

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

**на тему: «Дослідження та розробка ІС управління ІТ-проектами з DevOps
на прикладі АТ КБ «ПриватБанк»»**

Виконав: студент групи 6141 м

____  ____ **Морарь М.І.**
(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)

 ____ **Партас В.К.**
(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

В. Ажищев

к.т.н, доц. Ажищев В.Ф.

" ____ " _____ 2025 р.

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

Студенту Морарю Максиму Івановичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження та розробка ІС управління ІТ-проектами з DevOps на прикладі АТ КБ «ПриватБанк»

Керівник роботи к.т.н. доцент Партас Віктор Кирилович

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

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

Дослідження ІС управління ІТ-проектами з DevOps на прикладі АТ КБ «ПриватБанк». Проєктування ІС управління ІТ-проектами з DevOps на прикладі АТ КБ «ПриватБанк». Розробка ІС управління ІТ-проектами з DevOps на прикладі АТ КБ «ПриватБанк». Техніко-економічне обґрунтування розробки. Охорона праці. Охорона навколишнього середовища

5. Перелік презентаційних матеріалів Недоліки роботи, Постановка задачі, Завдання дослідження, Вимоги до функцій інформаційної системи управління ІТ-проектами, Варіант концепції системи з використанням «десктопного» клієнта, Варіант концепції системи з використанням браузерного клієнта, Порівняльна характеристика концепцій, Діаграма потоків даних,

Концептуальна модель даних, Фізична модель даних, Схема взаємодії модулів, Модель розгортання системи, Екранні форми розробки, Апробація, Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Партас В.К., доцент	05.09.2024	22.12.2024
2	Партас В.К., доцент	23.12.2024	15.06.2025
3	Партас В.К., доцент	15.06.2025	25.09.2025
4	Партас В.К., доцент	25.09.2025	14.11.2025
5		25.09.2025	14.11.2025
6		25.09.2025	14.11.2025

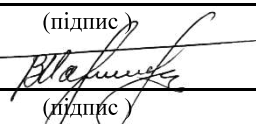
7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент

Керівник роботи


(підпис)

(підпис)

Морарь М.І.
(ПІБ)

Партас В.К.
(ПІБ)

Анотація

Магістерська робота присвячена дослідженню, проєктуванню та розробці інформаційної системи управління ІТ-проєктами з урахуванням принципів DevOps на прикладі АТ КБ «ПриватБанк». У роботі проаналізовано сучасні підходи до організації життєвого циклу програмного забезпечення, визначено ключові проблеми, що виникають під час управління ІТ-проєктами у великих фінансових установах, та обґрунтовано необхідність інтеграції DevOps-практик для підвищення ефективності процесів. Розроблена інформаційна система забезпечує автоматизацію основних процесів управління проєктами, включаючи планування, контроль виконання, моніторинг якості, управління змінами та безперервну інтеграцію й доставку програмних продуктів.

У роботі побудовано архітектуру системи, виконано моделювання бізнес-процесів і користувацьких сценаріїв, реалізовано функціональні компоненти, що сприяють підвищенню продуктивності команд, зменшенню часу на випуск релізів та покращенню якості програмного забезпечення. Представлені результати впровадження демонструють зростання рівня автоматизації, прозорості та контролюваності процесів у DevOps-середовищі банку.

Запропонована система може бути використана як основа для подальшої оптимізації процесів управління ІТ-проєктами та розробки інструментів підтримки прийняття рішень у великих організаціях із розгалуженою ІТ-інфраструктурою.

Робота представлена у вигляді 90 сторінок машинного тексту, що має у своєму складі 23 рисунки, 3 таблиці і 5 додатків, а також код програми.

Ключові слова: DevOps, інформаційна система, управління ІТ-проєктами, автоматизація, CI/CD, бізнес-процеси, ПриватБанк.

Abstract

The master's thesis is devoted to the research, design and development of an information system for managing IT projects taking into account DevOps principles using the example of JSC CB "PrivatBank". The work analyzes modern approaches to organizing the software life cycle, identifies key problems that arise during the management of IT projects in large financial institutions, and justifies the need to integrate DevOps practices to increase the efficiency of processes. The developed information system provides automation of the main project management processes, including planning, execution control, quality monitoring, change management, and continuous integration and delivery of software products.

The work builds the system architecture, models business processes and user scenarios, implements functional components that contribute to increasing team productivity, reducing release time, and improving software quality. The presented implementation results demonstrate an increase in the level of automation, transparency, and controllability of processes in the bank's DevOps environment.

The proposed system can be used as a basis for further optimization of IT project management processes and development of decision support tools in large organizations with extensive IT infrastructure.

The work is presented in the form of 90 pages of machine text, which includes 23 figures, 3 tables and 5 appendices, as well as program code.

Keywords: DevOps, information system, IT project management, automation, CI/CD, business processes, PrivatBank.

Зміст

Вступ.....	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА РОЗРОБКА ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DevOps НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»	9
1.1 Аналіз предметної області.....	9
1.2 Аналіз існуючих рішень в галузі управління ІТ проектами	12
1.2.1 Atlassian JIRA.....	13
1.2.2 Redmine.....	13
1.2.3 Trac.....	13
1.2.4 Порівняльна характеристика технологій	14
1.3 Варіанти концепцій системи управління ІТ проектами	15
1.3.1 Варіант концепції системи з використанням «десктопного» клієнта..	15
1.3.2 Варіант концепції системи з використанням браузерного клієнта	15
1.4 Постановка задачі.....	16
Висновки до розділу 1.....	17
РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DEVOPS НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»	19
2.1 Інформаційне забезпечення.....	19
2.1.1 Організаційна і функціональна структура.....	19
2.1.2 Побудова концептуальної моделі даних	21
2.1.3 Побудова логічної моделі даних.....	22
2.1.4 Побудова фізичної моделі даних	23
2.2 Технічне забезпечення	25
2.2.1 Визначення технічних засобів системи	25
2.2.2 Схема взаємодії модулів.....	26
2.2.3 Модель розгортання системи.....	27
2.3 Програмне забезпечення.....	28
2.3.1 Мова програмування.....	28
2.3.2 Інструмент управління та складання проекту	29
2.3.3 Інструмент створення web-інтерфейсу	30
2.3.4 Середовище програмування.....	31
2.3.5 Сервер бази даних	34

2.3.6 Сервер прикладних програм	35
Висновки до розділу 2.....	36
РОЗДІЛ 3. РОЗРОБКА ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DEVOPS НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»	38
Висновки до розділу 3.....	40
РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ	42
4.1. Мета та необхідність техніко-економічного обґрунтування	42
4.2. Визначення технічних вимог до системи	42
4.3. Оцінка витрат на розробку	42
4.4. Економічні вигоди від впровадження системи	43
4.5. Оцінка ефективності проєкту.....	44
Висновки до розділу 4.....	44
РОЗДІЛ 5. ОХОРОНА ПРАЦІ. ВИМОГИ ДО ПАРАМЕТРІВ МІКРОКЛІМАТУ ТА ВЕНТИЛЯЦІЇ РОБОЧИХ ПРИМІЩЕНЬ	45
5.1 Вимоги до параметрів мікроклімату робочих приміщень	45
5.1.1 Розкриття поняття виробниче приміщення та мікроклімат.....	45
5.1.2 Вимоги до параметрів мікроклімату виробничих приміщень.....	46
5.1.3 Оптимальні і допустимі умови мікроклімату виробничого приміщення.	46
5.2 Вимоги до вентиляції робочих приміщень.....	49
5.2.1 Природна вентиляція	49
5.2.2 Штучна вентиляція.....	50
5.2.3 Основні вимоги до систем вентиляції.....	54
Висновок до розділу 5.....	55
РОЗДІЛ 6. ОХОРОНА ОТОЧУЮЧОГО СЕРЕДОВИЩА.....	57
6.1. Вступ.....	57
6.2. Екологічні аспекти використання ІТ-систем у корпоративному середовищі.....	57
6.3. Екологічні переваги впровадження DevOps-підходу	57
6.4. Використання енергоефективних технологій	58
6.5. Управління електронними відходами	59
6.6. Заходи щодо мінімізації впливу на довкілля.....	59
Висновки до розділу 6.....	59
ВИСНОВКИ.....	60

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТОК А – Технічне завдання	64
ДОДАТОК Б – Інструкція користувача	67
ДОДАТОК В – Технічний опис	72
ДОДАТОК Г – Фрагмент програмного коду.....	74

ВСТУП

Управління проектами є невід'ємною частиною повсякденної діяльності керівників різного рівня. Багато керівників необхідність використання формалізованих методів управління проектами пов'язують з великими проектами. Однак, навіть в організаціях, орієнтованих на серійне виробництво, реалізація проектів становить значну частину діяльності.

Застосування формалізованих методів управління проектами дозволяє більш обґрунтовано визначати цілі інвестицій і оптимально планувати інвестиційну діяльність, більш повно враховувати проектні ризики, оптимізувати використання наявних ресурсів і уникати конфліктних ситуацій, контролювати виконання складеного плану, аналізувати фактичні показники і вносити своєчасну корекцію в хід робіт, накопичувати, аналізувати і використовувати надалі досвід реалізованих проектів. Таким чином, система управління проектами є однією з найважливіших компонент всієї системи управління організацією.

Системи автоматизації управління проектами — це програмні системи, що дозволяють автоматизувати одну або декілька складових управління проектами: складання календарного плану проекту, розподілення задач між співробітниками, а також стеження за процесом виконання задач та проекту в цілому.

Новітні комп'ютерні технології дозволяють створити зручну та надійну систему управління проектами з великим набором можливостей. Це різні види задач, функціональний і в той же час дружній інтерфейс для створення, перегляду та обробки задач, слідкування за змінами в проекті та багато іншого.

Дана система буде розроблена для автоматизації управління проектами інформаційних технологій. Вона буде надавати можливість створювати, переглядати, редагувати задачі, слідкувати за змінами в проекті за допомогою генератора звітів. Буде реалізовано систему авторизації користувача, що зробить неможливість неавторизованого доступу як до задач, так і до проектів. Інтерфейс користувача повинен бути максимально спрощений в частині елементів управління. Це спрощення з одного боку не повинно заважати інтуїтивній зрозумілості для користувачів різних ступенів знайомства з комп'ютерними технологіями, а з іншого приховувати в собі потужний інструмент для роботи з задачами.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА РОЗРОБКА ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DevOps НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»

1.1 Аналіз предметної області

«ПриватБанк» - один з найбільших банків України. Національна мережа банківського обслуговування ПриватБанку включає понад 3 000 філій і відділень на всій території України.

Основну частину корпоративних клієнтів, що обслуговуються банком, складають підприємства малого і середнього бізнесу. При цьому значну долю вжитку банківських продуктів забезпечують крупні фінансові і промислові корпорації. Особлива увага протягом року приділялася обслуговуванню компаній, що мають широку мережу структурних підрозділів, — мережевих корпорацій. Галузева характеристика клієнтської бази банку свідчить, що велику частину складають підприємства, основний вид діяльності яких торгівля і комерційна діяльність. Значну долю серед корпоративних клієнтів займають підприємства харчової, транспортної, сільськогосподарської і будівельної галузей [1].

Для успішного та комфортного обслуговування клієнтів банку потрібно своє спеціалізоване програмне забезпечення, та автоматизовані робочі місця. Для вирішення цієї проблеми існує відділ розробки програмного забезпечення «Біплан», який розробляє та підтримує внутрішні програмні комплекси банку.

В відділі інформаційних технологій у «Біплан» функціонує трирівнева система управління. Найвищим органом управління є так званий «бізнес» - відділ, який аналізує потреби клієнтів, економіку країни, банківську структуру та планує нововведення у комплексах банку. Запланувавши нововведення відділ засобами службової записки передає задачу на виконання у відділ розробки програмного забезпечення. Зазначаючи в ній підстави для розробки, строки виконання, задачу, та додаткову інформація, якщо така необхідна.

В свою чергу відділ розробки програмного забезпечення отримавши задачу від бізнесу, аналізує її та відправляє службову записку відповідальному бізнес аналітику, який повинен дослідити задачу та на її основі створити технічне завдання. На основі технічного завдання бізнес аналітик розбиває задачу на під задачі та розподіляє їх між співробітниками (програмістами) службовими записками, надаючи їм також технічне завдання. В службових записках вказується дата початку, дата завершення задачі, оціночний час на виконання, завдання до виконання, та додаткова інформація якщо така необхідна [1].

Отримавши службову записку співробітник з урахуванням строків виконання задачі та власного графіку навантаження планує приблизний план виконання задач. Після виконання деякої задачі співробітник надає бізнес аналітику службову записку з кінцевою датою реалізації задачі, затраченим часом та ресурсами, описом розробки.

Отримавши усі відповіді про завершення виконання задачі, бізнес аналітик звітує про завершення розробки «бізнесу».

Зі сторони розроблювальної системи, починаючи новий проект керівник повинен створити його в системі управління проектами, для цього він заповнює форму створення проекту:

- *Назва проекту* – унікальний ідентифікатор проекту;
- *Опис проекту* – пояснення до проекту;
- *Дата початку* – дата коли було почато проект;
- *Дата завершення* – дата завершення роботи над проектом;

Після чого додає до проекту співробітників котрі будуть приймати в ньому участь.

Проаналізувавши завдання проекту, керівник створює задачі шляхом заповнення форми створення задачі:

- *Назва задачі* – назва задачі для пошуку в таблиці задач;
- *Опис задачі* – опис того що треба зробити;

– *Тип задачі*

- Покращення – внесення змін в існуючі модулі проекту для покращення їх функціонування.

- Розробка – додавання нового функціоналу в проект.

- Помилка – некоректна робота деякого модуля або функції проекту.

- Тестування – перевірка роботи спроможності модулів або функцій проекту;

– *Пріоритет задачі*

- Низький – не виконання даної задачі не несе негативного впливу на проект.

- Нормальний – пріоритет задачі яка повинна бути реалізована до кінцевої дати проекту, не виконання даної задачі може понести негативний вплив на проект.

- Високий – задача яка повинна бути реалізована як най скоріше, не виконання задачі понесе негативний вплив на проект, можливо без виконання даної задачі неможливо буде почати виконувати деякі інші.

- Критичний – задача найвищого пріоритету, не виконання даної задачі веде до не роботи спроможності проекту;

– *Статус задачі*

- Відкрита – задача не була прийнята на виконання.

- В роботі – співробітник почав виконання даної задачі.

- Вирішена – співробітник закінчив виконання задачі, але вона ще не буда протестована.

- На тестуванні – задача була виконана співробітником, і тепер знаходиться на тестуванні.

- Закрита – задача повністю реалізована та протестована.

- Відхилена – виконання даної задачі більше не потребується.

– *Дата початку* – дата початку виконання задачі;

- *Дата завершення* – дата завершення виконання задачі;
- *Час на виконання* – час затрачений на виконання задачі;
- *Процент готовності* – процент виконання задачі;
- *Відповідальний* – співробітник відповідальний за виконання даної задачі;

В свою чергу співробітник отримавши задачі по пріоритетній черзі починає їх виконувати, поступово поновлюючи стан задач, а саме змінюючи значення статусу та процента виконання задачі.

Після чого за керівником залишається лише спостереження за виконанням задач проекту, для цього можна скористатись засобами пошуку та перегляду конкретних задач або сформувати звіт проекту [1].

Звіт проекту це сформована системою статистика по ходу виконання задач проекту:

- *Кількість вирішених задач* – кількість задач проекту, які на момент формування звіту були в стані вирішена, закрита або відхилена.
- *Кількість відкритих задач* – кількість задач проекту, які на момент формування звіту були в стані відкрита, в роботі або на тестуванні.
- *Кількість прострочених задач* – кількість відкритих задач проекту, у яких дата завершення раніше поточної дати.

1.2 Аналіз існуючих рішень в галузі управління ІТ проектами

Враховуючи проблеми з організації управління ІТ проектами в АТ КБ «ПриватБанк», розглянемо вже існуючі системи управління проектами.

Існуючі системи, що працюють за цими напрямками покликані вирішити проблеми контролю виконання задач, централізованого збереження та доступності задач в мережі, допомагають за коротші строки передавати задачі між співробітниками [2].

Для вирішення тих чи інших проблем існують технології управління ІТ проектами, такі як: Atlassian JIRA, Redmine, Trac.

1.2.1 Atlassian JIRA

Atlassian JIRA – система відстежування помилок, призначена для організації спілкування з користувачами, хоча в деяких випадках може бути використана для управління проектами.

Розроблена компанією Atlassian Software System. Платна. Має веб-інтерфейс. Назва системи (JIRA) отримано шляхом модифікації японської назви Годзіла ("Gojira"), що в свою чергу є алюзією на назву конкуруючого продукту — Bugzilla. JIRA створювалася в якості заміни Bugzilla і багато в чому повторює архітектуру Bugzilla.

Система дозволяє працювати з декількома проектами. Для кожного з проектів створює та веде схеми безпеки і схеми оповіщення.

Всередині компанії «Atlassian Software Systems» для управління процесом розробки використовується «стіна смерті». «Стіна смерті» — це дошка, на яку чіпляються роздруківки запитів користувачів з JIRA і по стану якої відстежується хід розробки. Після закінчення розробки, програмісти інформують користувачів про результати за допомогою JIRA [2].

1.2.2 Redmine

Redmine — вільний серверний веб-застосунок для управління проектами та відстеження помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на Ruby і є застосунком розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми [3].

1.2.3 Trac

Trac — вільний, оснований на веб-інструмент для управління проектом.

Він включає в собі: основу на веб платформу для управління subversion-репозиторієм; Вікі систему, для опису помилок які виникають, виконання заміток по ходу розробки, а також просто документування; Bug-Tracker.

В основному Trac розроблений для проектів по розробці програмного забезпечення, але також може бути використаний як звичайна вікі-система.

Trac забезпечує інтерфейс для Subversion. Проте існують також наробки для інших систем управління версіями, як Mercurial, Darcs та Perforce.

Trac є розроблений на мові програмування python та розповсюджується по модифікованій ліцензії BSD

1.2.4 Порівняльна характеристика технологій

З метою визначення технології, що якнайбільше повною мірою дозволяє подолати існуючі проблеми і є більш актуальною для цієї сфери – виконано порівняльну характеристику за критеріями, найбільш важливими для системи управління ІТ проектів АТ КБ «ПриватБанк», покликаної вирішити вищезазначені проблеми.

Таблиця 1 – Порівняльна таблиця характеристик технологій управління ІТ проектами.

Технологія	Безкоштовна	Ведення декількох проектів	LDAP автентифікація	Неможливість самореєстрації
Atlassian JIRA	-	+	+	+
Redmine	+	+	+	-
Trac	+ (Для некомерційних цілей)	-	+	+

Як к бачимо технологія Atlassian JIRA є комерційною, а система Trac є частково комерційною тому вони обидві нам не підходять. Система Redmine дозволяє само реєстрацію користувачів, що не є добре у закритих комерційних проектах [3].

Отже враховуючи це та специфіку даної предметної області, є необхідність в розробці власної системи.

1.3 Варіанти концепцій системи управління ІТ проектами

1.3.1 Варіант концепції системи з використанням «десктопного» клієнта

Концепція (Рисунок 1.1) заснована на трирівневій архітектурі з використанням «десктопного клієнта» - клієнтської прикладної програми, серверу прикладних програм та серверу бази даних.

