РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет математики та інформатики

Кафедра інформаційно-комунікаційних технологій та
методики викладання інформатики

До захисту допущено»
Завідувач кафедри ІКТ та МВІ

(підпис) (прізвище, ініціали)
«___»___2022р. протокол №_.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка автоматизованої інформаційної системи групового планування та контролю за виконанням завдань

здобувача другого (магістерського) рівня вищої освіти
спеціальності 015 Професійна освіта (за спеціалізаціями)
спеціалізація 015.39 Цифрові технології
Куряти Олександра Сергійовича _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник: _____ завідувач кафедри інформаційно-комунікаційних технологій та методики викладання інформатики РДГУ, професор, доктор педагогічних наук, Войтович Ігор Станіславович
(підпис)

Рецензент: _____
(підпис)

Рецензент: _____
(підпис)

Засвідчую, що у цьому кваліфікаційному проекті немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Рівне – 2022

АНОТАЦІЯ

Курята О.С. «Розробка автоматизованої інформаційної системи групового планування та контролю за виконанням завдань». – Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 015.39 Професійна освіта (Цифрові технології) – Рівненський державний гуманітарний університет – Рівне, 2022. – 79 с.

Кваліфікаційна робота викладена на 79 сторінках, вона містить 3 розділи, 33 ілюстрації, 1 таблицю, 2 додатки та 16 джерел з переліку посилань.

Розроблена автоматизована інформаційна система групового планування та контролю за виконанням завдань призначена для організацій, дозволяє автоматизувати процеси управління проектами, розподіл ресурсів за завданнями проекту.

Програма володіє інтуїтивно зрозумілим інтерфейсом, заснованим на повсюдно поширених технологіях, повністю адаптованим до потреб користувача при роботі з системою.

У процесі виконання дипломної роботи було досягнуто наступних результатів: обрано комплекс технічних, програмних та довідкових засобів, необхідних для роботи над створенням автоматизованої інформаційної системи, спроектовано модель бази даних з урахуванням нормалізації та цілісності даних, побудовано фізичну модель СУБД з визначенням полів та типів даних, реалізовано програмні модулі системи, проведено розрахунок економічних показників системи, що пред'являються при проектуванні та роботі з інтерфейсом користувача.

Ключові слова: автоматизована інформаційна система, система управління проектами, групове планування, контроль завдань.

ABSTRACT

Kuriata O.S. "Development of an automated information system for group planning and task performance control." - Qualifying work for obtaining a master's degree in the specialty 015.39 Professional education (Digital technologies) - Rivne State Humanitarian University - Rivne, 2022. - 79 p.

The qualification work is laid out on 79 pages, contains 3 chapters, 33 illustrations, 1 table, 2 appendices and 16 sources in the list of references.

The developed automated information system of group planning and task performance control is intended for organizations, allows to automate project management processes, allocation of resources according to project tasks, accounting of employees' working hours.

The program has an intuitive interface based on common technologies, fully adapted to the user's needs when working with the system.

The following results were achieved in the process of completing the thesis: a set of technical, software and reference tools necessary for work on the creation of an automated information system was selected, a database model was designed taking into account normalization and data integrity, a physical DBMS model was built with the definition of fields and data types, software modules of the system have been implemented, the economic indicators of the system, which appear during design and work with the user interface, have been calculated.

Keywords: automated information system, project management system, group planning, task control.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АІС – автоматизована інформаційна система.

HTML – Hypertext Markup Language.

CSS – Cascading Style Sheets.

СУБД – система управління базами даних.

ПЗ – програмне забезпечення.

SQL – Structured Query Language.

API – Application Programming Interface.

HTTP – Hypertext Transfer Protocol.

ВСТУП

Актуальність дослідження. Діяльність будь-якої комерційної компанії спрямована на одержання прибутку. Одним із засобів підвищення ефективності роботи, а, отже, і збільшення прибутку, є автоматизація бізнес-процесів компанії. Отже, очевидно, що жодна організація неспроможна нормально функціонувати, якщо у ній не налагоджено грамотне управління її діяльністю. У разі неякісного управління ситуація ще більше посилюється, якщо організація пов'язана з виконанням досить довгострокових проектів, управління якими передбачає виконання проекту (тобто виконання необхідного обсягу робіт із необхідною якістю) в умовах обмежених ресурсів. Очевидно, що основним фактором, який визначає успіх управління проектом, буде чіткий заздалегідь визначений план, що включає строгий графік виконання робіт, розподіл завдань та ресурсів, мінімізацію можливих ризиків. Також очевидно, що складання такого плану є досить трудомістким процесом. У тому випадку, коли йдеться про проекти, в яких завдання тісно пов'язані один з одним, наприклад, початок одного завдання можливий лише при досягненні певного етапу виконання іншого, або з просто великою кількістю завдань, складання плану вручну практично неможливо і його необхідно автоматизувати за допомогою відповідного програмного забезпечення.

Програмне забезпечення для управління проектами, як правило, включає додатки для планування завдань, складання розкладу, контролю ціни та управління бюджетом, розподілу ресурсів, спільної роботи, обміну повідомленнями, швидкого управління, документування та адміністрування системи, яке використовуються спільно для управління великими проектами. Однією з основних функцій такого ПЗ є планування подій та управління завданнями, яке включає: планування різних подій, що залежать один від одного, планування розкладу роботи співробітників та управління ресурсами, розрахунок часу, необхідного на вирішення кожного із завдань, сортування

завдань залежно від термінів їх завершення, управління кількома проектами одночасно.

Крім того, бажано, щоб таке програмне забезпечення надавало можливості управління даними та надання інформації, а саме: список завдань для співробітників та інформацію розподілу ресурсів, інформацію про терміни виконання завдань, ранні попередження про можливі ризики, пов'язані з проектом, інформацію про робоче навантаження співробітників.

Цілком очевидно, що програмне забезпечення такого типу може бути різного ступеня складності, починаючи від сучасних розрахованих на багато користувачів ERP систем, побудованих за технологією клієнт-сервер, до найпростіших однокористувальних систем типу органайзера. В даний час на ринку представлено величезну кількість засобів автоматизації різних бізнес-процесів: CRM - системи управління взаємовідносинами з клієнтами, HRM - системи управління персоналом, SCM - системи управління поставками та багато інших. Великі ERP-системи поєднують у собі функціонал таких модулів-підсистем. Допрацювання та використання ERP-систем під конкретного замовника – дуже трудомісткий процес, яким займаються компанії-інтегратори.

Більше того, як правило, такого роду автоматизовані системи або є вузькоспеціалізованими, або, навпаки, поєднують у собі таку кількість функцій, які дуже обтяжують систему: вона стає дорогою та складною як у налаштуванні, так і у використанні. Тому, незважаючи на уявне перенасичення ринку подібними продуктами, створення простої та надійної системи групового планування та контролю за виконанням завдань досі є важливим завданням у галузі розробки інформаційних систем.

Мета дипломної роботи полягає в розробці автоматизованої інформаційної системи групового планування та контролю за виконанням завдань, яка відповідає таким вимогам:

• Можливість розгортання у локальній мережі організації, побудованої на будь-якій поширеній технології;

• Кросплатформенність – можливість роботи системи під будь-якою поширеною операційною системою (Windows та Linux);

• Низькі системні вимоги та продуктивність – можливість одночасної роботи в системі, запущеній на дешевому, можливо, морально застарілому обладнанні, кількості користувачів, що відповідає невеликій організації;

• Простота підтримки та експлуатації системи за рахунок використання широко поширених технологій та програмного забезпечення.

Об'єкт дослідження – процес створення автоматизованої інформаційної системи групового планування та контролю за виконанням завдань.

Предмет дослідження – автоматизована інформаційна система групового планування та контролю за виконанням завдань для автоматизації процесів управління проектами.

Теоретичне значення дослідження. Проаналізовано літературу по створенню автоматизованих інформаційних систем групового планування та контролю за виконанням завдань.

Практичне значення дослідження полягає в розробці програмного продукту для Рівненського державного гуманітарного університету для їх власних потреб з врахуванням рекомандацій кожного із працівників.

Завдання дослідження:

• вивчити потреби організацій в автоматизації бізнес-процесів;

• проаналізувати аналогічні існуючі системи;

• визначити функціональні вимоги до ПЗ, що розробляється;

• розробити структуру ІС та екранні форми;

• перевірити працездатність створеної ІС.

Система, що розробляється, повинна реалізовувати наступні функції:

• Ведення списку завдань як окремого співробітника, так і структурної одиниці компанії в цілому, розподіл завдань структурної одиниці за її співробітниками;

• Поділ прав користувачів;

• Створення підзадач;

• Відстеження статусу виконання завдань;

• Пошук по завданням;

• Зручні інструменти для адміністрування системи.

Використання такої системи в організаціях дозволить збільшити ефективність роботи персоналу за рахунок:

• Прискорення роботи завдяки єдиному інтерфейсу доступу до необхідної для виконання завдань інформації;

• Підвищення якості спільної роботи різних структур організації у рамках одного проекту;

• Мотивація співробітників, через наочність завдань та терміни їх виконання;

• Підвищення надійності зберігання даних завдяки єдиній базі даних з можливістю резервного копіювання та відновлення.

Апробація результатів роботи. Основні положення та результати магістерського дослідження доповідались та обговорювались на XV Всеукраїнській науково-практичній конференції «Інформаційні технології у професійній діяльності» (1 листопада 2022 року, м. Рівне). Підтверджуючий сертифікат про участь міститься у Додатку А.

За матеріалами магістерського дослідження опубліковано наукову працю: «Розробка автоматизованої інформаційної системи групового планування та контролю за виконанням завдань» у електронному збірнику тез доповідей XV Всеукраїнської науково-практичної конференції «Інформаційні технології у професійній діяльності» [13].

Структура роботи. Дана робота складається з вступу, трьох розділів, висновків і списку використаних джерел.

Вступ дипломної роботи містить обґрунтування актуальності теми, визначені об’єкт і предмет, головна мета та основні завдання дипломної роботи, теоретичну та практичну значимість.

Перший розділ дипломної роботи розкриває сутність характеристики предметної області.

У другому розділі здійснено аналіз обраних інструментальних засобів для розробки автоматизованої інформаційної системи.

Третій розділ присвячений проектуванню та реалізації практичної частини.

Висновки відбивають сучасний стан автоматизованих інформаційних систем групового планування та контролю за виконанням завдань, їх стрімкий розвиток та зміни. Розглянуто представлені на ринку системи групового планування та контролю за виконанням завдань. Розглянуто основні принципи організації системи, що розробляється, обґрунтовано вибір інструментарію розробки системи.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи автоматизації виробництва та управління ресурсами

1.1.1 Роль та місце автоматизованих систем в управлінні організацією

Підвищення якості та ефективності управління - одне з головних завдань більшості компаній. Якісне управління відкриває шлях до реального підвищення конкурентоспроможності та прибутковості, до залучення інвестицій, а отже, до максимальних темпів зростання вартості бізнесу. Для підвищення якості управління менеджменту компанії необхідні сучасні та багатофункціональні інструменти, які дозволяють аналізувати велику кількість економічних параметрів та приймати грамотні управлінські рішення, адекватні ситуації, що склалася на ринку. Одним із таких інструментів є інформаційні системи управління підприємством.

Підприємство можна розглядати як інформаційний центр (рис. 1.1), у якому обробляється інформація, що міститься як у зовнішньому, так і у внутрішньому потоках, тобто реалізується інформаційний процес.

Рис. 1.1. Інформаційні потоки підприємства

Зовнішній потік інформації визначається взаємодією підприємства з економічними та політичними суб'єктами, що діють поза ним. Сюди належить

взаємодія підприємства з клієнтами та конкурентами як реальними, так і потенційними. Внутрішній потік включає інформацію, що описує відносини в колективі співробітників, а також знання, що породжуються у виробництві.

Підприємства мають і формують своє власне внутрішнє інформаційне середовище, в якому циркулюють потоки інформації. Як зовнішні джерела підприємства виступають держава, інформаційні центри та мережі, науково-дослідні організації, постачальники матеріалів, конкуренти, інфраструктура ринку тощо. Вхідний потік підприємства формується на підставі інформації, що надходить із зовнішнього середовища. Вихідний інформаційний потік направляється підприємством у зовнішнє середовище і містить інформацію про свої виробничі можливості, вироблений товар (рекламу), матеріальні, енергетичні, кадрові та інформаційні потреби тощо. Інформаційна система підприємства фільтрує інформаційний потік і виділяє інформацію, необхідну життєдіяльності підприємства, перетворюючи їх у зручну прийняття рішень форму.

Основними завданнями підприємства щодо формування інформаційних потоків є:

• формування адекватних інформаційних ресурсів системи управління підприємством;

• оптимізація інформаційних потоків шляхом виключення дублювання інформації;

• ліквідація розриву між впровадженням інформаційних технологій та техніки та станом інформаційних ресурсів (їх формування та використання).

Структуру будь-якої інформаційної системи, пов'язаної з економічною діяльністю, можна уявити суб'єктом та об'єктом управління (рис. 1.2), де основні інформаційні потоки між зовнішнім середовищем, об'єктом та суб'єктом управління позначені стрілками 1, 2, 3, 4 та підтримуються ІС.

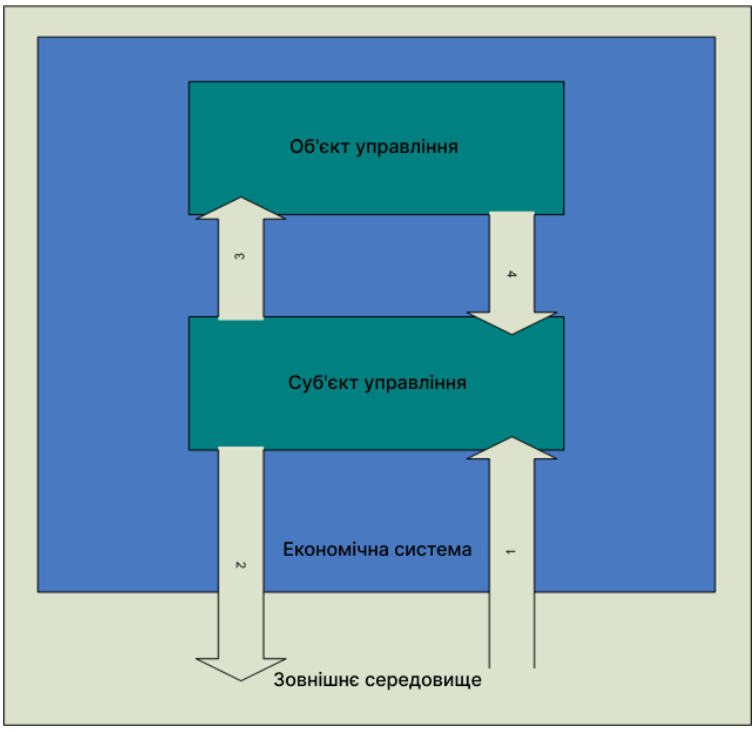

Рис. 1.2. Структура економічної системи

Об'єкт управління є підсистему матеріальних елементів економічної діяльності (сировина і матеріали, обладнання, готова продукція, працівники та ін.) і господарських процесів (основне та допоміжне виробництво, постачання, збут та ін.).

Суб'єкт управління є сукупність взаємодіючих структурних підрозділів економічної системи (дирекція, фінансовий, виробничий, постачальницький, збутовий та інші відділи), що здійснюють такі функції управління:

• планування - визначає мету функціонування економічної інформаційної системи на різні періоди часу (стратегічне, тактичне, оперативне планування);

• облік – відображає стан об'єкта управління внаслідок виконання господарських процесів;

• контроль – фіксує відхилення облікових даних від планових цілей та нормативів;

• регулювання — здійснює оперативне управління всіма господарськими процесами для виключення відхилень, що виникають між плановими та обліковими даними;

• аналіз – визначає тенденції у роботі економічної інформаційної системи та резерви, які враховуються при плануванні на наступний часовий період.

Інформаційна система є сукупність функціональної структури, інформаційного, математичного, технічного, організаційного та кадрового забезпечень, які об'єднані в єдину систему з метою збирання, зберігання, обробки та видачі необхідної інформації для виконання функцій управління. Вона забезпечує інформацією систему управління, формуючи такі інформаційні потоки:

• інформаційний потік 1 — із довкілля до системи управління, з одного боку, є потік нормативної інформації, створюваний державними установам у частині законодавства, з другого боку — потік інформації про кон'юнктуру ринку, створюваний конкурентами, споживачами, постачальниками;

• інформаційний потік 2 — із системи управління у зовнішнє середовище (звітна інформація, передусім фінансова до державних органів, інвесторам, кредиторам, споживачам; маркетингова інформація потенційним споживачам);

• інформаційний потік 3 - із системи управління на об'єкт, являє собою сукупність планової, нормативної та розпорядчої інформації для здійснення господарських процесів;

• інформаційний потік 4 - від об'єкта в систему управління, що відображає облікову інформацію про стан об'єкта управління економічною системою (сировини, матеріалів, грошових, енергетичних, трудових ресурсів, готової продукції та виконаних послуг) в результаті виконання господарських процесів.

Інформаційна система накопичує і переробляє облікову інформацію, що надходить, і наявні нормативи та плани в аналітичну інформацію, що служить основою для прогнозування розвитку економічної системи, коригування її цілей і створення планів для нового циклу відтворення.

1.1.2 Сучасні ERP-системи

Відповідно до Словника APICS, термін «ERP-система» можна використовувати у двох значеннях. По-перше, це інформаційна система для ідентифікації та планування всіх ресурсів підприємства, які необхідні для здійснення продажів, виробництва, закупівель та обліку в процесі виконання клієнтських замовлень. По-друге (у більш загальному контексті), це методологія ефективного планування та управління всіма ресурсами підприємства, які необхідні для здійснення продажів, виробництва, закупівель та обліку при виконанні замовлень клієнтів у сферах виробництва, дистрибуції та надання послуг. Таким чином, термін ERP означає сукупність інформаційної системи та відповідної методології управління, що реалізується та підтримується цією інформаційною системою.

