НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Хіміко-технологічний факультет
(повна назва інституту/факультету)
Кафедра кібернетики хіміко-технологічних процесів
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Комп'ютерно-інтегрована система управління ресурсами підприємства»

Виконав (-ла):
студент (-ка) VI курсу, групи ХА-61м
Захарчук Яна Олегівна _____

Науковий керівник:
Доц. каф. кібернетики ХТП,
Бондаренко Сергій Григорович _____

Консультант з розробки робочого проекту:
Керівник відділу аналітики та навчання ТОВ «Е С У»
Тихоліз Олександр Васильович _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Хіміко-технологічний факультет
(повна назва інституту/факультету)
Кафедра кібернетики хіміко-технологічних процесів
(повна назва кафедри)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 151 «Автоматизація та комп'ютерно-інтегровані технології» («Комп'ютерно-інтегровані технології сталих хімічних виробничих комплексів»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____

(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Захарчук Яна Олегівна
(прізвище, ім'я, по батькові)

1. Тема дисертації: «Комп'ютерно-інтегрована система управління ресурсами підприємства»,
науковий керівник дисертації Бондаренко Сергій Григорович, к.т.н., доц.,
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації 22 травня 2018 року

3. Об'єкт дослідження комплекс виробничих процесів підприємства

4. Вихідні дані: структура підприємства, основні інформаційні потоки,

5. Перелік завдань, які потрібно розробити: аналіз структури підприємства, визначення структури комп'ютерно-інтегрованої системи управління ресурсами підприємства, розробка структури бази даних системи, створення програмного

забезпечення для застосування комп'ютерно-інтегрованої системи управління ресурсами підприємства

6. Орієнтовний перелік графічного (ілюстративного) матеріалу

7. Орієнтовний перелік публікацій 3

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Тихоліз Олександр Васильович, Керівник відділу аналітики та навчання ТОВ «Е С У»		

9. Дата видачі завдання 01 вересня 2016 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз структури підприємства	11.09.2016 – 10.10.2016	
2	Пошук літаратурних джерел, постановка задачі для магістерської дисертації	11.09.2016 – 10.10.2016	
3	Виначення структури комп'ютерно-інтегрованої системи	18.11.2016 – 23.01.2017	
4	Розробка бази даних комп'ютерно-інтегрованої системи управління ресурсами підприємства	03.02.2017 – 15.03.2018	
5	Створення програмного забезпечення для застосування системи	21.05.2017 – 15.03.2018	
6	Оформлення пояснювальної записки	15.03.2018 – 10.04.2018	
7	Підготовка доповіді та створення електронної презентації	05.04.2018 – 22.04.2018	
8	Попередній захист магістерської дисертації	11.05.2018	

Студент _____ Я.О. Захарчук
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ С.Г. Бондаренко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Дипломна робота містить 109 с., 2 ч., 20 табл., 65 рис., 1 дод., 27 джерел

КОМП'ЮТЕРНО-ІНТЕГРОВАНА СИСТЕМА, АВТОМАТИЗОВАНА СИСТЕМА, ІНФОРМАЦІЙНА СИСТЕМА, УПРАВЛІННЯ, РЕСУРСИ ПІДПРИЄМСТВА, БАЗА ДАНИХ, СТРУКТУРА СИСТЕМИ

Об'єкт дослідження - комплекс виробничих процесів підприємства.

Предмет дослідження - Комп'ютерно-інтегрована система управління ресурсами підприємства.

Мета роботи – розробити комп'ютерно-інтегровану систему правління ресурсами підприємства, для підвищення прибутковості підприємства і конкурентоздатності за рахунок оперативного вирішення комплексу задач зі збирання, аналізу інформації та за результатами її обробки прийняття раціональних управлінських рішень.

Метою дослідження – розробка інструменту, для прискорення виконання певних видів робіт, наприклад, обробка замовлень, розрахунок фінансових показників, формування звіту з прибутків, зведення балансу.

Система працює в браузері «Google Chrome» і «IE 9 +», доступ до системи здійснюється по протоколу HTTPS. Результатом програми є накопичення, зберігання і переробка інформації.

Програму впроваджено у ТОВ «ESU», що допоможе підприємству забезпечити високу ефективність прийняття рішень, інтегрувати інформаційні процеси, вдосконалювати організацію документообігу підприємства, знизити витрати на інформаційний супровід функціонування підприємства.

РЕФЕРАТ

Дипломная работа содержит 109 с., 2 ч., 20 табл., 65 рис., 1 доп., 27 источников

КОМПЬЮТЕРНО-ИНТЕГРИРОВАННЫЕ СИСТЕМА, АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ, ИНФОРМАЦИОННЫЕ СИСТЕМЫ, УПРАВЛЕНИЕ, РЕСУРСЫ ПРЕДПРИЯТИЯ, БАЗА ДАННЫХ, СТРУКТУРА СИСТЕМЫ

Объект исследования - комплекс производственных процессов предприятия.

Предмет исследования - компьютерно-интегрированная система управления ресурсами предприятия.

Цель работы - разработать компьютерно-интегрированную систему правления ресурсами предприятия для повышения доходности предприятия и конкурентоспособности за счет оперативного решения комплекса задач по сбору, анализу информации и по результатам ее обработки принятия рациональных управленческих решений.

Целью исследования - разработка инструмента для ускорения выполнения определенных видов работ, например, обработка заказов, расчет финансовых показателей, формирование отчета по прибыли, сведения баланса.

Система работает в браузере «Google Chrome» и «IE 9 +», доступ к системе осуществляется по протоколу HTTPS. Результатом программы является накопление, хранение и переработка информации.

Программа внедрена в ООО «ESU», что поможет предприятию обеспечить высокую эффективность принятия решений, интегрировать информационные процессы, совершенствовать организацию документооборота предприятия, снизить затраты на информационное сопровождение функционирования предприятия.

ABSTRACT

The diploma work contains 109 pages, 2 hours, 20 tables, 65 figures, 1 additional, 27 sources

PROGRAM, INFORMATION SYSTEM, MANAGEMENT, ENTERPRISE, AUTOMATED SYSTEM

The object of research is the creation of a complex of production processes of the enterprise.

The purpose of the work is to develop a computer-integrated system of government with the resources of the enterprise, with the help of which the enterprise's profitability and competitiveness will increase due to an operative solution of a set of tasks to collect, analyze information and, based on the results of its processing, make rational management decisions.

The purpose of the study is to develop a tool for accelerating the performance of certain types of work, for example, processing orders, calculating financial indicators, generating a profit report, balance information.

The system works in the browser "Google Chrome" and "IE 9 +", the system is accessed via the HTTPS protocol. The result of the program is the accumulation, storage and processing of information.

The program is implemented in ESU, which will help the company to ensure high decision-making efficiency, to integrate information processes, to improve the organization of the document circulation of the enterprise, to reduce the costs of information support for the operation of the enterprise.

ЗМІСТ

ВСТУП

На основі впровадження сучасної обчислювальної техніки, промислових роботів, верстатів із числовим програмним управлінням, нових технологічних процесів здійснюється технічне переозброєння підприємств, де одним з найважливіших виробничих процесів є обробка інформації. Під час автоматизованої обробки інформації як об'єкт, який піддається перетворенням, виступають різноманітного роду дані, що характеризують ті або інші економічні явища. Такий процес називається технологічним процесом автоматизованої обробки інформації й являє собою комплекс взаємозалежних операцій, що проходить у встановленій послідовності. Тобто, це перетворення вхідної інформації у вихідну з використанням технічних засобів і ресурсів.

Стрімкий розвиток товарних і фінансових ринків в Україні послужив могутнім поштовхом до інтенсивного наростання процесів інформатизації всіх сфер життя суспільства, відповідно росте потреба в розробках автоматизованих систем обробки інформації. Кількісне накопичення подібних розробок супроводжується якісним оформленням і диференціацією ринку інформаційно-технічної продукції. Найбільш очевидною передумовою такого розвитку процесів інформатизації є перехід до використання новітніх інформаційних технологій, які вийшли на якісно новий рівень, що дозволяє без значних капітальних витрат вирішувати складні задачі не тільки в державному масштабі, але й на рівні підприємств, організацій, фірм.

Сучасна інформатизація суспільства потребує побудови сучасних автоматизованих систем управління різними об’єктами. Підвищення економічної ефективності підприємства є першим загальним принципом автоматизації управління.

Цілями автоматизації інформаційних процесів є підвищення продуктивності і ефективності праці працівників, поліпшення якості

інформаційної продукції та послуг, підвищення сервісу та оперативності обслуговування користувачів.

Автоматизація базується на використанні засобів обчислювальної техніки і необхідного програмного обладнання. Вона дозволяє істотно скоротити час обслуговування користувачів, значно підвищити рівень їх обслуговування, перетворює і видозмінює окремі технологічні процеси, а часом - всі основні традиційно використовувані технології.

Правильний вибір або розробка програмних продуктів для автоматизації інформаційних потоків у рамках автоматизованих систем можна вважати найпершим завданням для сучасних організацій. Введення нових безпаперових технологій, що використовують комп'ютер і нові організаційні форми їх застосування, дозволяють підвищити якість і оперативність інформаційного обміну. Підвищення ефективності використання виробничих ресурсів підприємства за допомогою автоматизованої системи управління можливе тільки в тому випадку, якщо управління виробничими процесами на всіх ієрархічних рівнях буде узгоджене.

Досягнення вказаної мети можливе за рахунок виконання системою таких завдань: забезпечення комплексного автоматизованого обліку в усіх сферах діяльності підприємства в межах єдиної інформаційної бази; прогнозування та моделювання діяльності підприємства; створення єдиного інформаційного простору; підвищення оперативності збору, передачі і обробки інформації; поліпшення продуктивності праці персоналу, за рахунок звільнення його від ручних робіт; отримання доступу керівництва підприємства до всіх інформаційних ресурсів системи; поліпшення якості фінансових розрахунків.

1 АНАЛІЗ СКЛАДУ ТА РОБОТИ ІНФОРМАЦІЙНИХ СИСТЕМ ПРИЗНАЧЕНИХ ДЛЯ МОНІТОРИНГУ РЕСУРСІВ ПІДПРИЄМСТВА

1.1 Особливості роботи інформаційних систем різного призначення

Інформаційна система, як система управління, тісно пов’язується, як з системами збереження та видачі інформації, так і з іншої - з системами, що забезпечують обмін інформацією в процесі управління. Вона охоплює сукупність засобів та методів, що дозволяють користувачу збирати, зберігати, передавати і обробляти відібрану інформацію [1].

Інформаційна система (ІС) — це взаємопов'язана сукупність пристроїв, методів і персоналу для обробки інформації.

Інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існує потреба в управлінні. Місією інформаційної системи є виробництво потрібної для організації інформації, потрібної для ефективного управління всіма її ресурсами, створення інформаційного та технічного середовища для управління її діяльністю. Інформаційна система може існувати і без застосування комп’ютерної техніки – це питання економічної необхідності. В будь-якій інформаційній системі управління вирішуються задачі трьох типів:

- задачі оцінки ситуації (деколи їх називають задачами розпізнавання образів);

- задачі перетворення опису ситуації (розрахункові задачі, задачі моделювання);

- задачі прийняття рішень (в тому числі і оптимізаційні).

Автоматизована інформаційна система – це взаємозв’язана сукупність даних, обладнання, програмних засобів, персоналу, стандартних процедур, які призначені для збору, обробки, розподілу, зберігання, представлення інформації у відповідності з вимогами, які випливають з цілей організації. Сьогодні, у вік інформації, практично кожна інформаційна система використовує комп’ютерні

технології, і тому надалі під інформаційними системами надалі будемо підрозумівати саме автоматизовані [2].

Інформаційні системи включають в себе: технічні засоби обробки даних, програмне забезпечення і відповідний персонал. Чотири складові частини утворюють внутрішню інформаційну основу:

- засоби фіксації і збору інформації;
- засоби передачі відповідних даних та повідомлень;
- засоби збереження інформації;
- засоби аналізу, обробки і представлення інформації.

Різноманітність інформаційних систем з кожним роком все зростає. В залежності від функціонального призначення можна виділити такі системи: управляючі (АСУТП, АСУВ), проектуючі (САПР), наукового пошуку (АСНД, експертні системи), діагностичні, моделюючі, систем підготовки прийняття рішення (СППР), а в залежності від сфери використання – на адміністративні, економічні, виробничі, медичні, навчальні, екологічні, криміналістичні, військові та інші [1].

Ручні ІС характеризуються відсутністю сучасних технічних засобів переробки інформації і виконанням всіх операцій людиною [3].

У автоматичних ІС всі операції з переробки інформації виконуються без участі людини.

Автоматизовані ІС передбачають участь в процесі обробки інформації і людини, і технічних засобів, причому головна роль у виконанні рутинних операцій обробки даних відводиться комп'ютера. Саме цей клас систем відповідає сучасному поняттю "інформаційна система".

Залежно від характеру обробки даних ІС діляться на інформаційно-пошукові та інформаційно-вирішальні [5].

Інформаційно-пошукові системи роблять введення, систематизацію, зберігання, видачу інформації по запиту користувача без складних перетворень

даних. (Наприклад, ІС бібліотечного обслуговування, резервування і продажу квитків на транспорті, бронювання місць у готелях і пр.) .

Інформаційно-вирішальні системи здійснюють, крім того, операції переробки інформації за певним алгоритмом. За характером використання вихідної інформації такі системи прийнято ділити на керуючі і радять. Результуюча інформація керуючих ІС безпосередньо трансформується в прийняті людиною рішення. Для цих систем характерні завдання розрахункового характеру і обробка великих обсягів даних. (Наприклад, ІС планування виробництва або замовлень, бухгалтерського обліку) [1].

ІС виробляють інформацію, яка приймається людиною до відома і враховується при формуванні управлінських рішень, а не ініціює конкретні дії. Ці системи імітують інтелектуальні процеси обробки знань, а не даних. (Наприклад, експертні системи).

У залежності від сфери застосування розрізняють такі класи ІС. Інформаційні системи організаційного управління - призначені для автоматизації функцій управлінського персоналу як промислових підприємств, так і непромислових об'єктів (готелів, банків, магазинів та ін.) [6].

Основними функціями подібних систем є: оперативний контроль і регулювання, оперативний облік та аналіз, перспективне і оперативне планування, бухгалтерський облік, управління збутом, постачанням і інші економічні і організаційні завдання.

ІС управління технологічними процесами (ТП) — служать для автоматизації функцій виробничого персоналу з контролю та управління виробничими операціями. У таких системах зазвичай передбачається наявність розвинених засобів вимірювання параметрів технологічних (температури, тиску, хімічного складу тощо), процедур контролю допустимості значень параметрів і регулювання технологічних процесів [1].

ІС автоматизованого проектування (САПР) — призначені для автоматизації функцій інженерів-проектувальників, конструкторів, архітекторів, дизайнерів під час створення нової техніки чи технології. Основними функціями подібних систем є: інженерні розрахунки, створення графічної документації (креслень, схем, планів), створення проектної документації, моделювання проектованих об'єктів [2].

Інтегровані (корпоративні) ІС — використовуються для автоматизації всіх функцій фірми і охоплюють весь цикл робіт від планування діяльності до збуту продукції. Вони включають в себе ряд модулів (підсистем), що працюють в єдиному інформаційному просторі і виконують функції підтримки відповідних напрямів діяльності.

Аналіз сучасного стану ринку ІС показує стійку тенденцію зростання попиту на інформаційні системи організаційного управління. Причому попит продовжує рости саме на інтегровані системи управління.

Замовники ІС стали висувати все більше вимог, спрямованих на забезпечення можливості комплексного використання корпоративних даних в управлінні та плануванні своєї діяльності [3].

Таким чином, виникла нагальна необхідність формування нової методології побудови інформаційних систем.

Мета такої методології полягає в регламентації процесу проектування ІС та забезпеченні управління цим процесом з тим, щоб гарантувати виконання вимог як до самої ІС, так і до характеристик процесу розробки. Основними завданнями, вирішення яких має сприяти методологія проектування корпоративних ІС, є наступні:

- забезпечувати створення корпоративних ІС, що відповідають цілям і завданням організації, а також вимогам, що пред'являються по автоматизації ділових процесів замовника;

- гарантувати створення системи з заданою якістю в задані терміни і в рамках встановленого бюджету проекту;
- підтримувати зручну дисципліну супроводу, модифікації та нарощування системи;
- забезпечувати наступність розробки, тобто використання в розробляється ІС існуючої інформаційної інфраструктури організації (зачепила в області інформаційних технологій) [5].

Впровадження методології повинно призводити до зниження складності процесу створення ІС за рахунок повного і точного опису цього процесу, а також застосування сучасних методів і технологій створення ІС на всьому життєвому циклі ІС - від задуму до реалізації.

Проектування ІС охоплює три основні області:

- проектування об'єктів даних, які будуть реалізовані в базі даних;
- проектування програм, екранних форм, звітів, які будуть забезпечувати виконання запитів до даних;
- облік конкретного середовища або технології, а саме: топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл-сервер або клієнт-сервер), паралельної обробки, розподіленої обробки даних і т.п.

Проектування інформаційних систем завжди починається з визначення мети проекту. У загальному вигляді мета проекту можна визначити як рішення ряду взаємопов'язаних завдань, які включають у себе забезпечення на момент запуску системи і протягом всього часу її експлуатації:

- необхідної функціональності системи та рівня її адаптивності до мінливих умов функціонування;
- необхідної пропускної спроможності системи;
- необхідного часу реакції системи на запит;

- незвідмовної роботи системи;
- необхідного рівня безпеки;
- простоти експлуатації і підтримки системи.

1.2 Структурний аналіз інформаційних систем

Структуру інформаційної системи складає сукупність окремий ее частин - підсістем. Підсистема — це частина системи, яка виділена за певною ознакою. Тому структура будь-якої інформаційної системи может буті представлена як сукупність підсістем, что забезпечують інформаційне, технічне, математичне, програмне, організаційне и правове забезпечення.

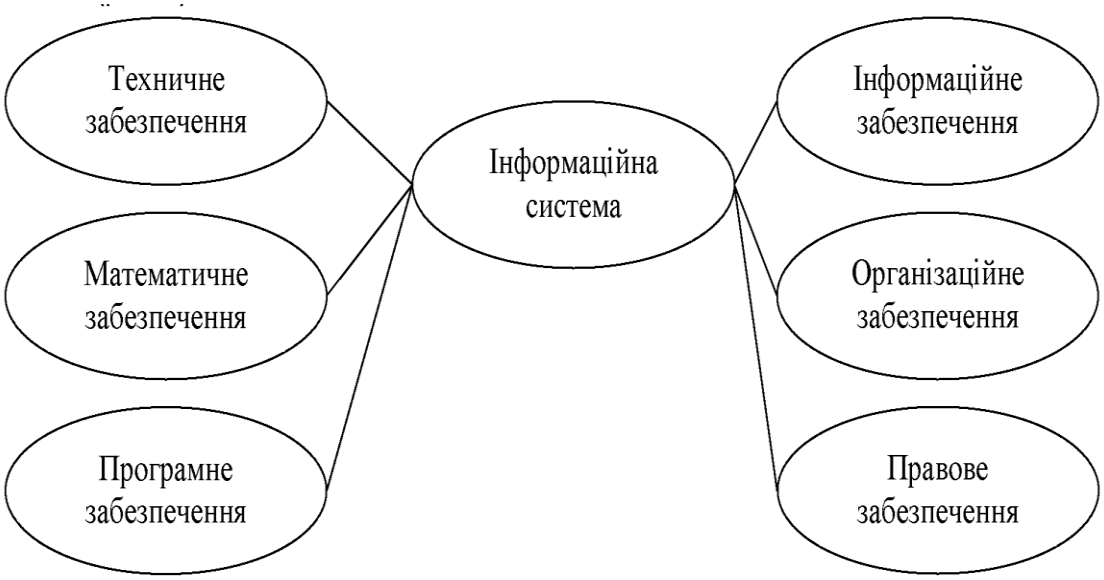


Рисунок 1.1 - Структура інформаційної системи

Інформаційне забезпечення — сукупність єдиної системи класіфікації й кодування повідомлень, уніфікованіх систем документації, схем інформаційних потоків, что ціркулюють в организации, а також методологія побудова баз Даних.

