Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Автоматики і комп'ютеризованих технологій_____
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____

Розроблення ERP-системи керування ресурсами складу приладобудівного виробництва
(тема)

Виконав:
здобувач ____4_____ року навчання,
групи ____АКТАКІТ-21-3
_____Філіп ПУСТОВОЙТЕНКО
(власне ім'я, прізвище)

Спеціальність _151 Автоматизація та комп'ютерно інтегровані технології_____
(код і повна назва спеціальності)

Тип програми ____освітньо-професійна_____

Освітня програма Автоматизація комп'ютерно-інтегрованих технологій
(повна назва освітньої програми)

Керівник _доцент Артем БРОННІКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _НЕВЛЮДОВ І. Ш.__
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Пустовойтенко Філіп Анатолійович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

02 червня 2025 р.  Філіп ПУСТОВОЙТЕНКО

Харківський національний університет радіоелектроніки

Факультет___Автоматики та комп'ютеризованих технологій _____

Кафедра_Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки_

Рівень вищої освіти_____перший (бакалаврський)_____

Спеціальність_151 Автоматизація комп'ютерно-інтегрованих технологій___
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма___Автоматизація комп'ютерно-інтегрованих технологій__
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«28»___квітня___ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві___Пустовойтенку Філіпу Анатолійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення ERP-системи керування ресурсами складу Приладобудівного виробництва

Затверджена наказом університету від 19.05 2025 р. № 390 Ст.

2. Термін подання здобувачем роботи до екзаменаційної комісії 06.06 2025 р.

3. Вихідні дані до роботи Фреймворк для розробки Django, база даних PostgeSQL, Клієнт-серверна архітектура

4. Перелік питань, що потрібно опрацювати в роботі

4.1 Вступ

4.2 Аналіз предметної області

4.3 Вибір і обґрунтування технічних засобів для створення програмного забезпечення ERP системи

4.4 Розроблення програмного забезпечення ERP системи

4.5 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Слайди у форматі PowerPoint у кількості 10 слайдів з розширенням .pptx

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	28.04.2025	виконано
2	Опрацювання літератури за темою роботи.	29.04.2025	виконано
3	Виконання розділу 1 Аналіз предметної області	01.05.2025	виконано
4	Виконання розділу 2 Вибір та обґрунтування технічних засобів для розробки ERP системи	04.05.2025	виконано
5	Виконання розділу 3 Розроблення програмного забезпечення ERP системи	10.05.2025	виконано
6	Подання роботи на перевірку Інтернет-сервісом StrikePlagiarism	04.06.2025	виконано
7	Оформлення пояснювальної записки	05.06.2025	виконано
8	Оформлення презентації	05.06.2025	виконано
9	Подання роботи на рецензію	07.06.2025	виконано
10	Подання роботи на підпис зав. кафедри		виконано
11	Подання атестаційної роботи в ЕК		виконано

Дата видачі завдання 28.04 2025 р.

Здобувач _____ Філіп ПУСТОВОЙТЕНКО
(підпис)

Керівник роботи _____ Доцент Артем БРОННІКОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 83 ст., 40 рис., 18 джерел, 3 додатки.

PYTHON, ERP, INDUSTRY, DJANGO, MES, СИСТЕМА, ПІДПРИЄМСТВО

Мета роботи – покращення ефективності роботи персоналу приладобудівного виробництва шляхом розробки автоматизованої ERP системи для керування ресурсами складу на приладобудівному виробництві, враховуючи переваги та недоліки існуючих продуктів. Також потрібно провести аналіз існуючих систем, з метою виявлення ключових особливостей і тонкощів при розробці системи з мінімальними витратами при інтеграції системи, та з мінімальними витратами для її підтримки, використовуючи програмні засоби, зазначені в 2 розділі кваліфікаційної роботи.

Об’єкт розробки – процес керування приладобудівним виробництвом.

Предмет розробки – система керування ресурсами складу приладобудівного виробництва.

ABSTRACT

Explanatory note 83 p., 40 figures, 18 sources, 3 appendicles

PYTHON, ERP, INDUSTRY, DJANGO, MES, SYSTEM, ENTERPRISE

The purpose of the work is to improve the efficiency of instrument-making staff by developing an automated ERP system for managing warehouse resources in instrument-making, taking into account the advantages and disadvantages of existing products. It is also necessary to analyze existing systems in order to identify key features and subtleties in the development of the system with minimal costs for system integration, and with minimal costs for its support, using the software tools specified in Chapter 2 of the qualification work.

Translated with DeepL.com (free version)Object of development – instrumentation production management.

Subject of development – a system for managing warehouse resources in instrumentation production.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КІТАР – комп’ютерно-інгегрованих технологій, автоматизації та робототехніки;

СКБД – система керування базами даних

ХНУРЕ – Харківський національний університет радіоелектроніки;

ШІ – штучний інтелект;

ERP – Enterprise resource planning;

MES – Manufacturing execution system.

ВСТУП

В 21-му столітті виробнича галузь отримала небачені раніше темпи розвитку. Масова цифровізація, збільшення темпів автоматизації різних частин технологічного процесу, підвищення продуктивності, та інші технологічні прориви постійно спостерігаються в наш час. З приходом Індустрії 4.0 та 5.0 підприємства почали активно наповнюватися цифровими приладами, які використовуються у всіх аспектах. Збільшення кількості, якості, та різноманіття датчиків для збору все більшого обсягу інформації, комп'ютери для обробки цих даних, хмарні технології для зберігання показників, та їх аналізу. Також досить популярним явищем є інтеграція різного програмного забезпечення, яка дає можливість працювати з різними аспектами підприємства завдяки комп'ютерам, мінімізуючи бюрократичні процеси, пов'язані з паперовою роботою, очікуванням необхідних підписів, тощо. Вже довгий час використовуються різні типи програмного забезпечення, які беруть на себе відповідальність за невелику підсистему підприємства, таку, як система контролю пропусків працівників, облік робочого персоналу, тощо.

Свого часу це був величезний технологічний прорив, адже більше немає необхідності наймати людину, що буде цілий день сидіти і записувати в зошит інформацію про працівників. Це дає змогу зробити дані прозорими, адже це є вкрай важливим фактором для безпеки роботі підприємства, це значно зменшує фактор людської помилки, оскільки людина, яка контролює це все вилучається, і дає змогу зробити дані мобільними, тобто значно полегшити до них доступ, зробити можливим їх отримувати онлайн, а не через зошит. І це лише один із прикладів, насправді, такі програми залучалися у велику купу аспектів.

Але з розвитком технологій і підприємств, прогрес прийшов до того, що необхідно створювати комплексне програмне забезпечення, здатне покривати одразу декілька аспектів роботи. Серед переваг варто виділити те, що одна

програма являє собою одну систему, де дані можуть ходити в різні її частини для різної обробки. Також неодмінно варто виділити те, що інтегрувати в підприємство значно легше буде комплексне рішення, адже його представляє одна й та сама компанія, і зазвичай, всі компоненти такого забезпечення виглядають схоже за стилем, а отже, і зменшує час на адаптацію персоналу. Але тут варто також виділити зворотній бік. Через те, що ведеться розробка такого потужного інструменту, вона може коштувати досить багато, що може бути нерентабельним для деяких підприємств.

Одним із таких інструментів є ERP системи, які використовуються для того, щоб автоматизувати процеси керування складом, замовленнями, поставками, постачальниками, та іншими суміжними аспектами, в залежності від того, яка саме система використовується.

Такого роду системи наразі є одними із найпотужніших інструментів для виробництв, оскільки дають змогу автоматизувати величезну купу процесів, спростити десятки годин роботи з документацією, відстежувати одразу всю інформацію, та керувати нею в залежності від необхідності. Наступним кроком є інтеграція ERP системи разом із системою для роботи з клієнтами, але це вже вкрай складна система, що потребує величезної робочої потужності для її розробки, великих фінансових витрат, та неодмінної необхідності в такій системі.

Таким чином, на меті кваліфікаційної роботи є покращення ефективності роботи персоналу приладобудівного виробництва шляхом розробки автоматизованої ERP системи для керування ресурсами складу на приладобудівному виробництві, враховуючи переваги та недоліки існуючих продуктів.

Задачі, що повинні бути виконані під час виконання виробничої практики, це проведення аналізу предметної області, того, як влаштована робота підприємств в Індустрії 4.0 та 5.0, потреби та вимоги таких підприємств, провести аналіз існуючого програмного забезпечення на ринку, та виділити їх

переваги, недоліки, та інші показники, які необхідні для розробки потужного програмного забезпечення, яке здатне замінити конкурентів. Також необхідно провести певний аналіз інструментів, з якими можна працювати задля досягнення кращих показників продуктивності додатку, та зручності його розробки, підтримки, та роботи з ним клієнтів.

Об’єкт розробки – процес керування приладобудівним виробництвом.

Предмет розробки – система керування ресурсами складу приладобудівного виробництва.

Кваліфікаційна робота виконана відповідно до [1-3]. Проведені дослідження відповідають цілям сталого розвитку (ЦСР): ЦСР 9, ЦСР 12.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні підприємства в епоху Індустрії 4.0 та 5.0

В 21-му столітті все більше і більше виробництв інтегрують та використовують нові виробничі моделі – Індустрія 4.0 та Індустрія 5.0, які стали сучасним трендом, та вкрай позитивно впливають на роботу підприємства.

Індустрія 4.0 – це четверта промислова революція, основним мотивом якої є максимальна інтеграція цифрових технологій у виробництво, задля покращення його роботи. Серед популярних технологій варто виділити Інтернет речей, що являє собою мережу між фізичними об’єктами, які взаємодіють між собою за допомогою інтернету, як для обміну інформацією, так і з більш конкретними цілями.

Поширеною практикою є інтеграція кіберфізичних систем, що дає можливість в певні фізичні процеси інтегрувати цифрове обчислення задля забезпечення кращої та продуктивнішої роботи.

Наразі дуже актуальним є сфера штучного інтелекту (ШІ), що активно впроваджується у більшість сфер діяльності людини, адже ШІ дає можливість максимально швидко отримувати знання, інструкції, підказки, та будь яку інформацію стосовно більшості галузей. Штучний інтелект також здатен до роботи з візуальними операціями, у випадку, якщо необхідно перевіряти виріб на наявність зовнішніх дефектів, навчена модель справляється дуже добре.

Для оптимізації виробничих процесів необхідно проводити аналіз показників, що збираються під час роботи, але, їх може бути дуже багато, адже самих показників багато, та обсяг даних сягає величезних значень. Тут використовується Big Data інструменти для аналітики процесів підприємства на основі зібраної інформації для оптимізації виробництва, оцінки його якості, та оцінки якості впливу на виробництво інших технологій.

З метою економії ресурсів для зберігання інформації, проведення аналітики та інших важких процесів Індустрія 4.0 передбачає використання хмарних технологій, що дають змогу орендувати обчислювальні потужності в їх постачальника.

Завдяки впровадженню Індустрії 4.0 виробництва стають розумними підприємствами, де автоматизованість доводиться до максимуму, разом із децентралізацією.

В цей ж час, Індустрія 5.0 – це наступне покоління, наступна революція у виробництві, яка вирішує проблеми минулих поколінь, та дає можливість виробництвам виходити на максимальну потужність. На рисунку 1.1 зазначено кожну промислову революцію, її лозунги, та особливості, які вона привнесла в розвиток виробництва та інших галузей промисловості.

Рисунок 1.1 – Візуалізація усіх промислових революцій

Перш за все, підприємства починають впроваджувати комплексні рішення, такі як ERP, CRM та інші, що спрямовані на вирішення комплексу проблем, та забезпечення оптимізації та прозорості різних процесів, що відбуваються на виробництвах. Значно посилюється інтеграція з ШІ, для забезпечення більшої

виробничої потужності, та збільшується кількість різних датчиків, сенсорів, задля того, щоб вони могли відстежувати все більше аспектів виробництва задля більш детального аналізу та пошуку нових рішень для покращення роботи підприємства.

Серед нових популярних практик можна виділити моделювання, створення цифрових двійників, які дають змогу на практиці перевіряти ті чи інші рішення, спрямовані на виробництво, задля того, щоб побачити їх симульований вплив на процеси.

Завдяки тому, що процес розвитку підприємств пришвидшується з кожною новою революцією, наразі, виробництва можуть думати не лише про те, як отримати максимальний прибуток, але й про інші аспекти, такі, як екологія, циркулярність економіки, кібербезпека, та соціальна відповідальність.

1.2 Особливості управління приладобудівним виробництвом в епоху Індустрії 4.0 та 5.0

Управління приладобудівним виробництвом в наш час поєднує в собі традиційні підходи разом з сучасними технологіями цифровізації, починаючи сенсорами Інтернету речей, закінчуючи модулями із залученням моделей штучного інтелекту, який аналізує тисячі показників у реальному часі, оброблює результати аналізу, та видає рішення щодо того чи іншого компоненту роботи приладобудівного виробництва. На рисунку 1.2 можна побачити абстрактну візуалізацію роботи керуючого на підприємстві.

Рисунок 1.2 – Робота керуючого підприємством

Сьогодні на підприємстві будь-яка деталь, така як мікропроцесор в автоматичному приводі або болт в корпусі приладу має свій датчик, який передає дані про температуру, вібрацію, тиск, електроживлення, або будь який інший технічний показник, в якому зацікавлено підприємство. Аналітичні платформи, використовуючи машинне навчання та різні алгоритми аналітики, в реальному часі обробляють отримані показники і виявляють проблемні ділянки, здатні спричинити простої або відмови. Завдяки цьому технічні спеціалісти в змозі запланувати технічне обслуговування різного обладнання до того, як воно вийде з ладу, значно зменшуючи фінансові витрати та простої виробництва.

На етапі проектування та випробувань найбільш поширеним інструментом управлінців і інженерів є цифрові двійники, які являють собою точні віртуальні копії складних механізмів і виробничих ліній, які вже існують, або плануються до впровадження на підприємстві. За допомогою цифрових двійників відбувається моделювання різних експлуатаційних умов, та середовищ, аналіз теплових і механічних навантажень, перевірка сценаріїв виходу з експлуатації й оптимізація графіків обслуговування забезпечення та пристроїв. Це дозволяє випробувати на міцність нові конструкції без виготовлення дорогих прототипів і значно скорочує час розробки та впровадження у використання інноваційних виробів.

Водночас впровадження ERP та MES систем дає можливість створити єдину платформу, де інтегровані системи для закупівлі сировини, логістики, виробничі замовлення, контроль якості та фінансові показники. У такому середовищі управлінці отримують цілісну картину в спеціалізованому інтерфейсі, в якому вони бачать, як зміни в графіку поставок вплинуть на завантаження обладнання, як відхилення стандартів якості на одній ділянці вплинуть на собівартість готової продукції, і де слід перерозподілити ресурси, щоб утримати виробничі показники в межах запланованих значень. Задля використання таких систем необхідно впроваджувати новий клас фахівців – інженери цифровізації, аналітики великих даних і кібербезпеки, які спільно з менеджерами виводять управління підприємством на якісно інший рівень [4].

На рисунку 1.3 зображено піраміду автоматизації процесів на приладобудівному виробництві.

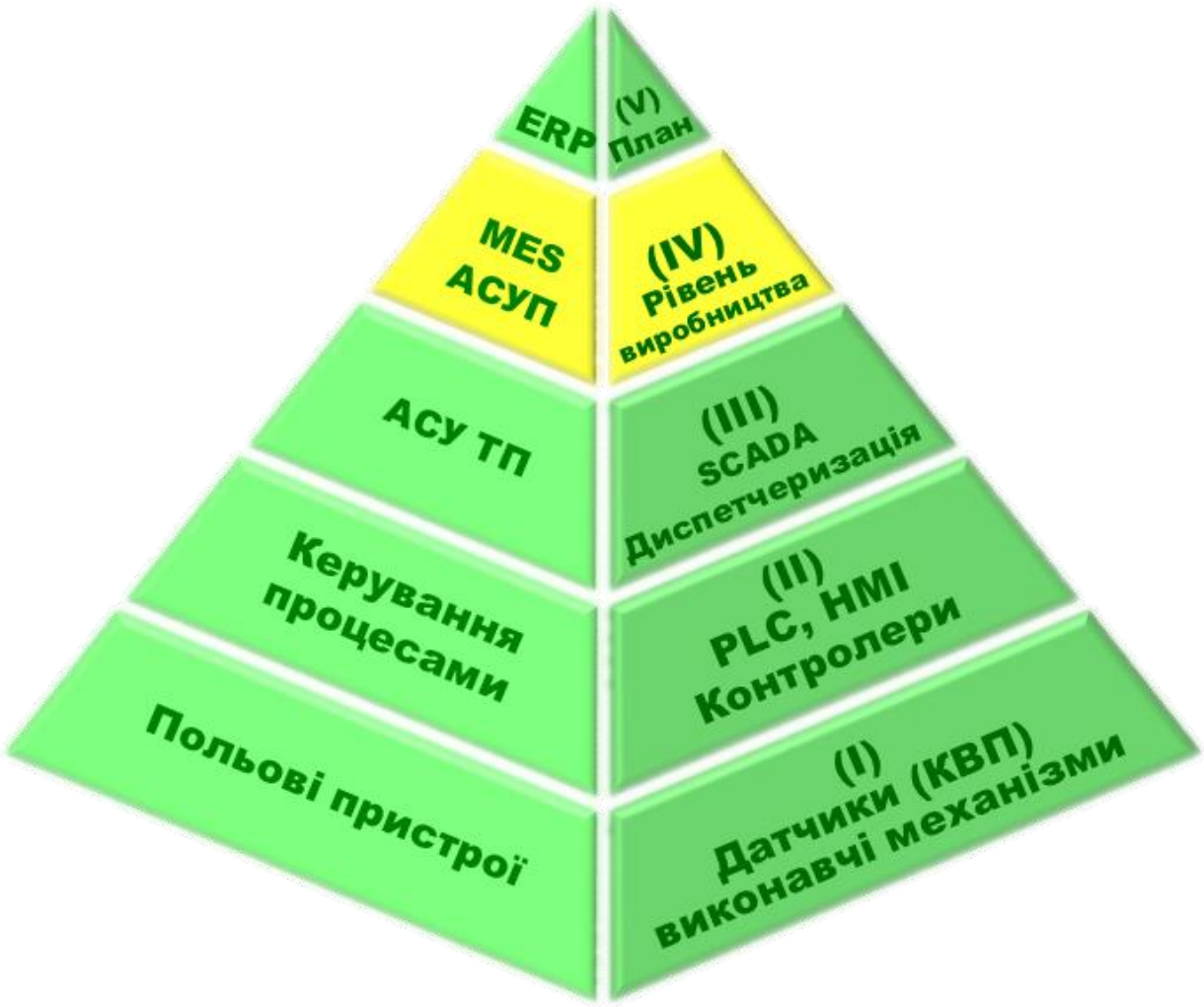


Рисунок 1.3 – Піраміда автоматизації технологічного процесу на приладобудівних виробництвах

Для того, щоб отримати максимальну користь від програмних засобів на підприємстві, необхідно правильно поєднувати обидва види систем. На рисунку 1.4 відображено поєднання цих двох видів систем, та їх зони відповідальності.

Рисунок 1.4 – Поєднання двох видів систем на підприємстві

Варто відзначити, що Індустрія 4.0, орієнтована, насамперед, на максимальну автоматизацію та оптимізацію механізмів, поступово доповнюється духом Індустрії 5.0, де в центрі уваги опиняється оператор. Сучасні приладобудівні виробництва впроваджують екзоскелети для зменшення навантаження на опорно-руховий апарат операторів та інших працівників, а системи доповненої реальності у вигляді окулярів чи планшетів забезпечують інструкції та вказівки під час виконання складних операцій. На рисунку 1.5 зображено використання доповненої реальності.

