

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

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

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «магістр»

**на тему: «Проектування та розробка інформаційної системи роздрібних
продажів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ»**

Виконала: студентка 6144 групи

_____ *Кухаренко В.С.*
(підпис)

Керівник роботи:

СТ. викладач
(посада, науковий ступінь вчене звання)

_____ *Беркунський Є.Ю.*
(підпис)

м. Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 126 "Інформаційні системи та технології"

Освітня програма "Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Чернова ЛБ.С.

" ____ " _____ 2023 р.

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

Студентці Кухаренко Вікторії Сергіївни

(прізвище, ім'я, по батькові)

1. Тема роботи: Проектування та розробка інформаційної системи роздрібних продажів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ

Керівник роботи Беркунський Є.Ю.

Затверджені наказом ректора № 1098-уч. від "18" вересня 2023 року.

2. Термін подання роботи: 15 грудня 2023 р.

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

4. Перелік питань, що належать до розробки (найменування розділів); Дослідження та аналіз предметної області, Розробка технічного проекту інформаційної системи, Реалізація проекту

5. Перелік презентаційних матеріалів Організаційна структура підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ, Паспорт бізнес-процесу «Оновлення наявності товару для продажу, модель IDEF0 «ЯК Є». модель IDEF0 «ЯК ПОВИННО БУТИ», діаграма прецедентів, діаграма концептуальної моделі, діаграма логічної моделі, діаграма фізичної моделі, діаграма послідовності, результати реалізації проекту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайдаєнко О.В.		

2	Беркунський Э.Ю.		
3			
4			

7. Дата видачі завдання 05 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження і аналіз предметної області		
2	Розробка технічного проекту інформаційної системи		
3	Реалізація проекту інформаційної системи		
4	Вирішення питань охорони праці		
5	Вирішення питань фінансів		
6	Вирішення питань охорони навколишнього середовища		
7	Виконання графічних документів		
8	Оформлення пояснювальної записки		
9	Подання дипломної роботи (ДР) на кафедральний захист		
10	Подання ДР рецензенту		
11	Подання ДР на державний захист		

Студент

(підпис)

Кухаренко Вікторія Сергіївна
(ПІБ)

Керівник роботи

(підпис)

Беркунський Євген Юрійович
(ПІБ)

АНОТАЦІЯ

У даній дипломній роботі розглядається завдання оновлення та реалізації новою інформаційної системи для підприємства роздрібного продажу товарів ЕЛІЗТАН АЛЬЯНС КОМПАНІ.

Робота викладена на 130 сторінках, містить 6 розділів, 52 рисунків, 23 таблиці, 4 додатки та список використаної літератури з 32 найменувань.

Робота виконана українською мовою.

ABSTRACT

This current degree work have the task of creation and implementation of a new information system for the enterprise of retail sales of goods ELIZTAN ALLIANCE COMPANY.

The work is laid out on 130 pages, contains 6 subsections, 52 drawing, 23 tables, 4 applications and 32 referes.

The work is done in Ukrainian.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1 Дослідження та аналіз предметної галузі.....	8
1.2 Функції та завдання організації.....	9
1.3 Структура організації.....	10
1.4 Аналіз існуючого інформаційного та програмного забезпечення	13
1.5 Постановка задачі.....	14
1.6 Аналіз бізнес-процесів «ЯК Є»	15
1.7 Виявлення проблематики за допомогою SWOT аналізу	29
1.8 Моделювання бізнес-процесу «ЯК ПОВИННО БУТИ»	32
1.9 Формування вимог до майбутньої інформаційної системи	36
РОЗДІЛ 2. РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ ІС.....	40
2.1 Інформаційне забезпечення.....	40
2.1.1 Розробка та аналіз системних рішень	40
2.1.2 Розробка концептуальної моделі даних	44
2.1.3 Розробка логічної моделі даних.....	51
2.1.4 Розробка фізичної моделі даних	52
2.1.5 Розробка діаграма послідовності	53
2.2 Рішення з математичного забезпечення.....	55
2.3 Програмне забезпечення	58
2.4 Технічне забезпечення проекту	64
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІС	66
3.1 Реалізація – сервер	66
3.2 Реалізація – клієнт.....	70
РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ ПІД ЧАС НЕЇ	82
4.1 Поняття охорона праці.....	82
4.2 Аналіз небезпечних факторів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ.....	83
4.3 Заходи зменшення впливу шкідливих факторів.....	85

РОЗДІЛ 5. ПРО ОХОРОНУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА	88
5.1 Актуальність питання, що стосується охорони навколишнього середовища.....	88
5.2 Аналіз факторів шкідливого впливу на екологію бізнес-процесу «Оновлення наявності товару для продажу».....	89
5.3 Заходи запобігання забруднення навколишнього середовища	90
РОЗДІЛ 6. РОЗРАХУНОК ВИТРАТ НА ВПРОВАДЖЕННЯ ІС	93
6.1 Фінансова актуальність розробленої ІС.....	93
6.2 Розрахунок витрат на створення і впровадження системи	94
ВИСНОВОК.....	98
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	99
Додаток А – Технічне завдання	103
Додаток Б – Код створення таблиць postgresql	105
Додаток В – Код програми backend.....	113
Додаток Г – Код програми frontend.....	122

ВСТУП

Аналіз різних процесів – це дуже важлива складова нашого життя. Так, наприклад, у своїй роботі «Переваги rpg жанру відеоігор» для конференції «Стан, досягнення та перспективи інформаційних систем і технологій» від Одеського національного технологічного університету Інформатики і прикладних знань, я виявила переваги та проаналізувала вплив відеоігор в наш час. Але існує і інший аналіз, а саме виявлення проблем роботи організацій.

У наш час важлива ідея автоматизації усіх процесів. Час – це важливий ресурс, який потрібно якомога більше зберігати. Завжди час – це гроші. Автоматизація робочих процесів також передбачає збереження великої кількості інформації. Для цього і існують інформаційні технології.

Буває таке, що у підприємства уже існує якась інформаційна система, але вона працює некоректно. Наприклад, відбувається втрата невизначеної кількості даних, передбачається використання великої кількості технологій, які гальмують усі бізнес-процеси у системі. У цьому випадку виникає питання та запит щодо проектування та реалізації такої інформаційної системи (далі ІС), які ці помилки зможе виправити.

Тема дипломної роботи – «Проектування та розробка інформаційної системи роздрібних продажів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ».

Мета роботи: визначити проблемні сторони підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ, а саме бізнес-процесів пов'язаних з логістикою та продажем товарів у торгових точках «Табако Маркет».

Завдання: проаналізувати компанію, визначити її функції та завдання, проаналізувати існуючу інформаційну систему за допомогою моделі «ЯК Є», виявити проблемні сторони та спроектувати модель «ЯК ПОВИННО БУТИ», розробити концепт майбутньої ІС – технічні, математичні, програмні сторони, застосувати програмне забезпечення, для реалізації складових інформаційної системи.

Розробка ІС передбачає уникнення виявлених під час аналізу системи підприємства проблем та покращення її роботи.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.

1.1 Дослідження та аналіз предметної галузі

Торгівля – це одна з головних складових нашого життя. Вона присутня всюди у багатьох наших діях. Здійснюючи покупки в будь-якому магазині, ми стикаємося з торговим бізнесом. Щоразу, коли ми контактуємо з продавцем у магазині продуктів біля будинку, ми стаємо на пару хвилин учасником величезного бізнес-процесу, який починається з ідеї створення цієї торгової точки. Продавець контактує з нами, а перед цим він пов'язаний з зовсім іншими людьми, які забезпечують повноцінне функціонування магазину, в якому ми здійснили покупки.

Бізнес-процес – це сукупність різних видів діяльності, які створюють результат, який має цінність для споживача. Кожен процес має входи і виходи [1].

Бізнес-процес у торгівлі – це безперервна справа, яка закінчиться лише тоді, коли перестане існувати компанія. Їх існує безліч і всі вони тісно взаємопов'язані між собою. Наприклад, звітність головного бухгалтера перед директором захисту зарплатні і є бізнес-процесом; а також передача товарів водієм продавцю в торговій точці. Усі вони тісно взаємопов'язані між собою.

Виробничі процеси закупівлі і логістики вимагають вирішення наступних завдань:

- Встановлення, планування та управління потребами матеріальних ресурсів (а саме товарів);
- Контроль запасів ресурсів;
- Планування використання матеріальних ресурсів упродовж невизначеного часу;
- Взаємодія складових організації щодо зв'язку товаробігу.

Інформаційна система — сукупність організаційних, технічних, програмних і інформаційних засобів, що об'єднані в єдину систему для реалізації мети і виконання задач [2].

Елементами інформаційної системи є:

- інформаційна модель, що є сукупністю правил і алгоритмів;
- регламент розвитку інформаційної моделі і правил по внесенню до неї змін;
- програмний комплекс, конфігурація якого відповідає вимогам інформаційної моделі;
- апаратно-технічна база (комп'ютери, периферійні пристрої, канали зв'язку, системне ПЗ, СУБД тощо).

Використання інформаційних систем – це невід'ємна частина автоматизації робочих бізнес-процесів. Жодне виробництво зараз не обходиться без втручання інформаційних технологій.

1.2 Функції та завдання організації

«ЕЛІЗТАН АЛЬЯНС КОМПАНІ» – товариство з обмеженою відповідальністю, основним видом діяльності якого є роздрібна торгівля в неспеціалізованих магазинах переважно продуктами харчування, включаючи напої, та тютюновими виробами. Вона має свою торгову марку «Табако Маркет», під якою продає продукцію різного виду по торгових точках, які розташовані по всьому Миколаєву.

Основними завданнями підприємства є:

- Надання послуг клієнтам у сфері роздрібної торгівлі тютюнових та інших виробів;
- Надання послуг клієнтам у сфері оптової торгівлі тютюнових та інших виробів;
- Надання послуг посередництва у торгівлі товарів різного типу;
- Надання послуг мобільного харчування на торгових точках «Табако Маркет».

Основними функціями підприємства є:

- Діяльність посередників у торгівлі товарами широкого асортименту;
- Роздрібна та оптова торгівля тютюновими виробами;
- Роздрібна та оптова торгівля алкогольними напоями;

- Роздрібна та оптова торгівля іншими виробами;
- Роздрібна торгівля товарами, що не використовуються, у спеціалізованих магазинах;
- Складське господарство;
- Послуги мобільного харчування (продаж кави, хот-догів тощо).

1.3 Структура організації

Структуру організації представлено на рисунку 1.1.

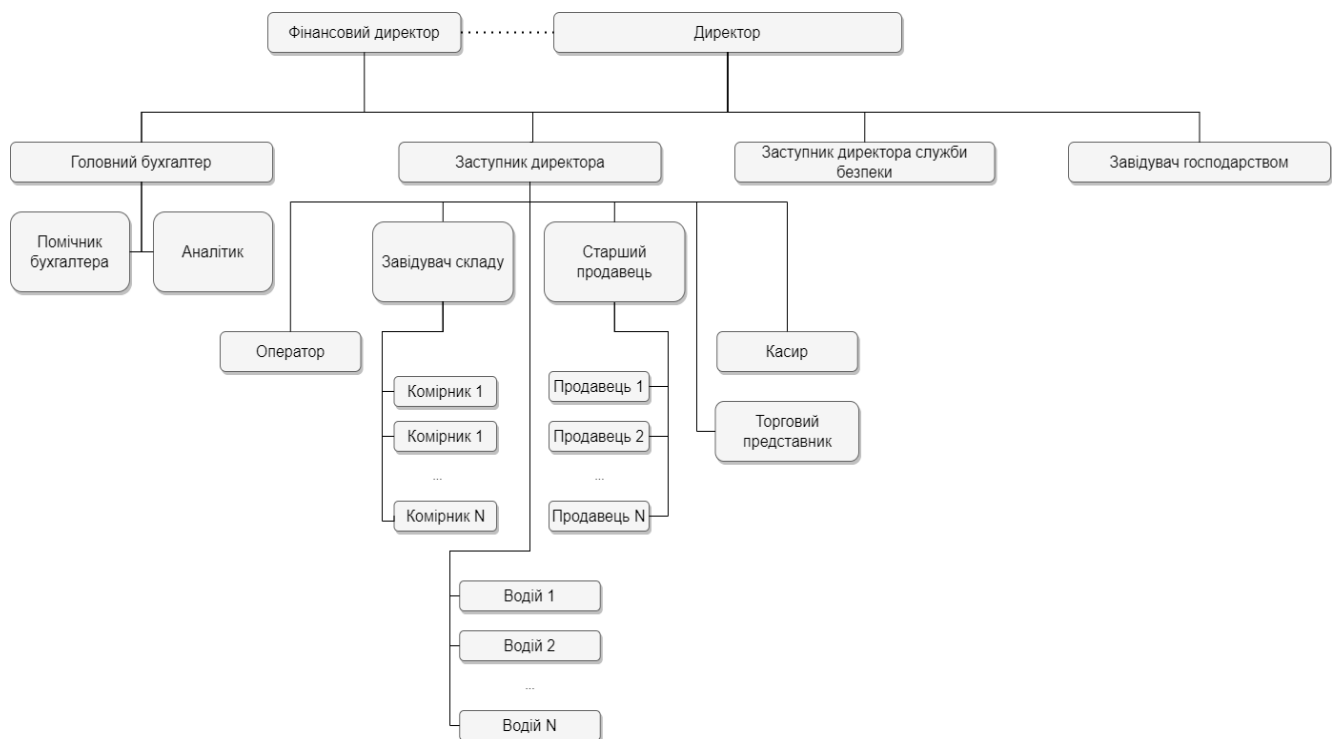


Рис. 1.1. Структура організації

Опишемо структурну організацію підприємства, кадри та їх функції.

Директор – один із головних людей компанії. Він займається створенням та розвитком ідеології компанії, визначає цілі та завдання, виявляє стратегічні напрями маркетингу, організовує планування та управління підприємством. Ще займається координацією діяльності підрозділів, перед ним звітують інші посади (наприклад, захист заробітної плати наприкінці кожного місяця), вирішує оперативні проблеми та відповідає за прибуток підприємства та її життєздатність.

Фінансовий директор – має більший вплив у компанії, практично на рівні з основним директором підприємства. Він займається забезпеченням необхідної

кількості фінансів для ведення бізнесу, забезпечує керівництво та інших співробітників інформацією для прийняття стратегічних рішень, забезпечує економічну безпеку підприємства та підтримкою позитивного іміджу компанії. Основною функцією фінансового директора є складання прогнозного плану та контролю виконання його в компанії.

Головний бухгалтер – підпорядковується фінансовому та головному директору. Жодне підприємство не обходиться без бухгалтерського відділу, яким керує головний бухгалтер. Він займається забезпеченням ведення бухгалтерського обліку, забезпечує перелік податків і зборів, займається оформленням договорів із постачальниками товарів, контролює веденням касових операцій, трудових та фінансових ресурсів, здійснює контроль за бухгалтерським обліком усіх торгових точок та розрахунки з іншими кредиторами відповідно до договірних зобов'язань.

Помічник бухгалтера – підпорядковується головному бухгалтеру. Займається всіма дочірніми процесами бухгалтерського спрямування, що не встигає головний бухгалтер. Він виконує роботу на дорученій ділянці функціонального спрямування, займається веденням документації та архіву, оформляє платіжні доручення, бере участь у проведенні інвентаризації, ставить накладні на прихід, проводить звітність з фінансових операцій.

Аналітик – підпорядковується головному бухгалтеру. Він займається аналізом потреб клієнта, вимог та покращень, працює зі статистикою структури та обсягів продажів, займається відстеженням популярних товарних позицій, прогнозує попит і складає план продажу та закупівель, складає звітність наданих вище функцій.

Заступник директора – підпорядковується головному директору. Він розробляє та узгоджує план розвитку підприємства, контролює та управляє посадами нижче його за статусом, їхню роботу, організовує поточну організаційно-виконавчу роботу всієї фірми, затверджує штат та займається організацією роботи у фірмі зі стимулювання персоналу. Основним його

завданням є забезпечення контролю основних напрямів розвитку підприємства та звітність перед головним директором.

Завідувач складу – підпорядковується заступнику директора. Він керує роботою складу з прийому, зберігання та відпустки товарно-матеріальних цінностей, організовує проведення вантажно-розвантажувальних робіт на складі з дотриманням норм, правил та інструкцій з охорони праці. Основною функцією завідувача є збирання та зберігання інформації пов'язаних з функціонуванням складу як важливої кадрової одиниці підприємства.

Оператор – підпорядковується заступнику директора. Займається введенням довідкової інформації та документів у єдину систему, обробкою даних, підготовкою звітної документації для керівництва.

Старший продавець – підпорядковується заступнику директора. Він керує колективом продавців, планує їх роботу, веде необхідну документацію торгових точок, працює з клієнтами у торгових залах, стежить за прямими продажами, виконує плани продажу. Основною функцією старшого продавця є контроль асортименту продукції у всіх торгових точках.

Касир – підпорядковується заступнику директора. Він здійснює операції з прийому та обліку коштів та цінних паперів, веде на основі прибуткових та видаткових документів касову книгу, складає описи старих купюр, а також відповідну документацію, вживає всіх заходів для забезпечення збереження коштів. Основною функцією є визначення зв'язку з банком та складання касової звітності.

Торговий представник – підпорядковується заступнику директора. Він навідується на місця продажу товарів та збирається інформації про те, що і як продається. Основною функцією є представлення компанії різних рівнях функціонування підприємства.

Продавці – підпорядковуються старшому продавцю. Вони взаємодіють з покупцями, представляють, презентують товар, а також надають послуги, запровадять клієнтську базу та здійснюють інформаційний зворотний зв'язок із

клієнтами. Основною функцією є надання всіх видів послуг, які пропонують компанія та торгова марка.

Комірники – підпорядковуються завідувачу складу. Розподіл обов'язків між кадровими одиницями – прийом, організація, зберігання та видача різних матеріальних цінностей (у випадку з підприємством тютюнових виробів і т), стеження за правильністю оформлення заявок на ввезення та вивезення товарної продукції.

Водії – підпорядковуються заступнику директора. Їх основне завдання – транспортування товарів у зазначені торгові точки у певний час. Додаткова функція – розвезення потреб підприємства у залежність від ситуації.

Заступник директора служби безпеки – підпорядковується головному директору. Він забезпечує захист об'єктів компанії від крадіжок та інших злочинів, займається організацій безпеки, здійснює перевірку та оцінку лояльності службовців об'єктів, що охороняються. Основною функцією є розробка та здійснення керівництва заходу щодо безпеки об'єктів, що їм належать.

Завідувач господарства – підпорядковується головному директору. Він забезпечує господарське обслуговування та належний стан торгових точок, займається організацією контролю раціональної витрати матеріалів та засобів, пов'язаних з господарством.

1.4 Аналіз існуючого інформаційного та програмного забезпечення

Розглядається інформаційна підсистема, яка пов'язана з товарообігом у компанії. Взаємодія кадрів та існуючої інформаційної системою представлено на рисунку 1.2.

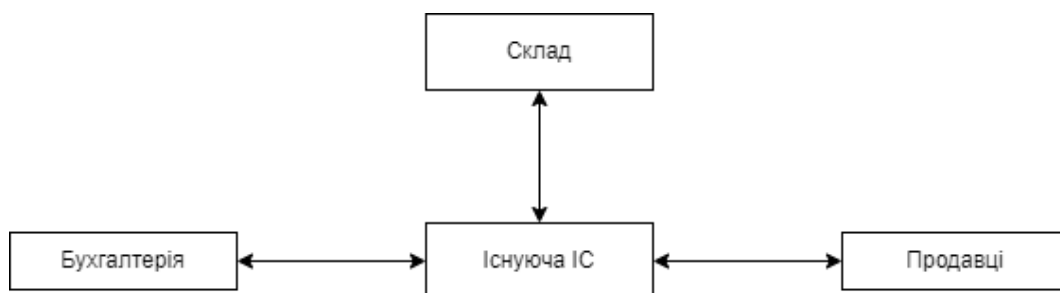


Рис. 1.2. Взаємодія кадрів та існуючої ІС

Для взаємодії використовується таке програмне забезпечення:

Motive Force Kvinta – інтегрована система, що дозволяє використовувати комплекс таких інструментів, як бухгалтерські системи та інші додатки. Пропонується ряд модулів для інтеграції з такими системами, як «1С-Бухгалтерія», «Клієнт-банк», відео спостереження за касовими місцями, використання банківських терміналів, терміналів збору даних, смартфонів, планшетів. Вона ґрунтується на використанні штрихового кодування товару; підтримує технологію роботи як локального ресторану, магазину тощо, так і мережі роздрібних підприємств з урахуванням віддаленості офісу, складів; приділяється особлива увага маркуванню товару, перевірці відповідності його штрих-коду тощо [3].

Gmail – безкоштовна служба електронної пошти від компанії Google. Вона надає доступ до поштових скриньок через вебінтерфейс і за протоколами POP3, SMTP, IMAP. Головною перевагою цієї служби є зберігання 15 гігабайтів простору. Gmail побудований на JS, застосовується технологія AJAX (оновлення сторінок без перенавантаження) [4].

Microsoft Office – офісний пакет, створений корпорацією Microsoft. До складу цього пакету входить програмне забезпечення для роботи з різними типами документів: текстами, електронними таблицями, презентаціями, базами даних тощо [5].

Telegram – глобально доступний багатоплатформовий зашифрований фріміум клауд-месенджер. Месенджер також надає опціональні наскрізні зашифровані чати, більш відомі як секретні чати та відеодзвінки, VoIP, обмін файлами та деякі інші функції [6].

1.5 Постановка задачі

Предметом дослідження буде встановлено: відділ продавців, відділ бухгалтерії, склад, оператор та інформаційна підсистема.

Об'єктом дослідження буде встановлено: бізнес-процес взаємодії заданих предметів дослідження та програмного забезпечення під час оновлення наявності товару.

Метою роботи буде встановлено: уникнення проблем, підвищення ефективності роботи та автоматизація робочого бізнес-процесу за рахунок перепроєктування існуючої робочої інформаційної системи.

1.6 Аналіз бізнес-процесів «ЯК Є»

Головною метою встановлення проблемних сторін підприємства було підвищення ефективності роботи та автоматизація робочого бізнес-процесу за рахунок перепроєктування існуючої робочої інформаційної системи. Назва цього бізнес-процесу «Оновлення наявності товару для продажу».

Проведемо декомпозицію бізнес-процесу на складові, а саме:

1. Оновлення наявності товару для продажу.

1.1. Контроль товару у торговій точці.

- 1.1.1. Аналіз потреб;
- 1.1.2. Контроль кількості товару/залишків;
- 1.1.3. Контроль ціноутворення товару.

1.2. Замовлення товару у постачальника.

- 1.2.1. Звіряння проведеного аналізу старшим продавцем;
- 1.2.2. Пошук необхідних товарів на джерелах постачальників;
- 1.2.3. Оформлення замовлення;
- 1.2.4. Звіряння дат отримання замовлення передачі інформації складу.

1.3. Прийом товару складом.

- 1.3.1. Контроль кількості товару, що прибув;
- 1.3.2. Контроль якості товару, що прибув;
- 1.3.3. Отримання накладної від оператора та його друкування за допомогою принтера;
- 1.3.4. Оформлення та підпис комірником накладної.

1.4. Транспортування товару до магазину.

- 1.4.1. Передача товару транспортувальнику (водію);

1.4.2. Переміщення товару до зазначеної торгової точки;

1.4.3. Прийом товару продавцем.

1.5. Занесення накладної до бази.

1.5.1. Надсилання сканованої копії накладної помічнику.

1.5.2. Передача накладної до архіву бухгалтерії;

1.5.3. Занесення даних накладної.

1.6. Занесення нової ціни до бази.

1.6.1. Передача інформації про нову та стару ціну товару.

1.6.2. Занесення даних ціни.

Створимо дерево цілей на основі ідентифікації бізнес-процесів, які були виявлені вище. Дерево цілей представлено на рисунку 1.3.

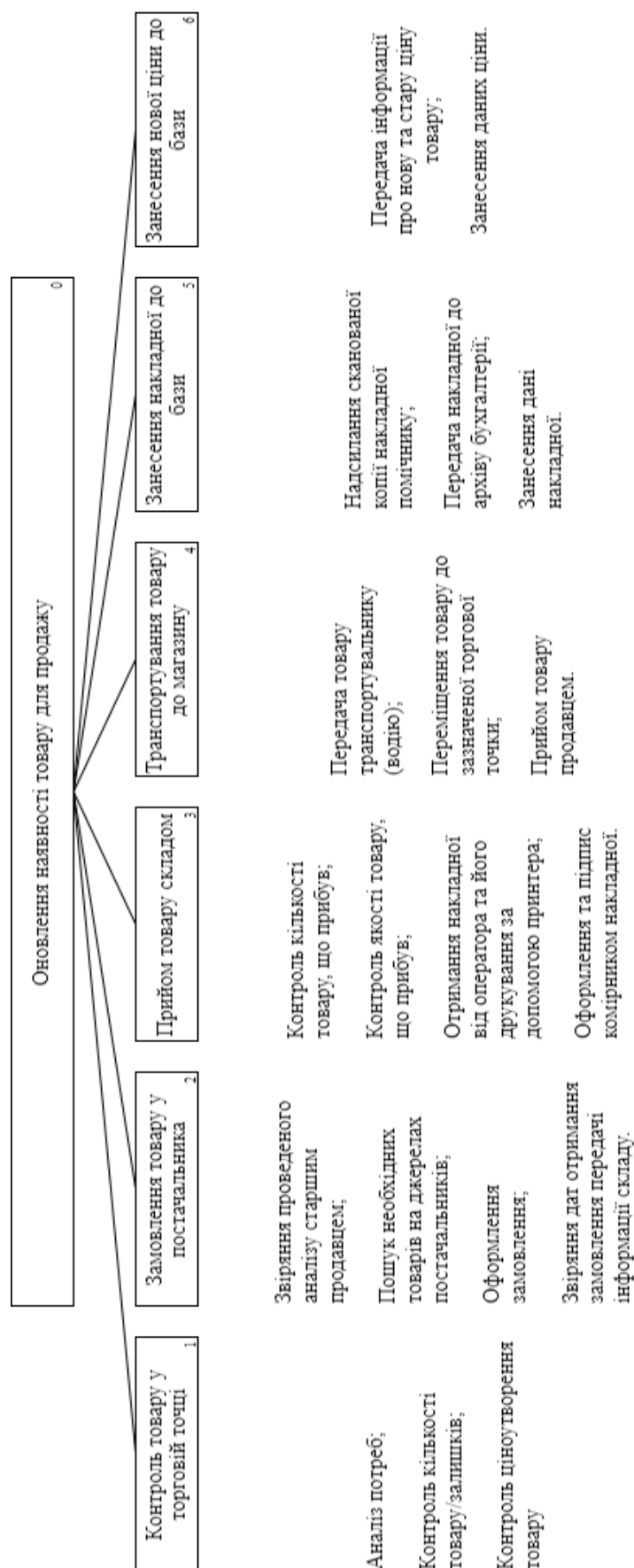


Рис. 1.3. Дерево цілей бізнес-процесу «Оновлення наявності товару для продажу»

Розпишемо більш детально бізнес-процес та його декомпозиції за допомогою паспортів бізнес-процесів для формалізації.

Паспорт бізнес процесу «Оновлення наявності товару для продажу» представлено у таблиці 1.1.

Таблиця 1.1. Паспорт бізнес процесу «Оновлення наявності товару для продажу»

Найменування процесу	Оновлення наявності товару для продажу
Індекс	1
Призначення	Перегляд та встановлення наявності товарів для продажу в торговій точці
Результати	Наявність фізичного товару та інформаційного матеріалу про нього в базі та в архіві.
Виходи бізнес-процесу	
Матеріальні	Товар
Інформаційні	Звіт про внесені дані
Входи бізнес-процесу	
Матеріальні	Товар
Інформаційні	Потреби
Власник бізнес-процесу	Торгова точка
Учасники бізнес-процесу	
Персонал	Помічник бухгалтера, продавці, старший продавець, завідуючий складом, комірник, оператор, водій
Засоби реалізації	MForce, Telegram, Microsoft Office, принтер, ІС постачальника, Gmail, стандарти занесення, документація
Споживач	Склад, торгова точка, архів бухгалтерії
Постачальник	Стороні організації-постачальники

1	2
Попередній процес	-
Наступний процес	-
Записи у ході виконання процесу.	-

Паспорт бізнес процесу «Контроль товару у торговій точці» представлено у таблиці 1.2.

Таблиця 1.2. Паспорт бізнес процесу «Контроль ціноутворення товару»

Найменування процесу	Контроль товару у торговій точці
Індекс	1.1
Призначення	Проведення контролю щодо наявності певного товару у торговій точці.
Результати	Встановлення товарного обігу у торговій точці.
Виходи бізнес-процесу	
Матеріальні	-
Інформаційні	Результат проведеного аналізу
Входи бізнес-процесу	
Матеріальні	-
Інформаційні	Потреби
Власник бізнес-процесу	Торгова точка
Учасники бізнес-процесу	
Персонал	Продавці, старший продавець.
Засоби реалізації	Документація, MForce
Споживач	Торгова точка
Постачальник	-
Попередній процес	-
Наступний процес	Замовлення товару у постачальника
Записи у ході виконання	Результати аналізу товарів у документацію

Паспорт бізнес процесу «Замовлення товару у постачальника» представлено у таблиці 1.3.

Таблиця 1.3. Паспорт бізнес процесу «Звіряння дат отримання замовлення передачі інформації складу»

Найменування процесу	Замовлення товару у постачальника
Індекс	1.2
Призначення	Замовлення товарів від постачальників для поповнення наявності їх у магазині.
Результати	Замовлення товару.
Виходи бізнес-процесу	
Матеріальні	-
Інформаційні	Оформлена постачальником накладна
Входи бізнес-процесу	
Матеріальні	-
Інформаційні	Результати проведеного аналізу
Власник бізнес-процесу	Торгова точка
Учасники бізнес-процесу	
Персонал	Продавці, старший продавець, оператор, склад
Засоби реалізації	Документація, ІС постачальника
Споживач	Торгова точка, склад
Постачальник	-
Попередній процес	Контроль товару у торговій точці
Наступний процес	Прийом товару складом
Записи у ході виконання процесу	Результати аналізу товарів у документацію

Паспорт бізнес процесу «Прийом товару складом» представлено у таблиці 1.4.

Таблиця 1.4. Паспорт бізнес процесу «Прийом товару складом»

Найменування процесу	Прийом товару складом
Індекс	1.3
Призначення	Прийом товару складом для подальшого транспортування в торгові точки.
Результати	Товар знаходиться на складі.
Виходи бізнес-процесу	
Матеріальні	-
Інформаційні	Підписана комірником накладна.
Входи бізнес-процесу	
Матеріальні	Товар
Інформаційні	Оформлена постачальником накладна
Власник бізнес-процесу	Склад
Учасники бізнес-процесу	
Персонал	Завідуючий складом, комірники, водій, оператор.
Засоби реалізації	Принтер, Gmail, Microsoft Office
Споживач	Склад
Постачальник	Постачальник товару
Попередній процес	Замовлення товару у постачальника
Наступний процес	Транспортування товару до магазину
Записи у ході виконання процесу	-

Паспорт бізнес процесу «Транспортування товару до магазину» представлено у таблиці 1.5.

Таблиця 1.5. Паспорт бізнес процесу «Прийом товару продавцем»

Найменування процесу	Транспортування товару до магазину
Індекс	1.4
Призначення	Передача інформації та матеріального товару на торгову точку.
Результати	Результат передачі товару торговій точці.

1	2
Виходи бізнес-процесу	
Матеріальні	Товар
Інформаційні	Накладна
Входи бізнес-процесу	
Матеріальні	Товар
Інформаційні	Накладна
Власник бізнес-процесу	Торгова точка
Учасники бізнес-процесу	
Персонал	Продавці, водій, склад
Засоби реалізації	-
Споживач	Торгова точка, водій
Постачальник	Склад, водій
Попередній процес	Прийом товару складом
Наступний процес	Занесення накладної до бази, занесення нової ціни до бази
Записи у ході виконання процесу	-

Паспорт бізнес процесу «Занесення накладної до бази» представлено у таблиці 1.6.

