Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація мобільного додатку для управління ERP-системою ”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М_____
ОПП «Комп'ютерна інженерія»
спеціальності 123 «Комп'ютерна інженерія»
_____А.М. Казанков
«____» _____ 2024р.

Керівник проекту
кандидат технічних наук, доцент
_____ О.А Кислун
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *магістр*
Галузь знань *12 "Інформаційні технології"*
Спеціальність *123 "Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Казанкову Андрію Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація мобільного додатку для управління ERP-системою*

2. Керівник роботи *Кислун Олег Андрійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 07.08.24

3. Строк подання роботи до захисту *15.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація ERP-системи за допомогою мобільного додатку*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Маркетингове та економічне обґрунтування ІТ-проєкту
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи	*3 аркуша*
Діаграма взаємодії процесів	*1 аркуш*
Маркетингове та економічне обґрунтування ІТ-проєкту	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Маркетингове та економічне обґрунтування ІТ-проєкту	Доренська А.О	09.11.2024 р.	17.11.2024 р.
Охорона праці	Марченко К.М., к.т.н., доцент	03.11.2024 р.	21.11.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	12.10.2024	
2.	Постановка задачі, оформлення ТЗ	18.10.2024	
3.	Розробка моделі компонента	23.10.2024	
4.	Розробка структур даних	25.10.2024	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2024	
6.	Програмування алгоритмів	11.11.2024	
7.	Маркетингове та економічне обґрунтування ІТ-проєкту	13.11.2024	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2024	
9.	Оформлення ПЗ	18.11.2024	
10.	Попередній захист роботи	04.12.2024	

Дата видачі завдання
«__»_____2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Казанков А.М. Дослідження та програмна реалізація мобільного додатку для управління ERP-системою. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації ERP-системи з розробкою мобільного додатку .

Метою розробки є дослідження та програмна реалізація мобільного додатку для промислового підприємства та аналізу впливу його роботи на ефективність роботи підприємства.

Об'єктом дослідження є можливості використання мобільного додатку для підвищення ефективності діяльності підприємства.

Предметом дослідження є сукупність теоретичних, технічних та практичних засад створення й впровадження мобільного додатку для збільшення обсягів виробництва та продажу.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення та методах порівняння та систематизації і синтезу.

Результат роботи – програмна реалізація та теоретичне узагальнення і нові практичні рекомендації для створення мобільного додатку.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення. Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися у на різних операційних системах (Android та iOS). Програму розроблено в середовищі розробки IntellijIDEA на мові програмування java, php-7.3.

Ключові слова: комп'ютерна інженерія, ERP, java, ООП, MySQL.

ANNOTATION

Kazankov A.M. Research and software implementation of the ERP system using a mobile application. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this master's work, software was developed, which is intended for the implementation of the ERP system with the development of a mobile application.

The purpose of the development is the research and software implementation of a mobile application for an industrial enterprise and the analysis of the impact of its operation on the efficiency of the enterprise.

The object of the study is the possibility of using a mobile application to improve the efficiency of the enterprise.

The subject of the study is a set of theoretical, technical and practical principles of creating and implementing a mobile application to increase the volume of production and sales.

Research methods are based on methods of coding theory, methods of mathematical statistics, methods of software development, and methods of comparison and systematization and synthesis.

The result of the work is software implementation and theoretical generalization and new practical recommendations for creating a mobile application.
In the process of working on the software model, an analysis of existing hardware and software was performed.

All components of the developed software are fully described. A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on different operating systems (Android and iOS). The program was developed in the IntellijIDEA development environment in the programming language java/

Keywords: computer engineering, ERP, java, OOP, MySQL.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

БД – база даних

КІС – корпоративна інформаційна система

МД – мобільний додаток

ІС – інформаційна система

API - Application Programming Interface

PWA - Progressive Web Applications

ERP - Enterprise Resource Planning System

MRP - Manufacturing Resources Planning

SWOT - Strengths, Weaknesses, Opportunities, Threats

SQL - structured query language

PHP - Hypertext Preprocessor

ABAP - Advanced Business Application Programming

CRM - Customer Relationship Management

AT (англ. AT – Automation Technology)

IT (англ. IT – Information Technology)

HMI - Human - Machine Interface

ВСТУП

Актуальність теми. На початковому етапі діяльності підприємства впровадження ERP-системи зазвичай не є необхідністю. Усі робочі процеси можуть виконуватись за допомогою стандартних офісних програм, що дозволяє швидко вносити зміни у документацію. Однак зі зростанням і розширенням підприємства управління різними підрозділами, такими як кадрова служба, бухгалтерія та інші, стає дедалі складнішим.

У таких умовах ефективне управління, оптимізація ресурсів та підвищення продуктивності потребують нових підходів. ERP-системи стають оптимальним рішенням

Мета й завдання дослідження. Метою дослідження є розробка та програмна реалізація мобільного додатку для ERP-системи управління підприємством, а також визначення його впливу на ефективність діяльності підприємства.

Для досягнення поставленої мети визначено такі завдання:

– аналіз існуючих ERP-систем;

– вивчення концепцій побудови ERP-систем;

– програмна реалізація мобільного додатку для ERP-системи;

– дослідження впливу мобільного додатку на ефективність діяльності підприємства.

Об’єкт дослідження – можливості використання мобільного додатку в інтеграції з ERP-системою для підвищення ефективності роботи підприємства. **Предмет дослідження** – теоретичні, технічні та практичні аспекти створення і впровадження мобільного додатку для управління підприємством.

Методи дослідження включають:

– методи порівняння та систематизації, аналізу та синтезу (для уточнення поняття "мобільний додаток");

– SWOT-аналіз (для визначення переваг і недоліків існуючих мобільних додатків);

– прототипування та моделювання (для створення мобільного додатку).

Наукова новизна дослідження:

– розроблено мобільний додаток для інтеграції з ERP-системою;

– сформульовано рекомендації щодо використання мобільного додатку.

Практична цінність

Розроблений мобільний додаток для підприємства дозволяє підвищити ефективність його діяльності.

Проведені дослідження підтвердили економічну ефективність та доцільність впровадження мобільного додатку, що робить його перспективним для застосування на українських підприємствах.

Достовірність наукових результатів забезпечена теоретичними висновками та експериментальними даними, отриманими під час тестування розроблених систем.

Отже, дослідження та розробка мобільного додатку для ERP-системи є актуальним завданням, що потребує вирішення в рамках цієї магістерської роботи.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1. Призначення системи

Розробка мобільних додатків є ключовим етапом для ефективного функціонування сучасного бізнесу. Мобільний додаток дозволяє виконувати робочі завдання в будь-якому місці, використовуючи лише смартфон або планшет. Завдяки можливості отримувати своєчасну інформацію про надходження замовлень, стан продукції чи статус угод, бізнес-процеси стають більш керованими та продуктивними.

Сучасні IT-рішення, такі як ГІС-технології, GPS-навігація, системи CRM, HRM, а також ERP-стратегії, відіграють важливу роль у створенні сприятливих умов для оптимізації діяльності приладобудівної галузі. Ці технології сприяють підвищенню ефективності управління, вдосконаленню організаційної структури та адаптації до змін ринку.

Зростання популярності смартфонів обумовлено їхньою багатофункціональністю та портативністю. Вони замінюють численні інструменти, такі як календар, калькулятор, годинник, будильник, а також виконують мультимедійні завдання. За даними Statista, у 2021 році кількість завантажень додатків на всіх мобільних платформах сягнула 197 мільярдів. Ці додатки суттєво розширюють функціональність пристроїв, відкриваючи нові канали комунікації та інструменти для бізнесу.

Мобільні додатки сприяють оптимізації інформаційних потоків у компанії, підтримують прямий зв’язок із клієнтами, дозволяють аналізувати їхню поведінку та вподобання, а також досягати маркетингових цілей. Використання мобільних додатків у професійній діяльності, особливо у сфері приладобудування, стає актуальним інструментом для підвищення продуктивності виробництва.

У загальному розумінні мобільний додаток - це програмне забезпечення, розроблене для роботи на смартфонах, планшетах та інших мобільних пристроях, що призначене для вирішення завдань користувача та оптимізації його діяльності. Вчені визначають мобільний додаток як спеціальну програму, яку можна завантажити та встановити через онлайн-магазини додатків, такі як App Store або Google Play.

Мобільні пристрої - це компактні портативні гаджети, які працюють на операційних системах (iOS, Android, Windows Phone) та підтримують доступ до мобільних мереж і Wi-Fi. Їх основні переваги включають компактні розміри, зручність використання, тривалу автономну роботу, швидке включення та вимикання, можливість підключення до Інтернету та сумісність зі стаціонарними комп'ютерами й ноутбуками.

Ключовою особливістю мобільних додатків є високий рівень опрацювання функціоналу, який визначає унікальність та привабливість кінцевого продукту.

До найпоширеніших видів мобільних додатків для бізнесу належать:

- додатки для автоматизації процесів;

- системи автоматизації виробництва, логістики та зберігання продукції, які знижують витрати порівняно з використанням стаціонарних робочих станцій на основі ПК.

Програми для підвищення продуктивності та співпраці:

- системи для спільного доступу та роботи з файлами;

- інструменти внутрішньої комунікації, такі як месенджери, трекери повідомлень;

- мобільні версії корпоративних соціальних мереж та платформи для електронних опитувань;

- додатки для управління проектами та завданнями, які забезпечують збирання, уточнення та синхронізацію інформації в режимі реального часу.

Використання таких рішень дозволяє бізнесу оптимізувати процеси, покращувати комунікацію та забезпечувати ефективну координацію роботи команди.

Серед клієнтських мобільних додатків можна виділити такі категорії:

Додатки для розширення функціоналу онлайн-сервісів:

- інтернет-каталоги та мобільні вітрини;
- платформи для онлайн-купівлі квитків;
- мобільний банкінг;
- трекери статусів замовлень.

Мобільні додатки як електронна картка постійного клієнта:

-інструменти для програм лояльності, зручного доступу до знижок та спеціальних пропозицій.

На сьогодні існує кілька основних підходів до технічної реалізації мобільних додатків:

- **Нативні додатки**: створюються окремо для кожної операційної системи (iOS, Android), забезпечують високу продуктивність і доступ до всіх функцій пристрою, але потребують більше ресурсів на розробку.
- **Гібридні додатки**: поєднують у собі веб- та нативні технології, дозволяють швидше створювати кросплатформні рішення, проте мають обмеження в продуктивності порівняно з нативними.
- **Прогресивні веб-додатки (PWA)**: це вебсайти, які мають функціонал, подібний до додатків, працюють через браузер, але забезпечують досвід, максимально наближений до нативних додатків, без потреби у встановленні.

Кожен із цих підходів має свої переваги та недоліки, вибір залежить від потреб бізнесу, цільової аудиторії та доступних ресурсів.

Головні їх характеристики представлені у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика основних видів мобільних додатків.

	Нативні	Гібридні	PWA
Можливість перевикористання коду	Код розробляється окремо для кожної платформи	Можливе перевикористння коду	Можливе перевикористння коду
Доступ до функцій пристроїв	Найбільш повний	Обмежений доступ	Дуже низький
Модель розповсюдження	Завантаження в магазині додатків	Завантаження в магазині додатків	Доступ по URL
Продуктивність	Висока	Низька	Низька
Підтримка пристроями	Висока	Висока	Середня
Популярність	Висока	Середня	Середня
Підтримка зовнішніх бібліотек	Висока	Середня	Висока

Метою магістерської роботи є демонстрація можливостей застосування ERP-системи шляхом розробки мобільного додатку. Це дозволить виявити переваги інтеграції мобільних рішень в ERP-систему для підвищення ефективності управління підприємством.

1.2 Область застосування

Управління складом передбачає облік товарів і оптимізацію бізнес-процесів, пов'язаних із зберіганням, інвентаризацією, прийманням,

відвантаженням та контролем поставок. Автоматизація складського обліку дозволяє підвищити ефективність цих процесів, зокрема шляхом прискорення інвентаризації, зменшення кількості помилок та оптимізації витрат часу на операції.

Запровадження мобільного додатку для управління складом, який інтегрується з існуючою ERP-системою, є перспективним кроком. Такий додаток забезпечує:

- підвищення швидкості виконання складських операцій;

- мінімізацію людських помилок;

- автоматизацію процесу передачі даних між системою обліку та мобільними пристроями.

Мобільний додаток розробляється для роботи на терміналах із ОС Android або iOS, обладнаних зчитувачами штрих-кодів. Принцип його функціонування включає перенесення складських документів із системи бухгалтерського обліку на мобільний пристрій, виконання складських операцій (збір, підбір товарів, внесення додаткових даних, наприклад, кількості або серійних номерів), а також передачу фактичних даних назад у систему обліку для порівняння з обліковими показниками.

Мобільний додаток забезпечує інтеграцію з ERP-системою, розширюючи її функціонал за допомогою таких можливостей:

- автоматична передача даних через доступні канали зв'язку (Wi-Fi, мобільний інтернет, кабельне з’єднання);

- друк документів безпосередньо з мобільного пристрою на принтері;

- підтримка логістичних процесів, що підвищує ефективність складської роботи.

Впровадження такого рішення сприяє вдосконаленню складської логістики, покращуючи управління запасами та загальну продуктивність підприємства.

2. ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1. Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським рівнем вищої освіти)

Основні принципи роботи мобільних додатків, розроблених за допомогою різних підходів, наочно представлені на рисунку 2.1. Цей рисунок демонструє відмінності між нативними, гібридними та прогресивними веб-додатками (PWA) щодо архітектури, взаємодії з пристроями, використання ресурсів та механізмів доступу до функціоналу мобільних пристроїв.

Нативні додатки: працюють безпосередньо на операційній системі (iOS або Android), забезпечують максимальну продуктивність і доступ до апаратних функцій пристрою.

Гібридні додатки: використовують веб-технології для створення інтерфейсу та нативну оболонку для доступу до функцій пристрою.

PWA: працюють через браузер, але забезпечують досвід, схожий із нативними додатками, зберігаючи універсальність і легкість впровадження.

Такий підхід до класифікації дозволяє обрати оптимальну технологію для розробки додатків, враховуючи потреби користувача та особливості бізнесу.

Рисунок 2.1 - Основні підходи в реалізації мобільних додатків

Успішний досвід таких українських компаній, як Нова Пошта, Rozetka та Prom.ua, підтверджує, що мобільні додатки можуть значно підвищити рівень продажів. Згідно з даними статистичного бюро Criteo, 27% усіх електронних комерційних платежів у 2023 році були здійснені через мобільні додатки. Дослідження аналітичного агентства eMarketer показує, що протягом останніх 5 років спостерігається тенденція до збільшення часу, проведеного користувачами в додатках, в той час як кількість часу, що витрачається на перегляд мобільних веб-сторінок, значно зменшується.

Однією з ключових переваг використання мобільних додатків для бізнесу є можливість ретаргетингу - механізм розсилки реклами тим користувачам, які вже виявили інтерес до продукту, але не здійснили покупку. Мобільні функції ретаргетингу включають:

- аналіз поведінки клієнтів і нагадування про незавершені завдання за допомогою push-повідомлень;

- використання геолокаційних служб і аналіз історії покупок для формування персоналізованих рекламних пропозицій.

Мобільні додатки забезпечують значно зручніший доступ до сервісів порівняно з веб-сайтами, оскільки вони можуть працювати без підключення до Інтернету та використовувати вбудовані API мобільного пристрою. Крім того, мобільні додатки дають підприємцям можливість створити сильний бренд і розширити ринок збуту.

Впровадження програм лояльності через мобільні додатки стало ефективним інструментом взаємодії з клієнтами, дозволяючи вивчати їхню поведінку та стимули. Інтеграція з соціальними мережами сприяє залученню нових клієнтів.

Мобільні додатки поєднують переваги зручного формату з швидким доступом до актуальної інформації. Однак важливо уникати поширених помилок, таких як надмірне скорочення коду додатку або зменшення його розміру, що може знижувати продуктивність. Для ефективної роботи необхідно зосередитися на оптимізації користувальницького досвіду, скороченні часу на виконання завдань і на забезпеченні простоти інтерфейсу.

Основні функції мобільних додатків для компаній включають:

- оперативний доступ до даних для користувачів на всіх рівнях;

- бездротовий двосторонній зв’язок із технологічним обладнанням та інформаційною системою;

- отримання актуальної інформації в реальному часі;

- швидке прийняття обґрунтованих рішень.

Ці функції реалізуються через мінімалістичні, інтуїтивно зрозумілі інтерфейси з оптимальним дизайном, що дозволяє користувачам повністю зосередитись на технологічних процесах і виконанні посадових обов'язків.

Огляд існуючих рішень ERP-систем

Система SAP, розроблена мовою програмування ABAP (Advanced Business Application Programming), є однією з найбільш впливових у світі в сфері корпоративних програмних рішень. Вона підтримує широкий спектр бізнес-процесів і використовується для автоматизації управлінських і облікових

операцій в організаціях різних розмірів. ABAP — це внутрішня мова програмування для розробки додатків в системі SAP, яка дозволяє створювати додатки для сервера SAP NetWeaver. З Java, ABAP є основною мовою розробки для додатків SAP.

SAP забезпечує високий рівень інтеграції і автоматизації для великих, середніх і малих підприємств, що дозволяє оптимізувати управлінські процеси, такі як фінансовий облік, управління запасами, продажі, відвантаження і багато інших. Це особливо важливо в галузях, таких як торгівля, фінанси, високі технології і державне управління. Основною моделлю в SAP є SAP Business Suite, яка включає інструменти для точного налаштування бізнес-процесів і досягнення конкретних цілей підприємства.

Однією з ключових особливостей є використання технології баз даних Hana, яка дозволяє здійснювати швидку обробку даних, оскільки інформація більше не зберігається на жорсткому диску, а працює в пам'яті, що суттєво збільшує швидкість роботи системи.

SAP включає вбудовані стандартні функції для автоматичного обміну даними між різними модулями, наприклад, між продажами, відвантаженням і управлінням запасами, що дозволяє знижувати ризик помилок і покращує ефективність управлінських процесів.

Система SAP є прикладом високоякісної корпоративної ERP-системи, яка інтегрує різні функціональні області і дозволяє компаніям ефективно управляти своїми ресурсами та процесами, підвищуючи продуктивність і знижуючи витрати.

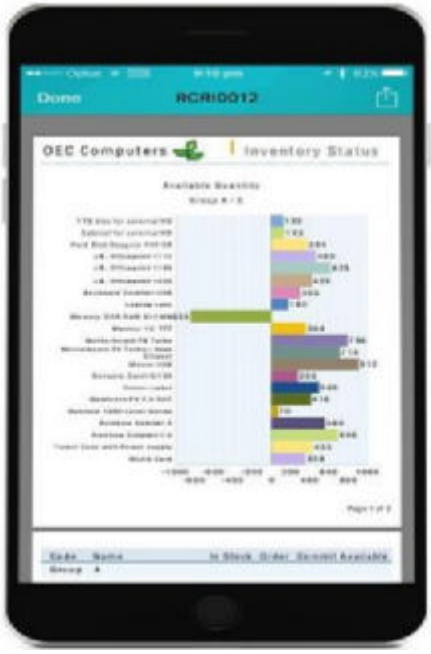

Рисунок 2.2 - Мобільна версія системи SAP

Хмарні ERP-системи, такі як Weclapp та Actindo, є революційними рішеннями для малих і середніх підприємств. Вони пропонують новий рівень доступності та зручності, оскільки підприємства можуть використовувати повноцінне ERP-обладнання без необхідності значних капіталовкладень у програмне або апаратне забезпечення та утримання власного IT-відділу.

Переваги хмарних ERP

Економічність: Витрати на хмарні ERP-системи можна обчислювати та контролювати набагато точніше, ніж для локальних рішень. Для малих підприємств це дозволяє скоротити витрати на інфраструктуру та технічне обслуговування.

Висока безпека: Хмарні провайдери пропонують багаторівневі рішення для захисту даних, що часто перевищують корпоративні стандарти безпеки.

Без необхідності власного IT-відділу: Програмне забезпечення як послуга (SaaS) не вимагає інвестицій у власні сервери та апаратне забезпечення, а також знижує потребу в спеціалізованому IT-обслуговуванні.

Швидке впровадження: Такі платформи, як Weclapp, дозволяють здійснити впровадження ERP-системи в короткі терміни. Модульний підхід

дозволяє підприємствам платити лише за потрібні функції, що дає змогу гнучко адаптувати рішення під конкретні потреби бізнесу.

Інтуїтивно зрозумілий інтерфейс: Weclapp має сучасний інтерфейс, який полегшує користування і швидку адаптацію співробітників до нової системи. Вбудовані довідки та рекомендації значно спрощують процес освоєння системи.

Платформи для інтеграції

Weclapp: Хмарна ERP-система, яка підтримує CRM, ERP, бухгалтерію та білінгове програмне забезпечення. Вона забезпечує низькі витрати на впровадження та безкоштовні оновлення, що робить її привабливою для малих підприємств, які хочуть зекономити на інфраструктурі.