Технічне забезпечення ІС — це комплекс технічних засобів, що забезпечують роботу ІС, відповідної документації на ці засоби і технологічні процеси.

У комплекс технічних засобів входять:

- пристрої збору, накопичення, обробки, передачі та виведення інформації;

- пристрої передачі даних і ліній зв'язку;
- експлуатаційні матеріали та ін.

Документацією оформляються попередній вибір технічних засобів, організація їх експлуатації, технологічний процес обробки даних, технологічне оснащення.

Документацію можна умовно розділити на три групи:

- загальносистемну, що включає державні та галузеві стандарти по технічному забезпеченню;
- спеціалізовану, що містить комплекс методик по всіх етапах розробки технічного забезпечення;
- нормативно-довідкову, використовувану при виконанні розрахунків по технічному забезпеченню.

Математичне і програмне забезпечення - це сукупність математичних методів, моделей, алгоритмів і програм для реалізації цілей і завдань ІС, а також нормального функціонування комплексу технічних засобів.

До засобів математичного забезпечення відносяться:

- засоби моделювання процесів;
- типові завдання;
- методи математичного програмування, математичної статистики, теорії масового обслуговування та ін.

До засобів програмного забезпечення (ПО) відносяться:

Общесистемное ПО — це комплекс програм, орієнтований на користувачів і призначений для вирішення типових завдань обробки інформації. Вони служать для розширення функціональних можливостей комп'ютерів, контролю та управління процесом обробки даних.

Спеціальне ПО являє собою сукупність програм, розроблених при створенні конкретної ІС. До його складу входять пакети прикладних програм, що

реалізують розроблені моделі різного ступеня адекватності, що відображають функціонування реального об'єкта.

Технічна документація на розробку програмних засобів повинна містити опис завдань, завдання на алгоритмізацію, економіко-математичну модель задачі, контрольні приклади.

Організаційне забезпечення — це сукупність методів і засобів, що регламентують взаємодію працівників з технічними засобами і між собою в процесі розробки і експлуатації ІС.

Організаційне забезпечення реалізує наступні функції:

- аналіз існуючої системи управління організацією, де буде використовуватися ІС, і виявлення завдань, що підлягають автоматизації;

- підготовку завдань до вирішення на комп'ютері, включаючи технічне завданя на проектування ІС і техніко-економічне обґрунтування її ефективності;

- розробку управлінських рішень по складу і структурі організації, методології вирішення завдань, спрямованих на підвищення ефективності системи управління. Організаційне забезпечення створюється за результатами передпроектного обстеження на 1-му етапі побудови БД.

Правове забезпечення — це сукупність правових норм, що визначають створення, юридичний статус і функціонування ІС, що регламентують порядок одержання, перетворення і використання інформації. До складу правового забезпечення входять закони, укази, постанови державних органів влади, накази, інструкції та інші нормативні документи міністерств, відомств, організацій, місцевих органів влади. У правовому забезпеченні можна виділити загальну частину, регулюючу функціонування будь-якої ІС, і локальну частину, регулюючу функціонування конкретної системи.

Правове забезпечення етапів розробки ІС включає нормативні акти, пов'язані з договірними відносинами розробника і замовника і правовим регулюванням відхилень від договору.

Правове забезпечення етапів функціонування ІС включає:

- статус ІС;
- права, обов'язки та відповідальність персоналу;
- правові положення окремих видів процесу управління;
- порядок створення і використання інформації і ін.

1.3 Основні етапи розробки та впровадження інформаційної системи

Відповідно до сучасної методології, процес створення ІС являє собою процес побудови і послідовного перетворення низки узгоджених моделей на всіх етапах життєвого циклу (ЖЦ) ІС. На кожному етапі ЖЦ створюються специфічні для нього моделі - організації, вимог до ІС, проекту ІС, вимог до програм і т.д. Моделі формуються робочими групами команди проекту, зберігаються і накопичуються в репозиторії проекту. Створення моделей, їх контроль, перетворення і надання в колективне користування здійснюється з використанням спеціальних програмних інструментів — CASE-засобів.

Процес створення ІС ділиться на ряд етапів (стадій), обмежених деякими тимчасовими рамками і закінчуються випуском конкретного продукту (моделей, програмних продуктів, документації тощо) [5].

Зазвичай виділяють наступні етапи створення ІС:

- формування вимог до системи;
- проектування;
- реалізація;
- тестування;
- введення в дію;
- експлуатація і супровід.

Початковим етапом процесу створення ІС є моделювання бізнес-процесів, що протікають в організації та реалізують її мету та завдання. Модель організації, описана в термінах бізнес-процесів та бізнес-функцій, дозволяє сформулювати основні вимоги до ІС. Це фундаментальне положення методології забезпечує об'єктивність у виробленні вимог до проектування системи. Безліч моделей опису вимог до ІС потім перетвориться в систему моделей, що описують концептуальний проект ІС. Формуються моделі архітектури ІС, вимог до програмного забезпечення (ПЗ) та інформаційного забезпечення (ІЗ). Потім

формується архітектура ПЗ і ІВ, виділяються корпоративні БД і окремі додатки, формуються моделі вимог до програм і проводиться їх розробка, тестування та інтеграція [3].

Метою початкових етапів створення ІС, що виконуються на стадії аналізу діяльності організації, є формування вимог до ІС, коректно і точно відображають цілі та завдання організації-замовника. Щоб специфікувати процес створення ІС, що відповідає потребам організації, потрібно з'ясувати і чітко сформулювати, в чому полягають ці потреби. Для цього необхідно визначити вимоги замовників до ІС та відобразити їх на мові моделей у вимоги до розробки проекту ІС так, щоб забезпечити відповідність цілям і завданням організації.

Завдання формування вимог до ІС є однією з найбільш відповідальних, важко формалізуються і найбільш дорогих і важких для виправлення в разі помилки. Сучасні інструментальні засоби і програмні продукти дозволяють досить швидко створювати ІС по готових вимогам. Але часто ці системи не задовольняють замовників, вимагають численних доробок, що призводить до різкого подорожчання фактичної вартості ІС. Основною причиною такого становища є неправильне, неточне або неповне визначення вимог до ІС на етапі аналізу [1].

На етапі проектування насамперед формуються моделі даних. Проектувальники в якості вихідної інформації отримують результати аналізу. Побудова логічної і фізичної моделей даних є основною частиною проектування бази даних. Отримана в процесі аналізу інформаційна модель спочатку перетвориться в логічну, а потім у фізичну модель даних. Паралельно з проектуванням схеми бази даних виконується проектування процесів, щоб отримати специфікації (опису) всіх модулів ІС. Обидва ці процесу проектування тісно пов'язані, оскільки частина бізнес-логіки зазвичай реалізується в базі даних (обмеження, тригери, збережені процедури). Головна мета проектування процесів полягає у відображенні функцій, отриманих на етапі

аналізу, в модулі інформаційної системи. При проектуванні модулів визначають інтерфейси програм: розмітку меню, вид вікон, гарячі клавіші і пов'язані з ними виклики [2].

Кінцевими продуктами етапу проектування є:

- схема бази даних (на підставі ER-моделі, розробленої на етапі аналізу);
- набір специфікацій модулів системи (вони будуються на базі моделей функцій).

Крім того, на етапі проектування здійснюється також розробка архітектури ІС, що включає в себе вибір платформи (платформ) і операційної системи (операційних систем). У неоднорідній ІС можуть працювати кілька комп'ютерів на різних апаратних платформах і під управлінням різних операційних систем.

Крім вибору платформи, на етапі проектування визначаються такі характеристики архітектури:

- чи буде це архітектура "файл-сервер" або "клієнт-сервер";
- чи буде це 3-рівнева архітектура з наступними шарами: сервер, ПЗ проміжного шару (сервер додатків), клієнтське ПЗ;
- чи буде база даних централізованої або розподіленою. Якщо база даних буде розподіленої, то які механізми підтримки узгодженості та актуальності даних будуть використовуватися;
- чи буде база даних однорідною, тобто, чи будуть всі сервери баз даних продуктами одного і того ж виробника (наприклад, всі сервери тільки Oracle або всі сервери тільки DB2 UDB). Якщо база даних не буде однорідною, яке ПЗ буде використано для обміну даними між СУБД різних виробників (вже існуюче або розроблене спеціально як частина проекту);.
- чи будуть для досягнення належної продуктивності використовуватися паралельні сервери баз даних (наприклад, Oracle Parallel Server, DB2 UDB і т.п.).

Етап проектування завершується розробкою технічного проекту ІС. На етапі реалізації здійснюється створення програмного забезпечення системи, встановлення технічних засобів, розробка експлуатаційної документації. Етап тестування зазвичай виявляється розподіленим в часі [1].

Після завершення розробки окремого модуля системи виконують автономний тест, який переслідує дві основні мети:

- виявлення відмов модуля (жорстких збоїв);
- відповідність модуля специфікації (наявність всіх необхідних функцій, відсутність зайвих функцій).

Після того як автономний тест успішно пройде, модуль включається до складу розробленої частини системи і група згенерованих модулів проходить тести зв'язків, які повинні відстежити їх взаємний вплив.

Далі група модулів тестується на надійність роботи, тобто проходять, по-перше, тести імітації відмов системи, а по-друге, тести напрацювання на відмову. Перша група тестів показує, наскільки добре система відновлюється після збоїв програмного забезпечення, відмов апаратного забезпечення. Друга група тестів визначає ступінь стійкості системи при штатній роботі і дозволяє оцінити час безвідмовної роботи системи. У комплект тестів стійкості повинні входити тести, що імітують пікове навантаження на систему [4].

Потім весь комплект модулів проходить системний тест — тест внутрішньої приймання продукту, що показує рівень його якості. Сюди входять тести функціональності і тести надійності системи.

Останній тест інформаційної системи — приймально-здавальні випробування. Такий тест передбачає показ інформаційної системи замовнику і повинен містити групу тестів, що моделюють реальні бізнес-процеси, щоб показати відповідність реалізації вимогам замовника.

Необхідність контролювати процес створення ІС, гарантувати досягнення цілей розробки і дотримання різних обмежень (бюджетних, тимчасових і ін)

призвело до широкого використання в цій сфері методів і засобів програмної інженерії: структурного аналізу, об'єктно-орієнтованого моделювання, CASE-систем.

Розробка складних ІС неможлива без ретельно обдуманого методологічного підходу. Які етапи необхідно пройти, які методи і засоби використовувати, як організувати контроль за просуванням проекту та якістю виконання робіт - ці та інші питання вирішуються методологіями програмної інженерії.

В даний час існує ряд загальних методологій розробки ІС. Головне в них - єдина дисципліна роботи на всіх етапах життєвого циклу системи, облік критичних завдань і контроль їх вирішення, застосування розвинених інструментальних засобів підтримки процесів аналізу, проектування і реалізації ІС. Для успішної реалізації проекту об'єкт проектування (ІС) повинен бути перш за все адекватно описаний, повинні бути побудовані повні і несуперечливі функціональні та інформаційні моделі ІС.

Для розробки складних інформаційних систем важливий ретельно обдуманий методологічний підхід. Які етапи необхідно пройти, які методи і засоби використовувати, як організувати контроль за просуванням проекту та якістю виконання робіт - ці та інші питання вирішуються методологіями програмної інженерії.

Серед основних вимог, що пред'являються до інформаційних систем, слід зазначити:

1. Ефективність інформаційної системи — визначається зіставленням всіх пов'язаних з розглянутими заходами витрат і одержуваних при цьому результатів.
2. Якість функціонування інформаційної системи — ступінь пристосованості системи до виконання заданих функцій.

Серед основних властивостей, що визначають якість функціонування інформаційної системи, виділяють:

- надійність і своєчасність подання інформації та виконання функціональних технологічних операцій;
- адекватність функціонування інформаційної системи;
- повнота, безпомилковість, актуальність і конфіденційність інформації, що представляється;
- наявність технічних можливостей інформаційної системи до взаємодії, вдосконалення і розвитку.

3. Надійність інформаційної системи визначається надійністю технічних засобів її оснащення та помилками виконавців.
4. Безпека інформаційної системи передбачає таке її функціонування, при якому забезпечується:

- захист інформаційної системи і її об'єктів від несанкціонованого зміни її заданих параметрів і режиму експлуатації;
- захист інформації, що циркулює в цій системі;
- захист користувачів інформаційної системи від шкідливого впливу як інформації, що циркулює в цій системі, так і об'єктів самої системи.

1.4 Застосування баз даних в інформаційних системах

У сучасному інформаційному суспільстві обробка інформації є необхідною умовою організації виробництва. Для прийняття ефективних управлінських рішень виникає необхідність, враховуючи різноманітні фактори, обробляти значну кількість інформації. Стрімке зростання обсягу науково-технічної інформації, з одного боку, і розвиток обчислювальної техніки, з іншого боку, викликали необхідність створення нових інформаційних технологій. Електронні сховища інформації дозволили компактно зберігати, багатократно використовувати, сортувати, редагувати, відображати на екрані дисплея, а при необхідності отримувати копії документів на папері. Такі інформаційні системи забезпечували доступ до інформації великому числу користувачів незалежно від їх географічного місцезнаходження і дозволяли оперувати великими об’ємами даних [8].

База даних (БД) — упорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система управління БД.

Система управління базами даних (СУБД) — це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації.

Централізований характер управління даними в базі даних передбачає необхідність існування деякої особи (групи осіб), на яку покладаються функції адміністрування даними, що зберігаються в базі [13].

Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Існує величезна кількість різновидів баз даних, що відрізняються за критеріями (наприклад, в Енциклопедії технологій баз даних визначаються понад 50 видів БД) [7].

Відзначимо тільки основні класифікації.

Класифікація БД за моделлю даних:

- ієрархічні;
- мережеві;
- реляційні;
- об'єктні;
- об'єктно-орієнтовані;
- об'єктно-реляційні.

Класифікація БД за технологією фізичного зберігання:

- БД у вторинній пам'яті (традиційні);
- БД в оперативній пам'яті (in-memory databases);
- БД у третинній пам'яті (tertiary databases).

Класифікація БД за вмістом:

- географічні;
- історичні;
- наукові;
- мультимедійні.

Класифікація БД за ступенем розподіленості:

- централізовані (зосереджені);
- розподілені.

Окреме місце в теорії та практиці займають просторові (англ. spatial), тимчасові, або темпоральні (temporal) і просторово-часові (spatial-temporal) БД.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий - об'єкти другого рівня і т.д.

Між об'єктами існують зв'язки, кожен об'єкт може включати в себе декілька об'єктів більш низького рівня. Такі об'єкти перебувають у відношенні предка (об'єкт більш близький до кореня) до нащадка (об'єкт більш низького рівня), при цьому можлива ситуація, коли об'єкт-предок не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок. Об'єкти, що мають загального предка, називаються близнюками [10].

Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію.

До основних понять мережевої моделі бази даних відносяться: рівень, елемент (вузол), зв'язок.

Вузол — це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. У мережній структурі кожен елемент може бути пов'язаний з будь-яким іншим елементом.

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то ця модель не є повністю незалежною від програми. Іншими словами, якщо необхідно змінити структуру даних, то потрібно змінити і додаток [8].

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має наступні властивості:

- кожен елемент таблиці - один елемент даних;
- всі осередки в стовпчику таблиці однорідні, тобто всі елементи в стовпчику мають однаковий тип (числовий, символьний тощо);
- кожен стовпчик має унікальне ім'я;
- однакові рядки в таблиці відсутні;

- порядок проходження рядків і стовпчиків може бути довільним.

Об'єктна СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін [11].

Об'єктно-орієнтована база даних (ООБД) — база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. Результатом поєднання можливостей (особливостей) баз даних і можливостей об'єктно-орієнтованих мов програмування є об'єктно-орієнтовані системи управління базами даних (ООСУБД). ООСУБД дозволяють працювати з об'єктами баз даних також, як з об'єктами у програмуванні в об'єктно-орієнтованих мовах програмування. ООСУБД розширює мови програмування, прозоро вводячи довготривалі дані, управління паралелізмом, відновлення даних, асоційовані запити й інші можливості.

Об'єктно-орієнтовані бази даних звичайно рекомендовані для тих випадків, коли потрібна високопродуктивна обробка даних, які мають складну структуру.

Система, яка забезпечує об'єктну інфраструктуру і набір реляційних розширювачів, називається "об'єктно-реляційною".

Об'єктно-реляційні системи поєднують переваги сучасних об'єктно-орієнтованих мов програмування з такими властивостями реляційних систем як множинні представлення даних і високорівневі непроцедурні мови запитів.

За технологією обробки даних бази даних поділяються на централізовані й розподілені [8].

Централізована база даних зберігається у пам'яті однієї обчислювальної системи. Якщо ця обчислювальна система є компонентом мережі ЕОМ, можливий розподілений доступ до такої бази. Такий спосіб використання баз даних часто застосовують у локальних мережах ПК.

Розподілена база даних складається з декількох, можливо пересічних або навіть дублюючих одна одну частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи управління розподіленою базою даних (СУРБД).

За способом доступу до даних бази даних поділяються на бази даних з локальним доступом і бази даних з віддаленим (мережевим) доступом.

Системи централізованих баз даних з мережевим доступом припускають різні архітектури подібних систем:

- файл-сервер;
- клієнт-сервер.

Файл-сервер. Архітектура систем БД з мережевим доступом передбачає виділення однієї з машин мережі в якості центральної (сервер). На такій машині зберігається спільно використовувана централізована БД. Усі інші машини мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувальницької системи до централізованої бази даних. Файли бази даних відповідно до призначених для користувача запитів передаються на робочі станції, де в основному і проводиться обробка. При великій інтенсивності доступу до одних і тих же даних продуктивність інформаційної системи падає. Користувачі можуть створювати також на робочих станціях локальні БД, які використовуються ними монопольно [9].

Клієнт-сервер. У цій концепції мається на увазі, що крім зберігання централізованої бази даних центральна машина (сервер бази даних) повинна забезпечувати виконання основного обсягу обробки даних. Запит на дані, який видається клієнтом (робочою станцією), породжує пошук і вилучення даних на

сервері. Витягнуті дані (але не файли) транспортуються по мережі від сервера до клієнта. Специфікою архітектури клієнт-сервер є використання мови запитів SOL.

2 ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

Підприємство відноситься до сфери телекомунікації та займається такими видами робіт:

- проектування та побудова базових станцій, їх поточне обслуговування та налагодження роботи;
- заміна телекомунікаційного обладняння;
- проектування та встановлення комп'ютерних мереж, забезпечення необхідного обладнаяння для роботи підрозділів;
- роботи по ремонту та встановленню обладнання в офісах.

Структура підприємства:

- транспортно-логістичний та диспетчерський віддли (обслуговування бригад на віїзді, закупка обладнання і т.п.);
- фінансовий відділ;
- відділ системного адміністрування;
- складські відділення;
- відділ зовнішніх відносин по роботі з клієнтами.

Видно, що підприємство має велику структуру. Аналіз показав, що ресурси підприємтва, такі, як обладняння, працівники, складські приміщення і т.п. використовуються не ефективно, тому це все призводить до додаткових витрат підприємства. Саме тому було прийнято рішення розробити, спроектувати та протестувати інформаційну систему для контролю та управління процесами підприємства.

Завдання магістерської дисертації полягає в тому, щоб виконати:

- аналіз структурних зв'язків підприємства;
- визначення структури комп'ютерно-інтегрованої системи управління ресурсами підприємства;

- розробку структури бази даних системи;
- розробку інформаційного забезпечення комп’ютерно-інтегрованої системи управління ресурсами підприємства;
- створення програмного забезпечення для застосування комп’ютерно-інтегрованої системи управління ресурсами підприємства

3 РОЗРОБКА РОБОЧОГО ПРОЕКТУ

3.1 Аналіз структурних зв'язків системи

Перевагою системи автоматизованого контролю і управління ресурсами підприємства є суттєве прискорення виконання певних видів робіт, наприклад, обробка замовлень, розрахунок фінансових показників, формування звіту з прибутків, зведення балансу.

