

УДК 681.3
№ держреєстрації 0105U002238
Інв. № 0207U002849

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 (032) 297-89-03, ndch@franko.lviv.ua

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
д-р. хім. наук, проф.
_____ Б.Котур
____.12. 2006

**ЗВІТ
ПРО НАУКОВО-ДОСЛІДНУ РОБОТУ**

**СИСТЕМА АВТОМАТИЧНОГО ГЕНЕРУВАННЯ ДРУКОВАНИХ ТА
ЕЛЕКТРОННИХ ВИДАНЬ УЧБОВОГО ЗАКЛАДУ
ОБ-29П
(Заключний)**

Начальник НДЧ,
канд.хім.наук, ст.наук.співроб.

С.Орищин

В.о. декана факультету
прикладної математики та інформатики,
канд.фіз.-мат.наук., доц.

М.Коркуна

Науковий керівник,
д-р.фіз.-мат.наук, проф.

Г.Шинкаренко

2006

Рукопис закінчено 15 листопада 2006 р.
Результати цієї роботи розглянуто Вченою Радою
факультету прикладної математики та інформатики,
протокол № 12 від 15.11.2006

Список авторів

науковий керівник теми, зав. кафедри інформаційних систем, д-р. фіз.-мат. наук	_____	Г.Шинкаренко
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	І.Бернакевич
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	П.Вагін
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	П.Венгерський
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	В.Вовк
доц. кафедри інформаційних систем, канд. фіз.-мат. наук	_____	В.Горlach
ст. наук. співр. інформаційно- обчислювального центру	_____	О.Левченко
мол. наук. співр. кафедри інформаційних систем	_____	Р.Малець
ст. лаб. кафедри інформаційних систем	_____	Т.Семенюта
студент	_____	В.Пасічник
студент	_____	А.Забавський
студент	_____	Т.Гула
студент	_____	С.Сидорчук

Нормоконтролер

М.Благітка

РЕФЕРАТ

Звіт про НДР: 85 с., 53 рис., 18 табл., 21 джерело.

Розглянуто питання проектування розподілених інформаційно-пошукових систем, методів збору інформації та розробки програмних засобів для генерування різноманітних інформаційних матеріалів про учбовий заклад у вигляді Інтернет-сторінок, презентацій, друкованих та мультимедійних видань.

Розвинуто сучасні методи проектування розподілених інформаційно-пошукових систем, методи об'єктного проектування баз даних, методи збору та обробки даних. Розроблено кросплатформенні структури для опису та зберігання інформації про учбовий заклад (а саме – структуру, підрозділи, персонал, спеціальності, навчальні курси, навчально-методичні матеріали, збірники наукових праць та періодичні видання тощо) для використання у системах автоматичного генерування друкованих та електронних видань.

На підставі інформації в центральній базі даних забезпечується вирішення декількох завдань – інформаційне забезпечення для функціонування Веб-сайту учбового закладу, автоматичне та напівавтоматичне генерування довідкових та презентаційних друкованих та/або мультимедійних видань у формі електронних документів на компакт-дисках або у вигляді Інтернет-сторінок.

Використані середовища проектування відповідають сучасному рівню розвитку комп'ютерних технологій.

ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, ПРЕЗЕНТАЦІЯ, ІНТЕРНЕТ, ВЕБ-САЙТ, ВЕБ-ТЕХНОЛОГІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ФОРМАТ ДАНИХ, МУЛЬТИМЕДІЙНИЙ КАТАЛОГ.

ЗМІСТ

ВСТУП.....	6
1. ЗАГАЛЬНІ ЗАСАДИ ПОБУДОВИ КОМПОНЕНТ АВТОМАТИЧНОГО ГЕНЕРУВАННЯ ДОКУМЕНТІВ	7
2. ОРГАНІЗАЦІЯ ПЕРВИННОГО ЗБИРАННЯ ІНФОРМАЦІЇ ДЛЯ СТВОРЕННЯ БАЗИ ДАНИХ УНІВЕРСИТЕТУ	8
2.1. Принципи підходу до інформаційного забезпечення	8
2.2. Структура інформації для факультету	9
2.3. Висновки	15
3. ЗАСТОСУВАННЯ ОБ'ЄКТНОГО ПІДХОДУ ДО ТОПОЛОГІЇ РОЗМІЩЕННЯ ДАНИХ У МЕРЕЖІ	16
3.1. Об'єктні моделі даних та процесів	16
3.2. Динамічна взаємодія клієнтів та серверів застосувань	17
3.3. Об'єктна топологія даних та серверів застосувань	18
4. ДВОСТОРОННІЙ ТРАНЗИТ ДАНИХ МІЖ СКБД ТА ОФІСНИМИ ДОКУМЕНТАМИ З ВИКОРИСТАННЯМ СОМ-ТЕХНОЛОГІЙ	21
4.1. Використані технології та програмне забезпечення проекту	21
4.1.1. СКБД та ODBC	21
4.1.2. Тришарова модель клієнт-сервер	23
4.1.3. Інтеграція компонентів Microsoft Office	24
4.1.4. Технологія СОМ	25
4.2. Програмна реалізація	28
4.3. Висновки	33
5. ФОРМУВАННЯ ЗВІТІВ ДОВІЛЬНОЇ СКЛАДНОСТІ В СВІТЛІ ДОКУМЕНТОЦЕНТРИЧНОГО ПРОГРАМУВАННЯ	34
5.1. Технології реалізації проекту	34
5.1.1. Платформа .NET та .NET Framework	35
5.1.2. Типова Архітектура Word – проекту	35
5.1.3. Порівняння VBA та .NET Framework для написання Office-аплікацій	36
5.1.4. Об'єктна модель Word	36
5.2. Опис системи	37
5.2.1. Структура каталогів проекту	37
5.2.2. Користування програмою	39
5.3. Висновки	43
6. РОЗРОБКА МЕТОДИКИ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ПРЕЗЕНТАЦІЙНИХ МАТЕРІАЛІВ	45
6.1. Структура даних проекту	46
6.2. Опис прикладної програми для редагування презентації	48
6.3. Опис презентаційних матеріалів	53
6.4. Висновки	56
7. РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ З ВИКОРИСТАННЯМ ПАТТЕРНОГО ПРОЕКТУВАННЯ	58
7.1. Вимоги до системи	58

	5
7.2. Паттерни проектування. Паттерн проектування MVC	59
7.3. Класифікація систем управління контентом	61
7.4. Об'єктно-орієнтований підхід та PHP-реалізація MVC	62
7.4.1. Реалізація класу Model	63
7.4.2. Реалізація класу Controller	63
7.4.3. Реалізація класу View	64
7.4.4. Реалізація функціональності GlobalController	64
7.5. Реалізація системи управління контентом	65
7.5.1. Структура даних	65
7.5.2. Доповнення та оновлення даних. Шаблони перегляду даних	66
7.6. Створення веб-сайту періодичних видань на основі створеної системи управління контентом	69
7.7. Висновки	69
8. СИСТЕМА АВТОМАТИЗОВАНОГО ГЕНЕРУВАННЯ ПЕРСОНАЛЬНИХ ВЕБ-СТОРИНОК	71
8.1. Технології реалізації системи	71
8.2. Структура проекту	71
8.2.1. Шаблон проектування	72
8.2.2. Модель даних	72
8.2.3. Структура бази даних	73
8.2.4. Структура класів	78
8.3. Модерування інформації. Рівні доступу до даних	79
8.4. Контролери. Адміністративна частина	79
8.5. Представлення результатів	80
8.6. Інтерфейс системи	81
8.7. Висновки	82
ВИСНОВКИ	83
ПЕРЕЛІК ПОСИЛАНЬ	84

ВСТУП

Комп'ютеризація суспільства та перехід на бездокументарні управлінські технології призводить до стрімкого росту кількості різноманітних електронних форм документів та лавиноподібного накопичення інформації в установах та організаціях, в тому числі й учбових закладах. Відсутність єдиних стандартів, велика кількість форматів зберігання однотипної інформації (текстової, табличної, графічної, у базах даних та ін.) призводять до проблем:

- ✓ дублювання інформації;
- ✓ швидкої втрати актуальності інформації;
- ✓ великих затрат на інтегрування різноманітної інформації у певний формат з метою підготовки друкованих видань, видання мультимедійних каталогів та презентацій, публікації матеріалів в мережі Інтернет тощо.

Представлена робота, разом з розробкою та розвитком сучасних методів проектування розподілених інформаційно-пошукових систем, методів об'єктного проектування баз даних, методів збору та обробки даних, пропонує кросплатформенні структури для опису та зберігання інформації, а також програмне забезпечення для автоматизованого та напівавтоматизованого генерування Веб-сторінок, різноманітних звітних форм, презентацій, довідників, мультимедійних каталогів тощо.

Окремі елементи запропонованої авторами концепції успішно апробовані при проектуванні та розробці низки окремих проектів [15-19,21].

1. ЗАГАЛЬНІ ЗАСАДИ ПОБУДОВИ КОМПОНЕНТ АВТОМАТИЧНОГО ГЕНЕРУВАННЯ ДОКУМЕНТІВ

Діяльність університету супроводжується значними інформаційними потоками різної природи. Нерідко, одна й та ж інформація збирається різними службами при підготовці друкованих довідників, Веб-сторінок чи компакт-дисків. Звідси виникає необхідність централізувати зберігання основної інформації про учбовий заклад як організаційно (шляхом створення спеціального підрозділу) так і технічно – у базах даних сервера інформаційно-пошукової системи. Однак такий підхід не вирішує проблеми своєчасної актуалізації даних, що зберігаються. Очевидно, що адміністрування цих процесів повинно бути децентралізованим, тобто самі користувачі (представники підрозділів та служб) братимуть на себе певну частину функцій щодо оновлення інформації. Доступ до цієї інформації має бути авторизованим, однак досить простим і доступним для широкого загалу. Використання різних комп'ютерних платформ, а також комп'ютерів з різними технічними характеристиками вимагає проектування клієнтської частини під Веб-технології.

Такий підхід є основою для проектування сервера бази даних та Веб-застосувань, які, в залежності від потреб, можуть динамічно (одночасно з оновленням баз даних) представляти вказану інформацію користувачам.

Однак автоматичною генерацією Веб-сайту учбового закладу використання інформації сервера баз даних не обмежується. Централізоване зберігання інформації дозволяє проектувати окремі компоненти, які генерують структури даних для подальшого використання при підготовці друкованих чи мультимедійних довідників різноманітного призначення.

2. ОРГАНІЗАЦІЯ ПЕРВИННОГО ЗБИРАННЯ ІНФОРМАЦІЇ ДЛЯ СТВОРЕННЯ БАЗИ ДАНИХ УНІВЕРСИТЕТУ

Питання побудови та забезпечення успішного функціонування інформаційної бази даних такої великої установи, як Університет, дуже складне і багатогранне [2,11,12,14]. Передусім у ньому можна виділити два аспекти: програмне забезпечення та інформаційне наповнення (надалі вживатимемо термін «інформаційне забезпечення»). Розглянемо це детальніше

2.1. Принципи підходу до інформаційного забезпечення

Під інформаційним забезпеченням розумітимемо процес **збирання інформації**, необхідної для функціонування інформаційної бази даних Університету, та її підготовки до використання програмним забезпеченням. Надалі зібрані відомості потрібно своєчасно **оновлювати**, однак ми проаналізуємо лише організацію першого їх одержання. Треба також мати на увазі, що виконання завдання забезпечують відповідальні працівники в кожній окремій структурній одиниці Університету (факультеті, коледжі, науковій установі тощо), а у центрі, що координує цю роботу, зводять одержані дані в єдиний масив і готують їх до подальшого використання.

Попередні спроби збирання інформації – у форматі текстового документа – засвідчили складність цієї справи. Попри певні вимоги до структури такого файлу, кожен виконавець бачив завдання по-своєму, і в підсумку різні документи містили неоднакові відомості й не завжди повністю подавали потрібні дані. Отож, недоліки такого способу виконання завдання – це неструктурованість, неодноманітність, неповнота зібраної інформації. Тому для якісного розв’язання проблеми потрібно відшукати засоби, позбавлені цих вад.

Зазначимо, що безпосереднє збирання інформації у бази даних (БД) дало б змогу найшвидше і найякісніше досягнути мети, причому зібрані дані були б відразу готові до використання відповідним програмним забезпеченням. Однак на практиці це нездійсненно, оскільки звичайний виконавець не володіє інструментарієм роботи з базами даних. Тому найліпшим виходом із ситуації є використання електронних таблиць, які дають змогу легко вводити інформацію, контролювати її правильність, коригувати помилки, а потім (на рівні координаційного центру) конвертувати дані у БД потрібного формату. Інструментом же для роботи з електронними таблицями вибрано широкоживаний табличний процесор Excel [10].

Найпершим питанням, що постає на початку організації процесу збирання інформації, є визначення оптимального набору документів, які потрібно заповнити:

- один загальний файл на кожну окрему структурну одиницю;
- набір взаємозалежних файлів для підрозділів;
- набір незалежних файлів для кожного підрозділу структурної одиниці.

Перший варіант дає змогу відразу зібрати всю інформацію про структурну одиницю в одному файлі, уникнути дублювання даних та пов'язаних з цим помилок, і є взагалі найліпшим з погляду координаційного центру. Однак він потребує складної організаційної роботи з боку відповідальної особи від структурної одиниці для забезпечення скоординованих дій працівників підрозділів. Набір взаємозалежних файлів дає змогу розподілити роботу між підрозділами структурної одиниці, уникнувши дублювання інформації, однак висуває деякі технічні вимоги (зокрема, спільне розташування всіх файлів для роботи з ними через мережу), які сьогодні важко забезпечити для всіх структур Університету.

Отже, найоптимальнішим варіантом є розроблення набору повністю незалежних файлів, що дає змогу розподілити всю роботу зі збирання інформації по підрозділах структурної одиниці з подальшим переданням у координаційний центр для опрацювання, інтегрування і конвертації у БД. Роль відповідальних осіб у цьому випадку зводиться до передання відповідних файлів у підрозділи, призначення (через керівника структурної одиниці чи її підрозділу) конкретних виконавців, перевірки зібраних даних і передання повного набору файлів у координаційний центр.

Конкретизуємо викладені принципи на прикладі факультету як найбільшої та найскладнішої структурної одиниці Університету.

2.2. Структура інформації для факультету

Передусім зазначимо, що вид і обсяг інформації визначає Стандарт Університету «Веб-сайт Університету: структура, дизайн та мінімальний перелік інформації» [3]. Отож, усі подальші викладки ґрунтуватимуться на цьому базовому документі.

Факультет містить кілька типів підрозділів, які мають різну будову і виконують різні функції [1]. Це деканат – адміністративний підрозділ факультету, випускові кафедри, що готують фахівців різного освітнього рівня, і загальні кафедри, які ведуть загальні курси, а також допоміжні (ненавчальні) підрозділи – лабораторії, бібліотеки, музеї тощо. Зрозуміло, що ці підрозділи можуть подавати про себе інформацію різного роду та обсягу, і тому для кожного типу потрібно розробити окремий вид файлів. Це допоможе також чіткіше окреслити структуру факультету.

Ключовим підрозділом факультету є випускова кафедра, яка виконує основну навчальну та наукову роботу. Для збирання інформації про неї розроблено файл **Department.xls** [20], який містить аркуші (електронні таблиці) **Відомості, Історія, Працівники, Докторанти та аспіранти, Наукова робота, Навчальна робота, Курси кафедри**, куди у відповідні поля таблиці потрібно внести відомості, визначені Стандартом. Передбачено паралельне внесення інформації українською та англійською мовами.

- Аркуш Відомості – загальні відомості про кафедру, поля: Назва кафедри; Адреса (місто (крім Львова), вулиця, будинок); №№ кімнат; Назва (англ.); Адреса (англ.); Телефон (без розділювачів); E-mail; Web-сторінка; останнє – адреса власної сторінки кафедри (якщо вона функціонує).
- Аркуш **Історія** – короткі історична довідка та інформація про сучасну діяльність кафедри, поля: **Історія та діяльність; Історія та діяльність (англ.)**; особливість цієї таблиці полягає в тому, що кожен абзац вводять в окрему комірку як окремий запис.
- Аркуш **Працівники** – відомості про всіх працівників кафедри, поля: **Посада; Прізвище; Ім'я; По батькові; Прізвище (англ.); Ім'я (англ.); По батькові (англ.); Вчений ступінь; Вчене звання; Службовий телефон (без розділювачів); Адреса електронної пошти; Оприлюднювати е-адресу (так, ні)?; Адреса персональної Веб-сторінки; Фото (файл у папці Photos); Дата народження (у форматі дд.мм.рррр); Оприлюднювати дату народження (так, ні)?; Домашній телефон (без розділювачів); Коротка біографічна довідка; Оприлюднювати біографію (так, ні)?; Додаткові посади; Додаткові звання, членство в наукових товариствах, редколегіях наукових журналів тощо; Сфера наукових інтересів; Перелік курсів, що читають (через ;); Коротка біографічна довідка (англ.); Додаткові посади (англ.); Додаткові звання (англ.), членство в наукових товариствах, редколегіях наукових журналів тощо; Сфера наукових інтересів (англ.); Перелік курсів, що читають (англ., через ;); Вибрані наукові публікації (через ;); у відповідних полях працівник повинен зазначити, чи згоден він з оприлюдненням певних персональних відомостей на веб-сайті, причому домашній телефон не буде оприлюднено у жодному разі, а використано для інших цілей (зокрема для оновлення внутрішнього телефонного довідника).**
- Аркуш **Докторанти та аспіранти** – відомості про всіх аспірантів та докторантів кафедри, поля: **Статус (аспірант, докторант); Прізвище; Ім'я; По батькові; Прізвище (англ.); Ім'я (англ.); По батькові (англ.); Посада, вчений ступінь, вчене звання керівника/консультанта (тільки не працівника ЛНУ); Прізвище керівника/консультанта; Ім'я; По батькові; Посада, вчений ступінь, вчене звання керівника/консультанта (англ., тільки не працівника ЛНУ); Прізвище керівника/консультанта (англ., тільки не працівника ЛНУ); Ім'я (англ., тільки не працівника ЛНУ); По батькові (англ., тільки не працівника ЛНУ); Рік вступу в аспірантуру/докторантуру; Форма навчання; Шифр спеціальності; Спеціальність; Тема дисертації; Спеціальність (англ.); Тема дисертації (англ.); Службовий телефон (без розділювачів); Адреса електронної пошти; Оприлюднювати е-адресу (так, ні)?; Адреса персональної Веб-сторінки; Фото (файл у папці Photos); Домашній телефон (без розділювачів).**

- Аркуш **Наукова робота** – інформація про договірні та бюджетні теми, які виконують співробітники кафедри, поля: **Шифр**; **Номер держреєстрації**; **Назва**; **Назва (англ.)**; **Посада, вчений ступінь, вчене звання керівника (тільки не працівника ЛНУ)**; **Прізвище керівника**; **Ім'я**; **По батькові**; **Посада, вчений ступінь, вчене звання керівника (англ., тільки не працівника ЛНУ)**; **Прізвище керівника (англ., тільки не працівника ЛНУ)**; **Ім'я (англ., тільки не працівника ЛНУ)**; **По батькові (англ., тільки не працівника ЛНУ)**; **Початок виконання (мм.рррр)**; **Кінець виконання (мм.рррр)**; **Коротка анотація**; **Коротка анотація (англ.)**; **Веб-сторінка теми**.
- Аркуш **Навчальна робота** – відомості про спеціальності та спеціалізації, за якими кафедра випускає фахівців різного освітнього рівня, поля: **Шифр спеціальності**; **Спеціальність**; **Спеціалізація**; **Спеціальність (англ.)**; **Спеціалізація (англ.)**; **Освітній рівень**; **Кваліфікація**; **Кваліфікація (англ.)**; **Форма навчання**; **Термін навчання (в роках)**.
- Аркуш **Курси кафедри** – інформація про всі курси, які читають викладачі кафедри в межах Університету, поля: **Назва курсу (для кожного семестру окремо)**; **Анотація**; **Назва курсу (англ.)**; **Анотація (англ.)**; **№ семестру**; **Кількість лекційних годин**; **Кількість практичних (лабораторних) годин**; **Форма звітності**; **Прізвище лектора**; **Ім'я**; **По батькові**; **Тип курсу (гуманітарний, фундаментальний, вибіркового, професійний)**; **Факультет**; **Спеціальність**; **Спеціалізація**; **Освітній рівень**; **Форма навчання**.

Як бачимо, назви полів доволі повно описують інформацію, яку потрібно внести у відповідну комірку таблиці, і спосіб її внесення, однак до файлу додана ще й інструкція, у якій є детальні пояснення щодо введення даних у кожне поле. Застереження щодо інформації для «**тільки не працівника ЛНУ**» спрямоване на усунення дублювання і пов'язаних з ним помилок, оскільки відомості про працівників Університету вносять у таблицю **Працівники**.

Загальна кафедра факультету подає подібну інформацію, крім відомостей про спеціальності та спеціалізації, оскільки сама не готує випускників. Тому у файлі **GeneralDep.xls**, розробленому для цього типу підрозділу, будуть ті самі аркуші, що й для випускової кафедри, за винятком **Навчальної роботи**. А файл **Subdivision.xls**, призначений для ненавчального підрозділу факультету, міститиме лише аркуші **Відомості**, **Історія**, **Працівники** та **Наукова робота** з полями, описаними раніше.

Інша ситуація з адміністративним підрозділом факультету – деканатом. Інформація, яку подаватиме він, стосується загалом усього факультету, і тому відповідний файл не може бути складений із тих самих таблиць, що й для інших підрозділів. Зокрема, повинні бути відомості про вчену раду факультету, спеціалізовані ради із захисту дисертацій, періодичні видання тощо, а також навчальні плани для кожної спеціальності з урахуванням освітнього рівня та форми навчання. Щоб не перенасичувати файл для деканату великою кількістю

аркушів, вирішено розділити його на два: **Faculty.xls** для загальної інформації і **FacultyPlan.xls** для відображення навчальної роботи. Конкретизуємо структуру їхніх даних, спочатку для **Faculty.xls**.

- Аркуш **Відомості** – загальні відомості про факультет, поля: **Назва факультету**; **Адреса (місто (крім Львова), вулиця, будинок)**; **№№ кімнат**; **Назва (англ.)**; **Адреса (англ.)**; **Телефон (без розділювачів)**; **E-mail**; **Веб-сторінка**; останнє – адреса власної (існуючої) сторінки факультету.
- Аркуш **Історія** – короткі історична довідка та інформація про сучасну діяльність факультету, поля: **Історія та діяльність**; **Історія та діяльність (англ.)**; кожен абзац уводять в окрему комірку як окремий запис.
- Аркуш **Вітальне слово** – вітальне слово декана факультету – коротка інформація, що має на меті зорієнтувати абітурієнта у специфіці та привабливості факультету, поля: **Вітальне слово декана**; **Вітальне слово декана (англ.)**; кожен абзац також подають в окремій комірці.
- Аркуш **Працівники** – відомості про всіх працівників деканату, поля: **Посада**; **Прізвище**; **Ім'я**; **По батькові**; **Прізвище (англ.)**; **Ім'я (англ.)**; **По батькові (англ.)**; **Вчений ступінь**; **Вчене звання**; **Службовий телефон (без розділювачів)**; **Адреса електронної пошти**; **Оприлюднювати e-адресу (так, ні)?**; **Адреса персональної Веб-сторінки**; **Фото (файл у папці Photos)**; **Дата народження (у форматі дд.мм.рррр)**; **Оприлюднювати дату народження (так, ні)?**; **Домашній телефон (без розділювачів)**; **Коротка біографічна довідка**; **Оприлюднювати біографію (так, ні)?**; **Додаткові посади**; **Додаткові звання, членство в наукових товариствах, редколегіях наукових журналів тощо**; **Сфера наукових інтересів**; **Перелік курсів, що читають (через ;)**; **Коротка біографічна довідка (англ.)**; **Додаткові посади (англ.)**; **Додаткові звання (англ.), членство в наукових товариствах, редколегіях наукових журналів тощо**; **Сфера наукових інтересів (англ.)**; **Перелік курсів, що читають (англ., через ;)**; **Вибрані наукові публікації (через ;)**.
- Аркуш **Рада факультету** – інформація про вчену раду факультету, поля: **Посада в раді**; **Посада (тільки для студентів та аспірантів)**; **Прізвище**; **Ім'я**; **По батькові**; **Прізвище (англ., тільки для студентів та аспірантів)**; **Ім'я (англ., тільки для студентів та аспірантів)**; **По батькові (англ., тільки для студентів та аспірантів)**; **Адреса електронної пошти (тільки для студентів та аспірантів)**; **Адреса персональної Веб-сторінки (тільки для студентів та аспірантів)**; **Фото (файл у папці Photos) (тільки для студентів та аспірантів)**.

- Аркуш **Спеціалізовані ради** – відомості про спеціалізовані вчені ради із захисту дисертацій, які діють на факультеті, поля: **Шифр ради**; **Надавані наукові ступені (через ;)**; **Перелік спеціальностей (через ;)**; **Надавані наукові ступені (англ., через ;)**; **Перелік спеціальностей (англ., через ;)**; **Посада, вчений ступінь, вчене звання голови ради (тільки не працівника ЛНУ)**; **Прізвище голови ради**; **Ім'я**; **По батькові**; **Посада, вчений ступінь, вчене звання голови ради (англ., тільки не працівника ЛНУ)**; **Прізвище голови ради (англ., тільки не працівника ЛНУ)**; **Ім'я (англ., тільки не працівника ЛНУ)**; **По батькові (англ., тільки не працівника ЛНУ)**; **Посада, вчений ступінь, вчене звання заступника голови (тільки не працівника ЛНУ)**; **Прізвище заступника голови**; **Ім'я**; **По батькові**; **Посада, вчений ступінь, вчене звання заступника голови (англ., тільки не працівника ЛНУ)**; **Прізвище заступника голови (англ., тільки не працівника ЛНУ)**; **Ім'я (англ., тільки не працівника ЛНУ)**; **По батькові (англ., тільки не працівника ЛНУ)**; **Посада, вчений ступінь, вчене звання секретаря ради (тільки не працівника ЛНУ)**; **Прізвище секретаря ради**; **Ім'я**; **По батькові**; **Посада, вчений ступінь, вчене звання секретаря ради (англ., тільки не працівника ЛНУ)**; **Прізвище секретаря ради (англ., тільки не працівника ЛНУ)**; **Ім'я (англ., тільки не працівника ЛНУ)**; **По батькові (англ., тільки не працівника ЛНУ)**; **Телефон (без розділювачів)**; **E-mail**; **Web-сторінка**.
- Аркуш **Організації** – інформація про громадські, наукові та інші організації, які діють на факультеті, поля: **Назва організації**; **Характеристика діяльності**; **Назва організації (англ.)**; **Характеристика діяльності (англ.)**; **Посада, вчений ступінь, вчене звання керівника (тільки не працівника ЛНУ)**; **Прізвище керівника**; **Ім'я**; **По батькові**; **Посада, вчений ступінь, вчене звання керівника (англ., тільки не працівника ЛНУ)**; **Прізвище керівника (англ., тільки не працівника ЛНУ)**; **Ім'я (англ., тільки не працівника ЛНУ)**; **По батькові (англ., тільки не працівника ЛНУ)**; **Адреса офісу (місто (крім Львова), вулиця, будинок)**; **№№ кімнат**; **Адреса офісу (англ.)**; **Телефон (без розділювачів)**; **E-mail**; **Web-сторінка**.