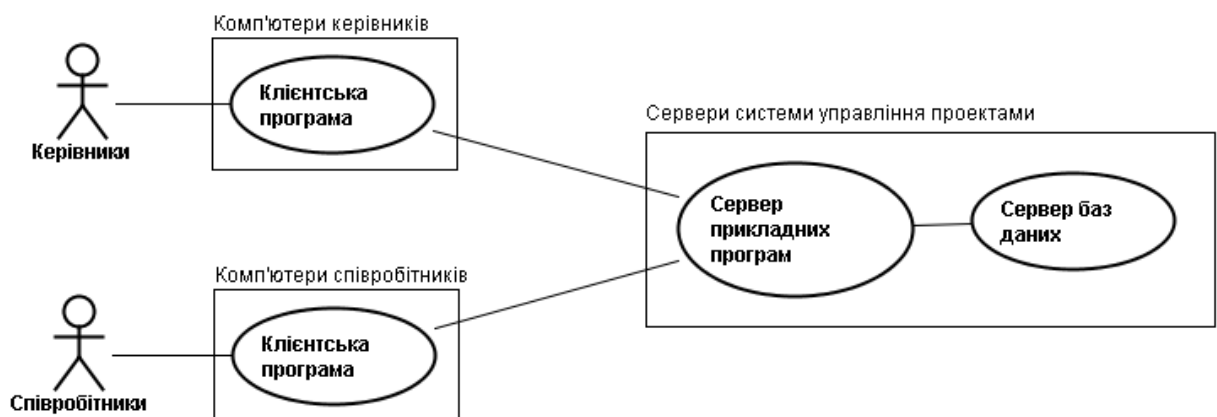


Рисунок 1.1 – Концепція з використанням «десктопного клієнта»

1.3.2 Варіант концепції системи з використанням браузерного клієнта

Концепція (Рисунок 1.2) також заснована на трирівневій архітектурі з використанням браузера як клієнту, серверу веб-додатків та серверу баз даних.

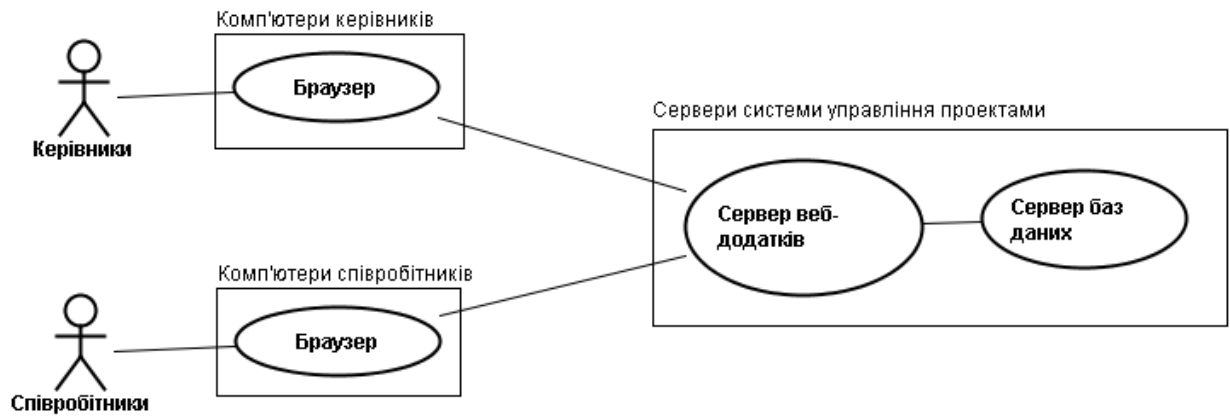


Рисунок 1.2 – Концепція з використанням браузерного клієнта

Порівняльну характеристику даних концепцій зображено в таблиці 2.

Таблиця 2 – Порівняльна характеристика концепцій

Концепція	Простота в обслуговуванні	Простота в розповсюдженні	Потреба в додатковому ПЗ	Платформозна незалежність
«десктопний» клієнт	-	-	+	-
браузерний клієнт	+	+	-	+

Для побудови системи мною було обрано другу концепцію у зв'язку з тим, що відсутня потреба у встановленні додаткового програмного забезпечення (необхідним є лише веб-браузер), спрощує обслуговування системи та процес розповсюдження.

1.4 Постановка задачі

Дана система повинна автоматизувати процес управління проектами інформаційних технологій для спрощення процесу управління проектом, підвищувати ефективність управління проектами, надавати зручні інструменти для спостереження за станом задач та проекту в цілому. Допомогти ефективно формувати послідовність виконання співробітником поставлених задач.

Даний програмний продукт повинен мати наступні функції:

- Функція авторизації для входу в систему тільки зареєстрованих

користувачів;

- Функція реєстрації нових користувачів системи;
- Функції створення, редагування та видалення проєктів;
- Функції створення, редагування та видалення задач;
- Функції пошуку задач та проєктів;
- Функція складання та перегляду звітів по проєкту.

Вимоги до надійності системи:

- Резервне копіювання бази даних;
- Тимчасове блокування форми входу в систему при 5 разовому вводі неправильного пароля.
- Паролі користувачів повинні зберігатися в зашифрованому вигляді;
- Всі дані, введені користувачем повинні перевірятися на коректність.

Висновки до розділу 1

У першому розділі було проведено всебічний аналіз особливостей організації процесів управління ІТ-проєктами в контексті діяльності АТ КБ «ПриватБанк» та окреслено ключові проблеми, що стримують ефективність розробки і супроводу програмного забезпечення. Встановлено, що традиційні підходи до управління проєктами не забезпечують достатнього рівня гнучкості, прозорості та швидкості реакції на зміни бізнес-вимог, що є критично важливим для фінансової установи з високими вимогами до стабільності та безпеки ІТ-рішень.

У результаті дослідження недоліків поточної організації роботи виявлено такі основні проблеми: фрагментованість процесів, відсутність єдиного середовища взаємодії команд розробки та експлуатації, низький рівень автоматизації розгортання, тестування та моніторингу, складність відстеження змін і контроль якості релізів. Ці недоліки призводять до

збільшення часу розробки, високої ймовірності виникнення помилок, затримок у випуску оновлень і надмірного навантаження на команди підтримки.

У межах розділу також проаналізовано альтернативні концепції та моделі організації процесів розробки ПЗ, серед яких гнучкі методології (Agile/Scrum), Lean-підхід, а також сучасні практики DevOps. Проведене порівняння показало, що саме DevOps як інтеграційна концепція, що поєднує автоматизацію, безперервну інтеграцію та доставку, культуру співпраці й прозорість процесів, найбільш повно відповідає потребам ПриватБанку та дозволяє усунути виявлені недоліки.

Отже, результати аналітичного дослідження підтверджують доцільність впровадження підходів DevOps як основи для розробки та вдосконалення інформаційної системи управління ІТ-проектами. Застосування DevOps-орієнтованої концепції забезпечить оптимізацію процесів, підвищення якості продуктів, скорочення часу розробки та підвищення загальної ефективності роботи ІТ-підрозділу АТ КБ «ПриватБанк».

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DEVOPS НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»

2.1 Інформаційне забезпечення

2.1.1 Організаційна і функціональна структура

Розробка діаграми варіантів використання переслідує мету:

- Визначити загальні межі і контекст модельованої наочної області на початкових етапах проектування системи.
- Сформулювати загальні вимоги до функціональної поведінки проектованої системи.
- Розробити початкову концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- Підготувати початкову документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. У свою чергу, варіант використання служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, ініційованих системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів з системою [4].

У процесі свого функціонування система взаємодіє з різними користувачами для виконання ними різних функцій. Схема організаційної структури системи при взаємодії із користувачами (у виді діаграми варіантів використання) приведена на рисунку 2.1.

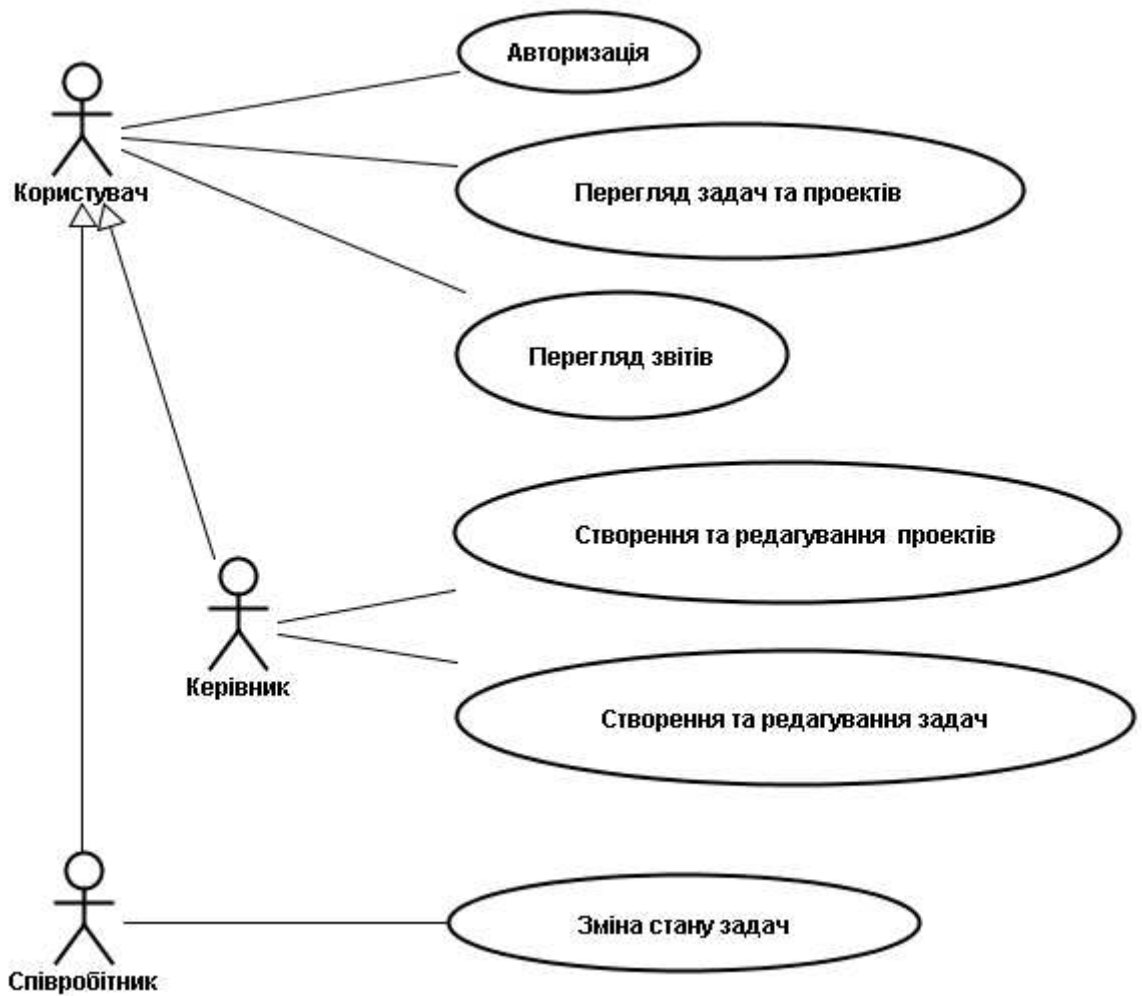


Рисунок 2.1 Діаграма варіантів використання

У ході своєї роботи, користувачі взаємодіють із системою на рівні визначених прав доступу та використовують наступні функції:

Керівник – для входу в систему повинен авторизуватись, потім має можливість створювати, редагувати або видаляти проекти та задачі, реєструвати в системі нових користувачів. Також має можливість перегляду задач, і звітів системи [5].

Співробітник - для початку роботи має авторизуватись, авторизувавшись може змінювати стан задач за які він відповідальний. Також має можливість перегляду задач, і звітів системи.

2.1.2 Побудова концептуальної моделі даних

Побудова концептуальної моделі являє собою процес моделювання смислового наповнення бази даних. Концептуальна модель складається з наступних трьох основних компонентів.

- Сутності - елементи реального світу, які можуть існувати незалежно.
- Атрибути - описують сутність.
- Зв'язки. Зв'язок представляє взаємодію між сутностями.

На основі аналізу системи будуюмо концептуальну модель даних, яка зображена на рисунку 2.2.

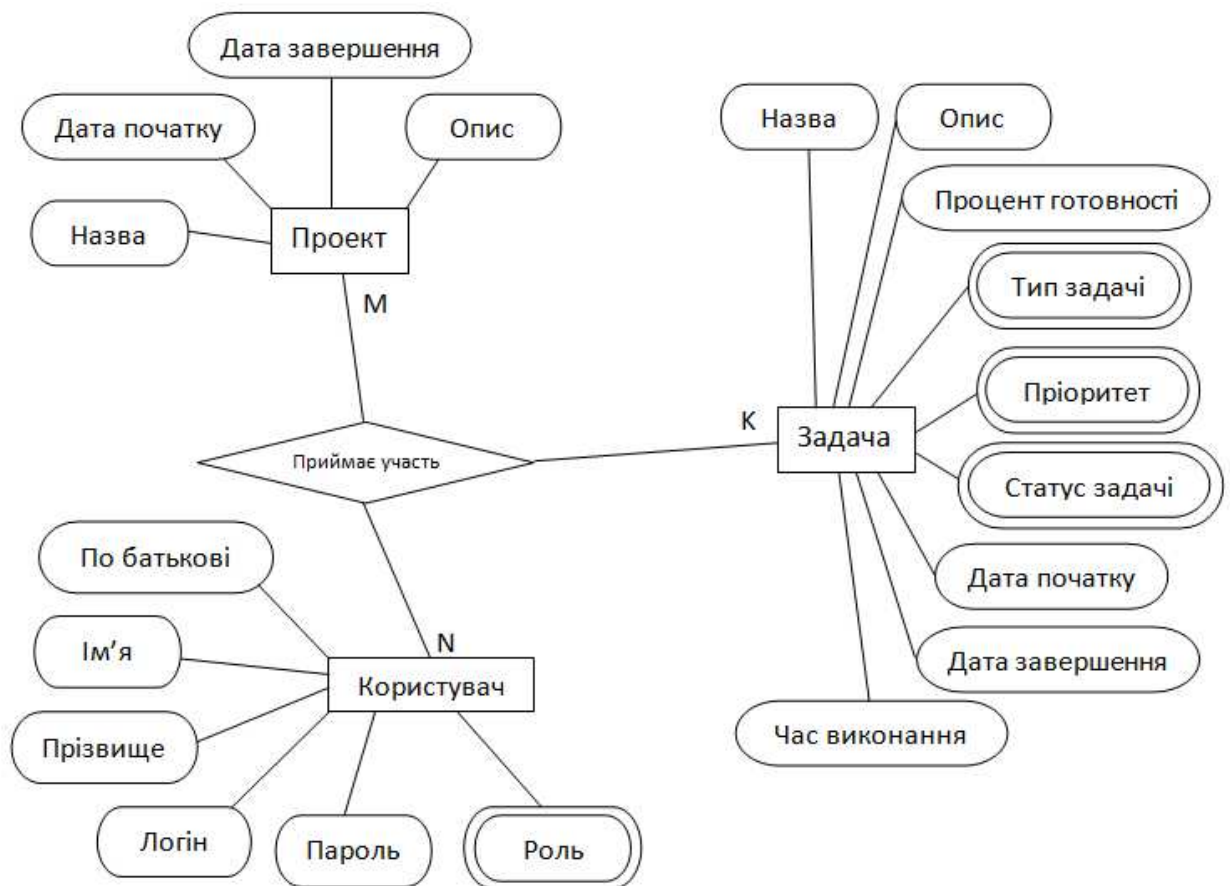


Рисунок 2.2 Концептуальна модель даних

2.1.3 Побудова логічної моделі даних

Логічна модель відображає логічні зв'язки між елементами даних. Вона формулюється в термінах бази даних, але не залежить від конкретної СУБД. Для кожного атрибута таблиці визначається тип даних, а саме: N – число, S – строчка, D – дата. Для кожного з ключів також прийняті скорочення: РК – основний ключ, FK – зовнішній ключ. Зовнішній ключ – один або декілька атрибутів, за допомогою яких підлегла таблиця пов'язується з батьківською [6].

На основі концептуальної моделі, розробленої в пункті 2.1.2 була розроблена логічна модель даних, яка зображена на рисунку 2.3.

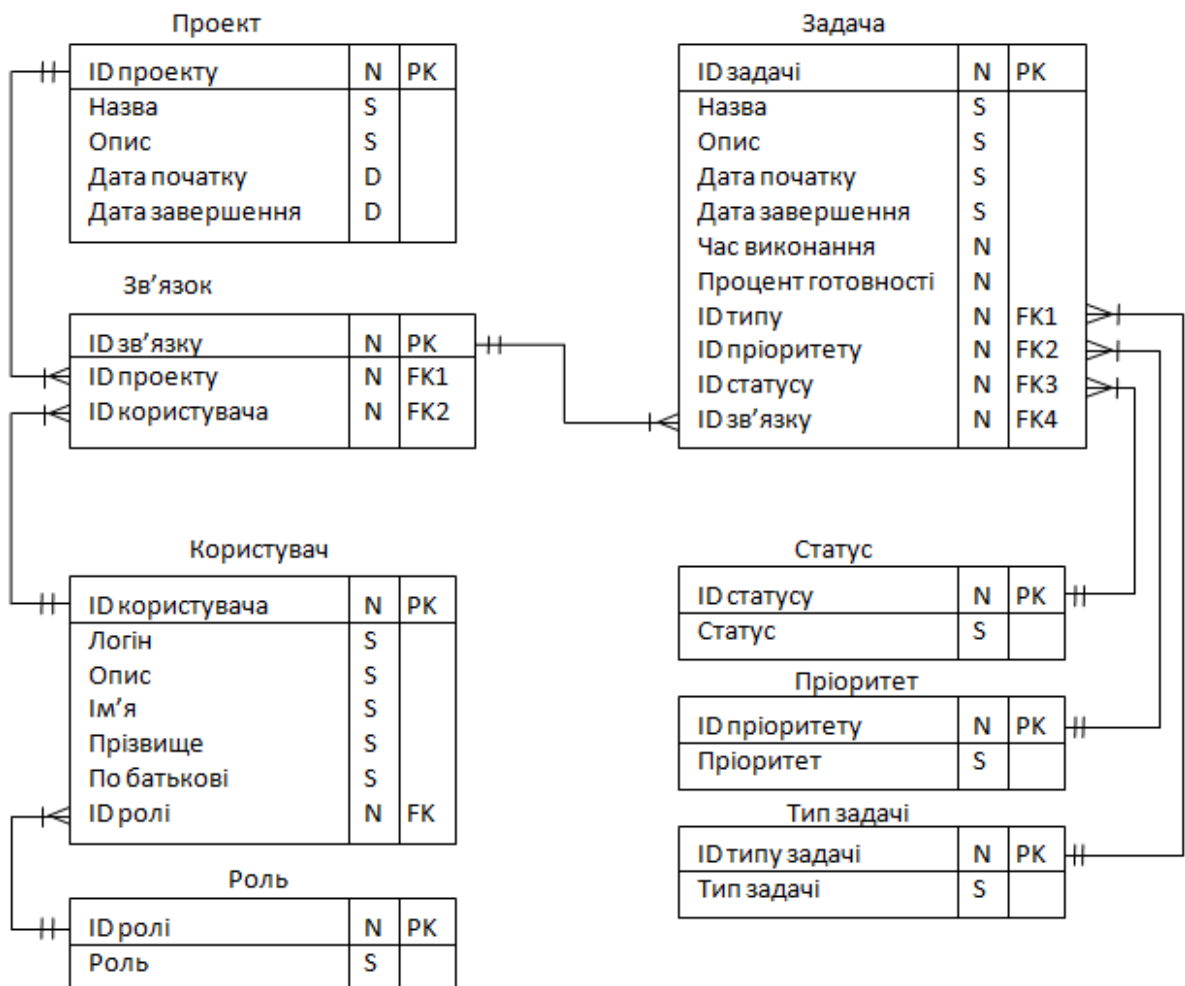


Рисунок 2.3 Логічна модель даних

2.1.4 Побудова фізичної моделі даних

На основі логічної моделі була розроблена фізична модель даних, яка зображена на рисунку 2.4

Ця модель відповідає практичній реалізації бази даних і визначає ті фізичні об'єкти, які необхідно реалізувати. При переході від логічної моделі до фізичної сутності перетворюються у таблиці, а атрибути — у поля (стовбці).