Таблиця 1.6. Паспорт бізнес процесу «Занесення дані накладної»

Найменування процесу	Занесення накладної до бази
Індекс	1.5
Призначення	Задokumentовано інформації в базі.
Результати	Задokumentовано інформації в базі.
Виходи бізнес-процесу	
Матеріальні	-
Інформаційні	Звіт про занесення даних

1	2
Входи бізнес-процесу	
Матеріальні	Накладна
Інформаційні	Накладна
Власник бізнес-процесу	Торгова точка
Учасники бізнес-процесу	
Персонал	Продавець, помічник бухгалтера
Засоби реалізації	Telegram, MForce, стандарти занесення
Споживач	Бухгалтерія
Постачальник	Продавець
Попередній процес	Транспортування товару до магазину
Наступний процес	-
Записи у ході виконання процесу	Інформація в базу даних.

Паспорт бізнес процесу «Занесення нової ціни до бази» представлено в таблиці 1.7.

Таблиця 1. 7. Паспорт бізнес процесу «Занесення нової ціни до бази»

Найменування процесу	Занесення нової ціни до бази
Індекс	1.6
Призначення	Контроль ціни товару.
Результати	Оновлення інформації о ціні товару.
Виходи бізнес-процесу	
Матеріальні	-
Інформаційні	Звіт про занесення даних
Входи бізнес-процесу	
Матеріальні	-
Інформаційні	Фотографія/текст про нову ціну
Власник бізнес-процесу	Торгова точка

1	2
Учасники бізнес-процесу	
Персонал	Продавці, помічник бухгалтера
Засоби реалізації	Telegram, MForce, стандарти занесення
Споживач	Бухгалтерія
Постачальник	Продавці
Попередній процес	Транспортування товару до магазину
Наступний процес	-
Записи у ході виконання процесу	Інформація в базу даних.

Після ідентифікації бізнес процесів було виконано діаграму IDEF0 «ЯК Є».

Зовнішні входи і виходи процесу «Оновлення наявності товару для продажу» представлено на рисунку 1.4.

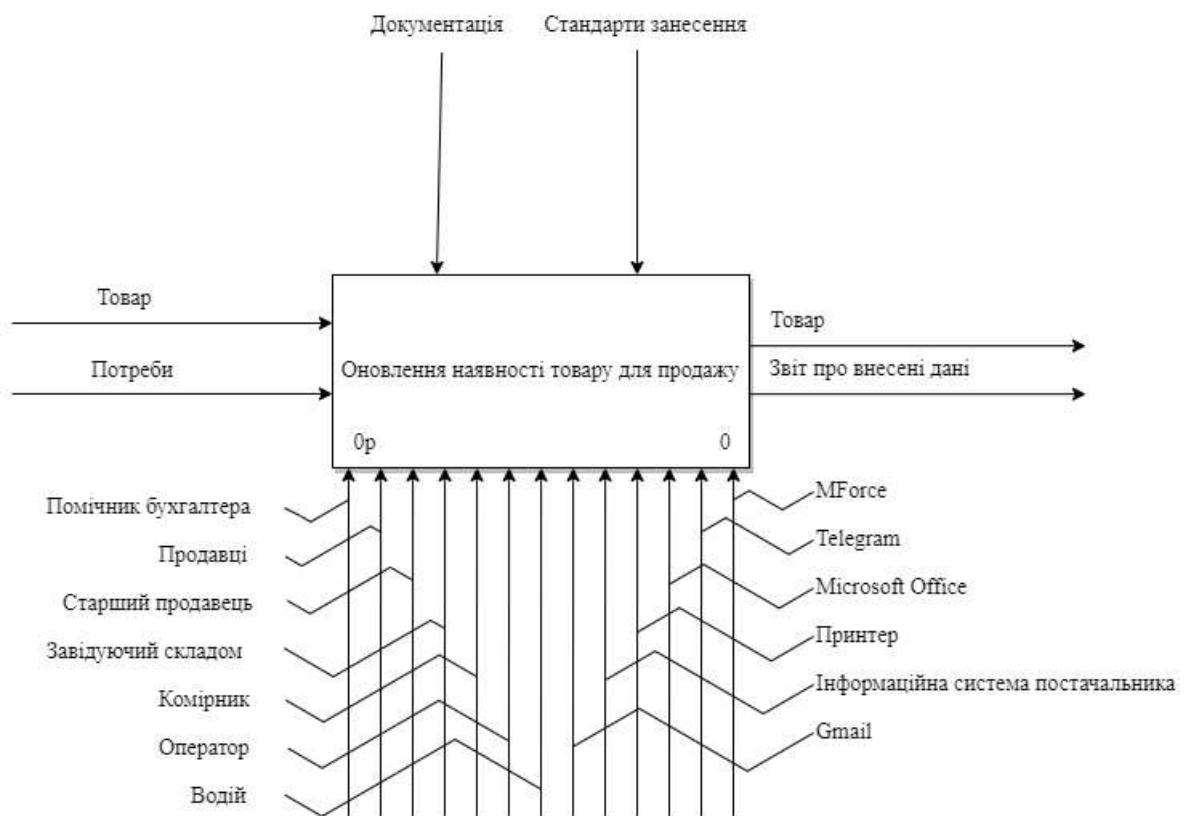


Рис. 1.4. Зовнішні входи і виходи процесу «Оновлення наявності товару для продажу»

Внутрішні входи і виходи процесу «Оновлення наявності товару для продажу» представлено на рисунку 1.5.

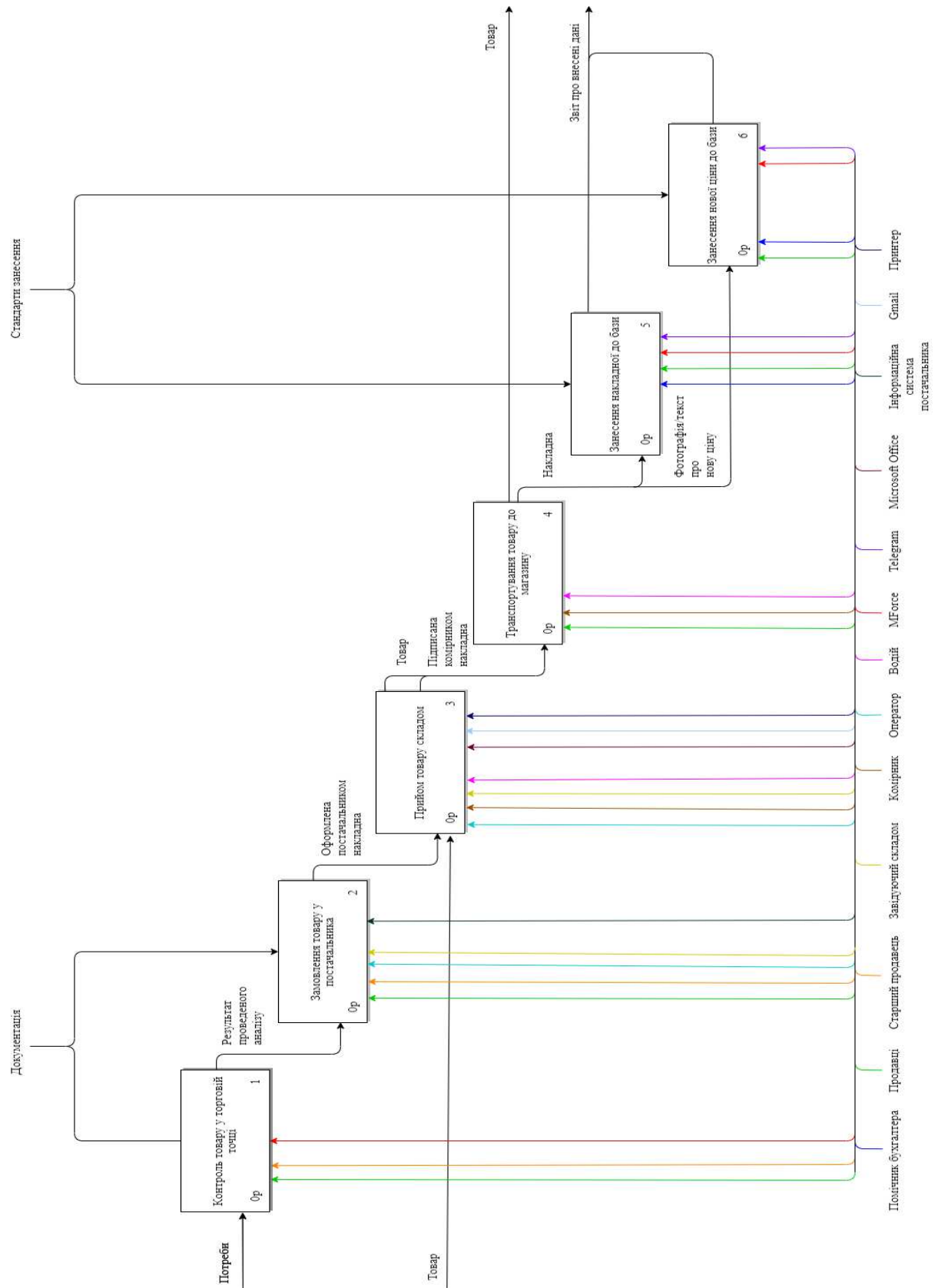


Рис. 1.5. Внутрішні входи і виходи процесу «Оновлення наявності товару для продажу»

Декомпозиція процесу «Контроль товару у торговій точці» представлено на рисунку 1.6.

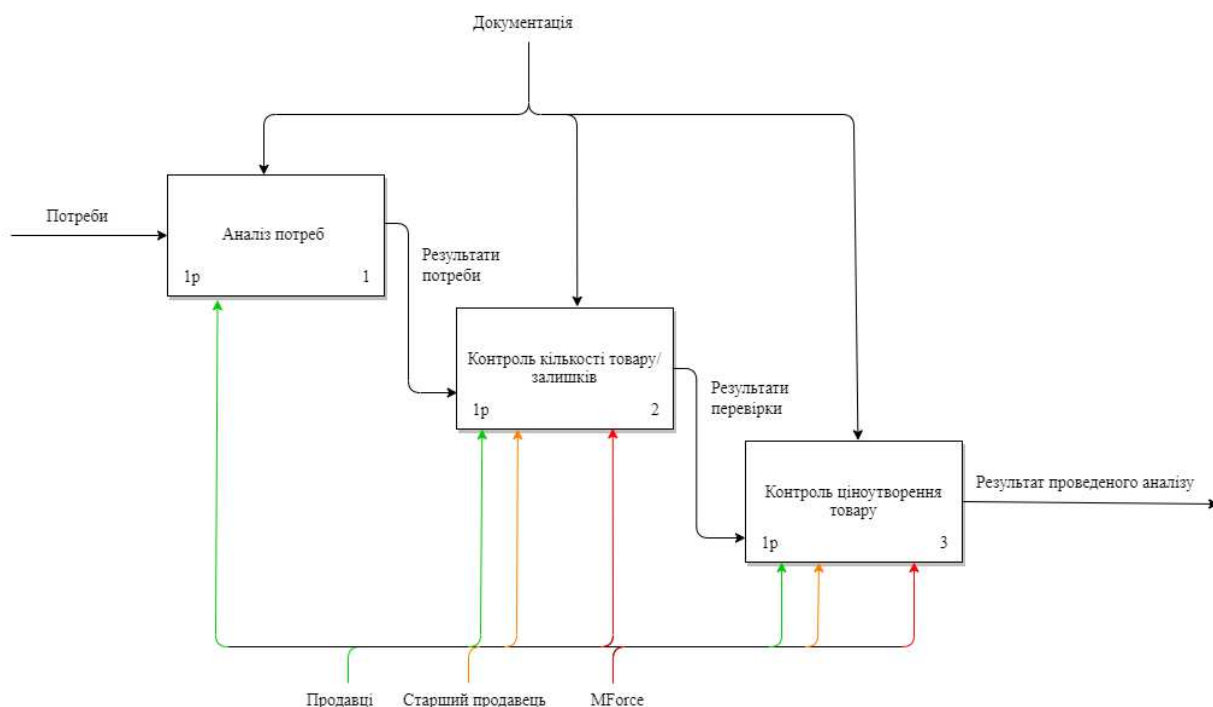


Рис. 1.6. Декомпозиція процесу «Контроль товару у торговій точці»

Декомпозиція процесу «Замовлення товару у постачальника» представлено на рисунку 1.7.

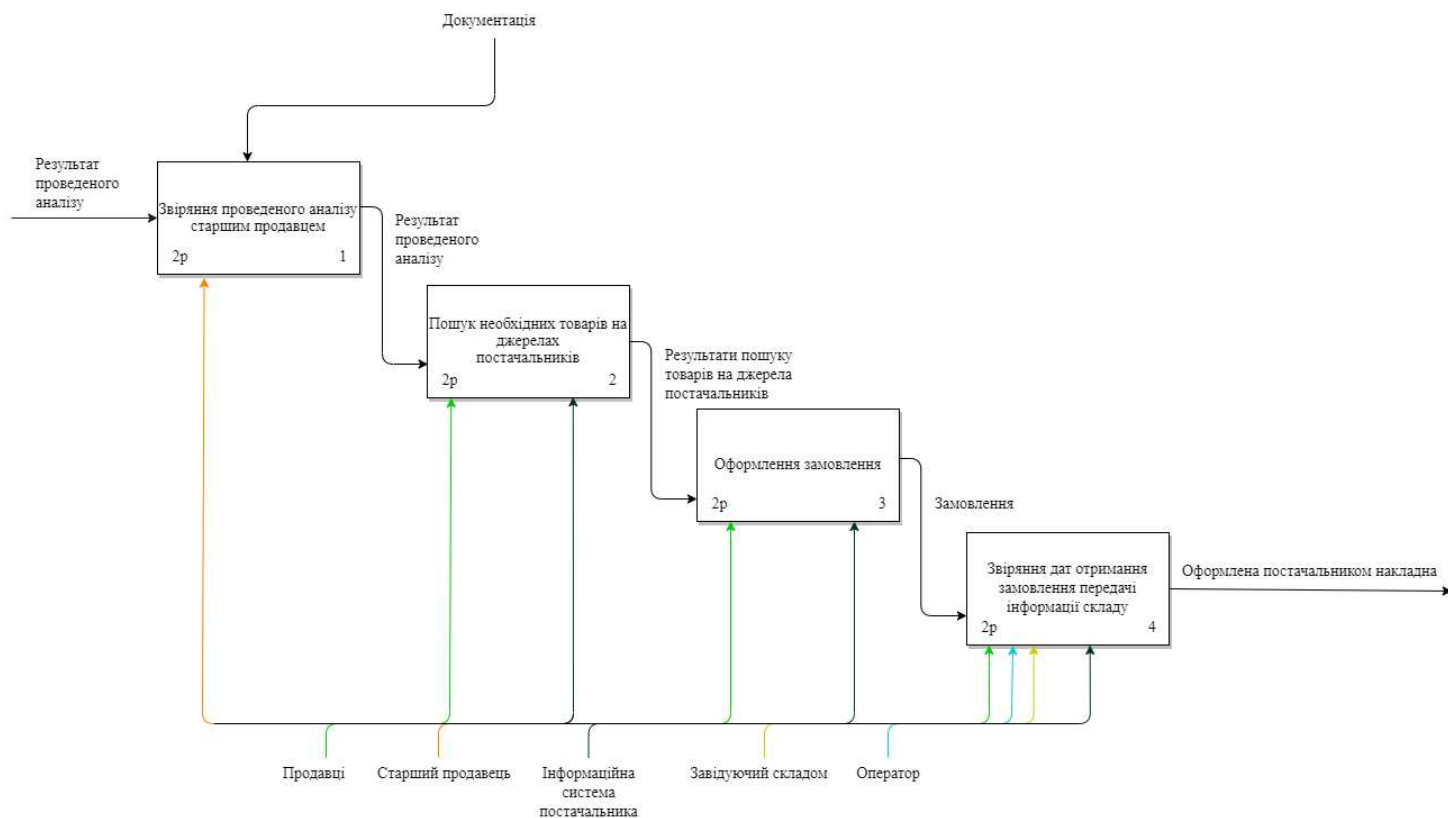


Рис. 1.7. Декомпозиція процесу «Замовлення товару у постачальника»» 26

Декомпозиція процесу «Прийом товару складом» представлено на рисунку 1.8.

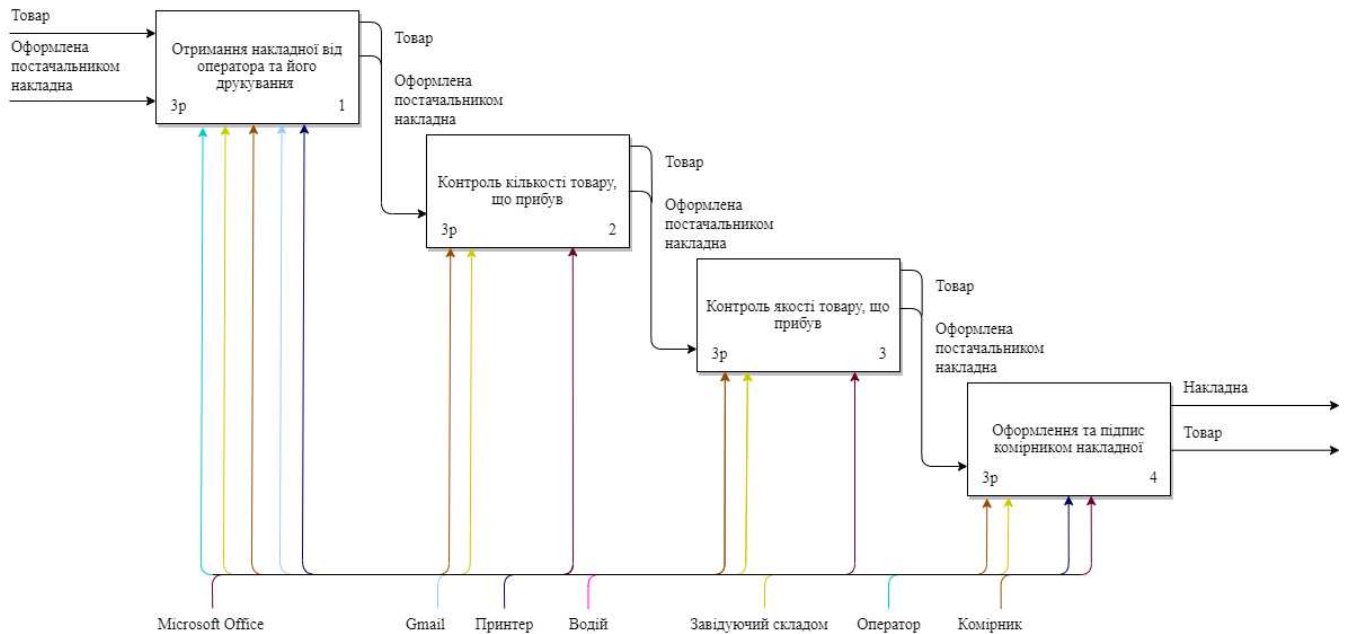


Рис. 1.8. Декомпозиція процесу «Прийом товару складом»

Декомпозиція процесу «Транспортування товару до магазину» представлено на рисунку 1.9.

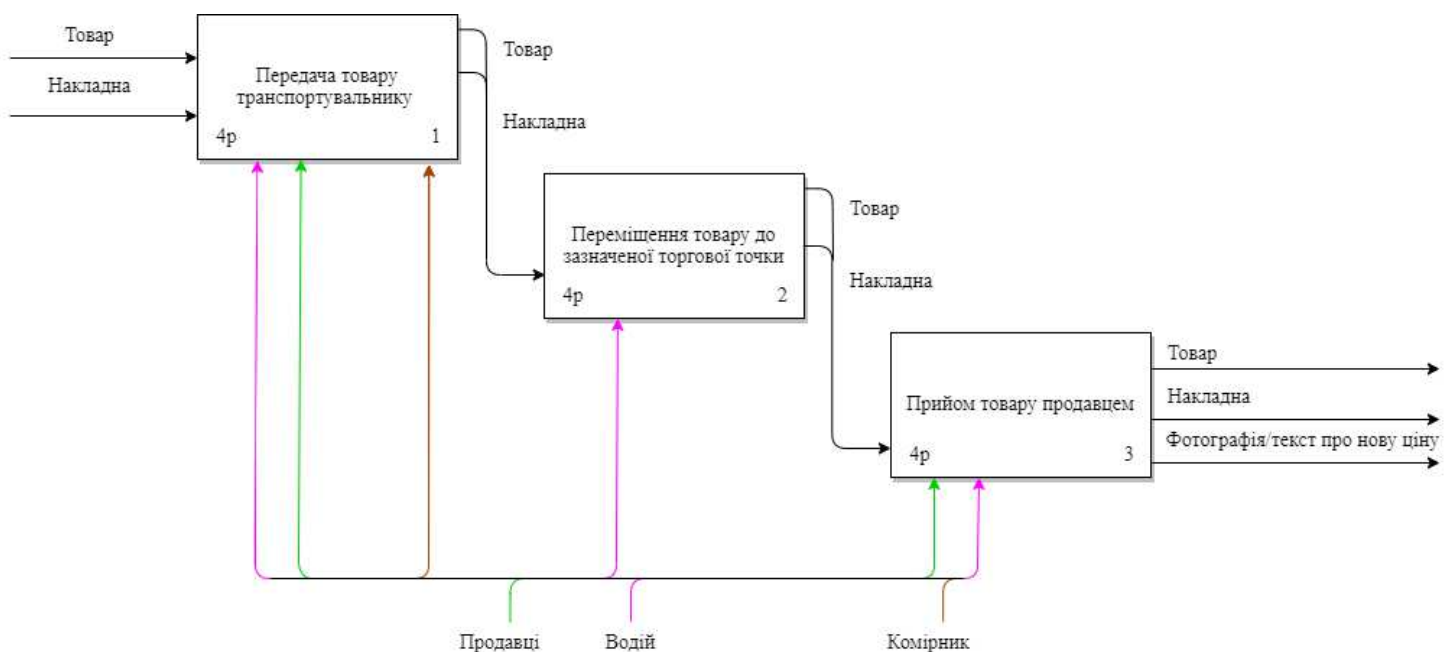


Рис. 1.9. Декомпозиція процесу «Транспортування товару до магазину»

Декомпозиція процесу «Занесення накладної до бази» представлено на рисунку 1.10.

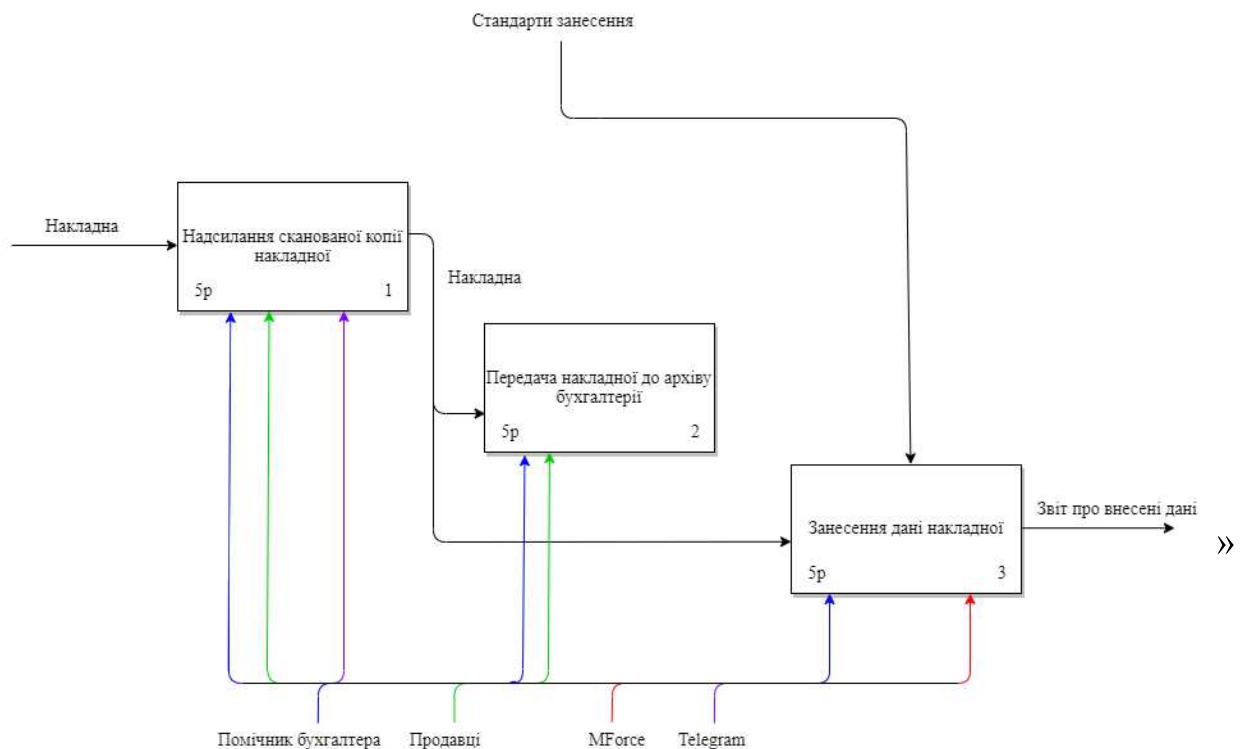


Рис. 1.10. Декомпозиція процесу «Занесення накладної до бази»

Декомпозиція процесу «Занесення нової ціни до бази» представлено на рисунку 1.11.

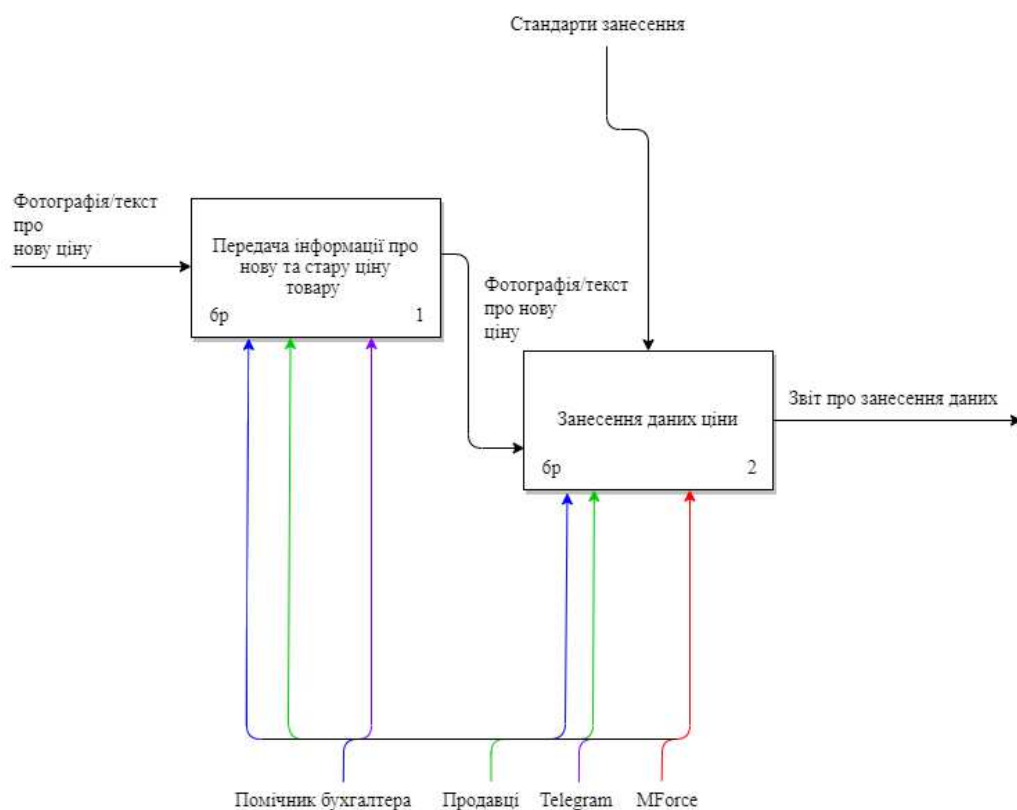


Рис. 1.11. Декомпозиція процесу «Занесення нової ціни до бази»

1.7 Виявлення проблематики за допомогою SWOT-аналізу

Для аналізу буде використовуватися якісні методики, а саме аналіз графічних схем процесу, тобто спроектовану раніше діаграма IDEF0.

Бізнес-процес «Оновлення наявності товару для продажу» має дуже багато потоків входу, а саме інфраструктури та персонал. Персонал неможливо виключити з процесу, бо вони всі мають велике значення при виконанні функцій:

- Продавці – вони задіяні в більшості процесів: аналізують наявність товару, роблять замовлення, приймають товар, передають інформацію щодо товару помічнику бухгалтера.
- Старший продавець – перевіряє роботу продавців та аналізує результати перевірки.
- Оператор – зв'язується з постачальниками та отримують накладні, які потім надсилають складу для подальшого переміщення товару.
- Склад (а сам завідуючий складом та комірники) – приймають товар, перевіряють їх кількість та якість, а потім передають їх по торговим точкам водію.
- Водій – займається транспортуванням товару по всім точкам від складу.
- Помічник бухгалтера – займається прийманням інформації щодо накладних на товари та заносить її за стандартами у базу.

Перейдемо до інформаційного програмного забезпечення, а саме їх дуже багато. Застосуємо SWOT-аналіз.

SWOT-аналіз — один із найпоширеніших аналітичних методів, який дозволяє в комплексі оцінити сильні й слабкі сторони компанії, а також можливості й загрози, що впливають на неї.

Матриця SWOT містить:

- S (strengths) — сильні сторони. Характеристики бізнесу, які вирізняють його на фоні конкурентів.

- W (weaknesses) — слабкі сторони. Ознаки, які роблять компанію вразливою на ринку.
- O (opportunities) — можливості. Їх компанія може використовувати для розвитку свого бізнесу.
- T (threats) — загрози, можуть завдати компанії збитків [7].

У більшості складових бізнес-процесу використовується програмне забезпечення MForce. Перелік сильних та слабких сторін цієї програми представлено в таблиці 1.8

Табл. 1.8. SWOT-аналіз програми MForce

Сильні	Слабкі
Великий асортимент функцій; Система збереження даних по різних вузлах;	Застарілий інтерфейс; Немає деяких можливостей, що потрібні під час роботи;
Систематизація даних.	Немає навчальних посібників, потрібна допомога представників компанії, що надає послуги програмою.
Загрози	Можливості
Втрата даних і неправильне трактування даних із-за людського фактору; Некоректне формування даних.	Використовуючи арсенал програми та її можливості можна відновити та покращити її можливості.

На етапі виконання бізнес-процесу використовується у «Контроль товару у торговій точці», «Занесення накладної до бази», «Занесення нової ціни до бази», а саме помічником бухгалтера та продавцями.

У процесі виконання передачі інформації (а саме накладних) використовується Gmail та Telegram, представлено в таблиці 1.9

Табл. 1.9. SWOT-аналіз програми Telegram і Gmail

Сильні	Слабкі
Швидкий доступ; Відомі у використанні.	Перенавантаження бізнес-процесу двома програмами; Немає можливості зробити нормальну
1	2
	архівацію даних.
Загрози	Можливості
Втрата даних.	Функції можна перенести в один інтерфейс.

На етапі виконання бізнес-процесу Gmail використовується у ««Прийом товару складом», а саме завідувачим складу та комірниками.

На етапі виконання бізнес-процесу Telegram використовується у «Занесення накладної до бази», «Занесення нової ціни до бази», а саме помічником бухгалтера та продавцями.

Microsoft Word використовується лише у ««Прийом товару складом», а саме завідувачим складу та комірниками, для аналізу накладних.

Вони перевантажують бізнес-процес, їх використовується багато, постійне пересилання різними технологіями, з чого виникають такі проблеми:

- 1) Відбувається втрата даних, неправильне їхнє трактування;
- 2) Некоректне формування накладних і їх подальше занесення в базу;
- 3) Несвоєчасне виконання всіх процесів або їх нестиковка;
- 4) Збільшується час відновлення інформації, що може призвести до втрати прибутку.

1.8 Моделювання бізнес-процесу «ЯК ПОВИННО БУТИ»

На основі проведеного аналізу буде сформована модель IDEF0 «ЯК ПОВИННО БУТИ». Зовнішні входи і виходи процесу «Оновлення наявності товару для продажу» моделі «ЯК ПОВИННО БУТИ» представлено на рисунку 1.12.