Рисунок 1.5 – Доповнена реальність

Такі технології не лише підвищують продуктивність і точність роботи, але й знижують ризик професійних травм, сприяють більш швидкій інтеграції нових працівників і зменшують вплив людського фактору на якість кінцевого виробу.

Не менш важливою складовою Індустрії 5.0 стає соціальна й екологічна відповідальність підприємств. Відмова від одноразових пластмас і перехід на екологічно чисті або перероблені матеріали, впровадження циклів виробництва, де відходи попереднього процесу стають сировиною для наступного, дають змогу істотно зменшити негативний вплив на екологію оточуючого середовища. Інтелектуальні системи моніторингу електрики та води контролюють споживання ресурсів і в реальному часі коригують роботу клімат-контролю, освітлення та вентиляції, щоб уникнути надлишкових витрат на підприємстві. На рисунку 1.6 зображено один з банерів, які створені для поширення ідеї створення екологічно чистих підприємств, їх переваги та недоліки [5].

Рисунок 1.6 – Створення екологічно чистого підприємства

Паралельно компанії розвивають програми підтримки працівників, такі як створення комфортних зон відпочинку з ергономічними меблями й тихими вентиляційними системами, впровадження гнучких графіків роботи та створення гібридного формату роботи шляхом поєднання очного формату із віддаленими консультаціями через системи для онлайн зустрічей.

Сучасний підхід до управління приладобудівним виробництвом полягає в поєднанні цифрових технологій Індустрії 4.0 і ідей Індустрії 5.0. У єдиній екосистемі, що поєднує хмарні та фізичні платформи, великі дані, штучний інтелект і оператора, вирішуються різні завдання від прогнозування попиту і планування виробництва до забезпечення якості й екологічної безпеки для навколишнього середовища. Саме такий режим роботи підприємства дозволяє як підвищити ефективність виробництва та знизити витрати, так і зробити підприємство привабливим роботодавцем, рушієм інновацій і відповідальним членом суспільства, здатним утримувати конкурентоспроможність у довгостроковій перспективі.

1.3 Концепція ERP систем, та їх роль в оптимізації процесів на підприємствах

ERP система являє собою корпоративну інформаційну систему, яка необхідна для того, щоб автоматизувати певні процеси, що пов’язані з обліком та керуванням. Розроблюване програмне забезпечення призначено для роботи на приладобудівному виробництві, але розроблене таким чином, що для зміни русла діяльності треба витратити мінімум часу.

Зазвичай, ERP системи допомагають оптимізувати процеси керування фінансами, керування виробництвом, керування запасами, постачанням зазначених запасів, керування реалізацією товарів, роботи із замовленнями, замовниками, постачальниками, робітниками та іншим персоналом, тощо.

Однією з найбільших переваг ERP систем є те, що одне програмне забезпечення замінює безліч програм під кожен відділ підприємства, що дає змогу значно полегшити обслуговування таких систем, та проведення робіт облікового характеру, адже вся інформація знаходиться в одному додатку.

Завдяки тому, що правильно побудована ERP система має працювати за схожими принципами у різних своїх блоках, таку конструкцію легше зробити безпечною для підприємства. Готова до роботи система має бути безпечною, має розмежувати інформацію, та доступ до неї для того, щоб запобігти більшості загроз, як ззовні, так і всередині.

До зовнішніх загроз можна віднести спроби окремих осіб, чи підприємств конкурентів отримати доступ до чутливої конфіденційної інформації, та використати її для досягнення своєї мети.

Внутрішніми загрозами прийнято вважати корупцію, і різні схеми, що мають в собі протиправні дії, що неодмінно негативно впливають на роботу підприємства, його результати [6].

Також варто зазначити, що часто ERP системи інтегруються разом з іншими системами, серед яких зазвичай є система контролю якості, системи управління відносинами з клієнтами, системи для впровадження додаткової безпеки, тощо.

Поєднання усіх цих систем разом дає змогу максимально задовольнити вимоги підприємства, витративши мінімум часу, грошей, та персоналу на впровадження і забезпечення роботи систем автоматизації.

Але варто також відзначити, що ERP системи мають перелік недоліків, які також необхідно враховувати перед впровадженням на підприємстві.

Найважливішою проблемою є рентабельність системи. Для деяких підприємств впровадження потужної ERP системи буде занадто коштовним. Також варто врахувати, що необхідно витратити кошти на навчання персоналу по роботі з системою, та на адаптацію підприємства до роботи з такими системами автоматизації. Також, обов’язково треба провести аналіз сумісності наявної документації, та наявних процесів із ERP системами, адже деякі специфічні виробництва можуть зіткнутися з тим, що інтегрувати систему дуже складно, або навіть неможливо.

Коректна система має бути дуже добре протестована, адже у випадку помилок під час розробки, підприємство ризикує отримати великі втрати та проблеми, адже чутливі дані будуть оброблені неправильно, що може призвести до складнощів та проблем різних рівнів.

Варто також враховувати, що під час навчання персоналу необхідно зробити так, щоб усі відділи підприємства працювали в системі однаково, за однаковими методами та алгоритмами, адже у випадку конфлікту різних даних можуть виникати помилки, які можуть бути критичними для підприємства.

На рисунку 1.7 зображено візуалізацію функціоналу класичної ERP системи.

Рисунок 1.7 – Функціонал класичної ERP системи

Виходячи з наведеного вище рисунку 1.7 можна побачити, що ERP – це багатомодульна, багатофункціональна система, яка покриває велику кількість модулів управління підприємством, та є невід'ємним інструментом для сучасного підприємства.

1.4 Історія ERP систем

Історія ERP почалася ще в перші десятиліття XX століття [7], коли великі виробничі підприємства почали впроваджувати стандартизовані підходи до обліку матеріалів, трудових ресурсів і виробничих процесів. На той час була відсутня будь яка автоматизація, тому всі процеси відбувалися з використанням паперових карток і механічних калькуляторів. Це був етап, коли управлінці

прагнули стандартизувати виробництво, запобігати надлишкам і зменшувати втрати на виробництво товару. Але зростання обсягів замовлень та збільшення вибору продукції підвищили й вимоги до гнучкості й точності обліку.

У 1960-х роках ситуація почала змінюватися з появою перших комп'ютеризованих систем MRP, які працювали на мейнфреймах і дозволяли планувати потреби у сировині, беручи за основу виробничі графіки. Такі системи, як IBM MAPICS, стали першими в цю епоху. Вони були орієнтовані виключно на виробничі підприємства, не враховуючи інші аспекти бізнесу через низький рівень технологічності тих часів. Проте це був радикальний крок у розвитку промисловості, тому що аналітика почала витісняти інтуїцію, а цифрові дані перемагали паперові таблиці.

У наступні два десятиліття концепція MRP розширилась до MRP II, яка вже охоплювала не лише сировину, а й управління виробничими потужностями, доступними в той час, планування робочих змін, витрат, логістики, персоналу. Це стало фундаментом для подальшого переходу до повноцінних інтегрованих систем управління підприємством. Паралельно із розвитком обчислювальної техніки й розподілених систем з'являлися нові можливості зберігання, обробки і візуалізації інформації, що дозволило MRP II стати все більш популярним і необхідним у великих корпораціях. Саме в цей момент система стає найближчою до розроблюваної у цій роботі системи типу ERP [8].

У 1990-х роках на тлі дуже стрімкого загального технологічного прогресу та поширення клієнт-серверної архітектури виникло поняття ERP – Enterprise Resource Planning. Це були системи зовсім нового типу, де замість розрізнених систем для кожного департаменту з'явилася єдина інтегрована платформа, яка охоплювала всі ключові бізнес-процеси компанії - фінанси, виробництво, відділ кадрів, логістику, продажі, маркетинг, закупівлі та інші. ERP-системи зменшили роздрібленість та дублювання інформації, дозволили отримувати повну картину

діяльності компанії в режимі реального часу. У цей період почали виникати системи, які зараз вважаються класикою, та прикладом для новітніх систем

Наприкінці 1990-х ERP-системи почали інтегрувати різні CRM-модулі, аналітику та інтерфейси, готуючись до майбутнього технологічного прогресу. Водночас, перехід до глобалізованих ринків вимагав мультивалютної і мультимовної підтримки. Зросла складність розробки систем, з'явилася потреба у спеціалізованих консультантах, бізнес-аналітиках, інтеграторах. Водночас стало зрозуміло, що великі обсяги системи не завжди є гарним рішенням, адже дедалі більше проектів завершувалося невдачею або коштували більше, ніж давали користі. Це підштовхнуло розвиток більш гнучких, модульних рішень.

У 2000-х роках настала ера ERP II – концепції, яку запропонувала компанія Gartner. ERP II не обмежувалась межами підприємства, а відкривала доступ до систем партнерам, постачальникам, клієнтам. Почали впроваджуватися різні сучасні механізми роботи, інтеграція з електронною комерцією, мобільними додатками та зовнішніми сервісами. На цьому етапі ERP-системи почали грати роль не лише інструменту управління, а й платформи для цифрової взаємодії у бізнес-екосистемах.

Паралельно почався перехід до хмарних технологій – спочатку як гібридні рішення, а потім і повноцінні платформи. NetSuite стала однією з перших успішних хмарних ERP, а Salesforce почала демонструвати, що хмара – це технологія майбутнього. SAP випустив S/4HANA як відповідь на нові виклики, Oracle просував свою Oracle Cloud ERP. Хмарні технології дали можливість малому й середньому бізнесу скористатися ERP, що раніше було доступно лише корпораціям, які мали достатньо великий бюджет на подібні рішення. На рисунку 1.8 візуалізовано процес розвитку ERP систем [9].

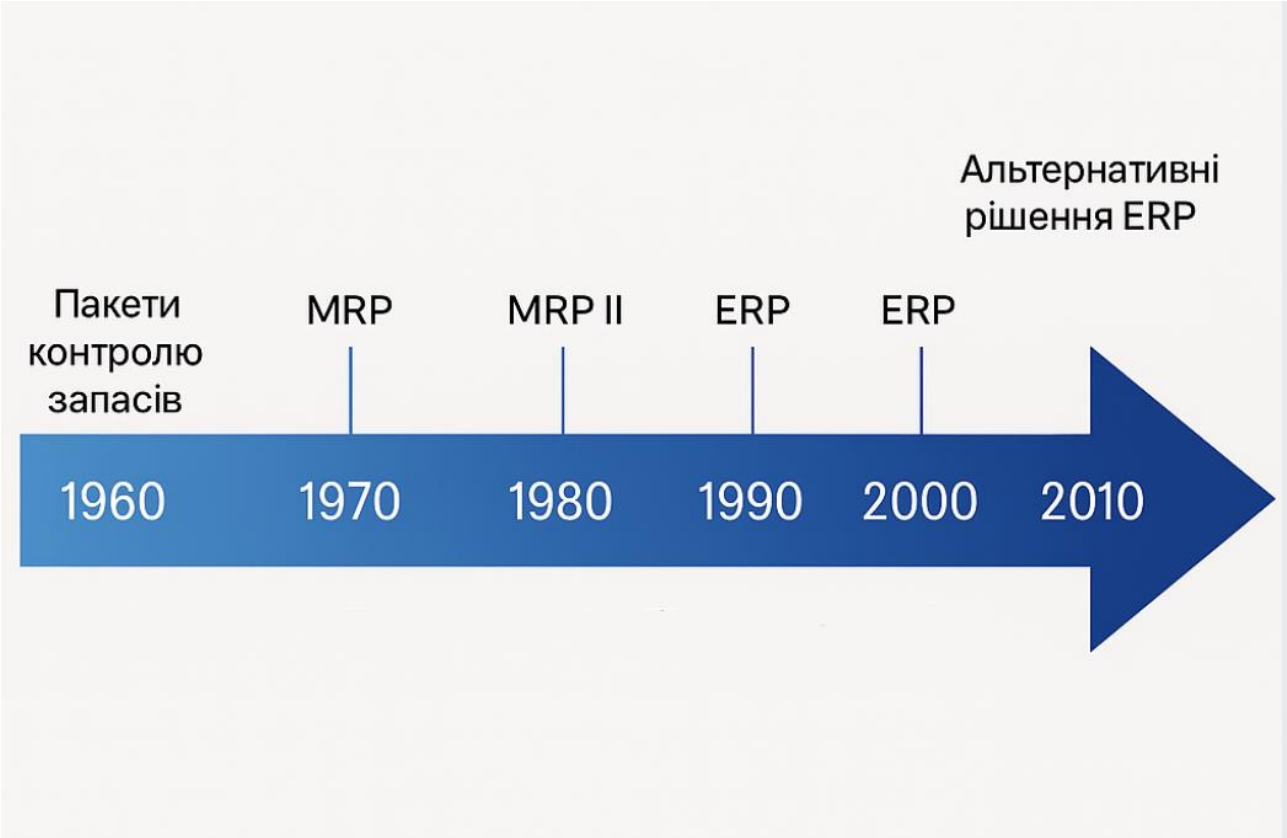


Рисунок 1.8 – Процес розвитку ERP систем із плином часу

Сьогодні ERP – це не просто система обліку. Це розумна, гнучка платформа, яка здатна інтегрувати дані з різних пристроїв, аналізувати тренди за допомогою штучного інтелекту, передбачати ризики і автоматизувати рішення. Вона може масштабуватись в реальному часі, адаптуватися до змін, забезпечувати гнучке ціноутворення, персоналізовані пропозиції й машинне навчання.

1.5 Аналіз існуючих ERP систем

Для того, щоб спроектувати та розробити таку ERP систему, що буде відповідати вимогам до такого роду програмного забезпечення, та мати переваги перед іншими системами, нам необхідно провести аналіз цих самих вимог, та вже існуючих рішень, які створені для вирішення цих проблем. Серед

найпопулярніших систем можна виділити Odoo, SAP S/4HANA та Microsoft Dynamics 365 Supply Chain Management. На рисунку 1.9 зазначено зовнішній вигляд інтерфейсу програмного забезпечення Odoo.

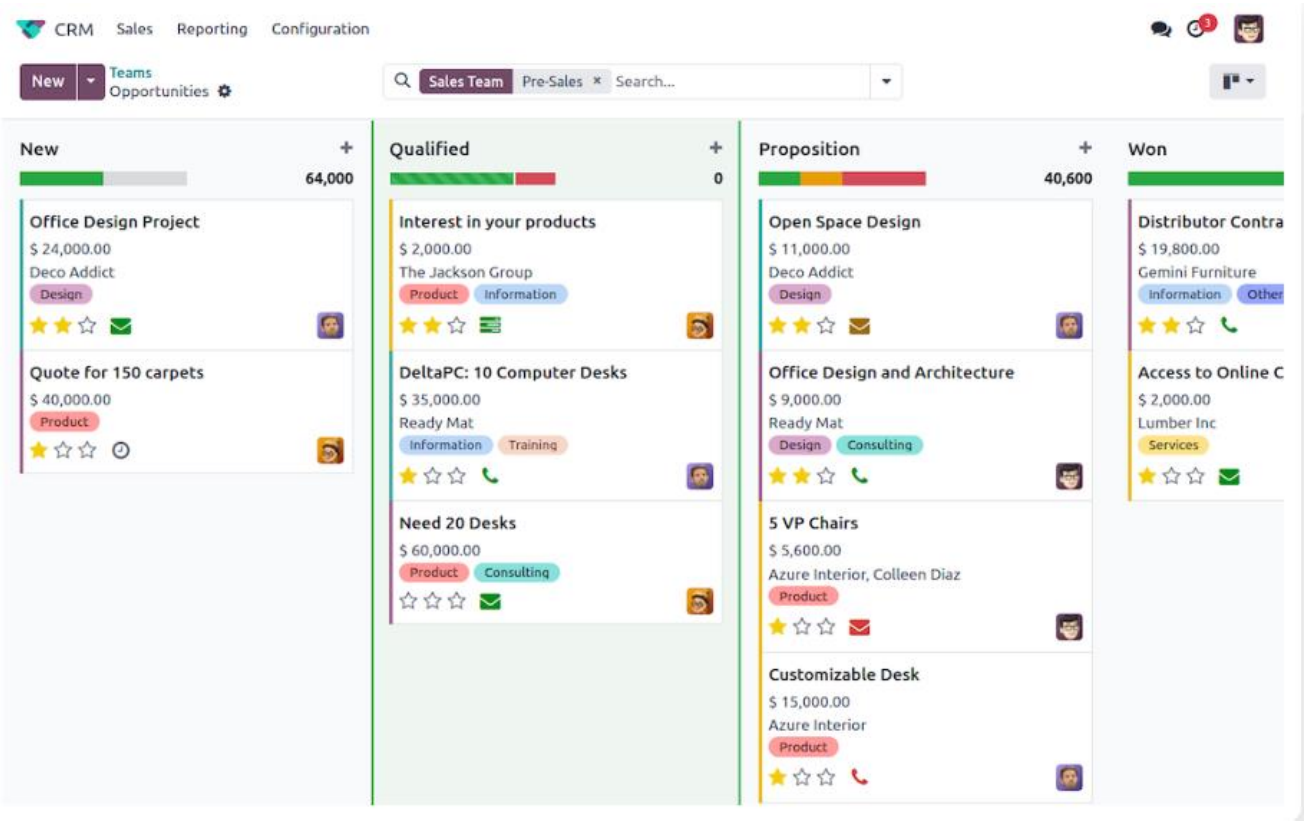


Рисунок 1.9 – Інтерфейс ERP системи Odoo

Аналізуючи рисунок 1.9, можна зазначити, що попри різний функціонал, дизайн є занадто навантажений, що може викликати проблеми в працівників, які просто не зрозуміють, як працювати з інтегрованою системою. Наступною необхідно розглянути програмне рішення від SAP S/4HANA. На рисунку 1.10 зазначено зовнішній вигляд цього рішення.

