

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»**

Кафедра інформаційних систем в економіці

**ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»**

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

Форма навчання: заочна

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Інформаційна управляюча система сервісної підтримки ІТ
інфраструктури АТ "Київгаз"»

здобувача Дівеева Ярослава Ігоровича
(ПІБ, підпис)

Науковий керівник: к.е.н., доцент Помазун О.М.
(науковий ступінь, учене звання, ПІБ)

(підпис)

**Робота допущена до захисту перед екзаменаційною
комісією з атестації здобувачів вищої освіти (ЕК)**

Завідувач кафедри: к.е.н., доцент Тішков Б.О.
(науковий ступінь, учене звання, ПІБ)

(підпис)

Київ 2024

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА

«Інформаційні управляючі системи і технології»

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

ПОГОДЖЕНО:

Керівник проєктної групи (гарант)
освітньо-професійної програми

_____ Устенко С.В.
« ____ » _____ 202_ р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри інформаційних
систем в економіці

_____ Тішков Б.О.
« ____ » _____ 202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Дівесєва Ярослава Ігоровича
(прізвище, ім'я, по батькові)

заочної форми навчання

очної (денної), заочної, дистанційної

на підготовку кваліфікаційної магістерської роботи

на тему: «Інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз"»

Тему затверджено наказом ректора Університету від « ____ » _____ 202_ р. № _____

Кваліфікаційна магістерська робота виконується на матеріалах АТ «Київгаз»

План кваліфікаційної магістерської роботи

Розділ I Дослідження та аналіз підходів до створення інформаційних управляючих систем сервісної підтримки ІТ інфраструктури

(назва розділу)

Розділ II _ Характеристика інформаційних управляючих систем та методи і моделі

(назва розділу)

Розділ III Розроблення проєктних рішень та їх реалізація

(назва розділу)

Об'єкт дослідження: процеси управління сервісної підтримки ІТ інфраструктури АТ "Київгаз"»

Предмет дослідження: методи та прийоми проєктування інформаційної управляючої системи сервісної підтримки ІТ інфраструктури для АТ "Київгаз»

Мета кваліфікаційної магістерської роботи: є підвищення ефективності процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи сервісної підтримки ІТ інфраструктури

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I Здійснити аналіз існуючих інформаційних управляючих систем сервісним обслуговуванням IT інфраструктур, дослідити IT-інфраструктуру «АТ Київгаз», обґрунтувати вибір підходів і технологій створення інформаційно-управляючих систем.

У розділі II Описати структуру і характеристику системи, побудувати інформаційну модель, розробити структуру системи, визначити методи і моделі і інформаційних системах управління сервісним обслуговуванням.

У розділі III Розробити проєктні рішення, зокрема: описати процес проєктування бази даних, сховища даних, визначити інформаційні масиви, визначити основні показники OLAP-аналізу. Описати програмне та технічне забезпечення. Розробити прототип інформаційної системи управління сервісної підтримки IT-інфраструктури

Завдання підготував
науковий керівник _____
(підпис)

Помазун О.М.
(ініціали, прізвище)

« ____ » _____ 2024 р.

Завдання одержав
здобувач _____
(підпис)

Дівесєв Ярослав Ігорович
(ініціали, прізвище)

« ____ » _____ 2024 р.

Реферат

Кваліфікаційна магістерська робота містить 85 сторінок, 11 таблиць, 15 рисунків, список використаних джерел з 44 найменувань, додатки.

«Інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз"»

Об'єктом дослідження кваліфікаційної магістерської роботи є процеси управління сервісною підтримкою ІТ інфраструктури АТ «Київгаз»

Предметом дослідження є методи та прийоми проєктування інформаційної управляючої системи сервісної підтримки ІТ інфраструктури для АТ "Київгаз" .

Мета і завдання дослідження. Основною метою кваліфікаційної магістерської роботи є підвищення ефективності процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи сервісної підтримки ІТ інфраструктури

Відповідно до поставленої мети визначені такі *завдання*:

- дослідити предметну область, виконати аналіз існуючих процесів сервісної підтримки ІТ-інфраструктури «АТ Київгаз»;
- представити обґрунтування вибору підходів і технологій для створення системи;
- визначити структуру та представити характеристику управляючої системи сервісної підтримки ІТ інфраструктури АТ "Київгаз";
- описати методи та моделі в системі сервісної підтримки;
- спроектувати базу даних для системи сервісної підтримки ІТ-інфраструктури;
- спроектувати засоби OLAP для управляючої системи сервісної підтримки ІТ-інфраструктури»;
- описати програмне забезпечення для управляючої системи ;
- описати технічне забезпечення для управляючої системи сервісної підтримки ІТ-інфраструктури;
- представити результати реалізації управляючої системи сервісної підтримки ІТ-інфраструктури

Теоретична, методична та практична значущість отриманих результатів. Під час дослідження було досліджено предметну область, проаналізовано існуюча ІТ-інфраструктура на АТ «Київгаз», визначені методи і моделі для підвищення ефективності управління сервісною підтримкою ІТ-інфраструктури, стпроектрована база даних, визначен показники еієктивності. Практичні результати дослідження полягають у вдосконаленні процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи.

Рік виконання кваліфікаційної магістерської роботи – 2024.

Рік захисту роботи – 2024

Ключові слова: ІТ-інфраструктура, сервісна підтримка, база даних, сховище даних, інформаційна система.

Відгук

про кваліфікаційну магістерську роботу
здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»

освітньо-професійної програми «Інформаційні управляючі системи і технології»

Дівеева Ярослава Ігоровича

(прізвище, ім'я, по батькові)

на тему «Інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз"»

1. Актуальність теми: Створення "Інформаційної управляючої системи сервісної підтримки ІТ інфраструктури АТ "Київгаз" може значно підвищити ефективність роботи компанії та якість обслуговування клієнтів. Вивантаження заявок з сервіс-деску та подальший аналіз цієї інформації можуть стати основою для інтелектуальних рішень.

2. Позитивні риси кваліфікаційної магістерської роботи: Цінними є дослідження процесів сервісної підтримки ІТ-інфраструктури в АТ «Київгаз», самостійно спроектована модель сховища даних, реалізовано фрагмент програми на мові програмування java.

3. Наявність самостійних розробок автора усі теоретичні та експериментальні дослідження автором виконані самостійно.

4. Цінність теоретичних висновків та практичних рекомендацій Теоретична, методична та практична значущість отриманих результатів. Під час дослідження було досліджено предметну область, проаналізовано існуюча ІТ-інфраструктура на АТ «Київгаз», визначені методи і моделі для підвищення ефективності управління сервісною підтримкою ІТ-інфраструктури, спроектована база даних, визначен показники ефективності. Практичні результати дослідження полягають у вдосконаленні процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи.

5. Наявність недоліків: в роботі є наявність стилістичних помилок.

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: робота виконана на високому професійному рівні і може бути допущена в екзаменаційній комісії з високою позитивною оцінкою.

Науковий керівник доцент кафедри, к.е.н., доцент Помазун О.М.

(посада, учене звання, науковий ступінь)

(підпис)

(прізвище, ініціали)

« ____ » _____ 20__ р.

Рецензія

на кваліфікаційну магістерську роботу
здобувача вищої освіти

Дівесва Ярослава Ігоровича

(прізвище, ім'я, по батькові)

Тема: «Інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз"»

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення Зростання обсягів ІТ-інфраструктури: Зі збільшенням кількості обладнання та програмного забезпечення, необхідність в ефективному управлінні ІТ-інфраструктурою стає важливішою. Інформаційна система дозволить централізовано керувати всіма аспектами інфраструктури, забезпечуючи своєчасне виявлення та вирішення проблем.

Якість проведеного дослідження : дослідження проведено якісно, усі теоретичні висновки мають преактивні пропозиції.

Позитивні риси кваліфікаційної магістерської роботи Цінними є дослідження процесів сервісної підтримки ІТ-інфраструктури в АТ «Київгаз», самостійно спроектована модель сховища даних, реалізовано фрагмент програми на мові програмування java.

Зауваження відсутні

Практична значимість висновків і рекомендацій Практичні результати дослідження полягають у вдосконаленні процесів управління ІТ інфраструктурою шляхом розроблення та використання інформаційної управляючої системи, які будуть враховані при впровадженні даної управляючої системи в АТ "Київгаз"

Місце роботи та посада рецензента АТ «Київгаз», начальник відділу Інформаційних технологій

Науковий ступінь, учене звання (за наявності) Клепко П.П.

(підпис, ПІБ)

Підпис засвідчую: _____

(посада, підпис)

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ сервісної підтримки ІТ інфраструктури	5
1.1 Аналіз існуючих інформаційних управляючих систем сервісної підтримки ІТ інфраструктури	5
1.1.1 Дослідження предметної області.....	5
1.1.2 Дослідження ринку інформаційних управляючих систем.....	9
1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи	13
РОЗДІЛ 2 ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ.....	20
2.1 Структура і характеристика системи	20
2.2 Методи та моделі в інформаційних управляючих системах і технологіях...	29
РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ	32
3.1 Проектування бази даних та сховища даних для інформаційної управляючої системи	32
3.2 Проектування засобів OLAP-аналізу даних	40
3.3. Програмне забезпечення.....	42
3.3.1 Вимоги до програмного забезпечення	42
3.3.2 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем.....	43
3.3.3 Архітектура програмного забезпечення	48
3.3.4. Тестування програмного забезпечення.....	48
3.4. Технічне забезпечення.....	48
3.5 Реалізація інформаційної управляючої системи.....	50
ВИСНОВКИ.....	54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	55
ДОДАТКИ.....	

ВСТУП

Актуальність проблеми.

ІТ інфраструктура – це комплексна структура, що об'єднує всі інформаційні технології та ресурси, які використовує конкретне підприємство. Правильно побудована ІТ інфраструктура дозволяє оптимізувати витрати підприємства та підвищити продуктивність встановлених ІТ сервісів і засобів

Актуальність розробки інформаційної управляючої системи сервісної підтримки ІТ інфраструктури для АТ "Київгаз" обумовлена кількома важливими аспектами, які відображають сучасні потреби підприємства в ефективному управлінні технологічними процесами та ресурсами: зростання обсягу ІТ-інфраструктури, підвищення вимог до безпеки, оптимізація витрат, покращення якості обслуговування, інтеграція з іншими системами.

Усі ці аспекти роблять інвестиції в розробку та впровадження інформаційної управляючої системи сервісної підтримки ІТ інфраструктури актуальними та виправданими з точки зору стратегічного розвитку і оперативної ефективності АТ "Київгаз".

Об'єктом дослідження кваліфікаційної магістерської роботи є процеси управління сервісною підтримкою ІТ інфраструктури АТ «Київгаз»

Предметом дослідження є методи та прийоми проектування інформаційної управляючої системи сервісної підтримки ІТ інфраструктури для АТ "Київгаз" .

Мета і завдання дослідження. Основною метою кваліфікаційної магістерської роботи є підвищення ефективності процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи сервісної підтримки ІТ інфраструктури

Відповідно до поставленої мети визначені такі завдання:

- дослідити предметну область, виконати аналіз існуючих процесів сервісної підтримки ІТ-інфраструктури «АТ Київгаз»;

- представити обґрунтування вибору підходів і технологій для створення системи;
- визначити структуру та представити характеристику управляючої системи сервісної підтримки ІТ інфраструктури АТ "Київгаз";
- описати методи та моделі в системі сервісної підтримки;
- спроектувати базу даних для системи сервісної підтримки ІТ-інфраструктури;
- спроектувати засоби OLAP для управляючої системи сервісної підтримки ІТ-інфраструктури»;
- описати програмне забезпечення для управляючої системи ;
- описати технічне забезпечення для управляючої системи сервісної підтримки ІТ-інфраструктури;
- представити результати реалізації управляючої системи сервісної підтримки ІТ-інфраструктури

Теоретична, методична та практична значущість отриманих результатів. Під час дослідження було досліджено предметну область, проаналізовано існуюча ІТ-інфраструктура на АТ «Київгаз», визначені методи і моделі для підвищення ефективності управління сервісною підтримкою ІТ-інфраструктури, спроектована база даних, визначен показники ефективності. Практичні результати дослідження полягають у вдосконаленні процесів управління ІТ інфраструктурою АТ "Київгаз" шляхом розроблення та використання інформаційної управляючої системи.

Структура роботи. Загальний обсяг роботи становить 85 сторінок друкованого тексту, 11 рисунків, 14 таблиць. 1 додаток. Список використаних джерел налічує 44 найменування.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ СЕРВІСНОЇ ПІДТРИМКИ ІТ ІНФРАСТРУКТУРИ

1.1 Аналіз існуючих інформаційних управляючих систем сервісної підтримки ІТ інфраструктури

1.1.1 Дослідження предметної області

ІТ інфраструктура – це комплексна структура, що об’єднує всі інформаційні технології та ресурси, які використовує конкретне підприємство. До її складу входять всі комп’ютери, встановлене програмне забезпечення, системи зв’язку, інформаційні центри, мережі та бази даних. Правильно побудована ІТ інфраструктура дозволяє оптимізувати витрати підприємства та підвищити продуктивність встановлених ІТ сервісів і засобів [1].

Компоненти ІТ-інфраструктури розглядають за трьома групами:

- апаратне забезпечення, яке включає фізичну частину інфраструктури;
- програмне забезпечення, яке забезпечує роботу апаратної частини та управління нею.
- мережева інфраструктура, яка забезпечує внутрішній та зовнішній зв’язок між пристроями та елементами системи.

Таким чином, до ІТ-інфраструктури відносять:

- сервери;
- персональні комп’ютери;
- маршрутизатори;
- встановлена операційна система;
- датчики моніторингу стану обладнання;
- елементи живлення;

– програмне забезпечення для профілактичного обслуговування.

ІТ-інфраструктура постійно розвивається, адаптуючись до нових технологій та потреб бізнесу. Хмарні обчислення, штучний інтелект, Інтернет речей - це лише деякі з трендів, які впливають на те, як будується та використовується ІТ-інфраструктура сьогодні.

Об'єктом дослідження даної роботи є процеси управління ІТ-інфраструктурою. АТ "Київгаз" забезпечує надання різноманітних послуг, пов'язаних із газопостачанням, включаючи встановлення та обслуговування лічильників, ремонт газового обладнання, приєднання до газових мереж та консультаційні послуги через контакт-центр (Kyivgaz) (Вікіпедія).

АТ "Київгаз". ІТ-інфраструктура АТ "Київгаз" забезпечена надійною та ефективною мережею для забезпечення безперебійної роботи і високої продуктивності.

В структурі АТ "Київгаз" – 28 філій, поєднаних між собою в одну ІТ-інфраструктуру.

Всередині компанії "Київгаз" ІТ-інфраструктура побудована для підтримки ефективного управління, забезпечення безпеки та оперативного обслуговування клієнтів. Основні компоненти ІТ-інфраструктури всередині компанії включають корпоративні системи:

ERP-система – система управління ресурсами підприємства, яка інтегрує всі основні процеси компанії, включаючи фінанси, постачання, управління проектами та інші бізнес-процеси. Це дозволяє оптимізувати управління ресурсами та покращити ефективність роботи.

CRM-система – система управління взаємодією з клієнтами, що дозволяє збирати, зберігати та аналізувати інформацію про клієнтів, їх запити та історію взаємодії. Це сприяє підвищенню якості обслуговування та задоволеності клієнтів.

Системи моніторингу та управління газовими мережами включає наступні системи:

SCADA-система – система диспетчерського управління та збору даних, яка забезпечує моніторинг і контроль за станом газопроводів та газорегуляторних

пунктів в режимі реального часу. Це дозволяє оперативно реагувати на аварійні ситуації та підтримувати безперебійне газопостачання АТ «Київгаз».

В ІТ-інфраструктурі діє система захисту від кібератак, яка включає антивірусні програми, брандмауери, системи виявлення та запобігання вторгненням (IDS/IPS), а також регулярні аудити безпеки. Це забезпечує захист інформаційних систем від несанкціонованого доступу та кібератак.

Одна з ключових складових цієї інфраструктури — це мережа з'єднань між різними підрозділами компанії, з'єднаною маршрутизаторами Cisco, розташованими в різних філіях, та центральною мережею оператора MPLS (Multi-Protocol Label Switching). На рисунку 1.1 показана схема логічних з'єднань АТ «Київгаз».

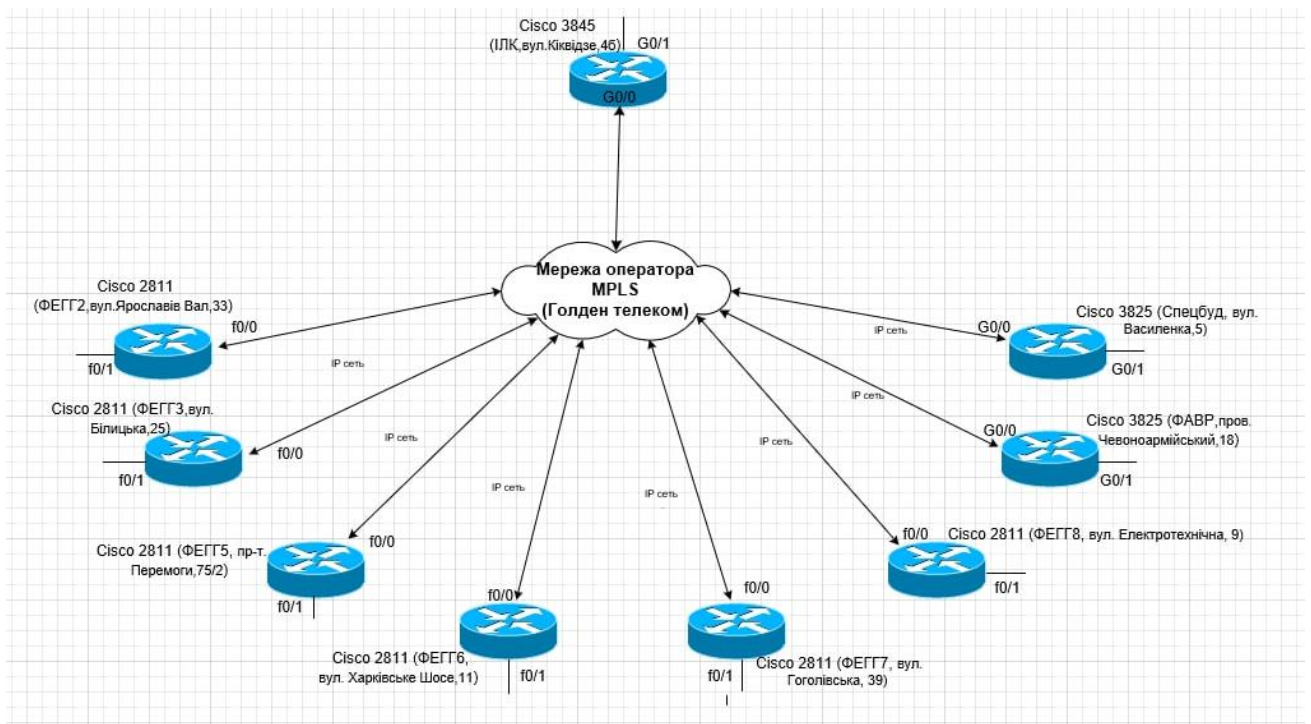


Рисунок 1.1 – Схема логічних з'єднань АТ «Київгаз», з'єднаних маршрутизаторами Cisco.

Джерело: розроблено автором самостійно на базі матеріалів АТ «Київгаз»

Американська компанія Cisco є найбільшим у світі виробником мережевого обладнання, призначеного для обслуговування мереж віддаленого доступу, сервісів безпеки, мереж зберігання даних, маршрутизації та комутації, а також для

потреб комерційного ринку IP-комунікацій і корпоративного ринку. На рисунку 1.2 показана детальна схема з'єднань в АТ «Київгаз».