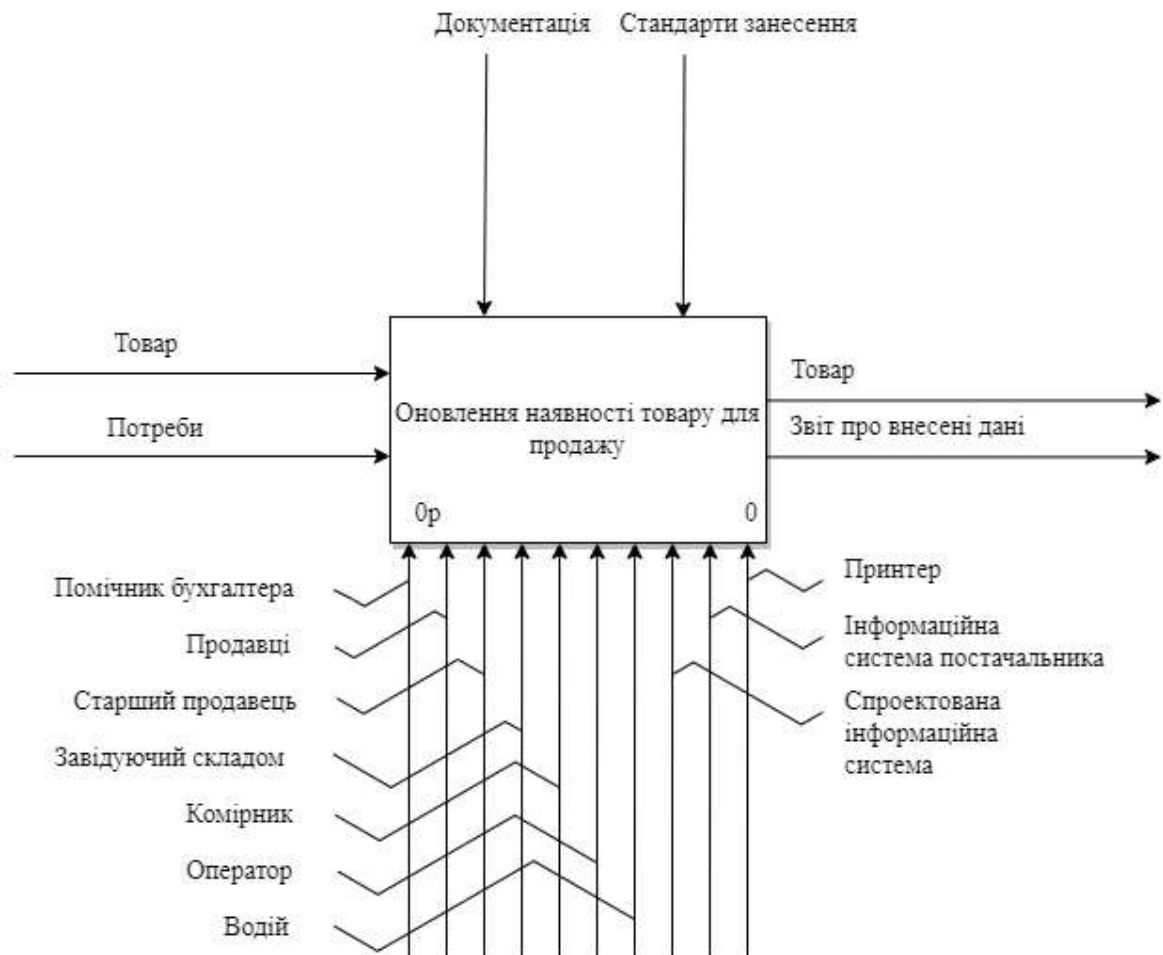


Рис. 1.12. Зовнішні входи і виходи процесу «Оновлення наявності товару для продажу»

Внутрішні входи і виходи процесу «Оновлення наявності товару для продажу» представлено на рисунку 1.13.

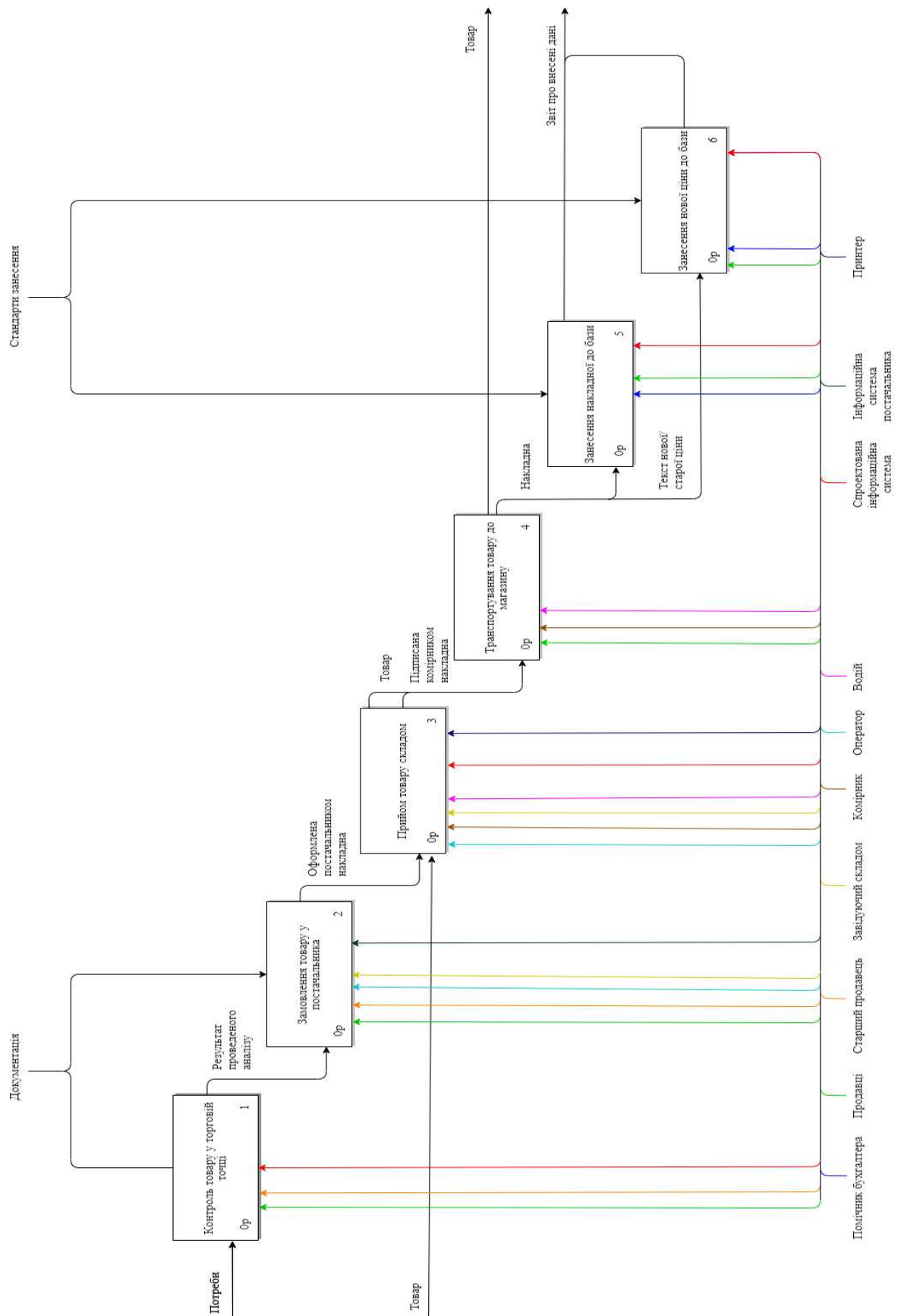


Рис. 1.13. Внутрішні входи і виходи процесу «Оновлення наявності товару для продажу»

Декомпозиція процесу «Контроль товару у торговій точці» представлено на рисунку 1.14.

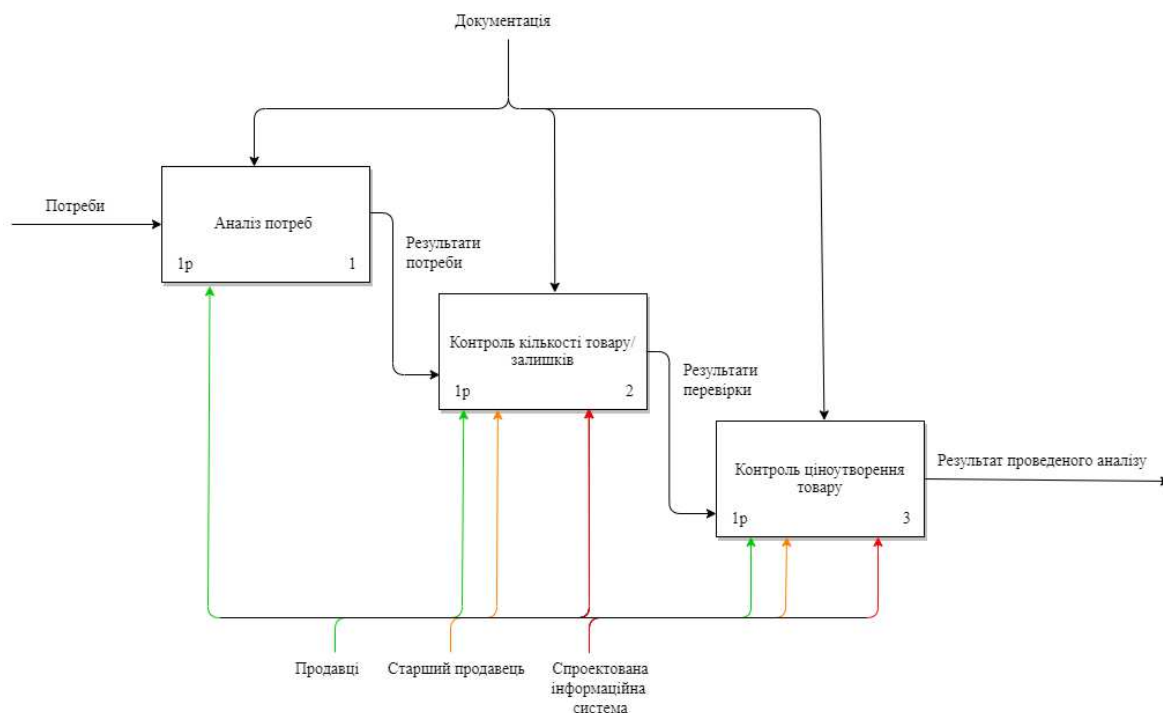


Рис. 1.14. Декомпозиція процесу «Контроль товару у торговій точці»

Декомпозиція процесу «Прийом товару складом» представлено на рисунку 1.15.

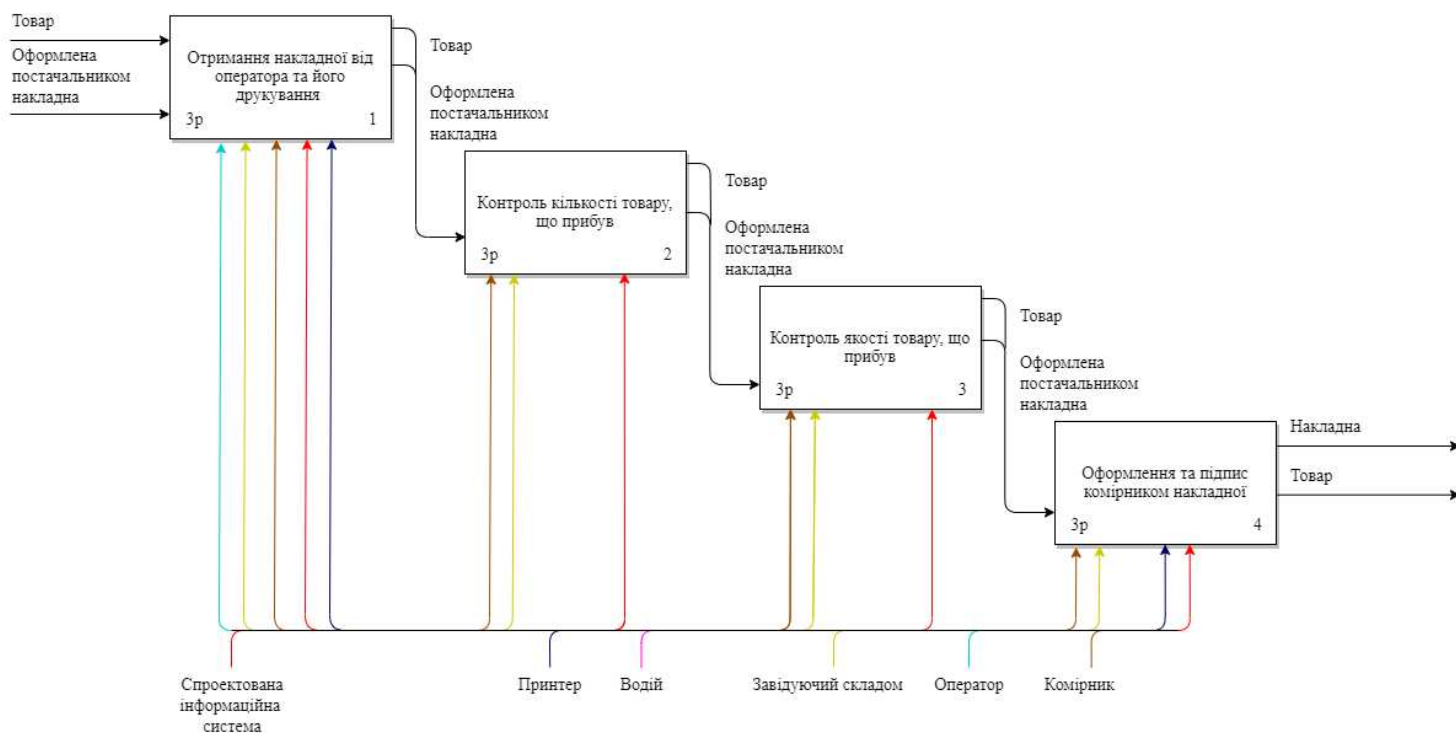


Рис. 1.15. Декомпозиція процесу «Прийом товару складом»

Декомпозиція процесу «Занесення накладної до бази» представлено на рисунку 1.16.

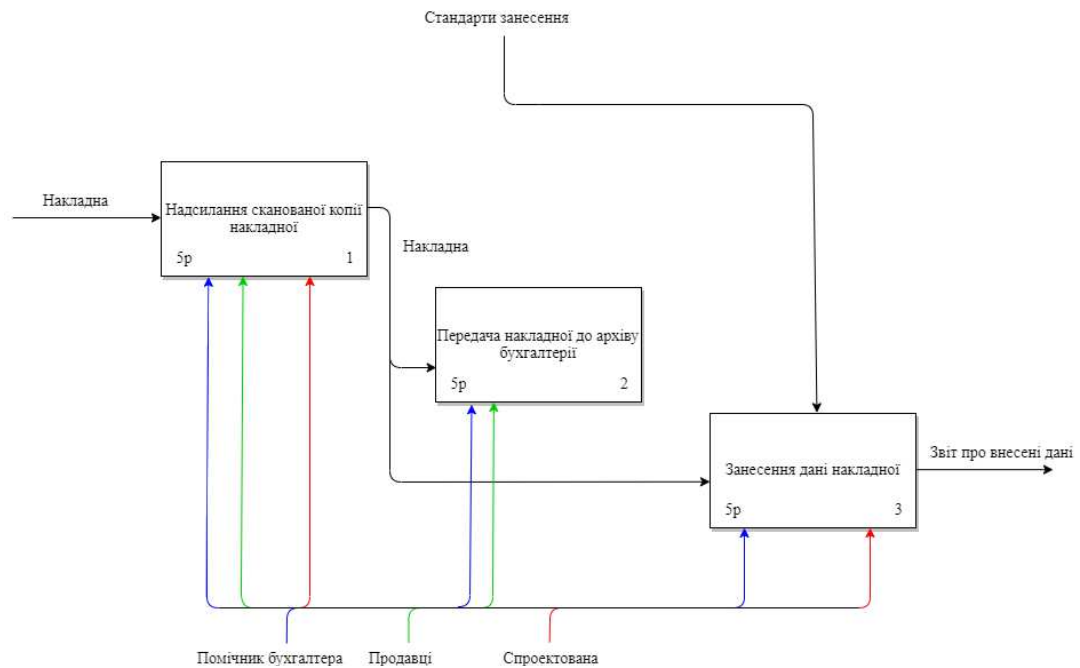


Рис. 1.16. Декомпозиція процесу «Занесення накладної до бази»

Декомпозиція процесу «Занесення нової ціни до бази» представлено на рисунку 1.17.

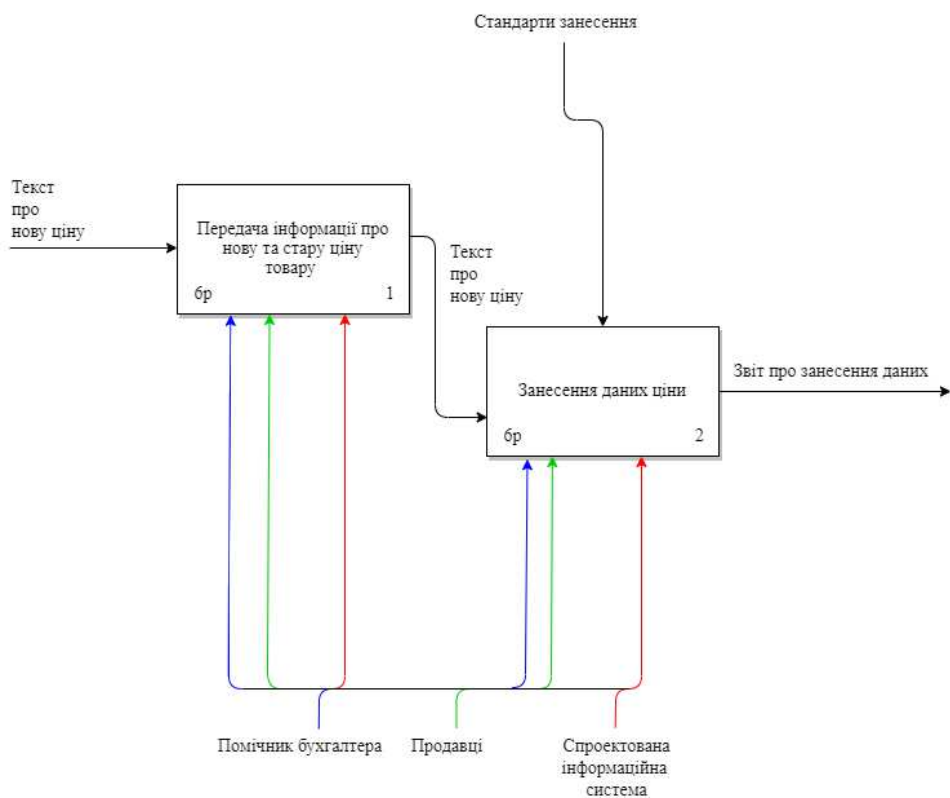


Рис. 1.17. Декомпозиція процесу «Занесення нової ціни до бази»

1.9 Формування вимог до майбутньої інформаційної системи

Порівняльна таблиця «ЯК Є» та «ЯК ПОВИННО БУТИ» представлена в таблиці 1.10.

Табл. 1.10. Порівняльна таблиця «ЯК Є» і «ЯК ПОВИННО БУТИ»

Завдання, які виконується під час бізнес-процесу	«ЯК Є»					«ЯК ПОВИННО БУТИ»
	Telegram	Microsoft Word	Gmail	MForce	ІС постачальника	Спроектвана на ІС
Контроль товару у торговій точці	-	-	-	+	-	+
Замовлення товару у постачальника	-	-	-	-	+	-
Прийом товару складу	-	+	+	-	-	+
Транспортування товару до магазину	-	-	-	-	-	-
Занесення накладної до бази	+	-	-	+	-	+
Занесення нової ціни до бази	+	-	-	+	-	+

Із цієї таблиці можна виявити, що спроектована таблиця зможе виконати майже усі задачі бізнес-процесу (окрім «Замовлення товару у постачальника»

та «Транспортування товару до магазину» із-за локальності виконаних задач без застосування іншого програмного забезпечення.

Модель запропонованої ІС представлено на рисунку 1.18.

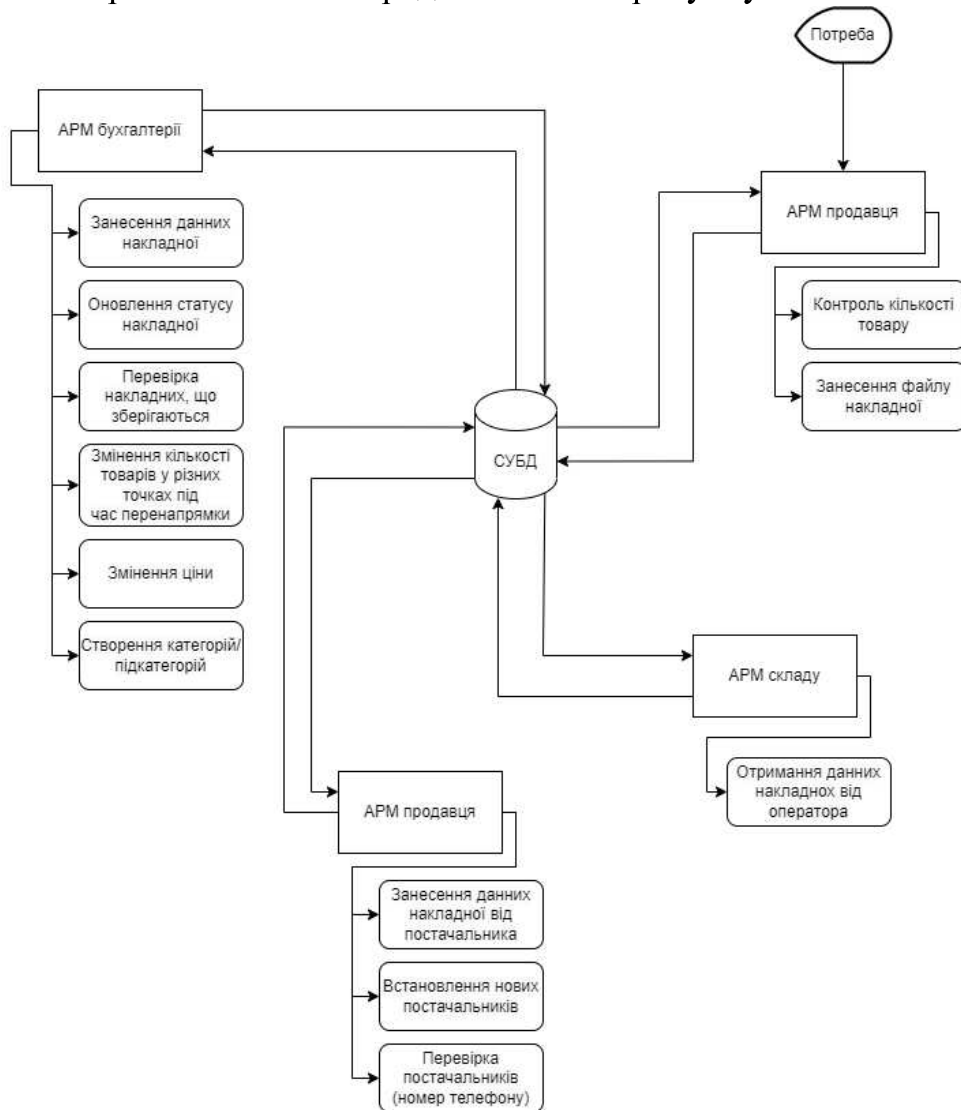


Рис. 1.18. Модель запропонованої ІС

Вимоги, які виконуються, щодо ІС:

- Всі дані, у тому числі файли накладних, ціни, накладні від оператора тощо, зберігатимуться в одному місці без втручання інших технологій;
- Своєчасне встановлення інформаційних матеріалів для подальшого проведення бізнес-процесу;
- Введення статистики проведення операцій у процесах, а саме: коли була отримана накладна складу, і коли вона з'явилась в системі після завантаження у

систему, що може допомогти верхньому керівництву контролювати процеси виконання бізнес-процес (для захисту зарплатні, наприклад).

- Усі учасники процесу зможуть використовувати систему у своїх задачах, які від них потрібні.

Функціонування системи передбачає використання таких програмних засобів:

1. Реляційна СУБД;
2. Мова програмування сервісної частини, фреймворк та допоміжні бібліотеки/пакети;
3. Мова програмування, мова розмітки та формальна мова опису зовнішнього вигляду документа клієнтської частини, допоміжні бібліотеки/пакети;
4. Браузер;
5. Середовище розробки програмного забезпечення.;
6. Графічний редактор для створення діаграм.

Функціонування системи передбачає використання таких сервісних засобів:

1. Постійний Інтернет зв'язок для динамічного виведення інформації;
2. Захист інформації від зовнішніх подразників та загроз.

Програмні засоби

Для того, щоб керувати базою даних, вибираємо реляційну систему PostgreSQL.

Розроблятиметься у середовищі розробки JetBrains WebStorm.

Сервер: розробляється на TypeScript з використанням фреймворку Nest (її пакети) та ORM Sequelize. При розробці використовуються бібліотеки: SwaggerUIExpress (для введення явної документації API), UUID.

Клієнт: розробляється на JavaScript з використанням фреймворку Vue. Мова розмітки – HTML5, SASS. При розробці використовуються бібліотеки: swagger-typescript-api (для автоматичної генерації файлів зв'язку з сервером з типізованими даними та методами), Vuex, VueRouter, vuetify, vue-cropperjs.

Браузер: Google Chrome.

Операційна система: Linux.

Для створення діаграм у процесі розробки використовуватиметься сервіс Draw.io.

Інтернет підключення – використовується технологія Ethernet. З'єднання серверу та робочого комп'ютера здійснюється за допомогою Інтернет кабеля.

Для захисту системи від зовнішніх загроз використовується антивірус Avast.

Технічні засоби

Системні вимоги щодо клієнтської частини реалізації:

- Процесор – починаючи з Intel® Core™ i3-10100F CPU @ 3.60 GHz
- Тип системи – 64-розрядна;
- Оперативна пам'ять – від 4Гб;
- Відеокарта – від 256 mb;
- Монітор – 1920 x 1080;
- Клавіатура;
- Миша;
- Місце на жорсткому диску – 4 гб;
- Мережевий адаптер – Intel® Ethernet Connection (12) I1219-V.

Системні вимоги щодо серверній частини реалізації:

- Процесор – починаючи з Intel® Core™ i3-10100F CPU @ 3.60 GHz;
- Тип системи – 64-розрядна;
- Оперативна пам'ять – від 4Гб;
- Відеокарта – від 256 mb;
- Монітор – 1920 x 1080;
- Клавіатура;
- Миша;
- Місце на жорсткому диску – 5 тб;

Мережевий адаптер – Intel® Ethernet Connection (12) I1219-V.

РОЗДІЛ 2. РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ ІС

2.1. Інформаційне забезпечення

2.1.1. Розробка та аналіз системних рішень

Аналіз предметної області існуючої організації, постановка задачі, виявлення проблем та шляхів їх вирішення, формування вимог до майбутньої інформаційної системи – це вже пройдений перший етап роботи над проектом.

Проектування архітектури ІС – це другий етап, який допомагає уникнути майбутніх проблем та помилок, які можуть виникнути у подальшій реалізації. Саме на цьому етапі формується концепція системи та шляхи її використання.

Одним із видів моделей, яка характеризує систему, є модель варіантів використання або діаграма прецедентів. В майбутньому вона допоможе в проектуванні концептуальної моделі. Проаналізуємо бізнес-процеси «ЯК ПОВИННО БУТИ» та винесемо всіх користувачів системи, а саме: помічник бухгалтера, продавець, комірник (склад) та оператор. Вони є основними користувачами та у кожного з них є задачі, які цією системою вирішуються.

Діаграма прецедентів ІС представлено на рисунку 2.1.

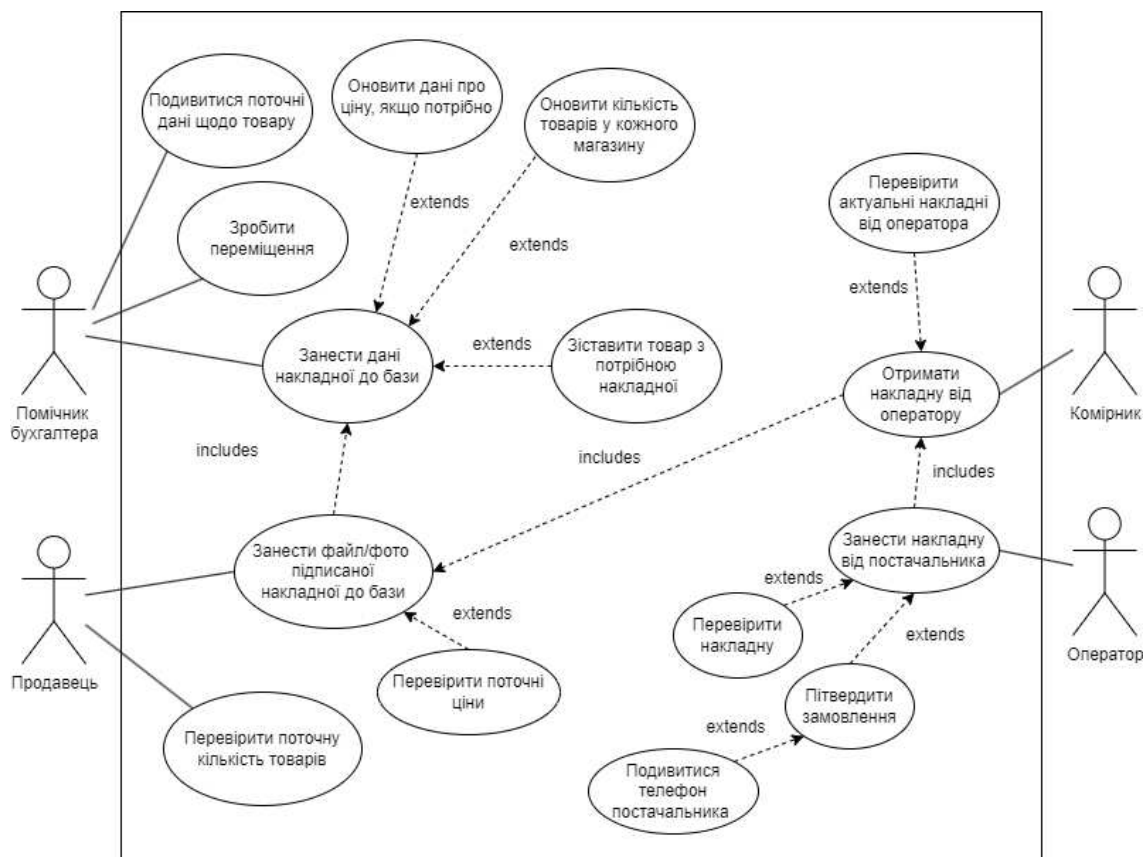


Рис. 2.1. Діаграма прецедентів

Приклади варіантів використання ІС представлено в таблицях 2.1-2.3.

Таблиця 2.1. Подивитися поточні дані щодо товару

Для чого використовується	Використовується для перегляду точної інформації щодо заданого товару
Діюча особа	Помічник бухгалтера
Передумова	Податкова вимагає накладну на певний товар
На що реагує	Блок «Наявні товари»
Сценарій	Помічник бухгалтера авторизується в системі; вибирає необхідний магазин; відкриваю сховище накладних; за допомогою пошуку відбувається сортування по назві товару та/або даті останнього оновлення у систему.
Що відбудеться	Система віддає необхідні дані про товар та надає можливість завантажити зображення/файл з бази даних необхідної накладної.

Таблиця 2.2. Занести дані з накладної

Для чого використовується	Використовується для відновлення даних щодо товару
Діюча особа	Помічник бухгалтера
Передумова	До магазину прийшов новий товар, продавець відправляє накладну
На що реагує	Кнопка «Надіслати накладну»
Сценарій	Помічник бухгалтера отримує повідомлення про те, що з'явилась нова підписана накладна від продавця; заходить в розділ «Оновлення даних про товар», заносить необхідні дані та нажимає кнопку «Зберегти».
Що відбудеться	Відбувається оновлення інформації щодо товару; в журналі накладних з'являється нова архівна
	запис щодо товару; відбувається зв'язування накладної від продавця та поточного товару у сховищі накладних.

