

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему:

“Розроблення підсистеми управління програмними проєктами АІС ІТ-компанії”

Виконав: студент групи 4147ст

Андрєєв В. С.



(підпис)

Керівник роботи:

викл. Бобровський О. С.

(посада, науковий ступінь вчене звання)



(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" " 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Андрєєву Вадиму Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: "Розроблення підсистеми управління програмними проектами АІС ІТ-компанії"

Керівник роботи викл. Бобровський О. С.

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проєкту (роботи) (ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем):
ДСТУ 3008:2015 – "Інформація та документація. Звіти у сфері науки і техніки"; ДСТУ ISO/IEC 15910:2012 "Інформаційні технології. Документація для програмного забезпечення"; ODOO: Управління проектами з відкритим кодом; Управління проектами в ODOO

4. Перелік питань, що належать до розробки (найменування розділів):

- Аналіз предметної області, розробка концепції автоматизованої системи та постановка задачі.
- Проект АРМ менеджера програмних проєктів АІС ІТ-компанії.
- Модель розгортання підсистеми АРМ менеджера проєктів.

5. Перелік презентаційних матеріалів

Структура управління проектами в проєктно-орієнтованій організації; Мета та задачі роботи; Модель кооперації існуючої системи ІТ-компанії; Структура інформаційного забезпечення процесів проєктування ПЗ; Концепції АРМ менеджера програмних проєктів у складі АІС ІТ-компанії; Модель прецедентів АРМ менеджера в частині адміністрування проєктів; Логічна модель даних АРМ менеджера (фрагмент); Структура програмних модулів та інтерфейс АРМ; Висновки по роботі

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	викл. Бобровський О. С.	03.02.2025	22.02.2025
2	викл. Бобровський О. С.	22.02.2025	21.03.2025
3	викл. Бобровський О. С.	21.03.2025	25.04.2025
4	викл. Бобровський О. С.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

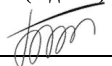
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Андреев В. С.
(ПІБ)

Керівник роботи


(підпис)

Бобровський О. С.
(ПІБ)

Анотація

Робота присвячена розробленню підсистеми управління програмними проєктами АІС ІТ-компанії. В роботі зроблено акцент на інформаційному забезпеченні процесів розробки та супроводу програмного забезпечення, управління програмними проєктами з можливістю використання вільного програмного забезпечення.

В роботі виконано аналіз предметної області, розроблено концепцію автоматизованого робочого місця керівника програмних проєктів, виконано постановку задачі, розроблені проєктні рішення та документація по підсистемі.

Робота викладена на ____ сторінках друкованого тексту, містить ____ рисунків, ____ таблиць та ____ додатків.

Ключові слова: корпоративна інформаційна система; управління проєктами; інформаційне забезпечення проєкту; база даних.

Abstract

The work is devoted to the development of a software project management subsystem AIS of an IT company. The work focuses on information support for the processes of software development and maintenance, software project management with the possibility of using free software.

The work includes an analysis of the subject area, the concept of an automated workplace for a software project manager, the formulation of the task, and the development of design solutions and documentation for the subsystem.

The work is presented on ____ pages of printed text, contains ____ figures, ____ tables and ____ appendices.

Keywords: corporate information system; project management; project information support; database.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ.....	4
Вступ.....	5
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, РОЗРОБКА КОНЦЕПЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ.....	8
1.1. Аналіз діяльності ІТ-компанії.....	8
1.1.1. Організація проєктного управління в ІТ-компанії.....	8
1.1.2. Організаційна робота менеджера проєктів ІТ-компанії.....	12
1.2. Визначення проблем діяльності ІТ-компанії.....	16
1.3. Аналіз предметної області.....	17
1.3.1. Термінологія, визначення та технології.....	17
1.3.2. Аналіз існуючих рішень та технологій інформаційної підтримки проєктів.....	22
1.3.3. Порівняльна характеристика технологій документування.....	26
1.4. Розробка концепції автоматизованої системи управління програмними проєктами.....	27
1.5. Функціонально-вартісний аналіз.....	30
1.5.1. Функціональний аналіз.....	30
1.5.2. Вартісний аналіз.....	32
1.6. Постановка задачі на створення підсистеми управління програмними проєктами ІТ-компанії.....	33
1.7. Висновки до розділу 1.....	33
2. ПРОЕКТ АРМ МЕНЕДЖЕРА ПРОГРАМНИХ ПРОЕКТІВ АІС ІТ- КОМПАНІЇ.....	35
2.1. Загальносистемні рішення.....	35
2.2. Рішення з інформаційного забезпечення.....	39
2.3. Рішення з програмного забезпечення.....	46
2.4. Рішення з технічного забезпечення.....	56
2.5. Рішення з організаційного забезпечення.....	57
2.6. Висновки до другого розділу.....	58

3. МОДЕЛЬ РОЗГОРТАННЯ ПІДСИСТЕМИ АРМ МЕНЕДЖЕРА ПРОЄКТІВ	59
Загальні висновки по роботі.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ.....	63
А.1. Призначення розробки.....	63
А.2. Вимоги до автоматизованої системи.....	63
А.3. Вимоги до функціональності системи.....	63
А.4. Вимоги до надійності й безпеки.....	64
А.5. Умови експлуатації.....	64
А.6. Вимоги до технічних засобів.....	65
А.7. Вимоги до інформаційної й програмної сумісності.....	65
А.8. Вимоги до програмної документації.....	65
А.9. Стадії й етапи розробки.....	65
А.10. Порядок контролю та приймання.....	66
ДОДАТОК Б. Код програмного забезпечення підсистеми (фрагмент)	
ДОДАТОК В. Інструкція з експлуатації та супроводу автоматизованої інформаційної системи управління програмними проєктами ІТ-компанії	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

AIC – автоматизована інформаційна система

APM – автоматизоване робоче місце

IC – інформаційна система

IT – інформаційні технології

ПЗ – програмне забезпечення

СКБД – система керування базами даних

ORM – (object relation mapper) система об'єктного відображення
реляційної бази даних

SVN – (subversion) система управління версіями

UML – уніфікована мова моделювання

Вступ

При вирішенні проблемних завдань, пов'язаних із переорієнтацією цілей організації або зміною шляхів їх досягнення, найбільш привабливою формою організаційної структури є проєктна команда. Це сформовані групи, в яких виконавці закріплені за проєктом на період його життєвого циклу і повністю підпорядковані менеджеру проєкту. При такій організації управління досягається тісний взаємозв'язок і взаємодія між виконавцями, високий рівень відповідальності, планування і контролю, легкість в управлінні конфліктами.

Для вирішення завдань перспективного розвитку в складі проєктної організації організації створюється спеціальний підрозділ, який займається винятково питаннями стратегії, а керівники проєктів зосереджують свою увагу на виконанні конкретних завдань

Управління проєктами – область знань з планування, організації та управління ресурсами з метою успішного досягнення цілей та завершення завдань проєкту. Іноді ототожнюється з управлінням програмами, але програма – це фактично більш високий рівень: група пов'язаних та взаємозалежних проєктів.

Головним завданням проєктного управління є досягнення всіх цілей та виконання завдань проєкту, одночасно виконуючи зобов'язання щодо наперед визначених обмежень проєкту. Типовими обмеженнями є межі та зміст проєкту, час, бюджет. Другорядним завданням, але більш амбіційнішим, є оптимізація, розподілення та інтеграція виконавців і завдань, необхідних для досягнення наперед визначених цілей.

Системи управління проєктами ІТ-компанії призначені для автоматизації бізнес-процесів з розробки, впровадження та підтримки автономного програмного забезпечення, програмного забезпечення в складі автоматизованих інформаційних систем.

ІТ-компанії під час розробки програмного забезпечення звичайно з самого початку свого існування приділяють значну увагу автоматизації власних бізнес-процесів з розробки та подальшого супроводу ПЗ. Цей інтерес

призводить до розробки окремих програмних продуктів та систем управління проєктами. Основні функції таких систем забезпечують ведення різних видів обліку проєктів, команд і проєктних груп; призначення та контролю завдань розробникам тощо. Такі системи дозволяють вести контроль статусу виконання робіт з проєктів, облік та ведення статистики помилок у проєктах; і контроль виконання робіт з усунення помилок; надання розробникам інформації про ті завдання, які їм призначені для розробки, з урахуванням пріоритетності їх виконання; надання менеджеру оперативної інформації про те, над чим в даний момент працюють розробники та рівень завантаженості розробників; облік, формування та аналіз статистики відпрацьованого часу в розрізі розробника і проєкту; збірку проєктів та установку на цільовий сервер.

Особливість управління проєктами в ІТ-компаніях полягає в тому, що розробка програмного забезпечення проводиться в межах робочих планів та програм, а супровід та виправлення помилок, як правило – по за межами цих планів. Виконання зазначених робіт передбачає використання різних інструментальних засобів при розробці проєктів різної складності – від завдань на декілька годин роботи (виправлення нескладних помилок) до завдань, що можуть виконуватись протягом етапу реалізації проєкту. Таким чином зазначені роботи можуть виконуватись безперервно або зі значними перервами. Це значить, що між цими роботами мають зберігатись отримані проміжні результати і надаватись дружнє середовище для виконання таких робіт.

Крім того, у проєктній структурі для вирішення конкретного завдання створюється спеціальна робоча група, яка після завершення роботи над проєктом розпускається. При цьому, відповідний персонал і ресурси, раніше притягнуті до роботи, повертаються у свої підрозділи, але інформаційне забезпечення проєкту, представлене, у тому числі, знаннями учасників проєкту, має зберігатись і бути доступним після розпуску команди проєкту. Разом це складає серйозну проблему інформаційного забезпечення проєктів.

Метою роботи, з врахуванням вищенаведеного, є підвищення ефективності операційної діяльності менеджменту і пересічних працівників ІТ-компанії за рахунок автоматизації процесів інформаційного супроводу програмних проєктів та управління проєктами ІТ-компанії — включаючи розробку технічної та експлуатаційної документації по проєктах, рекомендацій з оперативної діяльності виконавців проєктів, підтримку та оновлення інструментального програмного забезпечення, ведення журналів виконання робіт тощо.

Для досягнення цієї мети потрібно провести детальний аналіз процесів управління програмними проєктами та підтримки інструментальних програмних середовищ, складу, розробки та завантаження проєктних документів, розробити постановку задачі на здійснити проєктування автоматизованої системи, створити необхідне програмне забезпечення на основі результатів аналізу та проєктування, розробити інші рішення по АРМ адміністратора автоматизованої системи.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, РОЗРОБКА КОНЦЕПЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз діяльності ІТ-компанії

1.1.1. Організація проектного управління в ІТ-компанії

Інформаційна система управління проектами (project management information system, PMIS) – це логічна організація інформації, необхідної організації для успішного виконання проектів. PMIS зазвичай являє собою один або кілька програмних застосунків та методичний процес збору та використання інформації про проекти. Ці автоматизовані інформаційні системи допомагають планувати, виконувати та закривати цілі управління проектами [11]. Системи PMIS відрізняються за обсягом, дизайном та функціями залежно від операційних вимог організації.

Класичний підхід до управління ІТ, як допоміжними технологіями і послугами в поточній ринковій ситуації з урахуванням динаміки її розвитку (зміна бізнес-моделі на тлі глобальної фінансової кризи) є ризикованим, як в фінансовому, так і в іміджевому плані [1]. Розвиток ринкової ситуації вимагає впровадження сучасних моделей управління ІТ, що забезпечують:

- сервісну орієнтованість ІТ-діяльності;
- прозорість ІТ-послуг для бізнесу у фінансовому та технологічному плані;
- можливість надання основних типів послуг «на вимогу»;
- можливість оперативного надання нових послуг, оперативного переналаштування наявних сервісів;
- задоволення реальних потреб бізнесу в ІТ-послугах та ІТ-рішеннях;
- можливість управління співвідношенням «ціна / якість» при жорсткому контролі заявленої якості ІТ-послуг;
- можливість оперативного управління ресурсами та компетенціями, пов'язаними з наданням ІТ-послуг;
- отримання прибутку від ІТ-діяльності, створення реальної вартості ІТ як окремого виду діяльності.

Суть структурування ІТ полягає у створенні нової системи централізованого управління інформаційними технологіями та ІТ-послугами при розподіленій моделі надання ІТ-сервісів, з використанням сучасних моделей організаційно-юридичної взаємодії учасників ІТ-діяльності, а саме в напрямках:

- стандартизації та типізації процесів в області розробки програмного забезпечення та ІТ;
- створенні типових ІТ-послуг з використанням механізмів керованих сервісних ланцюжків і методів оцінки ефективності сервісів за допомогою розрахунково-технологічних карт;
- забезпечення моніторингу ІТ-діяльності на всіх рівнях управління, виробництва та надання послуг (технологічний моніторинг, моніторинг сервісів, моніторинг SLA / якості послуг).

Стратегічною метою всього комплексу робіт зі структурування ІТ-діяльності організації є створення діючого механізму (системи, що включає в себе методики, технології, організаційні структури), що дозволяє забезпечити прибутковість ІТ, як окремого виду діяльності, забезпечити його стійке функціонування і розвиток [10]. Структурування ІТ дозволить організації досягти наступних цілей:

- Кардинально (до 60%) знизити витрати на ІТ та підвищити ефективність інвестицій в ІТ інфраструктуру;
- Збільшити продуктивність блоку ІТ, а також, підвищити ефективність використання наявної ІТ-інфраструктури.
- Забезпечувати форсування інновацій при наданні ІТ-послуг функціональним замовникам;
- Забезпечити довгострокову мотивацію учасників проекту та менеджменту шляхом надання їм опціонів в спеціально сконфігурованій на базі ІТ блоку компанії – ІТ-сервіс провайдера;
- Отримати додатковий прибуток від ІТ, як окремого виду діяльності, а також, додатковий дохід при реалізації мінарітарного пакету послуг або припинення участі в структурі капіталу ІТ сервіс провайдера;

- Знизити ризики, пов'язані з інформаційними технологіями, зокрема:
 - Ризики, пов'язані зі стратегічною пасивністю і неефективністю ІТ-зниження конкурентоспроможності бізнесу, зниження маневреності бізнесу та інше;
 - Ризики зниження адаптивності ІТ до змін і, як наслідок ризики припинення інновацій та підвищення аварійності;
 - Ризики втрати загальної керованості і керованості ІТ-інфраструктурою при зростанні кількості угод зі злиття і поглинання;
 - Ризики, пов'язані зі збільшенням складності інфраструктури та розширенням операцій;
 - Іміджеві та фінансові ризики, пов'язані з неякісним наданням послуг споживачам.

Як видно з огляду сучасних моделей управління ІТ, забезпечення якісного управління та інформаційного обміну при реалізації програмних проєктів і діяльності ІТ-компанії в цілому є важливою задачею, складність якої визначається значним переліком виконавців, задач і технологій управління [1] (Рисунок 1.1).



Рисунок 1.1. Структура управління проєктами в проєктно-орієнтованій організації

Управління роботою ІТ-компанії спрямоване на забезпечення процесів раціонального управління проєктами, удосконалення методики здійснення цих процесів відповідно до прийнятих в компанії технологій проєктування та програмування, оптимізації й інтенсифікації діяльності, поліпшення організації самостійної роботи та контролю виконання робіт працівниками компанії, підвищення управлінської майстерності керівного персоналу компанії.

Керівництво цими процесами покладається на менеджмент компанії, який також несе відповідальність за стан, ефективність та якість, рівень організації проєктного управління та його методичне забезпечення.

Рядові працівники ІТ-компаній, відповідно до планів здійснення проєктів, керуються вимогами якості виконання основних робіт за проєктами та можливої необхідності виправлення виявлених в програмному коді помилок.

1.1.2. Організаційна робота менеджера проєктів ІТ-компанії

Організаційна робота менеджерів компанії розглядається як частина їх посадових обов'язків. Ця робота планується і враховується у відповідних розділах корпоративних планів.

Зміст організаційної роботи менеджерів середньої ланки ІТ-компанії розробляється одноразово – через посадові обов'язки, поквартально, щорічно або іншим чином і затверджується керівництвом компанії. Складовими цієї роботи є:

- визначення цілей, завдань та очікуваних результатів проєкту і їх документування;
- розробка документів щодо ініціації проєкту з визначенням параметрів проєкту, переліку зацікавлених сторін, вимог до складу команди проєкту;
- встановлення границь (рамки) проєкту;
- визначення доступності ресурсів і наявних ризиків проєкту;

- планування етапів проекту, складання розкладу і виконавців робіт, визначення переліку звітів про виконані роботи тощо.

Отже, проектна та організаційна робота менеджерів середнього рівня ІТ-компанії супроводжується використанням значного обсягу документів, що розробляються відповідно до проектів та напрямків роботи компанії, а також різного інструментального програмного забезпечення, що використовується в компанії [8, 9].

