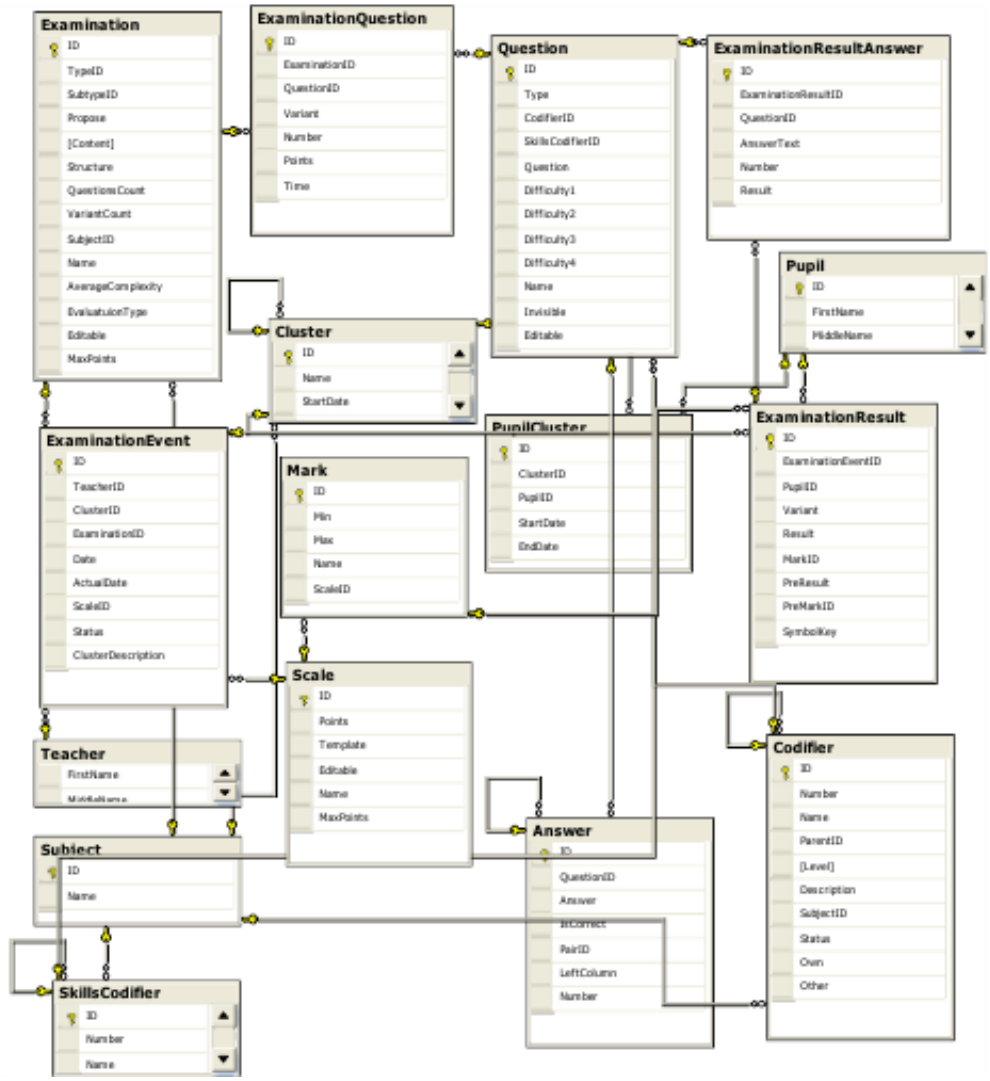


Рисунок 3.18 - Фрагмент схеми БД, що описує механізм проведення ОЗ

У даному розділі представлені далеко не всі фрагменти, з яких складається БД клієнтської програми. Залишилися неохопленими позанавчальні ОЗ, маса спільних довідників, функціональність, пов'язана з веденням паспорта школи і т.і. Однак, з наведеного опису можна отримати загальне уявлення про підхід, що використовувався при проектуванні схеми БД.

3.3 Висновки до розділу 3

1. Спроектована архітектура автоматизованої регіональної інформаційної системи моніторингу якості освіти, що забезпечує збір, обробку та інтерпретацію інформації про результати освітньої діяльності.
2. Реалізовано підсистеми доступу до даних та реплікації з ЦБД.
3. Реалізовано модуль інтеграції системи з АСК «Символ».
4. Створено робочий прототип клієнтської частини.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

У даному розділі проведено аналіз потенційних небезпечних і шкідливих виробничих факторів, причин пожеж. Розглянуто заходи, які дозволяють забезпечити гігієну праці та виробничу санітарію. На підставі аналізу розроблено заходи з техніки безпеки і рекомендації з пожежної профілактики.

Завданням даної роботи магістра було розглянути моделі та програмні засоби розробки клієнтської частини комп’ютерної системи моніторингу якості освіти. Так як в процесі проектування використовувався ПК, то аналіз потенційно небезпечних і шкідливих виробничих факторів виконується для робочого місця з ПК, на якому буде використовуватися розроблений засіб.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником,а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. У законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності.

4.2 Аналіз стану умов праці

Умови праці повинні задовольняти таким вимогам, які дали б можливість людині виконувати роботу без шкоди для здоров'я, без перевтоми і з високою продуктивністю. Для вибору показників умов праці при проведенні аналізу слід керуватися чинними в Україні "Правилами охорони праці під час експлуатації електронно-обчислювальних машин" ДНАОП 0.00-1.31-99 [19].

Робота з засобом комп’ютерною системою буде проходити в приміщенні Центру моніторингу якості освіти. Для даної роботи досить однієї людини, для якого надано робоче місце зі стаціонарним комп'ютером. Виконувана робота за ступенем тяжкості відноситься до категорії "легка 1б". До неї відносяться роботи, вироблені сидячи, стоячи або пов'язані з ходьбою, але не потребують систематичного фізичного напруження чи підняття і перенесення важких предметів.

4.2.1 Вимоги до приміщень

Згідно з [20] розмір площі для одного робочого місця оператора персонального комп'ютера повинно бути не менше 6 кв. м, а обсяг - не менше 20 куб. м. Отже, дане приміщення повністю відповідає зазначеним нормам.

4.2.2 Вимоги до організації місця праці

Робочий стіл на досліджуваному місці також містить досить простору для ніг. Крісло, використовується в якості робочого сидіння, є підйомно поворотним, має підлокітники і можливість регулювання по висоті і куту нахилу спинки, також воно м'яке і виконано з екологічної шкіри, що дозволяє працювати в комфорті. Екран монітора знаходиться на відстані 0.8 м, клавіатура має можливість регулювання кута нахилу 5-15°С. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам. Приміщення кабінету знаходиться на другому поверсі трьох поверхового будинку і має обсяг 78 м^3, площа - 24 м^2. У цьому кабінеті обладнано три місця праці, з яких два укомплектовані ПК.