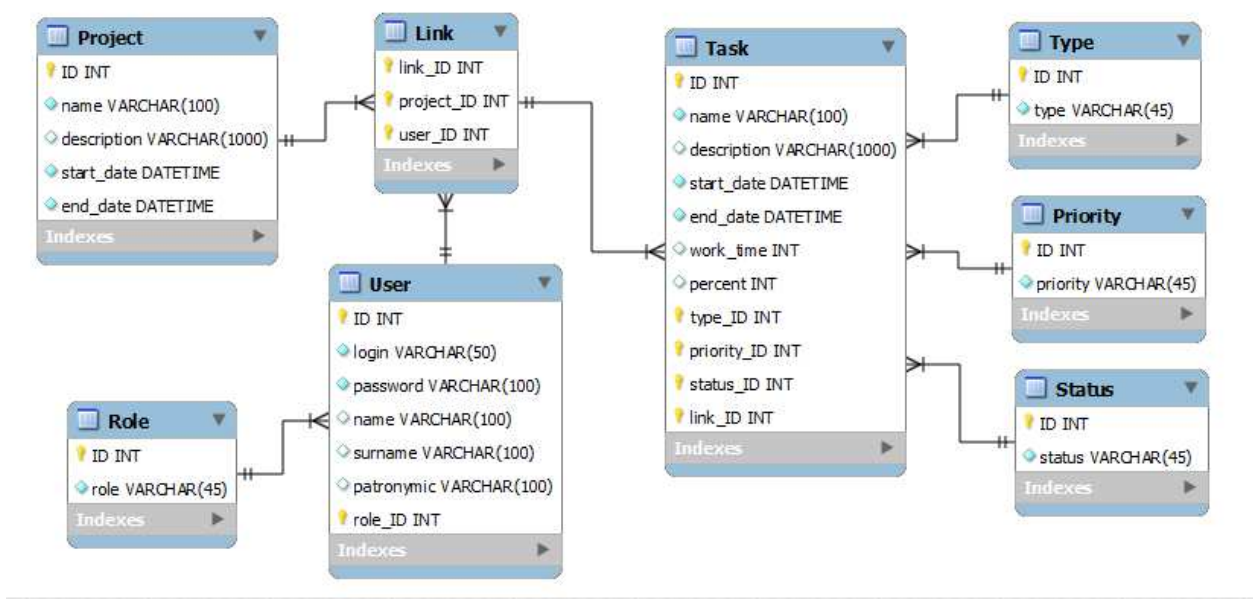


Рисунок 2.4 Фізична модель даних

Таблиця 2.1 – Дані таблиць фізичної моделі даних

Ім'я фізичної моделі	Тип поля	Розмір поля (байт)	Ключ (фізичної моделі)	Обов'язковість поля
Проект				
ID	INTEGER	4	PK	Так
name	VARCHAR	100	-	Так

description	VARCHAR	1000	-	Hi
start_date	DATETIME	8	-	Так
end_date	DATETIME	8	-	Так
Користувач				
ID	INTEGER	4	PK	Так
login	VARCHAR	50	-	Так
password	VARCHAR	100	-	Так
name	VARCHAR	100	-	Hi
surname	VARCHAR	100	-	Hi
patronymic	VARCHAR	100	-	Hi
role_ID	INTEGER	4	FK	Так
Роль				
ID	INTEGER	4	PK	Так
role	VARCHAR	45	-	Hi
Задача				
ID	INTEGER	4	PK	Так
name	VARCHAR	100	-	Так
description	VARCHAR	1000	-	Hi
start_date	DATETIME	8	-	Так
end_date	DATETIME	8	-	Так
work_time	INTEGER	4	-	Hi
percent	INTEGER	4	-	Hi
type_ID	INTEGER	4	FK1	Так
priority_ID	INTEGER	4	FK2	Так
status_ID	INTEGER	4	FK3	Так
link_ID	INTEGER	4	FK4	Так
Зв'язок				
link_ID	INTEGER	4	PK	Так

project_ID	INTEGER	4	FK1	Так
user_ID	INTEGER	4	FK2	Так
Тип задачі				
ID	INTEGER	4	PK	Так
type	VARCHAR	45	-	Так
Пріоритет				
ID	INTEGER	4	PK	Так
priority	VARCCHAR	45	-	Так
Статус				
ID	INTEGER	4	PK	Так
status	VARCHAR	45	-	Так

2.2 Технічне забезпечення

2.2.1 Визначення технічних засобів системи

Вибір технічних засобів, необхідних для впровадження автоматизованої системи буде значною мірою визначати загальний успіх чи невдачу проекту. На підставі прийнятих загальносистемних рішень були визначені склад і організація комплексу технічних засобів.

Вимоги до продуктивності: обчислювальний комплекс повинен забезпечувати достатню продуктивність для рішення передбачуваних задач. Крім того, потрібно забезпечити прийнятний час реакції системи при роботі декількох користувачів одночасно. Цим буде забезпечена їх комфортна робота. Проектована система не буде пред'являти підвищених вимог до обчислювальної потужності. Створювану систему можна віднести до класу облікових задач, у яких загальна продуктивність системи визначається швидкістю операцій введення-висновку при неповному залученні потужності центрального процесора. Розподіл програмних компонентів системи

дозволить не збільшити навантаження на існуючі клієнтські місця і не висувати вимог до їх модернізації [7].

Вимоги до надійності: розв'язувана задача висуває стандартні вимоги до надійності апаратних засобів без вимоги забезпечення повної відмово стійкості. Основною вимогою в даній групі буде вимога до збереження даних. Передбачаються наступні заходи для його виконання:

- включення сервера в електромережу через пристрій безперебійного живлення (UPS);
- використання пристроїв резервного копіювання.

2.2.2 Схема взаємодії модулів

На основі організаційної структури системи була створена схема взаємодії модулів, яка зображена на рисунку 2.5.



Рисунок 2.5 Схема взаємодії модулів

Модуль ідентифікації користувача призначений для авторизації користувача. Користувач може увійти в систему як керівник або співробітник, і далі може виконувати операції в системі, які відповідають

його правам доступу. Даний модуль звертається до бази даних, щоб співставити введені дані користувача та дані, які знаходяться в базі даних. Неавторизований користувач не може виконувати ніякі дії в системі.

Якщо користувач увійшов в систему як керівник, викликається модуль керівника, який в свою чергу може викликати модулі перегляду задач та проектів, модулі створення та редагування задач і проектів, а також модуль перегляду звітів [8].

Модуль перегляду задач та проектів надає змогу користувачу переглянути існуючі задачі і проекти та надає інструменти пошуку по деяким критеріям.

Модулі створення та редагування задач і проектів надають можливість керівнику створювати нові задачі і проекти або редагувати існуючі.

Модуль перегляду звітів надає можливість користувачу переглянути поточний стан проекту.

У випадку входу в систему співробітника, йому будуть доступні модулі зміни стану задач, перегляду задач та проектів, перегляду звітів.

Модуль зміни стану задачі дає змогу співробітнику у результаті роботи над задачею помічати поточний стан роботи: змінювати статус задачі, процент виконання та затрачений час.

2.2.3 Модель розгортання системи

Модель розгортання (синонім - діаграма розміщення) застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів по окремих вузлах системи.

На основі розробленої концепції була створена модель розгортання системи, яка зображена на рисунку 2.6.

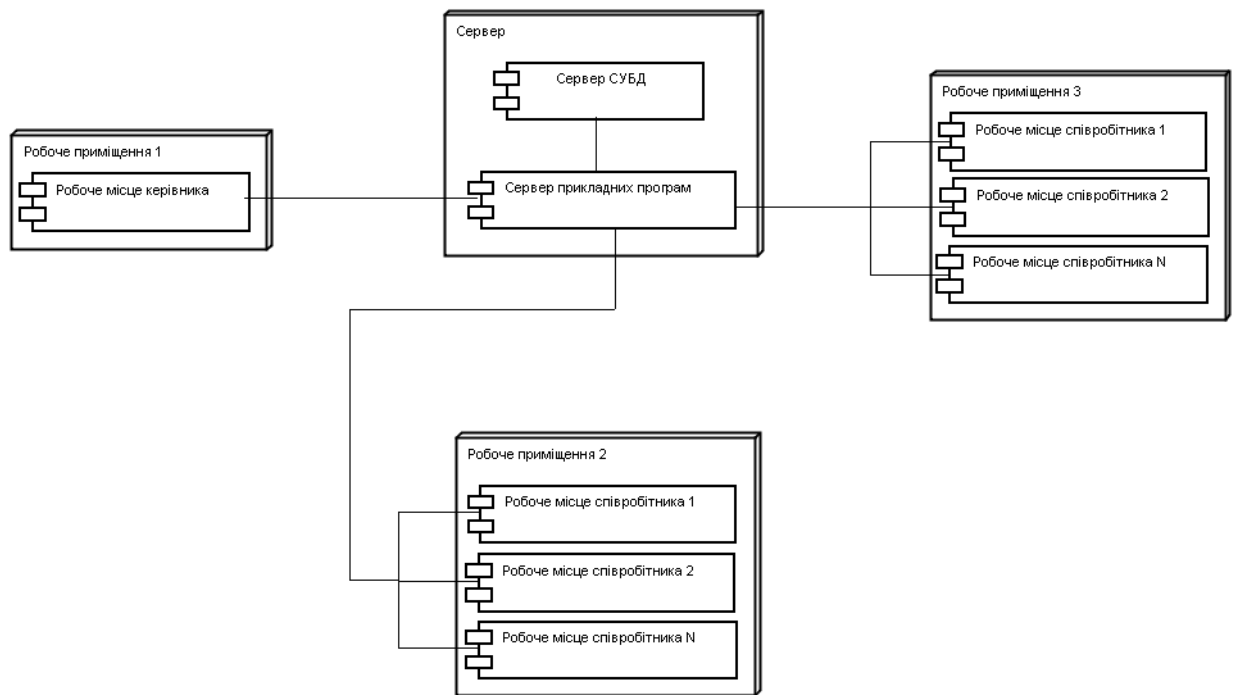


Рисунок 2.6 Модель розгортання системи

2.3 Програмне забезпечення

2.3.1 Мова програмування

Так як основною корпоративною мовою програмування в ПАО КБ «ПриватБанк» є мова Java, для реалізації проекту було обрано її.

Java — об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байт код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

JavaEE (Java Platform, Enterprise Edition) — набір специфікацій та відповідної документації для мови Java, яка описує архітектуру серверної платформи для задач середніх і великих підприємств.

Специфікації деталізовані настільки, щоб забезпечити переносимість програм з однієї реалізації платформи на іншу. Основна мета специфікацій — забезпечити масштабованість додатків і цілісність даних під час роботи системи.

2.3.2 Інструмент управління та складання проекту

Для вибору програмного інструменту управління та складання Java проекту було проаналізовано два інструменти: Apache Maven та Apache Ant.

Apache Ant – java-утиліта для автоматизації процесу збирання програмного продукту. Ant – платформо незалежний аналог UNIX-утиліти make, але з використанням мови Java, він вимагає платформи Java, і краще пристосований для Java-проектів.

Управління процесом складання відбувається за допомогою XML-сценарію, який також називають Build-файлом. У першу чергу цей файл містить визначення проекту, що складається з окремих цілей (Targets). Цілі порівняні з процедурами в мовах програмування і містять виклики команд-завдань (Tasks). Кожне завдання являє собою неподільну, атомарному команду, що виконує певну елементарну дію.

Ant дозволяє визначати власні типи завдань шляхом створення Java-класів, що реалізують певні інтерфейси.

Apache Maven – програмний інструмент для управління Java проектами (management) та складання (build) програм, створений Джейсоном ван Зилом (*Jason van Zyl*) у 2002 році. У функціонуванні подібний до Apache Ant, але має простішу build-модель конфігурації, яка базується на форматі XML.

Для опису програмного проекту який потрібно побудувати (*build*), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач - таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію, що забезпечує доступ до багатьох версій різних Java-проектів з відкритим кодом, від Apache та інших організацій та окремих розробників. Цей репозиторій та його реорганізований наступник, - Maven 2

репозиторій, - намагається бути де-факто механізмом для дистрибуції Java програм, але прийняття його в такій якості йде повільно.

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє як первісний засіб синхронізації виходу проектів на локальній системі.

Інструментом управління та складання Java проекту було обрано Apache Maven.

2.3.3 Інструмент створення web-інтерфейсу

Для вибору інструмента створення web-інтерфейсу проекту, було проаналізовано два фреймворка Java: JavaServer Faces, Vaadin.

JavaServer Faces (JSF) — це каркас програмування, технологія для веб-сторінок, що написані на Java. Він служить для того, щоб полегшувати розробку користувацьких інтерфейсів для Java EE сторінок. На відміну від більшості MVC фреймворків, які керуються запитами, підхід JSF ґрунтується на використанні компонентів. Стан компонентів користувацького інтерфейсу зберігається, коли користувач запитує нову сторінку й потім відновлюється, якщо запит повторюється. Для відображення даних звичайно використовується JSP, але JSF можна пристосувати й під інші технології, наприклад XUL.

Переваги:

- генерація серверної частини інтерфейсу користувача

- базується на компонентах (деякого HTML)

- наявна обробка подій (event) та станів (states)

- різноманітні view-технології — не тільки HTML та JavaScript

- розробка з урахуванням доступного інструментарію

- наявний стандарт і він включений до Java EE

Недоліки:

потрібно багато часу для вивчення та освоєння технології
потребуються потужні обчислювальні можливості серверу

Vaadin — відкритий набір бібліотек для створення так званих RIA веб-програм. На відміну від бібліотек на JavaScript і специфічних плагінів для браузерів, Vaadin пропонує сервер-орієнтовану архітектуру, яка базується на Java Enterprise Edition. Використання JEE дозволяє виконувати основну частину логіки програми на стороні сервера, тоді як технологія AJAX, використовувана на стороні браузера, дозволяє інтерактивно взаємодіяти з користувачем, не відстаючи від аналогічних десктоп-програм. Для відображення елементів інтерфейсу користувача та взаємодії з сервером на стороні клієнта Vaadin використовує Google Web Toolkit.

Можливості:

Використання Java як єдиної мови програмування при створенні веб-програм і веб-контенту — одна з найбільш значущих функцій в Vaadin.

Організація моделі даних і віджетів дозволяє відображати в браузері великі обсяги даних без значного завантаження пам'яті і без додаткових дій з боку розробника.

Можливість розширення стандартного набору віджетів Vaadin за рахунок інших віджетів, написаних для GWT, а також кастомізації його за допомогою CSS.

Інструментом створення web-інтерфейсу проекту було обрано фреймворк Vaadin.

2.3.4 Середовище програмування

Під час вибору середовища програмування було проаналізовано три найпопулярніших середовища програмування: NetBeans IDE, IntelliJ IDEA та Eclipse.

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, Python, Groovy.

Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за умовчанням підтримує розробку для платформ J2SE і J2EE.

Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle, проте розробка NetBeans ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

За якістю і можливостям останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, авто доповнення мовних конструкцій на льоту, шаблони коду та інше.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

IntelliJ IDEA — комерційна інтегрована среда розробки програмного забезпечення на багатьох мовах програмування, зокрема Java, JavaScript, Python та ін., розроблена компанією JetBrains.

Перша версія IntelliJ IDEA з'явилася в січні 2001 року і швидко набула популярності, як перша Java IDE з широким набором інтегрованих інструментів для рефакторінга, які дозволяли програмістам швидко реорганізувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи їм сконцентруватися на розробці функціональності, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

IntelliJ IDEA доступна в двох версіях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0. У ній реалізована повна підтримка Java SE, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. У версії Ultimate Edition реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов і фреймворків.

Eclipse — вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation. Написаний в основному на Java, і може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby(включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

Початок коду йде від IBM VisualAge, він був розрахований на розробників Java, складаючи Java Development Tools (JDT). Але користувачі могли розширяти можливості, встановлюючи написані для програмного каркасу Eclipse плагіни, такі як інструменти розробки під інші мови програмування, і могли писати і вносити свої власні плагіни і модулі.

Випущена на умовах Eclipse Public License, Eclipse є вільним програмним забезпеченням. Він став одним з перших IDE під GNU Classpath і без проблем робить під IcedTea.

Середовищем програмування було обрано IntelliJ IDEA.

2.3.5 Сервер бази даних

Для вибору системи управління базами даних було проаналізовано дві реляційні системи: MySQL, PostgreSQL.

MySQL — вільна реляційна система управління базами даних. Розробку та підтримку MySQL здійснює корпорація Oracle, що отримала права на торговельну марку разом з поглинанням Sun Microsystems.

MySQL вважається гарним рішенням для малих і середніх застосувань. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більше того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. Для некомерційного використання MySQL є безкоштовним.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють з БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

PostgreSQL — об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft

SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно до інших проектів з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

Сервер PostgreSQL написаний на мові С. Зазвичай розповсюджується у вигляді набору текстових файлів із програмним кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

Системою управління базою даних було обрано – MySQL.

2.3.6 Сервер прикладних програм

Для вибору серверу прикладних програм було проаналізовано три найпопулярніших сервера прикладних програм: Tomcat, JBoss.

Tomcat — контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software Foundation. Реалізує специфікацію сервлетів і специфікацію JavaServer Pages (JSP). Написаний на мові Java.

Tomcat дозволяє запускати веб-програми, містить ряд програм для самоконфігурування.

Tomcat використовується в якості самостійного веб-сервера, в якості сервера контенту в поєднанні з веб-сервером Apache HTTP Server, а також в якості контейнера сервлетів в серверах JBoss і GlassFish.

Розробка та підтримка Tomcat здійснюється фондом Apache Software Foundation і добровольцями. Користувачі мають вільний доступ до вихідного коду і бінарних файлів Tomcat згідно ліцензії Apache License 2.0.

JBoss Application Server або *JBoss* — сервер прикладних програм на платформі Java EE з відкритим сирцевим кодом, розроблений однойменною

компанією JBoss. Як і багато програм з відкритим кодом, розроблених комерційними організаціями, JBoss можна вільно завантажувати та використовувати, проте підтримка та консультації здійснюються за окрему платню. Достатньо якісна реалізація принципів J2EE робить JBoss конкурентом для аналогічних власницьких програмних рішень, таких як WebSphere або WebLogic.

JBoss використовує Apache Tomcat в якості контейнеру сервлетів.

Так як фреймворк Vaadin базується на Java Enterprise Edition та сервлетах, для його роботи буде достатньо контейнеру сервлетів Apache Tomcat.

