ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**Створення сервісу для автоматизації роботи складу товарів мовою C#**»

Виконав: студент 4 курсу, групи ПД–43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Воронецький М.А.
(прізвище та ініціали)

Керівник Коба А.Б.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормконтроль _____
(прізвище та ініціали)

Київ –2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Бакалавр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____Негоденко О.В.

“____” _____2022 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

Воронецький Микита Андрійович

(прізвище, ім’я, по батькові)

1. Тема роботи: **«Створення сервісу для автоматизації роботи складу товарів мовою C#»**

Керівник роботи: Коба Андрій Борисович, старший викладач

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «16» лютого 2022 року № .

2. Строк подання студентом роботи 03.06.2022

3. Вхідні дані до роботи

Методи складського обліку, технічна література програмного забезпечення з приводу складського обліку, технічна література .NET, технічні засоби розробки такі як Visual Studio – середа розробки, методи та бібліотеки C#, MongoDB, MongoDB Compass, Postman для керування і тестування API.

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

4.1 Опис предметної області автоматизації складського обліку.

4.2 Аналіз наявних засобів та технологій для організації складського обліку.

4.3 Моделювання та проектування інформаційної системи для автоматизації складського обліку.

4.4 Реалізація інформаційної системи для автоматизації складського обліку.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

5.1. Титульний слайд

5.2. Аналоги

5.3. Аналіз аналогів

5.4. Мета, об’єкт та предмет дослідження

5.5. Технічне завдання

5.6. Засоби реалізації

5.7. Сценарії використання

5.8. Проектування діяльності

5.9. Діаграма класів

5.10-5.13. Приклади запитів API

5.14. Апробація результатів дослідження

5.15. Висновки

5.16. Кінцевий слайд

6. Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.03.22-14.04.22	Виконано
2	Розробка вимог до системи	15.04.22-20.04.22	Виконано
3	Проектування системи	20.04.22-01.05.22	Виконано
4	Концепція та архітектура програмного забезпечення	01.05.22-05.05.22	Виконано
5	Програмне реалізація системи, написання коду	05.05.22-20.05.22	Виконано
6	Вступ, висновки, реферат	21.05.22-30.05.22	Виконано
7	Розробка обов‘язкових демонстраційних матеріалів	03.06.22-05.06.22	Виконано
8	Попередній захист роботи	06.06.22	Виконано
9	Здача роботи	15.06.22	

Студент _____ Воронецький М.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Коба А. Б.
(підпис) (прізвище та ініціали)

Реферат

Об'єкт дослідження – автоматизований облік товарів на складі

Предмет дослідження – API сервер та база даних для складського обліку

Мета роботи – створення доступного для малого бізнесу сервісу для автоматизації роботи складу, обліку товарів та створення звітності.

Методи дослідження – методи створення складу в базі даних та керування ним (додавання, видалення товару, користувачів).

При створені проведено розбір створених програмних продуктів, таких як SUBTOTAL, LiteBox та ін. систем складського обліку репрезентовані у роботі таких програмних продуктів – використання методів POST завантажень.

Особливістю є стабільна робота БД та серверу та отримання швидкого з'єднання з ними користувача. Програмне забезпечення враховує ці потреби.
Веб-додаток написано за допомогою ASP.NET Core 5 (Web API) з використанням бази даних MongoDb.

Таким чином було розроблено та описано серверну частину та базу даних який виконує потреби потенційного користувача у створенні віртуального складу та контролем за ним.

Галузь використання – бізнес, підприємство, складський облік.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система

ПК – персональний комп’ютер

ПЗ – програмне забезпечення

ARCnet – Attached Resource Computer NETwork

HDD - Hard Disk Drive

БЖ – блок живлення

DS – Data Storage

NAS – Network Area Storage

IP – Internet Protocol

VLAN – Virtual Local Area

НСД – несанкціонований доступ

ІС – інформаційна система

СЗД – система зберігання даних

ЦОЗД – центр зберігання та обробки даних

SQL – сервер баз даних

СУБД – система управління базами даних

HTTP – Hyper Text Transfer Protocol

QA – Quality Assurance

API - Application Programming Interface

DIP – Dependency Inversion Principle

IoC – Inversion of Control

DTO – Data Transfer objects

JWT – JSON web token

ВСТУП

Актуальність дослідження. Перш за все дане програмне забезпечення є актуальним для людей працюючих у сфері бізнеса та виробництва товарів. Актуальність ПЗ перш за все очевидна для таких професій як складський менеджер або керівник складу. Питання контролю за складом є дійсно важливим у період інформаційних технологій, адже набагато зручніше коли склад товарів завжди під рукою керівника – у ПК або на смартфоні, а не в складській документації.

В XXI сторіччі, коли підприємства виробляють величезні обсяги товарів, коли процвітають тисячі відомих компаній з продажу цих товарів, мати складське приміщення і навіть не одне – це необхідність. Навіть для середнього та малого бізнесу, який займається продажом будь-яких товарів зазвичай потрібно мати свій склад (для прикладу уявіть звичайний інтернет-магазин з продажу дитячих іграшок).

Завдяки сучасним засобам розробки та IT-технологіям в цілому ми маємо можливість створити проект який повність задовольняє потреби потенційних користувачів. В процесі роботи було розроблено ПЗ, яке має можливість створювати віртуальний склад та вести облік товарів в мережі, зекономивши час та ресурси витрачені на ведення спеціальної документації.