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5. Є аптечка для надання долікарської допомоги, а також в кабінеті роблять вологе прибирання і щодня провітрюють приміщення.

4.3 Виробнича санітарія

На підставі аналізу небезпечних і шкідливих факторів при експлуатації, пожежної безпеки можуть бути в подальшому вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення тощо [21].

4.3.1 Аналіз небезпечних і шкідливих факторів при експлуатації системи

Аналіз небезпечних і шкідливих виробничих факторів виконується в табличній формі (табл. 4.1).

Таблиця 4.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[22]
- підвищений рівень шуму на робочому місці	-//-	2	[23]
- підвищений рівень вібрації	-//-	2	[23] [24]
- підвищена або знижена вологість повітря	-//-	2	[22]
- підвищена або знижена рухливість повітря	-//-	1	[22]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[22] [25]
- підвищений рівень електромагнітного випромінення	-//-	2	[25]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[26] [27]
- підвищений рівень статичної електрики	-//-	2	[27]
- підвищена напруженість електричного поля	-//-	2	[27]
- підвищена напруженість магнітного поля	-//-	2	[27]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[28]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[28]
- підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[28]
- понижена контрастність	-//-	1	[20]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів;	4	[20] [29]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безпервна робота)	2	[20] [29]

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень - це клімат внутрішнього середовища цих приміщень, який визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт Іа. Отже оптимальні значення для температури, відносної вологості і рухливості повітря для зазначеного робочого місця відповідають [21] і наведені в табл. 4.2:

Таблиця 4.2 – Норми мікроклімату робочої зони об'єкта

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

4.4.2 Освітленість

Розрахунок освітлення.

Для виробничих і адміністративних приміщень світловий коефіцієнт приймається не менш -1/8, в побутових - 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$Sn = a \cdot b = 4 \cdot 6 = 24 \text{ м}^2, \tag{4.2}$$

$$S = 1/8 \cdot 24 = 3 \text{ м}^2.$$

Приймаємо 2 вікна площею S=3 м^2 кожне.

Розрахунок штучного освітлення проводиться за коефіцієнтами використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n проводиться за формулою (4.3):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \tag{4.3}$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 24 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання і ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, що залежить від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення в формулу (5.2), отримуємо:

$$n = \frac{300 * 24 * 1.1 * 1.5}{5400 * 0.575 * 2} \approx 1.9 \quad (4.4)$$

4.5 Заходи з організації виробничого середовища і попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Відповідно до класифікації приміщень за ступенем небезпеки ураження електричним струмом [19], приміщення в якому проводяться всі роботи належить до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, і 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \quad (4.5)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення в формулу (4.5), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом} \quad (4.6)$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом • м. Приблизне значення питомої опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахункова питомий опір ґрунту, $\rho_{розр}$, Ом•м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом•м по формулі:

$$\rho_{розр..} = \psi \cdot \rho, \quad (4.7)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом•м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м} \quad (4.8)$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлення $R_в$, Ом, по (4.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.9)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стрижня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (5.10):

$$t = h_в + \frac{l_в}{2}, \quad (4.10)$$

де $h_в$ – глибина закладення вертикальних заземлювачів (0,8 м); тоді

$$t = 0,8 + \frac{3}{2} = 2,3 \text{ м} \quad (4.11)$$

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом} \quad (4.12)$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.13)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлення без урахування впливу сполучної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.14)$$

7) Визначається довжина сполучної стрічки горизонтального заземлювача $l_с$, м:

$$l_c = 1,05 \cdot L_в \cdot (n_в - 1), \quad (4.15)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти $L_в$ = 3м);

$n_в$ – необхідну кількість вертикальних заземлювачів.

$$l_c = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м} \quad (4.16)$$

8) Визначається опір розтіканню струму горизонтального заземлювача (сполучної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_c} \cdot \ln \frac{2 \cdot l_c^{\,2}}{d_{см} \cdot h_г}, \quad (4.17)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладення горизонтальних заземлювачів (0,5 м);

l_c - довжина сполучної стрічки горизонтального заземлювача l_c, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом} \quad (4.18)$$

9) Визначається коефіцієнт використання горизонтального заземлювача ηс відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання сполучної смуги η_c=0,3 (табличне значення).

10) Розраховується результуючий опір заземлюючого електрода з урахуванням сполучної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_c + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.19)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпека будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д \quad (4.20)$$

У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки щодо запобігання можливих наслідків та усунення пожежі [19].

4.6 Охорона навколишнього природного середовища

4.6.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: розгляд моделей та програмних засобів розробки клієнтської частини комп'ютерної системи, у процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [30], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [31], Законом України «Про відходи» [32], Законом України «Про охорону атмосферного повітря» [33], Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру» [34], Водний кодекс України [35].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на видалення (знешкодження), утилізацію, тощо в ІТ галузі.

В процесі діяльності комп'ютерної системи виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- відпрацьовані люмінесцентні лампи - І клас небезпеки
- батарейки та акумулятори (малі) -ІІІ клас небезпеки
- змінні носії інформації - IV клас небезпеки
- відходи друкуючих пристроїв - IV клас небезпеки
- макулатура - IV клас небезпеки
- матеріали пакувальні пластмасові забруднені (ємності з-під тонеру, фарби, інш.) - IV клас небезпеки
- побутові відходи - IV клас небезпеки

4.6.2 Вимоги до збору, пакування та розміщення відходів ІТ галузі

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм [36].

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- відходи I класу небезпеки зберігаються в герметичній тарі (сталеві бочки, контейнери). У міру наповнення тару з відходами закривають герметично сталевий кришкою;

- відходи II класу небезпеки в залежності від агрегатного стану зберігаються в поліетиленових мішках, бочках, сховищах та інших видах тари, яка запобігає поширенню шкідливих речовин;

- відходи III класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

- відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і інш.

Особливий контроль наділяється збору і зберіганню відпрацьованих ртутьвмісних ламп (енергоощадних) як відходам I класу небезпеки, що збираються і обов’язково передаються на утилізацію підприємствам, що мають ліцензію на поводження з такими небезпечними відходами.

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Відвантаження таких відходів здійснюється відповідно до договору (контракту)

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.6.3 Визначення впливу та заходів щодо поводження з відходами IT галузі

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» [32] повинен здійснюватися моніторинг місць утворення, зберігання, і видалення

відходів. Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації / виробництва».