Серед такого інструментального програмного забезпечення слід відзначити операційні системи, програми проектування та профілювання, системи документування та планування, компілятори та інтерпретатори, текстові та табличні процесори, сервери баз даних та веб-сервери, бібліотеки прикладних програм тощо. Всі ці електронні документи і програмні файли повинні бути ретельно задокументовані і архівовані для збереження і подальшого використання, встановлені на потрібних для використання та доступних в мережі робочих місцях компанії, а в окремих випадках – і мережі Інтернет (Рисунок 1.2).

Зважаючи на те, що система має бути багатокористувацькою, наступним важливим аспектом розробки системи є визначення можливостей використання тих чи інших комунікаційних технологій. На сьогодні найбільш поширеною групою таких технологій є Інтернет технології, які забезпечують віддалений доступ до інформаційних ресурсів різних Інтернет-серверів шляхом впровадження цілого сімейства інтернет-протоколів, таких як: HTTP, HTTPS, FTP, SMTP, SMB та інших.

Мережа Інтернет на сьогодні також є популярним засобом обміну інформацією. Все це дозволяє адміністратору проектів ІТ-компанії здійснювати розміщення в мережі Інтернет різних текстові та мультимедійних документів, а також засобів їх форматування та іншої обробки, транспортування, захисту та представлення відповідно до ролей та повноважень працівників ІТ-компанії (а в окремих випадках – і клієнтів такої компанії) та виконувати дії з підтримки такого інформаційного середовища (рисунок 1.3).

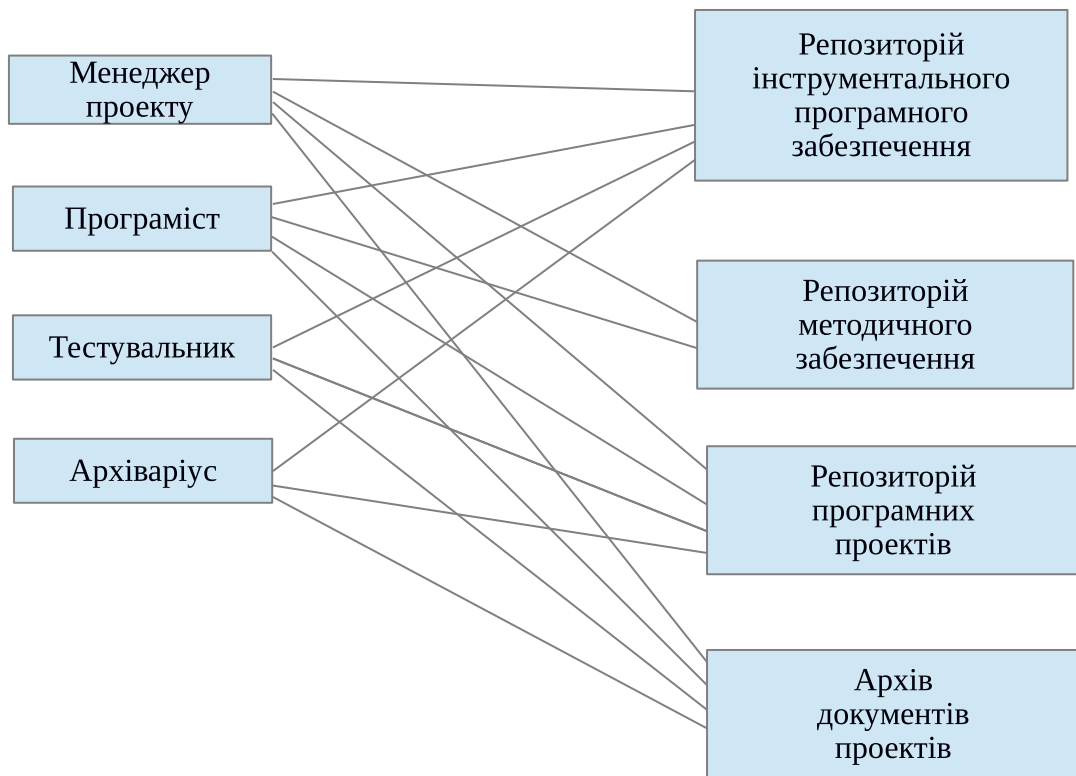


Рисунок 1.2. Модель кооперації наявної інформаційної системи ІТ-компанії

Завдяки своєму поширенню Інтернет-технології здатні вирішити задачі оперативності доступу до інформації та зменшення витрат на надання, збереження, обробку та транспортування інформації. Наприклад, в інтернет-торгівлі користувач, що має інтерес до товарів чи послуг, представлених в мережі Інтернет, може отримати їх після замовлення в інтернет за результатами пошуку в пошуковій системі. Так само і в проєктному управлінні типовим є пошук інформації з документів з подальшим використанням або наданням іншим учасникам проєкту такої інформації або копій збережених документів за результатами виконаного пошуку.

1.2. Визначення проблем діяльності ІТ-компанії

Більшість документів, що розробляються в ІТ-компанії, пов'язані один з одним або використовують спільну інформацію, що повторюється і повинна

використовуватись декілька разів. Окрім того, така інформація має ретельно перевірятись. На додаток, такі документи мають складну структуру, що визначається вимогами та стандартами щодо правил оформлення проектної і програмної документації.

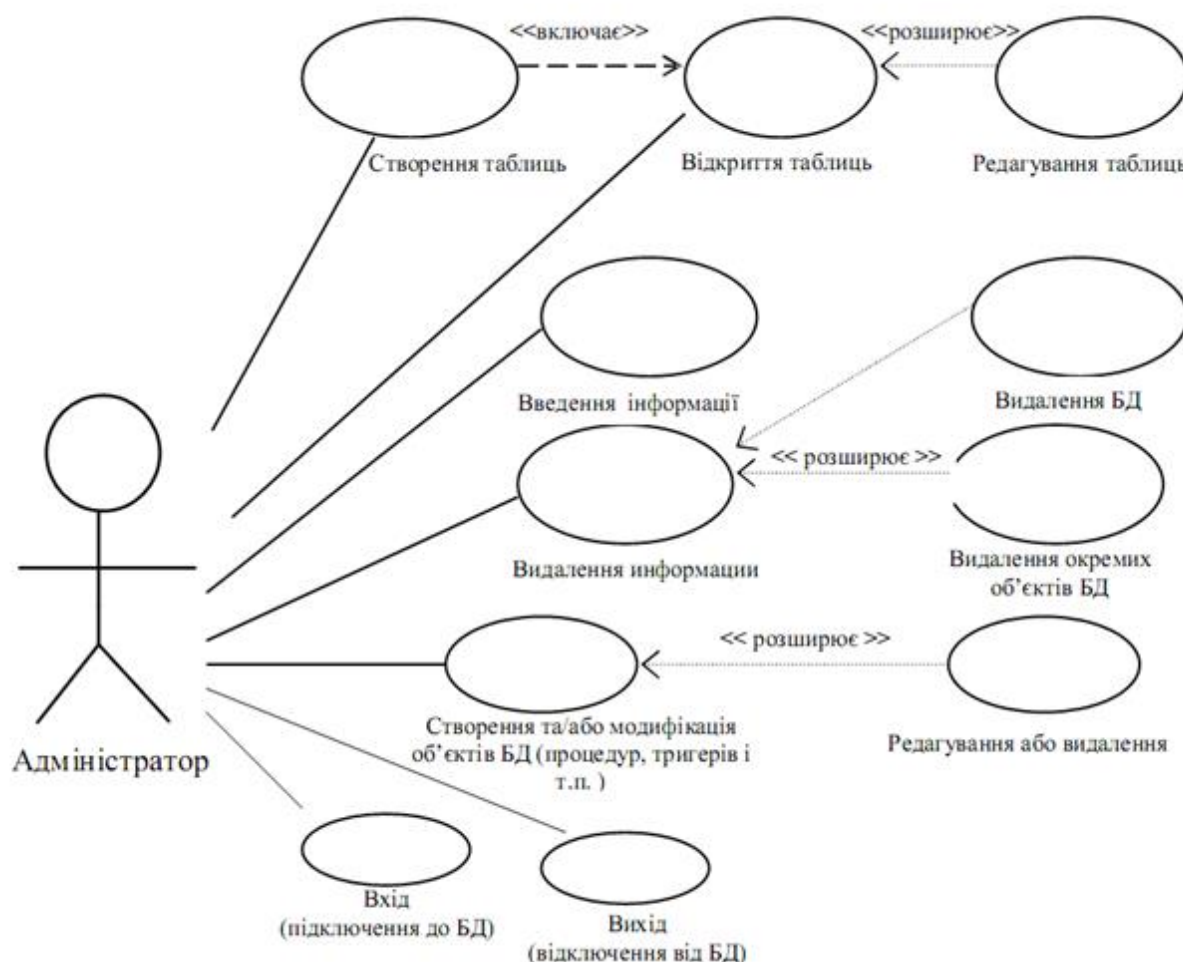


Рисунок 1.3. Модель операційної діяльності адміністратора

На даний момент проектна документація, яку розробляють проєктувальники та програмісти для використання в проєктах, зберігається у електронному або паперовому вигляді, або у вигляді текстових документів на власних носіях інформації, до яких не мають доступу менеджмент та клієнти компанії, або на закритих ззовні серверах у вигляді каталогів файлів, що не завжди розміщені зручно. Інструментальне програмне забезпечення встановлюється персоналом компанії на клієнтські робочі машини. При цьому, наприклад через ліцензійні обмеження, працівники компанії не завжди

можуть отримати необхідні програмні продукти самостійно – для цього менеджмент компанії можливо повинен узгодити використання такого продукту з іншими підрозділами компанії та дати дозвіл на отримання і встановлення файлів даних або програм окремо і кожному індивідуально, що призводить до втрати часу та не виключає ймовірності втрати або пошкодження даних або програм у архіві.

Внутрішній файловий архів компанії звичайно зберігає дані як у електронному так і у паперовому вигляді, при цьому більшість документів часто знаходиться у паперовому вигляді без дублювання в електронному. Це призводить до ускладнення пошуку конкретних документів а також і можливості втрати даних. Крім того, одні й ті самі документи можуть бути розроблені в окремій операційній або інструментальній системі і призначені для використання саме в ній (рисунок 1.4).

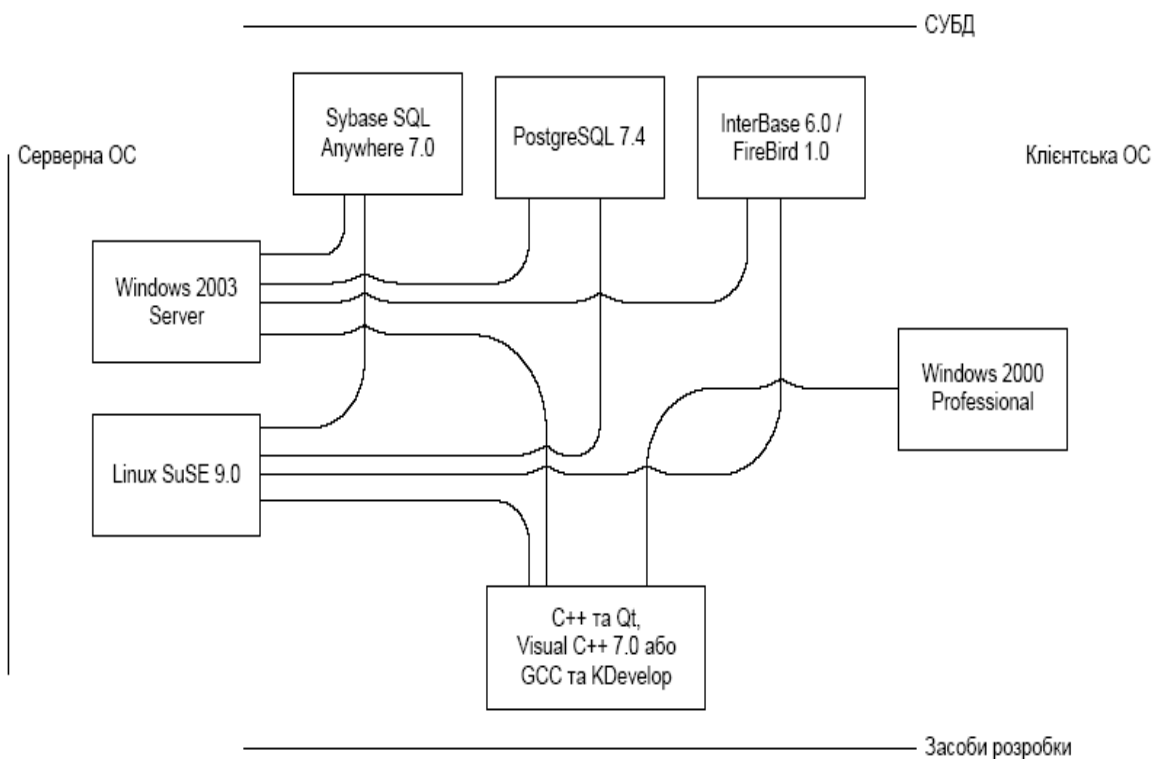


Рисунок 4. Приклад комбінації інструментальних засобів автоматизованої системи

1.3. Аналіз предметної області

1.3.1. Термінологія, визначення та технології

Проект – це обмежений часовими рамками процес, що має визначений початок та кінець, зазвичай обмежений датою, але також може обмежуватися фінансуванням або досягненням результатів, який здійснюється для реалізації унікальних цілей та завдань, зазвичай, щоб призвести до вигідних змін або створення доданої вартості. Тимчасова природа проєктів контрастує з бізнесом (процесами), які є повторюваною, постійною або частково постійною діяльністю з виробництва продуктів або послуг. На практиці, управління вищезазначеними двома системами часто різняться і таким чином вимагає розвитку окремих технічних навичок та використання розподіленого управління ними [1, 10].

Традиційно, управління проєктами включає наступний перелік елементів: чотири групи процесів та систему контролю. Незалежно від методології чи термінології, що застосовується, використовуються одні й ті самі базові процеси управління проєктами розробки ПЗ та моделі інформаційного забезпечення процесів аналізу (рисунок 1.5) та проєктування ПЗ (рисунок 1.6).

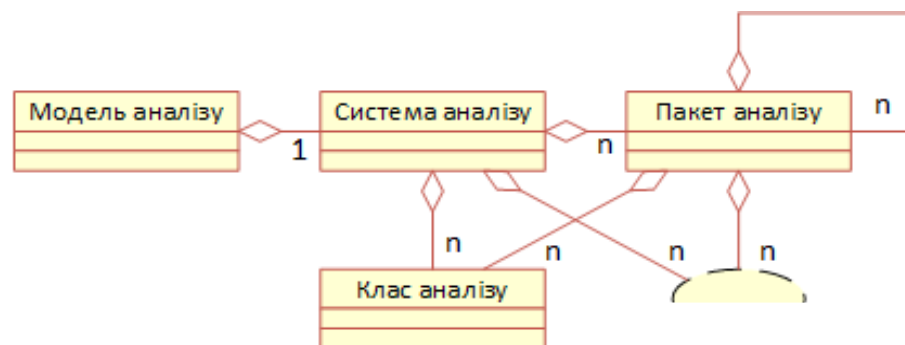


Рисунок 1.5. Структура інформаційного забезпечення процесів аналізу предметної області та вимог до ПЗ

Групи процесів при розробці складних продуктів, у тому числі інформаційних систем, зазвичай включають:

- Ініціювання;
- Планування чи розробка;
- Експлуатація чи виконання;
- Моніторинг і контроль;
- Завершення.

У проєктному середовищі зі спеціальним спрямуванням (наприклад, орієнтованому на проєкти з дослідження та проєктування) зазначені етапи можуть доповнюватися опціональними точками прийняття рішень, де визначається чи потрібно продовжувати проєкт, чи, можливо, його виконання або впровадження його результатів необхідно припинити через обставини, які не були враховані на етапі ініціації проєкту.

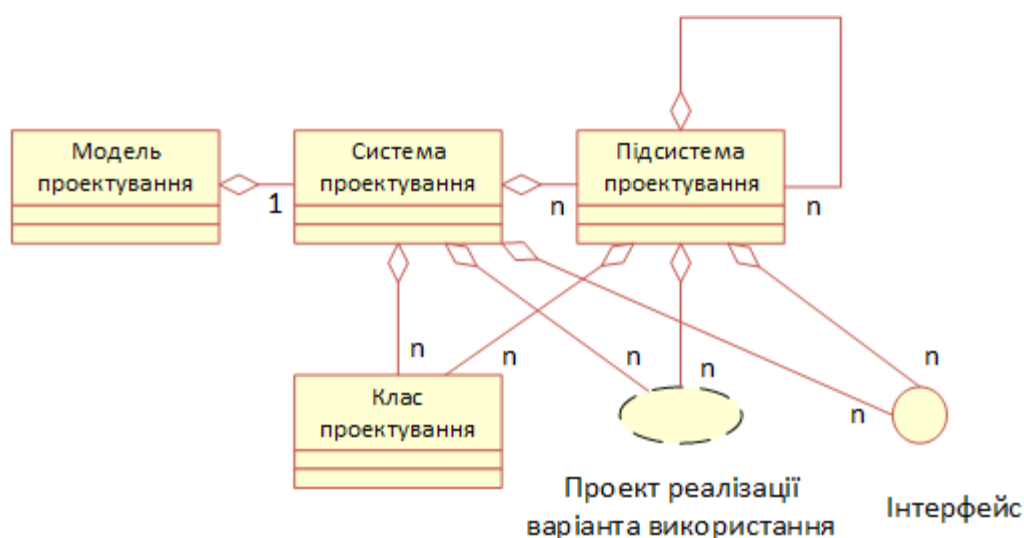


Рисунок 1.6. Структура інформаційного забезпечення процесів проектування ПЗ

Не зважаючи на те, що проєктне управління не залежить від області знань [9], під управлінням програмними проєктами звичайно маєтись на увазі ефективне здійснення програмного проєкту шляхом пошуку та застосування інструментальних засобів та технологій розробки, що найбільше

відповідають особливостям проекту, та організації робіт в цьому напрямку управління.