- **Аркуш Видання** – відомості про періодичні видання, які виходять на факультеті, поля: **Тип видання; Назва видання; Серія; Назва видання (англ.); Серія (англ.); Посада, вчений ступінь, вчене звання відповідального редактора (тільки не працівника ЛНУ); Прізвище відповідального редактора; Ім'я; По батькові; Посада, вчений ступінь, вчене звання відповідального редактора (англ., тільки не працівника ЛНУ); Прізвище відповідального редактора (англ., тільки не працівника ЛНУ); Ім'я (англ., тільки не працівника ЛНУ); По батькові (англ., тільки не працівника ЛНУ); Посада, вчений ступінь, вчене звання відповідального секретаря (тільки не працівника ЛНУ); Прізвище відповідального секретаря; Ім'я; По батькові; Посада, вчений ступінь, вчене звання відповідального секретаря (англ., тільки не працівника ЛНУ); Прізвище відповідального секретаря (англ., тільки не працівника ЛНУ); Ім'я (англ., тільки не працівника ЛНУ); По батькові (англ., тільки не працівника ЛНУ); Періодичність (разів на рік); Мова видання; Коротка анотація; Адреса редколегії (місто (крім Львова), вулиця, будинок); №№ кімнат; Коротка анотація (англ.); Адреса редколегії (англ.); Телефон (без розділювачів); E-mail; Web-сторінка.**

Файл **FacultyPlan.xls** містить такі електронні таблиці.

- **Аркуш Докторантура** – інформація про спеціальності докторантури на факультеті, поля: **Шифр спеціальності; Спеціальність; Спеціальність (англ.); Майбутній науковий ступінь; Термін перебування (в роках).**
- **Аркуш Аспірантура** – інформація про спеціальності аспірантури на факультеті, поля: **Шифр спеціальності; Спеціальність; Спеціальність (англ.); Майбутній науковий ступінь; Форма навчання; Термін навчання (в роках).**
- **Аркуш Навчальна робота** – інформація про спеціальності, за якими факультет готує фахівців різного освітнього рівня, поля: **Шифр спеціальності; Спеціальність; Спеціальність (англ.); Освітній рівень; Кваліфікація; Кваліфікація (англ.); Форма навчання; Термін навчання (в роках); Аркуш переліку курсів спеціальності.**
- **Аркуші Курси спеціальності (№)** – навчальний план спеціальності; відповідність конкретного номера аркуша № і номера певної спеціальності задано на аркуші **Навчальна робота** у службовому полі **Аркуш переліку курсів спеціальності**; поля: **Цикл дисциплін (гуманітарний, фундаментальний, вибіркового); Назва курсу; Назва курсу (англ.); Вивчення (№№ семестрів через ,); Іспит (№№ семестрів через ,); Залік (№№ семестрів через ,); Контрольна (№№ семестрів через ,); Курсова (№№ семестрів через ,); Кількість лекційних годин; Кількість практичних годин; Кількість лабораторних годин; Кількість самостійних годин.**

Додамо, що всі файли для збирання інформації з інших структурних одиниць Університету можна скомпонувати з описаних тут електронних таблиць.

2.3. Висновки

Отже, оптимальним варіантом для організації збирання інформації про Університет є розроблення набору незалежних файлів формату електронних таблиць Excel для всіх підрозділів кожної структурної одиниці. Наприклад, для найбільшої й найскладнішої структурної одиниці – факультету – такими файлами можуть бути: для випускової кафедри – **Department.xls**, для загальної кафедри факультету – **GeneralDep.xls**, для ненавчального підрозділу – **Subdivision.xls**, для адміністративного підрозділу (деканату) – **Faculty.xls** та **FacultyPlan.xls**. Розподіливши ці документи між відповідними підрозділами і призначивши там відповідальних виконавців, від яких потрібне лише володіння популярною офісною програмою Excel на рівні звичайного користувача, можна одержати потрібні відомості за мінімально можливий термін. Опрацювання, інтегрування і конвертацію цих відомостей у бази даних повинні забезпечити фахівці єдиного координаційного центру, наявність якого є необхідною організаційною передумовою для успішного проектування, реалізації та підтримки інформаційної бази даних Університету.

3. ЗАСТОСУВАННЯ ОБ'ЄКТНОГО ПІДХОДУ ДО ТОПОЛОГІЇ РОЗМІЩЕННЯ ДАНИХ У МЕРЕЖІ

Очевидна безальтернативність принципу відкритості розподілених інформаційних систем (ІС) є головним аргументом «глобалізації» сьогodнішніх ідеологій їхньої розробки. Провідні технології у цій галузі DCOM, CORBA, SOAP та інші надають підкреслено універсальний інструментарій для організації взаємодії довільних компонентів гетерогенної системи через набір інтерфейсів. Тобто, ніяких попередніх припущень щодо структури інформації та способів її опрацювання не роблять. Відповідно, і реалізація таких технологій є складною і недешевою. З іншого боку, відомо, що будь-яка універсальність методики максимально виправдана лише за наявності засобів її ефективного обмеження (з одночасним спрощенням реалізації) в разі появи додаткової інформації про специфіку вихідної задачі. На жаль, згадані вище супертехнології таких засобів практично не мають, і часто розробники залучають їх «супер» можливості без гарантій об'єктивної потреби у цьому.

Ми відійдемо від практики всезагальності підходу до розробки ІС і зробимо це за рахунок додаткових припущень щодо структури її даних. Припущення будуть полягати навіть не в конкретизації вигляду цієї структури, а лише в єдиному підході до принципів її побудови в усіх окремих піделементах системи, які містять частини загальної інформації. Виконання такого обмеження, зрозуміло, можливе лише за умови розробки нової ІС. Крім того, її специфікою є умова відсутності жорстких вимог замовника щодо топології розміщення даних у мережі комп'ютерів. Утім, ці обмеження є невеликою платою за можливість будувати легкорозширювані, прості в експлуатації і недорогі в розробці ІС.

3.1. Об'єктні моделі даних та процесів

Далі опиратимемось на ідеологію застосування об'єктного підходу до розробки моделей даних та процесів інформаційних систем, яка детально описана в [2,5,6]. Коротко нагадаємо її головні тези.

Структуру предметної області задачі подають у вигляді ієрархії класів (типів), кожному з яких відповідає окремий об'єкт (таблиця бази даних) моделі даних. Далі, без обмеження загальності, усі викладки виконуватимемо на спрощеному опорному прикладі моделі ІС вищого навчального закладу (ВНЗ).

Процеси обробки даних поділяють на дві головні категорії: природні (за формулюванням задачі) та технологічні (подання інформації користувачам, тощо). На рис. 3.1 праворуч зображена гілка ієрархії природних процесів, яка є проекцією моделі даних, а реальні об'єкти, що наповнюють інформаційну систему, мають власну ієрархію відображення користувачу.

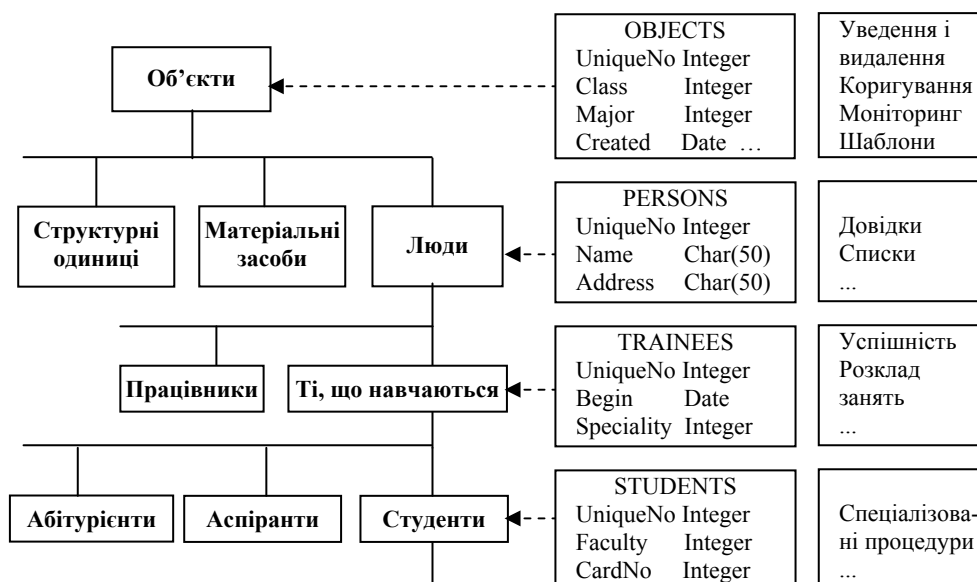


Рисунок 3.1 – Фрагмент моделі даних і процедур ІС ВНЗ

3.2. Динамічна взаємодія клієнтів та серверів застосувань

Типова реалізація багат шарової архітектури клієнт-сервер передбачає деревоподібну структуру, у якій кореневий шар становлять бази даних під керуванням відповідних систем керування базами даних (СКБД). Шар серверів застосувань виконує найбільшу обчислювано-затратну частину коду опрацювання інформації. Клієнти, здебільшого, лише відображають у зручному для користувачів вигляді отриману від серверів застосувань інформацію.

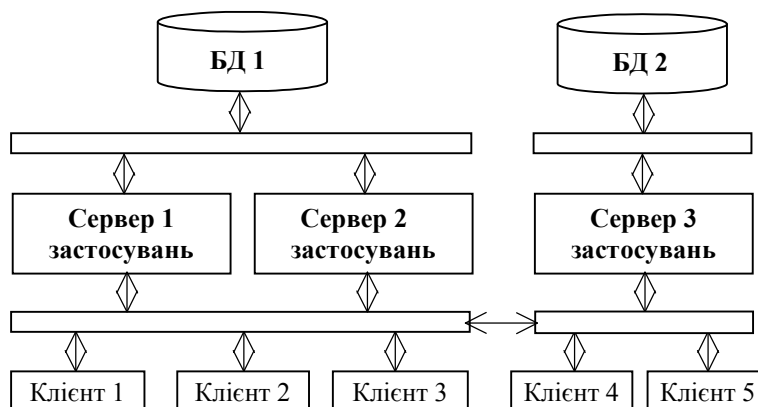


Рисунок 3.2 – Класична тришарова архітектура клієнт-сервер

Під номером сервера чи клієнта тут маємо на увазі його тип, а не конкретний екземпляр. Бази даних та відповідні сервери застосувань взаємодіють через локальну мережу. Клієнти входять у склад груп, що з'єднані між собою виділеними лініями або засобами Інтернету і, тому, здатні налагодити зв'язок з довільним сервером застосувань.

Програмний код опрацювання даних може міститись у самому сервері застосувань, або зберігатись у депозитаріях (як це, наприклад, передбачено технологією CORBA). Крім необхідності застосування недешевих технологій, за гнучкість другого підходу доводиться платити збільшенням завантаження

мережі. Тому нижче розглянемо випадок серверів та клієнтів із вбудованими процедурами опрацювання даних.

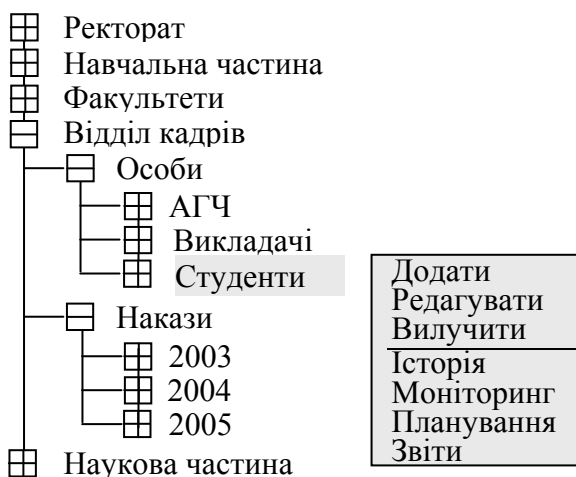


Рисунок 3.3 – Навігатор об'єктів

Зробимо головне припущення, яке визначатиме подальшу стратегію побудови ієрархії виконавчих серверів:

моделі даних і процесів усіх БД, що є в складі ІС, організовані за об'єктною методикою, що запропонована в [1,2].

Наголосимо, що це припущення ніяк не обмежує ні змісту інформації і процедур її опрацювання в БД, ні типів керуючих СКБД, що важливо для функціонування гетерогенних ІС. Водночас воно є основою для створення

єдиного для всіх клієнтських застосувань інтерфейсу навігації серед об'єктів системи, що означає значну уніфікацію процесів спілкування клієнта з серверами застосувань. Запропонована методика також задіює переваги об'єктного підходу у розподіленій розробці окремих елементів масштабної ІС.

Продемонструємо сказане на прикладі ІС ВНЗ [14]. Нехай показана на рис. 3.3 БД1 є базою даних ректорату, БД2 + сервер застосувань розміщені у відділі кадрів, а БД3 з частиною серверів та клієнтів міститься у віддаленому корпусі, скажімо, фізичного факультету, який має зв'язок з локальною мережею головного корпусу через Інтернет. Згідно з прийнятим припущенням, об'єкти кожної БД становлять ієрархічну структуру, яка в інтерфейсі користувача [5] є деревоподібною. Тому клієнт, під'єднуючись до будь-якого сервера застосувань (наперед налаштованого на визначену БД), відразу отримує дерево вузлів, як це, наприклад, зображено на рис. 3.4. Кожному вузлу дерева відповідає об'єкт певного класу. Для кожного класу наперед визначено набір технологічних процедур (візуальних форм), які реалізують окремі елементи моделі процесів [6] і які вибирають з меню. Важливо, що ця схема не змінюється у випадку з'єднання клієнта з сервером застосувань через мережу Інтернет. Це означає, що інформація про окремі вузли може бути у різних БД, а під'єднання до відповідних серверів застосувань виконуватиметься навігатором об'єктів динамічно. Зрозуміло, що відображення інформації окремих вузлів дерева буде відбуватись лише за наявності відповідних візуальних форм у програмному коді клієнта. Це дає змогу легко маніпулювати його функціональністю включенням у його склад лише потрібних елементів.

3.3. Об'єктна топологія даних та серверів застосувань

Взаємодія серверів застосувань у типовій архітектурі клієнт-сервер (рис. 3.2) не має системного характеру вже через те, що всіх їх уважають рівноправними і вони не утворюють ніякої ієрархії. Скористаємося

припущенням про об'єктність моделей даних і процесів усіх складових частин нашої ІС. Для цього ще раз звернемося до рис. 3.1. Кожному класу об'єктів тут відповідає власна таблиця даних. Розділимо всі таблиці між окремими БД структурних одиниць установи, які уводять і коригують відповідну інформацію. Зрозуміло, що способи й ефективність такого поділу сильно залежать як від умов вихідної задачі, так і від досвіду розробника.

Нехай, наприклад, БД відділу кадрів містить інформацію в електронному вигляді ту ж, яку, зазвичай, зберігають у паперовому. Зате тепер ніде вона не дублюється, як це властиво паперовим технологіям (зазначимо, що повна інформація про особу далеко не обмежена відділом кадрів, а рознесена по БД інших підрозділів). Кожний підрозділ, що має БД, комплектуваний також власним сервером застосувань (або кількома), котрий спеціалізований на технологічне опрацювання відповідної частини даних (див. рис. 3.4).

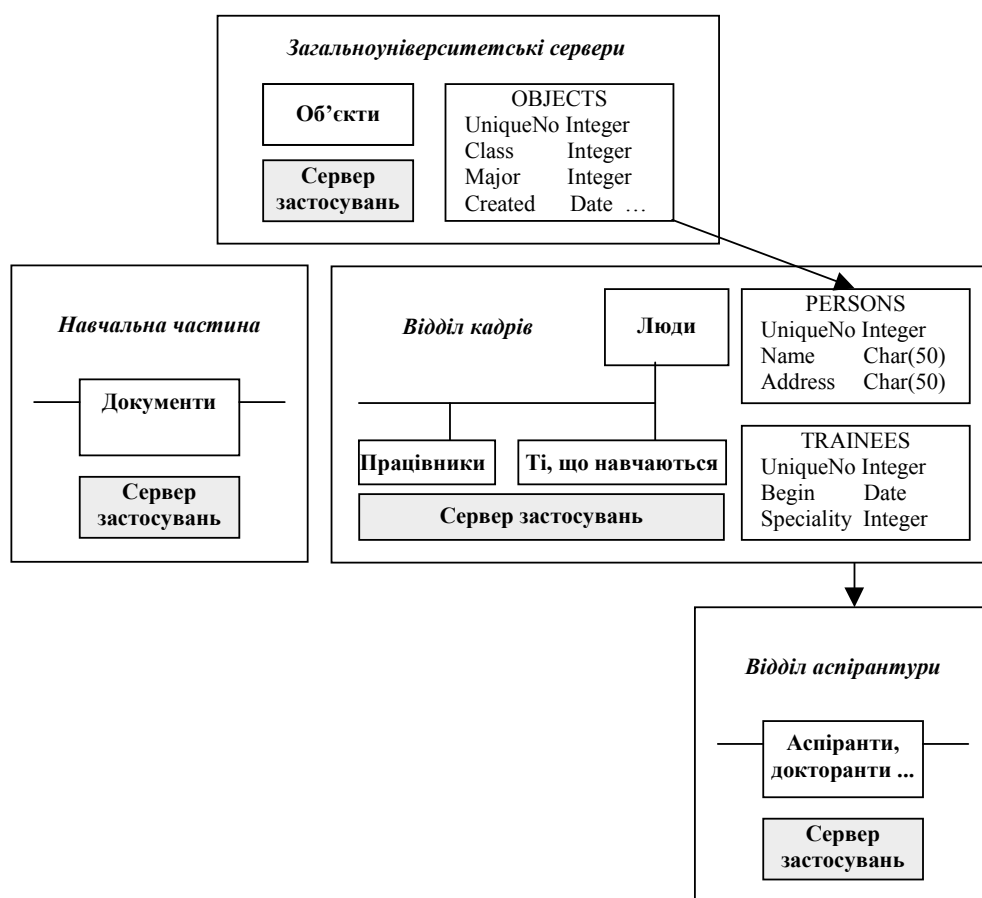


Рисунок 3.4 – Об'єктна топологія даних ІС ВНЗ

Зрозуміло, що інформація класів найвищої ієрархії повинна зберігатись на загальноуніверситетських серверах і бути максимально доступною для клієнтів. Крім того, деякі підрозділи (наприклад, відділ аспірантури) навіть не маючи власної БД, можуть на власному сервері застосувань мати низку технологічних процедур опрацювання інформації.

Наслідки запровадження цієї топології такі [4]:

- позитивні

- ІС у цілому є стійкішою до збоїв у роботі окремих підсистем, оскільки відсутність зв'язку з окремою БД (за винятком, можливо, OBJECTS) не порушує зв'язності решти інформації про об'єкт;
- дані локалізовано у підрозділах, що дає змогу посилити відповідальність за їх збереження та супровід;
- зменшуюються обсяги дублювання інформації;
- негативні
 - для збирання інформації про об'єкт необхідне почергове з'єднання з декількома серверами застосувань, що може призводити до затримок у часі виконання складних запитів.

Окремим позитивним моментом, який зумисно виокремлений з наведеного перерахунку, є спадкова ієрархія, яку утворює програмний код серверів застосувань (унаслідок об'єктності моделей даних та процесів). Завдяки особливості тришарової архітектури клієнт-сервер у перенесенні основної ваги програмного коду з клієнта на шар серверів застосувань цей факт обіцяє значні полегшення їхньої розробки та супроводу.

4. ДВОСТОРОННІЙ ТРАНЗИТ ДАНИХ МІЖ СКБД ТА ОФІСНИМИ ДОКУМЕНТАМИ З ВИКОРИСТАННЯМ СОМ-ТЕХНОЛОГІЙ

База даних – це сукупність взаємопов'язаних даних, що зберігаються разом, при чому для даних допускається така мінімальна надлишковість, що допускає їх оптимальне використання для кількох застосувань, забезпечується незалежність даних від програм, для пошуку, доповнення та модифікації даних застосовуються спільні засоби, забезпечується цілісність та захист від неавторизованого доступу.

Бази даних пройшли еволюцію від простих файлів, які характеризуються послідовною організацією фізичних даних, через файлові системи з типовим методом доступу та перші СКБД до сучасних систем керування базами даних.

Сучасні СКБД поділяються на сервери баз даних, тобто бази, до яких можна під'єднатися на віддаленому сервері (MySQL, MS SQL Server, PostgreSQL та ін.), та локальні, так звані настільні, системи керування базами даних. Кожна з СКБД має свої переваги та недоліки та свої застосування. Кожен розробник постачає базу даних із драйвером ODBC.

Важливою проблемою постає відображення інформації, яка знаходиться в базі даних, у зручній для користувача формі. Особливо, коли інформація потрібна для генерування довідкових матеріалів, а інформація зберігається в таблицях, у структуру яких, користувач не повинен вникати.

Зворотною проблемою є перенесення зібраної у якомусь довідковому файлі інформації в базу даних, або певні проміжні файли даних.

Спробуємо розглянути ці проблеми і запропонувати застосування для автоматичного генерування різноформатних матеріалів (електронних таблиць Excel, документів Word та XML файлів) з інформації у базі даних, та в зворотному напрямку, тобто з документа в таблицю бази даних. Для виконання роботи будемо використовувати різні СКБД та СОМ сервери Word та Excel.

4.1. Використані технології та програмне забезпечення проекту

4.1.1. СКБД та ODBC

Відкрите з'єднання з базою даних (Open Database Connectivity або ODBC) – розробка Microsoft, що надає розробникам додатків єдиний програмний інтерфейс додатку (API) для різних ядер баз даних, систем управління реляційними і не реляційними базами даних (database management system – DBMS). ODBC API призначений для надання розробникам аналогічних функціональних можливостей незалежно від типу даних, до яких виконується доступ – базам даних ISAM, базам даних SQL або текстовим даним. Ця мета досягається шляхом закріплення кожного драйвера ODBC за одним із зумовлених рівнів відповідності. Щоб вважатися драйвером ODBC, драйвер повинен відповідати специфікаціям ядра ODBC. Ці вимоги гарантують,

що розробник додатку завжди може розраховувати на одні і ті ж функціональні можливості незалежно від того, до яких даних відбувається звернення. Якщо формат використовуваних даних безпосередньо не підтримує основні функціональні можливості, драйвер ODBC повинен емулювати ці функції.

ODBC API абстрактно віддалений від реального джерела даних, як це показано на рис. 4.1.

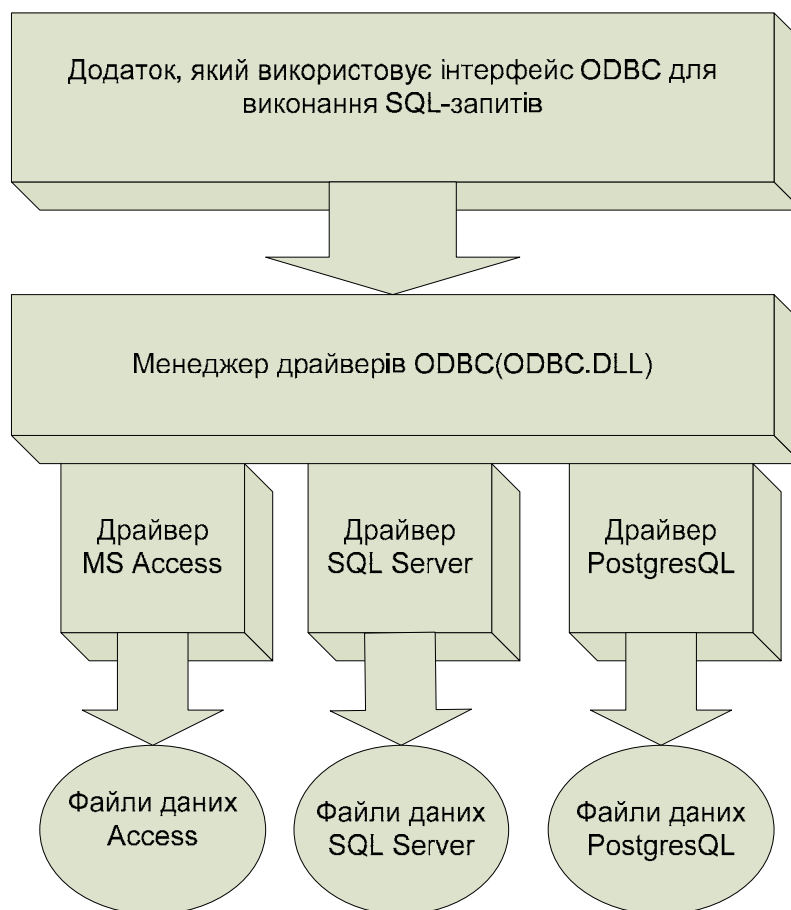


Рисунок 4.1 – Архітектура ODBC

Метод з'єднання з джерелом даних, отримання повідомлень про помилки, а також стандартні інтерфейси реєстрації є загальними для всіх драйверів. Для забезпечення уніфікованості драйвери дотримуються основних вимог API.

Відкритий інтерфейс доступу до баз даних є бібліотекою функцій, яка дозволяє прикладній програмі звертатися до різних СКБД, використовуючи структуровану мову запитів SQL. Інтерфейс ODBC пропонує незалежний від постачальника доступ до різних СКБД. Таким чином, розробник прикладної програми може створювати програму для віртуальної бази даних і дозволити завантажуваному драйверу перетворити логічні дані конкретної СКБД або систем, використовуваних даною прикладною програмою API.

Привабливість ODBC обумовлена її портативністю й взаємодією з кодом прикладної програми. ODBC функціонує як стандартний інтерфейс для розробників прикладних програм, а також для розробників бібліотек драйверів.

Архітектура ODBC має чотири основні компоненти: прикладна програма, диспетчер драйверів, драйвер і джерело або джерела даних. Додаток, що

використовує інтерфейс ODBC забезпечуватиме звичайні засоби користувача, включаючи:

- з'єднання і від'єднання від СКБД;
- виконання запитів і забезпечення областей зберігання і форматів даних для вибраних результатів;
- дозвіл транзакційної обробки в режимі on-line;
- засоби, зовнішні по відношенню до інтерфейсу ODBC.

Диспетчер драйверів, створений фірмою Microsoft, спільно з набором засобів розробника ODBC, є динамічно зв'язаною бібліотекою (DLL), яка завантажує драйвери, забезпечуючи єдину точку входу у функції ODBC для різних драйверів.

Функції інтерфейсу ODBC підрозділяються на сім груп. Ці сім груп містять функції, визначені проектом специфікацій фірми X/Open і інтерфейсу на рівні викликів (CLI) консорціуму SQL Access Group (SAG). Британська фірма X/Open є незалежною усесвітньою організацією, яка підтримується більшістю найбільших світових постачальників інформаційних систем. Американський консорціум SQL Access Group виконує аналогічну функцію. Ці дві групи працюють у кооперації по відношенню до проекту розробки специфікації SQL. Фірма SAG групує інтерфейсі функції таким чином:

1. призначення й відміна призначення:

- ідентифікатор оточення
- ідентифікатор з'єднання
- ідентифікатор оператора;

2. з'єднання;

3. виконання SQL-операторів;

4. отримання результатів;

5. управління транзакціями;

6. ідентифікація помилок

7. змішані функції.

