УДК 681.3
№ держреєстрації 0100U001428
Інв. № 0203U002140

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
ЛНУ ім. Івана Франка
79000 м. Львів, вул. Університетська, 1; тел.(0322)72 70 40
факс (0322) 97-89-03, ndch@franko.lviv.ua

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
докт. хім. наук, проф.

_____ Б.Котур

___.12. 2002

ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ

РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНИМИ МАСИВАМИ WEB-САЙТІВ УНІВЕРСИТЕТУ

(ПІ-97Б)
(Заключний)

Заступник проректора
з наукової роботи,
канд.хім.наук, ст.наук.співр. С.Орищин

Декан факультету
прикладної математики та інформатики,
докт.фіз.-мат.наук., проф. Я.Савула

Науковий керівник,
канд.фіз.-мат.наук, доц. В.Горлач

2002

Рукопис закінчено 15 листопада 2002 р.
Результати цієї роботи розглянуто Вченою Радою
ф-ту прикладної математики та інформатики,
протокол від 02.11.17 № 2

Список авторів

науковий керівник теми, доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	В.Горлач (розділи 1-2)
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	І.Бернакевич (1.3)
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	В.Вовк (1.3)
доц. кафедри програмування, канд. фіз.-мат. наук	_____	А.Музичук (2.3)
доц. кафедри програмування, канд. фіз.-мат. наук	_____	В.Черняхівський (1.1)
ст. наук. співр. кафедри інформаційних систем	_____	О.Левченко (2.4)
інж. І категорії	_____	Р.Малець (2.4)
інж. І категорії	_____	О.Шинкаренко (1.1,1.2,2.1,2.2)
лаб. кафедри інформаційних систем	_____	Т.Семенюта (2.5)
аспірант кафедри інформаційних систем	_____	Я.Кондратюк (1.1)
студент	_____	В.Крупач (2.7)
студент	_____	А.Петльований (2.3)
студент	_____	А.Табачин (2.3)
студент	_____	А.Федорович (2.6)
студент	_____	Д.Хромов (2.4)

Нормоконтролер М.Благітка

РЕФЕРАТ

Звіт про НДР: 61 с., 57 рис., 25 табл., 18 джерел.

Розробка програмного забезпечення та принципів керування інформаційними масивами WEB-сайтів Львівського національного університету імені Івана Франка, які б забезпечили покращення системи управління університетом на базі сучасних комп'ютерних технологій.

Розроблено Web-модель інформаційної системи корпоративного типу, яка включає архітектуру, основні класи, структуру баз даних, XML-формати передачі даних. Розроблено концепцію кросплатформенної технології проектування розподілених інформаційно-пошукових систем. Визначено вимоги до системного програмного забезпечення Web-сервера та клієнта.

Розроблені моделі, принципи керування інформаційними масивами та окремі підсистеми відповідають сучасному розвитку технологій проектування корпоративних інформаційних ресурсів.

Отриману Web-модель та принципи кросплатформенного проектування можна використати для побудови масштабованих розподілених багаторівневих інформаційних систем. Зокрема, це можуть бути системи супроводження документообігу в університеті та його підрозділах, системи кадрового обліку і інші облікові та інформаційні системи, а також системи для дистанційного навчання, проведення дискусій, опитування працівників, тестувань в навчальному процесі тощо.

Запропоновані моделі та принципи кросплатформенного проектування можуть бути використані для побудови масштабованих розподілених багаторівневих інформаційних систем вищих закладів освіти та інших установ та організацій. Зокрема, це можуть бути системи супроводження документообігу, системи кадрового обліку і інші облікові та інформаційні системи, а також системи для дистанційного навчання, проведення дискусій, опитування працівників, тестувань в навчальному процесі тощо. Готові підсистеми "Телефонний довідник", "Наукові видання", "Облік наукових публікацій", "Універсальна система тестувань", "Система "Інформаційний бюлетень" впроваджені у Львівському національному університеті імені Івана Франка та можуть бути, практично без додаткових модифікацій, встановлені та використовуватись в інших вищих закладах освіти, установах та організаціях.

ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-САЙТ, ІНТЕРНЕТ, ІНТРАНЕТ, ВЕБ-ТЕХНОЛОГІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ, ФОРМАТ ДАНИХ.

ЗМІСТ

ВСТУП 5
1. ЗАГАЛЬНІ ЗАСАДИ ПОБУДОВИ ТА СТРУКТУРА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ КОРПОРАТИВНОГО ТИПУ 7
1.1. Основні інформаційні масиви учбового закладу 9
1.2. Вимоги до програмно-апаратних засобів 10
1.2.1. Сервери 10
1.2.2. Клієнти 10
1.2.3. Дизайн 10
1.2.4. Програмування 10
1.3. Об'єктний підхід до проектування баз даних 11
1.3.1. Модель даних 11
1.3.2. Модель процесів індивідуальної поведінки об'єктів 15
1.3.3. Модель процесів корпоративної поведінки групи об'єктів 17
2. РЕАЛІЗАЦІЯ ОКРЕМИХ ПІДСИСТЕМ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ ЛЬВІВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ФРАНКА 21
2.1. Загальний дизайн та карта інформаційно-пошукової системи "Кампус" 21
2.2. Web-застосування "Інформаційний бюлетень" 23
2.2.1. Структура інформації 23
2.2.2. Модуль представлення даних 25
2.2.3. Адміністративний модуль 26
2.3. Модель університетської ІС на прикладі системи тестування 28
2.4. Web-сайт "Наукові видання Львівського університету" 33
2.4.1. Структура інформації 33
2.4.2. Представлення інформації 35
2.5. База даних наукових публікацій 39
2.5.1. Структура інформації 39
2.5.2. Представлення інформації 39
2.5.3. Внесення інформації 40
2.6. Реалізація універсальної тестової системи у вигляді Web-сервера інтранет навчального закладу 42
2.6.1. Формалізація тесту - опитувача 42
2.6.2. Структура інформації 43
2.6.3. Опис таблиць бази даних 44
2.6.4. Опис інтерфейсу адміністратора 48
2.6.5. Внесення інформації 50
2.6.6. Інтерфейс користувача 52
2.7. Web-застосування "Телефонний довідник" 56
ВИСНОВКИ 59
ПЕРЕЛІК ПОСИЛАНЬ ТА ПУБЛІКАЦІЙ ВИКОНАВЦІВ 60

ВСТУП

Сучасний стан розвитку інформаційних технологій та швидкий ріст числа користувачів глобальної мережі Інтернет не тільки роблять можливими, а і вимагають використання нових можливостей доступу до різнорідної інформації зокрема в навчальних закладах (надалі під терміном "навчальний заклад" матимемо на увазі в першу чергу вищі навчальні заклади). При цьому мова йде не лише про використання сучасних інформаційних технологій в навчальному процесі, але і про представлення інформації про сам навчальний процес як в локальних мережах навчального закладу (Інтранет), так і в мережі Інтернет.

Існуючі сьогодні Web-ресурси навчальних закладів в Україні здебільшого представляють їх структуру, тобто перелік факультетів, спеціальностей, кафедр та коротку інформацію про них (як правило це статичні Web-сторінки, оновлення яких у кращому випадку відбувається 1-2 рази на рік). Є окремі спроби використання Web-застосувань, що працюють з структурованою інформацією. Однак ці застосування орієнтовані в першу чергу на забезпечення подачі інформації про навчальний заклад назовні (усім користувачам Інтернет) і рідко мають використання в навчальному процесі.

Досвід проектування та експлуатації інформаційно-пошукових систем провідними університетами світу підтверджує доцільність об'єднання сучасних технологій проектування баз даних з технологіями Інтернет-Інтранет, що дозволяє використовувати накопичену в базах даних інформацію як для внутрішніх цілей, так і для представлення інформації про навчальний заклад всім користувачам мережі Інтернет.

Специфікою створення і супроводу масштабних інформаційних систем є неефективність їх розробки знизу (методом нарощування), без початкового єдиного плану структури даних і технологічних процесів. Саме це є основною перешкодою об’єднання значного числа існуючих різнорідних програмних наробок у ефективні системи керування діяльністю установ, у тому числі навчальних.

Прийнята на державному рівні стратегія демократизації освіти істотно підвищила рівень самоуправління у навчальних закладах, що принципово ускладнює розробку єдиної типової для навчальних закладів інформаційної системи, яка могла б бути створена на рівні міністерства. Застосування ж єдиного підходу до проектування таких систем, не обмежуючи різноманітність форм конкретних реалізацій, гарантує зменшення витрат на повторне фінансування кожного нового проекту і в майбутньому може значно спростити їх об’єднання у єдину національну мережу.

В даній роботі та ряді праць виконавців проекту [1-13] запропоновано концепцію інформаційної системи, яка, з однієї сторони, забезпечує гнучкість реалізації умов конкретного навчального закладу, а з іншої – є достатньо структурованою в рамках системи науково-освітніх установ. Її основу складають наступні основні принципи:

- застосування об’єктного підходу на всіх рівнях проектування, розробки та супроводу АІС, який, зокрема, передбачає:
  - об’єктну модель даних (під керуванням реляційних серверів БД);
  - об’єктну модель технологічних процесів обробки та представлення даних;
  - багатошарову ієрархічну архітектуру "клієнт/сервер" організації колективного доступу до даних;
  - організацію топології зв’язку між клієнтськими та серверними застосуваннями у вигляді графа;
- використання розподіленого (у тому числі гетерогенного) доступу до баз даних окремих підрозділів на основі технологій:
  - Інтранет - у межах кампусної мережі (адміністрація, факультети, бухгалтерія, відділ кадрів, тощо)
  - Інтернет - для віддаленого доступу до баз даних розподіленої інформаційної системи. Для динамічної генерації Web-сторінок використовуються сервлети та сторінки JSP.

Застосування запропонованої концепції до реалізації складної інформаційної системи надає наступні переваги:

- відкритість структури системи до доповнень та модифікацій
- повторне використання вже напрацьованого програмного коду
- максимальне використання потужностей Web – технологій
- стійкість загальної працездатності системи до можливого некоректного функціонування її підсистем.

Окремі елементи запропонованої авторами концепції інформаційно-пошукової системи успішно апробовані при проектуванні та розробці низки окремих програмних продуктів [14-18].

1. ЗАГАЛЬНІ ЗАСАДИ ПОБУДОВИ ТА СТРУКТУРА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ КОРПОРАТИВНОГО ТИПУ

Діяльність університету супроводжується значними інформаційними потоками різної природи. Значна їх частина в сукупності представляє типовий інформаційний простір підприємства середніх розмірів; як організаторами, так і споживачами такої інформації є службовці адміністративної частини університету. При цьому адміністрування здійснюється безпосередніми керівними посадовцями.

Якщо ж говорити про іншу частину інформації, безпосередньо пов’язану з навчальним процесом, то вона на даний час представлена розрізненими програмами, які розроблені без єдиної ідеології, не завжди надають змогу використовувати їх у корпоративній мережі, не мають уніфікованого підходу до використання баз даних, окремі вимагають спеціального навчання користувачів тощо. Що ж до людей, які мають відношення до цих інформаційних потоків, то, по-перше, їх на кілька порядків більше, ніж у адміністративній частині; практично це можуть бути всі студенти та весь професорсько-викладацький склад. По-друге, користувачі можуть належати до багатьох груп, які можуть динамічно утворюватися під час навчального процесу і змінювати свій склад. Очевидно, що адміністрування цих процесів повинно бути децентралізованим, тобто самі користувачі братимуть на себе певну частину адміністративних функцій. По-третє, способи підготовки цієї інформації мають бути досить простими і зручними, доступними для широкого загалу.

Перелічені вище особливості університетської ІС тісно пов’язані з її технічними аспектами. Зокрема, це експлуатація на різних комп’ютерних платформах, супроводження на комп’ютерах з різними технічними характеристиками, маштабування тощо. При цьому треба враховувати швидке старіння техніки, оскільки в межах університету реально експлуатуються комп’ютери різних поколінь. Отже, клієнтська частина ІС повинна бути максимально простою. Якщо ще передбачити доступ через Інтернет, як це вимагають сучасні тенденції та технології дистанційного навчання, то стає очевидним, що архітектура ІС повинна проектуватися під Web-технології.

Такий підхід до розгляду інформаційних масивів учбового закладу та пов'язаних з ними задач відображення інформації є основою для проектування відповідних баз даних та Web-застосувань, які, в залежності від потреб, можуть динамічно (одночасно з оновленням баз даних) представляти вказану інформацію як "внутрішнім" користувачам (викладачам, студентам традиційної чи дистанційної форми навчання), так і "зовнішнім" користувачам (викладачам та студентам інших факультетів чи вузів, абітурієнтам, іншим користувачам мережі Інтернет). Крім того, враховуючи можливу територіальну розподіленість підрозділів факультету чи вузу, такий підхід стирає різницю між

користувачами, що користуються різними апаратно-програмними середовищами чи лініями зв'язку різної пропускної здатності (локальна мережа, комутовані канали тощо) і дозволяє з однаковим успіхом забезпечити уніфікований доступ до інформації в спільних базах даних.

При цьому, завдяки використанню Інтернет-Інтранет-технологій та авторизації доступу до певної частини інформації, поняття "зовнішнього" чи "внутрішнього" користувача не пов'язане з місцем його фізичного розташування – робоча станція в локальній мережі вузу чи домашній комп'ютер з доступом до Інтернет.

Надалі для означення описаної вище інформаційно-пошукової системи навчального закладу, базованої на використанні технологій Інтернет-Інтранет вживатимемо також термін – система "Кампус" (campus – університетське містечко).

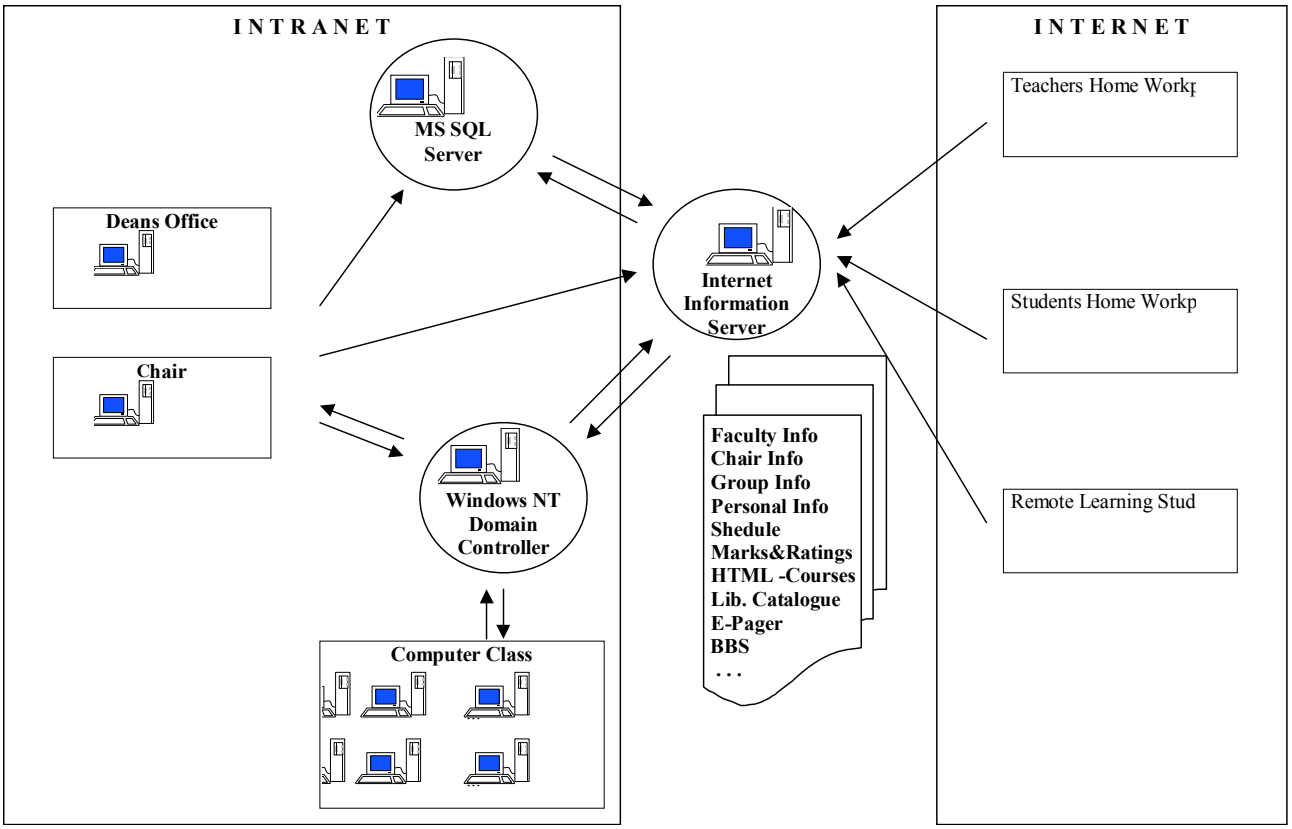

Рис.1.1. Схема функціонування інформаційно-пошукової системи, базованої на використанні технологій Інтернет-Інтранет.

1.1. Основні інформаційні масиви учбового закладу

Основними інформаційними масивами на рівні навчального підрозділу (факультету) вважаємо:

- викладачів, групи яких формують поняття "кафедра";
- студентів, групи яких формують поняття "академічна група";
- навчальні курси, групи яких формують поняття "спеціальність".

Як окремі елементи кожного інформаційного масиву, так і їх групи володіють певними властивостями. Якщо до персональної інформації викладача ми включаємо таку інформацію (наведений перелік не є повним) як : ім'я, освіта, посада, вчена ступінь, фото тощо, то інформація про навчальний курс містить назву, робочу програму, семестровий план та (можливо) зв'язки з переліком Web-ресурсів, що представляють матеріали лекційних та/або лабораторних занять.

На перетині цих основних інформаційних масивів виникають, зокрема, задачі відображення :

- розкладу занять у вибраній користувачем формі;
- успішності студентів.

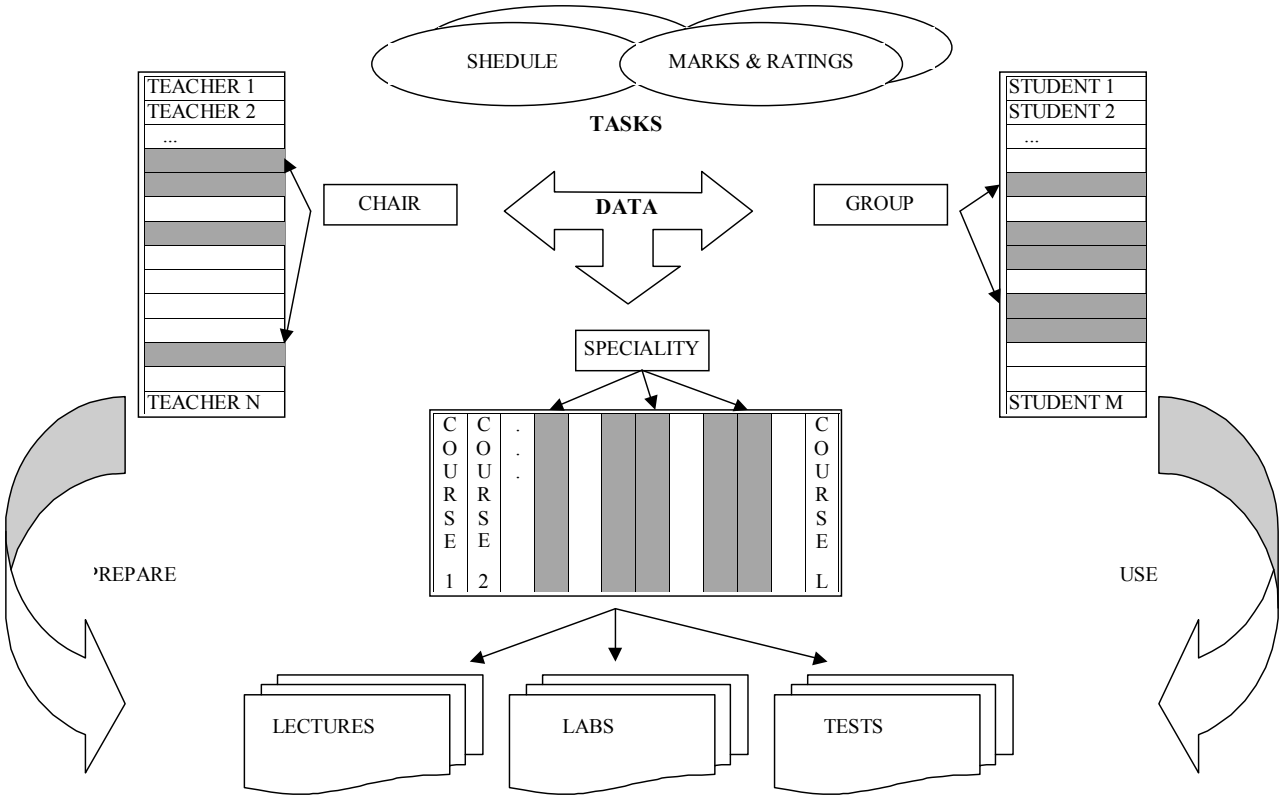

Рис.1.2. Структура даних та основні задачі системи "Кампус" на рівні факультету

Як додаткові можуть розглядатись задачі інформування викладачів, студентів, абітурієнтів через електронний інформаційний бюлетень (дошку оголошень), доступу та ведення електронного каталогу кафедральних, факультетських чи вузівських бібліотек та інші задачі.

1.2. Вимоги до програмно-апаратних засобів

Сформулюємо тут основні вимоги до програмно-апаратних засобів.

1.2.1. Сервери

- Операційна система сімейства UNIX (Sun, FreeBSD, Red Hat). Web-сервер Appache з підтримкою технологій JSP, PHP, XML. Сервер баз даних MySQL.
- Операційна система Windows 2000 Server. Web-сервер IIS 5.0 з підтримкою технологій JSP, PHP, XML, ASP.

1.2.2. Клієнти

У якості клієнтів допускається використання Web-браузерів починаючи з версій IE4, NC4.x.

1.2.3. Дизайн

Дизайн сторінок повинен забезпечувати їх функціонування в роздільній здатності 800x600px 256 кольорів.

Сторінки, які будуть видимі ззовні університетської мережі повинні створюватися в двох варіантах — україномовному (cp-1251) та англомовному.

Для оформлення вмісту сторінок слід використовувати стилі, визначені в існуючих CSS та HTML-шаблонах.

Для макетування бажано використовувати редактори Macromedia Dreamweaver та HomeSite.

1.2.4. Програмування

Програмування клієнтської частини HTML-сторінок повинно забезпечувати сумісність з браузерами IE4, NS4.x. Для програмування клієнтської динаміки слід використовувати JavaScript.

Для сторінок, які генеруються динамічно, слід використовувати кросплатформенні технології типу PHP та JSP.

У випадку використання програмної платформи сімейства UNIX (Sun, FreeBSD, Red Hat) та web-сервера Appache основними технологіями проектування є JSP та PHP та сервер баз даних MySQL.