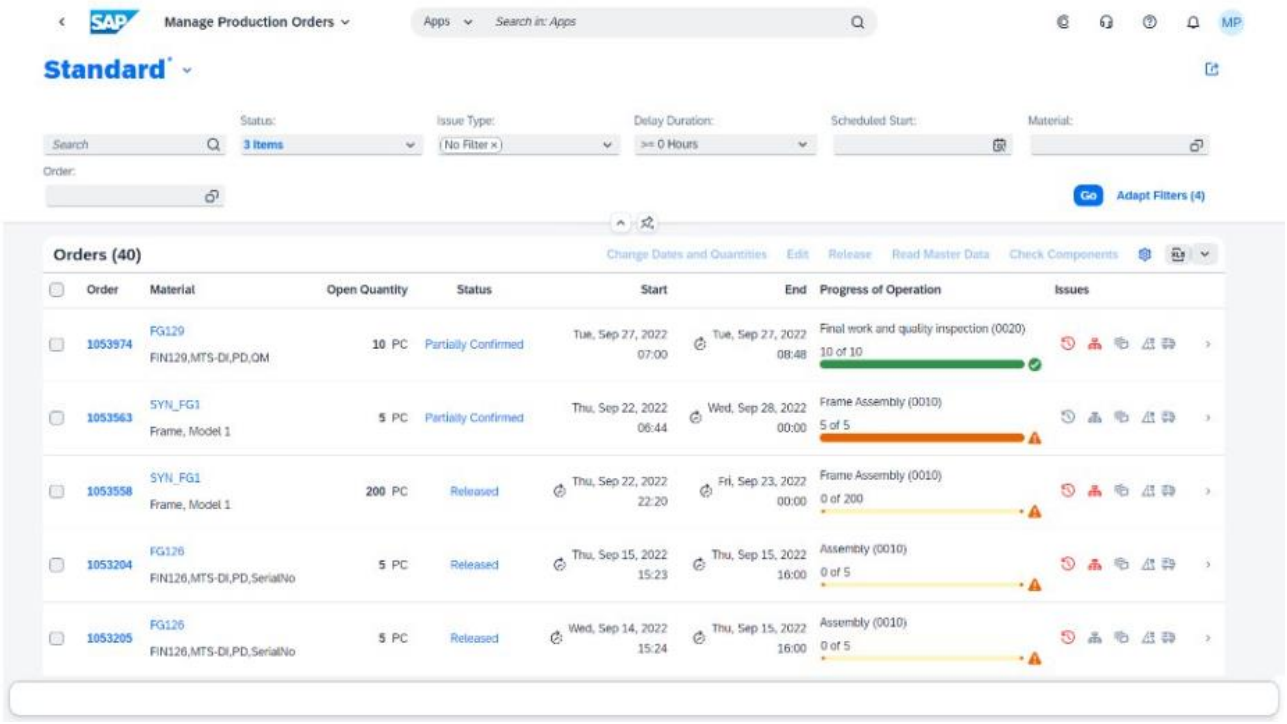


Рисунок 1.10 – Програмне рішення SAP S/4HANA

Тут також варто відзначити, що дизайн є досить складним, хоча і очевидно спрощений для покращення роботи з програмним забезпеченням. Для досягнення мети максимального полегшення інтеграції ERP системи у підприємство, необхідно продовжувати спрощувати інтерфейс, та зробити більш задовільними умови співпраці власників системи та підприємства. На рисунку 1.11 розглянуто третього конкурента – Microsoft Dynamics 365 від технологічного гіганту Microsoft.

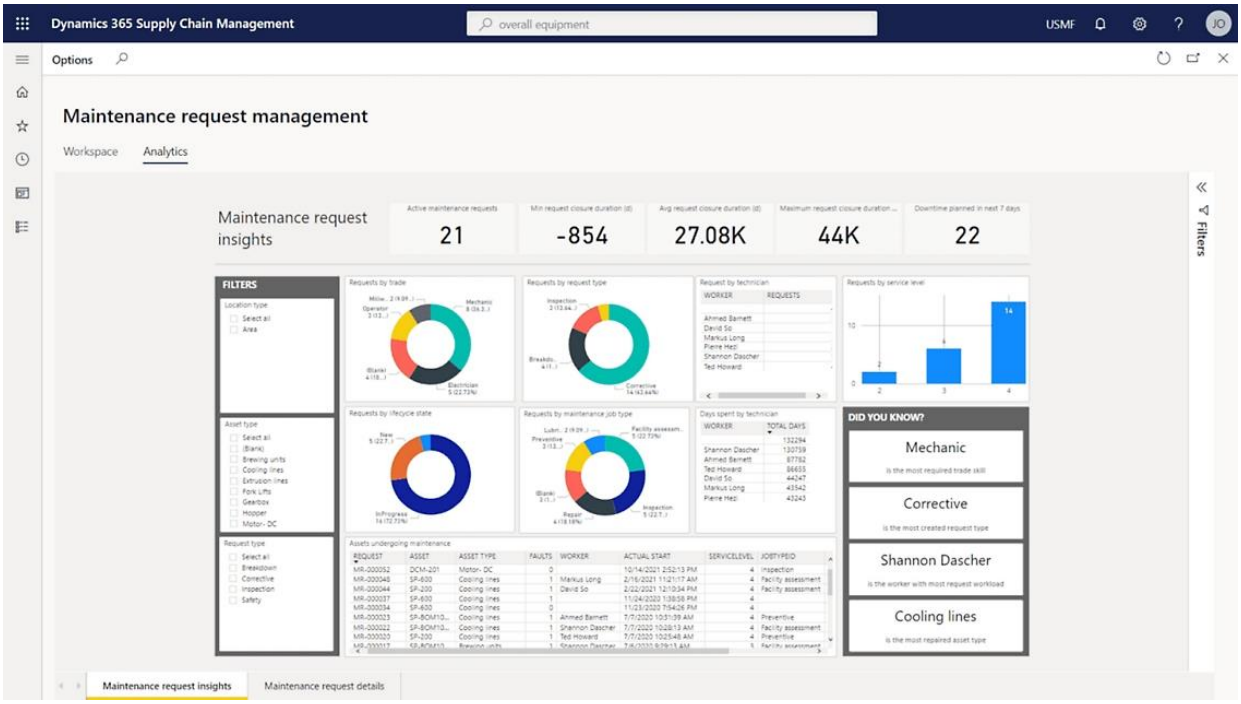


Рисунок 1.11 – ERP система від Microsoft

Як можна побачити на рисунку вище, ця система є найбільш перенавантаженою, але і функціонал вона має ширший. Проблема цієї системи в тому, що відсутній баланс між зрозумілістю та потужністю. Оскільки Microsoft – технологічний гігант, їх рішення зроблені для підприємств середнього та преміум сегменту, де є змога профінансувати цілий курс для персоналу щодо використання системи, та витратити величезні кошти на її інтеграцію, що робить систему не найкращим рішенням для обмежених бюджетом виробництв.

Виходячи з результатів детального аналізу, зазначеного в публікації, можна виділити дві основні проблеми ERP систем, а саме складність їх впровадження та високу ціну. Фінансова проблема є головною, адже більшість підприємств просто не мають можливості забезпечити роботу з таким програмним забезпеченням. Для вирішення цієї проблеми необхідно розробити доступну систему, яка буде коштувати значно нижче за аналоги. Для визначення ціни необхідно проаналізувати кошторис існуючого забезпечення, оцінити вартість розробки та підтримки програмного забезпечення, а також умови

експлуатації. Другою проблемою є складність інтеграції ERP системи у виробництво, та навчання персоналу по роботі з нею. Цю проблему необхідно вирішити створивши максимально зрозумілий інтерфейс, який дозволить мінімізувати процес адаптації працівників. В розроблюваному програмному забезпеченні передбачено мінімалістичний інтерфейс для всіх частин системи, виконаний в одному стилі, інтуїтивно зрозумілий, та облаштований різними підказками, що дасть змогу працювати за схожим сценарієм, та пришвидшити інтеграцію системи у роботу підприємства.

Прийнявши такі рішення, можна розробити дійсно потрібне ринку програмне забезпечення, що буде краще за конкурентів, та доступним для українського виробництва [10].

2 ВИБІР І ОБҐРУНТУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ERP СИСТЕМИ

2.1 Вибір мови програмування для створення програмного забезпечення

Під час проходження навчання було оглянуто на різних рівнях досить велику кількість мов програмування, серед яких є C++, Python, PHP, та інші. На основі отриманих навичок і досвіду, було прийнято рішення використовувати саме Python [11] для розробки ERP системи, адже:

– Python є мовою програмування, що широко використовується у всьому світі;

– Python розроблено таким чином, що з ним легко працювати, його синтаксис є зрозумілим;

– Python є відкритим для покращення іншими розробниками, і має величезну базу сторонніх бібліотек, фреймворків, та інструментів, які дають можливість проводити розробку максимально швидко, ефективно, та створити правильну архітектуру програмного забезпечення, що дає можливість в майбутньому без проблем додавати новий функціонал, чи змінювати вже існуючий, згідно вимогам до системи.

Варто також зазначити, що Python за своєю сутністю поєднує в собі зручність, та досить непогану швидкість, адже має в собі базу з мови програмування C, та можливість вільно працювати з бібліотеками, що написані на мовах цього роду.

Також, через велику поширеність цієї мови програмування, вона отримала ще одну перевагу – це стандартизація. PEP8 – перелік різного роду правил, рекомендацій, слідуючи яким, написаний код буде вважатися чистим, логічним, та читабельним, що значно покращує роботу з чужим кодом, або розробку в команді. Це також стосується і сторонніх бібліотек, які, якщо написані згідно

стандартам, можна спокійно модифікувати під свої потреби без необхідності вигадувати повністю нове програмне рішення.

Завдяки тому, що Python є мовою, що інтерпретується, а не компілюється, його код виконується рядок за рядком, що дає можливість легко відстежувати потік даних в програмі, та у разі помилки, буде одразу зрозуміло, де вона виникає, на що зверне увагу сам Python, та запропонує варіант рішення, що неодмінно позитивно впливає на розробку.

Не менш важливою перевагою мови програмування Python є те, що в ній використовується динамічна типізація, тобто, розробнику не обов'язково одразу вказувати тип змінної, яку він створює, і за потреби, змінювати його, працюючи з однією і тою самою змінною.

Однією із головних особливостей, та перевагою перед іншими мовами є автоматичний збір «сміття» (garbage collector), який завдяки внутрішнім складним алгоритмам дає можливість сфокусуватися на вирішенні проблеми, а не на відстеженні кожного біту, що використовує програмне забезпечення в ході свого функціонування.

2.2 Фреймворк Django

Фреймворк Django – це безкоштовна бібліотека [12] для мови програмування Python, яка дає можливість створювати веб-додатки, розроблюючи як серверну частину, так і візуальну також. Завдяки цьому фреймворку є можливість значно скоротити час розробки проекту, оскільки він передбачає в собі базові шаблони для різних компонентів, що присутні в сучасному веб додатку.

Завдяки вбудованим інструментам для роботою з базами даних, наявності підтримки роботи з візуалізацією сайтів за допомогою HTML, CSS, JS, та іншим компонентам було побудовано повноцінний сучасний веб додаток, що відповідає

усім вимогам для типового програмного забезпечення.

Варто відзначити, що у Django передбачена вбудована базова панель для адміністратора, що полегшує керування проектом, даними, персоналом, та будь-якими показниками, що присутні в проекті. Хоч панель і необхідно дороблювати, щоб працювати з створеними у проекті моделями, певний базовий шаблон є одразу, що вкрай позитивно впливає на розробку, її продуктивність, та значно її полегшує. Панель адміністратора дає можливість легко керувати усіма користувачами, надавати їм різні групи (робітник, адміністратор, бухгалтер), дозволи, змінювати логін, пароль, відновлювати його у разі виникнення проблем з авторизацією, чи створювати, або видаляти користувачів у випадку кадрових змін. Така сама можливість передбачена для сутностей самого проекту (ресурси, поставки, постачальники, тощо), можна будь яку сутність видалити, створити, або редагувати.

Django використовує архітектуру Model View Template, яка також передбачає легку розробку, через її зрозумілість, що дає можливість побудувати необхідний додаток за мінімальний час. Особливості та деталі архітектурного патерну будуть наведені в наступних розділах роботи.

Для взаємодії з базою даних фреймворк використовує власний інструмент – Object Relational Mapping, який дозволяє взаємодіяти з базами даних завдяки об’єктно-орієнтованій парадигмі програмування, використовуючи класи, атрибути та методи. Це дає можливість зменшити час на розробку, адже більшість запитів до бази даних можна зробити через код на Python, замість розробки більш ємних запитів на мові SQL.

2.3 Система керування базами даних PostgreSQL

В якості системи керування базами даних було обрано саме PostgreSQL, попри те, що Django має вбудовану інтеграцію з іншим засобом, а саме SQLite,

який має свої недоліки, та переваги.

Фундаментальною відмінністю між цими системами керування базами даних (СКБД) є те, що за архітектурою PostgreSQL [13] є кліент-серверним додатком, що потребує виділення окремого серверного процесу для роботи, в той час, як SQLite – це файл, в якому зберігається інформація. Зазвичай під кліент-серверні бази даних виділяється окремий сервер, що є менш вигідним з фінансової точки зору, але дає велику кількість переваг, що описані нижче.

Postgres дає можливість працювати з величезною кількістю типів даних, що зберігаються в таблицях бази даних, що зумовлено потужнішим та функціональнішим рушієм, через який, відповідно, і була обрана кліент-серверна архітектура. В цей час, простіша СКБД підтримує лише 5 основних типів даних для роботи з даними, яких не є достатньо для більшості проектів.

Одним з найважливіших аспектів вибору СКБД є здатність до масштабованості. PostgreSQL підтримує реплікацію, кластеризацію, що дає можливість горизонтально масштабувати проект, виділяючи нові сервера для бази даних. Згідно з показниками, підтримується обсяг даних до петабайту, чого буде більш, ніж достатньо для ERP системи. SQLite не розрахована на великий обсяг даних, і здатна містити в собі до 100 гігабайтів даних, погано працюючи з великим трафіком, що може стати проблемою при подальшій роботі підприємства, його розширення, та довготривалої роботи.

Не менш важливим є продуктивність баз даних. Оскільки SQLite передбачає собою лише файл, він підтримує лише один запис одночасно, блокуючи файл для інших операцій. В цей час, клієнт-серверна архітектура PostgreSQL дає можливість виконувати тисячі транзакцій одночасно, що є ідеальним для нашого додатку, в якому передбачається можливість багатьом користувачам з різними ролями користуватися додатком одночасно.

Оскільки розроблювана система використовується на приладобудівному виробництві, дані, що зберігаються в БД є приватними, тому є чітка задача

забезпечити безпеку даних. SQLite навіть не має авторизації, та, оскільки дані зберігаються в файлі, який можна видалити, та втратити всі дані. PostgreSQL передбачає багаторівневий захист даних, не даючи можливості використовувати його без авторизації. Також використовується шифрування даних, та різні системи захисту. Також, завдяки тому, що дані зберігаються на окремому сервері, їх неможливо видалити випадково.

Враховуючи перелічені аспекти, в результаті ретельного аналізу показників, саме PostgreSQL було обрано в якості рішення для збереження даних.

2.4 Docker та Nginx для забезпечення технічної стабільності та зручності

Варто відзначити, що програмне забезпечення має бути зручним не лише у використанні, а й в обслуговуванні. На основі цієї потреби було вирішено, що використання Docker [14] є необхідним. Docker – це спеціалізована платформа, яка дає можливість розгорнути свій додаток в окремих ізольованих «контейнерах». В середині контейнеру знаходиться ізольований фрагмент середовища додатку, разом з його залежностями, та усім необхідним забезпеченням для його функціонування.

Контейнеризація значно полегшує процес розгортання програмного забезпечення, адже дає можливість запустити програму будь де, та в будь якому середовищі, адже всередині є все необхідне для роботи проекту. Провівши певні налаштування, було отримано середовище, яке дає можливість переносити проект на інші сервери за пару операцій, та надає можливість масштабуватися, адже кількість контейнерів обмежена лише ресурсами машини, на якій він працює.

Для того, щоб значно покращити роботу системи разом з контейнерами, необхідно використати Nginx [15], що являє собою окремий веб сервер, який відповідає за розподіл навантаження між системами в програмному забезпеченні.

Цей інструмент є обов’язковим у разі, якщо необхідно масштабувати проект.

2.5 Візуалізація за допомогою HTML, CSS та JavaScript

Після аналізу різних методів візуалізації даних, було обрано декілька варіантів, це Python, HTML з CSS та JavaScript, C#. Оскільки перший та третій варіанти передбачають створення додатку, який необхідно встановлювати на комп’ютер для роботи з програмним забезпеченням, було обрано другий варіант, що надає можливість працювати з браузера на будь-якому пристрої [16].

Це рішення дає можливість взаємодіяти з програмним забезпеченням на будь якому з пристроїв, де є доступ до інтернету, і що найбільш важливо, без будь якої прив’язки до платформи. Програмне забезпечення буде однаково працювати як на Windows, так і на Linux, чи Android або IOS. Це робить розроблюване програмне забезпечення універсальним та доступнішим.

Варто відзначити, що це рішення значно скорочує витрати часу на розробку програмного забезпечення, адже в залежності від виробництва, можуть бути встановлені різні системи, що може викликати необхідність розробки різних версій додатку.

Поєднання цих трьох компонентів дає можливість створити приємний, інтуїтивно зрозумілий дизайн, який в свою чергу, позитивно вплине на продуктивність підприємства, адже персонал буде зайнятий робочими задачами, а не навчанням по роботі з розроблюваним програмним забезпеченням.

2.6 Розробка діаграми станів для ERP системи

Дана діаграма надає можливість описати послідовності станів і переходів, які в сукупності дають можливість побачити поведінку модельованої системи.

Діаграма станів розроблюється за допомогою уніфікованої мови

моделювання UML, що є невід'ємною частиною уніфікованого процесу розробки забезпечення. UML створений для того, щоб надати можливість візуалізувати проектування, і тому, саме за його допомогою розроблена наведена на рисунку 2.1 діаграма станів для розроблюваного програмного забезпечення для вкладки робітника.

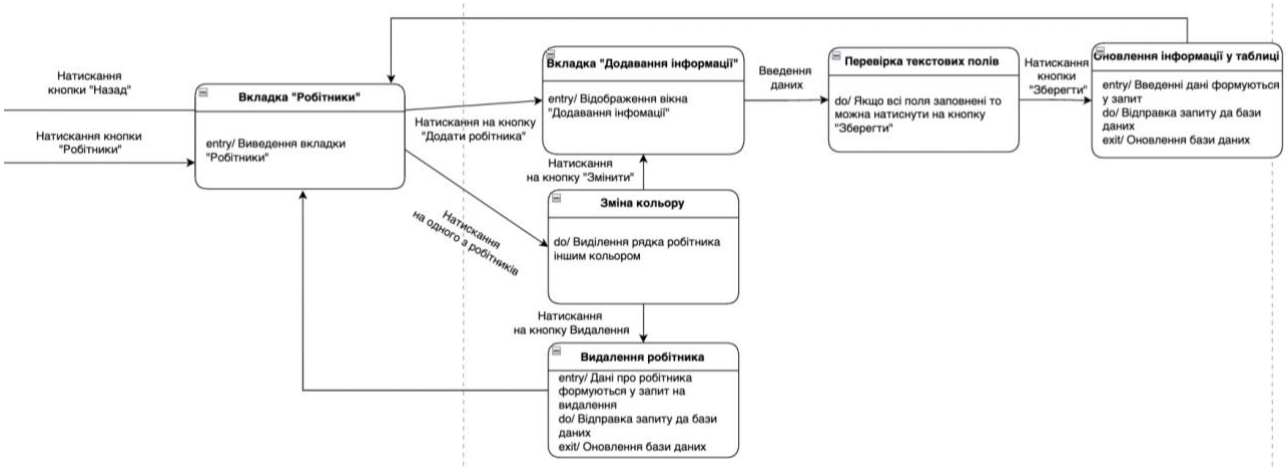


Рисунок 2.1 – Діаграма станів для розроблюваного програмного забезпечення для вкладки робітника

З наведеного вище зображення можна побачити, який функціонал, та яку взаємодію передбачає підсистема, що відповідає за вкладку робітників. В розроблюваному програмному забезпеченні передбачено функціонал перегляду, зміни, додавання та виділення необхідних компонентів, в даному випадку це робота з робітниками. Ми маємо можливість вести кадрову політику підприємства прямо в додатку, змінюючи посаду працівника, що в свою чергу впливає на його права доступу та можливості під час роботи програмного забезпечення.

На рисунку 2.2 наведено фрагмент діаграми станів для наступної вкладки товарів.