4.1.2. Тришарова модель клієнт-сервер

Проектування систем спільного користування має дві складові: фізичну і логічну. У випадку фізичного проектування програмних систем розглядають комп'ютери, диски, мережі та інші технічні засоби з метою створення цілісної надійної технічної бази з оптимальним співвідношенням ціна/продуктивність.

Логічне проектування призначене для розробки архітектури (структури) програмного забезпечення, яка дає змогу найефективніше розв'язати вихідну задачу при максимальному використанні технічних ресурсів. Основними критеріями ефективності такої структури є вартість розробки, надійність роботи та простота супроводу створеного програмного продукту.

Довільну програмну систему, за моделлю MVC, можна умовно розділити на три логічні компоненти: відображення даних, прикладні функції та доступ до інформації. Програмне забезпечення першої компоненти реалізує інтерфейс між людиною та системою, тобто надає користувачу засоби введення,

корегування, пошуку та відображення інформації, що зберігається у системі. Це функції клієнтської частини загальної системи. Третя компонента складається з програм, які реалізують безпосередній доступ до інформації на фізичних носіях (баз даних, файлів та ін.). Цю роботу виконує сервер даних. Відкритим ми залишили питання про приналежність прикладної компоненти, додатковою специфікою якої є те, що виконання деяких із прикладних процедур має ініціюватись не користувачем, а ситуацією, що склалась у базі даних. Програмна система має автоматично у потрібний момент запустити потрібні процедури на виконання. Залежно від розміщення прикладної компоненти у застосуванні та способу її взаємодії з іншими компонентами розрізняють декілька підходів до розробки загальної структури застосування.

Двошарова модель застосування передбачає, що прикладна компонента реалізується або в клієнтському програмному забезпеченні, або ж інтегрується в роботу сервера баз даних. Проте є ще один варіант архітектури застосувань, схема якого зображена нижче.



Рисунок 4.2 – AS – модель сервера застосувань

Тут прикладна компонента системи реалізована як окремий шар, який у літературі називається сервером застосувань (Application Server). Клієнтські програми (Application Client) звертаються до нього за виконанням окремих прикладних задач над базами даних. А сервер застосувань з допомогою SQL отримує дані від сервера баз даних. Модель сервера застосувань є основою для спеціального виду програмного забезпечення – моніторів транзакцій (Transaction Processing Monitor – TPM), які реалізують перехід від функціонування баз даних у локальних мережах до баз даних у розподілених системах глобальних мереж.

4.1.3. Інтеграція компонентів Microsoft Office

Широкі можливості включення компонентів Microsoft Office в інші програми забезпечується в першу чергу тим, що ці компоненти мають в своєму розпорядженні множину загальнодоступних інтерфейсів, дозволяючи використовувати їх як у якості серверів автоматизації, так і в якості СОМ об'єктів.

Використання механізмів автоматизації відкриває більш широкі можливості при інтеграції компонентів Microsoft Office у додатки.

У структурі кожного об'єкта автоматизації існує багато методів і властивостей доступних програмісту в процесі роботи з цим об'єктом. Методика роботи з цими методами і властивостями аналогічна методиці роботи з методами й властивостями елементів управління в екранній формі.

Середовище розробки Delphi підтримує два способи роботи з об'єктами оптимізації – через бібліотеку типів і через механізм автоматизації OLE. Розглянемо механізм автоматизації OLE.

Механізм автоматизації OLE передбачають, що розробнику відомо імена інтерфейсів, методів і властивостей об'єкта автоматизації.

В середовище Delphi змінні типу OleVariant значно спрощують роботу з об'єктами автоматизації. Такі змінні можуть містити об'єкти автоматизації, а функції, обслуговуючі ці об'єкти, повертають значення типу OleVariant.

Нижче перераховані методи отримання доступу й маніпулювання об'єктами автоматизації:

```
CreateObject( );
GetActiveObject( );
OleFunction( );
OleProcedure( );
OlePropertyGet( );
OlePropertySet( );
```

Методи CreateObject() і GetActiveObject() використовуються, відповідно, формування і отримання посилання на активний об'єкт автоматизації. Призначення решти методів зрозумілі з їх назви. Відзначу, що методи OleFunction(), OleProcedure(), OlePropertyGet(), OlePropertySet() є по суті оболонками метода Exec().

Метод Exec() використовується Delphi для запуску на виконання процедури чи функції, а також для витягання і установки значень властивостей. При виконанні цих операцій метод звертається до об'єктів чотирьох спеціальних класів.

```
Function;
Procedure;
PropertyGet;
PropertySet;
```

Клас Function використовується для визначення функцій, клас PropertyGet – для витягнення значення властивостей і т.д.

4.1.4. Технологія COM

Технологія COM (Component Object Model – компонентна модель об'єктів) надає можливість одній програмі (клієнту) працювати з об'єктом іншої програми (серверу). COM – це модель об'єкта, яка передбачає повну сумісність у взаємодії між компонентами, написаними різними компаніями і на різних мовах. При цьому неважливо, де виконуються програми: в одному потоці, в різних потоках, на різних комп'ютерах.

З точки зору COM додаток містить декілька об'єктів (в окремому випадку може бути один об'єкт). Кожний об'єкт має один або декілька інтерфейсів. В

інтерфейсі описані методи об'єкта, до яких можуть дістати доступ зовнішні програми. Якщо інтерфейсів децю, кожний з них експонує деяку підмножину методів, що виконують однорідні функції.

Об'єкт є частиною сервера COM. Сервером може бути виконуваний файл або бібліотека DLL. При установці сервера в Windows в системний реєстр заноситься інформація про всі його об'єкти. Ця інформація включає ідентифікатор класу CLSID (Class Identifier), однозначно визначаючий клас об'єкта. Заноситься інформація про тип серверу: внутрішній (in-process – усередині процесу) – DLL, що підключається до клієнта, локальний (local) – працюючий окремим процесом на комп'ютері клієнта, віддалений (remote) – працюючий на віддаленому комп'ютері. Для внутрішніх і локальних серверів в реєстр заноситься повне ім'я файлу, а для видалених – повна мережна адреса. Таким чином, в системі зберігається вся інформація про сервер COM, необхідна для виклику його в потрібний момент.

Внутрішнім сервером є програма, яка експортує автоматні об'єкти. Оскільки автоматні об'єкти поставляються з DLL, а не з інших додатків, вони є частиною додатка клієнта. Це позбавляє від великих витрат, супутніх кожному виклику автоматного сервера.

Локальний або видалений сервер – це автономний виконуваний файл, що експортує автоматні об'єкти.

Зовнішні додатки, які звертаються до об'єкта COM, є клієнтами COM. Клієнт отримує вказівник на інтерфейс об'єкту, що цікавить його, і через цей вказівник може викликати методи об'єкта. Специфікація COM забороняє змінювати інтерфейс, який уже був оголошений. Це забезпечує нормальну роботу клієнта при будь-яких модифікаціях сервера.

Таким чином, клієнту достатньо знати інтерфейси об'єкта й методи, що надаються ними. Про інше подбає система. В потрібний момент вона запустить сервер COM, якщо він був ще не запущений, сервер створить об'єкт, об'єкт завантажить всі необхідні йому дані і клієнту повернується вказівник на об'єкт і його інтерфейси, із якими він може працювати. Система подбає також про те, щоб забезпечити роботу об'єкта відразу з декількома клієнтами.

Інформація про об'єкти COM, їх інтерфейси, властивості, методи, параметри методів міститься в бібліотеці типів, яка створюється розробником об'єкта COM і розповсюджується разом з об'єктом. Бібліотека створюється за допомогою мови опису інтерфейсу IDL (Interface Definition Language).

Тепер розглянемо докладніше інтерфейси. Кожний інтерфейс має ім'я, що починається із символу «I», і GUID – глобальний унікальний ідентифікатор (Globally Unique Identifier). Кожний об'єкт COM має інтерфейс IUnknown. Цей інтерфейс має всього три методи: QueryInterface – отримання покажчика на інтерфейс, AddRef і Release – збільшення й зменшення на 1 числа посилань на об'єкт. Метод QueryInterface повертає вказівник на інтерфейс із заданим IID. Метод Release повинен викликатися після закінчення роботи з інтерфейсом, щоб повідомити об'єкт, що даний клієнт в ньому більш не має потреби. Ці два методи використовуються завжди при роботі з об'єктом COM. Ще один метод –

AddRef використовується тільки в тих випадках, коли один клієнт передав іншому посилання на інтерфейс. Оскільки при цьому метод AddRef, що автоматично збільшує число посилань, не викликається, клієнт, якому передано посилання, повинен викликати AddRef, щоб доповісти об'єкту, що він теж із ним працює.

Усі решта інтерфейсів об'єктів є спадкоємцями IUnknown і успадковують ті ж три методи, додаючи, звичайно, свої власні. Отже сказане вище застосовне до будь-яких інтерфейсів. Серед цих інтерфейсів-спадкоємців необхідно відзначити IDispatch. Це інтерфейс, що використовується клієнтами серверів автоматизації OLE для доступу до методів і властивостей об'єкту.

Описаний спосіб роботи з об'єктом COM припускає існування екземпляра цього об'єкта й знання клієнтом вказівника на нього. Якщо ж йдеться про створення першого екземпляра об'єкта, то спочатку треба звернутися до бібліотеки COM, яка забезпечує диспетчеризацію роботи з об'єктами. Імена всіх функцій бібліотеки починаються з «Co». Створення об'єкта здійснюється викликом функції CoCreateInstance цієї бібліотеки і передачею в неї CLSID необхідного класу, IID інтерфейсу і необхідного типу серверу. Бібліотека COM звертається до системного реєстру, в якому, як було описано вище, зберігається інформація про сервер, дає запит і повертає необхідний вказівник. Бібліотека COM за переданим їй CLSID звертається до відповідної фабрики класу – спеціального об'єкта, що управляє створенням екземплярів необхідних об'єктів. Фабрика класу підтримує інтерфейс IClassFactory. Виклик фабрики класу здійснюється функцією CoGetClassObject, в яку передається CLSID необхідного класу і IID інтерфейсу IClassFactory. Функція повертає покажчик на інтерфейс необхідної фабрики класу. За допомогою цього покажчика викликається метод CoCreateInstance інтерфейсу фабрики класу з передачею в нього IID інтерфейсу. Ця функція створює екземпляр об'єкту і повертає покажчик на його інтерфейс, який бібліотека COM повертає клієнту.

Технологія COM – реалізується спеціальними бібліотеками, включаючи OLE32.dll і OLEAut32.dll. Вони містять стандартні інтерфейси й API з функціями, що забезпечують створення об'єктів COM і управління ними.

Починаючи з Delphi 5, взаємодія з Word, і багатьма іншими поширеними програмами, що входять в стандартну установку Microsoft Office, може здійснюватися з додатків Delphi за допомогою компонентів, розміщених у бібліотеці на сторінці Office2k, або Office97, або Servers. Ці компоненти відображають безліч імпортованих серверів COM. Всі вони є нащадками свого базового класу TOIeServer. В цьому класі оголошені абстрактні методи і властивості, що дозволяють встановлювати зв'язок з сервером. Тому об'єкти класу TOIeServer не можна створювати безпосередньо. В додатках використовуються тільки нащадки цього класу – конкретні сервери COM. Вони створюються імпортом бібліотек типів, про які йшлося вище.

Якщо перенести на форму компонент WordApplication і подивитися в Інспекторі Об'єктів його властивості, то їх виявиться дуже мало. Окрім звичайних для всіх компонентів Name і Tag є всього 4 властивості (в багатьох

компонентах-серверах їх 3). Властивість `AutoConnect` визначає, чи повинен сервер автоматично завантажуватися з початком виконання додатку. Якщо встановити `AutoConnect = true`, то з'єднання з сервером відбудеться у момент початку виконання вашого додатку. Якщо ж залишити значення `AutoConnect = false`, прийняте за умовчанням, то з'єднання з сервером можна встановити викликом методу `Connect`.

Властивість `ConnectKind` визначає, як саме здійснюється з'єднання з сервером. Ця властивість може приймати наступні значення:

ckRunningOrNew	Під'єднатися до сервера, що виконується, або створити новий екземпляр сервера.
ckNewInstance	Завжди створювати новий екземпляр сервера.
ckHunningInstance	Тільки під'єднатися до сервера, що виконується.
ckRemote	Під'єднатися до видаленого сервера. Ця опція повинна поєднуватися із завданням властивості <code>RemoteMachineName</code> .
ckAttachToInterface	Не під'єднуватися до сервера. Натомість додаток забезпечує інтерфейс методом <code>ConnectTo</code> . Опція <code>ckAttachToInterface</code> не може використовуватися спільно з установкою в <code>true</code> властивості <code>AutoConnect</code> .

Після того, як з'єднання з сервером встановлено, він ще не стає видимим користувачу. Втім, додаток може працювати з цим сервером, викликати будь-які його методи, змінювати або читати властивості, але сам сервер залишиться для користувача за кадром. Якщо це небажано, якщо вимагається, щоб користувач бачив, що відбувається на сервері, або міг би сам перемкнутися на сервер, то треба задати властивості `Visible` серверу значення `true`.

Розрив з'єднання з сервером здійснюється методом `Disconnect`. Крім того, у таких компонентах серверів COM, як `WordApplication`, є властивість `AutoQuit`. Якщо встановити цю властивість в `true`, то при завершенні додатку автоматично викличеться метод, який вивантажує сервер.

4.2. Програмна реалізація

Дана програма підтримує тришарову модель клієнт-сервер, тому одразу після запуску програми потрібно задати налаштування віддаленого сервера. У даного застосування є можливість під'єднуватись до серверів різних типів.

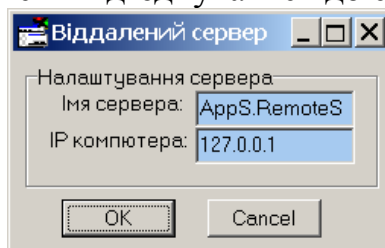


Рисунок 4.3 – Налаштування з'єднання з віддаленим сервером.

Далі (у головному вікні) користувач може вибирати, які наступні дії він хоче вчинити – генерувати звіти з таблиць, або ж заповнювати базу даних чи проміжні файли, використовуючи матеріали зібрані в документі Excel.

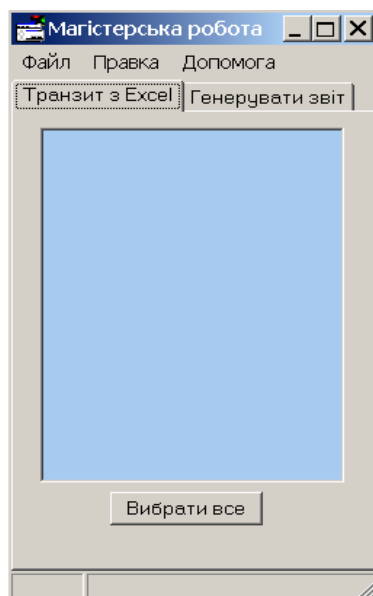


Рисунок 4.4 – Головне вікно програми.

Користувач може вибрати будь-яку базу даних, яка підтримує мову SQL і на яку налаштований драйвер ODBC (рис. 4.5).

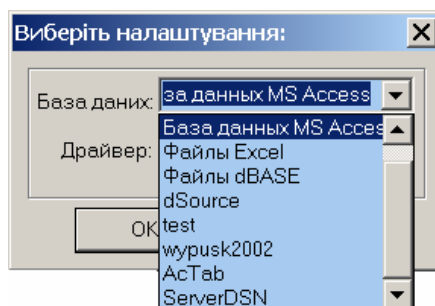


Рисунок 4.5 – З'єднання з базою даних через ODBC.

Є можливість відкрити файл Excel, у якому міститься зібрана інформація у вигляді таблиці (рис. 4.6).

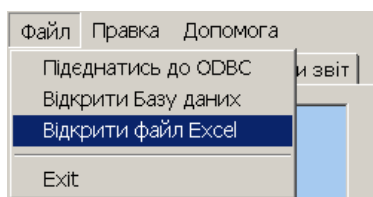


Рисунок 4.6 – Відкрити Excel файл.

Всі поля даних відображаються у вікні зображеному на рис. 4.7.

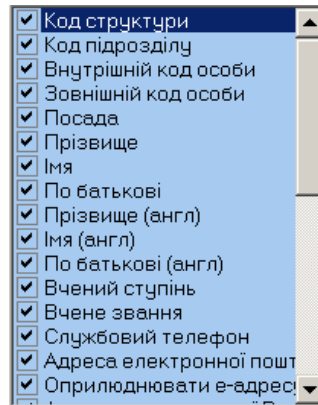


Рисунок 4.7 – Вікно вибору даних.

Користувач може вибрати поля, які його цікавлять, або ж усі поля, і передати всі дані у вже під'єднану базу даних, або ж у XML файл. Якщо користувач захоче передати дані у базу даних, програма запропонує ввести ім'я нової таблиці (рис. 4.8), якщо таблиця, до даного моменту не існувала, то вона створиться.

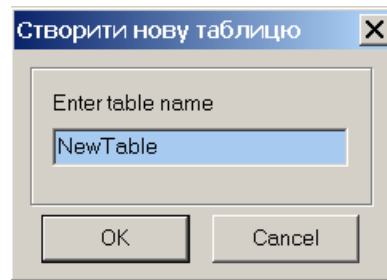


Рисунок 4.8 – Вікно вибору імені нової таблиці.

В залежності від вибору користувача, є різні варіанти збереження інформації. На рис. 4.9 зображені деякі з них.

Якщо ж користувач хоче згенерувати звітний документ з інформації збереженої в базі даних, то він аналогічно відкриває з'єднання через ODBC драйвер. Користувач сам вибирає таблиці та зв'язки між ними так, як це показано на рис. 4.10.

Установивши всі зв'язки між таблицями, користувач може приступити до генерування запиту. Для запиту можна вибрати тільки ті поля, які необхідно, і в потрібній послідовності. Є можливість сортування даних за певними полями і надання назви запиту. Всі запити, які були згенеровані зберігаються, з тією назвою, яку ми надали.

1	30001	Декан	Каличак	Ярослав	Михайлович
2	30002	Заступник	Пацай	Ігор	Орестович
3	30003	Заступник	Дмитрів	Григорій	Степанович
4	30004	Методист I	Міндрул	Ганна	Василівна
5	30005	Секретар	Гошко	Марія	Володимирівна
6	30101	завідувач	Гладишевський	Роман	Євгенович
7	30102	професор	Миськів		
8	30103	професор	Михалічк		
9	30104	професор	Павлюк		
10	30105	доцент	Дмитрів		
11	30106	доцент	Заремба		
12	30107	доцент	Кінжибал		
13	30108	доцент	Мокра		
14	30109	доцент	Стародуб		

NewTable : таблиця					
	Зовнішній	Посада			
17	30001	Декан			
18	30002	Заступник декана			
19	30003	Заступник декана			
20	30004	Методист I кат.			
	30005	Секретар			
	30101	завідувач кафедри			
	30102	професор			
	30103	професор	Михалічко	Борис	миронович
	30104	професор	Павлюк	Володимир	Васильович
	30105	доцент	Дмитрів	Григорій	Степанович
	30106	доцент	Заремба	Василь	Іванович
	30107	доцент	Кінжибало	Володимир	Васильович
	30108	доцент	Мокра	Іванна	Романівна


```

<?xml version="1.0" ?>
- <xml>
- <zapys number="1">
  <set4>Зовнішній код особи</set4>
  <set5>Посада</set5>
  <set6>Прізвище</set6>
  <set7>Імя</set7>
  <set8>По батькові</set8>
</zapys>
- <zapys number="2">
  <set4>030001</set4>
  <set5>Декан</set5>
  <set6>Каличак</set6>
  <set7>Ярослав</set7>
  <set8>Михайлович</set8>
</zapys>
- <zapys number="3">

```

Рисунок 4.9 – Транзит даних із Excel у таблиці та XML файл.

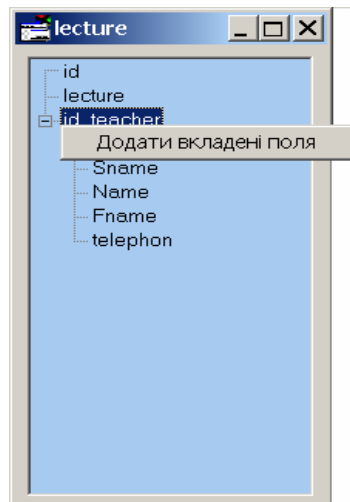


Рисунок 4.10 – Формування зв'язків між таблицями.

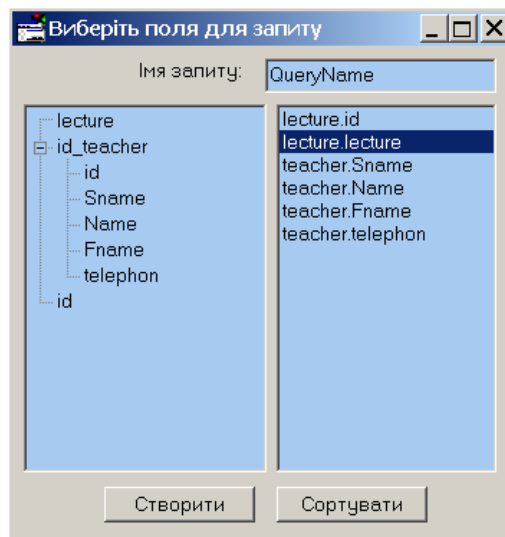


Рисунок 4.11 – Генерування запиту.

Для згенерованого запиту, користувач має можливість вибрати варіанти звітів: документ Word, електронні таблиці Excel чи XML файл.

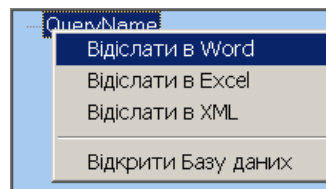


Рисунок 4.11 – Тип звітнього документа.

На рис. 4.13 зображені всі три варіанти автоматичного транзиту даних в звіт. Для генерування звіту у документ Word є можливість працювати зі стилями.

1							
ЧММФ							
Шинкаренко	A	B	C	D	E	F	
Георгій Андрійович 2232111	1	id	lecture	Sname	Name	Fname	telephon
	2	6	Клієнт-сер	Вовк	Володимир	Дмитрови	2222222
	3	4	Web serve	Горлач	Віталій	Михайлов	2233333
	4	5	HTML	Горлач	Віталій	Михайлов	2233333
	5	2	ПС	Венгерськ	Петро	Сергійови	2212345
	6	3	Java	Бернакеві	Ірина	Євстахієв	2213234
Address		D:\kurs\Новая папка\temp.xml		лф	Шинкарен	Георгій Андрійови	2232111

```

<?xml version="1.0" ?>
- <xml>
- <zapys number="1">
  <set1>id</set1>
  <set2>lecture</set2>
  <set3>Sname</set3>
  <set4>Name</set4>
  <set5>Fname</set5>
  <set6>telephon</set6>
</zapys>
- <zapys number="2">
  <set1>6</set1>

```

Рисунок 4.12 – Результати звітів.

4.3. Висновки

Розроблене програмне забезпечення дозволяє автоматично генерувати довідкові матеріали, зберігати їх в документах Word, Excel та XML файлах, а також переносити інформацію з електронних таблиць у вибрану базу даних, або в XML файл, як проміжну ланку.

Розроблене програмне забезпечення дозволяє користувачу відкривати будь-яку базу, самостійно задавати зв'язки між таблицями, вибирати головною будь-яку таблицю, створювати нові таблиці та вносити в них інформацію. Інформація відображається у тому порядку який задає користувач. Підтримується можливість роботи зі стилями програми Word.

Програмне забезпечення реалізоване у середовищі Delphi за допомогою технології ODBC, тришарової моделі клієнт-сервер та COM серверів Word та Excel.

5. ФОРМУВАННЯ ЗВІТІВ ДОВІЛЬНОЇ СКЛАДНОСТІ В СВІТЛІ ДОКУМЕНТОЦЕНТРИЧНОГО ПРОГРАМУВАННЯ

Формування звітів та інших довідкових матеріалів є на сьогоднішній день повсякденною задачею. Доволі часта зміна структури та форми документів приводить до того, що програми – генератори звітів, які орієнтовані на визначену структуру документів доволі швидко втрачають свою актуальність. Звідси слідує необхідність в розробці програмного забезпечення, незалежного від структури конкретного документу, а навпаки, орієнтованого на структуру, вказану користувачем. Разом з тим часто виникає необхідність у публікації сформованих документів. Інтернет є ідеальним інструментом для розповсюдження інформації, у зв'язку з чим публікація даних на веб-сторінці – це ефективний, простий і недорогий спосіб передачі інформації усім, хто в ній зацікавлений.

Можна легко простежити тенденцію переходу від паперових видань до електронних публікацій, що у свою чергу означає перехід від стандартних засобів видавництва до використання спеціалізованого програмного забезпечення для створення електронних документів.

Розглянемо один з підходів до створення сучасної системи побудови звітів, яка б, будучи орієнтованою на новітні розробки у формуванні звітних документів, вирішувала наведені вище проблеми та забезпечувала:

- можливість формувати звіт динамічно на основі інформації у базі даних (файлах даних);
- можливість формувати шаблони документів;
- можливість зв'язувати шаблон та схему даних, яка використовується у ньому;
- можливість побудови документів на основі шаблонів з вказанням джерела даних, що відповідає прив'язаній до шаблону схемі.

5.1. Технології реалізації проекту

Visual Studio 2005 for Team Developers запропонувала розробникам новий підхід до програмування Office-аплікацій. Цей підхід базується на програмуванні для продуктів Microsoft Office вже не з використанням скриптової мови VBA (Visual Basic for Applications), а використовуючи всі переваги об'єктно-орієнтованого підходу реалізованого у VisualBasic.NET чи C#.

При створенні даного програмного засобу (Office-аплікації) була надана перевага останньому. Для реалізації також було використано VSTO (Visual Studio Tools for Office), набір класів, який власне і забезпечує можливість програмувати Office-аплікації використовуючи об'єктну модель офісних застосувань (Word, Excel, Outlook Express та інші). Розглянемо ці засоби детальніше.

5.1.1. Платформа .NET та .NET Framework

.NET – це перш за все концепція, яка виражає цілісний погляд на нову епоху у розвитку інформаційних систем та Інтернету, коли надзвичайно різні програмні продукти потрапляють до користувачів як взаємодіючі сервіси, які доступні на самих різних пристроях: від мобільних телефонів до дуже потужних кластерів. Основа .NET уявно складається з двох частин – засобу розробки .NET – програм Visual Studio.NET і не менш важливого середовища виконання цих програм, яке отримало назву .NET Framework. Розглянемо останню частину детальніше.

Особливістю .NET Framework є наявність керованого коду (managed code), який працює у середовищі виконання CLR (Common Language Runtime). CLR підтримує багатший набір служб, ніж 32-розрядна операційна система Windows. Будь-яке середовище розробки, сумісне з CLR (зокрема, Visual Studio) компілює вихідний текст програми в проміжну мову IL (Intermediate Language). Для виконання коду на IL треба здійснити миттєву компіляцію (just-in-time, JIT). Компілятор (JITer) читає вихідний текст і виходячи з конкретної платформи запуску, продукує «рідний» машинний код, який потім і виконується. При запуску виконуваного файлу, написаного на CLR-сумісній мові .NET, системний завантажувач визначає, що це керований код, і передає його ядру CLR. Ядро, знайшовши IL у Exe-файлі, викличе JIT-компілятор, який перетворить програму на IL у машинні коди процесора і операційної системи, під керуванням якої він працює. Таким чином, для кожної архітектури має бути свій власний JITer, який відповідатиме особливостям системи.