Таблиця 2.3. Занести файл/фото накладної

Для чого використовується	Використовується для занесення файлу накладної для збереження її на сервері
Діюча особа	Продавець
Передумова	До магазину прийшов новий товар
На що реагує	Кнопка «Занести накладну від продавця»
Сценарій	Продавець авторизується у системі; він натискає на кнопку «Надіслати накладну», надсилає файл та вибирає постачальника.
Що відбудеться	Створення нового запису у базі даних з поточною датою та зберіганням файлу на сервері.

Діаграма прецедентів допомагає сформулювати точне представлення того, для чого система буде використовуватися. Так, наприклад, занесення даних накладної пов'язана з помічником бухгалтера та продавцем. Коли продавець заносить файл щодо накладної у системи, помічник бухгалтера має отримати повідомлення про нову накладну та з якого вона магазину/який в неї постачальник для автоматизації процесу занесення даних про товар далі.

При цьому відображення даних для кожного користувача системи будуть різні. Режими доступу кожного користувача представлено в таблиці 1.4.

Таблиця 2.4. Режими доступу ІС

Користувач	Привілеї
Помічник бухгалтера (ACCOUNTANT)	• фільтр товарів за параметром торгової точки; • АРМ занесення даних накладної у базу (усі запити CRUD)
	• відображення параметрів товарів – усі. • доступ до файлів накладної за типом STORE. • доступ до особистого профілю.
Продавець (SALESMAN)	• фільтр товарів за вказаним за продавцем магазином • відображення параметрів товару – ім'я та кількість. • доступ до особистого профілю. • АРМ продавця даних файлів (фото/документ) накладної (запит INSERT та UPDATE). • доступ до особистого профілю.
Комірник (STOCK)	• АРМ комірника перегляду даних накладної (запит SELECT) за типом STOCK. • доступ до особистого профілю.

1	2
Оператор (OPERATOR)	• АРМ оператора занесення даних накладної (запит INSERT) за типом STOCK.
	• відображення параметрів постачальників – ім'я, номер телефону. • доступ до особистого профілю.
Супер-адмін (ADMIN)	має усі рівні доступу. <i>Додатковий доступ:</i> • список користувачів (скидання паролів, оновлення статусу, оновлення рівні доступу для користувачів) • видалення файлів.

2.1.2. Розробка концептуальної моделі даних

Концептуальна модель – це відображення предметної області бази даних, що розробляється, та її причино-наслідкові зв'язки. База даних складається з сутностей та її атрибутів, а також зв'язків між цими сутностями. За концепцією UML (уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування[8]) сутності відображаються у вигляді прямокутника, атрибути – овалів, а зв'язки – лініями, які посередині з'єднуються в один ромб, який визначає, що робить цей зв'язок.

Використовуємо при розробці концептуальної моделі – модель «сутність-зв'язок» або ERD діаграма (Entity–relationship model). Вона є однією з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, що його реалізує [9]. Проектуючи, потрібно перетворити ERD модель у конкретну схему вибраної СУБД.

Створена функціональна модель ІС, а також проведений аналіз предметної області та моделі прецедентів, допомагає сформулювати

концептуальну модель, щоб покрити всі завдання. Наведемо конкретні сутності, їх атрибути та зв'язки.

Сутність Категорія має такі атрибути:

- Найменування – це найменування категорії

Сутність Зв'язок Категорія та Підкатегорія.

Сутність Підкатегорія має такі атрибути:

- Найменування – це найменування підкатегорії.

Сутність Найменування товару має такі атрибути:

- Найменування – це найменування товару.
- Штрих-код – це штрих-код товару.
- Підкатегорія – це підкатегорія найменування товару (пов'язана з сутністю Підкатегорія).

Сутність Товар має такі атрибути:

- ПДВ – це ПДВ товару.
- Статус – це статус товару (наявний чи не наявний)
- Дата створення – це дата занесення такого товару у базу даних.
- Дата оновлення – це дата останнього оновлення товару у базі даних.
- Ціна – це ціна товару (пов'язана з сутністю Ціна)
- Постачальник – це постачальник товару (пов'язана з сутністю Постачальник).
- Найменування товару – це найменування товару (пов'язана з сутністю Найменування товару).

Сутність Ціна має такі атрибути:

- Нова ціна – це нова ціна товару.
- Стара ціна – це стара ціна товару (при оновленні інформації сюди заноситься інформація з нової ціни, а нова ціна оновлюється новими даними).
- Дата створення – це дата створення цієї ціни у базі даних.
- Дата оновлення – це дата останнього оновлення ціни у базі даних.

Сутність Постачальник має такі атрибути:

- Найменування – це найменування постачальника.
- Телефон – це номер телефону call-центру постачальника.

Сутність Зв'язок магазину та товару має такі атрибути:

- Кількість – це кількість товару, яка є на зараз у магазині.

Сутність Магазин має такі атрибути:

- Найменування – це найменування магазину.
- Дата створення – це дата створення цього магазину у базі даних.
- Дата оновлення – це дата останнього оновлення магазину у базі даних.

Сутність Зв'язок Сховища накладних та Магазину.

Сутність Коефіцієнти магазинів має такі атрибути:

- Магазин №1 – це перший магазин у зв'язку (пов'язана з сутністю Магазин).

- Магазин №2 – це другий магазин у зв'язку (пов'язана з сутністю Магазин).

- Коефіцієнт – це нарахований коефіцієнт зв'язку магазинів, який потрібний для подальшої математичної моделі.

Сутність Зв'язок Сховища накладних та Магазину.

Сутність Сховище накладних має такі атрибути:

- Тип накладної – це тип занесеної накладної.
- Статус – це статус чи відпрацьована накладна у базі даних.
- Дата створення – це дата створення цієї накладної у базі даних.
- Дата оновлення – це дата останнього оновлення цієї накладної у базі даних.

- Постачальник – це постачальник, який пов'язаний з накладною (пов'язана з сутністю Постачальник).

Сутність Файл має такі атрибути:

- Найменування – це найменування категорії.
- Тип файлу – це тип файлу, який зберігається.

- Накладна – це накладні зі сховища накладних (пов’язана з сутністю Сховище накладних)

Сутність Користувач має такі атрибути:

- Ім’я – це ім’я користувача.
- Пошта – це пошта користувача.
- Пароль – це пароль користувача.
- Прихований – чи активний профіль чи видалений.
- Роль – це роль користувача у системі.
- Блокування – це чи заблокований користувач у системі.
- SLUG – це легко читані ключові слова щодо користувача.
- Файл – це аватар (фото) користувача (пов’язана з сутністю Файл).
- Магазин – це магазин, з яким пов’язаний користувач, якщо потрібно (пов’язана з сутністю Магазин)
- Дата створення – це дата створення профілю користувача.
- Дата оновлення – це дата останнього оновлення профілю користувача.

Сутність Сесія авторизації користувача має такі атрибути:

- IP – це айпі адрес користувача.
- Активний – це чи активна сесія на зараз.
- Агент користувача – це рядок користувача агента браузера.
- Користувач – це користувач сесії (пов’язаний з сутністю Користувач).
- Дата створення – це дата створення початку сесії деякого користувача.
- Дата оновлення – це дата останнього оновлення сесії деякого користувача.

Зв’язки між сутностями та їх типи представлені у таблиці 2.5.

Таблиця 2.5. Зв’язки між сутностями та їх типи

Назва сутностей	Їх зв’язок	Тип зв’язку
Категорія $\rightleftharpoons$ Зв’язок категорія та під категорія	Має	Один-до-багатьох

1		2	
Підкатегорія $\Leftrightarrow$ Зв'язок категорія та підкатегорія	Має	Один-до-багатьох	
Підкатегорія $\Leftrightarrow$ Найменування товару	Має	Один-до-багатьох	
Найменування товару $\Leftrightarrow$ Товар	Має	Один-до-багатьох	
Ціна $\Leftrightarrow$ Товар	Має	Один-до-багатьох	
Постачальник $\Leftrightarrow$ Товар	Має	Один-до-багатьох	
Товар $\Leftrightarrow$ Зв'язок магазину та товару	Є	Один-до-багатьох	
Магазин $\Leftrightarrow$ Зв'язок магазину та товару	Має	Один-до-багатьох	
Магазин $\Leftrightarrow$ Зв'язок сховища накладних та магазину	Має	Один-до-багатьох	
Постачальник $\Leftrightarrow$ Сховище	Має	Один-до-багатьох	
Сховище накладних $\Leftrightarrow$ Файл	Має	Один-до-багатьох	
Магазин $\Leftrightarrow$ Користувач	Надає	Один-до-багатьох	
Магазин $\Leftrightarrow$ Коефіцієнти Магазинів	Має зв'язок	Один-до-багатьох	
Користувач $\Leftrightarrow$ Файл	Встановлює	Один-до-багатьох	
Користувач $\Leftrightarrow$ Сесія авторизації користувача	Створює	Один-до-багатьох	

Також система передбачає створення користувацьких типів даних, а саме enum. Enum або перелічуваний тип даних – це тип даних, що складається з множини іменованих значень, які називаються елементами, членами або еnumerаторами типу [10].

У спроектованій системі будуть два таких типи:

1. AdminRole (Роль користувача):

- Admin
- Operator
- Accountant
- Stock
- Salesman

2. TypeInvoices (Тип накладної):

- Store
- Stock

Концептуальну модель даних будуємо за допомогою діаграми «сутність-зв'язок». Модель представлена рисунку 2.2.

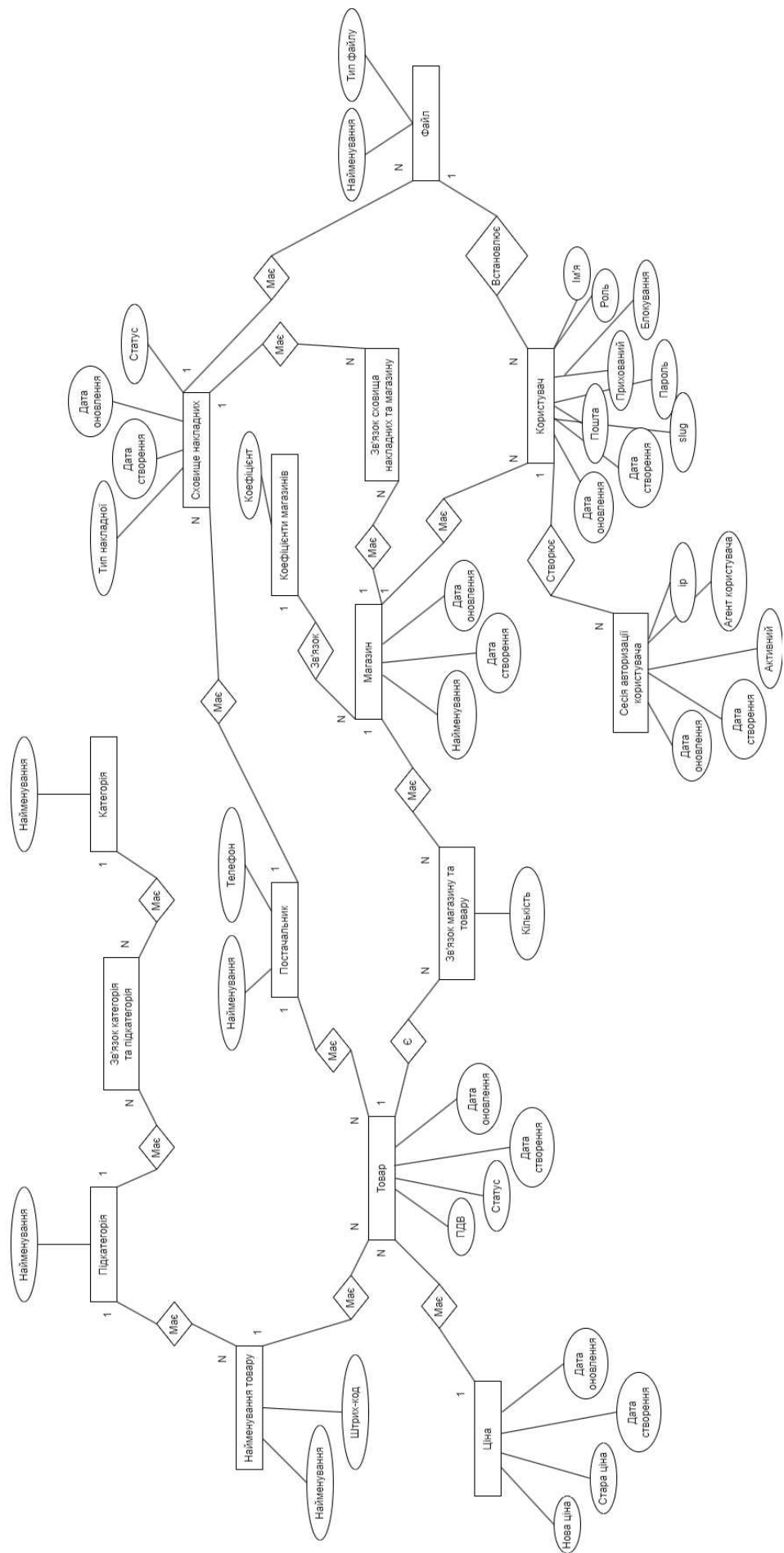


Рис. 2.2. Концептуальна модель

2.1.3. Розробка логічної моделі даних

Логічна модель даних моделюється за допомогою вже існуючої концептуальної моделі даних, яка була сформована незадовго до цього. Вона оновлює сутності, удосконалює їх, деталізує та трансформує, визначає майбутню структуру бази даних. Прототип бази даних, а саме логічна модель даних представлено на рисунку 2.3.

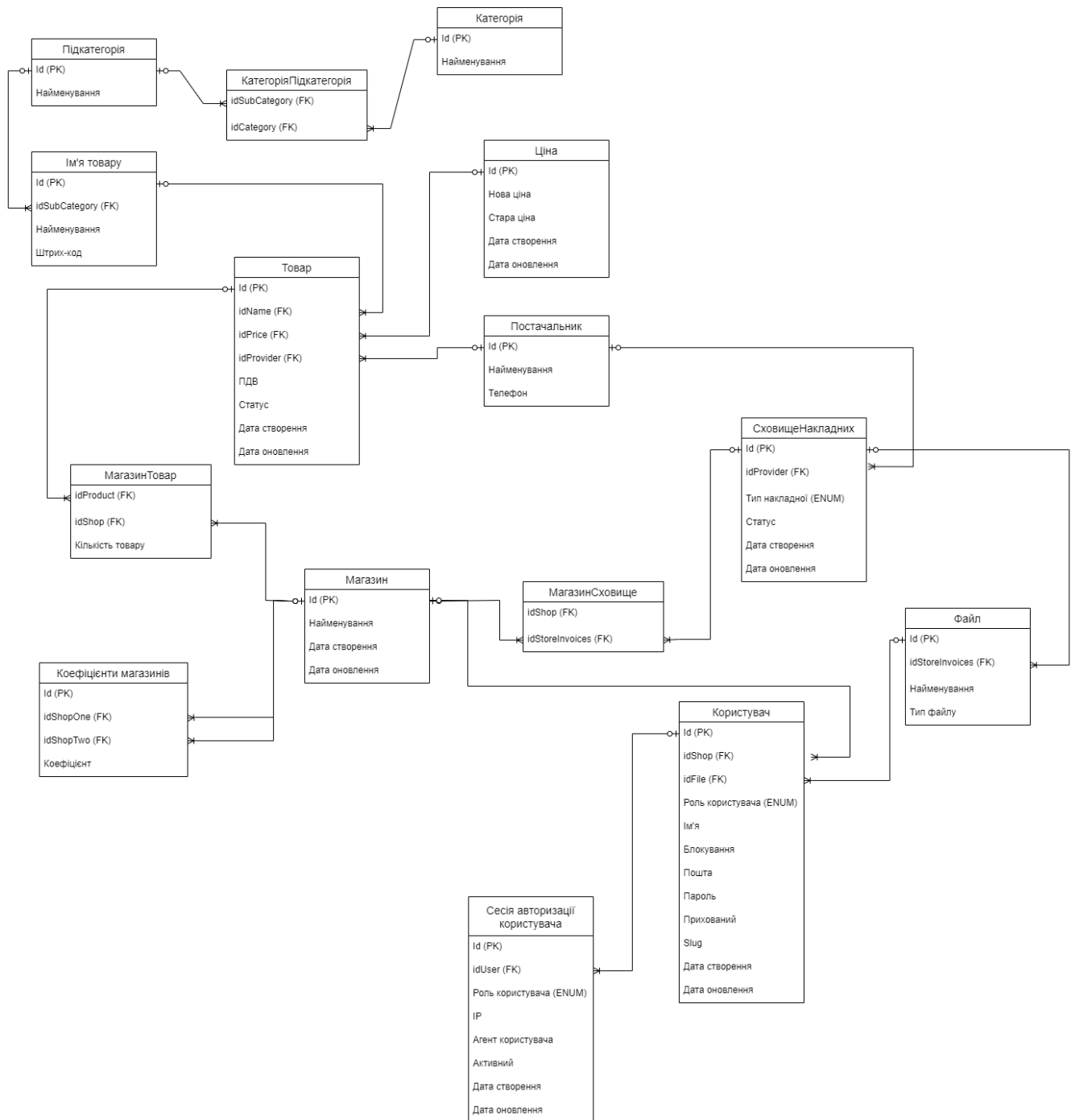


Рис. 2.3 Логічна модель даних

2.1.4. Розробка фізичної моделі даних

Фізична модель даних — це останній етап створення бази даних. Фізична модель даних використовує засоби вибраної СУБД, використовує її типізацію та засоби проектування для того, щоб вийшов необхідних результат для створення майбутньої бази даних.

Фізична модель даних представлена на рисунку 2.4

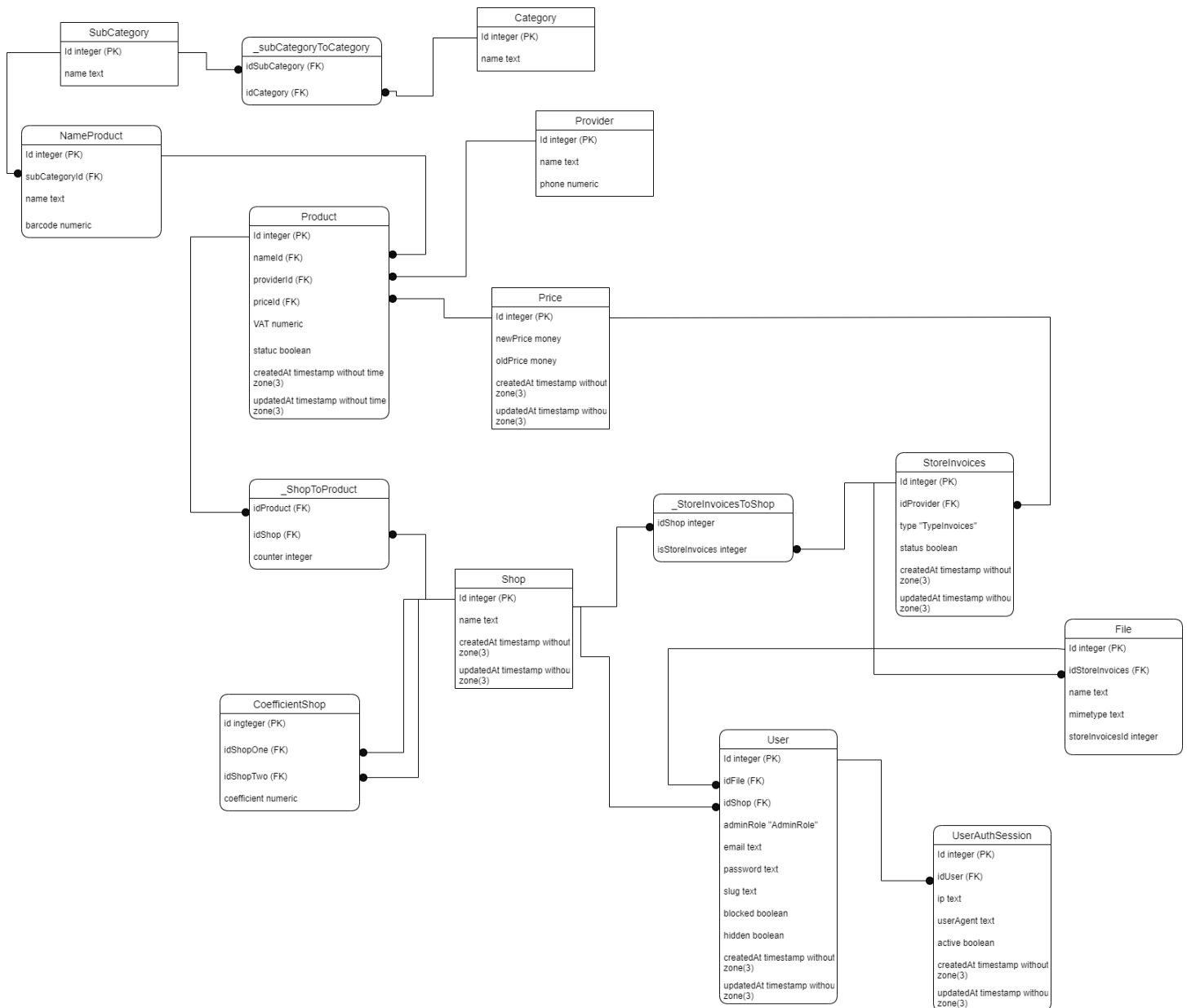


Рис. 2.4. Фізична модель даних

2.1.5. Розробка діаграма послідовності

Діаграма послідовності – дана діаграма графічно демонструє порядок взаємодії певних об’єктів програми у часі [11]. Наведемо, як приклад роботи ІС, сценарій авторизації помічника бухгалтера до системи, вибору магазину та надсилання нової кількості товару.

Спочатку помічник бухгалтера вводить дані для авторизації у поля, які надає клієнт (перед цим проводиться створення користувача супер-адміном та наданням ним користувачу ролі ACCOUNTANT). Після цього клієнт ці дані отримує та йде передача їх до сервера. Сервер перевіряє правильність даних, що він отримав з клієнту, та тих, які він отримав від бази даних, та повідомляє клієнту про успішність введених даних та передає йому JWT-token, який зберігається у користувача в браузері у LocalStorage.

Користувач продовжує роботу в системі вже з режимом доступу, який надав йому сервер. Він вибирає магазин, в якому продовжується робота, клієнт віддає роут для подальшої роботи з цим магазином.

Далі користувач, а саме помічник бухгалтера, визиває логіку додавання кількості товару до вибраного магазину, запит до сервера, який перевіряє ці дані, а вже після відбувається запит до БД, де ці дані оновлюється. Після цього сервер надсилає відповідь щодо вирішеного запиту, клієнт ці дані отримує, оновлює сторінку, виводить модальне вікно успішного виконаного запиту та показує оновлені дані щодо товару, які клієнт отримав з серверу.

Діаграма послідовності даного процесу представлено на рисунку 2.5

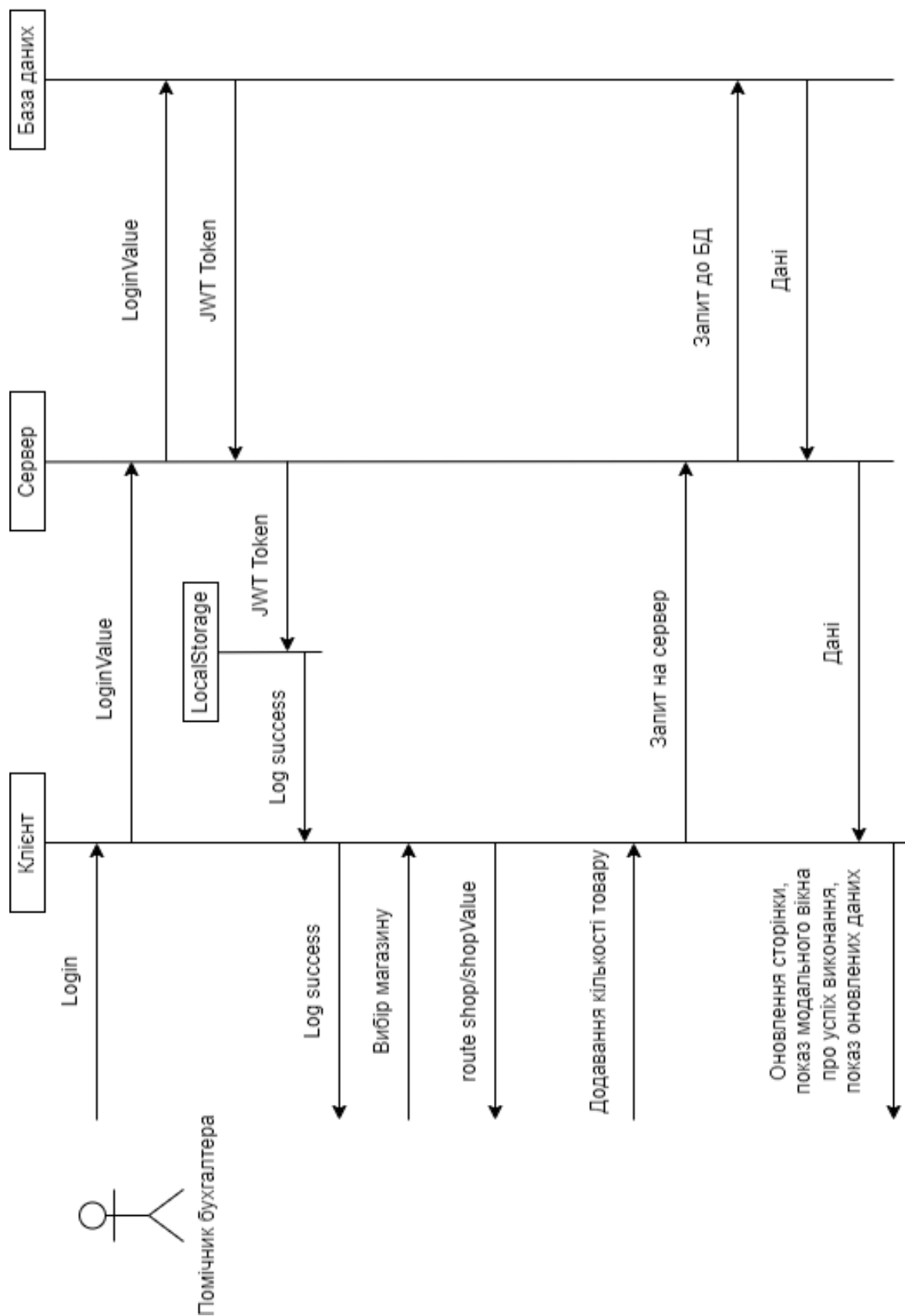


Рис. 2.5. Діаграма послідовності

2.2. Рішення з математичного забезпечення

У базі даних зберігатиметься інформація щодо товарів, які розподілені між собою за різними торговими точками. При цьому користувач зможе дивитися кількість кожного товару для кожного магазину окремо, а також загальна кількість товарів, які на даний момент є у компанії. Існують ситуації, коли необхідно зробити транспортування товару з одного магазину на інший, при цьому в базі даних також має оновитися дана інформація. Так само, помилково, величезна кількість товару прибула в одну з торгових точок. Помічник бухгалтера зможе зробити розподіл за допомогою системи, яка визначить, у яких торгових точках на даний момент недостатня кількість товарів. Передбачається створення можливості транспортування товару і в базі даних.

Для вирішення цієї проблеми можна використовувати теорію графів Дейкстри або алгоритм пошуку найкоротшого шляху. Суть алгоритму Дейкстри полягає в наступному – рухатися завжди через найкоротше ребро, з найменшою вагою [12]. Опишемо сам алгоритм Дейкстри покроково:

1. Початок представлено формулою 1.

$$d(v_1) = 0; d(v_i) = \infty; y = v_1 \quad (1)$$

де d – відстань, v_1 – початкова вершина.

2. Для сусідніх вершин на кожному кроці знаходимо найкоротший шлях.
3. Для наступного кроку обираємо вершину з ребром, що має найменшу вагу. Якщо потрібно, то перераховуємо відстані. Перерахунок відстані для вершин v_i , суміжних з y , виконується за допомогою формули 2.

$$d(v_i) = \min \{ d(v_i), d(y) + l(y, v_i) \} \quad (2)$$

де l – вага ребра, що інцидентне обом вказаним вершинам і повторюємо. [4]

Надаємо кожну торгову точку як вершину графа, а кількість товару необхідного виду та відстань між торговими точками – як ребра. Потім, для використання алгоритму Дейкстри, необхідно визначити початкову вершину графа (наприклад, торгова точка, в яку помилково попрямувала велика

кількість товарів) і запустити алгоритм. Він шукатиме найкоротший шлях від початкової вершини до решти вершин і поверне відстані від початкової вершини до кожної іншої вершини у графі.

В нас є п'ять торгових точок. Вага кожного ребра – кількість товару перемножена на коефіцієнт відстані між торговими точками (воно визначалось окремо за допомогою відстані, часу та ціни за одне перевезення). Коефіцієнти між магазинами представлені в таблиці 2.5

Таблиця 2.5. Таблиця коефіцієнтів зв'язків між магазинами.

	A	B	C	D	E
A		1,5		1,2	2,1
B			1,2	1,75	
C	2			2,3	
D					1
E		1	2		

Як приклад, візьмемо 40 одиниць товару, яку потрібно транспортувати. Перемножмо її на коефіцієнти та отримаємо. Представлено в таблиці 2.6.

Таблиця 2.6. Вага зв'язків.

	A	B	C	D	E
A		60		48	84
B			48	70	
C	80			92	
D					40
E		40	80		

Початковий магазин – С. Утворений граф Дейкстри представлено на рисунку 2.6.

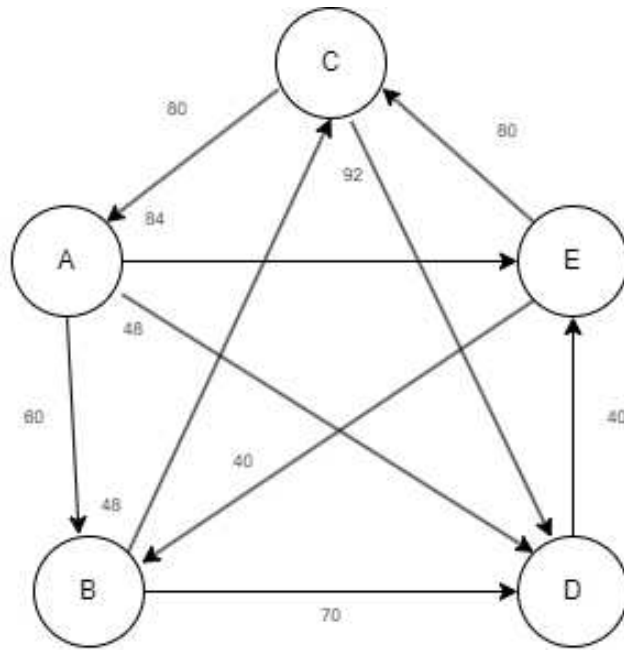


Рис. 2.6. Граф Дейкстри

Приклад розрахунку представлено в таблиці 2.6

Таблиця 2.6. Вага зв'язків.

Ітерація	S	w	D[A]	D[B]	D[D]	D[E]
Початок	{C}	-	80	-	92	-
1	{C,A}	A	-	140	128	164
2	{C,D}	D	-	-	-	132
4	-	-	80	140	92	132

2.3. Програмне забезпечення

Реалізація ІС передбачає собою задіяння певного програмного забезпечення.

Архітектурний шаблон програмного забезпечення. Архітектура клієнт-сервер – це домінуюча концепція у створенні розподілених мережних додатків і передбачає взаємодію та обмін даними між ними.

Вибір СУБД.

PostgreSQL – це об'єктно-реляційна система управління базами даних (ORDBMS). PostgreSQL – це нащадок оригінального коду Берклі з відкритим вихідним кодом. Він підтримує більшу частину стандарту SQL і пропонує безліч сучасних функцій:

- складні запити;
- зовнішні ключі;
- тригери;
- оновлювані уявлення;
- транзакційна цілісність;
- багатOVERсійний паралельний контроль.