Висновки до розділу 2

У другому розділі було здійснено системне проєктування інформаційної системи управління ІТ-проєктами з урахуванням принципів DevOps, орієнтованої на потреби та специфіку діяльності АТ КБ «ПриватБанк». На основі результатів аналізу, проведеного у попередньому розділі, сформовано архітектурну модель системи, визначено її функціональну структуру, інформаційні потоки та ключові компоненти, які забезпечують інтеграцію процесів розробки, тестування, реліз-менеджменту та експлуатації.

У процесі проєктування були визначені основні модулі системи, зокрема: модуль планування та управління завданнями, модуль моніторингу виконання робіт, модуль автоматизації DevOps-процесів (CI/CD, контроль версій, тестування), модуль аналітики та звітності, а також підсистема управління доступом та безпекою. Особливу увагу приділено побудові логічної та фізичної архітектури, що забезпечує масштабованість, надійність і можливість інтеграції з корпоративною інфраструктурою банку.

Створено діаграми варіантів використання, взаємодії, класів та компонентів, що відображають ключові користувацькі сценарії й

підтверджують узгодженість функціональної моделі. Розроблена модель даних забезпечує цілісність, структурованість і узгодженість інформаційних потоків, необхідних для ефективного управління проєктами у DevOps-середовищі.

Проєктні рішення дозволили сформувати комплексний концептуальний та технічний фундамент для реалізації системи, здатної автоматизувати критично важливі процеси, підвищити прозорість взаємодії між командами та забезпечити безперервність життєвого циклу розробки програмного забезпечення.

Таким чином, у розділі було створено цілісну проєктну основу майбутньої інформаційної системи, яка відповідає стратегічним вимогам АТ КБ «ПриватБанк» та закладає підґрунтя для її подальшої реалізації й впровадження.

РОЗДІЛ 3. РОЗРОБКА ІС УПРАВЛІННЯ ІТ-ПРОЕКТАМИ З DEVOPS НА ПРИКЛАДІ АТ КБ «ПРИВАТБАНК»

На основі аналізу системи та технічного завдання, яке приведено у додатку А, було створено необхідне програмне забезпечення. На рисунку 3.2 зображено діаграму проекту Apache Maven зі зв'язками між модулями та сторонніми бібліотеками.

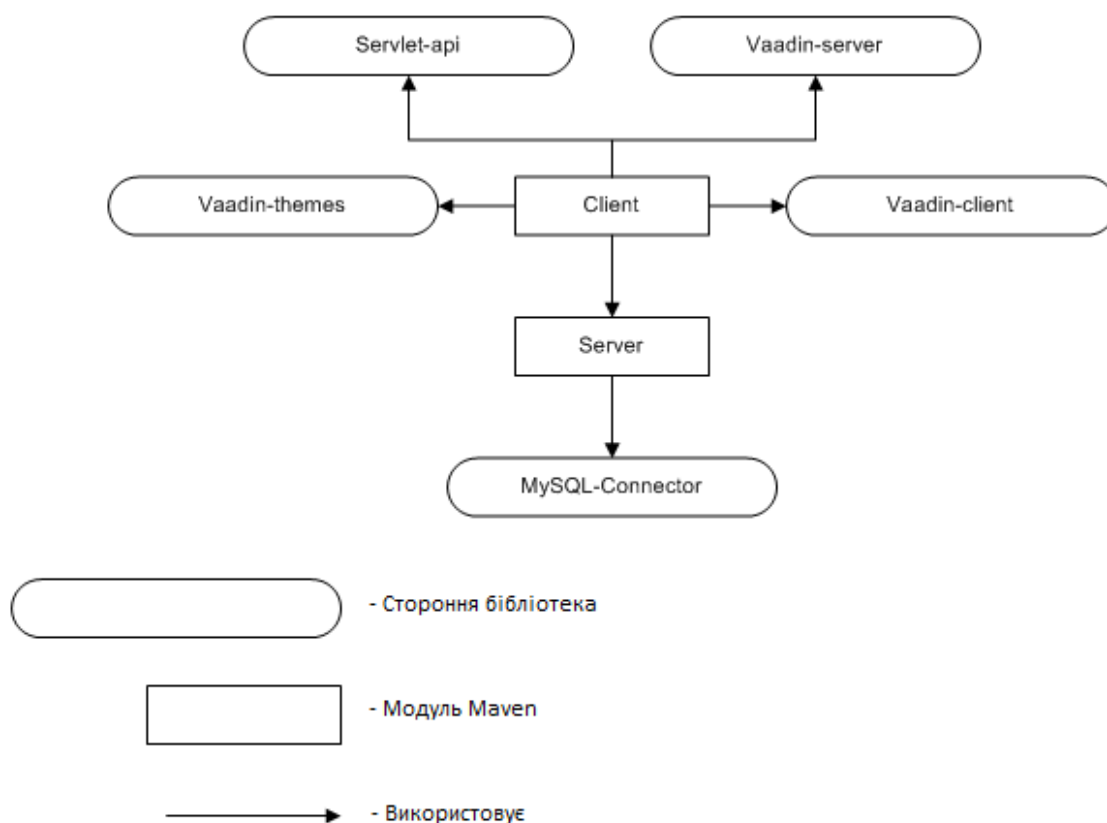


Рисунок 3.1 – Діаграма зв'язків Maven проекту

Так як проект має не малу кількість вихідних кодів, на діаграмах класів буде зображено вибіркові функціональності системи. Діаграми класів зображені на рисунках 3.2 - 3.4.

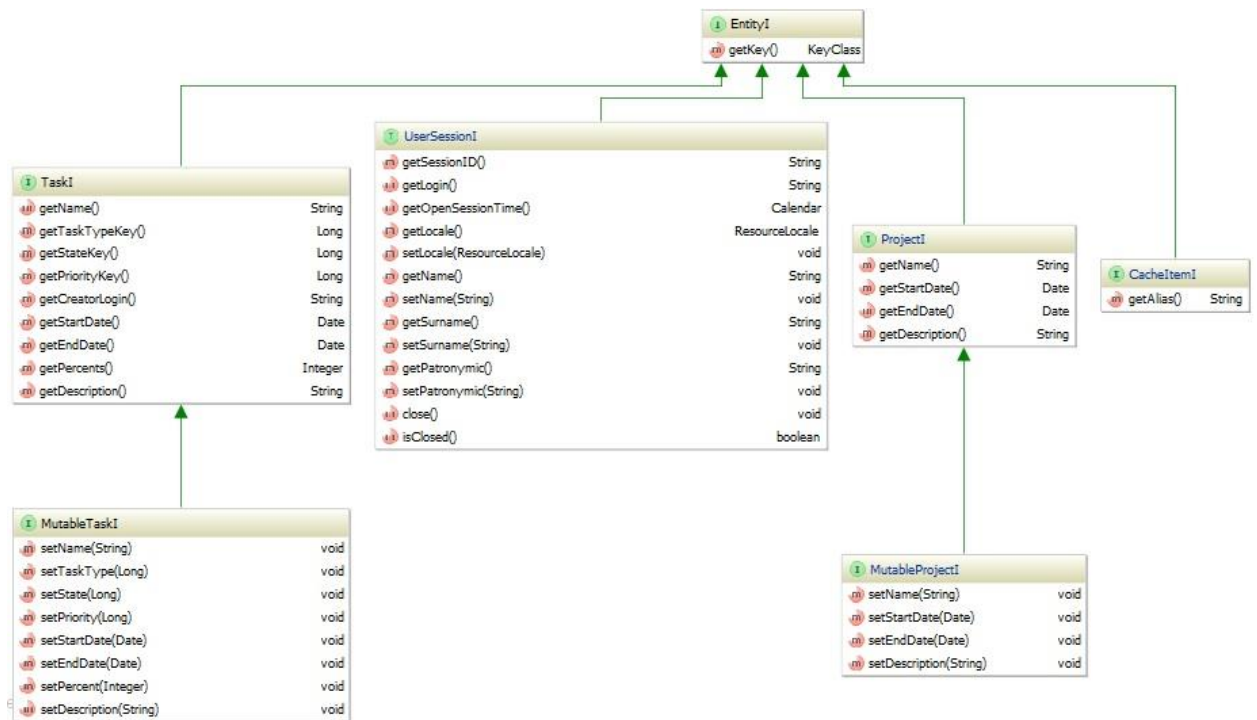


Рисунок 3.2 – Діаграма класів сутностей

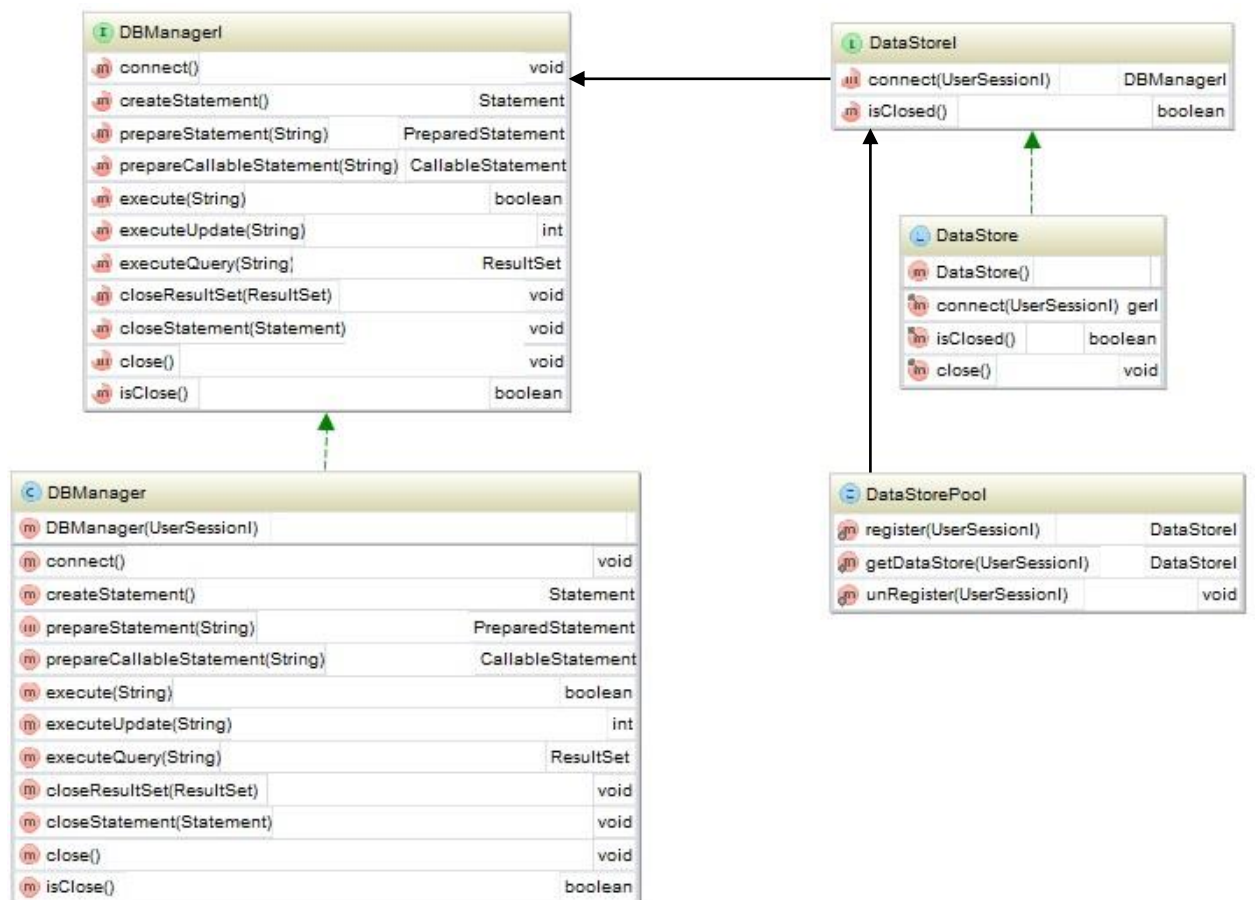


Рисунок 3.3 – Діаграма класів менеджера підключення до БД

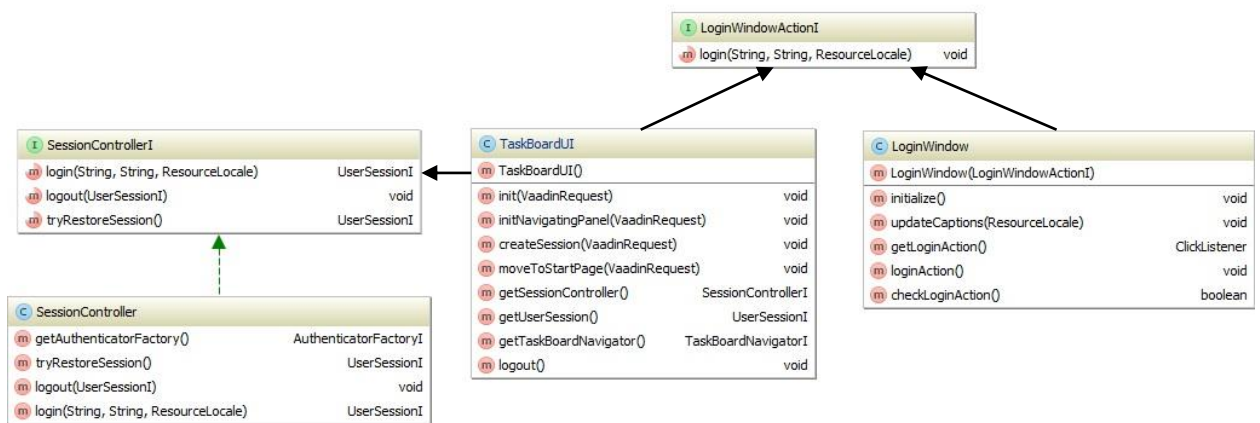


Рисунок 3.4 – Діаграма класів авторизації користувача

Висновки до розділу 3

У третьому розділі було виконано безпосередню розробку інформаційної системи управління ІТ-проєктами з використанням DevOps-підходів, що базуються на вимогах та архітектурних рішеннях, сформованих у попередніх розділах. Реалізовані функціональні модулі підтвердили працездатність проєктного рішення та довели можливість практичного застосування системи в умовах АТ КБ «ПриватБанк».

У ході розробки було створено та інтегровано ключові компоненти: модуль планування та управління завданнями, підсистема моніторингу стану проєктів, засоби автоматизації DevOps-процесів (CI/CD-конвеєри, автоматизоване тестування, контроль версій), а також модуль збору та аналізу метрик. Особливу увагу приділено реалізації механізмів безпеки, керуванню доступами та забезпеченню надійності обробки даних у відповідності до вимог банківської інфраструктури.

Було успішно побудовано інтеграційні зв'язки між системою та корпоративними інструментами, що використовуються в банку, що дозволило забезпечити безшовну взаємодію команд розробки й експлуатації. Результати тестування продемонстрували стабільну роботу системи,

коректність виконання сценаріїв, а також високу швидкість обробки операцій.

Розроблена інформаційна система підтвердила здатність оптимізувати життєвий цикл ІТ-проектів, підвищити рівень автоматизації, прозорості та контрольованості процесів, що є ключовими аспектами DevOps-культури. Отримані результати демонструють можливість впровадження системи в реальне середовище ПриватБанку для підвищення ефективності роботи ІТ-підрозділів.

Таким чином, у розділі було реалізовано повнофункціональну програмну систему, що відповідає визначеним вимогам, підтверджує правильність обраної концепції та створює передумови для масштабування й подальшого удосконалення ІС управління ІТ-проектами в DevOps-середовищі.

РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

4.1. Мета та необхідність техніко-економічного обґрунтування

Техніко-економічне обґрунтування (ТЕО) спрямоване на визначення доцільності створення та впровадження інформаційної системи (ІС) управління ІТ-проектами з DevOps, оцінку її економічної ефективності, витрат, термінів окупності та очікуваних вигод для АТ КБ «ПриватБанк». Проведення ТЕО дає змогу оцінити, наскільки впровадження ІС забезпечить підвищення продуктивності ІТ-підрозділів, скорочення тривалості розробки й випуску релізів, оптимізацію ресурсів та зменшення ризиків, пов'язаних із людським фактором.

4.2. Визначення технічних вимог до системи

На основі функціональних і нефункціональних вимог, сформованих у процесі аналізу та проєктування, визначено необхідні технічні ресурси для реалізації ІС:

Обчислювальні ресурси: серверні потужності для роботи CI/CD-конвеєрів, системи контролю версій, бази даних та аналітичних модулів.

Інструментальні засоби: Git, Docker, Kubernetes (або аналогічні системи оркестрації), Jenkins/GitLab CI, платформа моніторингу.

База даних: високонадійна СУБД для зберігання проєктних даних, журналів дій, конфігурацій.

Мережеві параметри: захищені канали зв'язку, VPN, внутрішні корпоративні мережі банку.

Засоби безпеки: механізми шифрування, ролева модель доступу, аудит активностей.

Вимоги обґрунтовані масштабом організації та потребою у безперервності роботи системи з високим рівнем надійності.

4.3. Оцінка витрат на розробку

До складу витрат на створення та впровадження ІС входять:

Витрати на розробку ПЗ зарплатний фонд команди (аналітик, розробники, тестувальник, DevOps-інженер). Витрати на проектування та моделювання системи. Тестування та виправлення помилок.

Витрати на програмні інструменти й інфраструктуру використання CI/CD-інструментів (більшість — open-source). Витрати на серверні ресурси та зберігання даних. Налаштування середовища розгортання.

Витрати на впровадження навчання персоналу, інтеграція з корпоративними системами банку, приймально-здавальні тестування.

Експлуатаційні витрати супровід і технічна підтримка, оновлення та модернізація системи.

Сукупні витрати визначаються залежно від обраного масштабу впровадження, але завдяки використанню DevOps-орієнтованих інструментів значна частина ПЗ не потребує ліцензійних платежів.

4.4. Економічні вигоди від впровадження системи

Розробка та використання ІС управління ІТ-проектами з DevOps забезпечує низку економічних переваг.

Скорочення часу випуску релізів на 20–40% за рахунок автоматизації тестування, інтеграції та деплоюменту.

Зменшення витрат на ручні операції, оскільки автоматизуються рутинні процеси DevOps.

Скорочення кількості помилок унаслідок стандартизації процесів і кращого контролю якості.

Підвищення продуктивності команд через прозорість роботи, централізацію даних і швидку комунікацію.

Оптимізація використання ресурсів, що знижує непрямі витрати.

Можливість масштабування системи без суттєвого збільшення витрат.

Таким чином, система дозволяє не лише зменшити поточні витрати, але й підвищити довгострокову ефективність ІТ-процесів банку.

4.5. Оцінка ефективності проекту

Під час оцінки ефективності розглядаються такі показники:

Прямі економічні вигоди (зменшення часу простоїв, оперативні релізи, менші витрати на підтримку).

Непрямі вигоди (підвищення якості сервісів банку, задоволеність клієнтів, зменшення ризику інцидентів).

Термін окупності проекту: очікується 1–2 роки залежно від інтенсивності використання.

ROI (рентабельність інвестицій): за рахунок автоматизації та прискорення процесів ROI є позитивним і зростає з кожним роком експлуатації.

Загалом реалізація ІС є економічно обґрунтованою та приносить фінансову вигоду вже у перші роки використання.

Висновки до розділу 4

Проведений аналіз підтверджує, що розробка та впровадження ІС управління ІТ-проектами з DevOps на прикладі АТ КБ «ПриватБанк» є технічно здійсненою та економічно вигідною. Система забезпечує комплексну автоматизацію ІТ-процесів. Підвищення якості програмних продуктів. Скорочення часу розробки та прискорення релізів. Зниження експлуатаційних витрат. Підвищення рівня безпеки та контрольованості.