Разом з тим .NET є продуктом нового часу, орієнтованим на розробку програм, які б складались з окремих компонентів, які б легко модифікувались, а не були монолітами, модифікація яких практично не можлива. Даний продукт, на відміну від інших, не намагався наздогнати і втілити новітні розробки у собі, а створювався відразу з врахуванням наявних новітніх технологій, таких, наприклад, як XML, WML, нових підходів до взаємодії з базами даних тощо. Новітній підхід Microsoft до концепції мережі Інтернет – створення засобів, які б дозволили взаємодіяти локальним комп'ютерам, мобільним пристроям та навіть побутовим електропристроєм відкриває перед розробниками програмного забезпечення величезні горизонти.

Нова версія .NET Framework – версія 2.0 надає ще більше можливостей і засобів для програмування зокрема Office-аплікацій.

5.1.2. Типова Архітектура Word – проекту

В найбільш загальній формі Word-проекти, створені з використанням VSTO складаються з 2 файлів: Word-документа та збірки у вигляді dll-файлу. В таких проектах збірка є прив'язаною до документа, але зберігається окремо. Документ має у собі невидимий елемент Runtime Storage Control, який містить деяку додаткову інформацію та місцезнаходження збірки. Така архітектура забезпечує обробку певних подій у збірці, що дозволяє здійснювати легкий

доступ до джерела даних (бази даних), заповнювати інформацією певні поля в документі чи просто реагувати на події зміною налаштувань самого Word.

5.1.3. Порівняння VBA та .NET Framework для написання Office-аплікацій

VBA використовує некерований код, який тісно інтегрований з Office-аплікаціями. Разом з тим керований код написаний на C# дає доступ до всієї функціональності платформи .NET.

VBA використовує код, який прив'язаний до документа та зберігається з ним. В .NET Framework код зберігається окремо, отже потенційно може бути використаний кількома документами.

VBA використовує об'єктну модель Office та API VBA. .NET Framework використовує об'єктну модель Office та .NET Framework API.

VBA використовується коли потрібна тісна взаємодія з Office-аплікацією (наприклад створення меню). .NET Framework використовується для написання проектів, які будуть використовувати переваги Office-аплікацій, але разом з тим і всю доступну інфраструктуру .NET.

Звідси слідує, що VBA використовується для невеликих проектів, в той час як .NET Framework легко можна застосувати у проектах рівня підприємства.

5.1.4. Об'єктна модель Word

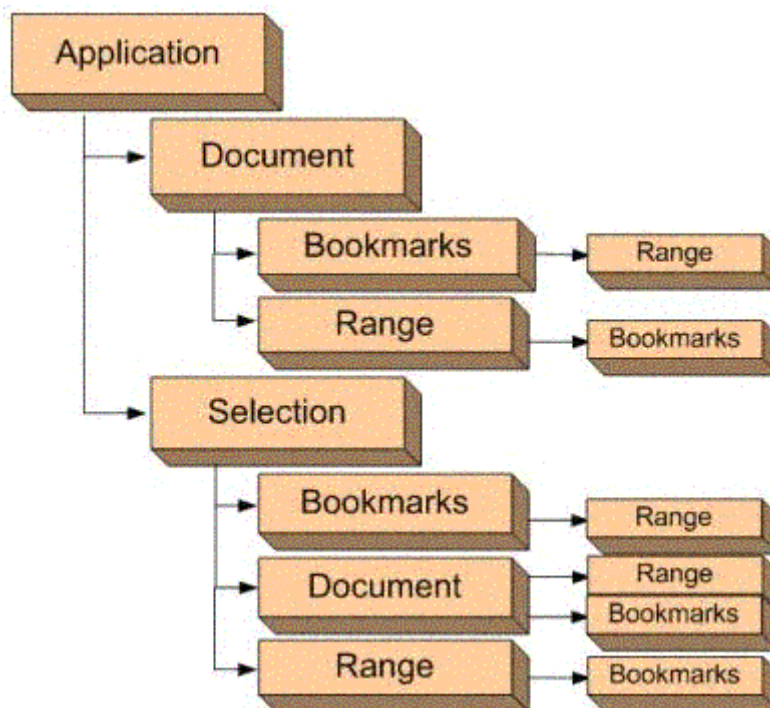


Рисунок 5.1 – Об'єктна модель Word

Application-об'єкт являє собою аплікацію та містить методи, які застосовуються до програми в цілому.

Document-об'єкт являє собою один з відкритих документів. Якщо даний документ має зараз фокус, то ми можемо звернутися до нього як до Application.ActiveDocument.

Selection-об'єкт представляє або виділення в документі або позицію курсора.

Range-об'єкт предсталає собою неперервну область в документі.

Bookmark-об'єкт майже аналогічний до Range, але даний елемент може зберігатись з документом, йому можна дати ім'я. Цей об'єкт має розширену, в порівнянні з Range, функціональність.

5.2. Опис системи

Для зручного встановлення на комп'ютері кінцевого користувача створено інсталяційний пакет. Даний пакет забезпечує швидкий, простий та зрозумілий процес встановлення файлів аплікації і запобігає можливим помилкам при встановленні файлів аплікації на комп'ютері кінцевого користувача (рис. 5.2).

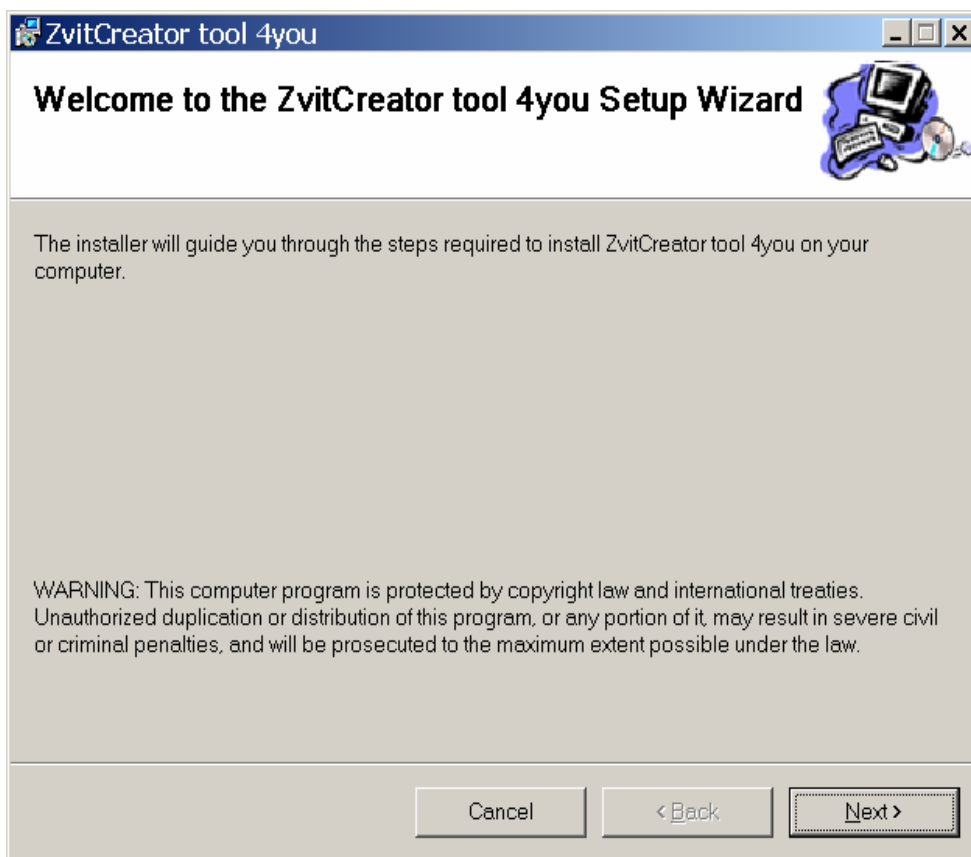


Рисунок 5.2 – Встановлення програми

5.2.1. Структура каталогів проекту

Функціонування програми базується на строго визначеній структурі каталогів, яка має такий вигляд (рис. 5.3):

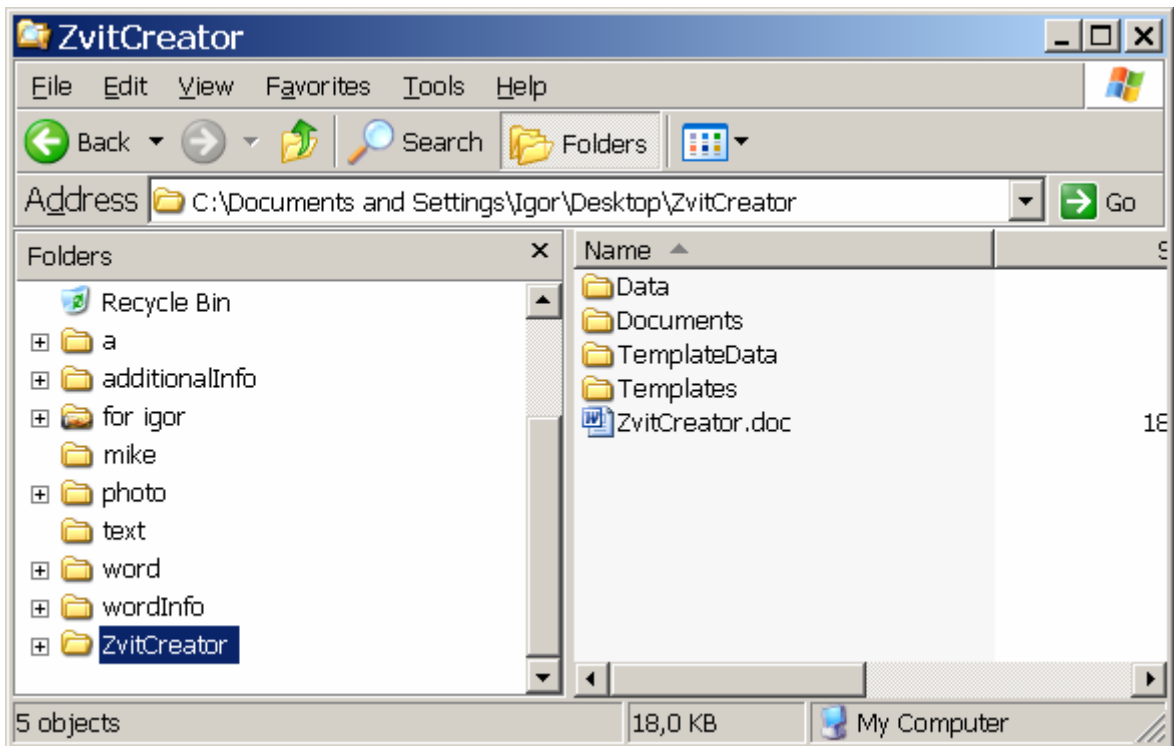


Рисунок 5.3 – Структура каталогів, які використовує програма

В папці Data є набір каталогів з файлами даних, які будуть використані при створенні звіту. Робота програми базується на використанні *.dbf файлів. В майбутньому планується розширення діапазону форматів, які будуть підтримуватись.

Папки Documents, Templates та TemplateData маю схожу структуру, яка виглядає так як показано на (рис. 5.4). Ці папки містять низку підпапок, назва кожної з яких є датою створення документів.

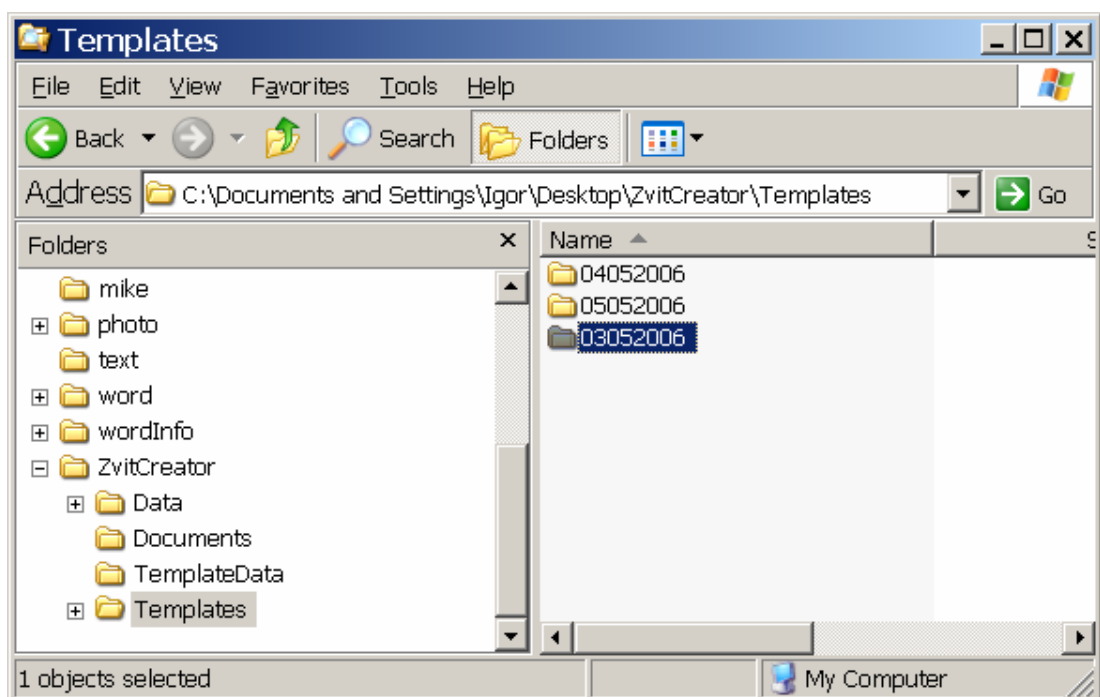


Рисунок 5.4 Структура папок Documents, Templates та TemplateData

Кожна з підпапок у папках Documents, Templates та TemplateData має таку структуру (рис. 5.5) :

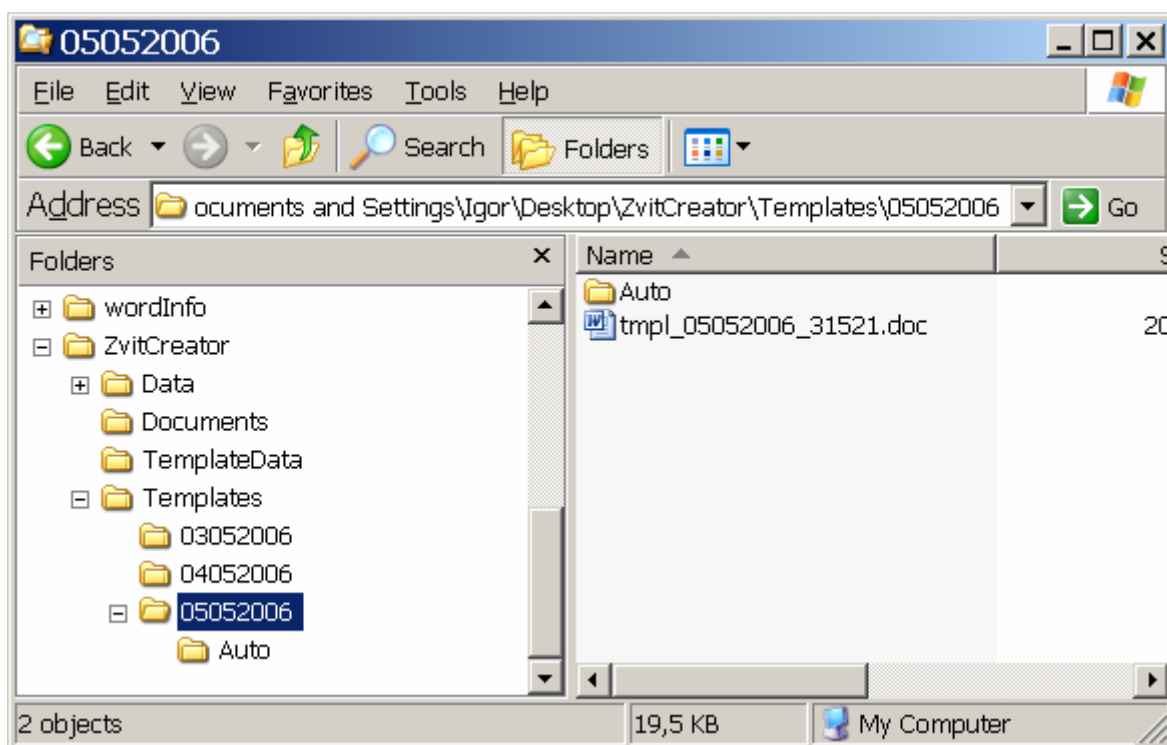


Рисунок 5.5 – Структура підпапок Documents, Templates та TemplateData

Кожна така папка містить набір файлів – документів, шаблонів, чи файлів які описують структуру даних, використаних у документі. Назва документа формується автоматично за таким принципом :

- Для папки Documents назва складається з таких частин `zvit_ПоточнаДата_ПоточнаГодина.doc`
- Для папки Templates назва матиме вигляд `tmpl_ПоточнаДата_ПоточнаГодина.doc`
- Для папки TemplateData назва файлу відрізнятиметься і матиме такий вигляд `НазваВідповідногоФайлу.xml` (де НазваВідповідногоФайлу – це `tmpl_ПоточнаДата_ПоточнаГодина`, тобто назва файлу для якого ми зберігаємо структуру зв'язків між даними та документом).

Разом з тим кожна така папка містить спеціальну папку Auto, яка зберігає документи відповідного типу, які не були явно збережені користувачем (наприклад, на запитання чи зберігати зміни у документі була дано відповідь «Ні»).

5.2.2. Користування програмою

Оскільки програма за суттю є розширенням MS Word, то при її запуску ми бачимо дещо модифіковане вікно текстового редактора (рис. 5.6). Модифікації стосуються елемента TaskPane – при запуску аплікації створюється додаткова закладка для цього елемента управління – DocumentActions, яку при відкритті звичайних документів користувач не бачить.

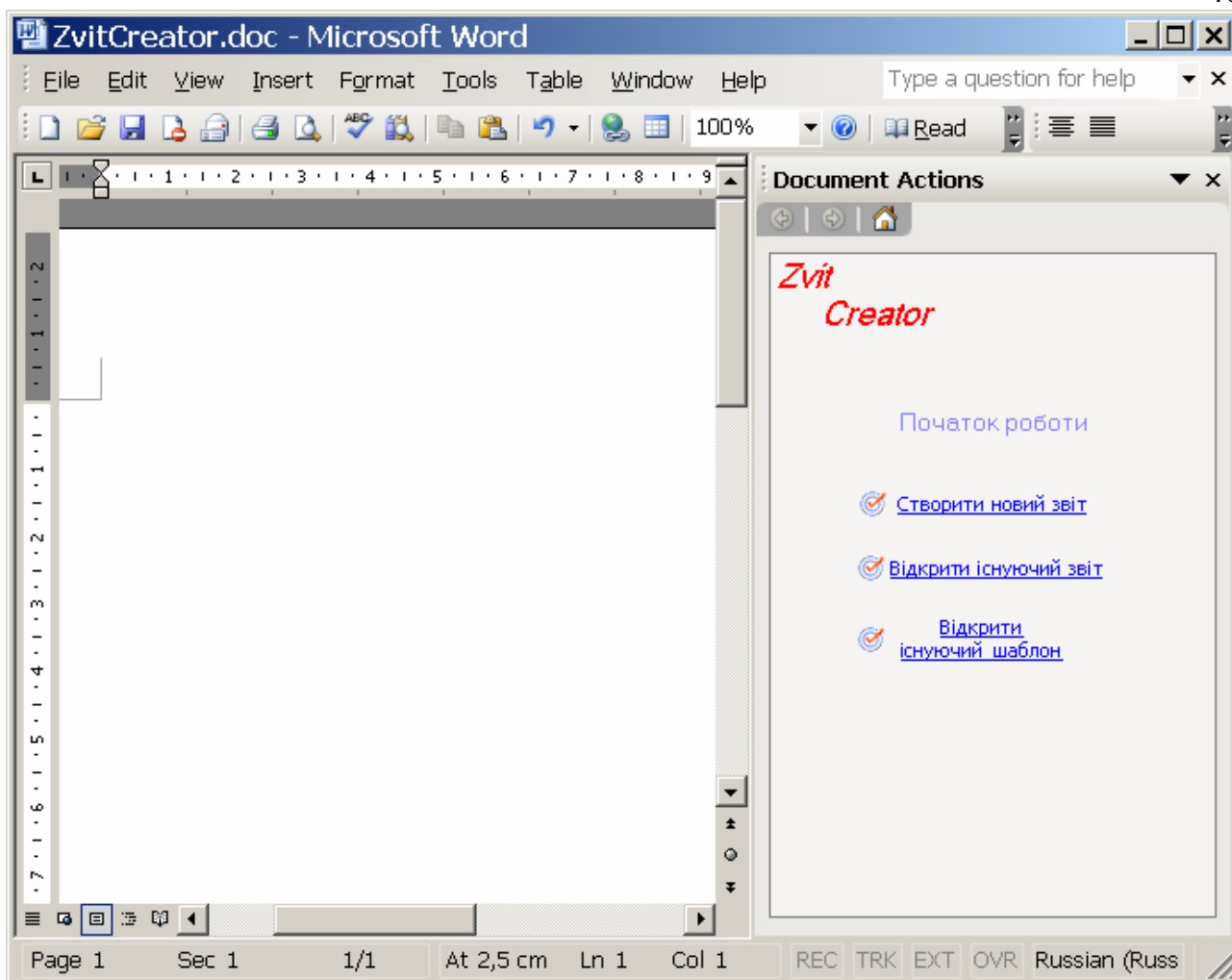


Рисунок 5.6 – Запуск аплікації

Крок 1. Створення нового звіту.

При виборі цього пункту відкривається вікно (рис. 5.7), яке запитує файли даних, що будуть використані у даному звіті. Оскільки папка Data містить інформацію у підпапках, то вибір файлів зручно робити попередньо вибравши одну з доступних директорій. Наступне вікно аналогічно запитує поля кожної з вибраних таблиць, які будуть використані у звіті.

Зауважимо, що ні для файлів, ні для полів немає необхідності відразу додавати всі необхідні файли чи поля відповідно, оскільки це просто зробити у процесі роботи – за необхідністю ми можемо як додавати потрібні елементи, так і вилучати зайві.

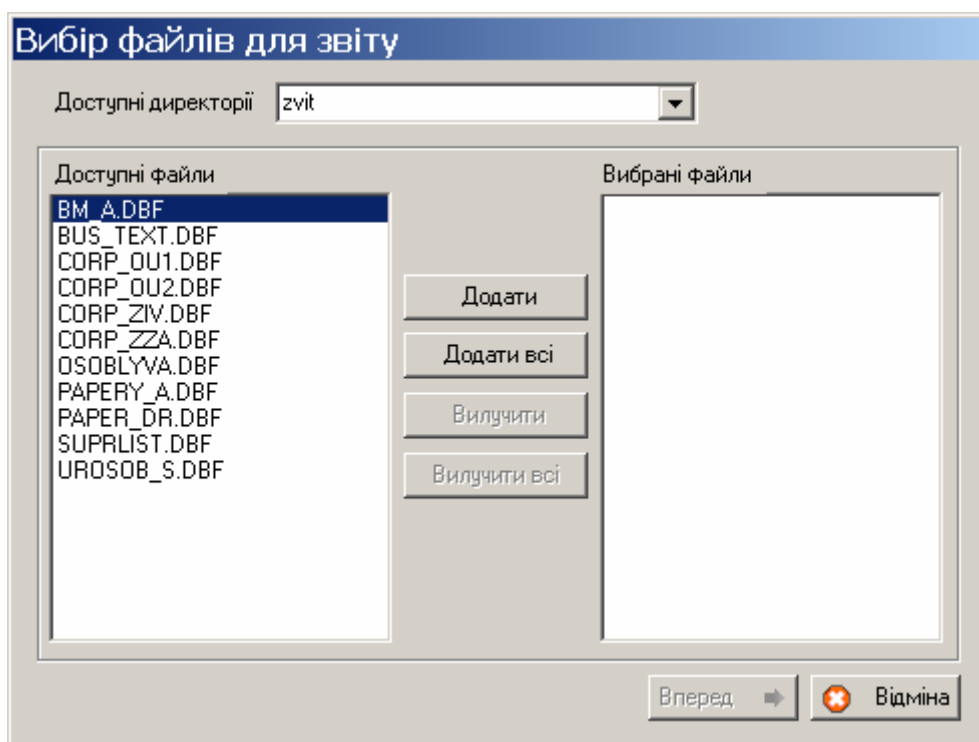


Рисунок 5.7 – Вибір файлів

На головній формі (рис. 5.8) можна за допомогою відповідних меню безпосередньо формувати звіт.

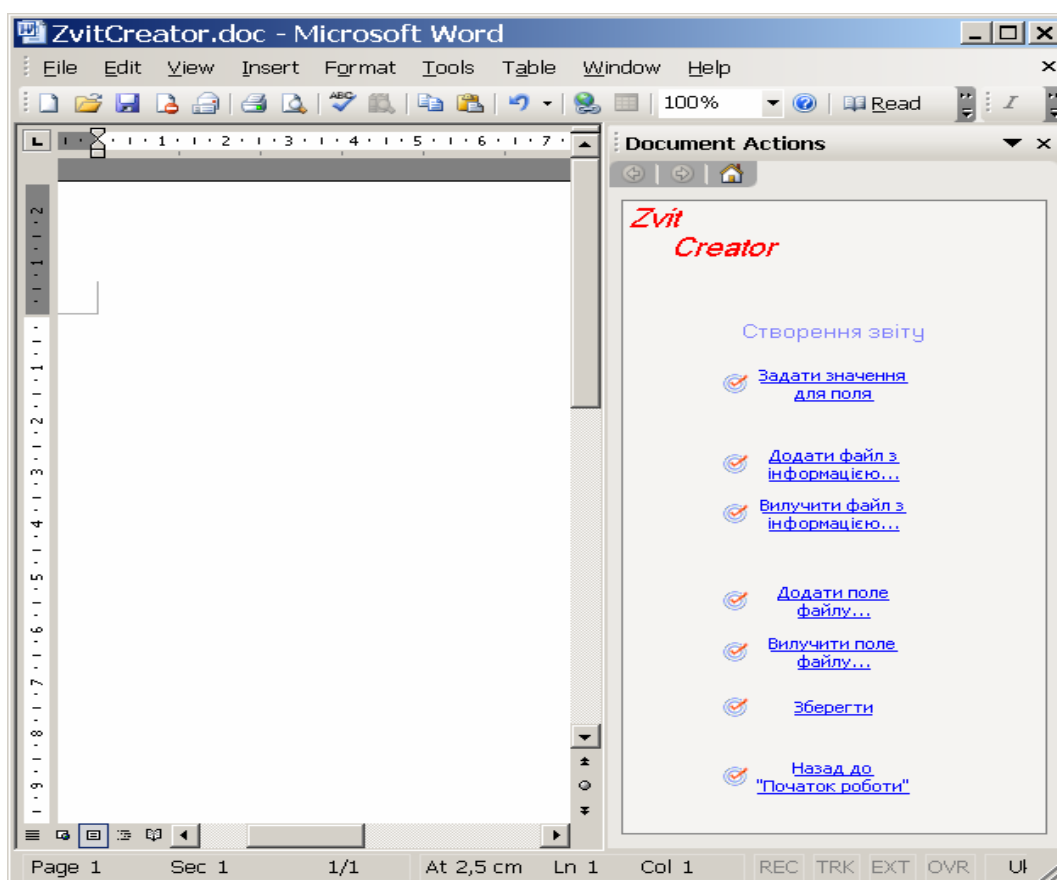


Рисунок 5.8 – Головна форма

Ця процедура виглядає наступним чином – вносимо в документ деякий статичний текст, який не залежить від даних. Нехай тепер потрібно внести певну інформацію з файлів даних. Для цього ставимо курсор у потрібну позицію та вибираємо меню «Задати значення для поля». Відкривається вікно (рис. 5.9) – з якого файлу, поля та який саме запис слід вибрати, якщо їх більше одного.