Крім того, PostgreSQL може розширюватись користувачем різними способами, наприклад, шляхом додавання нових:

- типи даних;
- функції;
- оператори;
- агрегатні функції;
- індексні методи;
- процедурні мови [13].

Це і є перевагою цієї СУБД над іншими.

Код створення таблиць подань у додатку Б.

Сервер

TypeScript – це мова, яка є розширеною версією JavaScript, перевіряє програму на наявність помилок перед виконанням та робить це на основі типів значень.

Фреймворк. Розвиток технологій передбачає виникнення фреймворків (програмна платформа, що визначає структуру програмної системи) на базі JS та інших мов, які створені для полегшення роботи. **NestJS** - це платформа для реалізації ефективних та масштабованих серверних програм Node.js.

Node.js - це кросплатформове середовище виконання JavaScript з відкритим вихідним кодом. Коли Node.js виконує операцію введення-виводу, наприклад, читання з мережі, доступ до бази даних або файлової системи, замість блокування потоку і витрати циклів ЦП на очікування Node.js відновить операції, коли прийде відповідь [14].

Під капотом Nest використовує надійні платформи HTTP-сервера, такі як Express.

Express.js, або просто Express — програмний каркас розробки серверної частини для Node.js[15]. Програма запускає сервер і слухає з'єднання на порт, видає необхідні дані за допомогою методів `get()`, `post()`, що адресовані маршруту.

ORM (Object-relational mapping, Об'єктно-реляційна проекція) Sequelize - це інструмент Node.js для роботи з різними СУБД, зокрема і з PostgreSQL. Він має надійну підтримку транзакцій, активне та відкладене завантаження, реплікацію читання тощо. Цей ORM передбачає виконання усіх запитів, які існують у SQL та встановлює характеристики (`where`), функції (`fn(SUM)`) тощо, наприклад:

- `create` – створює запит `INPUT`;
- `update` – створює запит `UPDATE`;
- `destroy` – створює запит `REMOVE`;
- `findAll` – створює запит `SELECT * FROM <Name Table>;`

- `findOne` – створює запит `SELECT * FROM <Name Table>` та виводить перший результат тощо.

`SwaggerUIExpress` – це модуль дозволяє обслуговувати автоматично згенеровані `swagger-ui` документи API з Express на основі файлу `swagger.json`. Результатом є жива документація для API, розміщена на сервері API через маршрут [16].

Код надано у додатку В.

Клієнт

Мова розмітки та зовнішнього уявлення. HTML – це мова розмітки, яка структурує зміст сторінки, а CSS – мова опису зовнішнього уявлення, яка створює зовнішній вигляд. Для прискорення роботи при реалізації використовуємо препроцесор SCSS.

Мова програмування. JavaScript або JS – це динамічна та об’єктно-орієнтована прототипна мова програмування. Вона допомагає створювати динамічно поновлювальний контент, керувати мультимедією, роботи анімації та інше [17].

Фреймворк. Vue – це фреймворк, який будує веб-додаток з компонентів. Компоненти створені зі стилів, скрипта та шаблону. При цьому усі компоненти можна поєднувати, викликати один з одного. Однією з переваг саме Vue є висока реактивність. При зміні даних у полі вводу користувачем, Vue одразу перехоплює ці зміни та передає клієнту ці дані.

`Vuetify` – це потужна інфраструктура компонентів Vue. Колекція компонентів інтерфейсу користувача підтримує єдиний стиль додатка і пропонує достатньо можливостей налаштування для будь-якого варіанту використання [18]. У проекті використовуємо можливості цієї бібліотеки для таблиць та алертів.

Приймання даних від серверу. Приймання даних з серверу відбувається за допомогою бібліотеки `swagger-typescript-api`. За допомогою цієї бібліотеки відбувається генерація готового API клієнта для спрощеного звернення до сервера. Генерація відбувається за допомогою команди:

```
swagger-typescript-api -p http://localhost:3000/api/ELIZTAN-  
json -o ./src/api -n api.client.js --api-class-name ApiClient --js  
--axios
```

Після цього у клієнта створюється два файли: `api.client.d.ts` `api.client.js`. Після підключення його до проекту, можна використовувати клас `api` з цих файлів для спрощених запитів з серверу. Також, в цієї команди вказано використання саме `Axios`, а не `Fetch`. Ця бібліотека з відкритим вихідним кодом, вона надає основні методи передачі через HTTP-запити – `get()`, `post()`. При роботі з `Axios` ми маємо прямий доступ до JSON (javascript object notation), який ми отримали в результаті прийому інформації з сервера. Перевага `Axios` над `Fetch` – отримання мережевої помилки при якихось збоях при запиті.

Структуроване збереження даних. `VueX` – це патерн налаштування станів, а якщо говорити звичайною мовою – це сховище даних, яке представляє собою три основні складові: дії, мутації та стани. Саме за допомогою `Vuex` та її модульного підходу вийде структурувати дані з серверу.

Код надано у додатку Г.

Роутінг. `VueRouter` - є офіційним маршрутизатором для `Vue.js`. Він глибоко інтегрується з ядром `Vue.js`, щоб зробити створення односторінкових програм за допомогою `Vue.js` легким. Особливості включають:

- Відображення вкладених маршрутів;
- Динамічна маршрутизація;
- Модульна компонентна конфігурація маршрутизатора;
- Параметри маршруту, запит, символи підстановки;
- Перегляньте ефекти переходу за допомогою системи переходів `Vue.js`;
- Точне керування навігацією;
- Посилання на автоматичні активні класи `CSS`;
- Режим історії `HTML5` або режим хешування;
- Настроювана поведінка прокручування;
- Правильне кодування `URL`-адрес [19].

Середовище розробки. Для розробки проекту використовуємо середовище розробки JetBrains WebStorm. WebStorm – це інтегроване середовище розробки для JavaScript та пов’язаних із ним технологій. Робота в цьому середовищі полегшується великою кількістю автоматизації рутинної роботи, більш приємним інтерфейсом та врегульованою системою співпраці серверу та клієнту [20].

Операційні системи. Розробка додатку передбачає використання засобів операційних систем.

Linux – це назва UNIX-подібних операційних систем, основою якого є однойменне ядро. Використовуємо дистрибутив Mint, який існує на основі Ubuntu та Debian. Основними перевагами Linux Mint є такі характеристики:

- Легка у розумінні та застосуванні;
- Повна мультимедійність;
- Безкоштовна у використанні;
- зручний інтерфейс;
- Встановлені заздалегідь та підтримка зазвичай ПЗ;
- Повна готовність до роботи із самого початку [21].

Типові програмні засоби. Для перегляду веб-програми нам знадобиться певний браузер. Вибираємо Chrome. Chrome - це безкоштовний веб-браузер, розроблений компанією Google на основі браузера з відкритим кодом Chromium та іншого відкритого програмного забезпечення. У браузер включений зневаджувач JS, консоль JS і менеджер завдань, в якому можна дізнатися, скільки пам'яті використовує кожна вкладка [22].

Draw.io – це інструмент проектування діаграм, блок-схем, інтелект-карт, відносини сутностей, програмних блоків та інше. Використовується цей сервіс для полегшення роботи з заздалегідь створеними шаблонами, програма володіє багатим набором функцій для візуалізації більшості завдань.

Сервісне програмне забезпечення. Захист інформації відбувається за допомогою антивірусного програмного забезпечення (далі ПЗ) Avast. Avast

може працювати на більшості ОС (Windows, macOS, Linux, а так на Android та iOS, що нам і потрібно). Можливості програми включають в себе перевірку комп'ютера на віруси, блокування шкідливих скриптів, резидентний захист незалежними видалення шпигунських ПЗ, підтримка голосового повідомлення, якщо виявляється вірус та багато інших. Тобто, avast виконає більшість функцій по захисту системи від зовнішніх та внутрішніх загроз.

2.4. Технічне забезпечення проекту

Системні вимоги щодо клієнтської частини реалізації:

- Процесор – починаючи з Intel® Core™ i3-10100F CPU @ 3.60 GHz
- Тип системи – 64-розрядна;
- Оперативна пам'ять – від 4Гб;
- Відеокарта – від 256 mb;
- Монітор – 1920 x 1080;
- Клавіатура;
- Миша;
- Місце на жорсткому диску – 4 гб;
- Мережевий адаптер – Intel® Ethernet Connection (12) I219-V.

Системні вимоги щодо серверній частини реалізації:

- Процесор – починаючи з Intel® Core™ i3-10100F CPU @ 3.60 GHz;
- Тип системи – 64-розрядна;
- Оперативна пам'ять – від 4Гб;
- Відеокарта – від 256 mb;
- Монітор – 1920 x 1080;
- Клавіатура;
- Миша;
- Місце на жорсткому диску – 5 тб;
- Мережевий адаптер – Intel® Ethernet Connection (12) I219-V.

Оглядаючи системні вимоги майбутньої реалізації, наведемо технічне забезпечення.

- Системний блок: QUBE QB Miditower 500Вт;
- Місце на жорсткому диску: 6 Тб.
- Процесор: Intel Core i3-10100F 4 ядра 3,6 (4,3) ГГц;
- Пам'ять: DDR4 SSD 1Тб 8 гб 2666 МГц;
- Відеокарта: Nvidia GeForce GTX 1650 4 гб;
- Миша: MSI Cultch GM11;

- Клавіатура: RZTK Mamba KTL Kalih;
- Монітор: HP M27fd;
- Мережевий адаптер – Intel® Ethernet Connection (12) I1219-V.

Так само для роботи комп'ютера, де знаходиться сервер, при стрибках напруги (щоб техніка не виходила з ладу через це), він має бути оснащений стабілізатором потужності – Logic Power.

Підключення відбувається за допомогою технології Ethernet, розповсюдження сигналу – за мережевого проводу. Комутатор – PowerLink SP401F. Швидкість підключення (перевірено за допомогою сервісу 2ip.ua) – 82.70 Мбіт/сек.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІС

3.1. Реалізація – сервер

Сервер передбачає використання бібліотеки SwaggerUIExpress, яка створює зручну документацію усіх існуючих до бази даних запитів. Запити до сервера представлені в таблиці 3.1.

Таблиця 3.1. Вага зв'язків.

<i>Запит</i>	<i>Тип запиту</i>
Авторизація	
Авторизація	POST
Реєстрація	POST
Отримання авторизованого користувача	GET
Вийти з профілю	GET
Користувач	
Отримання всіх користувачів	GET
Отримання користувача за айді	GET
Видалити користувача за айді	DELETE
Оновлення інформації	PATCH
Оновлення ролі користувача	PATCH
Оновлення магазину	PATCH
Встановити аватар користувачу	POST
Видалити аватар користувачу	DELETE
Категорія	
Створення категорії	POST
Отримання усіх категорій	GET
Видалити категорії за айді	DELETE
Отримання категорії за айді	GET
Підкатегорія	
Створення підкатегорії	POST

1	2
Отримання усіх підкатегорій	GET
Оновлення даних підкатегорії	PATCH
Видалити підкатегорії за айді	DELETE
Отримання підкатегорії за айді	GET
Постачальник	
Створення постачальника	POST
Отримання усіх постачальників	GET
Оновлення даних постачальника	PATCH
Видалити постачальника за айді	DELETE
Отримання постачальника за айді	GET
Ціна	
Створення ціни	POST
Отримання усіх цін	GET
Оновлення даних ціни	PATCH
Видалити ціну за айді	DELETE
Отримання ціни за айді	GET
Назва товару	
Створення назви товару	POST
Отримання усіх назв товару	GET
Оновлення даних назви товару	PATCH
Видалити назви товару за айді	DELETE
Отримання назви товару за айді	GET
Товар	
Створення товару	POST
Отримання усіх товарів	GET
Переміщення товару з одного магазину в інший	POST

1	2
Оновлення даних товару	PATCH
Видалити товару за айді	DELETE
Отримання товару за айді	GET
Магазин	
Створення магазину	POST
Створення товару та магазину	POST
Створення коефіцієнту магазинів	POST
Отримання усіх магазинів	GET
Оновлення кількості товарів	PATCH
Оновлення даних товарів	PATCH
Видалити магазин за айді	DELETE
Отримання магазину за айді	GET
Сховище накладних	
Створення сховища	POST
Створення зв'язку сховища та магазину	POST
Завантажити накладну	POST
Отримання усіх накладних	GET
Отримання накладної за датою створення/постачальником	GET
Оновлення статусу накладної	PATCH
Видалити файл накладної	DELETE
Видалити накладну за айлі	DELETE

На рисунку 3.1-3.2 представлено приклад категорії та запиту, які реалізовані на сервері за допомогою Swagger API.

Авторизація	
POST	/auth/login Login користувача
POST	/auth/registration Реєстрація користувача
GET	/auth Отримання авторизованого користувача
GET	/auth/exit Вийти з профілю
Користувач	
GET	/user Отримання всіх користувачів
GET	/user/{id} Отримання користувача за айді
GET	/user/{slug} Отримання користувача за slug
DELETE	/user/{userId} Видалення за айді
PATCH	/user/{idUser} Оновлення інформації щодо користувача
PATCH	/user/role/{idUser}/{role} Оновлення ролі користувача
PATCH	/user/shop/{idUser}/{idShop} Оновлення магазину
DELETE	/user/shop/remove/{idUser} Видалити магазин у користувача
POST	/user/avatar/{idUser} Встановити аватар користувачу
DELETE	/user/avatar/remove/{idUser} Видалити аватар

Рис. 3.1. Категорії Авторизація та Користувач

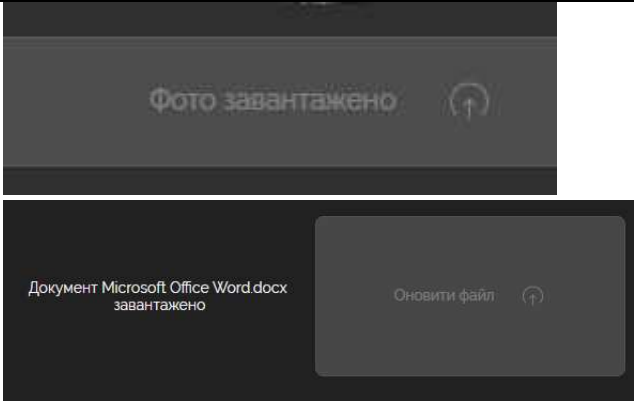
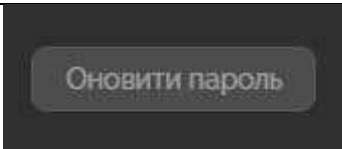
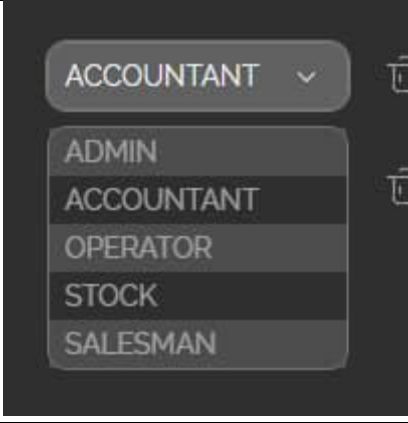
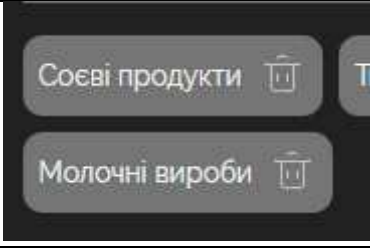
POST	/product/shopFrom/{idShopFrom}/shopTo/{idShopTo}/amount/{amount}
Parameters	
Name	Description
idShopFrom * required number (path)	idShopFrom
idShopTo * required number (path)	idShopTo
idProduct * required number (path)	idProduct
amount * required number (path)	amount

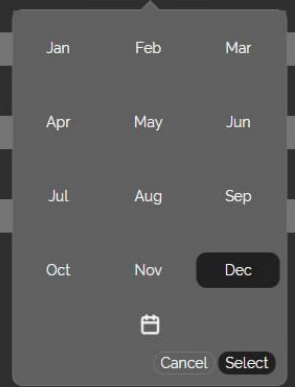
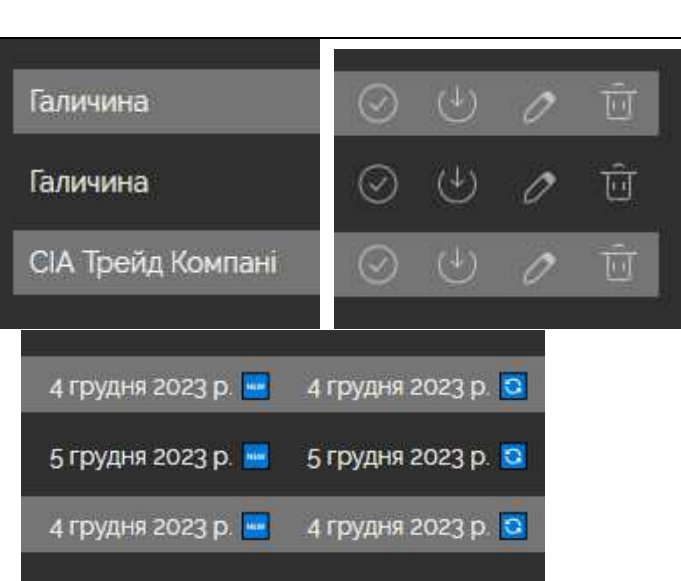
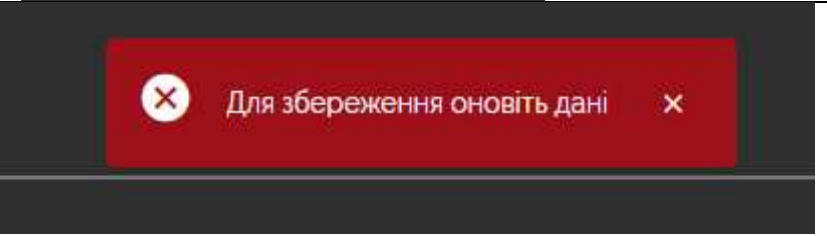
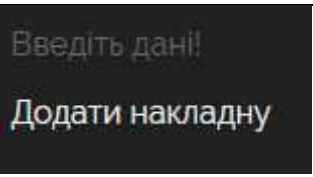
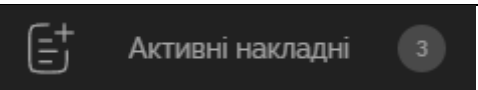
Рис. 3.2. Запит переміщення кількості товару з одного магазину до іншого

3.2. Реалізація – клієнт

Так як додаток побудований на основі Vue js, який має компонентний підхід побудови програми, то елементи графічного інтерфейсу, які використовуються постійно, представлено у таблиці 3.2.

Таблиця 3.2. Вага зв'язків.

Зовнішній вигляд	Функція
	Input file – елемент для завантаження файлів різних видів (фото, документи).
	Button – елемент кнопки для подій натискання (відкриття модальних вікон, надсилання інформації на сервер тощо)
	Dropdown list - випадаючий список з необхідними елементами (у цьому випадку, вибір ролі користувача у системі)
	Selected item - відображає вибрані елементи зі списку.

		Dropdown list date - випадаючий список для вибору дати.
		Toggle theme – перемикач темної та світлої теми.
		Item list – елемент списку з датою занесення до бази даних, датою оновлення цих даних, елементами зміни статусу, завантаження файлу, редагування та видалення даних.
		Main alert – основне попередження на сторінці.
		Sub alert – дочірнє попередження у модальних вікнах.
		Notification – повідомлення про нову накладну у системі.

У клієнті реалізовані багато функцій, які використовують логіку програми, а саме:

– Зміну кольорової схеми додатку з світлої на темну та навпаки.
Представлено на рисунку 3.3 та 3.4

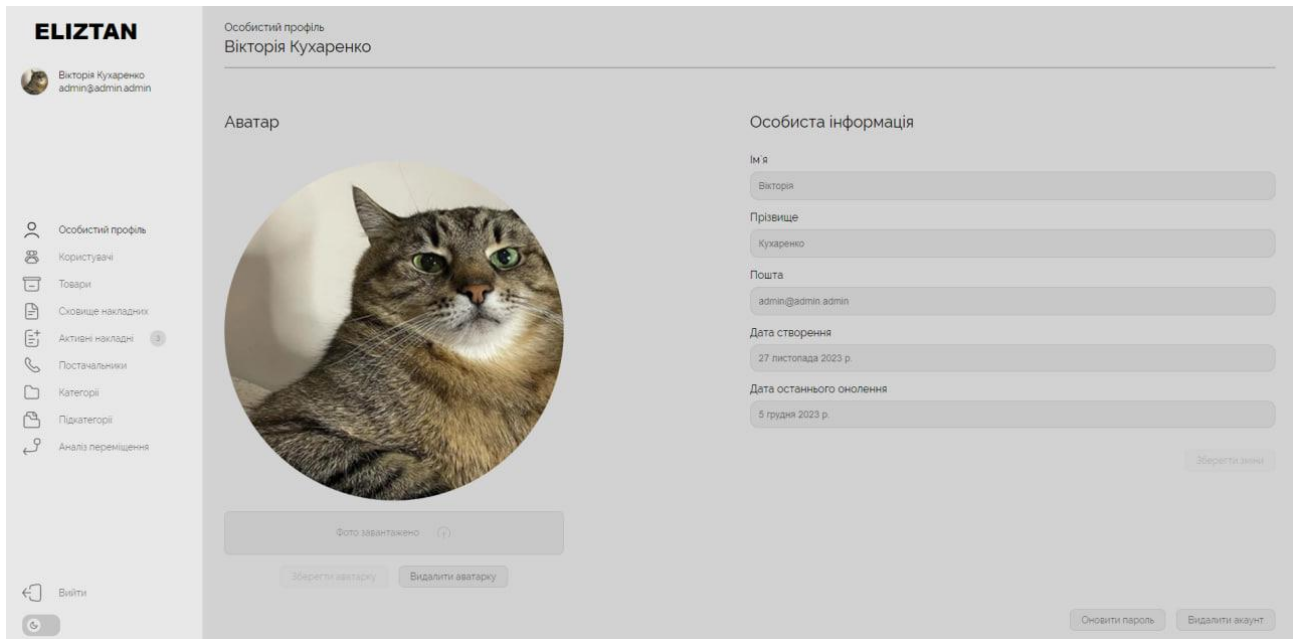


Рис. 3.3. Світла тема

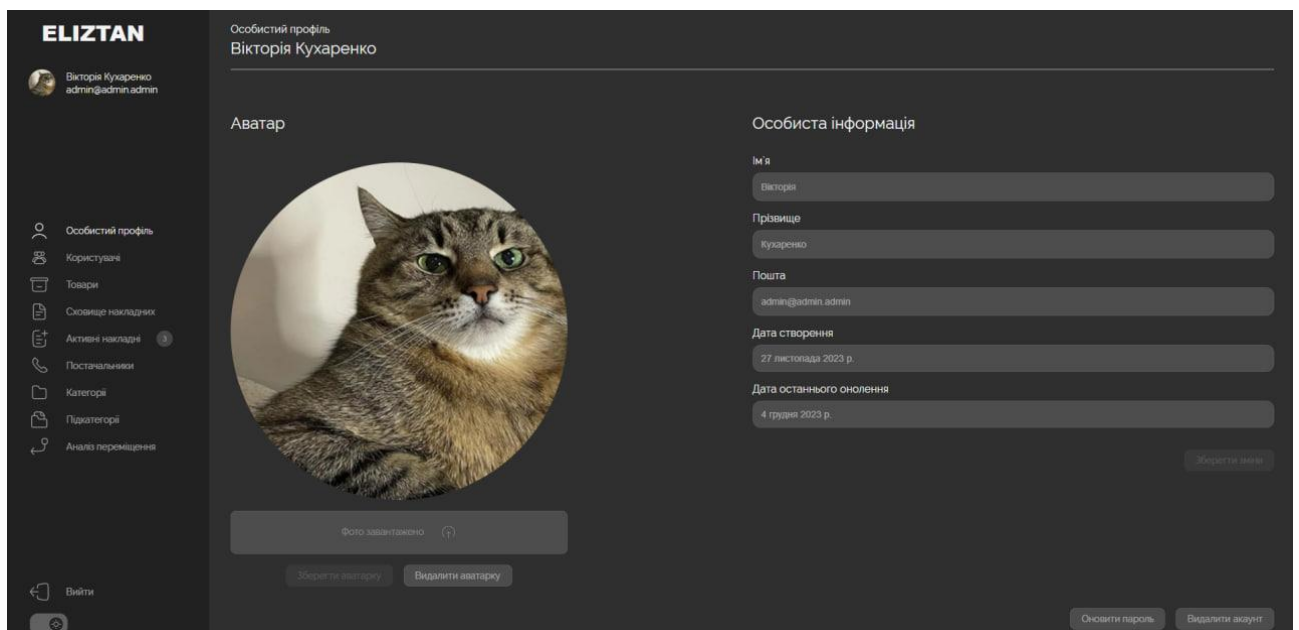


Рис. 3.4. Темна тема

– Авторизація користувача та відображення елементів меню за роллю користувача. Представлено на рисунках 3.5-3.7.

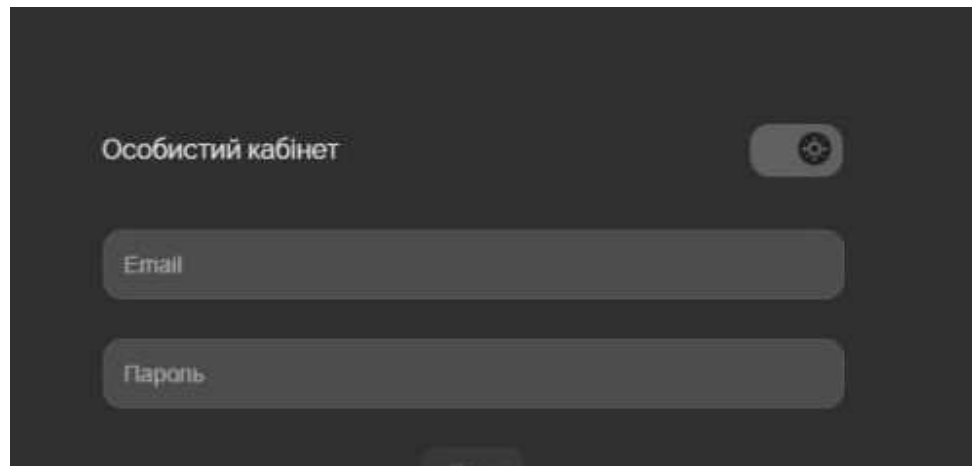


Рис. 3.5. Авторизація (роут /)

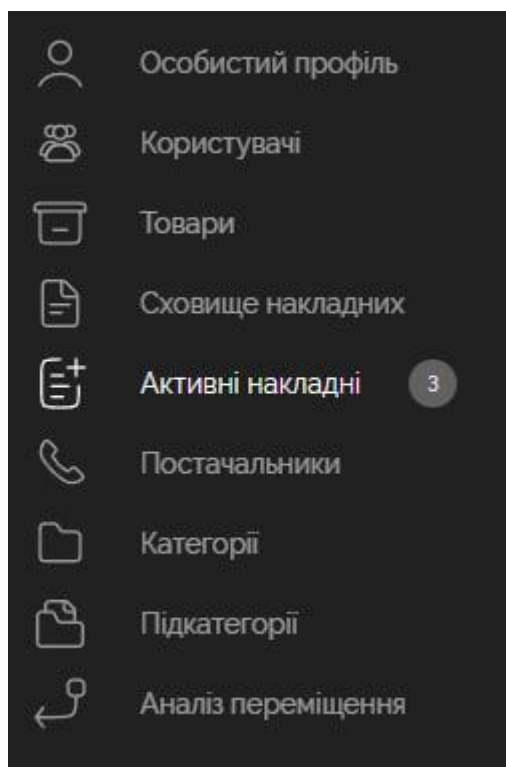


Рис. 3.6. Вигляд меню при ролі
ADMIN

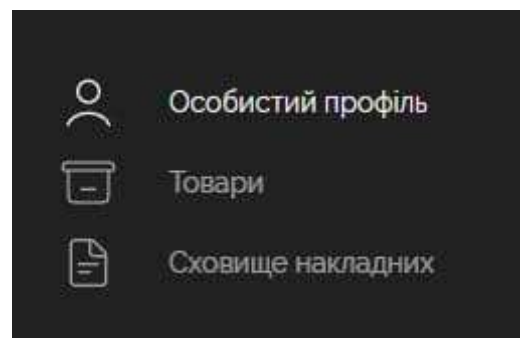


Рис. 3.7. Вигляд меню при ролі
SALESMAN

Так як роль користувача має визначати, які блоки додатку може відвідувати користувач, то наведемо приклади кожної ролі (без ADMIN, бо він має усі режими доступу та окремий до категорії *Користувачі*, та SALESMAN, бо є на рисунку 3.7):

- ACCOUNTANT – Особистий профіль, Товари, Сховище накладних, Активні накладні, Категорії, Підкатегорії, Аналіз переміщення.
- OPERATOR – Особистий профіль, Сховище накладних, Постачальники.
- STOCK – Особистий профіль, Активні накладні.
- Модальні вікна. Представлено на рисунках 3.8-3.16.