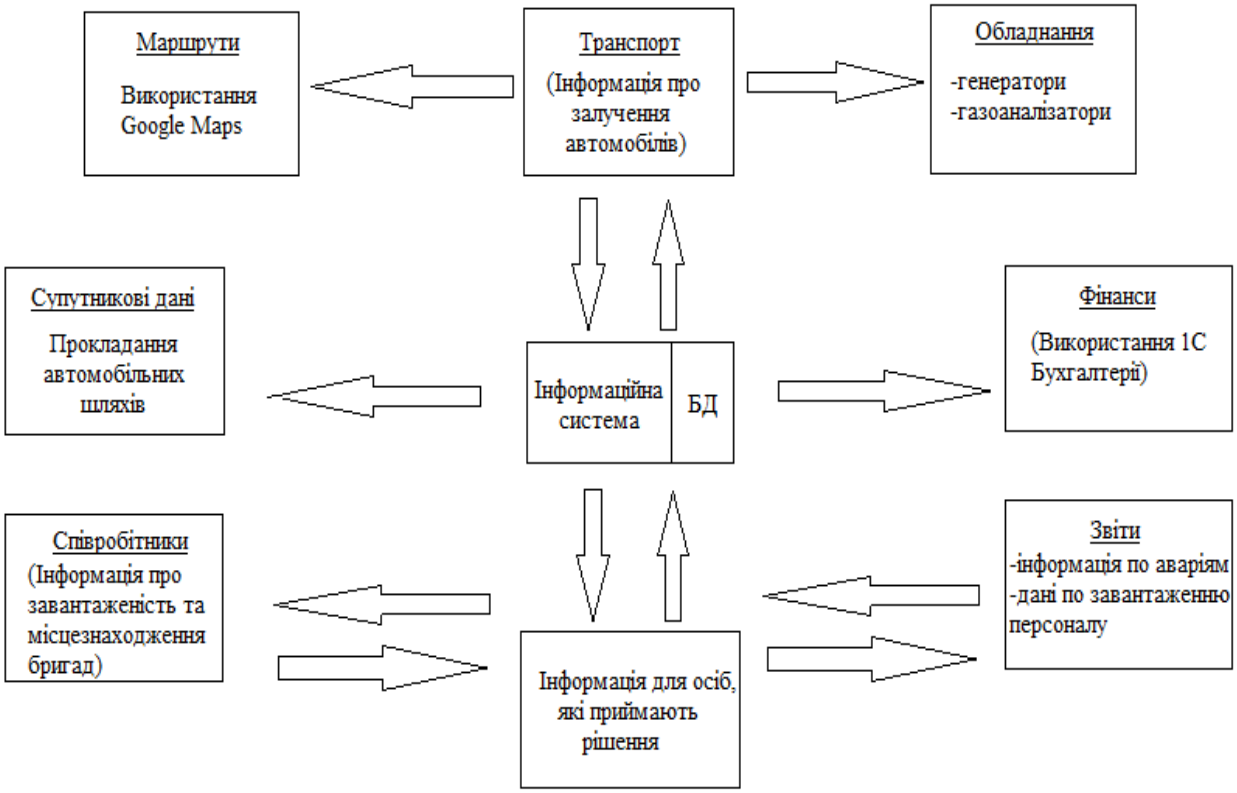


Рис. 3.1. Взаємозв'язок компю'терно-інтегрованої системи управління ресурсами підприємства

Також передбачено: налаштування різних типів заявок і нарядів; облік і супровід наряду по життєвому циклу; вказівка переліку необхідних робіт; вибір оптимальних ресурсів; облік трудовитрат персоналу і устаткування; актуальна і повна інформація про поточний статус робіт за нарядами; автоматичний

контроль планованих і фактичних показників; вивантаження даних в зовнішні системи та багато іншого.

3.2 Створення структури комп’ютерно-інтегрованої системи та її обгрунтування

Основним показником ефективності є можливість прийняття оперативних управлінських рішень на основі повної, достовірної інформації. При цьому скорочується час на виконання рутинних робіт.

Фінансовий ефект полягає в якісному управлінні закупками обладнання, а також у зменшенні виробничих запасів і витрат відповідно до реальних потреб і вивільненні оборотних коштів.

На рисунку 3.2 зображена структура комп’ютерно-інтегрованої системи управління ресурсами підприємства.

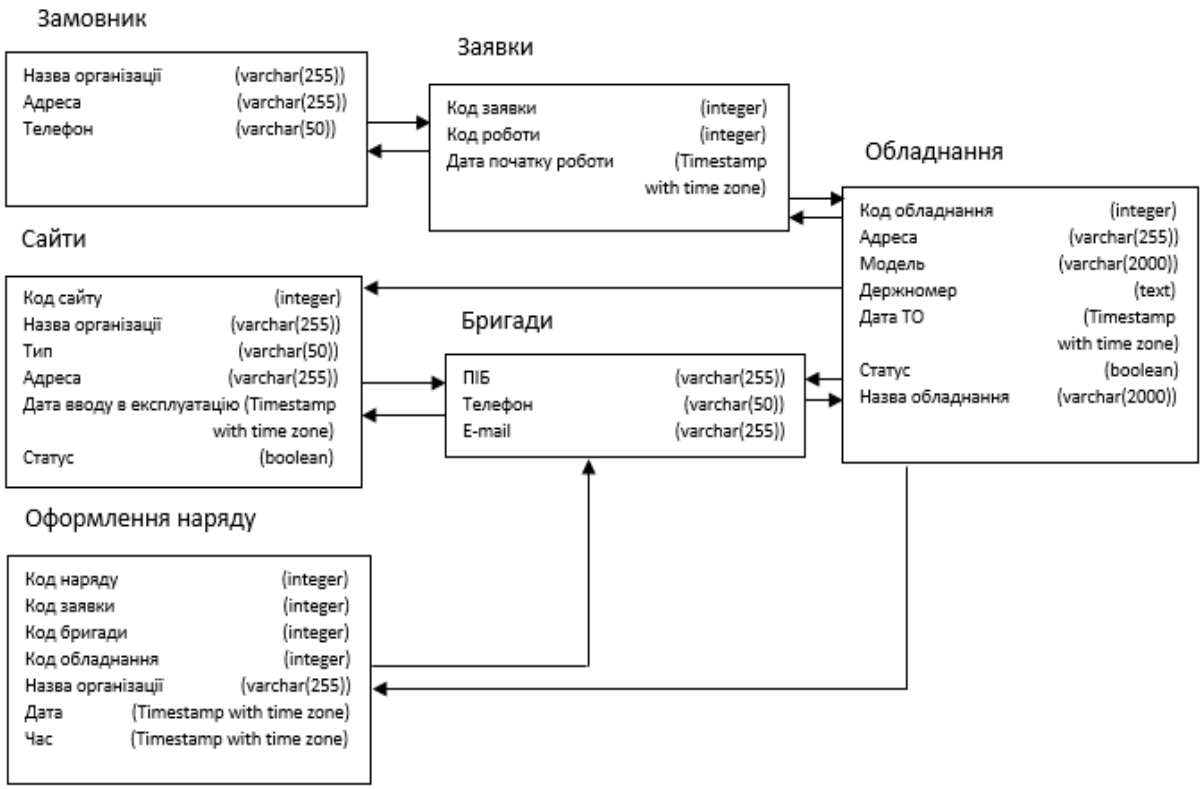


Рисунок 3.2 – Структура комп’ютерно-інтегрованої системи управління ресурсами підприємства

Показана взаємодія між модулями системи та зазначені типи даних, що обрані для введення інформації.

У системі використовуються такі типи даних:

- Integer - служить для представлення цілих чисел;
- Varchar (n) - n символів, від 1 до 32767, строковий тип змінної довжини;
- Boolean - приймає два можливих значення, іноді званих істиною і брехнею;
- Timestamp with time zone – мітки часу;
- Text – служить для вводу тексту.

Одним з основних властивостей інформаційних систем є подільність на підсистеми, яка має переваги з точки зору її розробки і експлуатації:

– спрощення розробки та модернізації інформаційної системи в результаті спеціалізації груп проектувальників по підсистемах;

– спрощення впровадження та постачання готових підсистем у відповідністю з черговістю виконання робіт;

– спрощення експлуатації інформаційної системи внаслідок спеціалізації робітниців предметної області.

Як правило, виділяють функціональні та забезпечуючі підсистеми.

Для опису структури інформаційної системи використовується поняття функціональної підсистеми.

Функціональні підсистеми реалізують і підтримують моделі, методи і алгоритми отримання інформації. Склад функціональних підсистем дуже різноманітний і залежить від предметної області використання інформаційної системи, специфіки господарської діяльності об'єкта, управління.

Функціональні підсистеми інформаційної системи обслуговують певні види діяльності об’єкта моніторингу (в даному випадку системи водопостачання другого контуру атомної електростанції), характерні для його структурних підрозділів та (або) функцій управління.

Інтеграція функціональних підсистем в єдину систему досягається за

рахунок створення і забезпечуючих підсистем, таких як:

– інформаційна;

– технічна;

– програмна;

– математична;

– лінгвістична.

Рисунок 3.3 – Розподіл інформаційної системи на підсистеми

Інформаційне забезпечення – це сукупність засобів і методів побудови інформаційної бази. Воно визначає способи і форми відображення стану об'єкта управління у вигляді даних всередині інформаційної системи, документів, графіків і сигналів поза інформаційною системою. Інформаційне забезпечення поділяють на зовнішнє і внутрішнє [21]. На рисунку 3.4 схематично зображено структуру інформаційного забезпечення.

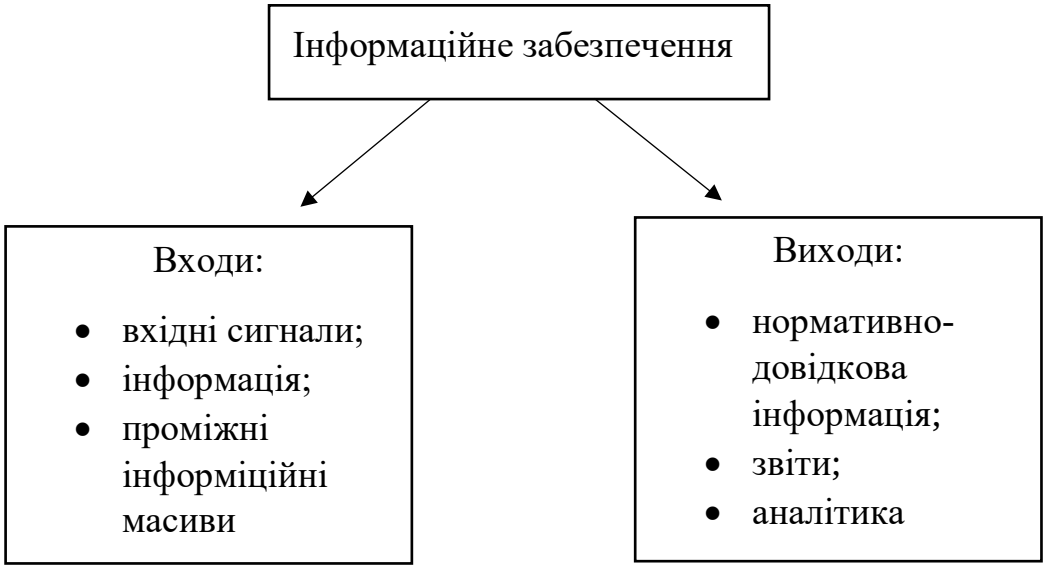


Рисунок 3.3 – Структура інформаційного забезпечення інформаційної системи в загальному вигляді

До входів інформаційного забезпечення відносяться:

– вхідні сигнали та дані – поточні значення показників якості водного режиму другого контуру атомної електростанції, що надходять від датчиків, та попередні значення показників якості, які зберігаються в базі даних.

– проміжні інформаційні масиви – вибірки, що формуються оператором клієнтської станції для аналізу та оцінки стану даного водного об’єкту.

До виходів інформаційного забезпечення спроектованої комп’ютерно-інтегрованої системи відносяться:

– вихідні сигнали і документи – звіти, які завантажуються оператором клієнтської станції на його персональний комп’ютер для подальшого опрацювання.

– нормативно-довідкова інформація – таблиця рекомендацій, до якої відбувається звертання системи в разі, коли поточні значення одного чи кількох

показників якості не відповідають нормативним значенням цих показників.

Математичне забезпечення складається з алгоритмічного і програмного забезпечення. Алгоритмічне забезпечення являє собою сукупність алгоритмів, які використовуються в системі для вирішення завдань і обробки інформації. Програмне забезпечення складається [21]:

– із загального ПО (ОС, транслятори, тести і діагностика та ін., тобто все те, що забезпечує роботу апаратних пристроїв);

– спеціального ПО (прикладне ПО, що забезпечує автоматизацію процесів управління в заданій предметній області).

Схематично математичне забезпечення зображено на рисунку 3.4.

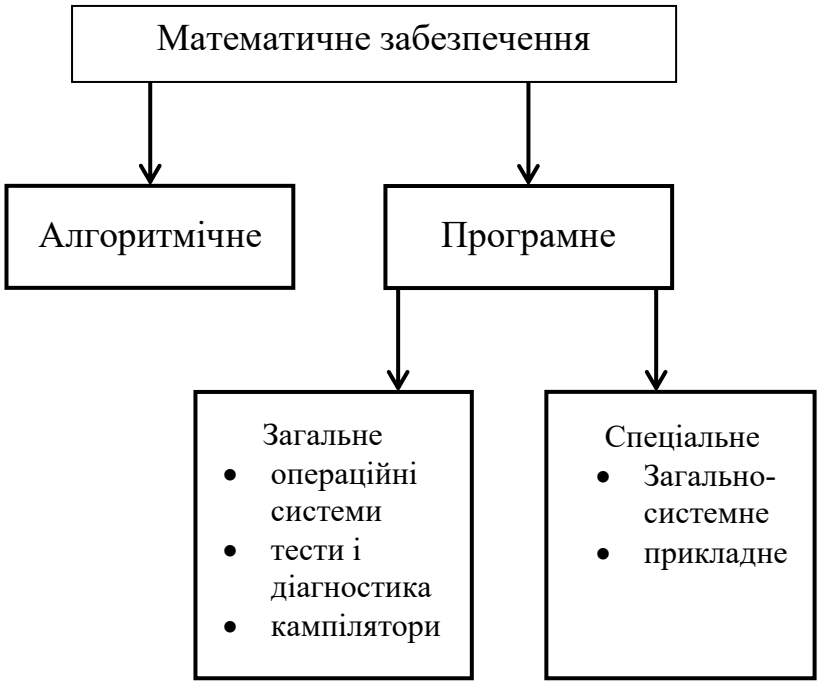


Рисунок 3.4 – Структура математичного забезпечення інформаційної системи в загальному вигляді

3.3 Розробка програмного забезпечення комп’ютерно-інтегрованої системи управління ресурсами підприємства

Мережевий протокол в комп'ютерних мережах — заснований на стандартах набір правил, що визначає принципи взаємодії комп'ютерів в мережі. Протокол також задає загальні правила взаємодії різноманітних програм, мережевих вузлів чи систем і створює таким чином єдиний простір передачі. Хости (будь-який вузол мережі що відправляє або приймає дані через мережу називають хостом (host)) взаємодіють між собою. Для того, щоб прийняти і обробити відповідним чином повідомлення, їм необхідно знати як сформовані повідомлення і що вони означають. Прикладами використання різних форматів повідомлень в різних протоколах можуть бути встановлення з'єднання з віддаленою машиною, відправка повідомлень електронною поштою, передача файлів. Зрозуміло, що різні служби використовують різні формати повідомлень.

Протокол описує:

Формат повідомлення, якому застосунки зобов'язані слідувати;

Спосіб обміну повідомленнями між комп'ютерами в контексті визначеної дії, як, наприклад, пересилка повідомлення по мережі.

Різні протоколи найчастіше описують лише різні сторони одного типу зв'язку й, узяті разом, утворюють стек протоколів. Назви «протокол» і «стек протоколів» також вказують на програмне забезпечення, яке реалізує протоколи.

Типи протоколів:

IP протокол (англ. IP — Internet protocol) — найбільш широко розповсюджена реалізація ієрархічної схеми мережної адресації. Використовуваний в мережі Інтернет, протокол відповідає за адресацію пакетів, але не відповідає за встановлення з'єднань, не є надійним і дозволяє реалізувати тільки негарантовану доставку даних. Термін «протокол без встановлення з'єднань» (англ. connectionless) означає, що протокол для взаємодії не потребує виділеного каналу, як це відбувається під час телефонної розмови і не існує

процедури виклику перед початком передачі даних між мережними вузлами. Протокол IP вибирає найбільш ефективний шлях з числа доступних на основі рішень прийнятих протоколом маршрутизації. Відсутність надійності і негарантована доставка не означає, що система працює погано або ненадійно, а вказує лиш на те, що протокол IP не докладає ніяких зусиль, щоб перевірити чи був пакет доставлений за призначенням. Ці функції делеговані протоколам транспортного та вищих рівнів. Транспортний рівень також відповідає за збірку пакетів у повідомлення в потрібній послідовності.

Transmission Control Protocol (TCP) - один з основних мережевих протоколів Інтернету,призначений для управління передачею даних в мережах і підмережах TCP/IP.

Інформацію, яку потрібно передати, TCP розбиває на порції-сегменти. Кожна порція нумерується, щоб можна було перевірити, чи вся інформація отримана, і розташувати інформацію в правильному порядку. Для передачі цього порядкового номера по мережі у протоколу є свій власний сегмент даних, в якому зокрема написана службова необхідна інформація. Порція ваших даних розміщується в сегмент TCP. Сегмент TCP в свою чергу розміщується в сегменті IP і передається в мережу.

На приймаючій стороні програмне забезпечення протоколу TCP збирає сегменти, витягує з них дані і розташовує їх в правильному порядку. Коли якихось сегментів немає, програма просить відправника передати їх ще раз. Після розміщення всієї інформації в правильному порядку ці дані передаються тій програмі, яка використовує послуги TCP.

В реальній ситуації пакети не тільки загублюються, але й отримують зміни у зв'язку з короткочасними неполадками в лінії передачі. TCP вирішує і цю проблему. При розміщенні даних виробляється так звана контрольна сума. Контрольна сума — це число, яке дозволяє приймаючому TCP виявити помилки в пакеті. Коли пакет прибуває в пункт призначення, приймаючий TCP обраховує

контрольну суму і порівнює її з тою, яку послав відправник TCP. Якщо значення не збігаються, то при передачі виникла помилка. Приймаючий TCP відкидає цей пакет і просить повторну передачу.

TCP/IP — це абревіатура терміну Transmission Control Protocol / Internet Protocol (Протокол керування передачею / міжмережевий протокол). Фактично TCP/IP не один протокол, а декілька. Саме тому ви часто чуєте, як його називають набором, або комплектом протоколів, серед яких TCP і IP — два основних. Фактично TCP/IP представляє цей базовий набір протоколів, відповідальний за розбивання вихідного повідомлення на пакети (TCP), доставку пакетів на вузол адресата(IP) і збирання (відновлення) вихідного повідомлення з пакетів (TCP).

Доменна система імен (англ. Domain Name System, DNS) — розподілена система перетворення імені хоста (комп'ютера або іншого мережевого пристрою) в IP-адресу.

Кожен комп'ютер в Інтернеті має свою власну унікальну адресу — число, яке складається з чотирьох байтів. Оскільки запам'ятовування десятків чи навіть сотень — не досить приємна процедура, то всі (чи майже всі) машини мають імена, запам'ятати які (особливо якщо знати правила утворення імен) значно легше.

Python використовується як засіб створення інструментів і прототипів, системи логіки в іграх, як засіб імпорту/експорту файлів (наприклад COLLADA), автоматизації завдань. Тому у нашому випадку буде доречно використовувати для створення інформаційної системи для автоматизованого контролю та управління ресурсами підприємства мову програмування Python.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та

пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносимість програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно

розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

Програмне забезпечення (застосування або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакунки. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування.