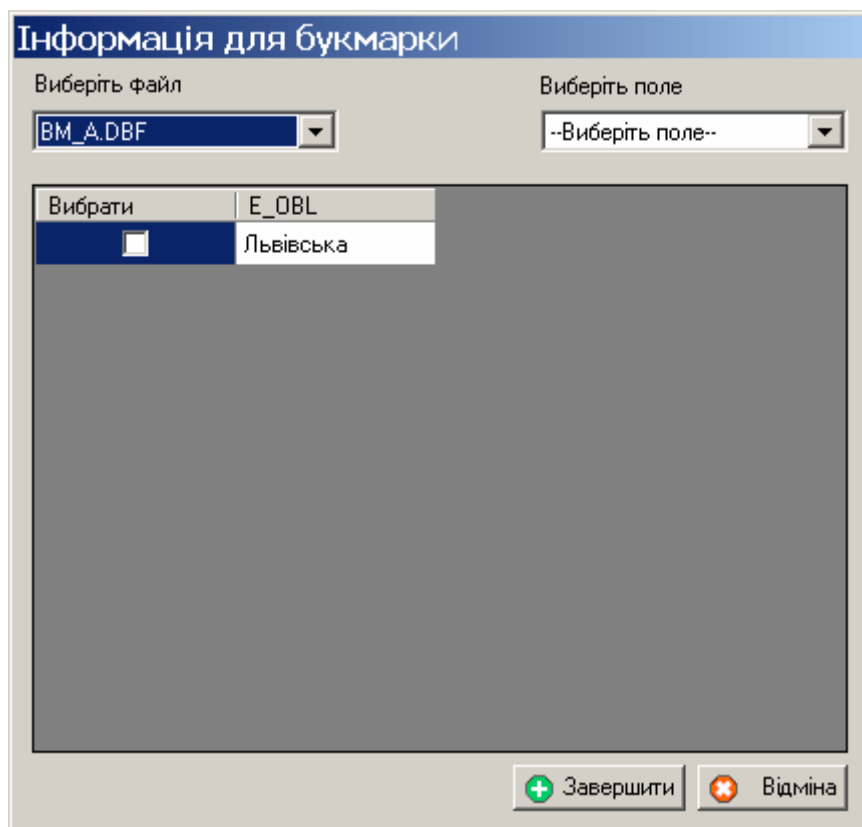


Рисунок 5.9 – Задання значення з файлу даних

В результаті після натискання кнопки «Закінчити» отримаємо ситуацію схожу до тієї, що на (рис. 5.10) : «Місцезнаходження» – деякий статичний текст тоді як «Львівська» – інформація з файлу даних.

Після того, як задано всю необхідну інформацію звіт необхідно зберегти клацнувши на кнопку «Зберегти». В наслідок цього буде створено наступний набір файлів:

1. zvit_05052006_42227.doc
2. tpl_05052006_42227.doc
3. tpl_05052006_42227.xml

де

zvit_05052006_42227.doc – файл з текстом звіту, готовий до друку чи публікації в разі необхідності;

tpl_05052006_42227.doc – шаблон звіту, який не містить інформації заповненої з файлу даних, а лише статичний текст та мітки де повинна бути інформація з файлів даних;

tpl_05052006_42227.xml – файл, який містить інформацію про зв'язок між файлом – шаблоном та файлами даних. Тобто інформацію про те, які саме

файли будуть використані при створенні нового документа на основі шаблону, та які поля цих файлів будуть використані при цьому.

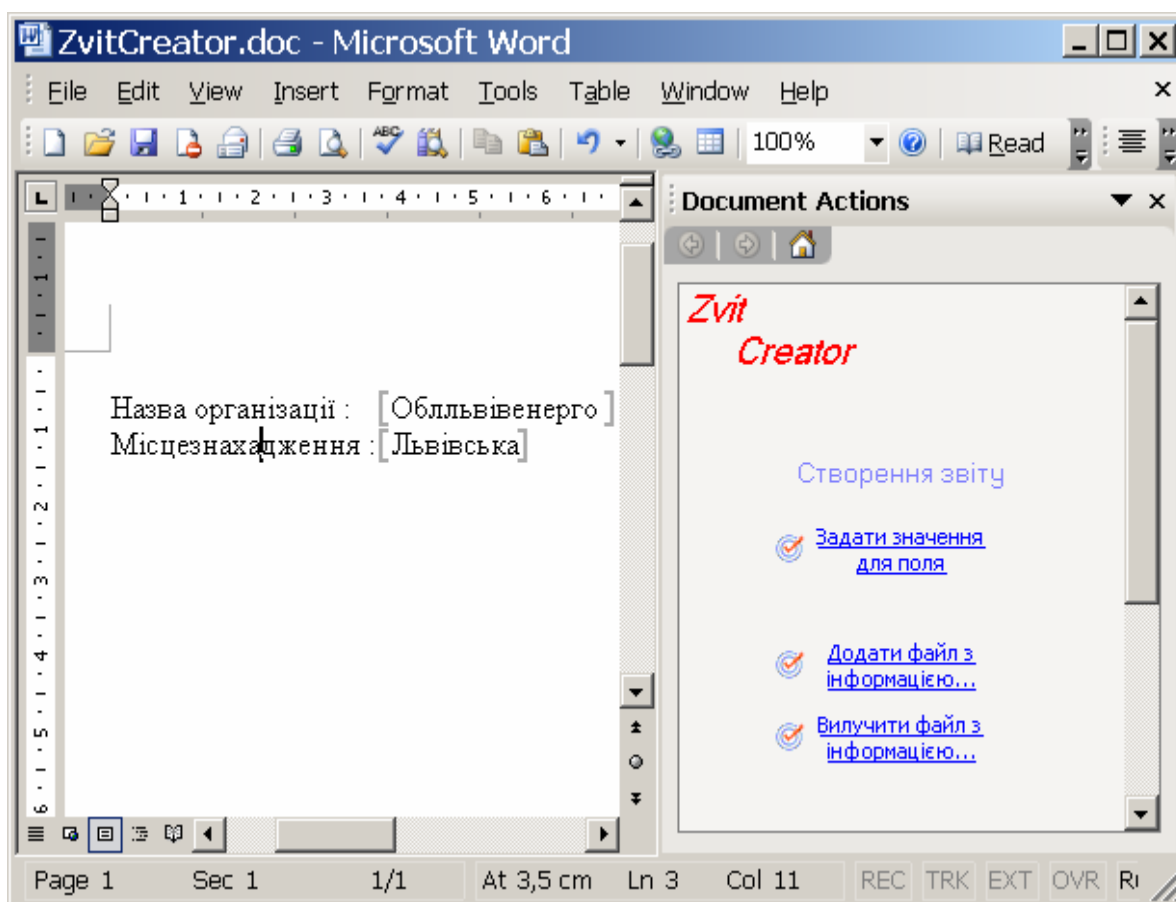


Рисунок 5.10 – Задання значення полів з файлу даних

Крок 2. Відкриття існуючого документа.

На цьому кроці ми можемо відкрити документ сформований раніше для того, щоб відредагувати його. Для редагування документів нам необхідно знову вказати файли даних та їх поля, які треба використовувати.

Крок 3. Відкриття існуючого шаблону.

Дії на цьому кроці аналогічні до дій на попередньому, єдина відмінність в тому, що для шаблонів зберігається схема привязки до даних, тому відкриваючи шаблон і вказуючи при відкритті кожен раз інші папки з файлами даних, ми будемо отримувати інші документи (документи з тою самою структурою, але різним наповненням). Єдина вимога для того, щоб шаблон був завантажений – це наявність у вказаній папці файлів з такою ж структурою, як і при створенні шаблону.

5.3. Висновки

В даному розділі розглянуто систему для створення документів з динамічною структурою, яка задається користувачем. Створені документи можуть бути використані для представлення в мережі Інтернет.

Система реалізована за допомогою платформи .NET, мови програмування C#.NET, та Visual Studio Tools for Office (VSTO). Скористатися цим проектом

можна на будь-якому комп'ютері за умови, що на комп'ютері встановлено .NET Framework, та кілька додаткових пакетів необхідних для взаємодії Microsoft Office і .NET Framework.

Завдяки повній інтегрованості з MS Word система має доступ до багатого набору функціональних можливостей Word для форматування тексту звіту та його вигляду в цілому, додаткових можливостей таких як друк документа, збереження файлу у кількох форматах тощо.

6. РОЗРОБКА МЕТОДИКИ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ПРЕЗЕНТАЦІЙНИХ МАТЕРІАЛІВ

Останні десятиліття характеризуються бурхливим розвитком інформаційних технологій. Зокрема, активно розвиваються і технології комп'ютерного дизайну. Якщо раніше основна увага зосереджувалась на самій інформації, то тепер все більше значення надають її представленню.

Залежно від призначення майбутнього проекту, розробник вибирає потрібний спосіб представлення інформації. Зупинимось на представленні інформації у вигляді мультимедійних презентацій.

Існуючі пакети програм для створення презентацій достатньо прості у роботі, активно використовують можливості мультимедіа, забезпечують зручний імпорт відео- та звукових файлів, анімацію зображень. Одна з перших програм цього класу – Harvard Graphics, містить бібліотеку графічних зображень для ілюстрацій, дозволяє створювати багато типів графіків та діаграм, передбачає різноманітні ефекти виводу зображення під час презентації. Найвідоміша та найпоширеніша на сьогодні програма – PowerPoint з пакету програм Microsoft Office. Вона має в своєму складі навчальну програму, готові зразки презентацій, ефективну довідкову систему, надає широкі можливості роботи з текстом. Adobe Persuasion призначена для підготовки презентацій за допомогою слайдів. Програмним забезпеченням передбачено створення шаблонів слайдів, вибір способу заміни одного слайду іншим та способу відтворення зображення на екрані. Для презентації на моніторі комп'ютера чи на екрані ідеально підходить програма Macromedia Director. Можливість створення професійних роликів з покадровим заповненням, вставки відеофрагментів, а також інтерактивний показ забезпечили програмі популярність серед широкого загалу. Зовсім інший підхід до розробки мультимедійних проектів реалізує Macromedia Flash. Головним призначенням цієї програми є створення відеокліпів з ціллю їх вставки в інші електронні документи, чи самостійної публікації. Наявні всі необхідні можливості і для розробки мультимедійних презентацій.

У даному розділі запропонована методика та розроблено програмні засоби для автоматичної генерації (оновлення, редагування) мультимедійних матеріалів для будь-якої структури даних. Дана задача вирішується в два етапи:

1. Розробка оболонки (прикладної програми) для створення структури для нових презентаційних матеріалів або для редагування вже існуючої презентації.
2. Генерація презентаційних матеріалів на основі структури даних, побудованих за допомогою оболонки.

Головний акцент ставиться на незалежності презентаційного проекту та інформаційних даних. За допомогою створеної оболонки, звичайний користувач на інтуїтивному рівні зможе легко і просто доступитись до будь-

якої інформації, будь це текст, графіка, відеоролик чи зовнішні ресурси. Фактично, це дозволяє просте редагування та оновлення інформації.

На рис. 6.1 схематично представлено ідею реалізації автоматичного генерування презентаційних матеріалів.



Рисунок 6.1 – Ілюстрація ідеї автоматичного генерування презентаційних матеріалів

Презентація реалізована засобами Macromedia Flash, а прикладна програма – Builder C++ .

6.1. Структура даних проекту

В основі реалізації оболонки лежить робота з базою даних, яка містить одну таблицю, кожен рядок якої відповідає пункту меню в презентації. У табл. 6.1 показано структуру таблиці бази даних:

Таблиця 6.1 – Структура таблиці бази даних

№	Назва поля	Тип поля	Опис поля
1	<i>Id</i>	integer	Ідентифікаційний номер елемента. Дане поле є ключовим.
2	<i>Name</i>	string	Ім'я елемента, яке використовується як назва кнопки в українському варіанті презентації.
3	<i>en_name</i>	string	Ім'я елемента, яке використовується як назва кнопки в англійському варіанті презентації.
4	<i>Parent</i>	integer	Ідентифікаційний номер елемента, який є батьківським для даного елемента.
5	<i>Inf</i>	integer	Тип інформації, яка буде відобразитись після натискання на відповідну кнопку.

Для поля *inf* можливими є лише шість значень, які відповідають таким варіантам:

- текст (*.txt);

- зображення (*.jpg);
- фільм (*.avi);
- зовнішні ресурси (будь-який файл, який можна відкрити за допомогою Інтернет-браузера);
- графічна галерея (декілька графічних файлів в форматі jpg);
- жоден з вище зазначених.

Використання поля *parent* дає можливість зв'язати дочірній елемент з батьківським. Отже, структуру даних з похідними елементами, на якій буде базуватись презентація, легко можна записати в таку таблицю, не порушуючи коректності зв'язування елементів.

Оболонка працює з даною базою даних. Доступні такі дії: редагування і знищення виділеного елемента, додавання нового елемента, знищення всіх елементів, налаштування презентації. Також можливим є завантаження і редагування інформації, яка відповідає обраному елементу, тобто відповідному пункту меню. Ця інформація записується в файли, ім'я яких формується з врахуванням виду інформації і унікального номеру пункту меню, при натисканні на яке завантажиться даний файл. Всі ці файли зберігаються в папці *Source*. Оскільки в презентації відображається різнопланова інформація, то типи файлів, в яких вона зберігається, теж різні. Щоб визначити, який саме файл необхідно відкрити, використовується поле таблиці *inf*.

Оскільки в Flash MX не реалізовано можливість роботи з базою даних, то вся інформація, яка зберігається в базі даних певним чином записується в текстовий файл *base.txt*, який зберігається у кодуванні UNICODE. Цей файл зчитується в презентації, розбивається на змінні, на основі яких будується структура меню презентації. Файл *base.txt* містить лише інформацію про елементи меню, а інформація, яка буде відображатись при натисканні кнопки меню, зчитується з папки *Source* з врахуванням її унікального номеру. Ця ідея проілюстрована на рисунку 2.

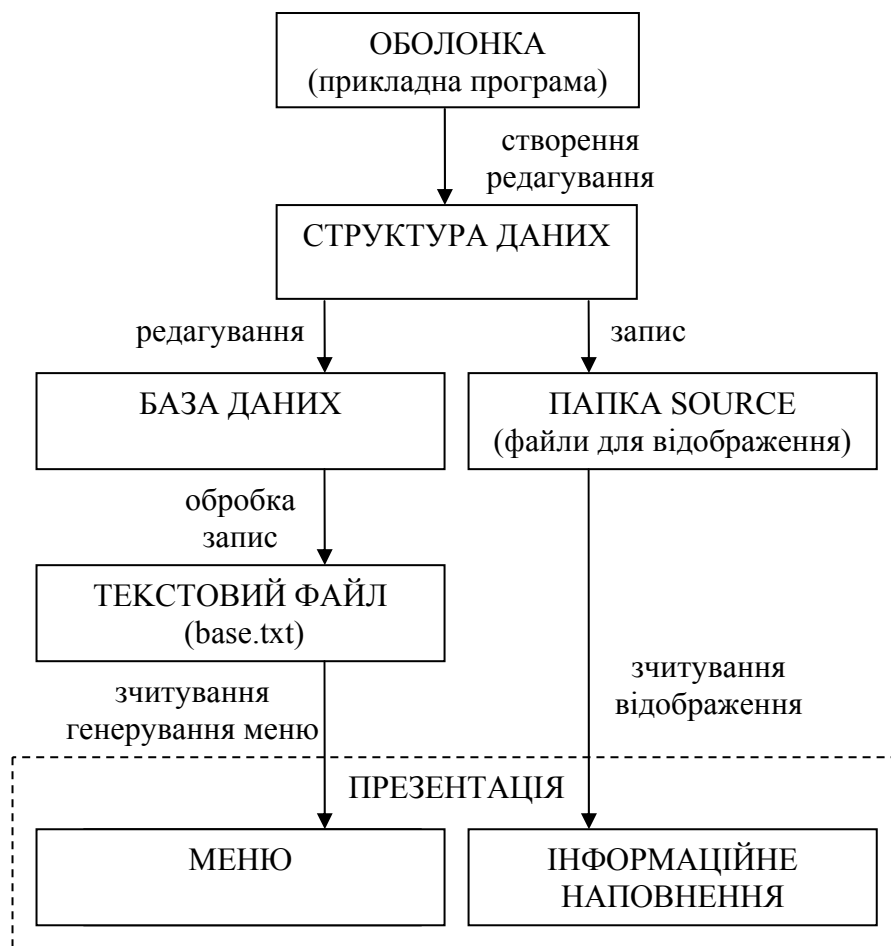


Рисунок 6.2 – Схема автоматичного генерування презентаційних матеріалів

Для створення оболонки використовувались компоненти, такі як TTntEdit, TTntMemo. Ці компоненти не відрізняються від їхніх аналогів, які є стандартними для Builder C++. Єдиною відмінністю цих компонент є те, що вони працюють з текстом в кодуванні Unicode, а не з текстом типу AnsiString.

6.2. Опис прикладної програми для редагування презентації

Оскільки робота оболонки пов'язана з базою даних, необхідно її підключити. Це робиться наступним чином:

- запускаємо програму *Адміністратор джерел даних(ODBC)*. Вона знаходиться в *Старт – Панель інструментів – Адміністрування – Джерела даних(ODBC)*;
- в закладці *Користувацький DNS* натискаємо кнопку *Додати...*;
- вибираємо драйвер *Microsoft Access Driver(*.mdb)* і натискаємо кнопку *Готово*;
- записуємо в полі *Ім'я джерела даних base* і за допомогою кнопки *Вибрати...* вибираємо базу даних;
- натискаємо *ОК*.

Після цих дій оболонка буде працювати з потрібною базою даних.

Прикладна програма завантажується з файла *Presentation_Editor.exe* (рис. 6.3).

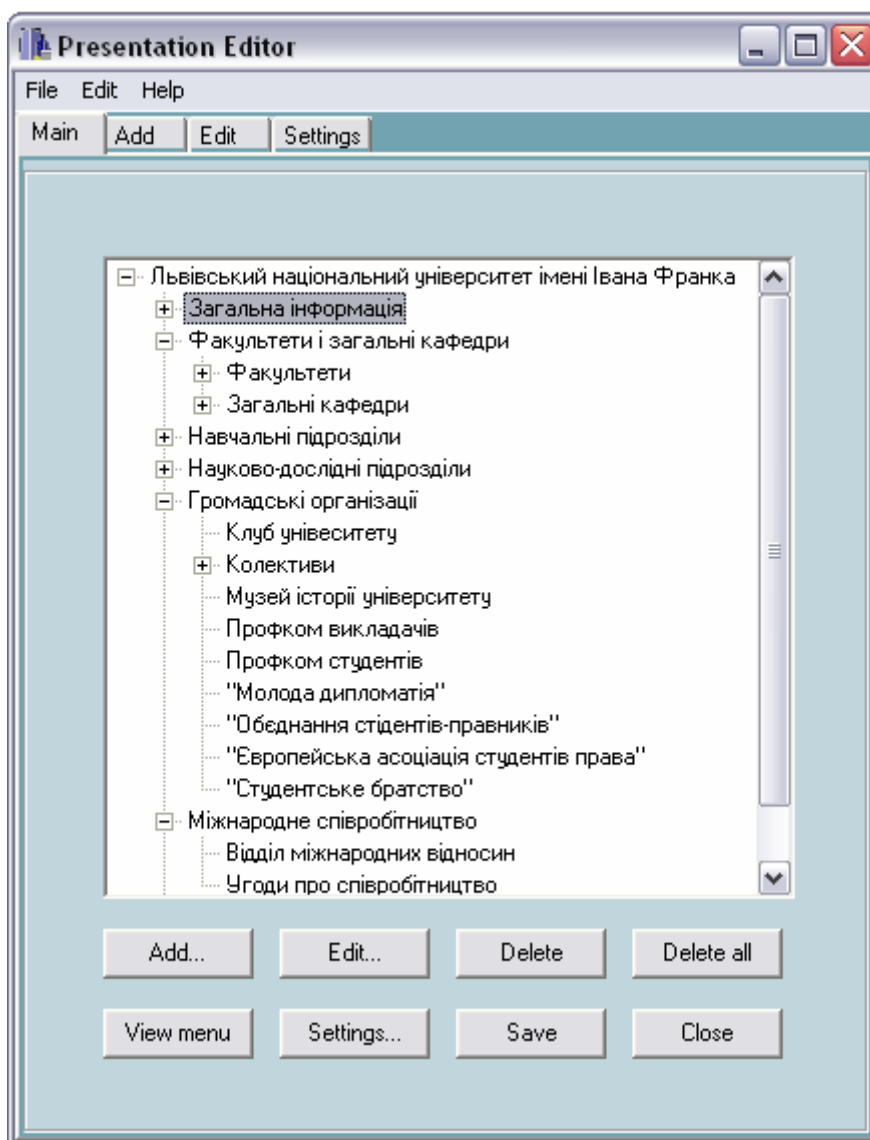


Рисунок 6.3 – Головне вікно програми, яке відображає структуру меню презентації

Кнопка *Delete* знищує виділений елемент зі структури даних. Кореневий елемент («Львівський національний університет імені Івана Франка» на рис. 6.3) не може бути знищений. Назва кореневого елемента використовується як назва презентації, а інформація, що йому відповідає завантажується на другу сторінку презентації.

Кнопка *Delete all*, що знаходиться на головній закладці, знищує всі елементи зі структури даних крім кореневого, а кореновому елементу присвоює назву *Презентація*.

Кнопка *Settings...* відкриває закладку для редагування параметрів презентації (рис. 6.4)

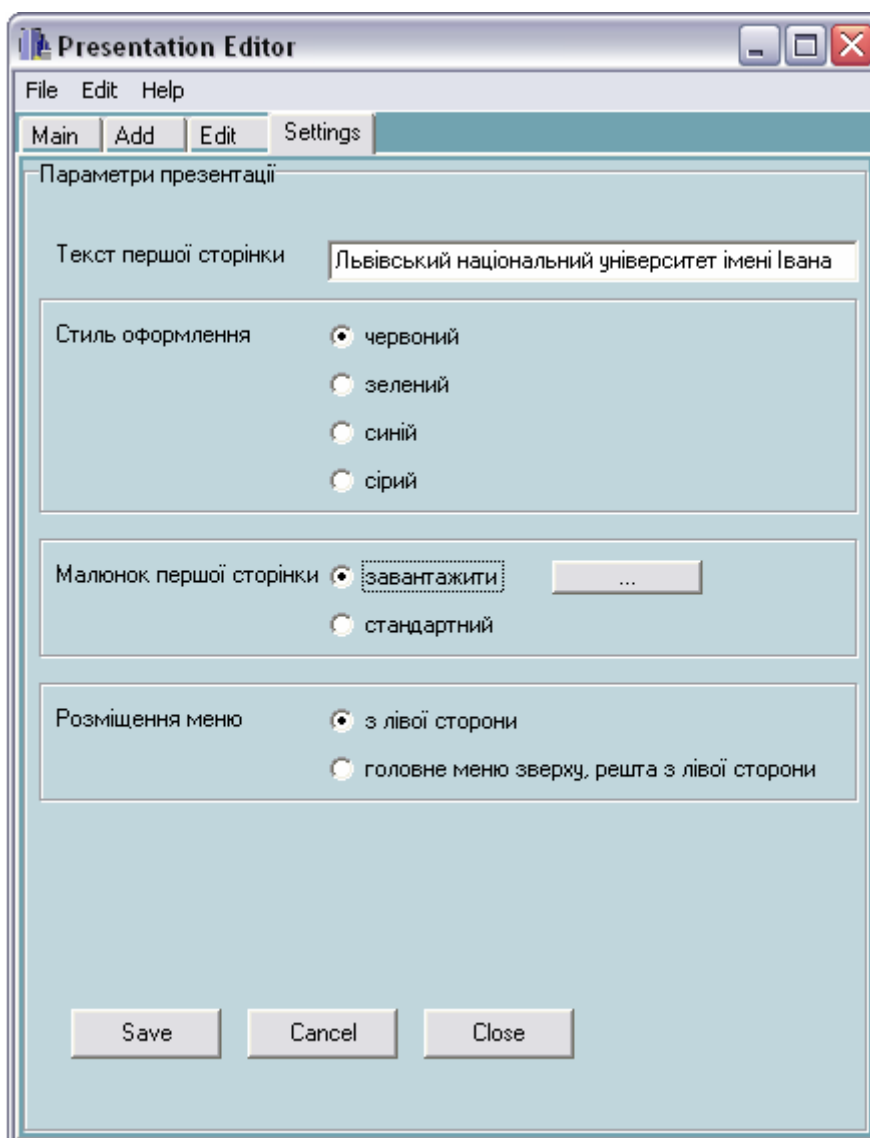


Рисунок 6.4 – Закладка *Settings...* для редагування параметрів презентації

У вікні редагування параметрів є такі поля:

- *Текст першої сторінки* – текст, який буде відображатись на першій сторінці.

- *Малюнок першої сторінки* – пропонується вибрати малюнок, який буде відображатись на першій сторінці. Варіант *стандартний* передбачає зображення малюнку, який відповідає стилю оформлення презентації. Також можливо завантажити будь-який інший малюнок. Це можна зробити, відмітивши індикатор *завантажити* і клацнувши мишкою по кнопці справа. Після цього з'явиться вікно для вибору графічного файлу (рис. 6.5).

- *Стиль оформлення* – відображає стиль оформлення презентації. Можливі такі варіанти: червоний, зелений, синій, сірий.

- *Розміщення меню* – пропонується вибрати місце розміщення меню у вікні презентації. Можливі такі варіанти розміщення: з лівої сторони або головне меню зверху, решта зліва.