Основними факторами, які впливають на впровадження інформаційних систем, є потреби організацій (в нашому випадку – ІТ-компанії) та користувачів (менеджерів, керівників проектів, програмістів, тестувальників, архіваріусів тощо) до машинної обробки документів, а також наявність відповідних засобів для їх створення, збереження, форматування та надання. Найсуттєвіше на розвиток інформаційних систем вплинули досягнення в галузі комп'ютерної техніки та телекомунікаційних мереж.

Дана робота направлена на полегшення праці фахівців ІТ-компанії та взаємодії їх між собою та, можливо, з клієнтами компанії за рахунок автоматизації їх діяльності. Передбачається, що інформаційна система буде створена як ресурс, доступний безпосередньо менеджерам, програмістам, тестувальникам та архіваріусам і буде розгорнута переважно на сегменті локальної мережі компанії та на її території.

Проблема полягає в тому, що менеджерам, тестувальникам та програмістам необхідно завантажувати документальне забезпечення (методичні вказівки, проектну документацію тощо) та інструментальне забезпечення (дистрибутиви операційних систем, засобів проектування та тестування, компіляторів та інтерпретаторів, текстових та табличних процесорів, серверів баз даних, FTP-серверів та веб-серверів, бібліотек, програм комп'ютерної графіки тощо) до сховища даних (Рисунок 1.7). При цьому документальне забезпечення повинне містити посилання на відповідне інструментальне забезпечення.

Забезпечення посилковості (references) між документами вимагає використання відповідних рішень щодо створення таких посилань між документами.

Однією з найбільш популярних технологій вирішення задачі забезпечення внутрішньої і зовнішньої зв'язності документів та їх структурних елементів ("посилковості") є технологія Вікі.

Вікі-засоби – це звичайний веб-сайт (або інша гіпертекстова збірка документів), що дозволяє користувачам змінювати самостійно вміст сторінок через браузер, використовуючи спрощену і зручнішу, порівняно з HTML, вікі-розмітку тексту. Термін “Вікі” також стосується спільного програмного забезпечення (collaborative software), яке розробляється для створення і підтримки такого сайту, а також – як назва спрощеної мови розмітки документів.

Організаційна компонента:	Змістова компонента:	Технологічна компонента:
• Організаційні структури; • Корпоративні стандарти;	• Навчальні ресурси; • Наукові ресурси; • Методичні ресурси; • Контролюючі ресурси; • Довідкові ресурси;	• ІТ-інфраструктура; • Централізовані сервіси; • Децентралізовані сервіси.

Рисунок 1.7. Компоненти інформаційного забезпечення проєктів ІТ-компанії

Таким чином, для забезпечення посилковості між документами, їх буде необхідно попередньо розставити внутрішні та зовнішні посилання, елементи форматування тощо (“вікіфікувати”). Така робота потребує певних зусиль та витрат часу. В окремих випадках для цього використовують спеціальні конвертери, які на основі деяких структурованих даних здатні за лічені хвилини переробити наявну інформацію в підготовлену та відформатовану документацію згідно до вікі-розмітки інформаційних ресурсів компанії та прив'язати таку документацію до структури інших документів, наприклад – через механізм категоризації документів.

Вікі-рушій (англ. wiki engine) – набір програм, служить для перетворення вікі-розмітки у зручно читаєме подання на мові HTML і взаємодії користувача з базою даних. Існуючі на сьогодні вікі-движки забезпечують підтримку різних мов програмування, доступні як бібліотечні модулі та можуть використовуватись з різними базами даних в різних

операційних системах. Основним та найбільше поширеним проектом Вікі є Вікіпедія – відкрита багатомовна ВІКІ-енциклопедія, якою опікується некомерційна організація «Фонд Вікімедіа». Назва утворена від слів «ВІКІ» (технології для створення сайтів) та «енциклопедія».

З метою подальшої розробки визначимо такі основні поняття, як контент, вікіфікація текстових документів, конвертація та архівування даних, та їх особливості.

У контексті програмування, контент – це різні дані, які програма повинна завантажити в пам'ять (з файлу), щоб пізніше оперувати ними (генерувати рисунки, креслити таблиці, відтворювати текст з посиланнями тощо).

Вікіфікація – це процес додавання або виправлення вікі-розмітки в існуючих статтях. Вікіфікація включає проставлення внутрішніх посилань на ключові слова або зовнішні ресурси статті там, де вони необхідні або будуть доречними.

Вікіфікатор – інструмент для автоматичного оброблення вікі-тексту згідно деяких правил вікі-розмітки. Звичайно вікіфікатор запускається натисканням відповідної клавіші над вікном редагування і надає середовище для обробки тексту та зображень.

Конвертер – комп'ютерна програма, яка перетворює дані вхідних файлів з одного формату представлення в інший формат, зазвичай – із збереженням у вихідному файлі основного логічно-структурного змісту основної інформації вхідного файлу.

З метою визначення можливостей представлення даних в системі розглянемо можливості форматування даних, що надаються сучасними вікі-рушіями (таблиця 1.1).

З метою визначення попередніх вимог щодо форматування даних в системі, розглянемо засоби форматування даних в системі та можливості поєднання окремих об'єктів даних в єдине представлення даних – контент автоматизованої системи. Перелік засобів форматування даних, що надаються сучасними вікі-рушіями, наведений в таблиці 1.2.

1.3.2. Аналіз існуючих рішень та технологій інформаційної підтримки проєктів

Загалом, в управлінні проєктами існує спеціальне програмне забезпечення, призначене для інформаційного забезпечення операційних процесів виконання та супроводу проєктів.

Таблиця 1.1. Засоби форматування даних, що надаються сучасними вікі-рушіями

Опис / вигляд прикладу	Wiki	MediaWiki
Заголовки	== Розділ == === Підрозділ === ==== 4 рівень ==== ===== 5 рівень ===== ===== 6 рівень =====	= Перший рівень = == Другий рівень == === Третій рівень === ==== Четвертий рівень ==== ===== П'ятий рівень =====
курсив	"курсив"	"курсив"
напівжирний шрифт	"жирний шрифт"	"жирний шрифт"
моноширинний шрифт	<tt>моноширинний шрифт </ tt>	`моноширинний шрифт`
код		{{{код}}}
підкреслення	<u>підкреслення </u>	__підкреслення __
перекреслення	<strike>перекреслення </ strike>	--(перекреслення)--
Внутрішнє посилання	[[Вікіпедія:Правила#Текст Текст]].	[[Головна сторінка посилання]]
Зовнішнє посилання	http://www.dia.com	[[http://www.dia.com]]

Як приклад, на рисунку 1.8 наведений інтерфейс програми-робота, яка автоматично збирає інформацію проєктів та генерує на її основі впорядковані та логічно зв'язані через гіперпосилання документи у форматі, що застосовується у проєкті “Вікіпедія”.

Таблиця 1.2. Порівняльна таблиця характеристик технологій обробки і вікіфікації документів

Технології	Управління доступом	Простота використання	Потреба в додатковому ПЗ
Пісочниця	+	+	+
MediaWiki:Wikificator	-	-	-
АвтоВікіБраузер	-	+	-

Тестовим інструментом “Вікіпедії”, що надається користувачам для тестування оформлення інформаційного наповнення та інструментального забезпечення вікіпедії, є «Пісочниця». Наявність цього інструменту відповідає головному принципу вікі — з документами може працювати користувач з мінімальною підготовкою і зміни будуть відразу ж враховані і представлені на сайті. Пісочниця вікі має зручний і простий інтерфейс користувача, що дозволяє забезпечити її просте використання в системі з часто змінюваними користувачами (переважно — програмістами та тестувальниками; менеджерами та керівниками проєктів) без необхідності їх додаткового навчання роботі в системі.

Для редагування поточного документа (в термінології Вікі такі документи звуться “статті”) треба внести бажані зміни у вигляді тексту та тегів форматування у вихідний текст, для перевірки правильного відображення редагованого документа (“сторінки”) достатньо завершити редагування і форматування, натиснути кнопку «Попередній перегляд», і якщо вміст і вигляд сторінки відповідає бажаному, опублікувати нову версію

документа на сервері автоматизованої системи натисканням кнопки «Зберегти».

Вище текстової форми редагування знаходяться кнопки для форматування тексту, нижче – панель швидкої вставки спеціальних символів, елементів вікі-розмітки і шаблонів.

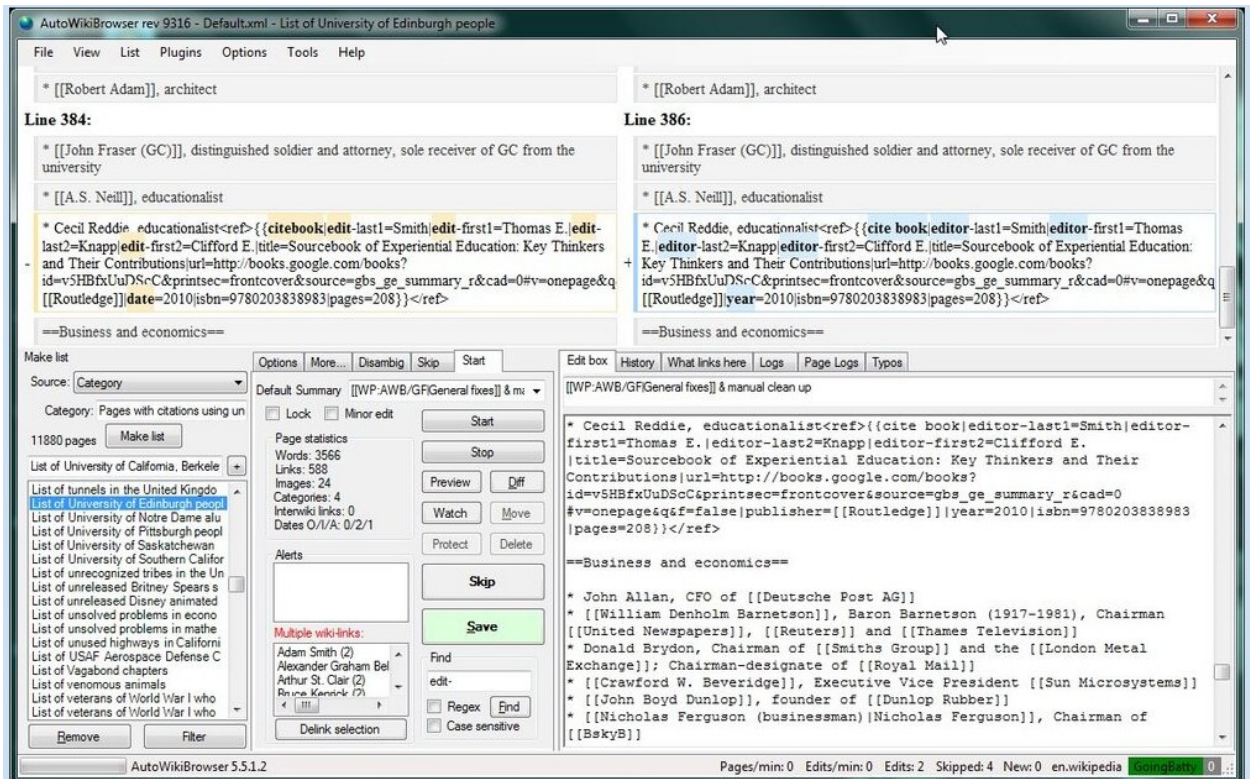


Рисунок 1.8. Приклад інтерфейсу програми автоматизованого збирання інформації з/у вікі сховище ІТ-компанії

MediaWiki:Wikificator – інструмент для вікіфікації інформаційних ресурсів Вікіпедії. Основні функції:

- Замінює деякі HTML-теги на аналогічну вікі-розмітку
- Виділяє теги жирного шрифту (</ b>, </ strong>);
- Виділяє теги курсиву (<i> </ i>, </ em>);
- Вставляє, де потрібно, тире і мінус;
- Розставляє правильні лапки і лапки всередині лапок;
- Замінює (tm), (r) і тому подібні на спеціальні символи;

- виправляє неправильно зазначені службові слова;
- розставляє пропущені та видаляє зайві пробіли;
- корегує і розкриває загальноприйняті скорочення.

Вікіфікатор не перетворює текст: в середині подвійних фігурних дужок (`{{i}}`), всередині подвійних квадратних дужок (`[[i]]`, або `[[i ]`), Вікіфікатором не обробляються рядки, що починаються з пробілів (звичайний текст). Видаляються непотрібні пропуски, а пропущені пробіли, навпаки, вставляються тощо.

АвтоВікіБраузер – інструмент для правок Вікіпедії в напівавтоматичному режимі, що працює під Windows 2000/XP/Vista/7. Для роботи потрібно .NET Framework версії 2.0. Для його використання потрібно занесення учасника в список дозволених учасників.

АвтоВікіБраузер має наступну функціональність:

- виправляє часті помилки в секціях «Див. також» і «Посилання», прибирає зайві пробіли;
- додає маркер списку у зовнішні посилання після “`==Зовнішні посилання==`”.
- замінює курсивну і жирну HTML-розмітку на вікі-розмітку.
- спрощує посилання за принципом “`[[Автор|Автор]]`” на “`[[Автор]]`”.
- спрощує посилання за принципом “`[[Автор|Автора]]`” на “`[[Автор]]а`”.
- перетворює посилання зі сторінки на саму себе у звичайний текст.
- дозволяє множинні пошук і заміну.
- додає вказаний текст у кінець сторінки (вибрано “Append”) або на початок сторінки (вибрано “Prepend”).

АвтоВікіБраузер має наступні органи управління:

- Створити опис – правка опису редагування.
- Переглянути статистику статті – надає доступ до різних статистик, наприклад, кількість знаків і зображень.
 - Дозволяє попередній перегляд.
 - Показує зміни статті до попередньої версії.
 - Містить кнопку «Зберегти».

АвтоВікіБраузер розповсюджується вільно і безкоштовно на умовах ліцензії GPL. Програмний код АвтоВікіБраузера можна завантажити з сервера SVN на sourceforge, для цього потрібна програма-клієнт SVN.

1.3.3. Порівняльна характеристика технологій документування

З метою визначення технологій, що якнайкраще дозволяють подолати існуючі в предметній області проблеми і є більш актуальними для забезпечення діяльності проєктного управління в ІТ-компанії складемо порівняльну характеристику за критеріями, найбільш важливими для попередньої обробки інформації і подальшого її завантаження (таблиця 2).

В таблиці зведені основні технології підготовки та представлення контенту, та їх експлуатаційні переваги та недоліки. Наявність тих чи інших можливостей технології позначена символом “+”, її відсутність або суттєва недосконалість – символом “-”.

Варто зазначити, що класифікація існуючих технологій за потребою у додатковому програмному забезпеченні проводиться від зворотнього. Тобто, MediaWiki:Wikificator і АвтоВікіБраузер мають потребу в даній функції, але для реалізації поставленої задачі це є зайвим.

MediaWiki:Wikificator і АвтоВікіБраузер заміняють HTML-теги на відповідну вікі-розмітку, але тут потрібен інструмент для конвертації однієї вікі-розмітки в іншу, використану одним з рушіїв вікі.

Пісочниця є зручним і простим у використанні програмним продуктом, спеціально створеним для завантаження контенту до Вікіпедії, що підтримують тільки одну версію вікі-розмітки, тому ми не можемо використовувати його для виконання поставленої задачі, але саме за допомогою цього інструменту менеджер та керівник проєкту зможе підготувати фрагмент документації до проєкту – статтю для подальшої конвертації в віддаленому режимі. Отже, враховуючи вищеописані положення та специфіку даної предметної області є необхідність в розробці власної системи.

1.4. Розробка концепції автоматизованої системи управління програмними проектами

Враховуючи результати аналізу діяльності об'єкта автоматизації та аналізу існуючих рішень, а також маючи за мету підвищення ефективності об'єкта автоматизації розробимо концепцію автоматизованої системи.