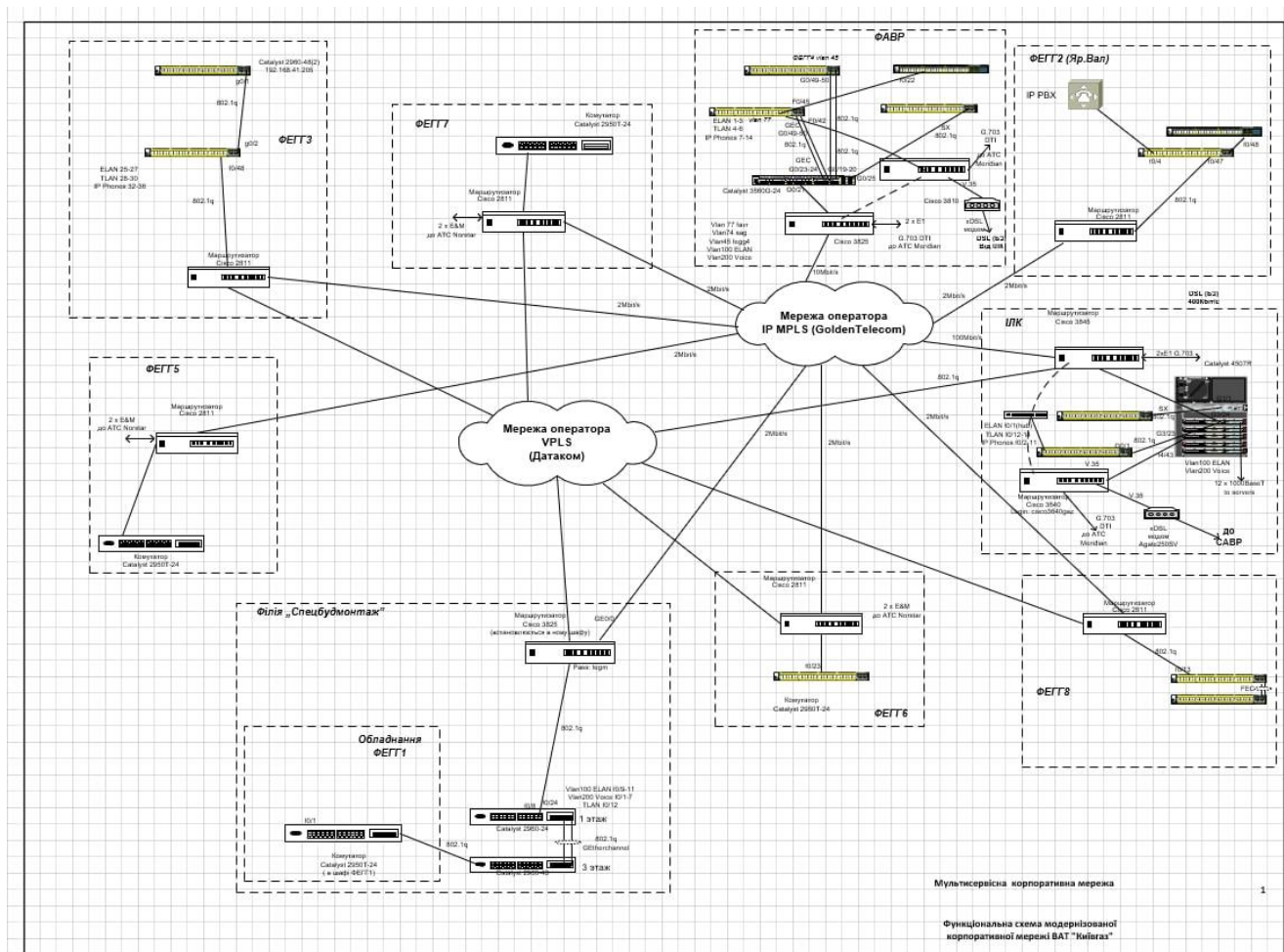


Рисунок 1.2 – Схема ІТ-інфраструктури АТ «Київгаз», з'єднаних маршрутизаторами Cisco.

Джерело: розроблено автором самостійно на базі матеріалів АТ «Київгаз»

Не дивлячись на те, що дана модель має чітко налагоджену ІТ-інфраструктуру, існує ряд задач, які потребують інтелектуальну підтримку, тому виникає потреба у створенні інформаційної управляючої системи сервісної підтримки ІТ інфраструктури АТ "Київгаз".

АТ "Київгаз" активно впроваджує сучасні технології для покращення своєї ІТ-інфраструктури. Одним із ключових проєктів є запуск оновленого особистого кабінету для споживачів, який дозволяє управляти кількома особовими рахунками, передавати показання лічильників, укладати договори розподілу газу,

завантажувати рахунки та оплачувати послуги онлайн. Це значно спрощує процес взаємодії з компанією для клієнтів та підвищує ефективність обслуговування.

Компанія також стикається з викликами у сфері кібербезпеки. На початку 2024 року "Київгаз" став мішенню для хакерів з "Russian Cyber Army Team", що підкреслює важливість захисту ІТ-інфраструктури від кіберзагроз.

Загалом, "Київгаз" продовжує розвиватися, впроваджуючи новітні технології та покращуючи якість своїх послуг для мешканців Києва.

1.1.2 Дослідження ринку інформаційних управляючих систем

Сервісна підтримки між філіями АТ «Київгаз» здійснюється через спеціальне програне забезпечення, призначене для ІТ-підтримки користувачів – систему ServiceDesk. На рисунку 1.3 показано інтерфейс існуючої системи.

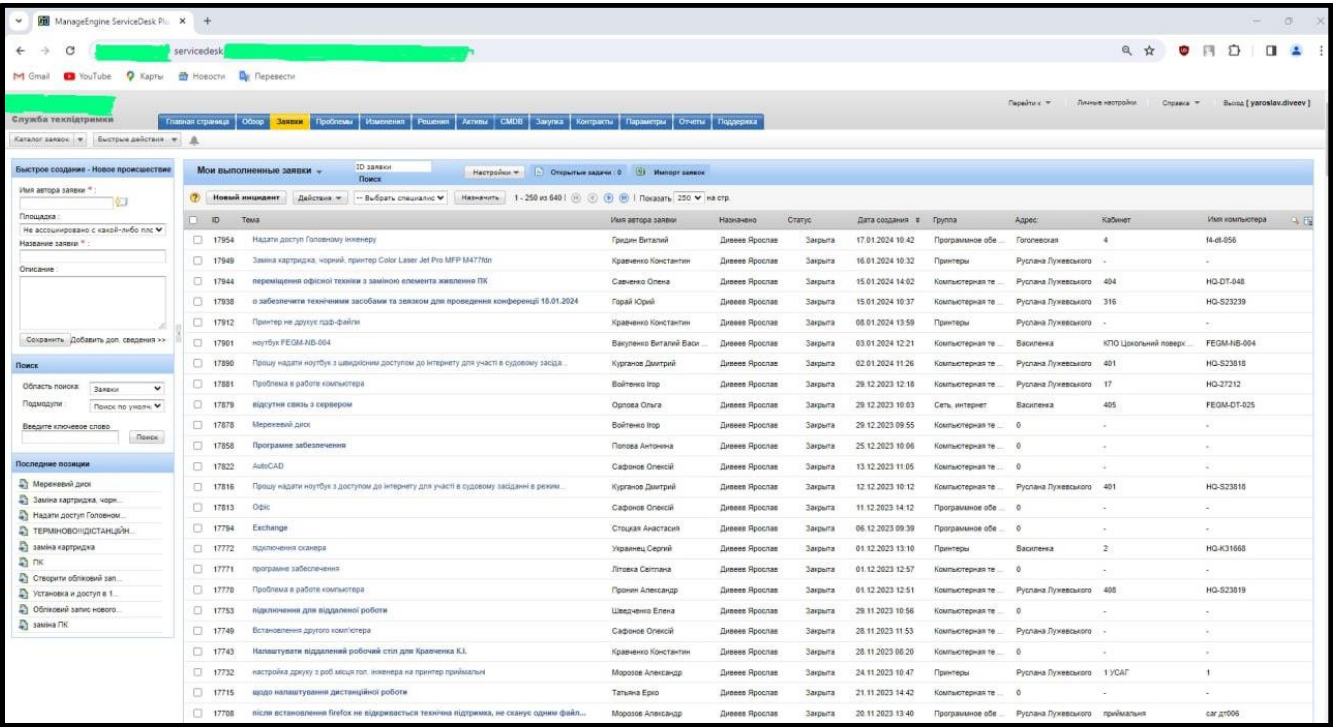


Рисунок 1.3 – Екранна форма ServiceDesk в АТ «Київгаз»

Джерело: розроблено автором самостійно на базі матеріалів АТ «Київгаз»

Сервіс-деск – це система, яка централізує всі запити на ІТ-підтримку, забезпечуючи чіткий та структурований підхід до їх обробки. Замість того, щоб надсилати листи на різні адреси або телефонувати в різні відділи, користувачі можуть створити тикет у системі сервіс деск, де його буде негайно взято на облік.

Таким чином, Сервіс-деск – програмне забезпечення для служби підтримки – це інструмент, який використовується для організації, управління та відповіді на запити щодо обслуговування з внутрішніх та зовнішніх джерел. Зазвичай запити від клієнтів надходять через різні канали, включаючи електронну пошту, телефон або соціальні мережі. Команди обслуговування клієнтів використовують програмне забезпечення для служби підтримки, щоб оптимізувати процеси підтримки та отримувати аналітику щодо взаємодії з клієнтами через всі канали зв'язку.

Основні переваги сервісдеск:

- покращена комунікація – сервісдеск забезпечує чіткий канал зв'язку між користувачами та ІТ-фахівцями, гарантуючи, що всі запити будуть зафіксовані та оброблені;
- підвищення ефективності – завдяки централізованому збору запитів та автоматизації робочих процесів сервіс деск значно економить час та ресурси ІТ-відділу;
- швидке вирішення проблем – система пріоритезації запитів у сервіс деск гарантує, що найкритичніші проблеми будуть вирішені першими, мінімізуючи час простою.
- підвищення задоволеності користувачів – зрозуміла система обробки запитів, чітка комунікація та швидке вирішення проблем ведуть до значного підвищення задоволеності користувачів ІТ-послугами.

До основних функцій сервіс-деск відносять:

- створення та реєстрація тикетів, користувачі можуть легко створювати тикети, описуючи свою проблему та додаючи необхідні файли;

- класифікація та пріоритезація тикетів, запити можна класифікувати за типом проблеми, терміновістю та іншими критеріями, щоб гарантувати, що найважливіші з них будуть вирішені першими;
- призначення тикетів, тикети можна призначати конкретним ІТ-фахівцям, які володіють необхідними знаннями та навичками для їх вирішення;
- відстеження статусу тикетів, користувачі можуть відстежувати статус своїх тикетів у будь-який час, отримуючи повідомлення про кожну зміну;
- наявність бази знань, система може містити базу знань з типовими проблемами та їх рішеннями, що дозволяє користувачам самостійно знаходити відповіді на свої запитання;
- звітність та аналітика, сервіс деск може генерувати звіти про кількість та тип запитів, час їх вирішення та інші показники, що допомагає ІТ-фахівцям аналізувати свою роботу та виявляти сфери для покращення.

Програми сервісдеск можуть містити додаткові інтеграції для самостійного обслуговування клієнтів, соціальних мереж або модулі для живого чату.

Щоб продукт був включений у категорію програмного забезпечення для служби підтримки, він повинен відповідати таким критеріям:

- організовувати зовнішні запити клієнтів у вигляді квитків для агентів підтримки;
- збирати запити з електронних листів та порталу клієнтів;
- призначати квитки агентам підтримки для швидкого обслуговування.

Існує два основних типи сервіс деск:

- локальний сервіс деск – встановлюється на серверах організації;
- хмарний сервіс деск – розміщується на віддалених серверах провайдера послуг.

На рисунку 1.4 показаний процес звернення в сервісдеск в АТ «Кивгаз»

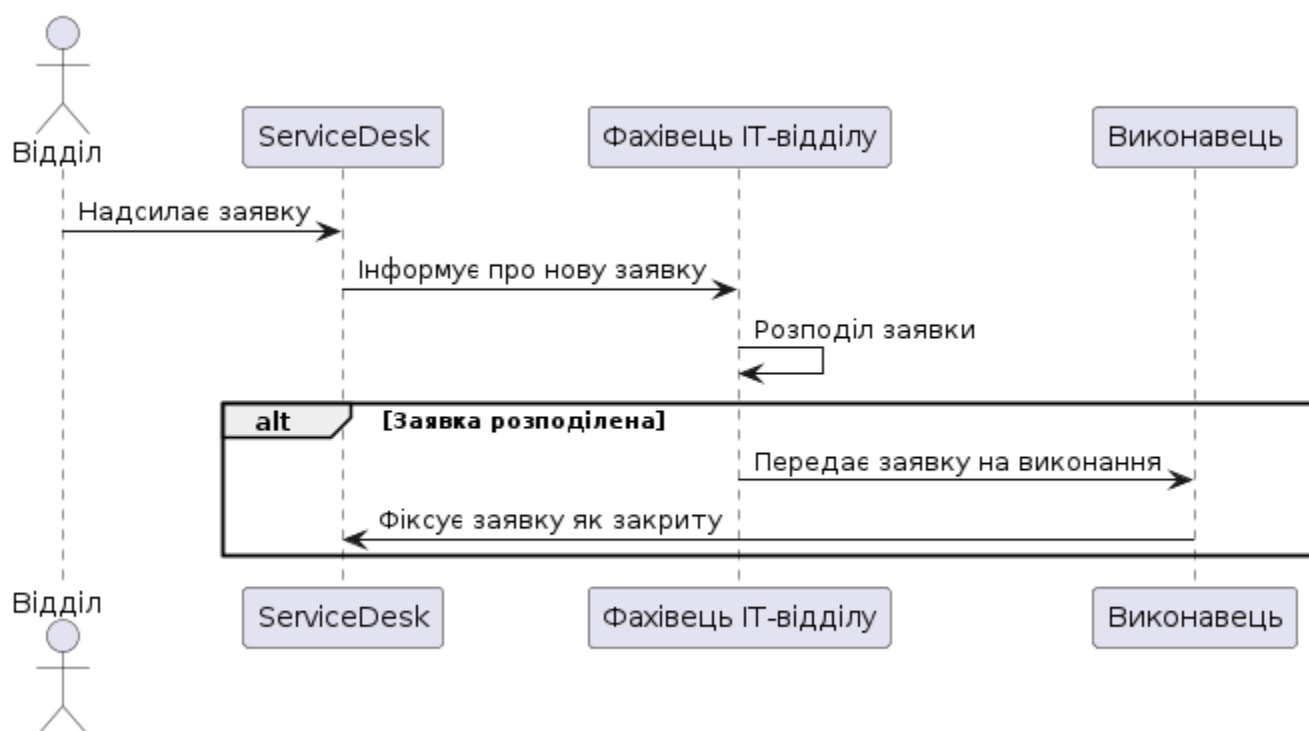


Рисунок 1.4 – Процес звернення до ServiceDesk в АТ «Київгаз»

Джерело: розроблено автором самостійно

Коли у відділі компанії виникла технічна проблема, вони звертаються до служби підтримки .ServiceDesk, яка є центром обробки всіх запитів на підтримку.

Коли заявка надходить до ServiceDesk, система автоматично інформує про це фахівця IT-відділу. Фахівець IT-відділу отримує повідомлення про нову заявку і починає її обробляти. Він переглядає деталі запиту та визначає, як найкраще розподілити завдання для швидкого і ефективного вирішення проблеми.

Якщо фахівець вирішує, що заявку потрібно передати іншому спеціалісту для виконання, він перенаправляє її до відповідного виконавця. Виконавець отримує завдання і приступає до його виконання. Після завершення роботи виконавець фіксує у системі ServiceDesk, що заявка закрыта і проблема вирішена.

Таким чином, відділ, який надіслав заявку, отримує необхідну допомогу, а ServiceDesk фіксує завершення роботи, забезпечуючи прозорість і контроль над процесом обробки заявок. Цей процес дозволяє забезпечити своєчасне вирішення проблем і підтримувати високий рівень обслуговування в компанії.

1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи

Управління ІТ-інфраструктурою відноситься до категорії слабоструктурованих задач, що є складними і потребують спеціальної підтримки. Використання спеціалізованих програмних засобів при управлінні є важливою частиною цього процесу.

Основна мета створення інформаційної системи (ІС) полягає в підвищенні якості управлінських рішень, моніторингу якості послуг та виявленні прихованих знань про проблеми.

Аналізуючи досвід створення подібних систем, можна виділити наступні етапи проектування ІС:

1. Дослідження проблемної області та формування вимог до системи.
2. Вибір технології проектування системи.
3. Визначення функціональних можливостей ІС.
4. Проектування компонентів ІС.

Детальніше розглянемо ці етапи:

1. Дослідження проблемної області та формування вимог до системи: На цьому етапі визначаються основні труднощі та специфічні аспекти роботи системи.

До основних проблем відносяться:

- неможливість прогнозування процесу;
- складнощі з обробкою необхідних даних;
- проблеми з аналізом і формуванням логічних висновків;
- труднощі з візуалізацією результатів;
- неточність евристичних суджень.

2. Вибір технології проектування системи: вибір технології проектування залежить від типу ІС. Технологія проектування інформаційної системи включає

методологічні підходи, засоби проектування та засоби організації проектування, що охоплюють управління процесом створення та модернізації ІС.

Основою технології проектування є технологічний процес, що визначає дії, їх послідовність, склад виконавців, а також засоби і ресурси, необхідні для виконання цих дій. Процес проектування інформаційної системи розбивається на низку послідовно-паралельних, взаємопов'язаних та супідрядних дій, кожна з яких може мати свій предмет. Дії при проектуванні системи можуть бути визначені як неподільні технологічні операції або як підпроцеси цих операцій.

При виборі конкретної технології для створення ІС, слід керуватися такими вимогами:

- проект, створений за допомогою цієї технології, повинен відповідати вимогам замовника;
- вибрана технологія має максимально відображати весь життєвий цикл проекту;
- вибрана технологія повинна забезпечувати мінімальні трудові та фінансові витрати на проектування і супровід проекту;
- технологія повинна служити основою для зв'язку між проектуванням і супроводом проекту;
- технологія повинна сприяти підвищенню продуктивності праці проектувальника;
- технологія повинна гарантувати надійність процесу проектування та експлуатації проекту;
- технологія повинна забезпечувати легкість ведення проектної документації.

Існує два основні класи технологій створення інформаційних систем (ІС): канонічна та індустріальна технології. Індустріальна технологія проектування ділиться на два підкласи: автоматизоване проектування з використанням CASE-технологій та типове проектування, яке може бути параметрично-орієнтованим або

модельно-орієнтованим. Використання індустріальних технологій не виключає застосування канонічної технології.

Канонічне проектування ІС відображає особливості ручного індивідуального проектування, яке здійснюється без застосування інструментальних засобів для інтеграції елементарних операцій. Цей підхід зазвичай використовується для невеликих локальних систем. Основою канонічного проектування є каскадна модель життєвого циклу ІС. Цей підхід також відомий як ресурсно-функціональний, оскільки ґрунтується на ресурсно-функціональній декомпозиції діяльності економічного об'єкта, для якого створюється система.