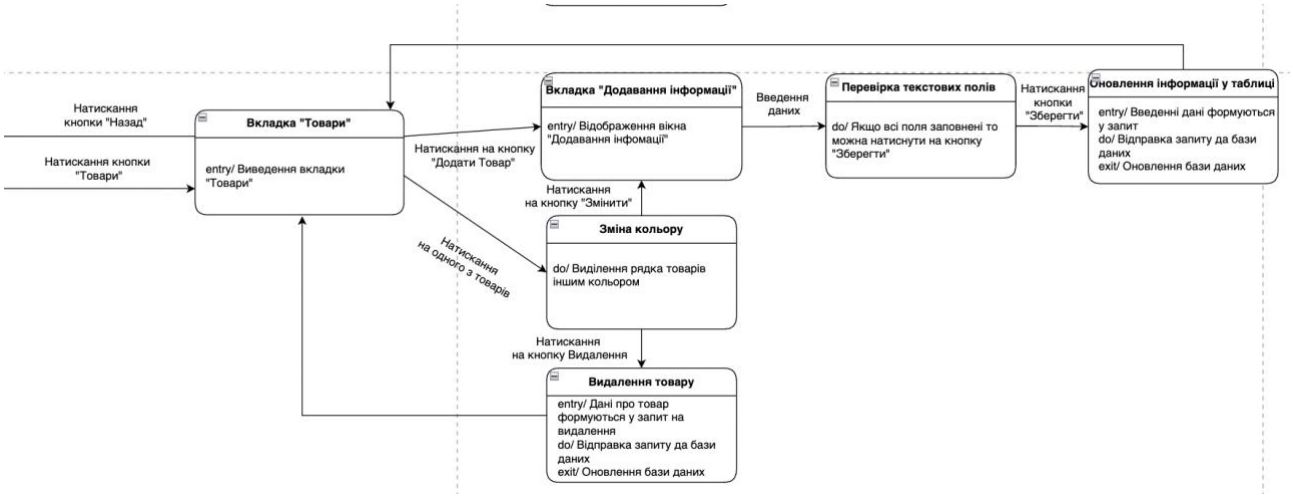


Рисунок 2.2 – Діаграма станів для розроблюваного програмного забезпечення для вкладки товарів

Варто зазначити, що, як було зазначено вище, усі вкладки виконані за схожою логікою, що дає можливість значно пришвидшити як розробку програмного забезпечення, так і полегшити його використання, адже усі підсистеми працюють за однаковим алгоритмом. На рисунку 2.3 зазначено аналогічний фрагмент діаграми для вкладки замовлень.

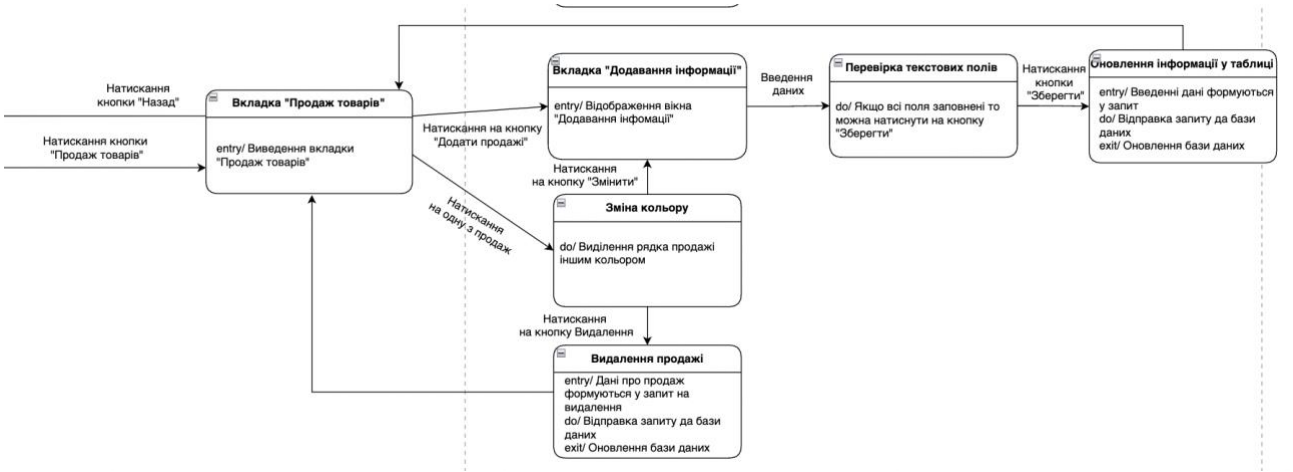


Рисунок 2.3 – Діаграма станів для розроблюваного програмного забезпечення для вкладки замовлень

Виходячи з рисунку 2.3 можна зазначити, що замовлення товарів виконується по схожому шаблону поведінки, наряду з іншими підсистемами. Для того, щоб оформити запит на покупку, указана інформація має пройти декілька рівнів перевірки.

Так само, доступні різні операції з замовленнями, їх обробка, зміна статусу, тощо. На рисунку 2.4 зазначено аналогічний фрагмент для вкладки ресурсів.

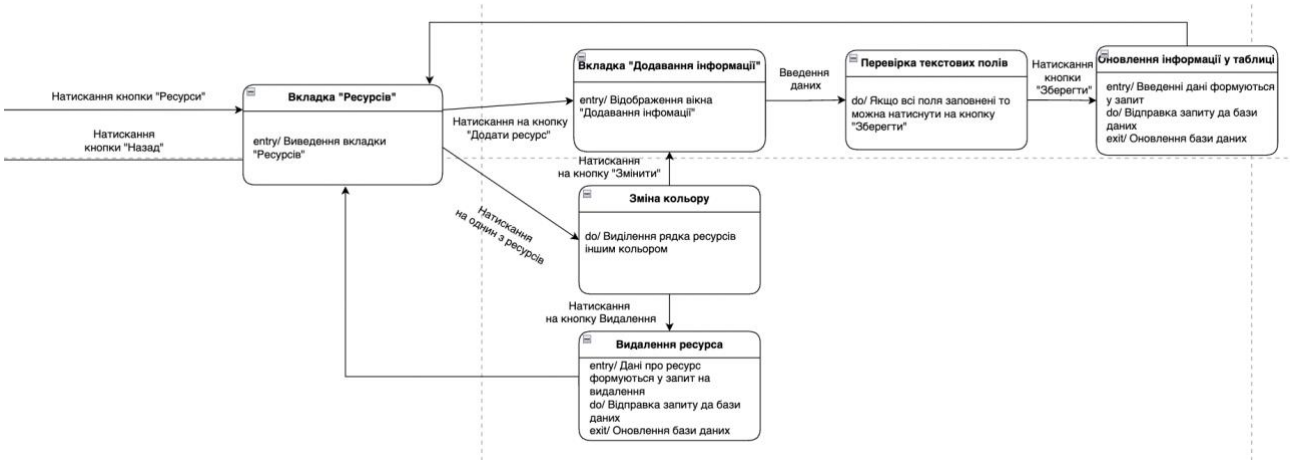


Рисунок 2.4 – Діаграма станів для розроблюваного програмного забезпечення для вкладки ресурсів

Виходячи з рисунку 2.4 можна зазначити, що зберігається однорідність підсистем, що позитивно впливає на розробку, та використання. На рисунку 2.5 зазначено фрагмент діаграми станів для наступної вкладки замовлень поставок ресурсів.

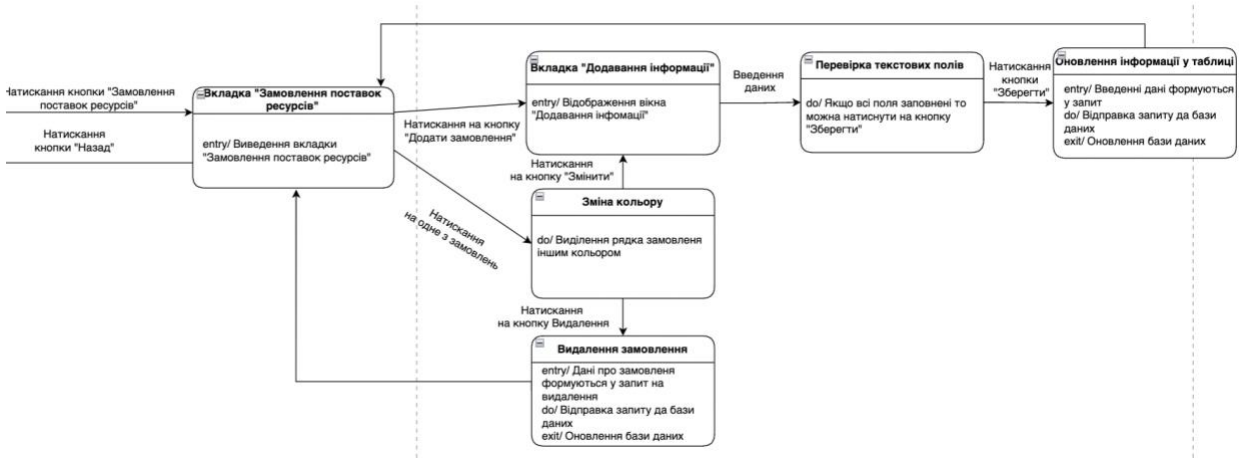


Рисунок 2.5 – Діаграма станів для розроблюваного програмного забезпечення для вкладки поставок

На рисунку 2.6 наведено фрагмент діаграми станів для вкладки матеріалів, що виконана в аналогічному стилі та містить аналогічну іншим фрагментам логіку, стани, та переходи.

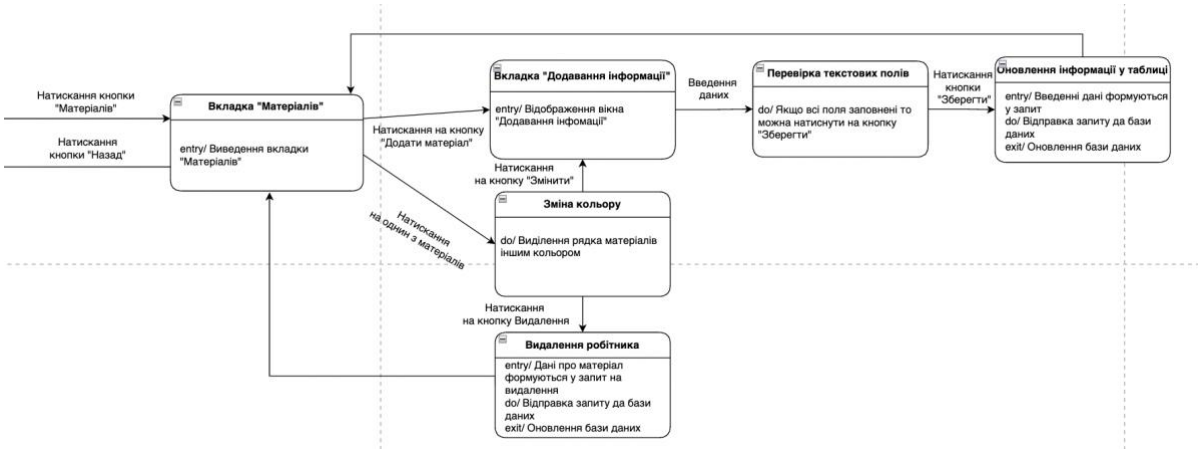


Рисунок 2.6 – Діаграма станів для розроблюваного програмного забезпечення для вкладки матеріалів

Наступною треба відзначити вкладку постачальників, яка, як і зазначено раніше, виконана за схожою логікою із минулими вкладками. Вкладку зазначено на рисунку 2.7.

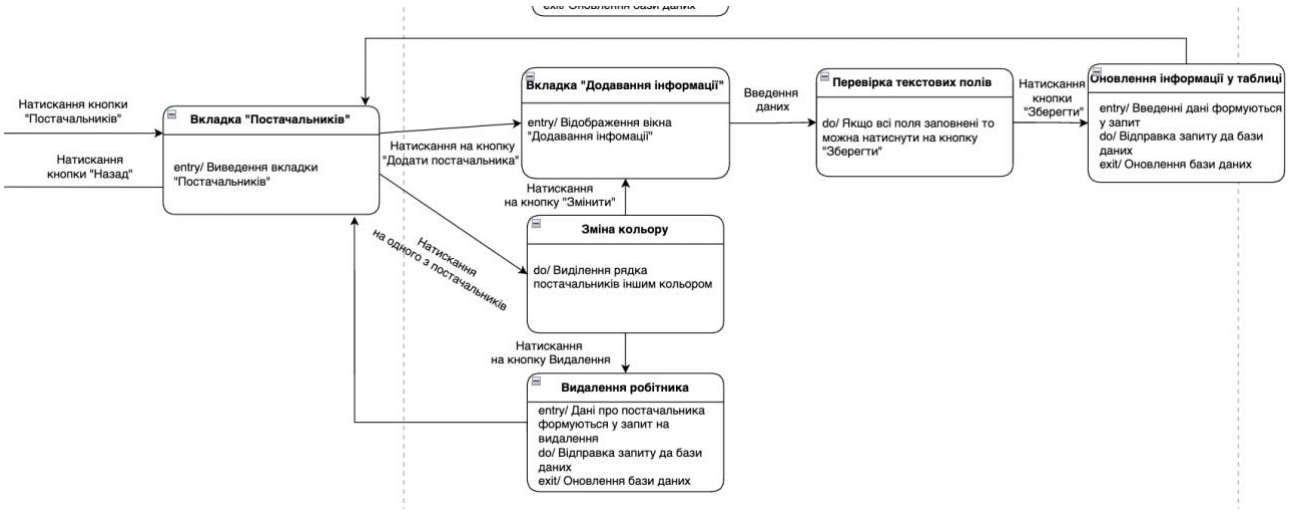


Рисунок 2.7 – Діаграма станів для розроблюваного програмного забезпечення для вкладки постачальників

Останньою вкладкою є система звітів, яка, в свою чергу, відрізняється від усіх інших, передбачає собою два процеси, це генерація звітності, та її збереження. Фрагмент діаграми зазначено на рисунку 2.8.

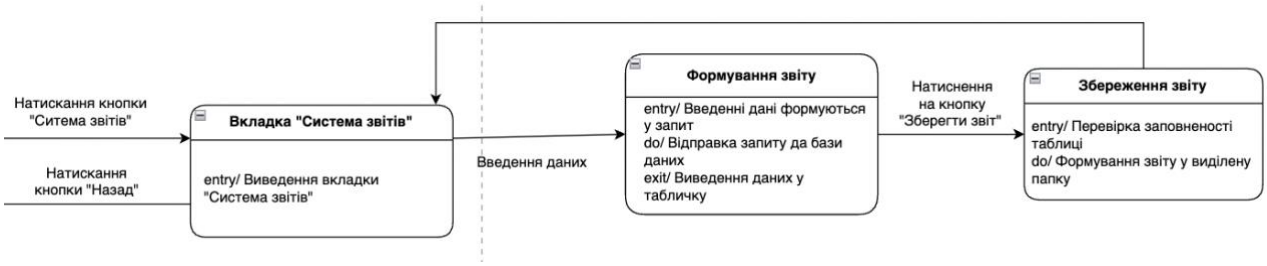


Рисунок 2.8 – Діаграма станів для розроблюваного програмного забезпечення для вкладки звітів

З наведених вище рисунків можна побачити, що програмне забезпечення має велику кількість різних вкладок, які, в свою чергу, є взаємопов’язаними підсистемами однієї системи. Виходячи з аналізу діаграми видно, що система має заходи безпеки, а саме авторизація для доступу до меню з усім іншим функціоналом. В діаграмі станів зазначено, що розроблювана ERP система

поділяється на вкладки робітників, товарів, замовлень товарів, ресурсів, замовлень поставок ресурсів, матеріалів, постачальників, та системи звітів.

Можна також побачити, що більшість підсистем спроектована в схожому стилі, що дає можливість інтуїтивно працювати із додатком, зменшуючи витрати часу персоналу, що позитивно впливає на продуктивність усього підприємства.

2.7 Розробка діаграми IDF0 для ERP системи

IDF0 передбачає собою методологію графічного опису бізнес процесів, та функціонального моделювання. Загальний вигляд системи через діаграму IDF0 наведено на рисунку 2.9.

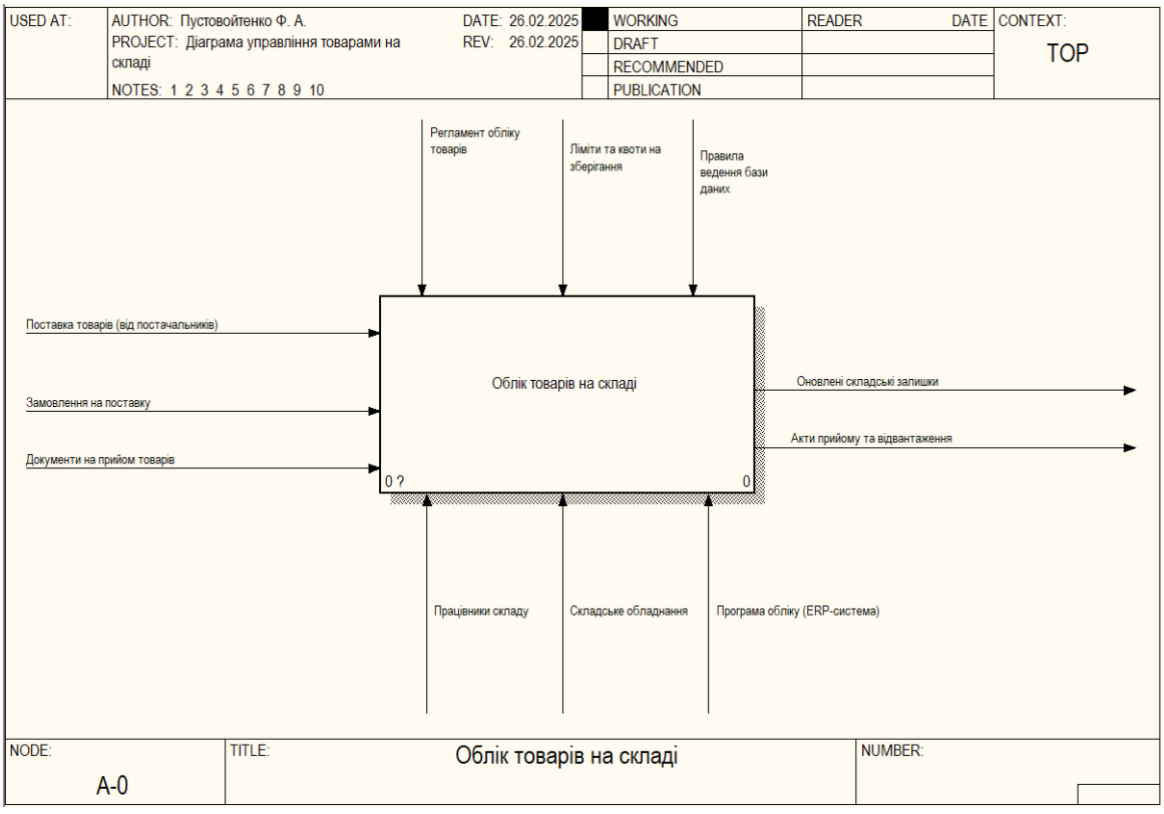


Рисунок 2.9 – Загальний вигляд діаграми IDF0 для розроблюваного програмного забезпечення

Наведена вище загальна діаграма відображає те, як функціонує програмне забезпечення в масштабному плані.

Для більш детального вивчення на рисунках нижче наведено декомпозиції діаграми. На рисунку 2.10 зазначено декомпозицію основної діаграми.