Виходячи з наявності істотних проблем пошуку та співставлення з проектами інструментального програмного забезпечення (а для програмістів та тестувальників – ще й методичного забезпечення) розробимо концепцію системи, яка буде забезпечувати збереження інформації про інформаційні ресурси системи співставлення проектів ІТ-компанії з іншими ресурсами системи.

Модель кооперації запропонованої концепції автоматизованої системи представлена на рисунку 1.9.

Запропонована концепція автоматизованої системи може мати наступні основні переваги над існуючою системою:

- буде корисною менеджерам проектів, бо містить не тільки описи проектів та робіт, але й приклади їх виконання (методичне забезпечення), та інструментальне програмне забезпечення, необхідне для виконання робіт по проекту;
- може мати більш зручний і простий у використанні інтерфейс, що враховує певне спрямування проектів компанії.

Реалізація подібної системи буде виконуватись на основі:

- пошуку символів синтаксису вікі-розмітки тої чи іншої категорії та заміні її на синтаксис використовуваної мови на ресурсі в репозиторії системи;
- пошуку посилань на локальні інструментальні ресурси з заміною імен таких ресурсів на їх адреси в репозиторії проектів та зберігаємого інструментального програмного забезпечення.

Наявність цих механізмів дозволить досить швидко завантажувати в систему необхідні текстові документи (наприклад – листування за проектом) та зв'язувати такі документи між собою. Зв'язування текстових документів з

проектною документацією (таблицями, схемами, кресленнями, графікою тощо) може виконуватись за іменами файлів в репозиторіях, наприклад – відповідно до назви проекту.

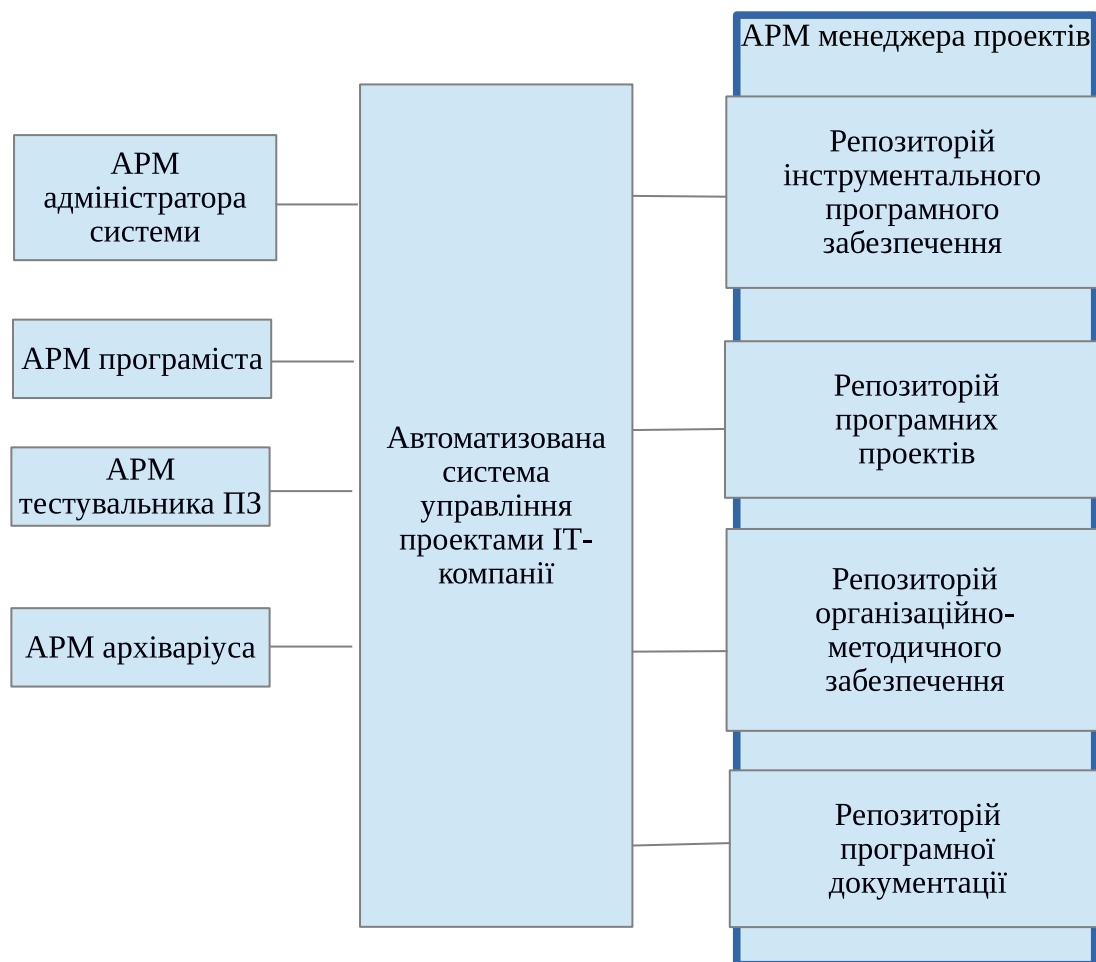


Рисунок 1.9. Модель кооперації варіанта концепції АРМ менеджера програмних проєктів АІС ІТ-компанії

Процедура аналізу тексту виконуватиметься за наступним принципом: в результаті аналізу, програмне забезпечення системи повинне видати готову для завантаження сторінку. Зважаючи на можливо різний рівень підготовки персоналу, в програмному забезпеченні системи потрібно реалізувати зручний користувацький інтерфейс. Редактор повинен дозволяти користувачеві маніпулювати текстовими об'єктами для створення контенту, тобто змінювати їх розмір, структуру і властивості, якими володіє об'єкт, та

документами з проєктною документацією (схемами, кресленнями, графікою тощо). Оцінка тексту в системі буде виконуватись за допомогою метода пошуку службових символів вікі та заміни на потрібні для зв'язування документів. Відмінності синтаксису для порівняння були представлена в таблиці 1.1.

Запропонований варіант концепції підсистеми автоматизованої системи (рисунок 1.3) може бути реалізований декількома мовами програмування. Зважаючи на те, що проєктована система є Інтернет-проєктом та враховуючи, що останнім часом найбільше популярними технологіями розробки веб-додатків є технології на основі мов програмування PHP та Python, надалі будемо розглядати два варіанти розробки програмного забезпечення для реалізації концепції системи – на основі високорівневих інтерпретованої мови програмування PHP та на основі інтерпретованої мови програмування Python.

1.5. Функціонально-вартісний аналіз

1.5.1. Функціональний аналіз

На основі вимог до функцій та якостей автоматизованої системи визначимо параметри системи:

- захищеність програмного забезпечення;
- швидкість розробки ПЗ проєкту;
- наочність коду;
- швидкість обробки запитів;
- валідність коду;

Результати експертних оцінок вагових параметрів методом попарного порівняння і визначення вагових коефіцієнтів ϕ_i наведені в таблиці 1.3.

Для уникнення рівності оцінок кількість експертів має бути непарною. В якості експертів можуть виступати спеціалісти з експлуатації, використання та тестування інформаційного забезпечення.

Усереднені оцінки (максимальне значення параметра = 10 балів, мінімальне = 0) експерти занесені в таблицю 1.4.

Визначимо коефіцієнти значущості ϕ_i і бальні оцінки B_i інформаційної системи (представлені в таблицях 1.4 та 1.5).

Крім того, визначимо коефіцієнт технічного рівня $K_{тр.} = \sum \phi_i * B_i$, який визначає рівень використовуваних в системі технологій.

Результати розрахунків по запропонованим варіантам (варіант 1 – PHP, варіант 2 – Python) концепції автоматизованої інформаційної системи представлені в таблиці 1.6.

Таблиця 1.3. Основні параметри ІС (варіант 1 – PHP, варіант 2 – Python)

Параметр	Варіант 1	Варіант 2
Захищеність програмного забезпечення	9	8
Швидкість розробки ПЗ проєкту	7	8
Наочність коду	9	10
Швидкість обробки запитів	8	9
Валідність коду	9	9

Таблиця 1.4. Визначення коефіцієнта значущості ϕ_i

Параметр	ϕ_i
Захищеність програмного забезпечення	0,40
Швидкість розробки ПЗ проєкту	0,35
Наочність коду	0,4
Швидкість обробки запитів	0,35
Валідність коду	0,35

1.5.2. Вартісний аналіз

Аналіз функціональних властивостей системи та її функцій має бути доповнений вартісним аналізом. Для цього розглянемо зведені витрати по кожному варіанту.

Зведені витрати Z_j розраховуються у сфері виробництва (у розробника) або у сфері експлуатації (у виконавця). У нашому випадку, досить розрахувати у сфері виробництва.

Поточні витрати при розробці інформаційної системи – це собівартість створення системи. Для усіх варіантів вважатимемо, що собівартість системи однакова.

Таблиця 1.5. Визначення коефіцієнтів ІС (варіант 1 – PHP, варіант 2 – Python)

Параметр	Варіант	Бальна оцінка Ві	Ваговий коефіцієнт ϕ_i	Ктр.
Захищеність програмного забезпечення	1	9	0,40	3,6
	2	8		3,2
Швидкість розробки ПЗ проекту	1	7	0,35	2,45
	2	8		2,8
Наочність коду	1	9	0,4	3,6
	2	10		4
Швидкість обробки запитів	1	8	0,35	2,8
	2	9		3,15
Валідність коду	1	9	0,35	3,15
	2	9		3,15

Таблиця 1.6. Коефіцієнти технічного рівня (варіант 1 – PHP, варіант 2 – Python)

Варіант 1	3,12
Варіант 2	3,26

Капітальні вкладення – це вкладення, які мають одноразовий характер. В даному випадку, в капітальні вкладення ми включимо купівлю ліцензійного програмного забезпечення, а так само технічного і організаційного забезпечення, необхідного для створення системи.

Зважаючи, що робота виконується як дипломна, вартість програмного забезпечення за обома варіантами на даний момент будемо вважати однаковою.

1.6. Постановка задачі на створення підсистеми управління програмними проектами ІТ-компанії

Виходячи з результатів аналізу діяльності об'єкта автоматизації та предметної області є нагальною необхідність і доцільність розробки автоматизованої системи управління проектами ІТ-компанії на основі вікі-технологій з використанням конвертеру текстових файлів, які разом складатимуть систему підтримки процесів завантаження і організації контенту документації за проектами в системі управління проектами, організації навчальних проєктів (методичного забезпечення) тощо.

Програмне забезпечення автоматизованої системи повинне забезпечити можливості створення та вікіфікації інформаційних ресурсів системи – документів в процесі редагування текстів, та надання прив'язок наявного інструментального забезпечення до проєктів, навчально-методичних матеріалів для підготовки фахівців компанії (навчальних проєктів) та основних проєктів інформаційної системи ІТ-компанії.

1.7. Висновки до розділу 1

В результаті проведеного аналізу визначено основні характеристики об'єкта автоматизації та проблеми його діяльності, що полягають в необхідності управління програмними проектами ІТ-компанії та відповідного інструментального забезпечення.

Для подолання зазначених проблем запропоновано використання вікі-технологій, які дозволять забезпечити логічне поєднання програмних проєктів ІТ-компанії, методичного інформаційного та інструментального забезпечення в єдиній автоматизованій системі.

2. ПРОЕКТ АРМ МЕНЕДЖЕРА ПРОГРАМНИХ ПРОЕКТІВ АІС ІТ-КОМПАНІЇ

2.1. Загальносистемні рішення

2.1.1. Вибір технології та засобів моделювання

Відповідно до наявності декількох технологій проєктування програмного забезпечення та враховуючи, що основою данної системи має бути саме програмне (а не, наприклад, технічне) забезпечення, розглянемо популярні технології проєктування ПЗ.

Об'єктно-орієнтоване програмування (ООП) – це спосіб організації коду в програмі шляхом його угруповання у вигляді об'єктів, тобто окремих елементів, які включають інформацію (значення даних) і функції. Об'єктно-орієнтований підхід до організації програми дозволяє групувати певні фрагменти інформації разом з часто використовуваними функціями або діями, пов'язаними з цією інформацією. Ці елементи об'єднуються в один елемент, так званий об'єкт. Можливість зв'язати разом ці значення і функції дає кілька переваг. Наприклад, потрібно відстежувати тільки одну змінну, а не декілька; можна згрупувати разом пов'язані між собою функції; можна організувати структуру програми так, щоб вона більше відповідала реальним потребам. Таким чином в програмному забезпеченні автоматизованої системи буде використовуватись інформація з бази даних та функції для роботи з нею, такі як додавання та редагування матеріалу [9].

В об'єктно-орієнтованому підході основна категорія об'єктної моделі – клас – об'єднує в собі на елементарному рівні як дані, так і операції, які над ними виконуються (методи). Саме з цієї точки зору зміни, пов'язані з переходом від структурного до об'єктно-орієнтованого підходу, є найбільш помітними. У даному програмному продукті автоматизованої системи буде зручно використовувати саме об'єктно-орієнтований підхід, тому що він має суттєві переваги:

- об'єктна декомпозиція дає можливість створювати програмні системи меншого розміру шляхом використання загальних механізмів, що забезпечують необхідну економію засобів вираження. Використання

об'єктного підходу істотно підвищує рівень уніфікації розробки і придатність для повторного використання не тільки програм, але і проєктів, що веде до створення середовища розробки і переходу до складеного створення ПЗ. Системи найчастіше виходять більш компактними, ніж їх структурні еквіваленти, що означає не тільки зменшення обсягу програмного коду, але і здешевлення проєкту за рахунок використання попередніх розробок.

- об'єктна декомпозиція зменшує ризик створення складних систем ПЗ, так як вона передбачає еволюційний шлях розвитку системи на базі відносно невеликих підсистем. Процес інтеграції системи розтягується на весь час розробки, а не перетворюється на одноразову подію;
- об'єктна модель цілком природна, оскільки в першу чергу орієнтована на людське сприйняття світу, а не на комп'ютерну реалізацію;
- об'єктна модель дозволяє повною мірою використовувати виразні можливості об'єктних і об'єктно-орієнтованих мов програмування [15].

Для розробки об'єктно орієнтованого проєкту потрібні об'єктно-орієнтовані засоби проєктування. Найбільш універсальним інструментом для цього є універсальна мова моделювання (Unified Modelling Language або UML) – мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об'єкти систем програмного забезпечення. Ця мова допомагає наочно переглядати компонування системи і полегшує співпрацю з іншими учасниками її розроблення. UML розроблено для моделювання структури і зорієнтованого на об'єкти програмного забезпечення [16].

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення [16]. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему [8].

ArgoUML – засіб UML моделювання. ArgoUML є відкритим програмним забезпеченням.

ArgoUML повністю написаний на Java і для роботи йому підходить будь-яка операційна система з встановленою Java 2 JRE або JDK версії 1.4 або вище. Редактор має різні елементи для побудови діаграм (класи, інтерфейси тощо), які зручно налаштовані для швидкої вставки і редагування. Ці елементи можуть бути класифіковані за пріоритетністю, метою або типом, завдяки чому можливо контролювати їх не тільки візуально, але і змінювати властивості кожного з об'єктів. Таким чином, крім налаштування різних прапорів і властивостей об'єктів, є можливість додавати нотатки, що безперечно дуже зручно, коли мова йде про розробку та впровадження. У редакторі ArgoUML для проєкту була використана нотація UML1.4.

У процесі розробки програмного продукту може бути застосована методологія Rational Unified Process.

Rational Unified Process (RUP) – методологія розробки програмного забезпечення, створена компанією Rational Software. RUP використовує ітеративну модель розробки. В кінці кожної ітерації (ідеально триваючої від 2 до 6 тижнів) проєктна команда повинна досягти запланованих на дану ітерацію цілей, створити або допрацювати проєктні артефакти і отримати проміжну, але функціональну версію кінцевого продукту. Ітеративна розробка дозволяє швидко реагувати на мінливі вимоги, виявляти і усувати ризики на ранніх стадіях проєкту, а також ефективно контролювати якість створюваного продукту. Ця методологія має 4 стадії [13]:

- початкова стадія (Inception). Формуються бачення і межі проєкту, створюється економічне обґрунтування, визначаються основні вимоги, обмеження і ключова функціональність продукту. Створюється базова версія моделі прецедентів.
- уточнення (Elaboration). Виробляється аналіз предметної області та побудова виконуваної архітектури.
- побудова (Construction). Відбувається реалізація більшої частини функціональності продукту. Завершується першим зовнішнім релізом системи.

- впровадження (Transition). Створюється фінальна версія продукту проєкту і передається передача результатів проєкту від розробника до замовника. Це включає в себе програму бета-тестування, навчання користувачів, а також визначення якості продукту. У разі, якщо якість не відповідає очікуванням користувачів або критеріям, встановленим у початковій фазі, фаза «Впровадження» повторюється знову.
- виконання всіх задач для досягнення цілей означає досягнення віхи готового продукту (Product Release) та завершення повного циклу розробки проєкту.

Визначення цих стадій є цінним не тільки з огляду на розробку програмного забезпечення системи, але й з тієї точки зору, що сама розроблювана система призначена для збереження програмних проєктів та інформаційних ресурсів про ці програмні проєкти.