4.7 Висновки до розділу 4

У результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, наведено інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

Розроблювана система забезпечить збір, обробку та інтерпретацію інформації про результати освітньої діяльності. Реалізована система реплікації даних прискорить збір інформації та забезпечить поширення загальних довідників в ЗЗСО. Більшість необхідних довідників вже зберігається на сервері, також розроблені механізми імпорту з excel для деяких з них, що забезпечує легкий механізм додавання нових даних.

Реалізовано механізм імпорту результатів ОЗ з АСК "Символ", що дозволяє не тільки розробляти оціночні заходи, але і швидко оцінювати отримані результати, при цьому самі відповіді учнів також зберігаються, що дозволяє оцінити коректність складених питань і самого оціночного заходи в цілому.

У перспективі система повинна бути доповнена аналізом даних ЦБД, підбиттям статистики. Має бути доопрацьований механізм поновлення, що підтримує не тільки оновлення даних, а й оновлення самої системи, а також формування дистрибутивів з останньою версією клієнта.

На даному етапі розробки отримані наступні результати:

- проаналізована предметна область;
- спроектована архітектура;
- реалізовано підсистеми доступу до даних та реплікації з центральної БД;
- реалізовано модуль інтеграції системи з АСК «Символ»;
- створено робочий прототип клієнтської частини.

Під час побудови архітектури було подолано значне число технічних ризиків, а при реалізації вибудувана стійка базова платформа для наповнення системи розширеною функціональністю. Таким чином, можна сказати, що перша фаза життєвого циклу проекту може бути успішно закрита, після створення остаточної версії інтерфейсу користувача і введення системи в дослідну експлуатацію.

Виконано аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник, визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Височина Н.О. Програмні засоби моніторингу якості освіти. // Збірник науково-практичних праці ІІІ молодіжного форуму «ІТ-ідея 2017». – Сєвєродонецьк. – 2017. – С. 54-55.
2. Височина Н.О., Шумова Л.О., Міняйло В.А. Комп'ютерні засоби розробки інформаційної системи моніторингу якості освіти. // Вісник Східноукраїнського національного університету ім. В. Даля. – 2018. – Вип №6(247). – С. 27-30.
3. Костенко К.И. Моделирование информационной системы оценки качества образования// Университетское управление: практика и анализ. – 2003. –№ 3(26). С. 77-83.
4. Макєєв С. Ю. Використання інформаційно-комунікаційних технологій у системі початкової освіти / С. Ю. Макєєв // Педагогіка та психологія. – 2011. – Вип. 40(2). – С. 97-102. – URL: http://nbuv.gov.ua/UJRN/znpkhnpu_ped_2011_40(2)__18.
5. Биков В.Ю. Сучасні завдання інформатизації освіти / В.Ю. Биков // Інформаційні технології і засоби навчання. – 2010. – №1 (15). – URL: http://www.ime.edu-ua.net/em/html
6. Бочаров Б.П., Воєводіна М.Ю. Інформаційні технології в освіті / Б.П. Бочаров, М.Ю. Воєводіна // Монографія. – Харків, 2015. – 197 с.
7. Титаренко Н.В. Використання комп'ютерних технологій під час моніторингу якості освіти / Н.В. Титаренко // Матеріали міжнародної науково-практичної конференції «Інформаційно-комунікаційні технології навчання». – Умань, 2008 – с. 169-171.
8. Пліш І.В. Використання інформаційно-комунікаційних технологій управління якістю освіти в загальноосвітніх навчальних закладах / І.В. Пліш // Дисертація канд. пед. наук. – Київ, 2012 – 200 с.
9. Moodle в Україні. [Електронний ресурс]. – URL: https://moodle.org/course/view.php?id=17228
10. Анисимов А.М. Работа в системе дистанционного обучения Moodle // Учебное пособие. – 2009.
11. Google for education. [Електронний ресурс]. – URL: https://edu.google.com/?modal_active=none
12. Компьютерная программа тестирования знаний Open TEST 2.0. [Електронний ресурс]. – URL: http://opentest.com.ua/kompyuternaya-programma-testirovaniya-znanij-opentest-2/
13. Компьютерная система тестирования знаний с украинского языка по программе Open TEST 2.0. [Електронний ресурс]. – URL: http://www.info-library.com.ua/libs/stattya/3884-kompjuterna-sistema-testuvannja-znan-z-ukrayinskoyi-movi-za-programoju-open-test-20.html
14. Программа для создания тестов и онлайн тестирования. [Електронний ресурс]. – URL: https://indigotech.ru/
15. Что такое ABBYY FormReader и как он работает. [Електронний ресурс]. – URL:

https://habr.com/company/abbyy/blog/107958/

16. Microsoft SQL Server. [Електронний ресурс]. – URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server

17. SPSS. [Електронний ресурс]. – URL: https://uk.wikipedia.org/wiki/SPSS

18. Білик О.О. Кваліативні моделі загальноосвітнього навчального закладу / В.А. Лужецький, О.О.Білик, В.М.Заячковський // Інформаційні технології та комп'ютерна інженерія. - 2007. - №1(8). – С. 153-163.

19. ДНАОП 0.00-1.31-99 Правила охорони праці під час експлуатації електронно-обчислювальних машин.

20. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

21. НАПБ Б.02.005-2003 Типове положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України.

22. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень.

23. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку.

24. ДСТУ ГОСТ 12.1.012-2008 Система стандартів безпеки праці. Вібраційна безпека. Загальні вимоги.

25. ГОСТ 12.1.006-84 Правила охорони праці під час оброблення і використання алюмінієвих і титанових сплавів.

26. ГОСТ 13109-97 Норми якості електричної енергії в системах електропостачання загального призначення.

27. ГОСТ 12.1.030-81 Система стандартів безпеки праці. Електробезпека. Захисне заземлення. Занулення.

28. ДБН В.2.5-28:2015 Освітлення у примішеннях.

29. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин.

30. Закон України «Про охорону навколишнього природного середовища».

31. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення».

32. Закон України «Про відходи».

33. Закон України «Про охорону атмосферного повітря».

34. Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру».

35. Водний кодекс України.

36. ДСанПіН 2.2.7.029 Гігієнічні вимоги щодо поводження з промисловими відходами та визначення їх класу небезпеки для здоров'я населення.