ERP призначені для інтеграції всіх відділів та функцій компанії в єдину комп'ютерну систему, яка зможе обслужити всі специфічні потреби окремих підрозділів. В основі ERP-систем лежить принцип створення єдиного сховища даних, що містить всю корпоративну бізнес-інформацію та забезпечує одночасний доступ до неї будь-якої необхідної кількості працівників підприємства, наділених відповідними повноваженнями. Таким чином, ERP замінює старі розрізнені комп'ютерні системи з фінансів, управління персоналом, контролю за виробництвом, логістикою, складом, однією уніфікованою системою, що складається з програмних модулів, які повторюють функціональність старих систем. Програми, які обслуговують фінанси, виробництво чи склад, тепер зв'язуються разом (рис. 1.3), і забезпечують легкий доступ до необхідної інформації для будь-якого співробітника.

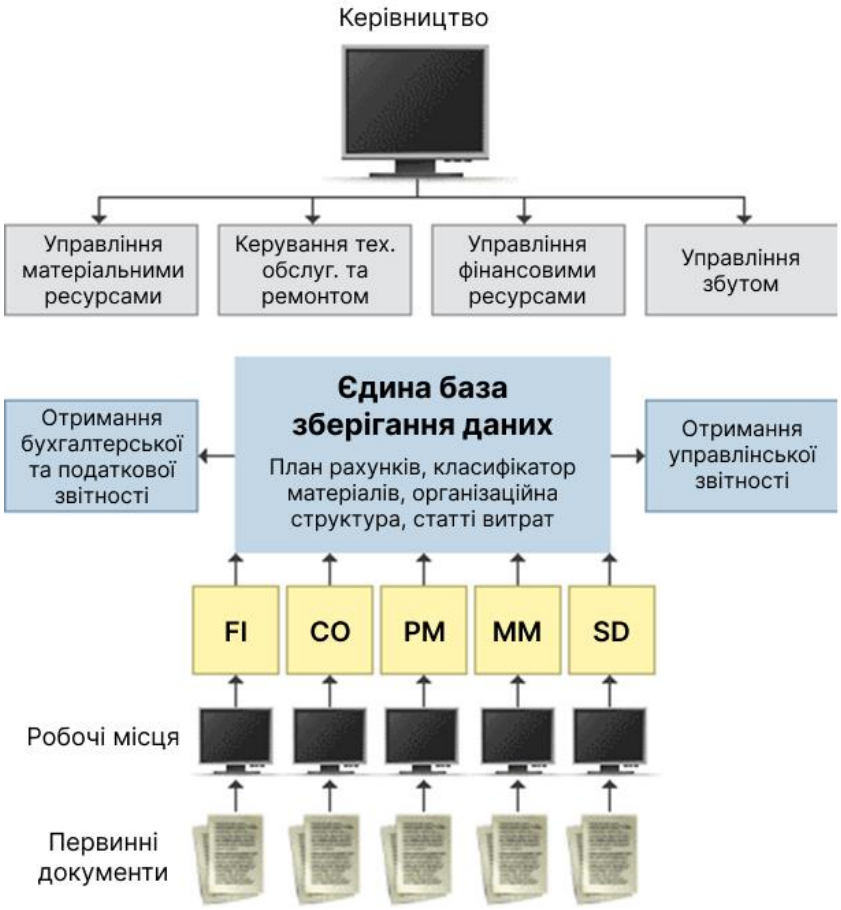

Рис. 1.3. Управління бізнес-процесами підприємства

Система розмежування доступу до інформації, що реалізується в ERP-системах, призначена (у комплексі з іншими заходами інформаційної безпеки підприємства) для протидії як зовнішнім загрозам (наприклад, промисловому шпигунству), так і внутрішнім (наприклад, розкраданням). Впроваджувані у зв'язку з CRM-системою і системою контролю якості, ERP-системи орієнтовані на максимальне задоволення потреб підприємств у засобах управління бизнесом.

ERP-системи більшості постачальників досить гнучкі та налаштовуються, їх можна встановлювати модулями, не купуючи відразу весь пакет.

Функції ERP-систем:

• ведення конструкторських і технологічних специфікацій, що визначають склад виробів, а також матеріальні ресурси та операції, необхідні для їх виготовлення;

• формування планів продажу та виробництва;

• планування потреб у матеріалах та комплектуючих, термінів та обсягів поставок для виконання плану виробництва продукції;

• управління запасами та закупівлями: ведення договорів, реалізація централізованих закупівель, забезпечення обліку та оптимізації складських та цехових запасів;

• планування виробничих потужностей від укрупненого планування до використання окремих верстатів та обладнання;

• оперативне управління фінансами, включаючи складання фінансового плану та здійснення контролю за його виконанням, фінансовий та управлінський облік;

• управління проектами, включаючи планування етапів та ресурсів.

В останнє десятиліття успішно розвивалися інтернет технології, що дозволяють підприємствам через інформаційну мережу обмінюватись даними та документами з покупцями та контрагентами. Нові функції роботи з інтернетом, що з'явилися в інтегрованих системах управління, вже виходять за традиційні рамки ERP, замкненої всередині виробничого циклу підприємства. Поєднання традиційної ERP системи підприємства з інтернет рішеннями для електронного бізнесу призвели до створення нового організаційного та управлінського середовища та нової якості системи. Результатом цього стала концепція систем нового покоління - ERP II - Enterprise Resource and Relationship Processing - управління ресурсами та зовнішніми відносинами підприємства, що мають як би два контури управління: традиційний внутрішній, керуючий внутрішніми бізнес процесами підприємства, і зовнішній – керуючий взаємодіями з контрагентами та покупцями продукції . При цьому традиційний внутрішній контур управління прийнято називати back-office – внутрішня система, а функції взаємодії з контрагентами та замовниками – front-office – зовнішня система. Таким чином, ERP II система - це методологія ERP системи з можливістю тіснішої взаємодії підприємства з клієнтами та контрагентами за допомогою інформаційних каналів, що надаються інтернет технологіями.

1.1.3 Недоліки ERP-систем

ERP системи є складним програмним продуктом. Їх використання вимагає глибокого дослідження бізнес-процесів організації, налаштування системи під ці процеси, інколи ж супроводжується і реінжинірингом самих бізнес-процесів. Основними недоліками ERP є:

• Висока вартість ERP-систем. Проводилося безліч досліджень з вивчення питання вартості володіння ERP системи (TCO - Total Cost of Ownership), включаючи апаратне і програмне забезпечення, консалтингові послуги та витрати на персонал. У підсумкову величину включалися витрати на інсталяцію ERP системи та період впровадження, протягом якого відбувається супровід системи фахівцями, її оновлення чи нарощування та оптимізація. Компанії, що брали участь у дослідженні, представляли різні галузі промисловості і ставилися як до малого або середнього, так і до великого бізнесу. Середня величина TCO на весь функціонал системи становила $1,5 млн. з розкидом від $200 тис. до $3 млн.

• Тривалі терміни застосування. Умовно ERP-системи можна поділити на 2 класи. Використання закритих систем може займати 2-3 роки. Інші ж ERP системи, гнучкіші, можна підлаштувати під роботу співробітників компанії. Їх налаштування може здійснюватися на будь-якій подальшій стадії розвитку компанії. І при цьому немає необхідності залучати консультантів фірми, яка займалася впровадженням, налаштування зможе виконати адміністратор системи. Але їх використання займає від 6 до 18 месяцев.

• Складність інтеграції з наявною інфраструктурою.

• Необхідність навчання працівників.

1.2 Специфіка автоматизованих систем для організацій

1.2.1 Потреби організацій в автоматизації бізнес-процесів

Більшість представників класу ERP-рішень відносять до так званих «важких» систем через широту класу розв'язуваних завдань, багату функціональність, високу вартість рішення та володіння, тривалі терміни впровадження. Для великих підприємств, промислових та сільськогосподарських агломерацій, дистриб'юторських мереж вибір «важкої» системи найчастіше виправданий: тимчасові, фінансові та трудові вкладення окупаються за рахунок упорядкування бізнес-процесів компанії, підвищення ефективності управління ресурсами, упорядкування бухгалтерського, управлінського, фінансового та інших видів обліку.

Однак серед невеликих підприємств, що належать до так званого «малого» бізнесу і активно працюють з інформацією, також велика потреба в упорядкуванні та автоматизації інформаційних процесів, що протікають.

Прикладами компаній такого типу, зокрема, є: невеликі організації, що постачають товари або надають професійні послуги (комп'ютерні магазини, установники мережного та комунікаційного обладнання, веб-студії, рекламні агенції), консультаційні компанії (юридичні, медичні, експертні), перекладацькі та копірайт -Агентства. Відмінними рисами таких компаній є:

• значна роль інформаційних процесів в управлінні компанією та різноманітність видів даних, що підлягають зберіганню та обліку;

• невелика кількість працівників (до 100);

• обмежена кількість рівнів управління процесами (зазвичай трохи більше 2) та його учасників (робота малими групами);

• невисока складність бізнес-процесів;

• простота бухгалтерського обліку (найчастіше застосування спрощеної системи оподаткування);

• активне використання надомної праці чи технологій мобільного офісу (територіально розподілені компанії).

На основі аналізу відмінних рис компаній можна сформулювати вимоги до корпоративної інформаційної системи:

• Наявність функціональних можливостей для організації зберігання, обліку та роздільного доступу до електронних документів будь-яких типів, ведення архіву, постановки та контролю завдань.

• Відсутність потреби в елементах електронного документообігу.

• Необхідність підтримки роздільної роботи над проектами, відсутність потреби в інтеграції із системами бухгалтерського обліку.

• Можливість віддаленого доступу, переважно використання т.зв. тонкого клієнта, наприклад веб-браузера.

Очевидно, що з одного боку, майже будь-яка «важка» ERP-система покриває всі ці потреби і вирішує завдання автоматизації, а з іншого, надмірність таких рішень у малому бізнесі та їх висока вартість робить застосування «важких» систем економічно та організаційно невиправданим.

З погляду процесів і принципів управління, що протікають, можна охарактеризувати аналізований клас завдань як проектний. Як правило, є чітко виділені етапи, терміни, результати, поетапне приймання робіт, відрядно-преміальна система заохочення, відповідальна особа на кожному етапі. "Важкі" КІС не застосовні в цих умовах. Розглянемо класи більш простих програмних продуктів, які можуть бути використані для вирішення задачі:

1. Системи управління проектами – Project Management Systems/Software;

2. Системи обліку відносин із замовниками (управління продажами) - Customer Relations Management;

3. Системи групової роботи з інформацією (wiki-проекти, дискусійні платформи, форуми);

4. Малі системи автоматизації документообігу та управління підприємствами (ERP, ECM – Enterprise Content Management, BPM – Business Processes Management);

5. Спеціальні системи замовлення.

Серед перерахованих лише група 5 повністю покриває описану потребу, однак, є і найбільш дорогою і трудомісткою при впровадженні, що значно обмежує сферу використання.

PM-системи дозволяють документувати поділ обов'язків, що склався, призначати завдання і отримувати звітність щодо їх виконання. Багато таких систем мають веб-доступ, дозволяють вести пов'язане листування, інтегруючись, наприклад з MS Exchange і поштовими службами.

Використання такої системи може бути виправданим за наявності в системі веб-модуля.

CRM-системи хороші для обліку прямих телефонних чи особистих продажів, при великій текучці кадрів, високій дисциплінованості співробітників та налагодженому бізнес-процесі продажу. Системи такого роду переважно вирішують завдання управління маркетингом, продажами та клієнтським обслуговуванням, але не командної роботи над проектом. Цей клас систем немає перспектив застосування вирішення всіх поставлених завдань.

Wiki-проекти, дискусійні платформи добре вирішують завдання групової роботи над документами, мають всі переваги засобів віддаленої роботи, проте майже не мають підтримки інших завдань. Зокрема, повністю відсутній інструментарій для прогнозування та аналітики.

Системи керування документообігом, підприємством, ресурсами (СЕД, ERP, BPM). Системи такого роду вирішують завдання якнайкраще, але вимагають спеціальної жорсткої постановки завдань: формалізації всіх сценаріїв проходження документа, прописування ролей, статусів та атрибутів, забезпечують збір звітності, підтримують нагадування. Ціна впровадження будь-якої коробкової системи буде дуже високою, а економічний ефект відносно малий. Також практично не поширені якісні веб-орієнтовані системи.

З проведеного аналізу, можна дійти невтішного висновку, що з малих організацій найперспективнішим є використання «легких» систем управління проектами.

1.2.2 Системи управління проектами та завданнями

Програмне забезпечення для управління проектами - визначення для комплексного програмного забезпечення, що включає додатки для планування завдань, складання розкладу, контролю ціни та управління бюджетом, розподілу

ресурсів, спільної роботи, спілкування, швидкого управління, документування та адміністрування системи, що використовуються спільно для управління проектами.

Однією з найпоширеніших можливостей є можливість планування подій та управління завданнями. Вимоги можуть відрізнятися залежно від того, як інструмент використовується. Найбільш поширеними є:

• Планування різних подій, що залежать один від одного;

• Планування розкладу роботи співробітників та управління ресурсами;

• Розрахунок часу, необхідного на вирішення кожної із завдань;

• Сортування завдань залежно від термінів їхнього завершення;

• Управління кількома проектами одночасно.

Програмне забезпечення для управління проектами надає велику кількість необхідної інформації, такої як:

• Список завдань для працівників та інформацію розподілу ресурсів;

• Огляд інформації про терміни виконання завдань;

• Ранні попередження про можливі ризики, пов'язані з проектом;

• Інформація про робоче навантаження;

• Інформація про хід проекту, показники та їх прогнозування.

1.2.3 Вимоги до системи

Сам факт того, що системи управління завданнями є «легкими» по відношенню до повноцінних ERP-систем, мають урізаний функціонал і простоту освоєння не позбавляє їх від недоліків даного сімейства ІС.

У організаціях особливо гостро стоїть питання із ціною та часом застосування, т.к. вони можуть собі дозволити великі довгострокові інвестиції. Крім того, системи для малих організацій повинні бути орієнтовані на мережу, що вже є, архітектуру, апаратне забезпечення організації, ІТ-інфраструктуру. Це означає підтримку найбільш поширених серверних платформ, якими на даний момент є Windows та Unix-подібні системи, можливість використання не на

професійному серверному обладнанні, інтеграцію з поширеними системами та підтримку популярних форматів документів.

Зниження витрат на використання можливе при використанні системою популярних відкритих технологій за рахунок подальшого доопрацювання сторонніми фахівцями. Зменшення вартості перенавчання співробітників може бути досягнуто при використанні графічного інтерфейсу, заснованого на поширених технологіях і принципах, наприклад веб-інтерфейсу.

Виходячи з вищевикладеного, до системи можна пред'явити такі вимоги:

Функціональні

• *Список завдань* – можливість створення завдань з описом, набором характеристик та термінами виконання, а також відображення списку завдань для цього працівника;

• *Інформування про нові події* – своєчасне відображення у системі інформації про зміни, пов'язані із завданнями співробітника;

• *Ієрархія задач* – можливість зберігання та подання завдань у вигляді ієрархічного дерева. Можливість декомпозиції завдань на підзавдання зі своїми делегуванням;

• *Коментування* – обговорення завдань у системі, зі збереженням історії обговорення;

• *Теги* – призначення задач необмеженої кількості міток – ключових слів з можливістю пошуку за тегом та набором тегів;

• *Пошук* – реалізація пошуку за параметрами завдання, відповідальними особами, описом тощо;

• *Зберігання зовнішніх контактів* – можливість зберігання в системі будь-яких важливих робочих контактів та прикріплення їх до конкретного завдання для прискорення взаємодії;

• *Поділ доступу* – призначення ролей із завдання та обмеження доступу на їх основі;

• *Зберігання файлів* – прикріплення до завдань пов'язаних із ними файлів;

• *Планування* – можливість менеджеру вести в системі облік та прогнозування робочого часу співробітників (в т.ч. відпустки та інше) та інших ресурсів;

• *Аналітика* – збирання статистики та аналіз ефективності співробітників, їх завантаженості в даний момент;

Вимоги до розгортання, налаштування, інтеграції та експлуатації

• *Модульність, розширюваність* – встановлення лише модулів з необхідними функціями, додавання функцій за потреби;

• *Відкриті технології* – для забезпечення можливості доопрацювання системи сторонніми розробниками

• *Інтеграція з наявною IT-інфраструктурою* – наприклад, авторизація у системі через домен організації, переносимість документів між системами тощо;

• *Налаштовуваність під структуру компанії* – можливість застосування системи без значної переробки під конкретну організацію;

• *Логування змін та резервне копіювання* – стійкість системи до втрати та помилкової зміни інформації, можливість відкату до неушкодженого стану;

• *Мультиплатформенність* – можливість розгортання більшості поширених операційних систем;

• *Робота віддалених користувачів* – можливість роботи у системі поза офісом;

• *Інтерфейс користувача* – зрозумілий та зручний графічний інтерфейс, що базується на поширених технологіях.

1.3 Існуючі рішення систем керування завданнями

1.3.1 Огляд сучасних систем керування завданнями

Redmine

Redmine (рис. 1.4) — відкритий серверний веб-додаток для управління проектами та відстеження помилок. Redmine є додатком на основі фреймворку Ruby on Rails. Поширюється згідно з GNU General Public License.