Як виняток, в окремих випадках допускається використання програмної платформи Windows NT 4.0 або Windows 2000 Server з технологіями JSP та ASP. В цьому випадку в якості web-сервера використовується MS IIS 5. Доступ до баз даних здійснюється засобами ADO. В серверних сценаріях бажаним є використання конструкцій JavaScript. SQL-запити програмуються як stored procedures SQL-сервера, а не включаються безпосередньо в текст ASP-сторінок.

Структура баз даних повинна забезпечувати коректну роботу SQL-запитів двома мовами - українською та англійською.

Кожна з підсистем, яка звертається до бази даних, повинна надавати web-засоби для модифікації інформації з бази даних з авторизованим доступом.

1.3. Об'єктний підхід до проектування баз даних

Універсальність і простота використання реляційної моделі баз даних суттєво зменшили гостроту низки специфічних проблем, які виникають у процесі розробки великомасштабних програмних систем. Проте постійне зростання обсягів програмного коду рано чи пізно знову виводить ці проблеми у ряд головних і змушує шукати шляхи їх вирішення в безальтернативному напрямку – об’єктному підході.

На сьогоднішній день об'єктний підхід залишається безальтернативною концепцією реалізації великомасштабних програмних проектів. Його поширенню сприяє швидкий ріст потужностей комп'ютерної техніки, який спонукає розробників до регулярного оновлення програмного забезпечення з використанням найновіших технологій програмування. Проте бази даних залишаються дещо осторонь цих процесів, оскільки задача внесення інформації про об'єктні взаємозв'язки у всю багаторічно накопичувану масу даних є надто громіздкою. Таким чином, розробка і впровадження об'єктних СКБД сильно гальмуються "історично міцними" позиціями реляційних серверів БД. Оскільки істотних змін у дану ситуацію найближчим часом не передбачається, природними виглядають спроби прилаштувати об'єктний підхід (бодай частково) до обробки реляційно організованої інформації.

1.3.1. Модель даних

На теперішній час об’єктний підхід є єдиною методологією, яка здатна подолати більшість специфічних проблем розробки великомасштабних програмних комплексів. Для новостворюваного програмного забезпечення його застосування є практично безальтернативним. Проте спроби переробки вже готових систем на об’єктну основу так чи інакше призводять до необхідності повної їх перебудови. Чи не найбільшою мірою це стосується інформаційних систем на основі баз даних, програмне забезпечення яких, крім клієнтської частини, містить засоби однієї з існуючих СКБД, що не підтримує об’єктного підходу в повному обсязі. Настільні варіанти локальних СКБД першими підхопили об’єктну ідею і реалізували її у засобах розробки клієнтських застосувань. Але реалізація об’єктної моделі даних виявилась значно серйознішою задачею, яка потребує масштабних наукових досліджень. Через це розробки сучасних СКБД з об’єктною моделлю даних на сьогодні ще не є широкого поширеними, хоча у пресі час від часу з’являються повідомлення про успіхи і навіть промислові зразки таких систем.

В роботі запропоновано компромісний варіант організаціїї моделі даних великих інформаційних систем, що має основні ознаки об’єктного підходу і не потребує наявності об’єктної СКБД.

Реляційна модель даних дає змогу в потрібний момент потрібним чином легко зв’язати в одне ціле інформацію про об’єкт, яка розміщена у декількох таблицях. Ця властивість широко використовується при різних формах нормалізаціїї бази даних і в цілому призводить до збільшення загальної

кількості таблиць завдяки зменшенню у них кількості інформаційних полів. Основними критеріями у цих процесах найчастіше виступають міркування ліквідації дублювання даних, збереження цілісності інформації тощо. Не випускаючи з уваги цих критеріїв, покладемо в основу алгоритму розділення інформації по таблицях **ієрархічність структури системи об'єктів інформаційної системи**. Наявність ієрархії гарантується вихідним припущенням про складність цієї системи в сенсі застосовності об'єктного підходу. Оскільки останній у сфері програмування трактує принцип ієрархії як принцип наслідування ознак, то тільки цей вид ієрархії може бути використаний для конструювання моделі даних системи. Проілюструємо сказане на довільно вибраному прикладі :

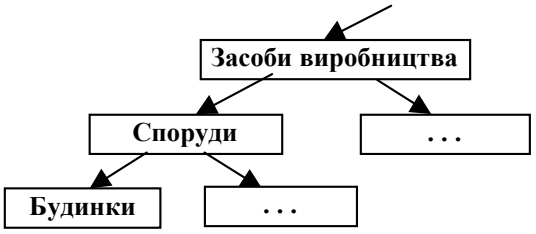

Рис. 1.3. Приклад ієрархії об'єктів

Кількісні й якісні характеристики ієрархії повністю залежать від задач, які розв'язує інформаційна система, тому наведений вище ланцюжок класів може мати багато інших варіантів, як наприклад:

Наземні конструкції -> Житлові споруди ->...Будинки ->

Достатньо лише дотримуватись принципу наслідування, оскільки тільки він підтримується стандартними програмними засобами і у нашому випадку є першою ознакою переходу від загального поняття об'єктного підходу до конкретних принципів об'єктно-орієнтованого програмування.

Приймаючи ієрархію у вигляді ланцюжка класів, розкладемо інформацію з таблиці властивостей об'єкта (далі все на прикладі будинку)

Будинки

1. Назва	4. Адреса	7. Поверхів
2. Власник	5. Матеріал	8. Під'їздів
3. Дата	6. Площа	9. Квартир
		10. . . .

на відповідний набір таблиць:

Засоби
1. Ун. номер
2. Назва
3. Власник
4. Дата
5. . . .

Споруди
1. Ун. номер
2. Адреса
3. Матеріал
4. Площа
5. . . .

Будинки
1. Ун. номер
2. Поверхів
3. Під'їздів
4. Квартир
5. . . .

Для зв'язування в одне ціле належних конкретному будинку записів у кожну з цих таблиць як ключове додано ще одне поле **Ун.номер.** Воно дає

змогу утворювати кортежі з повною інформацією про будинки. Головні переваги зробленого поділу:

- За необхідності введення в систему нового класу об’єктів, скажімо “**Склади**”, для зберігання більшої частини інформаціїї можна скористатись уже наявними таблицями “**Засоби**” та “**Споруди**”. Заново створити достатньо лише відносно невелику за кількостю стовпців таблицю **“Склади”,** яка містить тільки ті поля, що відрізняють поняття “склад” від поняття “споруди”. Це, очевидно, економніше з огляду на дискову пам’ять, аніж розширення первинної таблиці “**Будинки**”, і набагато ефективніше від нової повної таблиці **“Склади”** з точки зору програмного забезпечення, оскільки код для оперування більшою частиною даних вже готовий.
- Ієрархія таблиць даних дає змогу побудувати відповідну ієрархію програмних форм, які зображають ці дані. Сучасні засоби програмування, які підтримують об’єктно-орієнтований стиль, надзвичайно ефективно реалізують процеси спадкування програмних властивостей.
- Разом з формами зображення кожна таблиця отримує статус програмного типу (класу), де об’єктами виступають окремі записи, а властивості форми визначають їх поведінку.

Детально проаналізуємо названі результати.

Групування властивостей об’єктів конкретного типу у зв’язаний унікальним полем набір таблиць забезпечує високий рівень мобільності програмного коду при потребах модифікації, а також підвищення його стійкості до внесення випадкових помилок у систему. Проте при значному зростанні кількості класових типів загострюється проблема організації взаємодії між ними, в основі якої, як виявили перші ж спроби апробації запропонованого підходу, лежать труднощі однозначної ідентифікації об’єктів різних типів. Ця проблема легко усувається введенням єдиної наскрізної нумерації всіх об’єктів у системі. Для цього у вершину ієрархії типів доцільно ввести додатковий службовий клас, що визначається такою таблицею

OBJECTS

UniqueNo	Integer
Class	Integer
---	---
Major	Integer
Counter	Integer
Create_date	Date
Create_user	Integer
Security	Integer
. . .	

де поле UniqueNo має статус ключового. Принагідно зауважимо, що розподіл даних про властивості об’єкта по ієрархічно зв’язаних за спадковим принципом таблицях автоматично нормалізує базу даних за другою основною формою.

Отже, кожному об’єкту системи, який описується будь-яким кортежем з набору таблиць, відповідає єдиний запис у таблиці OBJECTS з унікальним значенням **UniqueNo** і числовим номером класу у полі **Class**. За значенням другого поля довільний об’єкт “знає”, до якого класу він належить, що дає змогу запустити на виконання відповідну форму його зображення на екрані.

Для оформлення елементів таблиці OBJECTS у програмний клас слід визначити їх поведінку у вигляді відповідної програмної форми, від якої будуть спадкуватись всі наступні класи. Зрозуміло, що в контексті об’єктного програмування такий клас FOBJECTS апріорі є абстрактним, тобто служить лише шаблоном для утворення наступних класів реальних об’єктів.

Залежно від потреб інформаційної системи OBJECTS є зручним місцем розміщення інших спільних для всіх об’єктів даних, наприклад: **Major** – номер батьківського по ієрархії об’єкта, **Counter** – лічильник звертань до об’єкта для контролю за можливістю його вилучення з системи, **Security** – індекс обмеження доступу тощо.

Отже, OBJECTS у моделі даних виконує роль стрижня, до якого на основі значення поля **UniqueNo** приєднуються всі решта таблиці.

Зобразимо це графічно:

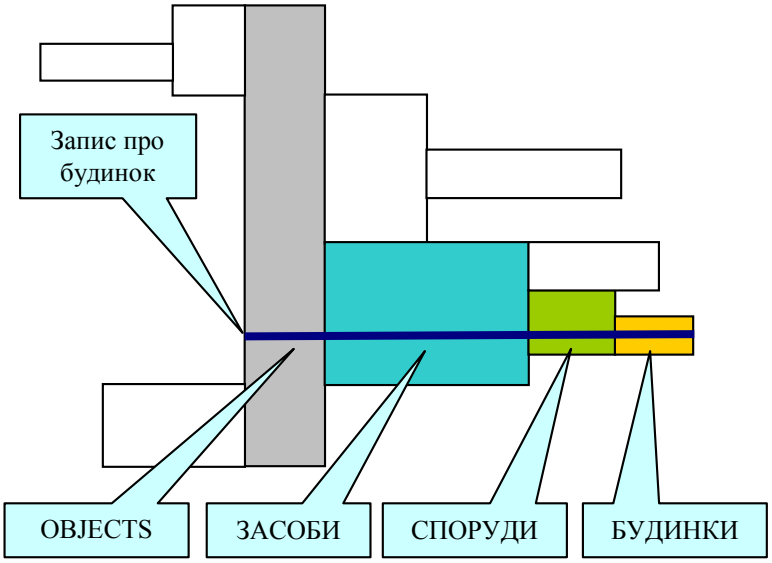

Рис.1.4. Графічне представлення зв'язку між таблицями

Отже, повна інформація про об’єкт (будинок) рознесена по декількох таблицях, але зв’язана в одне ціле ключем **UniqueNo**.

Пряма відповідність набору полів кожної таблиці наборові властивостей, які визначають особливості конкретного виду реальних об’єктів, дає змогу строго обмежити доступ до даних цього типу у межах однієї або декількох програмних форм. Ієрархія таблиць за спадковим принципом однозначно задовільняє принцип модульності об’єктного підходу, а його принцип абстрагування автоматично закладений у зміст набору таблиць бази даних. Таким чином, запропонований принцип побудови моделі даних може бути основою для застосування об’єктно-орієнтованого підходу до програмної реалізації моделі процесів довільної складної інформаційної системи. Для

ілюстрації сказаного наведемо фрагмент верхівки класів моделі даних системи підтримки менеджменту у сфері виробництва.

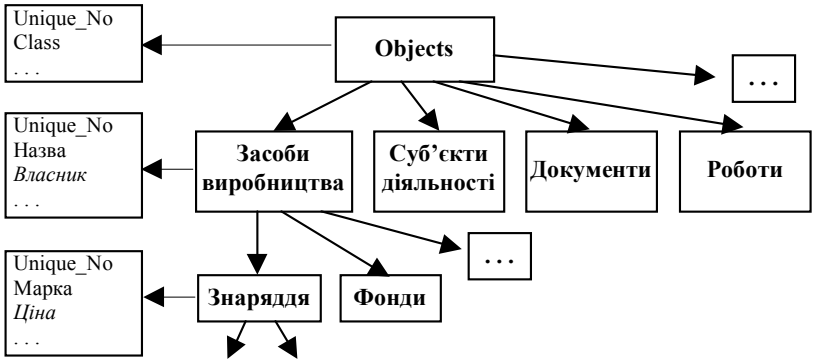

Рис.1.5. Верхівка класів моделі даних системи підтримки менеджменту

1.3.2. Модель процесів індивідуальної поведінки об'єктів

Цілком логічним виглядав би перехід від об’єктної моделі даних до об’єктної моделі процесів, де процедури як елементи технології обробки даних самі виступають окремими об’єктами. Проте природа процесів інкапсуляції, наслідування та поліморфізму у цьому випадку ще потребує вивчення. Тому нижче при побудові моделі процесов ми обмежимося використанням об’єктності моделі даних.

На відміну від звичайних програмних об’єктів, які знаходяться в оперативній пам’яті комп’ютера, властивості об’єктів класів, що приведені на рис. 3, розміщені у таблицях бази даних на зовнішніх носіях. Проте поведінка (методи) і надалі описується процедурами, які виконуються в оперативній пам’яті. Через це для реалізації об’єктно-орієнтованого підходу використання відповідних стандартних засобів компіляторів призводить до значних накладних витрат. Вирішенням проблеми може бути налагодження штучного об’єктного зв’язку між кортежами в БД та програмними формами, які реалізують процеси обробки даних. Важливо з максимальною повнотою зімітувати принципи інкапсуляції, наслідування та поліморфізму, для чого надалі спиратимемось на об’єктну природу побудованої вище моделі даних.

Обмеження доступу до даних об’єктів класу з достатньою для практичного застосування строгістю можна реалізувати введенням єдиної для кожного класу програмної форми, що має ексклюзивний доступ до відповідної класу таблиці в БД і в кожний конкретний момент часу налаштована на редагування єдиного вибраного об’єкта (кортежа). Для цього зручно використати поле CLASS у згаданій вище таблиці OBJECTS, оскільки воно включається у довільний кортеж з даними про будь-який об’єкт. Достатньо завести додаткову таблицю, що містить інформацію про наявні у системі класи і має, наприклад, таку структуру

CLASSREF

Class	Integer
Major	Integer
---	---
Form	Char
Description	Char
. . .	

Тут поле FORM відповідає за адресацію програмної форми, яка задає технологію подання будь-якого об’єкта класу з номером CLASS і, фактично, інкапсулює весь зовнішній інтерфейс по внесенню будь-яких змін у його дані. Нагадаємо, що дотепер йшлося лише про інформацію у відповідній класу таблиці. Повний же кортеж можна отримати із об’єднання (на основі поля UniqueNo) всіх таблиць батьківських класів, для чого CLASSREF містить поле MAJOR, яке моделює концепцію наслідування класів. З його допомогою об’єкт у потрібний момент здатний відновити повний генеалогічний ланцюжок батьківських класів (наслідуваних форм) і реалізувати всю свою успадковану поведінку. Оскільки остання не залежить від зовнішних (у БД) даних класу, то реалізація наслідування і поліморфізму підтримується штатними засобами компіляторів чи середовищ розробки програм. Для створення візуального інтерфейсу користувача найбільш логічним у цьому випадку є подання об’єктів у вигляді дерева, яке грунтується на значеннях полів **UniqueNo** та **Major у** таблиці OBJECTS. З іншого боку, модель процесів містить не тільки розглянуті до цього часу процедури роботи з одиночними екземплярами об’єктів. Потреби генералізації даних в інформаційній системі потребують наявності процедур класу, які визначають його поведінку в цілому. Наприклад, складання списків об’єктів з визначеними умовами фільтрації та ін. У цій частині модель процесів має самостійну відмінну від моделі даних деревовидну ієрархію на основі значень полів **Class** та **Major** з таблиці CLASSREF. Для відображення можливості розробки додаткових спеціальних процесів роботи з даними, що не опираються на їх об’єктну природу, на рисунку внизу відповідна гілка позначена знаком питання:

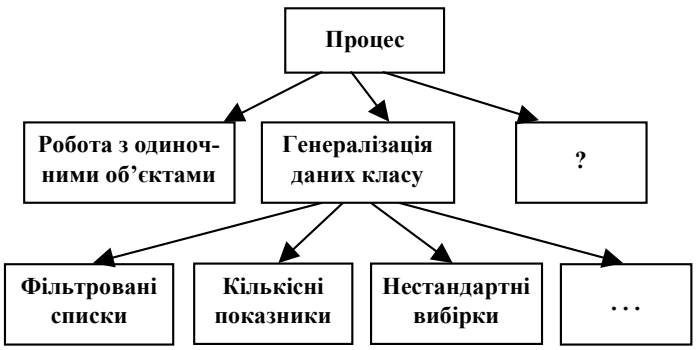

Рис.1.6. Приклад моделі процесів

На відміну від моделі процесів роботи з окремими об’єктами реалізація моделі поведінки класу, що грунтується на об’єктності моделі даних, залишається предметом подальших досліджень.

1.3.3. Модель процесів корпоративної поведінки групи об'єктів

У попередньому пункті запропоновано підхід до побудови моделі даних інформаційної системи, який володіє основними ознаками об'єктного підходу і не потребує наявності об'єктної СКБД. Додатково виконано спробу застосувати вже використані засоби до організації об'єктної моделі процесів. Проте, це вдалось зробити задовільно лише для тих процесів, які відображають індивідуальну поведінку об’єктів класу. Предметом подальшого обговорення є побудова моделі процесів для описання корпоративної поведінки групи об’єктів.

1.3.3.1. Проблеми побудови об’єктної моделі процесів

З одного боку, використання природної ієрархії класових типів дає змогу побудувати просту і ефективну ієрархію процесів створення, модифікації та знищення об’єктів у складній програмній системі. Більше того, далі буде показано, що її ж зручно застосовувати до реалізації групової поведінки цілих класів об’єктів.

З іншого боку, чи не найголовнішою перевагою зберігання інформації у вигляді бази даних є можливість уніфікації набору операцій її обробки. Завдяки саме їй СКБД перебирають на себе всі операції найнижчого рівня з пошуку, фільтрації та адміністрування даних. При цьому вивіреність та вдала стандартизація структури самого набору призводять до спроб її поширення на вищий щабель абстракції моделі процесів, яка реалізує весь ланцюжок технологічних операцій з вирішення задач бази даних.

Оскільки обидві згадані ієрархії мають різне походження, знаходження шляхів їх поєднання (комбінації) є нетривіальною задачею і потребує аналізу їхньої природи.

1.3.3.2. Дуальність моделі процесів

Об’єктно-орієнтоване програмування як застосування об’єктного підходу у царині комп’ютерної обробки даних описує довільний об’єкт у вигляді об’єднання списків його властивостей та процедур (методів). При цьому, окрім звичайних процесів окремих об’єктів передбачається можливість існування класових (групових) процедур, які задають спільну поведінку всіх екземплярів даного класу. Зрозуміло, що зміст цих списків визначається рівнем абстрагування (проекцією) функціонального призначення конкретних об’єктів у вибраній предметній області. Проте така проекція об’єкта реального світу на дискретну множину ресурсів комп’ютера призводить не тільки до відсікання малозначущих його властивостей, а й до появи нових, які характеризують саму проекцію. Так, наприклад, до “природних” даних про фондові капітали і операції банку додаються “штучні” властивості й особливості їх представлення на екрані монітора, способи зберігання, пошуку та обробки у комп’ютері.

Таким чином, приходимо до двох взаємодоповнюючих категорій процесів у системі, які надалі іменуватимемо природними та штучними відповідно.

1.3.3.3. Природна ієрархія моделі процесів

Природна ієрархія розглядає процес як елемент фізичної моделі задачі. Об'єкти похідного класу успадковують відповідні процеси батьківського із можливістю їх модифікації, а також доповнення списку власними процесами. В контексті прикладу з п.1.3.1 опису моделі даних, наведемо один із можливих варіантів такої ієрархії:

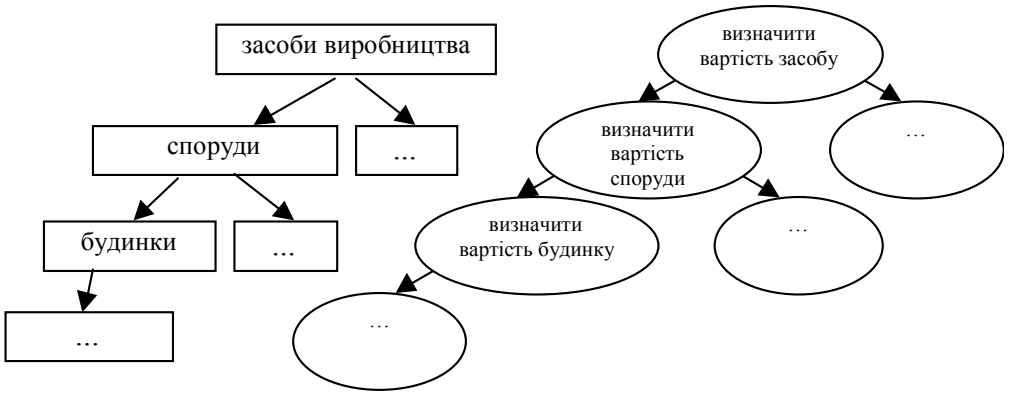

Рис.1.7. Приклад природної ієрархії процесів

Завдяки прямій відповідності між елементами структур даних та процесів програмна реалізація природної ієрархії легко вписується у запропоновану раніше схему. Для цього достатньо лише винести поля Form та Description з таблиці CLASSREF у нову таблицю, яка для кожного класу даних містить повний список можливих технологічних операцій із описом відповідних програмних форм. Запровадження нової операції ніяк не вплине на працездатність вже існуючого програмного коду, оскільки приведе лише до появи нової спадкованої форми і відповідного запису у таблиці операцій.

1.3.3.4. Штучна ієрархія

Даний тип ієрархії розглядає процес як об'єкт програмної моделі задачі. На відміну від природної штучна ієрархія визначається програмно-апаратним середовищем, у якому відбувається виконання процесів і з цієї причини значно краще надається для уніфікації програмного коду відносно самих різноманітних задач. Особливо це стосується програмного забезпечення баз даних, у яких програмне середовище не тільки мало залежить від фізичної постановки задачі, а й компонентно стабільне для більшості СКБД. Ця обставина підкреслює важливість побудови найкращої (серед багатьох можливих) штучної ієрархії. Для вирішення такої проблеми попередньо проаналізуємо анатомію узагальненого поняття процесу на предмет виділення спільних компонент (в сенсі програмної реалізації) для усіх процесів. У найбільш загальному випадку таких складових визначено чотири:

- *Модуль даних* (надалі *МД*) це складова процесу, яка забезпечує отримання вхідних даних від попереднього процесу і підготовку до їх наступної обробки

- *Аналітичний модуль* (надалі *АМ*) обробляє вихід з МД згідно з функціональним призначенням процесу
- *Модуль результату* (надалі *МР*) готує вихід з АМ до передачі наступним процесам.
- *Ітераційний модуль* (надалі *ІМ*) може виступати спеціалізованою частиною складного процесу, який повторює (ітерує) однотипні вкладені у нього підпроцеси

Зрозуміло, що навіть саму простеньку обробку інформації з БД із видачею результату на екран монітора можна представити у вигляді довгого ланцюжка процесів, МД першого з яких має отримати інформацію від БД, а МР останнього візуалізує результат.