ДОДАТОК А. Лістинг програми

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.ComponentModel;
using System.Collections;
using Fields = DataSource.XmlSerializableDictionary<string, DataSource.Field>;
using System.Xml.Serialization;
using System.IO;

namespace DataSource
{
   public delegate bool Comparer(BasicEntity row, string filter); public delegate void
ModificationEventWithCancel(BasicEntity sender,
   ModificationEventArgsWithCancel e);
   public delegate void ErrorEvent(BasicEntity sender, ErrorEventArgs e);
   public abstract class BasicEntity : IXmlSerializable
   {
      public enum Navigation
      {
         First = 0, Next
      }
      public event ModificationEventWithCancel OnBeforeModification; public event
EventHandler<ModificationEventArgsWithoutCancel>
      OnAfterModification;
      private Hashtable _dbTypeTable; public Comparer Comparer; private int _timeout = 70; protected string _idFieldName;
      private SqlConnection _connection; private SqlTransaction _transaction; private SqlTransaction _innerTransaction;
protected Fields _fields;
      protected List<string> _serviceFields; protected string _tableName;
      protected string _what; protected string _filter; protected string _order;
      protected ReplicationState _replicationState = ReplicationState.None; protected bool _logging;
      private object _tag;
      private LoggingType _loggingType;
      private bool _caching;
      public SqlConnection Connection
      {
         get
         {
         }
         set
return _connection;
         {
            _connection = value;
         }
         }
public SqlTransaction Transaction
      {
         get
         {
            if (IsTransactionValid(_transaction)) return _transaction;
         }
         set
         {
 else
 return DataSourceManagment.Transaction;
            _transaction = value;
         }
      }
      public int Timeout
      {
         get
         {
```

```
            }
            set
            {
            }
        }
 return _timeout;

_timeout = value;
 public bool Logging
        {
            get
            {
            }
            set
            {
            }
        }
 return _logging;
_logging = value;
 public LoggingType LoggingType
        {
           get
            {
            }
            set
            {
            }
        }
 return _loggingType;
_loggingType = value;
 public object this[string name]
        {
            get
            {
            }
 return Requisites(name).Value;
 set
{
 if (value == null)
value = DBNull.Value;
if (!_fields.ContainsKey(name))
{
 _fields.Add(name, new Field(value));
_fields[name].Update = true;
}
if (!value.Equals(_fields[name].Value))
{
_fields[name].Value = value;
_fields[name].Update = true;
if (_fields[name].ValueType == DBNull.Value.GetType())
_fields[name].ValueType = value.GetType();
if (name == _idFieldName)
foreach (Field field in _fields.Values) field.Update = true;
}
}
}
public object this[int index]
{
    get
    {
    }
    set
    {
    }
}
```

```
 return this[RequisitesNames(index)];
this[RequisitesNames(index)] = value;
public string TableName
{
    get
    {
    }
}
 return _tableName;
 public string PrimaryKeyName
{
    get
    {
    }
}
 return _idFieldName;
 public bool Caching
{
    get
    {
    }
    set
    {
        return _caching;
        _caching = value;
    }
}
public string Filter
{
    get
    {
    }
    set
    {
    }
}
 return _filter;
_filter = value;
 public string Order
{
    get
    {
    }
    set
    {
    }
}
 return _order;

_order = value;
 public object Tag
{
    get
    {
    }
    set
    {
    }
}
 return _tag;
_tag = value;
 public Guid ID
{
    get
    {
```

```
    }
    set
    {
    }
}
 return (Guid) Requisites(_idFieldName).Value;
Assignment(_idFieldName, value);
protected BasicEntity(SqlConnection connection)
{
    Connection = connection;
    _idFieldName = "ID"; Caching = false; Logging = true;
    Order = Filter = null;
    _what = "*";
    _serviceFields = new List<string>();
    _serviceFields.Add("_RowNumber_");
    _fields = new Fields(); LoggingType = LoggingType.Simple;
}
protected Field Requisites(string name)
{
    Field requisite;
    if (_fields.TryGetValue(name, out requisite)) return requisite;
    found.");
}
 else
 throw new Exception("Requisite with specified name not
public string RequisitesNames(int index)
{
    found.");
}
 if ((index< 0) || (index >= _fields.Keys.Count))
throw new Exception("Requisite with specified index not
string[] fieldsNames = new string[_fields.Keys.Count];
_fields.Keys.CopyTo(fieldsNames, 0); return fieldsNames[index];
 public bool HasRequisite(string name)
{
    return _fields.ContainsKey(name);
}
public object RequisitesValues(int index)
{
    return this[index];
}
public int RequisitesCount()
{
    return _fields.Count;
}
public void Clear()
{
    _fields.Clear();
    Order = null;
    Filter = null;
}
#region Queries
public virtual void Insert()
{
    if (!_fields.ContainsKey(_idFieldName))
        _fields.Add(_idFieldName, new Field(Guid.NewGuid()));
    if (!UserProceed(BasicEntityModification.Insert, true)) return; using (SqlCommand command = new SqlCommand())
    {
        StringBuilder query = new StringBuilder(); query.AppendFormat("INSERT INTO {0} (", _tableName);
        StringBuilder parameters = new StringBuilder(); parameters.Append(" VALUES (");
        foreach (string key in _fields.Keys)
        {
            if (_fields[key].Value as Guid? == Guid.Empty)
                _fields[key].Value = DBNull.Value;/////////
            if (_fields[key].Value == DBNull.Value) continue;
```

```
        query.AppendFormat("{0}, ", key); parameters.AppendFormat("@{0}, ", key); command.Parameters.Add(new
SqlParameter(key,
        Convert.ChangeType(_fields[key].Value, _fields[key].ValueType)));
        _fields[key].Update = false;
      }
      query.Remove(query.Length - 2, 2);
      parameters.Remove(parameters.Length - 2, 2); query.AppendFormat("){0})", parameters.ToString());
command.CommandTimeout = Timeout; command.CommandText = query.ToString(); command.Connection = Connection;
command.Transaction = Transaction; SafeConnectionOpen();
      InnerTransactionOpen(command); QueryLog(command.CommandText); command.ExecuteNonQuery();
      if (Logging) Log(ReplicationState.Create); InnerTransactionCommit();
      SafeConnectionClose(); UserProceed(BasicEntityModification.Insert, false);
   }
   Get(ID);
}
public virtual void Update()
{
   if (!UserProceed(BasicEntityModification.Update, true)) return; using (SqlCommand command = new SqlCommand())
   {
      StringBuilder query = new StringBuilder(); query.AppendFormat("UPDATE {0} SET ", _tableName); bool isUpdate =
false;
      foreach (string key in _fields.Keys)
      {
         if (_fields[key].Update && (key != _idFieldName))
         {
            if (_fields[key].Value as Guid? == Guid.Empty)
               _fields[key].Value = DBNull.Value;/////////
            if (_fields[key].ValueType == typeof(DBNull))
               throw new Exception("You must retrieve table row
               before updating.");
               query.AppendFormat("{0} = @{0}, ", key); if (_fields[key].Value != DBNull.Value)
               command.Parameters.Add(new SqlParameter(key,
               Convert.ChangeType(_fields[key].Value, _fields[key].ValueType)));
            else
            {
               command.Parameters.Add(new SqlParameter(key, GetDBType(_fields[key].ValueType)));
               command.Parameters[key].Value =
               _fields[key].Value;
            }
         }
      }
      _fields[key].Update = false; isUpdate = true;
      if (!isUpdate) return;
      query.Remove(query.Length - 2, 1); query.AppendFormat("WHERE ({0} = @ID)", _idFieldName);
command.CommandText = query.ToString(); command.CommandTimeout = Timeout;
      command.Connection = Connection; command.Transaction = Transaction; command.Parameters.Add(new
SqlParameter("ID", ID)); SafeConnectionOpen(); InnerTransactionOpen(command); QueryLog(command.CommandText);
command.ExecuteNonQuery();
      if (Logging) Log(ReplicationState.Update); InnerTransactionCommit(); SafeConnectionClose();
      UserProceed(BasicEntityModification.Update, false);
   }
}
public virtual void Delete()
{
   if (!UserProceed(BasicEntityModification.Delete, true)) return; StringBuilder query = new StringBuilder();
query.AppendFormat("DELETE FROM {0} WHERE {1} = @ID", _tableName,
   _idFieldName);
   using (SqlCommand command = new SqlCommand(query.ToString(),
   Connection))
   {
   }
   command.Parameters.Add(new SqlParameter("ID", ID)); command.Connection = Connection; command.CommandTimeout
= Timeout; command.Transaction = Transaction; SafeConnectionOpen();
   InnerTransactionOpen(command);
QueryLog(command.CommandText); command.ExecuteNonQuery();
```