Нижче на рисунку наведений фрагмент коду комп’ютерно-інтегрованої системи управління ресурсами підприємства.

```
Ext.define('portal.view.inventory.equipmentDetails.Common', {
    extend: 'portal.view.AbstractDisplayForm',
    alias: 'widget.equipmentDetailsCommon',
    border: 1,

    autoScroll: true,
    items: [
        {
            xtype: 'fieldset',
            editFormWidget: 'equipInfoEditForm',
            flex: 1,
            title: langGUI.common.commonInfo,
            items: [
                {
                    xtype: 'container',
                    layout: 'hbox',
                    items: [
                        {
                            margin: '10 10 0 10',
                            xtype: 'container',
                            name: 'labelAutoWidthAlign',

                            items: [

                                {
                                    fieldLabel: langGUI.common.titleName,
                                    name: 'name'
                                },
                                {
                                    fieldLabel: langGUI.common.organization,
                                    name: 'org_name',
                                    renderer: function (val, elem) {
                                        if (bDebugMode) {
                                            console.log("%c-- Start renderer ", "background: red; ");
                                        }
                                        if (elem.getForm().getRecord()) {
                                            var record = elem.getForm().getRecord().data;
                                            if (record.org_id) {
                                                val = '<a href="#/organizations/detail/' + record.org_id + '">' + val + '</a>';
                                            }
                                        }
                                        return val;
                                    }

                                },
                                {
                                    fieldLabel: langGUI.common.type,
                                    name: 'device_type_name'

                                },
```

Рисунок 3.5 - Фрагмент коду програми

У даному фрагменті показано створення пункту «Оборудование» системи навігаційного меню. А саме ствоерння боксу (встановлюємо величину відступу від кожного краю елементу, назви заголовків та ін.), на якому буде виводитись запити для пошуку та будуть виводитись результати пошуку.

```
log = init_logger('[DEVICE]')
class DeviceAPI(BaseEntity):
    """ Implement device api """
    DELETE = 'DELETE'
    CREATE = 'CREATE'
    UPDATE = 'UPDATE'
BJTITY_REQ_PARAMS = ( ' name ', ' model_id ', ' is_served ', ' status_id ', ' org_id ',
                ' site_id ' )
ENTITY_FIELDS = ('name', 'description', 'org_id', 'model_id', 'status_id', 'status2_id',
                'date_install', 'notes', 'date_commissioning', 'maintenance_date',
                'maintenance_wo', 'is_served', 'asset_number', 'serial_number',
                'device_code')
DATE = ('date_commissioning', 'date_install', 'maintenance_date')
DEF_S0RT = {'property's 'NAME', 'direction's 'DESC'}
device_type_id = ''
```

Рисунок 3.6 - Фрагмент коду програми

У даному фрагменті представлений ввід основної інформації про обладання, такої як: назва, модель, статус, до якого сату належить обладнання, серійний номер, дату вводу в єксплуатацію та ін.

UPDATE - оператор мови SQL, що дозволяє оновити значення в заданих стовпцях таблиці.

DELETE - в мовах, подібних SQL, DML-операція видалення записів з таблиці. Критерій відбору записів для видалення визначається виразом where. У разі, якщо критерій відбору не визначений, виконується видалення всіх записів.

CREATE - DDL оператор мови SQL, що використовується для створення об'єктів бази даних. У різних СУБД може використовуватися для створення різних об'єктів, наприклад таблиць, схем, уявлень.

3.4 Розробка інформаційного забезпечення комп’ютерно-інтегрованої системи управління ресурсами підприємства

Інформаційна система, яка вирішує завдання оперативного управління підприємством, будується на основі бази даних, в якій фіксується вся можлива інформація про підприємство.

Ми розглянули різні підходи до внутрішньої організації баз даних. І в результаті прийшли до висновку про необхідність використання реляційної моделі, так як вона вирішує одну з основних проблем - внесення змін до бази даних в процесі її використання.

Таблиця 3.1 Порівняння традиційних та реляційних баз даних

Виконувана операція	Традиційні бази даних	Реляційні бази даних
Розробка додатків	Необхідно визначити, яка інформація потрібна різних додатків і створити ряд загальних файлів.	Необхідно визначити види даних, що зберігаються і взаємозв'язку між ними.
Реалізація програм	Дані, що надходять записуються в основні файли; в кожний інформаційний осередок кожного основного файлу записується один елемент даних.	Різні види даних записуються в таблиці даних, що відповідають цим видам. В результаті кожен елемент інформації зберігається в одному єдиному місці.

Продовження таблиці 3.1

Модифікація додатків	Потрібен перегляд структури бази даних з подальшим перезаписом основних файлів, які порушені змінами, що вносяться, і з переробкою всіх додатків, що використовують ці файли.	Досить знайти і модифікувати таблицю, в якій повинно міститися визначення нового виду даних. Самі дані зберігаються в інших таблицях, які не торкаються при подібних змінах.
Внесення часткових змін в дані	Необхідно прочитати кожен основний файл з початку до кінця, модифікуючи змінювані осередки даних і залишаючи всі інші прочитані осередки без змін.	У відповідних таблицях досить виділити безліч рядків, в які необхідно внести зміни, та провести ці зміни за допомогою одного SQL-оператора.

Отже, основні риси реляційних баз даних:

- Структура реляційної бази даних визначається зберігаються в них даними і не фіксується в момент завершення розробки (тобто є гнучкою і масштабованої).
- Структурам даних можна давати вельми інформативні назви.
- Дані зберігаються в єдиному екземплярі; всі опції читання і модифікації даних виробляються тільки з цим екземпляром даних, що якісно полегшує синхронізацію даних між багатьма додатками і користувачами.
- Дані зберігаються відповідно до чітко визначеними і строго дотримуваними правилами.

- Вибір системи управління для корпоративної бази даних - один з ключових моментів в розробці інформаційної системи. На Українському ринку присутні практично всі СУБД, що належать до елітного класу - Oracle, Informix, Sybase, MS SQL Server. Питання, яку СУБД використовувати, можна вирішити тільки за результатами попереднього обстеження і отримання інформаційних моделей діяльності компанії.
- Сервер ORACLE забезпечує ефективні і дієві рішення для основних засобів баз даних:
- - ORACLE підтримує найбільші бази даних, потенційного розміру до сотень гігабайт. Щоб забезпечити дієвий контроль за використанням дорогих дискових пристроїв, він надає повний контроль розподілу простору.
- - ORACLE підтримує велику кількість користувачів, що одночасно виконують різноманітні додатки, які оперують одними і тими ж даними. Він мінімізує суперництво за дані і гарантує узгодженість даних.
- - На деяких установках ORACLE працює 24 години на добу, не маючи періодів розвантаження, що обмежують пропускну здатність бази даних. Нормальні системні операції, такі як відкат бази даних, а також часткові збої комп'ютерної системи, не переривають роботу з базою даних.
- Було проведено порівняння розробленої інформаційної системи з іншими існуючими системами, в тому числі, які з'явилися за останній час. Результати порівняння показали, що застосування наукового підходу при розробці системи дозволило забезпечити відчутний відрив від конкурентів.
- Серед існуючих програмних рішень з автоматизації інформаційні системи більше мають автоматизований документообіг, це стало приводом для створення системи, в якій також враховується детальна інформація про обладнання, його місцезнаходження; GPS-навігація, що дає змогу

прокладати потрібний маршрут для обслуговування об’єктів; можливість додавання фото з об’єкту та багато іншого, що дозволить використовувати систему у різних галузях.

- Аналіз досліджень дозволив визначити ймовірні ефекти від впровадження інформаційної системи та представлені в таблиці 3.1:

Табл. 3.2 - Можливий ефект від впровадження інформаційної системи

Сутність завдань	Очікуваний ефект
1. Підвищення ефективності планування та обліку, і, як наслідок, підвищення ефективності використання виробничих потужностей, можливість оптимізації складських запасів.	Підвищення продуктивності праці в 2-3 рази, економія оборотних коштів.
2. Автоматизація завдань планування виробництва.	Зниження трудомісткості виписки маршрутних листів виконавцями (в 1,5-2 рази).
3. Створення єдиної системи виробничого обліку.	Отримання повної і достовірної інформації про хід виконання нарядів, фактичні витрати в будь-який момент часу.
4. Автоматизація завдань отримання зведеної аналітичної звітності по виробництву.	Швидке отримання необхідних звітних документів, виконання планових завдань, фактичний виробіток робітників, фактичні витрати за станом на будь-який момент часу.

Продовження таблиці 3.2

5. Перехід на використання передових інформаційних технологій і методів управління виробництвом.	Підвищення продуктивності праці.

Інформаційною складовою комп'ютерно-інтегрованої системи управління ресурсами підприємства є база даних, що забезпечує не лише зберігання первинних даних про підприємство, а й надає легкий доступ для отримання необхідної інформації, порівняння результатів та створення звітів.

В ході аналізу предметної області було виявлено, що необхідно розробити структуру бази даних, що дозволить зберегти наступну інформацію: про співробітників підприємства; про замовників, з якими працює підприємство; про заявки, що подаються замовниками; про послуги які надаються підприємством (види робіт); про оформлення нарядів на обслуговування заявок; про обладнання, що використовується в ході виконання заявок за нарядами.

База даних розроблена у середовищі Oracle Database 12c Enterprise Edition Release 12.2.0.1.0. Вона дозволяє оперативно заносити отримані дані, довготривало їх зберігати, виконувати обробку та аналіз цих даних та надавати доступ до бази даних у віддаленому режимі.

Бази даних (БД) створюються спеціально для зберігання, обробки, проведення розрахунків, сортування, вибірки та подання будь-яких масивів даних по будь-яким критеріям. Подібні бази даних здатні зберігати найрізноманітнішу інформацію, в даному випадку у вікні системи навігаційного меню пункту «Оборудование» буде зберігатись інформація про наявність обладнання, його використання та ін.

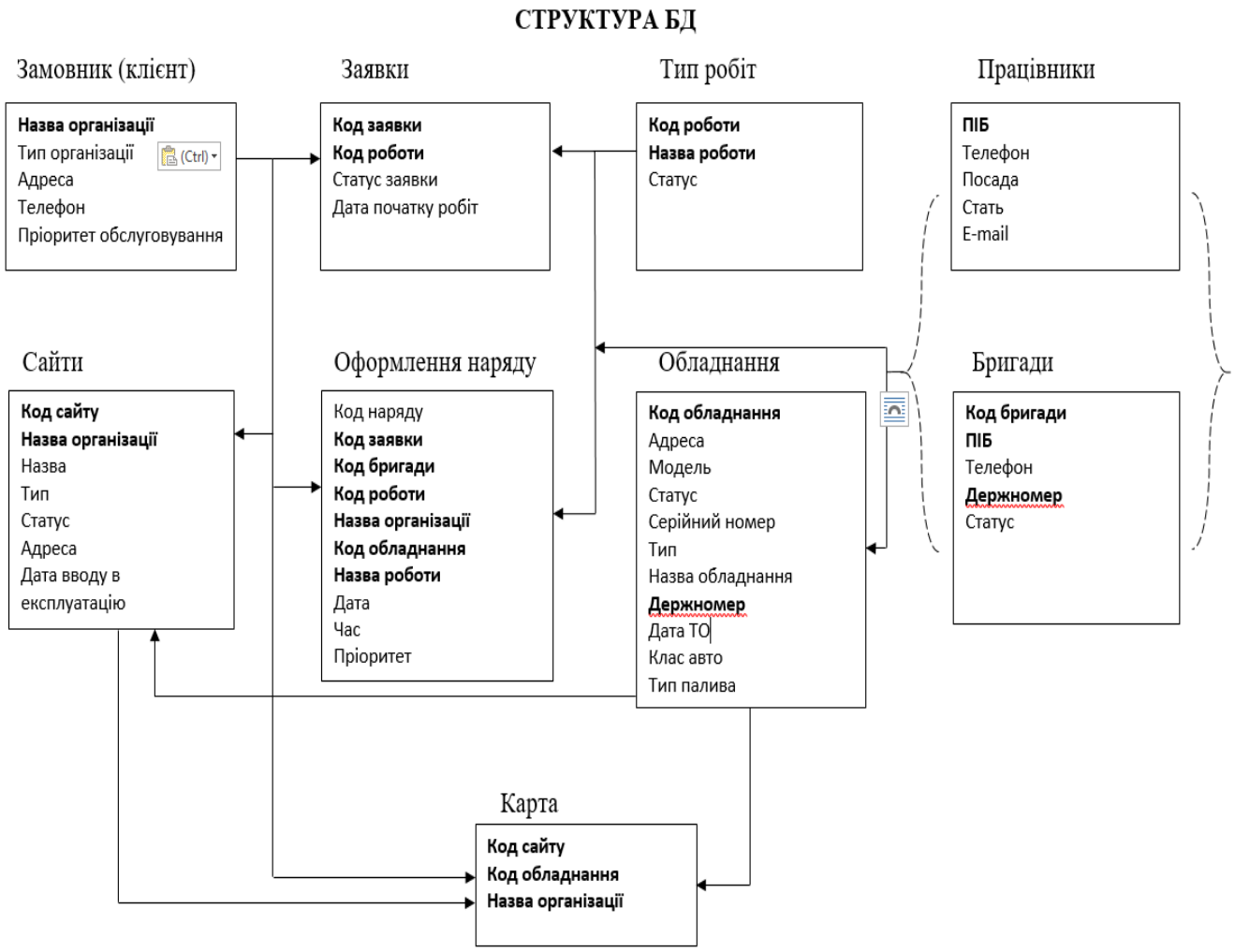
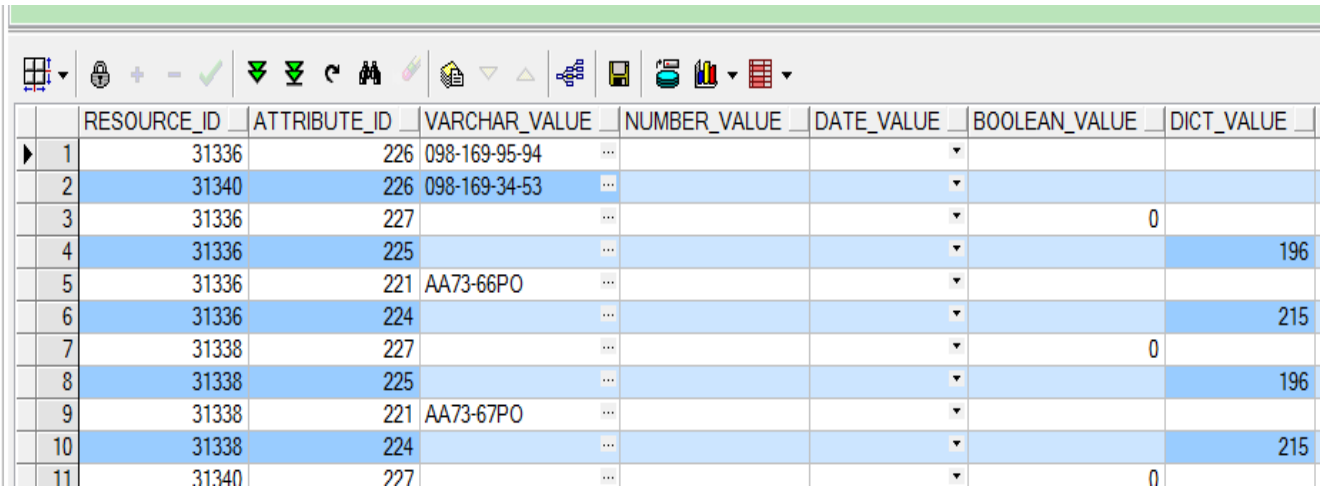


Рисунок 3.7 – Структура Бази Даних (БД)

Всі залежні поля пов'язані на рівні бази даних, що дозволяє реалізувати контроль цілісності, в тому числі каскадне оновлення. Деякі поля загальних і системних таблиць не використовуються, так як ця система є частиною програмного комплексу.



	RESOURCE_ID	ATTRIBUTE_ID	VARCHAR_VALUE	NUMBER_VALUE	DATE_VALUE	BOOLEAN_VALUE	DICT_VALUE
1	31336	226	098-169-95-94				
2	31340	226	098-169-34-53				
3	31336	227				0	
4	31336	225					196
5	31336	221	AA73-66PO				
6	31336	224					215
7	31338	227				0	
8	31338	225					196
9	31338	221	AA73-67PO				
10	31338	224					215
11	31340	227				0	

Рисунок 3.8 – Фрагмент БД

На рисунку 3. 4 можна побачити фрагмент бази даних (БД), де видно ID (номер) кожного обладнання. Якщо це машина, то її держномер, якщо в якості обладнання виступає мобільний телефон, то буде представлений його номер.

"Id" - унікальний ідентифікатор користувача. Це поле використовується в якості первинного ключа.

3.5 Архітектура системи керування підприємством

Архітектура системи автоматизованого контролю і управління ресурсами підприємства наведена на рисунку 3.7. Рожевим кольором виділені внутрішні модулі системи, а синім кольором зображені можливі зовнішні інформаційні системи для інтеграції.

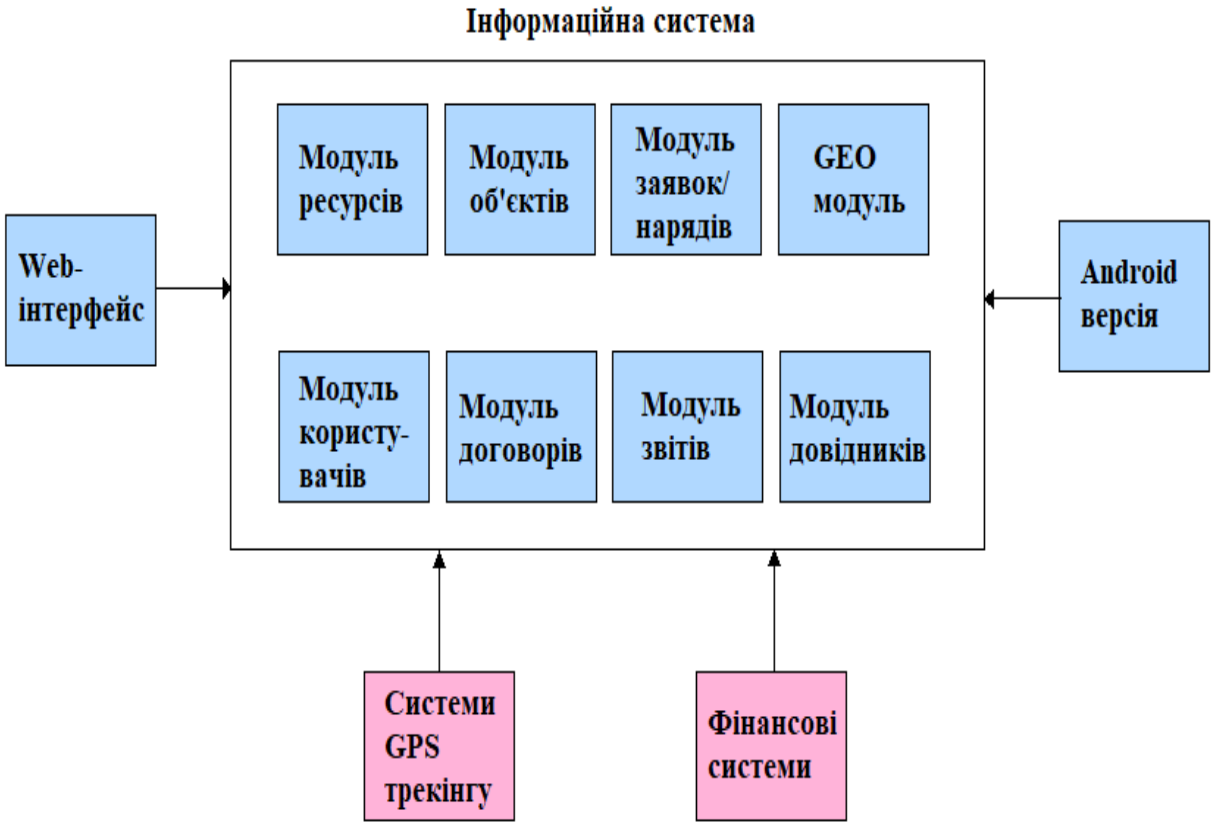


Рисунок 3.9 – Архітектура комп’ютерно-інтегрованої системи

WEB-інтерфейс – модуль, що забезпечує доступ користувачів до WEB-інтерфейсу системи.

Android версія – додаток для мобільного Android терміналу. Використовується бригадами для роботи з нарядами.