Вказаний поділ процесу на складові елементи покладемо в основу майбутньої ієрархії. Замість одного дерева, у корені якого лежить найбільш узагальнене поняття процесу, будемо будувати чотири окремих. При цьому у початок кожного з них покладемо найбільш узагальнене поняття відповідного модуля.

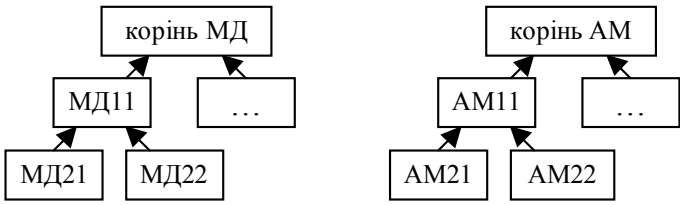

Рис.1.8. Приклад ієрархічних дерев модулів

Аналогічно виглядають ієрархічні дерева для решти модулів.

Тепер для кожного дерева необхідно конкретизувати кореневий елемент. Зрозуміло, що у найбільш загальному випадку зробити це досить непросто, оскільки складність такої задачі визначається бажаним рівнем універсальності кореневих елементів кожного дерева та ступенем деталізації виконуваних процесів. Тому успіх вирішення цієї проблеми сильно залежить від рівня кваліфікації та досвіду розробника. Найбільш логічним рішенням у даному випадку виглядає прив’язка функціональності кореневого елемента до предметної області конкретної інформаційної системи. Такий підхід дає змогу уникнути значного зменшення можливостей спадкування раніше створеного програмного коду внаслідок спрощення структури ієрархії. Зокрема авторами розглядався наступний спеціалізований випадок:

- процес не ділиться на ланцюжок підпроцесів
- МД є простим розіменуванням вказівника на об’єкт в оперативній пам’яті, а обмін даним з БД покладається на спеціалізоване дерево класів
- МР процесу виконує лише візуалізацію результатів на екрані монітора

Специфіка внесених спрощень, не зважаючи на їх значимість, приводить до важливого для програмної реалізації випадку: корінь дерева кожного модуля в головному визначає стартовий інтерфейс відповідної гілки ієрархії, а вузли дерева є модулями, які лише доповнюють початковий абстрактний інтерфейс

конкретною реалізацією. Це якнайкраще підходить до прийнятої у візуальному програмуванні стратегії опису технологічних процесів у вигляді ланцюжка спадкованих програмних форм.

Покажемо приклад формування процесу із модульних складових. Нехай для деякого класу даних (наприклад, “товар”) розроблено наступні модулі (для простоти зігноруємо їхніми спадковими відношеннями):

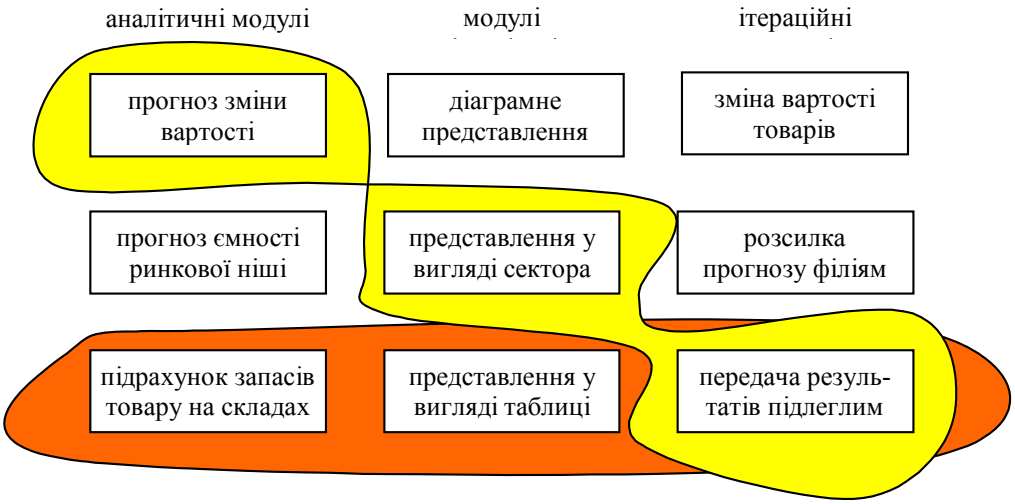

Рис.1.9. Приклад формування процесу з модульних складових

Послідовне виконання вибраних модульних складових реалізує потрібний технологічний процес.

Зауважимо, що хоча для побудови природної ієрархії процесів достатньо слідувати лише об’єктній структурі класів даних, у багатьох випадках для цього вигідно застосовувати описану вище методику поділу окремого процесу на складові елементи.

2. РЕАЛІЗАЦІЯ ОКРЕМИХ ПІДСИСТЕМ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ ЛЬВІВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ФРАНКА

В подальшому замість терміну "Корпоративна інформаційно-пошукова система Львівського національного університету імені Івана Франка" буде використовуватись скорочення ІПС (або ІС) та умовна назва "Кампус" (Campus – університетське містечко).

2.1. Загальний дизайн та карта інформаційно-пошукової системи "Кампус"

Одним з результатів робіт над проектом ІПС "Кампус" став новий дизайн Web-сайту Львівського університету, що представлений на рисунку:

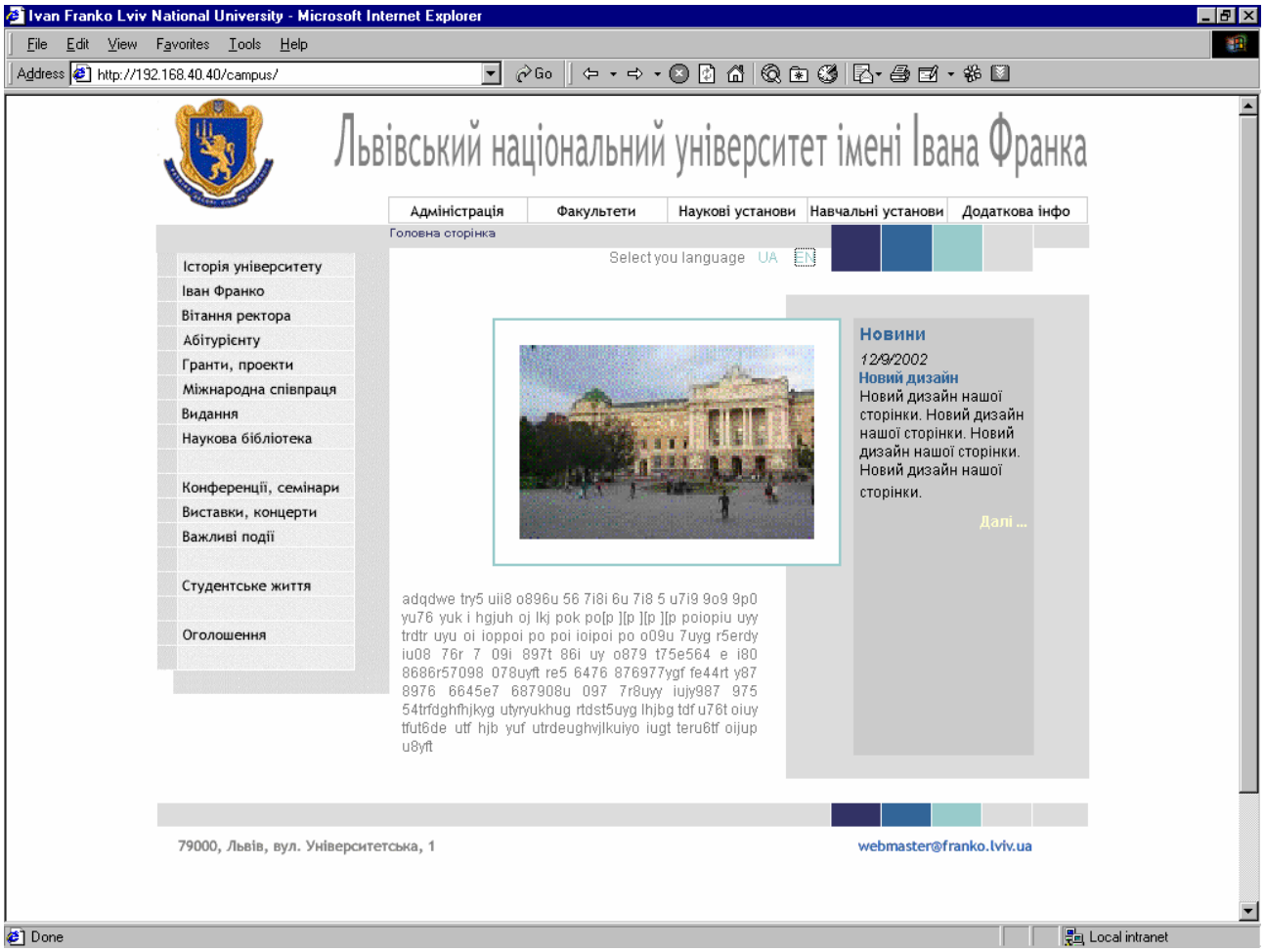

Рис. 2.1. Титульна сторінка Web-сайту університету та проекту "Кампус"

та карта Web-сайту університету (Табл.2.1).

Табл. 2.1. Верхні рівні карти Web-сайту університету

Вертикальна панель навігації	**Горизонтальна панель навігації**
⇒ **Історія університету** ⇒ **Іван Франко** ⇒ **Вітання ректора** ⇒ **Абітурієнту** ➢ Правила вступу ➢ Перелік спеціальностей ➢ План набору ➢ Пробне тестування ➢ ... ⇒ **Гранти, проекти** ⇒ **Міжнародна співпраця** ⇒ **Наукові видання** ➢ Вісник Львівського університету ➢ Збірники наукових праць ➢ Журнали ⇒ **Бібліотеки** ➢ Наукова бібліотека ➢ Студентські бібліотеки ⇒ **Конференції та семінари** ⇒ **Виставки, концерти** ⇒ **Важливі події** ⇒ **Студентське життя** ➢ Громадські організації ➢ Наукові гуртки ➢ Відпочинок ⇒ **Оголошення** ➢	⇒ **Адміністрація** ⇒ **Факультети** ➢ Біологічний ▪ Адміністрація ▪ Кафедри ▪ Спеціальності ▪ ... ➢ ... ➢ Юридичний ➢ Доуніверситетської підготовки ➢ Загальні кафедри ▪ Педагогіки ▪ ... ▪ Фізичного виховання ⇒ **Наукові установи** ➢ НДЧ ▪ Загальна інформація ▪ Науково-дослідні роботи ▪ ... ➢ Спеціалізовані учені ради ➢ Відділ аспірантури та докторантури ▪ Перелік спеціальностей ➢ ... ⇒ **Навчальні установи** ➢ Інститут ПКіПК ➢ Правничий коледж ➢ ... ⇒ **Довідкова інформація** ➢ Розташування підрозділів ➢ Телефонний довідник ➢ ...

2.2. Web-застосування "Інформаційний бюлетень"

Це застосування працює з сервером баз даних MySQL через механізм ASP, який підтримується web-сервером IIS 5 (входить в пакет Windows 2000). Застосування складається з двох частин – для представлення даних та для адміністрування.

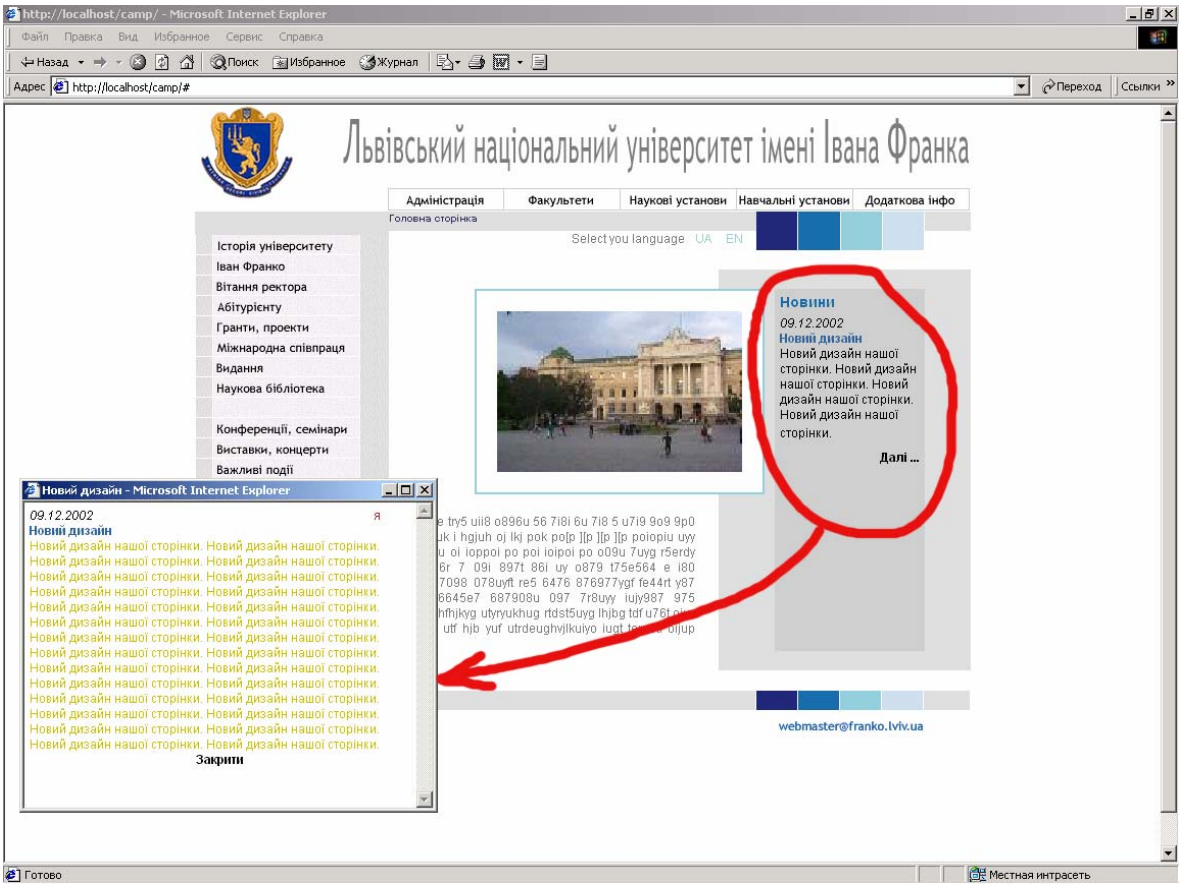

Рис. 2.2. Інтеграція системи "Інформаційний бюлетень" в сторінку університету

2.2.1. Структура інформації

Структура таблиць бази даних MySQL для функціонування web-застосування "Інформаційний бюлетень".

Табл. 2.2. Subdiv – зберігання інформації про підрозділи

Поле	**Призначення**
Id	Номер запису
Subdiv_name	Назва підрозділу
Subdiv_code	Кодова назва (латинськими літерами)
Subdiv_admin_name	ПІБ адміністратора підрозділу
Subdiv-admin_id	Login адміністратора

Скріпт для створення таблиці:

```
CREATE TABLE subdiv (
  id tinyint(4) NOT NULL auto_increment,
  subdiv_name CHAR(50) NOT NULL default '',
  subdiv_code CHAR(6) NOT NULL default '',
  subdiv_admin_name CHAR(50) default NULL,
```

```
 subdiv_admin_id CHAR(10) default NULL,
 PRIMARY KEY  (id)
);
```

Таблиці *_categ, *_design, *_messages, для зберігання інформації про категорії, оформлення та зміст повідомлень окремого підрозділу. У назві конкретних таблиць замість '*' фігурує кодова назва підрозділу з поля subdiv.subdiv_code відповідного запису в таблиці subdiv.

Табл. 2.3. *_categ – зберігання інформації про категорії повідомлень

Поле	**Призначення**
Id	Номер запису
Category_name_ua	Назва категорії українською мовою
Category_name_en	Назва категорії англійською мовою

Скріпт для створення таблиці:

```
CREATE TABLE *_categ (
 id tinyint(4) NOT NULL auto_increment,
 category_name_ua CHAR(50) default NULL,
 category_name_en CHAR(50) default NULL,
 PRIMARY KEY  (id)
) ;
```

Табл. 2.4. *_design – зберігання інформації про формат повідомлень

Поле	**Призначення**
Id	Номер запису
More_ua	Надпис на гіперпосиланні в короткому тексті українською
More_en	Надпис на гіперпосиланні в короткому тексті англійською
Close_ua	Надпис на гіперпосиланні для закриття вікна повного тексту українською
Close_en	Надпис на гіперпосиланні для закриття вікна повного тексту англійською
Full_bg_color	Колір фону для повного повідомлення
Css_path	Шлях до css-листа

Скріпт для створення таблиці:

```
CREATE TABLE *_design (
 id tinyint(4) NOT NULL auto_increment,
 more_ua CHAR(10) default 'Далі ...',
 more_en CHAR(10) default 'More ...',
 close_ua CHAR(10) default 'Закрити',
 close_en CHAR(10) default 'Close',
 full_bg_color CHAR(10) default 'white',
 css_path CHAR(50) default './',
 PRIMARY KEY  (id)
);
```

Табл. 2.5. *_messages – зберігання інформації про формат повідомлень

Поле	**Призначення**
Id	Номер запису
msg_categ	Id категорії запису
msg_start_date	Дата початку показу повідомлення
msg_expiry_date	Дата кінця показу повідомлення
msg_short_on	Ознака показу короткого тексту повідомлення
msg_short_ua	Короткий текст українською мовою
msg_short_en	Короткий текст англійською мовою
msg_full_ua	Повний текст українською мовою
msg_full_en	Повний текст англійською мовою
msg_author_ua	Автор українською мовою
msg_author_en	Автор англійською мовою
msg_header_ua	Заголовок українською мовою
msg_header_en	Заголовок англійською мовою

Скріпт для створення таблиці:

```
CREATE TABLE *_messages (
  id tinyint(4) NOT NULL auto_increment,
  msg_categ tinyint(4) NOT NULL default '0',
  msg_start_date datetime default NULL,
  msg_expiry_date datetime default NULL,
  msg_short_on tinyint(4) NOT NULL default '0',
  msg_short_ua text,
  msg_short_en text,
  msg_full_ua text,
  msg_full_en text,
  msg_author_ua CHAR(50) default NULL,
  msg_author_en CHAR(50) default NULL,
  msg_header_ua text,
  msg_header_en text,
  PRIMARY KEY  (id)
);
```

2.2.2. Модуль представлення даних

Модуль представлення даних складається зі сторінки для виведення короткого тексту повідомлень bbs-short.asp та для виведення повного тексту повідомлень bbs-single.asp. Bbs-short.asp спроектована таким чином, що інтегрується в довільну ASP-сторінку, розміщену на тому самому web-сервері шляхом звертання до об’єкта server.

```
<%Server.Execute /bbs/bbs-short.asp%>
```

Вхідні параметри передаються через об’єкти Session та Application

```
<%
  session("bbs.lang")="ua"         `мова повідомлень
  session("bbs.categ")=1`категорія повідомлень
  session("bbs.subdiv")="uni"     `підрозділ, якому належать повідомлення
`Параметри для підключення до сервера баз даних (файл global.asa)
  application("bbs.dsn") = "bbs"                    `DSN бази даних
  application("bbs.user") = "bogus"                 `користувач
  application("bbs.user.passwd") = "bogus"          `пароль користувача
%>
```

Цей користувач має право лише читати інформацію з бази даних.

Сторінка bbs-short.asp для форматування тексту використовує стилі CSS, які повинні бути визначенні серед стилів сторінки, в яку вона інтегрується

- bbs_link для форматування гіперпосилань;
- bbs_short для форматування короткого тексту;
- bbs_date для форматування дати;
- bbs_header для форматування заголовку повідомлення;
- bbs_categ для форматування назви категорії повідомлення.

Сторінка bbs-single.asp виводить форматований повний текст повідомлення в окремому вікні. Для форматування додатково використовуються стилі CSS:

- bbs_full для форматування повного тексту;
- bbs_author для форматування даних про автора.

Вхідні параметри передаються через URL, який формується строінкою bbs-short.asp.

Рис.2.2 ілюструє використання bbs-short.asp та bbs-single.asp для відображення новин.

2.2.3. Адміністративний модуль

Адміністративна частина застосування “Інформаційний бюлетень” дає можливості створення та видалення нових підрозділів та їх адміністраторів (для адміністратора застосування), а також створення, видалення та редагування повідомлень та їх категорій в межах одного підрозділу (для адміністратора підрозділу). Доступ до адміністративних сторінок авторизований і вимагає відповідної аутентифікації користувача бази даних.

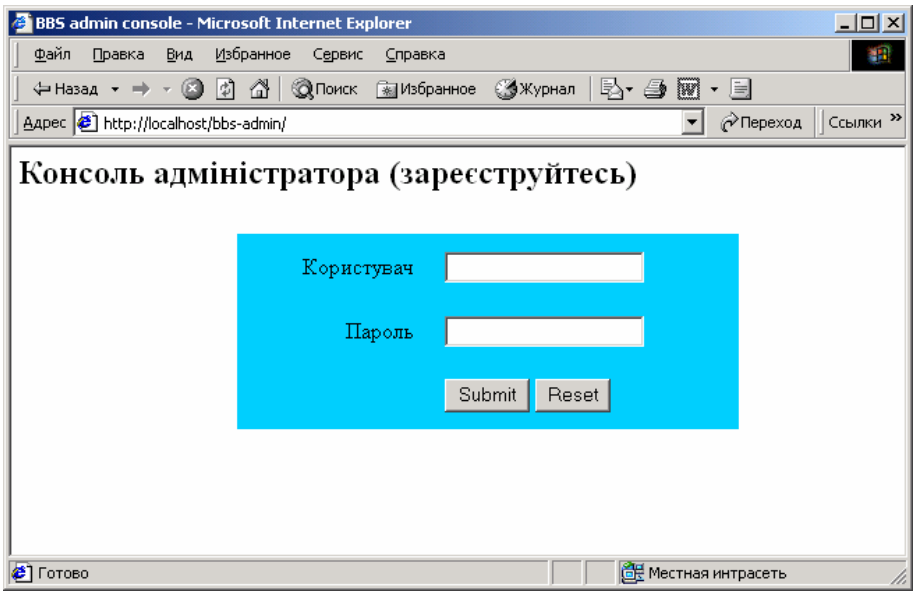

Рис.2.3. Реєстраційна форма

При створенні підрозділу створюється і його адміністратор, який має доступ лише до таблиць з бази даних, які відповідають його підрозділу. Таким чином забезпечується достатній рівень безпеки даних.