Об’єкт дослідження: API сервер для віртуального складу і обліку товарів.

Предмет роботи: програмне забезпечення для складського обліку.

Мета роботи: пришвидшення та спрощення процесу обліку товарів складу.

Завдання роботи: розробка програмного забезпечення для складського обліку товарів, а також бази даних та серверу для нього.

Методика дослідження:

1. Обрати найкращій метод упорядковування даних за допомогою сервера та БД;
2. Обрати архітектуру майбутньої системи управління складом;
3. Обрати модель розробки програмного забезпечення.

Таким чином, враховуючі потреби системи в цілому та існуючи методи

реалізації, було зроблено висновок що найбільш актуальним варіантом буде створення API-серверу та розробка бази даних.

Працювати сервер буде за основними принципами клієнт-серверної архітектури.

Наукова новизна роботи: наукова новизна роботи полягає в створенні API серверу та бази даних до нього, а також створенні інформаційної системи на обраній мові програмування, що дозволить зберігати дані складу на сервері, генерувати звіти у CSV форматі, а також створювати необмежену кількість віртуальних складів. Також варто додати використання MongoDB, що допомагає значно пришвидшити пошук по базі даних.

Практична значущість результатів*:* даний продукт може бути використаний у всіх сферах діяльності, де потрібен сервер з використанням IP для керування віртуальним складом (створення складу, додання та видалення товарів, додання користувачів до складу – потенційних працівників).

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ОРГАНІЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ

1.1 Загальні відомості про облік товарів

Реалізація складського обліку має великий пріоритет в умовах розвитку будь-якого бізнесу, пов’язаного з купівлею та продажом товарів або з виробництвом. Існує дуже багато варіантів складського обліку, в тому числі і найпростіший: ведення фізичної документації. Але в сучасному світі є тенденція автоматизувати та спрощувати такі процеси.

Усім успішним підприємцям відомо, що склад є важливою складовою будь-якої організації, яка відповідає за ефективне зберігання та переміщення товарів компанії. Якщо ж складським процесам не приділяється достатньо часу, це може мати значний негативний вплив на загальну ефективність компанії.

В умовах загострення конкуренції рівень технічного оснащення складу визначає такі критично важливі параметри як час і технологічність обробки вантажів з товаром. Ефективно організована складська діяльність позитивно впливає на скорочення логістичних витрат підприємства. Існує велика кількість умов, таких як глобалізація та розвиток мереж для поставок, на основі яких можна сформулювати певні вимоги до складського ПЗ:

- виконання більшої кількості операцій при менших обсягах партій нових та вже реалізованих товарів;
- складування найбільшого числа найменувань товарів та регулярного їх оновлення;
- постачання клієнтам товарів та послуг високої якості;
- збільшення діапазону послуг з передпродажної та підготовки товарів до продажу;
- збереження необмеженої кількості одиниць товару на складі для великих підприємств;

- можливість швидкого та зручного створення віртуального складу;
- можливість створювати необмежену кількість віртуальних складів;
- розширення додаткових шляхів поставок та закупівлі товару.

1.2 Облік товарів в торгівлі

Торгівельні організації залежно від сфери діяльності та виконуваних функцій можуть бути оптовими, роздрібними та оптово-роздрібними. У найскладнішому своєму вигляді торгова організація може здійснювати купівлю (заготовлення), переробку, зберігання та реалізацію перероблених чи придбаних для продажу товарів.

Торгівельна організація має вести бухгалтерський облік існуючих і реалізованих товарів усім етапах їх руху.

1.3 Облік товарів, що надходять до складу

Товари, що надійшли, приймаються організацією за фактичною кількістю в день їх надходження або не пізніше термінів, встановлених для приймання за якістю.

Матеріально-відповідальні особи ведуть облік товарів, що надійшли на склад, у картках або книгах кількісного обліку цінностей за найменуваннями товарів, сортами та з урахуванням інших їх особливостей.

Дані про кількісний облік товарів, що надійшли, узагальнюються в бухгалтерії організації у відомостях руху товарів, що є документами їх аналітичного обліку.

Регістром обліку руху товарів є зведена відомість руху товарів. Ця відомість складається з підсумкових даних відомостей про рух товарів, а також з даних відомостей надходження товарів.

Бухгалтерський облік товарів, що надійшли, ведеться за їх покупною (фактичною) собівартістю або продажною вартістю. При постановці на облік товарів їх оцінка проводиться виходячи з таких витрат, пов'язаних з їх придбанням:

- Вартості товарів, зазначеної у розрахункових документах продавця. У разі обліку товарів за продажними (роздрібними) цінами, вартість товарів буде їх продажною вартістю;
- Витрат на тару одноразового використання та упаковку, що оплачуються у складі одержуваних товарів та не підлягають до подальшого використання;
- Витрат, пов’язаних з транспортуванням до центрального складу організації.

1.4 Облік реалізації та продажу товарів

Реалізація товарів у торгових та інших посередницьких організаціях вимагає обліку вибуття даних цінностей у кількісному вираженні за напрямами їх вибуття, видами цінностей, і навіть характеру подальшої реалізації.

Облік повинен надавати дані для правильного визначення сум товарообігу та валового доходу від реалізації товарів. Розмір товарообігу та валового доходу визначається рівнем цін реалізації товарів та їх покупною (обліковою) вартістю.