Модуль об'єктів – ведення об'єктів (майданчики, сайти, обладнання), облік контрактів по майданчику.

Модуль ресурсів – ведення ресурсів (автомобілі, генератори, майданчики, обладнання і т.д.).

GEO модуль – забезпечує функціонал відображення майданчиків і автомобілів на мапі, геокодування і розрахунку маршрутів руху.

Модуль заявок/нарядів – забезпечує роботу з заявками та нарядами. Заявка може створюватися вручну системним диспетчером або автоматично на основі отриманих даних. Заявки та наряди мають відповідний життєвий цикл.

Модуль користувачів – модуль, який зберігає інформацію по користувачах і їх ролям, забезпечує функціонал додавання користувачів, призначення ролей, а також аутентифікацію і авторизацію користувачів.

Модуль довідників – надає адміністратору системи функціонал редагування довідкової інформації (моделі обладнання, роботи за нарядами і т.д.).

Модуль договорів – облік договорів на оренду, розміщення, електроживлення і т.д.

Модуль звітів – використовується для побудови звітів системи, також використовується для генерації друкованих форм різних об'єктів.

Системи GPS трекінгу – модуль забезпечує взаємодію з обладнанням, оснащеним GPS-передавачами для відстеження переміщення ресурсів (наприклад, автомобілів).

Фінансові системи – модуль взаємодії з фінансовими системами (наприклад, 1С), забезпечує імпорт і експорт об'єктів (ресурси, звіти за нарядами).

3.6 Економічний аналіз ефективності впровадження комп'ютерно-інтегрованої системи управління ресурсами підприємства

Головний економічний ефект від впровадження комп'ютерно-інтегрованої системи управління ресурсами підприємства полягає в поліпшенні економічних показників роботи підприємства, в першу чергу за рахунок підвищення оперативності управління та зниження трудовитрат на реалізацію процесу управління, тобто скорочення витрат на управління. Економічний ефект виступає у вигляді економії трудових і фінансових ресурсів, одержуваної від:

- зниження трудомісткості розрахунків;
- зниження трудовитрат на пошук і підготовку документів;
- економії на витратних матеріалах (папір, дискети, картриджі);
- скорочення службовців підприємства.

Зниження ж трудовитрат на підприємстві можливо за рахунок автоматизації роботи з документами, зниження витрат на пошук інформації.

Для розрахунку витрат на етапі проектування необхідно визначити тривалість кожної роботи, починаючи зі складання технічного завдання і закінчуючи оформленням документів.

Тривалість робіт визначається або за нормативами (при цьому застосовують спеціальні таблиці), або розраховують їх на підставі експертних оцінок за формулою:

$$T0 = (3 * Tmin + 2 * max) / 5 \quad (1)$$

де T0 - очікувана тривалість робіт;

Tmin і Tmax ~ відповідно найменша і найбільша на думку експерта тривалість роботи.

Дані розрахунків очікуваної тривалості робіт наведені в таблицю.

Таблиця 3.3 - Таблиця тривалості робіт на етапі проектування

Назва робіт	Тривалість робіт, днів		
	мінімум	максимум	Очікуван.
Розробка технічного завдання	14	31	21
Аналіз технічного завдання	10	19	15
Вивчення літератури	20	60	35
Робота з джерелами	20	60	35
Оформлення технічного завдання	7	21	14
Розробка алгоритма	120	315	180
Допрацювання програми	10	120	70
Налагодження програми	25	60	31

Таблиця 3.4 - Таблиця робіт користувачів

№ п/п	Вид робіт	До системи, хв	Економія часу, хв
1.	Ввід інформації	40	20
2.	Проведення розрахунків	5	4
3.	Підготовка та друк звітів	30	15
4.	Аналіз та вибірка даних	44	10

Витрати на розробку програмного продукту розраховуються по сліду-нього формулою:

$$K_{РПР} = З_{ФОТР} + З_{ОВФ} + З_{ЭВМ} + З_{СПП} + З_{ХОН} + Р_{Н}, \quad (2)$$

де $З_{ФОПР}$ - загальний фонд оплати праці розробників ПП;

$З_{ОВФ}$ - нарахування на заробітну плату розробників ПП у внебюджетні фонди;

$З_{ЕВМ}$ - витрати, пов'язані з експлуатацією техніки;

$З_{СПП}$ - витрати на спеціальні програмні продукти, необхідні для розробки ПП;

$З_{ГОП}$ - витрати на господарсько-операційні потреби (папір, література, носії інформації і т.п.);

$Р_Н$ - накладні витрати (РН = 30% від ЗФОТР).

При розробці програмного продукту загальний час розробки склало 6,5 місяця. З них машинний час (безпосередня робота з обчислювальною та оргтехнікою) становить 6,41 міс.

Фонд оплати праці за час роботи над програмним продуктом:

$$З_{ФОТР} = \sum_{j=1}^{m} O_{Pj} \cdot T_{РПР_j} \cdot (1+k_Д)(1+k_У), \quad (3)$$

де $О_{Pj}$ - оклад j-го розробника. У розробці брали участь 15 людей, оклад кожного становить 15 000 тис. грн .;

$Т_{РПРj}$ - загальний час роботи над системою в місцях, $Т_{РПР}$ = 6,5 міс .;

- коефіцієнт додаткової зарплати, = 0,2:

- районний коефіцієнт,.

Таким чином,

$З_{ФОПР}$ = 25 000 * 6,5 * (1 + 0,2) * (1 + 0,15) = 224 250 тис. грн. (4)

Критерієм ефективності створення і впровадження нових засобів автоматизації є очікуваний економічний ефект. Він визначається за формулою:

$Е = Е_р - Е_н * К_п$ = 3 322 400 - 0,15 * 1 760 000 = 3 058 400 млн грн. (5)

де $Е_р$ - річна економія;

$Е_н$ - нормативний коефіцієнт ($Е_н$ = 0.15);

$К_п$ - капітальні витрати на проектування і впровадження, включаючи початкову вартість програми.

Річна економія $E_р$ складається з економії експлуатаційних витрат і економії у зв'язку з підвищенням продуктивності праці користувача. Таким чином, отримуємо:

$E_р = (P_1-P_2) + \Delta P_п = (5\ 254\ 900 – 2\ 656\ 000) + 723\ 500 = 3\ 322\ 400$ млн грн. (6)

де P1 і P2 - відповідно експлуатаційні витрати до і після впровадження розроблюваної програми;

$\Delta P_п$ - економія від підвищення продуктивності праці додаткових користувачів.

Економічний ефект від впровадження ІС складається з двох складових. По-перше, це підвищення якості обробки інформації (непряма ефективність), по-друге, ефект, одержуваний за рахунок зниження трудомісткості виконання роботи. Отже, наша система дозволить скоротити за рік на 3 058 400 млн грн.

4 ВИКОРИСТАННЯ СИСТЕМИ

Система працює в браузері «Google Chrome» і «IE 9 +», доступ до системи здійснюється по протоколу HTTPS. Після переходу на сторінку системи, користувачеві відображається вікно входу в систему (рисунок 3.3), де слід вказати свій логін і пароль, й після чого відкриється сторінка з інтерфейсом системи.

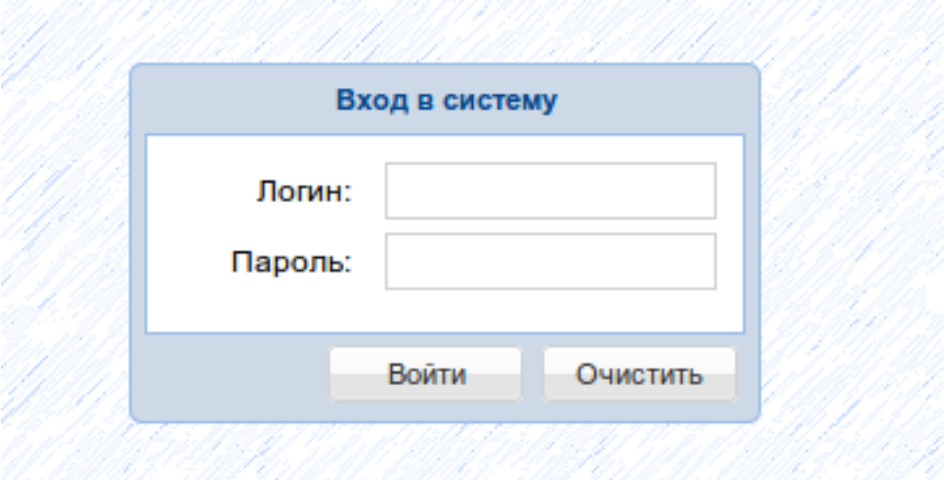


Рисунок 4.1 – Вікно входу в систему

Після відкриття вікна системи навігаційному меню Начни работу с є можливість отримати доступ до розділів Ресурсы (Сайты, Ключи, Оборудование, Заявка, Карта), Клиенты (Организации, Сотрудники, Бригады, Договора), Отчеты (Сайты, Заявки АВР, Пользователи), Администрирование (Группы, Процессы, Задачи, Планировщик, Динамические параметры, Дополнительные справочники, Администрирование). Доступ до певних розділів надається адміністратором системи.

Як приклад розглянемо пункт Работа с оборудованием важливого елементу системи – Модуль объектов за допомогою якого відбувається ведення об'єктів (майданчиків, сайтів, обладнання), облік контрактів по майданчику. Об'єкт може бути як локальним і знаходитись в системі, так і бути експортованим із зовнішніх

систем. Для роботи з цим елементом слід обрати у вікні системи навігаційному меню пункт Работа с оборудованием (рисунок 3.4).

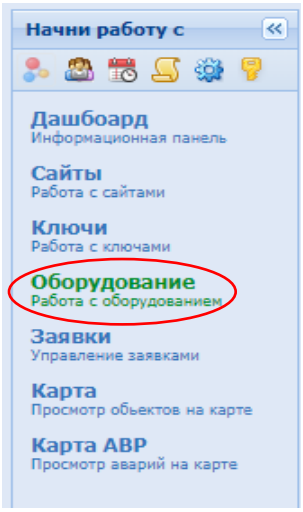


Рисунок 4.2 - Навігаційне меню

Далі слід обрати критерії пошуку, які були сформовані в процесі роботи над системою, із тих, що запропонує система. Можна здійснювати пошук як по одному, так і за кількома критеріями. Якщо жоден з критеріїв не вибрано. Показується весь список обладнання.

При натисканні на кнопку «Знайти» в нижній частині пошукової форми відображається список обладнання, що підходять під всі вибрані критерії пошуку.

При натисканні на кнопку «Очистити» значення всіх критеріїв пошуку очищається.

Для вивантаження обраного обладнання в файл використовується кнопка «Експорт в файл».

Сформований «csv» файл вивантажується з системи і зберігається на локальному комп'ютері користувача або відкривається програмою, яка призначена для перегляду файлів такого типу (в залежності від налаштувань конкретного робочого місця).

Для створення нового обладнання слід перейти на сайт, для якого додається обладнання, і в картці сайту перейти в закладку «Обладнання» і натиснути кнопку «Додати».

Редагування атрибутів обладнання можливо з картки обладнання, яка відкривається при подвійному натисканні на рядку обладнання на закладці «Обладнання» картки сайту або в пошуковій формі обладнання.

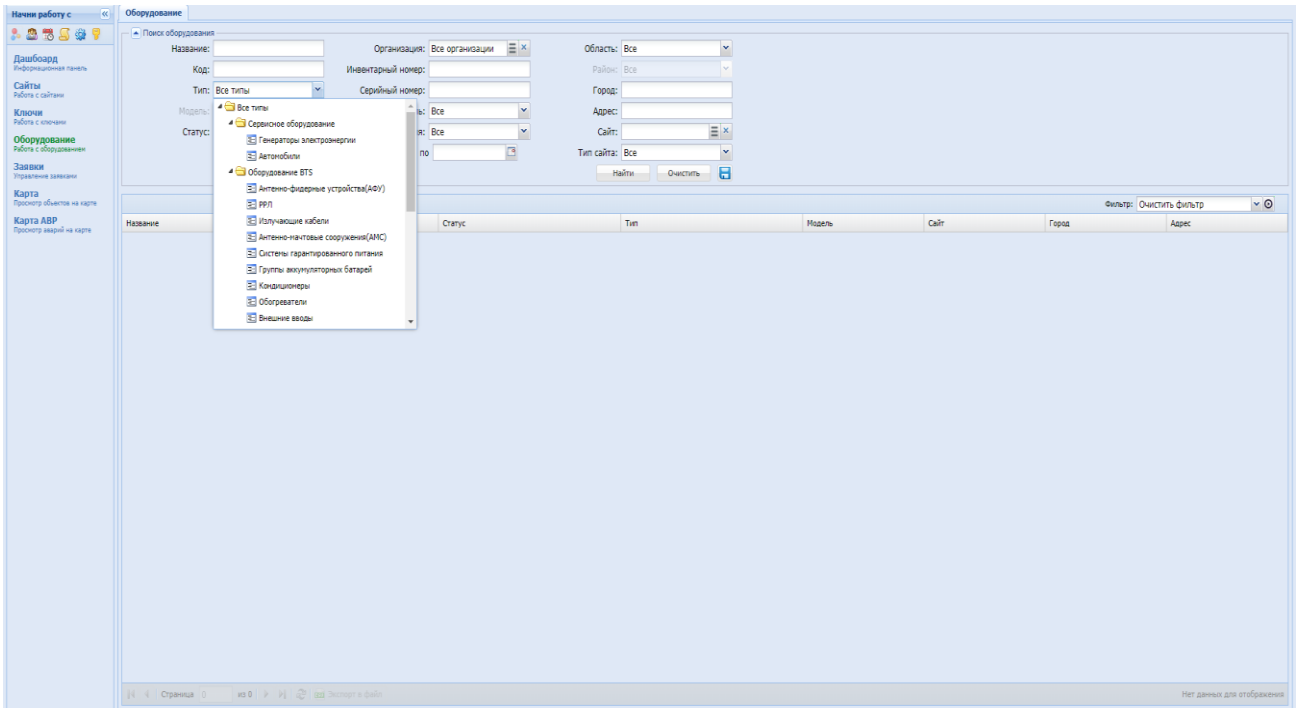


Рисунок 4.3 – Картка підпункту «Обладнання»

Після відкриття підпункту відкривається головна сторінка з пошуком обладнання. Багато критеріїв для пошуку, де можна обрати тип обладнання (генертори електроенергії, кондиціонери, лічильники електроенергії, обігрівачі, антенні споруди, автомобілі та ін.), модель обладнання, організація, до якої підпорядковується обладнання, серйний номер, дату початку/кінця експлуатації, область та місто, у якому нам потрібно знайти обладнання, також присутній пошук по адресі. У даному випадку на рисунку 4 показано розгорнуто критерій пошуку «Тип обладнання», де можна задати потрібний нам вид обладнання зы списку. На рисунку 5 показаний пошук груп акумуляторних батарей, задавши потрібні нам критерії пошуку натискаємо кнопку «Найти». На рисунку нижче наведений результат пошуку.

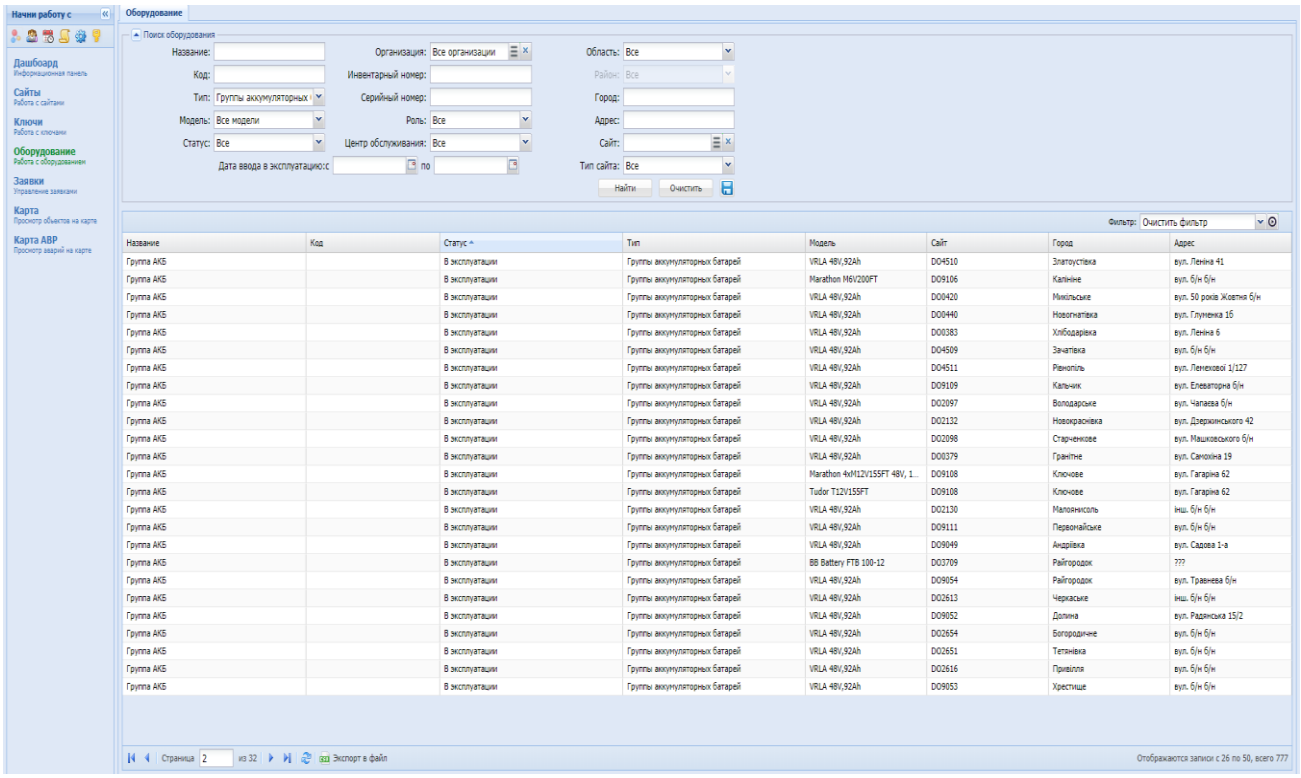


Рисунок 4.4 – Картка підпункту «Обладнання»

На рисунку 6 зображений пошук іншого типу обладняння, такого як Базові станції. Натиснувши «Найти» можна побачити всі знайдені результати. Також видно статус обладнання, модель, до якого сайту належить певне обладнання, місто, у якому знаходиться та точну адресу. Гортати сторінки можна за допомогою стрілочок у нижньому лівому куті сторінки. Є можливість експорту знайдених даних в файл та присутнє сортування знайденої інформації по стовбцям.