```
   if (Logging) Log(ReplicationState.Delete); InnerTransactionCommit();
   SafeConnectionClose();
   _fields = new Fields(); UserProceed(BasicEntityModification.Delete, false);
}
public virtual bool First()
{
   bool found = false; SafeConnectionOpen();
   using (SqlDataReader dataReader = GetFirstNRowsReader(1))
   {
      if (dataReader.HasRows)
      {
         dataReader.Read();
         _fields = GetFields(dataReader); found = true;
      }
   }
   SafeConnectionClose(); return found;
}
public virtual bool Next()
{
   bool found; SafeConnectionOpen();
   using (SqlDataReader dataReader = GetNextNRowsReader(1))
   {
      if (dataReader.HasRows)
      {
         dataReader.Read();
         _fields = GetFields(dataReader); found = true;
      }
      else
      {
         found = false;
      }
   }
   SafeConnectionClose(); return found;
}
public virtual bool Get(Guid id)
{
   if (Caching)
   {
      Connection);
      Fields fields = Hive.Get(TableName, _idFieldName, id,
   if (fields != null)
      {
         _fields = fields; return true;
      }
   }
   bool found = false;
   StringBuilder query = new StringBuilder(); query.AppendFormat("SELECT {2} FROM {0} WHERE {1} = @ID",
   _tableName, _idFieldName, _what);
   using (SqlCommand command = new SqlCommand())
   {
      command.Parameters.Add(new SqlParameter("ID", id)); command.CommandText = query.ToString();
command.Connection = Connection;
      command.CommandTimeout = Timeout; command.Transaction = Transaction; SafeConnectionOpen();
QueryLog(command.CommandText);
      using (SqlDataReader dataReader = command.ExecuteReader())
      {
         if (dataReader.HasRows)
         {
            dataReader.Read();
            _fields = GetFields(dataReader); found = true;
         }
      }
      SafeConnectionClose();
   }
   return found;
```

```
}
public virtual bool Get()
{
   return Get(ID);
}
protected SqlDataReader GetFirstNRowsReader(int count)
{
   StringBuilder query = new StringBuilder(); if (count > 0)
      query.AppendFormat("SELECT TOP({0}) {1} FROM {2}", count,
      _what, _tableName);
   else
      query.AppendFormat("SELECT {0} FROM {1}", _what, _tableName);
   if (!String.IsNullOrEmpty(Filter)) query.AppendFormat(" WHERE ({0})", Filter);
   if (!String.IsNullOrEmpty(Order)) query.AppendFormat(" ORDER BY {0}", Order);
   using (SqlCommand command = new SqlCommand(query.ToString(),
   Connection))
   {
   }
   command.Transaction = Transaction; command.CommandTimeout = Timeout; QueryLog(command.CommandText); return
command.ExecuteReader();
}
protected SqlDataReader GetNextNRowsReader(int count)
{
   if (count < 0)
      throw new Exception("Count value may not be negative."); using (SqlCommand command = new SqlCommand())
   {
      StringBuilder query = new StringBuilder(); query.Append("WITH _OrderedTable_ AS ");
query.AppendFormat("(SELECT {0}, ROW_NUMBER() OVER (ORDER BY
      { 1}) AS { 2}
      ", _what, String.IsNullOrEmpty(Order) ? _idFieldName : Order,
      _serviceFields[0]);
      if (String.IsNullOrEmpty(Filter))
         Filter);
      query.AppendFormat(" FROM {0}) ", _tableName); else
query.AppendFormat(" FROM {0} WHERE {1}) ", _tableName,
query.AppendFormat("SELECT TOP({0}) * ", count); query.Append("FROM _OrderedTable_ ");
query.AppendFormat("WHERE _RowNumber_       > (SELECT _RowNumber_
FROM _OrderedTable_ WHERE { 0} = @ID) ", _idFieldName); query.Append("ORDER BY _RowNumber_");
command.CommandText = query.ToString(); command.Connection = Connection; command.CommandTimeout = Timeout;
command.Transaction = Transaction; command.Parameters.Add(new SqlParameter("ID", ID));
QueryLog(command.CommandText);
      return command.ExecuteReader();
   }
}
public virtual int Count()
{
   Connection))
 StringBuilder query = new StringBuilder(); query.AppendFormat("SELECT COUNT(*) FROM {0}", _tableName); if
(!String.IsNullOrEmpty(Filter))
      query.AppendFormat(" WHERE ({0})", Filter);
   using (SqlCommand command = new SqlCommand(query.ToString(),
   {
      command.CommandTimeout = Timeout; command.Transaction = Transaction; SafeConnectionOpen();
QueryLog(command.CommandText);
   int count = (int)command.ExecuteScalar(); SafeConnectionClose();
   return count;
}
}
public virtual int DeleteAll()
{
   0;
   Connection))
 if (!UserProceed(BasicEntityModification.DeleteAll, true)) return
StringBuilder query = new StringBuilder(); query.AppendFormat("DELETE FROM {0}", _tableName); if
(!String.IsNullOrEmpty(Filter))
```