Індустріальне автоматизоване проектування ІС включає кілька підходів:

- структурно-орієнтований: зосереджується на структурі системи;
- об'єктно-орієнтований базується на об'єктній моделі, де система складається з об'єктів, що взаємодіють;
- процесно-орієнтований акцентує увагу на процесах, які відбуваються в системі;
- агентно-орієнтований: використовує агенти, які виконують автономні дії та взаємодіють між собою.

Кожен з цих підходів має свої переваги і використовується в залежності від конкретних вимог проекту та умов його реалізації.

Структурно-орієнтований підхід базується на декомпозиції економічного об'єкта, для якого розробляється інформаційна система, шляхом структурного аналізу. Цей метод дослідження починається з загального огляду системи і деталізується до ієрархічної структури з декількома рівнями.

Для структурного аналізу використовуються три основні групи засобів:

- функції: визначають, які завдання повинна виконувати система.
- відношення між даними ілюструють, як дані взаємодіють між собою.
- поведінка системи в часі відображають аспекти реального часу.

Найбільш поширеними графічними нотаціями для цих задач є:

- DFD (Data Flow Diagrams): діаграми потоків даних, які включають словники даних і специфікації процесів.
- ERD (Entity-Relationship Diagrams): діаграми "суть-зв'язок", що показують зв'язки між сутностями.
- STD (State Transition Diagrams): діаграми переходів станів, які ілюструють зміну станів системи залежно від подій.

Об'єктно-орієнтована методологія зосереджується на об'єктній декомпозиції предметної області, представляючи її як сукупність об'єктів, що взаємодіють між собою через передачу повідомлень. Цей підхід не протиставляється структурному, а навпаки, інтегрує деякі його елементи, такі як DFD, ERD і STD, для моделювання структури і поведінки об'єктів.

Процесно-орієнтований підхід базується на новітніх тенденціях у теорії менеджменту економічними об'єктами, зокрема на реінжинірингу бізнес-процесів. Бізнес-процеси включають ділові, адміністративні та технологічні процедури підприємства, такі як документообіг, управління фінансовими та матеріальними потоками, персоналом, організаційно-господарськими та технологічними процесами, а також процесами проектування виробів. Реінжиніринг бізнес-процесів полягає у кардинальному якісному переосмисленні та кількісному перепроєктуванні всіх бізнес-процесів конкретного економічного об'єкта з метою досягнення його стратегічних цілей. Основою методології реінжинірингу є системний та структурний аналіз, теорія управління великими системами, а також методи управління якістю та промислова інженерія.

Агентно-орієнтований підхід до створення інформаційних систем передбачає проектування та реалізацію систем як сукупності програмних агентів або як мультиагентну систему.

Типова (прототипна) технологія або технологія швидкої розробки додатків (RAD-технологія) базується на спіральній моделі життєвого циклу ІС. Відповідно до цієї технології, ІС розробляється шляхом поетапного розширення програмних

прототипів, повторюючи цикл від деталізації вимог до деталізації програмного коду.

3. Визначення функціональних можливостей ІС.

На цьому етапі визначаються функціональні можливості інформаційної системи (ІС). Пропонується порядок роботи користувача з системою, а також описується розподіл дій між персоналом та технічними засобами у різних ситуаціях вирішення задач. Описуються основні функції ІС, що включають:

- інтерфейс користувача;
- механізми обробки даних;
- функції управління та моніторингу;
- інтеграційні можливості з іншими системами;
- забезпечення безпеки та конфіденційності даних.

4. Проектування компонентів ІС. На цьому етапі проектуються основні компоненти ІС відповідно до обраних методів проектування:

- база даних: структура бази даних, типи даних, зв'язки між таблицями та індекси;
- база моделей: набір моделей, що використовуються для аналізу та прийняття рішень;
- інтерфейс користувача: проектування користувацького інтерфейсу з урахуванням механізмів взаємодії (формальний діалог, природна мова

5. Визначення функціональних можливостей ІС

На цьому етапі визначаються функціональні можливості інформаційної системи (ІС). Пропонується порядок роботи користувача з системою та розподіл дій між персоналом і технічними засобами при різних ситуаціях вирішення завдань. Описуються основні функції ІС, що включають:

- інтерфейс користувача: проектування зручного та інтуїтивно зрозумілого інтерфейсу;
- обробка даних: визначення методів і алгоритмів для обробки вхідних даних;

- управління та моніторинг: функції, що дозволяють відслідковувати та контролювати роботу системи;
- інтеграція: можливості взаємодії з іншими системами та базами даних;
- безпека: механізми захисту даних та забезпечення конфіденційності.

Розглянемо процес створення ІС для АТ "Київгаз".

1. Аналіз проблемної області. АТ "Київгаз" є великою компанією, яка обслуговує численні технічні системи та мережі, забезпечуючи постачання газу. Сервісна підтримка ІТ-інфраструктури такої компанії вимагає високого рівня автоматизації для забезпечення ефективності, швидкого реагування на інциденти та підтримки стабільної роботи всіх систем.

2. Визначення мети системи. Метою створення інформаційної управляючої системи є підвищення якості управління ІТ-послугами та підтримки користувачів, зниження часу простою систем і підвищення загальної продуктивності.

3. Вибір підходу до створення системи. Для реалізації цієї системи доцільно застосувати комбінований підхід, що включає елементи структурно-орієнтованого, процесно-орієнтованого та об'єктно-орієнтованого підходів.

Структурно-орієнтований підхід:

- використовується для декомпозиції ІТ-інфраструктури компанії, визначення ключових компонентів і зв'язків між ними;
- застосовуються діаграми потоків даних (DFD), діаграми сутність-зв'язок (ERD) та діаграми переходів станів (STD).

Процесно-орієнтований підхід:

- базується на реінжинірингу бізнес-процесів, що включає аналіз та оптимізацію всіх бізнес-процесів, пов'язаних із сервісною підтримкою ІТ;
- використовуються моделі бізнес-процесів для детального опису всіх етапів обслуговування та управління інцидентами.

Об'єктно-орієнтований підхід реалізується через об'єктну декомпозицію системи, де визначаються основні об'єкти (наприклад, заявки на підтримку, користувачі, технічне обладнання) і їх взаємодія.

Цей підхід забезпечує модульність системи та полегшує її масштабування і підтримку.

4. Визначення функціональних можливостей

На цьому етапі розробляється докладний опис функціональних можливостей системи:

- завантаження даних до сховища даних;
- автоматизований підхід до створення аналітичних звітів;
- моніторинг і управління станом технічного обладнання та програмного забезпечення;
- інтеграція з існуючими системами для забезпечення безперервного потоку даних.
- візуалізація та аналітика для відстеження ефективності роботи IT-підтримки.

5. Проектування компонентів.

Сховище даних: розробка структури сховища даних для зберігання інформації про заявки, обладнання, користувачів та інші необхідні дані.

Інтерфейс користувача: проектування зручного та інтуїтивно зрозумілого інтерфейсу для користувачів і технічного персоналу. Визначення механізмів взаємодії користувачів із системою (формальний діалог, графічний інтерфейс).

База моделей: включення моделей для аналізу та прийняття рішень, що дозволять автоматизувати процеси управління заявками та технічним обслуговуванням. Пропонується використання аналітичних моделей для аналізу сховища даних.

На основі аналізу предметної області, використання структурно-орієнтованого, процесно-орієнтованого та об'єктно-орієнтованого підходів, а також сучасних технологій, розробка інформаційної управляючої системи для АТ "Київгаз" забезпечить ефективну підтримку IT-інфраструктури, підвищить продуктивність і якість обслуговування користувачів.

РОЗДІЛ 2 ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ

2.1 Структура і характеристика системи

Інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз" призначена для автоматизації та оптимізації процесів управління технічною підтримкою та обслуговуванням інформаційно-технічної інфраструктури компанії. Її основні завдання включають в себе моніторинг стану обладнання, виявлення та вирішення проблем, планування та виконання регулярних обслуговувань, а також забезпечення оперативного реагування на виниклі аварійні ситуації. Ця система спрямована на підвищення ефективності роботи ІТ-інфраструктури, зменшення часу простою обладнання, а також покращення рівня обслуговування користувачів.

Техніко-економічна сутність інформаційної управляючої системи сервісної підтримки ІТ інфраструктури АТ "Київгаз" полягає в поєднанні технологічних можливостей системи з економічними вигодами, що вона приносить. Основні аспекти техніко-економічної сутності цієї системи включають:

- Зменшення витрат часу та ресурсів. система дозволяє автоматизувати багато рутинних процесів, які раніше вимагали великої кількості людських ресурсів та часу. Це сприяє оптимізації робочих процесів та зменшенню витрат компанії на обслуговування інфраструктури.
- Підвищення продуктивності. завдяки системі швидкого виявлення та вирішення проблем, час простою обладнання значно скорочується, що сприяє підвищенню загальної продуктивності ІТ-інфраструктури та сприяє збільшенню ефективності бізнес-процесів.

- Мінімізація ризиків. Система дозволяє оперативно реагувати на виникнення проблем та аварійних ситуацій, що допомагає зменшити ризики втрат даних, простоїв системи та негативного впливу на бізнес-процеси компанії.

- Оптимізація витрат. Впровадження системи дозволяє планувати та проводити обслуговування обладнання за оптимальними сценаріями, що дозволяє ефективно розподіляти ресурси та мінімізувати витрати на підтримку інфраструктури.

Отже, інформаційна управляюча система сервісної підтримки ІТ інфраструктури АТ "Київгаз" забезпечує економічну ефективність за рахунок оптимізації робочих процесів, підвищення продуктивності та мінімізації ризиків для бізнесу.

Потреба у використанні інформаційної системи управління сервісною підтримкою ІТ інфраструктури, може бути обґрунтована наступним чином:

- Обробка великого обсягу даних. ІТ інфраструктура підприємства, особливо такого великого, як АТ "Київгаз", містить величезний обсяг даних щодо обладнання, мережі, програмного забезпечення тощо. Використання ЕОМ дозволяє ефективно обробляти, зберігати та аналізувати ці дані.

- Швидкість обробки. ІС дозволить обробляти дані швидше, ніж люди. Швидкість обробки даних важлива для оперативного виявлення та вирішення проблем в ІТ інфраструктурі.

- Точність інформації. Використання ІС дозволяє зберігати та обробляти інформацію без помилок, що допомагає уникнути неправильних рішень та негативного впливу на бізнес-процеси.

- Автоматизація процесів. ІС дозволяють автоматизувати багато рутинних процесів управління сервісною підтримкою ІТ інфраструктури, таких як моніторинг стану обладнання, виявлення проблем, планування обслуговування тощо. Це допомагає зменшити навантаження на персонал та збільшити ефективність роботи.

Забезпечення доступу до інформації. ІС дозволяють забезпечити доступ до необхідної інформації шляхом впровадження спеціалізованих систем управління, які надають можливість швидкого та зручного доступу до потрібних даних для прийняття управлінських рішень.

Отже, використання інформаційної системи для розв'язання завдань управління сервісною підтримкою ІТ інфраструктури дозволяє оптимізувати процеси, підвищити продуктивність та ефективність, а також забезпечити точність та доступність інформації для прийняття управлінських рішень.

Управління сервісною підтримкою ІТ інфраструктури АТ "Київгаз" передбачає розв'язання задач управління різними об'єктами та процесами. Основні об'єкти:

- обладнання – включає в себе сервери, мережеве обладнання, робочі станції, принтери та інше обладнання, яке використовується в ІТ інфраструктурі;
- мережа – управління мережевою інфраструктурою, включаючи маршрутизатори, комутатори, фایрволи, бездротові точки доступу тощо;
- системне програмне забезпечення – включає операційні системи, бази даних, програмне забезпечення для підтримки бізнес-процесів, а також програмне забезпечення для забезпечення безпеки та моніторингу;
- дані – управління даними, включаючи їх зберігання, захист, резервне копіювання та відновлення;
- безпека – управління заходами забезпечення безпеки ІТ інфраструктури, включаючи аутентифікацію, авторизацію, шифрування, моніторинг та виявлення загроз;
- моніторинг та аналіз – правління системами моніторингу та аналізу, які забезпечують постійний контроль за станом інфраструктури та виявлення відхилень в її роботі;
- зв'язок з користувачами – управління системами зв'язку з користувачами, такими як системи заявок та тікетів, відділи технічної підтримки тощо.

Призначення та використання вихідної інформації в контексті управління сервісною підтримкою ІТ інфраструктури має важливе значення для ефективного функціонування системи.

Призначення вихідної інформації наступне:

- моніторинг та звітність: інформація використовується для постійного моніторингу стану системи та для створення звітів про продуктивність, доступність та інші показники;
- планування обслуговування та ремонту: використання вихідної інформації дозволяє планувати регулярне обслуговування, запобігати виникненню проблем та ефективно використовувати ресурси для ремонту та обслуговування;
- прийняття рішень інформація використовується для прийняття рішень щодо майбутнього розвитку та вдосконалення інфраструктури, вибору нових технологій та стратегій управління.

Таким чином, вихідна інформація відіграє ключову роль у функціонуванні системи управління сервісною підтримкою ІТ інфраструктури, допомагаючи забезпечувати ефективність, доступність та надійність інфраструктури та виявляти та усувати проблеми.

Періодичність розв'язання задач управління сервісною підтримкою ІТ інфраструктури може варіюватися в залежності від потреб компанії та характеру її діяльності. Проте, основні процеси розв'язання задач можуть бути розподілені так:

- Моніторинг та аналіз: цей процес може бути виконуваний неперервно або з певною періодичністю, наприклад, кожну годину або кожен день, в залежності від важливості та критичності системи.
- Планування та прийняття рішень: цей процес зазвичай відбувається на регулярних засіданнях або визначених інтервалах, наприклад, раз на тиждень або раз на місяць, залежно від потреб компанії.
- Виконання обслуговування та ремонту: цей процес також може бути регулярним, зазначений у планах обслуговування та ремонту, які зазвичай розробляються заздалегідь на певний період, наприклад, на місяць або на квартал.

Щодо терміну видачі вихідної інформації з сервіс-деску, якщо прийнято рішення про постачання цієї інформації раз у місяць, то зазвичай термін видачі вихідної інформації може бути приблизно 1-3 робочих днів після завершення місяця. Це дозволить збирати, обробляти та аналізувати всю необхідну інформацію перед її видачею.

Автоматизоване розв'язання задач управління сервісною підтримкою ІТ інфраструктури може припинитися в різних ситуаціях, в залежності від обставин і умов. Ось деякі з найбільш типових умов, за яких може статися припинення автоматизованого розв'язання задач:

Технічні проблеми: якщо виникають серйозні технічні проблеми з обладнанням або програмним забезпеченням, які перешкоджають правильному функціонуванню автоматизованих систем, це може призвести до припинення автоматизованого розв'язання задач.

Бізнес-процеси та стратегічні зміни: якщо в компанії змінюються бізнес-процеси або стратегічні напрямки, це може вимагати перегляду або зміни системи управління, що може призвести до припинення автоматизованого розв'язання задач.

Непередбачені обставини: іноді можуть виникати непередбачені обставини, такі як катастрофи, кібератаки або інші події, які можуть вплинути на можливість автоматизованого розв'язання задач.

Недостатність ресурсів: якщо компанія не має достатньо ресурсів для підтримки або розвитку автоматизованих систем, це може призвести до їх припинення.

Потреба у переоцінці стратегії: іноді компанія може виявити, що поточна стратегія автоматизації не ефективна або не відповідає її потребам, що може призвести до перегляду або припинення автоматизованих систем.

У будь-якому випадку припинення автоматизованого розв'язання задач потребує уважного аналізу, планування та можливої переоцінки стратегій, щоб забезпечити ефективне управління сервісною підтримкою ІТ інфраструктури в майбутньому.

Розподіл дій між персоналом і технічними засобами у процесі розв'язання задач управління сервісною підтримкою ІТ інфраструктури може змінюватися в залежності від конкретної ситуації та характеру проблеми. У кожній ситуації персонал і технічні засоби співпрацюють для ефективного та швидкого вирішення проблем та підтримки ІТ інфраструктури в робочому стані. Розподіл дій зазвичай залежить від складності проблеми, її критичності та доступності необхідних ресурсів.

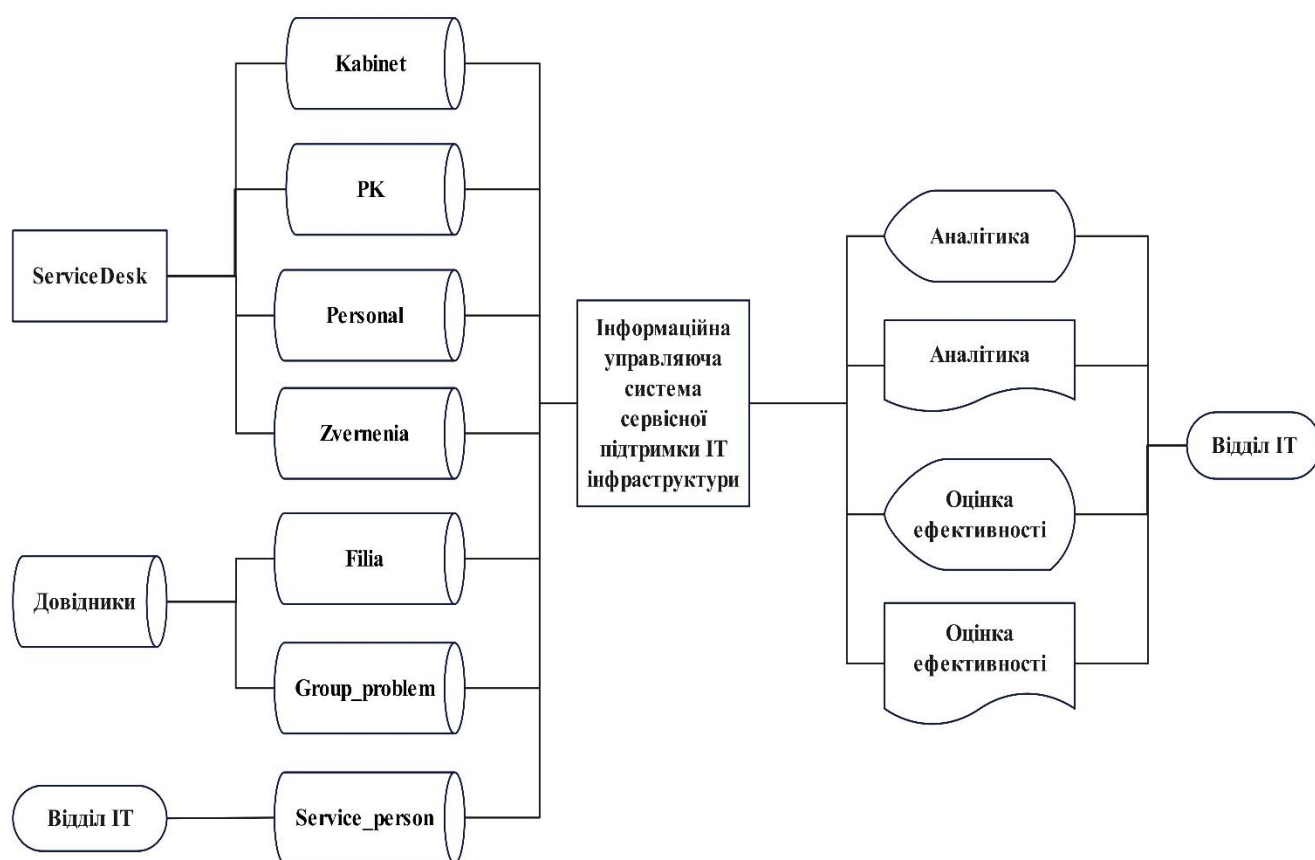


Рисунок 2.1 – Інформаційна модель задачі

Джерело: розроблено автором самостійно

В таблиці 2.1 показаний перелік і опис вихідних повідомлень. Вихідними повідомленням є аналітичні звіти за параметрами користувача. Також в системі можна отримати показники ефективності за параметрами: період, відділ, тип проблеми. Користувач може отримати звіти на екран або роздрукувати на принтер.