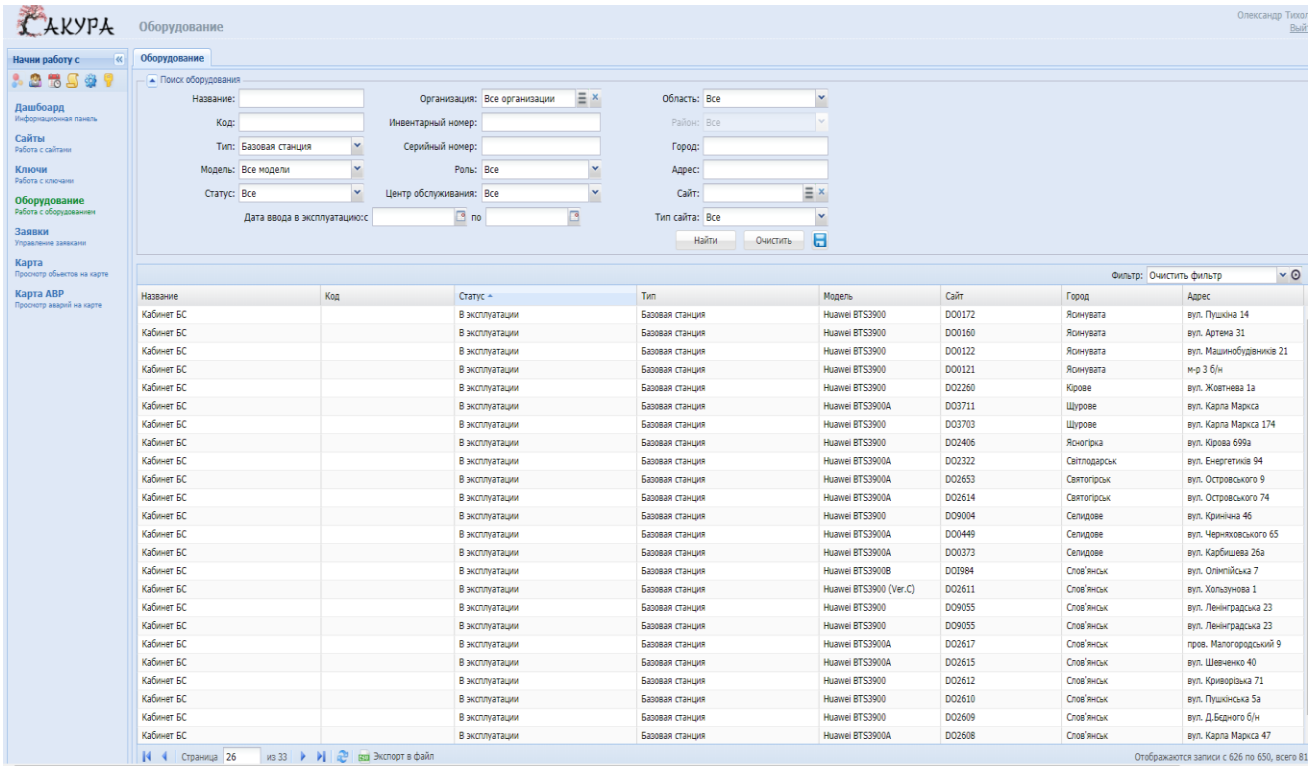


Рисунок 4.5 – Результат пошуку інформації по заданим критеріям пошуку на вкладці «Обладнання»

Обравши потрібне нам обладнання подвійним натиском відкривається додаткова вкладка, де можна побачити більш детальну інформацію. На рисунку 73.11 зображений рисунок з детальною інформацією про Генератор електроенергії. Можна побачити назву обладнання, тип, код, інвентарний номер, назву сайту та організацію, до якого дане обладнання належить, а також присутня інформація про роль обладнання (обслуговується чи приймає участь в обслуговуванні).

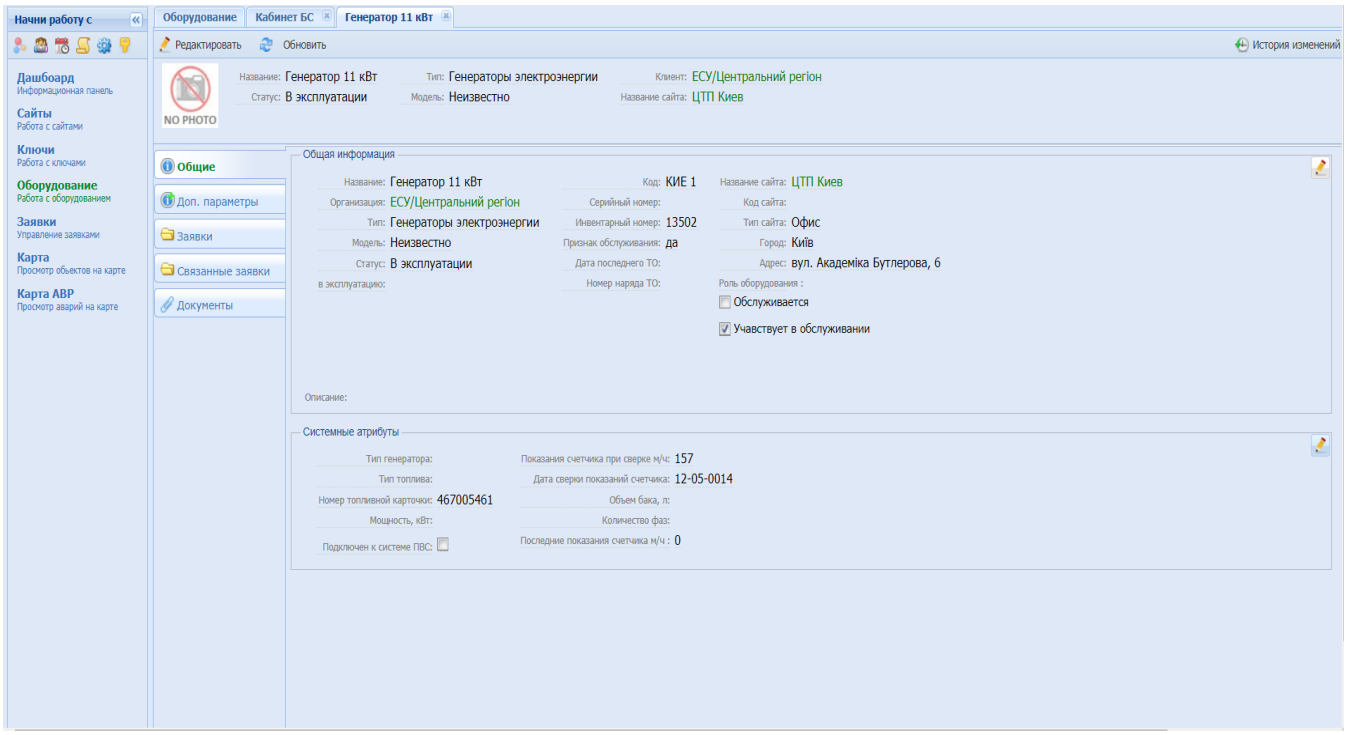


Рисунок 4.6 – Картка підпункту «Обладнання-Загальне»

На рисунку нижче зображена інформація про тип обладнання «Кондиціонер».

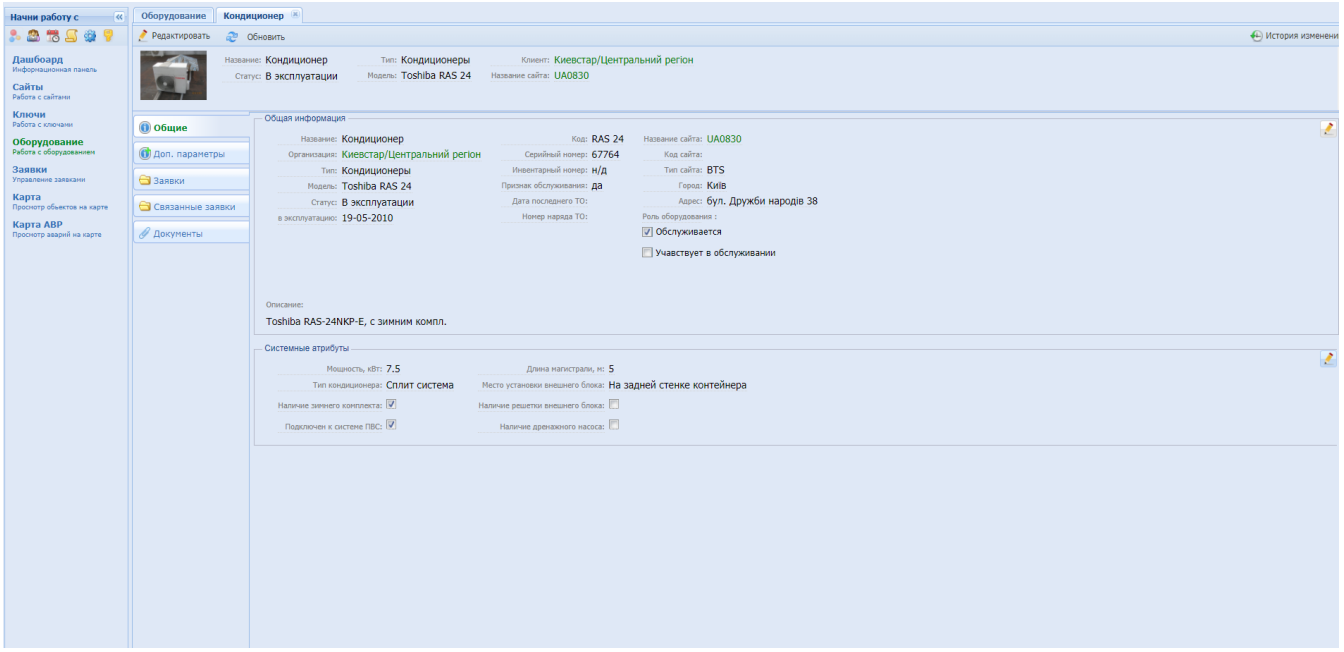


Рисунок 4.7 – Картка підпункту «Обладнання-Загальне»

Відкривши вкладку «Обладнання-Загальні заявки» будуть представлені всі заявки, у яких приймло участь обране обладнання. Натиснувши на заявку, буде доступна більш детальна інформація про заявку та участь обладнання у ній.

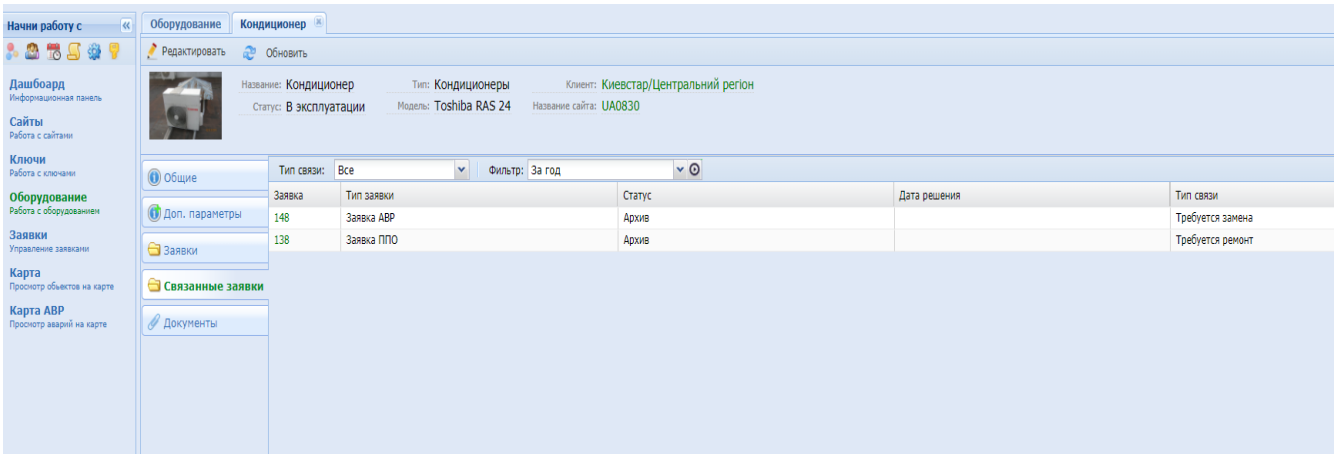


Рисунок 4.8 – Картка підпункту «Обладнання-Зв’язані заявки»

Далі є модливість відкрити вкладку «Обладнання-Документи», що зображена на рисунку 3.11, де можна побачити фото обладнання. Його можна скачати, видалити та додати нове фото за бажанням.

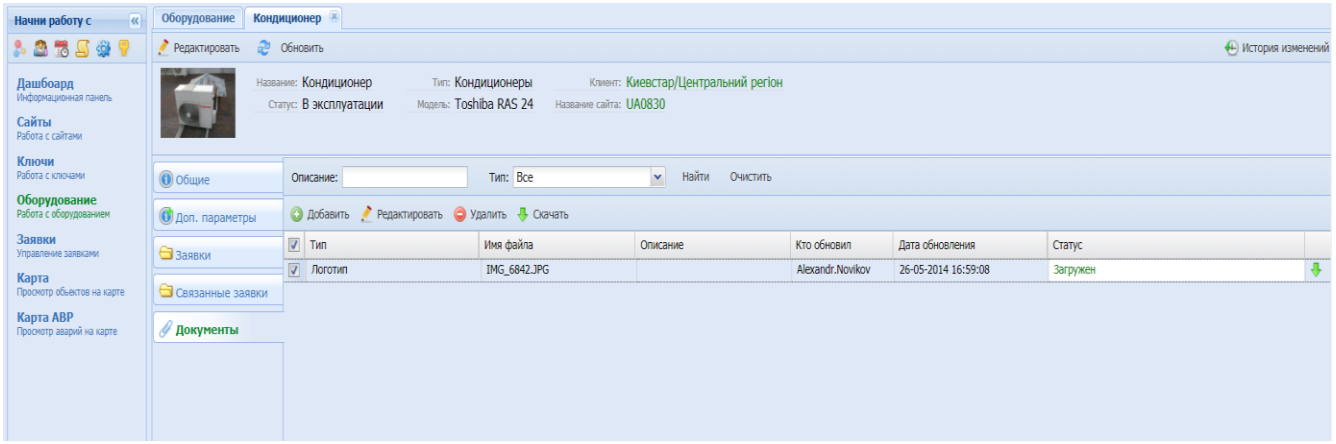


Рисунок 4.9 – Картка підпункту «Обладнання-Документи»

На рисунку 4.10 можна побачити відкрите фото обладнання.

Рисунок 4.10 – Картка підпункту «Обладнання»

На наступному рисунку зображено обладнання Авто. Є можливість дізнатись детальну інформацію, таку, як: назва, модель, інвентарний номер, класс авто, тип палива, показання одометра та ін.

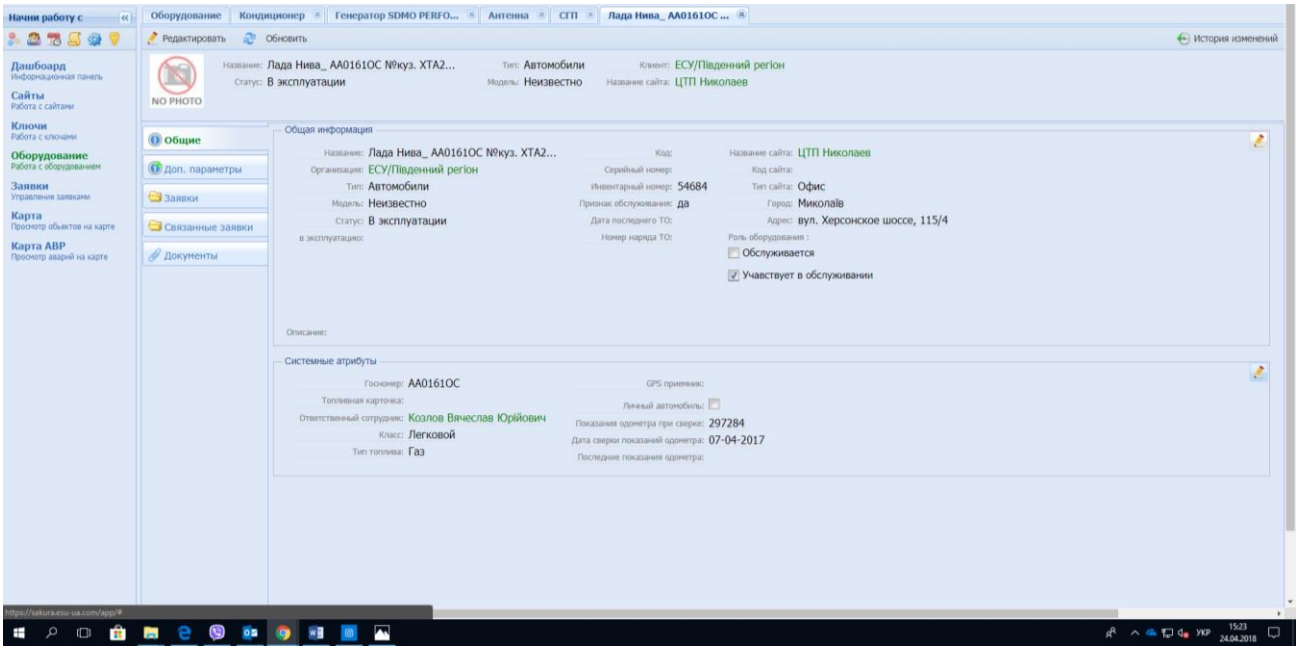


Рисунок 4.11 – Картка підпункту «Обладнання»

Інтеграція з системи з Google Maps дозволяє провести розрахунок оптимального маршруту для виконавців робіт, вказати контрольні точки маршруту і вести їх облік.

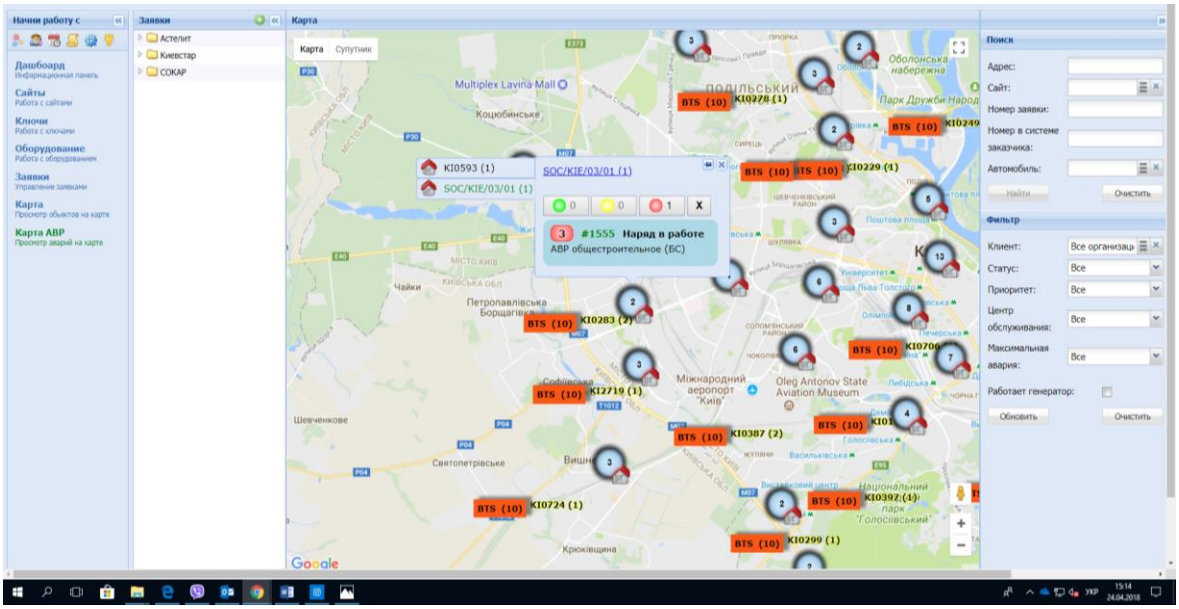


Рисунок 4.12 – Картка підпункту «Обладнання»

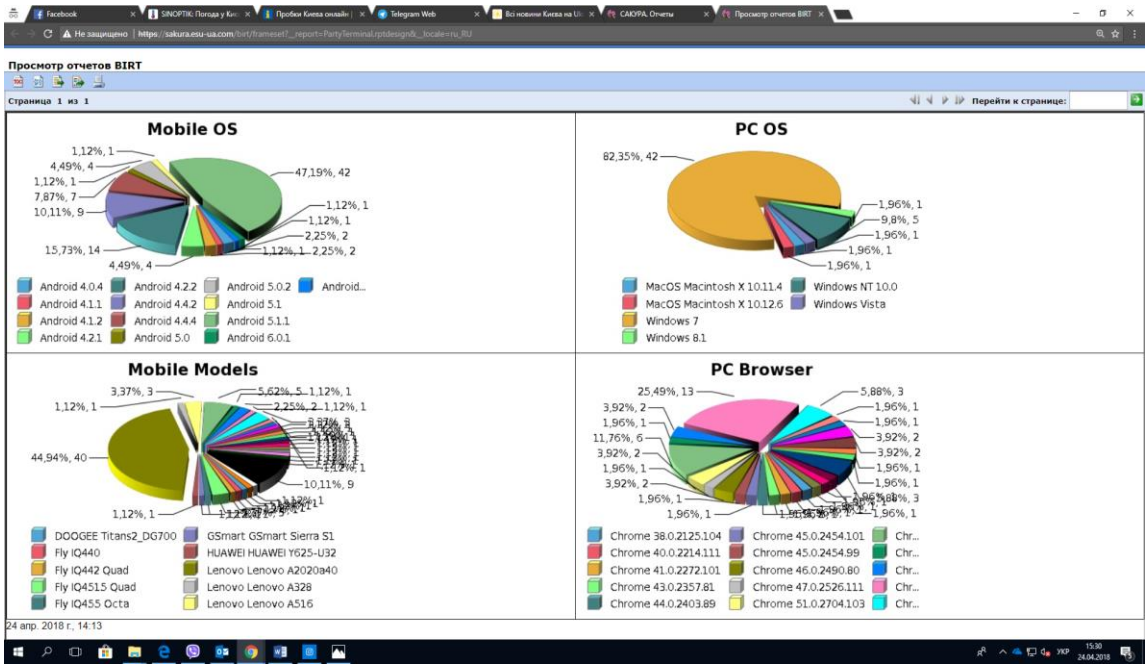


Рисунок 4.13 – Вікно формування звітів

У такому вигляді, як представлено на рисунку 6 показна інформація про звіти про обладнання (в даному випадку про мобільні телефони , програмні забепечення та ін.). Система передбачає можливість самостійного створення звітів в залежності від потреби замовника, створювати різноманітні друковані форми та виконувати експорт звітів в PDF, Word, Excel.