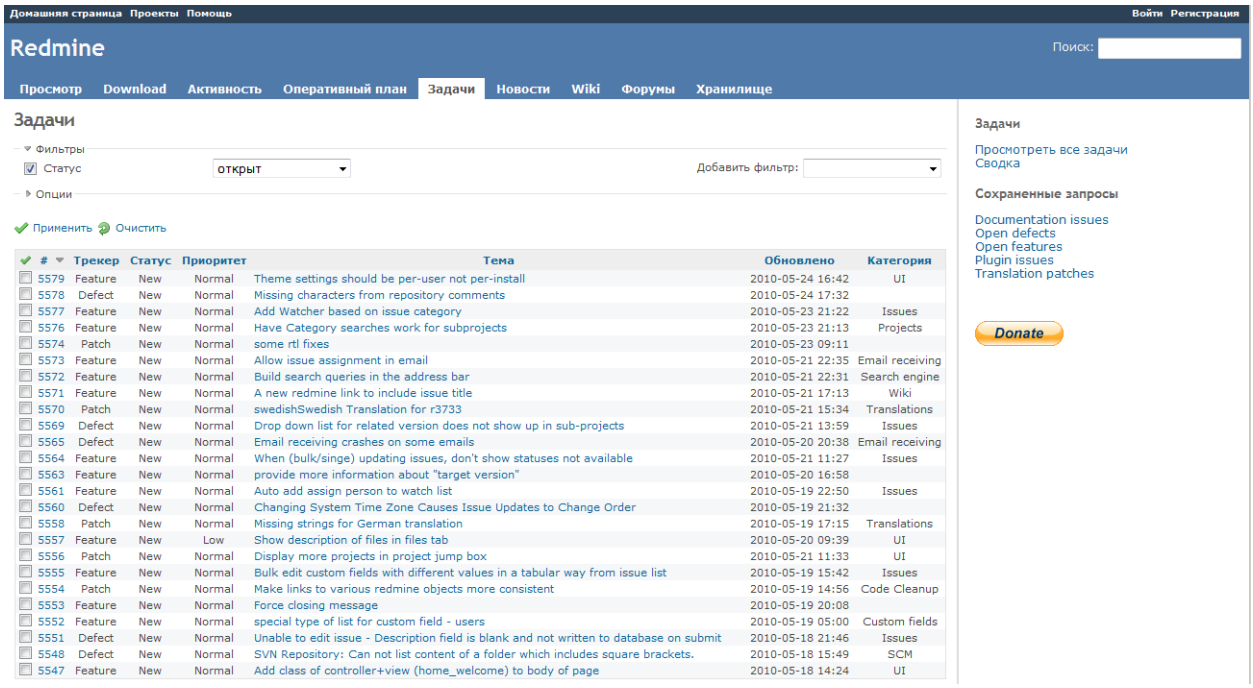

Рис. 1.4. Інтерфейс користувача користувача Redmine

Цей продукт надає такі можливості:

• ведення кількох проектів;

• гнучка система доступу, що базується на ролях;

• система відстеження помилок;

• діаграми Ганта та календар;

• ведення новин проекту, документів та управління файлами;

• оповіщення про зміни за допомогою RSS-потоків та електронної пошти;

• вікі для кожного проекту;

• форуми для кожного проекту;

• облік тимчасових витрат;

• довільні поля для інцидентів, тимчасових витрат, проектів і користувачів;

• легка інтеграція із системами керування версіями (SVN, CVS, Mercurial, Bazaar);

• створення записів про помилки на основі отриманих листів;

• підтримка множинної автентифікації LDAP;

• можливість самостійної реєстрації нових користувачів;

• багатомовний інтерфейс (у тому числі російська);

• підтримка СУБД MySQL, PostgreSQL, SQLite, Oracle.

Easy Projects .NET

Easy Projects .NET (рис. 1.5) - це веб-додаток для управління проектами з розробки програмного забезпечення, написаний на .NET компанією Logic Software.

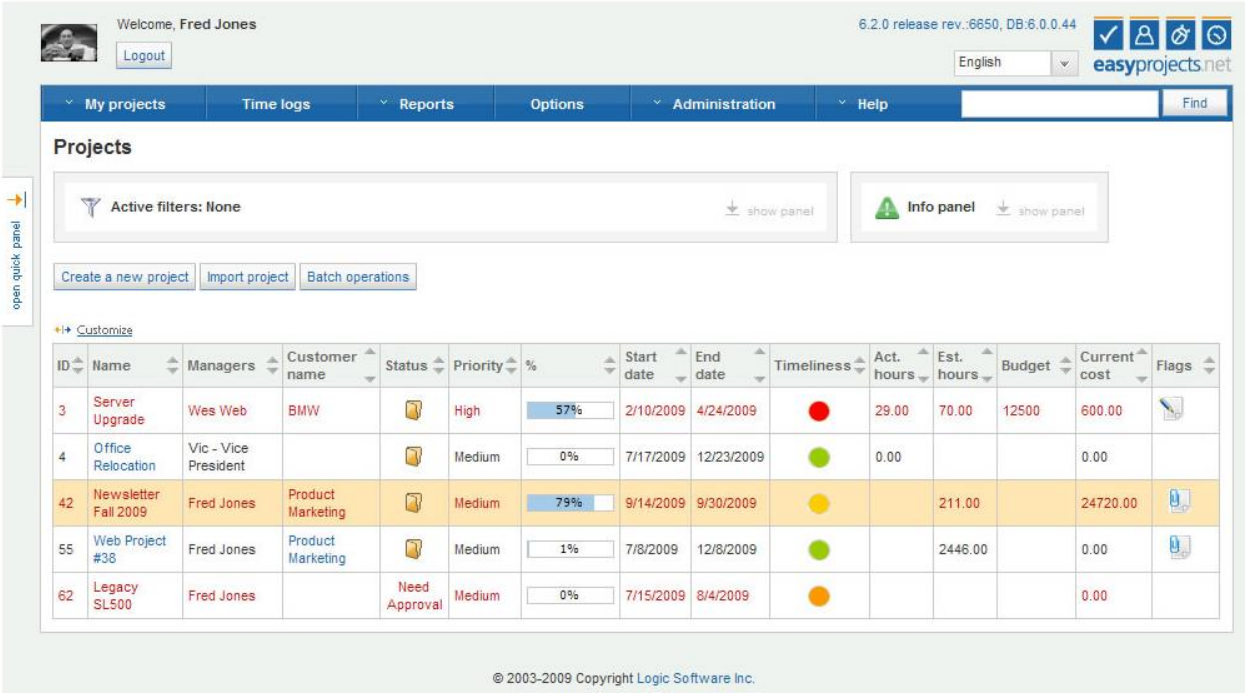

Рис. 1.5. Інтерфейс користувача користувача Easy Projects .NET

Easy Projects .NET дозволяє створювати необмежену кількість проектів, що містять різні поля, що настроюються. Пакетна обробка даних дозволяє виконувати типові операції для кількох проектів одночасно. Користувачам доступні інтерактивна діаграма Гантта, графіки та звіти. Окремі проекти можуть групуватися у портфелі проектів.

Easy Projects .NET підтримує необмежену кількість завдань та підзадач, а також налаштування статусів, категорій та пріоритетів завдань. Можливе створення завдань електронною поштою. Не лише розробники, а й клієнти мають можливість додавати запити та вимоги.

Програма дозволяє вносити та відстежувати оплачуваний та неоплачуваний час, витрачений на проект. Підтримуються особисті та

корпоративні розклади, а також є можливість перегляду графіка завантаженості ресурсів. Можливе гнучке налаштування прав доступу користувачів.

Зовнішній вигляд системи може бути налаштований шляхом додавання або видалення віджетів з інформацією про проекти. Користувачі можуть використовувати веб-конференції для спільної роботи. Підтримуються англійська, німецька, французька та російська мови інтерфейсу.

Мегаплан

Мегаплан (рис. 1.6) — комерційний програмний продукт, який дозволяє керувати завданнями та дорученнями, стежити за їх виконанням та зберігати базу даних співробітників компанії. Складається з трьох комерційних продуктів та двох безкоштовних. Включає кілька основних модулів:

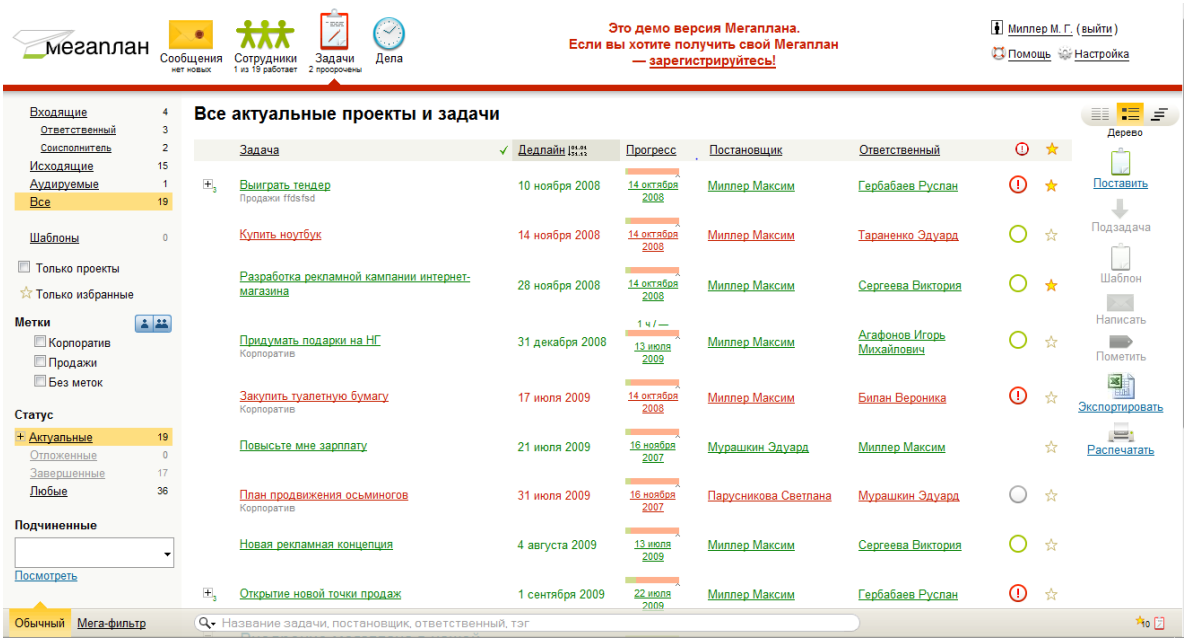

Рис. 1.6. Інтерфейс користувача МегаПлан

Task Manager - система управління завданнями та проектами. Продукт на вирішення внутрішніх завдань компанії. Складається з модулів: Робочий стіл, Повідомлення, Співробітники, Завдання, Справи. Інформація про кожного співробітника та ієрархія підпорядкування в компанії. Внутрішня пошта та сервіс повідомлень про поставлені завдання. Постановка та контроль виконання завдань. Делегування обов'язків.

Intranet - система управління компанією, співробітниками та завданнями. Продукт, що включає функціонал Task Manager + модулі «документи» і модуль «обговорення».

Project Manager - система управління проектами та взаємовідносинами з клієнтами. Продукт призначений для вирішення бізнес-процесів у проектних компаніях.

Складається зі всіх модулів Інтранету + модуль Клієнти (ведення клієнтської бази, планування комунікацій).

Time manager – особисті завдання співробітника. Виконує функції персонального інформаційного менеджера та органайзера.

Life manager - система управління особистими завданнями та проектами для фрілансерів та людей творчих професій.

Basecamp

Basecamp (рис. 1.7) - це онлайн-інструмент для управління проектами, спільної роботи та постановки завдань за проектами, створений компанією 37signals.

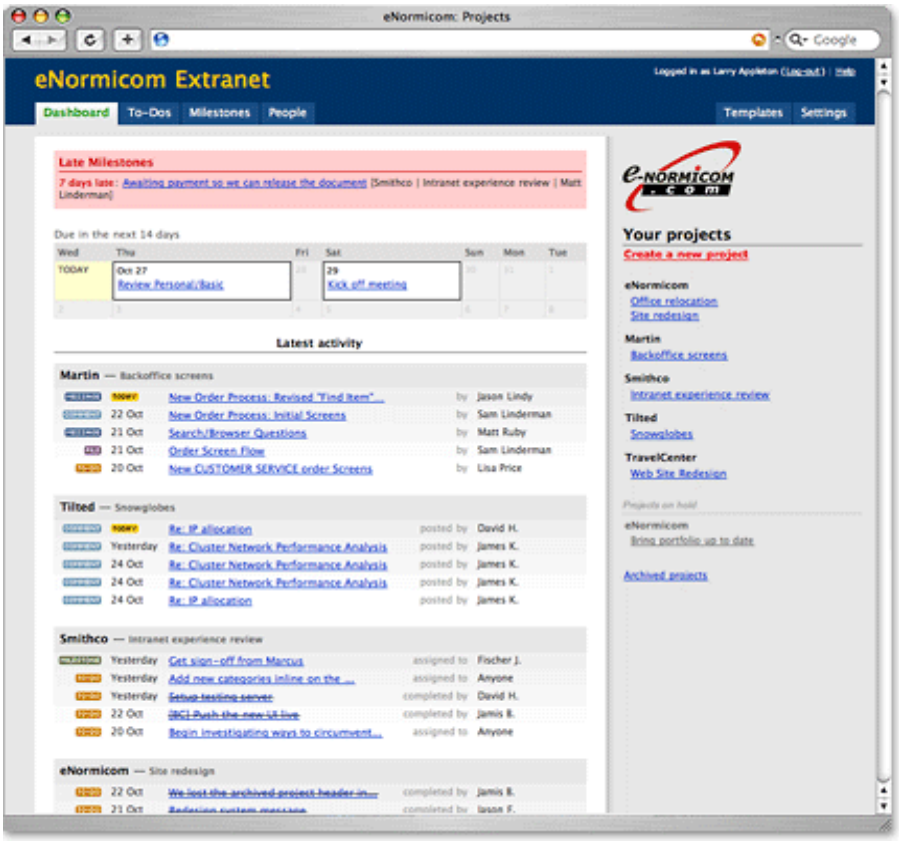

Рис. 1.7. Інтерфейс користувача Basecamp

В даний час Basecamp надає користувачам такі можливості:

- зміна колірної гами та логотипу системи,
- перегляд загальної інформації про клієнтів та проекти на одному екрані,
- призначення та відстеження завдань,
- завантаження, категоризація та відстеження версій файлів,
- форуми для обговорення завдань та проектів,
- ведення розкладу та управління ключовими точками проекту,
- відстеження витраченого часу,
- отримання основної інформації про проект на одному екрані,
- додавання повідомлень та коментарів.

Інтерфейс системи доступний англійською мовою.

Basecamp сумісний з багатьма програмами, віджетами та іншими програмами. На офіційному сайті доступні платні та безкоштовні доповнення у наступних категоріях:

- програми для айфона,
- мобільні додатки,
- звіти, графіки та планування,
- бухгалтерські системи,
- інструменти для розробки програмного забезпечення,
- облік робочого часу,
- віджети для робочого столу,
- інші доповнення.

Сторонні розробники можуть самостійно створювати доповнення до Basecamp.

Типи реєстрації

У Basecamp існує 4 типи платних облікових записів, що відрізняються кількістю проектів і користувачів, розміром файлового сховища, а також наявністю засобів обліку робочого часу. Компанія надає 30-денний

випробувальний термін для будь-якого з платних облікових записів. Також доступний безкоштовний план без обмежень за часом використання, за допомогою лише одного проекту, але без можливості завантажувати файли.

1.3.2 Порівняльний аналіз існуючих систем

Аналіз відповідності аналізованих систем висунутих у пункті 1.3.1 представлений на рисунку 1.8.

	Redmine	Easy Projects .NET	МегаПлан	BaseCamp
Список завдань	Так	Так	Так	Так
Інформування про події	Так	Так	Так	Так
Ієрархія завдань	Обмежена	Так	Так	Так
Коментарі	Так	Форум/Вікі	Так	Так
Теги	Так	Ні	Так	Ні
Пошук	Так	Так	Частково	Так
Зовнішні контакти	Так	Ні	Так	Так
Поділ доступу	Так	Так	Так	Так
Зберігання файлів	Так	Так	Так	Так
Планування	Ні	Ні	Ні	Ні
Аналітика	Так	Так	Обмежено	Обмежено
Модульність	Ні	Ні	Так	Так
Відкритість	Так	Ні	Ні	API
Інтеграція	Так	Так	Ні	Ні
Настроюваність	Так	Так	Так	Так
Логування	Ні	Ні	Ні	Ні
Мультиплатформенність	Так	Ні	Так	Так
Віддалена робота	Так	Так	Так	Так
Інтерфейс користувача	Складний	Так	Так	Так
Продуктивність	Висока	Низька	Низька	Висока
Ціна	GNU General Public License	75$ в місяць або 648$	176 грн. в місяць за користувача	24-149$ в місяць

Рис. 1.8. Порівняння систем управління проектами за ключовими параметрами

Очевидно, що жодна з систем, що розглядаються, повною мірою не відповідає наданим вимогам. Найменше реалізовані функції, необхідних керівників: облік і планування ресурсів, і робочого дня, аналіз ефективності співробітників тощо. Розглянуті системи складні в адмініструванні та налаштуванні під потреби конкретної організації. Більшість платних систем є досить дорогими і вимогливими до ресурсів, що накладає обмеження на їх

використання в малих організаціях. Розглянута open-source система має досить заплутаний інтерфейс користувача і не русифікована.

Таким чином, розробка простої в експлуатації та підтримці системи, що відповідає поставленим вимогам і має розвинені інструменти менеджера, що реалізують функції управління трудовими, тимчасовими та матеріальними ресурсами, планування, аналізу ефективності роботи, є актуальним завданням.

1.4 Принципи розробки кроссплатформенних розподілених додатків

1.4.1 Способи організації кроссплатформенності

Одна з сучасних тенденцій у програмуванні - можливість роботи програми під різними операційними системами та на різних архітектурах. Кросплатформенність дозволяє полегшити працю програміста, продовжити час життя програми, не обмежуючи його часом життя ОС та процесора. Принагідно покращується сумісність програми з різними версіями однієї ОС.

Існують різні способи організації кросплатформенності:

• Скриптові мови (PHP, Perl, Python);

• Розповсюдження програм у вихідних кодах. (Потрібна перекомпіляція);

• Виконання програм у байт-коді віртуальною машиною (Java);

• Емуляція апаратного забезпечення (VMWare, Plesk, Virtuozzo);

• Емулятори API (Wine).