Actindo: Це рішення орієнтоване на об'єднання всіх процесів бізнесу — від торгівлі до логістики та складування - в єдину централізовану систему на основі API. ActindoCore забезпечує високу продуктивність і масштабованість для майбутнього зростання, дозволяючи легко інтегрувати додаткові канали продажу через API. Використання інтерфейсів на базі Android дозволяє оптимізувати управління логістикою та доставкою.

Хмарні ERP-системи значно спрощують процеси управління підприємствами, знижуючи витрати на інфраструктуру і дозволяючи малим компаніям отримати доступ до потужних інструментів без необхідності великих інвестицій. Завдяки їх зручності і масштабованості, бізнес може швидко адаптувати систему під свої потреби, а також забезпечити високий рівень безпеки та ефективності.

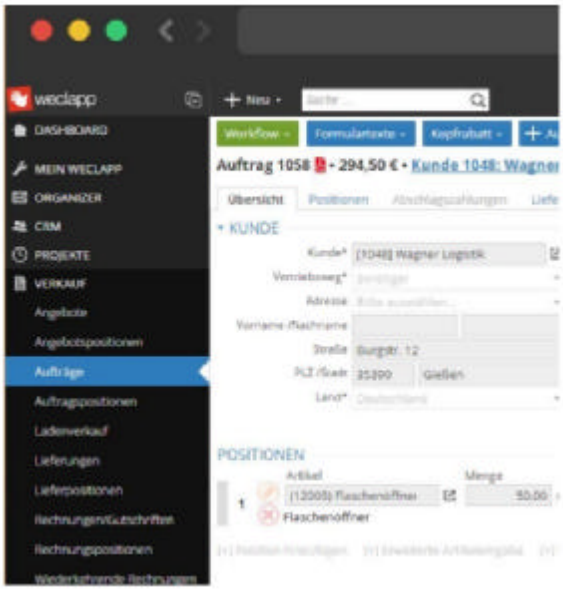

Рисунок 2.3 – Інтерфейс німецької системи WeClapp Actindo

Рисунок 2.4 – Система автоматизації Actindo

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Впровадження сучасних ERP-систем, які базуються на вимогах користувачів, дійсно відкриває нові можливості для підвищення продуктивності

та зниження витрат у компанії. У порівнянні з класичними статичними системами, сучасні рішення не лише збирають інформацію з бізнес-процесів, а й активно аналізують ці дані для прийняття обґрунтованих рішень. Вони охоплюють усі аспекти діяльності підприємства, включаючи управління матеріальними ресурсами, фінансами, облік, управління персоналом, маркетинг, дослідження та багато інших функцій. Важливо, що всі ці системи інтегровані та використовують одну базу даних, що дозволяє усунути функціональні розмежування між різними відділами.

Роль мобільних додатків у приладобудівній галузі

Хоча мобільні додатки досить повільно набирають популярність у приладобудівних компаніях, їх застосування в процесах розрахунків, планування та управління виробництвом має великий потенціал. Мобільні додатки можуть істотно спростити та прискорити такі процеси, як:

- **Розрахунок матеріальних потреб**: мобільний додаток може в реальному часі забезпечувати точний розрахунок необхідних матеріалів для виробничих процесів, що дозволяє зменшити витрати та оптимізувати запаси.
- **Керування клієнтською базою**: завдяки мобільним додаткам можна швидко отримати інформацію про стан замовлень і зворотний зв'язок з клієнтами.
- **Фінансові операції**: мобільні додатки можуть бути використані для отримання залишків коштів, їх перерахування або здійснення фінансових операцій на ходу.
- **Моніторинг виробничих процесів**: вони можуть надавати дані про стан виробничих приміщень та обладнання, що допомагає оперативно реагувати на можливі проблеми та здійснювати необхідні коригування.
- **Управління бухгалтерськими та кадровими питаннями**: враховуючи мобільні можливості, компанії можуть контролювати питання бухгалтерії та персоналу в реальному часі, не будучи прив’язаними до робочих станцій.

Класифікація мобільних додатків для приладобудівних компаній

- **Інформаційно-довідкові додатки**: вони забезпечують доступ до актуальної інформації про стан замовлень, виробничі ресурси, фінанси та інші аспекти бізнесу.
- **Торговельні майданчики**: додатки для здійснення онлайн-продажів або управління комерційними операціями.
- **GPS-вимірювання, навігація**: використовуються для управління логістикою, стеження за доставкою або планування маршрутів для персоналу.
- **Прогнозування та оцінка виконання замовлень**: додатки, які допомагають оцінити, як успішно буде виконано замовлення та чи буде воно прибутковим.
- **Економіка приладобудівної галузі**: мобільні додатки, які допомагають прогнозувати та оцінювати економічні аспекти, такі як витрати, доходи та прибутковість.

Тренди використання мобільних додатків

Як показує дослідження, значна частина підприємців у світі активно використовує мобільні додатки для оптимізації своїх бізнес-процесів. Опитування в США та Франції показують, що мобільні додатки можуть значно підвищити ефективність бізнесу, а доступ до Інтернету є ключовим фактором для впровадження цих технологій у сучасний бізнес.

Впровадження мобільних додатків у приладобудівних компаніях відкриває нові можливості для автоматизації та оптимізації процесів. Це не лише підвищує ефективність роботи, а й дозволяє зменшити витрати, прискорити реакцію на потреби клієнтів та підвищити якість управління виробничими процесами.

Згідно з даними Google Україна, майже 60% всього трафіку в країні генерується через мобільні пристрої. Як позитивний приклад можна навести запуск мобільного додатку m-Agri для малих фермерів, створеного Міністерством агрополітики України у співпраці з компанією «Київстар». Цей

додаток надає зручний доступ до 17 передових практик, галузевих цінових котирувань, новин, бази знань, ресурсів для професійного навчання та онлайн-консультацій. Ми також проаналізували лідерів ринку IT-продуктів для компаній у галузі приладобудування. Основними критеріями є якість програмної реалізації, конкурентні переваги порівняно з іншими системами, цінова політика та зручність інтерфейсу, а також наявність мобільних додатків. Однак розробка окремих додатків для різних мобільних операційних систем вимагає значних витрат часу і ресурсів, оскільки кожна система має свою мову програмування і специфічні вимоги. Порівняльні характеристики засобів розробки мобільних додатків для різних платформ наведені в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика інструментів розробки мобільних операційних систем

Назва критерію	**Apple iOS**	**Android**	**Blackberry OS**	**Windows Phone**
Мова програмування	Objective–C, C, C++, Swift	Java, Kotlin, інші	Java	C#, VB.NET, інші
Засоби розробки	Xcode	Android SDK	BB Java Eclipse Plug–in	Visual Studio, Windows Phone development tools
Формат пакування	.app	.apk	.cod	.xap
Магазин	Apple App Store	Google Play	Blackberry App World	Windows Phone Marketplace

У таких умовах компанії залучають різних фахівців, таких як розробники, дизайнери та тестувальники, для кожної окремої платформи, що призводить до збільшення вартості розробки. Витрати на створення програми залежать від низки факторів, таких як складність реалізації, гонорари розробників, інтеграція з нативними функціями та інші. Для порівняння вартості впровадження основних операційних систем (детальніше в таблиці 2.2) використовувався онлайн-калькулятор Venturepact, який враховує такі функції: шаблон інтерфейсу користувача для операційної системи, 7–12 робочих екранів програми, двоетапну аутентифікацію користувача, основні заходи захисту, інтеграцію з електронною платіжною системою, створення бази даних для програми та управління повідомленнями.

Таблиця 2.2 – Порівняння вартості розробки мобільних додатків для різних операційних систем, дол. США

Регіон	Платформи			
	Android	iOS	Android та iOS одночасно	Гібридний додаток
США, Канада, Західна Європа, Австралія	3900	3900	7800	2600
Східна Європа, Середній Схід, Центральна та Південна Америка	1950	1950	3900	1300
Південна Азія, Східна Азія, Південно–Східна Азія, Африка	1200	1200	2400	800

Підсумовуючи результати, можна зазначити, що кросплатформна гібридна розробка вимагає в середньому на 66,7% менше витрат, порівняно з одночасною нативною розробкою окремих додатків для Android та iOS. Для

вибору методу розробки мобільного додатку було проведено SWOT-аналіз (таблиця. 2.3).

Додатково, аналіз статистики використання PWA (Progressive Web App) порівняно з традиційними веб-сайтами показує такі результати:

- збільшення мобільного трафіку;
- 1,5-кратне пришвидшення завантаження та встановлення додатку;
- 2,5-кратне зменшення розміру додатку в пам'яті пристрою;
- збільшення середньої конверсії на 52%, залучення на 137%, кількості переглядів сторінок на 133,67%;
- продовження середньої сесії на 78%;
- зниження рівня відмов на 42,86% порівняно з мобільними веб-сайтами.

Таблиця 2.3 – SWOT–аналіз гібридних і PWA мобільних додатків

Сильні сторони	Слабкі сторони
Внутрішнє середовище	
Швидкість розробки Кросплатформність додатків Низькі витрати на розробку додатку Єдина база коду Безкоштовні інструменти розробки Безпечність Швидкий випуск на ринок	1 Нижча продуктивність, порівняно з нативними додатками. 2 Обмежений доступ до API смартфону
Можливості	Загрози
Зовнішнє середовище	
Можливість роботи додатку незалежно від версії операційної системи мобільного пристрою Реалізація PWA без зміни кодової бази	1 Висока залежність від сторонніх бібліотек та фреймворків

Серед основних переваг гібридних мобільних додатків і прогресивних веб-додатків можна виділити такі:

- Економічна ефективність. Розробка одного додатку для кількох платформ дозволяє значно знизити витрати часу та ресурсів. Такі технології також спрощують процес оновлення та додавання нового функціоналу для всіх пристроїв одночасно, забезпечуючи однаковий досвід для користувачів на різних платформах.

- Простота розробки та доступ до інструментів. Використання стандартних веб-технологій, таких як HTML, CSS та JavaScript, а також безкоштовних бібліотек, плагінів і фреймворків робить розробку простішою і доступнішою. У сучасних умовах будь-який веб-розробник може створити гібридний мобільний додаток або прогресивний веб-додаток без потреби вивчати нові технології.

Основні переваги прогресивних веб-додатків (PVA) включають:

- використання без доступу до Інтернету. API локального накопичувача дозволяє користувачам із повільним Інтернет-з'єднанням працювати з додатками навіть без активного підключення;

- зручний інтерфейс. PVA забезпечують інтерфейс, максимально схожий на нативний додаток, що забезпечує комфорт користувача;

- швидка збірка. На відміну від нативних додатків, PVA не потребують установки на пристрій і можуть бути завантажені безпосередньо з браузера.

Однак є й певні обмеження, особливо для пристроїв Apple iOS, такі як:

- підтримка з iOS 11.3;

- обмеження на зберігання локальних даних і файлів до 50 Мб;

- видалення даних при тривалому відмові від використання;

- обмежений доступ до нативних функцій, включаючи виконання коду у фоновому режимі, персональну інформацію, більшість сервісів Apple, push-повідомлення та інтеграцію з Siri.

Ці обмеження є суттєвим недоліком для користувачів iOS, які є важливою частиною цільової аудиторії українських підприємців. Цей сегмент традиційно вважається більш платоспроможним, ніж користувачі Android або Windows Phone. Структуру ринкових часток мобільних операційних систем можна побачити на рисунку 2.5.

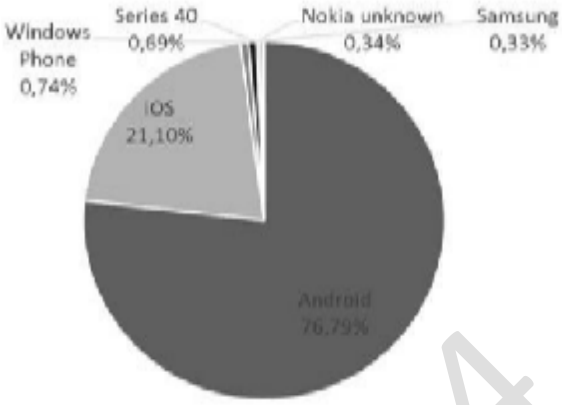

Рисунок 2.5 – Структура ринкових часток мобільних операційних систем України в 2023 році.

Вибір платформи для розробки гібридних мобільних додатків є критичним для забезпечення їхньої ефективності та функціональності. Існує низка популярних інструментів, фреймворків та бібліотек, які активно використовуються в індустрії для розробки таких додатків. Серед них можна виділити такі ключові технології, які дозволяють створювати мобільні додатки з мінімальними витратами часу та ресурсів, забезпечуючи при цьому високу продуктивність і стабільність:

- React Native - один із найбільш популярних фреймворків для створення гібридних мобільних додатків, який дозволяє використовувати одну кодову базу для обох платформ: Android і iOS;
- Flutter - ще один потужний фреймворк від Google, який дозволяє створювати нативні додатки для різних платформ за допомогою одного коду;

- Xamarin - інструмент для розробки додатків за допомогою C# та .NET, який дозволяє створювати кросплатформенні рішення для мобільних пристроїв;
- Ionic - фреймворк, який поєднує HTML, CSS та JavaScript для створення додатків, що працюють на різних платформах;
- PhoneGap - інструмент, який використовує веб-технології для створення мобільних додатків, які можна запускати на різних платформах.

Кожен із цих інструментів має свої переваги та обмеження, і вибір залежить від конкретних вимог проекту та рівня компетенції розробників. Структура та функціональність кожного з цих інструментів можуть бути розглянуті детальніше в графічному вигляді на рисунку 2.6.

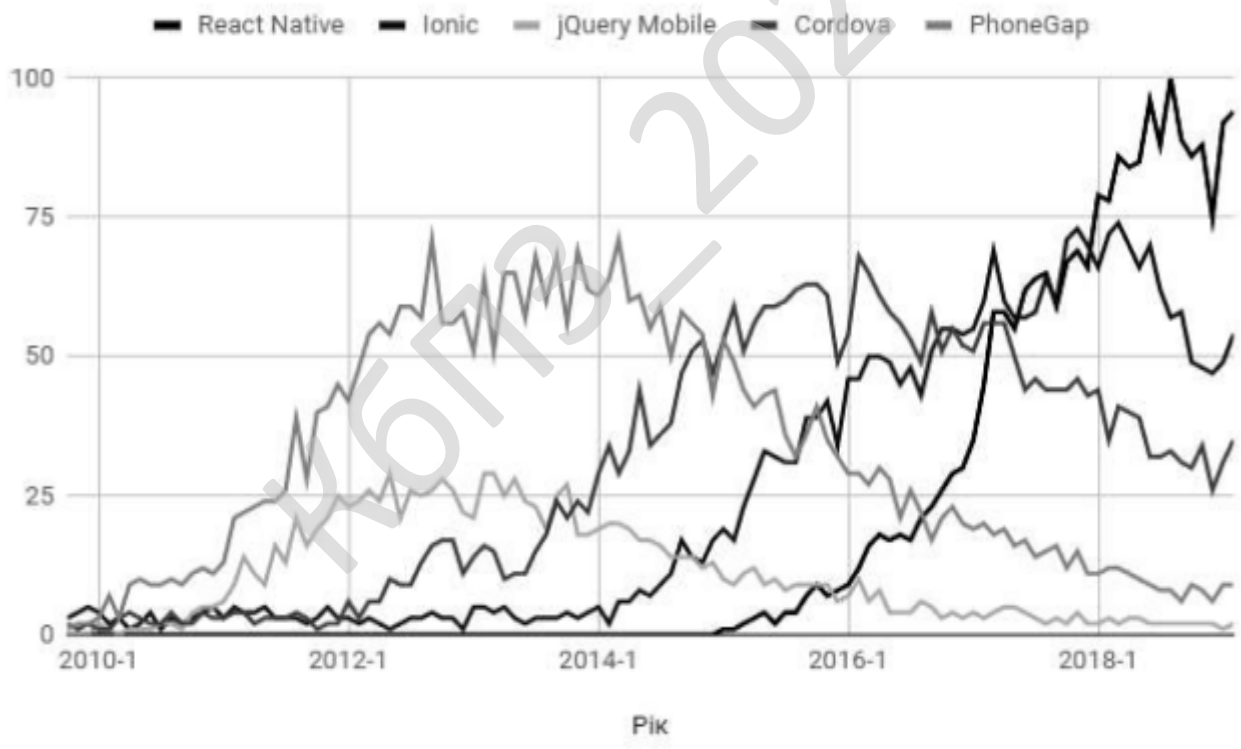

Рисунок 2.6 – Інтенсивність пошукових запитів назв інструментів розробки гібридних додатків у відсотках в системі Google в 2010–2018 роках.

Одним із найпопулярніших інструментів для розробки гібридних мобільних додатків і прогресивних веб-додатків (PVA) є **IonicFramework**. Цей

фреймворк працює на основі HTML5 та використовує веб-фреймворк **Angular**, дозволяючи розробникам створювати додатки з інтерфейсами, які схожі на нативні додатки для платформ Apple iOS і Google Android. Однією з ключових переваг Ionic є широкий функціонал, включаючи мобільні компоненти, інтерактивні парадигми, типографіку та базову тему, яку можна адаптувати під специфічні вимоги.

Проте за останні кілька років **ReactNative** отримав значний поштовх у розвитку. Це фреймворк для створення кросплатформних мобільних додатків, який не використовує технології WebView чи HTML. Програми на React Native створюються за допомогою поєднання JavaScript та XML-розмітки. Оскільки React Native компілює додатки з використанням власних API для Android (на основі Java) та iOS (на основі Objective-C), програми працюють значно швидше і ефективніше порівняно з гібридними додатками на основі WebView.

Програми, розроблені за допомогою React Native, включають популярні додатки, такі як **Facebook**, **Instagram**, **SoundCloud Pulse**, **Pinterest** і **Skype**. Цей фреймворк вважається майбутнім мобільної розробки завдяки своїй продуктивності та можливості створювати високоефективні мобільні додатки.

При виборі платформи для розробки гібридного мобільного додатку або прогресивного веб-додатку важливо враховувати конкретні бізнес-завдання, які потрібно вирішити, а також вимоги до функціональності та ефективності додатку.

2.3 Розгорнута постановка завдання

Ринок мобільних додатків сьогодні є одним із найперспективніших і активно розвиваються напрямків, що має значний потенціал для покращення ефективності виробничих процесів. У поєднанні з бездротовими мережами мобільні програми здатні збільшити продуктивність, оптимізувати

використання часу та ресурсів. Водночас вони відкривають нові можливості для бізнесу, забезпечуючи доступ до публічної інформації та основних послуг.

Дослідження показали, що одним з головних викликів є збільшення обсягів реалізації продукції, що потребує застосування сучасних засобів просування та реклами, включаючи мобільні додатки. Оскільки продукція компанії реалізується на внутрішньому ринку, специфіка та широкий асортимент аналогічних зарубіжних товарів вимагають впровадження ефективних інструментів просування для досягнення прибуткових результатів.

Мета роботи полягає в розробці мобільного додатку, який підвищить ефективність діяльності будівельних компаній через розширення ринків збуту та зростання доходів.

Для досягнення цієї мети передбачено виконання таких завдань:

- визначення ролі мобільного додатку в підвищенні ефективності виробництва;

- дослідження та аналіз сучасних типів мобільних систем і методів їх розробки для різних платформ;

- побудова типового структури та визначення функціональних принципів використання мобільних додатків;

- аналіз ринку мобільних додатків для виробничих компаній та визначення їх основних можливостей;

- розробка мобільного додатку для просування та реклами продукції приладобудівного підприємства в Україні.

Відповідно до технічного завдання на магістерську роботу, передбачається впровадження програмного забезпечення, яке демонструватиме функціонування додатку для управління ERP-системою. Для виконання цієї роботи необхідно виконати наступні етапи:

Аналіз існуючих аналогових систем:

- провести детальний аналіз наявних аналогічних систем для виявлення їх сильних і слабких сторін. Результати цього аналізу повинні бути враховані

при подальшій розробці системи, що допоможе уникнути повторення помилок і використати позитивний досвід.

Побудова системи контролю за технологічним обладнанням:

- обрати та обґрунтувати метод побудови автоматизованої системи контролю за роботою технологічного обладнання. Розробити функціональні та структурні схеми цієї системи, що дозволяють ефективно контролювати виробничі процеси.

Розробка системного програмного забезпечення:

- розробити програмне забезпечення, яке відповідатиме технічному завданню. Це включає створення блок-схем програмних алгоритмів та підпрограм для вирішення поставлених задач.

Організація інтерфейсу користувача:

- створити інтерфейс користувача, який буде зручним і зрозумілим. Він має забезпечувати можливість формування та відображення повідомлень про некоректні дії користувача і нестандартні ситуації.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Термін "корпорація" означає об'єднання кількох компаній. Однією з основних умов для компаній, які прагнуть стати корпорацією, є централізоване управління інформацією, що передбачає створення єдиної бази даних для всіх підрозділів, складів, магазинів тощо.

Корпорація має складну ієрархічну структуру управління, оскільки вона є багатопрофільним утворенням. Її система управління повинна функціонувати як взаємопов'язаний комплекс структурних елементів, що відповідає загальному поняттю системи, яка працює в інтересах компанії загалом.