ВИСНОВКИ

У даній магістерській дисертації була розроблема комп’ютерно-інтегрована система управління ресурсами підприємства на мові програмування Python з використанням системи управління базами даних Oracle Database 12c Enterprise Edition Release 12.2.0.1.0. , а саме було проведено:

1 Проведений аналіз структури підприємства ТОВ «Е С У», виявлена взіємодія інформаційних потоків.
2 На основі аналізу автоматизованих систем управління роботи підприємства та існуючих інформаційних систем ряду визначена структура комп’ютерно – інтегрованої системи управління ресурсами підприємства.
3 Обгрунтована структура бази даних для забезпечення інформаційних потоків системи та виконана її реалізація на мові програмування Python.
4 Зібрана інформація щодо технічного обладнання підприємства ТОВ «Е С У» та внесена у відповідні розділи бази даних.
5 Вибрані структури основних підсистем комп’ютерно – інтегрованої системи управління ресурсами підприємства.
6 Виконаний економічний аналіз ефективності впровадження комп’ютерно – інтегрованої системи управління ресурсами підприємства.
7 Виконана програмна реалізація візуалізації неробочого обладнання на мапі.
8 Розроблений користувацький інтерфейс та Android – додаток для комп’ютерно – інтегрованої системи управління ресурсами підприємства.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Воробйов Ю.М., Холод Б.І.Управління ресурсами підприємства – К.: Центр навчальної літератури, 2004. – 288 с.
2. Бесекерскій В.А., Попов О.П. «Теорія систем автоматичного управління» - 4-е вид., Перераб. і доп. - СПб.: Професія, 2003. - 747 с.
3. Бердтис А. Структуры данних. - М.: Статистика, 1974, - 408 с.
4. Васильев В.Н. Организация, управление и экономика гибкого интегрированного производства в машиностроении. – М.: Машиностроение, 1986. –312 с.
5. Капустін, М. М. Автоматизація виробничих процесів у машинобудуванні: Учеб. для втузів / Під ред. М. М. Капустіна. — М.: Вищу школу, 2004. — 415.
6. Воройский, Ф. З. Інформатика. Енциклопедичний систематизований словник-довідник. (Введення ЄІАС у сучасні інформаційні і телекомунікаційні технології в термінах і фактах). — М.: Физматлит, 2007. — 760.
7. Форрестер Дж. Основы кибернетики предприятия: индивидуальная динамика. — М.: Прогресс, 1971. — 340 с.
8. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
9. Економіка підприємства: навч. посібник / за заг. ред. В.Г. Герасимчука, А.Е. Розенплентера. – К.: ІВЦ «Видавництво «Політехніка», 2003. – 264 с.
10. А. Ю. Берко, О. М. Верес Організація баз даних: практичний курс : Навч. посіб. для студ.; Нац. ун-т "Львів. політехніка". - Л., 2003. - 149 с.

11. Основы конструирования и расчета химико-технологического и природоохранного оборудования. Справочник. / А.С. Тимонин Том 2, 2-е изд., перераб. и доп. - Калуга: Издательство Н. Бочкаревой, 2002. — 1030 с
12. Цыпкин Я. З. Основи теорії автоматичних систем. М., Наука, 1977.
13. Годин В.В., Корнеев И.К. Управление информационными ресурсами: 17-модульная программа для менеджеров «Управление развитием организации». Модуль 17. — М.: «Инфра-М», 2000. — 352 с.
14. Гудвін Г.К., С.Ф. Гребе, М.Е. Сальдаго «Проектування систем управління» . — пер. з англ. - М.: БІНОМ, Лабораторія знань, 2004. - 911 с.
15. Анхімюк В.Л., Олейко О.Ф., Міхєєв М.М. «Теорія автоматичного керування» - М.: Дизайн ПРО, 2002. - 352 с.
16. Пригорьев, Ю.А. Проблемы выбора доступа к данным при проектировании информационных систем на основе СУБД / Информационные технологии. - 1999 - №5. С. 4-10.
17. А.А. Оводенко, А.М. Смирнов, А.Г. Степанов, Т.В. Третьякова Формирование информационного обеспечения для поддержки принятия решений на предприятии: учебное пособие ; РИО ГУАП. СПб., 2002. 141 с.
18. Дейт, К., Дж. Введение в системы баз данных, 6-е издание. К.; М.; Спб.: Издательский дом «Вильяме», 2000. - 848 с.
19. Скворцов, А.В. Автоматизация управления жизненным циклом продукции: Учебник для студентов учреждений высшего профессионального образования / А.В. Скворцов, А.Г. Схиртладзе, Д.А. Чмырь. - М.: ИЦ Академия, 2013. - 320 с.
20. Марк Лутц. Программирование на Python, 4-е видання, I том - Переклад з англійської. - СПб.: Символ-Плюс, 2011. - 992 с.
21. Т. М. Басюк, П. І. Жежнич Методи та засоби мультимедійних інформаційних систем : навч. посіб. ; Нац. ун-т "Львів. політехніка". - Львів : Вид-во Львів. політехніки, 2015. - 426 с. - Бібліогр.: с. 413-416.

22. Перцовский М.И. Комплексная автоматизация промышленного предприятия: новые преимущества и новые проблемы//Мир компьютерной автоматизации. 2001.- №3 - С. 12-15.
23. Схиртладзе, А.Г. Автоматизация технологических процессов и производств: Учебник / А.Г. Схиртладзе, А.В. Федотов, В.Г. Хомченко. - М.: Абрис, 2012. - 565 с.
24. Практическое руководство по маркетинговой деятельности на предприятии./И.Е. Самусенко, Р.А. Логов и др.; Под ред. К.С. Гаврилова - М.: изд-во МГУ, 63 с.
25. Боэм Б.У. Инженерное программирование для проектирования программного обеспечения. -М.: Радио і связь, 1985, -512с.
26. Энсор Д., Стивенсон Й. Oracle. Проектирование баз данных К.: BHV, 1999.-557 с.
27. Лисицин Н.А. ”Экономика, организация и планирование промышленного производства”, Минск, ”Вышэйшая школа”, 1990г.
28. Грекул, В.И. Автоматизация деятельности предприятия розничной торговли с использованием информационной системы Microsoft Dynamics NAV: Учебное пособие / В.И. Грекул, Н.Л. Коровкина, Д.А. Богословцев. — М.: Бином, 2014. — 182 с.
29. Баин, А. М. Современные информационные технологии систем поддержки принятия решений: учебное пособие / А. М. Баин. – М.: ИД "ФОРУМ", 2009. – 260 с.
30. Мейер, М.В., Оценка эффективности бизнеса: Учебник. Пер. с англ.— М.: Вершина, 2007. – 300 стр.
31. Васильев А.А. Модели управления процессом бюджетного планирования на промышленных предприятиях. Межвузовский сборник трудов молодых

ученых по итогам работы межвузовского аспирантского семинара / ВолгГТУ. – Волгоград, 2001. – 0,16 п.л.

32. Грязнова, А.Г. , Оценка стоимости предприятия (бизнеса): Учебник – М.: Финансы и статистика, 2008. – 520 стр.
33. Том Кайт «ORACLE для профессионалов» Пер. с англ./ТомКайт- СПб.: ООО «ДиаСофт», 2003. — 672 с.