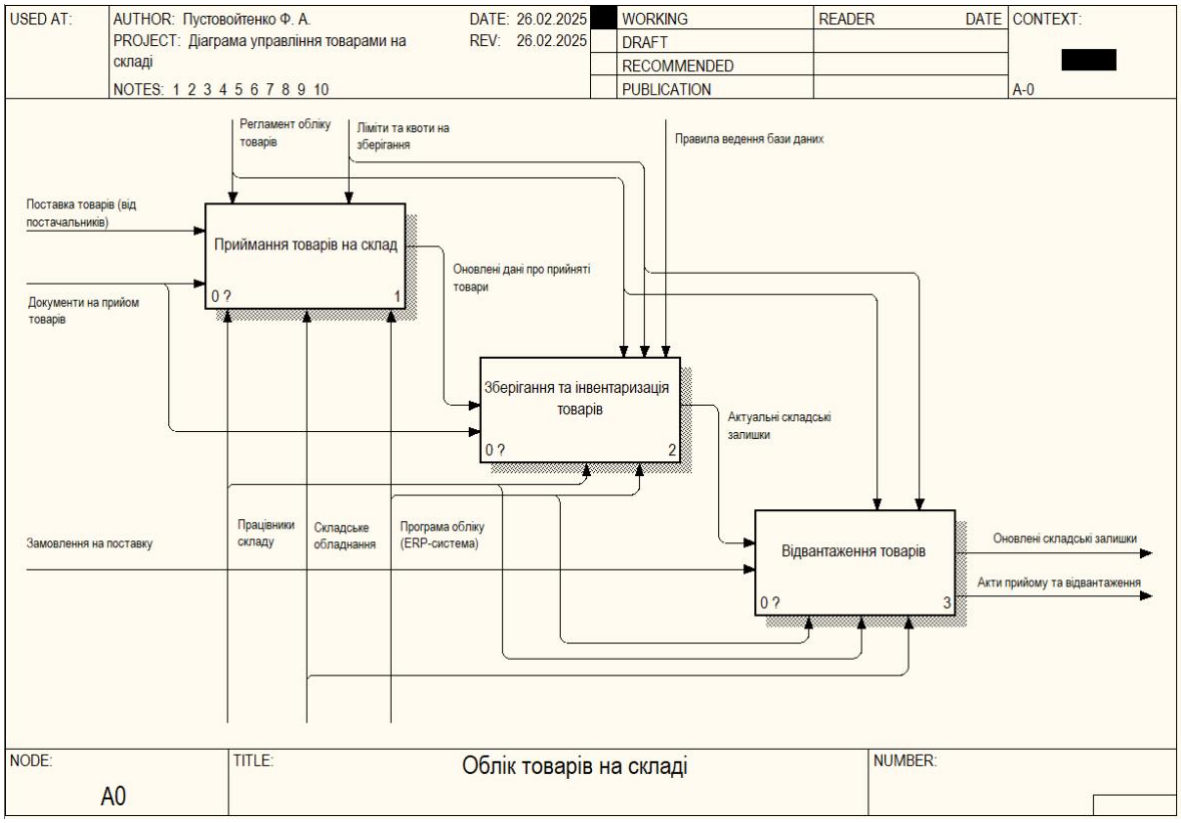


Рисунок 2.10 – Декомпозована IDF0 діаграма розроблюваного програмного забезпечення

На рисунку 2.10 зазначена діаграма IDEF0, що дозволяє деталізувати основний процес, розбиваючи його на менші процеси, для кращого розуміння взаємодії між функціональними блоками розроблюваного програмного забезпечення. Це дає змогу проаналізувати кожен етап роботи системи, визначити ключові входи, виходи, механізми та управлінські впливи.

Декомпозована діаграма використовується для обліку товарів на складі, що включає такі важливі підпроцеси, як приймання товарів, зберігання,

переміщення та відвантаження.

Для ліпшого розуміння роботи системи, нижче наведена декомпозиція діаграми IDF0 для кожного підпроцесу.

На рисунку 2.11 зазначено декомпозовану діаграму A1.

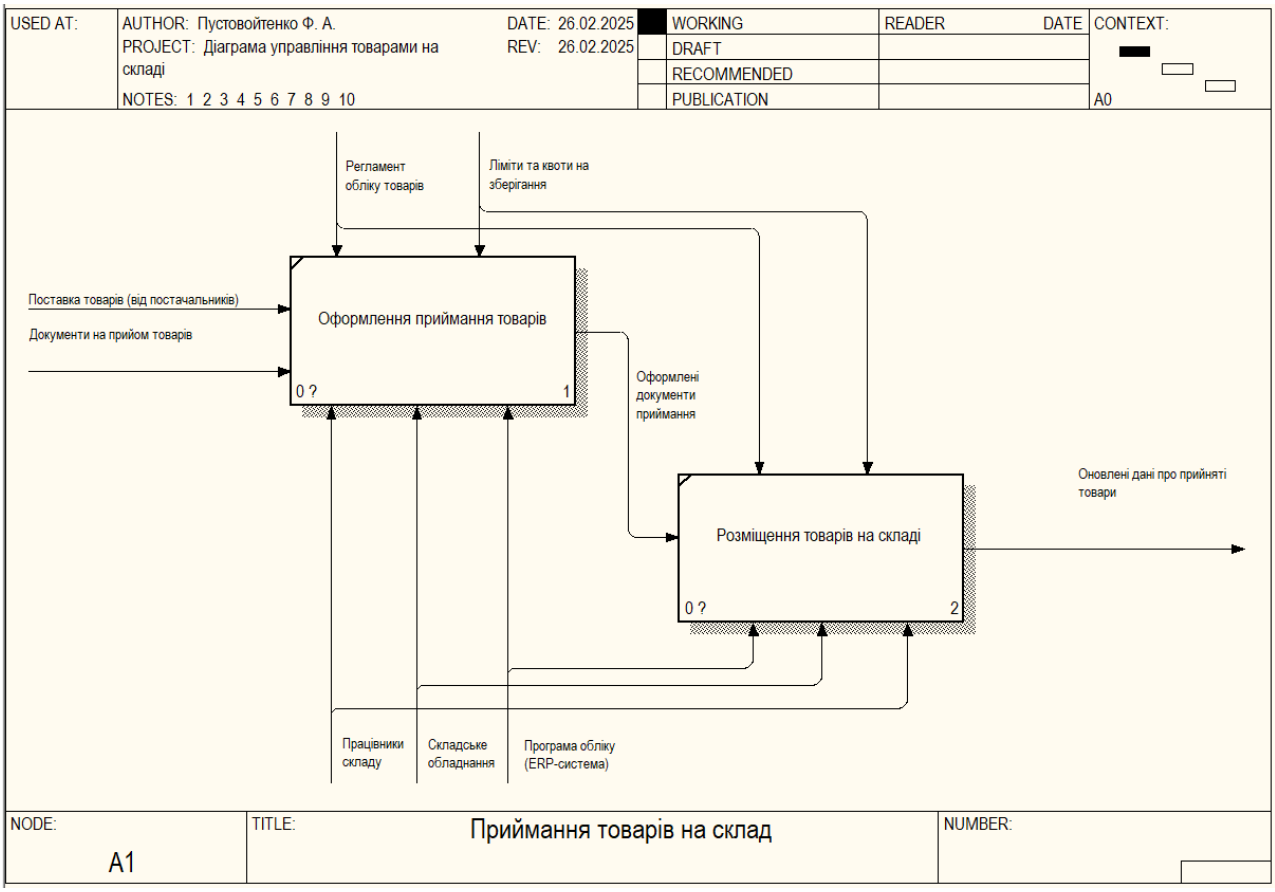


Рисунок 2.11 – Декомпозована діаграма A1

Декомпозована діаграма A1 деталізує процеси, пов’язані з надходженням товарів на склад. Вона включає такі ключові етапи, як перевірка відповідності отриманих товарів, внесення даних у систему та оформлення необхідної документації. Визначення всіх вхідних і вихідних потоків дозволяє оптимізувати процес реєстрації, зменшити ризик помилок та забезпечити узгодженість із закупівельним відділом і постачальниками. На рисунку 2.12 зазначено декомпозицію наступної діаграми A2.

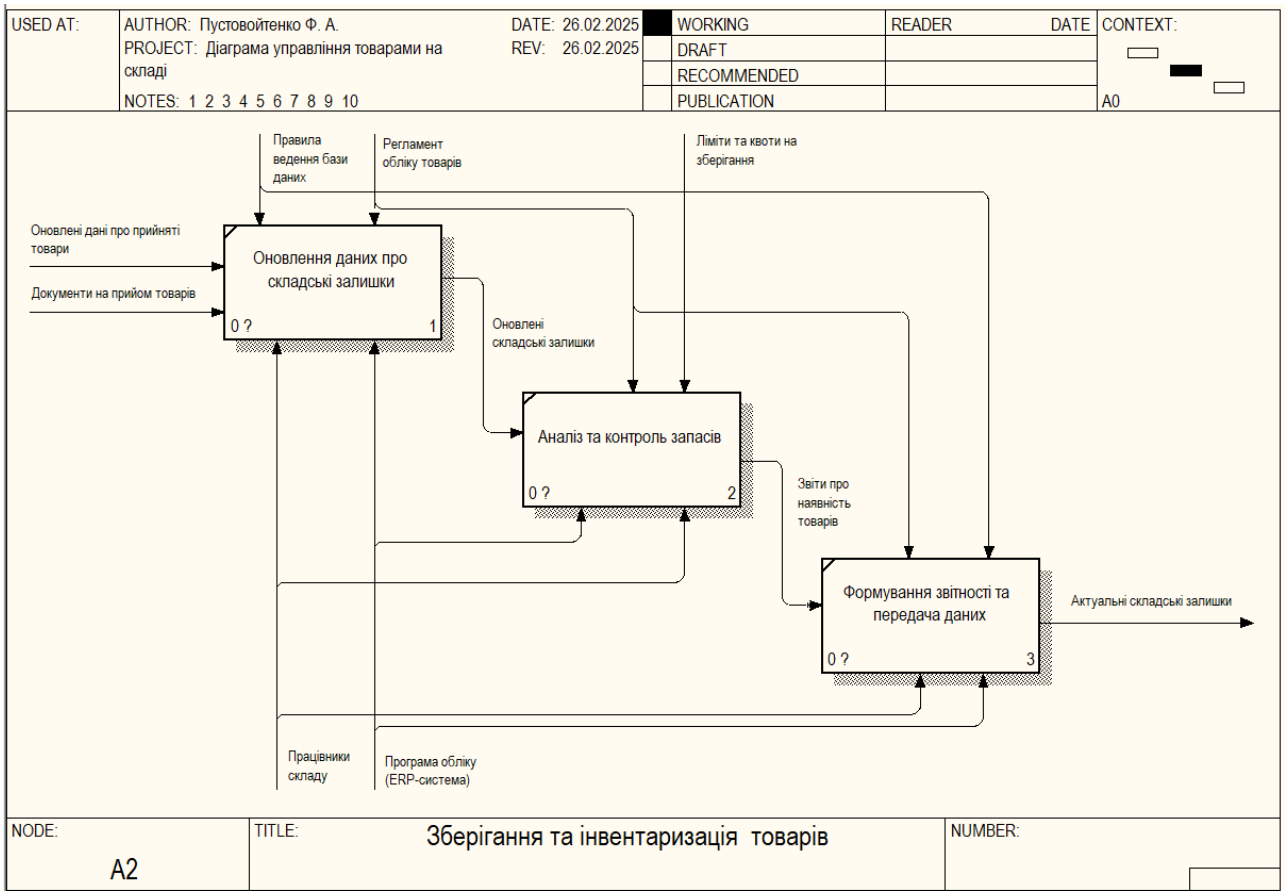


Рисунок 2.12– Декомпозована діаграма A2

Декомпозована діаграма A2 відображає механізм управління складськими запасами, включаючи розподіл товарів за зонами зберігання, контроль умов зберігання та проведення інвентаризацій. Декомпозиція дозволяє відстежити, як товари переміщуються всередині складу, які механізми використовуються для їхнього обліку та яким чином забезпечується точність даних у системі. Це допомагає уникнути нестачі або надлишку товарів, а також покращує швидкість пошуку та підготовки замовлень.

На рисунку 2.13 зазначено декомпозицію діаграми A3.

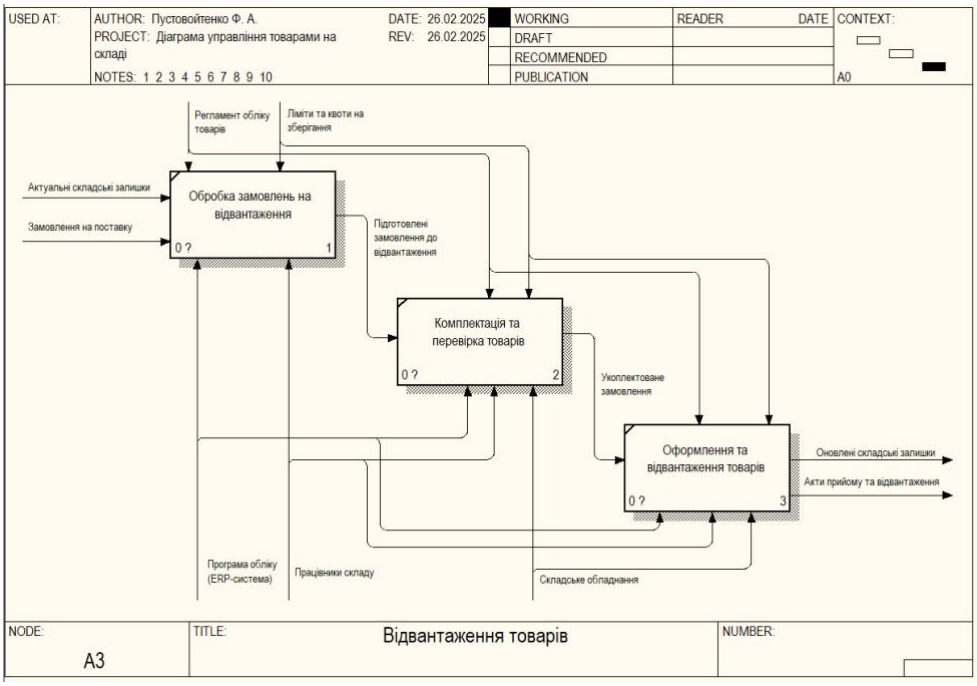


Рисунок 2.13 – Декомпозована діаграма A3

Декомпозована діаграма A3 деталізує процеси комплектації, перевірки та відправлення товарів клієнтам або іншим підрозділам підприємства. Вона дозволяє чітко визначити порядок підготовки замовлень, перевірку відповідності перед відвантаженням та оформлення супровідної документації.

Оптимізація цих процесів сприяє зменшенню затримок у логістиці, підвищенню точності виконання замовлень та покращенню координації з транспортними службами.

2.8 Розрахунки для забезпечення резерву ресурсів при поставках

Для забезпечення безупинної роботи підприємства необхідно подбати про нештатні ситуації, що можуть виникати під час робочого процесу. Саме з цією метою на підприємстві створюється резерв ресурсів на випадок, якщо заявлена потреба при замовленні виявиться неточною, чи виникне нестандартна ситуація.

Для забезпечення такого резерву в розроблюваному програмному рішенні

була прорахована формула для розрахунку рекомендованої кількості матеріалів до поставки з урахуванням потреби в резерві.

У формулі (2.1) зазначено алгоритм прорахунку кількості поставок.

$$S_{ЗАГ} = S_{БАЗОВА} + (S_{БАЗОВА} \times 20\%), \quad (2.1)$$

де $S_{БАЗОВА}$ – базова потреба в ресурсах на основі активних замовлень та стану складу;

$S_{ЗАГ}$ – загальна рекомендована кількість одиниць виміру продукції, рекомендована до поставки.

З наведеної вище формули (2.1) можна побачити, що резерв сягає розміру 20 відсотків від поточної потреби, що прораховується на основі властивостей продукції, що замовляється у підприємства. Поточна потреба розраховується за формулою (2.2).

$$S_{БАЗОВА} = \sum_{i=1}^{n} q_{Ti} \times c_i, \quad (2.2)$$

де $S_{БАЗОВА}$ – загальна кількість матеріалу, необхідного для виробництва всіх товарів для всіх активних замовлень;

q_T – кількість товару конкретного виду, що треба виробити;

c – кількість матеріалів, необхідних для виробу однієї одиниці виробу.

З наведеної вище формули (2.2) можна побачити, що розрахунок необхідних матеріалів ведеться на основі активних замовлень, що дає можливість створювати запити на поставки матеріалів лише за їх фактичної потреби, що дає зменшити витрату часу на облік поставок.

2.9 Розрахунок резерву на основі внутрішніх факторів

Для того, щоб правильно розрахувати необхідний обсяг резерву для ресурсів, треба враховувати різні фактори, такі, як потенційний брак матеріалу, потенційна зміна обсягу замовлення, та форс-мажорні ситуації, пов’язані з виходом з ладу компоненту підприємства, затримкою поставок, тощо. Формула (2.3) відображає правильний розрахунок для резерву.

$$R_{ЗАГ} = R_{БРАК} + R_{ЗАМОВЛЕННЯ} + R_{ФОРС}, \quad (2.3)$$

де $R_{БРАК}$ – брак матеріалу, оцінюється в 5%;

$R_{ЗАМОВЛЕННЯ}$ – потенційна зміна обсягу замовлення, оцінюється до 10%;

$R_{ФОРС}$ – форс-мажорна ситуація, оцінюється в 5%.

В результаті отримаємо вираз, з якого виходить, що загальна кількість необхідного резерву становить:

$$R_{ЗАГ} = 5\% + 10\% + 5\% = 20\%$$

Виходячи з результатів цього розрахунку саме значення 20 відсотків використовується в розроблюваному програмному забезпеченні.

2.10 Розрахунок вартості розробки та підтримки програмного забезпечення

Як і будь яке рішення, розроблювана ERP система потребує часу та кошти на розробку, та підтримку. Перед розробкою програмного забезпечення необхідно провести не лише аналіз потреб та вимог, але й доцільності розробки системи, для визначення пріоритетів, та ціни, за яку необхідно реалізовувати

розроблену систему.

Для початку варто виділити певні вхідні дані для розрахунків. Розробка системи зайняла 2 робочих місяці, рівень кваліфікації єдиного розробника – Middle, середньостатистична зарплатня – 2000 доларів США. Виходячи з цього, маємо формулу (2.4).

$$S_{\text{РОЗРОБКИ}} = (\text{с}_{\text{плата за місяць}} + \text{технічні витрати}) \times 2 + \text{разові витрати} \qquad (2.4)$$

де $S_{\text{РОЗРОБКИ}}$ – Сума ціни розробки за термін часу розробки;

$\text{с}_{\text{плата за місяць}}$– зарплатня розробника за місяць;

технічні витрати – покупка серверів, домену, та інші технічні витрати;

разові витрати – 7.95 доларів США на покупку імені домену.

В результаті отримаємо вираз, з якого виходить, що витрати на розробку становлять:

$$S_{\text{РОЗРОБКИ}} = (2000\ \$ + 50\ \$) \times 2 + 7.95\ \$ = 4107.95\ \$$$

Виходячи з результатів цього розрахунку саме значення в 4107.95 доларів США є собівартістю розробки програмного забезпечення. Аналізуючи ринок у вищезазначених пунктах кваліфікаційної роботи, було виявлено, що середня маржинальність такого забезпечення становить мінімум 100%, що становить додаткові 4 тисячі доларів США. Проте, оскільки програмне рішення розробляється для внутрішнього ринку, і націлене на допомогу національним підприємствам, прийнято рішення реалізації за собівартістю, як частина програми з спрощення роботи з програмним забезпеченням, вирішення фінансової проблеми є одним із ключових факторів.