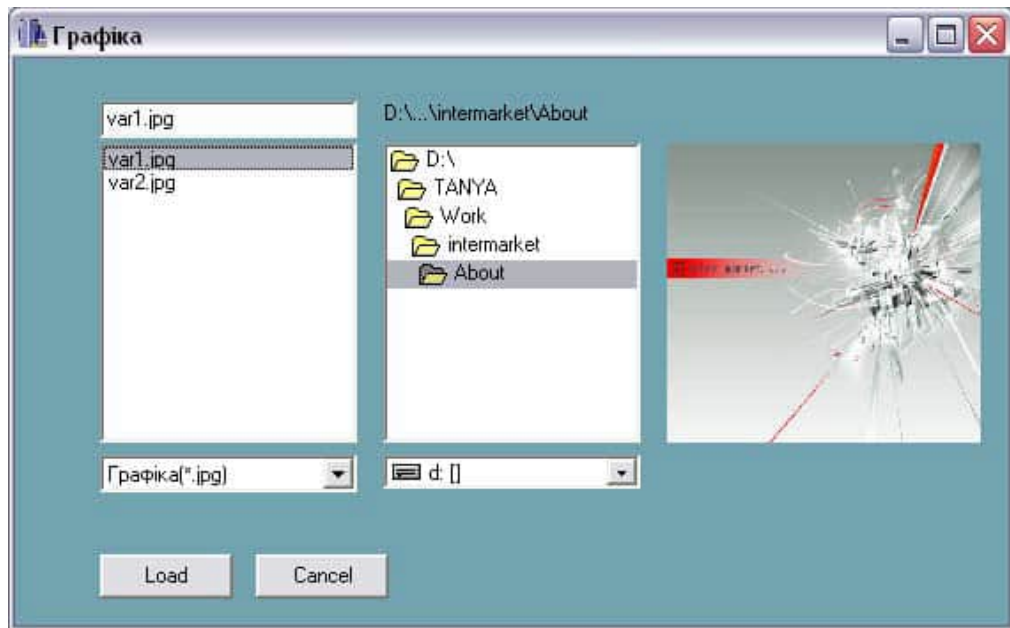


Рисунок 6.5 – Вікно завантаження графічної інформації

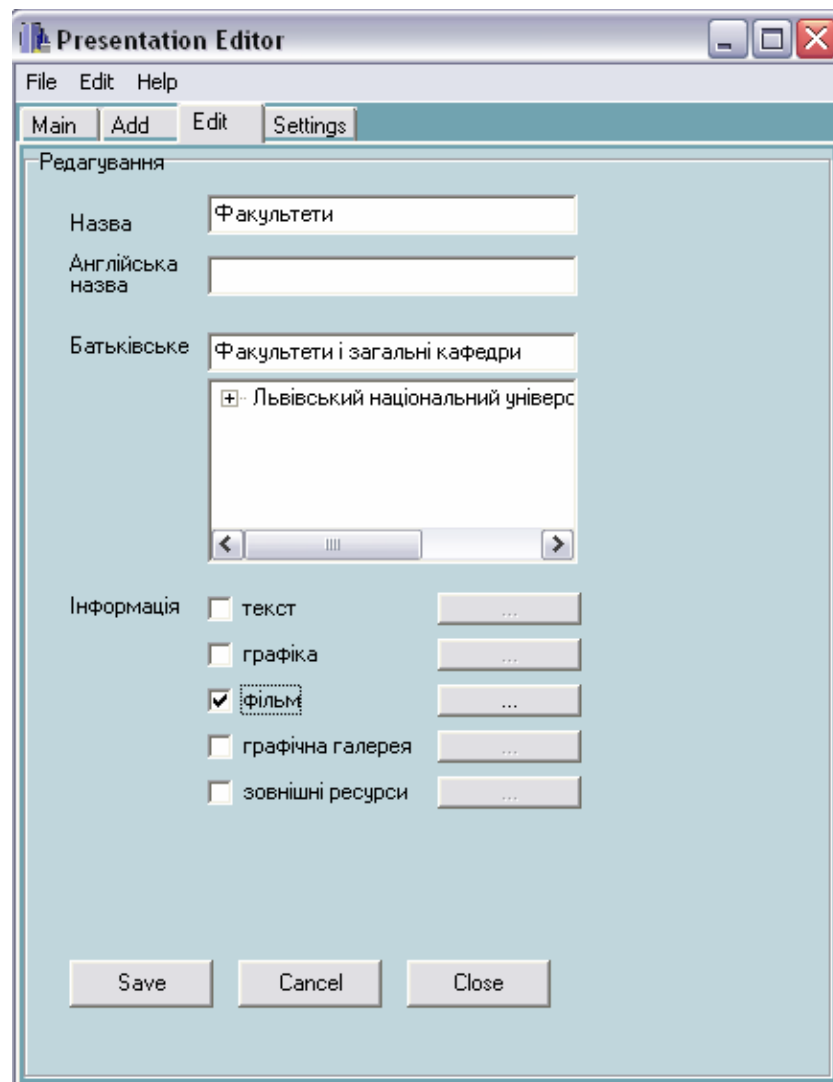


Рисунок 6.6 – Вікно редагування елемента структури

Кнопки *Add...* і *Edit...*, що знаходяться на головній закладці (рис. 6.3), призначені для додавання і редагування елементів структури меню. Після натискання на ці кнопки ми переходимо до закладки *Add* і *Edit* відповідно. Вигляд закладки *Edit* показано на рис. 6.6. Закладка *Add* виглядає аналогічно.

У вікні редагування є такі поля:

- *Назва* – відповідає українській назві кнопки.
- *Англійська назва* – відповідає англійській назві кнопки.
- *Батьківське* – у цьому полі відображається батьківське меню для даного. У поле нижче завантажується структура меню, де можна вибрати, клацнувши по ньому мишкою, батьківське меню.

- *Інформація* відображає той тип інформації, який відповідає даній кнопці. Такий вибір здійснюється за допомогою п'яти індикаторів: текст, графіка, фільм, графічна галерея, зовнішні ресурси. Можливий варіант, коли не відмічено жодний з індикаторів, тобто при натисканні на дану кнопку меню в презентації не буде відображатись інформація. Кожному з індикаторів відповідає своя кнопка редагування. Після натискання на кнопку, яка відповідає текстовому виду інформації, відкриється нове вікно під назвою *Текст*, яке зображено на рис. 6.7. Тут присутні дві закладки для *українського* і *англійського* тексту, на кожній з яких розміщено поля для заголовку і основного тексту. Передбачена можливість завантаження тексту з файлу. Це здійснюється за допомогою кнопки *Load...*, після натискання на яку відкривається вікно *Текст*, яке дозволяє завантажити будь-який текстовий файл. Після натискання на кнопки, які відповідають індикаторам: графіка, фільм, графічна галерея, завантаження зовнішніх ресурсів відкривається нове вікно для завантаження відповідної інформації. Загальний вигляд таких вікон зображено на рис. 6.5. Реалізована можливість вибору і завантаження інформації, яка буде відповідати даному елементу структури меню.

Кнопки *Save* (рис. 6.6), що знаходяться на закладках *Add* і *Edit*, зберігають створений або відредагований елемент структури. За допомогою кнопок *Cancel* можна перейти на головну закладку без внесення змін у вибраний елемент.

Кнопка *Save*, що знаходиться на головній закладці (рис. 6.3), зчитує інформацію з бази даних і закладки параметрів презентації і записує її певним чином в текстовий файл *base.txt*.

Кнопки *Close* на головній закладці (рис. 6.3) і закладках *Add* і *Edit* (рис. 6.6) закривають оболонку.

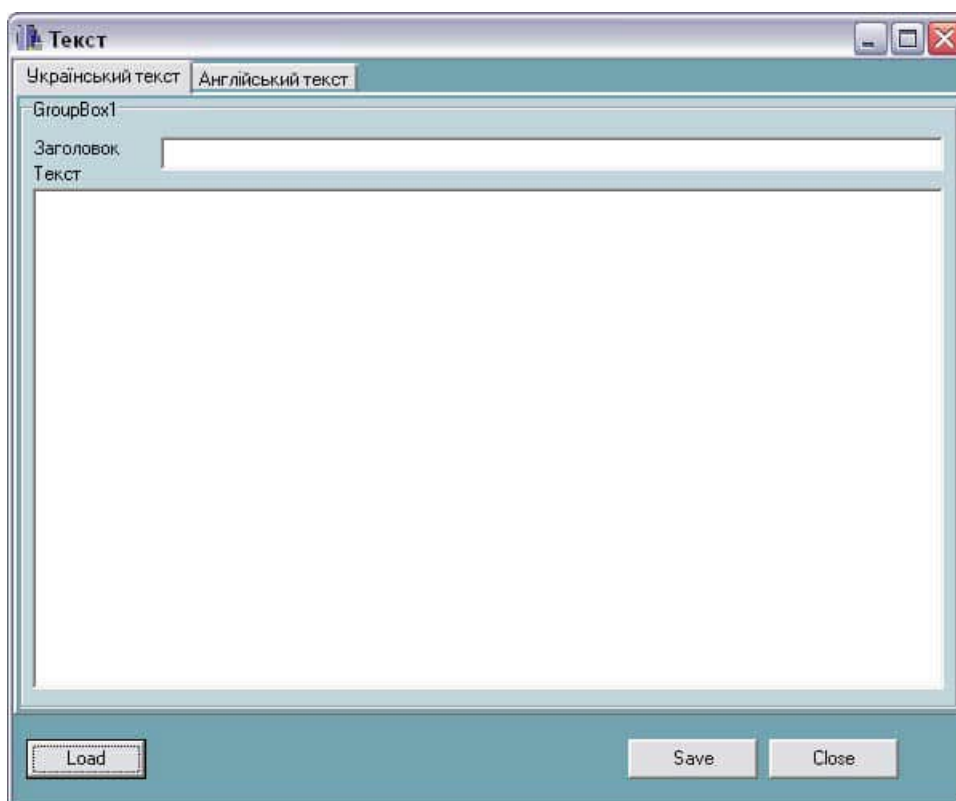


Рисунок 6.7 – Вікно редагування текстової інформації

6.3. Опис презентаційних матеріалів

Презентація завантажується з файлу *presentation.exe* (рис. 6.8). Для перегляду презентації на комп'ютері необхідно встановити Macromedia Flash Player.

На заставці презентації користувачу пропонується вибрати мову, після чого завантажується початкова сторінка. В залежності від параметрів презентації користувацьке меню може бути розміщене з лівого боку (рис. 6.9 б,г) або головне меню – зверху, решта – зліва (рис. 6.9 а,в). Пункти меню змінюються залежно від «місця знаходження» користувача (рухатись можна як вперед, так і назад). Реалізовано систему кнопок *назад*, які дозволяють перейти на один рівень вище відповідно. В лівому нижньому кутку вікна знаходиться кнопка для переходу в презентацію на іншій мові.



Рисунок 6.8 – Заставка презентації

Львівський національний університет імені Івана Франка

Загальна інформація

Факультети і загальні кафедри

Навчальні підрозділи

Науково-дослідні підрозділи

Громадські організації

Міжнародні співробітництво

Довідкова інформація

Львівський національний університет імені Івана Франка

Загальна інформація

- Історія університету
- Видатні особистості
- Адміністрація університету
- Відеофільми
- Університет на карті міста

Історія університету

Історія створення Львівського університету бере свій початок у XVII ст., але вона має і цікаву передісторію, що сягає корінням у глибину віку.

У XVI–XVII ст. центрами культурного життя на українських землях були церковні братства. Користуючись підтримкою міщан та духовенства, вони сприяли поширенню ідей гуманізму, розвитку науки і шкільництва. Найдавнішим в Україні було Успенське ставропігійське братство у Львові, яке стало визначним українським культурним центром. З 1586 р. у Львові діяла братська школа, яка була середнім навчальним закладом. Тут вивчалися церковно-слов'янська, грецька, латинська і польська мови, математика, граматика, риторика, астрономія, філософія та інші дисципліни. Члени Львівського братства планували навіть перетворити свій "гімнасійон" (так вони називали цю школу) у вищий навчальний заклад. У Львівській братській школі працювали і здобули освіту визначні діячі української культури кінця XVI–першої половини XVII ст.: Лаврентій Зизаній (Кукуль) і його брат Степан, Кирило Ставропільський, Іван Борещкий та ін.

До середини XVII ст. в Україні не було жодного вищого навчального закладу. Шляхетська Польща чинила опір створенню тут вищої школи, яка могла б стати небезпечним політичним і культурним центром. Українська молодь змушена була здобувати вищу освіту в інших Краківського чи інших європейських університетів.

Згідно зі статтями Гадяцької угоди (1658 р.) між Україною та Річчю Посполитою польський уряд обіцяв у майбутньому відкрити в Україні дві вищі школи-академії: одну в Києві, а другу там, де знайдеться для неї відповідне місце. Академіям було обіцяно ті самі права університету, якими користувався Краківський університет. Впливові кола Речі Посполитої не виключали й того, що під тиском певних політичних обставин в Україні могли утворитися власні національні університети. Тоді ж свідомий орден у справі захисту католицизму в Україні покладав особливі надії на свій осередок у Львові. Єзуїти з'явилися у Львові ще наприкінці XVI ст., а в 1608 р. відкрили тут свою середню школу-колегію. До середини XVII ст. ця колегія занепала, але все ж була врятована єзуїтами від загибелі, оскільки

а.

Львівський національний університет імені Івана Франка

Додаткова інформація

- Телефонні дзвінки
- Фотоальбом про університет
- Фотоальбом про Львів**
- Фотоальбом про Львівську область
- новини

Фотоальбом про Львів




6.

Львівський національний університет імені Івана Франка

- Загальна інформація
- Факультети і загальні кафедри
- Навчальні підрозділи
- Науково-дослідні підрозділи
- Промислові організації
- Міжнародне співробітництво
- Додаткова інформація

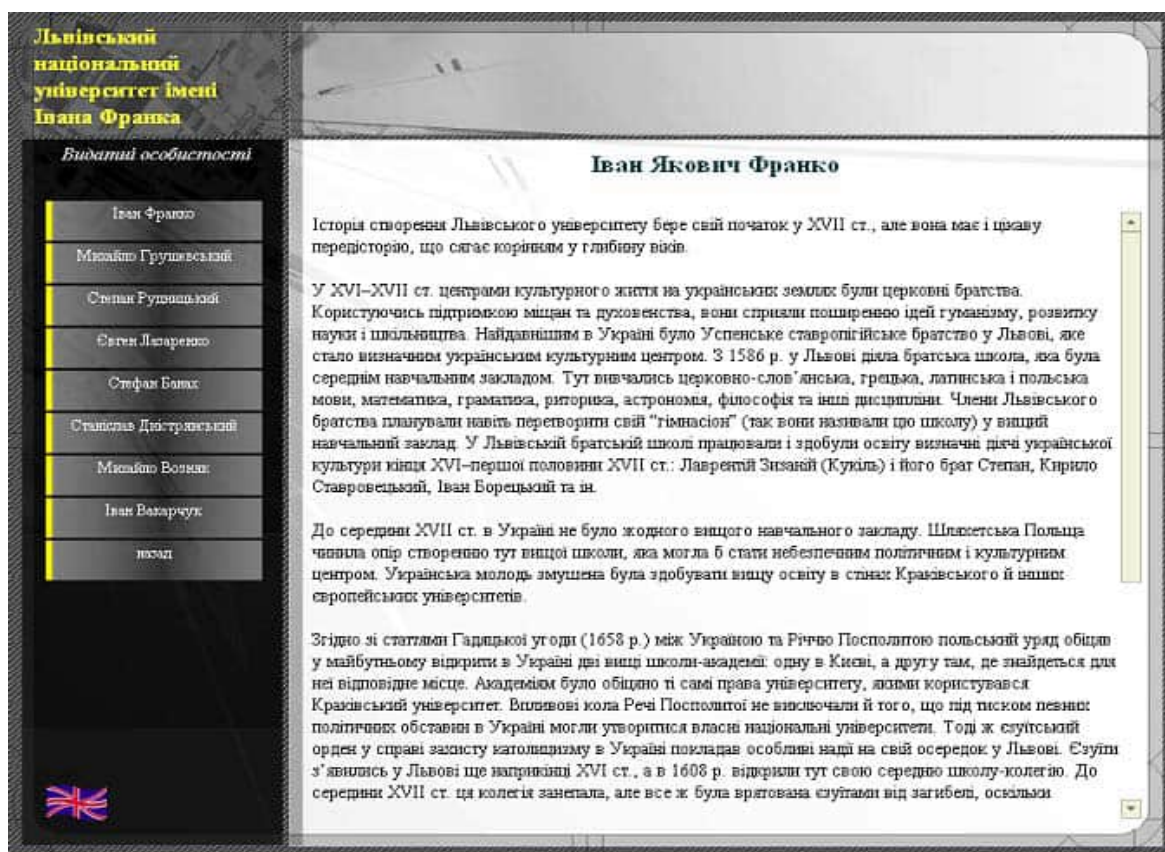
Додаткова інформація

- Телефонні дзвінки
- Фотоальбом про університет
- Фотоальбом про Львів
- Фотоальбом про Львівську область**

Фотоальбом про Львівську область




B.



г.

Рисунок 6.9 – Стилї оформлення презентації

Рисунок 9 в – демонструє сторінку «Фотоальбом про Львівську область» з завантаженою на неї графічною інформацією. Якщо розмір графіки або фільму не відповідає розмірам виділеного поля, тоді картинка масштабується під задані параметри.

6.4. Висновки

Таким чином, розроблено дві програми, одна – для створення структури для нових презентаційних матеріалів або для редагування вже існуючої презентації, друга – для генерації презентаційних матеріалів на основі структури даних, побудованих за допомогою першої.

Оскільки, презентаційний проект та інформаційні дані є незалежними один від одного, це дає можливість просто редагувати і оновлювати інформацію. Робота оболонки організована таким чином, що, навіть, недосвідчений користувач на інтуїтивному рівні зможе легко і просто нею користуватись. Функціональність оболонки дозволяє: створювати нову логічну схему даних, на базі якої будуть генеруватися презентаційні матеріали; редагувати вже створену структуру даних; змінювати налаштування параметрів презентації; завантажувати і редагувати інформацію різного плану: текст, зображення, відеоролики, зовнішні ресурси. В основі роботи презентації – логічна схема даних, на базі якої реалізовано систему меню і відображення

різнопланової інформації. Зручний інтерфейс дозволяє легко рухатись сторінками презентації.

Такий підхід до автоматичної генерації презентаційних матеріалів дозволяє звичайному користувачу створювати, редагувати і, що особливо важливо, актуалізувати презентацію, дизайн якої створений фахівцями.

7. РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ З ВИКОРИСТАННЯМ ПАТТЕРНОГО ПРОЕКТУВАННЯ

Аналіз розвитку різноманітних засобів побудови великих порталів чи веб-сайтів дозволяє констатувати, що дедалі більшу частку ринку займають системи управління «контентом» (інформаційним наповненням сайту). Проте останні існують в обмеженій кількості та придатні лише для невеликого кола завдань і проблем.

Важливість розробки системи управління контентом полягає у тому, що створення веб-сторінок стає все громіздкішим процесом, який вимагає певної автоматизації і використання готових продуктів і систем. Саме тому предметом для дослідження та створення стала система управління контентом як функціональний продукт.

У даному розділі представлено спробу розробки системи управління контентом на базі паттерна проектування MVC (Model-View-Controller), за допомогою якого можна вирішити одну з ключових проблем – відділення даних від їх візуального представлення. Засобами створення системи є мова програмування PHP, сервер баз даних MySQL.

Як приклад такої системи обрано своєрідний електронний архів видань або ж повноцінне Інтернет-видання газети (журналу) з швидким доступом до певного номера, сторінки та з реалізованими властивостями будь-якого динамічного сайту: доповнення інформації, легкість пошуку, функції адміністрування тощо.

7.1. Вимоги до системи

Основне завдання розробки можна розділити на такі блоки:

– *створення та проектування структури програми. Реалізація шаблону проектування MVC засобами мови PHP;*

Шаблон проектування MVC і об'єктно-орієнтовний підхід при проектуванні програмних продуктів надає багато переваг при розробці великих за об'ємом програмних продуктів. Основна його перевага в розділенні роботи програміста і дизайнера, програміста і іншого програміста, який буде доповнювати та оновлювати вже існуючу програмну розробку.

Тому програмний продукт повинен реалізовувати шаблон, що включає в себе набір класів трьох типів: Model, View, Controller, які будуть відповідати за свою частку функціональності системи, а також програмні засоби, які будуть дозволяти без зайвих труднощів дотримуватися шаблону і додавати нові функціональності і класи.

– *створення та проектування структури програми. Використання шаблону для створення системи управління контентом. Реалізація основної функціональності;*

Шаблон проектування MVC і об'єктно-орієнтовний підхід при проектуванні програмних продуктів надає багато переваг при розробці великих

за об'ємом програмних продуктів. Основна його перевага в розділенні роботи програміста і дизайнера, програміста і іншого програміста, який буде доповнювати та оновлювати вже існуючу програмну розробку.

Тому програмний продукт повинен реалізовувати шаблон, що включає в себе набір класів трьох типів: Model, View, Controller, які будуть відповідати за свою частку функціональності системи, а також програмні засоби, які будуть дозволяти без зайвих труднощів дотримуватися шаблону і додавати нові функціональності і класи.

– на основі системи управління контентом створення сайту з каталогом видань.

Всі системи управління повинні включати в себе деякі базові функціональності. В заданій системі, використовуючи шаблон MVC, повинні міститися розподіл користувачів за рівнем доступу до перегляду інформації чи її оновлення, система пошуку інформації, система управління вмістим сайту і можливість керувати виводом інформації на екран шляхом вибору певного шаблону.

Кожна система повинна адаптуватися для виконання певної роботи керування інформацією, яка, до того ж, може мати ще й різноманітну структуру: динамічну чи статичну. Тому завжди існує потреба вносити певні зміни до системи, щоби підлаштувати її під конкретну проблему, зокрема, створення каталогу видань.

Наперед не відомо, якою є структура каталога, відома тільки структура кінцевої одиниці перегляду – статті. Тому основна робота з адаптації системи до заданих вимог полягає в реалізації кінцевої одиниці, створенні шаблону перегляду, оновлення та доповнення інформації. При реалізації всіх функціональностей потрібно чітко дотримуватися об'єктно-орієнтовного підходу, правил шаблону проектування MVC, і реалізації шаблону засобами мови програмування PHP.

7.2. Паттерни проектування. Паттерн проектування MVC

Паттерни проектування стали невід'ємною частиною більшості середніх та великих програмних модулів. За допомогою шаблонів вирішується велика кількість різноманітних проблем, пов'язаних з проектуванням. Першим паттерни проектування почав застосовувати Christopher Alexander, який і дав визначення паттернам: «Паттерни проектування описують проблему, яка з'являється знову і знову, тоді описують основу для вирішення так, що ви можете використовувати готове рішення багато разів, а не двічі виконувати одну роботу».

Можна виділити такі основні властивості шаблонів:

- назва паттерна повинна відображати суть проблеми, бути легкою до запам'ятовування і асоціюватися з вирішенням;
- опис проблеми, яка є складною для вирішення, повинен містити всі деталі та приклади, де вона зустрічається;
- вирішення проблеми повинне містити набір бібліотек, класів, об'єктів;

- позитивний результат застосування паттерна, що виявляється у швидкості, зручності і застосовності паттерна.

Паттерн проектування Model-View-Controller (MVC) – один з перших шаблонів для вирішення проблем, який був реалізований на мові програмування SmallTalk і застосовувався для вирішення проблем у програмуванні графічного інтерфейсу в звичайних аплікаціях. Паттерн був розроблений основним чином для відділення даних від їх реалізації і для ефективного керування процесом роботи програм. Очевидною є незручність і неефективність розташування блоків коду, що відповідають за різну функціональність, у безпосередньому сусідстві. Особливо це відчутно при побудові масштабних, з великою кількістю інформації, порталів чи систем управління, які повинні чітко працювати з даними і без проблем при маніпулюванні ними. Паттерн MVC – трирівнева модель з такими частинами:

- Model;
- View;
- Controller.

На рис. 7.1 зображена структура паттерна із точки зору користувача.

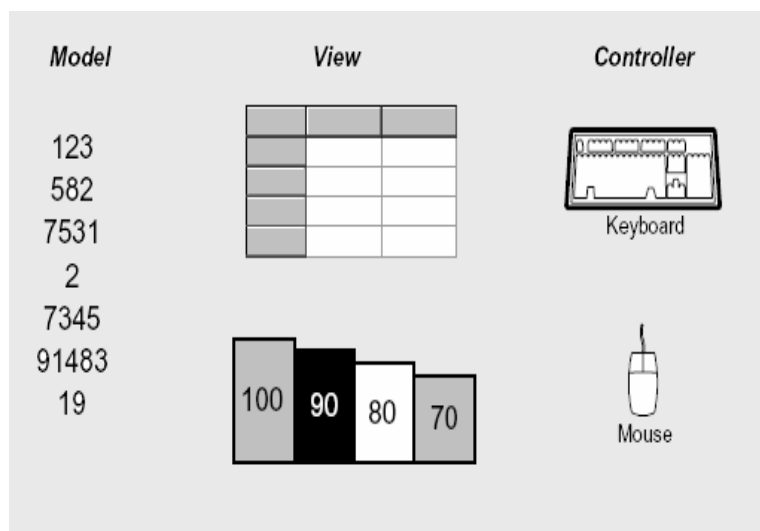


Рисунок 7.1 – Шаблон з точки зору користувача

Model – повинна містити дані і їх місце у системах управління базами даних, бізнес-логіку і інші програмні засоби, що не належать до засобів візуального представлення даних.

Controller – механізм, що управляє шляхами обміну інформацією між моделлю і відображенням даних. Включає засоби, які керують програмою залежно від введених даних, поставленим завданням чи виникненням помилок.

View – основні засоби для відображення даних на веб-сайті.

Схематично паттерн можна зобразити у вигляді діаграми:

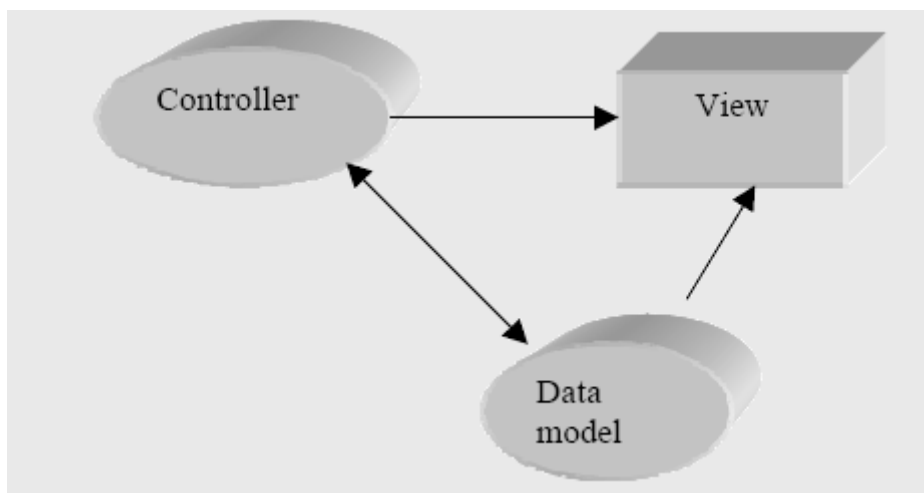


Рисунок 7.2 – Діаграма шаблону

Цей паттерн при проектуванні систем управління контентом відіграє суттєву роль для організації чіткого контролю над даними, відділення від реалізації, створення чіткого середовища для виявлення помилок, адже блок візуального представлення даних не залежить від їх оновлення і маніпулювання, і можливість оновлювати функціональність системи, не змінюючи дизайну сайту та не припиняючи його роботу.

7.3. Класифікація систем управління контентом

Всі системи управління контентом можна умовно поділити на дві категорії: статичні та динамічні.

Динамічні системи можна застосовувати до широкого ряду різноманітних за змістом задач. Такі системи, наприклад, PHPNuke, Plone є досить популярними, вони застосовуються для реалізації великих веб-проектів і є потужними механізмами, які керують діяльністю всього порталу. Вони також включають можливість використання різних серверів баз даних, в деяких випадках мають інтегровану власну базу даних, власну динамічну мову, інсталяцію нових продуктів і можливість повного керування сайтом відповідальною особою, яка може не володіти жодною з веб-технологій, проте з успіхом створювати складні статичні сторінки.