Останні два варіанти можуть застосовуватися для вирішення локальних завдань, однак як основа розробки кроссплатформенного програмного продукту не підходять. Поширення вихідного коду програм потребує відповідної бізнес-моделі. Крім того, більшість мов, що компілюються, досить незручні для розробки мережевих розподілених додатків і застосовуються тільки для високонавантажених програмних продуктів з високими вимогами до швидкодії. Найбільш поширеними та гнучкими способами організації кросплатформенності розподілених додатків зараз є використання Java та скриптових мов програмування.

Програми Java транслюються в байт-код, що виконується віртуальною машиною Java (JVM) - програмою, що обробляє байтовий код і передає інструкції обладнання як інтерпретатор. Байтовий код, причому, на відміну тексту, обробляється значно швидше.

Достоїнство подібного способу виконання програм — у повній незалежності байт-коду від операційної системи та обладнання, що дозволяє виконувати Java-програми на будь-якому пристрої, для якого є відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

До недоліків концепції віртуальної машини відносять те, що виконання байт-коду віртуальною машиною може знижувати продуктивність програм та алгоритмів, реалізованих мовою Java. Дане твердження було справедливим для перших версій віртуальної машини Java, проте останнім часом воно практично втратило актуальність. Цьому сприяла низка удосконалень:

• застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,

• широке використання платформно-орієнтованого коду (native-код) у стандартних бібліотеках,

• апаратні засоби, що забезпечують прискорену обробку байт-коду.

Основні можливості:

• автоматичне керування пам'яттю;

• розширені можливості обробки виняткових ситуацій;

• багатий набір засобів фільтрації вводу/виводу;

• набір стандартних колекцій, таких як масив, список, стек тощо;

• наявність простих засобів створення мережевих програм (у тому числі з використанням протоколу RMI);

• наявність класів, що дозволяють виконувати HTTP-запити та обробляти відповіді;

• вбудовані в мову засоби створення багатопотокових програм;

• уніфікований доступ до баз даних:

• на рівні окремих SQL-запитів – на основі JDBC, SQLJ;

• на рівні концепції об'єктів, що мають здатність до зберігання в базі даних - на основі Java Data Objects і Java Persistence API;

• підтримка шаблонів (починаючи з версії 1.5);

• паралельне виконання програм.

Скриптові мови — мови програмування для запису послідовностей операцій, які виконуються на комп'ютері. Вказані операції іноді називають сценаріями, а скриптові мови – мовами сценаріїв. В описі деяких мов програмування та програм скрипти називаються макросами. Ще одна назва скриптових мов – мови пакетної обробки.

Сценарії зазвичай інтерпретуються, а чи не компилируются, тому скриптові мови у класифікації позначають як интерпретируемые. Однак багато сучасних інтерпретаторів є компілюваними: компіляція виконується щоразу при завантаженні скрипта з подальшим виконанням відкомпілюваного коду.

Прикладами найпопулярніших універсальних скриптових мов є PHP, Perl, Python, Ruby.

Перевагами скриптових мов можна назвати просту і швидку розробку завдяки вбудованим в мову високорівневим інструментам, кросплатформеність, можливість вбудовування інтерпретатора у веб-сервер (mod_perl, mod_php для apache) для збільшення швидкодії веб-додатків.

Порівняно з Java універсальні скриптові мови мають низьку швидкодію – до 200-кратного уповільнення на окремих тестах. Але, у своїй, споживають восьмиразово менший обсяг пам'яті. Це пояснює використання Java лише у великих продуктах управління підприємствами таких компаній, як IBM, SAP тощо, тоді як скриптові мови використовуються малонавантажених системах з обмеженими апаратними ресурсами.

1.4.2 Клієнт-серверна архітектура

«Клієнт-сервер» – це модель взаємодії комп'ютерів у мережі. Зазвичай комп'ютери є рівноправними. Кожен із них має своє призначення. Деякі комп'ютери в мережі володіють та розпоряджаються інформаційно-обчислювальними ресурсами, такими як процесори, файлова система, поштова служба, служба друку, база даних. Інші мають можливість звертатися до цих служб, користуючись першими послугами. Комп'ютер, який керує тим чи іншим ресурсом, прийнято називати сервером цього ресурсу, а комп'ютер, який бажає ним скористатися - клієнтом. Конкретний сервер визначається видом ресурсу, яким він володіє. Так, якщо ресурсом є бази даних, то йдеться про сервер баз даних, призначення якого – обслуговувати запити клієнтів, пов'язані з обробкою даних; якщо ресурс - це файлова система, то говорять про файловий сервер і т.д.

У мережі той самий комп'ютер може виконувати як роль клієнта, і роль сервера. Такий самий принцип поширюється і взаємодія програм. Якщо одна з них виконує деякі функції, надаючи іншим відповідний набір послуг, така програма розглядається як сервер. Програми, які користуються цими послугами, називають клієнтами. Так, ядро реляційної SQL-орієнтованої СУБД часто називають сервером бази даних або SQL-сервером, а програму, яка звертається до нього за послугами з обробки даних - SQL-клієнтом.

В даний час фактичним стандартом для розрахованих на багато користувачів СУБД, стала архітектура «клієнт-сервер».

Якщо передбачається, що проектована ІС буде побудована за технологією «клієнт-сервер», це означає, що прикладні програми, реалізовані в її рамках, матимуть розподілений характер. Іншими словами, частина функцій прикладної програми буде реалізована у програмі-клієнті, інша – у програмі-сервері, причому для їхньої взаємодії буде визначено деякий протокол.

Основний принцип технології «клієнт-сервер» полягає у розподілі функцій стандартного інтерактивного додатка на чотири групи, що мають різну природу. Перша група – це функції введення та відображення даних. Друга група поєднує прикладні функції, притаманні даної предметної області. До третьої групи

належать фундаментальні функції зберігання та управління інформаційними ресурсами (базами даних, файловими системами тощо). Нарешті, функції четвертої групи – службові, які відіграють роль зв'язок між функціями перших трьох груп. Відповідно до цього у будь-якому додатку виділяються такі логічні компоненти:

• компонент уявлення, що реалізує функції першої групи;

• прикладний компонент, який підтримує функції другої групи;

• компонент доступу до інформаційних ресурсів, який підтримує функції третьої групи;

• протокол взаємодії.

Виділяються чотири підходи, реалізовані у таких моделях:

• модель файлового сервера (File Server – FS);

• модель доступу до віддалених даних (Remote Data Access – RDA);

• модель півночі бази даних (DataBase Server – DBS);

• модель сервера програм (Application Server – AS).

FS-модель є базовою для локальних мереж персональних комп'ютерів (Рис. 1.9). Відповідно до цієї моделі один із комп'ютерів у мережі вважається файловим сервером і надає послуги з обробки файлів іншим комп'ютерам. Файловий сервер працює під управлінням мережної операційної системи та відіграє роль компонента доступу до інформаційних ресурсів (файлів). На інших комп'ютерах у мережі функціонує програма, в кодах якої поєднані компонент подання та прикладний компонент.

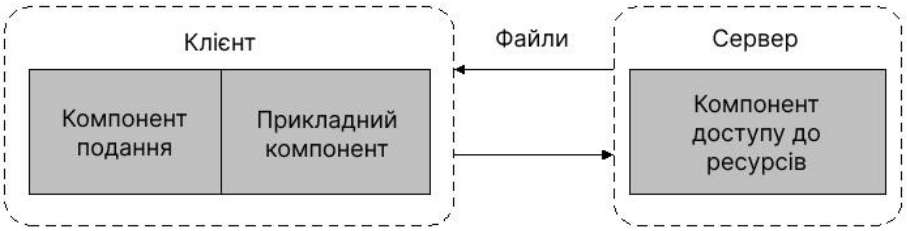

Рис. 1.9. Модель файлового сервера

Коли прикладна програма звертається до бази даних, СУБД надсилає запит на файловий сервер. У цьому запиті вказані файли, де знаходяться дані, що

запитуються. У відповідь на запит файловий сервер спрямовує по мережі потрібний блок даних. СУБД, отримавши його, виконує над даними дії, декларованих у прикладній програмі.

До технологічних недоліків моделі відносять високий мережевий трафік (передача безлічі файлів, необхідних додатку), вузький спектр операцій маніпулювання даними, відсутність адекватних засобів безпеки доступу до даних (захист лише на рівні файлової системи) тощо.

Більш технологічна модель RDA (Рис. 1.10) істотно відрізняється від FS-моделі характером компонента доступу до інформаційних ресурсів. Це зазвичай SQL-сервер. У RDA-моделі коди компонента представлення та прикладного компонента поєднані та виконуються на комп'ютері-клієнті. Останній підтримує як функції введення та відображення даних, так і чисто прикладні функції. Доступ до інформаційних ресурсів забезпечується або операторами спеціальної мови (мов SQL, якщо йдеться про бази даних) або викликами функцій спеціальної бібліотеки, за наявності відповідного інтерфейсу прикладного програмування – API.

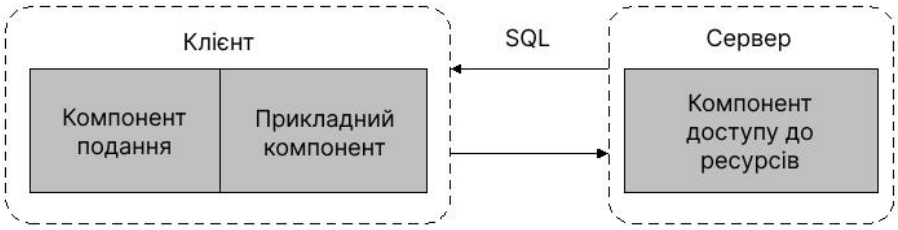

Рис. 1.10. Модель доступу до віддалених даних

Клієнт надсилає запити до інформаційних ресурсів через мережу віддаленого комп'ютера. На ньому функціонує ядро СУБД, яке обробляє запити, виконуючи вказані в них дії, і повертає клієнту результат, оформлений як блок даних. При цьому ініціатором маніпуляцій з даними виступають програми, що виконуються на комп'ютерах-клієнтах, тоді як ядру СУБД відводиться пасивна роль обслуговування запитів і обробка даних.

Основна перевага RDA-моделі полягає в уніфікації інтерфейсу "клієнт-сервер" у вигляді мови SQL, суттєво розвантажує сервер БД, сервер БД звільняється від невластивих йому функцій, зменшується завантаження мережі.

RDA-модель не позбавлена недоліків: задовільне адміністрування додатків у RDA-моделі практично неможливе через поєднання в одній програмі різних за своєю природою функцій (функції представлення та прикладні функції).

Поряд з RDA-моделлю все більшої популярності набуває перспективна DBS-модель (Рис. 1.11). Остання реалізована у деяких реляційних СУБД (Informix, Ingres, Sybase, Oracle, InterBase). Її основу складає механізм процедур, що зберігаються - засіб програмування SQL-сервера. Мова, у якому розробляються збережені процедури (SQL/PTL), є процедурне розширення мови запитів SQL і унікальний кожної конкретної СУБД.

Переваги DBS-моделі: можливість централізованого адміністрування прикладних функцій, і зниження трафіку (замість SQL-запитів по мережі направляються виклики процедур, що зберігаються), можливість поділу процедури між декількома додатками, економія ресурсів комп'ютера за рахунок використання одного разу створеного плану виконання процедури. До недоліків можна віднести обмеженість засобів, що використовуються для написання збережених процедур, які є різноманітними процедурними розширеннями SQL, які не витримують порівняння за функціональними можливостями з мовами третього покоління, такими як C або Pascal.

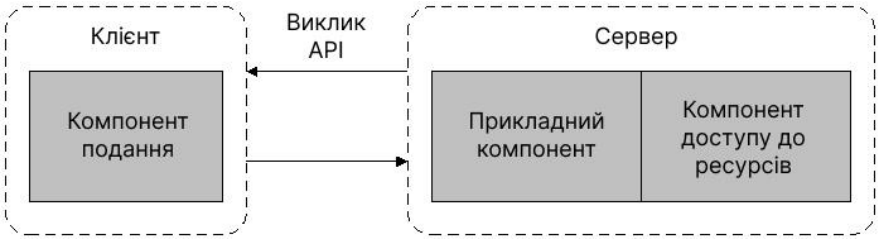

Рис. 1.11. Модель сервера баз даних

На практиці часто використовуються змішані моделі, коли підтримка цілісності бази даних і деякі найпростіші прикладні функції виконуються процедурами, що зберігаються (DBS-модель), а більш складні функції

реалізуються безпосередньо в прикладній програмі, яка працює на комп'ютері-клієнті (RDA-модель). Так чи інакше, сучасні розраховані на багато користувачів СУБД спираються на RDA- і DBS-моделі і при створенні ІВ, що передбачає використання тільки СУБД, вибирають одну з цих двох моделей, або їх розумне поєднання.

У AS-моделі процес, що виконується на комп'ютері-клієнті, відповідає, як завжди, за інтерфейс з користувачем (тобто реалізує функції першої групи). Звертаючись за виконанням послуг до прикладного компоненту, цей процес відіграє роль клієнта програми (Application Client - AC) (Рис. 1.12). Прикладний компонент реалізований як група процесів, що виконують прикладні функції, і називається сервером програми (Application Server - AS). Усі операції над інформаційними ресурсами виконуються відповідним компонентом, стосовно якого AS відіграє роль клієнта. З прикладних компонентів доступні ресурси різних типів – бази даних, черги, поштові служби та інших.

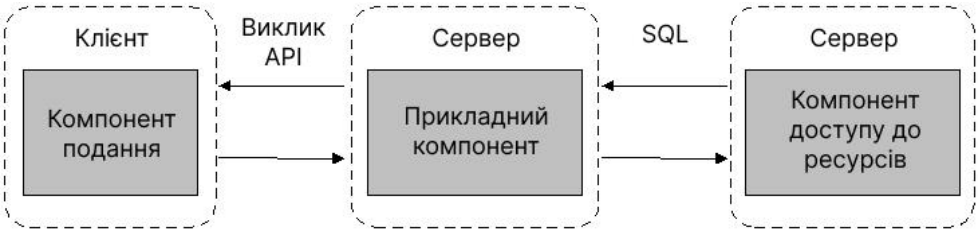

Рис. 1.12. Модель application server

РОЗДІЛ 2
ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ОБРАНІ ТЕХНОЛОГІЇ

2.1 Вибір інструментальних засобів розробки

2.1.1 AllFusion Process Modeler r7

AllFusion Process Modeler 7 (раніше BPwin) - інструмент для моделювання, аналізу, документування та оптимізації бізнес-процесів. AllFusion Process Modeler 7 можна використовувати для графічного представлення бізнес-процесів. Графічно представлена схема виконання робіт, обміну інформацією, документообігу візуалізує модель бізнес-процесу.

AllFusion Process Modeler 7 допомагає чітко документувати важливі аспекти будь-яких бізнес-процесів: дії, які необхідно вжити, способи їх здійснення та контролю, потрібні для цього ресурси, а також візуалізувати одержувані від цих дій результати. AllFusion Process Modeler 7 підвищує бізнес-ефективність IT-рішень, дозволяючи аналітикам і проектувальникам моделей співвідносити корпоративні ініціативи і завдання з бізнес-вимогами і процесами інформаційної архітектури і проектування додатків. Таким чином, формується цілісна картина діяльності підприємства: від потоків робіт в невеликих підрозділах до складних організаційних функцій.

Основні можливості системи:

- Підтримка різних технологій моделювання
- Аналіз показників витрат і продуктивності
- Інтеграція процесів / даних
- Підтримка стандартних нотацій
- Експорт об'єктів і властивостей в інші моделі
- Документування інформації в межах всієї моделі
- Масштабованість звітності без втрати якості графіків

2.1.2 Вибір СУБД

Сучасний ринок СУБД представлений двома основними напрямками: традиційними реляційними СУБД і активно розвивається сімейством нереляційних або об'єктних систем, об'єднаних під загальною назвою NoSQL.

Сімейство об'єктних СУБД досить різнорідне; вони, як правило, досить специфічні і застосовуються в даний час для вирішення обмеженого набору спеціалізованих завдань, пов'язаних зазвичай з високим навантаженням на БД.

Традиційні СУБД на основі реляційної мови SQL поширені повсюдно і вирішують широкий спектр завдань. Завдяки подібним принципам функціонування, різні реалізації реляційних СУБД прості у вивченні і ринок трудових ресурсів насичений фахівцями в даній області.

Реляційні СУБД можна умовно розділити на великі корпоративні системи, такі як MS SQL Server, Oracle Database, і невеликі системи для середніх, малих компаній і приватного використання – MySQL, PostgreSQL. Для перших характерні відмінна масштабованість, але і серйозні вимоги до апаратної частини, а також висока вартість самих СУБД. Другі можуть поширюватися під вільною ліцензією, не вимогливі до апаратного забезпечення і, хоч і поступаються корпоративним системам в робочих характеристиках, цілком відповідають поставленим в даному дипломному проекті вимогам.

У порівнянні з MySQL, PostgreSQL має кращу масштабованість, однак при невеликій кількості паралельних процесів (до 6) поступається у швидкодії. Результати тестування даних СУБД на різних апаратних конфігураціях представлені на рис. 2.1.