Інформаційна система (ІС) – це система, що за допомогою спеціальних інструментів збирає, передає та обробляє дані, надаючи оброблену інформацію співробітникам різних рівнів для виконання управлінських функцій та підтримки бізнес-процесів компанії. Її структуру можна уявити як набір окремих елементів, що працюють у взаємодії. Ці елементи утворюють підсистеми, як показано на рисунку 3.1.

Корпоративна інформаційна система (КІС) - це інтегрована платформа для управління компанією або корпорацією, що дозволяє ефективно використовувати всі наявні дані підприємства та виконувати їх детальний аналіз. Вона також забезпечує документообіг і електронне діловодство, керуючи всіма етапами життєвого циклу товару на підприємстві: від прийому, зберігання, переміщення до передачі.

КІС автоматизує методи та процеси управління за допомогою сучасних технологій, включаючи технічну підтримку, математичне моделювання, програмні рішення, ІТ-інфраструктуру, організаційні та юридичні інструменти.

Основна мета корпоративної системи - ефективне управління ресурсами компанії, що сприяє підвищенню прибутковості та оптимізації матеріальних

витрат. Вона дозволяє скоротити зайві переміщення працівників і підвищити рівень продажів.

Система може вважатися корпоративною інформаційною лише за умови відповідності таким критеріям:

- надійність і захист даних;
- віддалений доступ до системи та її додатків;
- можливість технічної підтримки та наявність необхідних інструментів;
- гнучкість для інтеграції з новими підрозділами, наприклад, передача даних до філій;
- інтеграція нових даних та консолідація інформації в загальній системі;
- обмін даними з іншими програмними продуктами та додатками через створення відповідних модулів;
- аналіз стану системи та процесів роботи.

Впровадження КІС забезпечує низку переваг, зокрема:

- керівники та власники компанії можуть оперативно отримувати інформацію про стан підприємства та відстежувати рух товарів, навіть через мобільні додатки.;
- надається огляд поточних процесів в організації;
- підвищується ефективність управління завдяки постійному оновленню даних і контролю;
- скорочуються строки збору замовлень і кількість операцій.

Сьогодні існує багато типів управлінських інформаційних систем, які допомагають вирішувати специфічні бізнес-завдання.

Нижче наведено деякі системи, які можуть бути ефективно впроваджені на підприємствах приладобудівної галузі.

CALS (Computer-Aided Logistic Support) – автоматизована підтримка логістики, що забезпечує ефективну доставку товарів.

SCM (Supply Chain Management) – система для управління та контролю всіх етапів ланцюга постачання.

MRP (Material Requirements Planning) – оптимізує управління складськими приміщеннями, розподіл матеріалів у відповідних зонах, створення календаря потреб для замовлень у постачальників, а також допомагає зменшити обсяги надлишкових запасів.

CAD (Computer-Aided Design) – автоматизує проектування готової продукції, підвищуючи швидкість і точність процесу.

CAM (Computer-Aided Manufacturing) – забезпечує управління інструментами та обладнанням для розробки продукції, включаючи машини, верстати, деталі, дерево- та металообробне обладнання.

CAE (Computer-Aided Engineering) – автоматизує виконання всіх інженерних розрахунків, що підвищує точність і ефективність розробки.

PLM (Product Lifecycle Management) – управляє повним життєвим циклом продукту, від проектування до утилізації.

MRP II (Manufacturing Resources Planning) – планує виробничі процеси, контролює завантаження і використання виробничих потужностей.

CSRP (Customer-Synchronized Resources Planning) – фокусується на синхронізації планування та контролю ресурсів відповідно до потреб клієнтів.

CRM (Customer Relationship Management) – система управління відносинами з клієнтами, яка забезпечує ефективну взаємодію, підвищення лояльності та управління замовленнями.

Використання цих систем дає можливість підприємствам підвищити продуктивність, автоматизувати ключові процеси, оптимізувати використання ресурсів та поліпшити взаємодію з клієнтами.

3.2 Розробка структурної схеми

Структурна схема відображає взаємодію основних компонентів програми, визначає їх функціональні завдання та спосіб комунікації між ними.

Вона не є повністю інформативною з точки зору відстеження передачі й обробки даних, тому для комплексних проєктів створюють окремі структурні схеми для різних частин програми відповідно до технічного завдання.

При створенні структурної діаграми часто використовують метод покрокової деталізації. Це дозволяє чітко визначити всі складові компоненти програми, такі як підпрограми, бібліотеки чи бази даних, а також взаємозв'язки між ними. Наприклад, за допомогою посилань можна показати обмін даними між компонентами або підключення нових бібліотек.

Під час розробки програмного забезпечення структурна схема допомагає:

- розподілити програму на окремі модулі для спрощення її розуміння та подальшої роботи над складною архітектурою;

- відобразити логіку програми, вказати місця підключення бібліотек та їх функціональність;

- продемонструвати ієрархічну взаємодію модулів та їх функції.

Хоча така схема зображає взаємозв’язки між модулями і дає загальне уявлення про функціональність системи, вона не показує конкретні процеси обробки даних. Тому до схеми додають розшифрування функцій, які забезпечують зв'язок між модулями.

На прикладі структурної схеми (рисунок 3.1) можна побачити зовнішні специфікації програми, що дають розуміння її функціональних компонентів, їх відповідальності та взаємодії із вхідними і вихідними даними. Це особливо важливо для визначення типів структур даних і їх використання.

Для ERP-систем взаємодія відбувається в межах мережевої інфраструктури. Дані про компанію (бухгалтерські документи, договори, кадрові записи тощо) надходять у базу даних, де вони аналізуються і зберігаються з урахуванням специфіки діяльності. Завдяки цьому система забезпечує швидке та коректне отримання аналітичної інформації, необхідної для ефективного управління компанією.

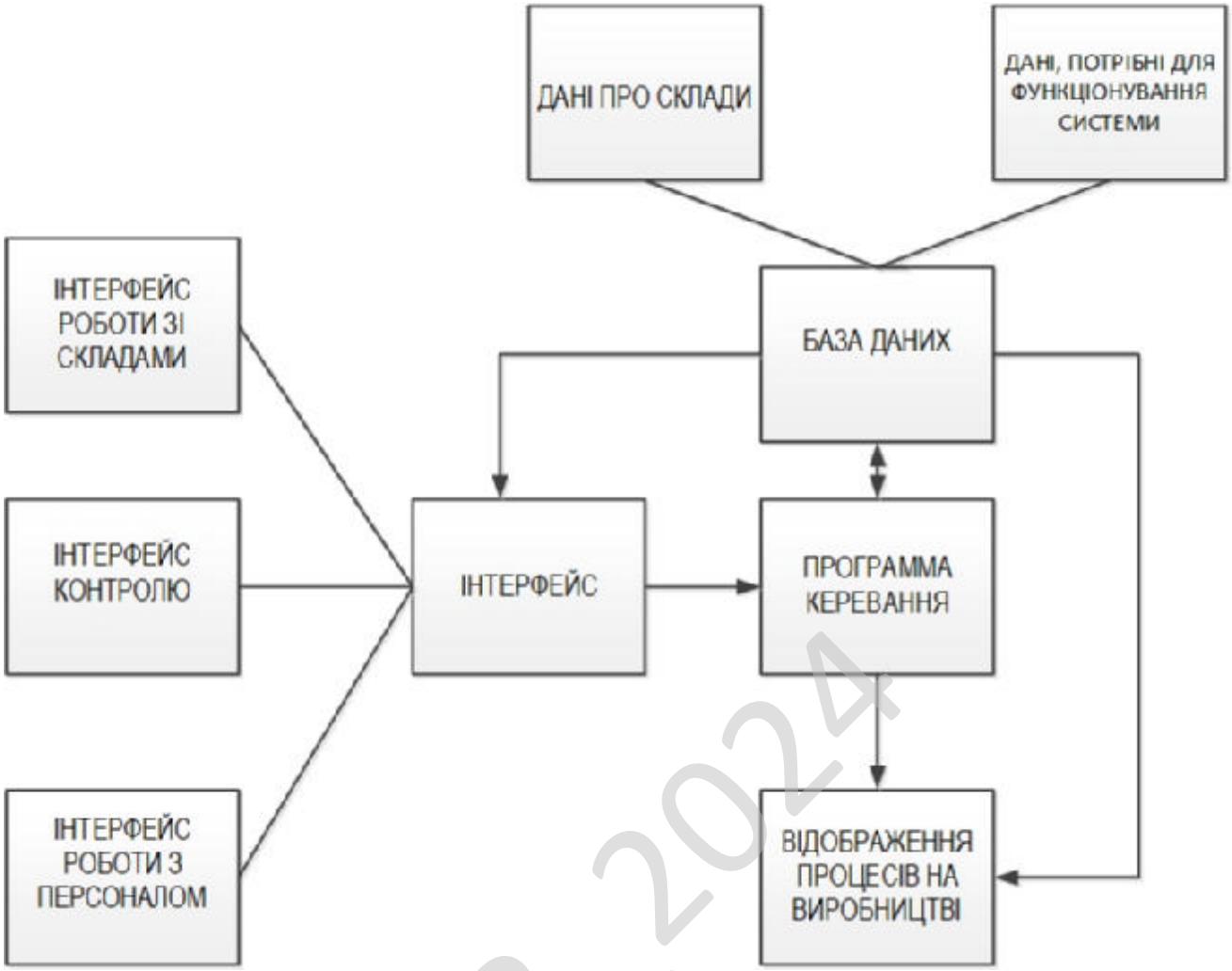

Рисунок 3.1 - Структурна схема

Поділ програми на окремі модулі здійснювався за принципом деталізації від загального до конкретного. Кожен модуль виконує окреме завдання, функціонуючи як окремий сервіс. Хоча існують альтернативні підходи, такі як побудова ієрархій, циклічний перегляд параметрів або використання шаблону MVC, обраний підхід забезпечує зручність налаштування, легкість тестування та підтримки окремих компонентів.

Структурна схема наочно відображає основний принцип роботи програми, включаючи основні та другорядні блоки, їхні функції та взаємодію між собою. Це дозволяє краще зрозуміти функціонування програми, що значно спрощує її подальше обслуговування та вдосконалення.

3.3 Розробка функціональної схеми

Функціональна схема, або схема даних, демонструє взаємодію компонентів програми, визначаючи склад даних і їх призначення. При створенні такої схеми використовуються стандартизовані позначення. Вона дозволяє відстежувати виконання існуючих вимог і функцій, аналізувати, які компоненти та модулі застосовуються, а також як вони взаємодіють між собою через вхідні та вихідні дані. Схема ілюструє роль кожного функціонального елемента в додатку і його зв'язок із іншими компонентами. Для її розробки було взято за основу конструктивну схему.

Функціональна схема дозволяє простежити принцип роботи програми, включаючи взаємодію клієнтської та серверної частин, а також бібліотеки й технології, які використовуються для зберігання та передачі даних. Такий тип діаграм зазвичай застосовується для спрощеної демонстрації алгоритму, що дає змогу зрозуміти виконувані функції, використовувані дані та результати, які отримуються. Це спрощує оцінку складності алгоритму, допомагає зрозуміти його принцип роботи та забезпечує контроль можливих змін або коригувань.

Для опису компонентів і їх взаємодії використовувався національний стандарт. У схемі прямокутники позначають окремі функціональні частини або групи, об'єднані за функціональним принципом, де кожна група має унікальну позначку та назву.

Функціональні схеми можуть бути представлені різними способами, зокрема у вигляді принципових схем або змішаних структур. Однак у будь-якому випадку вони мають чітко відображати призначення і логіку роботи системи.

На схемі (рисунок 3.2) показано:

- умовна позначка в прямокутнику, якщо це функціональна частина, зазвичай включає текстове описання, що визначає її призначення, наприклад:

- назва функції або процесу (наприклад, "Обробка даних");

- код чи ідентифікатор елемента (наприклад, "F1" або "Module_01");

- додаткові деталі (наприклад, входи/виходи: "Вхід: дані клієнта, Вихід: рахунок").

Для позначення положення та дії функціональної частини використовуються стандартні графічні елементи:

- прямокутник - основний блок, що символізує окрему функцію або модуль;
- стрілки між прямокутниками — позначають напрямок потоку даних чи виконання операцій;
- стрілки можуть мати підпис (наприклад, "Передача даних", "Вхідний потік");
- вхід/вихід - позначається у вигляді скошеного прямокутника або трапеції, що вказує на взаємодію з зовнішніми даними;
- процес - круг, овал або ромб, якщо позначається рішення чи умова (наприклад, "Так/Ні").

Ці позначення відповідають стандартам ISO або національним технічним стандартам.

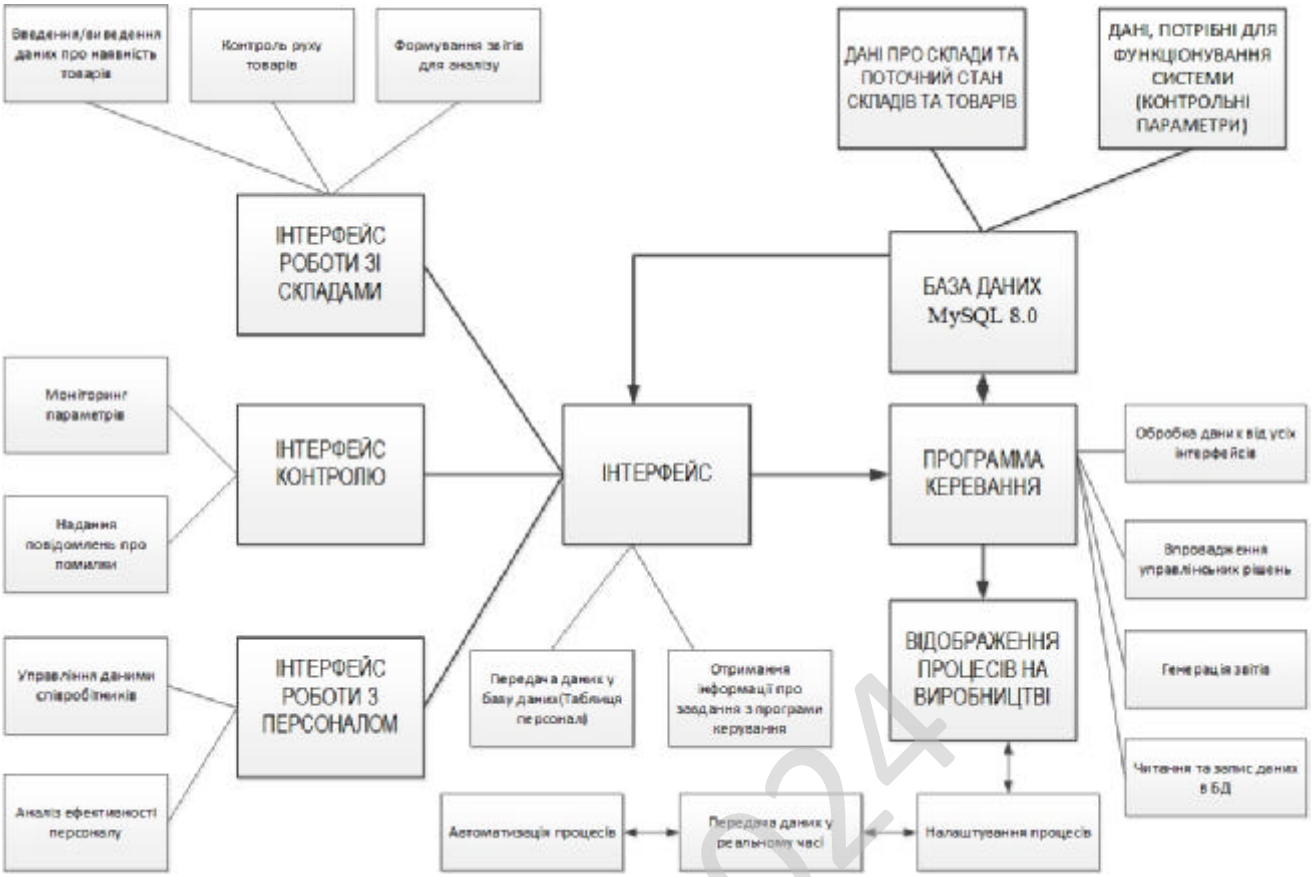

Рисунок 3.2 - Функціональна схема системи

У випадках, коли використовується принципова схема, функціональні частини мають бути відображені з урахуванням їхнього реального розташування у системі.

Це дозволяє відобразити фізичне або логічне розташування елементів у структурі програми або пристрою.

У таких схемах:

- кожна функціональна частина має своє графічне позначення, пов’язане з її роллю у системі (наприклад, блоки, з'єднання, вузли);

- відображаються усі зв’язки між частинами у вигляді стрілок, ліній або інших графічних елементів;

- список елементів не використовується, оскільки всі деталі та функції відображаються безпосередньо на схемі, що робить її самодостатньою;

Коли функціональна схема розробляється без використання принципової схеми, застосовуються загальні правила:

- функціональні частини позначаються спрощено, наприклад, за допомогою прямокутників, кожен із яких відповідає за певну функцію чи компонент;
- зв’язки між компонентами вказуються за допомогою напрямних стрілок, які описують потік даних, сигналів чи взаємодій;
- може бути використаний супровідний список елементів, де деталізуються всі компоненти, їхні функції та взаємодії;
- усі графічні позначення відповідають прийнятим стандартам, таким як ISO, IEC чи національні технічні стандарти.

Цей підхід допомагає представити структуру навіть за відсутності детальної принципової схеми, забезпечуючи зрозуміле зображення функціональних зв’язків.

3.4 Розробка діаграми процесів

При моделюванні поведінки розробленої системи можна спостерігати, як змінювалася її робота, аналізувати стан програми, вхідні й вихідні дані, а також оцінювати вплив цих змін на розробку.

Це дозволяє зрозуміти, як функціонують алгоритми в додатку і які результати вони генерують залежно від вхідних даних.

На діаграмі процесу за допомогою графічних об’єктів відображаються операції, які виконуються програмою, і їхні взаємозв’язки, що позначаються стрілками. Так на діаграмі (рисунок 3.3) можна побачити всі фази роботи програми: від моменту, коли користувач надсилає запит, до його потрапляння на сервер, обробки контролером, отримання даних із бази і формування відповіді для користувача. Графічні об’єкти діаграми відображають окремі процеси, а стрілки вказують послідовність передачі управління між ними.

Окрім процесних діаграм, існують також структурні діаграми, які зосереджуються на статичних аспектах даних і зв'язках між об'єктами. Відмінність полягає в тому, що діаграми процесів ілюструють динаміку програми, тоді як структурні акцентуються на статичних зв'язках. Основною характеристикою діаграм процесів є їх динамічність і неперервність, що дозволяє аналізувати сценарії виконання програмних дій.

Проблеми можуть виникати у випадках, коли об'єкт повинен одночасно виконувати кілька завдань у різних місцях. Це зазвичай трапляється через помилки у логіці або невідповідність послідовності дій. Тому важливо, щоб діаграма процесів точно відображала всі кроки алгоритму і сценарій виконання не суперечив фактичній послідовності.

Основна мета створення таких діаграм полягає у візуалізації послідовності дій, які виконуються для досягнення кінцевого результату. Спостерігаючи за тим, які дані отримують процеси, як вони обробляються і передаються користувачеві, можна передбачити, як програма має функціонувати.

Для побудови діаграм використовуються різні методи, проте найбільш ефективним вважається покроковий алгоритмічний підхід, який дозволяє чітко відобразити логіку і послідовність дій.

На основі проведених проектних робіт і реалізованих технічних рішень можна зробити висновок, що обрані дизайнерські підходи відповідають поставленим завданням.

Як видно з отриманих результатів, програма успішно виконує всі поставлені функції та відповідає початковим вимогам проекту.