ДОДАТОК А

Фрагмент коду програми

```
0#!/usr7bin/env python
        # encoding: UTF -8
#PYLINT: disable=C0103,W0212,W0613, C0301
"""
        Propose: Implement base API for AuthRole
        Author: 'yac'
        Date: 21.05.2017
"""
+ import ...
  log = init_logger('[AUTH_R0LE]')

  class AuthRoleApi(IndividualApi):
      """ Implement auth group api """
      DEF_S0RT = {'property': 'role_type_name', 'direction':
      'ASC'> role_type_id = ''
      def get_auth_role(self):
            "" Get auth group ""
          try:
              log.info('Get auth role type. ')
              select = "SELECT v.role_type_id, v.role_type_name,
              v.organization_needed, v.report_group_needed, "\
                      "COUNT(*) OVER () rn_count FROM AUTH_ROLE_TYPE v"
              bind = {}
```

```
            sql =     u"SELECT * FROM (SELECT dt.*, ROWNUM rn FROM ({0}
            {1})dt) "\

                    u"WHERE rn BETWEEN ^start_rn AND
            ^limit_rn".format(select, self.sort())

            bind.update(self.start_limit())

            res = self.db_api.get_info(sql, bind)

            return {'success': True, 'total_count': res.get('rn_count, 0),
        'result': res['res']}
except StandardError, err:

            log.error('ERROR: Cannot get auth role\n%s' % str(err))

                return SYS_ERR0R

    def get_auth_roles_by_id(self):

        """ Get auth roles by id """

        try:

            log.info( 'Get all roles by id')

            sql = "SELECT role_type_id, role_type_name, organisation_needed,
            report_group_needed "\

                  "FROM AUTH_ROLE_TYPE  WHERE role_type_id
               =:role_type_id"

            res = self.db_api.get_info(sql, bind={'role_type_id': self.role_type_id})

            return {'success': True, 'result': res['res']}
except StandardError, err:
```

```
            log.error('ERROR: Cannot get auth role by id\n%s' %
            str(err)) return SYS_ERR0R

def get_roles_by_type(self):
        """ Get all roles by type """
        try:
                log.info('Get al roles by type')
                sql = "SELECT r.role_id, r.role_name, r.role_descr "\
                  "FROM auth_role r "\
                  "WHERE r.role_type_id =:role_type_id " \
                  "ORDER BY r.role_name"
                Res = self.db_api.get_info(sql, bind={' role_type_id'
:self.role_type_id})
                Return {`success`: True, 'result': res ['res']}
        Except StandardError, err:
                Log.error ('ERROR: Cannot get all role by type\n%s' %
str(err))
                Return SYS_ERROR
      def role_2_group( self, process, group_id=None, role_id=None,
      role2group_id=None):
          """ Assign/delete
          roles to group
          @param process:
```

```
 ADD, DELETE
 @param
 group_id: group
 ID
 @param role_id:
 role ID
 @param role2group_id: role to group assigned ID
"""
try:
    log.info('Role to group %s' % process)
    param_list = [APP_NAME, self.user_login]
    if process == 'ADD':
        param_list.extend([group_id, role_id,
        self.params.get('organization_id',
        None),
        self.params.get('report_group_id',
        None)])
        self.conn = self.db_api.conn()
        role_group_id =
self.db_api.process_pkg('PKG_PARTY.AUTH_GROUP_ROLE_ADD',
self.conn, param_list=param_list, out=l)
        self.db_api.commit_close(self.conn)
        return {'success': True, 'role_2_group_id': role_group_id>
```

```
            elif process = 'DELETE':
                param_list.extend([role2group_id])
                self.conn = self. db_api. conn ()
                self.db_api.process_pkg('PKG_PARTY.AUTH_GROUP_ROLE_DEL
                ETE', self.conn, param_list=param_list)
                self.db_api.commit_close(self.conn)
                return {'success': True}
except StandardError, err:
        error, = err.args
        if isinstance(error, cx_Oracle._Error):
              if error.code = 20607:
                 return {'success': False. 'errorMessdge'^:
                'Role already in group}'
                 elif error.code = 20610:
                  return {' success': False, 'errorMessaqe': 'This role
                require report group!'}
                  elif error.code = 20611:
                  return {'success': False, 'errorMessage': 'This role require
                organization!'}
            log.error('ERROR: Cannot %s role to group\n%s' %
            (process, str(err))) return SYS.ERRO
```

```
#!/usr/bin/env python
# encoding: UTF-8 -*-
# pylint: disable=C0103,W0212,W0613, C0301
"""
Propose: Implement base API for AuthGroup
Author: `yac'
Date: 18.05.2017
+import ...
log = init_logger('[AUTH_GROUP]')
class AuthGroupApi(IndividualApi):
                        """ Implement auth group api """
                        DEF_S0RT = {'property': 'group_name', 'direction':
                'ASC'>
        # group_id = None
                def get_auth_group(self):
                """ Get auth group """
                try:
                        log.info('Get auth group.')
                        select = "SELECT v.group_id, v.group_name,
                        v.group_descr, COUNT(*) OVER () rn_count
                        FROM auth_group v"
                        bind, where = {}, ""
                        if self, params, get ('group_name', ` `):
                        bind ['group_name'] =
                        self.params['group_name'].strip()
                        where = ' '.join((where, u"WHERE
                        upper(v.group_name) LIKE
                        '%'||upper(:group_name)||'%'"))
```

```
                bind.update(self.sta rt_limit())
                sql = u"SELECT * FROM (SELECT dt.*,
                ROWNUM rn FROM ({0} {1} {2})dt) "\
                u" WHERE rn BETWEEN :start_rn AND
                :limit_rn". format (select, where, self.sortO)
                res = self.db_api.get_info(sql, bind)
            return {'success': True, 'total_count': res.get('rn_count', 0),
            'result': resl'res']}
except StandardError, err:
            log.error('ERROR: Cannot get auth group.\n%s' %
            str(err))
                SYS_ERR0R
    def get_auth_group_by_id(self):
    """ Get auth group by id """
    try:
            log.info('Get auth group by id')
            sql = "SELECT group_id, group_name, groupjdescr
            FROM auth_group WHERE group_id=:group_id"
            res = self.db_api.get_info(sql, bind={'group_id':
            self.entity_id})
            return {'success': True, 'result': resl'res']}
except StandardError, err:
            log.error('ERROR:Cannot get auth group by id\n%s' %
            str(err))
            return SYS_ERR0R
    def get_assign_roles(self):
            """ Get assign roles for group """
            try:
```

```
                    log.info('Get assign roles for group')
                    sql = "SELECT gr.id role2group_id, ar.role_id,
                    ar.role_name, ar.role_descr, "\
                        "rt. role_type_id, rt. role_type_name, "\
                        "gr.organization_id,
                    pkg_party.GET_ORGANIZATION_FULL_NAME
                    (o.organization_id) organization_name, " \
                        "gr.report_group_id,
                    repg.report_group_name, " \
                        ' COUNT(*) OVER () rn.count "\
        "FROM AUTH_R0LE ar "\
        "INNER JOIN auth_role_type rt ON ar.role_type_id =
        rt.role_type_id \
        "INNER JOIN auth_group2role gr ON ar.role_id = gr.role_id "\
        "LEFT OUTER JOIN organization o ON
        gr.organization_id=o.organization_id "\
        "LEFT OUTER JOIN report_group repg ON
        gr.report_group_id=repg.report_group_id "\
        "WHERE gr.group_id =:group_id %s ORDER BY
        rt.role_type_name, ar.role_name"
bind = {'group_id': self.entity_id> where = ' '
for param, val in self .params. iteritems():
if val and val != '0':
if param = 'role_type_id':
bind[param] = val
where = ' '.join([where, 'and rt.role_type_id=:role_type_id'])
elif param = 'organization_id':
bind[param] = val
```

```
where = ' '.join([where, 'and gr.organization_id=:organization_id'])
      elif param = 'role_id':
         bind[param] = val
         where = ' '.join([where, 'and ar.role_id=:role_id'])
         bind.update(self.sta rt_limit())
         sql = u"SELECT * FROM (SELECT dt.*, ROWNUM rn FROM
({0})dt) " \
u'WHERE rn BETWEEN :start_rn AND :limit_rn".format(sql %
where)
         res = self.db_api.get_info(sql, bind)
         return {'success': True, 'result': resl'res'], 'total_count':
         res.get('rn_count', 0)}
except StandardError, err:
         log.error('ERROR: Cannot get assign roles for group\n%s' %
         str(err))
         return SYS_ERR0R
         def get_assign_users(self):
                  """ Get assign users for group """
                  try:
         log.info('Get assign users for group')
         sql = "SELECT au.user_id party_id, au.active, au.auth_type,
         au.login, ind.first_name, ind.last_name, " \
                  'NVL(ind.second_name, ) second_name,
         TO_CHAR(au.valid_for, 'dd-mm-yyyy ) valid_for "\
                           "FROM auth_user au "\
                     "INNER JOIN individual ind ON
                     au.user_id=ind.party_id '\
                     "INNER JOIN party p ON au.user_id=p.party_id '\
```

```
                    "INNER JOIN auth_user2group ug ON (au.user_id
                    = ug.user_id) "\
                    "WHERE p.status_id=l and p.dd IS NULL "\
                    "AND ug.group_id=:group_id"
bind_params = {'group_id': self.entity_id> if self.params.get('query',
None):
bind_params[ query'] = self.params[ query']
sql = ' '.join((sql, u"AND (upper (ind.last_name ||' '|| ind.first_name ||' '||
ind.second_name)"
        u" LIKE '%'||upper(:query)||'%' "\ u"0R upper(au.login) LIKE
        u"OR upper(:query)||'%')"))
res = self.db_api.get_info(sql, bin =bind_params)


#!/usr7bin
/env
python #
encoding:
UTF-8
#          : disable=C0103,W0212,W0613, C0301
"""
  Propose: Implement base
  API for AuthRole
  Author: 'yac'
"""
import ...
```

```
  log = init_logger('[AUTH_R0LE]')

  class AuthRoleApi(IndividualApi):
      """ Implement auth group api """
      DEF_S0RT = {'property': 'role_type_name', 'direction':
      'ASC'> role_type_id = ''
      def get_auth_role(self):
          """ Get auth group """
          try:
              log.info('Get auth role type. )
              sql = "SELECT role_type_id, role_type_name, organization_needed,
              report_group_needed "\
                      "FROM AUTH_ROLE_TYPE WHERE
                      role_type_id=:role_type_id
              res = self.db_api.get_info(sql, bind={'role_type_id': self.role_type_id>)
              return {'success': True, 'result': res['res']}
except StandardError, err:
              log.error('ERROR: Cannot get auth role by id\n%s' % str(err))
                          SYS_ERR0R
      def get_roles_by_type(self):
          """ Get all roles by type """
          try:
```

```
            log.info( 'Get all roles by type')
            sql = "SELECT r.role_id, r.role_name, r.role_descr "\
                  "FROM auth_role r "\
                  "WHERE r.role_type_id =:role_type_id " \
                  "ORDER BY r.role_name'
            res = self.db_api.get_info(sql, bind={'role_type_id': self.role_type_id>)
            return {'success': True, 'result': res['res']}
except StandardError, err:
            log.error('ERROR: Cannot get all role by type\n%s' %
            str(err)) return SYS_ERR0R
      def role_2_group( ;elf, process, group_id=None, role_id=None,
      role2group_id=None):
         """

           param process:
         ADD, DELETE
           param
         group_id: group
         ID
           pa      role_id:
         role ID
           param role2group_id: role to group assigned ID
        """
```

```
try:
    log.info('Role to group %s' % process)
    param_list = [APP_NAME, self.user_login]
    if process = 'ADD':
        param_list.extend([group_id, role_id,
                           self.params.get('organiz
                           ation_id', None),
                           self.params.get('report_g
                           roup_id', None)])
                           self.conn *
                           self.db_api.conn()
        role_group_id =
        self.db_api.process_pkg('PKG_PARTY.AUTH_GROUP_ROLE_ADD
        ', elf.conn,
        param_list=param_list, out=l)
        self.db_api.commit_close(self.conn)
        return {'success': True, 'role_2_group_id': role_group_id>
    elif process = 'DELETE':
        param_list.extend([role2group_id])
                  = self. db_api. conn ()
        self.db_api.process_pkg('PKG_PARTY.AUTH_GROUP_ROLE_DEL
        ETE', self.conn, param_list=param_list)
        self.db_api.commit_close(self.conn)
```

```
                return {'success': True}
except StandardError, err: error, = err.args
            if isinstance(error, cx_Oracle._Error):
              if error.code = 20607:
                  turn {'success': False. 'errortlessdge'^ 'tole
                already in group}'} elif error.code = 20610:
                  return {' success': False, 'errprltessaqe': 'This role
                  uire report group!'} elif error.code = 20611:
                  return {'success': False, 'errorMessage': 'This role requ'
                                                    'ire organization!'}
            log.error('ERROR: Cannot %s role to group\n%s' %
            (process, str(err))) return SYS.ERROR
```

```
#!/usr/bin/env python
#•*- encoding: UTF-8
#pylint: disable=C0103, W0212, W0613, C0301
"""
Propose: Implement REST API for devices
Author: 'yac'
Date: 13.05.2017
"""
```

```
Import…
class DeviceController(object):
        """ implement device controller """
       class Device(object):
            """ implement REST api for device
        """
        exposed = True
        ^require(membe r_of("resou rce.device_read"))
        @csv_out
        @session_release_lock
        def GET(self, *vpath, **params):
        """ implement GET method """
        dev = DeviceAPK'resource',
        'device')
        dev.role_name = 'resou
        rce.device_read'
        dev.params = params
        if vpath and len(vpath) >= 1: #
        /resource/device/<device_id>
        if not to^tJ5^iJ^yJ^(vpath[0]):
        return OBJECT_N0T_F0UND
            if 'linked_order_csv' in vpath: #
                /resource/device/<device_id>/linked_order_csv
                dev.csv = True
```

```
        return dev.get_linked_order()
    if 'linked_order' in vpath: #
        /resource/device/<device_id>/linked_order
        return dev.get_linked_order()
    return dev.get_device_by_id()
path =
cherrypy.request.path_info.split
('/')
if params:
    dev.params = params
    if 'device' in path: # /resource/device/
    return dev.search_entity()
    elif 'device_csv' in path: # /resource/device_csv/
    dev.csv = True
    return dev.search_entity()
return EMPTY_REQUEST
@require(member_of("resource.device_write"))
@cherrypy.tools.json_in()
@che r rypy.tools.json_out()
def POST(self, *vpath, **params)'
""" Implement POST method /resource/device """
device = DeviceAPI('resou rce',
'device')
device.role_name =
```

```
'resource.device_write'
device.params =
cherrypy.request.json
if vpath and len(vpath) >= 1:
    if not device._set_entity_id(vpath[0]):
        return OBJECT_N0T_F0UND
    if 'del' in vpath:
        return device.del_entity() # /resource/device/<entity_id>/del
    elif  'test_set_coordinates' in vpath:
        return device.test_set_resource_coordinates() #
    /resource/device/<entity_id>/test_set_coordinates
if device.params:
    if len(vpath) = 1:
        return device.process_device(device.UPDATE) #
    /resource/device/<entity_id>
    return device.process_device(device.CREATE) # /resource/device
return EMPTY_REQUEST
class Model(object):
""" implement REST API for status """
exposed = True
@require(member_of('resource.device_read'))
@che r rypy.tools.j son_out()
def GET(self, *vpath, **params):
```

```
""" implement GET method """

dev = DeviceAPK'resource', 'device')

if params:

    dev.params = params

if len(vpath) = 1 and vpath[0] = 'tree': #
    resource/device_model/tree
    return dev.get_device_model(pa
    rams.get(1device_type_id', 1))

return EMPTY_REQUEST

@require(member_of('resource.device_write'))

@cherrypy.tools.json_in()

@che r rypy.tools.j son_out()

def POST>J(self, device_type_id, model_id=None, delete=None,

""" implement POST method """

dev = DeviceAPK'resource',
'device')
dev.params =
cherrypy.request.json

if not is_int(Idevice_type_id,
    model_idl):
    return NOT_INTEGER

dev.device_type_id =
device_type_id
dev.entity_id = model_id
```

```
        if delete: #
            resource/device_model/<device_type_id>/<m
            odel_id>/del
            return
            dev.process_model(action=dev.DELETE)
        if not model_id: #
            resource/device_model/<device_type_i
            d>
            return
            dev.process_model(action=dev.CREAT
            E)
        if device_type_id and model_id: #
            resource/device_model/<device_type_id>/<model_id>
            return dev.process_model(action=dev.UPDATE)
        return EMPTY_REQUEST



  #!/usr/bin/env python
  #♦*- encoding: UTF-8
  # pylint: disable=C0103,W0212,W0613, C0301
  """
  Propose: Implement simple api for device
  Author: *yac'
  Date: 13.05.2017
"""

import ...
```

```
  log = init_logger('[DEVICE]')
  class DeviceAPI(BaseEntity):
      """ Implement device api """
      DELETE = 'DELETE'
      CREATE = 'CREATE'
      UPDATE = 'UPDATE'
BJTITY_REQ_PARAMS = ( ' name ', ' model_id ', ' is_served ', ' status_id ', ' org_id ',
                    ' site_id ' )
ENTITY_FIELDS = ('name', 'description', 'org_id', 'model_id', 'status_id', 'status2_id',
                    'date_install', 'notes', 'date_commissioning', 'maintenance_date',
                    'maintenance_wo', 'is_served', 'asset_number', 'serial_number',
                    'device_code')
DATE = ('date_commissioning', 'date_install', 'maintenance_date')

DEF_S0RT = {'property's 'NAME', 'direction's 'DESC'}

device_type_id = ''

def search_entity(self):
    """ Search device """
    try:
        log.info('Sea rch device')
        role_sql = ''
        # add to query roles
        if self .pa rams, get ('resource_role_§jrj^iijg', '0') != '0':
            role_sql = "IMJER JOIN ENTITY_R0LE2ENTITY er2e ON er2e.
                        ENTITY_ID = v. resource_id "\
                    "inner join entity_class ec on ec.CLASS_ID =
                    er2e.BJTITY_CLASS_ID "\
                    "inner join ENTITY_ROLE er on er.R0LE_ID = er2e.
```

```
                BJTITY_ROLE_ID "
    select = "SELECT v.RESOURCE_ID, v.NAME,
    dbms_lob.substr(v.NOTES,600) notes, v.0RG_ID, v.0RG_NAME, v.IP,
    v.status_name, " \ "v.DEVICE_TYPE_ID, v.DEVICE_TYPE_NAME,
    v.STATUS_ID, v.STATUS2_ID, v.M0DEL_ID, v.MODEL_NAME, " \
    "v.SITE_ID, v.SITE_NAME, v.SITE_ADDRESS, v.CITY_ID,
    v.CITY_NAME, v.ASSET_NUMBER, "\
    "v.SERIAL_NUMBER, TO_CHAR(v.DATE.COMMISSIONING, 'dd-
    mm-yyyy') DATE_COMMISSIONING, " \
    "v.DISTRICT_NAME, v.STATE_NAME, v.device_code,
    v.SITE_TYPE_ID, v.SITE_TYPE_NAME %s "\
    "FROM V.DEVICE v {0} " \
"INNER JOIN PHYSICAL_DEVICE_MODEL pdm ON
pdm.model_id=v.model_id " \
"WHERE v.org_id in (select organization_id from v_organization start with
organization_id in ( "\
        "select g2r.organization_id from individual i inner join party p on
        p.party_id = i.party_id "\
        "inner join auth_user u on i.party_id=u.user_id "\
        "inner join auth_user2group u2g on u.user_id=u2g.user_id "\
        "inner join auth_group2role g2r on u2g.group_id=g2r.group_id "\
        "inner join auth_role r on g2r.role_id=r.role_id where u.active=l "\
        "and r.role_name=:role_name and u.login=:user_name) "\
```

```
                "connect by prior
                organization_id=parent_organization_id)".format(role_sql)
bind, where, is_where = {'user_name': self.user_login, 'role_name's self.role_name>,
"", False for pa ram, val in self, pa rams. iteritemsO:
    if (param not in
        IGNORE) and (val
        and val != '0'): val =
        val.strip()
        # todo need fix on front-end. When device_type is not selected and export to
            csv
        if val = 'undefined':
        continue
        if param in ('name', 'site_address'): bind[param] =
            val
            where = ' '.join((where, "and upper(v.{0>) LIKE
        '%'||upper(:{l>)||'%'".format(param, param)))
            elif param = 'date_commissioning_from':
            bind[param] = ' '.join((val, '00:00:00'))
            where = ' ',join((where, "and v.date_commissioning >=
        to_date(:date_commissioning_from, 'dd.mm.yjyyy hh24:mi:ss' elif param =
        'date_commissioning_to':
            bind[param] = ' '.join((val, '23:59:59'))
```

```
        where = ' ',join((where, "and v.date_commissioning <=
    to_date(:date_commissioning_to, 'dd.mm.yyy]y hh24:mi:ss')" elif param =
    'model_id's bind[param] = val

        where = ' '.join((where, "and pdm.model_id in (SELECT model_id FROM
    physical_device_model "\

                                "start with model_id =:model_id "\

                                "connect by prior model_id =
    parent_model_id) ")) elif param = ' resource_role_^jfjjg':
    bind[param] = val

        where = ' ' .join((where, "and er.R0LE_§$/£1£=:resource_roleand
    ec.CLASS_NAME = 'resource' [')) elif param = 'state_id' and self .pa rams, get
    ('districted', None) != '0':

        bind!'districted'] = self .pa rams!'districted'1

        where = ' ' .join((where, "and city_id in (select placeed from place p "\

                                "where dd is null connect by prior p.placeed =
    p.parent_placeed "\

                                "start with p.placeed = :districted)"))

elif                    param                    =                    'stateed's
bind[param] = val

        where = ' ' .join((where, "and cityed in (select placeed from place p "\

                                "where dd is null connect by prior p.placeed =
    p.parent_placeed "\

                                "start with p.placeed = sstateed)"))
```

```
        elif param = 'orged's bind[param] = val
            where = ' '.join((where, "and orged in (select o.organizationed "\
                                    "from organization o "\
                                    "inner join party p on p.partyed = o.organizationed "\
                                    "where p.dd is null "\
                                    "START with p.partyed =:orged "\
                                    "connect by prior p.partyed = p.parent_partyed)"))
        else:
        bind[param] = val
where = ' ',join((where, "and v.{0}=:{1>".format(param, param)))
DeviceAPI.DEF_SORT = {'property': 'NAME', 'direction': 'DESC'}
if self.csv:
    sql = "{0} {1} {2}
    ".format(select % ", where,
    self.sortO) return object2csv(sql,
    self.entity_type, bind)
sql = u"SELECT * FROM (SELECT dt.*, ROWNUM rn FROM ({0} {1} {2})dt) "\
      u'WHERE rn BETWEEN :start_rn AND :limit_rn".format(select %
      ",C0UNT(*) OVER () rn_count",
                                                    where, self.sortO)
bind, update (self ,start_limit()) res_info = self.db_api.get_info(sql, bind)
```

```
return {'success': True, 'total_count': res_info.get('rn_count', 0), 'result':
res_info['res']}


        #!/usr/bin/env python
        ##*- encoding: UTF-8
        #         :disable=C0103,W0212,W0613
          """
          Propose: implementapi for group
          Author: 'yac'
          Date; 27.05.2017


import ...
  log = init_logger('[GROUP]')
    all_ = ['ObjectNotFound', 'GroupAPI']

      """ Implement API for use group """

      DEF_S0RT = {'property': 'group_name', 'direction': 'desc'}
Idef get_group_list(self):
           .. Get group list  ..

           try:
               log.debug('Get group list')

                    sql = 'SELECT v.group_id, v.parent_group_id, v.group_type_id,
                    v.group_type_name, v.group_name, ' \ COUNT(*) OVER () rn_count
                    FROM v_party_group v

               bind = {>
```

```
            sql = u' SELECT * FROM (SELECT dt.*, ROWNUM rn FROM ({0}
            {l»dt) "\
                  u" WHERE rn BETWEEN :start_rn AND
            :limit_rn". format (sql, self.sortO) bind.update(self.sta
            rt_limit())

            res_info = self.db_api.get_info(sql, bind)

            return {'success': True, 'total_count': res_info['rn_count'], 'result':
        res_infoI'res']> except StandardError:
            log.exception('ERROR: Cannot get group list') return SYS ERROR


def get_ind4g roup(self):
    ... Get individuals for group

    try:
    log.debug('Get individual for group ID: %s', self.entity_id)
    sql = SELECT group_name, party_id, last_name, first_name, second_name,
    full_name, personal_number, '\
          org_party_id, org_name FROM v_party_group_individual WHERE
                                group_id=:entity_id'
    res = self.db_api.get_info(sql, {'entity_id': self.entity_id>)
    return {'success': True, 'result': res['res']} except StandardError:
    log.exception('ERROR: Cannot get individuals for group ID: %s', self.entity_id)
    return SYS ERROR
def process(elf, action):
    """ Add/delete individual from group
    """ try:

        log.debug('%s individual. Group ID: %s', action, self.entity_id)
```

```
        if not isinstance(self.params['individual_id'], list):
            return {'success': False, 'errorWessage': 'individual_id is not list or empty'}
        conn = self.db_api.conn()
        for ind_id in self.params['individual_id']:
            self.db_api.process_pkg('pkg_party.group_individual_%s' % action, conn,
            [APP_NAME,
                                                                    self.user_login,
                                                                    self.entity_id,
                                                                    ind_id])
        self.db_api.commit_close(conn)
        return {'success': True} except StandardError:
        log.exception('ERROR: Cannot %s individual. Group ID: %s',
        action, self.entity_id) return SYS ERROR
def update(self):
    ... Update group's name
    try:
        log.debug('Update name for group ID: %s', self.entity_id)
        if not self.params.get('group_name'):
            log.error('Empty required field: group_name')
            return EMPTY_REQUIRE_FIELD
        conn * self.db_api.conn()
        self.db_api.process_pkg('pkg_party.group_update', conn, [APP_NAME,
                                                        self .userjlogin,
                                                        self.entity_id,
                                                        self.pa rams['g roup_name']])
        self.db_api.commit_close(conn)
```

```
    return {'success': True} except StandardError:
    log.exception('ERROR: Cannot update name for group ID: %s',
    self.entity_id)
    return SYS ERROR



#!/usr/bin/env python
# encoding: UTF-8
#PYLINT: disable=C0103,W0212,W0613, C0301
"""
Propose: Im
                        hor:
'yac'
Date:
"""
import ...
log = init_logger(•[INDIVIDUAL]•) class IndividualApi(BaseEntity):
                """ Implement Individual api """
                ENTITY_REQ_PARAMS = (' first_name', ' last_name*, '
                gender')
                ENTITY_FIELDS = ('parent_party_id', 'first_name', 'last_name',
                'second_name', 'personal_number', 'gender', 'marital_status',
                'org_party_id', 'position', 'notes')
```

```
                DEF_S0RT = {>

                def search_entity(self):

        """ Search individual """""

        try:

            log.info('Search individual')

            role_sql = ' '

            # add to query roles

                    if self.params.get( party_role_§MJvg', None) and
                       self.params['party_role_£pjr9£'] != '0':
role.sql = "LEFT JOIN ENTITY_R0LE2ENTITY p2rr ON p2rr.BUITY_ID =
                                        p.party.id "\

  "LEFT JOIN BITITY_ROLE er on er.role_id = p2rr.entity_role_id " \

  "left join entity_class cs on cs.CLASS_ID = p2rr.ENTITY_CLASS_ID"

            select = "SELECT ind.party_id, ind.last_name, ind.first_name,
                        WL(ind.second_name, ' ') second_name, "\

                        "ind.personal_number, ind.gender, ind.marital_status,
                        org.organization_id org_party_id, " \
                        "pkg_party.get_organization_full_name(org.organization^)
                        org_name, ind.position, notes, " \
                        "pkg_party.get_main_party_contact_value(ind.party_id,3)
                        phone, "\
                        "pkg_party.get_main_party_contact_value(ind.party_id,2)
                        email %s "\

                    "FROM party p "\
```

```
        "INNER JOIN individual ind ON p.party_id=ind.party_id "\
        "LEFT OUTER JOIN organization org ON ind.org_party_id =
        org.organization_id {0} "\
        "WHERE p.party_id>0 and p.dd IS NULL ",format(role_sql)
                        , "
if self.params.get( name', None):
    bind['name'] = self.params['name'].strip()
    where = ' '.join((where, "AND upper (last_name ||' '|| first_name ||' ||
    second_name)"
                        "LIKE '%'||upper(:name)||'%'")) if
self.params.get( organization', None): if self.params[
organization'] != '0':
        bind!'org_id'] = self.paramsl'organization']
        where = ' '-joint(where, "and ind.org_party_id in (select
        o.organization_id "\
                "from organization o "\
                        "inner join party p on p.party_id =
                        o.organization_id "\
                        "where p.dd is null "\
                        "START with p.party_id =:org_id "\
                        "connect by prior p.party_id =
                        p.parent_party_id)"))
```

```
#roles

if self.params.get( party_role_§B3(pjg', None): if self .paramsl pa
    rty_role_§j35jpjg'] != '0':

        bind [' party_role_£pjfi)s' ] = self. pa rams [' party_role_§^jvg'].
        strip ()

        where = ' '.join((where, "AND er.ROLEJ*J^£=:party_rolej^syi)g
        and cs.class_name = 'party' "))

if self.params.get('contact', None):

    bind['contact'] = self.params[ contact'].strip() where = '
    '.join((where, "AND p.party_id IN "

                    "(SELECT party_id FROM party_contact
WHERE "

                    "upper(value) LIKE '%'||upper(:contact)||'%'
AND dd IS NULL)"))

if self.params.get('personal_number', None):

    bind[1personal_number'] =
    self.params['personal_number'].strip() where = '
    '.join((where, "AND ind.personal_number
    =:personal_number"))

if self.params.get('position', None):

    bind ['posit ion'] = self, paramsl'position'J.stripO

    where = ' '.join((where, "AND upper(ind.position) LIKE
    '%'||upper(:position)||'%'"))
```

```
        # sort
        order_by = 'ORDER BY last_name ASC' if self.params.get('sort',
        None):
            sort = json.loads(self.params[sort']) if sort and len(sort)
            = 1:
                direction = sort[0].get('direction', 'ASC')
                if sort[0].get('property', None) = 'full_name': order_by =
                'ORDER BY last_name %s' % direction elif
                sort[0].get('property', None) = 'org_name':
                    order_by = 'ORDER BY
                name_official %s' % direction
                elif sort[0].get('property',
                None):
    order_by = ' '.join(('ORDER BY', sort[0]['property'], direction))
        if self.csv:
            sql = "{0} {1} {2} ".format(select % '', where,
            order_by) return object2csv(sql, self.entity_type, bind)
        bind.update(self.sta rt_limit())
        sql = u"SELECT * FROM (SELECT dt.*, ROWNUM rn FROM ({0}
                {1} {2})dt) "\
              u"WHERE rn BETWEEN :start_rn AND :limit_rn".format(select
              % ",C0UNT(*) OVER () rn_count ", where, order_by)
        res = self.db_api.get_info(sql, bind=bind)
```

```
            return {'success': True, 'result': resl'res'], 'total_count':
        res.get('rn_count', 0)> except StandardError, err:
            log.error('ERROR: Cannot search individual.\n%s' % str(err)) return
            SYS.ERROR

                def get_individual_by_id(self):
""" Get individual by
id  """
try:
            sql_info = "SELECT ind.party_id, ind.last_name, ind.first_name,
                NVL(ind.second_name, '') second_name, "\
                "ind.personal_number, ind.gender, "ind.marital_status,
                org.organization_id org_party_id, "\
                "pkg_party.get_organization_full_name(org.organization_id)
                org_name, ind.position, notes, "\
                "pkg party.get main party contact value(ind.party id.3) phone, "\
                "pkg_party.get_main_party_contact_value(ind.party_id,2) email
                "\
                "FROM party p "\
```