Таблиця 2.1 – Перелік і опис вихідних повідомлень

№	Назва вихідного повідомлення	Ідентифікатор	Форма подання і вимоги до неї	Періодичність видання	Термін видання і допустимий час затримки	Користувачі інформації
1	Аналітичні звіти за параметрами корстувача	Аналітика	Відеокадр	За вимогою	П'ятниця, 18:00	Менеджери складу, лікарі
2	Аналітичні звіти за параметрами корстувача	Аналітика	Паперовий документ	Щодня	07:00 ранку	Хірурги, асистенти
3	Оцінка ефективності за параметрами корстувача	Оцінка ефективності	Відеокадр	Місячно	1 число місяця	Фінансовий відділ
4	Оцінка ефективності за параметрами корстувача	Оцінка ефективності	Паперовий документ	Постійно	В реальному часі	Менеджери замовлень

Джерело: розроблено автором самостійно

Вхідними повідомленнями є таблиця сховища даних, показані у таблиці 2.2

Таблиця 2.2 – Перелік і опис вхідних повідомлень

з/п	Назва вхідного повідомлення	Ідентифікатор	Форма подання	Термін і частота надходження	Джерело
1	Довідник кабінетів	Kabinet	масив	Постійно	База даних, ServiceDesk
2	Довідник комп'ютерів (або іншого обладнання)	PK	масив	Постійно	База даних, ServiceDesk
3	Довідник працівників (спеціалістів, які направляють заявку)	Personal	масив	Постійно	База даних, ServiceDesk
4	Звернення	Zvernienia	масив	1 раз у місяць	ServiceDesk
5	Довідник філій	Filia	масив	Постійно	База даних
6	Група проблем	Group_problem	масив	Постійно	База даних
7	Довідник працівників, які обслуговують IT-інфраструктуру	Service_person	масив	Постійно	База даних

Джерело: розроблено автором самостійно

Структура інформаційної управляючої системи сервісної підтримки ІТ інфраструктури показана на рис. 2.2.

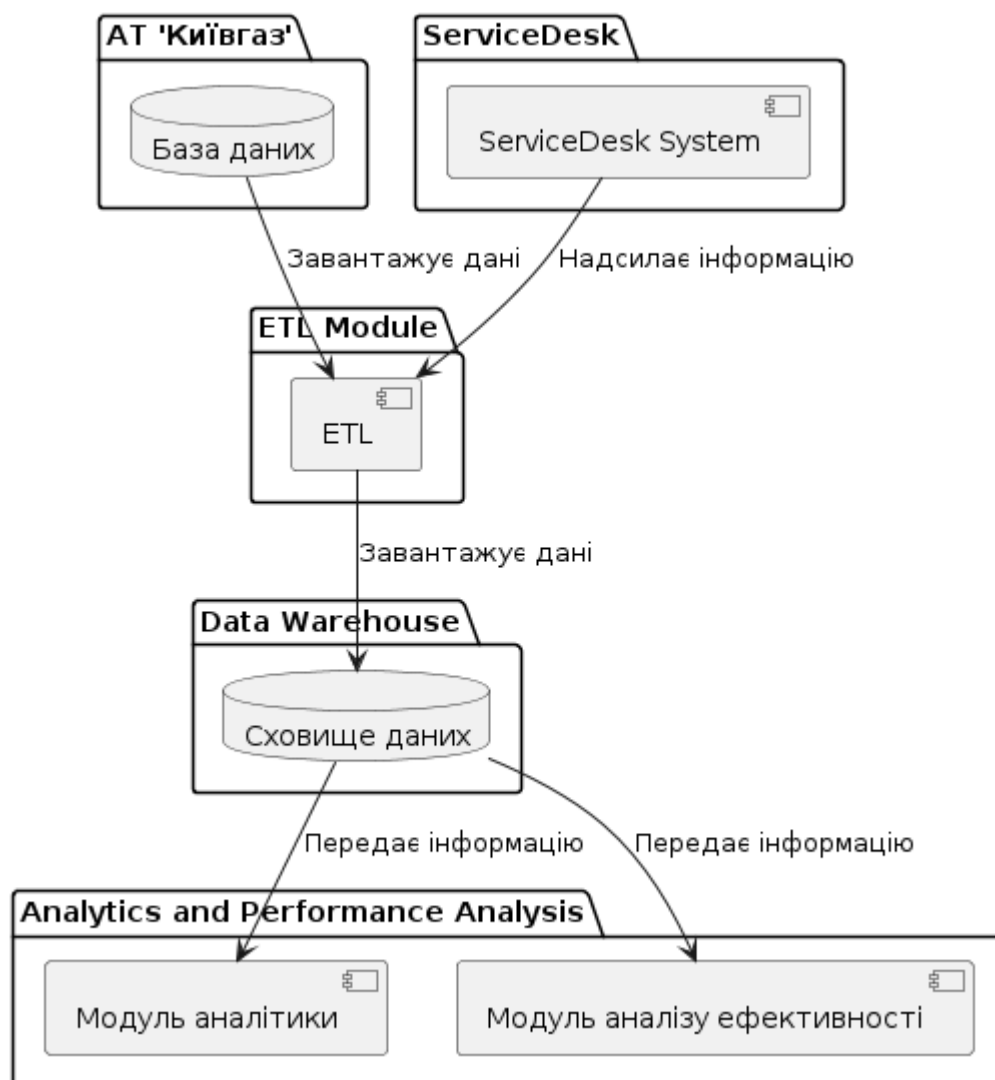


Рисунок 2.2 – Структура інформаційної управляючої системи сервісної підтримки ІТ інфраструктури

Джерело: розроблено автором самостійно

Розглянемо детально цю схему. З ServiceDesk надходить інформація із заявками у модуль ETL, який призначений для отримання інформації і перетворення її до необхідної структури. Також з центральної бази даних АТ "Київгаз" завантажуються у модуль ETL довідники.

Дана інформація перетворюється у структуру, необхідну для завантаження у сховище даних.

Завантажена у сховище даних інформація використовується для інтелектуального аналізу, аналізу ефективності та планування обслуговування.

Структура системи показана у вигляді діаграми класів, поданій на рис. 2.3.

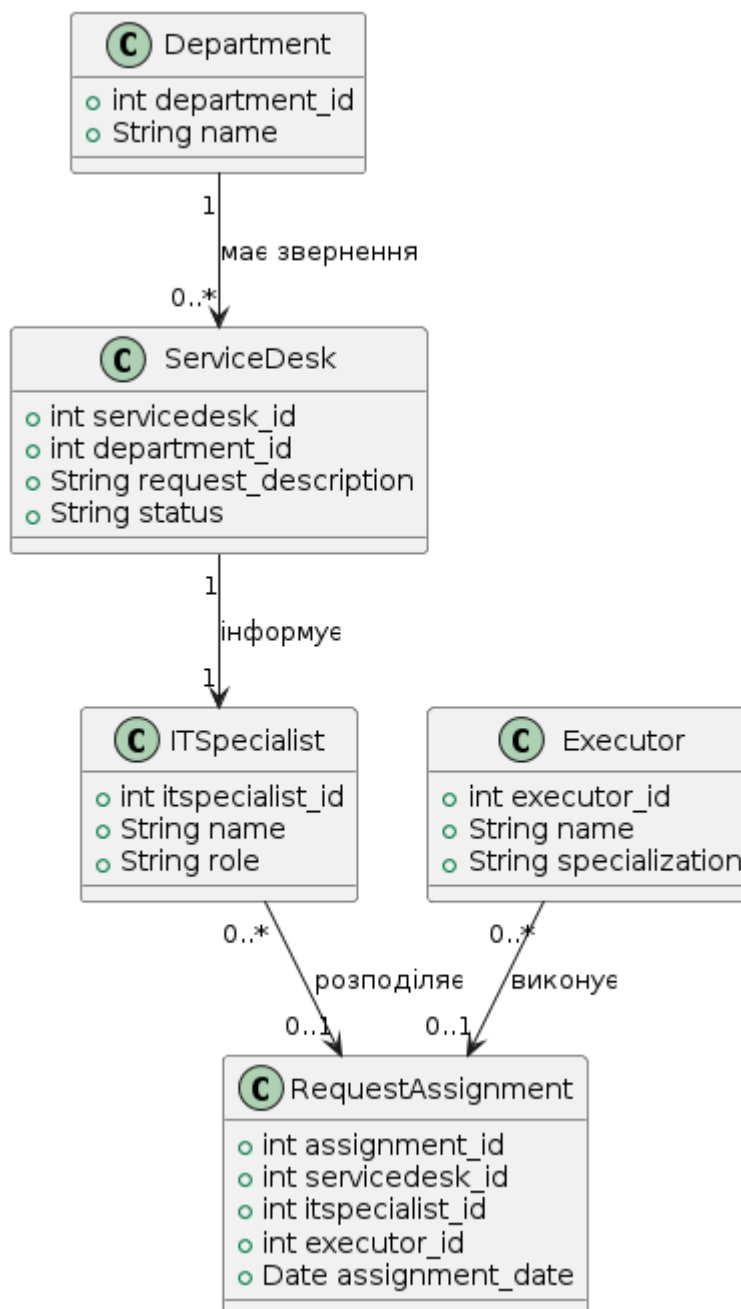


Рисунок 2.3 – Структура інформаційної управляючої системи сервісної підтримки ІТ інфраструктури у вигляді діаграми класів

Джерело: розроблено автором самостійно

2.2 Методи та моделі в інформаційних управляючих системах і технологіях

Створення "Інформаційної управляючої системи сервісної підтримки ІТ інфраструктури АТ 'Київгаз'" може значно підвищити ефективність роботи компанії та якість обслуговування клієнтів. Вивантаження заявок з сервіс-деску та подальший аналіз цієї інформації можуть стати основою для інтелектуальних рішень. Отриману інформацію можна використовувати для вирішення наступних завдань.

1. Аналіз даних та побудова звітів. Регулярне вивантаження заявок та їх аналіз може дати змогу виявити такі аспекти:

- частотність заявок: визначити найпоширеніші типи проблем та запитів;
- час реагування та вирішення: вимірювати середній час від подання заявки до її вирішення, ідентифікувати вузькі місця.

2. Виявлення трендів та аномалій. Використання інструментів бізнес-аналітики та машинного навчання для:

- виявлення сезонних трендів, наприклад, збільшення заявок у певні місяці чи дні тижня;
- прогнозування проблем, ідентифікація потенційних проблем на основі історичних даних;
- виявлення аномалій, знаходження відхилень від норми, які можуть свідчити про нові або невідомі проблеми.

3. Оптимізація процесів. На основі отриманих даних можна здійснити:

- покращення процесів: ідентифікація і усунення найслабших місць в процесі обслуговування заявок;
- розподіл ресурсів: оптимізація розподілу ресурсів та персоналу на основі аналізу навантаження.

4. Покращення якості обслуговування. Використання аналізу даних для:

- навчання та тренінги: визначення потреб у додатковому навчанні для співробітників на основі часто повторюваних проблем;
- покращення комунікації: створення стандартних відповідей та бази знань для швидшого вирішення типових заявок.

5. Інтелектуальна підтримка прийняття рішень. Використання систем підтримки прийняття рішень (DSS) для:

- пріоритизація заявок: використання алгоритмів машинного навчання для автоматичного визначення пріоритету заявок.
- рекомендації щодо дій: надання операторам рекомендацій щодо найкращих дій на основі аналізу попередніх випадків.

6. Моніторинг та контроль якості. Постійний моніторинг показників ефективності (KPI) для відстеження якості обслуговування: вимірювання задоволеності клієнтів та інших KPI.

За результатами аналізу існуючих методів запропонована таблиця 2.3, в якій визначені основні методи і моделі, які застосовують в різних напрямках управл.чих систем.

Таблиця 2.3 – Перелік методів і моделей в інформаційній управляючій системі сервісної підтримки ІТ інфраструктури

Напрямок аналізу	Методи та моделі	Опис
1	2	3
Аналіз частотності заявок	Дескриптивна статистика: підрахунок частот, модальні значення, гістограми. Аналіз тенденцій: часові ряди, сезонність.	Визначення найпоширеніших типів заявок, візуалізація розподілу заявок за типами, аналіз змін частотності з часом.
Аналіз часу реагування та вирішення	- Аналіз часових інтервалів: обчислення середнього часу, медіана та розподіл часу. Визначення вузьких місць: процесний аналіз, аналіз продуктивності.	Вимірювання середнього часу від отримання заявки до її вирішення, ідентифікація вузьких місць у процесах обслуговування.

Аналіз продуктивності команд	- КРІ: заявки на одного співробітника, час вирішення. - Аналіз ефективності: бенчмаркінг, аналіз результативності.	Оцінка ефективності команд, порівняння продуктивності різних команд, визначення найкращих практик.
Виявлення трендів	- Аналіз часових рядів: тренд та сезонність (STL decomposition), поточне згладжування (Exponential Smoothing). Прогнозування: ARIMA, LSTM.	Виявлення довгострокових трендів і сезонних коливань у заявках, прогнозування майбутніх трендів.
Виявлення вузьких місць (Bottleneck Analysis)	- Процесний аналіз: діаграми потоку (flowcharts), аналіз бізнес-процесів. - Аналіз продуктивності: аналіз тимчасових інтервалів, коефіцієнти використання.	Ідентифікація етапів процесу, які затримують вирішення заявок, визначення вузьких місць для їх подальшого усунення.
Моніторинг та контроль якості	- Візуалізація даних: створення дашбордів для моніторингу ключових показників. - Автоматизовані системи моніторингу: оповіщення про порушення. - Статистичний контроль процесів (SPC).	Створення інтерактивних дашбордів для моніторингу КРІ, автоматизовані системи для відстеження порушень у процесах, використання статистичних методів для контролю якості процесів, забезпечення оперативного реагування на проблеми.

Джерело: розроблено автором самостійно

РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРОЕКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

3.1 Проектування бази даних та сховища даних для інформаційної управляючої системи

Для роботи інтелектуальної системи необхідно спроектувати сховище даних, яке б дозволило здійснювати аналітику заявок в ServiceDesk.

Етапи проектування бази даних включають наступні етапи: дослідження предметної області, проектування бази даних на логічному рівні, проектування бази даних на фізичному рівні.

Проектування бази даних на логічному рівні було здійснено в середовищі vertabelo, результати проектування показані на рис. 3.1.

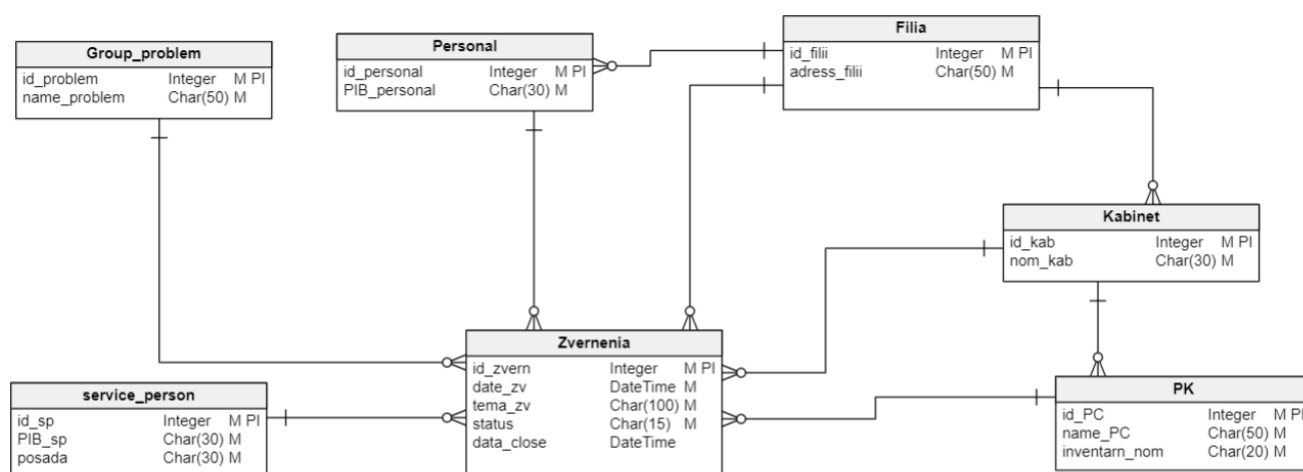


Рисунок 3.1 –Логічна модель бази даних

Джерело: розроблено автором самостійно

Логічну модель бази даних відобразимо на фізичну модель. Процес генерування бузи даних з логічної моделі на фізичну в середовищі vertabelo показаний на рис. 3.2.

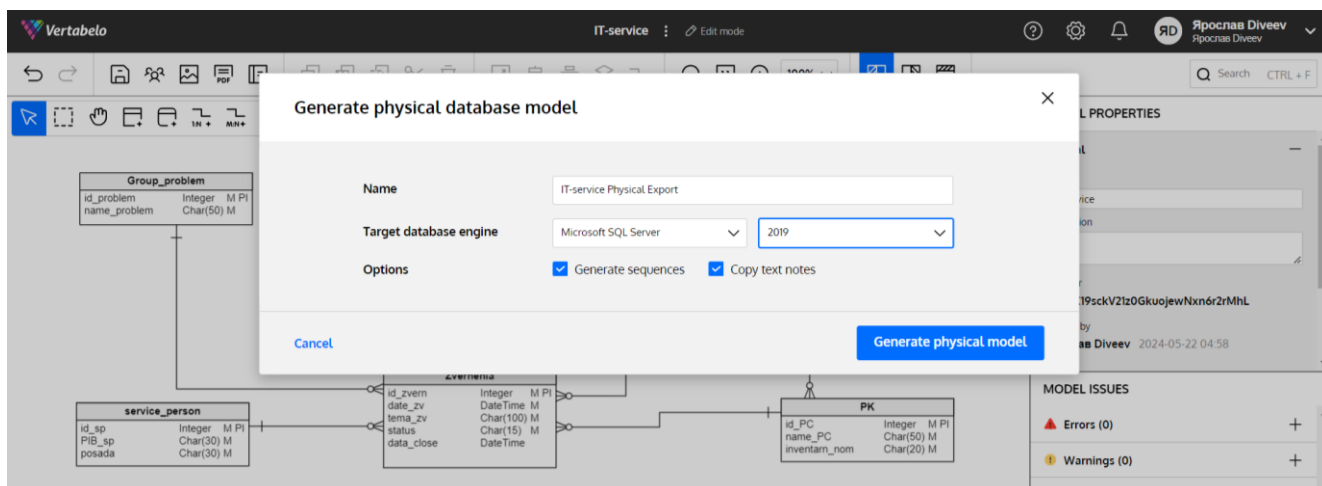


Рисунок 3.2 – Генерування фізичної моделі бази даних

Джерело: розроблено автором самостійно

На рисунку 3.3 показаний результат створення фізичної моделі, яка згенерована в середовищі vertabelo.