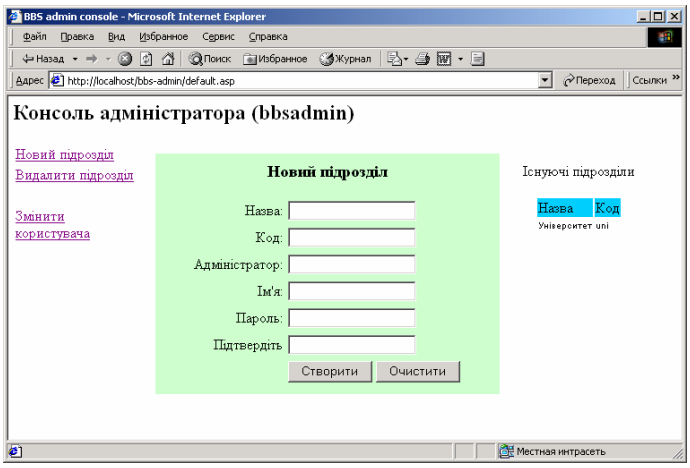

Рис.2.4. Форма реєстрації нового підрозділу

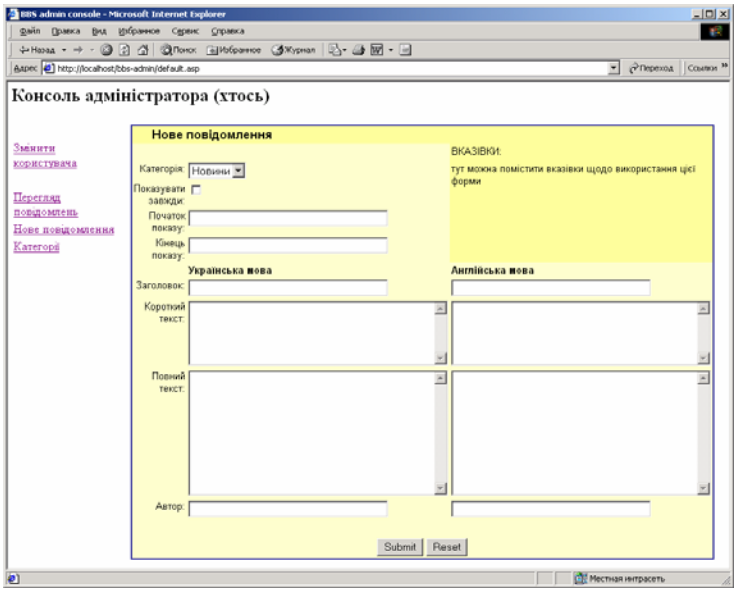

Рис.2.5. Реєстрація нового повідомлення

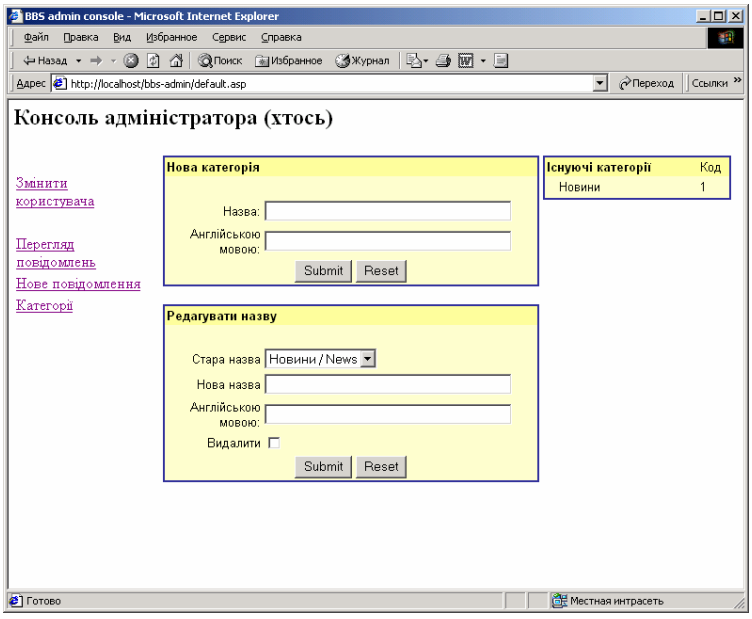

Рис.2.6. Створення нової категорії повідомлень

2.3. Модель університетської ІС на прикладі системи тестування

У даному розділі різні аспекти архітектури та функціонування університетської ІС досліджуються на прикладі системи тестування[13,18], яку за багатьма параметрами можна розглядати як модель для інших компонент.

Система є розподіленою. Web-сервер, сервери аплікацій та баз даних можуть розміщуватися на різних машинах, причому останній може бути не в одному екземплярі. Система підтримує такі групи функцій: адміністрування ресурсів і користувачів з санкціонуванням прав доступу, підготовку та зберігання у базі даних специфікованої за розділами інформації (у даному випадку – тестів різних видів), задання алгоритмів використання інформації (алгоритми тестування і аналізу); протоколювання роботи користувачів (робота з результатами).

Основний критерій класифікації користувачів грунтується на їхньому відношенні до інформації. В системі визначено такі категорії користувачів: адміністратори, компонувальники інформації (наприклад, тестів), керівники груп (наприклад, груп тестування) та звичайні користувачі, які завжди для роботи з системою мають належати принаймні до однієї з груп. Останні є найчисленішими користувачами системи, вони можуть динамічно об’єднуватися у різні групи для роботи з конкретними масивами інформації.

Логіка серверної частини реалізована за допомогою J2EE Enterprise JavaBeans (EJB). Тут використана основна перевага EJB-моделі – у визначенні стандартних операцій над компонентами сервера, включаючи підтримку сеансу взаємодії з клієнтом і роботу з базою даних. За аналогічною моделлю можна розробляти інші модулі університетської ІС, реалізуючи логіку роботи відповідних підрозділів учбового закладу з інформаційними потоками, а також їхню взаємодію.

Інтерфейс користувача з системою здійснюється за допомогою Web-браузерів. Такий вибір клієнтської частини сприяє швидкому освоєнню системи всіма категоріями користувачів. Інформація може компонуватися безпосередньо на сервері або завантажуватися з файла у спеціальному XML-форматі, розробленому для опису різних типів тестів. Такий підхід можна поширити на інші види інформації.

За відношенням до інформації з усієї множини функцій ІС можна умовно виділити такі три основні групи:

- комплектування документів;
- надання користувачам даних(документів) і засобів їхнього аналізу;
- адміністрування користувачів.

Такий поділ знайшов відображення в архітектурі ІС (рис.2.7). Вона складається з підсистем, які реалізують згадані групи властивостей. Підсистеми Completing і Using є практично взаємно незалежними, але кожна з них використовує типи і функції підсистеми адміністрування Admin.

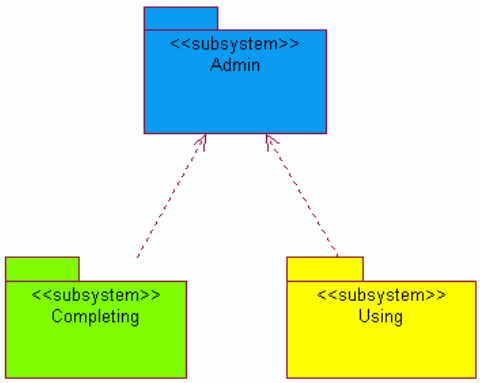

Рис.2.7. Підсистеми ІС

Поділ на підсистеми за відношенням до конкретних інформаційних потоків і функціонування за схемою комплектування-використання-аналіз дає змогу виділяти в ІС інші підсистеми(або їх модулі) і забезпечувати їх уніфіковане функціонування. Застосування xml-форматів дає змогу уніфіковано задавати будь-які види інформації.

2.2.1. Підсистема адміністрування Admin

На рис.2.8 подано діаграму моделі варіантів використання підсистеми Admin.

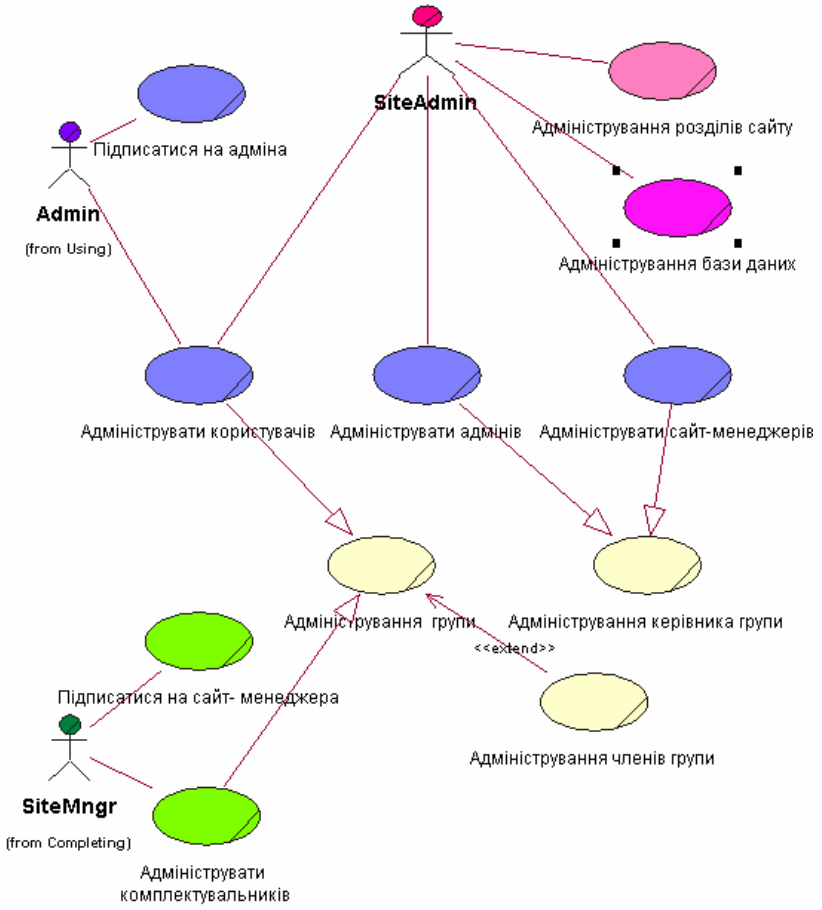

Рис.2.8. Діаграма варіантів використання підсистеми адміністрування

Підсистема адміністрування визначає, з одного боку, множини користувачів з різними правами доступу до документів, з другого– дає змогу керувати станом документів. Користувачі, які наділені правами адміністрування, представлені на сайті трьома акторами-адміністраторами:

- ***SiteAdmin*** – загальне керування політикою сайту, його ресурсами та іншими користувачами;
- ***SiteMngr*** – адміністратор розділу на сайті;
- ***Admin*** – керівник групи користувачів, які можуть лише використовувати документи.

SiteAdmin створює тематичні розділи в рамках сайту та призначає менеджерів розділів (SiteMngr) і адміністраторів використання (Admin). Перший є фактичним відповідальним за інформаційне наповнення розділу, другий – визначає способи використання документів. Ці дві категорії адміністраторів забезпечують децентралізацію сайту як по наповненню сайту документами, так і по керуванню різними користувацькими групами. Якщо користувач буде наділений правами обох адміністраторів, то це дасть йому змогу керувати процесом створення документів і їхнім використанням. Водночас SiteAdmin може адмініструвати всіх користувачів, забезпечуючи в такий спосіб повний контроль за доступом до сайту.

Адміністрування користувачів базується на функціях керування всією групою і її членами. Якщо адміністратор володіє правами керівника групи, то у випадку менеджера розділу це означає керування персоналом, який комплектує документи, а у випадку адміна – керування користувачами документів. Користувач отримує права адміністратора, якщо успішно завершується підписка – двосторонній процес, в ході якого користувач пропонує тему розділу, а адміністратор підтверджує цей запит і одночасно встановлює відповідні атрибути доступу для претендента.

2.2.2. Підсистема комплектування Completing

Діаграма моделі варіантів використання підсистеми Completing представлена на рис.2.9. Базовим типом актора в цій підсистемі є Completer, який володіє методами комплектування документів. Документ представляється у xml-форматі. Це дає змогу представити документ довільної структури і складності, який фактично сам себе документує. Його зміст можна безпосереднього формувати на сайті, або завантажувати з xml-файла, попередньо підготувавши за допомогою невеликої спеціальної програми, яку можна використовувати локально. Другий спосіб дає змогу залучати довільну кількість комплектувальників, не створюючи при цьому додаткового навантаження на сайт. Зазначимо, що безпосереднє редагування документу на сайті також виконується за допустимий час, оскільки EJB-компоненти є оптимізованими щодо роботи з базами даних. Іншою перевагою xml-файлів є зручність цієї форми зберігання документів поза базою даних. У такому вигляді інформація може передаватися в інші інформаційні системи та прикладні

програми, наприклад, для перегляду в Інтернет Explorer Browser, для передачі електронною поштою, для видруку тощо.

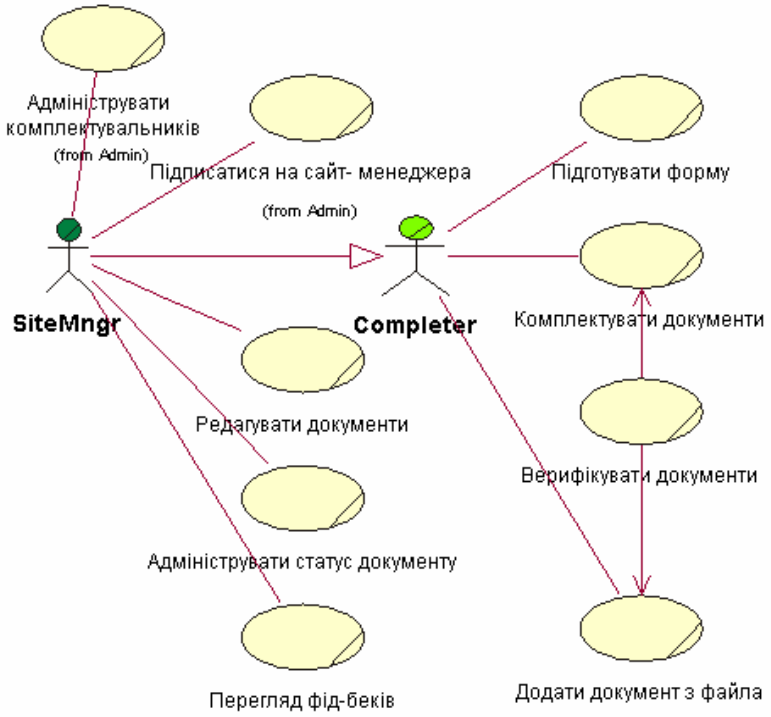

Рис.2.9. Варіанти використання підсистеми підготовки документів

Додатковими варіантами використання сайт-менеджера є встановлення статусу як цілого розділу, так і його складових – окремих документів. Це у першу чергу стосується етапу підготовки документів, оскільки лише на певному етапі готовності документ може стати доступним для використання різними категоріями користувачів. Якщо виникає потреба припинити на певний час доступ до документа певним категоріям користувачів, то для цього достатньо змінити лише його статус.

2.2.3. Підсистема опрацювання документів Using

Після логування на сайт користувач отримує доступ до тих документів, які санкціоновані для відповідного доступу в групах, у яких він є зареєстрованим. Наприклад, для тестування по одному з розділів необхідно, щоб адмін з підсистеми Using (рис.2.10), який у даному випадку має назву тестер, зареєстрував користувача у групі з доступним для тестування розділом. Результати тестування автоматично зберігатимуться в базі даних. Адмін фактично відповідає посаді керівника групи, яка має працювати з готовим документом. Отримати адміністраторські права адмін може за допомогою підписки на ці правами шляхом формування віповідного запиту до SiteAdmin.

Щоб забезпечити різні види алгоритмів роботи з документами, в ІС передбачена ієрархія типів. Користувач найзагальнішого типу User працює з документами шляхом перегляду і може заповнювати вибрані форми. Як додатковий вид зворотнього зв’язку використовується фідбек – коротке повідомлення адміністраторам розділу. Якщо потрібно, то користувач типу

QualifiedUser може розміщувати на сайті авторизований документ простої структури, наприклад, коментар, резюме, результати проведеного аналізу тощо. Ним встановлюється доступ до цього документу для членів групи, в якій сам автор зареєстрований.

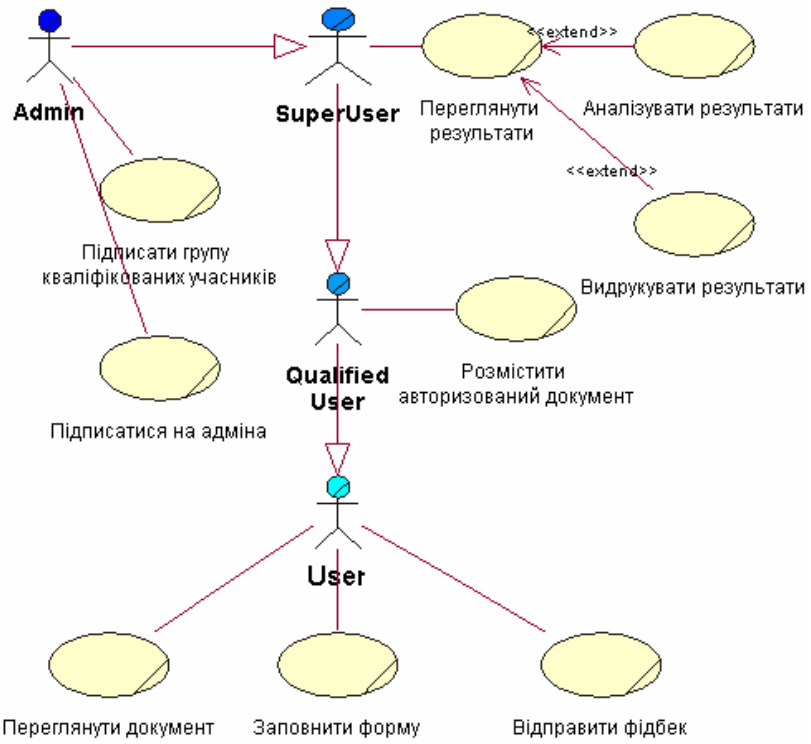

Рис.2.10. Діаграма варіантів використання підсистеми обслуговування користувачів

Дані форм, заповнених користувачем User, зберігаються в базі даних. Сайт може надавати адмінам різні способи їхнього використання. Зокрема, переглядати їх, робити передбачений аналіз і підводити по них певні підсумки з видруком результатів може користувач типу SuperUser. Фактично, користувачі цього типу можуть виконувати будь-які адміністративні функції щодо відповідної групи користувачів, крім адміністрування складу груп. Комплектувати групи може користувач типу Admin, який, у свою чергу, адмініструється безпосередньо актором типу SiteAdmin.

Усі три підсистеми описаної вище моделі реалізовані і пройшли апробацію у складі Web-сайта тестування [18]. Як сервер баз даних використано MySQL, який підтримує ліцензію GNU (цей факт є суттєвим у плані зменшення затрат на впровадження). Сайт відтестовано з метою перевірки його функціональності. На даний момент розроблено кілька розділів тестів з предметів “ОМ і програмування” та “Архітектура комп’ютерних мереж”.

У рамках університетської комп’ютерної мережі одночасна робота з сайтом близько півсотні користувачів відбувалася без помітної затримки. Аналіз продуктивності свідчить на користь використання застосованих технологій та розроблених класів для проектування та реалізації університетської ІС в повному обсязі.

2.4. Web-сайт "Наукові видання Львівського університету"

Прикладом незалежної підсистеми, що може працювати в складі проекту "Кампус" є Web-сайт "Наукові видання Львівського університету". Сайт запроектовано з орієнтацією на використання технології ASP. В даному звіті представлено структуру інформації та форми представлення інформації інформації кінцевому користувачу. Адміністративна частина системи знаходиться в стані тестування.

2.4.1. Структура інформації

Інформація про наукові видання структурована та пропонується до зберігання в базі даних СКБД Access такої структури:

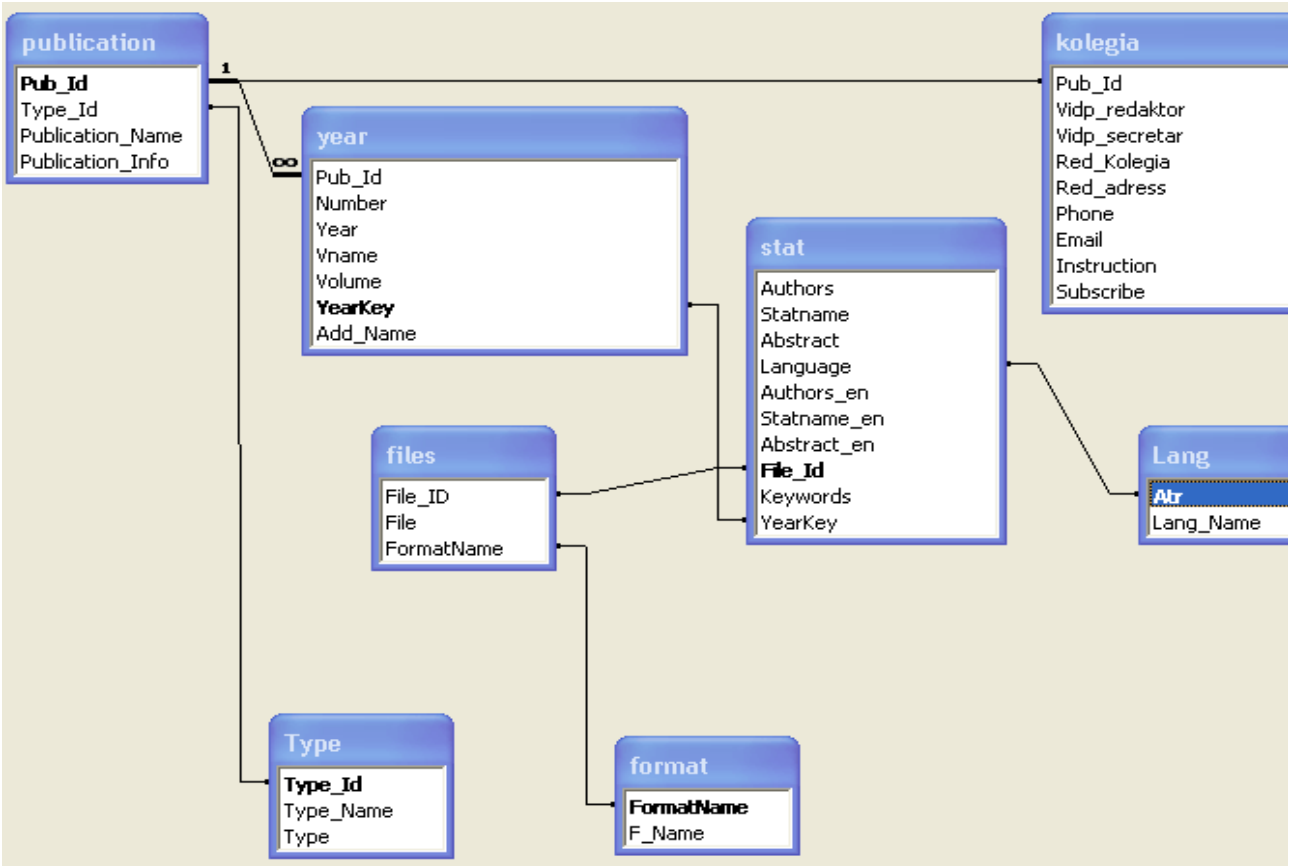

Рис.2.11. Структура бази даних сайту

Представлені на схемі таблиці мають таку структуру:

Табл. 2.6. Таблиця Publication - інформація про видання

Pub_Id	Лічильник	Ідентифікатор видання
Type_Id	Числове long int	Вказівник на тип видання
Publication_Name	String 255	Назва видання
Publication_Info	Memo	Додаткова інформація про видання

Табл. 2.7. Таблиця Type - словник типів видань

Type_Id	Числове long int	Вказівник на тип видання
Type_Name	String 50	Назва типу
Type	String 50	Вказує яким чином виходить видання (випуски, номери)

Табл.2.8. Таблиця Kolegia - інформація про редакційну колегію видання

Pub_Id	Long Int	Ідентифікатор видання
Vidp_redaktor	String 100	Відповідальний редактор
Vidp_secretar	String 100	Відповідальний секретар
Red_Kolegia	Memo	Редакційна колегія
Red_adress	Memo	Адрес редколегії
Phone	String 50	Телефон
Email	String 50	Електронна пошта
Instruction	String 255	URL інструкцій для авторів
Subscribe	String 255	URL умов підписки на це видання

Табл.2.9. Таблиця Year - інформацію про випуск (том) видання

Pub_Id	Long Int	Ідентифікатор видання
Number	Long Int	Номер
Year	Long Int	Рік видання
Vname	String 255	Загальна назва
Volume	Long Int	Том
YearKey	Лічильник	Ідентифікатор
Add_Name	String 255	Додаткове ім`я

Табл.2.10. Таблиця : Stat - інформація про публікації (статті) випуску (тому)

Authors	String 255	Автори статті
Statname	String 255	Назва статті
Abstract	Memo	Анотація
Language	String 2	Вказівник на мову статті
Authors_en	String 255	Автори статті (англ. мовою)
Statname_en	String 255	Назва статті (англ. мовою)
Abstract_en	Memo	Анотація (англ. мовою)
File_Id	Лічильник	Ідентифікатор на повний варіант статті
Keywords	String 255	Ключові слова
YearKey	Long Int	Ідентифікатор статті

Табл.2.11. Таблиця Files – інформація про розташування (URL) файлів зі статтею

File_Id	Long Int	Ідентифікатор на повний текст статті
File	String 255	URL документу
FormatName	Long Int	Тип документу (HTML, PDF, DOC)

Табл.2.12. Таблиця Format - словник типів форматів

FormatName	Long Int	Ідентифікатор формату
F_Name	String 50	Назва формату

Табл.2.13. Таблиця Lang - словник мов, якими може бути написана стаття

Atr	String 2	Ідентифікатор мови статті
Lang_Name	String 255	Мова якою написана стаття

2.4.2. Представлення інформації

Web-сайт розміщено за адресою http://blues.franko.lviv.ua/publish/
Представлення інформації користувачу ілюструють рис. 2.12 – 2.18.