Товари зі складів в оптову та роздрібну торгівлю відпускаються за вагою чи кількістю з оформленням відпустки бланками первинної облікової документації суворої звітності. У необхідних випадках (на продовольчих складах) товари перед їх відвантаженням піддаються перебиранню, сортуванню та калібруванню.

Облік відпустки товарів зі складу ведеться у відомості відвантаження та реалізації товарів. При цьому аналітичний облік відпуску товарів має надавати інформацію про вибуття товарів за такими напрямками:

- в оптову торгівлю;
- у роздрібну торгівлю;
- окремо до магазинів, що входять до єдиної торгової системи однієї організації з центральним складом;
- у магазини, які є дочірніми товариствами;
- у магазини та торгові організації, незалежні від організації – власника товарів.

З окремих відомостей відвантаження товарів, складених за напрямами їх вибуття, складається зведена відомість відвантаження та реалізації товарів у натуральному і вартісному вираженні.

Таким чином, однією з основних завдань обліку вибуття товарів є визначення вартості, якою вони списуються з балансу організації.

1.5 Облік втрачених товарів

Втрати товарів виникають у процесі їх транспортування (перевезення) до складу, зберігання та реалізації. Втрати товарів можуть бути спричинені їх природним убутком, результатом безгосподарності або обставинами непереборної сили. Поряд із цими втратами за окремими видами товарів, що вимагають переробки до моменту їх відпустки в торгівлю, втрати виникають також і безпосередньо на стадії переробки.

Результат перебирання, сортування, калібрування товарів за кожним їх видом відображається у спеціальному акті про їх перебирання, який підписується членами комісії, затвердженої з цією метою керівником організації. В акті показується

кількість товару (продукції), відпущеного у перебирання, допущеного до реалізації та непридатного для використання. Акт складається у двох примірниках – один для бухгалтерії, інший – залишається у матеріально-відповідальної особи.

Акт складається за кожен вид товару (продукції). Акт фіксує на дату переборки фізичний стан окремих товарів та служить для контролю за правильним оформленням вибуття товарів та списання їх вартості з підзвіту матеріально-відповідальної особи.

Втрати товарів усім стадіях їх руху поділяються на нормовані втрати і ненормовані (надпланові) втрати. До нормованих втрат товарів відносяться їх втрати в межах норм природних втрат, що відбулися з різних причин при перевезенні, зберіганні та продажу, пов'язані зі зміною їх фізико-хімічних властивостей (розпил, усушка, розсип, витік, розлив тощо). Норми встановлені у відсотках до продажної вартості товарів та диференційовані за видами товарів, часом року їх використання, кліматичними зонами, термінами та умовами зберігання.

Норми є визначними межами для списання втрат на витрати організації та застосовуються лише за наявності фактичних втрат товарів. Норми природних втрат можуть також встановлюватися у процентах на звітний рік до товарообігу організації.

Норми природних втрат застосовуються до товарів, відпущених зі складу у період між двома інвентаризаціями тих самих товарів з урахуванням терміну їх зберігання. Якщо облік товарів складі ведеться за сортами, то при визначенні розміру природних втрат необхідно розраховувати середній термін зберігання товару на складі. Нестача товарів у межах встановлених норм природних втрат списуються з матеріально-відповідальних осіб за цінами, якими товари були оприбутковані. Віднесення товарних втрат до витрат звернення проводиться у покупних цінах. Різниця між покупними та продажними (роздрібними) цінами відноситься за рахунок торгових знижок.

1.6 Особливості обліку складських операцій

Від постачальників або транспортних організацій (на товарних станціях) матеріали, що надходять, отримує експедитор підприємства. Отримуючи вантаж, експедитор звіряє його з транспортними документами. При отриманні вантажу від транспортних організацій перевіряється кількість, вага, збереження упаковки. У разі виявлення розбіжностей із транспортними документами складається комерційний акт. Він є основою пред'явлення претензій до транспортної організації.

Отриманий вантаж доставляється складу і передається комірнику під розписку. При прийманні на складі перевіряється кількість, асортимент, якість товарів, що надійшли. Прийняті товари оформлюються прибутковими ордерами. На складі ведеться кількісний облік товарів. І тому бухгалтерія відкриває картку складського обліку за кожен номенклатурний номер товарів. У картці вказуються номер складу, марка, сорт, профіль, розмір, одиниця виміру, ціна та найменування товару, а також дані про рух цінностей, що враховуються.

Потім у картці робляться записи за кожним прибутковим (витратним) документом окремим рядком. Дані лімітно-забірних карток записуються за підсумком. Після закінчення місяця всі залишки з карток переносяться у відомість залишків товарів на складі. Відомість залишків передається в бухгалтерію або обчислювальний центр, де проводиться оцінка залишків товарів на складі.