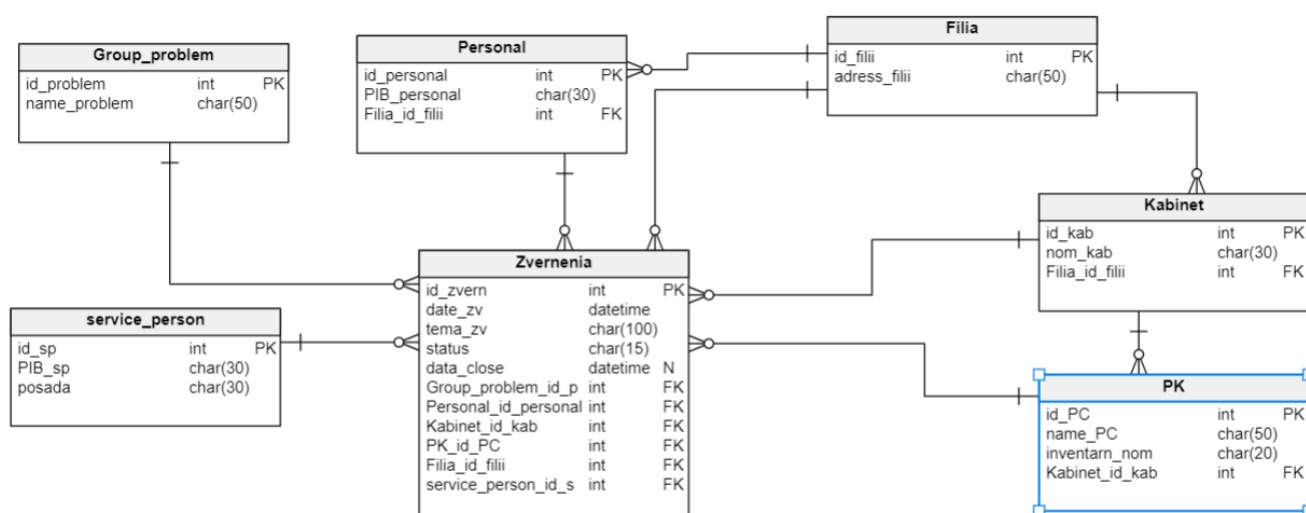


Рисунок 3.3 – Фізична модель бази даних

Джерело: розроблено автором самостійно

За результатами аналізу предметної області було прийнято про розроблення сховища даних, яке б зберігало інформацію про отримані заявки.

В якості моделі сховища даних обрана модель типу «зірка».

Основними перевагами моделі типу «зірка» є наступні:

- мають більш зрозумілу логіку об'єднання порівняно з іншими схемами для отримання даних із високо нормалізованих транзакційних схем;

- на відміну від високо нормалізованих транзакційних схем, зірчаста схема спрощує загальну логіку бізнес-звітності, таку як звітування та період за періодом;
- зіркові схеми широко використовуються системами OLAP для ефективного проектування кубів, зіркову схему можна використовувати як джерело без проектування кубічної структури в більшості основних систем OLAP;
- увімкнувши певні схеми продуктивності, які можна застосовувати до запитів, програмне забезпечення процесора запитів у зірковій схемі може запропонувати кращі плани виконання.

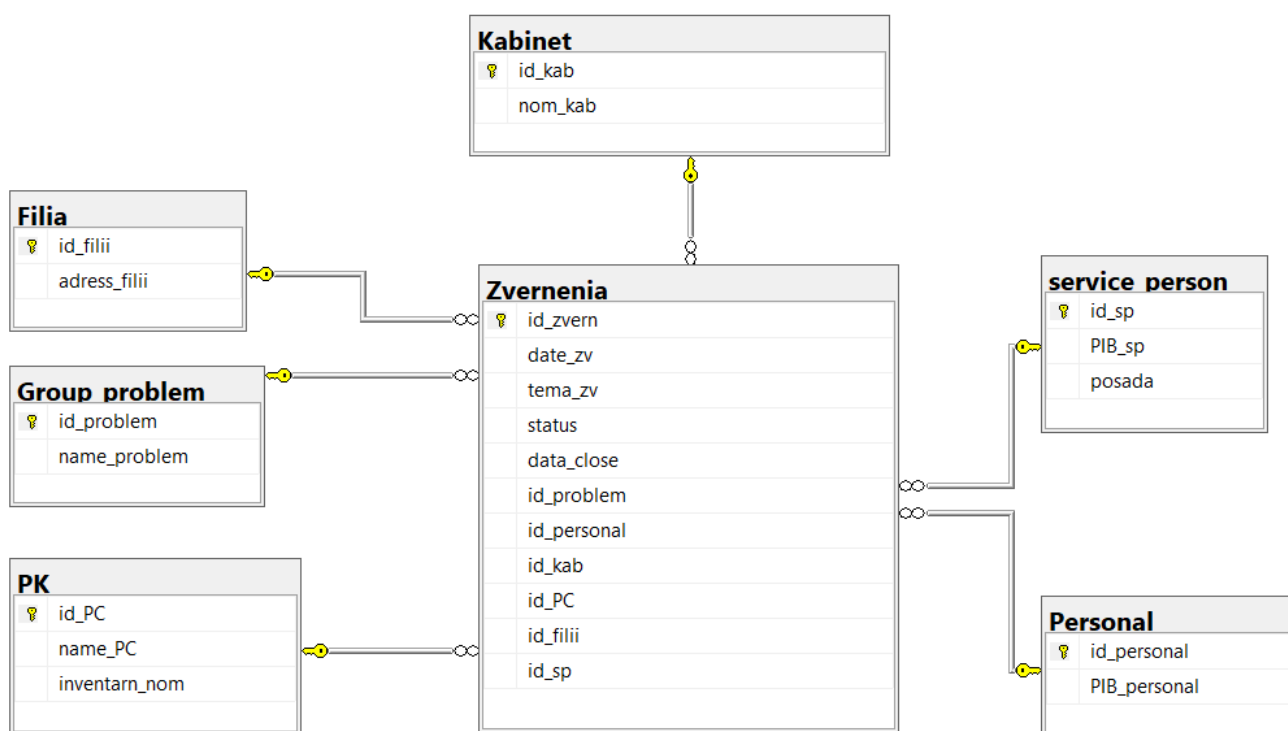


Рисунок 3.4 – Фізична модель сховища даних

Джерело: розроблено автором самостійно

Розглянемо таблиці даного сховища.

Сховище має таблицю-фактів, в якій відображена інформація про конкретну заявку.

Для створення таблиці-фактів скористалися командною послідовністю мовою SQL:

```

CREATE TABLE Zvernenia (
    id_zvern int PRIMARY KEY ,
    date_zv datetime NOT NULL,
    tema_zv char(100) NOT NULL,
    status char(15) NOT NULL,
    data_close datetime NULL,
    id_problem int NOT NULL,
    id_personal int NOT NULL,
    id_kab int NOT NULL,
    id_PC int NOT NULL,
    id_filii int NOT NULL,
    id_sp int NOT NULL
);

```

Дана таблиця має виміри: код проблеми (id_problem), код особи, що робить заявку (id_personal), код кабінету (id_kab), код комп'ютера (id_PC), код філії (id_filii), код фахівця IT-відділу (id_sp). Структура таблиці-фактів «Звернення» подана в таблиці 3.1. Наступні таблиці-виміри показані в таблицях 3.2-3.7.

Таблиця 3.1 – Опис масиву “Звернення”

Найменування масиву — Звернення

Ідентифікатор масиву — Zvernenia

Найменування носія інформації — Жорсткий диск

Максимальний об'єм масиву — 1000000 записів.

Довжина запису — 31 символ

Метод організації — послідовний.

Ключі упорядкування — id_zvern

Ідентифікатор індексного масиву — ind_id_zvern

Найменування	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код звернення	id_zvern	z	9(8)	РК	-	так	ІН Д	-

Дата звернення	date_zv	do	9(8)	-	-	так	-	-
Тема звернення	tema_zv		X(100)	-	-	так	-	-
Статус	status	s	X(15)	-	-	так	-	-
Дата закриття	data_close	dc	9(8)	-	-	так	-	-
Код групи проблеми	id_problem	k	9(3)	FK	-	так	ІД Д	Group _probl em
Код персоналу	id_personal	p	9(4)	FK	-	так	ІД Д	Person al
Код кабінету	id_kab	a	9(3)	FK	-	так	ІД Д	Kabine t
Код комп'ютера	id_PC	d	9(5)	FK	-	так	ІД Д	PK
Код філії	id_filii	f	9(2)	FK	-	так	ІД Д	Filia
Код спеціаліста ServiceDesk	id_sp	i	9(2)	FK	-	так	ІД Д	service _perso n

Джерело: розроблено автором самостійно

Таблиця 3.2 – Опис масиву “Довідник філій”

Найменування масиву — Довідник філій

Ідентифікатор масиву — Filia

Найменування носія інформації — Жорсткий диск

Максимальний об'єм масиву — 100 записів.

Довжина запису — 52 символи

Метод організації — послідовний.

Ключі упорядкування — id_id_filii

Ідентифікатор індексного масиву — ind_id_filii

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код філії	id_filii	f	9(2)	PK	-	так	ІН Д	
Адреса філії	adress_filii	-	X(50)					

Джерело: розроблено автором самостійно

Таблиця 3.3 – Опис масиву “Довідник груп проблем”

Найменування масиву — Довідник груп проблем

Ідентифікатор масиву — Group_problem

Найменування носія інформації — Жорсткий диск

Максимальний об’єм масиву — 100 записів.

Довжина запису — 53 символи

Метод організації — послідовний.

Ключі упорядкування — id_problem

Ідентифікатор індексного масиву — ind_id_problem

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв’язки
				Первинний /вторинний ключ	Умова на значення	Обов’язкове поле	Індексне поле	
Код групи проблеми	id_problem	gp	9(3)	PK	-	так	ІН Д	Zverne nia
Назва проблеми	name_problem	np	X(50)	-	-	так	-	-

Джерело: розроблено автором самостійно

Таблиця 3.4 – Опис масиву “Довідник кабінетів”

Найменування масиву — Довідник кабінетів

Ідентифікатор масиву — Kabinet

Найменування носія інформації — Жорсткий диск

Максимальний об’єм масиву — 1000 записів.

Довжина запису — 33 символи

Метод організації — послідовний.

Ключі упорядкування — id_kab

Ідентифікатор індексного масиву — ind_id_kab

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код кабінету	id_kab	kb	9(3)	РК	-	так	ІН Д	Zvernia
Номер кабінету	nom_kab	nk	X(30)	-	-	так	-	-

Джерело: розроблено автором самостійно

Таблиця 3.5 – Опис масиву “Довідник комп’ютерів та іншої техніки”

Найменування масиву — Довідник комп’ютерів та іншої техніки

Ідентифікатор масиву — РС

Найменування носія інформації — Жорсткий диск

Максимальний об’єм масиву — 1000 записів.

Довжина запису — 74 символи

Метод організації — послідовний.

Ключі упорядкування — id_id_PC

Ідентифікатор індексного масиву — ind_id_id_PC

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код комп'ютера	id_PC	idpc	9(4)	РК	-	так	ІН Д	Zvernia
Назва комп'ютера	name_PC	npc	X(50)	-	-	так	-	-
Інвентарний номер	inventarn_nom	inom	X(20)	-	-	так	-	-

Джерело: розроблено автором самостійно

Таблиця 3.6 – Опис масиву “Довідник працівників компанії”

Найменування масиву — Довідник працівників компанії

Ідентифікатор масиву — Personal

Найменування носія інформації — Жорсткий диск

Максимальний об’єм масиву — 1000 записів.

Довжина запису — 33 символи

Метод організації — послідовний.

Ключі упорядкування — id_id_personal

Ідентифікатор індексного масиву — ind_id_personal

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв’язки
				Первинний /вторинний ключ	Умова на значення	Обов’язкове поле	Індексне поле	
Код персоналу	id_personal	idp	9(3)	РК	-	так	ІН Д	Zverne nia
ПІБ персоналу	PIB_personal	pibp	X(30)	-	-	так	-	-

Джерело: розроблено автором самостійно

Таблиця 3.5 – Опис масиву “Довідник працівників ІТ відділу”

Найменування масиву — Довідник працівників ІТ відділу

Ідентифікатор масиву — service_person

Найменування носія інформації — Жорсткий диск

Максимальний об’єм масиву — 1000 записів.

Довжина запису — 66 символи

Метод організації — послідовний.

Ключі упорядкування — id_kab

Ідентифікатор індексного масиву — ind_id_kab

Найменування поля	Ідентифікатор у програмі	Умове позначення у формулах	Формат	Бізнес-правила				Логічні чи семантичні зв'язки
				Первинний /вторинний ключ	Умова на значення	Обов'язкове поле	Індексне поле	
Код особи ІТ відділу	id_sp	ids	9(3)	PK	-	так	ІН Д	Zverne nia
ПІБ особи ІТ відділу	PIB_sp	pibs	X(30)	-	-	так	-	-
Посада	posada	pos	X(30)	-	-	так	-	-

Джерело: розроблено автором самостійно

3.2 Проектування засобів OLAP-аналізу даних

При управлінні сервісною підтримкою ІТ інфраструктури АТ "Київгаз" можуть виникати наступні ситуації, показані в таблиці 3.8.

Таблиця 3.8 – Розподіл задач і ситуацій, пов'язаних із прийняттям рішення.

Ситуація	Персонал	Технічні засоби
Профілактика та планове обслуговування	Підготовка та планування профілактичних заходів, розробка планів обслуговування	Автоматизоване виконання планових обслуговувань, тестування, перевірка стану обладнання та систем
Виявлення та діагностика проблем	Виявлення сигналів проблем, аналіз вихідних даних, спілкування з користувачами для з'ясування деталей	Моніторинг стану системи, виявлення аномалій, виконання тестів та діагностичних процедур
Усунення проблем та відновлення роботи	Виявлення причин проблеми, прийняття рішення щодо методів усунення, виконання необхідних дій для відновлення роботи	Автоматичне виконання відновлювальних процедур, автоматичне відновлення системи до працездатного стану після усунення проблеми
Моніторинг та аналіз ефективності	Аналіз вихідних даних, оцінка ефективності вжитих заходів, розробка рекомендацій щодо подальших кроків	Збір та агрегація даних, створення звітів та аналітичних звітів, надання даних для аналізу персоналу

Джерело: розроблено автором самостійно

Розглянемо основні показники, які можна моніторити та аналізувати в спроектованому сховищі даних.

Показники (3.1) – (3.5) використовують при аналізі часу реагування при вирішенні проблем.

1. Середній час реагування (Average Response Time). Це середній час, який проходить від подання заявки до першої відповіді на неї.

Середній час реагування (Average Response Time). Це середній час, який проходить від подання заявки до першої відповіді на неї.

$$\text{Average Response Time} = \left(\frac{\sum_i \text{Response Time}_i}{n} \right) \quad (3.1)$$

Response Time_i - час від подання i -ї заявки до першої відповіді на неї.
 n - загальна кількість заявок.

2. Середній час вирішення (Average Resolution Time). Це середній час, який проходить від подання заявки до її повного вирішення.

$$\text{Average Resolution Time} = \frac{\sum_{i=1}^n (\text{Resolution Time}_i)}{n} \quad (3.2)$$

3. Процент заявок, вирішених у рамках SLA (Service Level Agreement Compliance Rate). Це відсоток заявок, які були вирішені у рамках встановленого SLA (договірного рівня обслуговування).

$$\text{SLA Compliance Rate} = \left(\frac{\text{Number of Requests Resolved within SLA}}{\text{Total Number of Requests}} \right) \times 100\% \quad (3.3)$$

$\text{Number of Requests Resolved within SLA}$ - кількість заявок, вирішених у встановлений **SLA** час.

$\text{Total Number of Requests}$ - загальна кількість заявок.

4. Середній час між етапами процесу (Average Time Between Process Stages). Це середній час, який проходить між різними етапами процесу обробки заявки.

$$\text{Average Time Between Stages} = \frac{\sum_{i=1}^n (\text{Stage Time}_i)}{n} \quad (3.4)$$

$Stage\ Time_i$ - час, що проходить між конкретними етапами для i -ї заявки (наприклад, від першої відповіді до ескалації).

n - загальна кількість заявок.

Наступні показники (3.5) – (3.9) використовують для ефективності обробки заявок.

5. Процент заявок, вирішених з першого разу (First Contact Resolution Rate). Це відсоток заявок, які були вирішені при першому зверненні.

Формула:

$$FCR\ Rate = \left(\frac{N_Requests\ Resolved\ on\ FC_Total\ Number\ of\ Requests}{Total\ Number\ of\ Requests} \right) 100\%$$

$N_Requests\ Resolved\ on\ FC_Total\ Number\ of\ Requests$ - кількість заявок, вирішених при першому зверненні.
 $Total\ Number\ of\ Requests$ - загальна кількість заявок.

3.3. Програмне забезпечення

3.3.1 Вимоги до програмного забезпечення

Для реалізації нового застосунку «IT infrastructure» нами використовується архітектурний шаблон MVC (Модель-Відображення-Контролер), який дозволив чітко виокремити серверну частину програми від клієнтської. Передачу даних між основними компонентами програми будемо здійснено за допомогою текстового формату XML.

3.3.2 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем

Розробку серверної частини програми реалізували на фреймворці SpringBoot, якій містить реалізацію обраного нами патерну програмування. Реалізація архітектурного рішення здійснювалась у середовищі IntelliJIdead. Також для автоматизованої збірки проекту використовувалась технологія Apache Maven. Ця технологія полегшила управління проектом, а саме: дозволила підключити необхідні бібліотеки, здійснити зборку проекту та його розміщення на сервері.

Для реалізації архітектурних рішень у проекті нами були підключені такі бібліотеки, як:

```
spring-boot-starter-web: 3.2.5;  
spring-boot-starter-devtools: 3.2.5;  
spring-boot-starter-data-jpa: 3.2.5;  
spring-boot-starter-freemarker: 3.2.5;  
spring-boot-starter-security: 3.2.5;  
hibernate-validator.
```

Перевагою роботи зі SpringBoot є реалізація пакетів за чітко визначеною структурою обраного патерну.