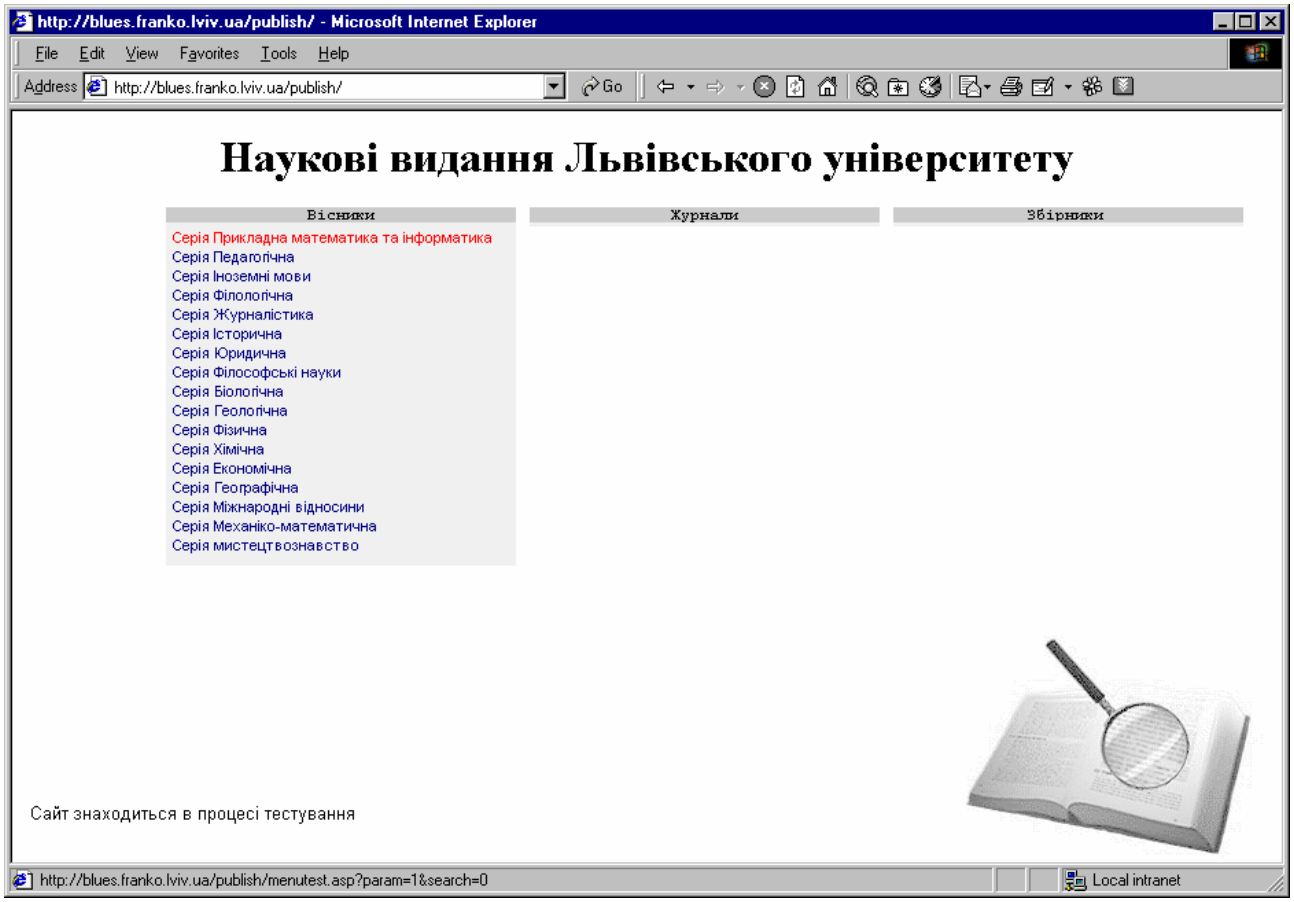

Рис.2.12. Титульна сторінка Web-сайту "Наукові видання Львівського університету"

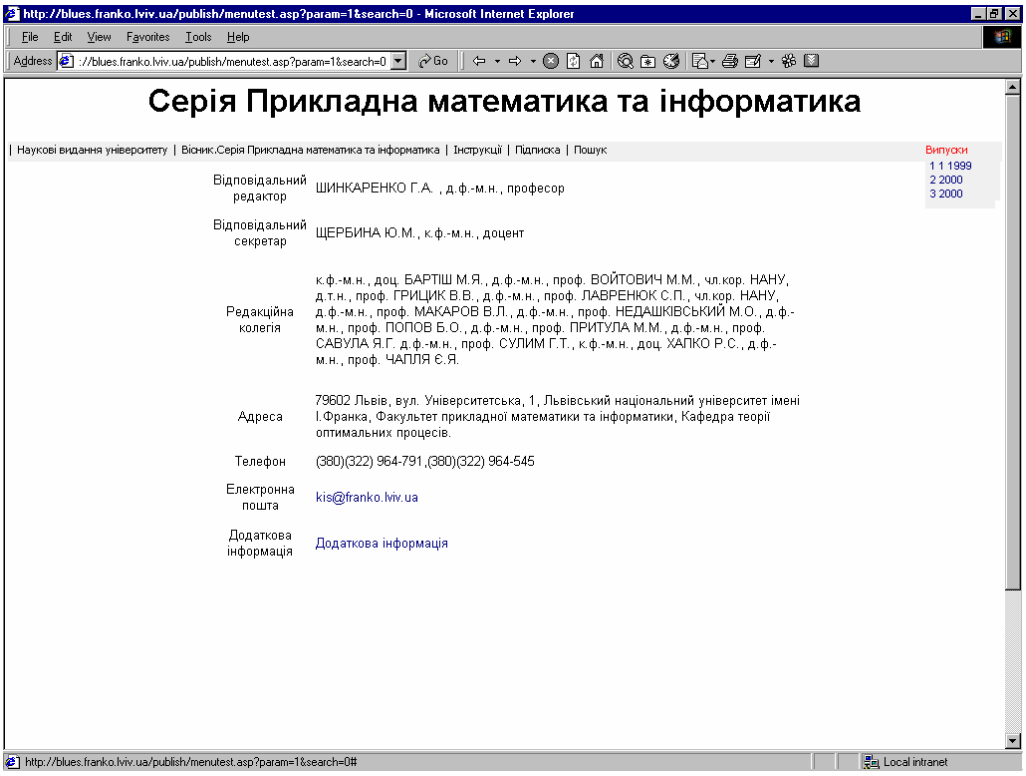

Рис.2.13. Сторінка Вісника серії "Прикладна математика та інформатика"

Рис.2.14. Зміст випуску

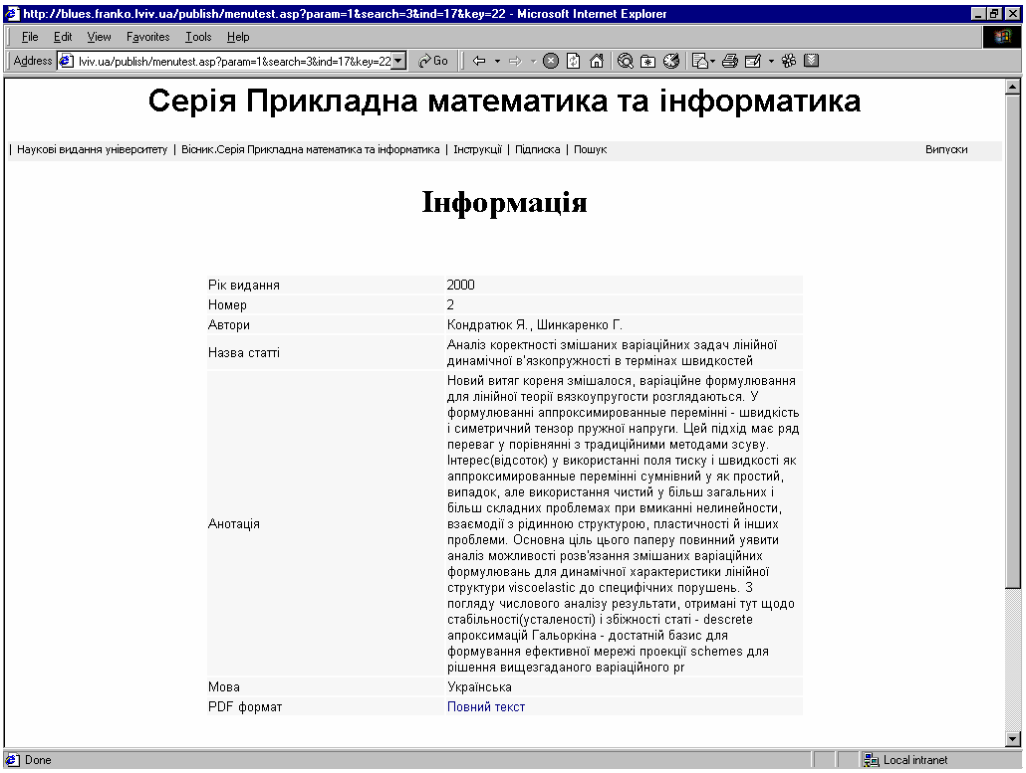

Рис.2.15. Інформація про статтю

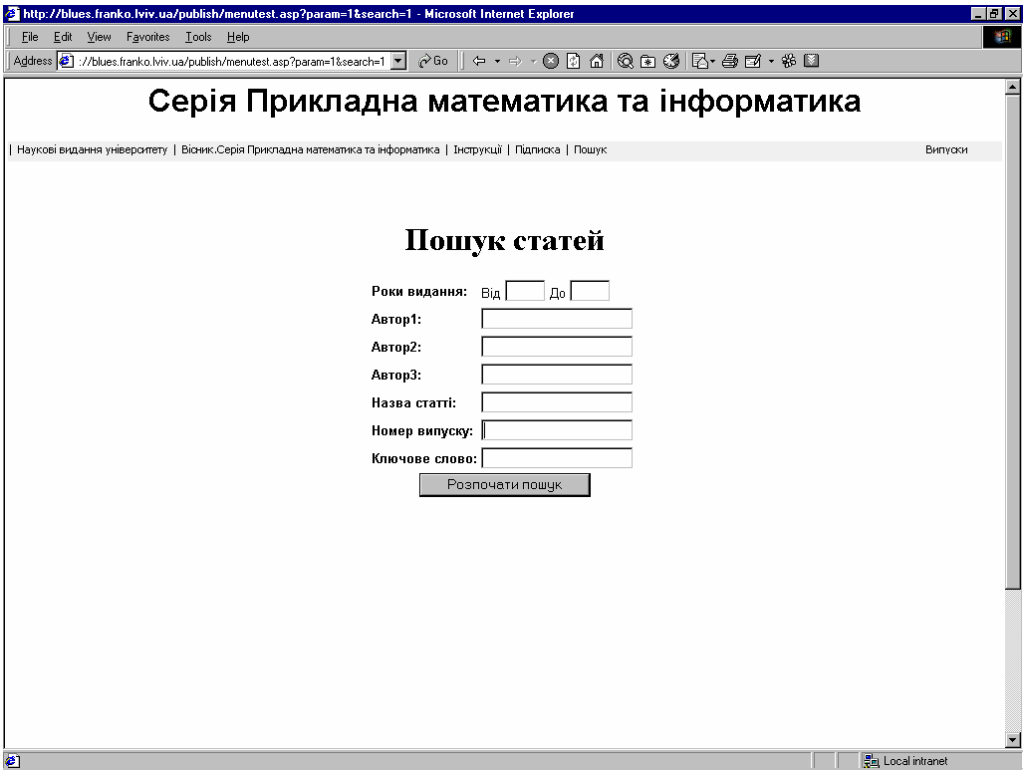

Рис.2.16. Форма пошуку публікацій

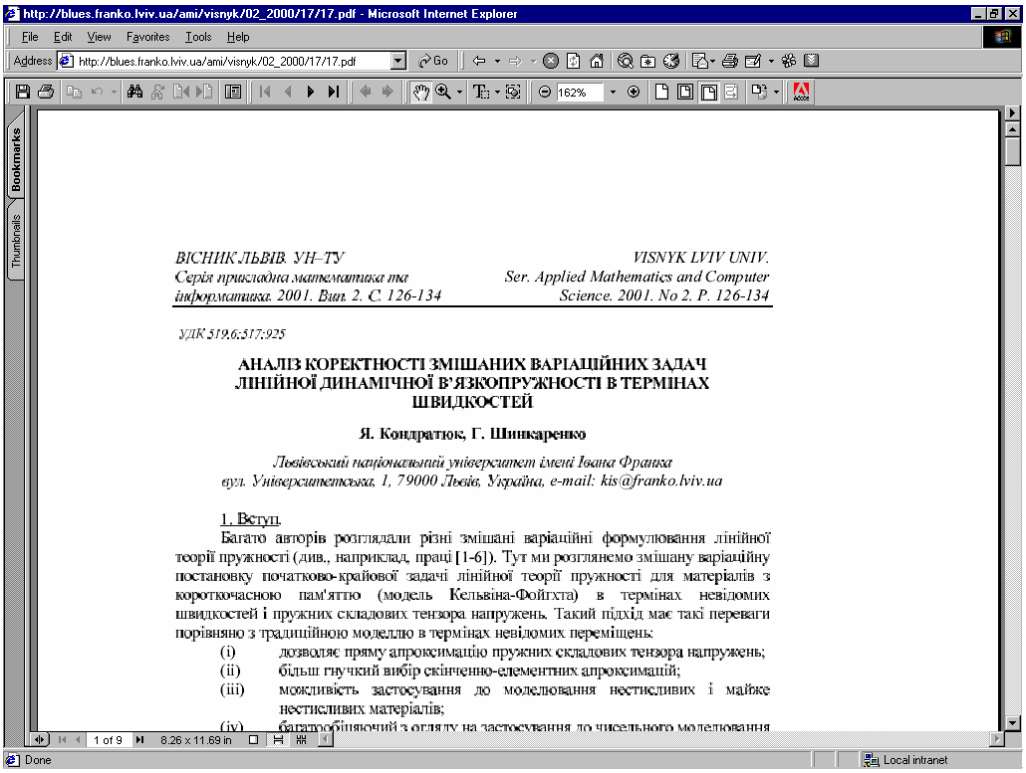

Рис.2.17. Стаття в pdf-форматі

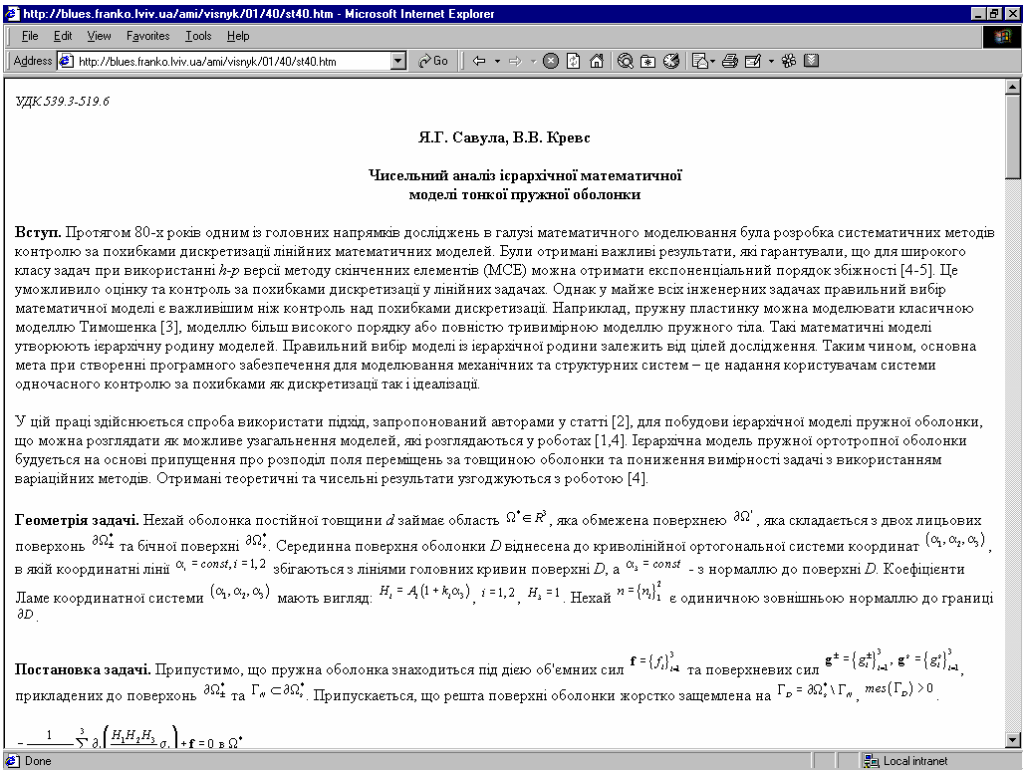

Рис.2.18. Стаття в html-форматі

2.5. База даних наукових публікацій

Web-застосування для пошуку та ведення переліку публікацій створено як ISAPI-застосування на платформі Windows.

2.5.1. Структура інформації

База даних публікацій представляє собою одну таблицю db-формату:

Табл.2.14. Структура таблиці Papers.db

Назва поля	Зміст
KLUST	тип публікації
AVTORY	автори статті
NAZVA	назва публікації
INFPTEX	назва журналу, видання
VYDAVN	місце видання, видавництво
RIK	рік видання
TOM	номер тому
NOMER	номер журналу
PERSTOR	перша сторінка статті в журналі
OSTSTOR	остання сторінка статті в журналі
MOVA	мова, на якій написана стаття
URL	адреса HTML-версії статті

2.5.2. Представлення інформації

Сторінка пошуку інформації має такий вигляд:

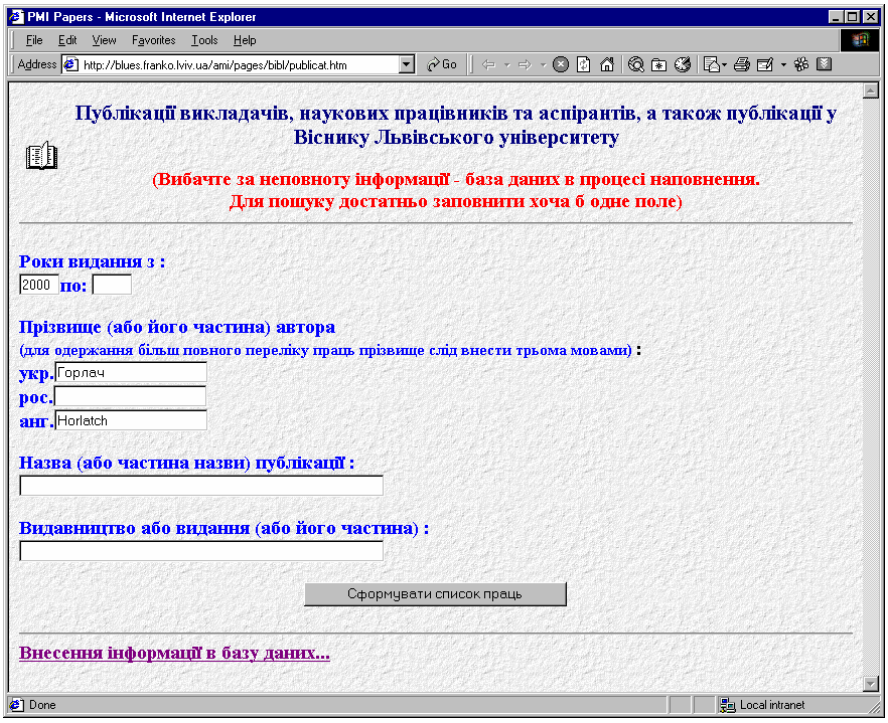

Рис.2.19. Титульна сторінка пошуку публікацій

Рис.2.20 ілюструє представлення інформації (відповідає стандартним вимогам до запису посилань та впорядкована в хронологічному порядку) за результатами виконання запиту.