2.1.2. Функціональна модель системи

Для вивчення особливостей роботи (функціонування) АРМ менеджера програмних проєктів АІС ІТ-компанії, його призначення та визначення взаємозв'язку АРМ з внутрішніми і зовнішніми елементами побудуємо функціональна модель (рис.2.1). Для цього виділимо основних акторів системи:

- адміністратор – особа, яка має в системі акаунт адміністратора, і можливість редагування інформації в БД, стосовно до предметної області в якості цього актора виступають менеджери та керівники проєктів ІТ-компанії;
- користувач – особа, яка використовує програму для пошуку та перегляду інформації, стосовно до функціональних обов'язків з предметної області; в якості цього актора виступають програмісти, тестувальники, архіваріуси, що працюють з проєктами компанії.

Для виявлених акторів представимо основні варіанти використання (прецеденти) в системі та, відповідно до вимог Технічного завдання (Додаток

А) до функціональних характеристик системи, розробимо діаграму варіантів використання (Use-Case Diagram).

Виявлені відповідно до Технічного завдання варіанти використання наведені у таблиці 2.1.

Структурну модель процесів проектування АРМ системи представлено на рисунку 2.1.

Діаграму варіантів використання, що моделює процеси обробки даних та потоки даних, зображено на рисунку 2.2.

Таблиця 2.1 – Виявлені варіанти використання

Основні актори	Найменування	Формулювання
Менеджер	Створити прпоект, отримати звіт за проектами	Створює новий ресурс в репозиторії; надає інформацію про стан ресурсів репозиторія.
Керівник проекту	Створити прпоект, отримати звіт за проектом	Створює новий ресурс в репозиторії; надає інформацію про стан ресурсу репозиторія.
Програміст	Змінити стан ресурсу	Встановлює значення атрибутів ресурсу відповідно до функціональних обов'язків
Тестувальник	Змінити стан ресурсу	Встановлює значення атрибутів ресурсу відповідно до функціональних обов'язків
Архіваріус	Змінити стан ресурсу	Встановлює значення атрибутів ресурсу відповідно до функціональних обов'язків
Файлове сховище	Розмістити статтю	Зберігає і розміщую створену статтю на ресурсах АРМ ІТ-компанії

2.2. Рішення з інформаційного забезпечення

Інформаційно-логічна модель відображає дані предметної області у вигляді сукупності інформаційних об'єктів і зв'язків між ними. Ця модель представляє дані, що підлягають зберіганню у базі даних.

Основними компонентами логічної моделі є сутності, атрибути та зв'язки [8]. Сутності трансформуються в таблиці даних, одна сутність може представлятися декількома таблицями, атрибути сутностей стають полями таблиць, уточнюються відношення між таблицями, виділяються первинні ключі [14].

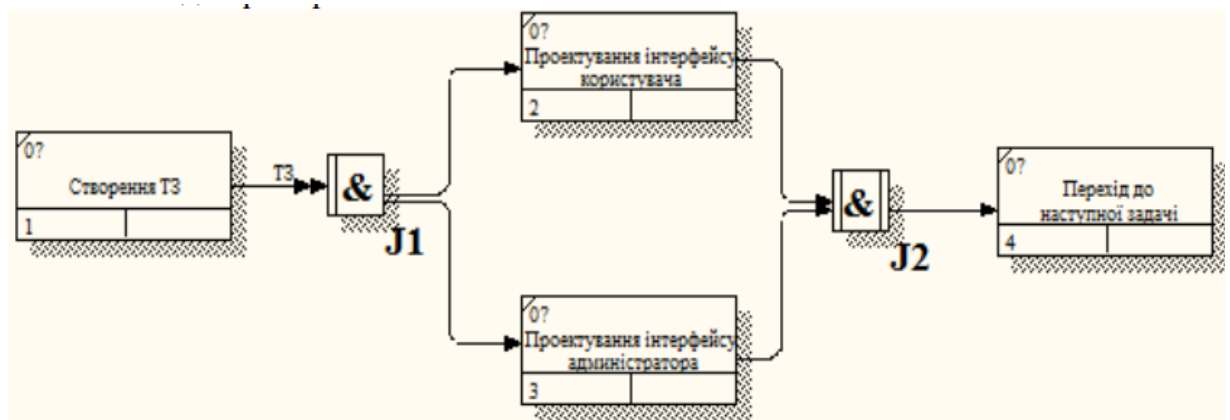


Рисунок 2.1. Структурна модель процесів проєктування АРМ системи

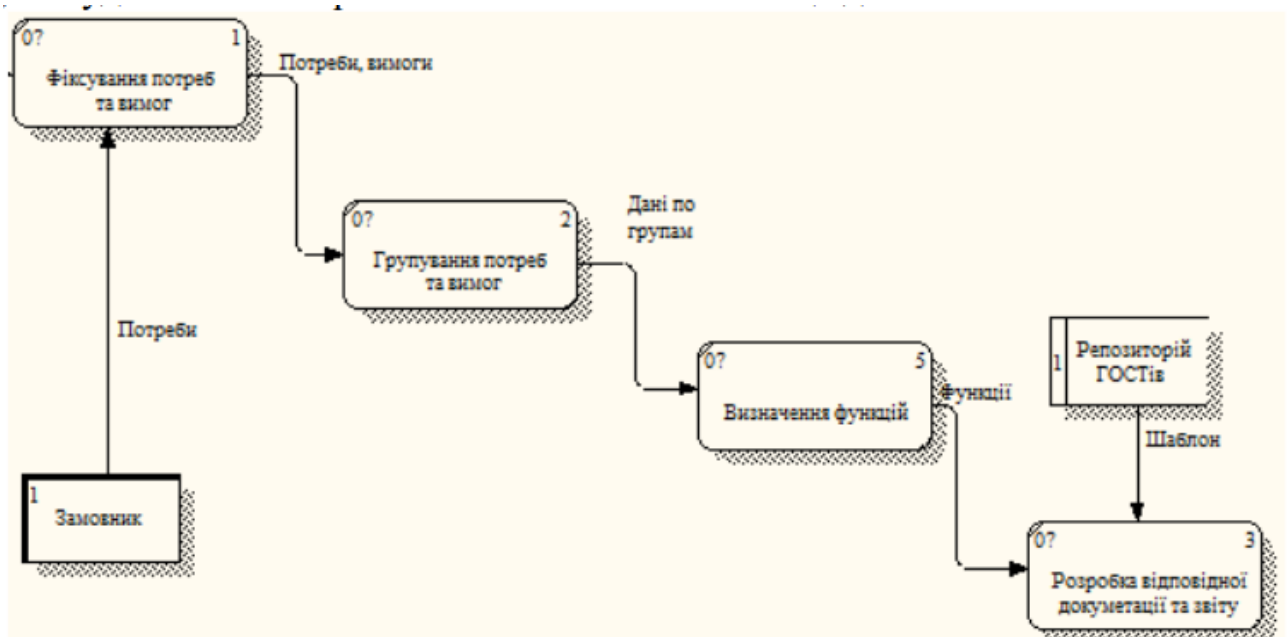


Рисунок 2.2. Діаграма процесу проєктування програмного забезпечення

Таблиця 2.2. Опис функцій акторів діаграми прецедентів

Основний актор	Назва	Формулювання
Адміністратор (менеджер)	Створення облікового запису адміністратора	створити обліковий запис адміністратора, а саме ввести особисті дані, що ідентифікують особу.
Адміністратор (керівник проєкту)	Авторизація адміністратора	авторизуватися адміністратору, а саме ввести логін та пароль.
Адміністратор	Функція додавання нового проєкту	додавати нові проєкти.
Адміністратор, програміст, тестувальник	Редагування матеріалу	редагувати назву, опис та зміст проєктів та відображати зміни до бази даних.
Адміністратор	Функція видалення матеріалу	видаляти матеріали.
Адміністратор	Функція перегляду повної інформації про матеріалі	переглянути повну інформацію про матеріал, переглянути текст, зміст БД.
Адміністратор, програміст, тестувальник, архіваріус	функція пошуку матеріалів	шукати матеріали.

Для розробки інформаційної моделі застосуємо діаграму класів. Діаграма класів – це набір статичних, декларативних елементів моделі. Діаграми класів можуть застосовуватись як при прямому проєктуванні, тобто в процесі розробки нової системи, так і при зворотному проєктуванні – описі існуючих і використовуваних систем. Інформація з діаграми класів безпосередньо відображається у вихідний код програми – у більшості

існуючих інструментів UML – моделювання можлива кодогенерація для певної мови програмування. Таким чином, діаграма класів – кінцевий результат проєктування і відправна точка процесу розробки [16].

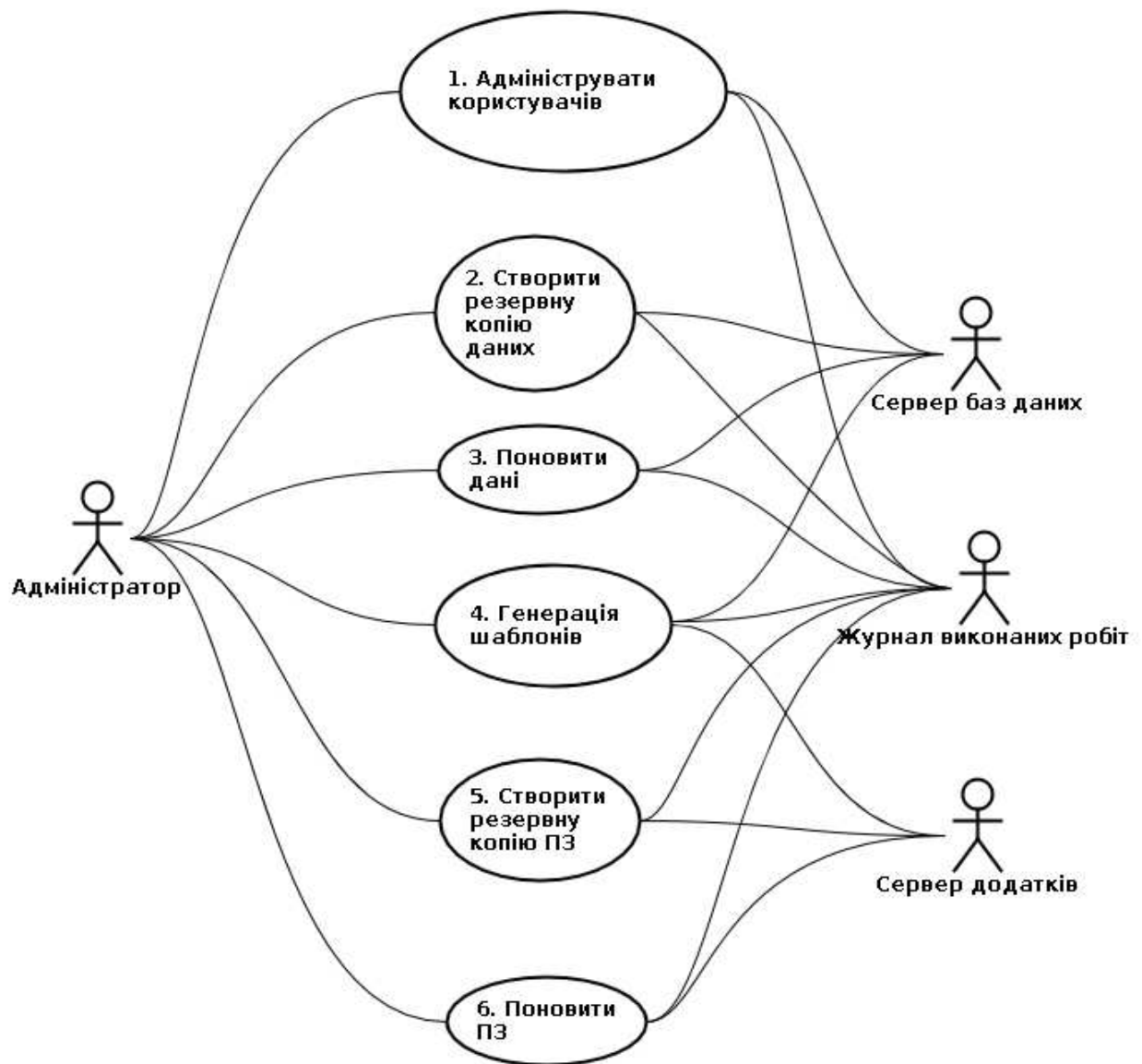


Рисунок 2.3. Діаграма варіантів використання процесу адміністрування системи

Клас у мові UML служить для позначення безлічі об'єктів, які мають однакову структуру, властивості, поведінку й відносинами з об'єктами з інших класів. Таким чином, клас представляє собою шаблон для створення об'єктів. Кожен об'єкт є екземпляром конкретного класу.

Існує два види діаграми класів:

- статичний вид діаграми розглядає логічні взаємозв'язки класів між собою;
- аналітичний вид діаграми розглядає загальний вигляд і взаємозв'язки класів, що входять в систему.

З метою опису сутностей та структури даних на основі проведеного аналізу розробимо модель даних системи. З урахуванням поставлених задач програма повинна включати сутності та зв'язки між ними. Діаграми моделі даних зображені на рисунках 2.5, 2.6.

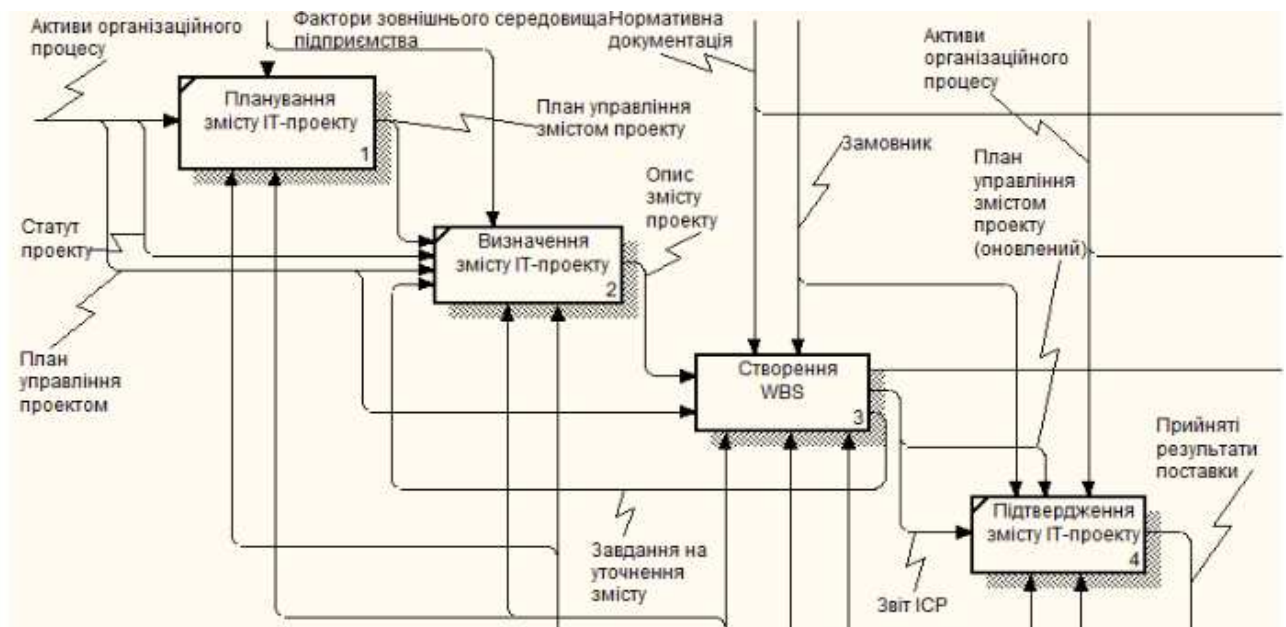


Рисунок 2.4. IDEF процесу управління IT-проектом

2.3. Рішення з програмного забезпечення

2.3.1. Вибір інструментальних засобів

З метою розробки необхідного програмного забезпечення автоматизованої системи та необхідності застосування вікі-сервера, сервера баз даних та сховища для збереження програмних проєктів IT-компанії та інструментального програмного забезпечення розглянемо можливості використання наявного системного програмного забезпечення. Маніпулюючи великими обсягами даних, виникає необхідність використовувати бази даних.



свої дані по різному, для доступу до них є спеціальна мова – SQL (структурована мова запитів) [2]. Не зважаючи на те, що в різних системах управління базами даних мова має свої тонкощі, основні команди SQL однакові для всіх систем управління базами даних.