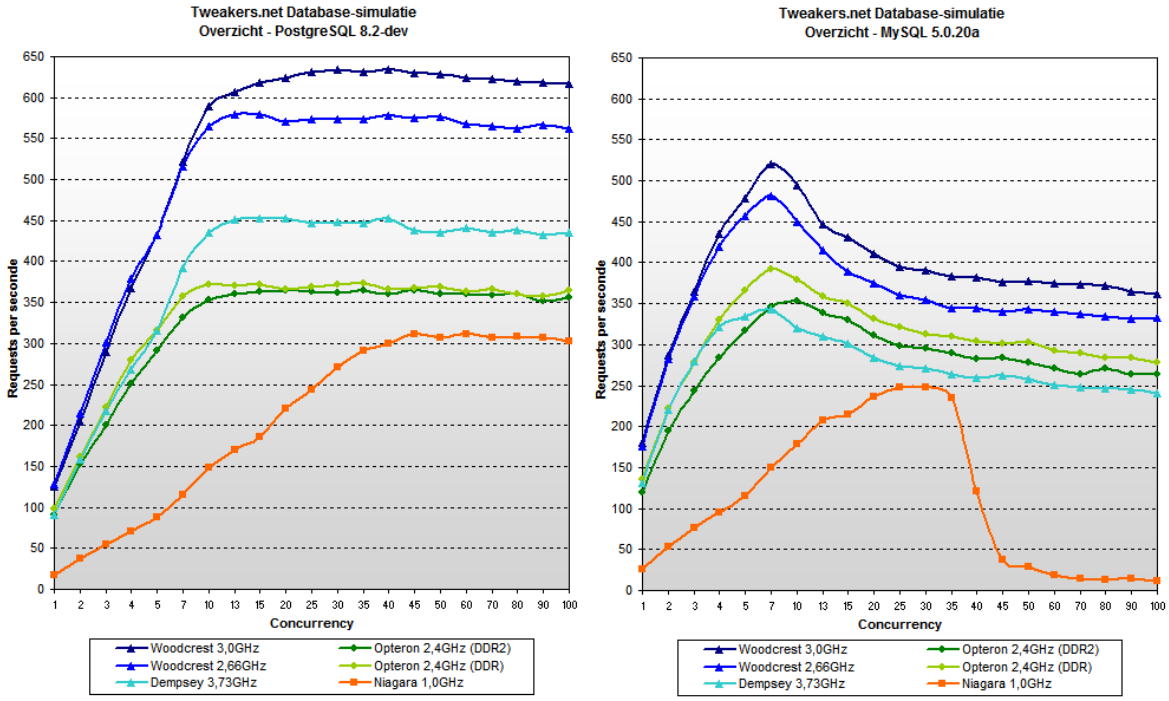

Рис. 2.1. Результати тестування PostgreSQL (ліворуч) і MySQL (праворуч)

MySQL

MySQL - це система управління реляційними базами даних з відкритим вихідним кодом.

Це означає, що застосовувати і модифікувати його може будь-хто. Використання програмного забезпечення MySQL регламентується ліцензією GPL (GNU General Public License). Якщо робота в рамках GPL не влаштовує або планується вбудовування MySQL-коду в комерційний додаток, є можливість купити комерційну ліцензовану версію у компанії MySQL AB.

Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечити більш високу швидкість роботи в порівнянні з існуючими на той момент аналогами

Основні можливості MySQL

Нижче наведено опис важливих характеристик програмного забезпечення MySQL.

- Внутрішні характеристики та портативність
- Написаний на C і c++. Протестований на безлічі різних компіляторів.

- Працює на різних платформах: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP і іншим версіями Windows. Існує також порт MySQL до OpenVMS. Важливо відзначити, що компанія MySQL AB надає для вільного завантаження не тільки вихідні коди СУБД, але і відкомпільовані і оптимізовані під конкретні операційні системи готові виконувані модулі, які можна завантажити з дзеркал, представлених на офіційному сайті.
- Для забезпечення переносимості використовується GNU Automake, Autoconf і Libtool.
- API для C, C++, Eiffel, Java, Perl, PHP, Python, Ruby і Tcl.
- Повністю багатопотоковий з використанням потоків ядра. Це означає, що, якщо така можливість забезпечується, можна легко організувати роботу з декількома процесорами.
- Дуже швидкі дискові таблиці на основі в-дерев зі стисненням індексів.
- Дуже швидка базується на потоках система розподілу пам'яті.
- Дуже швидкі з'єднання, що використовують оптимізований метод однопрохідного мультисоединения (one-sweep multi-join).
- Хеш-таблиці в пам'яті, використовувані як тимчасові таблиці.
- SQL-функції реалізовані за допомогою добре оптимізованої бібліотеки класів, тому вони виконуються настільки швидко, наскільки це можливо. Зазвичай після ініціалізації запиту розподілу пам'яті не відбувається взагалі.
- MySQL-код Протестований з використанням Purify (Комерційний детектор витоку пам'яті), а також Valgrind, одного з GPL-інструментів.
- Велика кількість: цілочисельні зі знаком / беззнакові, довжиною в 1, 2, 3, 4 і 8 байтів, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET і ENUM.

- Із записами фіксованої і змінної довжини.
- Всі стовпці мають значення за замовчуванням. За допомогою INSERT можна вставити підмножину стовпців таблиці; стовпці, для яких явно не задані значення, встановлюються в значення за замовчуванням.

Команди і функції

Повна підтримка операторів і функцій в SELECT-і WHERE-частинах запитів.

Повна підтримка для операторів SQL GROUP BY і ORDER BY з виразами SQL. Підтримка групових функцій (COUNT (), COUNT(DISTINCT ...), AVG(), STD(), SUM (), MAX () і MIN ()).

Підтримка LEFT OUTER JOIN і RIGHT OUTER JOIN з синтаксисом ANSI SQL і ODBC.

Дозволені псевдоніми для таблиць і стовпців відповідно до стандарту SQL92.

DELETE, INSERT, REPLACE, and UPDATE повертають число рядків, які були змінені. Замість цього можна задати повернення збіглися рядків. Для цього слід встановити прапор при з'єднанні з сервером.

Команду SHOW, яка є специфічною для MySQL, можна використовувати для отримання інформації про бази даних, таблиці та індекси. Щоб з'ясувати, як оптимізатор виконує запит, можна застосовувати команду EXPLAIN.

Імена функцій не конфліктують з іменами таблиць і стовпців. Наприклад, ABS є коректним ім'ям стовпця. Для виклику функції існує тільки одне обмеження: між ім'ям функції і наступною за ним відкриває дужкою `(' не повинно бути пробілів.

В одному і тому ж запиті можуть вказуватися таблиці з різних баз даних (з версії 3.22).

Безпека

Система, заснована на привілеях і паролях, за рахунок чого забезпечується гнучкість і безпеку, і з можливістю верифікації з віддаленого комп'ютера. Паролі

захищені, тому що вони при передачі по мережі при з'єднанні з сервером шифруються.

Масштабованість і обмеження

Управляє дуже великими базами даних. Компанія MySQL AB. використовує MySQL для роботи з декількома базами даних, які містять 50 мільйонів записів.

Для кожної таблиці дозволяється мати до 32 індексів. Кожен індекс може містити від 1 до 16 стовпців або частин стовпців. Максимальна ширина індексу 500 Біт (це значення може бути змінено при компіляції MySQL). Для індексу може використовуватися префікс поля CHAR або VARCHAR.

Встановлення з'єднань

Клієнти можуть з'єднуватися з MySQL, використовуючи сокети TCP / IP, сокети Unix або іменовані канали (named pipes, під NT).

Підтримка ODBC (Open-database-Connectivity) для Win32 (з вихідним кодом). Всі функції ODBC 2.5 і багато інших. Наприклад, для з'єднання з MySQL можна використовувати MS Access.

Локалізація

Сервер може забезпечувати повідомлення про помилки для клієнтів на різних мовах.

Повна підтримка декількох різних кодувань, включаючи ISO-8859-1 (Latin1), німецький, big5, ujis і багато інших. Наприклад, скандинавські символи дозволені в іменах таблиць і стовпців.

Для зберігання всіх даних використовується вибраний набір символів. Всі порівняння для стовпців з нормальними рядками проводяться з урахуванням регістру символів.

Сортування проводиться відповідно до обраного алфавітом (за замовчуванням використовується шведський). Цю установку можна змінити при запуску сервера MySQL. MySQL підтримує багато різних кодувань, які можна задавати під час компіляції і в процесі роботи.

Клієнти та інструментарій

Включає myisamchk, дуже швидку утиліту для перевірки, оптимізації та відновлення таблиць. Всі функціональні можливості myisamchk також доступні через SQL-інтерфейс.

Всі MySQL-програми можна запускати з опціями -- help або -? для отримання допомоги.

Ось уже протягом декількох років даний сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Незважаючи на те що MySQL постійно вдосконалюється, він вже сьогодні забезпечує широкий спектр корисних функцій. Завдяки своїй доступності, швидкості і безпеки MySQL дуже добре підходить для доступу до баз даних по Internet.

2.1.3 Інструмент візуального проектування БД MySQL Workbench

MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення та експлуатацію БД в єдине безшовне оточення для системи баз даних MySQL. Є наступником DBDesigner 4 з FabForce.

У MySQL Workbench реалізовані функції:

- Проектування таблиць;
- Проектування структури БД;
- Проектування уявлень (Views);
- Написання збережених процедур;
- Використання ролей для поділу доступу до даних;
- Документування;
- Back engineering;
- Forward engineering.

MySQL Workbench є кросплатформенним додатком. Поширюється під ліцензіями GNU GPL і пропрієтарної EULA.

2.1.4 Мова програмування PHP

PHP – скриптова мова загального призначення, що активно застосовується у сфері розробки інтерактивних Web-додатків. На даний момент є невід'ємною частиною великої кількості хостинг-провайдерів і вважається одним із лідерів серед мов програмування, що використовуються у розробці Web-ресурсів. Мова PHP створена спеціально для Web-розробки та може впроваджуватись у програмний код Web-сторінок.

Основна відмінність скриптів, написаних цією мовою, від скриптів, написаних, наприклад, C++ або Python – це те, що замість програми, яка створює HTML-код, формується програмний код з кількома впровадженими командами PHP. Для використання PHP коду на Web-сторінках його виділяють спеціальними тегами, які розташовуються на початку і в кінці коду, щоб процесор PHP міг визначити межі ділянки HTML-коду, який містить PHP.

PHP є багатофункціональною мовою, яка містить функції роботи з файловою системою, функції обробки рядків і масивів, функції роботи з датою і часом, функції роботи з протоколом HTTP і т.д. Ключовою особливістю даної мови є її відмінна взаємодія з усіма сучасними Web-технологіями та можливість підтримки великої кількості Web-протоколів, наприклад протокол FTP, протокол POP, протокол SNMP, протокол прикладного рівня для доступу до електронної пошти IMAP і т.д. Мова PHP ідеально підходить для роботи з базами даних. Більшість нинішніх СУБД має підтримку PHP.

Програма, написана на PHP, може складатися з 10000 рядків коду, а може і одного рядка. Все полягає в тому, яке перед нами поставлене завдання, яке необхідно вирішити за допомогою скрипту. Для того, щоб виконати PHP код на HTML-сторінці, необхідно почати зі спеціального символу (тега), який задасть послідовність дій (<?) і продовжити виконання цих дій до завершального тега (?>).

Для розробників PHP надає гнучкі та ефективні засоби забезпечення інформаційної безпеки, які зазвичай поділяють на дві групи: засоби рівня програми та засоби системного рівня. Механізми безпеки, написані на PHP,

знаходяться під контролем адміністраторів, і при правильному налаштуванні здатні забезпечити максимальну безпеку. Також, мовою PHP реалізовано так званий безпечний режим (safe mode), завдяки якому можна обмежити застосування коду конкретним категоріям користувачів. Наприклад, за допомогою безпечного режиму у програмістів є можливість обмежувати максимальний час, необхідний для виконання будь-якого скрипту, а також обсяг пам'яті, що використовується, так як необмежена витрата пам'яті негативно позначається на продуктивності Web-сервера. Крім цього, програміст може встановити обмеження на кількість каталогів, що використовуються, завдяки яким користувач переглядає інформацію і виконує скрипти PHP, а також переглядає конфіденційну інформацію на Webсервері за допомогою PHP скриптів.

У складі функцій мови PHP є низка надійних шифрувальних механізмів. Також код, написаний мовою PHP, сумісний із багатьма скриптами незалежних компаній. Це дає можливість легко інтегрувати його із захищеними технологіями електронної комерції.

Ще одна перевага мови PHP полягає в тому, що вихідний код скрипта PHP неможливо переглядати в Web-браузерах, тому що він компілюється безпосередньо перед відправкою на запит користувача. Це запобігає можливості розкрадання оригінального коду зловмисниками. По відношенню до вимог програміста мова PHP має достатню гнучкість та ефективність налаштування.

В силу того, що PHP код не містить у собі коду, який був би орієнтований на конкретний Web-сервер, він відмінно взаємодіє з сучасними серверами, такими як сервер Netscape Enterprise Server, сервер Microsoft IIS, сервер Apache, сервер Stronghold та інші.

2.1.5 Інструменти розробки користувальницького графічного інтерфейсу

Найбільш простим і логічним способом організації «тонкого» клієнта, є подання інформації за допомогою веб-браузера. Найбільш перспективним для

цього, в контексті активної розробки HTML5, в порівнянні, наприклад з Flash, бачиться зв'язка HTML і CSS, до яких, при необхідності, легко можна додати динамічні елементи на JavaScript.

HTML

HTML-стандартна мова розмітки. Більшість веб-сторінок створюються за допомогою мови HTML. Мова HTML інтерпретується браузером і відображається у вигляді документа, в зручній для людини формі.

HTML є додатком («окремим випадком") SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879.

За допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, в HTML внесена підтримка гіпертексту. Мультимедійні можливості були додані пізніше. Спочатку мова HTML був задуманий і створений як засіб структурування і форматування документів без їх прив'язки до засобів відтворення (відображення). В ідеалі, текст з розміткою HTML повинен був без стилістичних і структурних спотворень відтворюватися на обладнанні з різною технічною оснащеністю (кольоровий екран сучасного комп'ютера, монохромний екран органайзера, обмежений за розмірами екран мобільного телефону або пристрою і програми голосового відтворення текстів). Однак сучасне застосування HTML дуже далеко від його початкового завдання. З плином часу, основна ідея платформонезалежності мови HTML була віддана в своєрідну жертву сучасним потребам в мультимедійному і графічному оформленні.

Текстові документи, що містять код на мові HTML (такі документи традиційно мають розширення .html або .htm), обробляються спеціальними додатками, які відображають документ в його форматованому вигляді. Такі додатки, звані " браузерами, зазвичай надають користувачеві зручний інтерфейс для запиту веб-сторінок, їх перегляду (і виведення на інші зовнішні пристрої) і, при необхідності, відправки введених користувачем даних на сервер.

CSS. CSS (каскадні таблиці стилів) - технологія опису зовнішнього вигляду документа, написаного мовою розмітки. CSS використовується для

завдання кольорів, шрифтів, розташування та інших аспектів представлення документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншою мовою розмітки) і представлення документа (написаного на CSS). Цей поділ може збільшити доступність документа, надати більшу гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне представлення, друк, читання голосом (спеціальним голосовим браузером або програмою читання з екрану).

Переваги:

- Кілька дизайнів сторінки для різних пристроїв перегляду.
- Зменшення часу завантаження сторінок сайту за рахунок перенесення правил подання даних в окремий CSS-файл. У цьому випадку браузер завантажує тільки структуру документа і дані, що зберігаються на сторінці, а подання цих даних завантажується браузером тільки один раз і можуть бути закешовані.
- Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл.
- Додаткові можливості оформлення. Наприклад, за допомогою CSS-верстки можна зробити блок тексту, який інший текст буде обтікати (наприклад для меню) або зробити так, щоб меню було завжди видно при прокручуванні сторінки.

Недостатки:

- Різне відображення верстки в різних браузерах (особливо застарілих), які по різному інтерпретують одні і ті ж дані CSS.
- Часто зустрічається необхідність на практиці виправляти не тільки один CSS-файл, але і теги HTML і програмний код, які складним і не наочним способом пов'язані з селекторами CSS, що іноді зводить нанівець простоту застосування єдиних файлів стилів, і значно подовжує час редагування і тестування.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРАКТИЧНОЇ ЧАСТИНИ

3.1 Розробка бази даних

3.1.1 Логічна модель даних

Для побудови логічної моделі даних використовувався CASE-засіб ERWin, який дозволяє проектувати реляційні моделі даних. Найбільш поширеним засобом моделювання даних є діаграми "сутність-зв'язок" (ERD). З їх допомогою визначаються важливі для предметної області об'єкти (сутності), їх властивості (атрибути) і відносини один з одним (зв'язку). Модель даних представлена у вигляді ER-діаграми на Рис. 3.1.

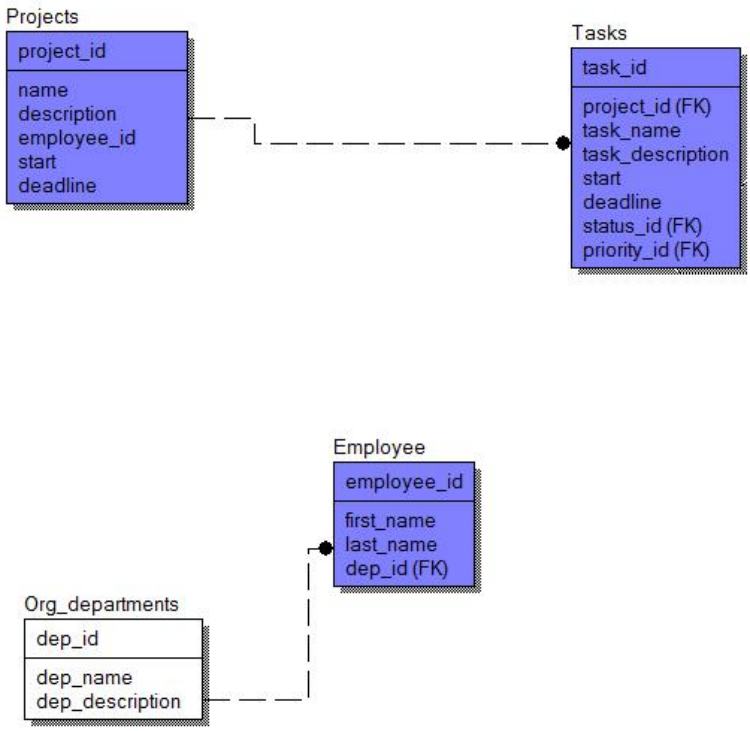

Рис. 3.1 «Полегшена» логічна схема БД

З метою підвищення наочності в логічній моделі свідомо опущена частина неключових атрибутів.

3.1.2. Фізична модель БД

На основі спроектованої логічної моделі, і з урахуванням обраного програмно-апаратного забезпечення, побудована фізична модель бази даних, представлена на рис. 3.2. Для побудови фізичної моделі бази даних використовувався додаток MySQL Workbench.