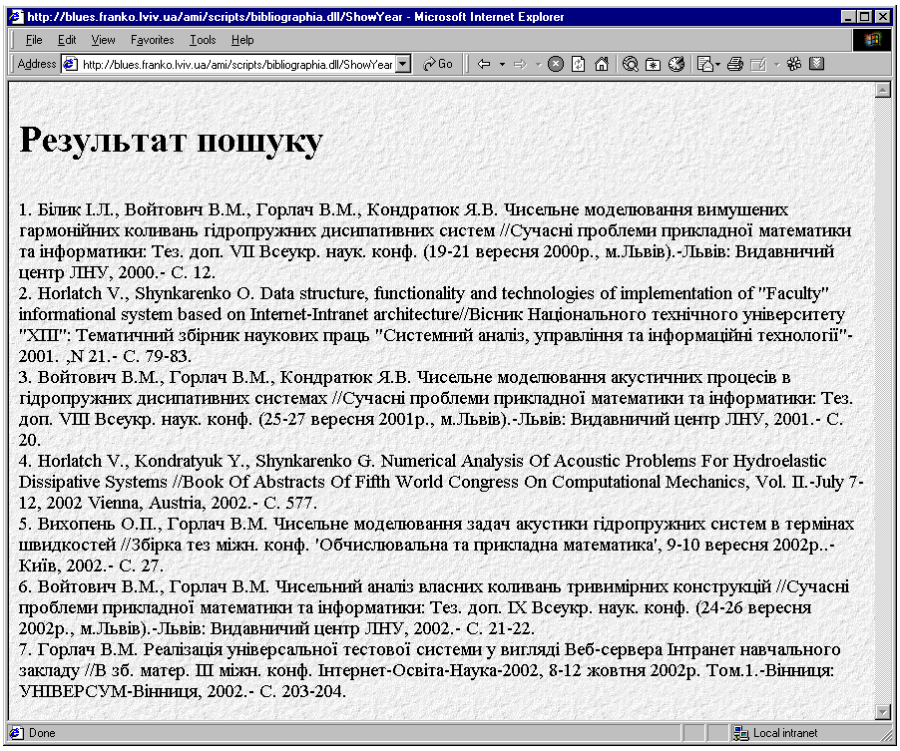

Рис.2.20. Результат виконання запиту

2.5.3. Внесення інформації

Внесення інформації здійснюють уповноважені адміністратором системи користувачі. Форма реєстрації користувача представлена на рисунку:

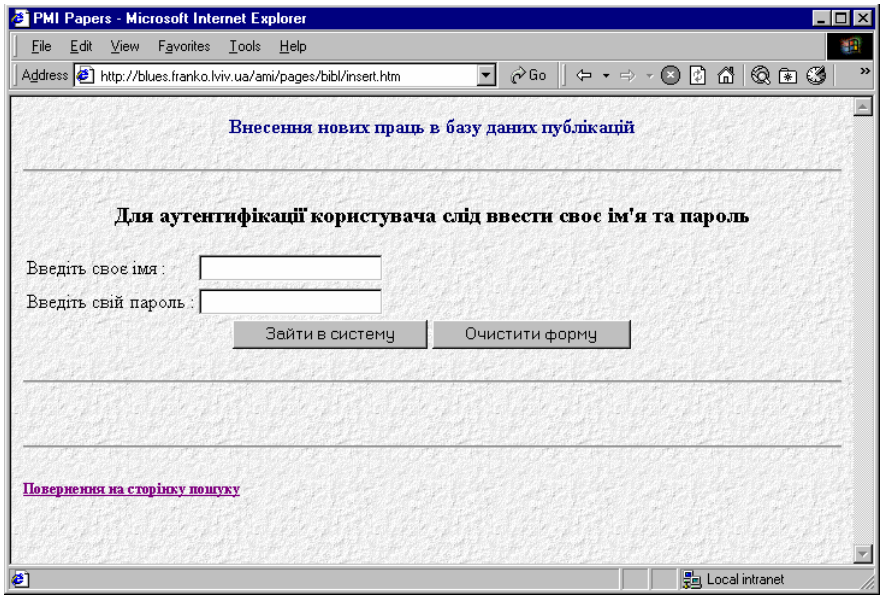

Рис.2.21. Реєстрація користувача для внесення інформації

Відповідно до класифікації (рис.2.22) публікацій створено відповідну кількість форм внесення інформації у відповідні поля таблиці Papers.

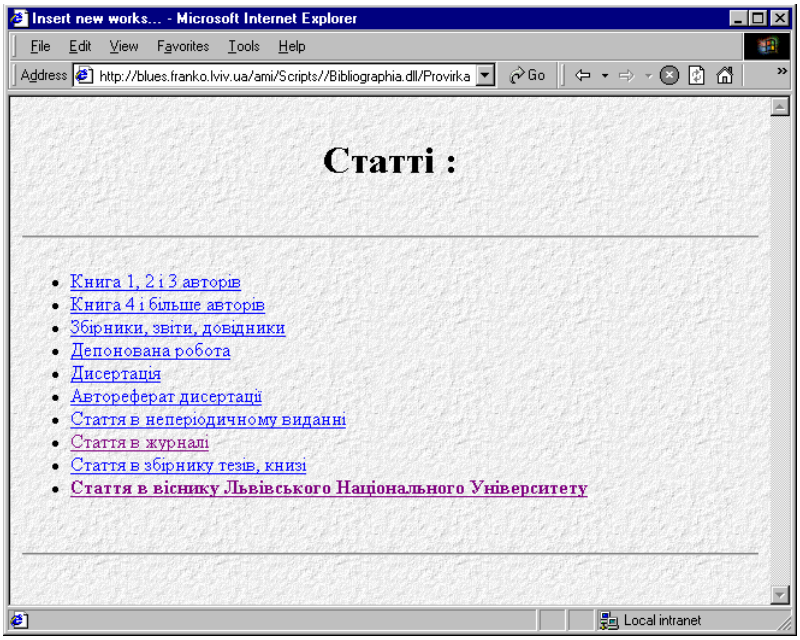

Рис.2.22. Вибір типу публікації

Рис.2.23 ілюструє одну з можливих форм внесення інформації.

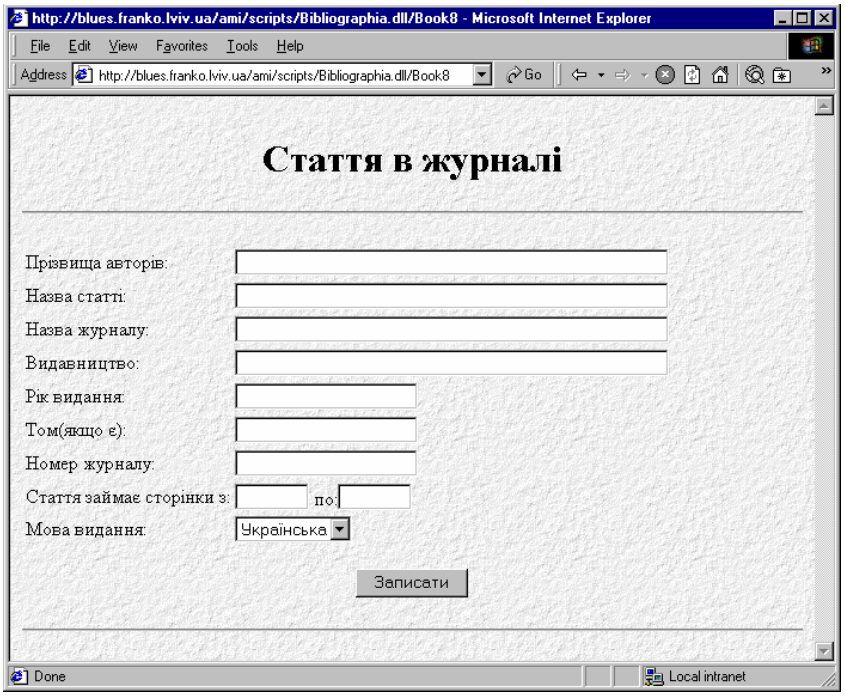

Рис.2.23. Форма внесення інформації про публікацію в журналі

База даних впроваджена на факультеті прикладної математики та інформатики Львівського національного університету імені Івана Франка та активно наповнюється інформацією. Сьогодні обсяг бази – біля тисячі записів.

2.6. Реалізація універсальної тестової системи у вигляді Web-сервера інтранет навчального закладу

Розглянуто принципи побудови та структуру універсальної системи тестування-опитування респондентів, яка реалізована у вигляді Web-сервера.

Система тестування-опитування здатна вирішувати широке коло задач, зокрема використовуватись для психодіагностики особи, оцінки інтелекту та рівня знань, розв’язання задач атестації і підбору кадрів, вивчення соціально-психологічного клімату в колективі, опитування громадської думки тощо.

Особливо важливу роль така система відіграє в навчальних закладах для поточної та підсумкової оцінки знань. Реалізація такої системи у вигляді Web-сервера інтранет навчального закладу дозволяє використовувати її як у будь-якому комп'ютерному класі навчального закладу, так і з-за його меж, зокрема – як частину системи дистанційного навчання. Клієнтською частиною виступає будь-який Інтернет-браузер, що суттєво спрощує процес освоєння системи кінцевим користувачем.

2.6.1. Формалізація тесту - опитувача

Під тестуванням будемо розуміти набір питань (тверджень), до кожного з яких додається перелік фіксованих варіантів відповідей з деякою числовою характеристикою.

Нехай T(i,n)-кількість балів, що нараховуються за n-у відповідь на i-е питання. Опитувач можна представити у вигляді набору P записів типу

< питання i >

< відповідь (i,1) > T(i,1)
< відповідь (i,2) > T(i,2)
. . .
< відповідь (i,N(i)) > T(i,N(i)) ,

де N(i) - кількість варіантів відповідей на i-е питання.

Первинним результатом тестування є числовий вектор:

$$S = (s(1), s(2), ... , s(M))$$

де s(l), l=1,...,M - сума набраних респондентом балів при відповідях на деяку l-у підмножину питань, що називається шкалою, а M - число шкал опитувача.

На практиці тестування часто зустрічається поняття корекції первинних результатів. Нехай K - деяке число, фіксоване для конкретного опитувача, або таке, що являється первинним результатом. Під скоригованим первинним результатом тестування розуміємо числовий вектор

$$V = (v(1), v(2), ... , v(M)) ,$$

де

$$v(l) = s(l) + t(l) * K \qquad l=1, ..., M,$$

а t(l) - коефіцієнти корекції, що задаються при розробці опитувача.

При обробці результатів тестування первинні (скориговані) оцінки часто приводять шляхом лінійного перетворення до єдиної шкали вимірювань.

Отримані оцінки назвемо вторинним результатом тестування і будемо позначати вектором

$$R = (r(1), r(2), ... , r(M))$$

де

$$r(l) = a(l)*v(l) + b(l) , \quad l=1, ..., M,$$

а a(l), b(l) - задані при визначенні опитувача числові значення.

І, накінець, результатом тестування є текстова, числова або деяка інша інформація про респондента, яка визначається набором записів інтерпретації, що являються частиною тексту, в залежності від значень вторинних результатів.

Під спрощеним опитувачем будемо розуміти опитувач з наведеною вище структурою, якщо M=1, тобто первинним результатом тестування є сума S набраних респондентом балів при відповідях на всі питання, а вторинним результатом – число:

$$R = a * s + b ,$$

де a, b - деякі задані в опитувачі числа (звичайно a=1,b=0, тобто R=S).

Результатом тестування є текстова, числова або деяка інша інформація, що відповідає тому числовому інтервалу, в який попадає оцінка R.

В структуру баз даних пропонованої системи тестування можна додавати довільні тести, що можуть бути описані вказаним вище способом.

2.6.2. Структура інформації

Для зберігання інформації в системі використовуються база даних СКБД Access.

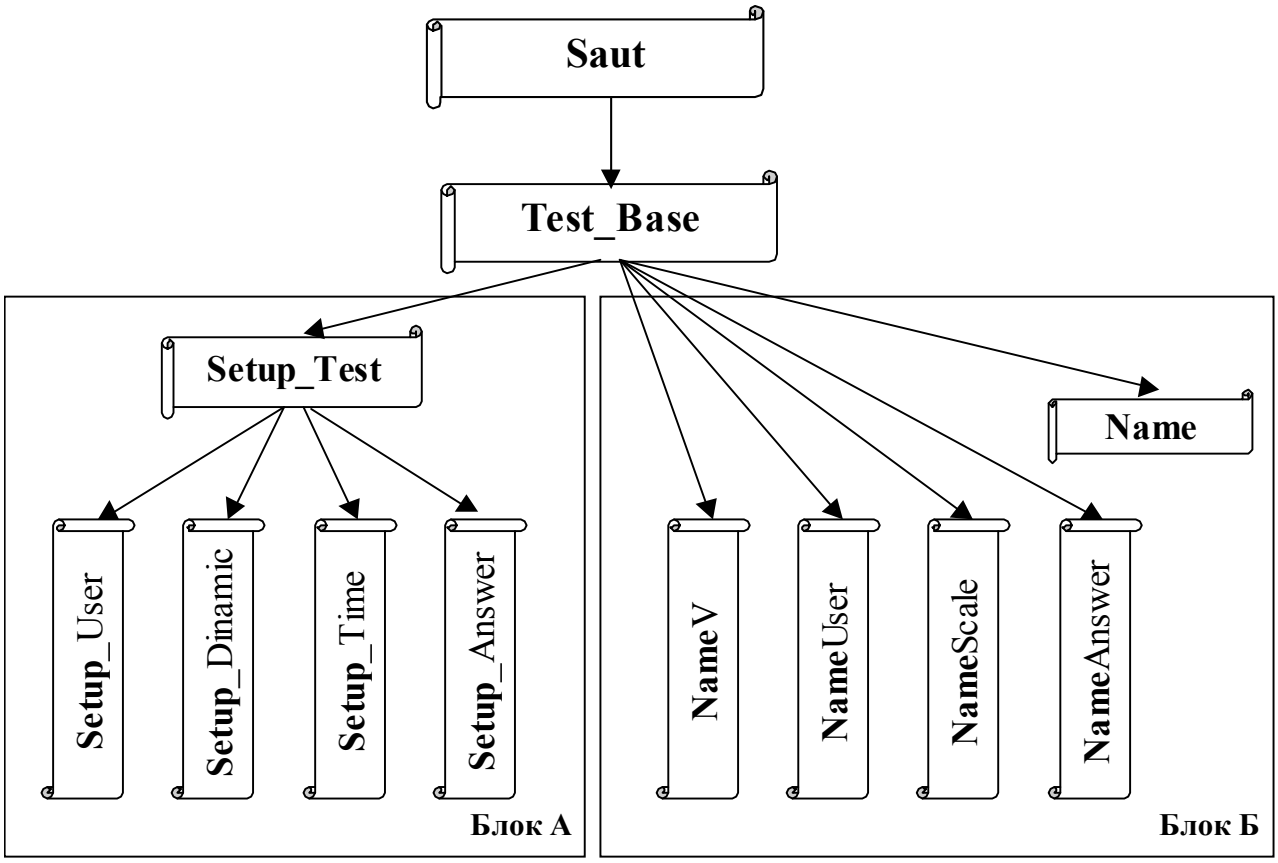

Рис. 2.24. Структура інформації

Під сайтом будемо розуміти деяку підмножину тестів, згрупованих за певною ознакою (наприклад за темою, або за курсами, які читаються для певної спеціальності чи певним викладачем).

В основі ієрархії таблиць системи лежить таблиця **Saut**. тобто ця таблиця головним чином призначена для зберігання інформації про групи тестів.

Базова таблиця тестів **Test_Base** містить основну інформацію про певний тест. З одного боку, за допомогою ключового поля вона зв`язана з таблицями, що визначають властивості тестів (Блок А), а з іншого боку, за допомогою поля імені тесту, вона взаємодіє з таблицями, які містять дані конкретного тесту (Блок Б).

До таких даних входять:

- умови завдань
- варіанти відповіді та їх числова характеристика
- дані про респондентів
- база результатів тестування та інша інформація.

Таких таблиць блоку Б є стільки ж, скільки тестів в системі, причому Name - означає ім`я конкретного тесту, що задане в таблиці Test_Base. Всі ці таблиці створюються динамічно для кожного тесту, що додається в систему.

2.6.3. Опис таблиць бази даних

В вершині ієрархії бази даних лежить таблиця **Saut** з інформацією про сайти системи.

Табл. 2.15. Таблиця Saut

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Name	CHAR(15)	Назва
List	CHAR(250)	Список номерів сайту
Passw	CHAR(15)	Пароль виклику сайту
Css_Path	CHAR(15)	Директорія знаходження сайту
Css_File	CHAR(90)	Назва CSS-файлу

Базовою таблицею тестів є **Test_Base**.

Табл. 2.16. Таблиця Test_Base

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Name	CHAR(60)	Назва тесту
Title	MEMO	Опис проведення тесту
Name_file	CHAR(20)	Ім'я файлу тесту, для зв'язку з блоком Б
Comment	CHAR(90)	Коментар

Таблиці блоку А мають такі структуру та призначення:

Setup_Test - таблиця в основі якої лежать налаштування основних властивостей системи.

Табл. 2.17. Таблиця Setup_Test

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Type	CHAR(10)	Елементи вибору питань
D_Type	CHAR(10)	Динамічний/Статичний тест
Time	CHAR(10)	Кількість хвилин
Present	CHAR(10)	Видача питань по одному/всі разом
User	CHAR(10)	Чи проводити реєстрацію користувача
Timer	CHAR(5)	Визначення часу складання тесту

Таблиця **Setup_User** містить інформацію про поля реєстрації користувача, кожного з тестів.

Табл. 2.18. Таблиця Setup_User

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Us1	CHAR(5)	Присутність поля прізвища
Us2	CHAR(5)	Присутність поля імені
Us3	CHAR(5)	Присутність поля по-батькові
Us4	CHAR(5)	Присутність поля рік народження
Us5	CHAR(5)	Присутність поля спеціальність
Us6	CHAR(5)	Присутність поля домашня адреса
Add_Filds	CHAR(250	Присутність додаткових полів
Kil	INTEGER	Кількість спроб складання тесту

Таблиця **Setup_Dinamic** містить інформацію про налаштування динамічних тестів, основні характеристики вибору питань.

Табл. 2.19. Таблиця Setup_Dinamic

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Coun	INTEGER	Кількість вибраних питань
P1	CHAR(5)	Видача питань по рівнях складності
P11	CHAR(10)	Кількість рівнів складності
P12	CHAR(15)	Кількість питань кожного рівня складності
P2	CHAR(5)	Видача питань за групами приналежності
P21	CHAR(5)	Кількість питань з групи приналежності

Таблиця **Setup_Time** містить інформацію про часовий проміжок, в якому дозволено проводити тестування.

Табл. 2.20. Таблиця Setup_Time

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
T1	CHAR(5)	Чи визначати дні проведення тесту
T2	CHAR(5)	Початкова дата
T3	CHAR(5)	Кінцева дата
T4	CHAR(5)	Чи визначати години проведення тесту
T5	CHAR(5)	Початковий час
T6	CHAR(5)	Кінцевий час

Таблиці блоку Б мають такі призначення:

Таблиця **Name** зберігає інформацію про умови питань тесту з назвою Name, правильні відповіді на них, їх числову характеристику та інші параметри.

Табл. 2.21. Таблиця Name

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код питання
Ymova	MEMO	Текст питання
Answer	CHAR(6)	Правильна відповідь
Ball	INTEGER	Числова характеристика
Group	CHAR(5)	Приналежність питання групі
Premitka	CHAR(40)	Примітка
Options	CHAR(10)	Додаткові параметри

Таблиця **NameV** містить інформацію про варіанти відповідей на питання (якщо така інформація є присутня в даному тесті). Ця таблиця зв`язана з таблицею Name з допомогою ключового поля Unic зв`язком один до багатьох.

Табл. 2.22. Таблиця NameV

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код питання
Un_variant	INTEGER	Код варіанта відповіді
Ymova	MEMO	Текст
Ball	INTEGER	Числова характеристика варіантів відповідей
Premitka	CHAR(40)	Примітка

В таблиці **NameUser** міститься інформація про користувачів, їх успішність та дата складання тесту.

Табл. 2.23. Таблиця NameUser

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код респондента
Coun	INTEGER	Кількість спроб тестування
Score	INTEGER	Кількість набраних балів
D	DATA	Дата тестування
Us1	CHAR(30)	Статичні поля зберігання інформації про користувача. Поле –1
- " -	- " -	- " -
Us6	CHAR(30)	Поле – 7
Add1	CHAR()	Перше додаткове поле інформації про користувача
- " -	- " -	- " -
Add n	CHAR()	n- додаткове поле
Ans1	CHAR (250)	Запитання на які давав відповіді респондент
Ans2	CHAR (250)	Варіанти відповідей, які вибирав респондент

Таблиця **NameScale** містить інформацію про оцінювання користувача.

Табл. 2.24. Таблиця NameScale

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Код
Text	MEMO	Текст шкали
Sc1	CHAR(6)	Числовий проміжок
Sc2	CHAR(6)	Числовий проміжок
Name	CHAR(10)	Ім'я шкали

В таблиці **NameAnswer** запам'ятовується список питань, на які відповідав користувач.

Табл. 2.25. Таблиця NameAnswer

Назва поля	Тип даних	Пояснення
Unic	INTEGER	Номер тесту
List	CHAR (250)	Список номерів питань що ввійшли в цей тест

2.6.4. Опис інтерфейсу адміністратора

Застосування реалізовано двома незалежними частинами – адміністративною, що працює на комп'ютері викладача під час підготовки та налаштувань того чи іншого тесту – та клієнтською, що представляє собою Web-застосування для проведення тестувань.