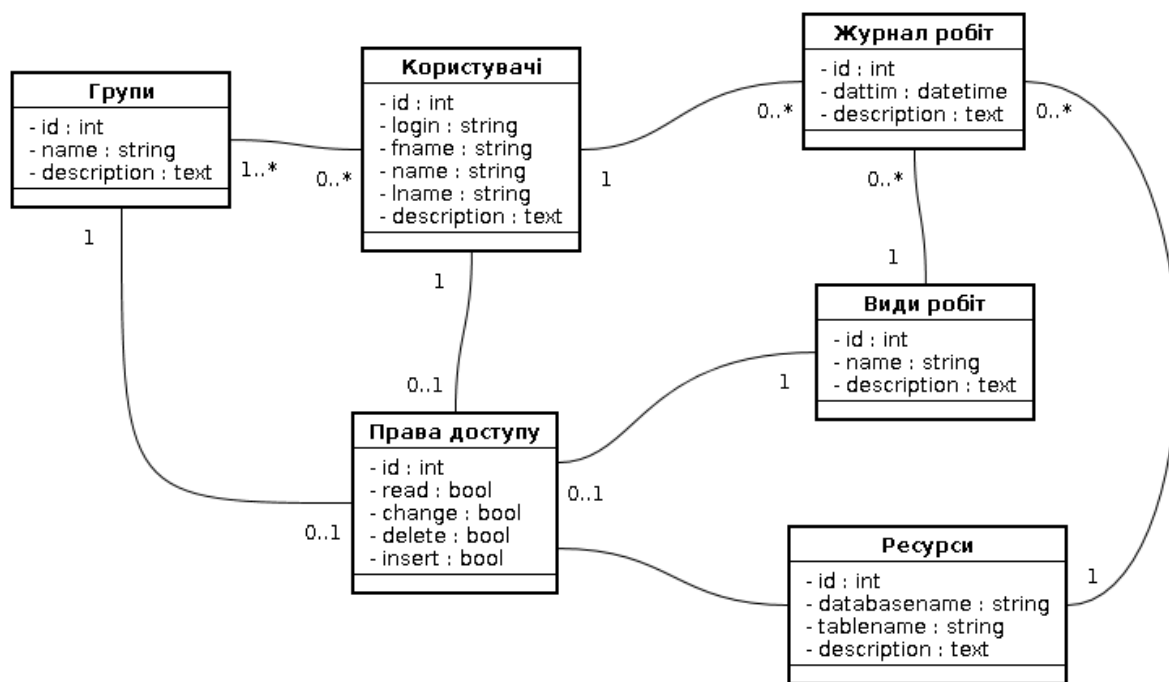


Рисунок 2.6. Діаграма моделі даних системи в частині засобів обліку робіт системи управління проєктами ІТ-компанії

ХАМРР (аббревіатура: Х – будь-яка з чотирьох операційних систем, А – Apache, М - MySQL, Р - PHP, Р – Perl) – кросплатформена збірка веб-сервера, що містить Apache, MySQL, інтерпретатор скриптів PHP, мову програмування Perl і велику кількість додаткових бібліотек, що дозволяють запустити повноцінний веб-сервер. ХАМРР працює з усіма 32-х розрядними ОС Microsoft (98/2000/XP/2003/Vista/7), а також з Linux, Mac OS X і Solaris. Програма вільно поширюється згідно ліцензії GNU General Public License і є безкоштовним, зручним у роботі веб-сервером, здатним обслуговувати динамічні сторінки. Локальний сервера ХАМРР простий в установці.

Програма містить приклади на PHP та Perl, що дозволяє вивчити деякий функціонал цих мов.

PhpMyAdmin – веб-додаток з відкритим кодом на мові PHP з графічним веб-інтерфейсом для адміністрування СКБД MySQL. PhpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Програма дозволяє керувати СКБД MySQL без безпосереднього вводу SQL команд, тому що має зручний інтерфейс. Керування можливе з будь-якого комп'ютера, під'єданого до інтернету, без необхідності встановлення додаткового програмного забезпечення.

Програма розповсюджується під ліцензією GNU General Public License і тому деякі інші розробники інтегрують його у свої розробки, наприклад XAMPP, Denwer.

PHP (англ. PHP: Hypertext Preprocessor – гіпертекстовий препроцесор, попередня назва: Personal Home Page Tools) – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. На додачу, PHP – проєкт з пакету відкритого програмного забезпечення.

Головним фактором забезпечення переваг і популярності мови PHP є практичність. PHP надає засоби для швидкого та ефективного вирішення поставлених завдань. Практичний характер PHP обумовлений такими важливими характеристиками, як традиційність, простота і ефективність.

Традиційність PHP визначається тим, що код PHP дуже схожий на той, що зустрічається в типових програмах на C або Pascal. Багато конструкцій мови запозичені з C, Perl. Це помітно знижує початкові зусилля при вивченні PHP. PHP – мова, що поєднує переваги Perl і C і спеціально націлена на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. На даний момент є однією з найпопулярніших мов для створення веб-додатків (скриптів).

Простота PHP базується на великому різноманітті бібліотек PHP. Сценарій PHP може складатися з 10 000 рядків, або з одного рядка – залежить від специфіки завдання. Немає необхідності довантажувати бібліотеки, вказувати спеціальні параметри компіляції тощо. PHP – мова, яка може бути вбудована безпосередньо в HTML-код сторінок, які, в свою чергу, будуть коректно оброблятися PHP-інтерпретатором. Велике різноманіття функцій PHP позбавляє від написання багаторядкових функцій.

Ефективність PHP є винятково важливим фактором при програмуванні для багатокористувацьких середовищ, до числа яких належить і веб. Дуже важлива перевага PHP полягає в його «движку».

2.3.2. Розробка моделей програмного забезпечення АРМ системи

Враховуючи склад модулів, розробимо моделі програмного забезпечення автоматизованої системи. Модель переходів станів є динамічною моделлю системи і дозволяє визначити поведінку об'єктів системи у часі. Модель переходів станів представлена на рисунку 2.6.

Модель потоків даних АРМ Адміністратора системи управління проєктами ІТ-компанії представлена на рисунку 2.7.

Модель структури програмного забезпечення автоматизованої системи є статичною моделлю системи і дозволяє визначити об'єкти системи і відношення між ними. Модель структури програмного забезпечення представлена на рисунку 2.8.

Також важливою з точки зору забезпечення цілісності системи є механізм аутентифікації та авторизації користувачів в системі. Останнім часом найбільш поширеними є два таких механізми: на рівні додатків і на рівні серверів баз даних [2].

Також слід зазначити, що логічна модель системи, розроблена для реляційної моделі даних, не відповідає повною мірою можливостям сучасних ОО-мов програмування, бо отримана на її основі фізична модель даних не може бути об'єктно-реляційною. З метою застосування ОО-технологій на рівні програмного забезпечення і на рівні структур даних бази даних в

системі використано ORM-технологію, яка дозволяє повною мірою реалізувати потенціал ОО-технологій в тому числі і на рівні моделей даних сервера баз даних [5, 6].

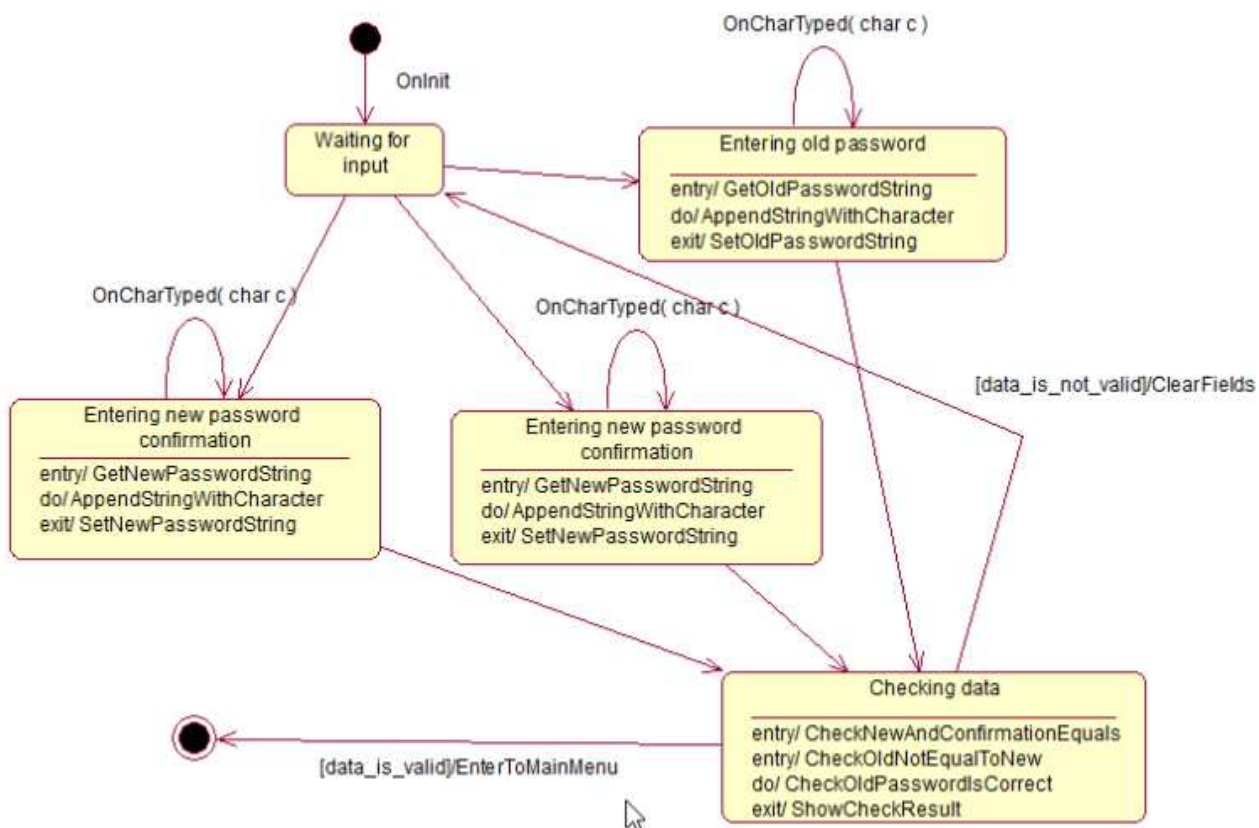


Рисунок 2.6. Діаграма переходів станів аутентифікації АРМ
Адміністратора системи управління проектами ІТ-компанії

2.3.3. Технології доступу до даних

Суттєвою проблемою розробки та експлуатації (в тому числі – розширення функціональності) корпоративних автоматизованих інформаційних систем є складність структур даних, які складаються в результаті розвитку таких систем. Як наслідок, розширення функціональності корпоративних інформаційних систем часто супроводжується змінами в структурах даних бази даних, що для складних інформаційних систем є нетривіальною задачею і часто потребує суттєвої переробки програмного забезпечення таких систем, яке використовує такі дані і структури.



Рисунок 2.7. Діаграма потоків даних АРМ системи управління проєктами ІТ-компанії

Складність вирішення задачі реорганізації даних в системах з реляційною базою даних (яких більшість) тим значніше зростає в наслідок невідповідності методів організації даних в програмах, що використовують базу даних – вони звичайно розробляються відповідно до методологій об’єктно-орієнтованого програмування – та методів організації даних в необ’єктних сховищах даних, що загалом призводить до необхідності виконання над базами даних вкрай складних запитів. Розробка таких запитів на складних структурах даних є невиправдано трудомісткою задачею та, в окремих випадках, вимагає створення допоміжних таблиць даних (*Temporary tables*) або видів (*Views*), що, в свою чергу, також ускладнює використання і обслуговування такої бази даних.

Все це разом визначає складність не тільки створення, але і проєктування корпоративних інформаційних систем і їх сховищ даних.

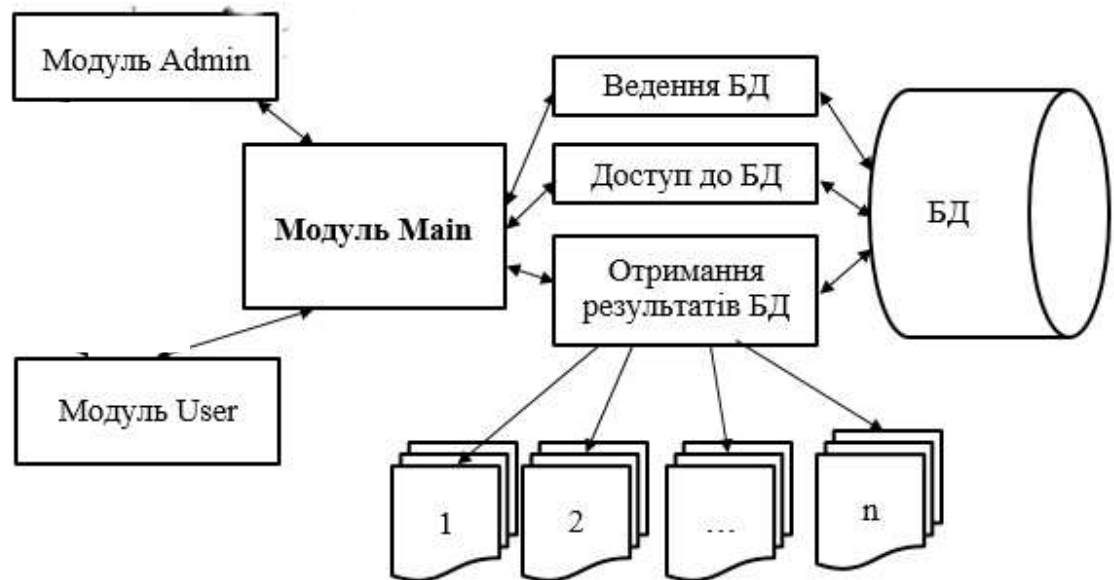


Рисунок 2.8. Модель структури програмного забезпечення автоматизованої системи в частині доступу до даних

Зазначена проблема могла б бути подолана за рахунок використання об'єктно-орієнтованих систем управління базами даних (СУБД), які б дозволили повною мірою задовольнити потреби щодо організації даних з боку об'єктно-орієнтованого програмного забезпечення, але, на жаль, об'єктно-орієнтовані СУБД на сьогодні не отримали широкого розповсюдження – найперше, через свою складність, обмежену підтримку та надмірну вартість [8].

З метою подолання означених проблем останнім часом широко використовують об'єктно-орієнтовані надбудови над реляційними базами даних (*ORM, Object Relational Mapping*), які забезпечують відповідність організації даних на рівні програми та на рівні структур даних. Використання *ORM* надає розробнику можливість маніпулювати даними на рівні об'єктів та абстрагуватись від особливостей реалізації структур даних на конкретному сервері баз даних [7]. Це також дозволяє відносно простими засобами

отримувати зрозумілий і якісний об'єктно-орієнтований код та спрощує його тестування модифікацію та супровід.

Деякі пакети вирішують цю проблему, надаючи бібліотеки класів, здатних виконувати перетворення даних автоматично. Маючи список таблиць в базі даних і об'єктів у програмі, вони автоматично перетворюють запити з одного виду в інший. У результаті запиту генеруються і виконуються необхідні запити, а результати перетворюються в об'єкти відповідного типу.

З точки зору програміста *ORM*-система повинна виглядати як постійне сховище об'єктів. На практиці всі *ORM*-системи, зазвичай, виявляють себе саме такими в тому чи іншому вигляді, забезпечуючи, таким чином, можливість ігнорування реляційної природи бази даних як набору таблиць та відношень між ними.

Враховуючи, що програмне забезпечення для складних систем звичайно розробляється відповідно до принципів об'єктно-орієнтованого програмування, необхідність використання *ORM*-систем виникає в наслідок невідповідності способів організації структурованих даних реляційних систем управління базами даних способам організації об'єктів як структурованих даних та механізмів їх обробки в програмах, які використовують такі бази даних. Ця невідповідність полягає в тому, що об'єкти програми можуть знаходитись у відношеннях агрегації, асоціації та використання, а реляційні бази даних підтримують лише відношення асоціації, причому, відношення асоціації багато-до-багатьох реляційними базами даних не підтримується.

Використання *ORM*-технології має і негативні риси, насамперед, через деяке зменшення пропускнуєї можливості сервера баз даних. Більш того, шар транзакцій при цьому виявляється більш повільним і неефективним (особливо в термінах згенерованого запиту, записаного мовою запитів – *SQL*).

Взагалі, для проєктів, які характеризуються високим навантаженням на сховища даних, питання отримання необхідної продуктивності гостро стоїть і без застосування *ORM*. Але для таких систем також мають велике значення і

отримання відносно простих можливостей розширення системи, а саме такі можливості і забезпечуються використанням *ORM*.

Таким чином, використання *ORM* може призвести до того, що програми будуть працювати повільніше і використовувати більше пам'яті, ніж програми, написані «вручну», але *ORM* позбавляє розробника необхідності написання великої кількості коду, часто одноманітного і схильного до помилок, тим самим значно підвищуючи швидкість розробки. Крім того, більшість сучасних реалізацій *ORM* дозволяють жорстко визначати код SQL запитів, який буде використовуватися при тих чи інших діях (збереження в базу даних, завантаження, пошук тощо) з постійним об'єктом.

Впровадження *ORM* звичайно призводить до деградації продуктивності системи на близько 30%.