Для того, щоб підтримувати програмне забезпечення після його реалізації,

необхідно також витрачати певні кошти.

Перш за все, це 30 доларів США для оренди серверу, для забезпечення працездатності системи, у випадку, якщо підприємство не має можливості розгорнути його в себе, приблизно 10 доларів США на рік для підтримки права власності на ім’я домену, та технічна підтримка 10 годин на місяць, становить 120 доларів США, та може змінюватися. Отже, ціна підтримки – 150 доларів США на місяць, та 10 доларів США на рік на домен. За цю ціну покупець отримує повну технічну підтримку, та забезпечення працездатності системи, що дає змогу змінити фокус суто на використання системи.

3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ERP СИСТЕМИ

3.1 Проектування структури бази даних, розробка та створення таблиць у СКБД POSTGRESQL

Оскільки розроблювана ERP система працює з даними, їх необхідно зберігати в базі даних. В наведеному вище аналізі різних систем керування базами даних було обрано PostgeSQL. Для того, щоб розробити ефективний застосунок, необхідно подбати про ефективну архітектуру зберігання даних, яка готова до масштабування, та роботи під великим навантаженням. На рисунку 3.1 зазначено розроблену структуру бази даних у програмному засобі PostgreSQL.

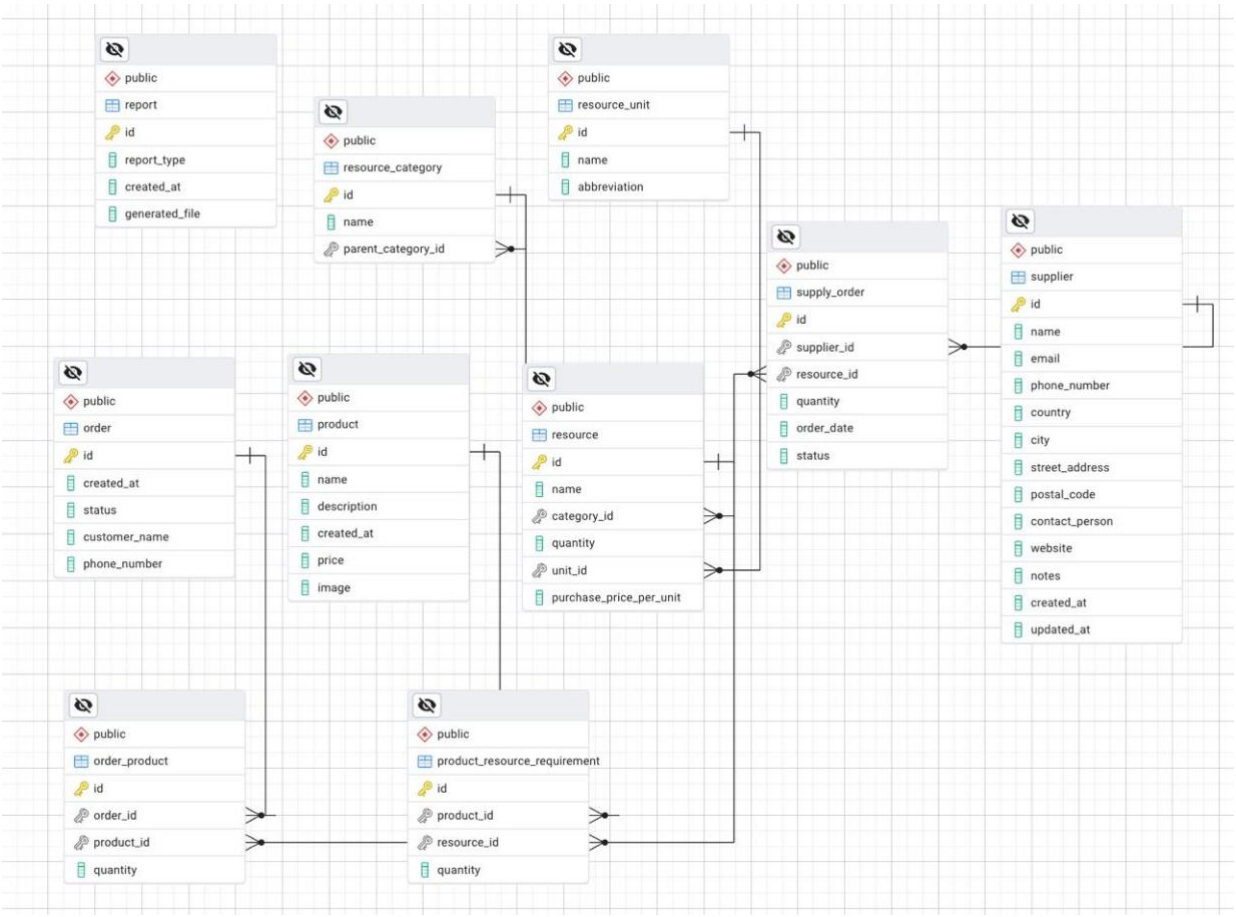


Рисунок 3.1 – Загальна структура бази даних

Виходячи з наведеної структури бази даних, можна відзначити, що як і було зазначено в пунктах раніше, дані сильно пов’язані між собою, що робить нашу систему поєднанням різних підсистем, і дає перевагу перед децентралізованими рішеннями, адже усі ці дані налаштовані на співпрацю між собою, і не потребують сил на їх нормалізацію для обробки. Варто відзначити, що робота з базою даних велася на різних рівнях, а саме за допомогою SQL запитів, за допомогою графічного інтерфейсу PgAdmin, та за допомогою вбудованого в фреймворк Django інструменту Django ORM, який надає можливість працювати з реляційними базами даних в парадигмі об’єктно-орієнтованого програмування, де таблиця – це клас, кожна колонка – це властивість класу, і кожний запис до цієї таблиці, це об’єкт, який є екземпляром цього класу. Такий підхід до роботи з даними в базі даних значно спрощує розробку, та зменшує витрачений на неї час, що позитивно впливає на вартість програмного забезпечення, вартість його оновлення та підтримки.

Для початку, варто розібрати таблицю, присвячену звітам, оскільки вона є ізольована від інших, і унікальна в своєму алгоритмі роботи в системи. На рисунку 3.2 зазначено програмний код моделі Django ORM, який відповідає за роботу з таблицею системи звітів у розроблюваному програмному забезпеченні.

```python

class Report(models.Model):
    REPORT_TYPES = [
        ("stock_status", "Стан складу"),
        ("last_supplies", "Останні 10 поставок"),
        ("supply_recommendations", "Рекомендації до поставок"),
    ]

    report_type = models.CharField(max_length=50, choices=REPORT_TYPES)
    created_at = models.DateTimeField(default=now)
    generated_file = models.FileField(upload_to="reports/", null=True, blank=True)

    def __str__(self):
        return f"Звіт: {self.get_report_type_display()} ({self.created_at})"

```

Рисунок 3.2 – Програмна модель системи звітів

На наведеному вище фрагменті коду можна побачити, що основним полем в нас є тип звіту, на даний момент в системі існує 3 види звітності – інформація про стан складу, інформація про останні 10 поставок, та рекомендації до поставок на основі стану складу. Наступним параметром в нас є created_at, який має тип даних DataTime, що відповідає збереженню дати. Це поле відповідає за дату та час створення кожного звіту. Передостаннім, найважливішим полем є generated_file, яке відповідає за зберігання самого файлу звіту. Це поле формату FileField, що дає базі даних інструкцію, що це поле буде зберігати файл. В самому проекті він буде зберігатися в папці reports, що зазначено в параметрах цієї властивості класу звітів. Останнім параметром, який не впливає на базу даних, але впливає на поведінку звітів в самому проекті є прописана функція, яка відповідає за коректне відображення звіту в палені адміністратора.

Наступною необхідно розглянути таблицю ресурсів, адже вона пов’язана з великою кількістю інших таблиць. На рисунку 3.3 зазначено фрагмент коду, який відповідає за роботу із таблицею ресурсів.

```python
class Resource(models.Model):
    name = models.CharField(max_length=255)
    category = models.ForeignKey(
        ResourceCategory, on_delete=models.CASCADE, related_name="resources"
    )
    quantity = models.DecimalField(max_digits=10, decimal_places=2)
    unit = models.ForeignKey(ResourceUnit, on_delete=models.CASCADE)
    purchase_price_per_unit = models.DecimalField(
        max_digits=10, decimal_places=2
    )  # Ціна закупівлі за одиницю ресурсу

    def __str__(self):
        return self.name

```

Рисунок 3.3 – Модель ресурсів

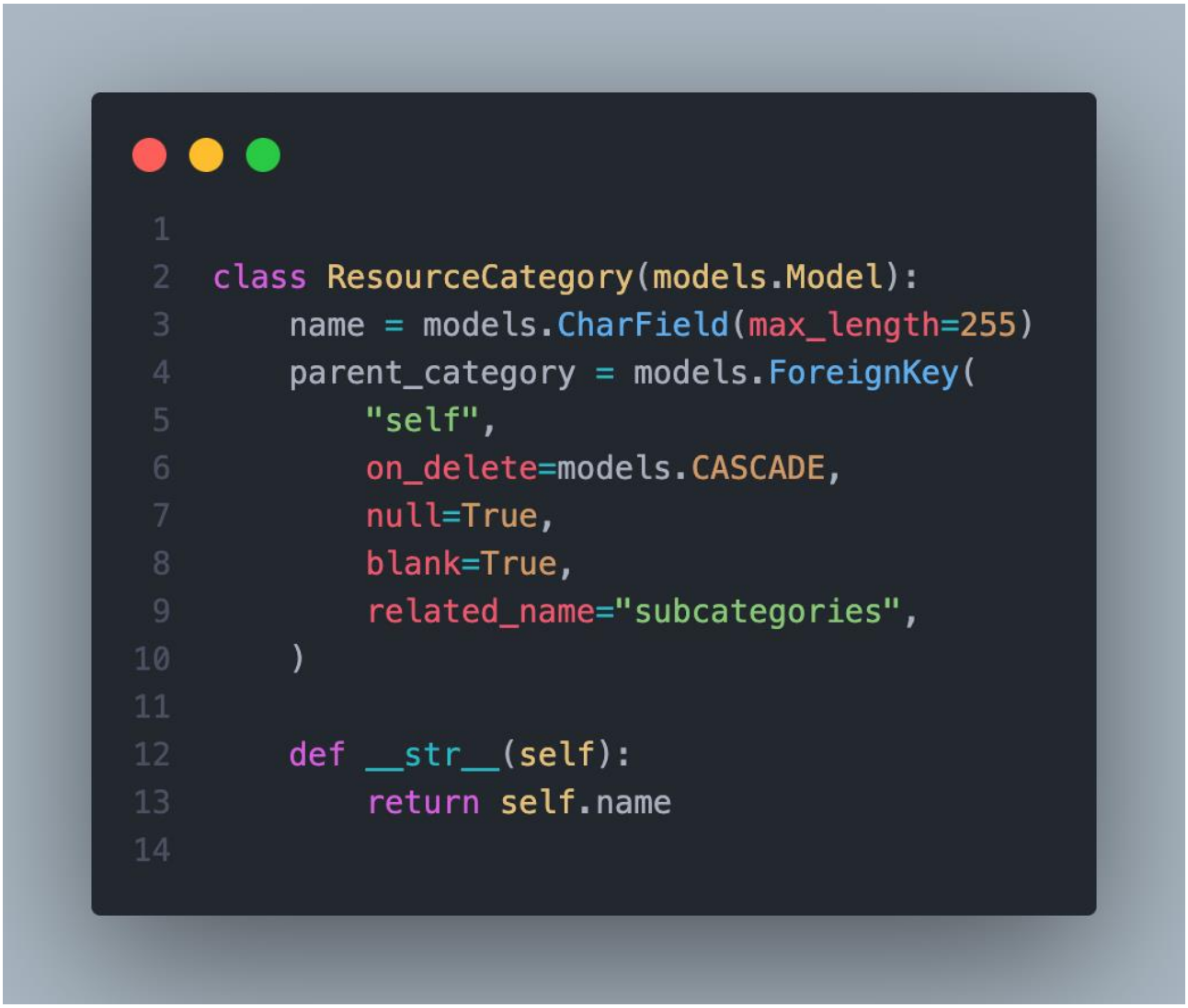
Виходячи з наведеного вище фрагменту коду можна побачити, що ресурси мають назву, яка зберігається у вигляді символьної строки, категорії, яка пов’язана з іншою таблицею категорій за допомогою ForeignKey зв’язку, який передбачає зв’язок типу «Один до багатьох», що означає, що в ресурсу може бути лише одна категорія, а в категорії може бути безліч ресурсів. Модель категорій буде розглянута далі. Наступне поле – кількість. Воно налаштовано таким чином, що зберігає дані з точністю до сотих одиниць. Це рішення прийняте для роботи з особливими матеріалами, яким необхідна така точність. Наступне поле unit – це одиниці виміру ресурсу. Воно, подібно полю категорії, встановлює зв’язок «Один до багатьох» з таблицею, присвяченою одиницям виміру. Це створено для того, щоб уникати плутанини з ресурсами, та полегшити процес роботи з додатком, адже це зробить його інтуїтивно зрозумілішим. Наступне поле прокоментоване в самому коді, і, як зазначено, відповідає вартості закупівлі ресурсу за одну одиницю. Ця інформація необхідна для того, щоб утворювати поставки,

розраховувати їхню вартість, та розраховувати вартість виробу, який використовує конкретні ресурси в конкретній кількості. Останнім параметром є функція, яка відповідає за правильне візуальне відображення ресурсу в роботі з ним. Варто відзначити, що для усіх відношень в цій таблиці використовується аргумент models.CASCADE, який дає інструкцію, що у випадку, якщо видаляється пов’язана модель (категорія, чи одиниця виміру), то й видаляється кожен пов’язаний запис в базі даних. Наступними будуть розглянуті моделі категорій та одиниць виміру, адже вони використовуються для опису ресурсів, і мають відношення лише до них. На рисунку 3.4 зазначено модель, яка відповідає за роботу із системою одиниць виміру для ресурсів.

Рисунок 3.4 – Модель для роботи з одиницями виміру

З наведеного вище рисунку можна побачити, що ця система максимально проста. Ми в базі даних зберігаємо всього два поля, це назва одиниці виміру, та її абревіатура для скороченого використання, як от «шт.», «л.» та інші. І, як і раніше, присутня функція, яка відповідає за коректне відображення у панелі

адміністратора кожної конкретної одиниці виміру. Наступним кроком необхідно розглянути систему категорій, адже вона включає в себе цікаві інженерні рішення, та вкрай важлива для функціонування розроблюваного програмного забезпечення. На рисунку 3.5 зазначено фрагмент коду, який відповідає за цю систему.



```
1
2  class ResourceCategory(models.Model):
3      name = models.CharField(max_length=255)
4      parent_category = models.ForeignKey(
5          "self",
6          on_delete=models.CASCADE,
7          null=True,
8          blank=True,
9          related_name="subcategories",
10     )
11
12     def __str__(self):
13         return self.name
14
```

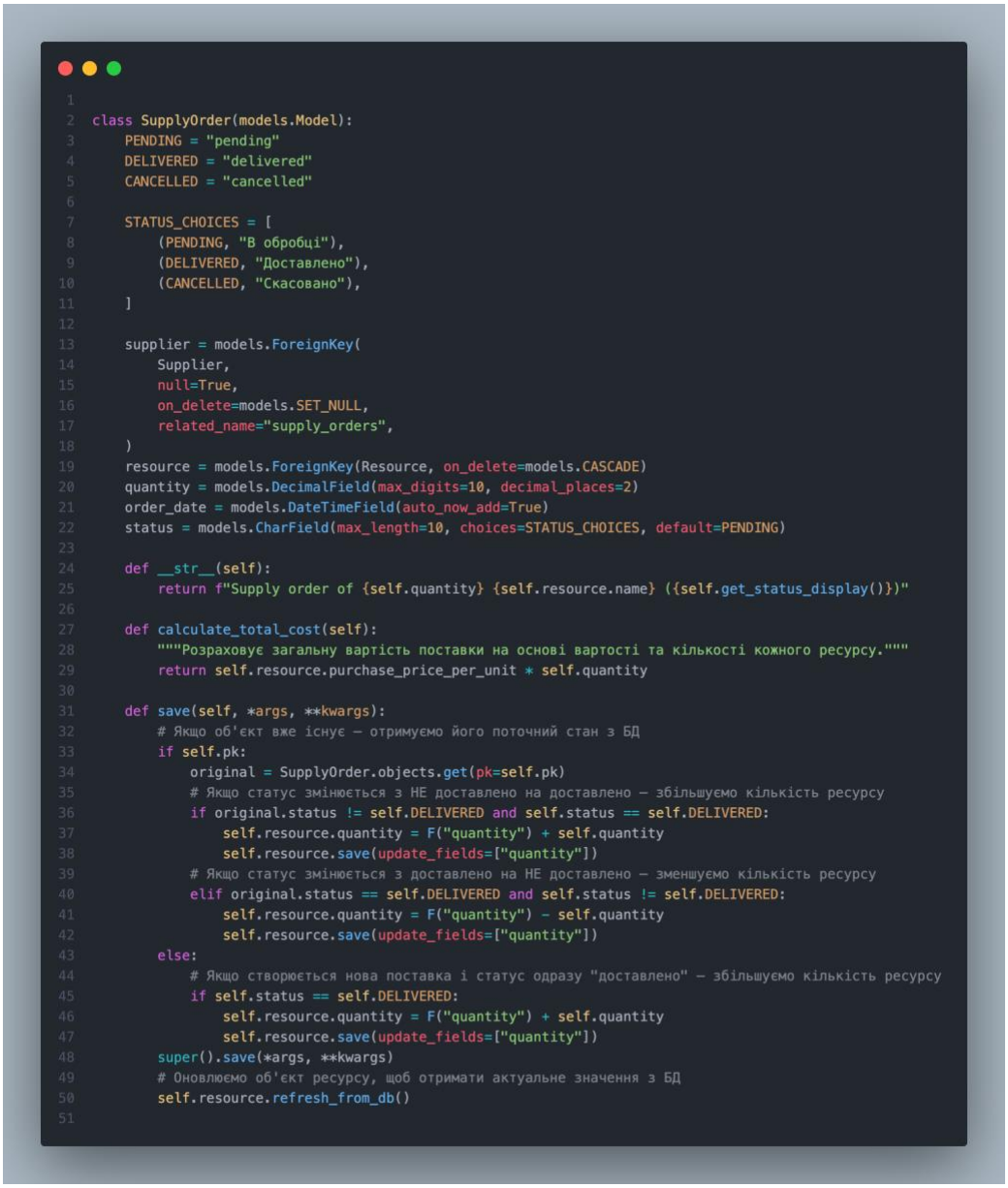
Рисунок 3.5 – Функціонал системи категорій

На зазначеному вище рисунку можна побачити, що модель категорій включає в себе всього два поля, а саме назву, яка зберігається в базі даних як символьний рядок, та поле батьківської категорії.