Таким чином, впровадження розробленої ІС забезпечує значний позитивний техніко-економічний ефект і є доцільним для використання в інфраструктурі АТ КБ «ПриватБанк».

РОЗДІЛ 5. ОХОРОНА ПРАЦІ. ВИМОГИ ДО ПАРАМЕТРІВ МІКРОКЛІМАТУ ТА ВЕНТИЛЯЦІЇ РОБОЧИХ ПРИМІЩЕНЬ

5.1 Вимоги до параметрів мікроклімату робочих приміщень

Для забезпечення високої продуктивності праці, збереження здоров'я, доброго самопочуття, а також продовження активного довголіття необхідно контролювати та оптимізувати сукупність параметрів виробничого середовища.

Так, метеорологічні умови виробничого середовища, або мікроклімат виробничих приміщень - це клімат всередині приміщення, який визначається сукупністю значень температури (t , °C); відносної вологості (j , %); швидкості руху повітря (V , м/с); температури поверхонь обладнання.

5.1.1 Розкриття поняття виробниче приміщення та мікроклімат

На підприємствах на самопочуття, стан здоров'я людини впливає мікроклімат виробничих приміщень, який визначається дією на організм людини температури, вологості, рухомості повітря і теплового випромінювання. Виробничий мікроклімат, як правило, відрізняється значною мінливістю, нерівномірністю по горизонталі та вертикалі, різноманітністю сполучень температури, вологості, рухомості повітря, інтенсивності випромінювання залежно від особливостей технології виробництва, кліматичних особливостей місцевості, конструкцій споруд, організації повітрообміну із зовнішнім середовищем.

Отже, виробниче приміщення - це замкнутий простір в спеціально призначених будинках та спорудах, в яких постійно (по змінах) або періодично (протягом частини робочого дня) здійснюється трудова діяльність людей.

Мікроклімат виробничих приміщень - це умови внутрішнього середовища цих приміщень, що впливають на тепловий обмін працюючих з

оточенням шляхом конвекції, кондукції, теплового випромінювання та випаровування вологи.

5.1.2 Вимоги до параметрів мікроклімату виробничих приміщень

Мікрокліматичні умови виробничих приміщень характеризуються такими показниками:

- температура повітря,
- відносна вологість повітря,
- швидкість руху повітря,
- інтенсивність теплового (інфрачервоного) опромінення,
- температура поверхні.

За ступенем впливу на тепловий стан людини мікрокліматичної умови поділяють на оптимальні та допустимі. Їх ми розглянемо у наступному пункті.

Для робочої зони виробничих приміщень встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням важкості виконуваної роботи та періоду року. При одночасному виконанні в робочій зоні робіт різної категорії важкості рівні показників мікроклімату повинні встановлюватись з урахуванням найбільш чисельної групи працівників.

5.1.3 Оптимальні і допустимі умови мікроклімату виробничого приміщення.

Оптимальні мікрокліматичні умови - поєднання параметрів мікроклімату, які при тривалому та систематичному впливі на людину забезпечують зберігання нормального теплового стану організму без активізації механізмів терморегуляції. Вони забезпечують відчуття теплового комфорту та створюють передумови для високого рівня працездатності.

Оптимальні умови мікроклімату встановлюються для постійних робочих місць. Показники температури повітря в робочій зоні по висоті та по горизонталі, а також протягом робочої зміни не повинні виходити за межі

нормованих величин оптимальної температури для тієї чи іншої категорії робіт.

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання (екрани і т. ін.), зовнішніх поверхонь технологічного устаткування, огорожуючи конструкцій не повинна виходити більш ніж на 2 град.С за межі оптимальних величин температури повітря для даної категорії робіт.

При виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням в кабінетах, пультах і постах керування технологічними процесами, в залах обчислювальної техніки та інших приміщеннях повинні дотримуватися оптимальні умови мікроклімату (температура повітря 22-24 град.С, відносна вологість 60-40%, швидкість руху повітря не більш 0,1 м/сек.).

Допустимі мікрокліматичні умови - поєднання параметрів мікроклімату, які при тривалому та систематичному впливі на людину можуть викликати зміни теплового стану організму, що швидко минають і нормалізуються та супроводжуються напруженням механізмів терморегуляції в межах фізіологічної адаптації. При цьому не виникає ушкоджень або порушень стану здоров'я, але можуть спостерігатися дискомфортні тепловідчуття, погіршення самопочуття та зниження працездатності.

Допустимі величини мікрокліматичних умов встановлюються у випадках, коли на робочих місцях не можна забезпечити оптимальні величини мікроклімату за технологічними вимогами виробництва, технічною недосяжністю та економічно обґрунтованою недоцільністю.

Перепад температури повітря по висоті робочої зони при забезпеченні допустимих умов мікроклімату не повинен бути більше 3 град.С для всіх категорій робіт, а по горизонталі робочої зони та протягом робочої зміни - виходити за межі допустимих температур для даної категорії роботи.

Температура внутрішніх поверхонь приміщень (стіни, підлога, стеля), а також температура зовнішніх поверхонь технологічного устаткування або його захисних обладнань (екранів і т. ін.) не повинна виходити за межі допустимих величин температури повітря для тієї чи іншої категорії робіт.

Інтенсивність теплового опромінення працюючих від нагрітих поверхонь технологічного устаткування, освітлювальних приладів, інсоляція від зашкленних огорожень не повинна перевищувати 35,0 Вт/м² - при опроміненні 50% та більше поверхні тіла, 70 Вт/м² - при величині опромінюваної поверхні від 25 до 50%, та 100 Вт/м² - при опроміненні не більше 25% поверхні тіла працюючого.

При наявності джерел з інтенсивністю 35,0 Вт/м² і більше температура повітря на постійних робочих місцях не повинна перевищувати верхніх меж оптимальних значень для теплого періоду року, на непостійних - верхніх меж допустимих значень для постійних робочих місць.

При наявності відкритих джерел випромінювання (нагрітий метал, скло, відкрите полум'я) допускається інтенсивність опромінення до 140,0 Вт/м². Величина опромінюваної площі не повинна перевищувати 25 % поверхні тіла працюючого при обов'язковому використанні індивідуальних засобів захисту (спецодяг, окуляри, щитки).

У виробничих приміщеннях, які розташовані в районах з середньою максимальною температурою найбільш жаркого місяця вище 25 град. С згідно з БНіП "Будівельна кліматологія" допускаються відхилення від величин показників мікроклімату для даної категорії робіт, але не більше ніж на 3 град.С. При цьому швидкість руху повітря повинна бути збільшена на 1,1 м/сек., а відносна вологість повітря знижена на 5 % при підвищенні температури на кожний градус вище верхньої межі допустимих температур повітря.

У виробничих приміщеннях, в яких не можна встановити допустимі величини мікроклімату через технологічні вимоги до виробничого процесу,

технічну недосяжність або економічно обґрунтовану недоцільність передбачаються заходи щодо захисту від можливого перегрівання та охолодження.

5.2 Вимоги до вентиляції робочих приміщень

Вентиляція являє собою сукупність заходів та засобів призначених для забезпечення на постійних робочих місцях та зонах обслуговування виробничих приміщень метеорологічних умов та чистоти повітряного середовища, що відповідають гігієнічним та технічним вимогам. Основне завдання вентиляції — вилучити із приміщення забруднене або нагріте повітря та подати свіже.

Вентиляція класифікується за такими ознаками:

- за способом переміщення повітря — природна, штучна (механічна) та суміщена (природна та штучна одночасно);
- за напрямком потоку повітря — припливна, витяжна, припливно-витяжна;
- за місцем дії — загально обмінна, місцева, комбінована.

5.2.1 Природна вентиляція

Природна вентиляція відбувається в результаті теплового та вітрового напору. Тепловий напір обумовлений різницею температур, а значить і густини внутрішнього і зовнішнього повітря. Вітровий напір обумовлений тим, що при обдуванні вітром будівлі, з її навітряної сторони утворюється підвищений тиск, а підвітряної — розрідження (рис. 4.1).

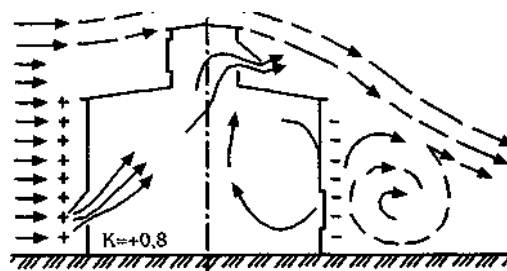


Рис. 4.1. Розподіл тиску в будівлі при дії вітру

Природна вентиляція може бути неорганізованою і організованою. При неорганізованій вентиляції невідомі об'єми повітря, що надходять та вилучаються із приміщення, а сам повітрообмін залежить від випадкових чинників (напрямку та сили вітру, температури зовнішнього та внутрішнього повітря). Неорганізована природна вентиляція включає інфільтрацію — просочування повітря через нещільності у вікнах, дверях, перекриттях та провітрювання, що здійснюється при відкриванні вікон та квартир.

Організована природна вентиляція називається аерацією. Для аерації в стінах будівлі роблять отвори для надходження зовнішнього повітря, а на даху чи у верхній частині будівлі встановлюють спеціальні пристрої (ліхтарі) для видалення відпрацьованого повітря. Для регулювання надходження та видалення повітря передбачено перекривання на необхідну величину аераційних отворів та ліхтарів. Це особливо важливо в холодну пору року.

5.2.2 Штучна вентиляція

Штучна (механічна) вентиляція, на відміну від природної, дає можливість очищувати повітря перед його викидом в атмосферу, вловлювати шкідливі речовини безпосередньо біля місць їх утворення, обробляти припливне повітря (очищувати, підігрівати, зволожувати), більш цілеспрямовано подавати повітря в робочу зону. Окрім того, механічна вентиляція дає можливість організувати повітрозбір в найбільш чистій зоні території підприємства і навіть за її межами.

Загально обмінна штучна вентиляція

Загальнообмінна вентиляція забезпечує створення необхідного мікроклімату та чистоти повітряного середовища у всьому об'ємі робочої зони приміщення. Вона застосовується для видалення надлишкового тепла при відсутності токсичних виділень, а також у випадках, коли характер технологічного процесу та особливості виробничого устаткування виключають можливість використання місцевої витяжної вентиляції.

Розрізняють чотири основні схеми організації повітрообміну при загальнообмінній вентиляції: зверху вниз, зверху вверх, знизу вверх, знизу вниз (рис 4.2).

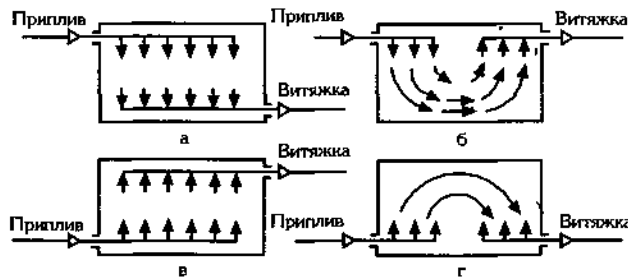


Рис.4.2. Схема організації повітрообміну при загальнообмінній вентиляції.

Схеми зверху вниз (рис. 4.2а) та зверху вверх (рис. 4.2.б) доцільно застосовувати у випадку, коли припливне повітря в холодний період року має температуру нижчу температури приміщення. Припливне повітря перш ніж досягти робочої зони нагрівається за рахунок повітря приміщення. Інші дві схеми (рис 4.2.в та 4.2.г) рекомендується використовувати тоді, коли припливне повітря в холодний період року підігрівається і його температура вища температури внутрішнього повітря.

Якщо у виробничих приміщеннях виділяються гази та пари з густиною, що перевищує густину повітря (наприклад, пари кислот, бензину, гасу), то загальнообмінна вентиляція повинна забезпечити видалення 60% повітря з нижньої зони приміщення та 40% — з верхньої. Якщо густина газів менша за густину повітря, то видалення забрудненого повітря здійснюється у верхній зоні.

Припливна вентиляція

Схема припливної механічної вентиляція (рис 4.3.) включає: повітрозабірний пристрій 1; фільтр для очищенні повітря 2; повітронагрівач (калорифер) 3; вентилятор 5; мереж, повітроводів 4 та припливні патрубки з насадками 6. Якщо нема необхідності підігрівати припливне повітря, то його пропускають безпосередньо у виробничі приміщення через обвідний канал 7.

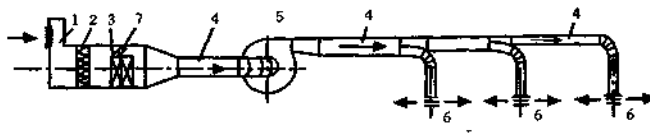


Рис. 4.3 Схема припливної вентиляції

Повітрозабірні пристрої необхідно розташовувати в місцях, *і* повітря не забруднене пилом та газами. Вони повинні знаходитись і нижче 2 м від рівня землі, а від викидних каналів витяжної вентиляції по вертикалі — нижче 6 м і по горизонталі — не ближче 25 м.

Припливне повітря подається в приміщення, як правило, розсіяним потоком для чого використовуються спеціальні насадки.

Витяжна та припливно-витяжна вентиляція

Витяжна вентиляція (рис. 4.4.) складається із очисного пристрою 1, вентилятор 2, центрального 3 та відсмоктуючих повітроводів 4.

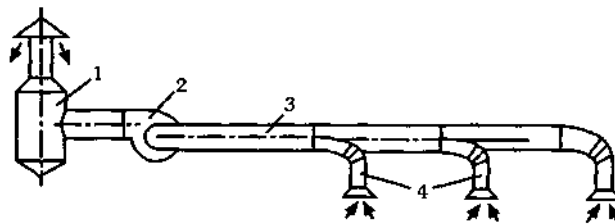


Рис.4.4. Схема витяжної вентиляції

Повітря після очищення необхідно викидати на висоті не менше ніж 1 м над гребенем даху. Забороняється робити викидні отвори безпосередньо у вікнах.

В умовах промислового виробництва найбільш розповсюджена припливно-витяжна система вентиляції із загальним припливом в робочу зону та місцевою витяжкою шкідливих речовин безпосередньо з місць їх утворення.

У виробничих приміщеннях, де виділяється значна кількість шкідливих газів, парів, пилу витяжка повинна бути на 10% більшою ніж приплив, щоб

шкідливі речовини не витіснялись у суміжні приміщення з меншою шкідливістю.

В системі припливно-витяжної вентиляції можливе використання не лише зовнішнього повітря, але й повітря самих приміщень після його очищення.

Таке повторне використання повітря приміщень називається *рециркуляцією* і здійснюється в холодний період року для економії тепла, витраченого на підігрівання припливного повітря. Однак можливість рециркуляції обумовлюється цілою низкою санітарно-гігієнічних та протипожежних вимог.

Місцева вентиляція

Місцева вентиляція може бути припливною і витяжною.

Місцева припливна вентиляція, при якій здійснюється концентроване подання припливного повітря заданих параметрів (температури, вологості, швидкості руху), виконується у вигляді повітряних душів, повітряних та повітряно-теплових завіс.

Повітряні душі використовуються для запобігання перегріванню робітників в гарячих цехах, а також для утворення так званих повітряних оазисів (ділянок виробничої зони, які різко відрізняються своїми фізико-хімічними характеристиками від решти приміщення).

Повітряні та повітряно-теплові завіси призначені для запобігання надходження в приміщення значних мас холодного зовнішнього повітря при необхідності частого відкривання дверей чи воріт. Повітряна завіса створюється струменем повітря, що подається із вузької довгої щілини, під деяким кутом назустріч потоку холодного повітря. Канал зі щілиною розміщують збоку чи зверху воріт (дверей).

Місцева витяжна вентиляція здійснюється за допомогою місцевих витяжних зонтів, всмоктуючих панелей, витяжних шаф, бортових відсмоктувачів.

Конструкція місцевої витяжки повинна забезпечити максимальне вловлювання шкідливих виділень при мінімальній кількості вилученого повітря. Крім того, вона не повинна бути громадкою та заважати обслуговуючому персоналу працювати і наглядати за технологічним процесом.

Основними чинниками при виборі типу місцевої витяжки є характеристики шкідливих виділень (температура, густина парів, токсичність), положення робітника при виконанні роботи, особливості технологічного процесу та устаткування.

У випадках, коли джерело виробничих шкідливостей можна помістити всередині простору, обмеженого стінками, місцеву витяжну вентиляцію влаштовують у вигляді витяжних шаф, кожухів, вітринних відсмоктувачів. Якщо за умовами технології або обслуговування джерело шкідливостей не можна ізолювати, тоді встановлюють витяжний зонт або всмоктувальну панель. При цьому потік повітря, що видаляється, не повинен проходити через зону дихання робітника.

Окремим випадком місцевої витяжної вентиляції є бортові відсмоктувачі, якими обладнують ванни (гальванічні, травильні) чи інші ємкості з токсичними рідинами, оскільки необхідність використати при їх завантаженні підйимально-транспортного обладнання унеможливило використання витяжних зонтів та всмоктувальних панелей. При ширині ванни 1 м і більше необхідно встановлювати бортовий відсмоктувач з передувом (рис. 2.6г), у якого з одного боку ванни повітря відсмоктується, а з іншого — нагнітається. При цьому рухоме повітря ніби екранує поверхню випаровування токсичних рідких продуктів.

5.2.3 Основні вимоги до систем вентиляції

Природна та штучна вентиляції повинні відповідати наступним санітарно-гігієнічним вимогам:

- створювати в робочій зоні приміщень нормовані метеорологічні умови праці (температуру, вологість і швидкість руху повітря);
- повністю усувати з приміщень шкідливі гази, пари, пил та аерозолі або розчиняти їх до гранично допустимих концентрацій;
- не вносити в приміщення забруднене повітря ззовні або шляхом засмоктування забрудненого повітря з суміжних приміщень;
- не створювати на робочих місцях протягів чи різкого охолодження;
- бути доступними для управління та ремонту під час експлуатації;
- не створювати під час експлуатації додаткових незручностей (наприклад, шуму, вібрацій, попадання дощу, снігу).

Найбільш повно вище перерахованим вимогам відповідає система кондиціонування повітря, яка також застосовується на підприємствах. За допомогою кондиціонерів створюються і автоматично підтримуються у виробничому приміщенні задані параметри повітряного середовища. При вирішенні питання щодо доцільності кондиціонування повітря слід враховувати і економічні чинники.

Необхідно зазначити, що до вентиляційних систем, встановлених у пожежонебезпечних та вибухонебезпечних приміщеннях висувається ціла низка додаткових вимог, які в цьому розділі не розглядаються.

Висновок до розділу 5

Метою даної дипломної роботи була розробка системи управління ІТ проектами АТ КБ «ПриватБанк».

Для досягнення цієї мети було проведено аналіз обороту технічних документів та службових записок, розроблено постановку задачі на проектування автоматизованої системи, виконано проектування системи. Було описано повний цикл створення системи, включаючи технічне завдання, а також інформаційне, технічне та програмне забезпечення. Система забезпечує можливість працювати з будь якого комп'ютера мережі

банку, з контролем доступу до проектів та задач, з можливістю створення задач та проектів, з єдиним сховищем даних.

Отриманий проект системи може бути використаний лише у банку «ПриватБанк» так як розробка велася на правах комерційної системи.