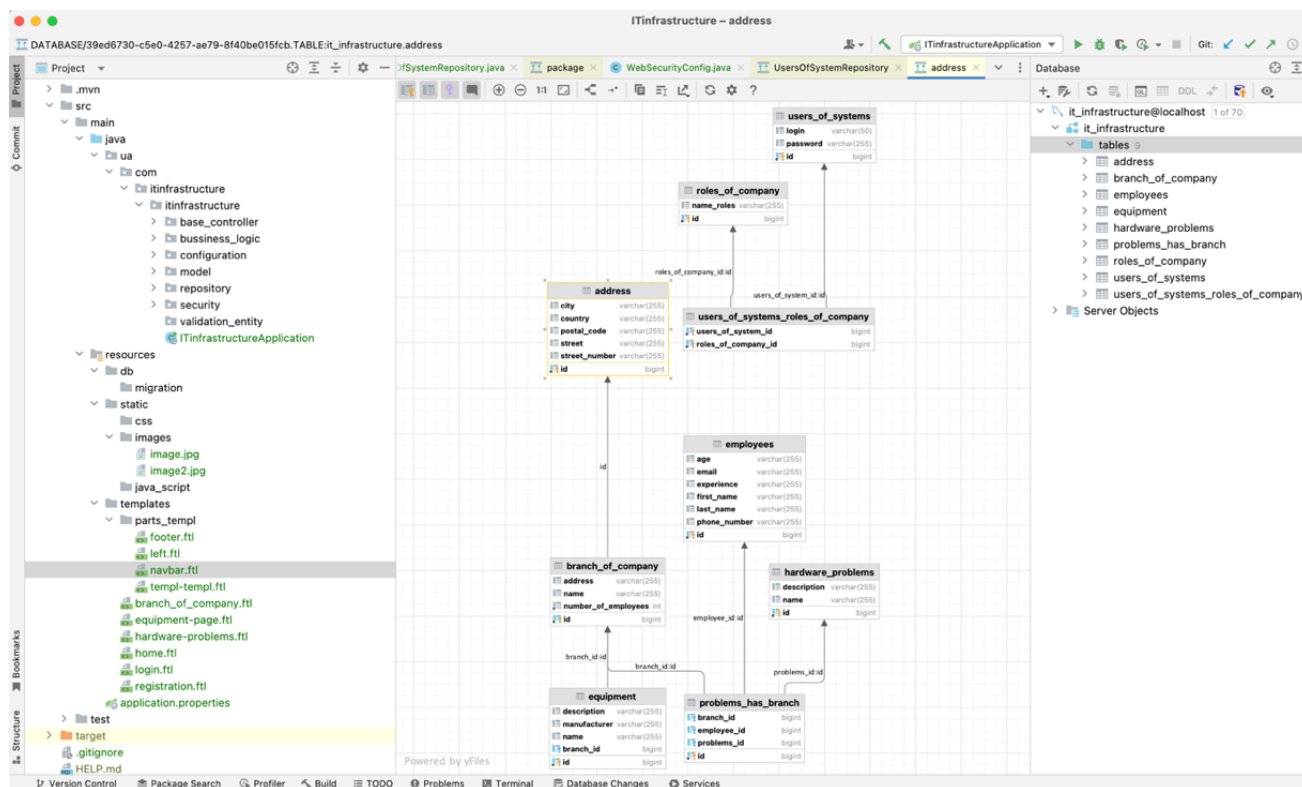


Рисунок – 3.5 - Середовище розробки застосунку «IT infrastructure»

Джерело: розроблено автором самостійно

Нами були розроблені та наповнені такі пакети програми, як:

- **model** – містить основні класи сутностей за проектом: Employees, Equipment, HardwareProblems, HardwareProblemsHasBranchOfCompany, RolesOfCompany, BranchOfTheCompany, UsersOfSystem, Address of the Company, зі структурою яких ви можете ознайомитись у додатку А.1;
- **repository** - містить реалізацію GRUD-операцій для описаних вище класів-сутностей, для цього нами були розроблені такі інтерфейси: EmployeesRepository, EquipmentRepository, HardwareProblemsRepository, HardwareProblemsHasBranchOfCompanyRepository, RolesOfCompanyRepository, BranchOfTheCompanyRepository, UsersOfSystemRepository, AddressOfCompanyRepository;
- **bussiness_logic** – пакет, необхідний для реалізації основної логіки роботи з визначеними класами-сутностей;

- security – пакет, необхідний для реалізації системи аутентифікації та авторизації з використанням фреймворку Spring Security;
- configuration – для розширення налаштувань SpringBoot;
- validation_entity – для валідування та верифікації даних.

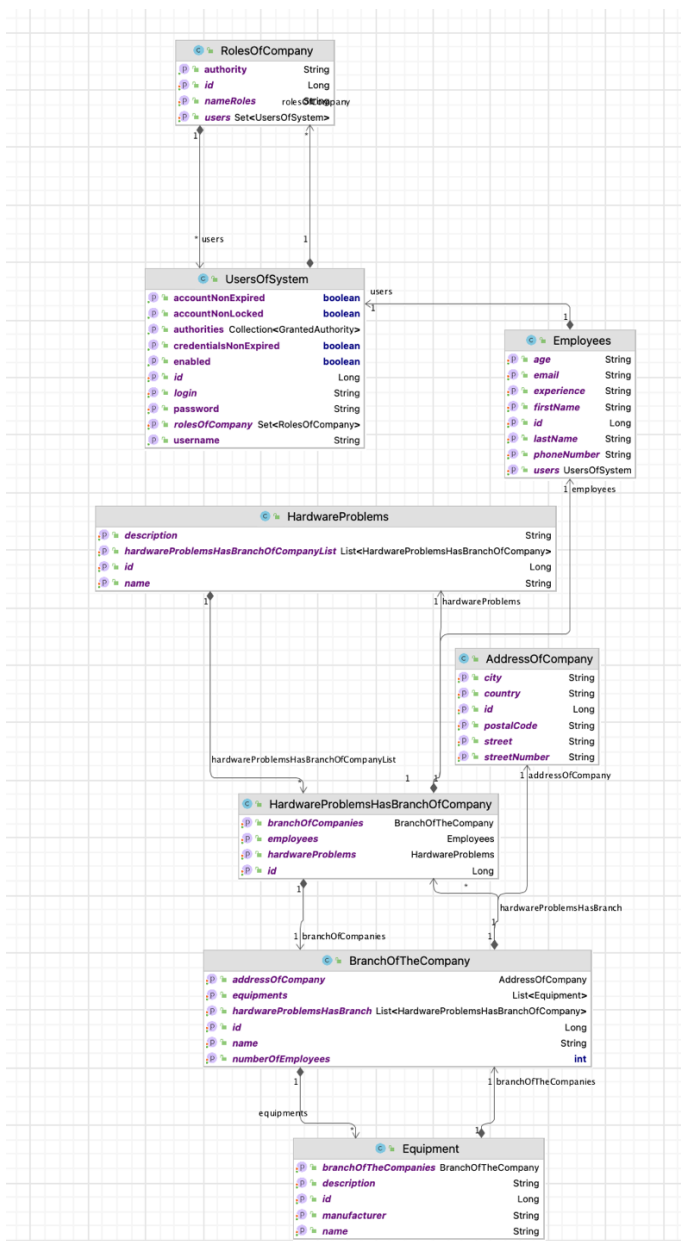


Рисунок – 3.6 - Структура класів-сутностей у програмі «IT infrastructure»

Джерело: розроблено автором самостійно

Передача даних між основними компонентами програми відбувається за допомогою шаблонізатора Apache Freemarker. Технологія дозволяє вставляти

динамічні дані з моделі у шаблон, тим самим вирішувати питання трансформації даних з байт-коду у текстовий формат і навпаки.

Для взаємодії з БД та реалізації запитів до її таблиць нами було використано технологію Hibernate, - це технологія, яка дозволила зв'язати наші класи-сутностей з таблицями БД, а також підключити технологію JPA-Repository.

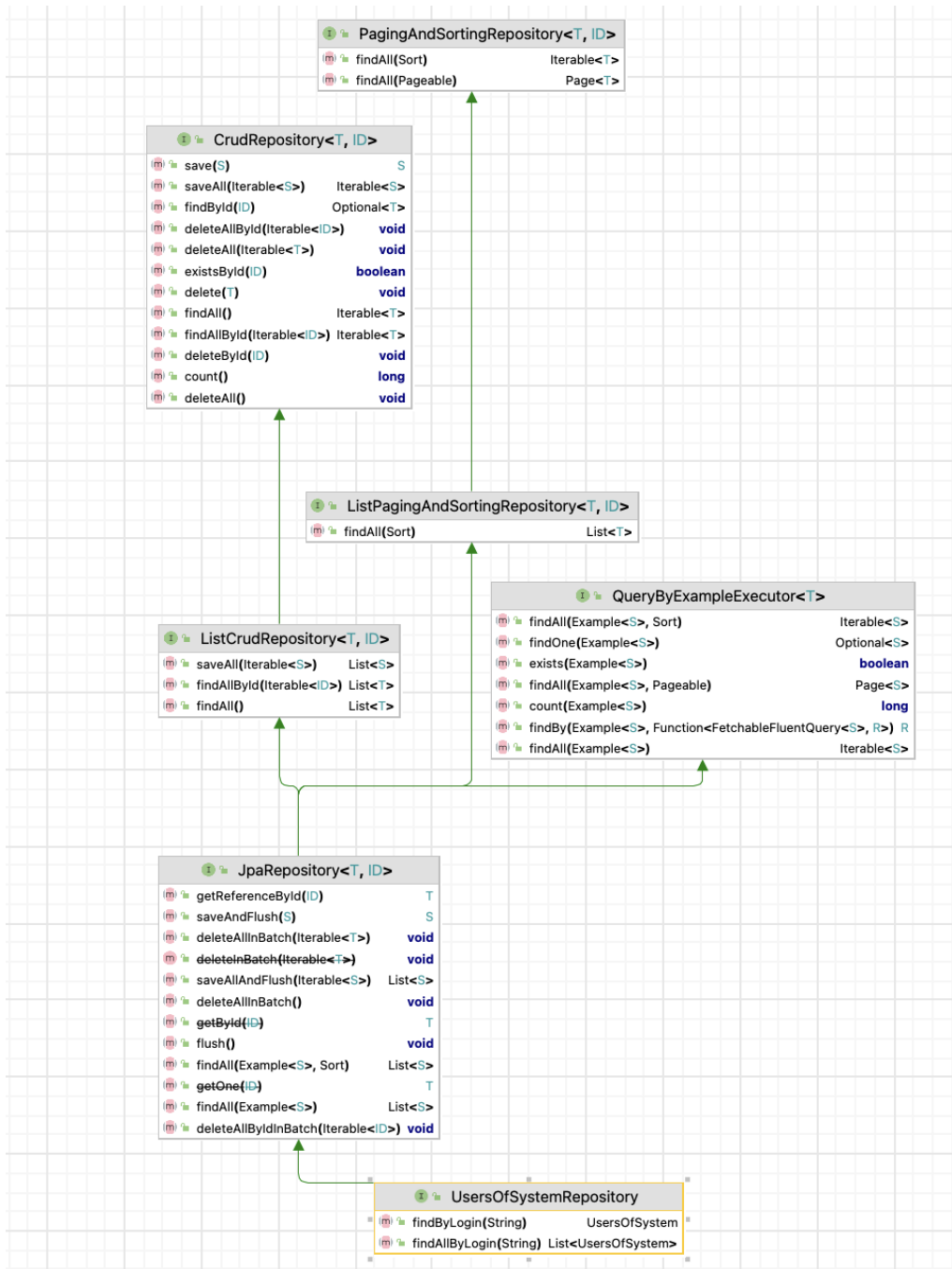


Рисунок - 3.7 - Підключення JPA-Repository до класу користувачів системи

Джерело: розроблено автором самостійно

Технологія JPA-Repository дозволила нам підключити GRUD-запити для роботи з основними класами-сутностей. Для цього у відповідні інтерфейси та параметри дженеріка нами передавалась назва класу та тип ідентифікатора. Наприклад, для реалізації JpaRepository з користувачами системи, ми передавали у дженерік назву класу «UsersOfSystem» та тип Id-параметру у вигляді Long.

Для розподілу бізнес-логіки між основними користувачами системи нами була налаштована і реалізована система фільтрів в середовищі SpringSecurity. Для забезпечення профілювання (розподілу ролей) та систем ідентифікації користувачів нами був налаштований SecurityFilterChain.

Реалізацію налаштування фільтру представлено у вигляді лістингу коду.

```
@Bean
public SecurityFilterChain authenticationSecurityFilterChain(HttpSecurity http)
throws Exception {

    http.csrf((csrf)->csrf.disable())
        .authorizeHttpRequests((authorize)->

            authorize.requestMatchers("/",
                "/login",
                "/static/**",
                "/registration",
                "/registration/**").permitAll()
                .requestMatchers("/user", "/equipment", "/new-branch-
of-company",
                    "/saveNewEq", "/hardware-
problems", "/saveHardwareProblems"
                        ).hasRole("User")
                .requestMatchers("/manager").hasRole("Manager")
                .requestMatchers("admin").hasRole("Admin")
                .anyRequest().authenticated()
            )
        .formLogin((form)->
            form.loginPage("/login")
                .defaultSuccessUrl("/")
                .permitAll()
            )
        .logout((logout)->logout.permitAll().logoutSuccessUrl("/"))
        .authenticationProvider(authenticationProvider());

    return http.build();
}
```

У цьому фільтрі нами було описано доступ до основних сторінок веб-сайту з урахуванням ролей користувачів системи, налаштовано сторінку аутентифікації(auth) та виходу з системи (logout).

3.3.3 Архітектура програмного забезпечення

Розробку нового застосунку «IT infrastructure» здійснювали на основі монолітної архітектури, а для реалізації обліку користувача в системі – технологію session. У застосунку використовується єдина база даних і єдина система безпеки, що полегшує взаємодію між компонентами та не потребує додаткових рішень щодо захисту та передачу даних, як у мікросервісній архітектурі.

3.3.4. Тестування програмного забезпечення

Для тестування програмного забезпечення, розробленого застосунку «IT infrastructure» нами було використано комплексно-інтегровані тести. Використання бібліотек SeleniumChromeDriver та WebDriverManager дозволило реалізувати тести та перевірити передачу даних з клієнтської частини на сервер, а також основні переходи між сторінками сайту.

Тестування реалізованих запитів до БД здійснювалось на основі бібліотеки Junit5. Проведене тестування показало, що запити до БД і робота по передачі даних через контролери реалізується коректно.

3.4. Технічне забезпечення

Мінімальні системні та технічні вимоги для виконання даного проекту зі сторони користувача та зі сторони компанії є обов'язковими для планування даного проекту в реалізацію. Баз даних характеристик можливі проблеми як на стадії користувача, який може використовувати застарілу ОС або техніку, або недостатньо потужна система з боку серверів у Дата-центрі, що спричинить

падіння системи або перевантаження і зависання. Для клієнта мінімальними вимогами є:

- CPU – 1x Intel Celerin N3350 (1.1-2.4 GHz);
- RAM – 4 GB DDR3;
- 1x 500 GB SATA HDD;
- Інтегрована відеокарта HD Graphics 500. Інтегрована в процесор.

Також потрібен стабільний доступ до глобальній мережі Internet бажано через інтернет провайдера, мобільний інтернет не має стабільного підключання що може привести до великих затримок між користувачем та серверами. Підійде стандартний маршрутизатор Wi-Fi з підтримкою швидкості 100 Mb/секунду з підтримкою стандартом зв'язку Wi-Fi 802.11b/g/a на частоті 2.4 ГГц. Після вихода в мережу він іде до обладнання у Дата – центрі. Там є своє мережеве обладнання, до якого під'єднується окрема машина. Мінімальними характеристиками даної машини для обробки даних та видачі результатів є:

- CPU – 1x Broadcom BCM2711, Quad core Cortex-72 (ARM v8) 64-bit SoC (1.5 GHz);
- RAM - 8 GB LPDDR4-2400 SDRAM;
- Підключення через 2.4 GHz and 5.0 GHz IEEE 802.11ac або Gigabit Ethernet;
- Micro-SD card або зовнішній розміром до 500 Gb;
- OpenGL ES 3.0 graphics.

Підключення в такій системі може бути або через гігабітний дрідт напряду або через Wi-Fi як резервне підключення. Усе підключення має форму – користувач по запиту із особистого кабінта – інтернет – сервер по запиту дає запит на потрібний віддалений сервер користувача. Сервер користувача мож бути будь-якої форми і конфігурації.

Також не слід забувати що у Дата – центрі росташований окремий сервер з даними користувачів, який постійно оновлюється реєстрацією або запитами на видалення записів, і у якому росташовані дані профілів робітників за для

зурочності роботи. Також автономна система живлення, яка не дає у випадку подання світла на якийсь проміжок часу і далі працювати усім серверам а також працівникам і клієнтам.

Треба розуміти, що сервер – це не одна стійка, а великий масив із десятків машин, як і користувачі це не 1 ПК, а велика кількість з'єднань з різних частин світу до їх серверів. Тому збільшення ланцюгу між Дата – центром та користувачем має під собою додаткові затрати а також втрату швидкості між клієнтом і його сервером. До того ж сервера мають декілька підмережей з різним пулом IP – адрес що приводить до висновку що сервер який відповідатиме на запити повинен знаходитись на самій першій ланці комутації щоб мати більш швидкий доступ до інших частин інфраструктури Дата – центру.

3.5 Реалізація інформаційної управляючої системи

Розробку нового застосунку «IT Infrastructure» реалізує клієнт-серверна архітектура рішень. Сайт містить горизонтальне меню, ліве меню, робочу область та нижню частину сайту. Всі сторінки сайту є адаптивними. Клієнт може працювати з будь-якого пристрою, бо у програмі реалізовано адаптивний шаблон. На наступних рисунках представлені динамічні макети сторінок, які були інтегровані у програмний продукт.

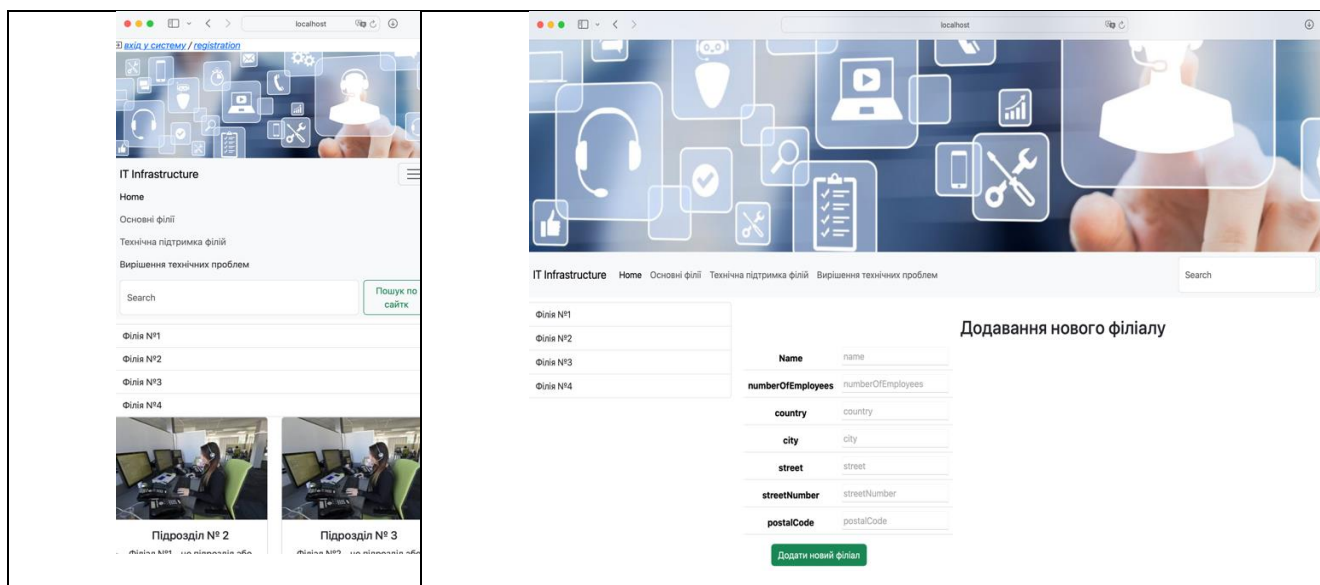


Рисунок – 4. Відображення даних на лептоп та мобільних пристроях

Для входу в систему реалізовано реєстрацію та авторизацію.