Маючи на увазі швидкий прогрес в підвищенні продуктивності обчислювальних систем (близько 30% – 50% на рік) та тенденції до зменшення вартості обладнання, можна розраховувати на те, що протягом року недоліки системи, пов'язані з деградацією обчислень внаслідок використання *ORM* будуть повністю нівельовані за рахунок підвищення продуктивності апаратного забезпечення. При цьому переваги від впровадження *ORM*, пов'язані з можливостями більш простого розширення структур даних, залишаться в повній мірі і надалі.

2.3.4. Тестування програмного забезпечення

Тестування є невід'ємною частиною проєктування програмного забезпечення [9]. Тестування програми – це процес виконання програми з метою виявлення помилок. Кроки процесу задаються тестами.

Існує два принципи тестування програми:

- функціональне тестування (тестування "чорного ящика");
- структурне тестування (тестування "білого ящика").

При тестуванні "чорного ящика" розглядаються системні характеристики програм, ігнорується їх внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Об'єктом тестування "білого ящика" є не зовнішня, а внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їх взаємодії. Зазвичай аналізуються управляючі зв'язки елементів, інколи – інформаційні зв'язки. Тестування за цим принципом характеризується ступінню, в якій тести виконують або покривають логіку (вхідний текст) програми. Вичерпне тестування також викликає певні труднощі.

В результаті тестування за принципом "білого ящика" перевірена коректність побудови всіх елементів програми та правильність їх взаємодії.

Були проаналізовані умови в операторах умовного вибору, циклах, параметри модулів. Виявлено, що:

- в текстах програм всіх програмних модулів не використовуються неініціалізовані змінні;
- у програмі немає описів елементів, змінних, функцій тощо, що не використовуються у тексті;
- відсутні фрагменти коду, що ніколи не виконуються;
- умови в умовних операторах, операторах вибору та циклічних операторах поставлені коректно;
- складні логічні вирази побудовані коректно і відповідають задачам, що пред'являються до програмного комплексу;
- не існує можливостей не завершення ітераційних циклів.

Проведені випробування показали, що розроблене програмне забезпечення функціонує нормально.

2.4. Рішення з технічного забезпечення

Відповідно до оцінок навантаження на систему, для функціонування серверної частини системи необхідні засоби, що відповідають таким мінімальним вимогам:

- Апаратні:
 - ЦП 4 ГГц
 - 4Гб ОЗУ

- Жорсткий диск з 1 Тб вільного простору
- Програмні
 - ОС Ubuntu Server 24.04 LTS
 - Веб-сервер Apache з підтримкою mod_python
 - Сервер баз даних MySQL

Для функціонування клієнтської сторони необхідна наявність з'єднання з локальною мережею підприємства з підтримкою TCP-протоколу і наявність веб-браузера на клієнтському робочому місці.

2.5. Рішення з організаційного забезпечення

Виходячи з призначення системи необхідно забезпечити віддалений доступ менеджерів та керівників проєктів до проєктного середовища та можливості взаємодії учасників виробничого процесу між собою. Як видно зі схеми інформаційних потоків системи (рис. 1.2, 2.2) менеджер взаємодіє з інструментальними засобами автоматизованої системи. Для реалізації його діяльності відповідно до мети автоматизації необхідно змінити структуру та інформаційні потоки діючої системи таким чином, щоб реалізувати взаємодію менеджерів та інших працівників компанії через автоматизовану систему. При цьому необхідно внести зміни в структуру системи управління та в схему інформаційних потоків (Рисунок 2.9).

Зміни в структурі і складі інформаційних потоків потрібно відобразити в наказах про зміну посадових обов'язків для вище зазначених виконавців.

2.6. Висновки до другого розділу

Розроблений проєкт АРМ менеджера програмних проєктів АІС ІТ-компанії дозволить пов'язати в єдиний інформаційний простір інформаційні ресурси професійної підготовки та інструментальні засоби розробки програмного забезпечення. Програмне забезпечення автоматизованої системи може бути корисним у випадках, коли вже є напрацьовані раніше документи, які необхідно перекодувати та зв'язати з іншими документами згідно до вимог вікіфікації описаного вище ресурсу ІТ-компанії.



Рисунок 2.9. Організаційна модель процесів розробки програмного забезпечення

3. МОДЕЛЬ РОЗГОРТАННЯ ПІДСИСТЕМИ АРМ МЕНЕДЖЕРА ПРОЄКТІВ

Модель розгортання підсистеми представимо як для невеликої ІТ-компанії Soft-service (Рисунок 3.1).

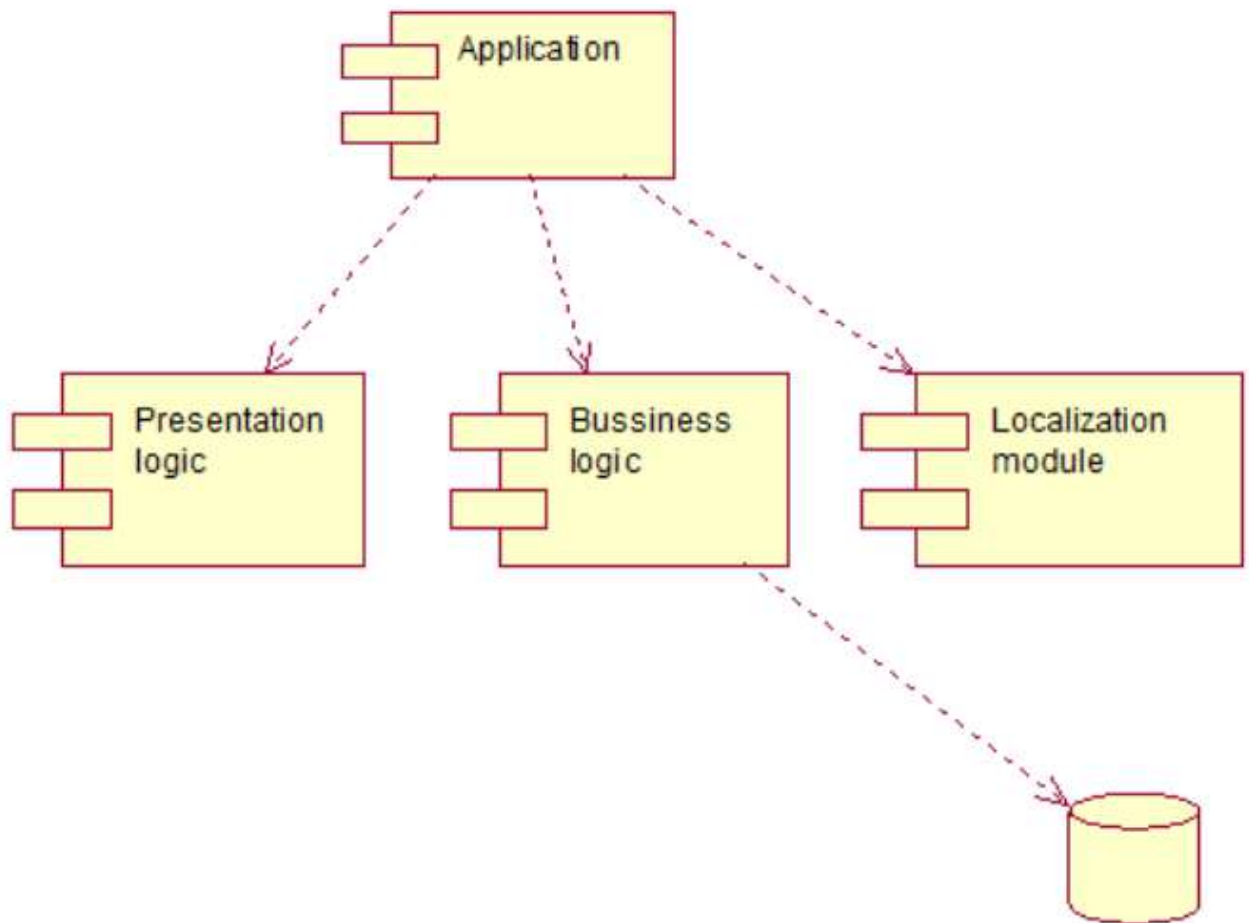


Рисунок 3.1. Модель розгортання підсистеми

Інформаційна система розроблена як веб-сервіс мовами програмування Python та SQL. Програма використовує базу даних під управлінням СКБД MySQL.

Тестування даної програмної системи показало, що програма успішно справляється з поставленими перед нею задачами з інформування відвідувачів системи та покращення роботи адміністратора.

Робота із програмою здійснюється з використанням зручного інтерфейсу користувача (Рис. 3.2).

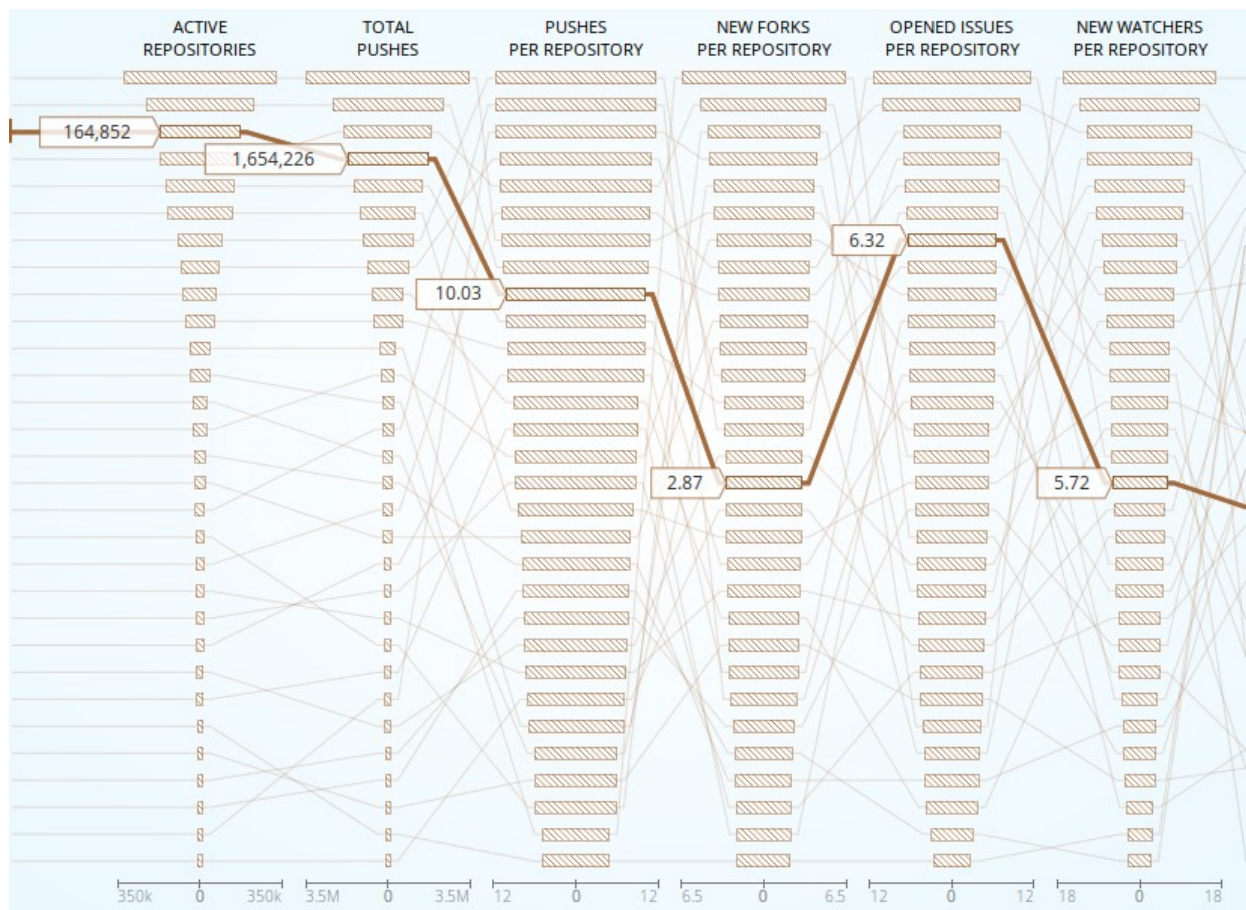


Рисунок 3.2. Приклад реалізації інтерфейсу АРМ

Загальні висновки по роботі

В результаті виконання даної дипломної роботи було розроблено АРМ менеджера програмних проєктів АІС ІТ-компанії. Розроблене програмне забезпечення задовольняє вимогам, які поставлені у технічному завданні.

Програмне забезпечення АРМ системи призначене для підвищення ефективності операційних процесів працівників ІТ-компанії та для підвищення якості процесів інформаційного обміну та комунікацій користувачів системи в частині обліку розрахунків за житлово-комунальні послуги шляхом автоматизації процесів ведення обліку надходжень за надані послуги та витрат на поточні заплановані або незаплановані роботи, визначення боржників і планування фінансових потоків.

Програмне забезпечення підсистеми розроблене мовою програмування Python 3 з використанням фреймворку Django.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вайсман, В.О. Моделі, методи та механізми створення і функціонування проектно-керованої організації: Монографія [Текст] / В.О. Вайсман – К. : Наук. світ, 2009. – 146 с.
2. Гайна, Г.А. Основи проектування баз даних / Навчальний посібник – К. : Кондор, 2024. – 280 с.
3. ДСТУ 3008:2015 – “Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання”
4. ДСТУ ISO/IEC 15910:2012 "Інформаційні технології. Документація для програмного забезпечення, облаштованого інтерфейсом користувача"
5. Заверач, М.М. Бази даних. Інформаційні системи [Текст]: навч. посіб. / М. М. Заверач,, В. В Третько. – Хмельницький : ХНУ, 2007, – 303 с.
6. Зарицька, О. Л. Бази даних та інформаційні системи: методичний посібник. – Житомир: Вид-во ЖДУ ім. І. Франка, 2010. – 132 с.
7. Організація реляційних баз даних : навч. посіб. / Я.Г. Куваєв, О.А. Жукова, І.А. Сечкін – 2-ге вид., допов. та переробл. – Дніпро : НГУ, 2017. – 157 с.
8. Пасічник, В.В., Литвин, В.В., Шаховська, Н.Б. Проектування інформаційних систем. Навчальний посібник (затв. МОН України) – Львів: 2013. – 380 с.
9. Томашевський, О. М. Інформаційні технології та моделювання бізнес-процесів / Навчальний посібник / О. М. Томашевський, М. Б. Цегелик, Г. Г. Вітер, В.І. Дубук. К.: Центр учбової літератури, 2005.– 296 с.
10. Управління проектами з відкритим кодом / Електронний ресурс. URL: https://www.odoo.com/uk_UA/app/project

11. Управління проєктами в ODOO / Електронний ресурс. URL: <https://ubmsoft.com.ua/crm/proekt>
12. Цибко, Г. Ю., Рамський, Ю. С. Проєктування й опрацювання баз даних : Посібник для вчителів. 2-ге вид., перероб. і доп. Тернопіль : Навчальна книга. Богдан, 2005.
13. Agile Web Development with Rails / Third Edition Sam Ruby, Dave Thomas, David Heinemeier Hansson, The Pragmatic Programmers LLC, Printed in the United States of America printing, June 4, 2008 printing, June 4, 2008.
14. Bart Voorn, Marieke L. van Genugten, Sandra van Thiel. The efficiency and effectiveness of municipally owned corporations: a systematic review // Електронний ресурс. URL: <https://www.tandfonline.com/doi/full/10.1080/03003930.2017.1319360>
15. Martin Fowler. Patterns of Enterprise Application Architecture. Addison Wesley Longman, Reading, MA, 2003.
16. Unified Modeling Language User Guide, The (2 ed.). Addison-Wesley. 2005. p. 496.

ДОДАТОК А.

ТЕХНІЧНЕ ЗАВДАННЯ

А.1. Призначення розробки

Назва – «Проект автоматизованої інформаційної системи управління програмними проєктами ІТ-компанії».

Розроблювана автоматизована інформаційна система призначена для збереження і надання інформації про програмні проєкти, що виконуються і ІТ-компанії в рамках її діяльності. Функціональне призначення системи – підвищити швидкість та якість роботи з програмними проєктами шляхом забезпечення кращої поінформованості учасників виробничого процесу компанії.

А.2. Вимоги до автоматизованої системи

Для забезпечення ефективної роботи інформаційна система повинна мати зрозумілий користувачеві інтерфейс, зрозуміле призначення функціональних частин інтерфейсу і наочний результат роботи системи; надійно працювати в умовах експлуатації, володіти можливостями захисту від збоїв і ушкодження даних.

Система повинна бути відкритою для подальшого вдосконалення й збільшення функціональних можливостей.

А.3. Вимоги до функціональності системи

Інформаційна система повинна бути побудована у вигляді взаємозалежних модулів, де кожний модуль відповідає за своє обмежене завдання, що полегшить роботу з потоками даних.

Також має бути виділена вхідна та вихідна інформація.

Вхідною інформацією є:

- Дані про операційні проекти;
- Дані про допоміжні (тестові та навчальні проекти);
- Дані про інструментальне програмне забезпечення та його призначення.