За умови автоматизації обліку складський облік товарів замість картотеки ведеться за допомогою комп'ютерів. Нижче представлена схема автоматизованої форми обліку:

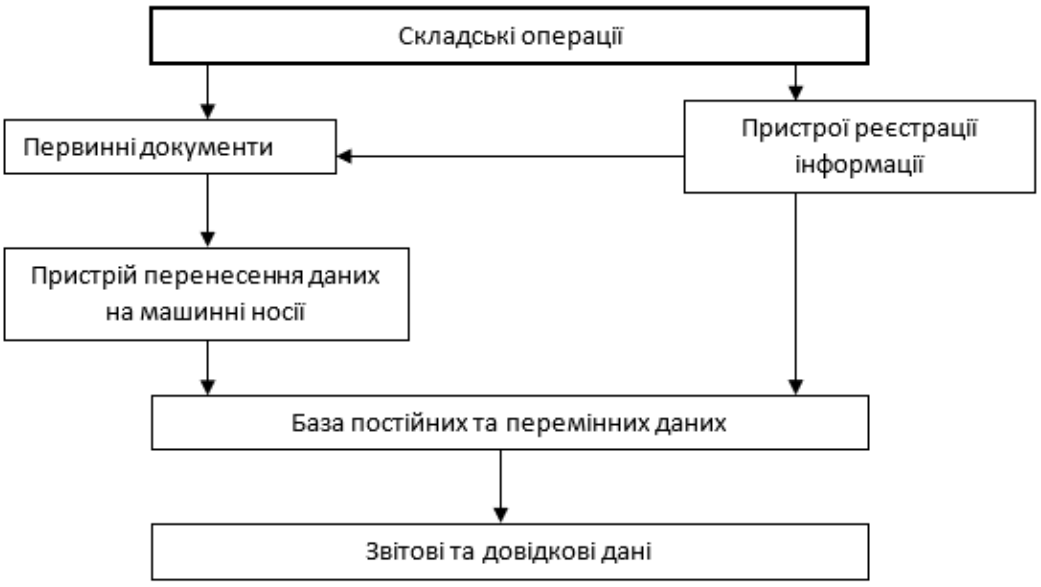


Рис. 1 - Схема автоматизованої форми обліку

Реєстри складаються окремо після приходу та витрати товарів на складі. За даними реєстрів та доданих до них первинних документів бухгалтерія складає накопичувальні відомості щодо приходу та витрати товарів за місяць.

Для обліку руху товарів застосовується первинна облікова документація, що відповідає вимогам Основних положень з обліку матеріалів, та пристосована для автоматизованої обробки. Кількість примірників документів, що виписуються, та їх документообіг кожна організація встановлює самостійно виходячи з виду діяльності, рівня автоматизації та системи обліку.

Картка обліку товару застосовується для обліку руху товарів на складі по кожному сорту, виду, розміру. Картка заповнюється на кожний номенклатурний номер. Веде картку відповідальна особу на складі. Записи у картці вносяться виходячи з первинних прибутково-витратних документів у день здійснення операції. На кожен номенклатурний номер заводиться окрема картка, де заповнюються всі реквізити даного товару (ціна, одиниці виміру, постачальник і т.д.).

РОЗДІЛ 2. АНАЛІЗ НАЯВНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ДЛЯ ОРГАНІЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ

Отже розглянемо існуючі програми для виконання складського обліку. Першим аналогом буде програмний модуль «Kataloger». Kataloger – безкоштовний універсальний каталогізатор, завдяки якому користувач може систематизувати абсолютно будь-яку інформацію, будь то контактні дані друзів і знайомих (П.І.Б., адреса, телефон, email та ін.), список дисків, книг, монет, велосипедів, одягу і багато іншого. У програмі є можливість прикріплювати до будь-якого запису в таблиці коментар, а також файли і папки (рис. 2). Додаток не вимагає установки.