РОЗДІЛ 6. ОХОРОНА ОТОЧУЮЧОГО СЕРЕДОВИЩА

6.1. Вступ

Сучасні інформаційні системи, включно з системами управління ІТ-проєктами на базі DevOps, хоч і не є матеріалоемними або енергоінтенсивними продуктами виробництва, проте мають непрямий вплив на навколишнє середовище. Відповідальне використання енергоресурсів, оптимізація роботи серверної інфраструктури та скорочення кількості апаратного забезпечення є важливими аспектами екологічної безпеки в ІТ-секторі.

Розробка ІС для АТ КБ «ПриватБанк» передбачає врахування вимог екологічної безпеки, впровадження енергоощадних технологій та дотримання міжнародних підходів до «зелених» ІТ.

6.2. Екологічні аспекти використання ІТ-систем у корпоративному середовищі

Основні потенційні джерела впливу ІТ-проєктів на довкілля енергоспоживання серверів, дата-центрів та мережевого обладнання. Утворення електронних відходів при оновленні або списанні техніки. Викиди CO₂, пов'язані з виробництвом та експлуатацією електронного обладнання. Використання паперових носіїв, якщо відсутня цифровізація документообігу. Охолодження дата-центрів і витрати водних ресурсів у великих інфраструктурах.

Оскільки ПриватБанк є масштабною фінансовою установою, оптимізація ІТ-процесів та застосування DevOps-технологій безпосередньо впливає на загальне ресурсоспоживання.

6.3. Екологічні переваги впровадження DevOps-підходу

Використання DevOps у розробленій ІС дозволяє зменшити навантаження на навколишнє середовище завдяки таким чинникам:

Оптимізація використання серверних ресурсів.

Автоматизація розгортання, контейнеризація (Docker), віртуалізація та оркестрація (Kubernetes) дають змогу зменшити кількість фізичних серверів, збільшити ефективність використання наявних потужностей, скоротити витрати енергії.

Перехід до безпаперового документообігу.

Усі процеси управління проєктами виконуються в електронному вигляді, що знижує використання паперу на 80–95%.

Зменшення потреби у фізичній інфраструктурі.

Завдяки хмарним сервісам та мікросервісній архітектурі знижується потреба у додатковому обладнанні.

Скорочення вуглецевого сліду завдяки автоматизації.

Менше поїздок персоналу між офісами, зменшення кількості ручних операцій та часу простою систем.

Підтримка політики "Green IT".

DevOps-практики сприяють зниженню екологічних ризиків, включаючи контроль версій, стандартизацію процесів і зменшення кількості помилкових релізів.

6.4. Використання енергоефективних технологій

Під час розробки та експлуатації системи було враховано принципи енергоефективності застосування оптимізованих контейнерів, що використовують мінімальний обсяг ресурсів. Підтримка хмарної або гібридної інфраструктури з динамічним масштабуванням. Мінімізація навантаження на сервери завдяки оптимізованим алгоритмам обробки даних. Використання енергоощадних робочих станцій та моніторів у рамках загальної політики ПриватБанку. Централізований моніторинг та керування навантаженням.

6.5. Управління електронними відходами

У межах ІТ-підрозділу банку реалізуються заходи відповідальна утилізація та переробка списаного обладнання. Продовження життєвого циклу техніки шляхом модернізації та повторного використання. Здача непридатних елементів до сертифікованих центрів утилізації. Документування та контроль екологічно безпечного списання.

Розробка та впровадження нової ІС не потребує значного збільшення апаратного фонду, що мінімізує кількість нових електронних відходів.

6.6. Заходи щодо мінімізації впливу на довкілля

Для зниження екологічного впливу передбачено такі заходи використання «зелених» дата-центрів з енергоефективним охолодженням. Впровадження систем моніторингу енергоспоживання. Застосування електронних документів у всіх процесах. Оптимізація алгоритмів і сервісів системи для зменшення навантаження. Підвищення рівня екологічної свідомості співробітників через внутрішні тренінги. Мінімізація відходів за рахунок використання багаторазових та цифрових матеріалів.

Висновки до розділу 6

Розроблена інформаційна система управління ІТ-проєктами з DevOps сприяє зменшенню негативного впливу на довкілля завдяки використанню сучасних енергоефективних технологій, автоматизації процесів, скороченню паперового документообігу та оптимізації серверних ресурсів. Впровадження системи відповідає світовим тенденціям "Green IT" та сприяє формуванню екологічно відповідальної ІТ-інфраструктури в АТ КБ «ПриватБанк».

Таким чином, проєкт не лише підвищує ефективність управління ІТ-проєктами, але й забезпечує значний позитивний екологічний ефект, що є важливою складовою сталого розвитку організації.

ВИСНОВКИ

У результаті проведеного дослідження отримано такі основні результати та висновки.

Проведено системний аналіз існуючих процесів управління ІТ-проєктами в АТ КБ «ПриватБанк».

Виявлені основні недоліки традиційних підходів — низький рівень автоматизації, фрагментованість процесів, труднощі координації команд, тривалі цикли розробки та високі ризики помилок при розгортанні. Це підтвердило необхідність впровадження DevOps як інтеграційної концепції.

Досліджено альтернативні моделі роботи та сучасні методології, серед яких Agile, Lean, DevOps.

Порівняльний аналіз показав, що DevOps найбільше відповідає вимогам банківської ІТ-інфраструктури, оскільки забезпечує безперервну інтеграцію, доставку, автоматизацію процесів, а також прозору взаємодію між командами розробки й експлуатації.

Розроблено архітектурну модель майбутньої ІС.

Створено функціональні діаграми, моделі даних, сценарії використання, логічну та фізичну архітектури. Визначено ключові модулі системи: управління завданнями, моніторинг, автоматизація CI/CD, аналітика, система доступів та безпеки.

Реалізовано програмне рішення, що забезпечує підтримку DevOps-процесів.

Упроваджено інструменти планування, контролю виконання проєктів, автоматизовані конвеєри CI/CD, інтеграцію з системою контролю версій та корпоративною інфраструктурою. Результати тестування підтвердили стабільність і працездатність ІС.

Проведено техніко-економічне обґрунтування проєкту.

Встановлено, що впровадження системи є економічно вигідним: скорочується час випуску релізів, зменшується кількість помилок, оптимізуються ресурси. Очікуваний термін окупності — 1–2 роки.

Оцінено вплив системи на навколишнє середовище.

Показано, що розроблена ІС сприяє реалізації принципів «Green IT» — зменшує потребу в паперових носіях, оптимізує використання серверних ресурсів, знижує енергоспоживання.

Загалом виконана робота підтверджує доцільність застосування DevOps-підходу для створення сучасних систем управління ІТ-проєктами. Розроблена ІС є комплексним інструментом, що дозволяє підвищити продуктивність ІТ-підрозділу, прискорити життєвий цикл розробки, зменшити кількість помилок і забезпечити стабільність та надійність програмних рішень АТ КБ «ПриватБанк».

Отже, поставлена у роботі мета досягнута, а всі визначені завдання успішно виконано. Результати дослідження можуть бути використані для подальшого вдосконалення ІТ-процесів, масштабування DevOps-практик та впровадження аналогічних систем у інших організаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kim G., Humble J., Debois P., Willis J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. 2nd ed. Portland : IT Revolution Press, 2021. 520 p.
2. Forsgren N., Humble J., Kim G. Accelerate: The Science of Lean Software and DevOps. Portland : IT Revolution Press, 2018. 288 p.
3. Hüttermann M. DevOps for Developers. Cham : Springer International Publishing, 2018. 235 p.
4. Ажищев, В. Ф., Качурка, Д. І., Леус, С. О., & Морарь, М. І. (2025). Дослідження методів системи динамічного призначення ролей у системах керування доступом підприємства. *У матеріалах VI Міжнародної науково-практичної інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025)» (16–17 листопада 2025 р.)*. Миколаїв: НУК імені адмірала Макарова С. 245-247
<https://itconf.nuos.edu.ua/2025/publications/investigation-of-methods-for-dynamic-role-assignment-in-enterprise-access-control-systems/>
5. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Updated ed. Boston : Addison-Wesley, 2020. 512 p.
6. Bass L., Weber I., Zhu L. DevOps: A Software Architect's Perspective. 2nd ed. Boston : Addison-Wesley, 2020. 352 p.
7. Erich F., Amrit C., Daneva M. A Qualitative Study of DevOps Usage in Practice. *Journal of Software: Evolution and Process*, 2017, vol. 29, no. 6. DOI: 10.1002/smr.1838.
8. Gruver G. Starting and Scaling DevOps in the Enterprise. Portland : IT Revolution Press, 2018. 180 p.
9. Sachdeva S. DevOps Adoption Strategies: Frameworks, Methods, and Techniques. Boca Raton : CRC Press, 2022. 260 p.

10. Leite L., Rocha C., Kon F., Milojicic D., Meirelles P. A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys*, 2019, vol. 52, no. 6, pp. 1–35. DOI: 10.1145/3359981.
11. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. Newtown Square, PA : PMI, 2021. 370 p.

ДОДАТОК А – Технічне завдання

Підстави для розробки

Підставою для розробки інформаційної системи управління ІТ проектами ПАТ «ПриватБанк» є завдання на дипломну роботу.

Функціональне призначення системи

Дана система автоматизує процес управління проектами інформаційних технологій, спрощує процес управління проектом, підвищує ефективність управління проектами, надає зручні інструменти для спостереження за станом задач та проекту в цілому. Допомогає ефективно формувати послідовність виконання співробітником поставлених задач.

Даний програмний продукт має наступні функції:

- Функція авторизації для входу в систему тільки зареєстрованих користувачів;
- Функція реєстрації нових користувачів системи;
- Функції створення, редагування та видалення проектів;
- Функції створення, редагування та видалення задач;
- Функції пошуку задач та проектів;
- Функція складання та перегляду звітів по проекту.

Вимоги до надійності системи

- Резервне копіювання бази даних;
- Тимчасове блокування форми входу в систему при 5 разовому вводі неправильного пароля.
- Паролі користувачів повинні зберігатися в зашифрованому вигляді;
- Всі дані, введені користувачем повинні перевірятися на коректність.

Умови експлуатації

- Безперебійна робота комп'ютерів сервера прикладної програми та сервера реляційної бази даних;
- Надійне Інтернет з'єднання між сервером прикладної програми та клієнтами.

Вимоги до програмної документації

В програмну документацію повинні входити:

- Текст програми;
- Опис програми;
- Програма та методика випробувань;
- Керівництво користувача.

Вимоги до складу та параметрів технічних засобів

В зв'язку з тим, що програмне забезпечення має трирівневу архітектуру, то й технічне забезпечення складається з трьох частин.

Технічне забезпечення серверу реляційної бази даних

На сервері буде встановлено СУБД MySQL 5.5, у якій буде зберігатися база даних, яка включає в себе список користувачів, проекти, задачі а також додаткові довідкові дані. Сервер повинен уміти обслуговувати декілька серверів прикладних програм одночасно. В зв'язку з цим рекомендовані технічні характеристики сервера наступні:

- процесор - Intel® Core™ i5-2450M Processor (2.5 GHz), або краще;
- оперативна пам'ять - 4Гб або більше;
- вільне місце на жорсткому диску – 30Гб або більше;
- надійне з'єднання з мережею Інтернет;
- монітор;
- маніпулятори – клавіатура та миша.

Технічне забезпечення серверу прикладних програм

На сервері Tomcat буде розгорнуто серверний додаток який повинен обслуговувати багатьох клієнтів, та відсилати запити до серверу реляційної бази даних. В зв'язку з цим рекомендовані технічні характеристики сервера такі ж як і у сервера бази даних:

Технічне забезпечення клієнта

Технічне забезпечення клієнта це браузер, тому воно менш вимогливе до ресурсів ніж у серверів. Рекомендовані технічні характеристики клієнта:

- процесор - Intel® Pentium® Processor E5200 (2.50 GHz), або краще;
- оперативна пам'ять – 1Гб або більше;
- надійне з'єднання з мережею Інтернет;
- кольоровий монітор;
- відео карта з 64Мб власної пам'яті, або краще;
- маніпулятори – клавіатура та миша.

Техніко-економічні показники

Даний програмний продукт розробляється на базі внутрішньо банківської комерційної системи.

Порядок контролю

Контроль за розробкою кожної окремої частини автоматизованої системи тестування знань відбувається на кожному етапі з урахуванням вимог, визначених в цьому технічному завданні. Згідно програми та методики випробувань приймання системи до експлуатації відбувається після остаточного тестування всієї системи та відповідності її тестової роботи всім вимогам, викладеним в технічному завданні.

ДОДАТОК Б – Інструкція користувача

Інструкція системного адміністратора.

Для початку необхідно налаштувати сервер реляційних баз даних MySQL, для цього необхідно встановити його та виконати скрипт первинної ініціалізації бази даних, який приведено у додатку Г.

Далі необхідно встановити сервер прикладних програм Apache Tomcat, та розгорнути на ньому модуль системи управління проектами який був раніше зібраний та знаходиться згорнутий у файлі TaskBoard.war. Далі необхідно налаштувати з'єднання розгорнутої системи управління проектами з сервером реляційних баз даних, для цього необхідно в теці «./Apache Tomcat/conf/» створити файл налаштувань «Connection.properties», та вказати в ньому наступні параметри:

JDBC.URL= // Адреса реляційної бази даних

JDBC.LOGIN= // Ім'я користувача для підключення до бази даних

JDBC.PASSWORD= // Пароль користувача для підключення до бази даних.

Після чого можна вмикати сервер та починати працювати з ним.

Інструкція користувача.

На комп'ютерах клієнтах повинен бути встановлений браузер та налаштоване підключення до мережі Інтернет.

Для початку роботи користувач повинен зайти в систему та авторизуватися, на рисунку 1 зображено вікно авторизації користувача.



The image shows a simple login interface. At the top, there is a text label 'Username' followed by a rectangular input field. Below this, there is a text label 'Password' followed by another rectangular input field. At the bottom of the form, there is a blue rectangular button with the text 'Sign In' in white.

Рисунок 1 – Вікно авторизації користувача

Якщо були введені не коректні дані авторизації, користувачу буде показано повідомлення яке зображено на рисунку 2.



Рисунок 2 – Введені некоректні дані користувача

Якщо все було введено вірно, користувач переходить на стартову сторінку системи управління проектами, яка зображена на рисунку 3.

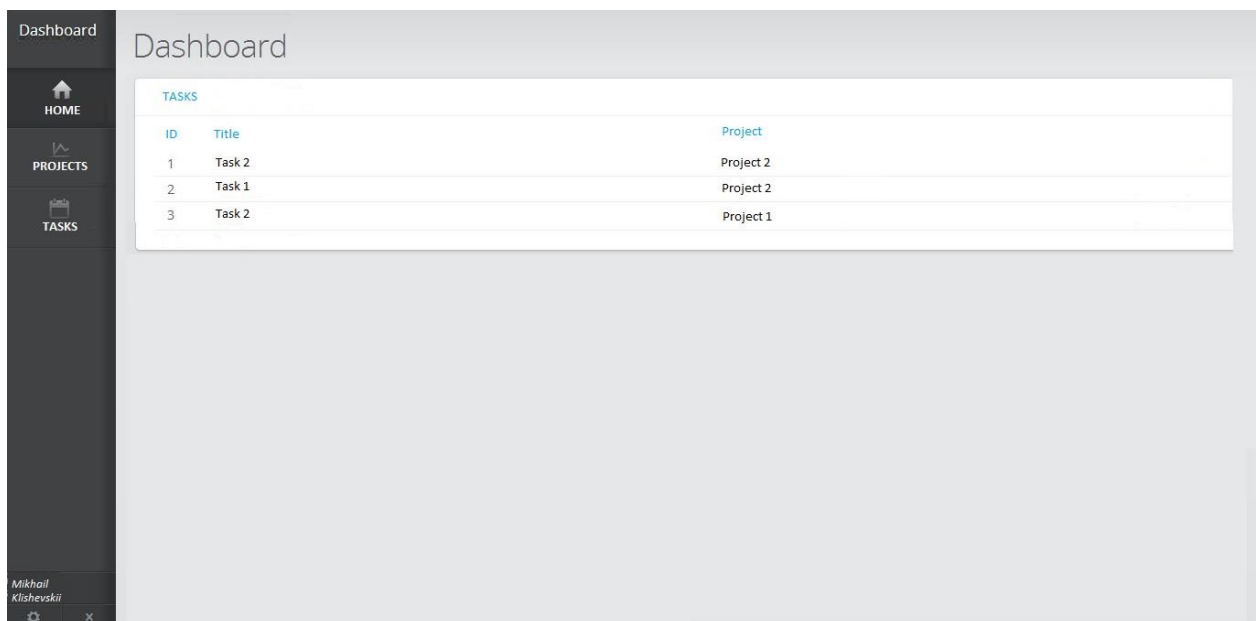


Рисунок 3 – Стартова сторінка системи

Як видно з малюнка 3, зліва користувачу доступна панель навігації, на котрій доступно декілька дій:

-  - кнопка виходу з текучої сесії користувача, після виходу з сесії користувача буде пере направлено на сторінку авторизації;
-  - кнопка налаштувань профілю користувача. Вікно з налаштуваннями користувача зображено на рисунку 4.

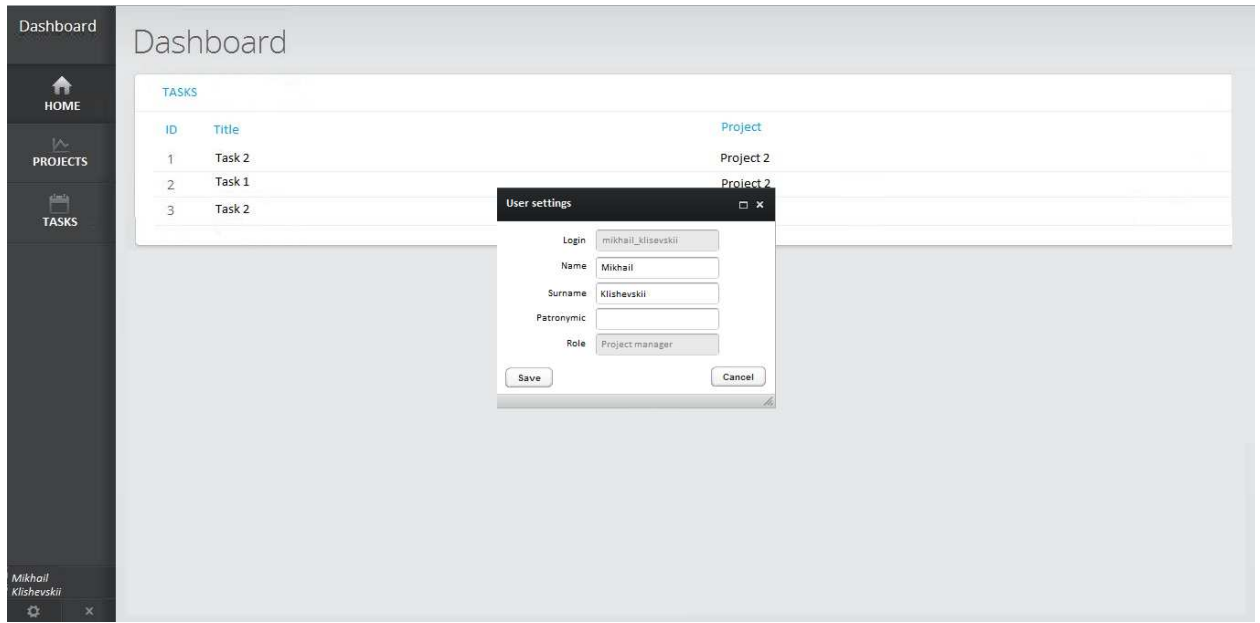


Рисунок 4 – Вікно налаштування профілю користувача

-  - кнопка повернення до стартової сторінки.
-  - кнопка переходу до сторінки перегляду, редагування та створення проектів. Вікно з відображенням проектів зображено на рисунку 5.