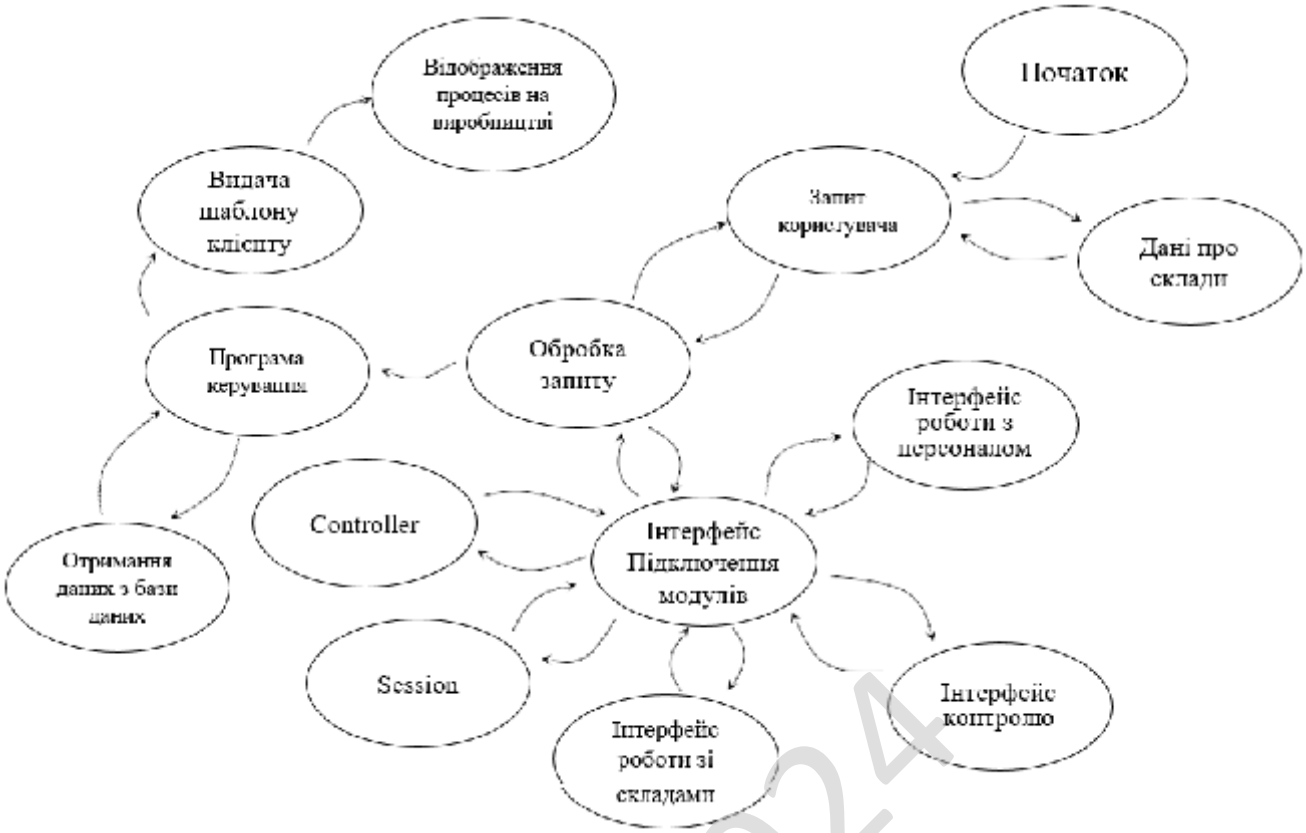

Рисунок 3.2 – Діаграма процесів системи

Розроблена діаграма відображає функціонування додатку, включаючи як клієнтську, так і серверну частини. Вона демонструє взаємодію окремих компонентів програми та результат, який отримує користувач під час роботи з програмним забезпеченням.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Сутність ERP-систем

ERP-система (Enterprise Resource Planning) є інтегрованою платформою для управління ресурсами підприємства, що дозволяє додавати нові модулі з часом. Вона об’єднує всі ресурси компанії, працюючи на основі єдиної бази даних, яка забезпечує контроль над усіма бізнес-процесами. Така система дозволяє одночасно встановлювати декілька з’єднань, підтримуючи роботу з кількома клієнтами, кожен із яких може мати різні рівні доступу та повноважень.

ERP-система виникла завдяки розвитку класичних систем MRP (Material Requirements Planning) та MRP II (Manufacturing Resource Planning), до яких додали модулі фінансового управління (FRP). В результаті ці інтегровані системи отримали нові функції та були названі ERP-системами. Завдяки своїй здатності ефективно планувати діяльність підприємств, ERP-системи стали одними з найпопулярніших рішень на ринку.

На рисунку 4.1 наведено основні функції, що виконують різні модулі ERP-систем. З часом у ці системи додали нові модулі та інтеграції, що значно розширили можливості управління та посилили контроль над безпекою даних.

З 1999 року розробники активно працюють над вдосконаленням ERP-систем, додаючи нові функціональні можливості, оптимізуючи бізнес-процеси підприємств і пришвидшуючи виконання робочих завдань. Це сприяло автоматизації підприємств, підвищенню продуктивності та ефективності їхньої діяльності.

Рисунок 4.1 – Історія розвитку ERP–системи

Еволюція ERP-систем: від ERP до ERP II

Ідея створення ERP-систем полягала у створенні інструменту, здатного працювати з усіма внутрішніми ресурсами компанії. Завдяки своїй гнучкості, ці системи з часом отримали нові функції та права шляхом інтеграції додаткових модулів. Наприклад, модуль SCM (Supply Chain Management) надає можливість

контролювати та керувати ланцюгами постачання, а CRM (Customer Relationship Management) спрощує управління взаємовідносинами з клієнтами.

З розвитком ERP-систем виник новий їх тип — ERP II (Enterprise Resource and Relationship Processing). Відмінністю ERP II стало те, що вони не тільки оптимізують внутрішні процеси компанії, але й забезпечують управління зовнішніми зв'язками. Це дозволяє не лише координувати бізнес-процеси всередині компанії, але й налагоджувати ефективну взаємодію з партнерами, клієнтами та постачальниками.

На рисуцнку 4.2 представлено схему, яка ілюструє функції та можливості ERP II, зокрема інтеграцію внутрішніх процесів компанії з управлінням зовнішніми взаємовідносинами. Завдяки цьому ERP II стали потужним інструментом для комплексного управління підприємством у сучасних умовах бізнесу.

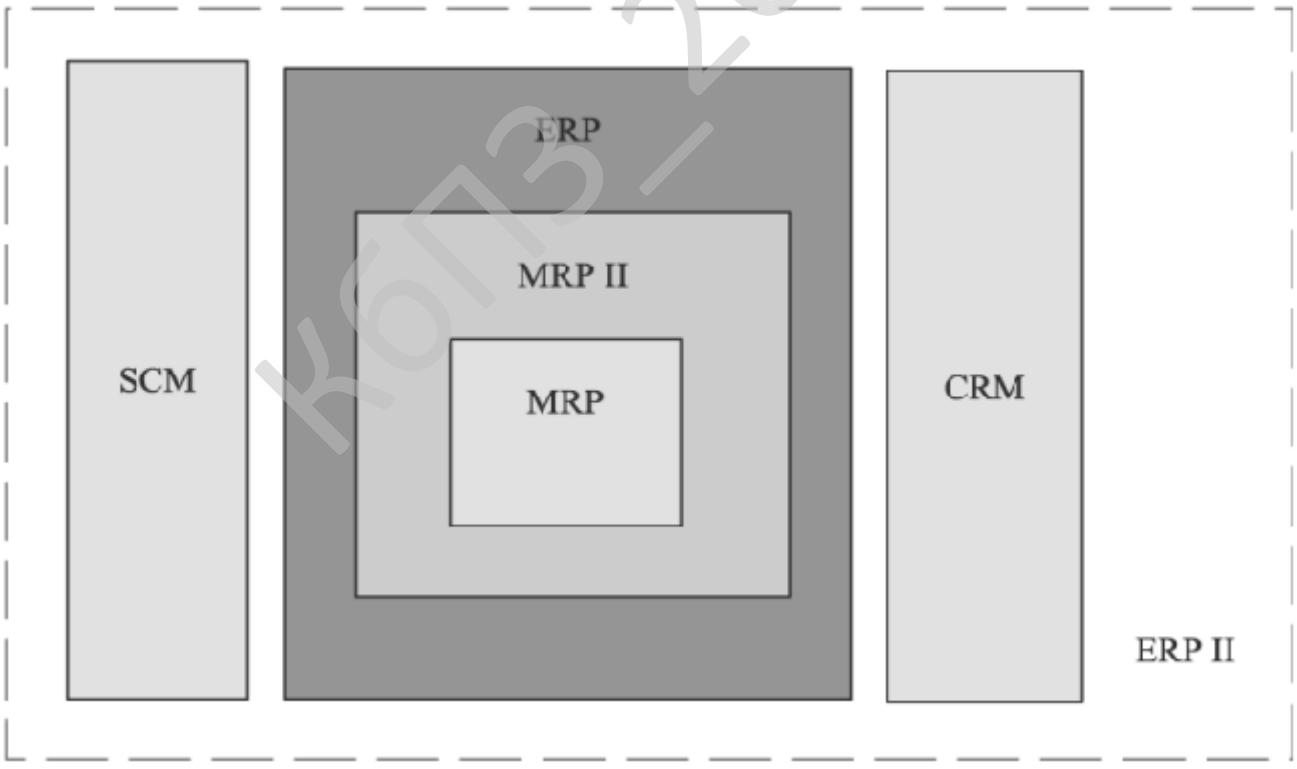

Рисунок 4.2 – Система класу ERP II

Основні функції ERP II та особливості архітектури

ERP II, як наступний етап еволюції ERP-систем, має кілька ключових функцій:

- розширена функціональність: дозволяє виконувати більш складні завдання, включаючи галузеві специфічні рішення;

- сприяння вузькоспеціалізованим галузевим рішенням: завдяки новим інструментам створення програм;

- удосконалення міжкорпоративних бізнес-процесів: додаються нові моделі управління та інтеграції;

- основний функціонал дистанційної системи керування приладобудівним підприємством.

ERP-система для управління підприємством у галузі приладобудування включаэ такі основні функції.

Формування планів:

- план продажів;

- план виробництва.

Моніторинг потреб:

- оцінка ресурсного забезпечення підприємства;

управління закупівлями та залишками товару;

складання виробничих планів та проєктів.

Фінансове регулювання:

- оперативний контроль фінансових потоків.

Проєктний менеджмент:

- моніторинг, контроль та управління проєктами.

Архітектура ERP-системи

ERP-система створюється на основі трирівневої архітектури клієнт-сервер, яка включає:

Рівень представлення:

- інтерфейс для введення та виведення даних оператора системи.

Прикладний рівень:

- обробка даних та виконання програмних операцій.

Рівень даних:

- постійне оновлення даних;
- єдина база для всіх додатків;
- постійне резервне копіювання;
- розміщення даних на кількох серверах для підвищення безпеки.

Робота ERP-системи

ERP має мережеву інфраструктуру. Дані, наприклад бухгалтерські документи, кадрові записи чи інформація про договори, зберігаються у єдиній базі даних. Це дозволяє системі швидко аналізувати інформацію та генерувати корисні аналітичні звіти.

На рисунку 4.3 наведено приклад архітектури клієнт-сервер та послідовності виконання запитів.

Ця модель демонструє:

- як запити від клієнта передаються на сервер;
- як обробляються дані та передаються назад як відповіді клієнту;

Особливості для українського ринку

На українському ринку актуальним є питання оплати ERP-систем. Зазвичай вартість включає або покупку ліцензії, або її річне оновлення, що може створювати бар'єри для малих і середніх підприємств.

Рисунок 4.3 – Структура ERP

Тут наглядно можна побачити адекватну модель поведінки системи, що дозволяю створити якісну та правильну модель бюджетування, оцінити стан на підприємстві та виробити план подальшої роботи. Усі розроблені ERP-системи об’єднує однакова модель проектування:

– платформа. Надає стандартні можливості для роботи модулів та компонентів. Розробник системи має доступ до програмного коду та може вносити зміни. До складу платформи входять: ядро (середовище до якого в перспективі можна буде додавати нові компоненти) та базовий функціонал (не може бути відключений, входить до систему за замовчуванням. В ньому знаходяться усі інструкції для користувачів системи, довідка іт.д.);

– керування даними. Обробка та інтерпретація даних під систему та для передачі їх у інші програмні модулі або додатки, база даних підприємства, положення даних на сервері, програмне забезпечення для роботи с базами даних;

– модулі. Це будь–які компоненти які можна впровадити у систему та розширити її функціонал. Через те, що вони працюють незалежно один від одного, та з єдиною базою, не має проблеми оновлення всієї системі при підключенні нового модуля. Це одна з головних плюсів ERP системи.

Їх можна поділити на такі типи:

- модулі для роботи із зовнішніми замовниками, зовнішніми користувачами, реальними та потенційними партнерами, постачальниками, клієнтами тощо. Це, наприклад, інтернет-магазин, особисті кабінети клієнтів, постачальників і менеджерів по роботі з клієнтами, облікові записи посередників.

Також це може бути модуль, який перенаправляє на CMS систему для створення сайту або на дизайнер. Конвектори беруть API із ядра та підключають його до зовнішньої програми. Завдяки їм можна додати телефонію, налаштувати обмін даними з додатком, сайтом тощо. Ми бачимо цю структуру на рисунку 4.4.

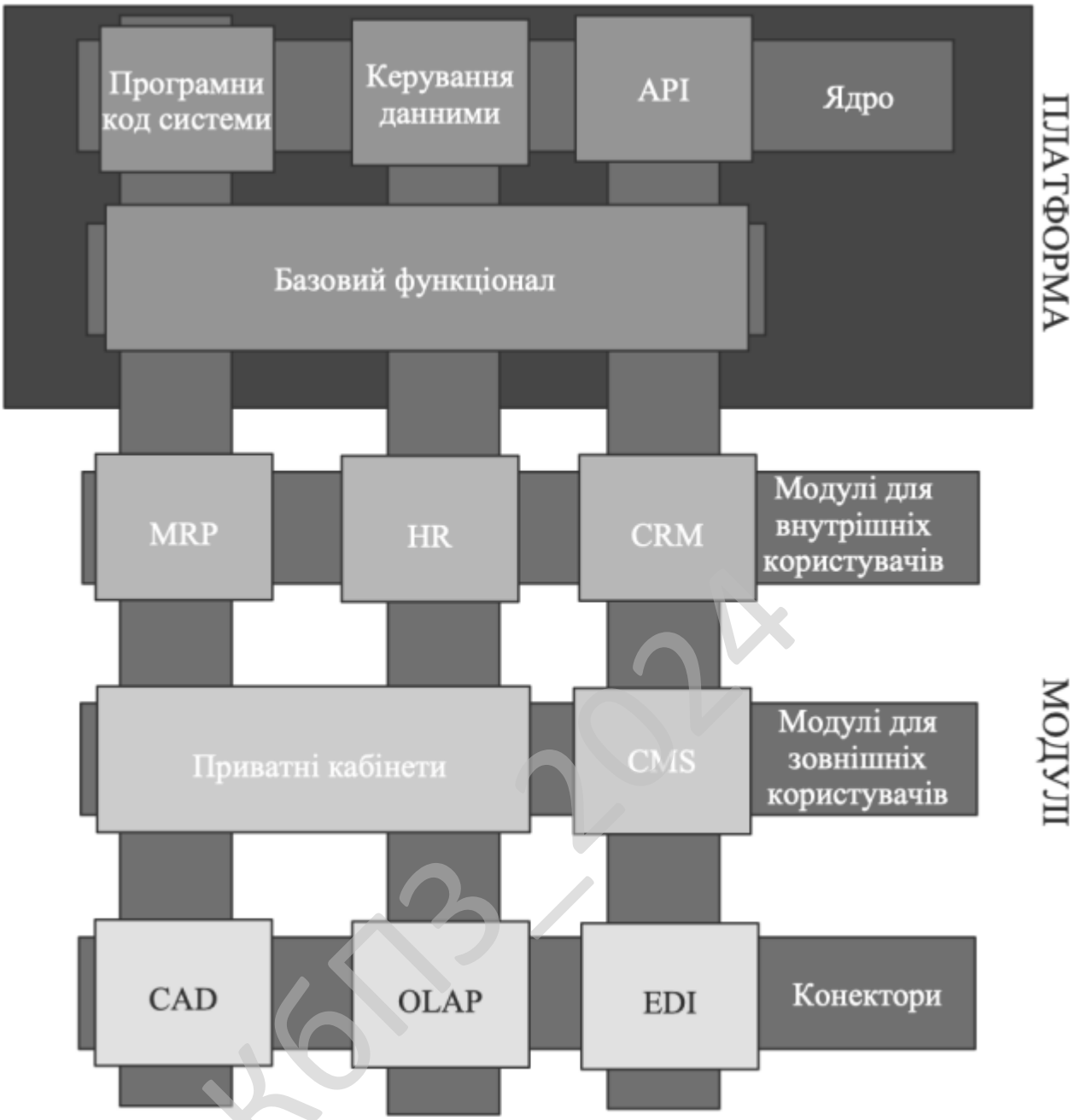

Рисунок 4.4 – Архітектура платформи та модулів ERP системи

Як можна побачити, конектори використовують EDI, OLAP та CAD хоч вони і не входять до ERP. Конектори можуть працювати з ними, тому що вони займаються тільки передачею даних та обміном інформацією. Архітектура розроблюваної системи також модульна і передбачає в перспективі підключення нових інструментів. Завдяки такій структурі, користувач отримує систему, яка буде працювати багато років адже в неї є дуже широкий перелік можливостей

для розвитку. Власнику підприємства не потрібно змінювати програмний продукт, достатньо підібрати та підключити готові модульні рішення.

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Розгляувши компанію виробничого типу були зясовані , які переваги дає впровадження ERP або будь-якої іншої системи такого типу. Для бізнесу такого типу дуже важлива постійна підтримка клієнтів і швидкість виконання замовлень. Адже вони виготовляють як стандартні, так і додаткові замовлення: складські тендери, спецпроекти, спеціальні ексклюзивні замовлення клієнтів, оптові замовлення тощо, для яких потрібно окремо замовляти сировину, металоконструкції, тобто працювати з іншими постачальниками. Тому лише щоденне безпосереднє спілкування з клієнтами дозволяє оптимізувати швидкість роботи (адже клієнт може в будь-який момент внести зміни в замовлення, які потрібно внести в систему негайно та вчасно), та покращити сервіс.

Філософія компанії полягає у створенні гнучких баз для роботи із зовнішніми клієнтами на українському ринку, з партнерами, щоб успішно виживати на ринку та витримувати конкуренцію.

Також компанія має оптові склади та магазини для внутрішніх продажів, комісійні цехи, склади, які також повинні бути введені в єдину інтегровану базу даних. Кількість матеріалів, що використовуються компанією для забезпечення технологічного процесу, варіюється в залежності від замовленої продукції та вимагає швидкої взаємодії з постачальниками. Також логістика або доставка здійснюється в найкоротші терміни після отримання замовлення. Замовлення передається у відділок, де його збирають, складають у вантажівку та вивозять. Все під одним дахом: зберігання продукції, весь асортимент із понад 10 000 товарів тощо. Тому база даних, а особливо безпека даних, є дуже важливим атрибутом для такого виду бізнесу. Також може бути постійне постачання матеріалів з інших країн, що також необхідно враховувати в системі

автоматизації. Щоб визначити вимоги до автоматизованої системи, розглянемо структуру типової виробничої компанії на рисунку 4.5.

Рисунок 4.5 – Типова структура виробничого підприємства

Нижче наведено блок-схему роботи запропонованої ERP-системи.

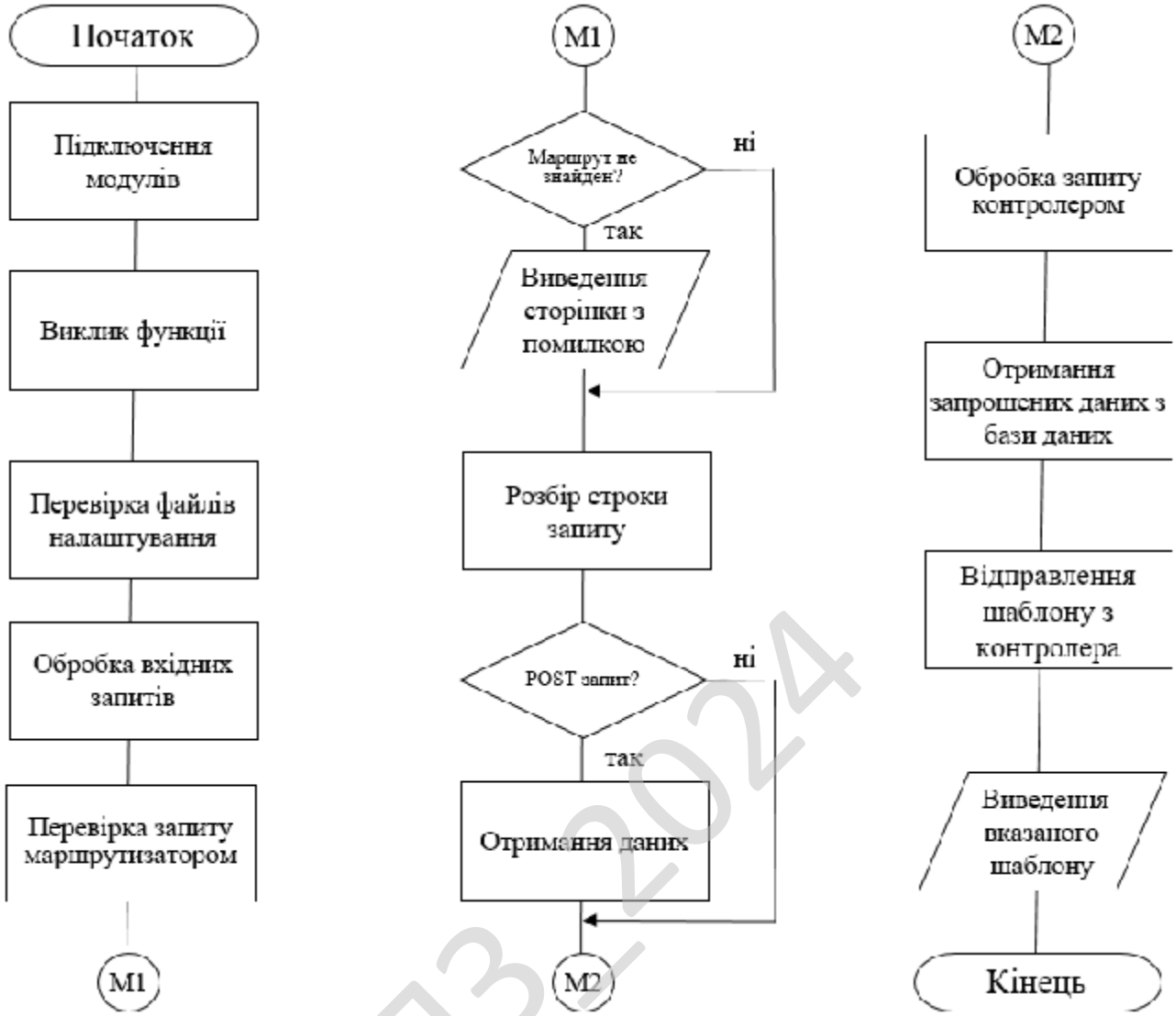

Рисунок 4.6 - Блок-схеми роботи системи

Опис інформаційної системи

У магістерській роботі буде реалізована ERP – система. Є зрозумілим, що питання про актуальність програмного забезпечення ERP є зайвим. Звичайно, багато компаній покладаються на звичайні програмні програми, такі як Excel або Word, для написання правильних та візуально привабливих замовлень. Тим не менше, ці програми закривають розрив у вхідних платіжках і працюють лише за допомогою великої кількості програмних та людських зусиль. Сучасні рішення корпоративного управління ERP можуть робити набагато більше. Вони створюють прозорість із запиту клієнта, управління замовником і постачальником, покупки товарів, планування виробництва та персоналу,

виробництва: з доставки, до документа про оплату. Натисканням однієї кнопки, прибутковість та будь-які недоліки у виробництві або розрахунку можна покласти на папір за допомогою модулю які переводить усі данні у файл .txt. Компанії швидко і легко відстежують вкладений капітал та отримують уявлення про рентабельність інвестицій. Повністю інтегрована система ERP також може спростити існуючі процеси та заощадити ресурси. Система, яка розроблюється, повинна працювати також під інтернет–магазин, замовлення якого надходять електронною поштою і не пов'язані безпосередньо з системою управління товарами. Завдяки пропонованому програмному рішенню, клієнти можуть покладатися на широкий спектр варіантів та галузевих рішень, адже завжди можна додати ще щось нове до системи або усі елементи можуть з легкістю бути адаптовані для індивідуальних потреб завдяки інтегруючим модулям.