Статичні системи виконують ряд завдань щодо оновлення інформації, наприклад, у форумах, бібліотеках. Загалом, великі динамічні системи складаються з менших, окремо функціонуючих статичних систем управління, які незалежно виконують поставлену перед ними задачу. Система повинна надавати доступ до контенту тільки користувачам з певними правами, і дозволяти виконувати їм дії відповідно до наданих прав. Дані не повинні залежати від їх реалізації, тобто повинні існувати засоби, які за посередництвом технологій зберігання інформації (сервери баз даних, XML), надають надавати можливість переглядати дані.

Основною причиною використання і розробки систем управління контентом є великий масштаб веб-сайту, для якого система має значно спростити весь процес керування інформацією. Програмістів, що займаються

розробкою таких систем цікавить інше питання: як без значних затрат виявляти помилки, додавати нові функціональності, не турбуючись про налагодження зв'язків з іншими структурами чи блоками програми. Розширення проекту, яке відбувається за умови його первинного успіху, викликає появу нових ідей, необхідність додавання нових можливостей та засобів, які не може охопити один програміст.

7.4. Об'єктно-орієнтований підхід та РНР-реалізація MVC

Об'єктно-орієнтовний підхід до програмування великих систем надає суттєві переваги для підтримки порталу, сприяє виявленню помилок, спрощує додавання нових функціональностей, полегшує взаємодію програмістів, що відповідають за окремі структурні і функціональні блоки та з точністю показує реальну структуру проекту. Таким чином відбувається відображення структури програми та інформації у термінах класів, типів, відбувається точна класифікація об'єктів, їх можливостей і засобів маніпулювання ними. При проектуванні системи управління контентом об'єктно-орієнтовний підхід є особливо важливий, оскільки за допомогою типізації чітко прослідковується структура інформації, керування та додавання нових функціональностей. Слід зазначити, що паттерне проектування відбувається в термінах класів і типів, оскільки це найкраще підходить в плані застосовності паттерна. Паттерн проектування MVC, реалізований в термінах класів стає зручним інструментом для створення великих функціональних блоків і систем.

Таким чином, найкраще шаблони проектування реалізовувати використовуючи об'єктно-орієнтований підхід. Реалізація шаблону ділиться на чотири функціональні блоки:

- класи Model;
- класи View;
- класи Controller;
- скрипт GlobalController.

Три класи відповідають трьом частинам шаблону. Проте існує необхідність ввести ще один функціональний блок, який пов'язаний з обмеженістю мови програмування РНР. Це – скрипт GlobalController, що здійснює взаємодію класів View і Controller.

На рис. 7.3 схематично зображена взаємодія між класами і реалізація шаблону.

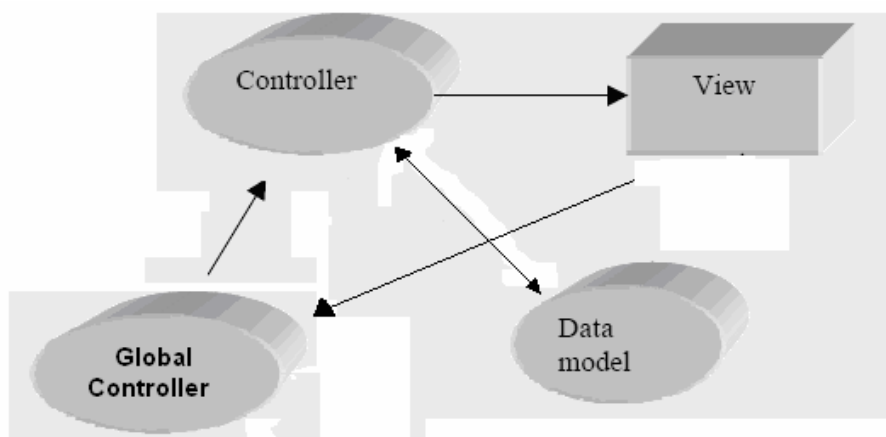


Рисунок 7.3 – Реалізація шаблону – Взаємодія класів.

7.4.1. Реалізація класу Model

Клас моделі повністю відображає всю інформацію про певний тип даних. Найчастіше він повністю копіює структуру бази даних. Клас повинен включати методи, що дозволяють повністю створити об'єкт зі всіма властивостями за унікальним значенням. Одному класу моделі відповідає один тип даних. Повинні бути передбачені можливості редагування, видалення, оновлення даних. Модель не містить елементів дизайну, не здійснює ніякого контролю над виконанням програми. Це просто структура даних, і набір функцій, за допомогою яких здійснюється керування даними. Модель даних включає в себе засоби, за допомогою яких можна здійснити вибірку з бази даних всієї інформації про даний тип, знаючи тільки один параметр. Це може бути унікальний ідентифікатор чи деяке інше унікальне значення. Також в моделі містяться функції для маніпулювання даними, такі як вставка, видалення, оновлення. Всі класи також містяться в каталозі Model. Дані класи використовують класи Controller.

7.4.2. Реалізація класу Controller

Класи Controller встановлюють певні правила:

- клас Controller приймає дані тільки з однієї форми.
- для однієї функціональності з форми, за котру відповідає один елемент розмітки HTML, тег Submit, існує одна функція в класі Controller.
- клас Controller працює тільки з моделлю даних і не містить в собі елементів дизайну.
- кінцевою метою класу Controller є обробка даних, зміна чи оновлення інформації та перенаправлення користувача на сторінку представлення даних.

Клас Controller є розширенням базового класу, який приймає певні параметри, за допомогою яких можна створити екземпляр моделі, з якою відбувається дія. У класі містяться функції, кожна з яких виконує певну дію над даними, які вона отримала від користувача. Кінцевим результатом роботи функції є виклик певного класу View, що відповідає за вивід інформації.

7.4.3. Реалізація класу View

Кожна веб-сторінка містить у собі візуальне представлення деяких даних за допомогою HTML-розмітки. Проте іноді виникає потреба в обробці даних, що надходять від користувача, чи конкретизації відображення різної за структурою інформації. Для такого випадку існує таке поняття як форма. Веб-дизайнер створює форму, яка містить у собі різноманітні поля заповнення, а програміст пише програму, яка обробляє введені дані. Досить часто на одній сторінці існує декілька форм. В таких випадках виникає потреба винесення обробітників форм в окремий файл, який і буде заданий в параметрах форми як обробітник. Проте виникає потреба веб-дизайнеру синхронізувати свою роботу з програмістом у плані назв обробітників та подальших дій після виконання програми. Для такого випадку дизайнер повинен знати наперед, який обробітник задавати для певної форми. Тому доцільно для такого випадку встановити ряд правил:

- обробітник форми, вказаний у параметрах форми, завжди повинен мати одну назву.
- кожна форма включає в себе поля, що містять інформацію, необхідну програмісту (ця інформація повинна носити сталий характер).
- дизайн міститься у відповідному класі і файлі.
- маніпулює з даними форми єдиний клас – Controller, котрий приймає всю введену інформацію за посередництва скрипта GlobalController і який і буде вказаний у параметрах форми як обробітник.

У даному випадку відбувається створення класу відображення, який реалізовує функцію, що обов'язкова для всіх класів даного типу – view(). Дана функція містить в собі елементи веб-дизайну і обов'язкові поля, що вказують, який клас Controller використовується і над яким екземпляром моделі працює користувач. Обробітник форми, вказаний у параметрах форми завжди незмінний. Його назва – GlobalController

7.4.4. Реалізація функціональності GlobalController

Даний скрипт є параметром, що вказується у кожній формі. Його основне призначення у перерозподілі запитів між класами View і Controller. Скрипт приймає дані з форми, що реалізована класом View, обов'язковими атрибутами якої є назва класу потрібного обробітника і унікальні дані моделі. Після цього він створює екземпляр класу даного обробітника і викликає потрібну функцію. Одним зі складових елементів скрипта є масив елементів Submit і назв функцій, що використовуються скриптом для вибору правильної назви функції класу Controller, яка обробляє запит. Масив містить в якості ключа назву елементів Submit, а в якості значення – назву функції. Приймаючи значення елементів Submit, GlobalController викликає функцію, назву якої отримує з масиву. Необхідність такого підходу зумовлена кириличними назвами елементів форми. В іншому разі можна давати назву функціям таку саму як і назва елементів Submit.

7.5. Реалізація системи управління контентом

7.5.1. Структура даних

Основною метою системи управління у плані структурованого підходу до організації даних є можливість будувати різноманітні ієрархії, не турбуючись про будь-які обмеження. Таким чином можна структурувати інформацію в будь-якому форматі. Інформація зберігається у двох виглядах: на диску та в базі даних (каталоги та документи). Відповідно, каталоги можна зберігати в інших каталогах, таким чином створювати ієрархії даних. Кінцевою точкою ієрархії є документ. База даних повністю відображає всю структуру інформації, містить дані про батьківські та дочірні елементи, а також додаткові властивості одиниці інформації, що не можуть бути відображені фізично на диску без допомоги додаткових файлів чи функціональностей, що працюють з операційною системою.

База даних містить усю необхідну інформацію, за допомогою якої можна відтворити ієрархію даних. База поділена на таблиці, кожна з яких відповідає за певний тип інформації. Це каталоги, файли та користувачі. В цих таблицях міститься необхідна інформація та додаткова, що включає властивості, які доповнюють уявлення про даний тип об'єкту. Основна інформація повинна повністю відображати суть структури і об'єктів, що збережені фізично на диску, і при потребі відновлювати структуру даних.

Основним і початковим каталогом є каталог під назвою «catalog». В ньому зберігається вся інформація у вигляді каталогів та файлів. Кінцевою точкою ієрархії є файл. Папка може містити велику кількість каталогів чи файлів. В каталозі може міститися один файл в форматі .vfi . В цьому файлі містяться дані, які будуть відображатися на сайті, коли користувач зайде на відповідний рівень ієрархії, за який відповідає даний каталог. Якщо такого файлу немає, тоді користувач буде бачити список файлів, що містить дана папка.

На рис. 7.4 показана деяка структура даних, що відображається у дереві каталогів. Дерево генерується шляхом рекурсивного перегляду програмою всіх каталогів і пошуку дочірніх елементів.

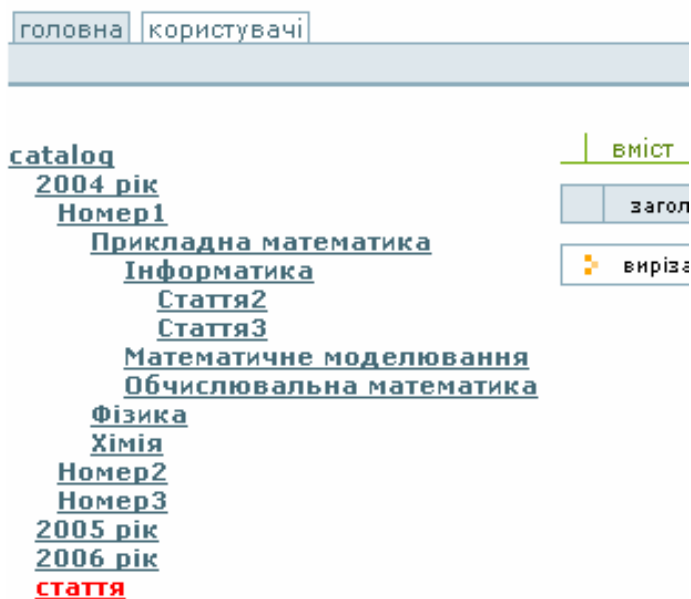


Рисунок 7.4 – Реалізація дерева каталогів та документів

7.5.2. Доповнення та оновлення даних. Шаблони перегляду даних

Оскільки веб-сайт містить інформацію, яка може динамічно змінюватися, то потрібно передбачити функціональність зміни і доповнення інформації.

Функціональність для зміни інформації про каталоги:

- видалити
- копіювати
- вирізати
- додати папку
- додати файл
- властивості:
 - зробити папку видимою для певної категорії користувачів
 - чи відображати вміст папки для певної категорії користувачів.
 - шлях до файлу, що буде відображати дана папка (.vfi файл).
- редагувати додаткові параметри і властивості папки

Для адміністратора система виведе всі файли, що містяться в каталозі. Також з даним каталогом адміністратор може виконувати дії, які приведені у списку. На рис. 7.5 зображена сторінка адміністрування каталогу.

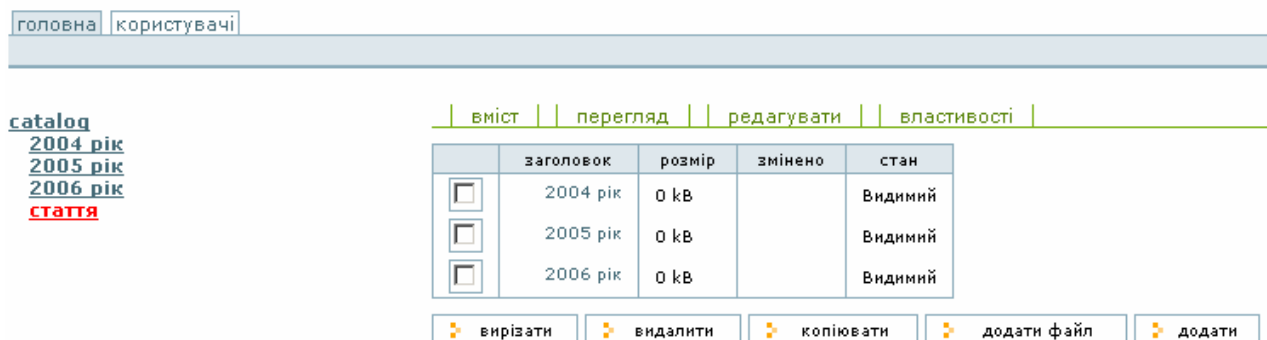


Рисунок 7.5 – Сторінка адміністрування каталогу

Функціональність для зміни інформації про файли:

- вирізати
- видалити
- копіювати
- редагувати властивості
- змінювати файл

Кожен файл міститься у певному каталозі. Принцип редагування файлу такий самий, як і каталогу. Для кожного файлу існує сторінка адміністрування. Файл можна перемістити або копіювати в іншу папку. На рис. 7.7 показано сторінку редагування властивостей файлу. У даному випадку файлом виступає стаття. Стаття містить додаткову інформацію, яка зберігається в базі даних.

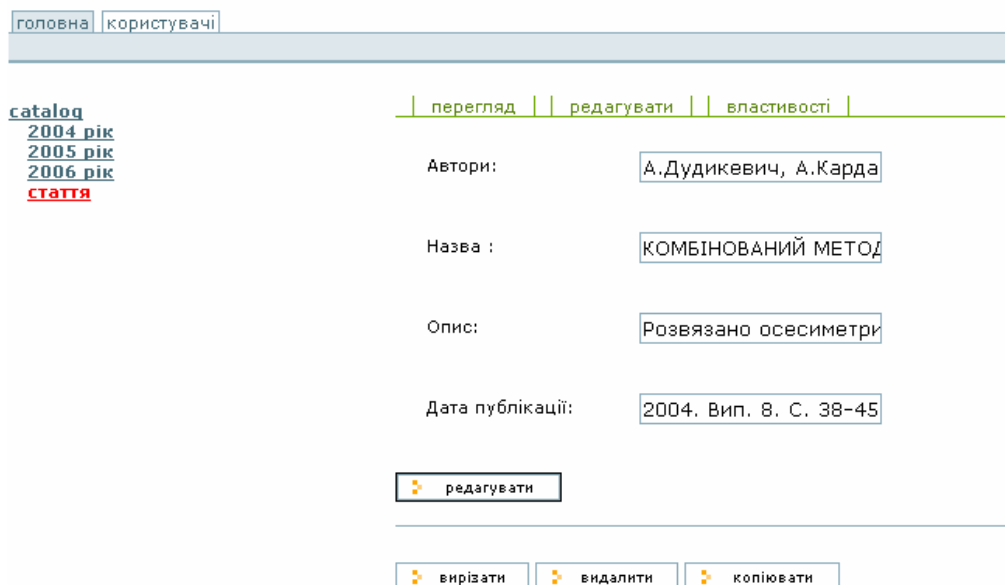


Рисунок 7.7 – Сторінка редагування властивостей статті.

Функціональність для зміни інформації про користувачів:

- додати користувача
- надати права доступу до зміни інформації
- видалити користувача

Основною вимогою і характеристикою систем управління контентом являється строгий поділ користувачів у їх доступі до інформації і обмеження прав на редагування та зміну інформації. Виділимо такі три типи користувачів з їх правами доступу:

1. *адміністратор.*

Здійснює редагування та доповнення інформації. Отримує всі права на редагування інформації, а також надає права доступу іншим користувачам.

2. *зарєєстрований користувач.*

Отримує доступ на перегляд каталогів та файлів, дозвіл на завантаження файлів.

3. *анонімний користувач.*

Доступ тільки до перегляду даних.

При завантаженні сайту кожен користувач за замовчуванням анонімний. Для входу на сайт зарєєстрованих користувачів існує форма логування. Після логування система за унікальними значеннями, такі як логін і пароль, розпізнає користувача та його права. Якщо користувач має всі права доступу до редагування інформації, то завантажується сторінка адміністрування каталогу. Для реєстрації користувачів існує форма реєстрації. Після реєстрації користувач може залогуватися і використовувати ті права доступу, які надані йому адміністраторами. На рис. 7.8 і рис. 7.9 показані відповідно форми логування та реєстрації.

вхід приєднатися

Реєстраційна форма

Особисті дані

Повне ім'я
Введіть повне ім'я, наприклад Максим Кривоніс

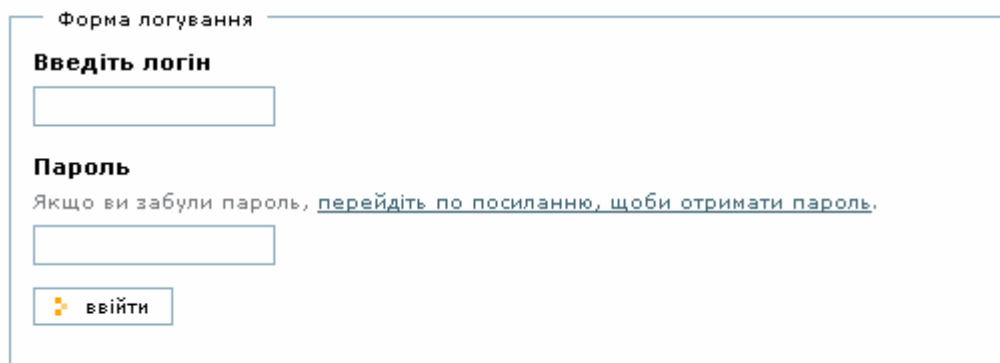
Псевдонім
Введіть псевдонім користувача, по-бажанню, звичайно щось на зразок 'mkryvonis'. Латинкою, без пропусків чи спеціальних символів. Псевдоніми та паролі є чутливі до регістру, переконайтеся, що Caps Lock вимкнено. Це буде псевдонім, яким ви користуватиметеся для входу в систему.

Електронна пошта
Введіть вашу електронну адресу. Це необхідно у разі, якщо ви забудете пароль.

Пароль
Введіть пароль. Мінімум 5 символів.

Підтвердіть пароль
Повторно введіть пароль. Впевніться, що вони ідентичні.

Рисунок 7.8 – Реєстраційна форма



Форма логуювання

Введіть логін

Пароль

Якщо ви забули пароль, [перейдіть по посиланню, щоби отримати пароль.](#)


 **ВВІЙТИ**

Рисунок 7.9 – Сторінка логуювання користувачів

7.6. Створення веб-сайту періодичних видань на основі створеної системи управління контентом

Сайт періодичних видань містить статті видання, які структуровані в певному порядку. Відомо, що статті діляться за роками, номерами та розділами. Стаття може бути як в номері, так і спеціальна, окрема від номера стаття. Всі дані статті містяться в базі даних, а сама стаття міститься на диску. Відповідно, статті містяться у папках, які відповідають тому рівню ієрархії, до якого належить стаття.

Сама стаття доступна для завантаження користувачам. Поділ користувачів відбувається за рівнем доступу до файлу статті. Анонімний користувач має право тільки на перегляд інформації, що стосується даної статті. Завантажити і прочитати статтю може користувач, що попередньо зареєструвався на сайті періодичного видання. Оскільки побудова сайту відбувається з використанням системи управління контентом, то стають очевидні переваги використання останньої, скільки вже реалізована вся функціональність щодо редагування інформації. Проте систему можна розширювати і модифікувати, тому до файлу додалися наступні властивості статті:

- автори;
- анотація ;
- дата публікації.

7.7. Висновки

Таким чином, у даному розділі зроблена спроба дослідити системи управління контентом, їхню роль в розробці великих і середніх веб-сайтів, дослідити характеристики систем та класифікувати їх за типом завдань і проблем, які вони вирішують. Створена повноцінна система управління контентом, розроблена на базі шаблону проектування MVC засобами мови програмування PHP. Реалізовані всі функціональні властивості, які притаманні

системам управління контентом. Розроблена система використана при побудові сайту періодичних видань, створення якого не зайняло багато часу і який отримав разом з системою всі її переваги.

8. СИСТЕМА АВТОМАТИЗОВАНОГО ГЕНЕРУВАННЯ ПЕРСОНАЛЬНИХ ВЕБ-СТОРИНОК

На тлі бурхливого розвитку Інтернету дуже важливим є подання якомога ширшого представлення необхідної інформації у всесвітній мережі. При цьому коло людей, що можуть кваліфіковано створити персональну веб-сторінку є обмеженим а їх послуги є не безкоштовними.

Частковим розв'язанням цієї проблеми є запропонована веб-система, що дозволяє автоматично, за певним шаблоном, генерувати сторінки для представлення інформації про працівників університету, дозволяє внесення, редагування, пошук інформації та її відображення з дотриманням єдиного шаблону. Система забезпечує зберігання інформації в базі даних для її подальшого використання, модифікації та оновлення.

8.1. Технології реалізації системи

Реалізувати дану систему можна за допомогою багатьох різних Веб-технологій, таких як ASP, JSP, CGI, ISAPI, Perl та багато інших.

Вибір здійснено на користь таких засобів, як препроцесор гіпертексту PHP та сервер баз даних PostgreSQL, що володіють рядом переваг над конкурентами.

До переваг PHP відносяться :

- Висока продуктивність.
- Наявність інтерфейсів до різних систем баз даних. Використовуючи Open Database Connectivity Standart (ODBC), можна підключатися до будь-якої бази даних, для якої існує ODBC-драйвер.
- Вбудовані бібліотеки для виконання багатьох загальних задач, пов'язаних з Веб.
- Простота вивчення і використання. Синтаксис PHP базується на інших мовах програмування, в першу чергу на C і Perl.
- Здатність до легкого перенесення на різні платформи. Пакет PHP можна використовувати під керуванням багатьох різних операційних систем.
- Доступність вихідного коду.

PostgreSQL теж володіє багатьма перевагами, в тому числі високою продуктивністю, низькою вартістю, простотою конфігурування і вивчення, легкістю перенесення і доступністю вихідного коду.

8.2. Структура проекту

Проект можна розділити на структурні елементи, які різняться для користувачів та адміністраторів

Для адміністратора реалізовані:

- внесення даних з контролем коректності
- пошук наявних сторінок, що представлений у двох варіантах:
 1. скорочений вигляд (пошук за частиною імені)

2. розширений пошук (дозволяє здійснювати пошук за багатьма полями одночасно)

- модифікація створених записів
- перегляд наявних зареєстрованих сторінок
- можливість підвантажувати дані з файлів
- робота із словниками посад працівників університету, вчених звань та вчених ступенів
- робота із деревом структури університету для внесення нових гілок структури або редагування існуючих
Для користувачів реалізовано
- пошук у короткому та розширеному вигляді без можливості модифікації
- можливість перегляду уже зареєстрованих сторінок

8.2.1. Шаблон проектування

Система побудована згідно шаблону проектування MVC (Model View Controller), тобто підтримується ідеологія, згідно якої збереження даних та логіка роботи з ними є відділеними від їх представлення та інтерфейсу їх зміни, що надає можливість легшого супроводу системи, її розвитку і розбудови.

Згідно згаданого шаблону система поділяється на такі компоненти:

1. MODEL: база даних то логіка роботи з даними.
2. CONTROLLER: адміністративна частина: інтерфейс, що надає доступ до пошуку, модифікації, адміністрування прав доступу до даних.
3. VIEW: представлення даних – згенеровані сторінки.

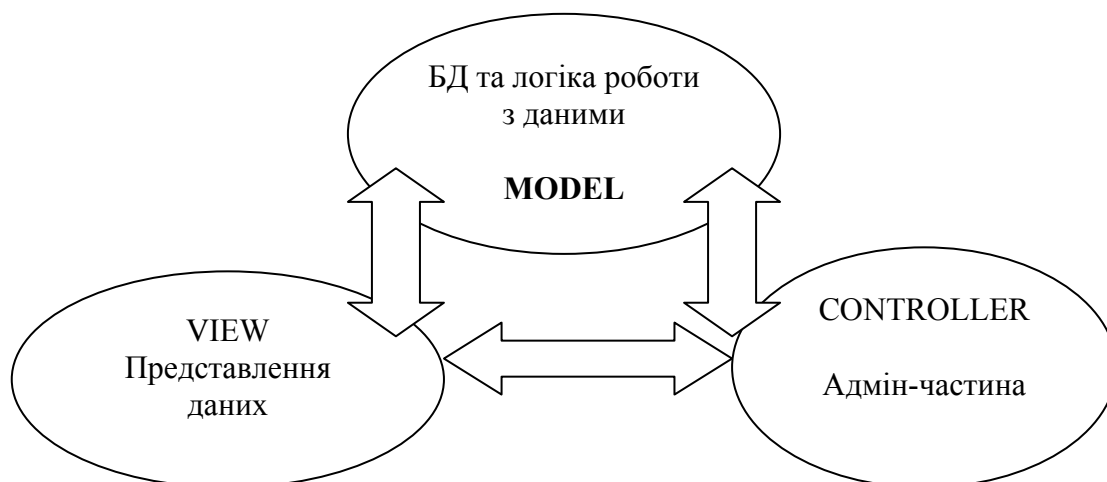


Рисунок 8.1 – Схема шаблону проектування MVC

8.2.2. Модель даних

Модель даних включає два структурні елементи:

1. База даних, в якій зберігається вся необхідна інформація про працівників, навчальні курси, допоміжна інформація про вчені звання, ступені, посади, а також інформація про модераторів та рівень їх доступу до даних.

2. Ієрархія класів, що реалізують логіку роботи з даними. Значною мірою дерево наслідування класів нагадує структуру таблиць бази даних. Для всіх головних таблиць створено відповідні класи.

8.2.3. Структура бази даних

Таблиці з даними можна розділити на такі логічні групи.

Головна таблиця:

- Personnel

Таблиці-словники:

- Positions
- AcademicDegrees
- AcademicStatuses
- AddAcademicDegrees
- AddAcademicStatuses
- ContactTypes

Таблиці-зв'язки:

- PersonnelContacts
- PersonnelAddAcademicDegrees
- PersonnelAddAcademicStatuses
- PersonnelCourses

Інші таблиці, що використовуються:

- Courses
- UniStructure
- Translation_String
- Translation_Text

Таблиці з інформацією про модераторів:

- Moderators

Взаємозв'язки між таблицями можна побачити на рис. 8.2.