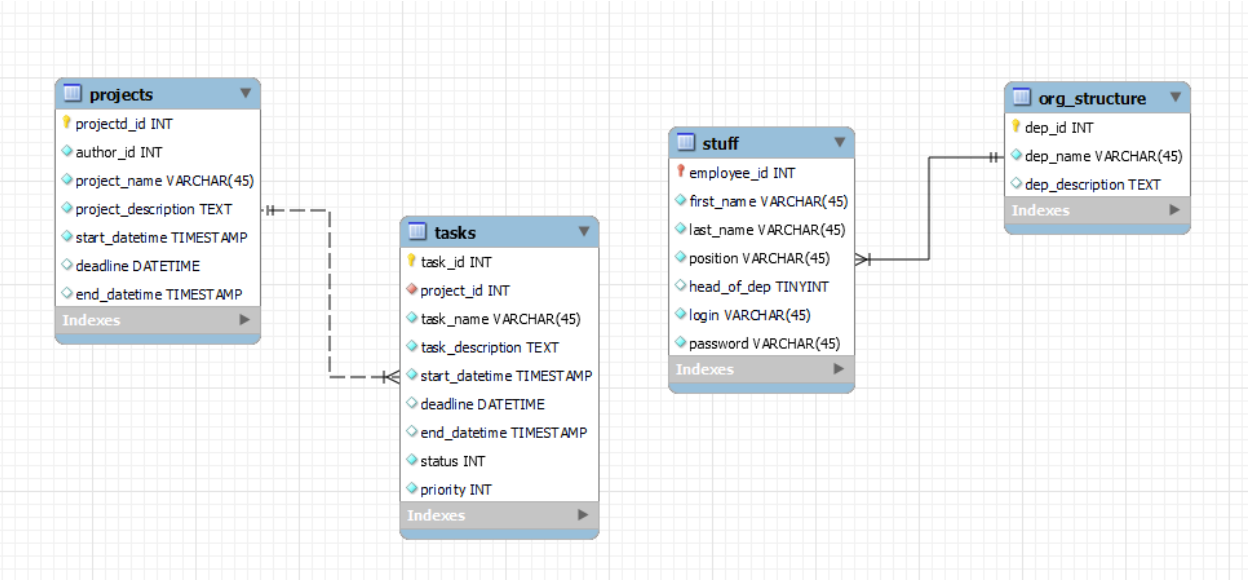

Рис. 3.2 Повна фізична схема БД

Повна структура сутностей БД представлена в таблиці 3.1.

Таблиця 3.1. Структура таблиць БД

Сутність	Опис		
Projects	Містить проекти організації		
Атрибут	Ключ	Тип	Опис
Project_id	K	int	Ідентифікатор проекту
Author_id		int	Ідентифікатор автора проекту
Project_name		varchar(45)	Назва проекту
Project_description		text	Детальний опис проекту
Start_datetime		timestamp	Дата створення проекту
Deadline		datetime	Дата необхідного завершення
End_datetime		timestamp	Дата фактичного завершення

Сутність	Опис		
Tasks	Містить завдання проектів		
Атрибут	Ключ	Тип	Опис
Task_id	K	int	Ідентифікатор завдання в системі
Project_id	FK	int	Ідентифікатор батьківського проекту
Task_name		varchar(45)	Ім'я завдання
Task_description		text	Опис завдання
Start_datetime		timestamp	Дата створення завдання
Deadline		datetime	Дата необхідного завершення
End_datetime		timestamp	Дата фактичного завершення
Next_task		int	Ідентифікатор залежної задачі
Status	FK	int	Ідентифікатор статусу завдання
Priority	FK	int	Ідентифікатор пріоритету завдання
Сутність	Опис		
Stuff	Містить дані про співробітників організації		
Атрибут	Ключ	Тип	Опис
Employee_id	K	int	Ідентифікатор співробітника
First_name		varchar(45)	Ім'я співробітника
Last_name		varchar(45)	Прізвище співробітника
Department	FK	int	Ідентифікатор підрозділу, в якому складається співробітник
Position		varchar(45)	Посада
Head_of_dep		boolean	Прапор начальника підрозділу
Login		varchar(45)	Логін для входу в систему
Password		varchar(45)	Пароль для входу в систему
Сутність	Опис		
Org_structure	Містить список підрозділів компанії		

Атрибут	Ключ	Тип	Опис
Dep_id	K	int	Ідентифікатор підрозділу
Dep_name		varchar(45)	Назва підрозділу
Dep_description		text	Опис підрозділу

3.2 Розробка графічного інтерфейсу користувача

Однією із завдань дипломного проекту є розробка користувальницького графічного інтерфейсу. Ключовою складовою зручності інтерфейсу користувача є продумана система навігації. Від того, наскільки проста, зрозуміла та зручна система навігації, багато в чому залежить якість роботи працівників організації. У системі відображається велика кількість інформації, яка потрібна для роботи співробітникам організації, і важливим є те, де і в якому вигляді інформація представлена.

Хорошим тоном вважається:

• можливість для відвідувача потрапити з головної сторінки сайту до найважливіших розділів за 1 клік;

• можливість із будь-якої сторінки сайту за 1 клік повернутися на головну сторінку сайту;

• можливість дістатися від будь-якої сторінки сайту до будь-якої сторінки сайту не більше ніж за 3 кліки;

• дублювання системи навігації (наприклад, зверху та знизу); використання кількох альтернативних способів навігації – щоб відвідувач міг потрапити на сторінку кількома способами – з меню, через систему пошуку тощо.

• облік технічних можливостей різних користувачів: система навігації повинна дозволяти переміщатися сайтом користувачам, у яких відключена графіка, не відображаються флеш-елементи тощо.

• візуальне виділення елементів навігації на тлі інших елементів сайту.

• для спрощення - використання стандартних елементів навігації, наприклад стандартних піктограм.

Інтерфейс системи спроектовано з урахуванням цих вимог.

3.2.1 Приклади інтерфейсу для різних ролей користувачів

У системі є три основних типи користувачів, залежно від яких, після авторизації, будуть запропоновані сторінки різного виду.

Адміністратор системи з встановленим обліковим записом admin має доступ до додавання користувачів системи. Інтерфейс користувача-адміністратора представлений на рис. 3.3.

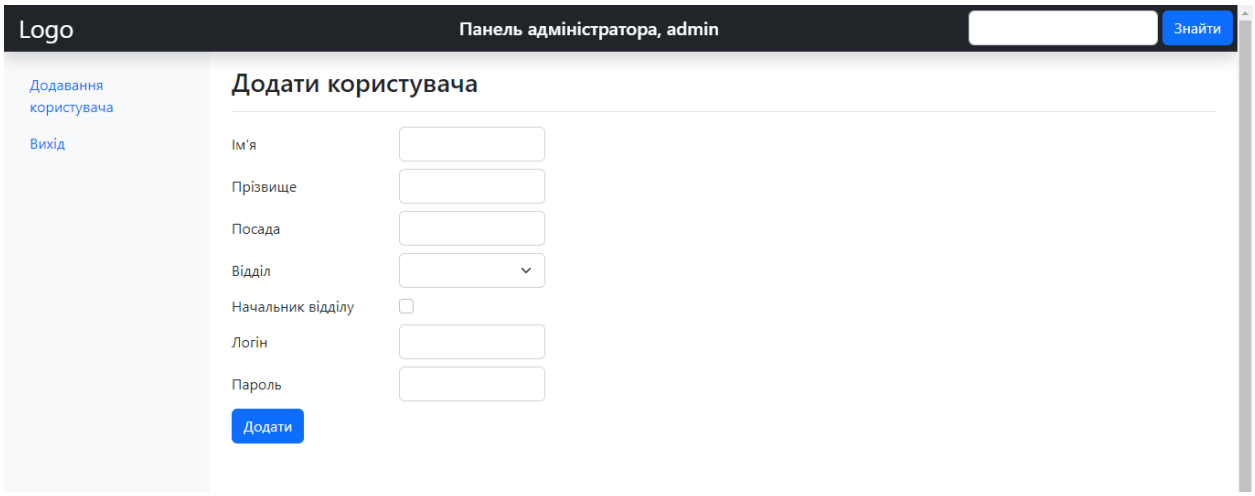

Рис. 3.3. Інтерфейс адміністратора. Панель додавання нового користувача

Рядовий співробітник з правами користувача має доступ із правами перегляду до більшості довідкової інформації в системі, а також списку своїх завдань. Інтерфейс користувача з логіном Ivan Ivanov представлений на рис. 3.4.

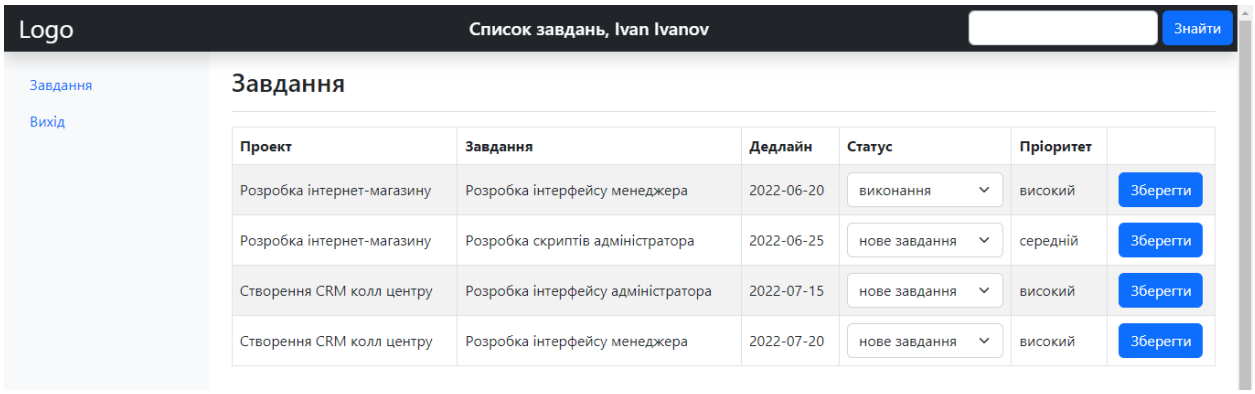

Рис. 3.4. Інтерфейс співробітника. Список завдань

Користувач з правами голови підрозділу одночасно має доступ до функцій звичайного співробітника, так і до розділу управління проектами: створення нового проекту, видалення проекту, створення завдання для певного проекту, видалення завдання в певному проекті. Інтерфейс користувача - голови підрозділу з логіном Alexey Ivanov представлений на рис. 3.5.

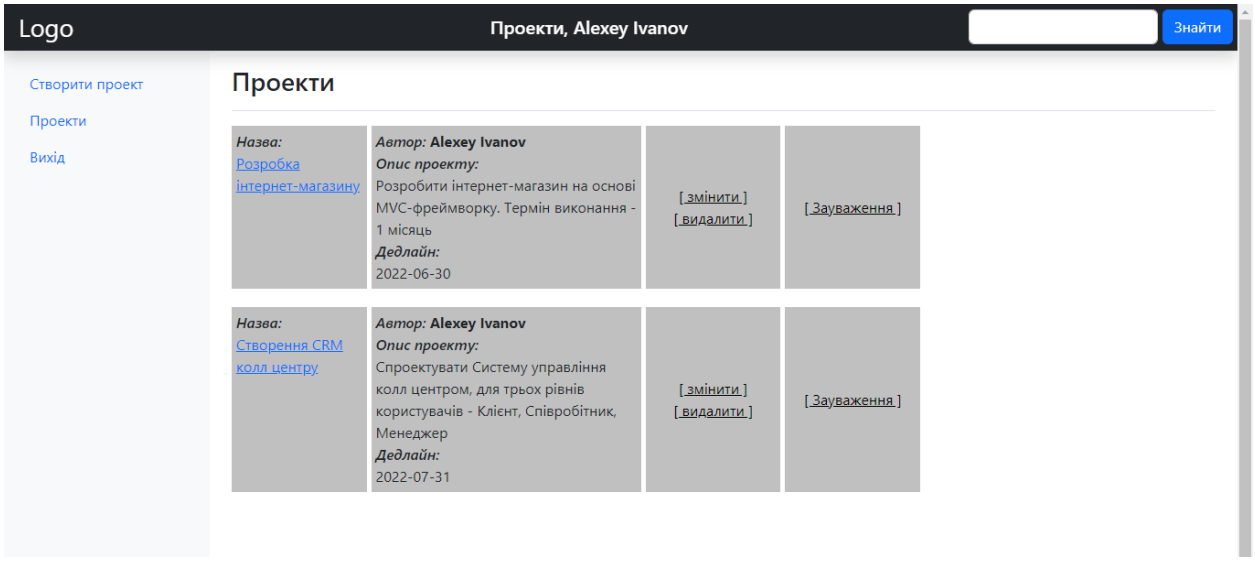

Рис. 3.5. Інтерфейс голови підрозділу. Список проектів

3.3 Тестування програмного продукту

Тестування програмного забезпечення – це дослідження, проведене з метою надання зацікавленим сторонам інформації про якість досліджуваного програмного продукту або послуги.

Таким чином, розробивши веб-застосунок можна зробити тестування. Тестування веб-застосунку здійснювалося на власному пристрої.

У даній роботі було застосоване ручне тестування програмного продукту.

При введенні логіну та паролю admin користувач потрапляє в панель адміністратора.

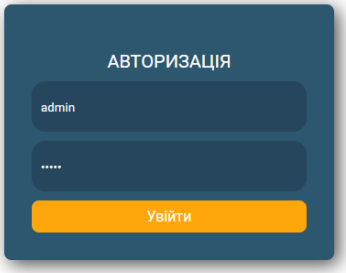

Рис. 3.6 Введення логіна і пароля admin

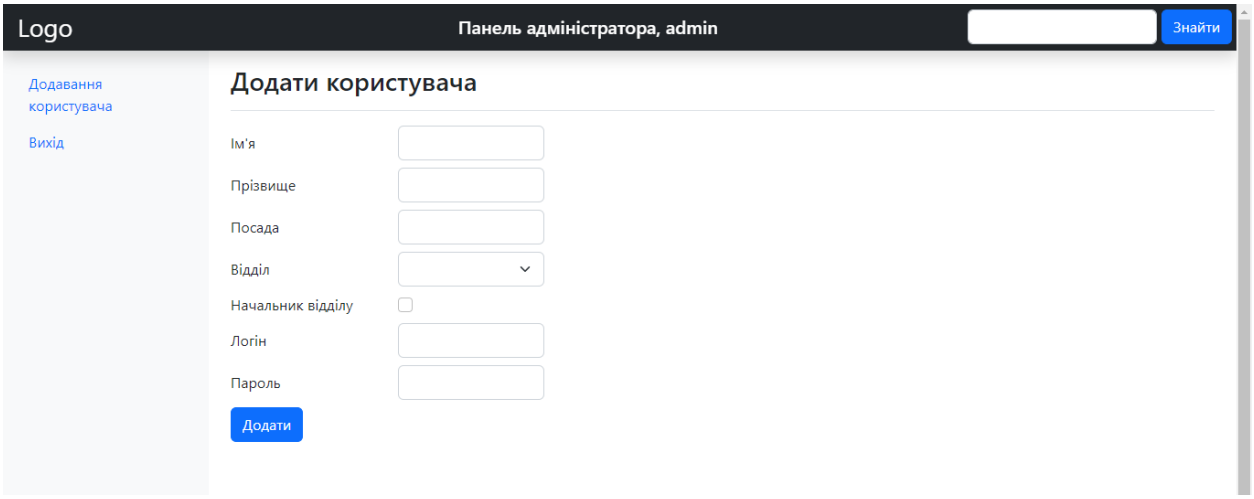

Рис. 3.7 Панель адміністратора

У вікні панелі адміністратора користувач може додати нових користувачів системи.

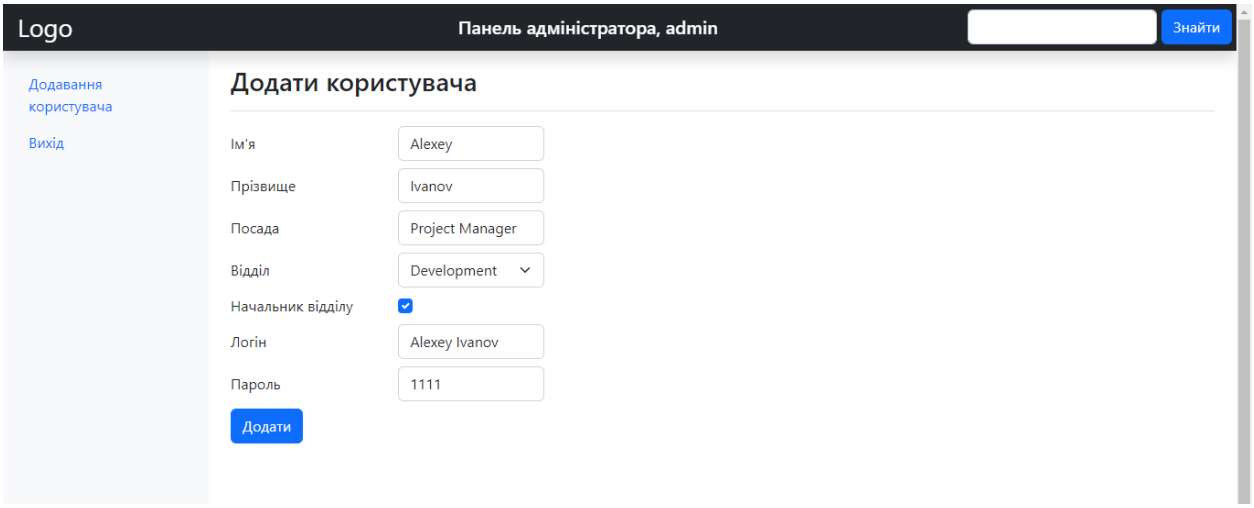

Рис. 3.8 Додавання нового користувача

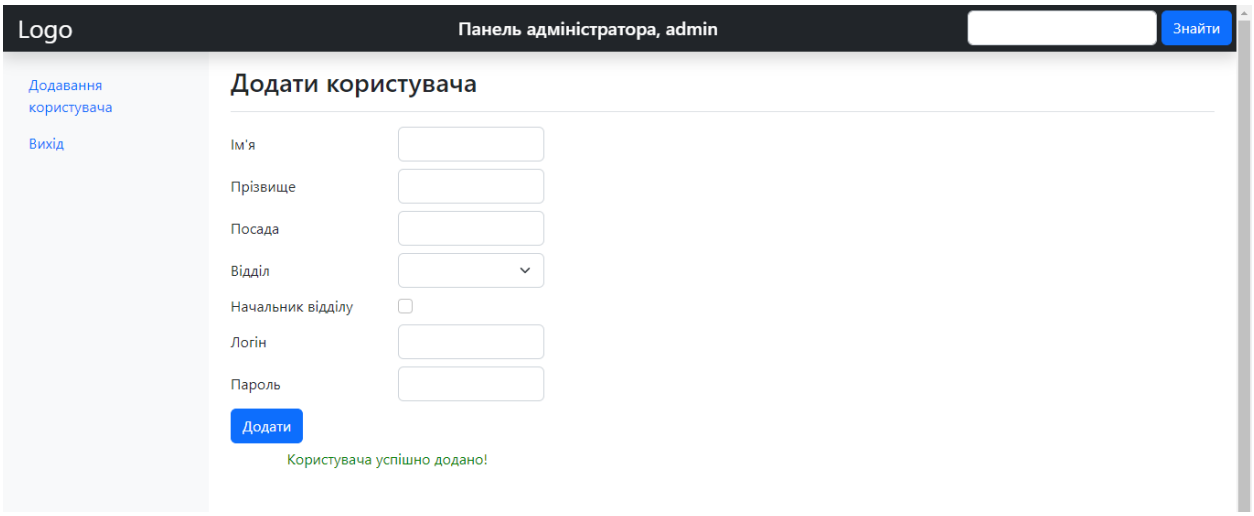

Рис. 3.9 Користувача успішно додано

Тепер нові додані користувачі можуть успішно авторизовуватися в системі.