Реалізація даної системи на підприємстві буде полягати у створенні програмного забезпечення, за допомогою якого користувач зможе з легкістю проводити операції. Класична система такого типу, є статичною та націлена на збір інформації з бізнес процесів без даних та аналізу цих даних. Сучасні ERP або WMS системи, навпаки, засновані на вимогах користувачів. Доведено, що системи такого типу, підвищують продуктивність і знижують витрати на персонал. На відміну від класичного управління товарами, вони відображають всі сфери і бізнес-процеси компанії. У центрі уваги не лише управління матеріальними ресурсами, але також фінанси і бухгалтерський облік, управління персоналом, продажу, маркетинг, дослідження та інші сфери діяльності компанії. Функціональні розмежування, які раніше були загальними для логістики, фінансового обліку та контролінгу, наприклад, усуваються за допомогою комплексної системи. Всі області спілкуються між собою і використовують одну і ту ж базу даних. Управління матеріальними ресурсами або планування ресурсів підприємства це планування, управління і контроль всіх рухів матеріалів в компанії, а також між компанією та іншими суб'єктами бізнесу, такими як клієнти і постачальники. Поняття матеріалу є широким, і описує як закупка

товарів для перепродажу, сировину і напівфабрикати, необхідні для виробництва. Їх необхідно закуповувати, зберігати і відправляти таким чином, щоб вони були в наявності в достатній кількості, з необхідною якістю, в потрібний час і в потрібному місці.

Реалізація ERP-системи

Мова java та середовище розробки IntellijIDEA

Серед об'єктно-орієнтованих мов програмування високого рівня для аналізу були відібрані наступні мови: C++ та Java. Мова C++ підтримує покажчики, перевантаження операторів, адресну арифметику покажчиків та структури. В мові присутня можливість реалізації деструкторів для знищення об'єктів класу. Це допомагає більш ретельно контролювати пам'ять, що використовується програмою, але ускладнює її розробку. C++ не має вбудованої підтримки потоків, для цього виникає необхідність використовувати сторонні розробки. C++ підтримує оператор «goto», але слід зауважити, що його вживання не рекомендується, тому що код стає важко сприймати, а також це може призвести до непередбачуваних помилок та наслідків. В C++ є необов'язковим використання операторів «try catch» навіть якщо функція генерує винятки. C++ не підтримує можливість написання документації у вигляді коментарів. C++ не генерує об'єктний код, це означає, що то ж самий код може не запускатися на різних платформах.

Мова програмування Java не підтримує покажчики, перевантаження операторів, адресну арифметику покажчиків та структури. В Java немає можливості реалізації деструкторів для знищення об'єктів класу. Для цього існує так званий «збирач сміття», який сам визначає, який об'єкт більше не використовується та підлягає знищенню. Через це програми на Java, як правило, мають меншу швидкість виконання ніж програми на C++, але процес написання коду на Java стає більш простим. В Java є вбудовані інструменти для роботи з потоками. Для цього існує клас потоків, який унаслідується, для створення нового потоку і реалізації методу «run», що його запускає. В Java немає

оператора «goto», але існує його більш прийнятний аналог «break», який використовується в основному для виходу з циклів, але також передбачене його використання, схоже з «goto», коли після «break» можна вказати мітку блоку коду, після цього керування передається зазначеному блоку коду та здійснюється вихід з нього. Обробка винятків в Java є обов'язковою процедурою при використанні функцій, що їх генерують, через неможливість прописати деструктори. Основна властивість Java, яка дозволяє програмі виконуватись на різних платформах, складається в тому, що компілятор Java видає не виконуваний код, а так званий «байт-код» – в вищому ступені оптимізований набір інструкцій, призначений для виконання в спеціальній системі Java, що іменується віртуальною машиною Java. Перша версія віртуальної машини розроблялась у якості інтерпретатора байт-коду. Це може дивувати, бо для забезпечення максимальної продуктивності компілятори багатьох сучасних мов програмування призвані створювати виконуваний код. Але те, що програма на Java інтерпретується в віртуальній машині Java, допомагає вирішити основні проблеми розробки програм. Трансляція програми Java в байт-код значно спрощує її виконання в різнотипних середовищах, оскільки на кожній платформі необхідно реалізувати тільки віртуальну машину Java. Якщо в окремій системі є виконуючий пакет, в ній можна виконувати будь-яку програму на Java. Слід, однак, мати на увазі, що 53 всі віртуальні машини Java на різних платформах, не дивлячись на деякі відмінності і особливості їх реалізації, здатні правильно інтерпретувати один і той же байт-код. Якби програма на Java компілювалась в код, залежний від машини, то для кожного типа процесорів, підключених до Інтернету, повинні були б існувати окремі версії однієї і тієї ж програми. Таке рішення є неприйнятним.

Таким чином, організація виконання байт-коду віртуальної машини Java – найпростіший спосіб створення по-справжньому переносних програм. Розглянемо аргументи, на основі яких буде зроблений висновок. Для початку перевіримо, чи підходять ці мови для перелічених цілей. На обидвох мовах є

можливість розробити модель. Для обидвох мов передбачені драйвери для взаємодії з сервером SQL. На обидвох мовах можна розробити інтерфейс користувача, за допомогою якого потрібно буде здійснювати контроль автоматизації моделі. Перейдемо до інших аргументів. Як було з'ясовано, в зручності написання програми переважає Java, але краще оптимізованою буде програма на C++. Мною перевага була віддана зручності, бо таким чином імовірність зробити помилку зменшується, а продуктивність при розробці зросте. Треба зауважити, що при розробці програми планується використовування багатопотоковість. В мові Java цей інструмент вже вбудований, в C++ потрібно використовувати сторонні ресурси. Тому більш зручною реалізація потоків передбачається в Java. При розробці програми на C++ потрібно мати на увазі, що при перенесенні програми на іншу платформу, вона може навіть не запуститись. В Java з її віртуальною машиною такої проблеми нема, це дозволяє слідувати принципу «працює на одній платформі – працює на всіх». Як можна побачити, більшість переваг на стороні Java. Таким чином був зроблений вибір мови програмування. Наступним кроком є вибір середовища розробки. Серед можливих варіантів: Intellij IDEA, Eclipse та JDeveloper. Розглянемо особливості кожного з варіантів.

Особливості IntellijIDEA:

– можливість безкоштовно отримати версію Ultimate Edition через програму підтримки студентів; – зручний інтерфейс;

– велика кількість типів проектів, які можна вибрати при його створенні (звичайний java проект, Maven, Andorid);

– велика кількість гарячих клавіш, можливість використання скорочень (наприклад, замість написання «System.out.println()» можна написати «soup» та натиснути клавішу «Tab» на клавітурі);

– при допусканні синтаксичної помилки, вона виділяється, та пропонується варіанти її виправлення;

– підтримка мов програмування окрім Java; – легке налаштування зв’язку з БД.

Особливості Eclipse:

– підтримка мов програмування окрім Java;

– безкоштовність;

– велика бібліотека плагінів, розроблених користувачами;

– середовище написане на Java, це означає, що воно не залежить від платформи;

– легке налаштування зв’язку з БД.

Особливості JDeveloper:

– безкоштовність;

– написане на Java, це означає, що воно не залежить від платформи;

– складність налаштування зв’язку з БД.

Після аналізу середовищ розробки першим зробленим висновком є виключення зі списку середовища JDeveloper, бо дуже великим недоліком є складність взаємодії з БД. Наступне рішення – признання переваг IntellijIDEA. Через зручність розробки програм в цьому середовищі гарантована велика продуктивність та менша імовірність зробити помилки. Особливо великим плюсом є можливість створення Maven-проекту, який допоможе встановити необхідний драйвер для налагодження зв’язку з БД, а також побудувати проект з виконуваним файлом.

Наступним кроком є вибір засобу, за допомогою якого буде будуватись інтерфейс програми. Для реалізації інтерфейсу в Java існує бібліотека Swing. В ній присутні всі елементи, за допомогою яких можна розробити зручну програму: вікна, кнопки, меню, таблиці і т.д.

Потрібно конкретизувати процес роботи програми. В БД зберігається інформація про обладнання системи та тривалі процеси. Необхідна інформація з БД подається на інтерфейс, за допомогою якого можна коректувати БД. На основі інформації з БД або взаємодії користувача з інтерфейсом програма

керування розробляє планування роботи. Стеження за процесом виконання розроблених замовлень до окремих частин підприємств здійснюється за допомогою інтерфейсу.

Створення проекту програми. При створенні проекту в середовищі розробки «IntellijIDEA» потрібно обрати тип проекту. Тип проекту був обраний Maven, бо це допоможе встановити необхідні залежності в файлі «pom.xml», серед яких – драйвер для налагодження зв'язку з БД. Також це допоможе побудувати проект з виконуваним файлом.

Процес роботи програми

Схематичне зображення роботи програми знаходиться на рисунку 4.9.

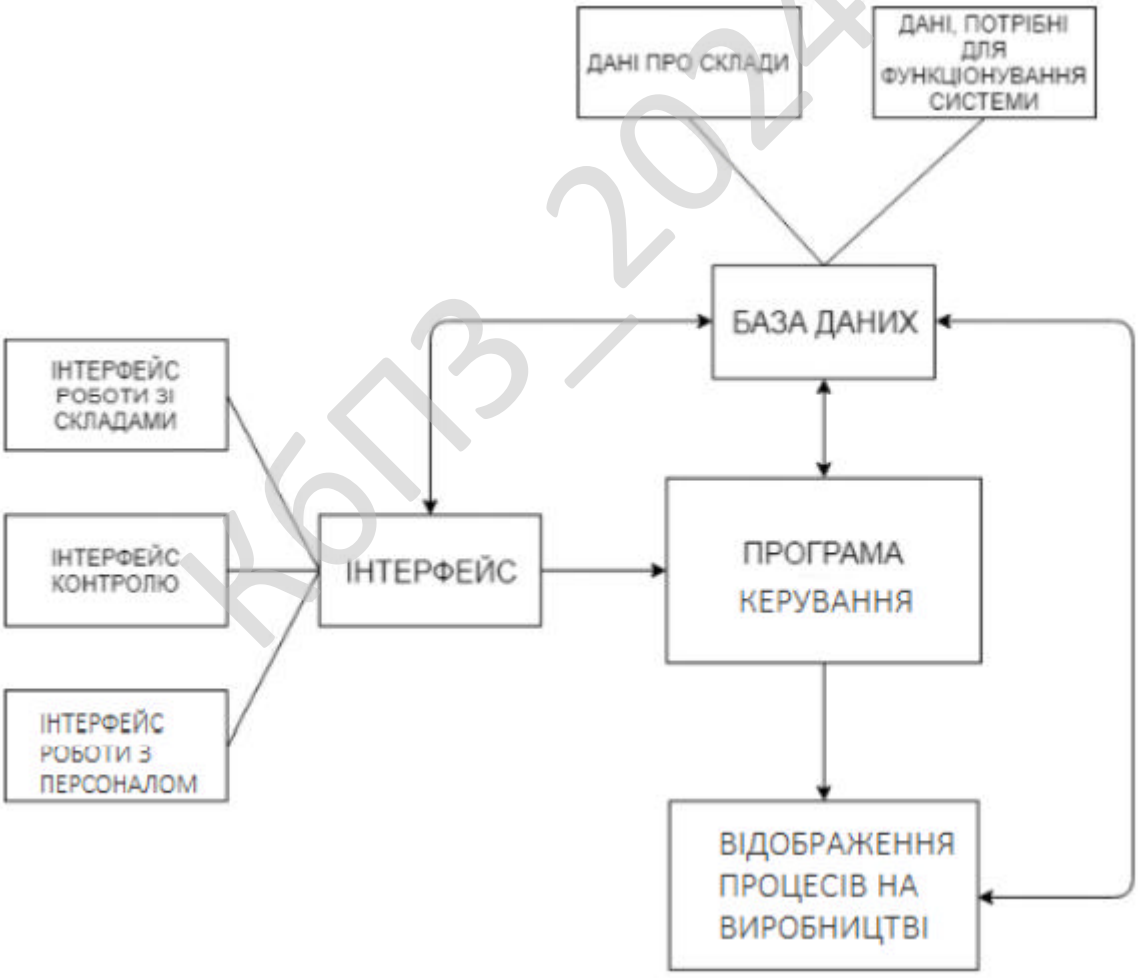

Рисунок 4.7 – Схематичне зображення роботи програми

Розробка програмного забезпечення

При створенні проекту в середовищі розробки «IntellijIDEA» потрібно обрати тип проекту. Тип проекту був обраний Maven, бо це допоможе встановити необхідні залежності в файлі «pom.xml», серед яких – драйвер для налагодження зв'язку з БД. Також це допоможе побудувати проект з виконуваним файлом.

Наступний крок – розробити структуру проекту, тобто спланувати, які класи будуть присутні в програмі, та як вони будуть взаємодіяти.

Перелік класів:

– Main (необхідний клас для запуску програми);

– LoginFrame (вікно входу, використовуючи логін та пароль); – DBConnection (відповідає за зв'язок з БД);

– ProgressThread (потік, що відповідає за відображення прогресу виконання завдань);

– Errors (допомагає працювати з виключеннями та виводити повідомлення про помилки);

– ResultSetTableModel (модель таблиці, що виводить дані з бази БД);

– WarehouseFrame (вікно з інформацією про склади);

– OrderFrame (вікно створення наказу до працівників);

– SellFrame (вікно продажу);

– NewProject (вікно створенню проекту);

– InformationFrame (вікно з інформацією про виконання замов та наказів);

– AccessFrame (вікно з інформацією про права окремих користувачів та груп).

Отримання зв'язку з базою даних

Для взаємодії програми з БД потрібно налагодити з нею зв'язок. Так як вона знаходиться на локальному сервері, зробити це буде легко. При встановленні СУБД MySQL була можливість разом із нею встановити драйвер JDBC (Java DataBase Connectivity), який забезпечує взаємодію з БД. Після цього

треба налаштувати цей драйвер в середовищі IntellijIDEA. Для цього треба відкрити репозиторий Maven, знайти там JDBC та скопіювати те, що нам потрібно, в файл pom.xml, після цього почнеться автоматичне завантаження драйвера. Наступним кроком є розробка класу «DBConnection», з об'єктом класу «Connection» та реалізацією метода «getConnection», який повертає цей об'єкт. Для цього потрібно передати менеджеру драйверів URL нашого локального серверу, потім виконати з'єднання і перевірку.

В результаті було отримано таке повідомлення в консолі: «Успішно». Зв'язок з БД налагоджено. Для створення запитів до БД існують наступні класи:

– Statement (дослівний запит у вигляді String);

– PreparedStatement (підготовлений запит, у якому деякі дані можуть позначатись знаком запитання, потім ці дані уточняються);

– CallableStatement (викликає збережені процедури).

Для отримання результатів існує клас ResultSet. Можна сказати, що цей клас представляє собою таблицю, з якої можна отримати дані, що зберігаються в певній колонці. В класі передбачені методи для переходу між строками цієї таблиці.

Реалізація вікна входу в програму

Для безпечного початку роботи з системою потрібно розробити вікно вводу логіна та пароля. Воно допомагає запобігти несанкціонованому доступу до системи. В ньому мають бути присутніми два поля: одне – для логіна, друге – для пароля, при цьому символи, що вводяться у поле пароля, мають виглядати, як крапки. Також повинна бути присутня кнопка входу. За допомогою бібліотеки Swing реалізувати його інтерфейс не складно. Для цього будуть потрібні об'єкти наступних класів: JTextfield, JPasswordField, JButton, JLabel.

Для реалізації кнопки входу треба додати до неї так званий «ActionListener», який буде спрацьовувати після натискання на кнопку. Була розроблена блок-схема алгоритму, який буде виконуватись при натисканні кнопки.

4.2 Захист розробленого програмного забезпечення

В першу чергу варто перевірити розроблений додаток автоматизованими засобами, вони зможуть перевірити додаток на найрозповсюдженіші помилки. Вони перевіряють не тільки код написаний розробником, а й бібліотеки та модулі які використовує розробник, а вони складають найбільшу частку зломів. Згідно з останніми перевірками, понад 50% npm модулів, не оновлюються більше декількох років.

Є декілька способів перевірити модулі перед їх використанням, в першу чергу варто переглянути код, та перевірити чи повністю модуль покриває потреби розробника. Він повинен перевірити які ще бібліотеки використовує цей модуль і коли вони останній раз оновлювалися, чим менше він має залежностей, тим краще.

Великі компанії мають спеціальних працівників, які проводять ревізії пакетів перед їх використанням, та складають спеціальні білі списки дозволених модулів.

Сама npm занепокоєна тим, як часто у бібліотеки почали впроваджувати майнерів та віруси для збору приватних даних. Тому вони випустили спеціальний модуль npm audit. Він сканує встановленні модулі в проект і порівнює їх з чорним списком модулів, які містять уразливості. Сьогодні навіть команда npm install підкаже, чи мають встановленні модулі вразливості. А нова команда npm audit fix, пропонує замінити вразливі пакети, або оновити до більш захищених версій, якщо такі існують. Але так можна знайти лише ті модулі, про які вже повідомили користувачі та розробники.

Також для перевірки безпеки додатку, важливим є написання тестів. Зазвичай розробники на AngularJS використовують концепцію Jasmine's Behavior-Driven Development (BDD), при написанні тестів. Jasmine − це вільний фреймворк для тестування коду написаного за допомогою JavaScript, його можна запустити на будь-якій платформі. Його перевагою є зручний інтерфейс, та

зрозуміле налаштування. Для тестування одиничних тестів часто використовують бібліотеку Karma, головна мета якого наблизити тестування додатку до його налаштувань в продакшен.

Також важливо використовувати протокол HTTPS, він є набагато надійнішим HTTP. HyperText Transfer Protocol Secure (HTTPS) — забезпечує шифрування даних і дозволяє надійно захищати дані користувачів при передачі в Інтернеті. Сьогодні більшість сайтів використовують саме цей протокол, оскільки він є обов'язковим при передачі приватних даних на сервер, особливо коли передаються паролі або дані кредитних карт. Також варто бути обережним с cookie файлами, які передаються під час авторизації. Зловмисник може перехопити їх та підробити запит на сервер, щоб цього уникнути потрібно використовувати HTTPS протокол.

Також слід брати до уваги поширені CORS та CSRF атаки. CSRF- атака це коли на сторінки, робиться непомітна форма, щоб відправляє запит з приватними даним користувача. Якщо на сайті авторизація відбувається тільки за допомогою кукі, то така атака може бути успішна. Щоб цього уникнути потрібно використовувати спеціальні токени. Для підпису XMLHttpRequest токен також зберігається в кукі. Тоді JavaScript може прочитати його з домену і додати в заголовок, а сервер - перевірити, що в заголовку міститься коректний токен.