В цьому випадку застосовується досить цікавий та важливий метод оптимізації бази даних. В розроблюваній ERP системі передбачається можливість створення підкатегорій для кожної категорії, тобто, для прикладу, існує категорія «Електроніка», яка є досить загальною, та викликатиме труднощі, якщо ресурсів з цієї категорії буде сотні, або тисячі. Для запобігання проблемі створено систему підкатегорій, які слугують для детальності, до чого саме відноситься той чи інший ресурс. Наприклад, є загальна категорія «Електроніка», і в неї є підкатегорії «Компоненти», «Вироби», в яких є свої підкатегорії, тощо.

З точки зору реалізації цієї системи найкращим варіантом є створення поля для батьківської категорії, яке посилається на цю ж таблицю, оскільки це зменшує навантаження на базу даних, та полегшує роботу з нею. Саме тому було встановлено зв’язок типу ForeighKey з аргументом self, який прямо вказує, що посилання йде на саму ж таблицю, з вказанням того, що якщо видаляється батьківська категорія, то видаляються усі дочірні, та інші інструкції для бази даних, які необхідні для коректної роботи. Також, як і раніше, знизу фрагменту коду створено функцію, яка відповідає за правильне відображення категорії в панелі адміністратора.

Наступною необхідно розглянути систему поставок, яка буде поставляти підприємству зазначені вище ресурси. На рисунку 3.6 зазначено модель, яка відповідає за роботу з системою поставок в розроблюваному програмному забезпеченні.



```python

class SupplyOrder(models.Model):
    PENDING = "pending"
    DELIVERED = "delivered"
    CANCELLED = "cancelled"

    STATUS_CHOICES = [
        (PENDING, "В обробці"),
        (DELIVERED, "Доставлено"),
        (CANCELLED, "Скасовано"),
    ]

    supplier = models.ForeignKey(
        Supplier,
        null=True,
        on_delete=models.SET_NULL,
        related_name="supply_orders",
    )
    resource = models.ForeignKey(Resource, on_delete=models.CASCADE)
    quantity = models.DecimalField(max_digits=10, decimal_places=2)
    order_date = models.DateTimeField(auto_now_add=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES, default=PENDING)

    def __str__(self):
        return f"Supply order of {self.quantity} {self.resource.name} ({self.get_status_display()})"

    def calculate_total_cost(self):
        """Розраховує загальну вартість поставки на основі вартості та кількості кожного ресурсу."""
        return self.resource.purchase_price_per_unit * self.quantity

    def save(self, *args, **kwargs):
        # Якщо об'єкт вже існує — отримуємо його поточний стан з БД
        if self.pk:
            original = SupplyOrder.objects.get(pk=self.pk)
            # Якщо статус змінюється з НЕ доставлено на доставлено — збільшуємо кількість ресурсу
            if original.status != self.DELIVERED and self.status == self.DELIVERED:
                self.resource.quantity = F("quantity") + self.quantity
                self.resource.save(update_fields=["quantity"])
            # Якщо статус змінюється з доставлено на НЕ доставлено — зменшуємо кількість ресурсу
            elif original.status == self.DELIVERED and self.status != self.DELIVERED:
                self.resource.quantity = F("quantity") - self.quantity
                self.resource.save(update_fields=["quantity"])
        else:
            # Якщо створюється нова поставка і статус одразу "доставлено" — збільшуємо кількість ресурсу
            if self.status == self.DELIVERED:
                self.resource.quantity = F("quantity") + self.quantity
                self.resource.save(update_fields=["quantity"])
        super().save(*args, **kwargs)
        # Оновлюємо об'єкт ресурсу, щоб отримати актуальне значення з БД
        self.resource.refresh_from_db()

```

Рисунок 3.6 – Модель для роботи із поставками

Ця модель досить важка для розробки, але дуже важлива для функціонування розроблюваної ERP системи. Для початку, можна побачити, що в нас встановлений ForeignKey зв’язок із таблицею постачальників, яку буде розглянуто далі. Оскільки поставки – це чутлива інформація, яка впливає на склад, тут було встановлено інший параметр зв’язку – models.SET_NULL, який, у випадку видалення не видалить інформацію про поставку, а встановить показник постачальника в Null. Наступне поле – аналогічне посилання на

таблицю ресурсів, адже це поставки ресурсів. Після цього ми вказуємо кількість ресурсу, який бажаємо замовити. Наступним кроком є поле, яка відповідає за дату та час створення поставки, яке автоматично генерується, та статус заявки, який є дуже важливим та впливовим. Всього в системі передбачено 3 статуси – В обробці, доставлено та скасовано. При створенні поставки їй автоматично надається статус «В обробці». Наступною йде функція, яка відповідає за відображення поставки належним чином в панелі адміністратора. Далі створено функцію, яка розраховує загальну вартість поставки – вкрай важливий, критичний показник, який може впливати на вибір постачальників, та коригування плану поставок, плану виробництва, тощо. Розраховується за принципом множення кількості ресурсу на його вартість за одиницю. Далі було перероблено базову функцію save(), яка відповідає за процес збереження інформації в базі даних. Було створено інтерактивну систему, яка на основі змін інформації про поставку може впливати на інші підсистеми в межах розроблюваної ERP системи. Завдяки коментарям в фрагменті коду можна побачити, що при зміні статусу доставки в нас коригується інформація про ресурси на складі. Якщо відповідальна особа зареєструвала поставку як прийняту, то автоматично надсилається інформація про кількість ресурсів, і оновлюється в відповідальній за склад підсистемі, і навпаки. Це досить складна та важлива система, яка допомагає зберігати дуже великий обсяг часу персоналу, зменшуючи потреби ревізії складу до планових періодичних. Варто відзначити, що розробка такої системи потребує достатнього тестування, та уважності, адже у випадку неправильного налаштування, вона може викликати великі розбіжності між тим, що відбувається на складі, і тим, що зафіксовано в базі даних.

Наступною, найбільш наближеною системою є система постачальників, відповідальний фрагмент коду якої зазначено на рисунку 3.7.

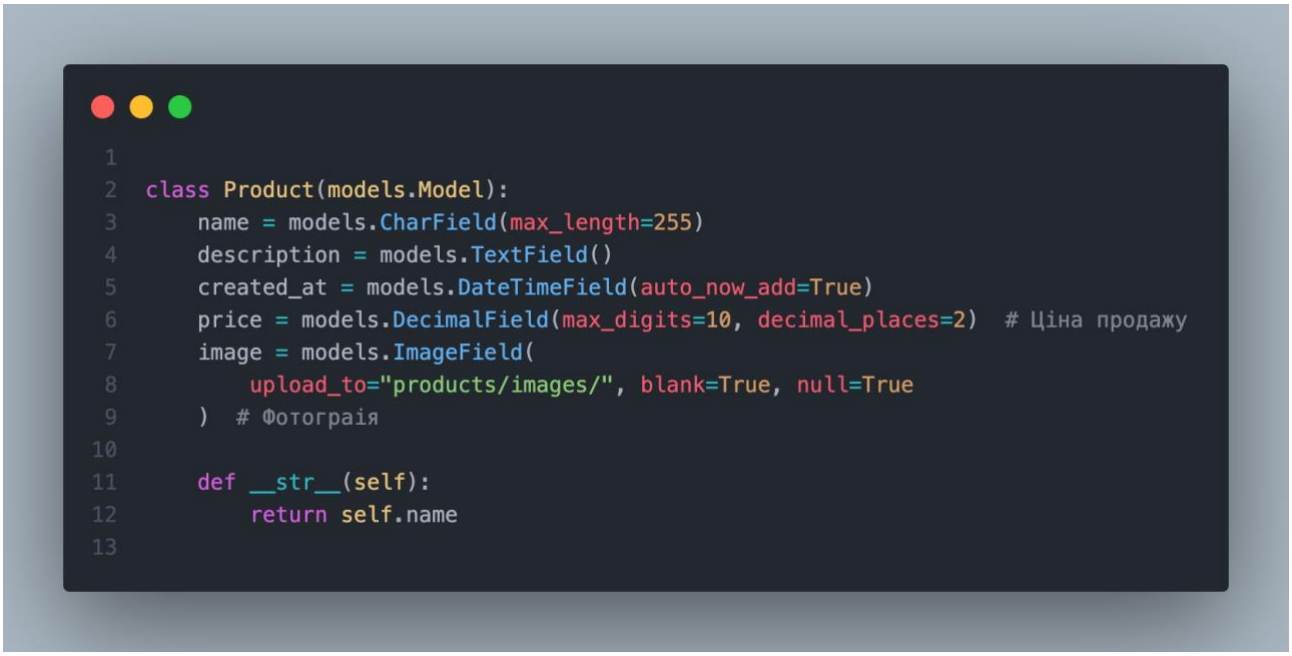
```python
class Supplier(models.Model):
    name = models.CharField(max_length=255, verbose_name="Назва постачальника")

    email = models.EmailField(verbose_name="Електронна пошта")
    phone_number = PhoneNumberField(
        region="UA",
        verbose_name="Номер телефону",
        help_text="Введіть номер у форматі +380 XX XXX XXXX",
    )
    country = models.CharField(max_length=100, verbose_name="Країна")
    city = models.CharField(max_length=100, verbose_name="Місто")
    street_address = models.CharField(
        max_length=255, verbose_name="Вулиця та номер будинку"
    )
    postal_code = models.CharField(max_length=20, verbose_name="Поштовий індекс")

    contact_person = models.CharField(
        max_length=100, verbose_name="Контактна особа", blank=True, null=True
    )
    website = models.URLField(verbose_name="Вебсайт", blank=True, null=True)
    notes = models.TextField(verbose_name="Примітки", blank=True, null=True)

    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")
    updated_at = models.DateTimeField(auto_now=True, verbose_name="Дата оновлення")

    class Meta:
        verbose_name = "Постачальник"
        verbose_name_plural = "Постачальники"

    def __str__(self):
        return self.name

```

Рисунок 3.7 – Модель для роботи із таблицею постачальників

З наведеного вище рисунку можна побачити, що модель не використовує ніяких складних відношень із іншими таблицями, та наповнена різного виду контактною інформацією про постачальника. В залежності від типу інформації використовується відповідний тип даних в базі даних, окремий для посилань, нотаток, телефону, тощо. Наведено коригуючі класи та функції для коректного відображення постачальників в панелі адміністратора.

Наступним кроком необхідно розглянути систему продукції, яку

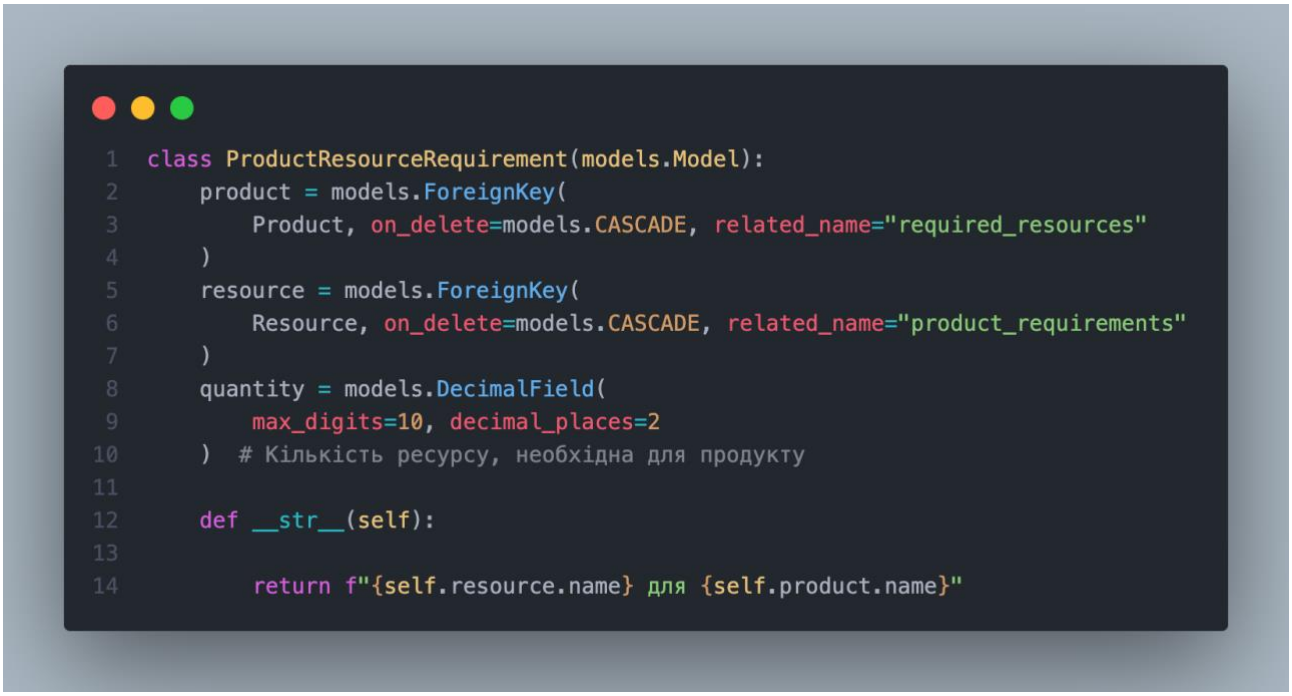
виготовляє приладобудівне виробництво, адже це його сенс, та головна мета існування. На рисунку 3.8 зазначено модель для роботи із продукцією.



```python

class Product(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)  # Ціна продажу
    image = models.ImageField(
        upload_to="products/images/", blank=True, null=True
    )  # Фотографія

    def __str__(self):
        return self.name

```

Рисунок 3.8 – Фрагмент програмного коду для роботи із продукцією приладобудівного виробництва

З наведеного вище рисунку можна побачити, що система досить проста, містить технічну інформацію про продукт, його фотографію, ціну, опис, назву, та дату, коли такий виріб було впроваджено на підприємстві. Також присутня вже відома функція для відображення коректної назви. Однак, це лише невеличкий фрагмент системи. Кожен виріб потребує ресурсів на його виготовлення, які так само необхідно фіксувати задля оцінки вартості виробу, та оцінки витрат складу на виготовлення партії виробів. Для цього було розроблено особливу модель, яка відповідає за вимоги виробу по ресурсам. Модель цієї підсистеми зазначено на рисунку 3.9.



```python
class ProductResourceRequirement(models.Model):
    product = models.ForeignKey(
        Product, on_delete=models.CASCADE, related_name="required_resources"
    )
    resource = models.ForeignKey(
        Resource, on_delete=models.CASCADE, related_name="product_requirements"
    )
    quantity = models.DecimalField(
        max_digits=10, decimal_places=2
    )  # Кількість ресурсу, необхідна для продукту

    def __str__(self):

        return f"{self.resource.name} для {self.product.name}"
```

Рисунок 3.9 – Модель для вимог продукту по ресурсам

З наведеного вище фрагменту коду можна побачити, що ми встановлюємо зв’язки з моделями для продукту і ресурсу, а також встановлюємо показник кількості. Це технічне рішення створено для того, щоб була можливість оцінити розроблюваний товар у ресурсах, які в свій час мають грошову оцінку за одиницю виробу, що за допомогою калькуляцій допоможе визначити собівартість продукту. Так само наявна функція, яка відповідає за коректне відображення у панелі адміністратора.

Наступною необхідно розглянути систему замовлень, яка і є джерелом прибутку для підприємств. На рисунку 3.10 зазначено модель системи замовлень для роботи із базою даних.

```python
class Order(models.Model):
    CREATED = "created"
    IN_PROGRESS = "in_progress"
    COMPLETED = "completed"
    CANCELLED = "cancelled"

    ORDER_STATUS_CHOICES = [
        (CREATED, "Створено"),
        (IN_PROGRESS, "Виконується"),
        (COMPLETED, "Виконано"),
        (CANCELLED, "Скасовано"),
    ]

    created_at = models.DateTimeField(auto_now_add=True)
    status = models.CharField(
        max_length=20,
        choices=ORDER_STATUS_CHOICES,
        default=CREATED,
    )
    customer_name = models.CharField(max_length=255, verbose_name="Имя заказчика")
    phone_number = PhoneNumberField(
        region="UA",
        verbose_name="Номер телефону",
        help_text="Введіть номер у форматі +380 XX XXX XXXX",
    )

    def __str__(self):
        return f"Order #{self.id} - {self.get_status_display()}"

    def calculate_resource_requirements(self):
        resource_requirements = []
        shortages = []

        for order_product in self.order_products.all():
            product = order_product.product
            quantity = order_product.quantity

            for req in product.required_resources.all():
                total_required_quantity = req.quantity * quantity
                resource_requirements.append(
                    {
                        "resource": req.resource,
                        "required_quantity": total_required_quantity,
                    }
                )

                if total_required_quantity > req.resource.quantity:
                    shortages.append(
                        {
                            "resource": req.resource.name,
                            "shortage_quantity": total_required_quantity
                            - req.resource.quantity,
                        }
                    )

        return resource_requirements, shortages
```

Рисунок 3.10 – Фрагмент коду, який відповідає за систему замовлень в базі даних

Наведена вище система досить складна для проектування та розробки, адже коректна фіксація інформації про замовлення – дуже важливий етап роботи

ERP системи, який формує вимоги, витрати, які впливають на роботу інших підсистем.

Можна побачити, що використовується схожа система статусів на систему поставок, далі йде певна технічна інформація – дата та час фіксації замовлення, контактна інформація замовника. Далі йде функція, яка відповідає за коректну назву в панелі адміністратора, та важлива технічна функція, яка відповідає за розрахунок загальної потреби в ресурсах в цьому замовленні шляхом розкладання замовлення на товари, які в свій час розкладаються на ресурси, і завдяки цьому можна розрахувати вартість замовлення, що й описано в функції. Використовується система, схожа за логікою на систему продуктів, адже використовується допоміжна модель, зазначена на рисунку 3.11.

```
1
2  class OrderProduct(models.Model):
3      order = models.ForeignKey(
4          Order, on_delete=models.CASCADE, related_name="order_products"
5      )
6      product = models.ForeignKey(Product, on_delete=models.CASCADE)
7      quantity = models.PositiveIntegerField()
8
9      def __str__(self):
10         return f"{self.quantity} of {self.product.name} in Order #{self.order.id}"
11
```

Рисунок 3.11 – Допоміжна модель замовлення для реалізації необхідного функціоналу

Зазначена вище допоміжна модель пов’язує замовлення із продукцією, яку виготовляє виробництво, яка в свій час пов’язана із ресурсами. Саме таке «дерево» доступу до ресурсів через кілька ієрархічних рівнів дає можливість оцінити вартість замовлення в ресурсах, і його собівартість, через можливість

оцінити ресурси грошовою мірою. Це вкрай важлива система, яка дає багато цінної, чутливої та важливої інформації, та допомагає автоматизувати вкрай великий обсяг потенційної паперової роботи.

Варто відзначити, що зображено лише найважливіші таблиці та системи, в повній роботі програмного забезпечення участь беруть велика кількість технічних таблиць, які необхідні для функціонування додатку, але їх розглядати немає нагоди, адже вони н мають відношення до ERP системи.

3.2 Візуальна складова розроблюваної ERP системи

Оскільки розробка системи передбачає не лише серверну, а й клієнтську частину, чимало часу необхідно виділити на проектування та розробку графічного інтерфейсу користувача, який має бути інтуїтивно зрозумілим, в міру навантаженим та мати приємний вигляд, який буде виділятися на фоні інших систем. На основі досліджень інтернет ресурсів було виявлено, що синій колір позитивно впливає на продуктивність працівників [17]. Саме тому в якості кольорової палітри веб застосунку було обрано поєднання синього та білого кольорів. Щодо стилю оформлення, було обрано мінімалізм, адже це дає змогу фокусуватися на роботі, і не відволікатися на інші речі. Для полегшення навігації системою було розроблено меню, та достатнього розміру кнопки. На рисунку 3.12 зображено вікно входу в додаток для авторизації.