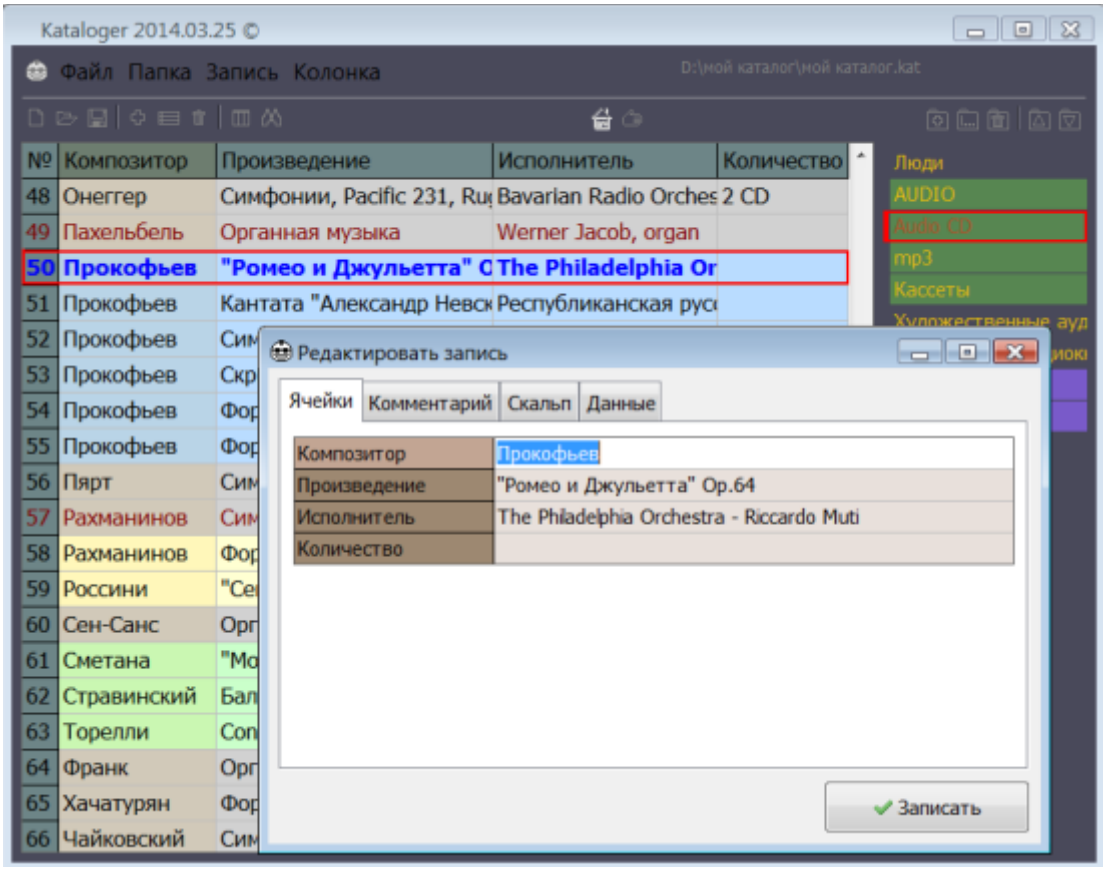


Рис. 2 - Програмний модуль «Kataloger»

Основні можливості «Kataloger»:

- табличне відображення даних;
- не потрібно спеціальної установки: досить просто скопіювати файл .exe формату. Всі налаштування зберігаються в файлі «settings.ini» в папці програми);
- простий мінімалістичний зрозумілий інтерфейс;
- розумне сортування;
- можливість настроювання шрифтів (тип, розмір, стиль, колір).

До недоліків даного програмного продукту слід віднести відсутність україномовної версії, що сьогодні є обов’язковою вимогою до підприємств на території нашої країни.

Розглянемо програмний модуль «ТріумфV2.0» як один з аналогів. «ТріумфV2.0» – програма призначена для оперативного обліку товару і грошей (рис. 3).

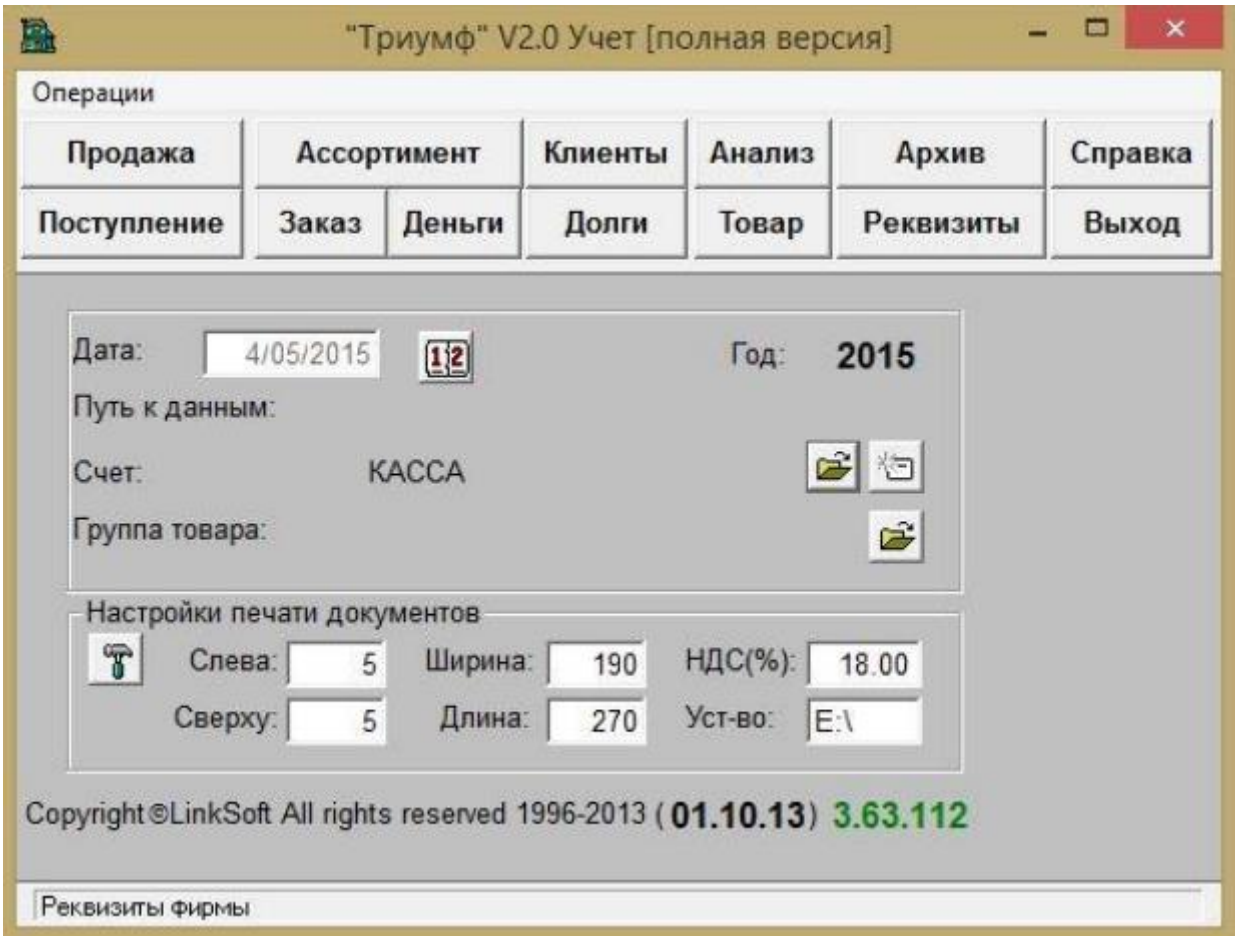


Рис. 3 - Програмний модуль «ТріумфV2.0»