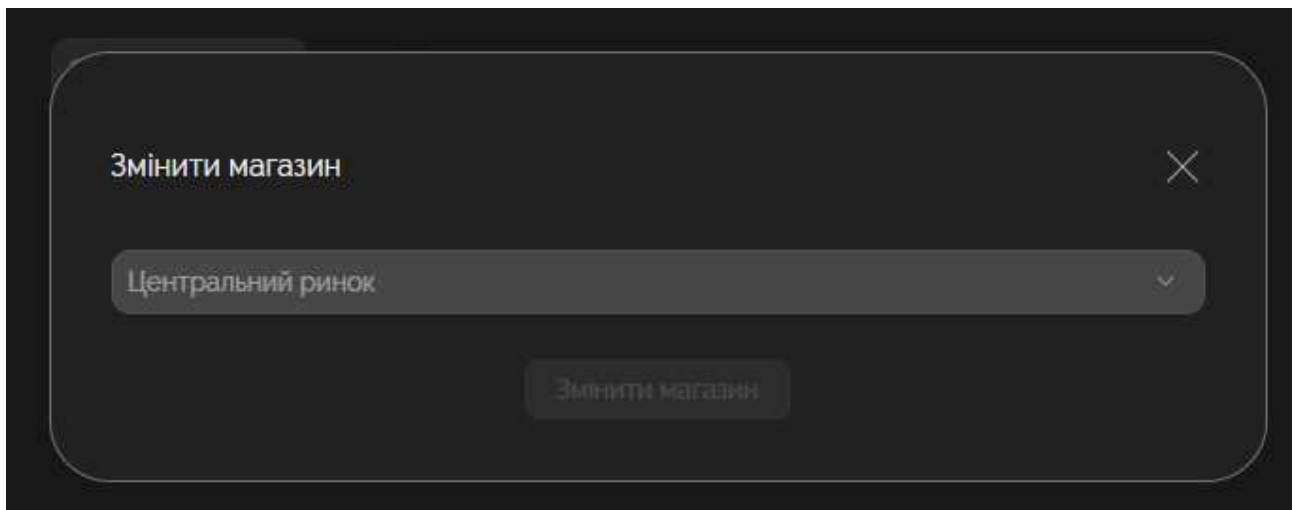


Рис. 3.8. Модальне вікно зміни магазину у користувача

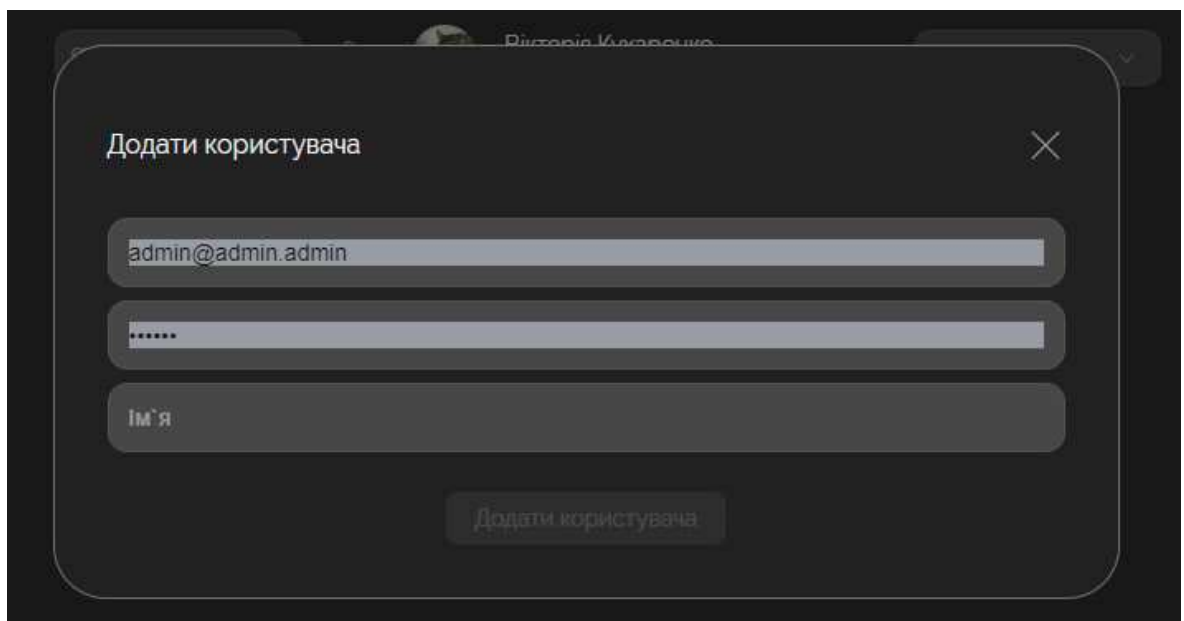


Рис. 3.8. Модальне вікно реєстрації нового користувача

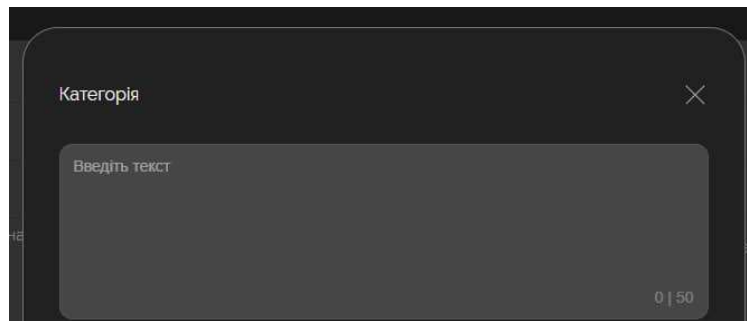


Рис. 3.9. Модальне вікно додавання/редагування категорії

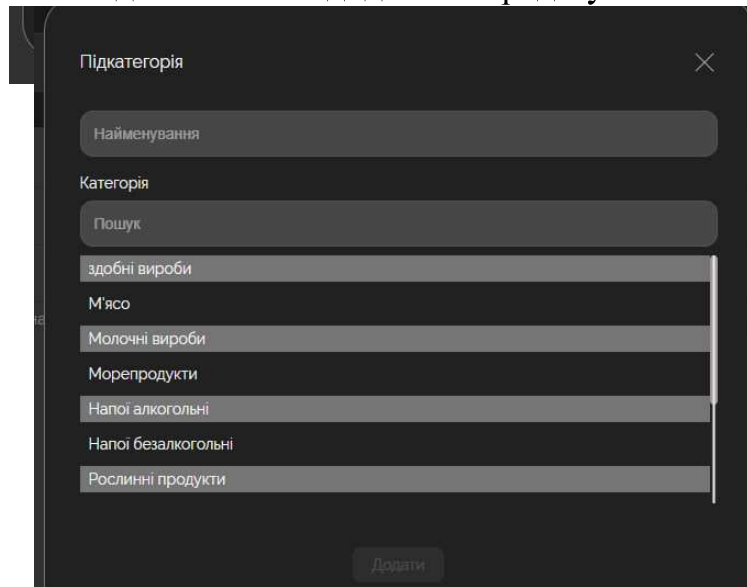


Рис. 3.10. Модальне вікно додавання/редагування підкатегорії

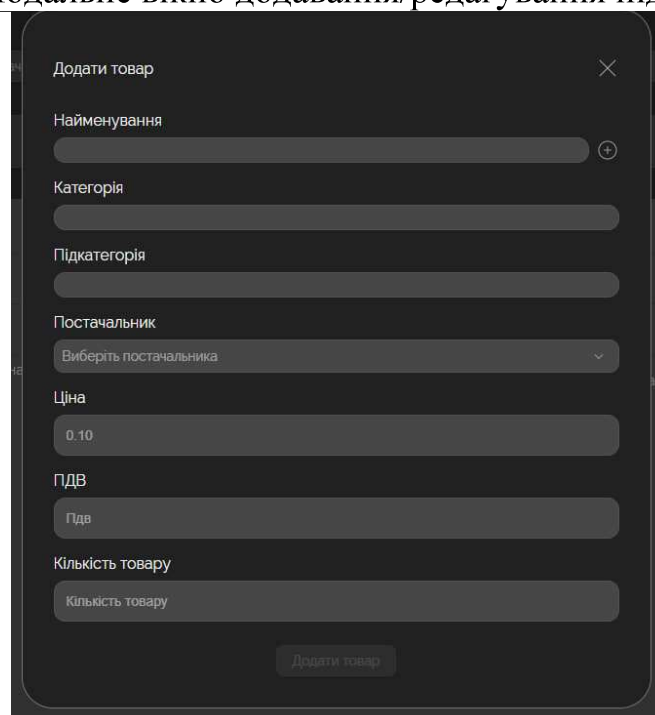
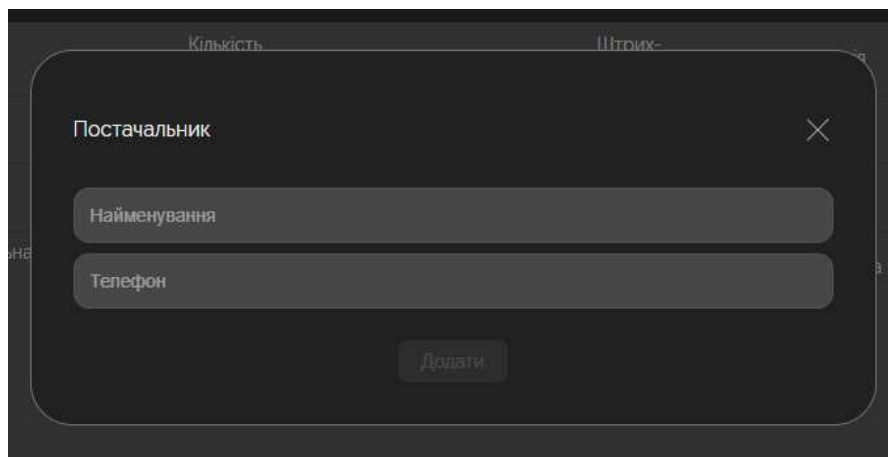


Рис. 3.11. Модальне вікно додавання товару



Кількість Штрих-

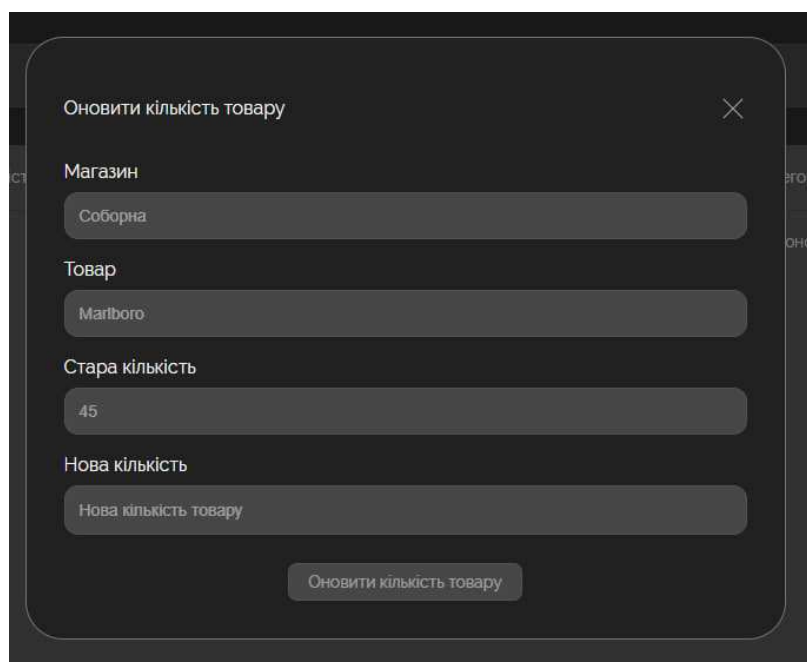
Постачальник

Найменування

Телефон

Додати

Рис. 3.12. Модальне вікно додавання постачальника



Оновити кількість товару

Магазин

Соборна

Товар

Marlboro

Стара кількість

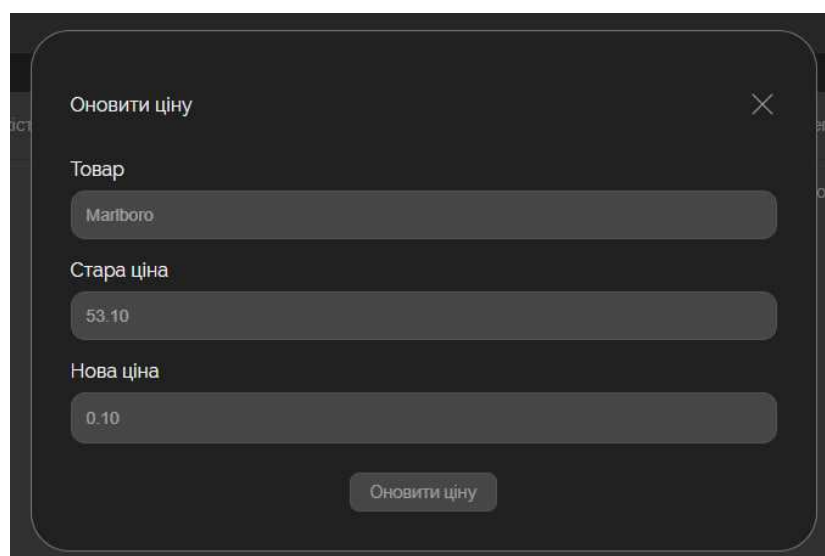
45

Нова кількість

Нова кількість товару

Оновити кількість товару

Рис. 3.13. Модальне вікно оновлення кількості товару



Оновити ціну

Товар

Marlboro

Стара ціна

53.10

Нова ціна

0.10

Оновити ціну

Рис. 3.14. Модальне вікно оновлення ціни товару

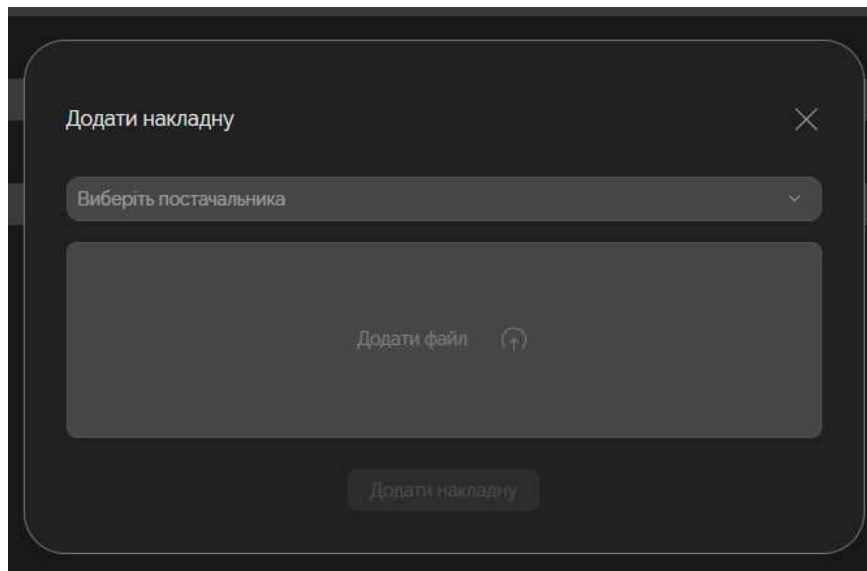


Рис. 3.15. Модальне вікно додавання товару

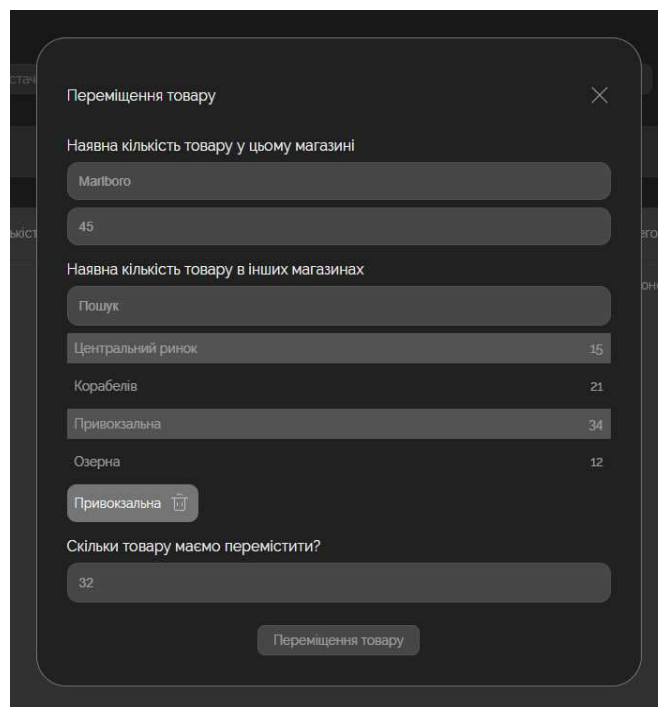


Рис. 3.16. Модальне вікно переміщення товару

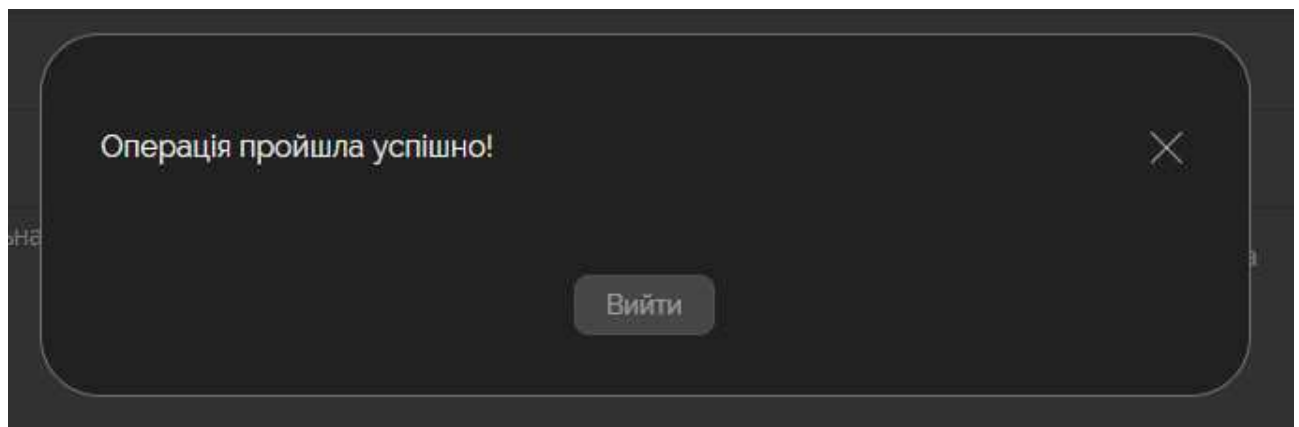


Рис. 3.17. Модальне вікно успіху операції після оновлення сторінки

— Усі доступні у додатку вікна. Представлено на рисунках 3.18-3.27.

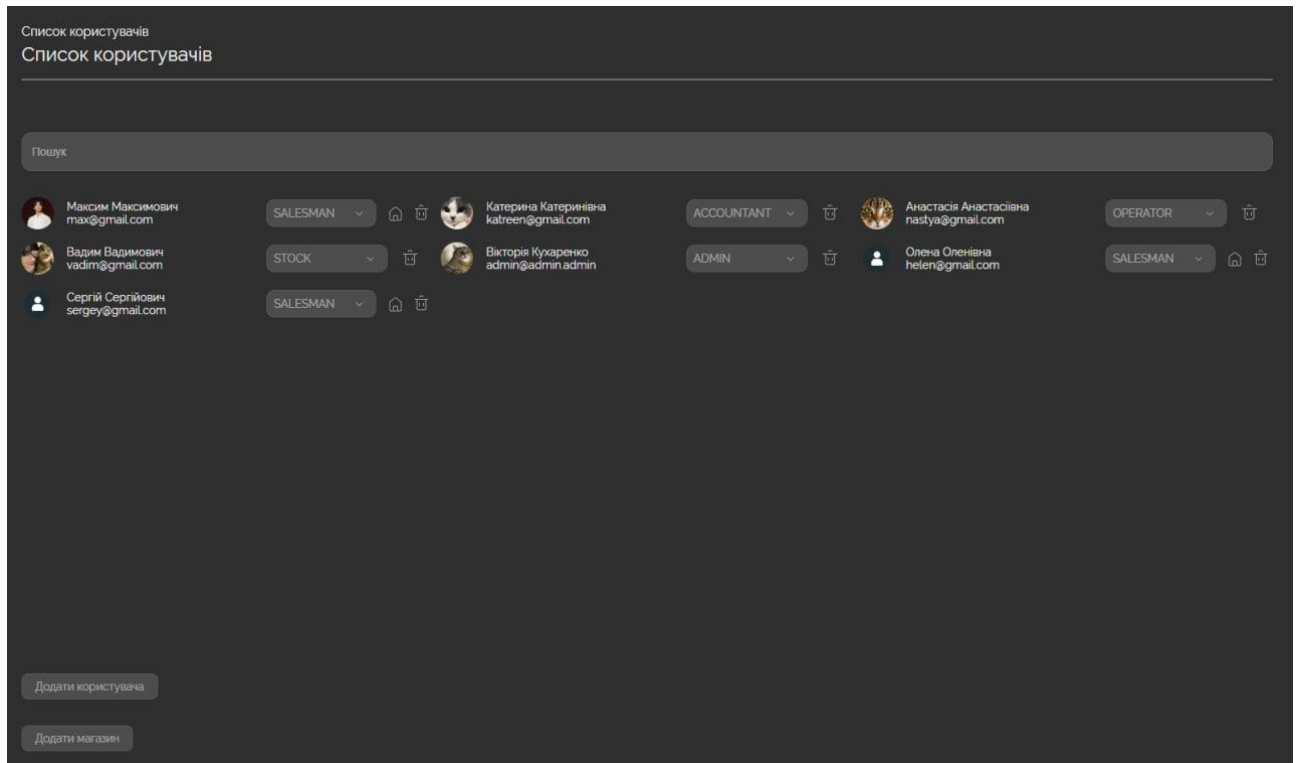


Рис. 3.18. Вікно користувачів. Передбачає оновлення ролей користувачів, надавання продавцям магазинів.

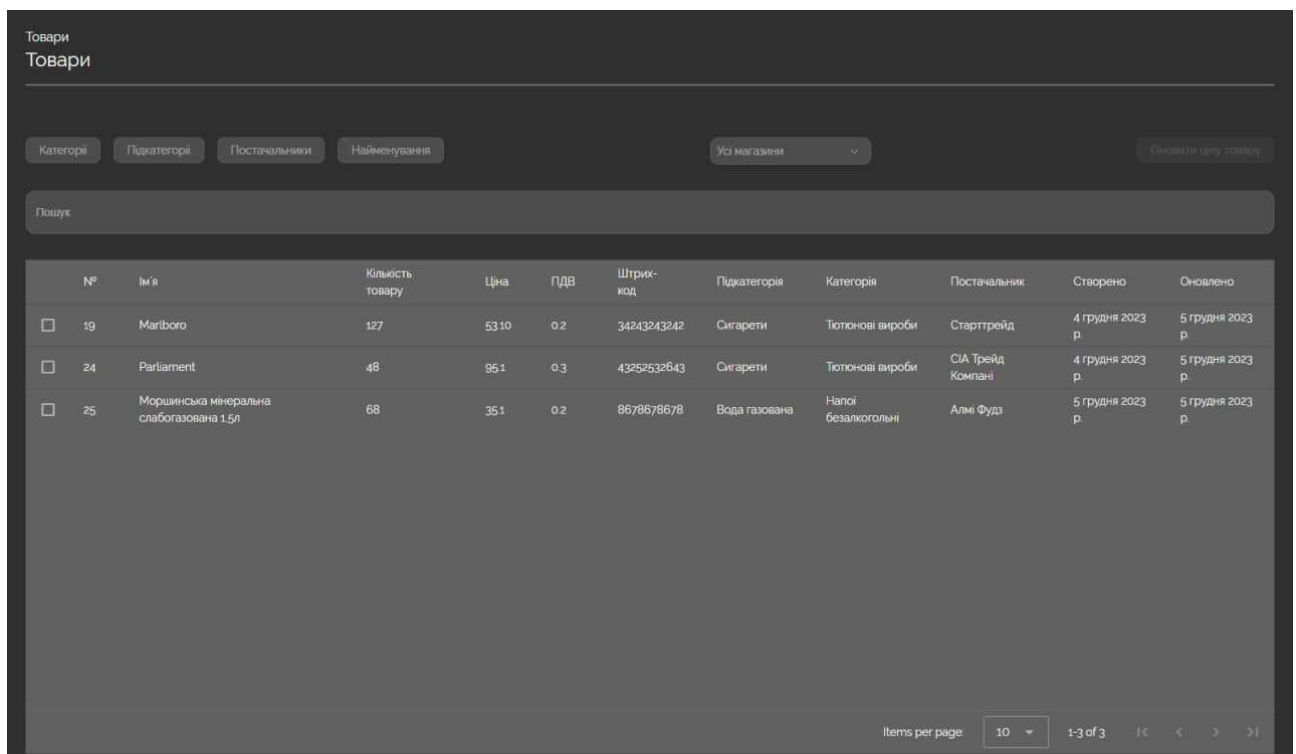


Рис. 3.19. Вікно товарів усіх магазинів. При переході на це вікно продавцем відображається тільки товари його магазину та доступна кнопка «Оновлення кількості товару»

Товари

Озерна

Категорія

Підкатегорія

Постачальники

Найменування

Озерна

Додати товар

Оновити кількість товару

Оновити ціну товару

Переміщення товару

Пошук

	№	Ім'я	Кількість товару	Ціна	ПДВ	Штрих-код	Підкатегорія	Категорія	Постачальник	Створено	Оновлено
☐	19	Marlboro	12	53.10	0.2	34243243242	Сигарети	Топінові вироби	Старттрейд	4 грудня 2023 р.	5 грудня 2023 р.
☐	24	Parliament	15	95.1	0.3	43252532543	Сигарети	Топінові вироби	СІА Трейд Компані	4 грудня 2023 р.	5 грудня 2023 р.
☐	25	Моршинська мінеральна слабгазована 1.5л	12	35.1	0.2	8678678678	Вода газована	Напої безалкогольні	Алім Фуз	5 грудня 2023 р.	5 грудня 2023 р.

Items per page

10

1-3 of 3

<

>

Рис. 3.20. Вікно товарів вибраного магазину.

Сховище накладних
Магазин

Виберіть постачальника Додати накладну Скинути фільтри Додати накладну

Адел груп	Центральний ринок	4 грудня 2023 р.	5 грудня 2023 р.	📄	📄	⬇️	✎	🗑️
Галичина		4 грудня 2023 р.	4 грудня 2023 р.	📄	📄	⬇️	✎	🗑️
Галичина		5 грудня 2023 р.	5 грудня 2023 р.	📄	📄	⬇️	✎	🗑️
СІА Трейд Компані		4 грудня 2023 р.	4 грудня 2023 р.	📄	📄	⬇️	✎	🗑️
СІА Трейд Компані		4 грудня 2023 р.	4 грудня 2023 р.	📄	📄	⬇️	✎	🗑️

< 1 >

Рис. 3.21. Вікно сховища накладних. Відображаються накладні за типом Store та Stock (в залежності від ролі користувача)

Активні накладні
Магазин

Галичина	4 грудня 2023 р.	4 грудня 2023 р.	📄	📄	✅	⬇️	✎	🗑️
Галичина	5 грудня 2023 р.	5 грудня 2023 р.	📄	📄	✅	⬇️	✎	🗑️
СІА Трейд Компані	4 грудня 2023 р.	4 грудня 2023 р.	📄	📄	✅	⬇️	✎	🗑️

Рис. 3.22. Вікно активних накладних. Відображаються накладні за типом Store та Stock (в залежності від ролі користувача)

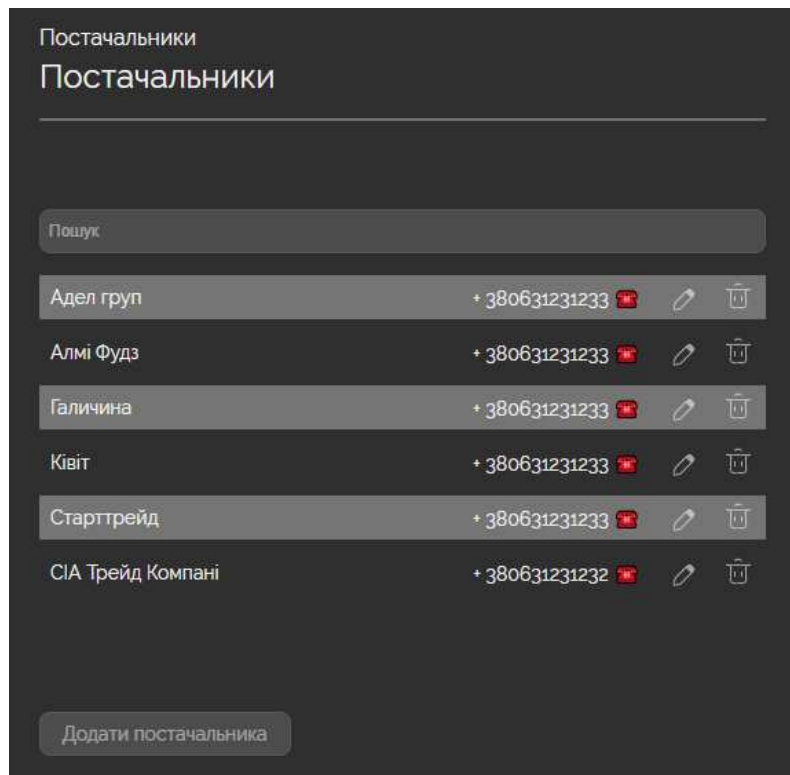


Рис. 3.23. Вікно постачальників

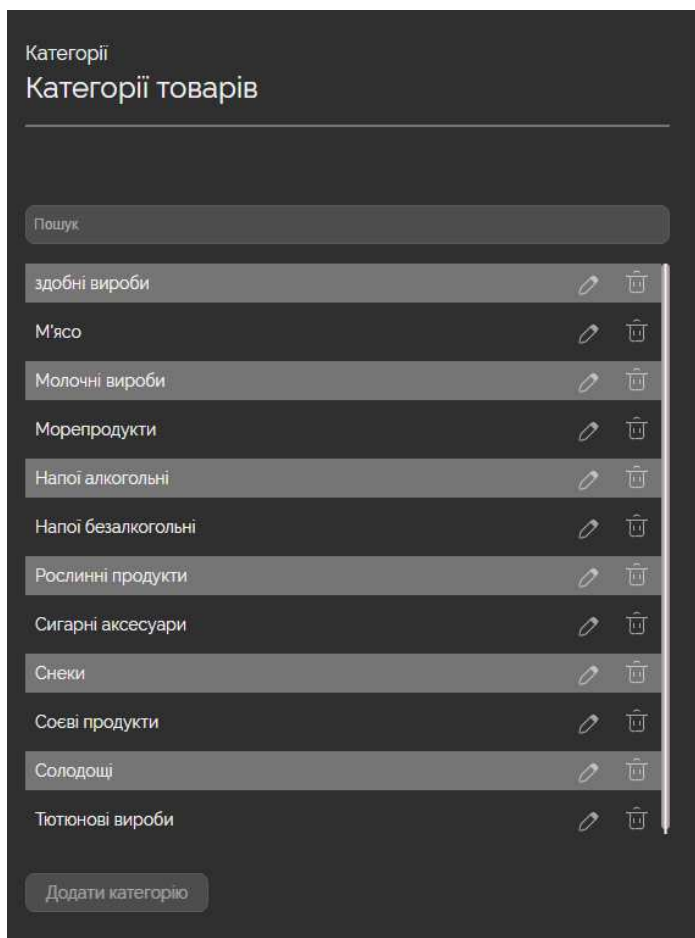


Рис. 3.24. Вікно категорій



Рис. 3.25. Вікно підкатегорій

Аналіз переміщення

Аналіз переміщення

Коефіцієнти між магазинами

№	Назва магазину № 1	Назва магазину № 2	Коефіцієнт
9	Корабелів	Соборна	1
10	Корабелів	Привокзальна	2
1	Озерна	Соборна	15

Items per page:

10

1-10 of 10

<< >>

Додати коефіцієнти

Магазин

Корабелів

Озерна

Привокзальна

Соборна

Центральний ринок

Привокзальна

Кількість товару

40

Знайти шлях

Результат

Чекаємо на Ваші дані

Рис. 3.26. Вікно аналізу переміщення товарів

Привокзальна	0
Озерна	80
Центральний ринок	92
Корабелів	132
Соборна	140

Рис. 3.27. Відображення результатів аналізу переміщення товару

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ ПІД ЧАС НЕЇ

4.1. Поняття охорона праці

За законом, який був введений у Конституцію України, згідно з Постановою Верховної Ради України від 14 жовтня 1992 року, охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [23].

При порушенні законодавчих чи нормативних актів, які стосуються охорони праці, передбачається введення адміністративної (передбачається застосування штрафних санкцій, порушення вимог законодавчих та інших актів тягне за собою штрафи, які стосуються винних осіб), матеріальної (передбачається відшкодування збитків працівникам при виникненню нещасних випадків та профзахворювань) та кримінальної відповідальності (за Кримінальним кодексом України ст. 135, 218, 219 та 220, карається строком виправними роботами на строк до одного року чи штрафом 15 мінімальних розмірів заробітної плати). Прикладами таких порушень є відсутність спеціальних засобів безпеки працівників, відсутність чи не пройдена перевірка техніки безпеки працівників, завдання шкоди чи нараження життя працівників небезпеки під час робочих процесів.

На державному рівні характеризуються такі рівні охорони праці: технічний контроль щодо безпечних умов праці у технологічній сфері роботи працівник, розв'язання завдання охорони праці у різних напрямках політики, таких як економічні, соціальні та екологічні; захист працівників, які стосуються відшкодування витрат при нещасних випадках, захворюваннях, сімейних обставин тощо; підвищення рівня праці за допомогою кваліфікаційних курсів, проведенням лекцій, запровадженням допомоги зі сторони інших підприємств, які пов'язані з тематикою роботи підприємства.

Розділ IV, а саме «Стимулювання охорони праці», встановлює, що відшкодування збитків за порушення правил охорони праці згідно з діючим

законодавством, витрати на рятування потерпілих під час аварії та ліквідації її наслідків, на розслідування її причин, відшкодовує роботодавець [24].

Охорона праці має дуже важливий аспект у процесі формування бізнес-процесів та їх впровадження у систему роботи. Якщо дивитися зі сторони продажу товарів, якій надано дуже багато часу у роботі, у цьому процесі залучено дуже багато працівників, і у кожного з них є свої аспекти, які пов'язані з охороною праці.

Кожний гарний директор має встановити за мету виявлення та встановлення правил щодо охорони праці, які будуть діяти за чинним законодавством України. Під час воєнного стану – тим більше.

Потрібно актуалізувати ці правила, проводити заздалегідь техніки безпеки працівників щодо безпечної роботи та кожний працівник має знати, які дії він має застосувати в тій чи іншій ситуації.