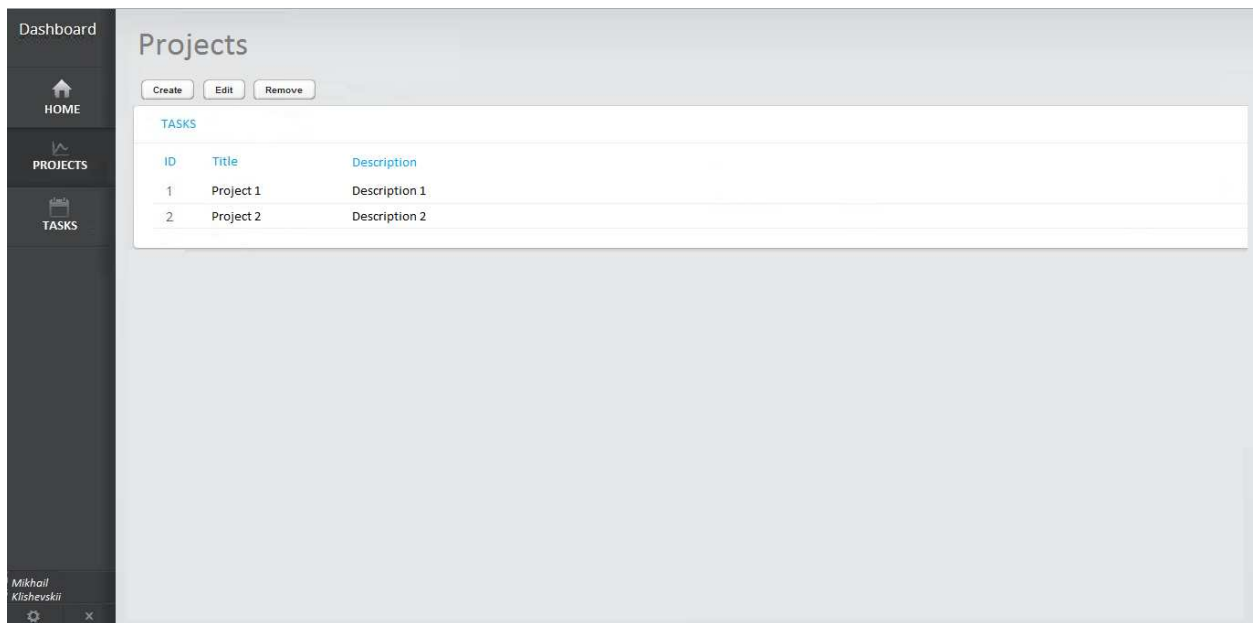


Рисунок 5 - Вікно з відображенням проектів

У даному вікні доступно 3 дії, створення, редагування та видалення проектів. На рисунку 6 зображено вікно створення/редагування проекту.

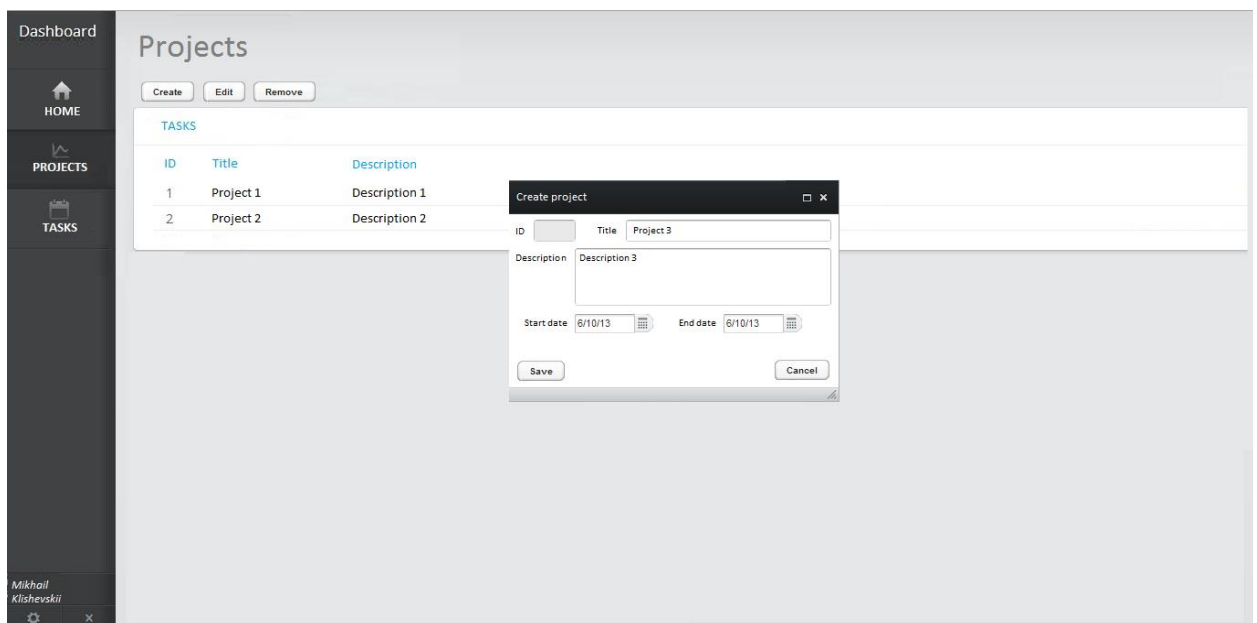


Рисунок 6 – Вікно створення/редагування проекту



- кнопка переходу до сторінки перегляду, редагування та створення задач. Вікно з відображенням задач зображено на рисунку 7.

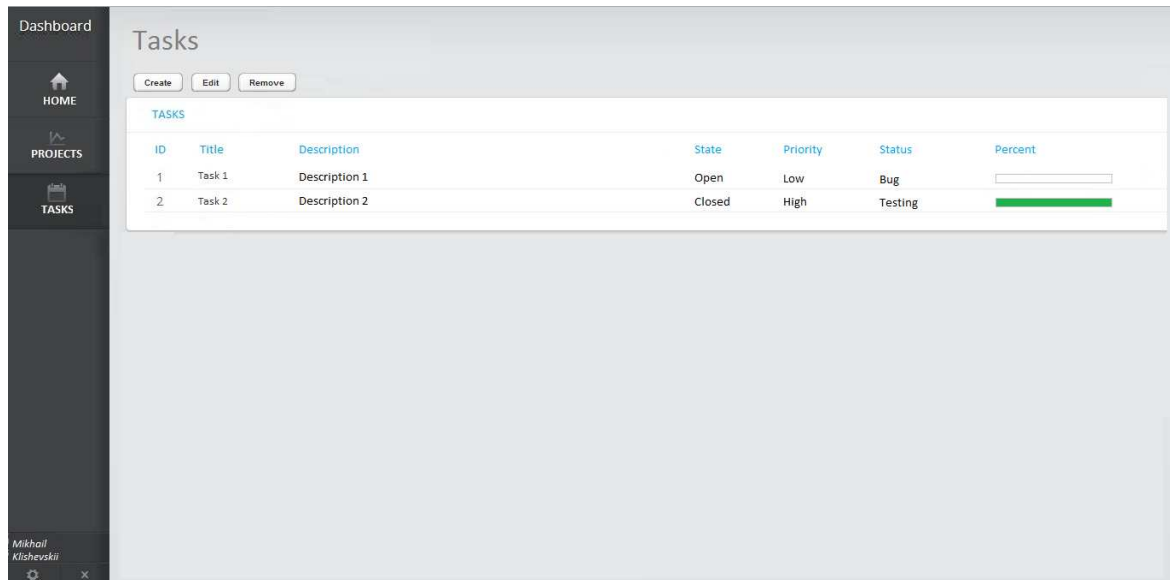


Рисунок 7 – Вікно відображення задач

У даному вікні також доступно 3 дії, створення, редагування та видалення задач. На рисунку 8 зображено вікно створення/редагування задачі.

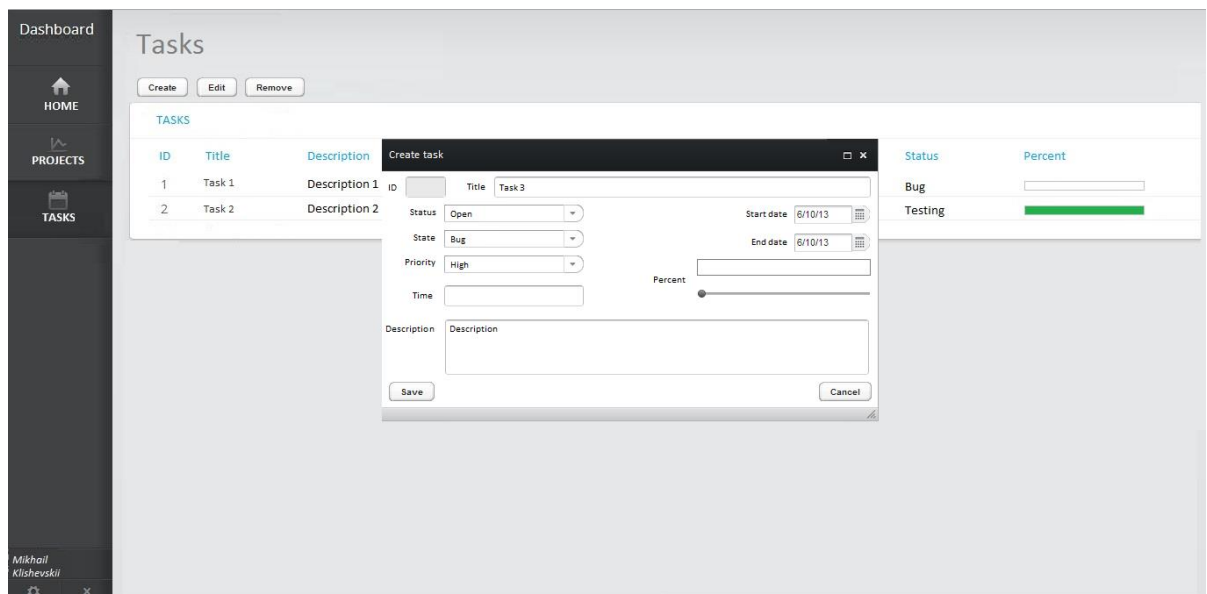


Рисунок 8 – вікно створення/редагування задачі

ДОДАТОК В – Технічний опис

Загальні відомості

Назва: «Інформаційна система управління ІТ проектами ПАТ «ПриватБанк»».

Основним засобом розробки програмного забезпечення системи є програмне середовище IntelliJ IDEA. Програмне забезпечення виконане мовою програмування java. Інструмент зборки проекту – Apache Maven, інструмент створення web-інтерфейсу фреймворк Vaadin, для ведення бази даних сервер реляційних баз даних – MySQL, сервер прикладних програм – Tomcat.

Функціональне призначення

Дана система автоматизує процес управління проектами інформаційних технологій, спрощує процес управління проектом, підвищує ефективність управління проектами, надає зручні інструменти для спостереження за станом задач та проекту в цілому. Допомогає ефективно формувати послідовність виконання співробітником поставлених задач.

Даний програмний продукт має наступні функції:

- Функція авторизації для входу в систему тільки зареєстрованих користувачів;
- Функція реєстрації нових користувачів системи;
- Функції створення, редагування та видалення проектів;
- Функції створення, редагування та видалення задач;
- Функції пошуку задач та проектів;
- Функція складання та перегляду звітів по проекту.

Опис логічної структури

При розробці програмного забезпечення системи використані методи об'єктно-орієнтованого проектування і розробки. Основними частинами програмного забезпечення є клієнтський web-додаток, web-серверна частина та серверна частина реляційної бази даних.

Клієнтський web-додаток реалізує графічний інтерфейс користувача, надає інструменти перегляду та редагування інформації, функції передачі даних на web-сервер.

Web-сервер відповідає за обробку інформації з клієнта, виконує запити до бази даних та конвертує результативні списки у внутрішні об'єкти системи.

Сервер бази даних відповідає за зберігання та надання інформації системи.

Використовувані технічні засоби

Рекомендовані технічні характеристики сервера наступні:

- процесор - Intel® Core™ i5, або краще;
- оперативна пам'ять - 4Гб або більше;
- вільне місце на жорсткому диску – 30Гб або більше;
- надійне з'єднання з мережею Інтернет;
- монітор;
- маніпулятори – клавіатура та миша.

Рекомендовані технічні характеристики клієнта:

- процесор - Intel® Pentium® Processor E5200, або краще;
- оперативна пам'ять – 1Гб або більше;
- вільне місце на жорсткому диску – 1Гб;
- надійне з'єднання з мережею Інтернет;
- кольоровий монітор;
- відео карта з 64Мб власної пам'яті, або краще;
- маніпулятори – клавіатура та миша.

ДОДАТОК Г – Фрагмент програмного коду

Нижче приведено лістинги файлів модуля Server.

Лістинг 1 – файл ConnectionCacheManager.java

```
package com.nk.utils.resources;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

public class ConnectionCacheManager {

    private static Map<String, String> connectionCache = new
HashMap<>();

    private static final String CONNECTION_CACHE_FILE_PATH =
"conncache/Connection.properties";

    static {
        String line, line_t;

        try (InputStream resourceAsStream =
ResourceManager.class.getResourceAsStream(
ConnectionCacheManager.CONNECTION_CACHE_FILE_PATH);
            InputStreamReader inputStreamReader = new
InputStreamReader(resourceAsStream);
            BufferedReader bufferedReader = new
BufferedReader(inputStreamReader)) {

            while ((line_t = bufferedReader.readLine()) != null) {
                line = new String(line_t.getBytes(), "UTF-8");
                int indexOf = line.indexOf("=");
                if (line.startsWith("#") || indexOf == -1 ||
line.length() == indexOf) continue;

                ConnectionCacheManager.connectionCache.put(line.substring(0, indexOf),
                    line.substring(indexOf + 1,
line.length()));
            }
        } catch (Exception e) {
            // TODO add log out
            e.printStackTrace();
        }
    }

    public static String getProperty(String property) {
        return
ConnectionCacheManager.connectionCache.get(property);
    }
}
```

Лістинг 2 – файл ResourceLocale.java

```
package com.nk.utils.resources;

/** Перечисление локализаций системы. */
public enum ResourceLocale {
    DEFAULT(0, "ENG"),
    RUS(1, "RUS");

    private int languagePKey;
    private String language;

    private ResourceLocale(int languagePKey, String language) {
        this.languagePKey = languagePKey;
        this.language = language;
    }

    /**
     * Метод получения алиаса языка.
     *
     * @return {@link String} алиас языка.
     */
    public String getLanguage() {
        return language;
    }

    /**
     * Метод получения пкея локализации системы.
     *
     * @return {@link Integer} пкей локализации системы.
     */
    public int getLanguagePkey() {
        return languagePKey;
    }

    /**
     * Метод поиска локали системы по пкею.
     *
     * @param pkey {@link Integer} пкей локали.
     * @return {@link ResourceLocale} локаль.
     */
    public static ResourceLocale getLocaleByPkey(int pkey) {
        for (ResourceLocale locale : values()) {
            if (locale.getLanguagePkey() == pkey) {
                return locale;
            }
        }

        return ResourceLocale.DEFAULT;
    }

    /**
     * Метод поиска локали системы по языку.
     *
     * @param language {@link String} язык.
     * @return {@link ResourceLocale} локаль.
     */
}
```

```

        */
        public static ResourceLocale getLocaleByLanguage(String
language) {
            for (ResourceLocale locale : values()) {
                if (locale.getLanguage().equalsIgnoreCase(language)) {
                    return locale;
                }
            }

            return ResourceLocale.DEFAULT;
        }

        @Override
        public String toString() {
            return ResourceManager.getString(this,
"Resource.Language");
        }
    }
}

```

Лістинг 3 – файл ResourceManager.java

```

package com.nk.utils.resources;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

/** Класс языковых ресурсов системы. */
public class ResourceManager {

    private static Map<ResourceLocale, Map<String, String>>
resources;
    //*****
    private static final String RESOURCE_FILE_PATH_MASK =
"lang/%s.resource";

    static {
        ResourceManager.resources = new HashMap<>();
        Map<String, String> language;

        for (ResourceLocale locale : ResourceLocale.values()) {
            String filePath =
String.format(ResourceManager.RESOURCE_FILE_PATH_MASK,
locale.getLanguage());
            String line, line_t;

            language = new HashMap<>();
            try (InputStream resourceAsStream =
ResourceManager.class.getResourceAsStream(filePath);
InputStreamReader inputStreamReader = new
InputStreamReader(resourceAsStream);
BufferedReader bufferedReader = new
BufferedReader(inputStreamReader)) {

```

```

        while ((line_t = bufferedReader.readLine()) !=
null) {
            line = new String(line_t.getBytes(), "UTF-8");
            int indexOf = line.indexOf("=");
            if (line.startsWith("#") || indexOf == -1 ||
line.length() == indexOf) continue;

            language.put(line.substring(0, indexOf),
line.substring(indexOf + 1, line.length()));
        }
        } catch (Exception e) {
            // TODO add log out
            e.printStackTrace();
        }

        ResourceManager.resources.put(locale, language);
    }
}

/**
 * Метод получения локализованной строки.
 *
 * @param key {@link String} ключ.
 * @return - {@link String} локализованная строка.
 */
public static String getString(ResourceLocale locale, String
key) {
    Map<String, String> language =
ResourceManager.resources.get(locale);
    if (language != null) {
        return (language.isEmpty() || language.get(key) ==
null) ? key : language.get(key);
    } else {
        // TODO Add log out
        return key;
    }
}
}

```

Лістинг 4 – файл DBManagerI.java

```

package com.nk.api.dao;

import com.nk.api.exception.ServerException;
import com.nk.api.exception.ServerRuntimeError;

import java.sql.CallableStatement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

/** Интерфейс менеджера подключения к БД */
public interface DBManagerI extends AutoCloseable {

    /**
     * Метод подключения к базе данных.
     */
}

```

```

        *
        * @throws ServerException исключение при работе системы.
        * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    void connect() throws ServerException, ServerRuntimeError;

    /**
     * Метод создания заявления в БД.
     *
     * @return {@link java.sql.Statement} стейтмент.
     * @throws ServerException исключение при работе системы.
     * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    Statement createStatement() throws ServerException,
    ServerRuntimeError;

    /**
     * Метод создания подготавливаемого стейтмента.
     *
     * @param sql {@link String} запрос в базу данных.
     * @return {@link java.sql.PreparedStatement} подготавливаемый
стейтмент.
     * @throws ServerException исключение при работе системы.
     * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    PreparedStatement prepareStatement(String sql) throws
    ServerException, ServerRuntimeError;

    /**
     * Метод создания подготавливаемого стейтмента.
     *
     * @param sql {@link String} запрос в базу данных.
     * @return {@link java.sql.CallableStatement} подготавливаемый
стейтмент.
     * @throws ServerException исключение при работе системы.
     * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    CallableStatement prepareCallableStatement(String sql) throws
    ServerException, ServerRuntimeError;

    /**
     * Выполнение запроса в БД.
     *
     * @param sql {@link String} запрос в базу данных.
     * @throws ServerException исключение при работе системы.
     * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    boolean execute(String sql) throws ServerException,
    ServerRuntimeError;

    /**