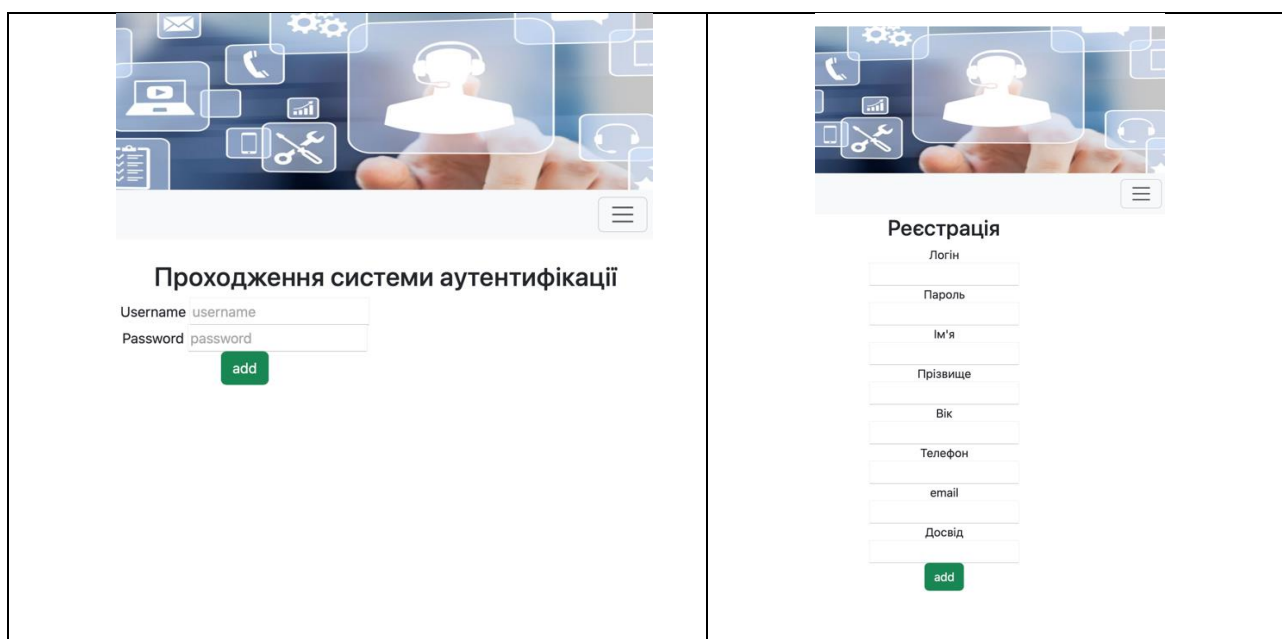


Рисунок – 5. Реалізація системи аутентифікації та реєстрації у системі «IT infrastructure»

Для додавання нових даних щодо обладнання філій нами реалізована форма вводу даних.

IT Infrastructure

Філія №1
Філія №2
Філія №3
Філія №4

Додавання нового обладнання

Name

Description

Manufacturer

BranchOfTheCompany Філіял №2 - це підрозділ або відділення основної компанії, функціонує під її політикою та стандартами. Філія виконує різноманітні функції, залежно від стратегії та потреб компанії.

Додати новий вид обладнання

Список обладнання

Назва	Опис	Виробник	Підрозділ(філія)
Обладнання	ПК	Asus	Філіял №2 - це підрозділ або відділення основної компанії, функціонує під її контролем і відповідно до її політики та стандартів. Філія виконує різноманітні функції, залежно від стратегії та потреб компанії.
ПК 13.3 pro	Apple 13.3	Apple	Філіял №2 - це підрозділ або відділення основної компанії, функціонує під її контролем і відповідно до її політики та стандартів. Філія виконує різноманітні функції, залежно від стратегії та потреб компанії.

Рисунок -6. Закріплення обладнання за відділом(філією) компанії

Інформування про технічні проблеми здійснюється на основі іншої форми. Для фіксування певної проблеми достатньо обрати тип обладнання та зазначити технічну проблему.

IT Infrastructure

Філія №1
Філія №2
Філія №3
Філія №4

Опис технічних проблем обладнання

Опис технічних проблем

Обладнання ПК 13.3 pro

Додати новий вид обладнання

Опис технічних проблем

Назва	Опис проблеми	Підрозділ(філія)
HP 1300	Заміна картриджа	Філіял №2 - це підрозділ або відділення основної компанії, функціонує під її контролем і відповідно до її політики та стандартів. Філія виконує різноманітні функції, залежно від стратегії та потреб компанії.
ПК 13.3 pro	Шлейф	Філіял №2 - це підрозділ або відділення основної компанії, функціонує під її контролем і відповідно до її політики та стандартів. Філія виконує різноманітні функції, залежно від стратегії та потреб компанії.

Рисунок - 7. Опис технічних проблем за основними філіями компанії у середовищі «IT infrastructure»

Результатом заповнення даної форми є формування таблиці, у якій здійснюється відображення інформації щодо зазначеної проблеми з прив'язкою до обладнання, яке, в свою чергу, закріплене за певним відділенням чи філією компанії.

Менеджер може переглядати, контролювати та аналізувати звіти, які містять інформацію про технічну підтримку філій чи відділів компанії.

Для розгортання застосунку «IT Infrastructure» використовується веб-сервер TomCat 10 на OS Linux.

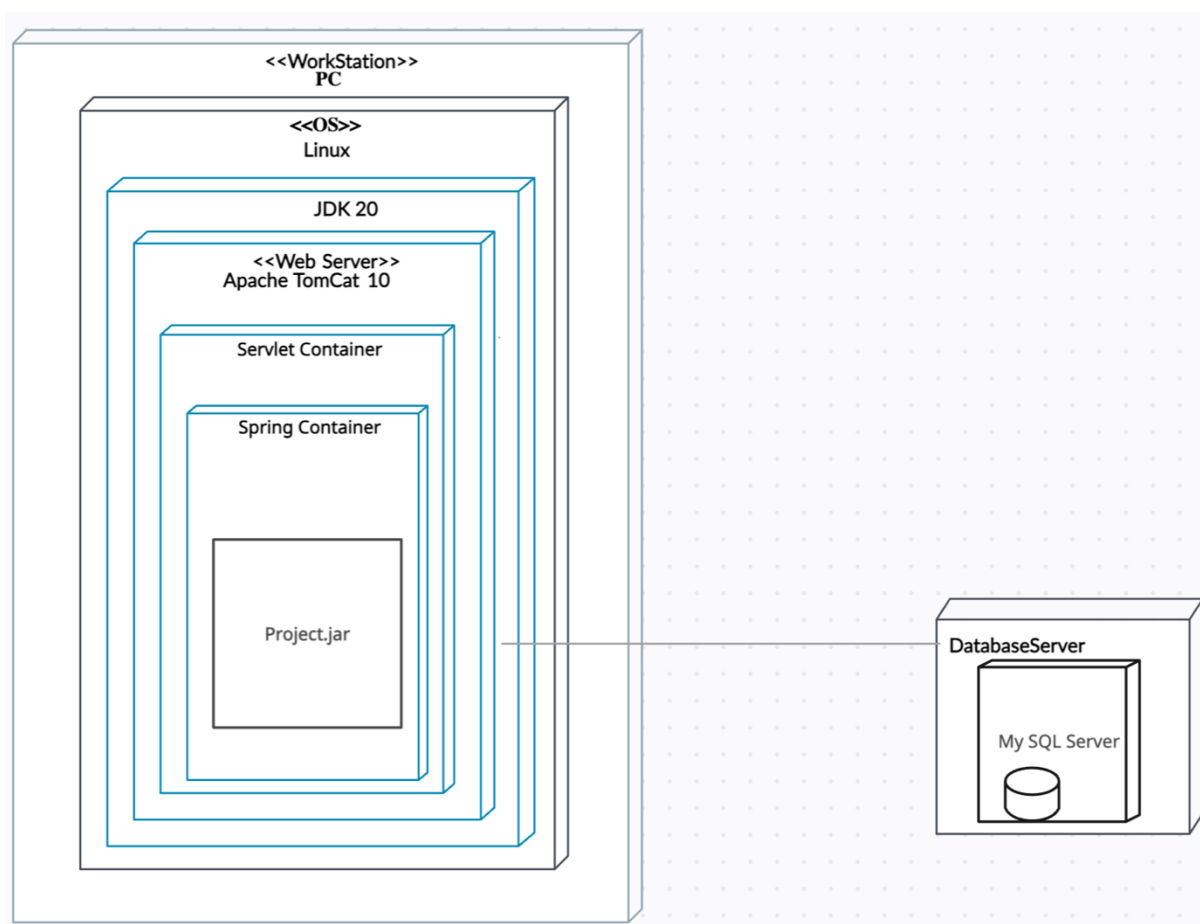


Рисунок - 8. Особливості розгортання проекту застосунку на веб-сервері

ВИСНОВКИ

В кваліфікаційній бакалаврській роботі досліджено питання проектування інформаційної управляючої система сервісної підтримки ІТ інфраструктури АТ "Київгаз".

Досліджено теоретичні відомості щодо ІТ-інфраструктури, визначено основна структура АТ «Київгаз», визначено, що не дивлячись на те, в АТ «Київгаз» розвинута ІТ-інфраструктура, яка підтримується спеціальним програмним забезпеченням сервіс-деск, існує потреба у створенні такої систми.

Впровадження інформаційної системи сервісної підтримки дозволяє підвищити контроль, управління та моніторинг ІТ-інфраструктури. Це сприяє зниженню витрат через оптимізацію ресурсів і підвищення продуктивності.

Покращення якості обслуговування, система забезпечує значне покращення в якості обслуговування, зменшуючи час реагування на заявки та підвищуючи задоволеність користувачів, що в кінцевому підсумку сприяє підвищенню загальної ефективності компанії.

Оптимізація процесів, систематичний аналіз даних заявок дозволяє ідентифікувати та усувати вузькі місця в ІТ-процесах, оптимізувати використання ресурсів і автоматизувати повторювані процеси.

Підтримка прийняття рішень, збір і аналіз даних забезпечує цінну інформацію для підтримки прийняття рішень на вищому рівні, допомагаючи керівництву ефективно розподіляти бюджет і ресурси.

В роботі описана структура і характеристика системи, визначені методи і моделі, які використовують в інформаційній систему управління ІТ-інфраструктурою. Запропоновані рішення щодо проектування бази даних та сховища даних для інформаційної системи управління. Запропонована нова інформаційна система може бути викристана не лише на АТ «Київгаз», а на інших підприємствах з подібною інфраструктурою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IT інфраструктура [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://koeebox.com/ua/terminy-i/it-infrastruktura/>.
2. IT-інфраструктура. Як розібратися в цьому складному понятті і чим вона допомагає в роботі підприємства [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://indevlab.com/uk/blog-ua/it-infrastruktura-yak-rozibratisya-v-tsomu-skladnomu-ponyatti-i-chim-vona-dopomagaye-v-roboti-pidpriyemstva/>.
3. IT інфраструктура [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://koeebox.com/ua/terminy-i/it-infrastruktura/>.
4. Pro vnesennia zmin do deiakykh zakoniv Ukrainy shchodo zabezpechennia kibernetichnoi bezpeky Ukrainy: proekt Zakonu. Verkhovna Rada Ukrainy [Elektronnyi resurs]. Rezhym dostupu: http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_1?id=&pf3511=44208. (Ukr)
5. Чек-ліст надійної IT-інфраструктури: 5 питань, на які важливо відповісти «так» [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://www.sim-networks.com/ukr/blog/it-infrastructure-checklist>.
6. ITIL Управління інформаційними технологіями підприємства [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <http://alpha-com.od.ua/products/obuchauschie-i-informatsionnie/84.php>.
7. Федоренко О. І. 4. АНАЛІЗ СКЛАДНОСТІ ТА ФАКТОРІВ ВИБОРУ VPS ДЛЯ ДЕРЖАВНИХ УСТАНОВ / О. І. Федоренко // Сучасні інформаційні технології та системи в управлінні [Електронний ресурс] : Зб. матеріалів IV Міжнар. наук.-практ. конф. молодих вчених, аспірантів і студентів ; 20–21 квітня 2023 р. / О. І. Федоренко. – Київ: КНЕУ, 2023. – С. 50–52.
8. Dorohyi, Ya. Iu., Mokhor, V. V., Kozliuk, I. O., Tsurkan, V. V. (2015). Krytychna infrastruktura: vrazlyvosti, zahrozy, ryzyky. Tezy dopovidei. II

- mizhnarodna naukovo-praktychna konferentsiia «Informatsiini tekhnolohii ta vzaiemodii», 3-5 lystopada. Kyiv, Pp. 46-47. (Ukr)
9. Mashchenko, E. N., Shevchenko, V. Y. (2013). Yssledovanye protsessov servysnoho obsluzhyvaniya krytycheskykh YT-ynfrastruktur na osnove polumarkovskoi modely. Radioelektronni i kompiuterni systemy, №5. SevNTU. Pp.57-63. (Rus)
 10. Mashchenko, E. N., Shevchenko, V. Y. (2013). Yssledovanye krytycheskykh sytuatsyi v YTyinfrastrukturakh metodamy klasternoho analyza. Radioelektronni i kompiuterni systemy, №5. SevNTU, Pp.191-196. (Rus)
 11. Информационные и коммуникационные технологии в дистанционном образовании: специализированный учебный курс /пер.с англ./ Майкл Г.Мур, Линда Блэк и др. – М: ИД "Обучение-Сервис", 2006. с.632.
 12. Э.В. Жариков. Основные направления оптимизации ИТ-инфраструктуры учебных заведений // Научный журнал "Вестник Восточного Южноукраинского национального университета имени Владимира Даля" – 2011. -№3(157).
 13. О. Балашова, Д. Козлов, Р. Смелянский. ИТ-услуги в вузе: учет стоимости и потребления // Открытые системы» , - 2006. - №5.
 14. Д.В. Логунов. Внедрение сервисного подхода к управлению небольшим ИТ-подразделением (на примере ИТ-службы вуза) [Электронный ресурс] // Электронный журнал "Системы управления бизнес-процессами", - 2011. - MBA СІО, Зима 2011. - Режим доступа: <http://journal.itmane.ru/node/603>.
 15. ITIL® and IT Service Management [Электронный ресурс].- Режим доступа: <http://www.itil.org.uk/>.
 16. Microsoft Operations Framework (MOF) [Электронный ресурс].- Режим доступа: <http://technet.microsoft.com/en-us/solutionaccelerators/dd320379.aspx>.
 17. Service Management: Not Just for IT Anymore [Электронный ресурс].- Режим доступа:

<https://www.thinkhdi.com/topics/research/infographics/~media/HDICorp/Files/Industry-Reports/sm-not-just-for-it-anymore.pdf>

18. Друкер Ф. Корпоративное управление сервисами: как вынести ИТ за рамки ITSM (... и вернуть инновации) [Электронный ресурс].- Режим доступа: <http://www.osp.ru/itsm/2014/02/13039603.html>.
19. Аншина М. Инь и янь ИТ-сервиса [Электронный ресурс].- Режим доступа: <http://www.osp.ru/itsm/2014/03/13040069.html>.
20. Воробьев Д. ITIL вне ИТ и почти без процессов [Электронный ресурс].-. Режим доступа: <http://www.osp.ru/itsm/2013/02/13033945.html>.
21. Skatkov, A. V., Skydan, A. A., Voronyn, D. Iu., Kuznetsova, N. Y. (2013). Matematycheskaia model dlia resheniya zadachy parametrycheskoho synteza v krytycheskykh IT-ynfrastrukturakh. Zbirnyk naukovykh prats SNUiAetaP. SevNTU, Pp. 229-233. (Rus)
22. Надійний хостинг в Україні [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://mirohost.net/ua..>
23. Хмарна інфраструктура для будь-яких задач [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://gigacloud.ua/>.
24. Н Хостинг номер один для Вашего бізнесу [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://hostiko.com.ua/>.
25. Splunk [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: https://www.splunk.com/en_us/download.html?utm_campaign=google_amer_en_search_brand&utm_source=google&utm_medium=cpc&utm_content=free_trials_downloads&utm_term=splunk&_bk=splunk&_bt=656682900952&_bm=e&_bn=g&_bg=43997962767&device=c&gclid=Cj0KCQJw7PCjBhDwARIsANo7CglSqJLwsC2xcQCQWi2Bven1Hhwd5yKUwaUfbv1tzSe9UMB30Jv9qiAaAu1WEALw_wcB.
26. Splunk Acquires SignalFx [Электронный ресурс]. – 2019. – Режим доступа до ресурсу: https://www.splunk.com/en_us/about-splunk/acquisitions/signalfx.html.

27. Modern monitoring & security See inside any stack, any app, at any scale, anywhere. [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://www.datadoghq.com/>.
28. The Total Economic Impact of ScienceLogic SL1 For Capgemini [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://sciencelogic.com/discovery/the-total-economic-impact-of-sciencelogic-sl1-for-capgemini?utm_source=adwords&utm_medium=cpc&utm_adgroupid=149356357393&utm_campaign=NAMER_-_Search_-_Branded&utm_term=sciencelogic&utm_content=654627764846&gclid=Cj0KCQjw7PCjBhDwARIsANo7CgkNqvIPwibh4aJpPwT_Zqg4usGr_puyz8AeYj17gw9zxo94-YmdUMAaAgMdEALw_wcB.
29. Zenoss Service Dynamics [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://www.zenoss.com/resources/zenoss-service-dynamics>.
30. Aytorsung yak instrymnt servisnoi logistuku [Outsourcing as a tool for service logistics]. (n.d.). https://studme.com.ua/1595021014291/logistika/autsorsing_kak_instrument_servisnoy_logistiki.htm. [In Ukrainian].
31. Glushenko, T.M. (2014). Analiz rozvutky logistichnuh poslyg na sychasnomy svitovomy runky [Analysis of the development of logistics services in the modern world market]. Naykovui visnuk Chersonskogo derjavnogo yuniversytety – Scientific Herald of Kherson State University, 6, 169-171. [In Ukrainian].
32. Dovba M.O. Aytorsung na runky logistichnuh poslyg: perspektuvu v Ukraini. [Outsourcing in the market of logistics services: prospects in Ukraine]. (n.d.). file:///C:/Users/danul/Downloads/Nzlubp_2011_7_20.pdf. [In Ukrainian].
33. Zavgorodniy, K.V. (2016). Stanovlennia logistichnogo aytorsingy yak instrymenty ypravlinniy regionalnumu transformaciiymu [Development of logistics outsourcing as a tool for managing regional transformations].

- Naykovui visnuk Chersonskogo derjavnogo yuniversutety – Scientific Herald of Kherson State University, 21, 96-99. [In Ukrainian].
34. Krasnoshapka, V.V. & Trohumets, I.I. (2015). Aytorsung ta jogo zastosyvannia na pidpruiemstvah Ukrainu [Outsourcing and its use at Ukrainian enterprises]. Efektivna ekonomika – Effective economy, 5. (n.d.). <http://www.economy.nayka.com.ua> Retrieved from <http://www.economy.nayka.com.ua/?op=1&z=4097> [In Ukrainian].
35. Logistuka yak instrymet optumizatsii biznes-protseviv ta napryam naykovogo piznannya. [Logistics as a tool for optimizing business processes and the direction of scientific knowledge]. (n.d.). https://pidruchniki.com/71570/logistika/logistika_instrument_optimizatsiyi_biznesprotseviv_napryam_naukovogo_piznannya. [In Ukrainian].
36. Logistuchnui aytorsing. [Logistics outsourcing]. (n.d.). <https://tsl.kname.edu.ua/index.php/en/8-glavnaya/26-logistic-outsourcing>. [In Ukrainian].
37. Osnovni tendentsii zastosyvannia aytorsingy na runky logistuchnuh poslyg v Ykraini. [The main tendencies of outsourcing use in the market of logistic services in Ukraine]. (n.d.). <http://calculino.info/ru/usefull/articles/id208457>. [In Ukrainian].
38. Tishchenko, O. M. (2012). Outsorsing yak factor efektyvnosti pidpruyemstv bydivelnoi galyzi [Outsourcing as a factor in the efficiency of the construction industry]. Problemu ekonomiku – Problems of the economy, 4, 129-135. [In Ukrainian].
39. 2019 third-party logistics study: The State of Logistics Outsourcing. (n.d.). https://dsqapj1larkkc.cloudfront.net/media/sidebar_downloads/2019-3PL-Study.pdf. [United States of America].
40. Percentage of 3PL users who outsource logistics services as of 2018, ranked by the most frequently outsourced services. (n.d.). <https://www.statista.com/statistics/660142/transportation-and-logistics-mostoutsourced-services/>. [United States of America].