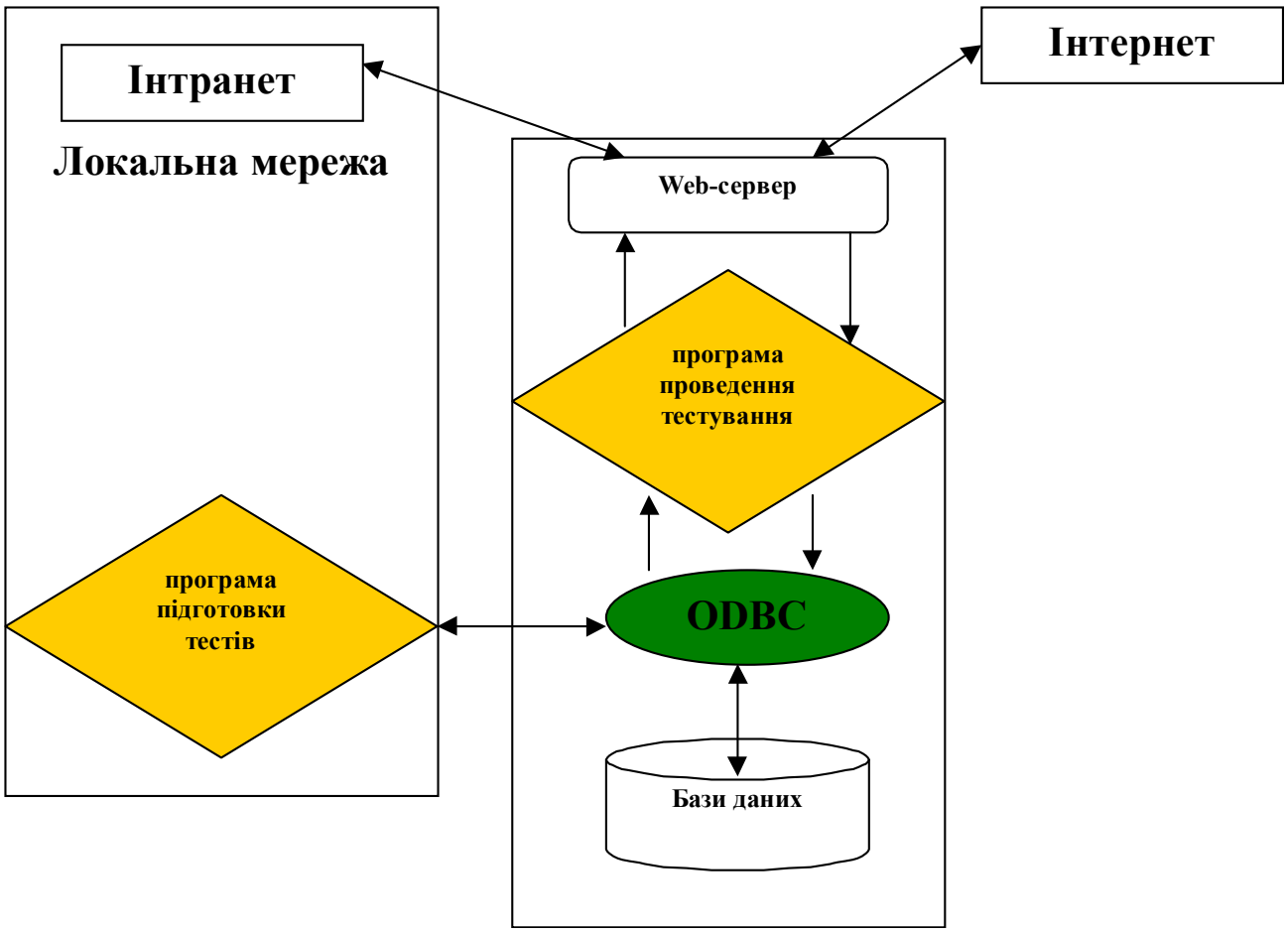

Рис.2.25. Схема функціонування системи

Рисунки 2.26 – 2.30 ілюструють окремі форми адміністративної частини для підготовки та налаштувань тестів та окремих сайтів (груп тестів).

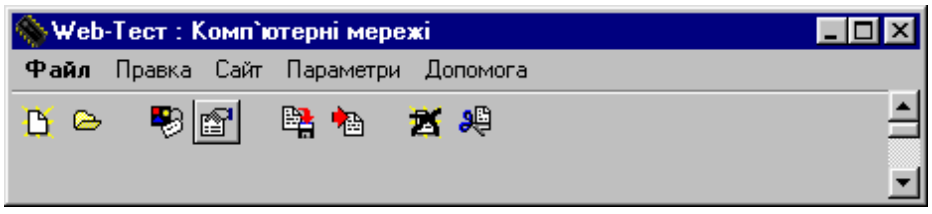

Рис.2.26. Консоль адміністратора

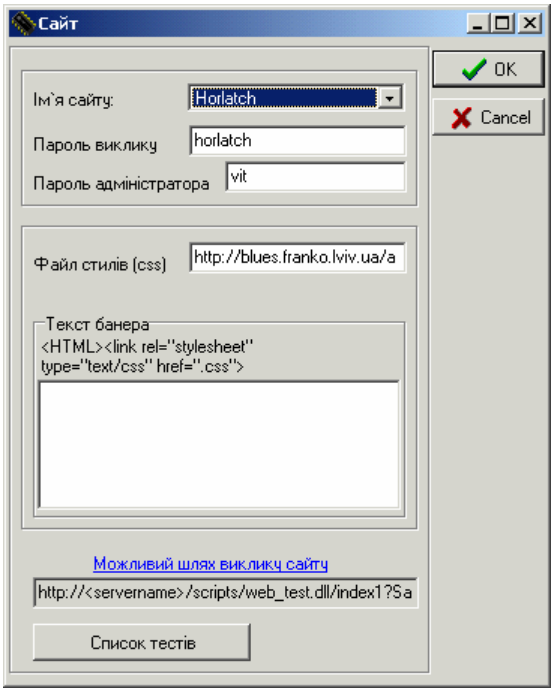

Рис. 2.27. Налаштування окремого сайту тестувань

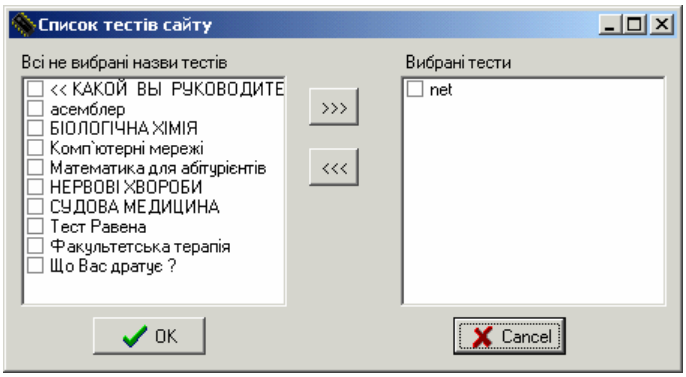

Рис. 2.28. Формування переліку тестів окремого сайту

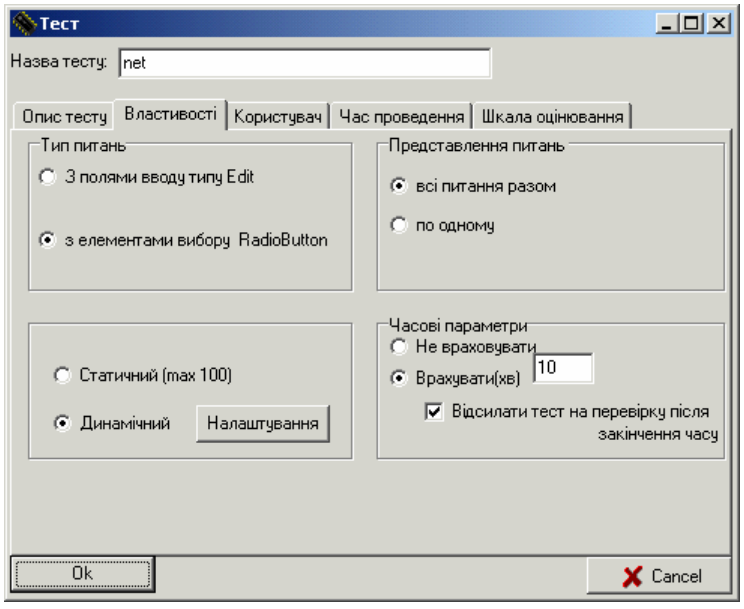

Рис. 2.29. Налаштування властивостей тесту

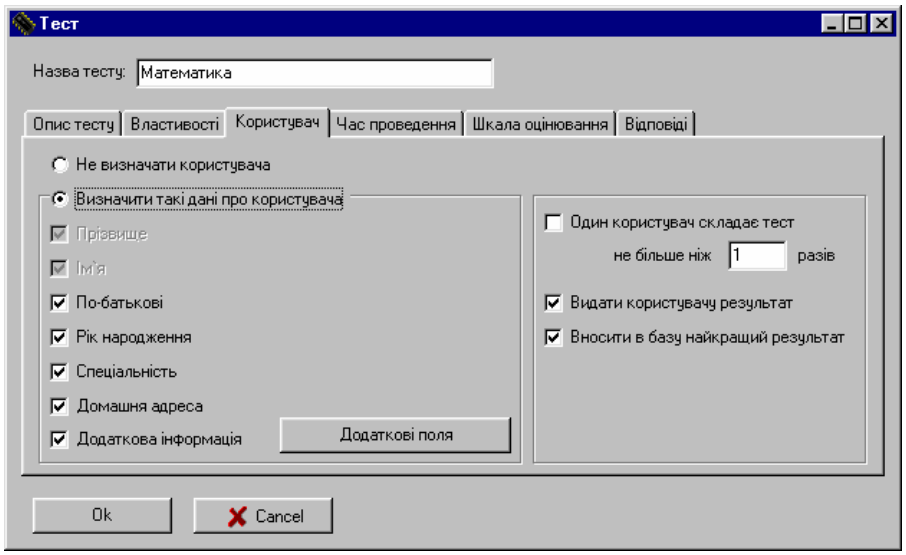

Рис. 2.30. Налаштування тесту стосовно визначення даних про користувача

2.6.5. Внесення інформації

Внесення інформації забезпечується в консолі адміністратора-викладача, як це ілюструє рисунок:

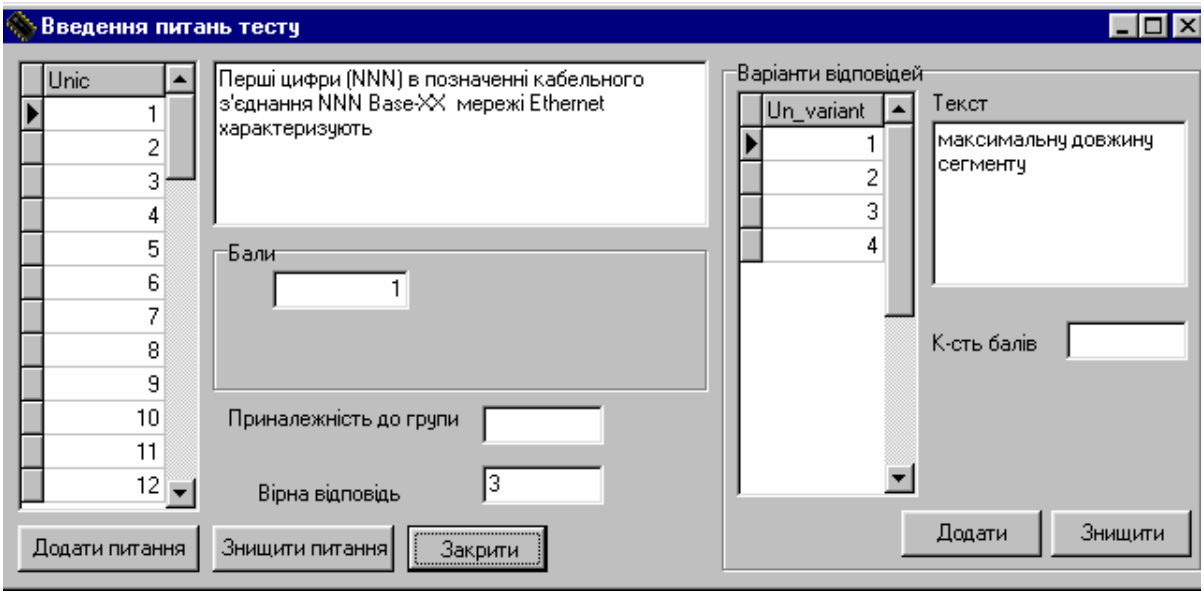

Рис. 2.31. Форма внесення тесту в базу даних системи

Ще один, і напевно простіший та наглядніший, спосіб внесення інформації – це створення будь-яким текстовим редактором текстового файлу з варіантами питань, відповідей та деякою додатковою інформацією. Підготований таким чином файл сканується програмою і додається в структуру бази даних. Після цього адміністратору залишається використовуючи консоль задати налаштування даного тесту.

Файл, який сканує програма для подальшого впорядкування питань повинен задовольняти таким вимогам:

1. Весь текст файлу зберігається в текстовому форматі і повинен мати розширення txt.

2. Текст питання складається з трьох блоків:

<текстова частина>
[<частина з вказанням параметрів шкали>]
[<частина з вказанням параметрів тесту>]

(друга і третя частини є необов'язковими)

3. Формат текстової частини :

<текстова частина>=<текст питання>
[<варіанти відповідей>]

<текст питання>
[<варіанти відповідей>]

4. Формат тексту питання :

<текст питання>=(число.[(параметри питання)]текст)
[<стрічка тексту>]

[<стрічка тексту>]

Кожне питання записується з нової стрічки.

5. Формат опису варіантів відповідей:

<варіанти відповідей>=(варіант відповіді 1)

(варіант відповіді N)

- Кожен варіант відповіді записується з нової стрічки.
- Варіанти відповіді можуть бути присутні лише у випадку коли у властивостях тесту задано тип питань з перемикачами типу CheckBox або елементами вибору RadioButton.

Кожен варіант відповіді може :

а) починатися зі знаку '-' або '+'

Знак '+' вказує на те, що цей варіант відповіді вважається правильним. Можлива також ситуація, коли в одному питанні кілька варіантів визначені як правильні, в цьому випадку правильною відповіддю вважається ситуація, коли користувач вказує саме ці варіанти.

Після знаку '-' або '+' слідує сам текст питання, який може бути записаний в кілька стрічок.

b) починатися з '(кількість балів)' , цей варіант можливий лише у випадку коли у властивостях тесту задано опцію: "Вказати кількість набраних балів для кожного з варіантів відповідей". В цьому випадку "кількість балів" означає кількість балів, які отримає користувач при виборі цього варіанту відповіді.

6. У випадку коли у властивостях тесту задано тип питань з полями вводу типу "Edit", після тексту кожного питання можна вказати правильну відповідь, це здійснюється за допомогою дужок ( ) в яких записується відповідь, і вони повині записуватися з нового рядка.

7. <частина з вказанням параметрів шкали> :

Після внесення всіх питань, в другій частині текстового файлу можна задати параметри шкали оцінювання за допомогою такого запису :

<SCALE>
------ текст блоку-------
</ SCALE >

Опис тексту блоку подано нижче.

7) <частина з вказанням параметрів тесту> :

В кінці текстового файлу можна задати додаткові параметри питань за допомогою такого запису :

<PARAM>
------ Параметри-------
</PARAM>

Правильні варіанти відповідей на відповідні питання, записані по порядку в стрічці параметрів записуються, наприклад, так:
vid=2;4;3;3;3;2;3;3;4;2;3;4;3;4.

Стрічка обовязково повинна закінчуватися крапкою.

Бали кожному питанню ставляться у відповідність стрічкою подібною до bal=1-10:1;11:4,12:4;13-20:2;21-30:3.

Тут питанням з першого по десяте ставиться у відповідність – один бал, 11-ому - чотири бали і так далі.

Система має можливість при скануванні перевіряти текст на присутність дескрипторів, що задають малюнки, наприклад : <img src="1.jpg">. Шлях до малюнків задається при вказанні параметрів системи в полі "Розміщення img".

2.6.6. Інтерфейс користувача

Рисунки 2.32 – 2.40 ілюструють взаємодію користувача з системою під час тестування.

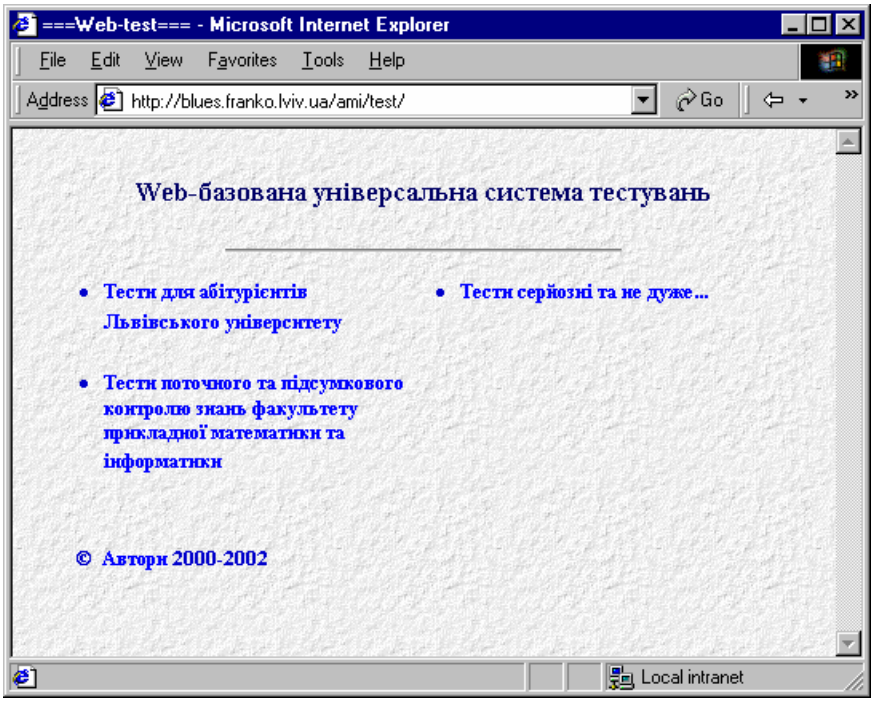

Рис. 2.32. Сторінка з переліком кількох функціонуючих в системі сайтів

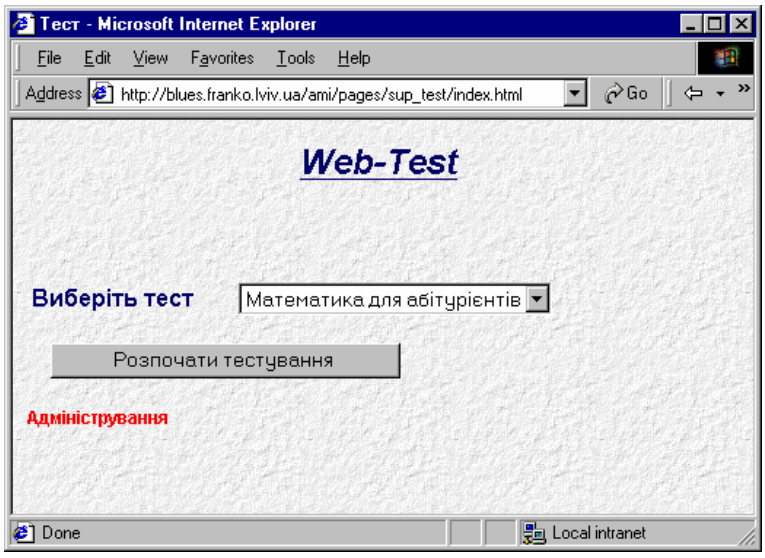

Рис.2.33. Вибір тесту

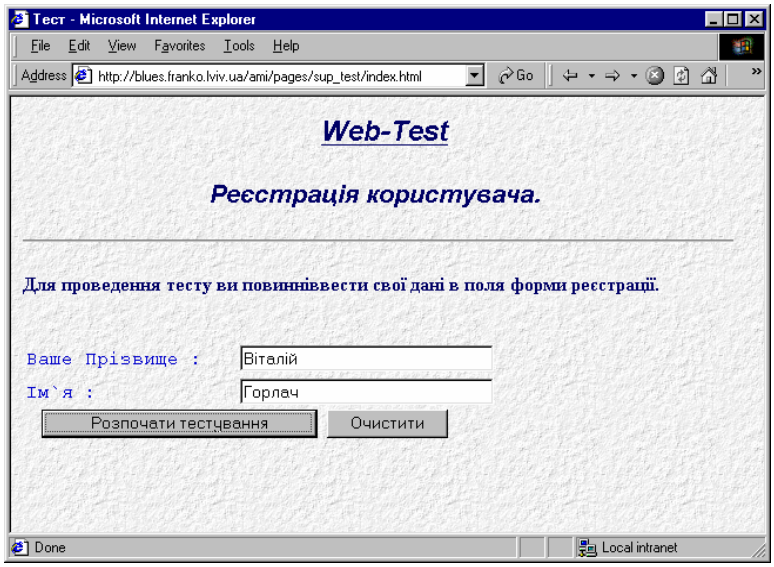

Рис.2.34. Реєстрація користувача

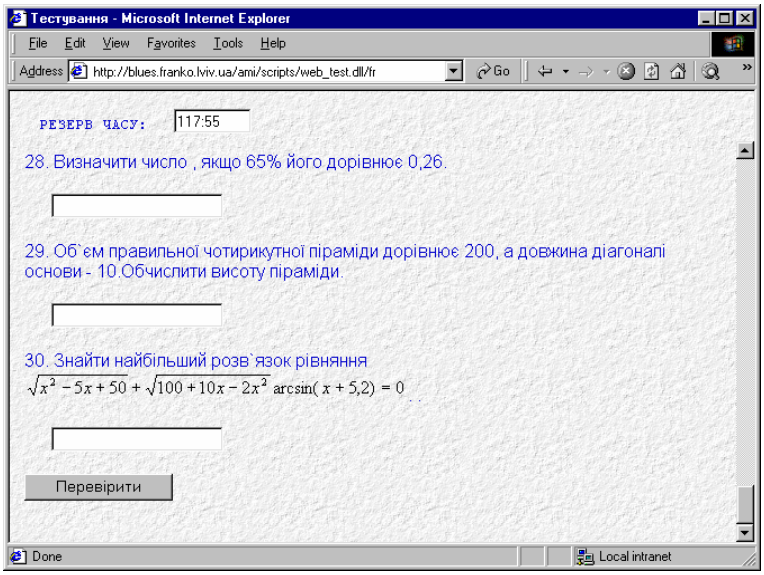

Рис.2.35. Приклад сторінки, що містить тестові питання з математики

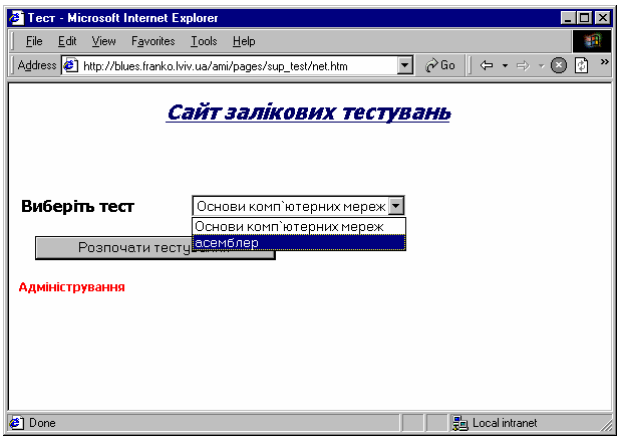

Рис.2.36. Приклад сайту, що оформлений з використанням іншого файлу стилів

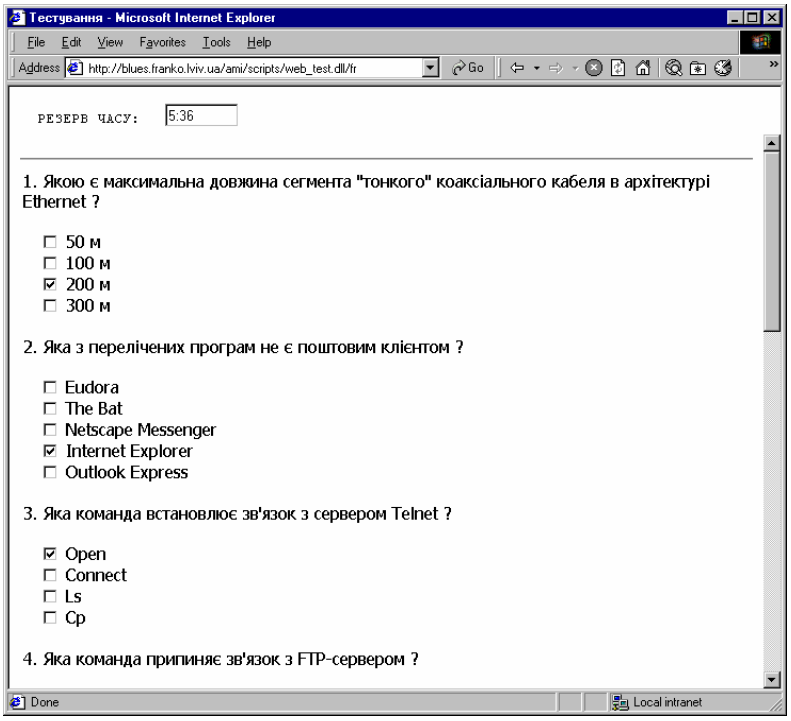

Рис.2.37. Сторінка з тестовими питаннями з курсу "Основи комп'ютерних мереж"