4.2. Аналіз небезпечних факторів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ

Конституція України, Кодекс законів про працю та Закон України «Про охорону праці» – забезпечення захисту життєдіяльності працівників товариства обмеженої відповідальності ЕЛІЗТАН АЛЬЯНС КОМПАНІ.

Так як, при обробці бізнес-процесу «Оновлення наявності товару для продажу» основним прецедентом цього процесу – є помічник бухгалтера, то за аналіз небезпечних та шкідливих факторів беремо саме роботу цього працівника.

Він працює в офісі. Проаналізуємо усі шкідливі виробничі фактори та встановимо ті, які можуть бути під час роботи помічника бухгалтера.

Існує декілька небезпечних та шкідливих виробничих факторів: хімічні, фізичні (електричні, фактори освітлення, шум), біологічні та психофізіологічні.

Фізичні фактори впливу виявляються за допомогою мікроклімату, а саме температури, вологості, вентиляції повітря, випромінювання тощо. Підвищена чи знижена вологість повітря ускладнює роботу тим, що працівник може погано себе почуватися, а також може призвести до поломки електронних

приладів праці, а саме портативних комп'ютерів, принтерів, освітлення тощо. Температура повітря та її зміна може вплинути на самопочуття працівників, а також призвести до теплових ударів чи переохолодження. За ДСН 3.3.6.042-99 Міністерством Охорони Здоров'я України було встановлено такі нормативні показники мікроклімату у приміщені:

- Теплий період року – температура 23-25 градуси Цельсія, вологість при цьому 40-60%;
- Холодний період року – температура 22-24 градуси Цельсія, вологість при цьому 40-60%.

Електричний струм та електричне поле з їх випромінюванням може бути шкідливим чи навіть небезпечним для працівників, які знаходяться в приміщені виконання роботи. Ураження струмом може призводити до різних порушень в організмі. За наказом №239 Міністерством охорони Здоров'я України граничні норми напруженості електричного поля визначаються суто в залежності від частоти/довжини хвилі і режимів цього випромінювання.

Рівень шумів також впливає на роботу працівників. При не нормативних показниках рівня шуму та звуків може погіршуватися гострота слуху, що може призвести до захворювань. Нормування відбувається за допомогою шкали шумоміра А, а допустимими параметрами рівнів звуку не повинні перевищувати 12кГц – $\leq 80\text{дБ}$; 20кГц – $\leq 90\text{дБ}$; 25кГц – $\leq 105\text{дБ}$; 31,5 – 100кГц – $\leq 110\text{дБ}$ (передбачається за ДСТУ 12.1.001-83).

Освітлення приміщення – одне із важливих складових, особливо при роботі із комп'ютером, що за відсотковим відношенням у праці помічника бухгалтера майже 90 відсотків. Погане освітлення чи навпаки дуже велика яскравість пристроїв освітлення може призвести до погіршення зору у працівника, його втомленості та навіть викликати захворювання різного ступеню важкості (синдром сухого ока, близорукість тощо).

Хімічні шкідливі виробничі фактори розділяються на:

- Характер впливу на людину (токсичні, дратівні, сенсibilізуючі, канцерогенні, мутагенні тощо);
- Шлях проникнення у організм людини (через органи дихання, шлунковокишечний тракт, шкіру, слизисті оболонки тощо).

Біологічні небезпечні виробничі фактори складається з об'єктів, а саме: мікроорганізми, продукти їх життєдіяльності, макроорганізми тощо.

Хімічні та біологічні фактори з дуже малим шансом можуть бути виявлені під час роботи працівника бухгалтера, але викорінювати їх повністю також неможливо.

А ось психофізіологічні та нервові перевантаження мають важливу ступінь зосередження при робочих процесах (повний перелік небезпечних та шкідливих виробничих факторів дається у ДСТУ 12.0.003-74), бо саме робоча атмосфера у колективі дуже сильно впливає на працездатність усіх робітників, у тому числі і на помічника бухгалтера.

4.3. Заходи зменшення впливу шкідливих факторів

Негативних факторів впливу на працівників дуже багато, і головною метою підприємства є встановлення причин їх виникнення та шляхів вирішення чи послаблення.

По-перше, виявимо умови праці працівника, а саме робоче місце помічника бухгалтера. Умовами комфортної роботи є:

- Висота робочої поверхні, а саме комп'ютерного стола, має бути встановлено з фізичних особливостей помічника бухгалтера. Ноги працівника мають доставати до підлоги, він має вільно рухати ними та тримати рівно, голова не повинна бути нахилена більше ніж на 20 градусів, плечі повинні бути розслаблені, а лікті – під кутом 80-100 градусів, передпліччя та кисті рук – у горизонтальному положенні.
- Стіл передбачає збереження на собі та в собі: портативного комп'ютера, місце для комп'ютерної миші та клавіатури, принтера,

органайзерів для канцелярських приладь та документації. Усі предмети на столі повинні не приносити незручності під час роботи працівника.

- Висота робочої поверхні рекомендується в межах 680-760 мм;
- Монітор має бути встановлено на відстані 50-60 см від очей;
- Співвідношення яскравості фона екрану і символів на екрані – від 1:2 до 1:15;
- Робоче крісло, а саме його сидіння, має мати висоту в межах 420-550 мм. Поверхня має бути м'якою, спинка – врегульована.

Встановимо вимоги роботи у приміщенні:

- Верхній одяг має зберігатися в окремому місці (у шафі, у окремій кімнаті, яка пристосована як гардеробна);
- Працівник має зберігати своє робоче місце у чистоті, усі речі мають лежати на своїй місцях;
- Уся документація, яка є архівом та не має актуальності, має бути у окремому для цього місці (інший кабінет, шафа тощо);
- Прибирання приміщення має бути у встановлений адміністрацією час іншим для цього працівником;
- Наприкінці робочого дня працівник має йти по такому алгоритму дій: залишити своє робоче місце у чистоті, прибрати непотрібні речі зі стола, вимкнути усі електронні прилади (комп'ютер, принтер тощо) свого робочого місця або, якщо він залишає своє робоче місце останнім, і інших працівників задля попередження залишення приладів увімкненими до наступного робочого дня;
- Забороняється ремонтувати будь-які прилади самостійно, це може призвести до порушення правил техніки безпеки. Це мають робити спеціалізовані фахівці.

У робочому приміщенні мають діяти заходи щодо освітлення, рівня шуму, мікроклімату та електричної безпеки.

Освітлення має бути як і природне, так і штучне. Люмінесцентні лампи типу ЛБ повинні застосовуватися в якості джерела світла при штучному висвітленні. Перегоріли лампи мають бути замінені вчасно, світильники мають протиратися від пилу заздалегідь, щоб уникнути погіршення освітлення на робочому місці. Фарбування стелі та стін має бути у світлих відтінках для підвищення рівня освітленості та позбавлення від різких тіней, які також впливають на зір працівника.

Щодо електричної безпеки, усі електроні прилади повинні мати електричну ізоляцію, захисне заземлення, має проводитися контроль їх аварійного стану. Працівники повинні безпечно вимикати прилади, щоб уникнути їх замикань, які можуть призвести до небезпечних ситуацій.

Мікроклімат приміщення повинно регулюватися за допомогою провітрювання приміщення, правильною вентиляцією та кондиціонуванням. Як у холодну, так і в теплу пори року необхідно досягати нормованої температури повітря у приміщенні за допомогою водної, парової, повітряної чи комбінованої системи опалення.

Шум може виникати в залежності від різних факторів, які потрібно уникати. Офіс має знаходитися у приміщенні з звукоізолюючими стінами. Якщо при оренді приміщення, це неможливо досягти, можливе встановлення звукоізолюючих приладів на стінах кімнати.

Сприятливі умови праці та дотримання стандартних правил безпеки під час бізнес-процесів покращать рентабельність підприємства, збільшить показники виконання робочих питань та підвищить рівень продуктивності праці у всього колективу.

РОЗДІЛ 5. ПРО ОХОРОНУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1 Актуальність питання, що стосується охорони навколишнього середовища

За законом України, який був введений в дію за Постановою ВР №1268-ХІІ від 26.06.1991, завданням законодавства про охорону навколишнього середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною [25].

Екологічна ситуація з кожним роком стає всі більш гострішою для обговорення. Глобальне потепління, забруднення повітря, вимирання великої кількості тварин.

Під час воєнного часу ситуація зовсім вийшла на новий рівень. Наприклад, провокації, які пов'язані з атомною енергетикою. Якщо не брати до уваги численні пожежі на території Чорнобильської зони, через які в повітря було піднято безліч шкідливих речей, був ризик викиду величезної кількості радіоактивних елементів із самої станції.

Ще одним з факторів, що впливають на екологію, – це викиди самих ракет. Навіть у разі знищення, усі речовини проходять стадію окиснення, хімічне забруднення атмосфери хімічними сполуками цих компонентів (чадний газ, бурий газ, азот тощо).

Знищення ландшафту, рельєфу, екосистем. Під час бойових дій відбуваються лісові пожежі, які складно загасити спеціальними службами через небезпеку обстрілу; відбувається ерозія ґрунту через техніку, а особливо, коли снаряди падають і знищують верхні шари. Безліч тварин також страждають – стрес, знищення харчового ланцюжка, просто вбивства заради забави російськими загарбниками. Ще однією з катастроф, яка б могла статися, але не

сталася через роботу українців, щоб екологія не постраждала, це масове вимирання курок на птахофабриці «Чорнобаївська». Якби курей не роздали, а померлих не поховали, шкоди екосистемі було б завдано дуже багато і яку не можна було б виправити.

Екодія, організація, яка розпочала моніторинг впливу російської агресії на Україну, нарахувала 1529 випадків шкоди нашої екології [26]. І це лише ті, що базуються на офіційних джерелах. Повністю оцінити збитки буде можливо лише після закінчення війни.

Тому важливо, під час війни, приділяти увагу охороні навколишнього середовища. Навіть маленькі річі можуть допомогти стабілізувати хоч якимось ситуацію з природними ресурсами, які втрачаються та знищуються.

5.2 Аналіз факторів шкідливого впливу на екологію бізнес-процесу «Оновлення наявності товару для продажу»

Існує багато кількості екологічних факторів впливу на природне середовище. Вони розділені на три групи, а саме абіотичні, біотичні та антропогенні. Абіотичні виникають з неорганічних умов, а саме хімічних та фізичних (температура, вологість, світло, радіація тощо). Біотичні фактори виникають за форми взаємодії хазяїна та паразиту. У випадку антропогенного впливу – форми діяльності людини. Саме цю групу факторів можна віднести до категорій впливу на середовищу бізнес-процесів.

Також фактори впливу можна розділити на класифікації, які розрізняються за часом, походженням, характером, ступенем впливу та умовами дії.

Бізнес-процес «Оновлення наявності товару для продажу» складається з великої кількості складових, які ми побачили, проаналізували та зробили висновки під час роботи над минулими розділами. Кожне розгалуження цього процесу має взаємодію між собою, а також з навколишнім середовищем. Так, наприклад, взаємодія помічника бухгалтера з комп'ютером – витрати електроенергії, також використання великої кількості паперу для документації,

різна канцелярія, яка з часом також приходить у непридатність. Ці відходи можуть здатися дрібницею, хоча насправді з таких дрібниць все й складається. Пізніше ми подивимося шляхи вирішення того, як зменшити вплив цих факторів на бізнес-процес.

Розвиток усіх галузей промисловості (енергетика, транспорт, виробництва) призвели до того, що зміни у навколишньому середовищі стали непередбачувані та практично незворотними. Але є такі, які можливо укоренити чи зменшити їх вплив. Наприклад, забруднення біосфери. До такого забруднення відносять транспортні засоби та автомобілі.

У бізнес-процесі є складова, яка відповідає за логістику товару, який надходить до магазину. Водій кожного дня використовує транспортний засіб для того, щоб перевозити товар з однієї торгової точки до іншої, чи з складу до магазину. Бувають відрядження, які пов'язані з перевезення товару з міста Миколаєва до міста Одеси. Це все – великі викиди шкідливих речовин у повітря. Гази двигунів, що потрапляють у повітря, містять складні суміші, вони складається з більш ніж двохсот компонентів, серед яких чимало канцерогенів. Також може відбуватися викиди у ґрунти.

Забруднення повітря транспортними засобами негативно впливає на загальну захворюваність суспільства. Воно знижує імунітет, негативно впливає на органи дихання, спричиняє різні захворювання таких органів: серця, судин, спричиняє виникнення мутагенних, канцерогенних наслідків тощо. Європейське Бюро ВООЗ стверджує, що забруднення атмосферного повітря може бути провокуючою причиною до 7-10% всіх випадків респіраторних захворювань серед дітей, 3-7% нових випадків хронічних обструктивних захворювань, 3-15% нових випадків бронхіальної астми [27].

5.3 Заходи запобігання забруднення навколишнього середовища

Для запобігання забруднення навколишнього середовища виявленими факторами бізнес-процесу, а саме офісні відходи, можна застосувати такі прості правила:

– Сортивання сміття. Папір – окремо, канцелярію – окремо, побутові та харчові відходи – окремо. При цьому можливе складання контракту з сміттєвозними компаніями про співпрацю. Наприклад, проведення акцій у магазинах Табако Маркет, які будуть пов’язані з сортуванням сміття. Це підніме актуальність підприємства в очах молоді, що також є плюсом. Сортиванням сміття також вирішиться проблема продажу табачних виробів, які також мають шкідливий вплив на довкілля. Правильна утилізація не викоринити проблему, але хоча б її зменшить.

– Перевикористання паперу. Архівні файли, у яких вже вийшов строк придатності, можна використовувати як чернетки для окремих записів чи не офіційної документації. Це зменшить споживання паперових предметів, що гарне запобігає забрудненню навколишнього середовища та навіть, у якійсь мірі, зменшить витрати підприємства.

Що стосується забруднення повітря під час переміщення товарів. Шляхами вирішення є:

- Використання електроавтомобілів. Це зменшить кількість викидів в атмосферу, а також може зекономити гроші на пальному. Але не всі електроавтомобілі підходять для транспортування великих обсягів товарів, тому цей шлях не є практичним з цієї думки. Якщо і рухатися в цьому русі, то потрібно це робити поетапно, як з погляду об’єктів нормування, так і з погляду значень прийнятих нормативів.

- Використання якісного пального для автомобілів. Це вже більш реальний шлях.

- Перевірка справності автомобілів якомога частіше. Також на викиди шкідливих речовин впливає несправність транспортних засобів, тому потрібно застосувати перевірку хоча б раз на місяць усіх автомобілів, які належать підприємству.

Якщо дивитися на ситуацію більш широко, на державному рівні, то потрібно переглянути систему оцінок екологічних якостей транспортних засобів, а саме обмеження концентрації оксиду вуглецю і вуглеводів при роботі

двигунів, обмеженням димності газів, регламентацією викидів картерних газів тощо.

Для зменшення впливу автомобільних викидів на навколишнє середовище та забезпечення сталих рішень, необхідно впроваджувати нові технології та підходи до транспортного сектору. Такий підхід сприятиме створенню сталого екологічно чистого та ефективного транспортного середовища. Зменшення кількості відходів від транспортних засобів цивільного населення – хоч якесь полегшення для природи, коли у країні й так відбувається екологічна катастрофа (і не тільки вона).

А починання щодо сортування сміття будуть сприяти маленькому, але вкладенню в оновлення та збереження ресурсів природи, в якій ми живемо і яка є частиною нас.

РОЗДІЛ 6. РОЗРАХУНОК ВИТРАТ НА ВПРОВАДЖЕННЯ ІС

6.1 Фінансова актуальність розробленої ІС

Будь-яка робота має бути оплачена. Аналіз, проектування та розробка інформаційних систем не є винятком із правил. Задіяння системи має збільшити показники продуктивності роботи системи, а також зменшити кількість втрати даних, які були до розробки оновленої системи.

Наведемо простий приклад, що стався у робочому процесі. Помічнику бухгалтера надійшов запит, пов'язаний із перевіркою з податкової служби. Необхідно було надати дані накладної на товар у зазначених датах, у зазначеній кількості та від зазначеного постачальника. Список складається з n-кількості найменувань. При використанні минулої інформаційної системи, де не зберігалася документація систематично та структуровано, а файли – у сховищі месенджера Telegram, пошук дуже затягнувся. Співробітник витратив кілька днів, щоб зіставити усі дати, відсортувати потрібні накладні. Працівнику доводилося заходити в особисті діалоги до кожного продавця, шукати потрібну дату, а потім накладну та постачальника від неї. У цьому підприємстві потрібно було сплатити понаднормову суму, оскільки працівник займався цим в окремі години від роботи. Так і виходять збитки. Чи можлива ситуація, що співробітник просто міг не знайти її? Паперовий архів міг також не дати жодного результату. Накладна втрачена, інформація, що підтверджує товар, теж. Податкова служба виявляє порушення, може звернути увагу і на інші складові системи, виявити ще якісь порушення, що призведе до збитків у підприємства.

Створена інформаційна система викорінює цю проблему, зберігаючи всі необхідні файли та дані в одному місці. Помічник бухгалтера тепер може витратити мінімальну кількість часу на пошук необхідної накладної за допомогою пошуку за постачальниками та датами занесення накладної до бази даних. Отже, вирішується питання збитків, оскільки втрата даних мінімальна. Система актуальна, може принести прибуток з прискоренням бізнес-процесу товарообігу у компанії та уникнути подальших проблем з ними пов'язаних. Але потрібно розуміти, що створення нової системи не викоринить усі проблеми, але

вона може полегшити їх вирішення та зменшити їх кількість, що зменшить збитки та навіть збільшить дохід компанії.

Після підтвердження корисності розробленої системи, можна розпочати розрахунків витрат за її розробку загалом.

6.2 Розрахунок витрат на створення і впровадження системи

Важливим моментом для застосування аналізу, проектування та розробки системи є процес формування її ціни.

Створення автоматизованої системи вимагає одноразових витрат на її розробку, придбання необхідних технічних засобів і поточних витрат на функціонування системи. Витрати складаються з:

- людських ресурсів;
- витрат на прилади розробки;
- витрати на експлуатацію приладів розробки;
- витрати на програмне забезпечення, в якому відбувається розробка системи;
- витрати на впровадження системи.

Впровадження системи приведе до значної економії, наприклад: підвищення продуктивності праці; вивільнення робочого часу; вивільнення працівників для виконання додатковою роботи під час екстрених ситуацій; уникнення штрафів чи їх позбавлення зовсім.

Для розробки нам необхідні:

- Frontend-розробник, який займається клієнтом;
- Backend-розробник, який займається сервером.

Мінімальна заробітна плата на місяць Frontend-розробника 18 000грн, Backend-розробника – 35 00 грн. Час, який потрібно на розробку системи – 3 місяці. З цього випливає, що затрати на заробітну плату двох розробників за три місяці становить за формулою (1).

$$Z_{\text{зп}} = (Z_{\text{зпF}} + Z_{\text{зпB}}) * T_{\text{роз}} \quad (1),$$

де $Z_{\text{пF}}$ – заробітня плата Frontend-розробника;

$Z_{\text{пB}}$ – заробітня плата Backend –розробника;

$T_{\text{роз}}$ – тривалість розробки проекту.

З цього випливає:

$$Z_{\text{п}} = (35000 + 18000) * 3 = 159000 \text{ грн.}$$

Загальна сума витрат на заробітню плату складає 159000 грн.

Прилади розробки вже є (у двох розробників стаціонарні комп'ютери, миші, клавіатури є), у поточних розрахунках витрат вони не приймають участь. Потрібно лише актуалізувати витрати на їх експлуатацію.

Витрати на експлуатацію приладів розробки полягаються в оплаті споживчої електроенергії та вираховуються за формулою (2).

$$Z_{\text{експл}} = P_{\text{еом}} * T_{\text{роз}} * N_{\text{рд}} * N_{\text{рг}} * C_e \quad (2),$$

де $P_{\text{еом}}$ – потужність приладу для розробки, 0,18 квт./год ;

$N_{\text{рд}}$ – число робочих днів у місяці, 20 днів;

$N_{\text{рг}}$ – число годин у робочому дні, 9 годин;

C_e – вартість 1 квт. електроенергії, 1,68 грн.

$$Z_{\text{експл}} = 0,18 * 3 * 20 * 9 * 1,68 = 362,88 \text{ грн.}$$

Перемножуємо на 2, тому що два працівника.

$$Z_{\text{експл}} = 362,88 * 2 = 725,76 \text{ грн.}$$

Для розробки необхідні програмні засоби, які представлені в таблиці 6.1

Таблиця 6.1. Витрати на програмні засоби на місяць.

№	Найменування	Вартість, грн./міс
	Ліцензія WebStorm, 2шт.	1160
	Програма DrawIO для діаграм	0
	Фреймворки (Vue, NestJS)	0
	Бібліотеки (SwaggerJS, Sequelize, Vuetify)	0
	Сервер	2400
Разом		3560

Сервер – це постійні витрати, але порахуємо її вартість під час розробки, ліцензія середовища розробки – оплата лише на три місяця. Актуалізуємо в таблиці 6.2

Таблиця 6.2. Витрати на програмні засоби на три місяці розробки.

№	Найменування	Вартість, грн
	Ліцензія Webstorm, 2шт.	3480
	Програма DrawIO для діаграм	0
	Фреймворки (Vue, NestJS)	0
	Бібліотеки (SwaggerJS, Sequelize, Vuetify)	0
	Сервер	7200
Разом		10680

Загальний кошторис витрат представлено в таблиці 6.3

Таблиця 6.3. Загальний кошторис витрат на три місяці розробки.

№	Найменування	Вартість, грн
	Заробітня плата за 3 місяці	159000
	Ліцензія Webstorm, 2шт.	3480
	Сервер	7200
	Витрати на експлуатацію приладів розробки	725,76
	Супроводження системи	2500
Разом		172905,76

Загальний кошторис – 172905,76 грн.

Для того, щоб порахувати ефективність системи та те, як вона може себе окупити, візьмемо за основу той приклад з штрафом від податкової служби. Один з штрафів мав суму 18000грн, за не виявлення необхідної накладної на товар. І це тільки одне порушення, а їх може бути безліч. Втрата даних може до цього призвести, а система це виправляє. Середня кількість штрафів тільки за не виявлення накладних за рік може доходити до 25. Зменшуємо її кількість до 20, тому що викоренення усіх штрафів неможливе через людський фактор

Таким чином, можна порахувати строки окупності розробленої системи. Вона розраховується за формулою 3.

$$k = \frac{B}{\Delta\Pi * N} \quad (3),$$

Де B – це загальна сума витрат на створення і впровадження системи,

$\Delta\Pi$ – це сума штрафу,

N – це середня кількість штрафів за рік.

$$k = \frac{172905,76}{18000 * 20} = 0.48$$

З цього ми можемо зробити висновок, що система може окупити себе через 0,48 від року або через 5,7 місяців, що без урахування виплат за сервер кожного місяця, але навіть так є прекрасним результатом для впровадження системи у роботу та її розвиток у подальшому, щоб закрити і інші проблеми, які можуть виникнути під час бізнес-процесів підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ.

ВИСНОВОК

У дипломній роботі було розроблено та реалізовано ІС підприємства ЕЛІЗТАН АЛЬЯСН КОМПАНІ.

Було сформовано та проаналізовано предметну область, встановлено структуру організації, проаналізовані складові цієї структури, за допомогою SWOT-аналізу було проведено дослідження моделі «ЯК Є», сформовано модель «ЯК ПОВИННО БУТИ» ІС підприємства та встановлені вимоги до майбутньої ІС за допомогою цієї моделі. На основі готової концепції був розроблений технічний проект ІС, а саме розробка діаграми прецедентів, концептуальної, логічної та фізичної моделі даних та діаграми послідовності. Після цього були сформовані математичне (вирішення аналізу переміщення товару за допомогою алгоритма Дейкстри), програмне та технічне забезпечення. Реалізацію було виконано на архітектурній структурі сервер-клієнт. Сервер було реалізовано за допомогою мови програмування TypeScript, фреймворку NestJs та Sequelize із використанням засобів СУБД PostgreSQL. Клієнт було реалізовано за допомогою мови програмування JavaScript, фреймворку VueJS та її бібліотек. Виконані розділи щодо охорони праці, фінансовий розділ та розділ з охорони довкілля.

Розроблена ІС передбачає подальший розвиток у реалізації, а саме розширення функціоналу системи ведення бухгалтерського обліку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В.С. Пономаренко, С.В. Мінухін, С.В. Знахур Теорія та практика моделювання бізнес-процесів – 244с.
2. Інформаційна система [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Інформаційна_система – дата звернення: 01.12.2022
3. Motive Force Kvinta [Електронний ресурс]: Веб-додаток Kvinta. – Режим звернення: http://kvinta.od.ua/motive_force/price/ – дата звернення: 01.12.2022
4. Gmail [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: <https://uk.wikipedia.org/wiki/Gmail> – дата звернення: 01.12.2022
5. Microsoft Office [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Microsoft_Office – дата звернення: 01.12.2022
6. Telegram [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: <https://uk.wikipedia.org/wiki/Telegram> – дата звернення: 01.12.2022
7. SWOT [Електронний ресурс]: Espotnik. – Режим звернення: <https://esputnik.com/uk/blog/swot-analiz-iz-prikladami>– дата звернення: 20.11.2023
8. UML [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Unified_Modeling – дата звернення: 28.04.2023
9. ERD діаграма [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Модель_«сутність_—_зв'язок» – дата звернення: 30.04.2023
10. Перелічуваний_тип_даних [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення:

- https://uk.wikipedia.org/wiki/Перелічуваний_тип_даних— дата звернення: 02.12.2023
- 11.Діаграма послідовності [Електронний ресурс]: – Режим звернення: https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciyних-tehnologiy – дата звернення: 28.04.2023
- 12.Теорія графів. Навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем» спеціальності 122 «Комп'ютерні науки»/ І.М. Кузьменко; КПІ ім. Ігоря Сікорського. — Електронні текстові дані (1 файл: 1,7 Мбайт). — Київ: КПІ ім. Ігоря Сікорського, 2020. — 71 с.
- 13.Офіційна документація PostgreSQL. [Електронний ресурс]: Документація PostgreSQL. – Режим звернення: <https://www.postgresql.org/docs/current/intro-what-is.html> – дата звернення: 20.11.2023
- 14.Офіційна документація NodeJs. [Електронний ресурс]: Документація NodeJs. – Режим звернення: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> – дата звернення: 20.11.2023
- 15.Express.js [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: <https://uk.wikipedia.org/wiki/Express.js> – дата звернення: 20.11.2023
- 16.Офіційний репозиторій SwaggerUIExpress. [Електронний ресурс]: Репозиторій SwaggerUIExpress. – Режим звернення: <https://www.npmjs.com/package/swagger-ui-express> – дата звернення: 20.11.2023
- 17.JavaScript [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: <https://uk.wikipedia.org/wiki/JavaScript> – дата звернення: 28.04.2023

- 18.Офіційна документація Vuetify. [Електронний ресурс]: Документація Vuetify. – Режим звернення <https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started> – дата звернення: 04.12.2023
- 19.Офіційна документація VueRouter. [Електронний ресурс]: Документація VueRouter. – Режим звернення: <https://router.vuejs.org/introduction.html> – дата звернення: 20.11.2023
- 20.WebStorm [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: <https://uk.wikipedia.org/wiki/WebStorm> – дата звернення: 28.04.2023
- 21.Linux Mint [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Linux_Mint – дата звернення: 28.04.2023
- 22.Chrome [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. – Режим звернення: https://uk.wikipedia.org/wiki/Google_Chrome – дата звернення: 28.04.2023
23. Закон України Про охорону праці [Електронний ресурс]: – Режим звернення: <https://zakon.rada.gov.ua/laws/show/2694-12#Text> – дата звернення: 29.11.2023
24. Закон України Про охорону праці розділ IV [Електронний ресурс]: – Режим звернення: <https://zakon.rada.gov.ua/laws/show/2694-12#Text> – дата звернення: 29.11.2023
25. Закон України Про охорону навколишнього природного середовища [Електронний ресурс]: – Режим звернення: <https://zakon.rada.gov.ua/laws/show/1264-12#Text> – дата звернення: 29.11.2023
26. Випадки потенційної шкоди довкіллю, спричинені російською агресією [Електронний ресурс]: Центр екологічної ініціативи «Екодія». – Режим звернення: <https://ecoaction.org.ua/warmap.html> – дата звернення: 29.11.2023

27. Шкідлива дія забрудненого атмосферного повітря на людей [Електронний ресурс]: Державна установа "Харківський обласний центр контролю та профілактики хвороб Міністерства охорони здоров'я України" – Режим звернення: <https://labcenter.kh.ua/?p=3815> – дата звернення: 05.12.2023
28. Конспект-матеріали з лекцій пройденого курсу «Охорона праці» на 4 курсі Національного університету кораблебудування імені адмірала Макарова Навчально-наукового інституту комп'ютерних наук та управління проектами Кафедри інформаційних управляючих систем та технологій на спеціальності Інформаційні системи та технології.
29. Національного Миколаївського кораблебудівного університету імені Адмірала Макарова, .
30. К.В. Кошкін, Г.Ю. Яні Проектування інформаційних систем промислових підприємств – 87с.
31. О.В. Гайдаєнко, Ю.О. Казимиренко, Г.С. Морозова Методичні вказівки ДСП – 31с.
32. Стан, досягнення та перспективи інформаційних систем і технологій / Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. Одеса, 20-21 квітня 2023 р. - Одеса, Видавництво ОНТУ, 2023 р. – 449 с.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Призначення розробки

Назва проекту: «ІС підприємства ЕЛІЗТАН АЛЬЯНС КОМПАНІ». Розробка ІС передбачає аналіз існуючої ІС, виявлення проблем та шляхів їх вирішення. Призначенням є: розробка та реалізація ІС для виконання виявлених під час аналізу проблем. Метою створення ІС є виправлення існуючих проблем у роботі бізнес-процесу «Оновлення наявності товару для продажу».

Вимоги до системи

ІС має бути мобільною та багато структурною, щоб реалізовувати різний функціонал. Система має:

- зберігати актуальну інформації, оновлювати її при запиті;
- складатися з трьох рівнів архітектурного рішення, а саме: клієнт (або декілька клієнтів, в залежності від потреб) – сервер – сервер БД;
- передбачає збереження файлів різних форматів на сервері;
- необхідно мати читабельний, зручний та зрозумілий інтерфейс.

Вимоги до програмного забезпечення

СУБД: PostgreSQL;

Вимоги до програмного забезпечення сервера:

- Операційна система: Linux;
- Середовище розробки: WebStorm;
- Мова програмування: TypeScript;
- Фреймворк: NestJS з використання технологій Sequelize для роботи з СУБД.
- API documentation: SwaggerUIExpress;

Вимоги до програмного забезпечення клієнту:

- Операційна система: Linux;
- Середовище розробки: WebStorm;
- Мова програмування: JavaScript;

- Фреймворк: Vue.

Вимоги до документації

Потрібно, щоб були наявні такі документи:

- Загальний опис системи;
- Посібник системного адміністратора програмного забезпечення;
- Програми та інструкції виконання;
- Інструкції введення бази даних та її використання;
- Текст програм;
- Посібник користувача.

Порядок контролю і прийому системи

Весь процес створення затверджується у вигляді акта-договору, які підписує директор компанії та розробник (або розробники з організацій). Отже, на законодавчому рівні буде затверджено співробітництво.

На кожному етапі проектування відбувається перевірка та огляд існуючих рішень. Передбачається переговори кожного тижня у п'ятницю для контролю процесів розробки.

Співпраця може бути розірвана у разі не виконання цілей та послуг з обох сторін за допомогою спеціальних юридичних процесів.

Вимоги до захисту інформації

Система має забезпечувати повний контроль за інформацією та її збереженням. Доступ до даних ІС має бути встановлено за ролями користування. Кожна роль користувача затверджується та має певні доступи до системи, а саме; Бухгалтер, Оператор, Комірник, Продавець, Супер-Адмін.