- 41.12.Langley, J. Third-Party Logistics Study – 2018. (n.d.).
https://mymaritimeblog.files.wordpress.com/2017/10/3pl_2018_study.pdf.
[United States of America].
- 42.Управление Производительностью Приложений Факторный анализ [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://old.prolan.ru/solutions/technology/apm/factoranalysis.html>.
- 43.Базовый SNMP тест оценки 'здоровья' серверов Linux/UNIX [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://www.you-expert.ru/index.php?KM=299>.
- 44.You-expert [Электронный ресурс]. – 2023. – Режим доступа до ресурсу: <https://www.you-expert.ru/files/299/%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B9%20SNMP%20%D1%82%D0%B5%D1%81%D1%82%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D0%B7%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%D1%8C%D1%8F%20%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BE%D0%B2%20Linux-UNIX.pdf>.

ДОДАТКИ

Додаток А

Код програми

Додаток А.1. Структура моделей (класів-сутностей)

```
package ua.com.itinfrastructure.itinfrastructure.model;

import jakarta.persistence.*;
import lombok.*;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString(of={"country", "city", "street", "streetNumber", "postalCode"})

@Entity
@Table(name = "address")
public class AddressOfCompany {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String country;
    private String city;
    private String street;
    private String streetNumber;
    private String postalCode;
}

package ua.com.itinfrastructure.itinfrastructure.model;

import jakarta.persistence.*;
import lombok.*;

import java.util.ArrayList;
import java.util.List;

@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString(of={"name", "address", "numberOfEmployees"})

@Entity
@Table(name = "branch_of_company")
```

```

public class BranchOfTheCompany {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private int numberOfEmployees;

    @OneToMany(mappedBy="branchOfTheCompanies")
    private List<Equipment> equipments = new ArrayList<>();

    @OneToMany(mappedBy = "branchOfCompanies")
    private List<HardwareProblemsHasBranchOfCompany> hardwareProblemsHasBranch;

    @OneToOne
    @MapKey
    @MapsId
    @JoinColumn(name = "id")
    public AddressOfCompany addressOfCompany;

}

```

```

package ua.com.itinfrastructure.itinfrastructure.model;

```

```

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

```

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

```

```

@Entity
@Table(name = "employees")
public class Employees {

```

```

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;
    private String age;
    private String email;

    private String phoneNumber;
    private String experience;

```

```

    @Transient
    @OneToOne
    @MapKey
    @MapsId
    @JoinColumn(name = "id")
    private UsersOfSystem users;
}

```

```

package ua.com.itinfrastructure.itinfrastructure.model;

```

```

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

```

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

```

```

@Entity
@Table(name = "equipment")
public class Equipment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String description;
    private String manufacturer;

    @ManyToOne
    @JoinColumn(name = "branch_id")
    private BranchOfTheCompany branchOfTheCompanies;

}

```

```

package ua.com.itinfrastructure.itinfrastructure.model;

```

```

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

```

```

import java.util.List;

```

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "hardware_problems")
public class HardwareProblems {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String description;

    @OneToMany(mappedBy = "hardwareProblems")
    private List<HardwareProblemsHasBranchOfCompany>
hardwareProblemsHasBranchOfCompanyList;

}

```

```

package ua.com.itinfrastructure.itinfrastructure.model;

```

```

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

```

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

```

```

@Entity
@Table(name = "problems_has_branch")
public class HardwareProblemsHasBranchOfCompany {

```

```

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

```

```

    @ManyToOne
    @JoinColumn(name = "problems_id")
    private HardwareProblems hardwareProblems;

```

```

    @ManyToOne

```

```

    @JoinColumn(name = "branch_id")
    private BranchOfTheCompany branchOfCompanies;

    @ManyToOne
    @JoinColumn(name = "employee_id")
    private Employees employees;

}

package ua.com.itinfrastructure.itinfrastructure.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;

import java.util.Set;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "roles_of_company")
public class RolesOfCompany implements GrantedAuthority {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nameRoles;

    @Transient
    @ManyToMany(mappedBy = "rolesOfCompany")
    private Set<UsersOfSystem> users;

    @Override
    public String getAuthority() {
        return nameRoles;
    }

    public RolesOfCompany(Long id, String nameRoles) {
        this.id = id;
        this.nameRoles = nameRoles;
    }
}

```

```

package ua.com.itinfrastructure.itinfrastructure.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.Size;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor

@Entity
@Table(name = "users_of_systems")
public class UsersOfSystem implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Size(min = 3, max = 50, message = "має бути більше 3, але менше 50 символів")
    private String login;
    @Size(min = 3, message = "має бути більше 3 символів")
    private String password;

    @ManyToMany(fetch = FetchType.EAGER)
    private Set<RolesOfCompany> rolesOfCompany = new HashSet<>();

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRolesOfCompany();
    }

    @Override
    public String getPassword() {
        return password;
    }

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

    @Override

```

```

    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}

```

Додаток А.2. Структура репозиторію (класів-сутностей)

```

package ua.com.itinfrastructure.itinfrastructure.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import ua.com.itinfrastructure.itinfrastructure.model.AddressOfCompany;

public interface AddressOfCompanyRepository extends JpaRepository<AddressOfCompany, Long>
{
}

```

```

package ua.com.itinfrastructure.itinfrastructure.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import ua.com.itinfrastructure.itinfrastructure.model.BranchOfTheCompany;

public interface BranchOfTheCompanyRepository extends JpaRepository<BranchOfTheCompany,
Long> {
}

```

```

package ua.com.itinfrastructure.itinfrastructure.repository;

import org.springframework.data.jpa.repository.JpaRepository;

```

```
import ua.com.itinfrastructure.itinfrastructure.model.Employees;
```

```
public interface EmployeesRepository extends JpaRepository<Employees, Long> {  
}
```

```
package ua.com.itinfrastructure.itinfrastructure.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import ua.com.itinfrastructure.itinfrastructure.model.Equipment;
```

```
public interface EquipmentRepository extends JpaRepository<Equipment, Long> {  
}
```

```
package ua.com.itinfrastructure.itinfrastructure.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import ua.com.itinfrastructure.itinfrastructure.model.HardwareProblemsHasBranchOfCompany;
```

```
public interface HardwareProblemsHasBranchOfCompanyRepo extends  
JpaRepository<HardwareProblemsHasBranchOfCompany,Long> {  
}
```

```
package ua.com.itinfrastructure.itinfrastructure.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import ua.com.itinfrastructure.itinfrastructure.model.HardwareProblems;
```

```
public interface HardwareProblemsRepository extends JpaRepository<HardwareProblems,Long> {  
}
```

```
package ua.com.itinfrastructure.itinfrastructure.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import ua.com.itinfrastructure.itinfrastructure.model.RolesOfCompany;
```

```
public interface RolesOfCompanyRepository extends JpaRepository<RolesOfCompany, Long> {  
}
```

```
package ua.com.itinfrastructure.itinfrastructure.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;  
import ua.com.itinfrastructure.itinfrastructure.model.UsersOfSystem;
```

```

import java.util.List;

public interface UsersOfSystemRepository extends JpaRepository<UsersOfSystem, Long> {

    UsersOfSystem findByLogin(String username);

    List<UsersOfSystem> findAllByLogin(String login);
}

```

Додаток А.3. Реалізація контролерів (класів-сутностей)

```

package ua.com.itinfrastructure.itinfrastructure.base_controller;

import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import ua.com.itinfrastructure.itinfrastructure.bussiness_logic.EmployeesService;
import ua.com.itinfrastructure.itinfrastructure.bussiness_logic.UsersService;
import ua.com.itinfrastructure.itinfrastructure.model.Employees;
import ua.com.itinfrastructure.itinfrastructure.model.RolesOfCompany;
import ua.com.itinfrastructure.itinfrastructure.model.UsersOfSystem;

import java.util.Collections;

@Controller
@RequiredArgsConstructor
public class UsersController {

    private final UsersService usersService;
    private final EmployeesService employeesService;

    @GetMapping("/login")
    public String getPageLogin() {
        return "login";
    }

    @GetMapping("/registration")
    public String getPageRegistration(Model model) {

        model.addAttribute("usersOfSystem", new UsersOfSystem());
        model.addAttribute("employees", new Employees());
    }
}

```

```

        return "registration";
    }

    @PostMapping("/registration")
    public String saveNewRegistration(@Valid UsersOfSystem users,
                                     BindingResult bindingResult1,
                                     @Valid Employees employees,
                                     BindingResult bindingResult2
    ){

        if (bindingResult1.hasErrors()) {
            return "registration";
        }

        if (bindingResult2.hasErrors()) {
            return "registration";
        }

        if(usersService.getUserLogic(users.getLogin())){
            return "redirect:/registration";
        }

        users.setPassword(new BCryptPasswordEncoder().encode(users.getPassword()));
        users.setRolesOfCompany(Collections.singleton(new RolesOfCompany(1L, "ROLE_User")));

        UsersOfSystem user = usersService.save(users);

        employees.setUsers(user);
        employeesService.saveEmployees(employees);

        return "redirect:/login";
    }
}

```

```

package ua.com.itinfrastructure.itinfrastructure.base_controller;

```

```

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import ua.com.itinfrastructure.itinfrastructure.model.BranchOfTheCompany;
import ua.com.itinfrastructure.itinfrastructure.repository.BranchOfTheCompanyRepository;

@Controller
@RequiredArgsConstructor
public class HomePageController {

```

```

private final BranchOfTheCompanyRepository branchOfTheCompanyRepository;

@GetMapping("/")
public String getPageHome(Model model) {

    model.addAttribute("branchOfTheCompany", branchOfTheCompanyRepository.findAll());

    return "home";
}

}
package ua.com.itinfrastructure.itinfrastructure.base_controller;

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import ua.com.itinfrastructure.itinfrastructure.model.BranchOfTheCompany;
import ua.com.itinfrastructure.itinfrastructure.model.Equipment;
import ua.com.itinfrastructure.itinfrastructure.model.HardwareProblems;
import ua.com.itinfrastructure.itinfrastructure.model.HardwareProblemsHasBranchOfCompany;
import ua.com.itinfrastructure.itinfrastructure.repository.EquipmentRepository;
import ua.com.itinfrastructure.itinfrastructure.repository.HardwareProblemsHasBranchOfCompanyRepo;
import ua.com.itinfrastructure.itinfrastructure.repository.HardwareProblemsRepository;

@Controller
@RequiredArgsConstructor
public class HardwareProblemsController {

    private final EquipmentRepository equipmentRepository;
    private final HardwareProblemsRepository hardwareProblemsRepository;
    private final HardwareProblemsHasBranchOfCompanyRepo hardwareProblemsHasBranchOfCompanyRepo;

    @GetMapping("/hardware-problems")
    public String getHardwareProblems(Model model) {

        model.addAttribute("hardwareProblems",
hardwareProblemsHasBranchOfCompanyRepo.findAll());
        model.addAttribute("equipments", equipmentRepository.findAll() );

        return "hardware-problems";
    }

    @PostMapping("/saveHardwareProblems")
    public String saveHardwareProblems(
//      @RequestParam(name = "name") String name,

```

```

        @RequestParam(name = "description") String description,
        @RequestParam(name = "equipment") String equipment){

    HardwareProblems hardwareProblems = new HardwareProblems();

    hardwareProblems.setName(equipmentRepository.findById(Long.valueOf(equipment)).get().getName());
    hardwareProblems.setDescription(description);

    HardwareProblems hp = hardwareProblemsRepository.save(hardwareProblems);

    HardwareProblemsHasBranchOfCompany hardwareProblemsHasBranchOfCompany = new
    HardwareProblemsHasBranchOfCompany();
    hardwareProblemsHasBranchOfCompany.setHardwareProblems(hp);

    Equipment equipment1 = equipmentRepository.findById(Long.valueOf(equipment)).get();
    BranchOfTheCompany branch = equipment1.getBranchOfTheCompanies();

    hardwareProblemsHasBranchOfCompany.setBranchOfCompanies(branch);

    hardwareProblemsHasBranchOfCompanyRepository.save(hardwareProblemsHasBranchOfCompany);

    return "redirect:/hardware-problems";
}

}

package ua.com.itinfrastructure.itinfrastructure.base_controller;

import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import ua.com.itinfrastructure.itinfrastructure.bussiness_logic.EquipmentService;
import ua.com.itinfrastructure.itinfrastructure.model.AddressOfCompany;
import ua.com.itinfrastructure.itinfrastructure.model.BranchOfTheCompany;
import ua.com.itinfrastructure.itinfrastructure.model.Equipment;
import ua.com.itinfrastructure.itinfrastructure.repository.AddressOfCompanyRepository;
import ua.com.itinfrastructure.itinfrastructure.repository.BranchOfTheCompanyRepository;

@Controller
@RequiredArgsConstructor
public class EquirmentController {

    private final EquipmentService equipmentService;
    private final BranchOfTheCompanyRepository branchOfTheCompanyRepository;
    private final AddressOfCompanyRepository addressOfCompanyRepository;

```

```

@GetMapping("/equirment")
public String getPageEquipment(Model model) {

    model.addAttribute("branch", branchOfTheCompanyRepository.findAll() );
    model.addAttribute("equipments", equipmentService.findAll() );

    return "equipment-page";
}

@PostMapping("/saveNewEq")
public String saveNewEq(@RequestParam(name = "name") String name,
                        @RequestParam(name = "description") String description,
                        @RequestParam(name = "manufacturer") String manufacturer,
                        @RequestParam(name = "branchOfTheCompany") String branchOfTheCompany
                        ) {

    Equipment equipment = new Equipment();
    equipment.setName(name);
    equipment.setDescription(description);
    equipment.setManufacturer(manufacturer);

    equipment.setBranchOfTheCompanies(branchOfTheCompanyRepository.findById(Long.valueOf(branchOfTheCompany)).get());

    equipmentService.save(equipment);

    return "redirect:/equirment";
}

```

```

@GetMapping("/new-branch-of-company")
public String getPageNewBranchOfCompany(Model model) {
    model.addAttribute("branch", branchOfTheCompanyRepository.findAll());
    return "branch_of_company";
}

@PostMapping("/saveNewBranchOfCompany")
public String saveNewBranchOfCompany(
    @RequestParam(name = "name") String name,
    @RequestParam(name = "numberOfEmployees") String numberOfEmployees,

    @RequestParam(name = "city") String city,
    @RequestParam(name = "country") String country,
    @RequestParam(name = "street") String street,
    @RequestParam(name = "streetNumber") String streetNumber,
    @RequestParam(name = "postalCode") String postalCode
) {

```

```

){

    AddressOfCompany addressOfCompany = new AddressOfCompany();
    addressOfCompany.setCity(city);
    addressOfCompany.setCountry(country);
    addressOfCompany.setStreet(street);
    addressOfCompany.setStreetNumber(streetNumber);
    addressOfCompany.setPostalCode(postalCode);

    AddressOfCompany addressOfCompany1 =
addressOfCompanyRepository.save(addressOfCompany);

    BranchOfTheCompany branchOfTheCompany = new BranchOfTheCompany();
    branchOfTheCompany.setAddressOfCompany(addressOfCompany1);
    branchOfTheCompany.setName(name);
    branchOfTheCompany.setNumberOfEmployees( Integer.valueOf(numberOfEmployees));

    branchOfTheCompanyRepository.save(branchOfTheCompany);

    return "redirect:/new-branch-of-company";
}
}

```

Додаток А.4. Налаштування Spring Security

```

package ua.com.itinfrastructure.itinfrastructure.security;

import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfigurat
ion;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import ua.com.itinfrastructure.itinfrastructure.bussiness_logic.UsersService;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfig {

```

```
private final UsersService userService;
```

```
@Bean
```

```
public PasswordEncoder passwordEncoder() {  
    return new BCryptPasswordEncoder();  
}
```

```
@Bean
```

```
public AuthenticationProvider authenticationProvider() {
```

```
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();  
    authProvider.setUserDetailsService(userService);  
    authProvider.setPasswordEncoder(passwordEncoder());
```

```
    return authProvider;  
}
```

```
@Bean
```

```
public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws  
Exception {  
    return config.getAuthenticationManager();  
}
```

```
@Bean
```

```
public SecurityFilterChain authenticationSecurityFilterChain(HttpSecurity http) throws Exception  
{
```

```
    http.csrf((csrf)->csrf.disable())  
        .authorizeHttpRequests((authorize)->  
            authorize.requestMatchers("/",  
                "/login",  
                "/static/**",  
                "/registration",  
                "/registration/**").permitAll()  
            .requestMatchers("/user", "/equirment", "/new-branch-of-company",  
                "/saveNewEq", "/hardware-problems", "/saveHardwareProblems"  
            ).hasRole("User")  
            .requestMatchers("/manager").hasRole("Manager")  
            .requestMatchers("admin").hasRole("Admin")  
            .anyRequest().authenticated()  
        )  
        .formLogin((form)->  
            form.loginPage("/login")  
                .defaultSuccessUrl("/")  
                .permitAll()  
        )  
        .logout((logout)->logout.permitAll().logoutSuccessUrl("/"))  
        .authenticationProvider(authenticationProvider());
```

```

    return http.build();
}

}

```

Додаток А.4. Реалізація шаблону HTML

```

<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">IT Infrastructure</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        <li class="nav-item">
          <a class="nav-link" href="/new-branch-of-company">Основні філії</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/equirment">Технічна підтримка філій</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/hardware-problems">Вирішення технічних
проблем</a>
        </li>
      </ul>
      <form class="d-flex" role="search">
        <input class="form-control me-2" type="search" placeholder="Search" aria-
label="Пошук">
        <button class="btn btn-outline-success" type="submit">Пошук по сайтку</button>
      </form>
    </div>
  </div>
</nav>

<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action">Філія №1</a>
  <a href="#" class="list-group-item list-group-item-action">Філія №2</a>
  <a href="#" class="list-group-item list-group-item-action">Філія №3</a>
  <a href="#" class="list-group-item list-group-item-action">Філія №4</a>
</div>

```

```

<p></p>
<div class="row">
  <div class="col-4">
    <ul class="list-group">
      <li class="list-group-item">Про компанію</li>
      <li class="list-group-item">Філії</li>
      <li class="list-group-item">Адреси</li>
      <li class="list-group-item">Контактні дані</li>
    </ul>
  </div>
  <div class="col-4">
    <ul class="list-group">
      <li class="list-group-item">Філії</li>
      <li class="list-group-item">Технічне забезпечення</li>
      <li class="list-group-item">Вирішення завдань</li>
      <li class="list-group-item">Статистика</li>
    </ul>
  </div>
  <div class="col-4">
    <ul class="list-group">
      <li class="list-group-item">Технічна підтримка</li>
      <li class="list-group-item">Вирішення технічних завдань</li>
      <li class="list-group-item">Пошук</li>
      <li class="list-group-item">Фінанси</li>
    </ul>
  </div>
</div>

```

```

<#macro pages>

```

```

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Templ</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMHjY6hW+ALEwIH"
crossorigin="anonymous">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.3/font/bootstrap-icons.min.css">
</head>
<body>

<div class="controller-fluid">
  <div class="row">
    <i class="bi bi-box-arrow-in-right"> <a href="/login"> вхід у систему</a> / <a

```

```
href="/registration">registration</a></i>
```

```
</div>
```

```
<div class="row">
```

```
<div class="col">
```

```

```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
<div class="col-12">
```

```
<#include "navbar.ftl"/>
```

```
</div>
```

```
</div>
```

```
<div class="row mt-2">
```

```
<div class="col-12 col-sm-3">
```

```
<#include "left.ftl"/>
```

```
</div>
```

```
<div class="col-12 col-sm-9">
```

```
<div class="col registration-form text-center">
```

```
<#nested>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="row">
```

```
<div class="col-12">
```

```
<#include "footer.ftl"/>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
```

```
integrity="sha384-
```

```
YvpcrYf0tY3lHB60NNkmXc5s9fdVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
```

```
crossorigin="anonymous"></script>
```

```
</body>
```

```
</html>
```

```
</#macro>
```