Вихідною інформацією є:

- Дані про заплановані до виконання, поточні, виконані та передані замовнику проекти;
 - Дані про методичні матеріали (тестові та навчальні проекти);
- Дані про інструментальне програмне забезпечення та його призначення.

A.4. Вимоги до надійності й безпеки

ПЗ повинне забезпечувати надійну роботу з даними. Необхідна можливість збереження резервних копій даних на випадки непередбачених збоїв в апаратному або програмному забезпеченні комп'ютерної системи, на базі якої буде працювати програма.

Надійність технічних засобів системи має підтримуватись технічним обслуговуванням (виконання профілактичних робіт і ремонту) технічних засобів.

A.5. Умови експлуатації

Програмний продукт автоматизованої системи повинен мати простий і зрозумілий інтерфейс.

Перед початком роботи з даним програмним продуктом необхідно ознайомитись з інструкцією використання програмного продукту.

Умови експлуатації повинні відповідати умовам експлуатації апаратного забезпечення. Умови експлуатації носіїв інформації відповідати занесеним у їхню супровідну документацію.

A.6. Вимоги до технічних засобів

Для установки й експлуатації програмного продукту необхідний персональний комп'ютер, що володіє системними характеристиками не гіршими, ніж:

Процесор: не менше Pentium 2ГГц.

Пам'ять: не менше 4 Gb;

Відеокарта: не менше 64 Mb;

Маніпулятори: миша, клавіатура.

A.7. Вимоги до інформаційної й програмної сумісності

Для роботи програмного забезпечення системи необхідно:

- Інтерпретатор Python версії 3.10 або вище;
- СУБД MySQL або MariaDB версії 5 або вище;
- Веб-фреймворк Django.

A.8. Вимоги до програмної документації

Програмна документація повинна бути розроблена в наступному складі:

- технічне завдання;
- інструкція користувача.

A.9. Стадії й етапи розробки

Стадії та етапи розробки з потрібними строками вказані в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадії	Етапи роботи	Початок робіт	Закінчення робіт
1. Ескізний проєкт	Розробка ескізного проєкту:	13.03.2025	04.03.2025
	Затвердження ескізного проєкту	04.04.2025	28.04.2025
2. Технічний проєкт	Розробка технічного проєкту:	28.04.2025	12.05.2025
	Затвердження технічного проєкту	12.05.2025	17.05.2025
3. Робочий проєкт	Розробка програми:	17.05.2025	23.05.2025
	Розробка програмної документації	23.05.2025	01.06.2025
	Випробування програми	01.06.2025	14.06.2025

A.10. Порядок контролю та приймання

Для контролю и прийому повинен бути наданий опис програмного забезпечення, а також програма і методика випробування.

Порядок контролю и прийому даної розробки здійснюється представником замовника у присутності представника розробника згідно з програмою та методикою випробувань.

Якщо програмне забезпечення не пройшло випробування, виконавець зобов'язаний виправити помилки та недоліки у строк не більш ніж 1 місяць зі дня випробування.

За результатами прийому складається акт, який підписується представником замовника та представником розробника та утверджується керівниками організації-замовника і організації-розробника.

У випадку виявлення помилок під час прийому програмного виробу складається акт про виявлені помилки, який підписується представниками замовника та розробника й стверджується керівниками організації-замовника й організації-розробника. Розробник повинен впродовж не більш як 1 місяця видалити вказані зауваження й повідомити Замовника про повторне проведення перевірки не пізніше як за 2 тижня до початку прийому програмного виробу.

ДОДАТОК Б. Код програмного забезпечення підсистеми (фрагмент)

Конфігурація системи

```
USE_I18N = True
```

```
STATIC_DOC_ROOT = os.path.join(os.path.dirname(__file__), 'wt/static').replace("\\", '/')
```

```
COUNTRIES_FLAG_PATH = 'img/flags/%s.gif'
```

```
MEDIA_ROOT = ''
```

```
MEDIA_URL = '/static/'
```

```
ADMIN_MEDIA_PREFIX = '/media/'
```

```
SECRET_KEY = ''
```

```
TEMPLATE_LOADERS = (
```

```
    'django.template.loaders.filesystem.load_template_source',
```

```
    'django.template.loaders.app_directories.load_template_source',
```

```
)
```

```
MIDDLEWARE_CLASSES = (
```

```
    'django.middleware.common.CommonMiddleware',
```

```
    'django.contrib.sessions.middleware.SessionMiddleware',
```

```
    'django.contrib.auth.middleware.AuthenticationMiddleware',
```

```
    'wps.middleware.django403.middleware.Django403Middleware',
```

```
    'wps.middleware.RequireLoginMiddleware.RequireLoginMiddleware',
```

```
    'wps.middleware.RequirePermissionMiddleware.RequirePermissionMiddleware',
```

```
)
```

```
TEMPLATE_CONTEXT_PROCESSORS = (  
    'django.core.context_processors.auth',  
    'django.core.context_processors.debug',  
    'django.core.context_processors.i18n',  
    'django.core.context_processors.media',  
    'django.core.context_processors.request',  
)
```

```
ROOT_URLCONF = 'wps.urls'
```

```
TEMPLATE_DIRS = (  
    os.path.join(os.path.dirname(__file__), 'wt/templates').replace('\\', '/'),  
)
```

```
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
    'django.contrib.admin',  
    'wps.wt',  
)
```

```
SESSION_EXPIRE_AT_BROWSER_CLOSE = True
```

```
LOGIN_REQUIRED_URLS = (  

```



```
    r'/(.*)$',  
)
```

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (  
    r'/accounts/login(.*)$',  
)
```

```
AUTH_PROFILE_MODULE = 'wt.Account'
```

```
RESTRICTED_VIEWS = (  
    (r'wps.wt.form_views.group_add', 'auth.add_group'),  
    (r'wps.wt.form_views.group_edit', 'auth.change_group'),  
    (r'wps.wt.form_views.group_delete', 'auth.delete_group'),  
    (r'wps.wt.views.(w+)_1.*', 'wt.can_view_%s'),  
    (r'wps.wt.views.(w+)_$', 'wt.can_view_%s'),  
    (r'wps.wt.form_views.(w+)_add', 'wt.add_%s'),  
    (r'wps.wt.form_views.(w+)_edit', 'wt.change_%s'),  
    (r'wps.wt.form_views.(w+)_delete', 'wt.delete_%s'),  
    (r'wps.wt.views.(w+)_add', 'wt.add_%s'),  
    (r'wps.wt.views.(w+)_edit', 'wt.change_%s'),  
    (r'wps.wt.views.(w+)_delete', 'wt.delete_%s'),  
    (r'wp.wt.views.(w+)by.*', 'wt.can_view_%s'),  
    (r'.*', None),  
)
```

```
RESTRICTED_VIEWS_EXCEPTIONS = (  
    r'wps.wt.views.group_',
```

```
r'wps.wt.views.group_1',  
r'wps.wt.form_views.object_delete_wrapper',  
r'wps.wt.views.index',  
r'[^(ps.wt)].*',  
)
```

Код моделей даних

```
# -*- coding: utf-8 -*-
```

```
from django.db import models
```

```
from django.contrib.auth.models import User, UserManager
```

```
from django.db.models import Avg, Max, Min, Count, Sum
```

```
from django.db.models import F
```

```
from datetime import datetime, date, time, timedelta
```

```
from utils import DateTime
```

```
class Account(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default='')
```

```
    description = models.TextField(verbose_name=u'Описание')
```

```
    teatcher = AdvancedForeignKey('Account', 'Manager')
```

```
    user = models.ForeignKey(User, unique=True, related_name='profile')
```

```
    def __unicode__(cls):
```

```
        return cls.user.username
```

```
    @property
```

```
    def name(self):
```

```
        return self.user.username
```

```
    @property
```

```
    def count_work(self):
```

```
        return Work.count({'account__id' : self.id})
```

```

@property
def duration_work(self):
    """ calc work duration in hours """
    return Work.sum_duration({'account__id' : self.id})

```

```

@property
def role(self):
    """ role (group) """
    query_set = self.user.groups.all()
    return query_set.all()[0] if query_set.count() > 0 else None

```

```

class Manager(models.Model, _Base):
    enable = models.CharField(max_length=1, default='y')
    icon = models.CharField(blank=True, max_length=60, default='')
    fname = models.CharField(max_length=100, unique=True,
verbose_name=u'Прізвище')
    name = models.CharField(max_length=100, verbose_name=u'Ім'я')
    lname = models.CharField(max_length=100, verbose_name=u'ПоБат.')
    phone1 = models.CharField(max_length=100, verbose_name=u'Тлф1')
    phone2 = models.CharField(max_length=100, verbose_name=u'Тлф2')
    email = models.CharField(max_length=50, verbose_name=u'eMail')
    skype = models.CharField(max_length=50, verbose_name=u'Скайп')
    born = models.DateField(verbose_name='Д.Н.')
    department = AdvancedForeignKey('Teatcher', 'Department',
verbose_name=u'Дкпартамент')
    post = AdvancedForeignKey('Teatcher', 'Post', verbose_name=u'Посада')
    description = models.TextField(verbose_name=u'Прим.')

```

```

class Meta:

    ordering = ["fname"]

    verbose_name = u'Менеджер'

def __unicode__(cls):

    return "%s %s %s" % (cls.fname, cls.name, cls.lname)

@property
def program_count(self):

    return Program.count({'teatcher__id' : self.id})

@property
def fio(self):

    return "%s %c. %c." % (self.fname, self.name[0], self.lname[0])

@property
def bay_debt(self):

    return self.bay_summ - Productionpayment.sum('pay',
{'productionorder__customer__id' : self.id})

@property
def sale_count(self):

    return Productorder.count({'customer__id' : self.id})

@property
def sale_summ(self):

```

```
return Productstructure.calculated({'productorder__customer__id' : self.id})
```

```
@property
```

```
def sale_debt(self):
```

```
    return Productpayment.sum('pay', {'productorder__customer__id' : self.id}) -  
self.sale_summ
```

```
class Department(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default='')
```

```
    name = models.CharField(max_length=100, unique=True,  
verbose_name=u'Назва')
```

```
    short_name = models.CharField(max_length=20, unique=True,  
verbose_name=u'Наз.')
```

```
    description = models.TextField(verbose_name=u'Опис')
```

```
class Meta:
```

```
    ordering = ["name"]
```

```
    verbose_name = u'Департамент'
```

```
def __unicode__(cls):
```

```
    return cls.name
```

```
@property
```

```
def teatchr_count(self):
```

```
    return Manager.count({'department__id' : self.id})
```

```
@property
```

```
def program_count(self):  
    return Program.count({'teatcher__department__id' : self.id})
```

```
class Direction(models.Model, _Base):  
    enable = models.CharField(max_length=1, default='y')  
    icon = models.CharField(blank=True, max_length=60, default='')  
    kode = models.CharField(max_length=20, unique=True, verbose_name=u'Код')  
    name = models.CharField(max_length=100, verbose_name=u'Назва')  
    description = models.TextField(verbose_name=u'Опис')
```

```
class Meta:  
    ordering = ["name"]  
    verbose_name = u'Напрям'
```

```
def __unicode__(cls):  
    return cls.name
```

```
@property  
def program_count(self):  
    return Program.count({'direction__id' : self.id})
```

```
@property  
def studgroup_count(self):  
    return Studgroup.count({'special__direction__id' : self.id})
```

```
@property  
def special_count(self):
```

```
    return Special.count({'direction__id' : self.id})
```

```
@property
```

```
def specialisation_count(self):
```

```
    return Specialisation.count({'special__direction__id' : self.id})
```

```
class Special(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default='')
```

```
    kode = models.CharField(max_length=20, unique=True, verbose_name=u'Код')
```

```
    name = models.CharField(max_length=100, verbose_name=u'Назва')
```

```
    direction = AdvancedForeignKey('Special', 'Direction', verbose_name=u'Напря́м')
```

```
    description = models.TextField(verbose_name=u'Опис')
```

```
class Meta:
```

```
    ordering = ["name"]
```

```
    verbose_name = u'Спеціальність'
```

```
def __unicode__(cls):
```

```
    return cls.name
```

```
@property
```

```
def group_count(self):
```

```
    return Studgroup.count({'special__id' : self.id})
```

```
@property
```

```
def program_count(self):
```



```
return Program.count({'special__id' : self.id})
```

```
@property
```

```
def duration_work(self):
```

```
    """ calc work duration in hours """
```

```
    return Work.sum_duration({'task__id' : self.id})
```

```
class Specialisation(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default='')
```

```
    kode = models.CharField(max_length=20, unique=True, verbose_name=u'Код')
```

```
    name = models.CharField(max_length=100, verbose_name=u'Назва')
```

```
    special = AdvancedForeignKey('Specialisation', 'Special',  
verbose_name=u'Спеціальність')
```

```
    description = models.TextField(verbose_name=u'Опис')
```

```
class Meta:
```

```
    ordering = ["name"]
```

```
    verbose_name = u'Спеціалізація'
```

```
def __unicode__(cls):
```

```
    return cls.name
```

```
class Project(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default='')
```

```
    kode = models.CharField(max_length=20, unique=True, verbose_name=u'Код')
```

```
name = models.CharField(max_length=100, unique=True,
verbose_name=u'Назва')

short_name = models.CharField(max_length=20, unique=True,
verbose_name=u'Наз.')

description = models.TextField(verbose_name=u'Опис')

class Meta:

    ordering = ["name"]

    #verbose_name = u'Дисципліна'

def __unicode__(self):

    return "%s" % self.name

@property
def program_count(self):

    return Program.count({'object__id' : self.id})

@property
def short_name(self):

    return "%c.%s" % (self.firstname[0], self.surname)

@property
def count_account(self):

    return Account.count({'person__id' : self.id})

@property
def list_account(self):

    account = Account.all({'person__id' : self.id})
```

```
return account
```

```
class Post(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default="")
```

```
    name = models.CharField(max_length=100, unique=True,  
verbose_name=u'Назва')
```

```
    description = models.TextField(verbose_name=u'Опис')
```

```
    def __unicode__(self):
```

```
        return "%s" % self.name
```

```
    @property
```

```
    def teatcher_count(self):
```

```
        return Teatcher.count({'post__id' : self.id})
```

```
    @property
```

```
    def program_count(self):
```

```
        return Program.count({'teatcher__post_id' : self.id})
```

```
class ProjectProgram(models.Model, _Base):
```

```
    enable = models.CharField(max_length=1, default='y')
```

```
    icon = models.CharField(blank=True, max_length=60, default="")
```

```
    create_dat = models.DateField(verbose_name=u'Дата')
```

```
    object = AdvancedForeignKey('Program', 'Object', verbose_name=u'Project')
```

```
    manager = AdvancedForeignKey('Program', 'Manager',  
verbose_name=u'Розробник')
```

```
    target = models.TextField(verbose_name=u'Цільове призначення')
```

```
tasks = models.TextField(verbose_name=u'Задачі')
description = models.TextField(verbose_name=u'Опис')
direction = AdvancedForeignKey('Special', 'Direction', verbose_name=u'Напря́м')
```

```
def __unicode__(self):
    return "%s" % self.object.name
```

```
class User(models.Model, _Base):
    enable = models.CharField(max_length=1, default='y')
    icon = models.CharField(blank=True, max_length=60, default='')
    fname = models.CharField(max_length=100, unique=True,
verbose_name=u'Прізвище')
    name = models.CharField(max_length=100, verbose_name=u'Ім\''я')
    lname = models.CharField(max_length=100, verbose_name=u'ПоБат.')
    description = models.TextField(verbose_name=u'Прим.')
    phone1 = models.CharField(max_length=100, verbose_name=u'Тлф')
    born = models.DateField(verbose_name='Д.Н.')
```

```
class Meta:
    ordering = ["fname"]
    verbose_name = u'Користувач'
```

```
def __unicode__(cls):
    return "%s %s %s" % (cls.fname, cls.name, cls.lname)
```

ДОДАТОК В

Інструкція з експлуатації та супроводу автоматизованої інформаційної системи управління програмними проєктами ІТ-компанії

Автоматизована інформаційна система управління програмними проєктами ІТ-компанії має використовуватись сумісно з іншими автоматизованими системами ІТ-компанії: проєктування, кодування, документування тощо.

В межах системи має забезпечуватись захист веб-ресурсів від несанкційованого доступу шляхом налаштування штатних засобів операційних систем та спеціалізованого серверного програмного забезпечення. Це дозволить досягти необхідного рівня взаємодії систем між собою та уникнути дублювання інформації при документуванні моделей, коду та проєктів в цілому.

Надійне збереження інформації має забезпечуватись адміністратором системи шляхом дублювання всієї інформації системи з обраною періодичністю, можливо — визначивши контрольні точки відповідно до моментів завершення важливих проєктів або їх етапів.

Доступ до функцій перегляду, редагування, створення, видалення або перейменування документів має виконуватись через авторизацію та по захищеним каналам даних. Для цього у веб-браузерах на клієнтських машинах має бути увімкнено підтримку протоколів HTTPS.