При крос-домених запитів, браузер додає заголовок Origin, який зберігає домен, з якого відбуваються запити. Сервер у цьому випадку повинен перевірити домен і відповісти спеціальним заголовком. Якщо сервер дозволяє доступ, він повинен відправити заголовок Access-Control-Allow-Origin, що зберігає домен запиту. Якщо Access-Control-Allow-Origin відсутній, то сервер завершує запит з помилкою. При таких запитах не передаються куки і заголовки HTTP-авторизації.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

У Магістерській роботі реалізована ERP система підприємства.

Проект створюється в організації, або в розділі «Особисті проекти». Створювати проект в організації можуть власник і адміністратори організації. Створення проекту можливо декількома способами. Скористатися пунктом «Створити» -> «Проект» головного меню програми. Скористатися контекстним меню на вкладці «Проекти» (пункт «Створити» -> «Проект»). На вкладці «Проекти» вибрати потрібну організацію або розділ «Особисті проекти» і в контекстному меню вибрати пункт «Створити». Як ми бачимо, простота процесу створення проекту одразу помічається і не підлягає сумніву. Саме через це був обраний цей метод розробки мобільного додатку, і це не говорячи про низький затрачений час та ресурси.

Вікно створення проекту складається з двох вкладок. На першій вкладці користувач вказує назву проекту, може додати опис і коментар до проекту, вибирає тип проекту. Існує три типи проектів:

Звичайний проект - не відрізняється особливими ознаками, завдання використовуються в різних цілях користувача;

База знань - проект, завдання якого служать для збору інформації про будь-якої предметної області;

Сховище рахунків – під час створення завдання в даному проекті в нього автоматично додається рахунок.

На вкладці «Доступ» користувач може призначити права доступу до нового проекту. Вікно властивостей проекту надає користувачеві інформацію про завдання проекту, показує історію коментування завдань, надає доступ до категорій. В даному вікні користувач також може змінити назву, опис або тип проекту. Відкрити вікно властивостей проекту можна за допомогою

відповідного пункту контекстного меню проекту на вкладці «Проекти» головного вікна програми (Enter). Інформація про проект розбита на кілька вкладок:

Вкладка «Інформація» надає такі відомості про проект, як назва, опис проекту, коментар, тип проекту (база знань, сховище рахунків) і список завдань проекту.

Опція «Показувати виконані»

В інтелект-карті або перелік містить завдання зі статусом «Виконано». За замовчуванням ця опція вимкнена.

Опція «Завантажувати коментарі» - при включеній опції коментарі завантажуються по всіх завдань проекту, незалежно від того відкривав їх користувач чи ні. Інакше, коментарі завантажені тільки за завданнями, в яких брав участь цей користувач, по іншим завданням коментарі завантажуються з сервера тільки в момент відкриття вікна завдання.

Опція «Показувати виконані» - в інтелект-карті або перелік містить завдання зі статусом «Виконано». За замовчуванням ця опція вимкнена. Опція «Завантажувати коментарі» - при включеній опції коментарі завантажуються по всіх завдань проекту, незалежно від того відкривав їх користувач чи ні. Інакше, коментарі завантажені тільки за завданнями, в яких брав участь цей користувач, по іншим завданням коментарі завантажуються з сервера тільки в момент відкриття вікна завдання. Опція «Не показувати в дереві» - при включеній опції даний проект не відображається в дереві проектів користувача. Прихований таким чином проект відображається в швидкому пошуку дерева проектів (якщо задовольняє запиту пошуку). Опція «Сповіщати при створенні нових завдань» - якщо опція включена, то користувач отримує повідомлення, коли в проекті з'являється нова задача. На вкладці «Дерево» у вигляді інтелект-карти (mind map) представлена графічна інформація про завдання даного проекту. Інтелект-карта дає наочнеуявлення про структуру проекту, графічна інтерпретація властивостей завдань (статус, важливість і інше) збігається з деревом проектів.

За допомогою контекстного меню і перетягування елементів інтелект-карти можна легко міняти структуру проекту, роблячи завдання вкладеними, створювати нові завдання, видаляти завдання. При наведенні курсора миші на завдання з'являється вікно підказки з першим коментарем в задачі. На вкладці «Список» завдання виводяться не у вигляді інтелект-карти, а звичайним списком. Для великих проектів (більш 50 завдань в корені проекту) інтелект-карта не будується, завдання виводяться тільки у вигляді списку. Видимість проекту можна обмежувати. Керувати правами доступу до проекту можуть власник і адміністратор проекту, власник і адміністратор організації, в якій знаходиться проект. На вкладці «Доступ» представлено список користувачів і перелік прав, які можна їм призначити. «Читання» - проект відображається в дереві проектів користувача. Користувач бачить всі завдання проекту, бачить коментарі цих завдань, але не може додавати нові коментарі;

«Зміна» - користувач може коментувати завдання проекту, змінювати властивості задач, створювати нові завдання в проекті, переміщати завдання;

«Адміністратор» - повний доступ до проекту. Користувач може створювати нові завдання в проекті, видаляти завдання, переміщати проект, управляти правами доступу до проекту. Поле у верхній частині вікна служить для пошуку користувачів. Під час введення ПІБ користувача список відповідним чином фільтрується. Опція «Є доступ» - при включеній опції в дереві відображаються тільки ті групи і користувачі, у яких є доступ до проекту. Вкладка «Коментарі» містить стрічку коментарів по всіх завдань проекту з фільтрацією за категоріями. У лівій частині вікна знаходиться список категорій організації, а також пункт «Всі коментарі» (обраний за замовчуванням при відкритті вікна проекту). При виборі будь-якої з категорій, в правій частині вікна завантажуються останні 100 коментарів завдань проекту, відмічені даною категорією. Завантажити наступні 100 коментарів можна за допомогою посилання «Завантажити ще ...», розташованої в кінці поточного списку. В отриманому списку також можна здійснювати додатковий пошук

коментарів по тексту повідомлення, автору коментаря, назвою вкладених файлів. Для виконання пошуку можна скористатися пунктом «Знайти» в контекстному меню стрічки коментарів.

Висновок

Програмна реалізація системи дистанційного керування на підприємстві полягає у створенні програмного забезпечення, за допомогою якого користувач зможе з легкістю проводити операції. Класична система такого типу, є статичною та націлена на збір інформації з бізнес процесів без даних та аналізу цих даних. Сучасні ERP або WMS системи, навпаки, засновані на вимогах користувачів. Також в експерименті доведено, що використання розробленої програми позитивно впливає на процес керування підприємством а саме – знижує затрачений час на формування замов та наказів стосовно роботи, а також зменшує імовірність допустити помилку при керуванні.

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення, яке призначено продемонструвати можливості застосування мобільних додатків при впровадженні ERP-систем.

Метою розробки є дослідження та програмна реалізація мобільного додатку для ERP системи управління підприємства і визначення його впливу на ефективність роботи підприємства.

Об'єктом дослідження є можливості використання мобільного додатку сумісно з ERP-системою для підвищення ефективності діяльності підприємства.

Предметом дослідження є сукупність теоретичних, технічних та практичних засад створення й впровадження мобільного додатку для керування роботою підприємства.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення. Використовувались методи порівняння та систематизації, синтезу та аналізу та прототипування і моделювання (при розробці мобільного додатку).

Наукова новизна отриманих результатів.

У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Розроблено мобільний додаток для роботи ERP-системи;
2. Здійснено теоретичне узагальнення і запропоновано нові практичні рекомендації для створення мобільного додатку.
3. Обґрунтовано, що ключовою перевагою використання мобільних додатків для підприємців є можливість ретаргетингу який передбачає створення потужного бренду та розширення ринків збуту.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації мобільного додатку для управління ERP-системою можуть бути цікавими для різних груп зацікавлених сторін (рисунок 7.1).

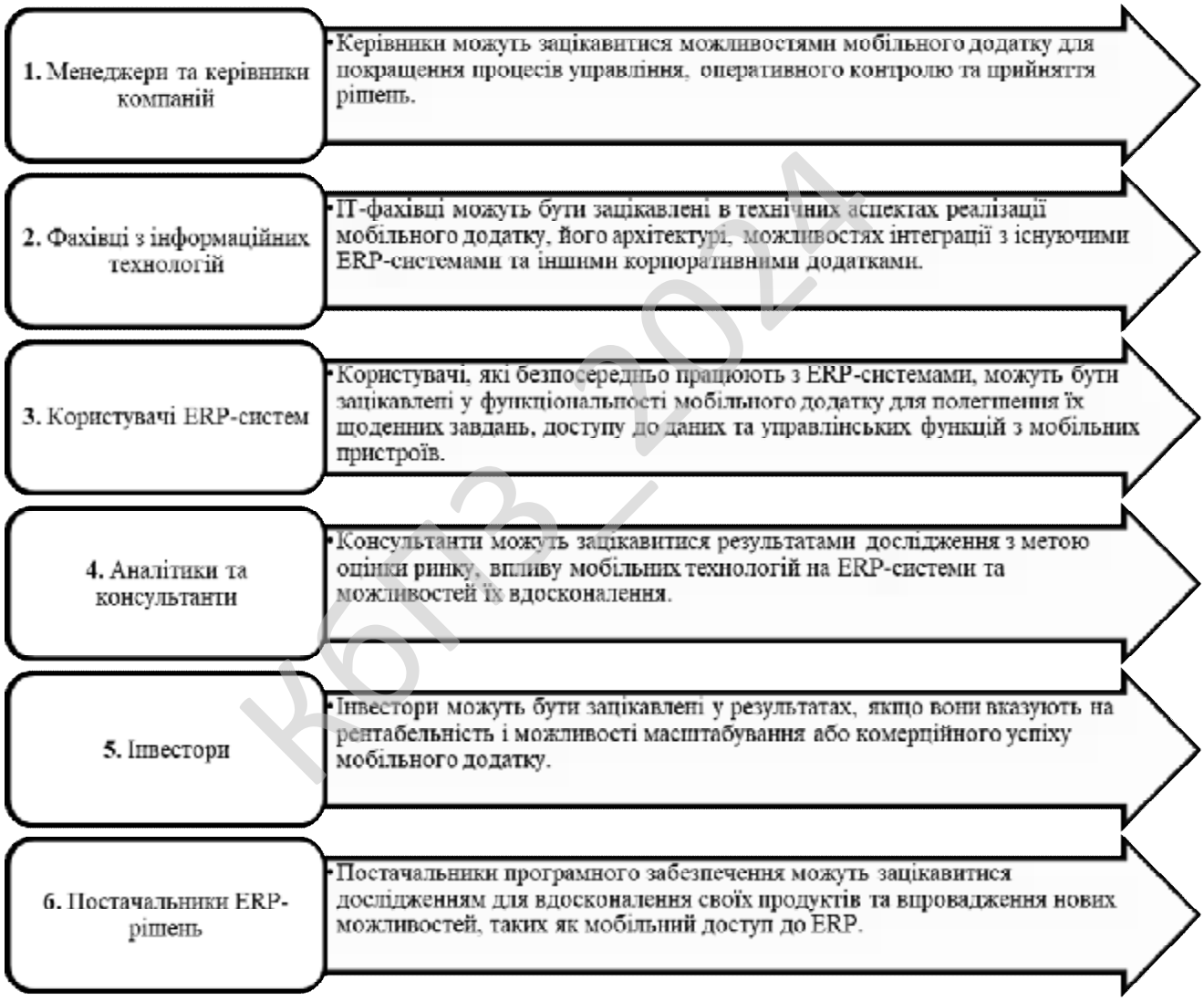

Рисунок 7.1 – Цільова аудиторія

Результати дослідження та програмної реалізації мобільного додатку для управління ERP-системою можуть зацікавити широкий спектр фахівців,

включаючи керівників, IT-фахівців, аналітиків, інвесторів і науковців, оскільки вони можуть мати значний вплив на ефективність управлінських процесів і прийняття рішень в організаціях.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації мобільного додатку для управління ERP-системою шляхом застосування методів експертних оцінок може бути здійснена через кілька етапів. Один з підходів — це використання методу експертних оцінок на основі бальних систем або метода «Делфі».

Першочергово, для оцінки привабливості мобільного додатку можуть бути використані такі критерії (рисунок 7.2):

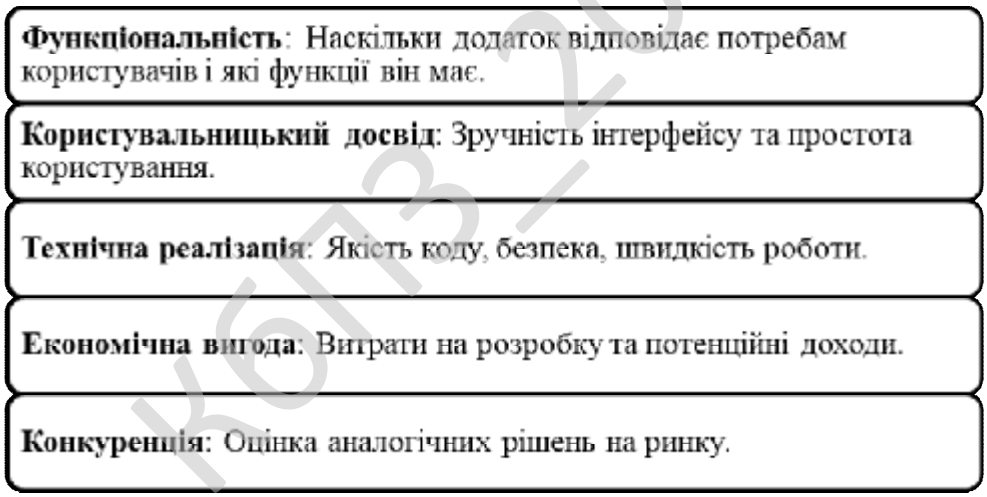

Рисунок 7.2 – Критерії експертної оцінки

Формуємо групу експертів з різних областей, таких як: фахівці з управління бізнесом, IT-експерти, спеціалісти з UX/UI дизайну, маркетологи.

Експерти оцінюють кожен критерій за шкалою, наприклад, від 1 до 10, де 1 — дуже погано, а 10 — відмінно. Щоб уникнути групового тиску та упереджень, оцінки збираються анонімно.

Розраховується середнє значення оцінок для кожного критерію: функціональність – 8, користувальницький досвід – 9, технічна реалізація – 7, економічна вигода – 6, конкуренція – 7.

Важаться критерії відповідно до їхньої важливості та підрахувати загальну оцінку привабливості проєкту: функціональність - 0.25, користувальницький досвід - 0.30, технічна реалізація - 0.20, економічна вигода - 0.15, конкуренція - 0.10.

Загальна оцінка=(8×0.25)+(9×0.30)+(7×0.20)+(6×0.15)+(7×0.10)=2+2.7+1.4 +0.9+0.7=7.7

Отримана загальна оцінка привабливості 7.7 (з максимальною оцінкою 10) свідчить про те, що програмна реалізація мобільного додатку для управління ERP-системою має високу ймовірність успішності на ринку. Ця інформація може бути корисною для інвесторів, керівництва компанії та інших зацікавлених сторін у прийнятті рішень про подальші кроки в проєкті.

7.3 Вибір методу оцінки вартості ПЗ

Оцінка вартості програмної реалізації мобільного додатку для управління ERP-системою може бути здійснена за допомогою кількох методів, і вибір методу залежить від конкретних обставин, цілей проєкту та доступних даних. Вибір методу залежить від:

– стадії проекту: на ранніх етапах може бути доцільніше використовувати метод аналогій або оцінку на основі витрат;

– доступних даних: якщо у вас є доступ до інформації про аналогічні проєкти, метод аналогій буде доречним;

– типу додатку: якщо ви плануєте розробити складний додаток з багатьма функціями, метод оцінки на основі функціональності може бути найбільш підходящим.

Для програмної реалізації мобільного додатку для управління ERP-системою рекомендує поєднання методів розрахунку витрат і дисконтованих

грошових потоків, оскільки це забезпечить комплексний підхід до оцінки вартості, враховуючи як витрати на розробку, так і потенційні доходи від проекту.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність впровадження мобільного додатку для управління ERP-системою може бути проілюстрована через кілька ключових аспектів (рисунок 7.3).

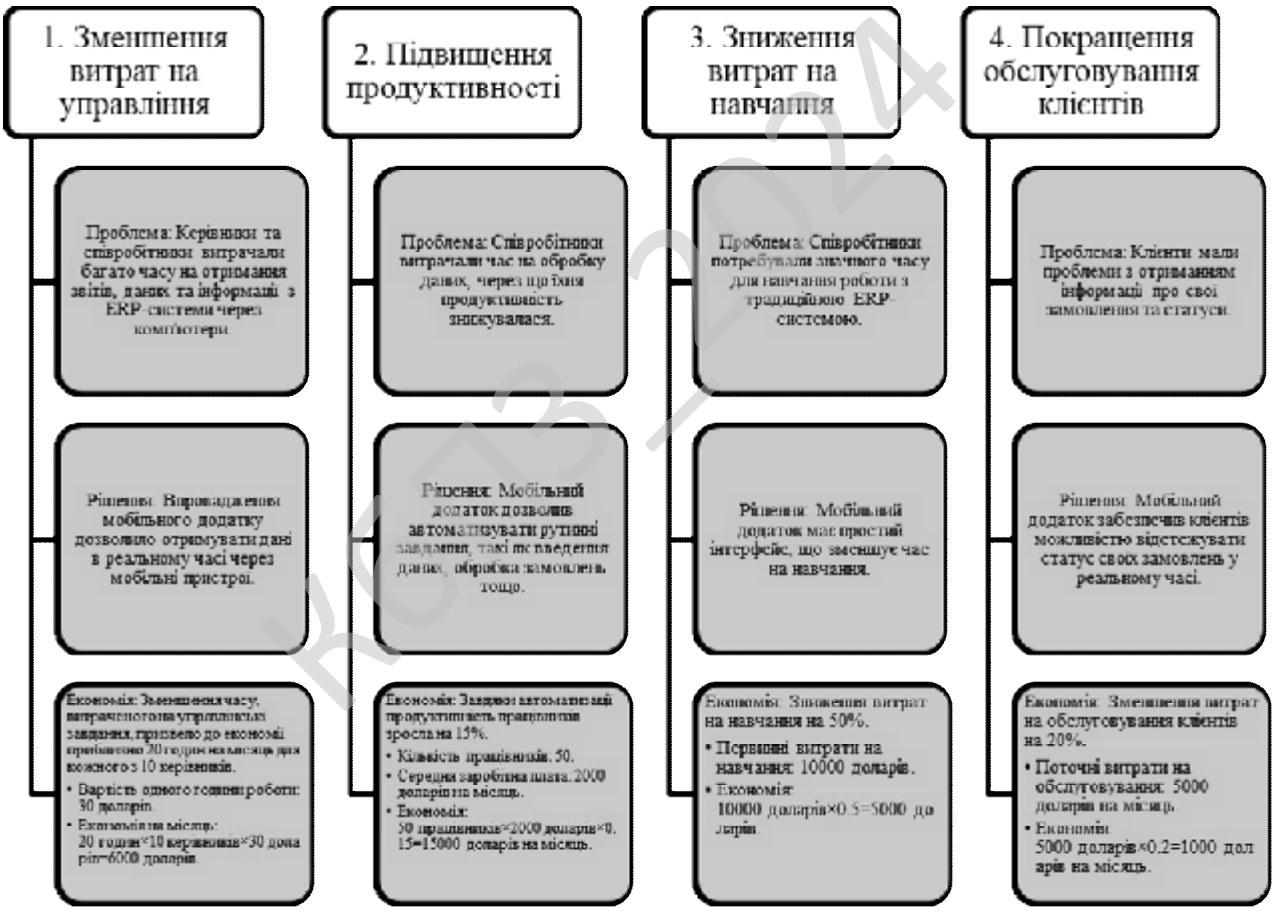

Рисунок 7.3 – Ключові аспекти ефективності впровадження проєкту для клієнта

Загальна економічна ефективність включатиме в себе: зменшення витрат на управління: 6000 доларів на місяць, підвищення продуктивності: 15000

доларів на місяць, зниження витрат на навчання: 5000 доларів (одноразово), покращення обслуговування клієнтів: 1000 доларів на місяць. Загальна щомісячна економія: 6000+15000+1000=22000 доларів. Одноразова економія на навчанні – 5000 доларів.

Впровадження мобільного додатку для управління ERP-системою приносить компанії щомісячну економію у розмірі 22000 доларів та одноразову економію на навчанні у розмірі 5000 доларів. Це підкреслює високу економічну ефективність проекту та вигоди від його реалізації.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації мобільного додатку для управління ERP-системою схематично подано на рисунку 7.4.