```

```

        * Выполнение запроса в БД.
        *
        * @param sql {@link String} запрос в базу данных.
        * @return {@link Integer} количество измененных строк.
        * @throws ServerException исключение при работе системы.
        * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    int executeUpdate(String sql) throws ServerException,
ServerRuntimeError;

    /**
    * Выполнение запроса в БД.
    *
    * @param sql {@link String} запрос в базу данных.
    * @return {@link java.sql.ResultSet} результирующий список.
    * @throws ServerException исключение при работе системы.
    * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    ResultSet executeQuery(String sql) throws ServerException,
ServerRuntimeError;

    /**
    * Метод для закрытия результирующего списка.
    *
    * @param resultSet {@link java.sql.ResultSet} результирующий
список.
    * @throws ServerException исключение при работе системы.
    * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    void closeResultSet(ResultSet resultSet) throws
ServerException, ServerRuntimeError;

    /**
    * Метод для закрытия стейтмента.
    *
    * @param statement {@link java.sql.Statement} стейтмент.
    * @throws ServerException исключение при работе системы.
    * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    void closeStatement(Statement statement) throws
ServerException, ServerRuntimeError;

    /**
    * Метод для закрытия подключения к БД.
    *
    * @throws ServerException исключение при работе системы.
    * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
    */
    void close() throws ServerException, ServerRuntimeError;

    /**

```

```

        * Метод проверки закрыт менеджер или нет.
        *
        * @return {@code true} - в случае если менеджер закрыт, иначе
{@code false}
        * @throws ServerException исключение при работе системы.
        * @throws ServerRuntimeError исключение при внутренней ошибке
системы.
        */
        boolean isClose() throws ServerException, ServerRuntimeError;
    }

```

Листинг 5 – файл DBManager.java

```

package com.nk.core.dao;

import com.nk.api.dao.DBManagerI;
import com.nk.api.entities.session.UserSessionI;
import com.nk.api.exception.ExceptionCode;
import com.nk.api.exception.ExceptionFactory;
import com.nk.api.exception.ServerException;
import com.nk.api.exception.ServerRuntimeError;
import com.nk.utils.Constants;
import com.nk.utils.resources.ConnectionCacheManager;

import java.sql.*;

public class DBManager implements DBManagerI {

    private Connection connection;

    DBManager(UserSessionI session) throws ServerException,
ServerRuntimeError {
        try {

Class.forName(ConnectionCacheManager.getProperty(Constants.JDBC_DRIVER
));
                } catch (ClassNotFoundException e) {
                    throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.DB_CONNECTION_RUNTIME_EXCEPTION);
                }
        }

    @Override
    public void connect() throws ServerException,
ServerRuntimeError {
        try {
            connection = DriverManager.getConnection(

ConnectionCacheManager.getProperty(Constants.JDBC_URL),

ConnectionCacheManager.getProperty(Constants.JDBC_LOGIN),

ConnectionCacheManager.getProperty(Constants.JDBC_PASSWORD));
        } catch (SQLException e) {

```



```

        throw
        ExceptionFactory.createServerException(e.getMessage(), e,
        ExceptionCode.SQL_EXCEPTION);
    }

    @Override
    public Statement createStatement() throws ServerException,
    ServerRuntimeError {
        try {
            return connection.createStatement();
        } catch (SQLException e) {
            throw
            ExceptionFactory.createServerException(e.getMessage(), e,
            ExceptionCode.SQL_EXCEPTION);
        }
    }

    @Override
    public PreparedStatement prepareStatement(String sql) throws
    ServerException, ServerRuntimeError {
        try {
            return connection.prepareStatement(sql);
        } catch (SQLException e) {
            throw
            ExceptionFactory.createServerException(e.getMessage(), e,
            ExceptionCode.SQL_EXCEPTION);
        }
    }

    @Override
    public CallableStatement prepareCallableStatement(String sql)
    throws ServerException, ServerRuntimeError {
        try {
            return connection.prepareCall(sql);
        } catch (SQLException e) {
            throw
            ExceptionFactory.createServerException(e.getMessage(), e,
            ExceptionCode.SQL_EXCEPTION);
        }
    }

    @Override
    public boolean execute(String sql) throws ServerException,
    ServerRuntimeError {
        try {
            return createStatement().execute(sql);
        } catch (SQLException e) {
            throw
            ExceptionFactory.createServerException(e.getMessage(), e,
            ExceptionCode.SQL_EXCEPTION);
        }
    }

    @Override

```

```

        public int executeUpdate(String sql) throws ServerException,
ServerRuntimeError {
            try {
                return createStatement().executeUpdate(sql);
            } catch (SQLException e) {
                throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
            }
        }

        @Override
        public ResultSet executeQuery(String sql) throws
ServerException, ServerRuntimeError {
            try {
                return createStatement().executeQuery(sql);
            } catch (SQLException e) {
                throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
            }
        }

        @Override
        public void closeResultSet(ResultSet resultSet) throws
ServerException, ServerRuntimeError {
            if (resultSet != null) {
                try {
                    resultSet.close();
                } catch (SQLException e) {
                    throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
                }
            }
        }

        @Override
        public void closeStatement(Statement statement) throws
ServerException, ServerRuntimeError {
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException e) {
                    throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
                }
            }
        }

        @Override
        public void close() throws ServerException, ServerRuntimeError
{
            if (connection != null) {
                try {

```

```

        connection.close();
    } catch (SQLException e) {
        throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
    }
}

@Override
public boolean isClose() throws ServerException,
ServerRuntimeError {
    if (connection != null) {
        try {
            return connection.isClosed();
        } catch (SQLException e) {
            throw
ExceptionFactory.createServerException(e.getMessage(), e,
ExceptionCode.SQL_EXCEPTION);
        }
    }
    return true;
}
}

```

Лістинг 6 – файл UserSessionI.java

```

package com.nk.api.entities.session;

import com.nk.api.entities.EntityI;
import com.nk.utils.resources.ResourceLocale;

import java.util.Calendar;

/** Интерфейс сессии пользователя. */
public interface UserSessionI extends EntityI<String> {

    /**
     * Метод получения идентификатора сессии пользователя.
     *
     * @return {@link String} идентификатор сессии пользователя.
     */
    String getSessionID();

    /**
     * Метод получения имени пользователя.
     *
     * @return {@link String} имя пользователя.
     */
    String getLogin();

    /**
     * Метод получения времени открытия сессии пользователя.
     *
     * @return {@link java.util.Calendar} время открытия сессии
пользователя.

```

```

    */
    Calendar getOpenSessionTime();

    /**
     * Метод получения локали пользователя.
     *
     * @return {@link com.nk.utils.resources.ResourceLocale}
    локаль пользователя.
     */
    ResourceLocale getLocale();

    /**
     * Метод установки локализации пользователя.
     *
     * @param locale {@link ResourceLocale} локализация.
     */
    void setLocale(ResourceLocale locale);

    /**
     * Метод получения имени пользователя.
     *
     * @return {@link String} имя пользователя.
     */
    String getName();

    /**
     * Метод установки имени пользователя.
     *
     * @param name {@link String} имя пользователя.
     */
    void setName(String name);

    /**
     * Метод получения фамилии пользователя.
     *
     * @return {@link String} фамилия пользователя.
     */
    String getSurname();

    /**
     * Метод установки фамилии пользователя.
     *
     * @param surname {@link String} фамилия пользователя.
     */
    void setSurname(String surname);

    /**
     * Метод получения отчества пользователя.
     *
     * @return {@link String} отчество пользователя.
     */
    String getPatronymic();

    /**
     * Метод установки отчества пользователя.
     *

```

```

    * @param patronymic {@link String} отчество пользователя.
    */
    void setPatronymic(String patronymic);

    /** Метод закрытия сессии пользователя. */
    void close();

    /**
     * Метод проверки состояния пользователя.
     *
     * @return <p/> - Boolean.TRUE - если сессия открыта.
     *          <p/> - Boolean.FALSE - если сессия закрыта.
     */
    boolean isClosed();
}

```

Лістинг 7 – файл UserSession.java

```

package com.nk.core.entities.session;

import com.nk.api.entities.session.UserSessionI;
import com.nk.utils.resources.ResourceLocale;

import java.util.Calendar;

public class UserSession implements UserSessionI {

    private String sessionId;
    private String login;
    private String name;
    private String surname;
    private String patronymic;
    private Calendar openSessionTime;
    private ResourceLocale locale = ResourceLocale.DEFAULT;
    private Boolean openSession = Boolean.FALSE;

    public UserSession(String login) {
        initializeSession(login, ResourceLocale.DEFAULT);
    }

    protected void initializeSession(String login, ResourceLocale
locale) {
        this.login = login;
        this.locale = locale;

        this.openSessionTime = Calendar.getInstance();
        this.sessionID = openSessionTime.getTime().toString() +
        "_" + this.login;

        this.openSession = Boolean.TRUE;
    }

    @Override
    public String getSessionID() {
        return sessionId;
    }
}

```

```
}

@Override
public String getLogin() {
    return login;
}

@Override
public Calendar getOpenSessionTime() {
    return openSessionTime;
}

@Override
public ResourceLocale getLocale() {
    return locale;
}

@Override
public void setLocale(ResourceLocale locale) {
    this.locale = locale;
}

@Override
public String getName() {
    return name;
}

@Override
public void setName(String name) {
    this.name = name;
}

@Override
public String getSurname() {
    return surname;
}

@Override
public void setSurname(String surname) {
    this.surname = surname;
}

@Override
public String getPatronymic() {
    return patronymic;
}

@Override
public void setPatronymic(String patronymic) {
    this.patronymic = patronymic;
}

@Override
public void close() {
    openSession = Boolean.FALSE;
}
```

```

@Override
public boolean isClosed() {
    return !openSession;
}

@Override
public String getKey() {
    return login;
}
}

```

Лістинг 8 – файл EntityI.java

```

package com.nk.api.entities;

import java.io.Serializable;

/**
 * Интерфейс определяющий сущность в БД.
 *
 * @param <KeyClass> тип значения первичного ключа.
 */
public interface EntityI<KeyClass> extends Serializable {

    /**
     * Метод получения первичного ключа сущности.
     *
     * @return {@link KeyClass} первичный ключ сущности.
     */
    KeyClass getKey();
}

```

Далі приведено лістинги файлів модуля Client.

Лістинг 9 – файл web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <display-name>Vaadin Web Application</display-name>

    <context-param>
        <description>Vaadin production mode</description>
        <param-name>productionMode</param-name>
        <param-value>false</param-value>
    </context-param>

    <servlet>
        <servlet-name>Vaadin Application Servlet</servlet-name>

```

```

        <servlet-class>com.vaadin.server.VaadinServlet</servlet-
class>

        <init-param>
            <description>Vaadin UI to display</description>
            <param-name>UI</param-name>
            <param-value>com.nk.client.main.TaskBoardUI</param-
value>
        </init-param>

        <init-param>
            <description>Application widgetset</description>
            <param-name>widgetset</param-name>
            <param-
value>com.nk.client.main.TaskBoardWidgetSet</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>Vaadin Application Servlet</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <session-config>
        <session-timeout>10</session-timeout>
    </session-config>
</web-app>

```

Листинг 10 – файл SessionControllerI.java

```

package com.nk.client.main.login;

import com.nk.api.entities.session.UserSessionI;
import com.nk.client.common.exceptions.ClientError;
import com.nk.client.common.exceptions.ClientException;
import com.nk.utils.resources.ResourceLocale;

/** Интерфейс контроллера создания\завершения сессии. */
public interface SessionControllerI {

    /**
     * Метод вызывается на событие входа на форме аутентификации.
     *
     * @param login    {@link String} логин пользователя
     * @param password {@link String} пароль пользователя
     * @param locale   {@link
com.nk.utils.resources.ResourceLocale} локализация
     * @throws ClientError    ошибка.
     * @throws ClientException исключение.
     */
    UserSessionI login(String login, String password,
ResourceLocale locale) throws ClientException, ClientError;

    /**
     * Метод вызывается на событие выхода.

```



```

        *
        * @param session {@link UserSessionI} сессия пользователя.
        * @throws ClientError      ошибка.
        * @throws ClientException  исключение.
        */
        void logout(UserSessionI session) throws ClientException,
ClientError;

    /**
     * Метод восстановления сессии пока не закрыта HTTP сессия.
     *
     * @throws ClientError      ошибка.
     * @throws ClientException  исключение.
     */
    UserSessionI tryRestoreSession() throws ClientException,
ClientError;
}

```

Листинг 11 – файл SessionController.java

```

package com.nk.client.main.login;

import com.nk.api.controllers.authenticator.AuthenticatorFactory;
import com.nk.api.controllers.authenticator.AuthenticatorFactoryI;
import com.nk.api.controllers.authenticator.AuthenticatorI;
import com.nk.api.entities.session.UserSessionI;
import com.nk.api.exception.ServerException;
import com.nk.api.exception.ServerRuntimeException;
import com.nk.client.common.exceptions.ClientError;
import com.nk.client.common.exceptions.ClientException;
import com.nk.client.utils.VaadinSessionManager;
import com.nk.utils.resources.ResourceLocale;

public class SessionController implements SessionControllerI {

    private String SESSION_ATTRIBUTE_NAME = "user_session";

    private AuthenticatorFactoryI getAuthenticatorFactory() {
        return new AuthenticatorFactory();
    }

    @Override
    public UserSessionI tryRestoreSession() throws
ClientException, ClientError {
        return
VaadinSessionManager.getAttribute(SESSION_ATTRIBUTE_NAME);
    }

    @Override
    public void logout(UserSessionI session) throws
ClientException, ClientError {
        VaadinSessionManager.removeAttribute(SESSION_ATTRIBUTE_NAME);
        session.close();
    }
}

```

```

        @Override
        public UserSessionI login(String login, String password,
ResourceLocale locale) throws ClientException, ClientError {
            try {
                AuthenticatorI userAuthenticator =
getAuthenticatorFactory().createUserAuthenticator();
                UserSessionI session =
userAuthenticator.authenticate(login, password, locale);

VaadinSessionManager.addAttribute(SESSION_ATTRIBUTE_NAME, session);
                return session;
            } catch (ServerRuntimeError e) {
                throw new ClientError(e.getMessage(), e);
            } catch (ServerException e) {
                throw new ClientException(e.getMessage(), e);
            }
        }
    }
}

```

Лістинг 12 – файл LoginWindowActionI.java

```

package com.nk.client.main.login;

import com.nk.client.common.exceptions.ClientError;
import com.nk.client.common.exceptions.ClientException;
import com.nk.utils.resources.ResourceLocale;

/** Интерфейс события на форме аутентификации. */
public interface LoginWindowActionI {

    /**
     * Метод вызывается на событие входа на форме аутентификации.
     *
     * @param login      {@link String} логин пользователя
     * @param password   {@link String} пароль пользователя
     * @param locale     {@link ResourceLocale} локализация
     * @throws ClientError      ошибка аутентификации.
     * @throws ClientException  исключение аутентификации.
     */
    void login(String login, String password, ResourceLocale
locale) throws ClientException, ClientError;
}

```

Лістинг 13 – файл LoginWindow.java

```

package com.nk.client.main.login;

import com.nk.client.common.components.AbstractAction;
import com.nk.client.common.exceptions.ClientError;
import com.nk.client.common.exceptions.ClientException;
import com.nk.utils.resources.ResourceLocale;
import com.nk.utils.resources.ResourceManager;
import com.vaadin.data.Property;

```

```

import com.vaadin.event.ShortcutAction;
import com.vaadin.ui.*;

import java.util.Arrays;

public class LoginWindow extends Window {

    private final LoginWindowActionI loginAction;
    //*****
    private TextField loginTextField;
    private PasswordField passwordField;
    private Button button;
    private ComboBox comboBox;
    private ResourceLocale locale = ResourceLocale.DEFAULT;

    public LoginWindow(LoginWindowActionI loginAction) {
        super(ResourceManager.getString(ResourceLocale.DEFAULT,
"LoginWindow.Caption"));
        this.loginAction = loginAction;
        initialize();
    }

    private void initialize() {
        this.setModal(true);
        this.setResizable(false);
        this.setClosable(false);
        this.setDraggable(false);

        HorizontalLayout horizontalLayout = new
HorizontalLayout();
        VerticalLayout verticalLayout = new VerticalLayout();

        this.setContent(horizontalLayout);

        horizontalLayout.addComponent(verticalLayout);
        horizontalLayout.setComponentAlignment(verticalLayout,
Alignment.MIDDLE_CENTER);
        verticalLayout.setMargin(true);
        verticalLayout.setSpacing(true);

        loginTextField = new TextField();
        passwordField = new PasswordField();
        button = new Button();
        button.addClickListener(getLoginAction());

        comboBox = new
ComboBox(ResourceManager.getString(ResourceLocale.DEFAULT,
"Resource.Language.ComboBox"),
Arrays.asList(ResourceLocale.values()));
        comboBox.setValue(ResourceLocale.DEFAULT);
        comboBox.setImmediate(true);
        comboBox.setNullSelectionAllowed(false);
        comboBox.addValueChangeListener(new
Property.ValueChangeListener() {
            @Override

```

```

        public void valueChange(Property.ValueChangeEvent
event) {
            updateCaptions((ResourceLocale)
event.getProperty().getValue());
        }
    });

    verticalLayout.addComponent(loginTextField);
    verticalLayout.setComponentAlignment(loginTextField,
Alignment.MIDDLE_CENTER);

    verticalLayout.addComponent(passwordField);
    verticalLayout.setComponentAlignment(passwordField,
Alignment.MIDDLE_CENTER);

    verticalLayout.addComponent(button);
    verticalLayout.setComponentAlignment(button,
Alignment.MIDDLE_CENTER);

    verticalLayout.addComponent(comboBox);
    verticalLayout.setComponentAlignment(comboBox,
Alignment.BOTTOM_CENTER);

    addAction(new Button.ClickShortcut(button,
ShortcutAction.KeyCode.ENTER));
    loginTextField.focus();

    updateCaptions(locale);
}

private void updateCaptions(ResourceLocale locale) {
    this.locale = locale;

    setCaption(ResourceManager.getString(locale,
"LoginWindow.Caption"));

    loginTextField.setInputPrompt(ResourceManager.getString(locale,
"LoginWindow.Login.Field"));

    passwordField.setInputPrompt(ResourceManager.getString(locale,
"LoginWindow.Password.Field"));
    button.setCaption(ResourceManager.getString(locale,
"LoginWindow.Button.LogIn"));
    comboBox.setInputPrompt(ResourceManager.getString(locale,
"Resource.Language.ComboBox"));
}

private Button.ClickListener getLoginAction() {
    return new AbstractAction("") {
        @Override
        public void actionPerformed(Button.ClickEvent event,
String command) throws ClientException, ClientError {
            loginAction();
        }

        @Override

```

```

        public boolean beforeAction(Button.ClickEvent event,
String command) throws ClientException, ClientError {
            return checkLoginAction();
        }
    };
}

    private void loginAction() throws ClientException, ClientError
{
        loginAction.login(loginTextField.getValue(),
passwordField.getValue(), locale);
        close();
    }

    private boolean checkLoginAction() throws ClientException,
ClientError {
        if (loginTextField.getValue().trim().length() > 0
            && passwordField.getValue().trim().length() > 0) {
            return true;
        } else {
            throw new
ClientException(ResourceManager.getString(locale,
"LoginWindow.Empty.Fields"));
        }
    }
}

```