Даний додаток дозволяє вести фактичний облік на складі або магазині. Програмне забезпечення веде облік товарів, контролює процеси продажу, витрати і надходження коштів. Окрім цього є властивість контролювати борги та залишки товарів, додавати коментарі до записів у таблиці, додавати файли та папки до записів в таблиці.

Складський облік можна вести по одному або декількох підприємствах. Кількість складів, кас, розрахункових рахунків не обмежена.

Програма, за допомогою якої можна керувати процесом виробництва, починаючи з надходження окремих матеріалів на склад і закінчуючи випуском готового виробу. Склад виробництва буде корисний технологам, логіста, збирачам, фахівцям з постачання, начальникам виробництв, фахівцям складського обліку, бухгалтерам і генеральним директорам підприємств різних сфер діяльності і

масштабів бізнесу. Для кожного фахівця в програмі є інформація тому програма охоплює весь виробничий цикл.

Кожен користувач має свій рівень доступу до програми і працює тільки в своїй робочій області. Безліч звітів, які дозволять відобразити оборот виробництва, відомість закупівлі матеріалів з урахуванням поставки його в упаковках та багато інших.

Додаток не вимагає установки ви зможете контролювати продавців в магазині. Окрім цього додаток дуже простий у використанні, що дозволяє прискорити процес навчання персоналу складу, магазину і т.д. Оскільки програма підтримується вже 19 років, зворотній зв’язок з користувачем дозволив покращити програму та додати нові можливості для роботи зі сховищем. Розглянемо можливості користувача в програмі:

- будь-яка кількість складів матеріалів;
- будь-які технологічні процеси виробництва;
- будь-яка кількість складів готових виробів;
- можливість автоматичного завантаження прайс-листа, щоб не набивати довідник вручну;
- можливість градації цін на різних складах;
- заміна відсутнього матеріалу при відправці в виробництво.

Основні можливості програми «ТріумфV2.0»:

- перегляд руху товару в надходженні і продажу;
- друк цінників, прайс-листів, залишків товарів, нових рахунків фактур, накладних, прибуткових і видаткових касових ордерів, нових книг продажів і покупок.

Ще одним аналогом автоматизованої системи для керування складом є програмне забезпечення «BAS Управління торгівлею» (рис.4). Це сучасний інструмент підвищення ефективності бізнесу торгівельного підприємства. Рішення

дозволяє збільшити продуктивність праці всіх служб торгівельного підприємства. ПЗ працює з оперативною інформацією, яка відображає поточний стан підприємства, а також дозволяє швидко та у зручній формі отримувати звіти для прийняття рішень на різних рівнях.

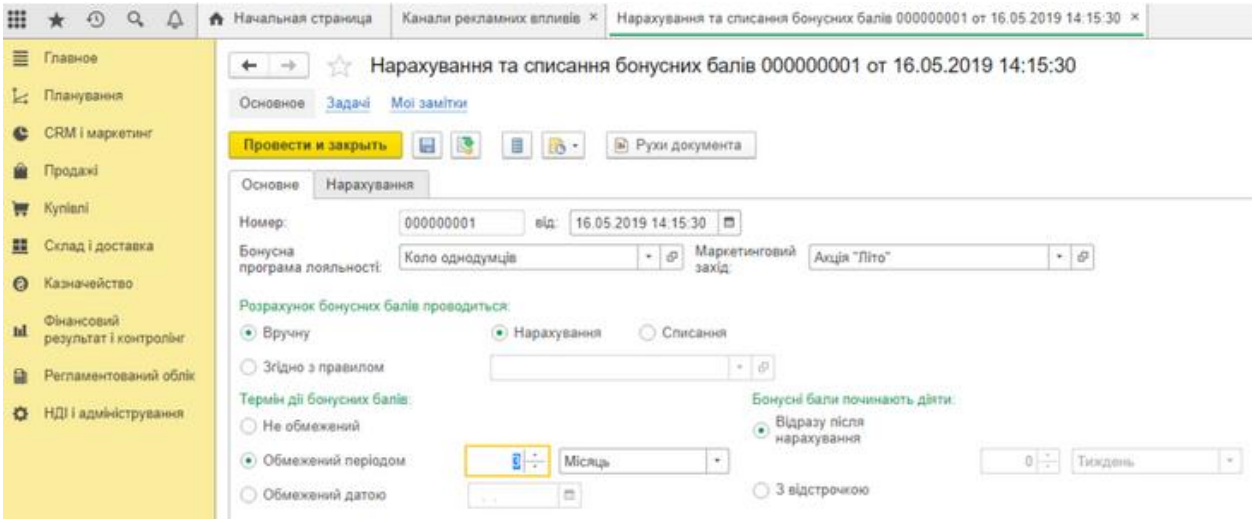


Рис. 4 - Програмне забезпечення «BAS Управління торгівлею»

Програмне забезпечення дозволяє виконувати наступні завдання складського обліку:

- планування, аналіз продажів, закупівель, збирання (розбирання);
- управління продажами, постачаннями, складськими запасами;
- управління відносинами з постачальниками та замовниками;
- управління замовленнями;
- самообслуговування клієнтів через Інтернет;
- робота торгових представників;
- обробка претензій;
- управління грошима;
- облік і аналіз комерційних витрат;
- керування взаєморозрахунками;

- аналіз цін і управління ціновою політикою;
- інтеграція з торговим обладнанням;
- аналітична звітність з торгової діяльності;
- спільна робота з «BAS Роздрібна торгівля» та «BAS Бухгалтерія».