Наступним рівнем доступу до АІС є начальник відділу. При введенні логіну та паролю начальника відділу користувач потрапляє у вікно адміністрування для начальників відділу рис. 3.11.

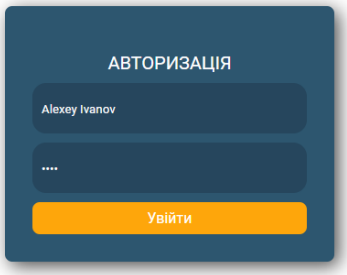

Рис. 3.10 Введення логіна і пароля начальника відділу

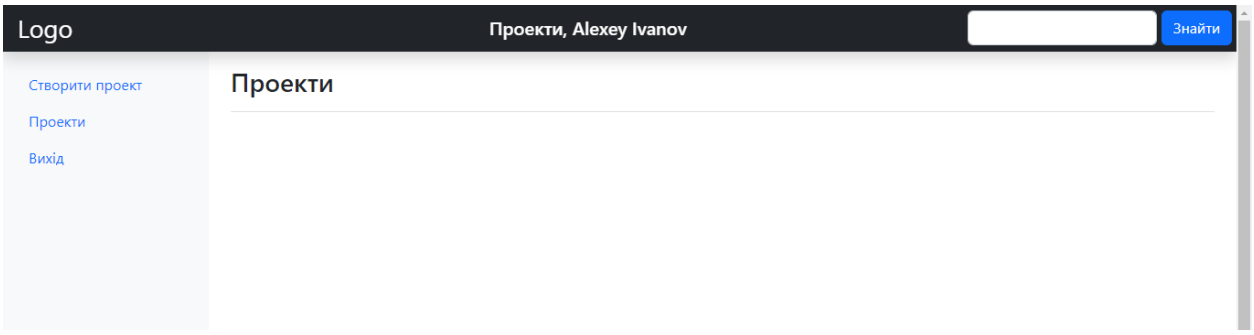

Рис. 3.11 Панель начальника відділу. Список проектів

В панелі начальника відділу користувач може створити новий проект, видалити проект, створити завдання для певного проекту, видалити завдання в певному проекті.

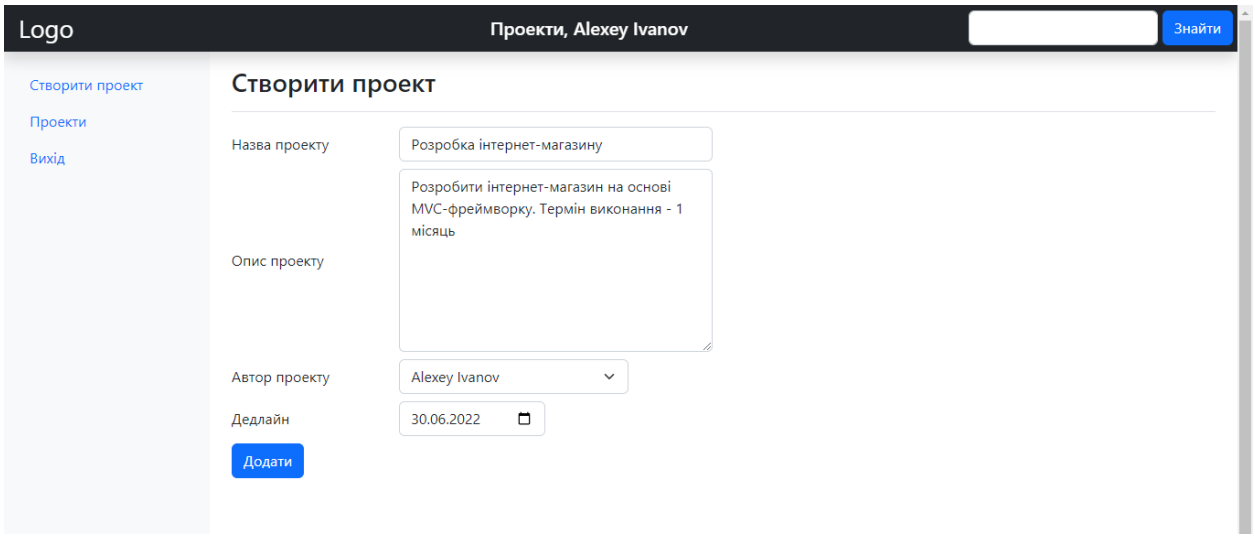

Рис. 3.12 Створення нового проекту

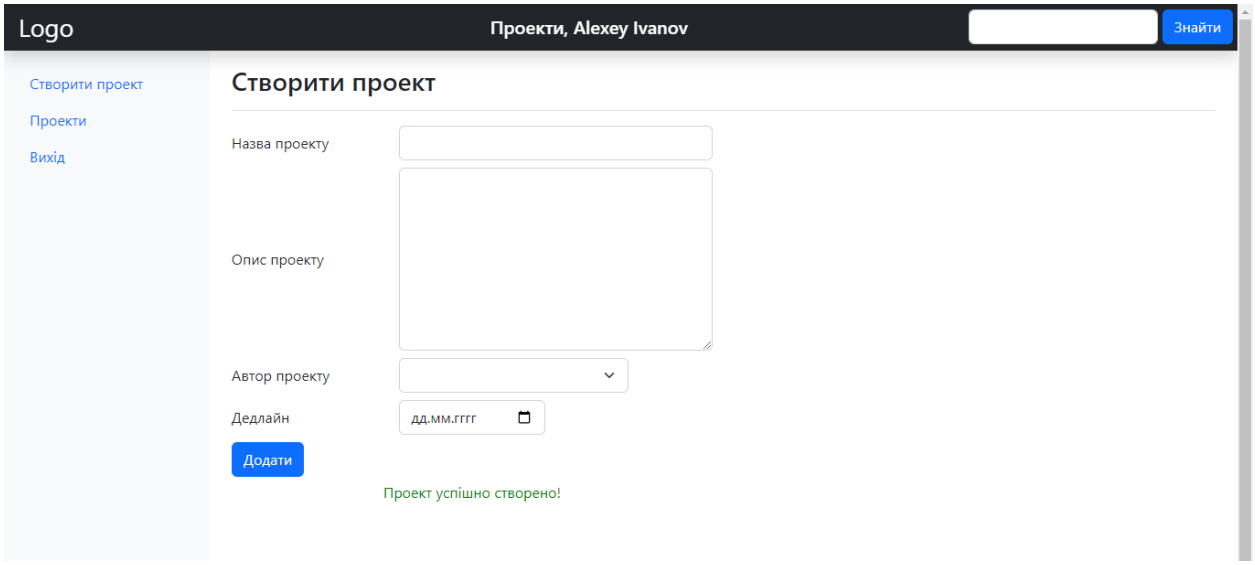

Рис. 3.13 Проект успішно створено

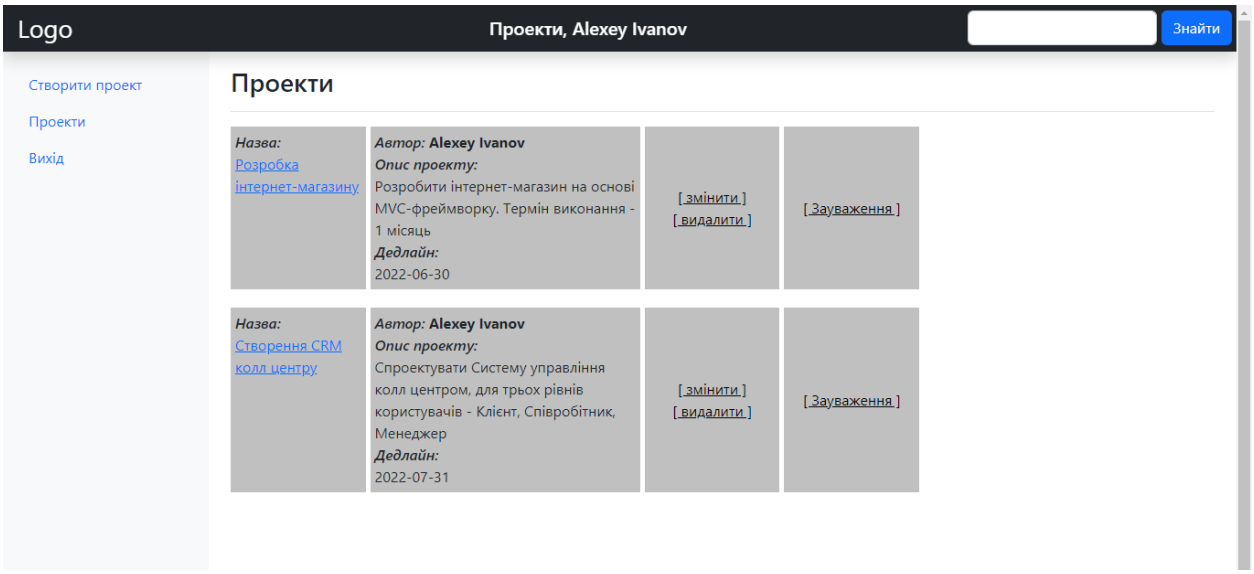

Рис. 3.14 Список проектів

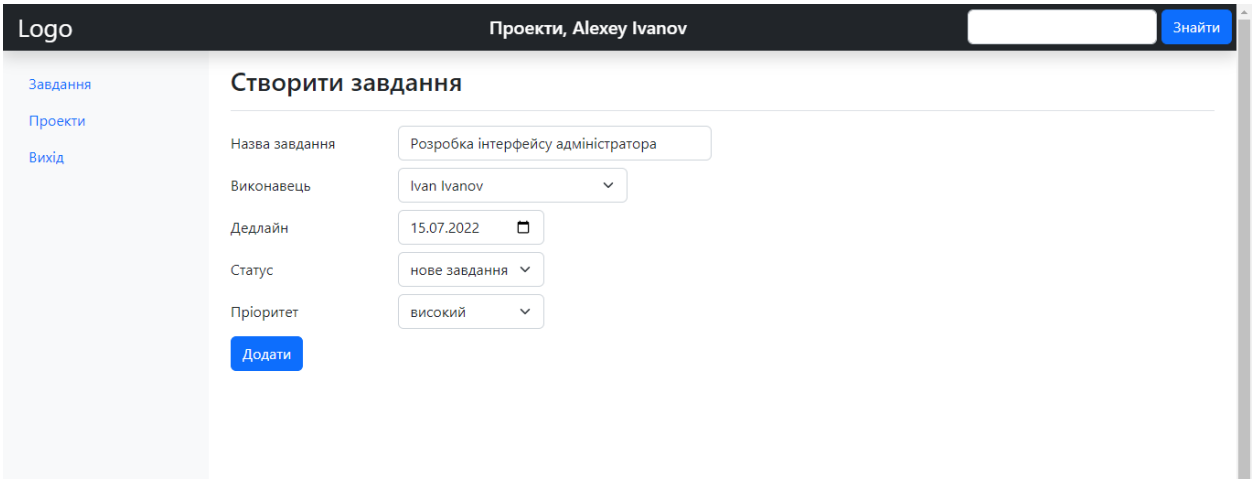

Рис. 3.15 Створення завдання проекту

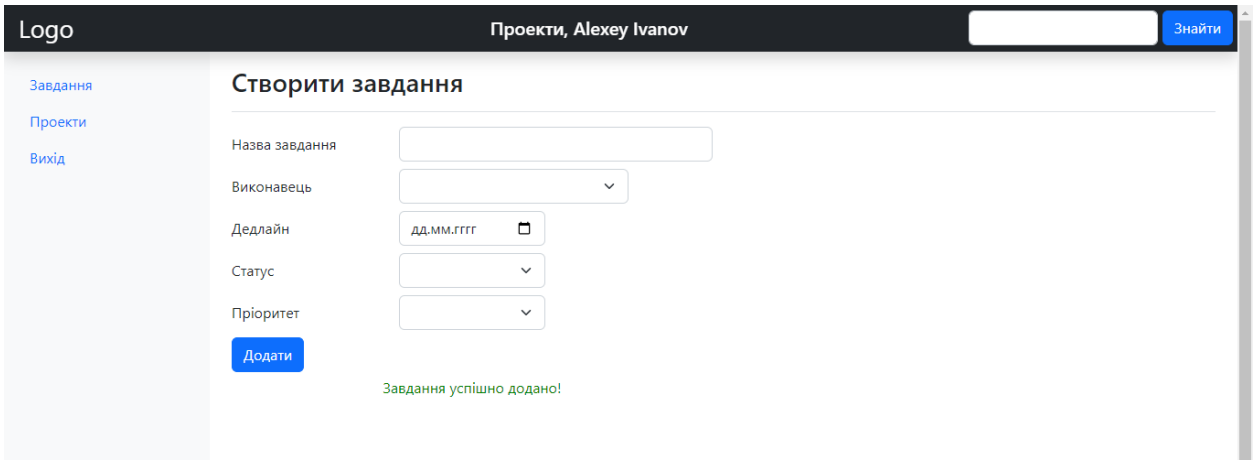

Рис. 3.16 Завдання успішно додано

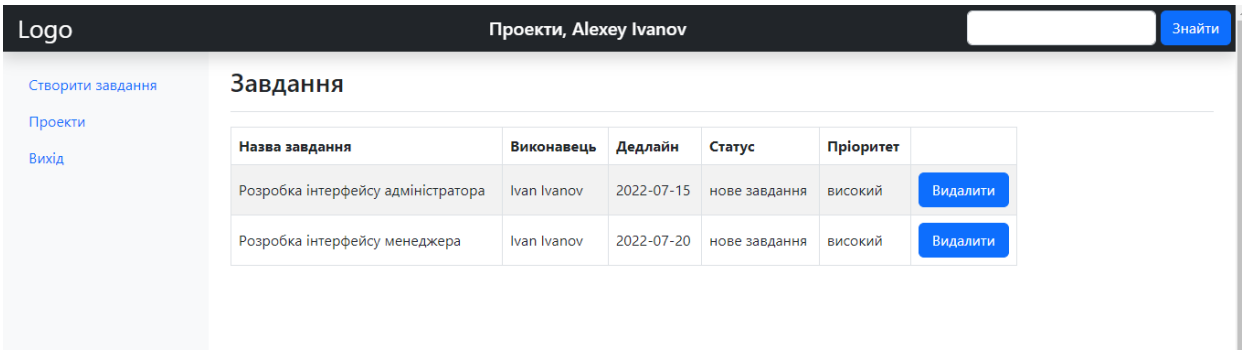

Рис. 3.17 Список завдань проекту

Третій рівень доступу до АІС – це співробітник. Вікно адміністрування для співробітника представлене на рис. 3.19.

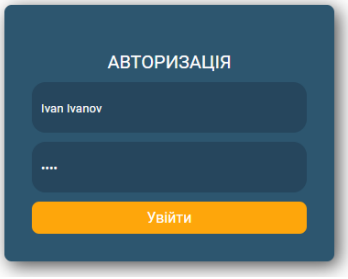

Рис. 3.18 Введення логіна і пароля співробітника

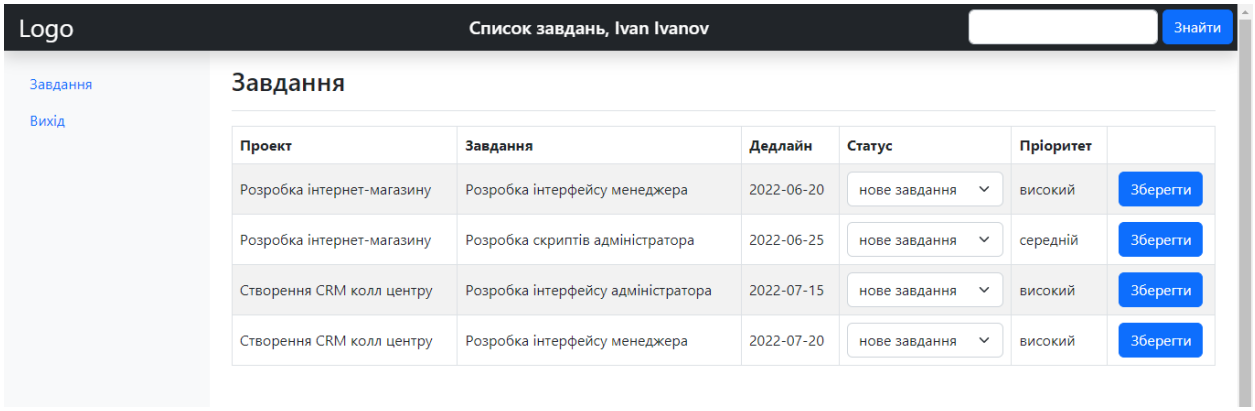

Рис. 3.19 Панель співробітника. Список завдань

В панелі співробітника користувач може переглядати завдання, змінювати статус завдань.