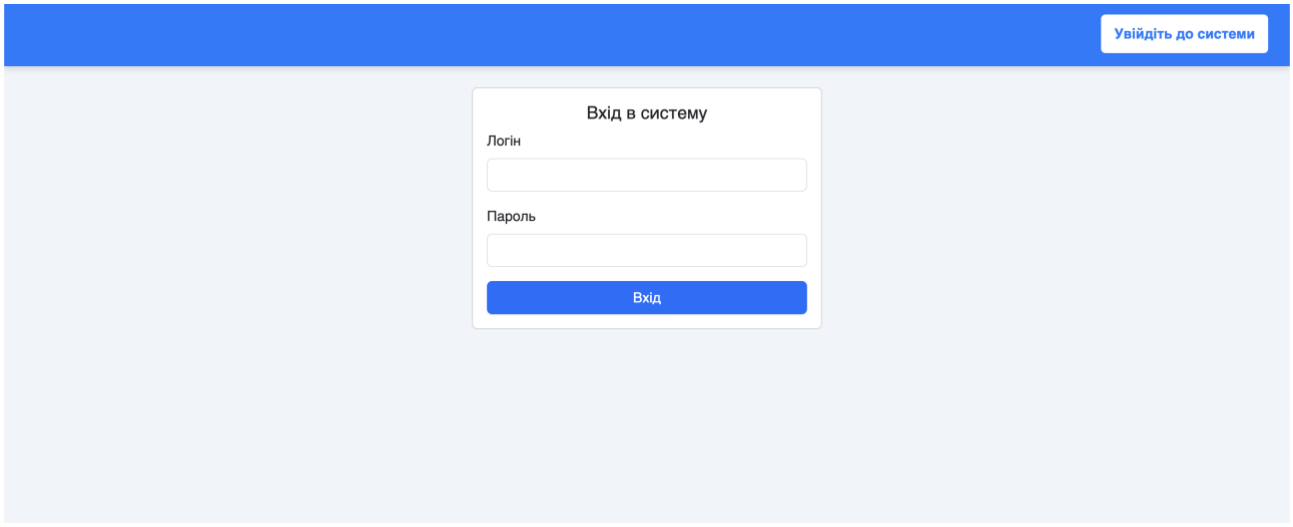


Рисунок 3.12 – Вхід в систему

Як було зазначено вище, використовується мінімалістичний стиль з поєднанням синього та білого кольорів. Саме за такою схемою було створено і інші фрагменти. При вдалій авторизації користувача зустрічає меню, із можливістю обрати ті, чи інші пункти, в залежності від його ролі. З метою розробки було створено користувача з усіма правами, тому йому доступний увесь функціонал. На рисунку 3.13 зображено зовнішній вигляд меню.

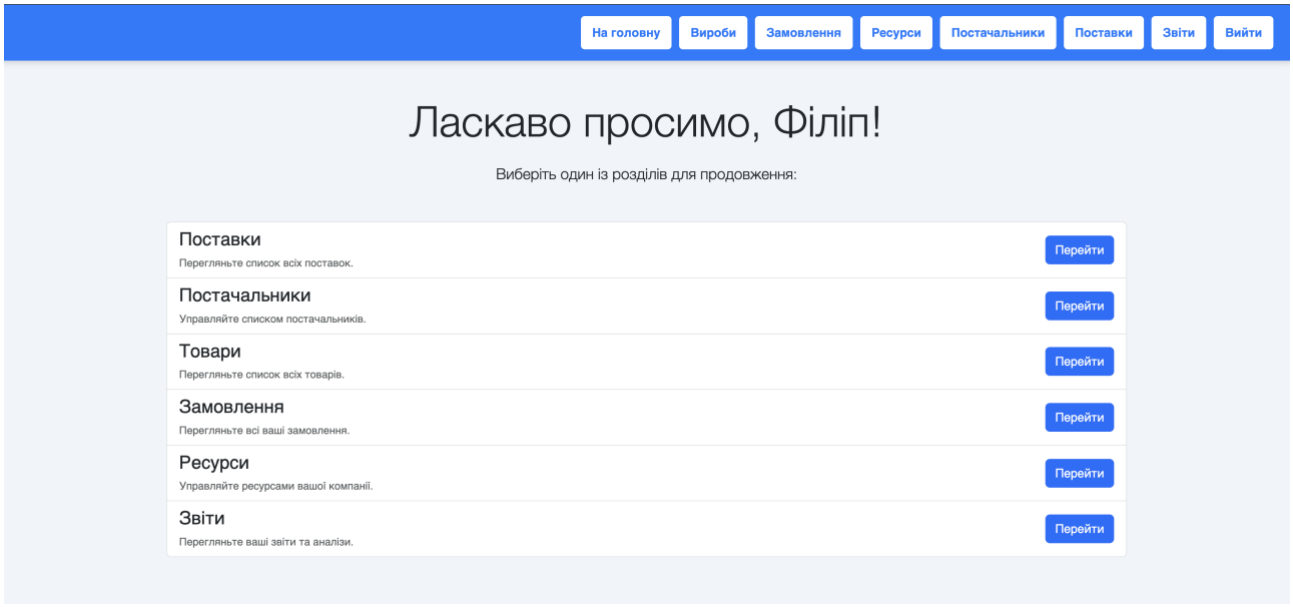


Рисунок 3.13 – Зовнішній вигляд меню системи

Як можна побачити, меню розроблено в мінімалістичному стилі із використанням зазначеної вище кольорової схеми, і з використанням зручного меню вгорі, яке забирає необхідність повертатися на головну сторінку для переходу в інший розділ підсистеми.

Задля забезпечення інтуїтивності дизайну, для інших вкладок було обрано табличний вигляд, де дизайн спрямований на створення секційного інтерфейсу, який дає змогу швидше орієнтуватися користувачу. На рисунку 3.14 наведено приклад однієї з таких вкладок.

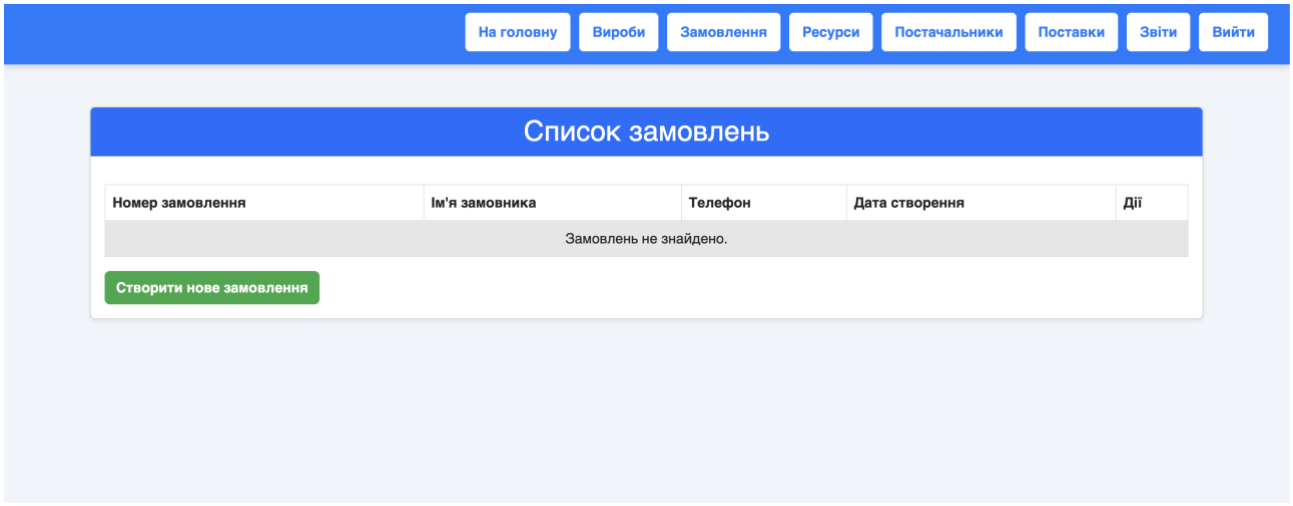


Рисунок 3.14 – Система замовлень з інтерфейсом у табличному дизайні

Такий варіант відображення структурованої інформації було обрано з метою забезпечення максимальної швидкодії працівників. Варто відзначити, що дизайн аналогічний для більшості вкладок, що теж є кроком для забезпечення максимальної швидкодії при роботі з розроблюваною ERP системою. На рисунку 3.15 зображено форму, завдяки якій працівник може зберігати інформацію в додатку, зображено на прикладі додавання замовлення.

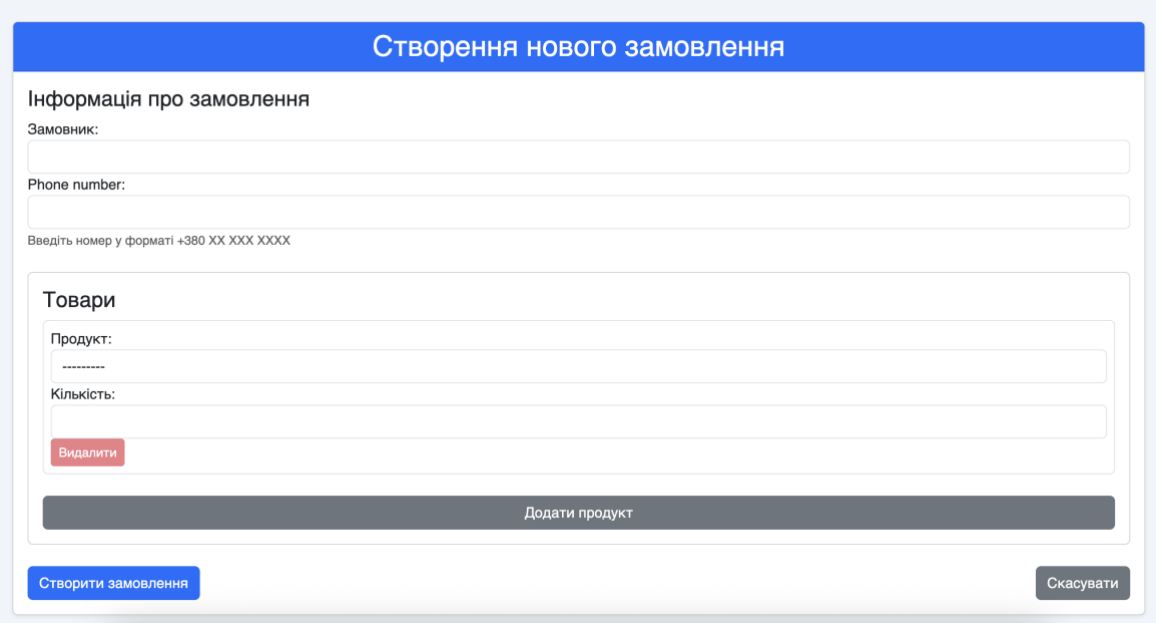


Рисунок 3.15 – Графічний інтерфейс користувача для створення нового замовлення

Всі наступні системи розроблені в аналогічному дизайні, який використовує вищезазначені кольори та елементи дизайну. На рисунку 3.16 зазначено зовнішній вигляд панелі адміністратора, яка була налаштована, та запрограмована на відображення усієї необхідної інформації та керування нею.

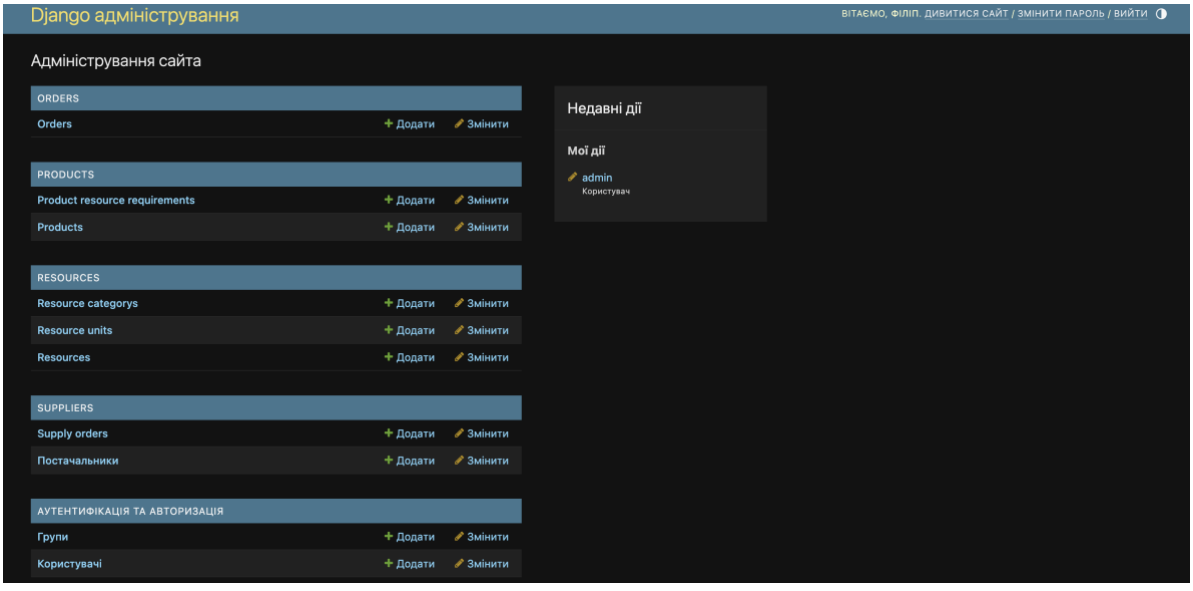


Рисунок 3.16 – Вбудована в Django панель адміністратора

Дана панель є однією з ключових підсистем системи. Вона дає можливість повністю керувати проектом, робити будь які зміни в базі даних, впливати на користувачів, їхні можливості, тощо. Код, який використовувався для налаштування панелі адміністратора зазначено в додатку Б.

3.3 Охорона праці

У процесі розроблення та впровадження ERP-системи управління ресурсами складу належну кількість уваги необхідно приділити питанням безпеки праці, адже інтеграція цифрових технологій у виробництво без належного супроводу спеціалістами може призвести до великих ризиків.

Робота з інформаційними системами супроводжується електромагнітним випромінюванням, високим емоційним навантаженням через необхідність обробки значного обсягу даних, а також потребою в постійному зоровому та інтелектуальному напруженні організму.

Не менш важливим є і фактор ергономіки: неправильна організація робочого місця може спричинити хронічні захворювання та зниження загальної працездатності оператора, наприклад, через погане положення столу, крісла, тощо. Водночас складські приміщення, якими керує ERP-система, залишаються джерелом підвищеної небезпеки через використання підйомних механізмів, стрічкових конвеєрів та іншого складського обладнання. В разі помилок у логіці системи або збоїв у її роботі можуть виникнути аварійні ситуації, такі як одночасна подача команд на переміщення товарів у кількох напрямках або неправильний розрахунок залишків, що ускладнює роботу персоналу і створює додаткові навантаження на систему та персонал. Для запобігання таким ситуаціям необхідно ще на етапі проектування ERP-системи враховувати чинні стандарти в галузі охорони праці, такі як Закон України Про охорону праці [18], санітарно-гігієнічні норми, правила протипожежної безпеки та безпеки при

роботі з електронікою. Працівники, що взаємодіють із системою, повинні проходити обов'язкове навчання, інструктажі з техніки безпеки, а також періодичні медичні огляди. ІТ-персонал має бути поінформований про правила безпечної експлуатації серверного обладнання, якщо воно розміщується в окремих приміщеннях із підвищеними вимогами до навколишнього середовища, таких як вентиляція, температура, вологість, тощо.

Додатковою перевагою сучасних ERP-систем є можливість інтеграції функцій контролю за станом охорони праці, а саме ведення електронних журналів інструктажів, облік медичних оглядів, фіксація трагічних випадків та порушень техніки безпеки, автоматичне нагадування про терміни проходження навчання або заміну захисного обладнання.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було досягнуто головну мету – спроектовано та створено ERP систему, яка відповідає усім поставленим вимогам, яка враховує проведений детальний аналіз існуючих систем, виявлені в них недоліки та підкреслені переваги. Розроблена система при інтеграції у виробництво дійсно підвисить продуктивність персоналу, та зменшить витрати часу на різні, пов’язані з ERP операції, що і було метою кваліфікаційної роботи.

До задач, що були вирішені під час кваліфікаційної роботи належить детальний аналіз предметної області, аналіз роботи підприємств в Індустрії 4.0 та 5.0, створено унікальну ERP систему спеціально для внутрішнього ринку, яка практично готова до реалізації підприємствам України. Окрім розробки було проведена оцінка розробки програмного забезпечення, що дає можливість правильно оцінити вартість розробленої системи для справедливої реалізації. Також проведено різного роду розрахунки, завдяки яким було визначено, що стратегічний резерв ресурсів на підприємстві має сягати 20% від поточної потреби виробництва.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. 29 с;

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти денної і заочної форми навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд.: І.Ш. Невлюдов, О.І. Филипенко, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2023. 64 с;

3. Навчальний посібник з підготовки кваліфікаційної роботи бакалавра для здобувачів вищої освіти денної і заочної форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми «Автоматизація та комп'ютерно-інтегровані технології» : Навчальний посібник / І. Ш. Невлюдов, В.А. Андрусевич, О. В. Токарєва, С. П. Новоселов, О. В. Сичова. – Харків : Видавництво Іванченка І. С., 2022. 151 с.;

4. What is ERP? History, benefits, modules - whatfix. The Whatfix Blog | Drive Digital Adoption. URL: https://whatfix.com/blog/erp/;

5. Що таке ERP (enterprise resource planning)? – SMART localization. SMART Localization. URL: https://bc.smart-it.com/news-and-articles/what-is-erp/;

6. Учасники проектів Вікімедіа. Планування ресурсів підприємства – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Планування_ресурсів_підприємства;

7. A brief history of ERP. Genius ERP. URL: https://www.geniuserp.com/resources/blog/a-brief-history-of-erps/;

8. ERP: through the decades. Oracle NetSuite. URL: https://www.netsuite.com /portal /resource/articles/erp/erp-history.shtml;

9. Hayes A. Manufacturing resource planning (MRP II): definition and example. Investopedia. URL: https://www.investopedia.com/terms/m/manufacturing-resource-planning.asp;

10. Пустовойтенко Ф. А. Аналіз існуючих рішень серед систем планування ресурсів підприємства та їх проблематики. CITAR 25. 2025. С. 97–100. URL: https://tapr.nure.ua/dijalnist-kafedri/naukova-robota/computer-integrated-technologies-automation-and-robotics;

11. Welcome to Python.org. Python.org. URL: https://www.python.org/;

12. Django. Django Project. URL: https://www.djangoproject.com/;

13. PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/;

14. Docker: Accelerated Container Application Development. Docker. URL: https://www.docker.com/;

15. Nginx. nginx. URL: https://nginx.org/;

16. Aqeel O. A Beginner's Guide to HTML, CSS, and JavaScript | HackerNoon. HackerNoon - read, write and learn about any technology. URL: https://hackernoon.com/a-beginners-guide-to-html-css-and-javascript;

17. Як кольори впливають на роботу?. Yaware.TimeTracker - онлайн система обліку часу і продуктивності. URL: https://yaware.com.ua/ua/blog/kak-tsveta-v-ofise-vliyayut-na-produktivnost-rabotyi-sotrudnikov/;

18. Про охорону праці : Закон України від 14.10.1992 № 2694-XII : станом на 4 квіт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text.