```
        query.AppendFormat(" WHERE ({0})", Filter);
    using (SqlCommand command = new SqlCommand(query.ToString(),
    {
        command.CommandTimeout = Timeout; command.Transaction = Transaction; SafeConnectionOpen();
    InnerTransactionOpen(command);
    if (Logging) LogAll(ReplicationState.Delete); QueryLog(command.CommandText);
    int count = command.ExecuteNonQuery();
    InnerTransactionCommit();
    SafeConnectionClose(); UserProceed(BasicEntityModification.DeleteAll, false); return count;
}
}
public virtual int UpdateAll(string arguments)
{
    0;
    Connection))
 if (!UserProceed(BasicEntityModification.UpdateAll, true)) return
StringBuilder query = new StringBuilder(); query.AppendFormat("UPDATE {0} SET {1}", _tableName, arguments); if
(!String.IsNullOrEmpty(Filter))
        query.AppendFormat(" WHERE ({0})", Filter);
    using (SqlCommand command = new SqlCommand(query.ToString(),
    {
        command.CommandTimeout = Timeout; command.Transaction = Transaction; SafeConnectionOpen();
    InnerTransactionOpen(command);
    if (Logging) LogAll(ReplicationState.Update); QueryLog(command.CommandText);
    int count = command.ExecuteNonQuery(); InnerTransactionCommit();
    SafeConnectionClose(); UserProceed(BasicEntityModification.UpdateAll, false); return count;
}
}
#endregion #region Service
public void DropCache()
{
    Hive.DropCache(TableName);
}
public static void DropAllCache()
{
    Hive.DropCache();
}
public void CopyTo(BasicEntity target)
{
    if (target == null) throw new NullReferenceException("BasicEntity object required.");
    target._fields = _fields; target.Connection = Connection; target._idFieldName = _idFieldName; target.Caching = Caching;
target.Logging = Logging; target.Order = Order;
    target.Filter = Filter; target._what = _what;
}
private void InnerTransactionOpen(SqlCommand command)
{
    if ((command.Transaction == null) || (command.Transaction.Connection == null))
    {
        _innerTransaction = Connection.BeginTransaction(); command.Transaction = _innerTransaction;
    }
}
private void InnerTransactionCommit()
{
    if (_innerTransaction != null)
    {
        _innerTransaction.Commit();
        _innerTransaction.Dispose();
        _innerTransaction = null;
    }
}
private void InnerTransactionRollback()
{
    if (_innerTransaction != null)
    {
        _innerTransaction.Rollback();
```

```
        _innerTransaction.Dispose();
        _innerTransaction = null;
    }
}
protected Fields GetFields(SqlDataReader dataReader)
{
    Fields fields = new Fields();
    for (int i = 0; i < dataReader.FieldCount; i++)
    {
      string name = dataReader.GetName(i); if (!_serviceFields.Contains(name))
          fields.Add(name,
          new Field(dataReader.GetValue(i), dataReader.GetFieldType(i)));
    }
    return fields;
}
protected bool IsActual(BasicEntity reference, string fieldName)
{
    if (reference != null)
    {
        if (Requisites(fieldName).Value != DBNull.Value) return (reference.ID ==
        (Guid)Requisites(fieldName).Value);
        else
        {
            reference = null; return true;
        }
    }
    else
    {
    }
}
 return (Requisites(fieldName).Value == DBNull.Value);
 private void Log(Guid recordID, ReplicationState state)
{
    Update log = new Update(Connection);
    log.Transaction = (_innerTransaction == null) ? Transaction :
    _innerTransaction;
    if (LoggingType == LoggingType.WithConfirmations)
    {
        log.Filter = String.Format("RecordID = '{3}' AND Type = {0} AND (Status = {1} OR Status = {2})",
        (byte)RowStatus.Created, (byte)ReplicationState.Create, (byte)ReplicationState.Update, recordID);
        switch (state)
        {
            case ReplicationState.Update: if (log.First()) return; break;
            case ReplicationState.Delete:
                log.DeleteAll();
                break;
        }
    }
    log.Status = (byte)state;
    log.Type = (byte)RowStatus.Created; log.Table = TableName;
    log.RecordID = recordID; log.Insert();
}
protected void Log(ReplicationState state)
{
    Log(ID, state);
}
protected void LogAll(ReplicationState state)
{
    SomeTable st = new SomeTable(TableName); st.Filter = Filter;
    st.What = "ID";
    st.Transaction = (_innerTransaction == null) ? Transaction :
    _innerTransaction;
    if (st.Count() > 0)
        foreach (SomeTable row in st.GetAllRows())
        {
```

```
            Log(row.ID, state);
        }
}
private bool IsTransactionValid(SqlTransaction transaction)
{
    return (transaction != null) && (transaction.Connection ==
    Connection);
}
private void QueryLog(string query)
{
    Configurator config = Configurator.Get(); if (config.QueryLog == false)
    {
        StringBuilder str = new StringBuilder();
        DateTime.Now);
    }
    str.AppendFormat("---{0:dd.MM.yyyy HH:mm:ss}---",
    str.AppendFormat("\r\n{0}\r\n", query); str.AppendFormat("-------------------------\r\n");
    string dirName = Path.GetDirectoryName(config.QueryLogPath); if (!Directory.Exists(dirName))
        Directory.CreateDirectory(dirName); File.AppendAllText(config.QueryLogPath, str.ToString());
}
#endregion
#region Connection
protected void SafeConnectionOpen()
{
    if (Connection.State == ConnectionState.Closed)
    {
        Connection.Open();
    }
    else if (!IsTransactionValid(Transaction))
    {
        Connection.Close(); Connection.Open();
    }
}
protected void SafeConnectionClose()
{
    if (!IsTransactionValid(Transaction))
    {
        Connection.Close();
    }
}
#endregion
#region Operators
public static bool operator !=(BasicEntity a, BasicEntity b)
{
    if (Object.ReferenceEquals(a, b)) return false;
    if (((object)a == null) || ((object)b == null)) return true;
    if (a._tableName != b._tableName)
        throw new Exception("Entities must be the same type."); return (a.ID != b.ID);

}
public static bool operator ==(BasicEntity a, BasicEntity b)
{
    if (Object.ReferenceEquals(a, b)) return true;
    if (((object)a == null) || ((object)b == null)) return false;
    if (a._tableName != b._tableName)
        throw new Exception("Entities must be the same type."); return (a.ID == b.ID);
}
public override int GetHashCode()
{
    return base.GetHashCode();
}
public override bool Equals(object obj)
{
    return base.Equals(obj);
}
```