Таким чином, можна зробити висновок що дане ПЗ виконує має широкий спектр можливостей та виконує майже всі основні потреби складського обліку. Додаток, що не менш важливо, інтегрований з іншим та дозволяє працювати на обох рівнях – складського обліку та бухгалтерії.

Розглянемо ще один додаток, який має назву «SAP» (рис. 5). Додаток є одним з провідних світових постачальників корпоративних програмних рішень як на території Німеччини, так і на території Європи. Вона організовує різні процеси всередині підприємства і між компаніями. Система включає в себе бізнес-додатки для великого і середнього бізнесу, а також стандартні рішення для малих і середніх компаній.

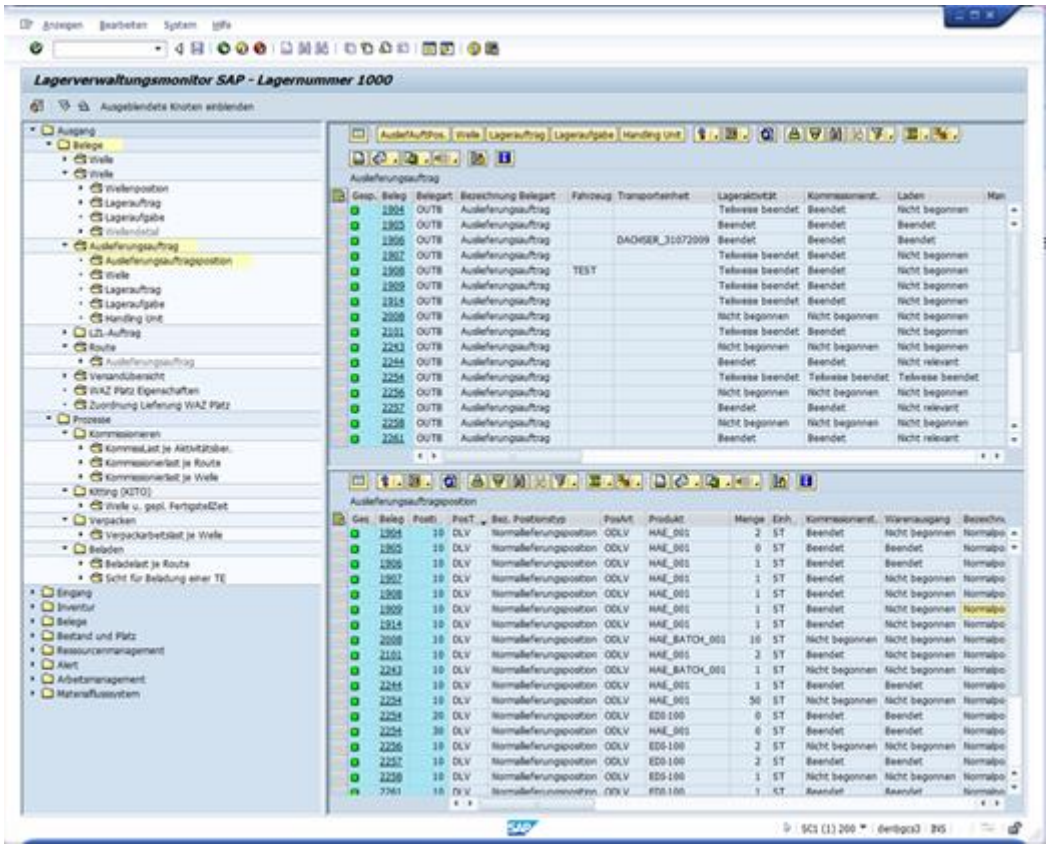


Рис. 5 - Програмне забезпечення «SAP»

SAP має широкий спектр вбудованих стандартних функцій. Наприклад, в процесах «Продажі», «Відвантаження» або «Управління запасами» дані автоматично переносяться в функції обліку. Додатки SAP Business Suite допомагають індивідуально керувати найважливішими бізнес-процесами. В цілому, вони надають тісно інтегроване пакетне рішення.

Хоч вищезгадані рішення для багатьох малих підприємств значно знизили б поріг для використання системи ERP, справжнім проривом є ERP хмарні, які стали доступними лише протягом декількох років. Це програмне забезпечення як сервісне рішення, де користувач може отримати доступ до готового програмного пакету ERP онлайн і використовувати його для планування і контролю ресурсів. Незалежно від того, в яких галузях активний користувач - в разі хмарної ERP витрати набагато більш точно розраховуються і контролюються, ніж витрати на локальні рішення.