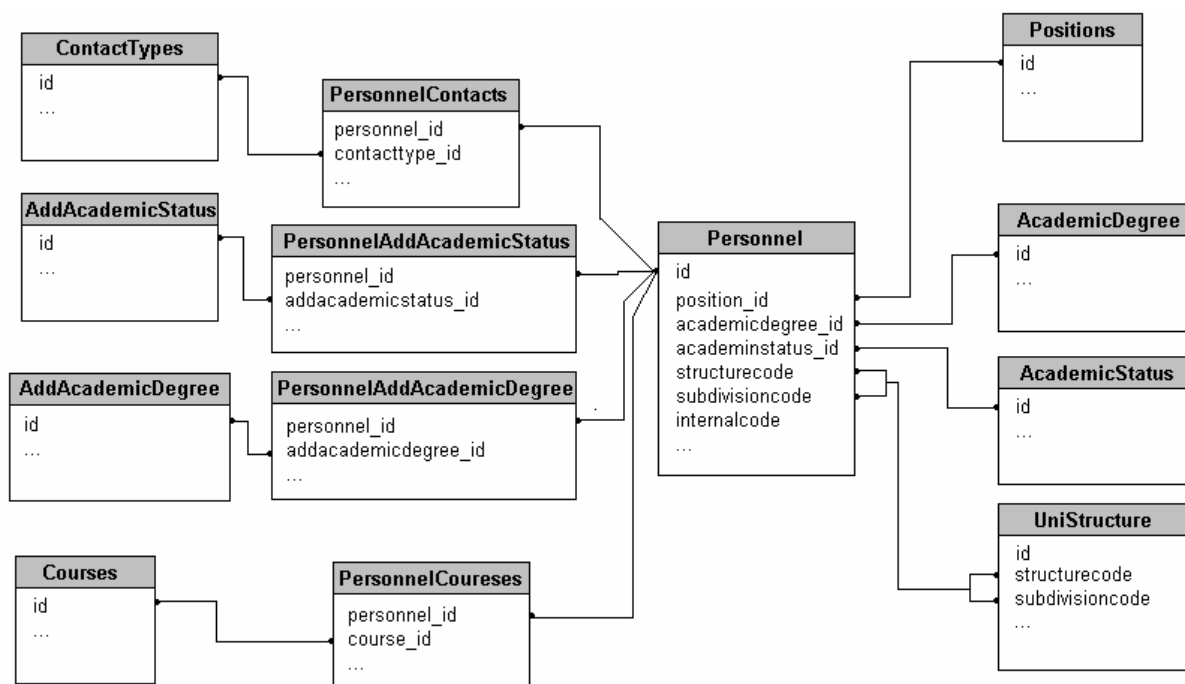


Рисунок 8.2 – Зв'язки між таблицями

Детальніша інформація про структуру таблиць подана у наступних таблицях.

Таблиця 8.1. Таблиця **PERSONNEL**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ працівника	ID	SERIAL	
Ім'я	FNAME_STRING	CHAR(15)	Ключ з таблиці TRANSLATION_STRING
Прізвище	LNAME_STRING	CHAR(15)	Ключ з таблиці TRANSLATION_STRING
По-батькові	MNAME_STRING	CHAR(15)	Ключ з таблиці TRANSLATION_STRING
Посада	POSITION_ID	INTEGER	Ключ з таблиці POSITIONS
Вчена ступінь	ACADEMICDEGREE_ID	INTEGER	Ключ з таблиці ACADEMICDEGREES
Вчене звання	ACADEMICSTATUS_ID	INTEGER	Ключ з таблиці ACADEMICSTATUSES
Фото *	PHOTO	CHAR(30)	
Дата народження *	BIRTHDAY	DATE	
Згода на публікування дати народження	PRINTBIRTHDAY	CHAR(1)	'Y' або 'N'
Біографія	BIOGRAPHY_TEXT	CHAR(20)	Ключ з таблиці TRANSLATION_TEXT

Згода на публікування біографії	PRINTBIOGRAPHY	CHAR(1)	‘Y’ або ‘N’
Наукові інтереси	SCIENTIFICINTERESTS_TEXT	CHAR(20)	Ключ з таблиці TRANSLATION_TEXT
Наукові публікації	PUBLICATIONS_TEXT	CHAR(20)	Ключ з таблиці TRANSLATION_TEXT
Код структури	STRUCTURECODE	CHAR(2)	
Код підрозділу	SUBDIVISIONCODE	CHAR(2)	
Внутрішній код особи	INTERNALCODE	CHAR(2)	

Таблиця 8.2 – Таблиця POSITIONS

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.3 – Таблиця ACADEMICDEGREES

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.4 – Таблиця ADDACADEMICDEGREES

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.5 – Таблиця ACADEMICSTATUS

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.6 – Таблиця ADDACADEMICSTATUS

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.7 – Таблиця COURSES

Зміст	Поле таблиці	Тип поля	Примітки
Ключ посади	ID	SERIAL	

Назва посади	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING
--------------	-------------	----------	--------------------------------------

Таблиця 8.8 – Таблиця **CONTACTTYPES**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING

Таблиця 8.9 – Таблиця **PERSONNELCONTACTS**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Ключ	PERSONNEL_ID	INTEGER	Ключ з таблиці PERSONNEL
Ключ	CONTACTTYPE_ID	INTEGER	Ключ з таблиці CONTACTTYPES

Таблиця 8.10 – Таблиця **PERSONNELADDACADEMICDEGREES**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Ключ	PERSONNEL_ID	INTEGER	Ключ з таблиці PERSONNEL
Ключ	ADDACADEMICDEGREE_ID	INTEGER	Ключ з таблиці ADDACADEMICDEGREES

Таблиця 8.11 – Таблиця **PERSONNELADDACADEMICSTATUSES**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Ключ	PERSONNEL_ID	INTEGER	Ключ з таблиці PERSONNEL
Ключ	ACADEMICSTATUS_ID	INTEGER	Ключ з таблиці CONTACTTYPES

Таблиця 8.12 – Таблиця **PERSONNELCOURSES**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Ключ	PERSONNEL_ID	INTEGER	Ключ з таблиці PERSONNEL
Ключ	COURSE_ID	INTEGER	Ключ з таблиці COURSES

Таблиця 8.13 – Таблиця **COURSES**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	

Назва	NAME_STRING	CHAR(20)	Ключ з таблиці TRANSLATION_STRING
-------	-------------	----------	--------------------------------------

Таблиця 8.14 – Таблиця **UNISTRUCT**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	US_ID	SERIAL	
Допоміжний номер	US_LEFT	INTEGER	
Допоміжний номер	US_RIGHT	INTEGER	
Допоміжний номер	US_LEVEL	INTEGER	
Назва	NAME_STRING	CHAR(30)	Ключ з таблиці TRANSLATION_STRING
Посилання	URL	CHAR(255)	
Код	STRUCTURECODE	CHAR(2)	
Код	SUBDIVISIONCODE	CHAR(2)	
Код	INTERNALCODE	CHAR(2)	

Таблиця 8.15 – Таблиця **TRANSLATION_STRING**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME	CHAR(20)	
Мова	LANGUAGE	CHAR(2)	
Значення	CONTENT	CHAR(255)	

Таблиця 8.16 – Таблиця **TRANSLATION_TEXT**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Назва	NAME	CHAR(20)	
Мова	LANGUAGE	CHAR(2)	
Значення	CONTENT	TEXT	

Таблиця 8.17 – Таблиця **MODERATORS**

Зміст	Поле таблиці	Тип поля	Примітки
Ключ	ID	SERIAL	
Логін	LOGIN	CHAR(100)	
Електронна адреса	EMAIL	CHAR(100)	
Пароль	PASS	CHAR(32)	
ХешКлюч	HASH	CHAR(32)	
Активність	ISACTIVE	CHAR(1)	
Рівень доступу	PERMITLEVEL	INTEGER	
Доступний код структури	PERMITSTRUCTURECODE	CHAR(2)	
Доступний код підрозділу	PERMITSUBDIVISIONCODE	CHAR(2)	
Доступний внутрішній код особи	PERMITINTERNALCODE	CHAR(2)	

8.2.4. Структура класів

Логіка роботи з даними, що пов'язані з інформаційним наповненням проекту зосереджена в ієрархії класів, що реалізовані мовою програмування РНР.

Серед головних функціональних елементів, що реалізовані в системі, слід виділити наступні:

- робота із з'єднаннями до бази даних;
- внесення та модифікація інформації у словники;
- внесення та модифікація інформації про працівників;
- робота з деревовидною структурою університету;
- реалізація модерування із гнучкою системою розподілу прав.

Структура взаємозв'язків між класами нагадує структуру самої бази даних, проте містить деякі допоміжні класи, що узагальнюють структурні елементи.

Ієрархія класів та їх зв'язки зображені на рис. 8.3.

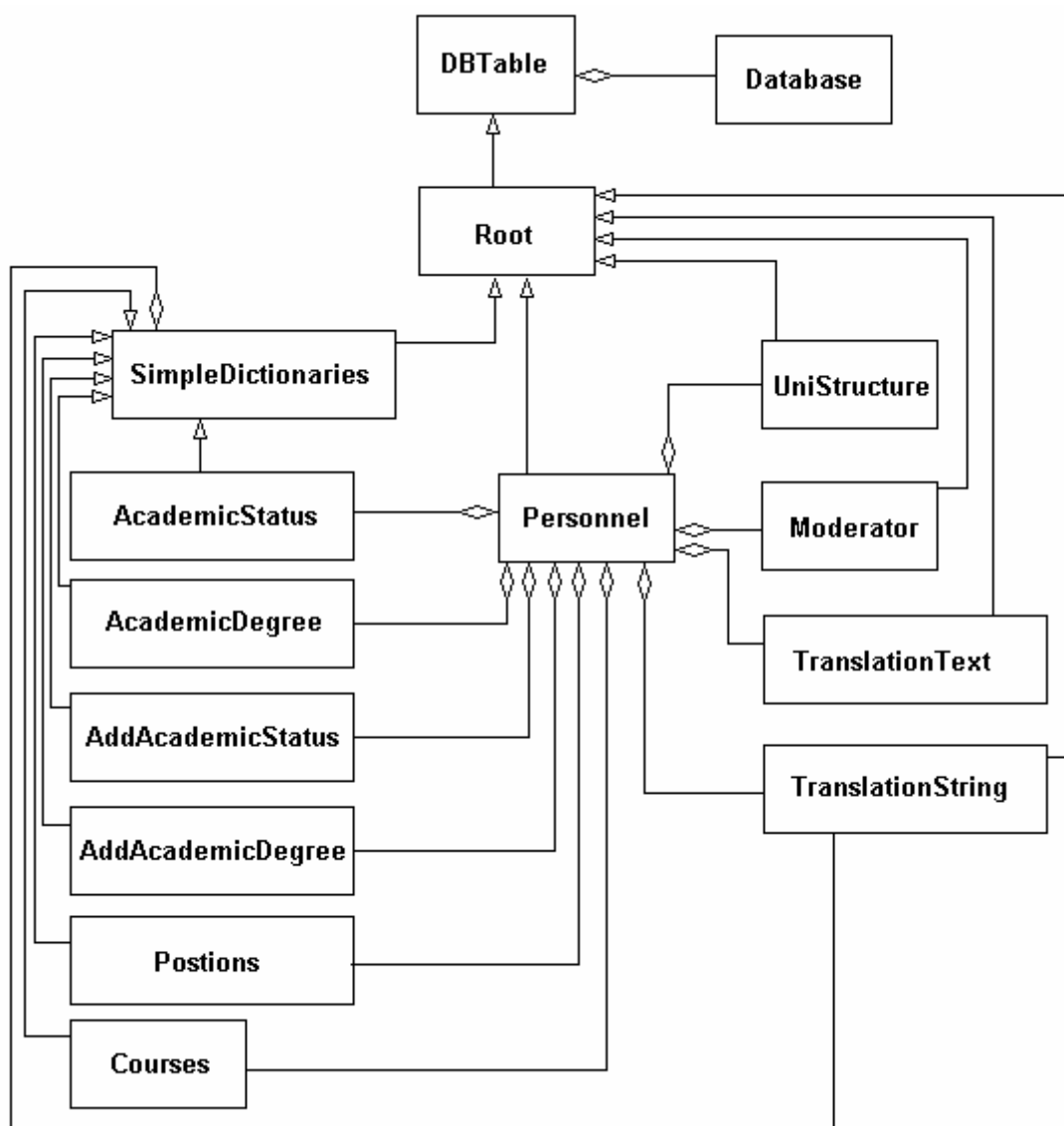


Рисунок 8.3 – Ієрархія класів

8.3. Модерування інформації. Рівні доступу до даних

В системі реалізовано ієрархічну систему модерування інформації. Після входу в систему визначаються права доступу до даних. Ці права асоційовані з певною гілкою у структурі університету і мають 4 рівні доступу:

1. Рівень 0: доступ до всієї інформації системи;
2. Рівень 1: доступ до інформації про працівників в рамках структурної одиниці університету, що відповідає першому рівню, наприклад, факультету.
3. Рівень 2: доступ до інформації про працівників в рамках певного підрозділу університету, наприклад, кафедри.
4. Рівень 3: доступ до інформації про одного працівника.

Кожен модератор належить до одного з цих чотирьох рівнів доступу. При цьому він може додати до переліку модераторів іншу людину, з правами доступу, які не перевищують його власні.

8.4. Контролери. Адміністративна частина

Адміністративна частина надає інтерфейс повного доступу до даних. Інтерфейс складається із заголовку, бокового меню, що містить посилання на основні сторінки для модифікації, та основної частини, яка містить контекстне меню в залежності від місця знаходження на сторінці.

Зовнішній вигляд панелі адміністрування представлено на рис. 8.4:

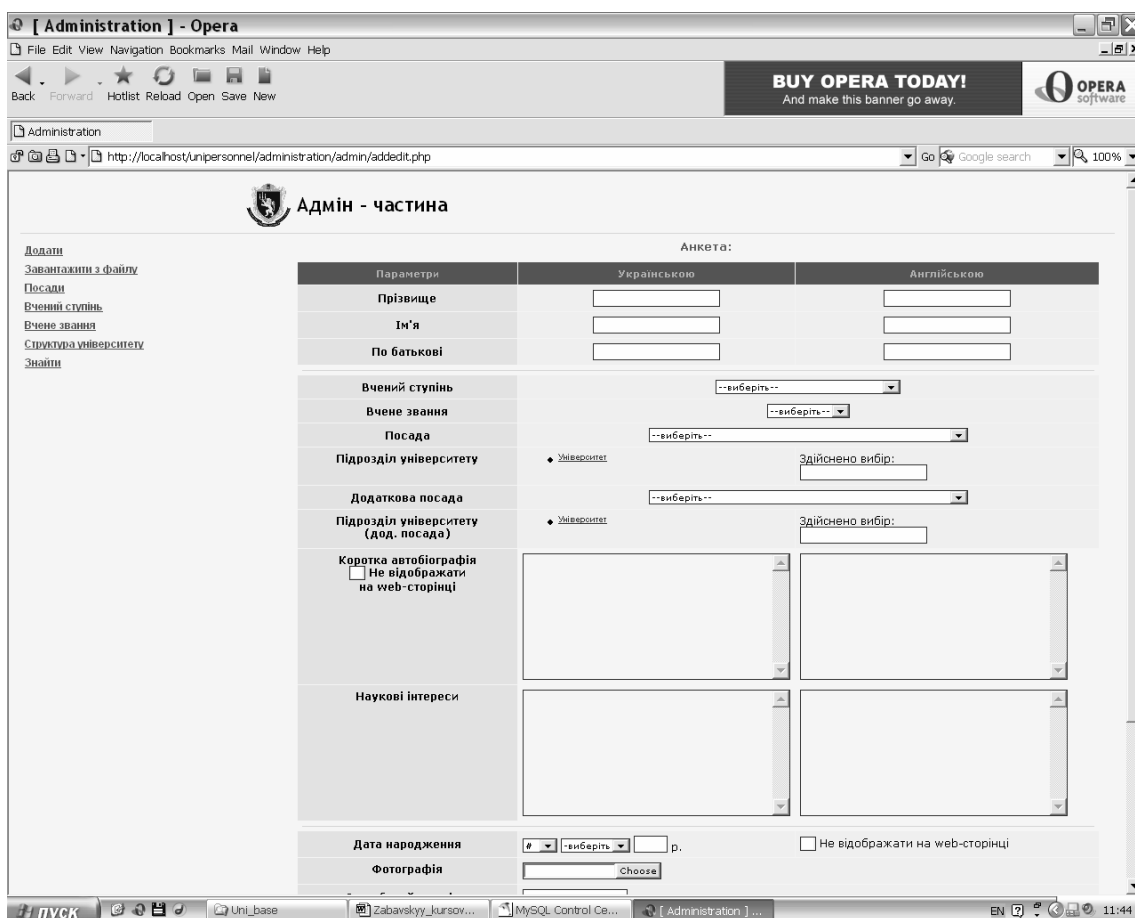


Рисунок 8.4 – Зовнішній вигляд панелі адміністрування

Блок меню «Модератор» дає можливість редагування інформації про модератора, а також додавання нового модератора з правами доступу меншими або рівними ніж власні. При додаванні нового модератора вказується електронна адреса, на яку система відсилає згенерований пароль, а також підрозділ університету, в межах якого новий адміністратор матиме змогу редагувати інформацію про працівників. При цьому кореневим вузлом дерева вибору підрозділу є підрозділ, за який відповідає поточний модератор, таким чином є неможливим надати комусь права більші ніж власні.

Після створення облікового запису новий модератор може увійти в систему використовуючи в якості логіна свою електронну адресу, та згенерований пароль.

Додавання нового модератора ілюструє рис. 8.5.

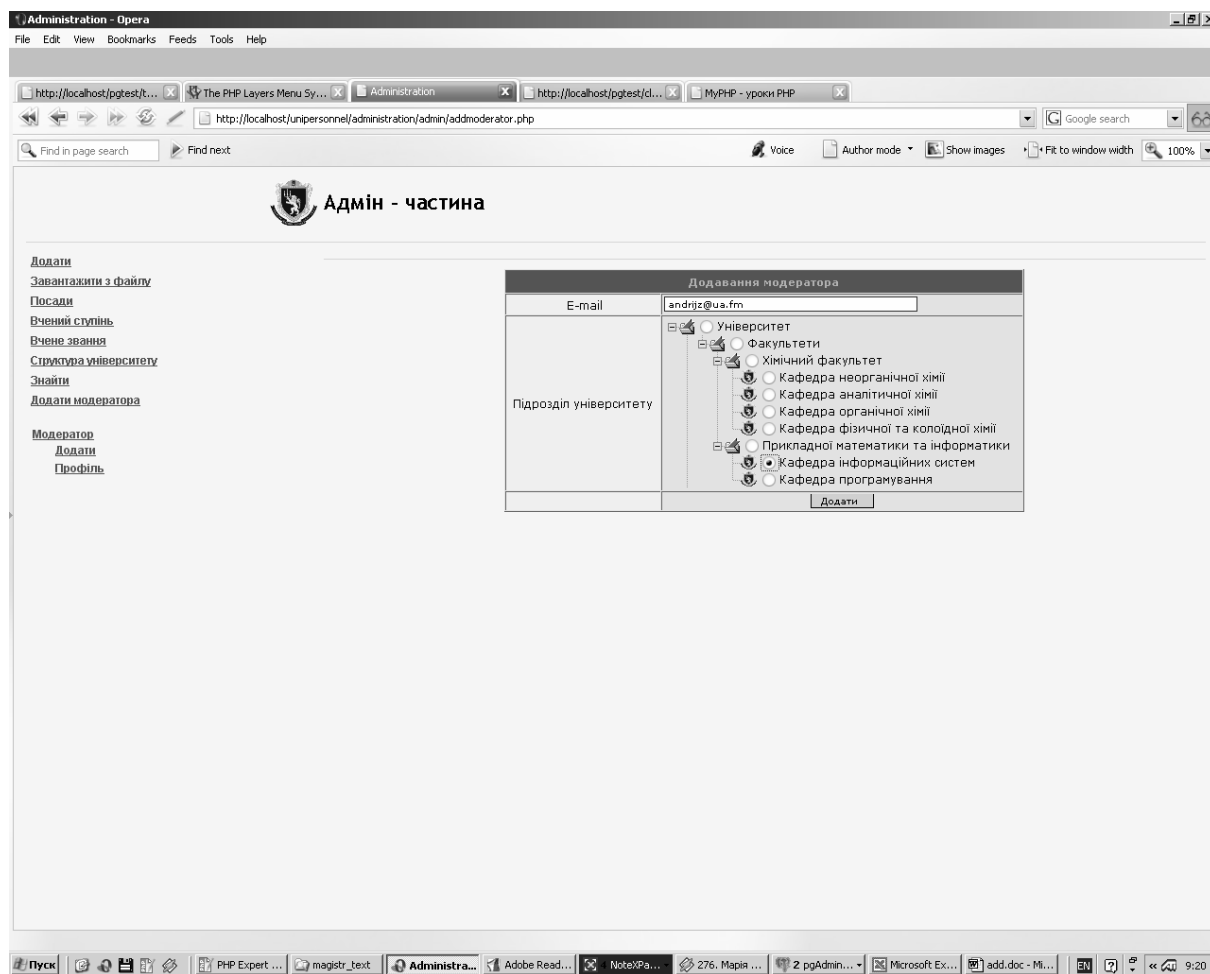


Рисунок 8.5 – Додавання модератора

8.5. Представлення результатів

Результатом роботи системи є персональна сторінка працівника університету. Сторінка відповідає вимогам щодо дизайну та інформаційного наповнення. Реалізована можливість зміни дизайну, що не прив'язаний до логіки роботи з даними. Для цього використано програмний продукт, що вільно поширюється для роботи з дизайном – бібліотека шаблонів Smarty, що дає

змогу зручно співпрацювати програмістам та дизайнерам. Зі сторони програмної частини на вхід сторінки передається масив з даними, а відображення цих даних повністю залежить від структури шаблону, який оформлюється згідно певних вимог щодо дизайну.

Приклад сторінки працівника університету, яку згенерувала система показано на рис. 8.6.



Рисунок 8.6 – Приклад згенерованої веб-сторінки

8.6. Інтерфейс системи

На головній сторінці системи розміщені форма “скороченого” пошуку для, що дозволяє здійснювати пошук за частиною імені або перейти на сторінку розширеного пошуку, а також розміщена коротка інформація про систему та інструкції користування.

Вхід на панель адміністратора розміщений за іншою адресою, що пояснюється іншим логічним призначенням цієї частини проекту. На головній сторінці адміністратора розміщено форму для логування. Після успішного входу розпочинається сеанс роботи з даними, параметрами якого є ідентифікатор модератора, що визначає права доступу до даних і є його невід’ємною частиною.

Під час роботи на панелі адміністрування постійно присутнє головне меню, що містить посилання на сторінки модерування усіх головних складових

системи. Сторінка за замовчуванням – сценарій внесення даних про працівника. Після внесення даних на першій сторінці, завантажується друга сторінка, яка перевіряє коректність внесених даних на першій сторінці, і при потребі виводить повідомлення з інформацією про помилку і прохання виправити її. Окрім цього є сторінки редагування інформації у словниках, інтерфейс для роботи з деревовидною структурою університету, а також сторінка додавання нових модераторів з вибором відповідних можливостей доступу до даних.

На кожній з вищеописаних сторінок адміністратора є форма скороченого пошуку, що дозволяє здійснювати пошук та отримувати можливість модифікувати наявні записи та створені сторінки.

8.7. Висновки

Реалізовано систему генерування веб-сторінок згідно вимог щодо дизайну та інформаційного наповнення. Система забезпечує надання прав модерування, що відповідає деревовидній структурі підрозділів університету.

ВИСНОВКИ

Результати проведених досліджень дозволяють:

- ✓ зменшити затрати на збір та оцифрування інформації;
- ✓ зменшити дублювання при зберіганні даних;
- ✓ публікувати інформацію у вигляді друкованих видань (каталогів, довідників, збірників наукових праць), у вигляді Інтернет-сторінок, а також мультимедійних інформаційно-довідкових та презентаційних каталогів на компакт-дисках;

Результати проекту:

- ✓ використані при підготовці нових навчальних курсів з сучасних комп'ютерних технологій, зокрема таких як проектування великих інформаційно-довідкових систем з використанням архітектури «клієнт-сервер», проектування систем обробки та візуалізації лексикографічної, картографічної, графічної та мультимедійної інформації;
- ✓ можуть знайти використання в інших учбових закладах для збору, актуалізації та представлення різномірної інформації у вигляді друкованих, мультимедійних та Інтернет-публікацій.

ПЕРЕЛІК ПОСИЛАНЬ

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., 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.
3. Веб-сайт університету: структура, дизайн та мінімальний перелік інформації // Стандарт Львівського університету СТП 1.04-2005.-Львів, 2005.-15 с.
4. Вовк В.Д. Застосування об'єктного підходу до топології розміщення даних у мережі // Вісник Львів. ун-ту. Сер. прикл. матем. та інформ., 2005.–Вип. 10.– С. 155-160.
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., Кревс В.Є., Пасічник В.Т., Шинкаренко О.Г. Система керування базою даних на платформі .NET з підтримкою структурованих поштових повідомлень // Сучасні проблеми прикладної математики та інформатики: Тези доп. XIII Всеукр. наук. конф. 3-5 жовтня 2006р., м.Львів.-Львів: Видавничий центр ЛНУ, 2006.- С. 43.
9. Горлач В.М., Левченко О.М. Організація збирання інформації для створення Веб-сайту університету // Вісник Львів. ун-ту. Серія прикл. мат. та інформ.-2006. Вип 11.- С. 217-224.
10. Інформатика. Комп'ютерна техніка. Комп'ютерні технології: Підручник. – К.: Каравела, 2003. – 464 с.
11. Розробка інструментальних засобів для формування друкованих, мультимедійних та Інтернет видань учбового закладу: Звіт про ДКР; держ. реєстр. № 0103U001925; держ. облік. № 0205U002159. – Львів, ЛНУ, 2004. – 43 с.

12. Розробка системи управління інформаційними масивами web-сайтів університету: Звіт про ДКР; держ. реєстр. № 0100U001428; держ. облік. № 0203U002140. – Львів, ЛНУ, 2002.– 61 с.
13. Створення програмного та інформаційного забезпечення для формування інформаційних компакт-дисків: Звіт про ДКР; держ. реєстр. № 0100U001429; держ. облік. № 0203U002130. – Львів, ЛНУ, 2002.– 18 с.
14. Тушницький Р. Методика створення технічного завдання на розробку автоматизованої інформаційної системи // IX Всеукр. студентська наукова конференція з прикладної математики та інформатики. Львів, 5-6 квітня 2006 року. м.Львів: ЛНУ, 2006. – С.106-107.

Електронні ресурси:

15. Веб-сайт «Львівський національний університет ім. І.Франка» (www.lnu.edu.ua).
16. Веб-сайт «Міжнародний науковий конгрес «Іван Франко: дух, наука, думка, воля»» (www.lnu.edu.ua/conference/franko150/).
17. Веб-сайт «Наукові видання Львівського університету» (<http://blues.lnu.edu.ua/publish/>).
18. Веб-сторінка інституту післядипломної освіти (www.lnu.edu.ua/institutes/ipk/).
19. Веб-сторінка підготовчого відділення для іноземних студентів (www.lnu.edu.ua/faculty/psis/).
20. Комплекти документів та інструкцій для збирання інформації про Університет (www.lnu.edu.ua/Geninf/Web-standard/documentation.html).
21. Веб-сайт Всеукр. конференції «Сучасні проблеми прикладної математики та інформатики» (www.sppmi.org.ua).