```
#endregion
#region Assignment
protected void Assignment(string fieldName, BasicEntity value)
{
   if (value != null)
   {
   }
   else
   {
   }
}
this[fieldName] = value.ID;
this[fieldName] = DBNull.Value;
protected void Assignment(string fieldName, object value)
{
   this[fieldName] = value;
}
before)
 #endregion #region Events
protected Boolean UserProceed(BasicEntityModification type, bool
{
   if (before)
   {
      ModificationEventArgsWithCancel e = new ModificationEventArgsWithCancel(type);
      if (OnBeforeModification != null) OnBeforeModification(this, e);
      return !e.Cancel;
   }
   else
   {
      ModificationEventArgsWithoutCancel e = new
      ModificationEventArgsWithoutCancel(type);
      if (OnAfterModification != null) OnAfterModification(this, e);
      return true;
   }
}
#endregion
#region Conversion
private SqlDbType GetDBType(Type type)
{
   if (_dbTypeTable == null)
   {
      _dbTypeTable = new Hashtable();
      _dbTypeTable.Add(typeof(System.Boolean), SqlDbType.Bit);
      _dbTypeTable.Add(typeof(System.Int16), SqlDbType.SmallInt);
      _dbTypeTable.Add(typeof(System.Int32), SqlDbType.Int);
      _dbTypeTable.Add(typeof(System.Int64), SqlDbType.BigInt);
      _dbTypeTable.Add(typeof(System.Double), SqlDbType.Float);
      _dbTypeTable.Add(typeof(System.Decimal), SqlDbType.Decimal);
      _dbTypeTable.Add(typeof(System.String), SqlDbType.NVarChar);
      _dbTypeTable.Add(typeof(System.DateTime), SqlDbType.DateTime);
      _dbTypeTable.Add(typeof(System.Byte[]), SqlDbType.Image);
      _dbTypeTable.Add(typeof(System.Guid), SqlDbType.UniqueIdentifier);
      _dbTypeTable.Add(typeof(System.Byte), SqlDbType.TinyInt);
   }
   SqlDbType dbType; try
   {
      dbType = (SqlDbType)_dbTypeTable[type];
   }
   catch
   {
      dbType = SqlDbType.Variant;
   }
   return dbType;
}
#endregion
```