ДОДАТОК Б – КОД СТВОРЕННЯ ТАБЛИЦЬ POSTGRESQL

Створення таблиці Category

```
CREATE TABLE IF NOT EXISTS public."Category"
(
    id integer NOT NULL DEFAULT
nextval('"Category_id_seq" '::regclass),
    name text COLLATE pg_catalog."default",
    CONSTRAINT "Category_pkey" PRIMARY KEY (id)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."Category"
    OWNER to postgres;
```

Створення таблиці File

```
CREATE TABLE IF NOT EXISTS public."File"
(
    id integer NOT NULL DEFAULT
nextval('"File_id_seq" '::regclass),
    name text COLLATE pg_catalog."default" NOT NULL,
    mimetype text COLLATE pg_catalog."default",
    "storeInvoicesId" integer,
    CONSTRAINT "File_pkey" PRIMARY KEY (id),
    CONSTRAINT "File_storeInvoicesId_fkey" FOREIGN KEY
("storeInvoicesId")
    REFERENCES public."StoreInvoices" (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE SET NULL
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."File"
    OWNER to postgres;
```

Створення таблиці NameProduct

```
CREATE TABLE IF NOT EXISTS public."NameProduct"
(
    id integer NOT NULL DEFAULT
nextval('"NameProduct_id_seq" '::regclass),
```

```

        name text COLLATE pg_catalog."default",
        "subcategoryId" integer,
        barcode numeric,
        CONSTRAINT "NameProduct_pkey" PRIMARY KEY (id),
        CONSTRAINT "NameProduct_subcategoryId_fkey" FOREIGN KEY
("subcategoryId")
        REFERENCES public."SubCategory" (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE SET NULL
    )
    TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."NameProduct"
    OWNER to postgres;

```

Створення таблиці Price

```

CREATE TABLE IF NOT EXISTS public."Price"
(
    id integer NOT NULL DEFAULT
nextval('"Price_id_seq"'::regclass),
    "newPrice" money,
    "oldPrice" money,
    "createdAt" time(3) without time zone DEFAULT
CURRENT_TIMESTAMP,
    "updatedAt" time(3) without time zone,
    CONSTRAINT "Price_pkey" PRIMARY KEY (id)
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."Price"
    OWNER to postgres;

```

Створення таблиці Product

```

CREATE TABLE IF NOT EXISTS public."Product"
(
    id integer NOT NULL DEFAULT
nextval('"Product_id_seq"'::regclass),
    "nameId" integer,
    "priceId" integer,

```



```

        "VAT" numeric,
        status boolean NOT NULL DEFAULT true,
        "providerId" integer,
        "createdAt" timestamp(3) without time zone DEFAULT
CURRENT_TIMESTAMP,
        "updatedAt" timestamp(3) without time zone,
        CONSTRAINT "Product_pkey" PRIMARY KEY (id),
        CONSTRAINT "Product_nameId_fkey" FOREIGN KEY ("nameId")
            REFERENCES public."NameProduct" (id) MATCH SIMPLE
            ON UPDATE CASCADE
            ON DELETE SET NULL,
        CONSTRAINT "Product_priceId_fkey" FOREIGN KEY ("priceId")
            REFERENCES public."Price" (id) MATCH SIMPLE
            ON UPDATE CASCADE
            ON DELETE SET NULL,
        CONSTRAINT "Product_providerId_fkey" FOREIGN KEY
("providerId")
            REFERENCES public."Provider" (id) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
    )
    TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."Product"
    OWNER to postgres;
CREATE INDEX IF NOT EXISTS "fki_Product_providerId_fkey"
    ON public."Product" USING btree
    ("providerId" ASC NULLS LAST)
    TABLESPACE pg_default;

```

Створення таблиці Provider

```

ALTER

```

Створення таблиці Shop

```

CREATE TABLE IF NOT EXISTS public."Shop"
(
    id integer NOT NULL DEFAULT
nextval('"Shop_id_seq"'::regclass),
    name text COLLATE pg_catalog."default",

```

```

        "createdAt" timestamp(3) without time zone DEFAULT
CURRENT_TIMESTAMP,
        "updatedAt" timestamp(3) without time zone,
        CONSTRAINT "Shop_pkey" PRIMARY KEY (id)
    )
    TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."Shop"
    OWNER to postgres;

```

Створення таблиці StoreInvoices

```

CREATE TABLE IF NOT EXISTS public."StoreInvoices"
(
    id integer NOT NULL DEFAULT
nextval('"StoreInvoices_id_seq"'::regclass),
    type "TypeInvoices" NOT NULL,
    "createdAt" timestamp(3) without time zone NOT NULL
DEFAULT CURRENT_TIMESTAMP,
    status boolean NOT NULL DEFAULT true,
    "updatedAt" timestamp(3) without time zone DEFAULT
CURRENT_TIMESTAMP,
    "providerId" integer,
    CONSTRAINT "StoreInvoices_pkey" PRIMARY KEY (id),
    CONSTRAINT "StoreInvoices_provierId_fkey" FOREIGN KEY
("providerId")
        REFERENCES public."Provider" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
    TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."StoreInvoices"
    OWNER to postgres;

```

Створення таблиці SubCategory

```

CREATE TABLE IF NOT EXISTS public."SubCategory"
(
    id integer NOT NULL DEFAULT
nextval('"SubCategory_id_seq"'::regclass),

```

```

        name text COLLATE pg_catalog."default",
        CONSTRAINT "SubCategory_pkey" PRIMARY KEY (id)
    )
    TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."SubCategory"
    OWNER to postgres;

```

Створення таблиці User

```

CREATE TABLE IF NOT EXISTS public."User"
(
    id integer NOT NULL DEFAULT
nextval('"User_id_seq"'::regclass),
    name text COLLATE pg_catalog."default" DEFAULT 'Новий
користувач'::text,
    "adminRole" "AdminRole" DEFAULT
'ACCOUNTANT'::"AdminRole",
    blocked boolean NOT NULL DEFAULT false,
    email text COLLATE pg_catalog."default" NOT NULL,
    "createdAt" timestamp(3) without time zone NOT NULL
DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" timestamp(3) without time zone NOT NULL
DEFAULT CURRENT_TIMESTAMP,
    "fileId" integer,
    password text COLLATE pg_catalog."default" NOT NULL,
    hidden boolean DEFAULT false,
    slug text COLLATE pg_catalog."default",
    "shopId" integer,
    CONSTRAINT "User_pkey" PRIMARY KEY (id),
    CONSTRAINT "User_fileId_fkey" FOREIGN KEY ("fileId")
        REFERENCES public."File" (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE SET NULL,
    CONSTRAINT "User_shopId_fkey" FOREIGN KEY ("shopId")
        REFERENCES public."Shop" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

```

```

)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."User"
CREATE UNIQUE INDEX IF NOT EXISTS "User_email_key"
    ON public."User" USING btree
    (email COLLATE pg_catalog."default" ASC NULLS LAST)
    INCLUDE(email)
    WITH (deduplicate_items=False)
    TABLESPACE pg_default;
CREATE UNIQUE INDEX IF NOT EXISTS "User_slug_key"
    ON public."User" USING btree
    (slug COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=False)
    TABLESPACE pg_default;

```

Створення таблиці UserAuthSession

```

CREATE TABLE IF NOT EXISTS public."UserAuthSession"
(
    id integer NOT NULL DEFAULT
nextval('"UserAuthSession_id_seq"'::regclass),
    "userId" integer NOT NULL,
    ip text COLLATE pg_catalog."default",
    "userAgent" text COLLATE pg_catalog."default",
    active boolean NOT NULL DEFAULT true,
    "createdAt" timestamp(3) without time zone NOT NULL
DEFAULT CURRENT_TIMESTAMP,
    "updatedAt" timestamp(3) without time zone NOT NULL,
    CONSTRAINT "UserAuthSession_pkey" PRIMARY KEY (id),
    CONSTRAINT "UserAuthSession_userId_fkey" FOREIGN KEY
("userId")
        REFERENCES public."User" (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."UserAuthSession"

```

OWNER to postgres;

Створення таблиці _ShopToProduct

```
CREATE TABLE IF NOT EXISTS public."_ShopToProduct"
(
    "idProduct" integer NOT NULL,
    "idShop" integer NOT NULL,
    counter integer NOT NULL DEFAULT 0,
    CONSTRAINT "File_productId_fkey" FOREIGN KEY
("idProduct")
    REFERENCES public."Product" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION,
    CONSTRAINT "File_shopId_fkey" FOREIGN KEY ("idShop")
    REFERENCES public."Shop" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."_ShopToProduct"
    OWNER to postgres;
```

Створення таблиці _StoreInvoicesToShop

```
CREATE TABLE IF NOT EXISTS public."_StoreInvoicesToShop"
(
    "idShop" integer NOT NULL,
    "idStoreInvoices" integer NOT NULL,
    CONSTRAINT "_StoreInvoicesToShop_A_fkey" FOREIGN KEY
("idShop")
    REFERENCES public."Shop" (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    CONSTRAINT "_StoreInvoicesToShop_B_fkey" FOREIGN KEY
("idStoreInvoices")
    REFERENCES public."StoreInvoices" (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE
```

```

)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."_StoreInvoicesToShop"
    OWNER to postgres;

```

Створення таблиці _subCategoryToCategory

```

CREATE TABLE IF NOT EXISTS public."_subCategoryToCategory"
(
    "idSubCategory" integer NOT NULL,
    "idCategory" integer NOT NULL,
    CONSTRAINT "_subCategoryToCategory_pkey" PRIMARY KEY
("idSubCategory", "idCategory"),
    CONSTRAINT "_subCategoryToCategory_A_fkey" FOREIGN KEY
("idSubCategory")
        REFERENCES public."SubCategory" (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE SET NULL,
    CONSTRAINT "_subCategoryToCategory_B_fkey" FOREIGN KEY
("idCategory")
        REFERENCES public."Category" (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE SET NULL
)
TABLESPACE pg_default;
ALTER TABLE IF EXISTS public."_subCategoryToCategory"
    OWNER to postgres;

```

Створення ENUM AdminRole

```

CREATE TYPE public."AdminRole" AS ENUM
    ('ADMIN', 'OPERATOR', 'ACCOUNTANT', 'STOCK', 'SALESMAN');
ALTER TYPE public."AdminRole"
    OWNER TO postgres;

```

Створення ENUM AdminRole

```

CREATE TYPE public."TypeInvoices" AS ENUM
    ('Store', 'Stock');
ALTER TYPE public."TypeInvoices"
    OWNER TO postgres;

```

ДОДАТОК В – КОД ПРОГРАМИ BACKEND

```
//сербic AuthModule

import {Body, HttpException, HttpStatus, Injectable, Post,
UnauthorizedException} from '@nestjs/common';
import {CreateUserDto, AuthLoginDto} from "../user/dto/create-
user.dto";
import {UserService} from "../user/user.service";
import {JwtService} from "@nestjs/jwt";
import * as bcrypt from 'bcryptjs'
@Injectable()
export class AuthService {
    constructor(private userService: UserService,
                private jwtService: JwtService) {

    }

    async login(userDto: AuthLoginDto) {
        const user = await this.validateUser(userDto);
        return this.generateToken(user)
    }

    async registration(userDto: CreateUserDto) {
        const candidate = await
this.userService.getUserByEmail(userDto.email);
        if(candidate) {
            throw new HttpException('Користувача з такою поштою
існує', HttpStatus.BAD_REQUEST)
        }
        const hashPassword = await bcrypt.hash(userDto.password,
5)
        const user = await this.userService.createUser({
            ...userDto, password: hashPassword
        })
        return this.generateToken(user)
    }

    private async generateToken(user) {
        const payload = {
            email: user.email,
            id: user.id,
            role: user.adminRole
        }
        return {
            token: this.jwtService.sign(payload)
        }
    }

    private async validateUser(userDto: AuthLoginDto) {
        const user = await
this.userService.getUserByEmail(userDto.email);
```

```

        const passwordEquals = await
bcrypt.compare(userDto.password, user.password)
        if(user && passwordEquals) {
            return user;
        }
        throw new UnauthorizedException({message: 'Некоректні
почта або пароль'})
    }
}

async getMe(request: Request) {
    const authHeader = request.headers.authorization;
    const bearer = authHeader.split(' ')[0];
    const token = authHeader.split(' ')[1];
    if(bearer !== 'Bearer' || !token) {
        throw new HttpException('Користувач не авторизований',
HttpStatus.BAD_REQUEST)
    }
    const user = this.jwtService.verify(token)
    return user
}

logout(request: Request) {
    if (request.headers.authorization) {
        delete request.headers.authorization;
    }
    throw new HttpException('Користувач не авторизований',
HttpStatus.BAD_REQUEST)
}

```

//сербіс FileModule

```

import {HttpException, HttpStatus, Injectable} from
'@nestjs/common';
import {InjectModel} from "@nestjs/sequelize";
import {File} from "../file.model";

import * as path from 'path'
import * as fs from 'fs';
import * as uuid from 'uuid';

@Injectable()
export class FileService {

    constructor(
        @InjectModel(File)
        private fileModel: typeof File,
    ) {}

    fileUpload(file) {
        try {

```



```

        const mimeType = file.mimetype.split('/').pop();
        const fileName = uuid.v4() + '.' + mimeType;
        const filePath = path.resolve(__dirname, '..',
'static')
        if (!fs.existsSync(filePath)) {
            fs.mkdirSync(filePath, {recursive: true})
        }
        fs.writeFileSync(path.join(filePath, fileName),
file.buffer)
        const fileDTO = {
            name: fileName,
            mimeType: mimeType
        }
        return this.fileModel.create(fileDTO);
    } catch (e) {
        throw new HttpException('Виникла помилка під час
запису файлу', HttpStatus.INTERNAL_SERVER_ERROR)
    }
}

fileUploadWithId(file, id) {
    try {
        const mimeType = file.mimetype.split('/').pop();
        let fileDTO = {};
        let fileName;
        if(mimeType.includes('vnd')) {
            const mimeTypeDOC =
file.originalname.split('.').pop();
            fileName = uuid.v4() + '.' + mimeTypeDOC;
            fileDTO = {
                name: fileName,
                mimeType: mimeTypeDOC,
                storeInvoicesId: id
            }
        } else {
            fileName = uuid.v4() + '.' + mimeType;
            fileDTO = {
                name: fileName,
                mimeType: mimeType,
                storeInvoicesId: id
            }
        }
        const filePath = path.resolve(__dirname, '..',
'static')
        if (!fs.existsSync(filePath)) {
            fs.mkdirSync(filePath, {recursive: true})
        }
        fs.writeFileSync(path.join(filePath, fileName),
file.buffer)
        return this.fileModel.create(fileDTO);
    } catch (e) {
        throw new HttpException('Виникла помилка під час

```

```

запису файлу', HttpStatus.INTERNAL_SERVER_ERROR)
    }
  }

  async fileRemove(id) {
    let fileRemovePath = (await this.fileModel.findOne({where:
{id}})).dataValues.name;
    let fileRemoveDB = await this.fileModel.destroy({where:
{id}});
    if(fileRemoveDB === 1) {
      const filePath = path.resolve(__dirname, '..',
'static/' + fileRemovePath)
      fs.unlinkSync(filePath)
      return {
        success: true
      }
    } else {
      return {
        success: false
      }
    }
  }
}

```

//сербic ProductModule

```

import {HttpException, HttpStatus, Injectable} from
'@nestjs/common';
import {InjectModel} from "@nestjs/sequelize";
import {NameProduct} from "../name-product/name-product.model";
import {Product} from "../product.model";
import {CreateProductDto} from "../dto/create-product.dto";
import {SubCategory} from "../sub-category/sub-category.model";
import {Price} from "../price/price.model";
import {Provider} from "../provider/provider.model";
import {Category} from "../category/category.model";
import {Shop} from "../shop/shop.model";
import {Op} from "sequelize";
import {Sequelize} from "sequelize-typescript";
import {ProductToShop} from "../shop/product-to-shop.model";

@Injectable()
export class ProductService {
  constructor(
    @InjectModel(Product)
    private productModel: typeof Product,

    @InjectModel(ProductToShop)
    private productToShopModel: typeof ProductToShop,

    private readonly sequelize: Sequelize
  ) {}
}

```

```

    ) {}

    async createProduct(dto: CreateProductDto) {
        return await this.productModel.create(dto);
    }

    async destroyProduct(id) {
        return await this.productModel.destroy({where: {id: id}})
    }

    async updateProduct(productId: number, productDTO:
CreateProductDto) {
        const productBeforeUpdate = await
this.findOne(productId)
        if(productBeforeUpdate) {
            await this.productModel.update(productDTO, {
                where: {
                    id: productId
                }
            });
            return this.findOne(productId);
        }
        throw new HttpException('Продукту з таким айді не існує',
HttpStatus.BAD_REQUEST)
    }

    async findAll(limit: number, offset: number, search: string) {
        if(search) {
            return this.productModel.findAndCountAll({
                offset: offset,
                limit: limit,
                include: [
                    {
                        model: NameProduct,
                        where: {
                            [Op.or]: [
                                { name: { [Op.iLike]:
`%${search}%` } }
                            ],
                        },
                        include: [
                            {
                                model: SubCategory,
                                include: [Category], // Включаем
данние из еще одной дочерней таблицы
                            },],
                        },
                    {
                        model: Price
                    },
                    {
                        model: Provider
                    },
                ],
            })
        }
    }
}

```

```

        {
            model: Shop
        }
    ]
    })
} else {
    return this.productModel.findAndCountAll({
        offset: offset,
        limit: limit,
        include: [
            {
                model: NameProduct,
                include: [
                    {
                        model: SubCategory,
                        include: [Category], // Включаем
данные из еще одной дочерней таблицы
                    }, ],
            }, ],
            {
                model: Price
            },
            {
                model: Provider
            },
            {
                model: Shop
            }
        ]
    })
}

}

async findByOne(id: number) {
    return this.productModel.findOne({where: {id}, include:
{all: true}})
}

async updateAmount(idShopA: number, idShopB: number,
idProduct: number, amount: number) {
    await this.sequelize.transaction(async (t) => {
        const productA = await
this.productToShopModel.findOne({where: {
            idProduct: idProduct,
            idShop: idShopA
        },
        transaction: t
    },
    )

        const productB = await
this.productToShopModel.findOne({where: {
            idProduct: idProduct,

```

```

        idShop: idShopB
      },
      transaction: t
    },
  )

  if(productA && productB) {
    if(productA.counter - amount > 0) {
      productA.counter = productA.counter - amount;
      productB.counter = Number(productB.counter) +
Number(amount);

      await productA.save({transaction: t});
      await productB.save({transaction: t});
    }
  }
})
}
}

```

```

//контролер UserModule
import {
  Controller,
  Delete,
  Get,
  Param,
  Patch,
  Post,
  UseGuards,
  Query,
  UseInterceptors,
  UploadedFile
} from '@nestjs/common';
import {UserService} from '../user.service';
import {
  ApiBearerAuth,
  ApiBody,
  ApiConsumes,
  ApiOperation,
  ApiParam,
  ApiQuery,
  ApiResponse,
  ApiTags
} from '@nestjs/swagger';
import {User} from '../user.model';
import {JwtAuthGuard} from '../auth/jwt.auth.guard';
import {adminRole} from '../enums/adminRole.enums';
import {FileInterceptor} from '@nestjs/platform-express';
@ApiBearerAuth('access-token')
@ApiTags('Користувач')
@Controller('user')
export class UserController {
  constructor(private userService: UserService) {

```

```

    }
    // @ApiOperation({summary: 'Створення користувача'})
    // @ApiResponse({status: 200, type: User})
    // @Post()
    // create(@Body() userDto: CreateUserDto) {
    //     return this.userService.createUser(userDto)
    // }

    @ApiOperation({summary: 'Отримання всіх користувачів'})
    @ApiResponse({status: 200, type: [User]})
    @ApiQuery({ name: 'limit', required: false})
    @ApiQuery({ name: 'offset', required: false})
    @ApiQuery({ name: 'search', required: false})
    // @UseGuards(JwtAuthGuard)
    @Get()
    getAll(@Query('limit') limit: number, @Query('offset') offset:
number, @Query('search') search: string) {
        return this.userService.findAll(limit, offset, search);
    }

    @ApiOperation({summary: 'Отримання користувача за айді'})
    // @UseGuards(JwtAuthGuard)
    @Get('/:id')
    getUserById(@Param('id') id: number) {
        console.log(id)
        return this.userService.getUserById(id);
    }

    @ApiOperation({summary: 'Отримання користувача за slug'})
    @UseGuards(JwtAuthGuard)
    @ApiResponse({status: 200, type: [User]})
    @Get('/:slug')
    getUserBySlug(@Param('slug') slug: string) {
        return this.userService.getUserBySlug(slug);
    }

    @ApiOperation({summary: 'Видалення за айді'})
    @ApiParam({ name: "userId" })
    @Delete("/:userId")
    async remove(@Param("userId") userId: number) {
        return this.userService.destroyUser(userId);
    }

    @ApiOperation({summary: 'Оновлення ролі користувача'})
    @ApiParam({ name: "idUser" })
    @ApiQuery({ name: 'role', enum: adminRole })
    @Patch("role/:idUser/:role")
    async updateRole(@Param("idUser") idUser: number,
@Query('role') role: adminRole = adminRole.ADMIN) {
        return this.userService.updateRole(idUser, role);
    }
}

```

```

    @ApiOperation({summary: 'Оновлення магазину'})
    @ApiParam({ name: "idUser"})
    @ApiParam({ name: 'idShop' })
    @Patch("shop/:idUser/:idShop")
    async updateShop(@Param("idUser") idUser: number,
    @Param("idShop") idShop: number) {
        return this.userService.updateShop(idUser, idShop);
    }

    @ApiOperation({summary: 'Видалити магазин у користувача'})
    @ApiParam({ name: "idUser"})
    @Delete("shop/remove/:idUser")
    async removeShop(@Param("idUser") idUser: number) {
        return this.userService.removeShop(idUser);
    }

    @ApiOperation({summary: 'Встановити аватар користувачу'})
    @Post('avatar/:idUser')
    @ApiConsumes('multipart/form-data')
    @ApiBody({
        schema: {
            type: 'object',
            properties: {
                file: {
                    type: 'string',
                    format: 'binary',
                },
            },
        },
    })
    @ApiParam({name: "idUser"})
    @UseInterceptors(FileInterceptor('file'))
    uploadFile(@Param('idUser') idUser: number,
    @UploadedFile('file') file) {
        return this.userService.uploadAvatar(idUser, file);
    }

    @ApiOperation({summary: 'Видалити аватар'})
    @ApiParam({ name: "idUser"})
    @Delete("avatar/remove/:idUser")
    async removeFile(@Param("idUser") idUser: number) {
        return this.userService.removeAvatar(idUser);
    }
}

```

Повний код проекту за посиланням: <https://github.com/CoraLion/backend-degree-work>

ДОДАТОК Г – КОД ПРОГРАМИ FRONTEND

//Код класів, які вираховують дані за допомогою алгоритму Дейкстри

```
import PriorityQueue from "@mixins/priorityClass";

export default class Graph {
  constructor() {
    this.nodes = {};
  }

  addNode(node, edges) {
    this.nodes[node] = edges;
  }

  dijkstra(startNode) {
    const distances = {};
    const visited = {};
    const priorityQueue = new PriorityQueue();

    for (const node in this.nodes) {
      distances[node] = node === startNode ? 0 : Infinity;
      priorityQueue.enqueue(node, distances[node]);
    }

    while (!priorityQueue.isEmpty()) {
      const currentNode = priorityQueue.dequeue().element;

      if (!visited[currentNode]) {
        for (const neighbor in this.nodes[currentNode]) {
          const currentDistance =
distances[currentNode];
          const edgeWeight =
this.nodes[currentNode][neighbor];
          const newDistance = currentDistance +
edgeWeight;

          if (newDistance < distances[neighbor]) {
            distances[neighbor] = newDistance;
            priorityQueue.enqueue(neighbor,
newDistance);
          }
        }

        visited[currentNode] = true;
      }
    }

    return distances;
  }
}

export default class PriorityQueue {
  constructor() {
```



```

        this.items = [];
    }

    enqueue(element, priority) {
        const queueElement = { element, priority };
        let contain = false;

        for (let i = 0; i < this.items.length; i++) {
            if (this.items[i].priority > queueElement.priority) {
                this.items.splice(i, 0, queueElement);
                contain = true;
                break;
            }
        }

        if (!contain) {
            this.items.push(queueElement);
        }
    }

    dequeue() {
        if (this.isEmpty()) {
            return "Underflow";
        }
        return this.items.shift();
    }

    isEmpty() {
        return this.items.length === 0;
    }
}

```

//Скрипт для зміна кольорової схеми сайту

```

import store from '@store/index';

(function () {
    if (localStorage.getItem('theme') === 'light-theme'){
        localStorage.setItem('theme', 'light-theme');
        store.dispatch('ACTIVE_COLOR', false)
        document.body.className = 'light-theme'
    } else {
        localStorage.setItem('theme', 'dark-theme');
        document.body.className = 'dark-theme';
        store.dispatch('ACTIVE_COLOR', true)
    }
})();

export default {
    methods: {
        setTheme(themeName) {

```

```

        localStorage.setItem('theme', themeName);
        document.body.className = themeName;
        this.changeActiveColor();
    },

    changeActiveColor() {
        if(localStorage.getItem('theme') === 'light-theme') {
            this.$store.dispatch('ACTIVE_COLOR', false)
        } else {
            this.$store.dispatch('ACTIVE_COLOR', true)
        }
    },

    toggleTheme() {
        if (localStorage.getItem('theme') === 'light-theme'){
            this.setTheme('dark-theme');
        } else {
            this.setTheme('light-theme');
        }
    },

},

}

```

```

//Модуль із Vuex, який відповідає за сховище Користувачів

import {api} from "@providers/api";
import store from "@store";

export const userModules = {

  state: () => ({
    activeMe: undefined,
    infoUser: undefined,
    activeUser: {},
    infoEducationLevel: []
  }),
  getters:{
    getUsersAll: (state) => {
      return state.infoUser
    },

    getEducationLevelAll: (state) => {
      return state.infoEducationLevel;
    },

    getMeInfo: (state) => {
      return state.activeMe;
    },

    getActiveUser: (state) => {
      return state.activeUser;
    }

  },
  mutations: {
    SET_ME(state, userInfo) {
      state.activeMe = userInfo
    },
    SET_USERS(state, userInfo) {
      state.infoUser = userInfo
    }
  },
  actions: {
    async getUsers({commit}, { params } = { params: {}}) {
      let response = await api.user.userControllerGetAll({
        ...params
      }).then(r => r.data);
      commit('SET_USERS', response?.rows)
    },

    async getUsersSearch({commit}, search) {
      console.log(search)
      let response = await api.user.userControllerGetAll({
        search: search
      }).then(r => r.data);
    }
  }
}

```

```

        commit('SET_USERS', response?.rows)
    },

    async getMe({commit}, { params } = { params: {} }) {
        let response = await
api.auth.authControllerGetMeAuth({
        ...params
    }).then(r => r.data)
        commit('SET_ME', response)
        let activeTypeInvoices;
        if(response?.adminRole === 'STOCK') {
            activeTypeInvoices = 'Stock'
        } else {
            activeTypeInvoices = 'Store'
        }
        await store.dispatch('getActiveInvoices', {
            params: {
                typeInvoices: activeTypeInvoices
            }
        })
        return response
    },
}
}

```

//Компонент модального вікна переміщення товару

<template>

```

<modal-frame :button-text="buttonName"
             :active-alert-vuetify="true"
             :main-title="buttonName"
             :disabled-button="!isValid"
             @submitInfo="submitInfo">

```

```

<v-alert
  :color="'#9D1019'"
  closable
  v-model="isActiveAlert"
  icon="$error"
  :text="alertText"
  type="error"
></v-alert>

```

```

<div class="input-info-content" v-if="!isLoadingSearch">

```

```

  <div class="input-block-content">
    <sub-title>
      {{ title[0] }}
    </sub-title>

```

```

    <block-input-static>
      {{ activeShopAmount?.name }}
    </block-input-static>

    <block-input-static>
      {{ activeShopAmount?.amount }}
    </block-input-static>
  </div>

  <div class="all-shop-container">

    <sub-title>
      {{ title[1] }}
    </sub-title>

    <input-main :name-block="'modal-frame-container'"
      @changeTextInput="changeSearchText"
      :info="inputInfoSearch"/>
    <list-block :info-list="filterNameProductShop"
      @clickTo="clickTo"
      :active-hover="true"
      :controller-block="true"
      :active-download="false"/>

    <div class="result-container" v-if="activeShop !==
undefined">
      <transition-group name="fade-translate-x">
        <mini-item :item="activeShop.text" :id="0"
@removeItem="removeItem"/>
      </transition-group>
    </div>
  </div>

  <div class="input-block-content">
    <sub-title>
      {{ title[2] }}
    </sub-title>

    <input-main :name-block="'modal-frame-container'"
      @changeTextInput="changeTextInput"
      :info="inputInfo"/>
  </div>

</div>
</modal-frame>
</template>

<script>
import ModalFrame from "@components/entities/ModalFrame.vue";
import InputMain from "@components/shared/InputMain.vue";
import {ca} from "vuetify/locale";
import {api} from "@providers/api";

```

```

import BlockInputStatic from
"@/components/shared/BlockInputStatic.vue";
import SubTitle from "@/components/shared/text/SubTitle.vue";
import {mapGetters} from "vuex";
import ListBlock from "@/components/entities/ListBlock.vue";
import MiniItem from "@/components/entities/MiniItem.vue";

export default {
  name: "ModalTransportProduct",
  components: {MiniItem, ListBlock, SubTitle, BlockInputStatic,
InputMain, ModalFrame},
  props: {
    activeShopAmount: {
      type: Object,
      required: true
    },
  },
  data() {
    return {
      title: ['Наявна кількість товару у цьому магазині', 'Наявна
кількість товару в інших магазинах', 'Скільки товару маємо
перемістити?'],
      buttonName: 'Переміщення товару',
      alertText: 'Такої кількості товару у магазині немає!',
      isActiveAlert: false,
      inputInfoSearch:
        {
          id: 0,
          dataIndex: 'name',
          placeholder: 'Пошук',
          type: 'text'
        },

      inputInfo:
        {
          id: 1,
          dataIndex: 'text',
          placeholder: 'Кількість товару',
          type: 'number'
        },
      activePrice: {},

      searchText: '',

      activeShop: undefined,
      activeAmount: 0
    }
  },
  computed: {
    ...mapGetters(['getShopAll']),
    isValid() {
      return this.activeShop !== undefined && this.activeAmount

```

```

    !== 0
  },
  nameProductShop() {
    let a = this.activeShopAmount?.shop.map(item => ({
      id: item?.id,
      text: item?.name,
      amount: item?.ProductToShop?.counter
    }))
    let idShopActive = a.map(item => item.id)
    let filterShop = this.getShopAll?.filter( itemShop =>
    !idShopActive.some(item => item === itemShop.id)).map(item => ({
      id: item?.id,
      text: item?.name,
      amount: 0
    }))
    return [...a, ...filterShop].filter(item => item.id !==
    Number(this.$route.params.idShop))
  },

  filterNameProductShop() {
    if(this.searchText === '') {
      return this.nameProductShop;
    } else {
      return this.nameProductShop.filter(item =>
item?.text.toLowerCase().indexOf(this.searchText.toLowerCase())
!== -1)
    }
  }
},
methods: {

  changeTextInput(text, dataIndex) {
    this.activeAmount = text
  },

  async submitInfo() {
    if(this.activeAmount <= this.activeShopAmount?.amount) {
      try {
        await api.product.productControllerUpdateAmount(
          Number(this.$route.params.idShop),
          this.activeShop.id,
          this.activeShopAmount.id,
          this.activeAmount).then(r => {
            localStorage.setItem('successItem', 'true')
            this.$router.go(0)
          })
      } catch (e) {
        console.log(e)
      }
    } else {
      this.isActiveAlert = true
    }
  },

```

```

        closeModal() {
            this.$emit('closeModal')
        },

        clickTo(item) {
            this.activeShop = item
        },
        removeItem() {
            this.activeShop = undefined
        }
    }
}
</script>

<style scoped lang="scss">

.v-alert {
    position: absolute;
    left: 50%;
}

.input-info-content {
    display: flex;
    flex-direction: column;
    row-gap: 16px;

    .input-block-content {
        display: flex;
        flex-direction: column;
        row-gap: 8px;
    }

    .all-shop-container {
        display: flex;
        flex-direction: column;
        row-gap: 8px;
    }
}
</style>

```

Повний код проекту за посиланням: <https://github.com/CoraLion/frontend-degree-work>