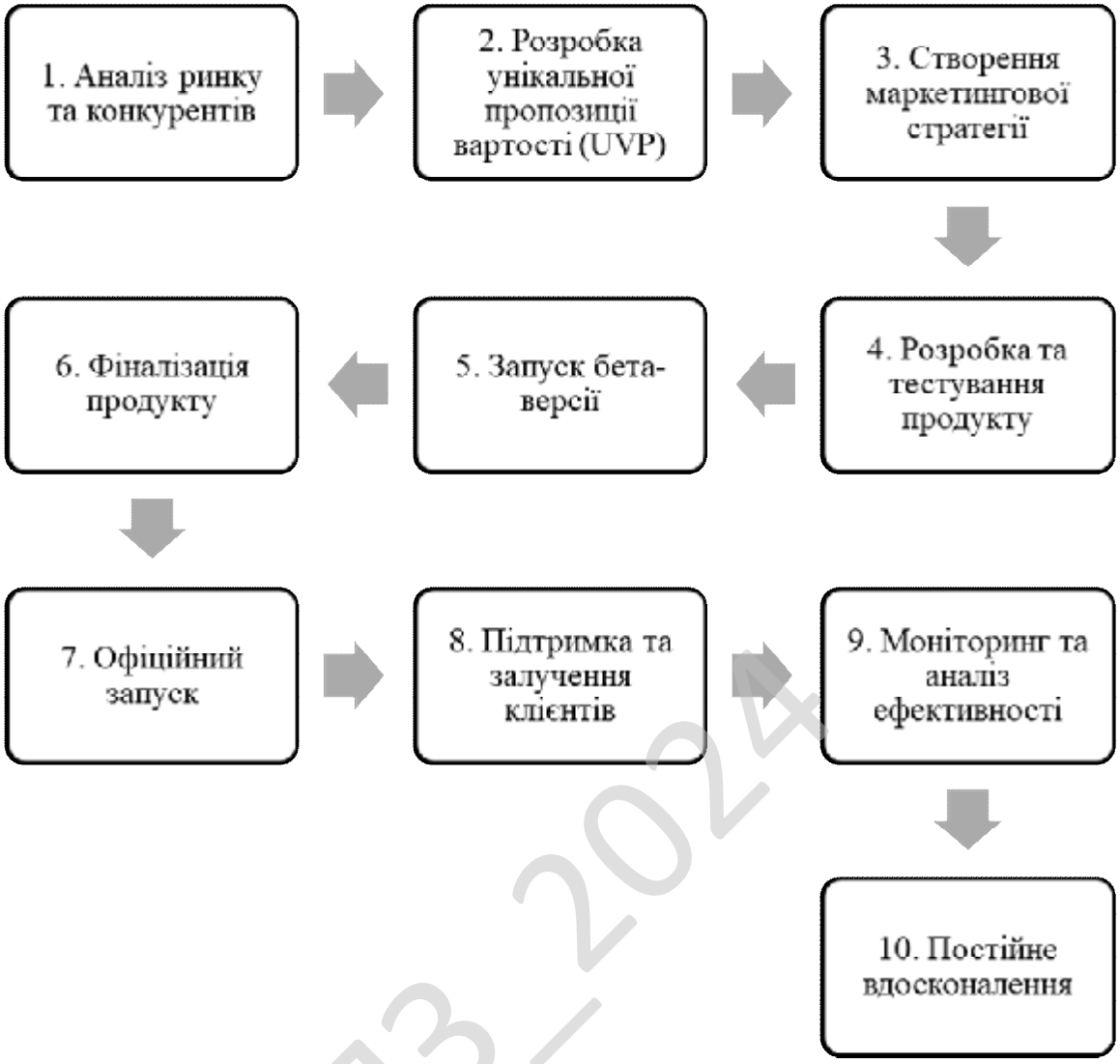

Рисунок 7.4 – Алгоритм просування проєкту

Цей алгоритм дозволяє систематично підходити до просування проєкту мобільного додатку для управління ERP-системою. Кожен етап допомагає забезпечити успішну реалізацію, залучення користувачів і підтримку конкурентоспроможності на ринку.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації мобільного додатку для управління ERP-системою можна розглянути стратегії наведені на рисунку 7.5.

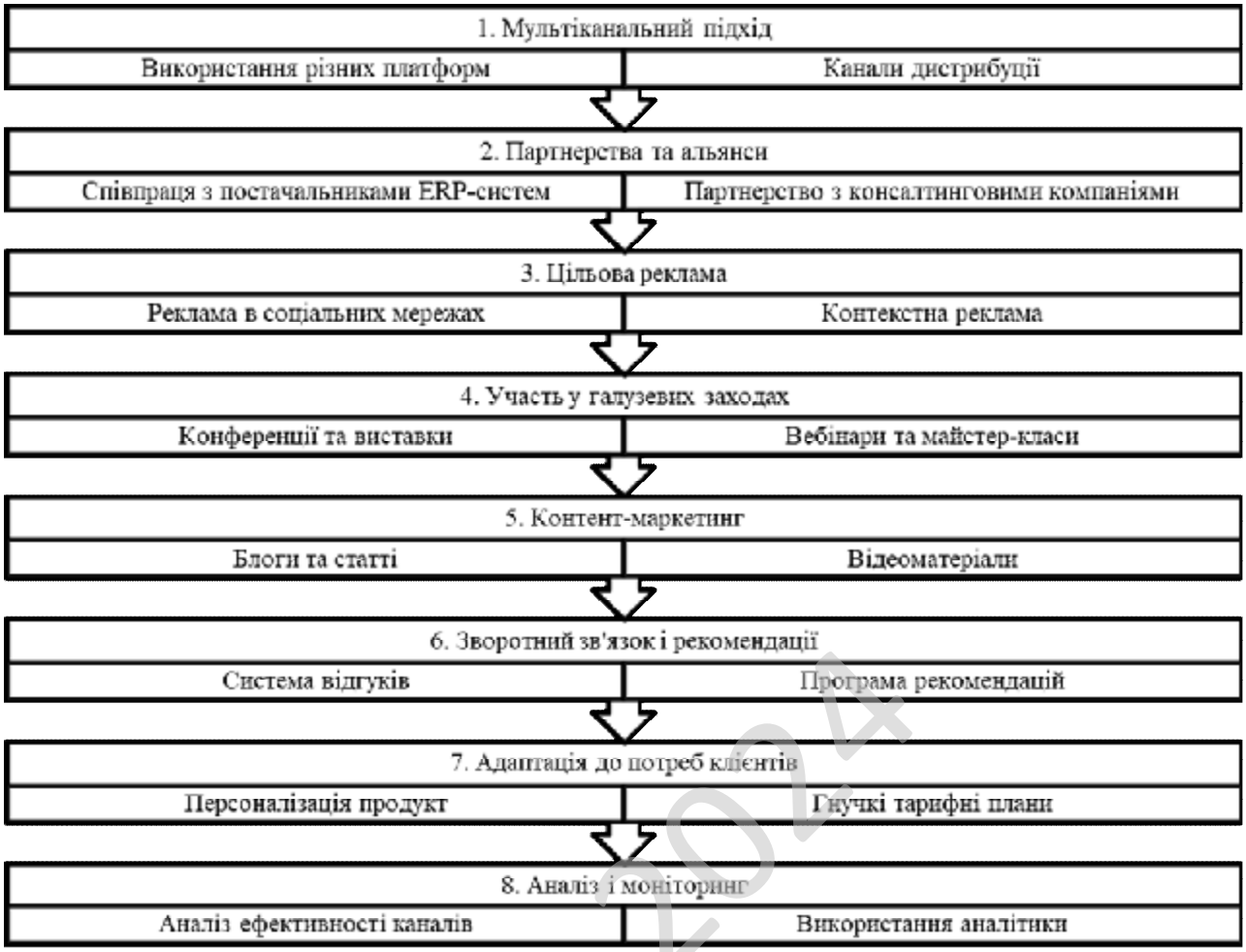

Рисунок 7.5 – Шляхи оптимізації каналів збуту

Оптимізація каналів збуту та шляхів реалізації проєкту мобільного додатку для управління ERP-системою вимагає комплексного підходу, що включає використання різних каналів, партнерств, маркетингових стратегій та безперервного зворотного зв'язку з користувачами. Це дозволить не тільки залучити нових клієнтів, але й утримувати існуючих.

7.7 Визначення ключових факторів успіху конкретного проєкту

Схематично ключові фактори успіху проєкту програмної реалізації мобільного додатку для управління ERP-системою виглядають так.

Рисунок 7.5 – Шляхи оптимізації каналів збуту

Ці фактори разом складають основу для успішної реалізації проєкту програмної реалізації мобільного додатку для управління ERP-системою. Їх врахування підвищить шанси на успіх і задоволеність кінцевих користувачів.

8 ЗАХОДИ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Комп'ютеризація на сучасному етапі відкрила якісно нові можливості в сфері виробництва. Вона дозволила автоматизувати процес проектування , без чого стали б нереальними сучасний рівень автомобіле – і літакобудування, створення космічних літальних апаратів і багато іншого. Комп'ютери стали невід'ємною частиною сучасного виробництва (верстати з числовим програмним управлінням, автоматизовані виробничі лінії), бізнесу (автоматизація бухгалтерського обліку, організація банківських операцій, використання пластикових карт мережі Internet для взаєморозрахунків), проникаютьу сферу обміну (Internet -магазини) і повсякденний побут (програмовані побутові прилади і персональні ЕОМ). Сучасні технології здатні полегшувати життя, гарантувати безпеку. Стрімко набувають популярності система «розумного будинку», яка створена для покращення умов проживання та полегшення щоденних клопотів у побуті, економії часу і коштів.

Небувалих висот досягла швидкість поширення інформації. Тепер події, що відбулися в одному кінці світу, стають відомі всьому світу всього за кілька хвилин. Саме тому XXI століття назвали інформаційним. Є багато сфер людської діяльності, де комп’ютер використовується як звичайний робочій інструмент та в деяких сферах його використання призводить до корінних змін в процедурі самої діяльності і функціях працівників, в їхніх ідеологічних позиціях та навіть в психологічних властивостях особистості, а також вливає на стан здоров’я користувача комп’ютера.

Темою магістерської роботи є розробка програмного продукту, дослідження та програмна реалізація програмного продукту. Виконання цих робіт пов’язана безпосередньо з роботою за комп’ютером.

У розділі даної магістерської роботи висвітлюються основні питання охорони праці працівників, робота яких пов’язана з роботою за комп’ютером,

планування робочого приміщення, де працюють програмісти ; параметри мікроклімату, освітленість робочих місць та приміщень; шумові завади та інтелектуальні навантаження..

8.1 Аналіз умов праці програміста

Приміщення відділу, що розраховане на 4 робочих місць, має 6 м в довжину, 5м в ширину та 3,2м у висоту. Приміщення розміщене на першому поверсі , обладнане системами опалення і припливно-витяжної вентиляцією, що є необхідністю в приміщеннях з роботою на комп'ютерах. Площа на одне робоче місце складає 7,5 $м^2$, об’єм – 24 $м^3$. Це відповідає нормальній організації робочого місця, так як за нормами площа відповідно до ДСанПіН 3.3.2.007-98 “Державні санітарні норми і правила роботи з візуальними дисплейними терміналами електронно-обчислювальних машин” площа на одне робоче місце користувача ПК повинна бути не менше 6 $м^2$, а об’єм не менше 20 $м^3$. Таким чином нормативи забезпечення працюючих робочою площею у відділі дотримано.

Поверхня підлоги рівна, покрита лінолеумом. Це оптимальне покриття для таких приміщень. Так як лінолеум неслизький і має антистатичні властивості. Вологе прибирання приміщення проводиться 1 раз на два дні, хоча за нормами повинна проводитися щоденно. Це негативно позначається на вологості повітря і чистоті приміщення, що безпосередньо пов'язано із самопочуттям персоналу. Забарвлення стін ясно-коричнева, стелі – біла, підлогу – паркетний, ясно-коричневий. Оздоблення приміщення виконано з урахуванням лікарів - психологів: кольору стін, стелі, статі гармоніюють між собою. З погляду колірної терапії, жовтий і ясно-коричневий кольору покращують настрій, позитивно впливають на нервову систему та внутрішні органи. У приміщенні стоять спеціальні шафи для зберігання документів і шафа для верхнього одягу, на вікнах встановлені регулюючі жалюзі.

Щодо стану мікроклімату у приміщенні з комп'ютерною технікою, то протягом року підтримуються нормальні значення температури, вологості повітря, і швидкість руху повітря, Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця користувачів ПК, встановлені оптимальні параметри мікроклімату, а при неможливості їх дотримання використовують допустимі параметри. Робота в нашому випадку, користувача ПК за енерговитратами відноситься до категорії легких робіт Іа, Іб. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях, де виконуються роботи операторського типу. (Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99).

Таблиця 8.1 - Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С; 40...60%; 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщенні відповідають вказаним у таблиці для теплого і холодного періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату у приміщені за допомогою потужного кондиціонера проводиться кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції.

Приміщення має природне і штучне освітлення. Природне - за рахунок великих широких вікон, які виходять на захід, а штучне - за рахунок люмінесцентних ламп. Термін експлуатації деяких ламп підходить до кінця, тому вони світяться помаранчевий кольором. Однак такі лампи на підприємстві не міняють до повного їх перегорання. Це знижує рівень освітленості приміщення

до 250 лк, що дає додаткове навантаження на зір співробітників. Так як нормований рівень освітлення робочого місця з ПК, згідно ДБН В.2.5- 28 – 2006 «Природне і штучне освітлення» повинен складати 300-500 лк, то це негативно впливає на роботу і здоров'я персоналу.

Інтелектуальне і психологічно - емоційне перенапруження програміста

Діяльність програміста вимагає напруження волі для забезпечення необхідного рівня уваги, що змушує докладати великі зусилля і супроводжується подальшим виснаженням енергетичних ресурсів організму. Праця характеризується високим рівнем психічного навантаження, Наслідками впливу такого роду навантажень призводять до швидкого перевтоми, головних болів, поганому сну і зниження працездатності. Інтелектуальних навантажень людина піддається не тільки сидячи за ПК, а й поза його при вирішенні інших завдань, не пов'язаних з набором тексту, формул, програмуванням. Однак при наявності ПК ступінь інтелектуальних і психологічно - емоційних навантажень зростає. Така небезпека проявляється при виконанні відповідальних завдань з використанням ПК. Ступінь її впливу залежить від рівня складності виконуваного завдання. Програміст схильний її впливу протягом усієї робочої зміни. Для того щоб знизити ризик перевтоми, виснаження нервово-психічних ресурсів організму, потрібно робити регламентовані перерви в роботі.

8.2 Розробка заходів з умов поліпшення охорони праці

Проводячи аналіз умов праці в розглянутому приміщенні, ми одержали наступні результати:

- обсяг приміщення, що приходиться на одному працюючого, відповідає нормативному значенню;
- показники мікроклімату відповідають нормативному значенню;
- акустичні умови роботи в нормі.

Виходячи із наступного можна зробити висновок, що основною причиною втомлюваності та зниження працездатності програміста є психофізіологічний фактор, тому основною пропозицією правильна організація робочого місця з урахуванням ергономічних вимог , а також дотримання регламентованого режиму праці та відпочинку.

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи.

8.3 Дослідження інформаційного навантаження на програміста

Програміст, у залежності від підготовки і досвіду, вирішує задачі різної складності, але в загальному випадку робота програміста будується по наступному алгоритму наведеному в таблиці 8.1 – 8.2.

Таблиця 8.1 – Постановка задачі

Етап	Зміст	Витрата часу, %
III	Постановка задачі Вивчення матеріалу за поставленою задачею	6.25
III	Визначення методу рішення задачі	6.25
IV	Складання алгоритму рішення задачі	12.5
V	Програмування	25
VI	Налагодження програми, складання звіту	50

Таблиця 8.2 – Алгоритм рішення задачі

Етап	Член алгоритму	Зміст роботи	Літерне позначення
1	2	3	4
I	1	Одержання першого варіанта	A
	7	Складання і уточнення технічного	Ri
		Одержання остаточного варіанта технічного завдання	Сіл
	4	Складання переліку матеріалів,	HII2
	5	Вивчення матеріалів по тематиці	Аг
	6	Вибір метолу рішення	C2J3
	7	Уточнення й узгодження обраного	B2
	8	Остаточний вибір методу рішення	СчТ4
	9	Аналіз вхідної і вихідної інформації	B2
	10	Вибір мови програмування	C4TS
	11	Визначення структури програми	H3C5q
	12	Складання блок-схеми програми	C6q2
	13	Логічний аналіз програми і	F1H4W2
	14	Компіляція програми	F2
	15	Виправлення помилок	D1W2
	16	Редагування програми	F2H5B3W
	17	Виконання програми	F3

Підрахуємо кількість членів алгоритму і їхню частоту (імовірність) щодо загального числа, прийнятого за одиницю. Імовірність повторення і-ої ситуації визначається по формулі:

$$Pi = к/п, \quad (8.1)$$

де к - кількість повторень кожного елемента одного типу, п - сумарна кількість повторень від джерела інформації, одного типу.

Результати розрахунку зведемо в таблицю 8.3:

Таблиця 8.3 – Кількісні характеристики алгоритму

Джерело	Члени алгоритму	Символ	Кількість	Частота повторень
1	2	3	4	5
1	Аферентні -		6	1,00
	Вивчення	А	2	0,33
	Спостереження	Р	4	0,67
2	Еферентні -		18	1,00
	Уточнення	В	3	0,17
	Вибір	С	8	0,44
	Виправлення	0	1	0.06
	Аналіз	н	6	0,33
	Виконання	к	0	0
3	Логічні умови		13	1,00
	Прийняття рішень на основі	І	5	0,39
	Грабічні	Ч	2	0.15
	Отриманого	V	6	0.46
	Усього:		37	

Кількісні характеристики алгоритму (табл.8.3) дозволяють розрахувати інформаційне навантаження програміста. Ентропія інформації елементів кожного джерела інформації розраховується по формулі, біт/сигн:

$$H_j = \sum_{i=1}^{m} p_i \log_2 p_i, \quad (8.2)$$

де т - число однотипних членів алгоритму розглянутого джерела інформації:

H1=2 × 2+2 × 4 = 12

H2 = 3 × 1,585 +8 × 3+0+6 × 2,585 = 44,265

НЗ = 5 × 2,323 + 2 × 1+6 + 2,585 = 29,125

Потім визначається загальна ентропія інформації, біт/сигн:

Hs=Hl+H2+H3, (8.3)

де HI, H2, НЗ - ентропія аферентних, еферентних елементів і логічних умов відповідно.

Hs = 10 + 44,265 + 29,125 = 83,39. Далі визначається потік інформаційного навантаження біт/хв,

$$F = \frac{H_{\sum} \cdot N}{t}, \quad (8.4)$$

де N - сумарне число всіх членів алгоритму; I - тривалість виконання всієї роботи, хв.

Від кожного джерела в інформації (члена алгоритму) у середньому надходить 3 інформаційних сигнали в годину, час роботи - 225 годин,

$$\Phi = \frac{83{,}39 \cdot 37 \cdot 3 \cdot 225}{13500} = 2{,}6 \text{ біт/с}$$

Розраховане інформаційне навантаження порівнюється з припустимою. При необхідності приймається рішення про зміни в трудовому процесі.

Умови нормальної роботи виконуються при дотриманні співвідношення:

$$\Phi_{доп.мин} < \Phi_{расч} < \Phi_{доп.макс}, \quad (8.5)$$

де $\Phi_{доп.мін}$, і $\Phi_{доп.макс}$ мінімальний і максимальний припустимі рівні інформаційних навантажень (0,8 і 3,2 біт/з відповідно);

$\Phi_{расч}$ - розрахункове інформаційне навантаження 0,8 < 2,6 <3,2

Висновки до розділу

У цій частині магістерської роботи були розглянуті вплив факторів трудового і виробничого середовища програмістів, дослідження інформаційного навантаження на програміста. Дотримання умов, що визначають оптимальну організацію робочого місця програміста і навантаження, отримані ним в процесі роботи, дозволить зберегти гарну працездатність протягом усього робочого дня та підвищить продуктивність праці програміста.

9 ОСНОВНІ ВИСНОВКИ

У сучасних умовах інноваційно-інформаційного суспільства практично всі суб’єкти використовують різні мобільні пристрої та програми, які дозволяють отримати доступ до відповідних даних. Мобільні додатки дозволяють ефективно інтегрувати інформацію з соціальними мережами, веб-сайтами компаній, мультимедійним контентом і засобами зв’язку. Вони активно впроваджуються приладобудівними підприємствами і значно полегшують роботу з розрахунків, характеристики місцевості та планування робіт. В магістерській роботі зроблено теоретичне узагальнення та запропоновано нові практичні рекомендації щодо розробки мобільного додатку.

Результати дослідження, проведеного на основі зазначених електронних та письмових джерел, дають змогу визначити, що мобільний додаток – це автономне програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях, яке встановлюється через маркет: портали. , магазини, маркети з метою оптимізації та вирішення завдань користувача

Сьогодні існує кілька підходів до технічної реалізації додатків для мобільних пристроїв, а саме нативний, гібридний, прогресивний (PVA). Серед їх видів, які найчастіше використовуються для підприємств і клієнтів, слід виділити наступні:

- автоматизувати процеси;

- підвищення продуктивності, кооперації та спільної праці;

- для «продовження» онлайн-сервісів і мобільних додатків як звичайна картка користувача.

Ключовою перевагою використання мобільних додатків для підприємців стала можливість ретаргетингу, що передбачає побудову потужного бренду та розширення ринків збуту за рахунок аналізу поведінки клієнтів та нагадування

про невиконані завдання за допомогою push. повідомлень, використання інформації про вподобання клієнтів на основі історії пошуку та покупок для доставки реклами, врахування даних сервісів геолокації, інтеграція з соціальними мережами.