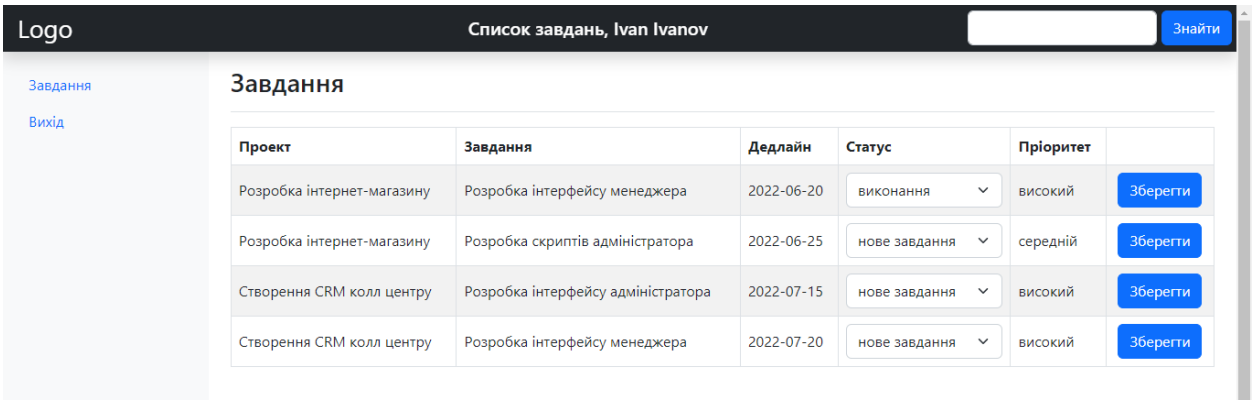

Рис. 3.20 Змінений список завдань

Таким чином розроблена автоматизована інформаційна система групового планування та контролю за виконанням завдань підтверджує свою працездатність, ефективність та зручність у використанні.

ВИСНОВКИ

В результаті виконання дипломної роботи було досягнуто наступних результатів:

Розглянуто представлені на ринку системи групового планування та контролю за виконанням завдань, зроблено висновок про недостатню реалізацію в них функцій планування та контролю завдань. Висунуто вимоги до системи, що розробляється, обґрунтовано необхідність розробки.

Вивчено методології управління проектами.

Розглянуто основні принципи організації системи, що розробляється, обґрунтовано вибір інструментарію розробки системи: СУБД MySQL, мова сценаріїв PHP, засоби проектування та розробки AllFusion Data Modeler і MySQL Workbench, технології розробки інтерфейсу HTML, CSS.

На основі висунутих вимог до функціональності системи, розроблені логічна та фізична модель бази даних, наведено повний опис сутностей БД та їх атрибутів. Розроблено прототипи інтерфейсів для користувачів із різними привілеями. Наведено приклад їх використання.

За результатами виконаної роботи, можна зробити висновок, що поставлені завдання вирішені і мета даного дипломного проекту можна вважати досягнутою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. CSS. *Wikipedia :* веб-сайт. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення 12.11.2021)
2. HTML. *Wikipedia :* веб-сайт. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення 10.10.2021)
3. HTTP. *Wikipedia :* веб-сайт. URL: https://uk.wikipedia.org/wiki/HTTP (дата звернення 12.12.2021)
4. MySQL. *Wikipedia :* веб-сайт. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення 10.04.2022)
5. PHP. *Wikipedia :* веб-сайт. URL: https://uk.wikipedia.org/wiki/PHP (дата звернення 10.04.2022)
6. PHP. *Керівництво по PHP – Manual :* веб-сайт. URL: http://php.net/manual/ru/ (дата звернення 22.05.2022)
7. Ручне тестування. *Wikipedia :* веб-сайт. URL: https://ru.wikipedia.org/wiki/Ручное_тестирование (дата звернення 13.09.2022)
8. Аткінсон Л. MySQL. Бібліотека професіонала, Вільямс, 2002, 624 стор.
9. Васвані В.Повний довідник з MySQL, - М.: Видавничий дім "Вільямс", 2006 - 526 с.
10. Гудвін Г.К., С.Ф. Гребі, М.Е. Сальдаго "Проектування систем управління"; пров. з англ. - М.: БІНОМ, Лабораторія знань, 2004. - 911 с.
11. Девід Фленаган «JavaScript. Детальний посібник», Символ-Плюс, 2008 р., 992 стор.
12. Іссі Коен Лазаро «Повний довідник з HTML, CSS та JavaScript. Довідник професіонала», ЕКОМ Паблішерз, 2007, 1168 стор.
13. Курята О.С. *Розробка автоматизованої інформаційної системи групового планування та контролю за виконанням завдань* : зб. Матеріалів доп. учасн. XV Всеукр. наук.- практ. конф., 1 листопада 2022 р. Рівне: РДГУ, 2022. С.166.

14. Мазур І.І., Шапіро В.Д., Ольдерогге Н.Д. "Управління проектами". - Омега - Л, 2004.
15. Харрінгтон Дж. Л. Проектування реляційних баз даних, - М.: Лорі, 2005. - 230 с.
16. Шенк, Джеффрі Д. Технологія клієнт-сервер та її застосування - М.: Лорі, 1995, - 418с.

ДОДАТКИ

ДОДАТОК А
СЕРТИФІКАТ УЧАСНИКА XIV ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРОФЕСІЙНІЙ ДІЯЛЬНОСТІ»

Міністерство освіти і науки України
Департамент освіти і науки Рівненської ОДА
Громадська спілка «Рівне ІТ-освіта»
Рівненський державний гуманітарний університет

СЕРТИФІКАТ № 44-22

учасника
XV Всеукраїнської науково-практичної конференції
"ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРОФЕСІЙНІЙ ДІЯЛЬНОСТІ"

1 листопада 2022 року, м. Рівне

Курята Олександр

Декан факультету
математики та інформатики РДГУ доц. Шахрайчук М.І.

ДОДАТОК Б

Лістинг файлу signin.php:

```
<?php
   require "db.php";
   $data = $_POST;
   $showError = FALSE;

   if(isset($data['signin'])){
      $errors = array();
      $showError = TRUE;

      if(trim($data['username']) == ''){
         $errors[] = "Вкажіть ім'я користувача!";
      }
      if(trim($data['password']) == ''){
         $errors[] = "Вкажіть пароль!";
      }

      $id = 1;
      $user = R::findOne('users', 'id = ?', [$id]);
      if($user){
         if($data['username'] == $user->username && $data['password'] == $user->password){
            $_SESSION['user'] = $user;
            header('Location: index.php');
         }else{
            $errors[] = 'Невірний пароль!';
         }
      }else{
         $errors[] = 'Користувач не знайдений!';
      }

      $employee = R::findOne('staff', 'login = ?', array($data['username']));
      if($employee){
         if($employee->login == $data['username'] && $employee->password == $data['password'] &&
$employee->head_of_dep == '0' || $employee->login == $data['username'] && $employee->password ==
$data['password'] && $employee->head_of_dep == NULL){
```

```
            $_SESSION['employee'] = $employee;
            header('Location: employee.php');
        }elseif($employee->login == $data['username'] && $employee->password == $data['password'] &&
$employee->head_of_dep == '1'){
            $_SESSION['headOfDep'] = $employee;
            header('Location: headOfDep.php');
        }else{
            $errors[] = 'Невірний пароль!';
        }
    }else{
        $errors[] = 'Користувач не знайдений!';
    }
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Авторизація</title>
  <link rel="stylesheet" href="css/style.css">
  <style>
    * {
      font-family: "Roboto", sans-serif;
    }
  </style>
</head>
<body>
  <div class="content">
    <form action="signin.php" method="post" class="auth_form">
      <p>Авторизація</p>
      <input type="text" name="username" placeholder="Ім'я користувача" value="<?php echo
@$data['username'] ?>"><br>
      <input type="password" name="password" placeholder="Пароль" value="<?php echo
@$data['password'] ?>"><br>
      <button type="submit" name="signin">Увійти</button>
```

```
        </form>
        <p class="error"><?php if($showError) {echo showError($errors);} ?></p>
    </div>
</body>
</html>
```

Лістинг файлу addProject.php:

```
<?php
  require "db.php";
  if(!R::testConnection()) die('No DB connection!');

  $headOfDep = R::findOne('staff', 'id = ?', array($_SESSION['headOfDep']->id));

  $data = $_POST;
  date_default_timezone_set('Europe/Kiev');

  if(isset($data['add_project'])){
    $project = R::dispense('projects');
    $project->project_name = $data['projectname'];
    $project->project_description = $data['projectDescr'];
    $project->start_datetime = date('Y-m-d H:i:s');
    $project->deadline = $data['deadline'];
    R::store($project);
    $successfulMessage = 'Проект успішно створено!';
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Створити проект</title>
    <link rel="stylesheet" href="css/bootstrap-reboot.min.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/style.css">
</head>
```

```
<body>
  <header class="navbar navbar-dark sticky-top bg-dark flex-md-nowrap p-0 shadow">
    <a class="header__logo navbar-brand col-md-3 col-lg-2 me-0 px-3" href="#">Logo</a>
    <div class="header__info">
     <h5>Проекти, <?php echo $headOfDep->login ?></h5>
    </div>
    <div class="header__search d-flex">
      <input class="form-control" type="text">
      <button class="btn btn-primary">Знайти</button>
    </div>
  </header>

  <div class="container-fluid">
    <div class="row">
     <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
      <div class="position-sticky pt-3">
       <ul class="nav flex-column">
        <li class="nav-item">
         <a class="nav-link" href="addProject.php">
          Створити проект
         </a>
        </li>
        <li class="nav-item">
         <a class="nav-link active" aria-current="page" href="headOfDep.php">
          Проекти
         </a>
        </li>
        <li class="nav-item">
         <a class="nav-link" href="#">
          Очищення сесій
         </a>
        </li>
        <li class="nav-item">
         <a class="nav-link" href="#">
          Статус
         </a>
        </li>
        <li class="nav-item">
```

```
            <a class="nav-link" href="#">
              Пріоритет
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">
              Зовнішні організації
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">
              Календар
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">
              Робочі дні
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">
              Структура організації
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="logout.php">
              Вихід
            </a>
          </li>
        </ul>
      </div>
    </nav>

    <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4"><div class="chartjs-size-monitor"><div class="chartjs-size-monitor-expand"><div class=""></div></div><div class="chartjs-size-monitor-shrink"><div class=""></div></div></div>
      <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
```

```
  <h1 class="h3">Створити проект</h1>
</div>

<form action="addProject.php" method="POST">
  <div class="row align-items-center mb-2">
    <div class="col-2">
      <label for="projectname">Назва проекту</label>
    </div>
    <div class="col-4">
      <input class="form-control" id="projectname" type="text" name="projectname">
    </div>
  </div>
  <div class="row align-items-center mb-2">
    <div class="col-2">
      <label for="projectDescr">Опис проекту</label>
    </div>
    <div class="col-4">
      <textarea style="height: 200px;" class="form-control" id="projectDescr" type="text"
name="projectDescr"></textarea>
    </div>
  </div>
  <div class="row align-items-center mb-2">
    <div class="col-2">
      <label for="projectAuthor">Автор проекту</label>
    </div>
    <div class="col-3">
      <select class="form-select" name="projectAuthor" id="projectAuthor" aria-label="Default
select example">
        <?php
          $headsOfDepartment = R::getAll('SELECT * FROM `staff`');
          foreach($headsOfDepartment as $headOfDepartment){
            if($headOfDepartment['head_of_dep'] == '1'){
              echo '
                <option>' . $headOfDepartment['login'] . '</option>
              ';
            }
          }
        ?>
```

```
                </select>
              </div>
            </div>
            <div class="row align-items-center mb-2">
              <div class="col-2">
                <label for="deadline">Дедлайн</label>
              </div>
              <div class="col-2">
                <input class="form-control" id="deadline" type="date" name="deadline">
              </div>
            </div>
            <div class="row">
              <div class="col-2">
                <button type="submit" name="add_project" class="btn btn-primary">Додати</button>
              </div>
            </div>
            <div class="row">
              <div class="col-6" style="color: green; text-align: center; margin-top: 5px;">
                <?php echo $successfulMessage; ?>
              </div>
            </div>
          </form>

        </main>
      </div>
    </div>

    <?php
      require "footer.php";
    ?>

    <script src="js/script.js"></script>
</body>
</html>
```

Лістинг файлу adduser.php:

```
<?php
    require "db.php";
```

```
    if(!R::testConnection()) die('No DB connection!');

    $data = $_POST;

    if(isset($data['add_user'])){
        $employee = R::dispense('staff');
        $employee->first_name = $data['firstname'];
        $employee->last_name = $data['lastname'];
        $employee->department = $data['department'];
        $employee->position = $data['position'];
        $employee->head_of_dep = $data['headOfDep'];
        $employee->login = $data['login'];
        $employee->password = $data['password'];
        R::store($employee);
        header('Location: index.php');
        $_SESSION['successfulMessage'] = 'Користувача успішно додано!';
    }
?>
```

Лістинг файлу addTask.php:

```
<?php
  require "db.php";
  if(!R::testConnection()) die('No DB connection!');

  $headOfDep = R::findOne('staff', 'id = ?', array($_SESSION['headOfDep']->id));

  $data = $_POST;
  date_default_timezone_set('Europe/Kiev');

  if(isset($data['add_task'])){
    $task = R::dispense('tasks');
    $task->project_id = $_SESSION['project_id'];
    $task->task_name = $data['task_name'];
    $task->performer = $data['performer'];
    $task->start_datetime = date('Y-m-d H:i:s');
    $task->deadline = $data['deadline'];
    $task->status = $data['status'];
    $task->priority = $data['priority'];
```

```
    R::store($task);
    $successfulMessage = 'Завдання успішно додано!';
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Створити завдання</title>
    <link rel="stylesheet" href="css/bootstrap-reboot.min.css">
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <header class="navbar navbar-dark sticky-top bg-dark flex-md-nowrap p-0 shadow">
        <a class="header__logo navbar-brand col-md-3 col-lg-2 me-0 px-3" href="#">Logo</a>
        <div class="header__info">
          <h5>Проекти, <?php echo $headOfDep->login ?></h5>
        </div>
        <div class="header__search d-flex">
            <input class="form-control" type="text">
            <button class="btn btn-primary">Знайти</button>
        </div>
    </header>

    <div class="container-fluid">
        <div class="row">
          <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
            <div class="position-sticky pt-3">
              <ul class="nav flex-column">
                <li class="nav-item">
                  <a class="nav-link" href="project.php">
                    Завдання
                  </a>
                </li>
```

```
<li class="nav-item">
  <a class="nav-link active" aria-current="page" href="headOfDep.php">
    Проекти
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Очищення сесій
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Статус
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Пріоритет
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Зовнішні організації
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Календар
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Робочі дні
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">
    Структура організації
```

```
            </a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="logout.php">
              Вихід
            </a>
          </li>
        </ul>
      </div>
    </nav>

    <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4"><div class="chartjs-size-monitor"><div class="chartjs-size-monitor-expand"><div class=""></div></div><div class="chartjs-size-monitor-shrink"><div class=""></div></div></div>
      <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
        <h1 class="h3">Створити завдання</h1>
      </div>

      <form action="addTask.php" method="POST">
        <div class="row align-items-center mb-2">
          <div class="col-2">
            <label for="task_name">Назва завдання</label>
          </div>
          <div class="col-4">
            <input class="form-control" id="task_name" type="text" name="task_name">
          </div>
        </div>
        <div class="row align-items-center mb-2">
          <div class="col-2">
            <label for="performer">Виконавець</label>
          </div>
          <div class="col-3">
            <select class="form-select" name="performer" id="performer" aria-label="Default select example">
              <?php
                $performers = R::getAll('SELECT * from `staff`');
                foreach($performers as $performer){
```

```
                    if($performer['head_of_dep'] == NULL || $performer['head_of_dep'] == '0'){
                      echo '
                        <option>' . $performer['login'] . '</option>
                      ';
                    }
                  }
                ?>
              </select>
            </div>
          </div>
          <div class="row align-items-center mb-2">
            <div class="col-2">
              <label for="deadline">Дедлайн</label>
            </div>
            <div class="col-2">
              <input class="form-control" id="deadline" type="date" name="deadline">
            </div>
          </div>
          <div class="row align-items-center mb-2">
            <div class="col-2">
              <label for="status">Статус</label>
            </div>
            <div class="col-2">
              <select class="form-select" name="status" id="status" aria-label="Default select example">
                <option selected value="нове завдання">нове завдання</option>
                <option value="виконання">виконання</option>
                <option value="виконано">виконано</option>
              </select>
            </div>
          </div>
          <div class="row align-items-center mb-2">
            <div class="col-2">
              <label for="priority">Пріоритет</label>
            </div>
            <div class="col-2">
              <select class="form-select" name="priority" id="priority" aria-label="Default select
example">
                <option selected value="середній">середній</option>
```

```
                    <option value="високий">високий</option>
                  </select>
                </div>
              </div>
              <div class="row">
                <div class="col-2">
                  <button type="submit" name="add_task" class="btn btn-primary">Додати</button>
                </div>
              </div>
              <div class="row">
                <div style="color: green; text-align: center; margin-top: 5px;" class="col-6">
                  <?php echo $successfulMessage; ?>
                </div>
              </div>
            </form>

          </main>
        </div>
    </div>

    <?php
      require "footer.php";
    ?>

    <script src="js/script.js"></script>
</body>
</html>
```