```
#region IXmlSerializable Members
public System.Xml.Schema.XmlSchema GetSchema()
{
   return null;
}
public void ReadXml(System.Xml.XmlReader reader)
{
   reader.Read();
   _tableName = reader.ReadString(); reader.Read();
   reader.Read();
   _replicationState = (ReplicationState)Enum.Parse(typeof(ReplicationState), reader.ReadString());
   reader.Read();
   XmlSerializer serializer; switch (_replicationState)
   {
      case ReplicationState.Delete:
         serializer = new XmlSerializer(typeof(Guid)); ID = (Guid)serializer.Deserialize(reader); break;
      default:
         serializer = new XmlSerializer(typeof(Fields));
         _fields = (Fields)serializer.Deserialize(reader); break;
   }
   reader.Read();
}
public void WriteXml(System.Xml.XmlWriter writer)
{
   XmlSerializer serializer = new XmlSerializer(typeof(string));
   serializer.Serialize(writer, TableName);
   serializer = new XmlSerializer(typeof(ReplicationState)); serializer.Serialize(writer, _replicationState);
   switch (_replicationState)
   {
      case ReplicationState.Delete:
         serializer = new XmlSerializer(typeof(Guid)); serializer.Serialize(writer, ID);
         break;
      default:
         serializer = new XmlSerializer(typeof(Fields)); serializer.Serialize(writer, _fields);
         break;
   }
}
#endregion
}
}
```

ДОДАТОК Б. Електронна презентація

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені В. Даля

Моделі та програмні засоби розробки клієнтської частини комп'ютерної системи моніторингу якості освіти

Автор:
ст. групи КІ-17дм
Височина Надія Олександрівна

Керівник дипломної роботи:
Шумова Лариса Олександрівна

2019

1

Рисунок Б.1 – Слайд №1

Актуальність теми і мета проекту

- Формування ефективної системи моніторингу якості освіти неможливе без функціонування високопродуктивних **інформаційних систем**, за допомогою яких генерується, обробляється й аналізується інформація, отримана в результаті здійснення моніторингових досліджень. Це свідчить про **актуальність** проведення моніторингових досліджень якості освіти та розробки інструментальних засобів, що забезпечують його ефективність.
- **Метою дослідження** є підвищення оперативності обробки інформації, отриманої під час проведення моніторингових заходів оцінювання якості освіти.

2

Рисунок Б.2 – Слайд №2

Задачі дослідження

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

- аналіз предметної області;
- формування основних напрямків розробки комп'ютерної системи;
- аналіз методів і засобів отримання інформації;
- розроблення моделей обробки інформації для подальшого аналізу;
- розроблення системи проведення моніторингових досліджень, обробки інформації та аналітичного аналізу;
- проектування загальної схеми роботи комп'ютерної інформаційної системи;
- розроблення програмних засобів і елементів інформаційної технології для проведення моніторингових досліджень.

Об'єкт дослідження – процеси створення та експлуатації програмно-технічних комплексів для моніторингу якості освіти.

Предмет дослідження – програмне забезпечення комп'ютерної системи моніторингу якості освіти.

3

Рисунок Б.3 – Слайд №3

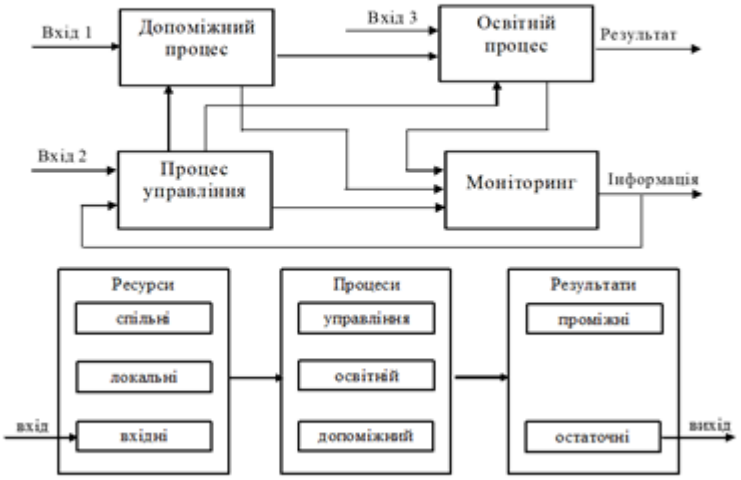
Основні характеристики системи

- система повинна мати централізоване сховище даних;
- кожний ЗЗСО повинен мати власну базу даних;
- повинна бути реалізована гнучка система реплікації між центральним сховищем і локальними базами даних ЗЗСО;
- системні вимоги, що пред'являються клієнтською частиною, повинні бути мінімізовані, що має забезпечити прийнятну швидкість роботи навіть на досить слабких комп'ютерах;
- у рамках одного освітнього закладу можуть функціонувати кілька клієнтських програм, розташованих на різних робочих станціях локальної мережі, при цьому результати роботи кожного клієнта повинні зберігатися в загальну базу, розташовану в тій же локальній мережі, яка також є сховищем власних даних ЗЗСО;
- клієнтська частина програми повинна бути максимально проста в установці і розгортанні, а її графічний інтерфейс має бути інтуїтивно зрозумілим навіть для користувача, що володіє тільки початковими навичками роботи в середовищі Windows.

4

Рисунок Б.4 – Слайд №4

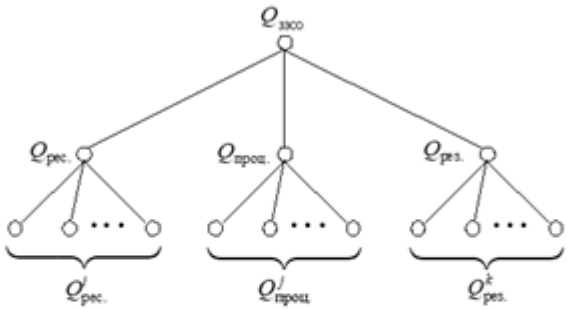
Узагальнена процесна модель ЗЗСО



5

Рисунок Б.5 – Слайд №5

Дерево якості ЗЗСО за РПР-моделлю



$$Q_{\text{ЗЗСО}} = \langle Q_{\text{рес.}}; Q_{\text{проц.}}; Q_{\text{рез.}} \rangle$$

Q рес. – якість ресурсів
Q проц.- якість процесів
Q рез. – якість результатів

6

Рисунок Б.6 – Слайд №6

Рисунок Б.7 – Слайд №7

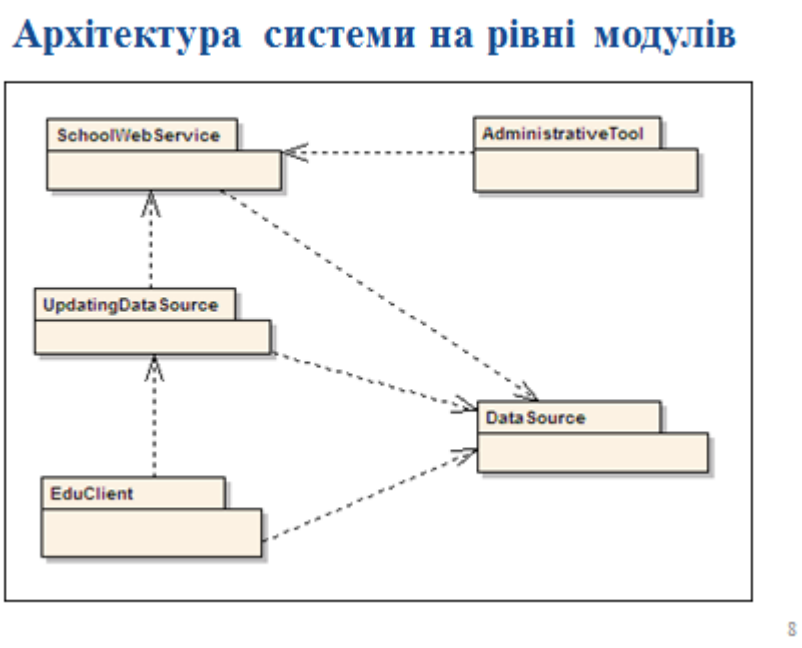


Рисунок Б.8 – Слайд №8

Рисунок Б.9 – Слайд №9

Рисунок Б.10 – Слайд №10

Інтерфейс програми

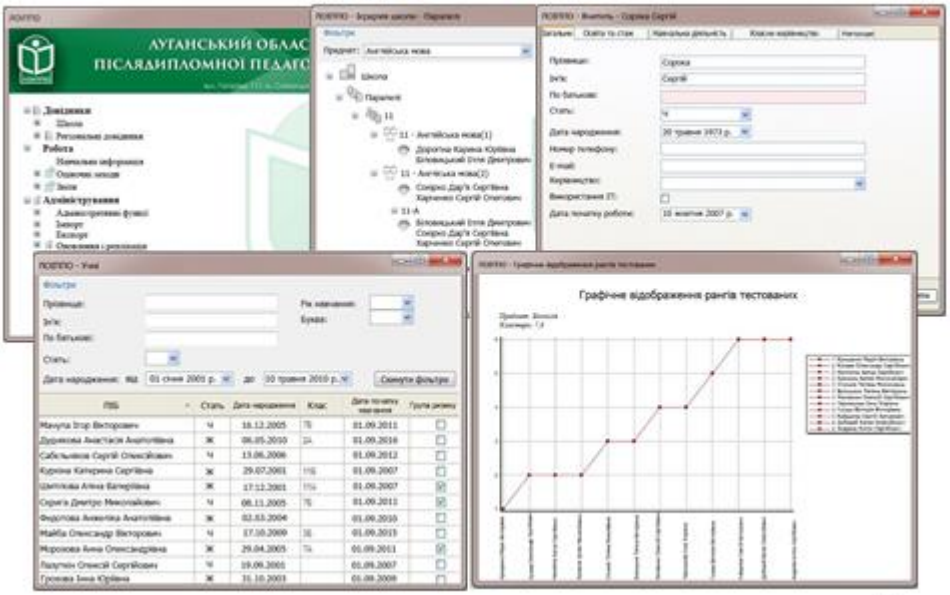


Рисунок Б.11 – Слайд №11

Висновки

Розроблювана система забезпечить збір, обробку та інтерпретацію інформації про результати освітньої діяльності. Реалізована система реплікації даних прискорить збір інформації та забезпечить поширення загальних довідників в ЗЗСО. Більшість необхідних довідників вже зберігається на сервері, також розроблені механізми імпорту з excel для деяких з них, що забезпечує легкий механізм додавання нових даних.

У перспективі система повинна бути доповнена аналізом даних ЦБД, підбиттям статистики. Має бути доопрацьований механізм поновлення, що підтримує не тільки оновлення даних, а й оновлення самої системи, а також формування дистрибутивів з останньою версією клієнта.

На даному етапі розробки отримані наступні результати:

- проаналізована предметна область;
- спроектована архітектура;
- реалізовано підсистеми доступу до даних та реплікації з центральної БД;
- реалізовано модуль інтеграції системи з АСК «Символ»;
- створено робочий прототип клієнтської частини.

12

Рисунок Б.12 – Слайд №12