Проведений SWOT-аналіз дозволив визначити, що гібридна крос-платформна розробка є найбільш оптимальним методом, який потребує в середньому на 66,7% менше витрат порівняно з одночасною нативною розробкою окремих програм для Android та iOS. Основні переваги: економічність, простота розробки та використання інструменту, використання без доступу до Інтернету, зручний інтерфейс користувача та швидке встановлення. Однак основним недоліком прогресивних веб-додатків є недостатня підтримка пристроїв.

У результаті була розроблена програма та проведено тестування з нею, щоб можна було оцінити її вплив на процес управління компанією. В результаті тестів, згідно з діаграмами, було зроблено висновок, що програма прискорює процес видачі замовлень в компанії під час виконання виробничого процесу, а також зменшує кількість помилок, які можуть бути допущені під час управління. Отже, створена система відповідає заявленим вимогам, всі поставлені завдання виконано, система повністю впроваджена.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бахрушин В. Є. Методи аналізу даних: Навч. посібник / В. Є. Бахрушин. – Запоріжжя: КПУ, 2011. – 268 с

2. Шумейко А. А. Интеллектуальный анализ данных (Введение в Data Mining) / А. А. Шумейко, С. Л. Сотник. – Днепропетровск: Белая Е. А., 2015. – 212 с.

3. http://www.nbuv.gov.ua/eb/ep.html - Національна бібліотека України імені В.І.Вернадського

4. Node.js v14.0.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/

5. М. Кантелон , М. Хартер, Т. Головайчук, Н. Райлих // “Node.js в действии”.

6. Янг А., Мек Б., Кантелон М. // “Node.js в действии. 2-е издание”.

7. John Resig, Bear Bibeault, Josip Maras // “Secrets of theJavaScript Ninja”.

8. Express [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/

9. Фреймворк AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS

10. AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://angularjs.org/

11. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/

12. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://https://uk.reactjs.org/

13. AngularJS MVC [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

14. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

15. Angular 2 [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/

16. Односторінковий застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Односторінковий_застосунок

18. Build Node.js Apps [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

19. REST [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/REST

20. What is REST [Електронний ресурс] – Режим доступу до ресурсу: https://restfulapi.net/

22.Веб приложение [Електронний ресурс] – Режим доступу до ресурсу: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/

23.Мова розмітки гіпертексту [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

24. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML

25. Каскадні таблиці стилів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS

26. Стек MEAN [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MEAN_(веброзробка)

27. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

28. Веб-технології для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web

29. CORS, XSS [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/maleta/cors-xss-and-csrf-with-examples-in-10-minutes-35k3

30. AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AJAX

31. Fetch API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API

32. AngularJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/angular/default.asp

33. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JQuery

34. Основи Web-технологій [Електронний ресурс] – Режим доступу до ресурсу: https://pidruchniki.com/1243020547796/informatika/web-tehnologiyi_pidpriyemstvah

35. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/

36. Install MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/administration/install-on-linux/

37. Как установить Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/node-js-ubuntu-18-04-ru

38. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linux

39. npm-audit [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/cli/audit

40. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

41. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

42. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/compass

43. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/

44. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

45. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

46. Охорона праці [Електронний ресурс]: реферат – Режим доступу до ресурсу: https://revolution.allbest.ru/life/00468031_0.html

47. Природне і штучне освітлення ДБН В.2.5-28:2018: державні будівельні норми України [Електронний ресурс] / Ю. Громадський, С. Облакевич, М.Громадський, Г. Фаренюк, Є. Фаренюк, О. Підгорний, О. Сергейчук, Є.Рейцен, В. Єгорченков, Л. Коваль, Д. Радомцев, В. Злоба, Н. Кучеренко, Г.Кожушко, О. Гончар, О. Козенко, Б. Шабашкевіч, Ю. Добровольський, В.Акіменко, С. Гозак, А. Яригін, В. Назаренко, В. Мартиросова, В. Сорокін, Є.Пугачов - Київ 2018 - Режим доступу до ресурсу: https://ledeffect.com.ua/images/__branding/dbn2018.pdf

48. Розрахунок світлодіодного освітлення кімнати [Електронний ресурс] – Режим доступу до ресурсу: https://luxled.biz.ua/rozrahynok-svitlodoidnogo-osvitlennja-kimnatu-v-kvarturi-abo-bydunky

49. Охорона праці, охорона праці та безпека в надзвичайних ситуаціях : метод. вказ. до викон. розділів у дипломних роботах / [укл. В.М. Челябієва, О.Л. Гуменюк] - Чернігів ЧДТУ 2013 - [Електронний ресурс] – Режим доступу до ресурсу: http://ir.stu.cn.ua/bitstream/handle/123456789/12461/Охорона праці та безпека. в надзв. ситуац;метод.вказ..pdf?sequence=1&isAllowed=y

50. Освітленість робочих місць [Електронний ресурс] – Режим доступу до ресурсу:https://ua-referat.com/Освітленість_робочих_місць_сучасні_підходи_до_вимірів_і_оцінки

51. Температурний режим праці [Електронний ресурс] – Режим доступу до ресурсу: http://poltava.medprof.org.ua/poltava/zakhist-trudovikh-ta-socialno-ekonomichnikh-prav-pracivnikiv-galuzi/pravova-dopomoga/temperaturnii-rezhim-praci-jakim-vin-maje-buti/

52. Шум. Методи захисту від його дії : метод. вказ. до лабораторної роботи / [укл. В. І. Шмирко, С. М. Журавель] — Запоріжжя: ЗНТУ, 2014. - 14 с. [Електронний ресурс] – Режим доступу до ресурсу: https://zp.edu.ua/sites/default/files/konf/oop_shum-2014.pdf

53. Системи протипожежного захисту ДБН В.2.5-56:2014 / Б. Платкевич, В. Носач, В. Федюк, В. Мусійчук, В. Євстіфєєв, Г. Дубінський, В. Сокол, А.Бушиленко, В. Дунюшкін, Р. Уханський, С. Пономарьов, В. Приймаченко, А.Приймаченко, С. Пітайчук, Н. Морозова, І. Колосов, О. Лагода, П. Мізін, В.Савченко, М. Федорович, П. Шаповалов, Л. Фесенко — Київ 2015 — [Електронний ресурс] – Режим доступу до ресурсу: http://kbu.org.ua/assets/app/documents/dbn2/98.1. ДБН В.2.5-56~2014. Системи протипожежного захисту.pdf

54. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ IBM−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с.[Електронний ресурс] – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

55. The Top JavaScript Frameworks [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

56. JavaScript Frameworks 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/migrate-mongo

57. Web Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/web-technology

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку програмного забезпечення ERP- системи та мобільного застосунку для її використання

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою розробки є дослідження та програмна реалізація мобільного додатку ERP системи для промислового підприємства та аналізу впливу його роботи на ефективність роботи підприємства.

4 Джерела розробки

Джерелом цієї магістерської роботи є відносна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– Розробку додатку ;

– систему підключення та тестування баз даних ;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Мова програмування Java, PHP-7.3.

СУБД MySQL 8.0

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Виконати маркетингове та економічне обґрунтування іт-проєкту

7.2 Виконати розрахунок економічної ефективності від впровадження реалізованого

8 Вимоги щодо охорони праці

В частині охорони праці та техніки безпеки в магістерській роботі повинен бути розглянутий аналіз умов праці програміста та розрахунок інформаційного навантаження на програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 3 аркуша.
– Маркетингове та економічне обґрунтування іт-проєкту – 1 аркуш.
– Пояснювальна записка – 98 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

11.1 Подання кваліфікаційної магістерської роботи на попередній захист 04.12.2024 р.

1.2 Подання магістерської роботи на захист 04.12.2024 р.

Додаток Б(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Кислун О.А.

Дослідження та програмна реалізація мобільного додатку для управління ERP-системою

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2024 року

Підключення до бази даних (DatabaseConnection.java)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL =
"jdbc:mysql://localhost:3306/erp_system";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Модель даних для клієнта (Client.java)

```
java
Копіювати код
public class Client {
    private int id;
    private String name;
    private String email;
    private String phone;

    public Client(int id, String name, String email, String phone) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.phone = phone;
    }

    // Getters and Setters
}
```

CRUD-операції для клієнтів (ClientDAO.java)

```
java
Копіювати код
import java.sql.*;
```

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

public class ClientDAO {

    public void addClient(Client client) throws SQLException {
        String query = "INSERT INTO clients (name, email, phone) VALUES
(?, ?, ?)";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, client.getName());
            stmt.setString(2, client.getEmail());
            stmt.setString(3, client.getPhone());
            stmt.executeUpdate();
        }
    }

    public Client getClientById(int id) throws SQLException {
        String query = "SELECT * FROM clients WHERE id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, id);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                return new Client(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("email"),
                    rs.getString("phone")
                );
            }
        }
        return null;
    }

    public List<Client> getAllClients() throws SQLException {
        List<Client> clients = new ArrayList<>();
        String query = "SELECT * FROM clients";
```

```
        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            while (rs.next()) {
                clients.add(new Client(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("email"),
                    rs.getString("phone")
                ));
            }
        }
        return clients;
    }

    public void updateClient(Client client) throws SQLException {
        String query = "UPDATE clients SET name = ?, email = ?, phone = ?
WHERE id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, client.getName());
            stmt.setString(2, client.getEmail());
            stmt.setString(3, client.getPhone());
            stmt.setInt(4, client.getId());
            stmt.executeUpdate();
        }
    }

    public void deleteClient(int id) throws SQLException {
        String query = "DELETE FROM clients WHERE id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, id);
            stmt.executeUpdate();
        }
    }
}
```

Схема бази даних MySQL

```
CREATE DATABASE erp_system;
USE erp_system;

CREATE TABLE clients (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(50),
    phone VARCHAR(15)
);
```

Головний клас для взаємодії з ERP-системою (Main.java)

```
java
import java.sql.SQLException;

public class Main {
    public static void main(String[] args) {
        ClientDAO clientDAO = new ClientDAO();

        // Створення нового клієнта
        Client client = new Client(0, "John Doe", "john@example.com",
"123456789");
        try {
            clientDAO.addClient(client);
            System.out.println("Клієнта додано!");

            // Отримання всіх клієнтів
            clientDAO.getAllClients().forEach(c ->
                System.out.println("ID: " + c.getId() + ", Name: " +
c.getName())
            );

            // Оновлення клієнта
            client.setName("John Smith");
            clientDAO.updateClient(client);
            System.out.println("Клієнта оновлено!");

            // Видалення клієнта
```

```
                clientDAO.deleteClient(client.getId());
                System.out.println("Клієнта видалено!");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

Контролер (Java Spring Boot)

```
@RestController
@RequestMapping("/api/clients")
public class ClientController {

    @Autowired
    private ClientService clientService;

    @GetMapping
    public List<Client> getAllClients() {
        return clientService.getAllClients();
    }

    @GetMapping("/{id}")
    public Client getClient(@PathVariable int id) {
        return clientService.getClientById(id);
    }

    @PostMapping
    public Client createClient(@RequestBody Client client) {
        return clientService.addClient(client);
    }

    @PutMapping("/{id}")
    public Client updateClient(@PathVariable int id, @RequestBody Client client)
{
        client.setId(id);
        return clientService.updateClient(client);
    }
```

```
    @DeleteMapping("/{id}")
    public void deleteClient(@PathVariable int id) {
        clientService.deleteClient(id);
    }
```

Мобільний застосунок (Android)

Залежності у build.gradle

groovy

```
dependencies {
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.recyclerview:recyclerview:1.3.0'
}
```

Модель даних Client.java

```
public class Client {
    private int id;
    private String name;
    private String email;
    private String phone;

    // Getters and Setters
}
```

API-інтерфейс ApiService.java

```
import retrofit2.Call;
import retrofit2.http.*;

import java.util.List;

public interface ApiService {
    @GET("clients")
    Call<List<Client>> getClients();

    @POST("clients")
    Call<Client> addClient(@Body Client client);

    @PUT("clients/{id}")
```

```
    Call<Client> updateClient(@Path("id") int id, @Body Client client);

    @DELETE("clients/{id}")
    Call<Void> deleteClient(@Path("id") int id);
}
```

Клієнт Retrofit RetrofitClient.java

```
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
    private static final String BASE_URL = "http://your-server-address/api/";
    private static Retrofit retrofit;

    public static Retrofit getInstance() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit;
    }
}
```

Головний екран MainActivity.java

```
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

import java.util.List;
```

```
public class MainActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private ClientAdapter clientAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        fetchClients();
    }

    private void fetchClients() {
        ApiService apiService =
RetrofitClient.getInstance().create(ApiService.class);
        apiService.getClients().enqueue(new Callback<List<Client>>() {
            @Override
            public void onResponse(Call<List<Client>> call,
Response<List<Client>> response) {
                if (response.isSuccessful() && response.body() != null) {
                    clientAdapter = new ClientAdapter(response.body());
                    recyclerView.setAdapter(clientAdapter);
                } else {
                    Toast.makeText(MainActivity.this, "Failed to load clients",
Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<List<Client>> call, Throwable t) {
                Toast.makeText(MainActivity.this, "Error: " + t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Адаптер для відображення клієнтів ClientAdapter.java

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;

public class ClientAdapter extends
RecyclerView.Adapter<ClientAdapter.ClientViewHolder> {

    private final List<Client> clients;

    public ClientAdapter(List<Client> clients) {
        this.clients = clients;
    }

    @NonNull
    @Override
    public ClientViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.client_item, parent,
false);
        return new ClientViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ClientViewHolder holder, int position)
{
        Client client = clients.get(position);
        holder.nameTextView.setText(client.getName());
        holder.emailTextView.setText(client.getEmail());
    }

    @Override
    public int getItemCount() {
```

```
        return clients.size();
    }

    static class ClientViewHolder extends RecyclerView.ViewHolder {
        TextView nameTextView;
        TextView emailTextView;

        public ClientViewHolder(@NonNull View itemView) {
            super(itemView);
            nameTextView = itemView.findViewById(R.id.nameTextView);
            emailTextView = itemView.findViewById(R.id.emailTextView);
        }
    }
}
```

Інтерфейс

activity_main.xml

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

client_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/nameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Client Name"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/emailTextView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Client Email"
        android:textSize="14sp" />
</LinearLayout>
```

Основний стиль у файлі styles.xml

```
<resources>
    <!-- Загальний стиль додатка -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">#6200EE</item>
        <item name="colorPrimaryDark">#3700B3</item>
        <item name="colorAccent">#03DAC5</item>
        <item name="android:textColorPrimary">#000000</item>
        <item name="android:windowBackground">#FFFFFF</item>
    </style>

    <!-- Стиль для елементів RecyclerView -->
    <style name="RecyclerViewStyle">
        <item name="android:divider">?android:attr/listDivider</item>
        <item name="android:dividerHeight">1dp</item>
        <item name="android:padding">8dp</item>
    </style>

    <!-- Стиль для кожного елемента списку -->
    <style name="ListItemStyle">
        <item name="android:textColor">#333333</item>
        <item name="android:textSize">16sp</item>
        <item name="android:padding">8dp</item>
        <item name="android:background">?attr/selectableItemBackground</item>
    </style>

    <!-- Стиль для кнопок -->
    <style name="ButtonStyle">
        <item name="android:backgroundTint">#6200EE</item>
        <item name="android:textColor">#FFFFFF</item>
```

```
        <item name="android:padding">12dp</item>
        <item name="android:cornerRadius">8dp</item>
    </style>
</resources>
```

Файл colors.xml (ресурси кольорів)

```
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
    <color name="textColorPrimary">#000000</color>
    <color name="textColorSecondary">#757575</color>
    <color name="listItemBackground">#F5F5F5</color>
</resources>
```

Файл dimens.xml (розміри елементів)

```
<resources>
    <dimen name="text_size_large">18sp</dimen>
    <dimen name="text_size_medium">16sp</dimen>
    <dimen name="text_size_small">14sp</dimen>
    <dimen name="padding_small">8dp</dimen>
    <dimen name="padding_medium">16dp</dimen>
    <dimen name="padding_large">24dp</dimen>
</resources>
```

Оформлення списків у RecyclerView

Оновлений client_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="@color/listItemBackground"
    android:padding="@dimen/padding_medium">

    <TextView
        android:id="@+id/nameTextView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/textColorPrimary"
        android:textSize="@dimen/text_size_large"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/emailTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/textColorSecondary"
        android:textSize="@dimen/text_size_medium" />
</LinearLayout>
```

Оформлення кнопок

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="@dimen/padding_medium">

    <Button
        android:id="@+id/addButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Add Client"
        style="@style/ButtonStyle" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        style="@style/RecyclerViewStyle" />
</LinearLayout>
```

Тема у AndroidManifest.xml

Переконайтеся, що у вашому AndroidManifest.xml використовується створена тема: android:theme="@style/AppTheme">

```
    ...
</application>
```

Додавання нового клієнта через форму

Файл AddClientActivity.java

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class AddClientActivity extends AppCompatActivity {

    private EditText nameEditText, emailEditText, phoneEditText;
    private Button saveButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_client);

        nameEditText = findViewById(R.id.nameEditText);
        emailEditText = findViewById(R.id.emailEditText);
        phoneEditText = findViewById(R.id.phoneEditText);
        saveButton = findViewById(R.id.saveButton);

        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saveClient();
            }
        });
```

```
    }

    private void saveClient() {
        String name = nameEditText.getText().toString();
        String email = emailEditText.getText().toString();
        String phone = phoneEditText.getText().toString();

        if (name.isEmpty() || email.isEmpty() || phone.isEmpty()) {
            Toast.makeText(this, "Please fill all fields",
Toast.LENGTH_SHORT).show();
            return;
        }

        Client newClient = new Client(0, name, email, phone);
        ApiService apiService =
RetrofitClient.getInstance().create(ApiService.class);
        apiService.addClient(newClient).enqueue(new Callback<Client>() {
            @Override
            public void onResponse(Call<Client> call, Response<Client> response)
{
                if (response.isSuccessful()) {
                    Toast.makeText(AddClientActivity.this, "Client added
successfully!", Toast.LENGTH_SHORT).show();
                    finish();
                } else {
                    Toast.makeText(AddClientActivity.this, "Failed to add
client", Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<Client> call, Throwable t) {
                Toast.makeText(AddClientActivity.this, "Error: " +
t.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Файл activity_add_client.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="@dimen/padding_medium">

    <EditText
        android:id="@+id/nameEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/emailEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/phoneEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Phone"
        android:inputType="phone" />

    <Button
        android:id="@+id/saveButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Save"
        style="@style/ButtonStyle" />
</LinearLayout>
```

Видалення клієнта зі списку

Оновлення ClientAdapter.java

```
import android.view.View;
import android.widget.Button;

public class ClientAdapter extends
RecyclerView.Adapter<ClientAdapter.ClientViewHolder> {

    private final List<Client> clients;
    private final ApiService apiService =
RetrofitClient.getInstance().create(ApiService.class);

    public ClientAdapter(List<Client> clients) {
        this.clients = clients;
    }

    @NonNull
    @Override
    public ClientViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.client_item, parent,
false);
        return new ClientViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ClientViewHolder holder, int position)
{
        Client client = clients.get(position);
        holder.nameTextView.setText(client.getName());
        holder.emailTextView.setText(client.getEmail());

        holder.deleteButton.setOnClickListener(v -> {
            apiService.deleteClient(client.getId()).enqueue(new Callback<Void>()
{
                @Override
                public void onResponse(Call<Void> call, Response<Void> response)
{
                    if (response.isSuccessful()) {
```

```
                        clients.remove(position);
                        notifyItemRemoved(position);
                        Toast.makeText(holder.itemView.getContext(), "Client
deleted", Toast.LENGTH_SHORT).show();
                    }
                }

                @Override
                public void onFailure(Call<Void> call, Throwable t) {
                    Toast.makeText(holder.itemView.getContext(), "Error: " +
t.getMessage(), Toast.LENGTH_SHORT).show();
                }
            });
        });
    }

    @Override
    public int getItemCount() {
        return clients.size();
    }

    static class ClientViewHolder extends RecyclerView.ViewHolder {
        TextView nameTextView;
        TextView emailTextView;
        Button deleteButton;

        public ClientViewHolder(@NonNull View itemView) {
            super(itemView);
            nameTextView = itemView.findViewById(R.id.nameTextView);
            emailTextView = itemView.findViewById(R.id.emailTextView);
            deleteButton = itemView.findViewById(R.id.deleteButton);
        }
    }
}
```

Оновлений client_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="@dimen/padding_small">

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:id="@+id/nameTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/textColorPrimary"
            android:textSize="@dimen/text_size_large" />

        <TextView
            android:id="@+id/emailTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/textColorSecondary"
            android:textSize="@dimen/text_size_medium" />
    </LinearLayout>

    <Button
        android:id="@+id/deleteButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Delete"
        android:textColor="@color/white"
        android:backgroundTint="@color/colorAccent" />
</LinearLayout>
```

Пошук клієнтів

activity_main.xml

```
<EditText
```

```
    android:id="@+id/searchEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Search clients"
    android:inputType="text" />

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />
```

пошук MainActivity.java

```
java
searchEditText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int
after) {}

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count)
{
        clientAdapter.getFilter().filter(s);
    }

    @Override
    public void afterTextChanged(Editable s) {}
});
```