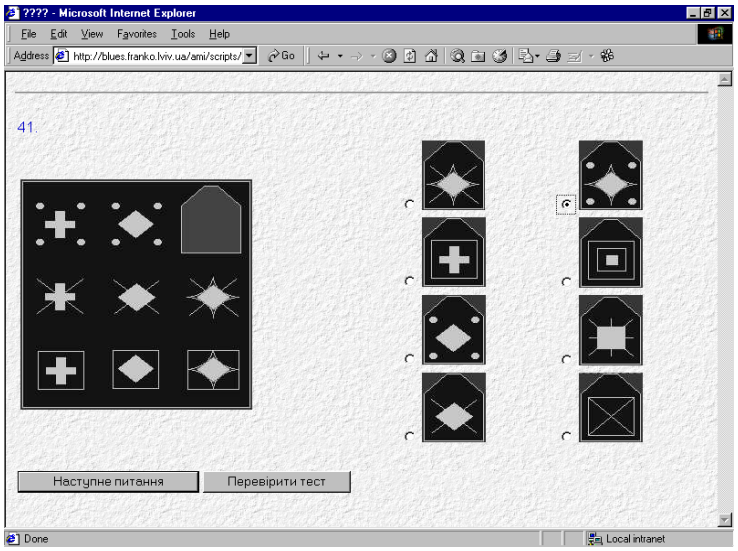

Рис.2.38. Приклад тесту з використанням графічних зображень (тест Равена)

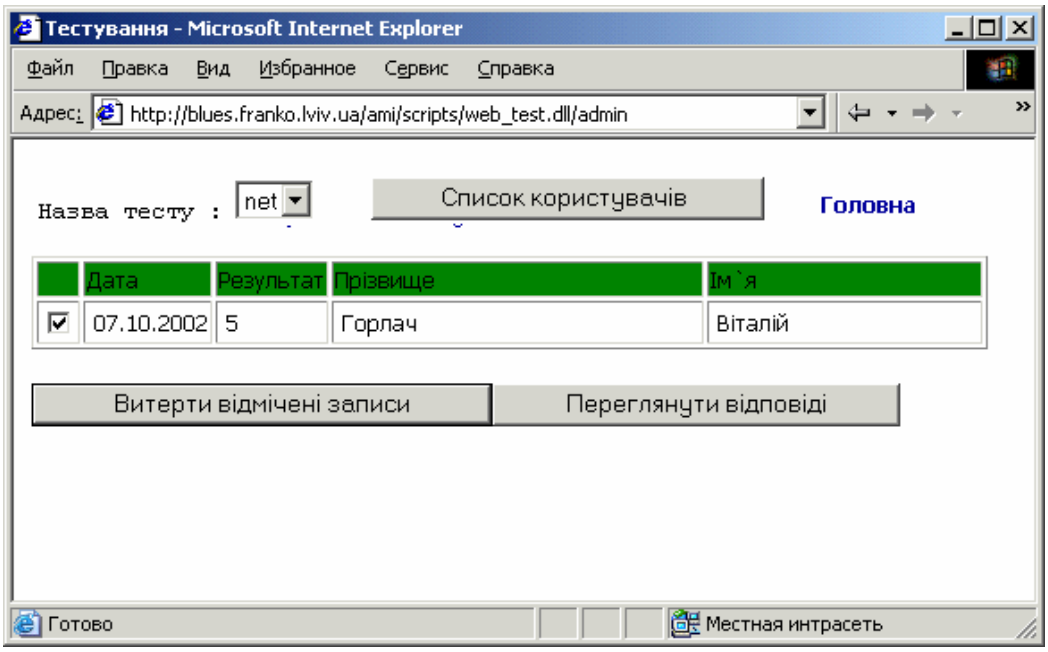

Рис.2.39. Сторінка викладача з переліком результатів тестування

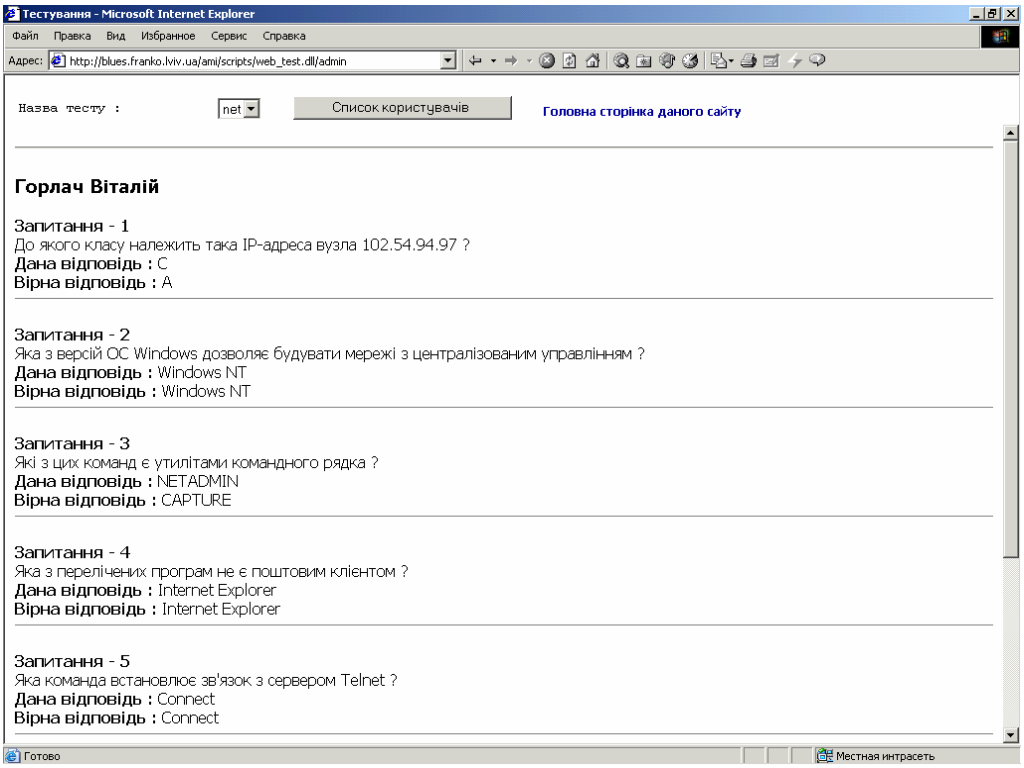

Рис.2.40. Протокол тестування

2.7. Web-застосування "Телефонний довідник"

Web-застосування "Телефонний довідник" запроектовано з орієнтацією на використання сервера баз даних MS SQL.

Застосування реалізовано двома незалежними частинами – адміністративною, що працює як CGI-застосування для внесення інформації в бази даних та клієнтською, що представляє собою ISAPI-застосування для відображення та пошуку інформації.

В даному звіті рис.2.41 – 2.48 ілюструють форми представлення інформації кінцевому користувачу.

Адміністративна частина системи знаходиться в стані тестування.

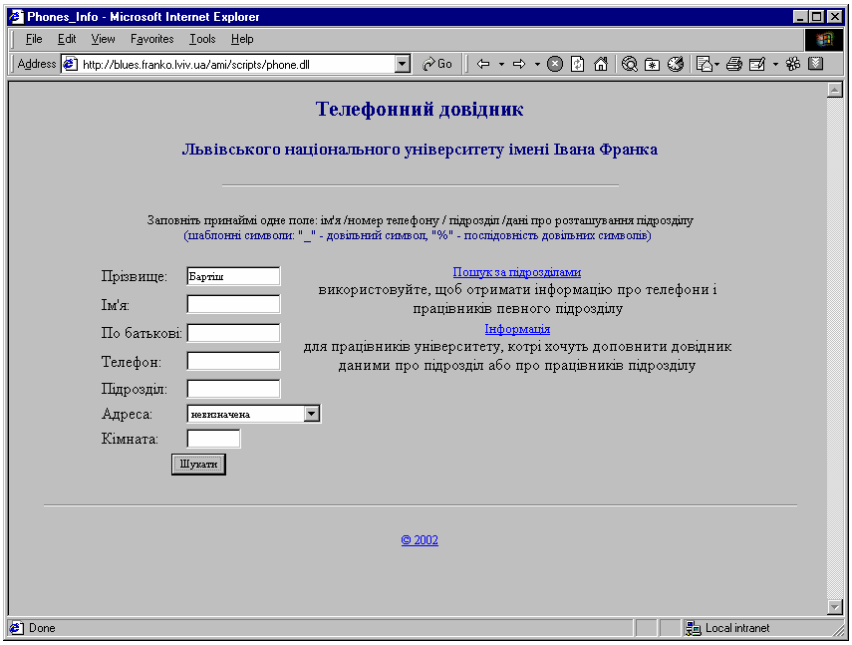

Рис.2.41. Титульна сторінка системи "Телефонний довідник"

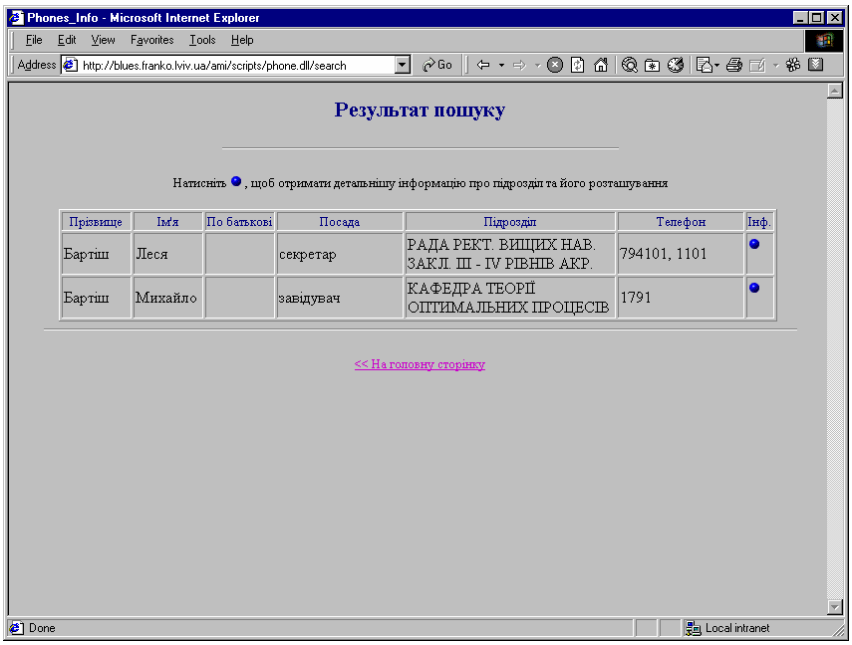

Рис.2.42. Результат пошуку за прізвищем

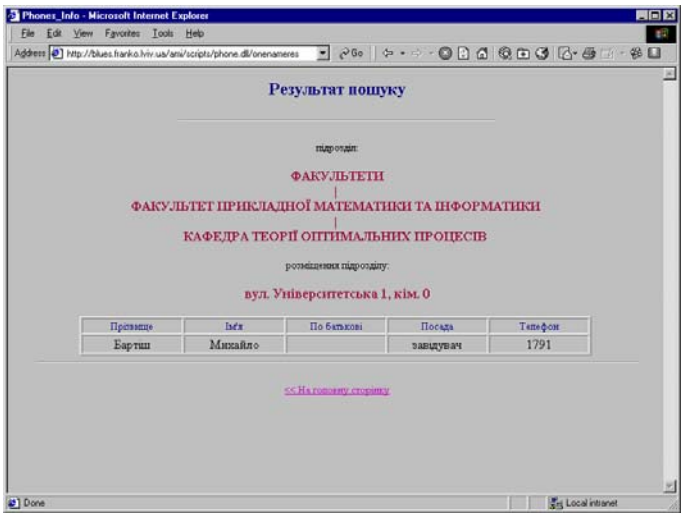

Рис.2.43. Інформація про знайдену особу

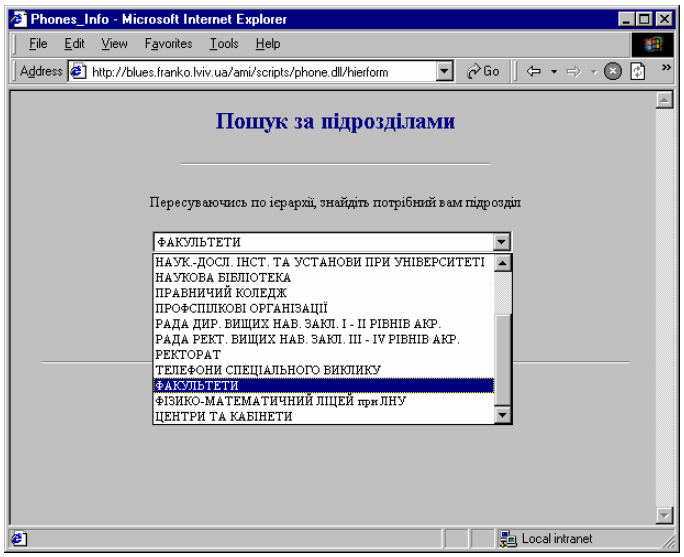

Рис.2.44. Пошук за ієрархією підрозділів (I - Факультети)

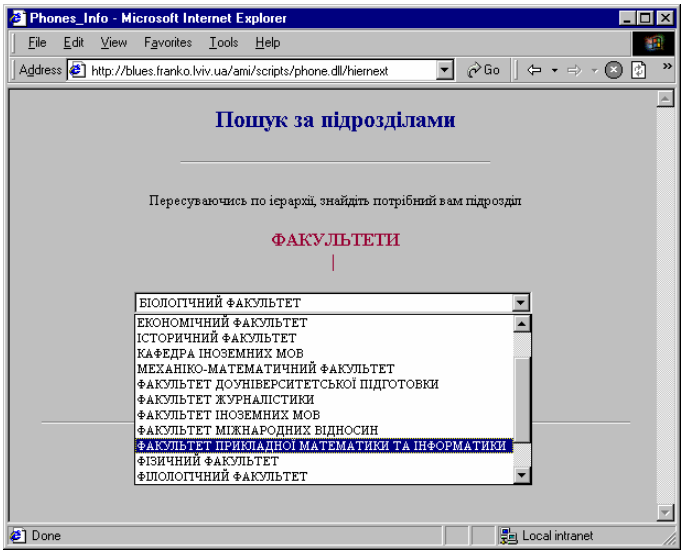

Рис.2.45. Пошук за ієрархією підрозділів (II – Факультет прикладної математики та інформатики)

Рис.2.46. Пошук за ієрархією підрозділів (III – телефони підрозділу "Факультет прикладної математики та інформатики")

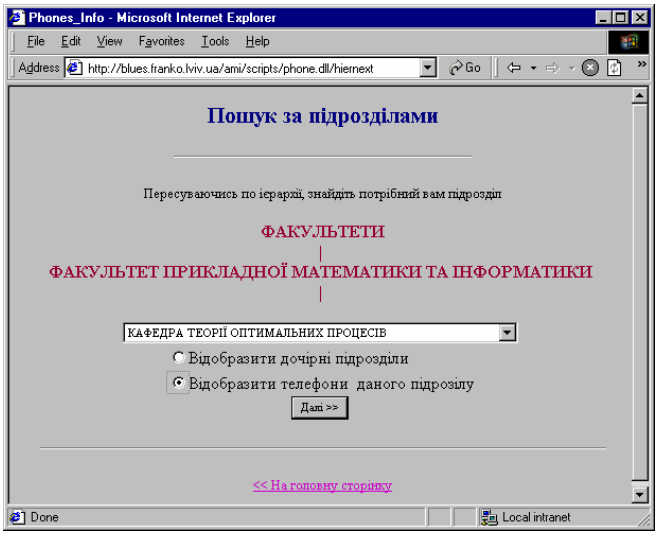

Рис.2.47. Пошук за ієрархією підрозділів (IV – відобразити телефони підрозділу)

Рис.2.48. Пошук за ієрархією підрозділів (IV – телефони підрозділу "Кафедра теорії оптимальних процесів")

ВИСНОВКИ

Розроблено Web-модель інформаційної системи корпоративного типу, яка включає архітектуру, основні класи, структуру баз даних, XML-формати передачі даних. Розроблено концепцію кросплатформенної технології проектування розподілених інформаційно-пошукових систем. Визначено вимоги до системного програмного забезпечення Web-сервера та клієнта.

Побудовано карту Internet-сайту університету. Створено базовий дизайн та таблиці каскадних стилів університетського Internet-сайту.

Створено програмне забезпечення системи "Інформаційний бюлетень", що являє собою електронну дошку оголошень університету з структуризацією інформації за тематикою та підрозділами.

Створено програмне забезпечення для наповнення баз даних, відображення інформації та автоматичного генерування твердої копії телефонного довідника університету.

Створено структуру каталогів, розроблено вимоги до форм електронного представлення публікацій, розроблено програмне забезпечення пошуку та відображення інформації сайту “Наукові видання Львівського університету”.

Створено програмне забезпечення Web-базованої універсальної тестової системи для використання в учбовому процесі для поточного та підсумкового контролю знань.

Отриману Web-модель та принципи кросплатформенного проектування можна використати для побудови масштабованих розподілених багаторівневих інформаційних систем. Зокрема, це можуть бути системи супроводження документообігу в навчальному закладі та його підрозділах, системи кадрового обліку і інші облікові та інформаційні системи, а також системи для дистанційного навчання, проведення дискусій, опитування працівників, тестувань в навчальному процесі тощо. Готові підсистеми "Телефонний довідник", "Наукові видання", "Облік наукових публікацій", "Універсальна система тестувань", "Система "Інформаційний бюлетень" впроваджені у Львівському національному університеті імені Івана Франка та підтверджують ефективність їх використання. Роботи у вказаному напрямку доцільно продовжувати, розширюючи сферу їх застосувань.

ПЕРЕЛІК ПОСИЛАНЬ ТА ПУБЛІКАЦІЙ ВИКОНАВЦІВ

1. Horlatch V., Shynkarenko O. Data structure, functionality and technologies of implementation of “Faculty” informational system based on Інтернет-Інтранет architecture // Вісник Національного технічного університету «ХПІ»: Тематичний збірник наукових праць «Системний аналіз, управління та інформаційні технології». – Харків: НТУ «ХПІ». – 2001. № 21. – С. 79-83.
2. Бернакевич І.Є., Вагін П.П. Програмування мовою Java: використання фундаментальних класів. – Львів: Видавн. центр ЛНУ, 2002. – 76 с.
3. Бернакевич І.Є., Вовк В.Д., Левченко О.М. Концепція інформаційної системи науково-освітньої установи // В зб. матер. III міжн. конф. Інтернет-Освіта-Наука-2002, 8-12 жовтня 2002р. Том.1. – Вінниця: УНІВЕРСУМ-Вінниця, 2002. – С. 18.
4. Білик М.В., Горлач В.М., Трескот І.І. Основні принципи побудови інформаційної системи “Факультет” // Вісник Львів. ун-ту. Серія мех.-мат. – 1998., Вип. 50. – С. 10-11.
5. Вовк В.Д. Застосування об'єктного підходу до розробки інформаційних систем на основі реляційних баз даних // Вісн. Львів. ун-ту. Сер. прикл. матем. та інформатика. 2000. – Вип.2. – С. 184 – 190.
6. Вовк В.Д., Мушкевич Б.Я. Застосування об'єктного підходу до розробки інформаційних систем під керуванням реляційних СКБД // Сучасні проблеми прикладної математики та інформатики: Тез. доп. IX Всеукр. наук. конф. (24-26 вересня 2002р., м.Львів). – Львів: Видавничий центр ЛНУ, 2002. – С. 19-20.
7. Вовк В.Д., Мушкевич Б.Я. Застосування об'єктного підходу до розробки моделі процесів інформаційних систем з використанням реляційних баз даних // Вісник Львів. ун-ту. Сер. прикл.матем. та інформатика. – 2002. Вип. 4. – С. 184-188.
8. Горлач В.М. Реалізація універсальної тестової системи у вигляді Веб-сервера Інтранет навчального закладу // В зб. матер. III міжн. конф. Інтернет-Освіта-Наука-2002, 8-12 жовтня 2002р. Том.1. – Вінниця: УНІВЕРСУМ-Вінниця, 2002. – С. 203-204.
9. Горлач В.М., Макар В.М. Побудова та адміністрування INTRANET-мереж. Ч.1. Основи мережних технологій: Тексти лекцій.-Львів: Видавничий центр Львів. ун-ту, 1999. – 45с.
10. Горлач В.М., Макар В.М. Побудова та адміністрування INTRANET-мереж. Ч.2. Адміністрування мереж Windows NT: Тексти лекцій.-Львів: Видавничий центр Львів. ун-ту, 1999. – 41с.
11. Дмитроца Я.Б., Музичук А.О. Про проект електронної бібліотеки Української академічної і освітньої мережі // Тези доповідей сьомої Всеукраїнської наукової конференції: Сучасні проблеми прикладної математики та інформатики.– Львів, 2000. – С. 38-39.

12. Костів О.В., Черняхівський В.В. Інформатизація суспільного життя України: стан і перспективи // Вісник Львів. ун-ту. Сер. прикл. матем. та інформатика. – Львів, ЛНУ, 1999. Вип. 1. – С. 143-145.
13. Музичук А.О., Петльований А.Т., Скуратов А.О., Табачин А.М. Про деякі аспекти розробки університетської інформаційної системи // Тези доповідей дев'ятої Всеукраїнської наукової конференції: Сучасні проблеми прикладної математики та інформатики. – Львів, 2002. – С. 93.
14. Web-сайт "Наукові видання Львівського університету" - http://www.franko.lviv.ua/publish/
15. Web-система "Телефонний довідник університету" - http://blues.franko.lviv.ua/ami/scripts/phone.dll
16. Універсальна система тестувань - http://blues.franko.lviv.ua/ami/test/
17. База даних наукових публікацій співробітників факультету прикладної математики та інформатики - http://blues.franko.lviv.ua/ami/pages/bibl/publicat.htm
18. Система тестувань (інтранет Львівського університету) – http://kprog:80