Крім того, постачальники надають багаторівневі рішення для захисту даних, які у багато разів перевищують рівень безпеки на рівні підприємства. Для малих підприємств, які не можуть дозволити собі ні програмного забезпечення, ні обладнання, не кажучи вже про ІТ-відділі, це може бути кращим інструментом.

Таким чином ПЗ має дуже важливі плюси, серед яких:

- висока ефективність автоматизації складських процесів;
- використання сучасних хмарних технологій;
- підвищений захист даних користувача;
- вирішує проблему автоматизації складу для малого бізнесу за рахунок ефективних технологій керування та доступності використання.

Одним з відомих додатків не тільки для складського обліку, але й для автоматизації торгівлі є Subtotal (рис. 6). Це хмарна система автоматизації торгівлі та складського обліку для роздрібного бізнесу. Сумісна з 1С та онлайн-бухгалтерією «Моя справа».

Таким чином програмне забезпечення виконує наступні задачі складського обліку:

- облік та оптимізація залишків на складі;
- замовлення та повернення товару постачальникам;
- інвентаризація.

Додаткові можливості ПЗ для бізнесу:

- сума грошей у касі, звіти з продажу, середньому чеку та інше;
- оптимізація асортименту;
- управління акціями, знижками;
- друк цінників.

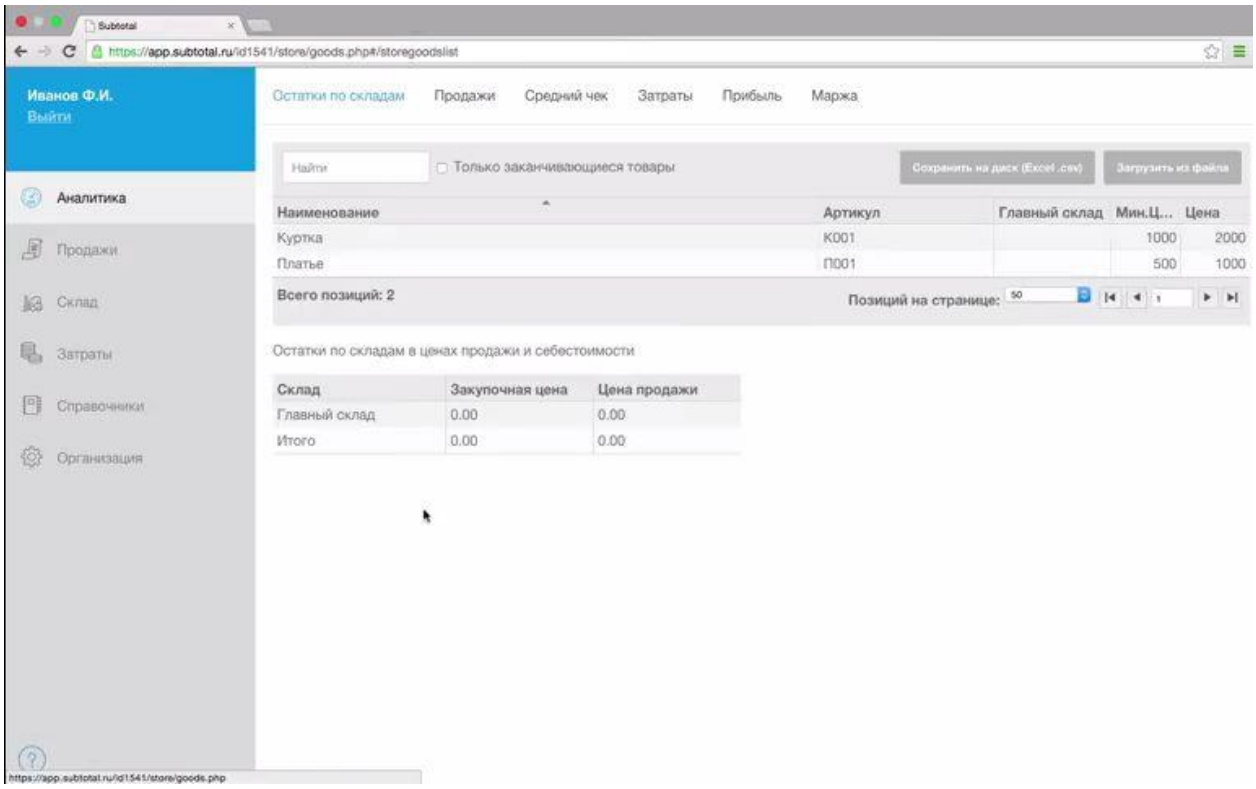


Рис. 6 - Програмне забезпечення "Subtotal"

Але ж також варто відмітити той факт, що розглядаєме ПЗ «Subtotal» лише частково підходить під потреби автоматизації складу, тому варто розглянути інші аналоги з більшим акцентом на автоматизацію саме процесів складського обліку.

Розглянемо як аналог наступне програмне забезпечення Weclapp (рис. 7). Weclapp – німецька компанія з Марбурга. Всі дані зберігаються в Німеччині відповідно до строгих німецьких правил захисту даних. З головних плюсів, є швидке впровадження, що зробило цю систему – вибором 2018 року. Weclapp підтримує всі сфери бізнесу: CRM, ERP-систему, 28 програму обліку та білінгу. Використання єдиного рішення забезпечує значну економію часу та коштів, а також прозорість а роботі. Також, система має інтуїтивно зрозумілий і простий у використанні сучасний, призначений для користувача, інтерфейс. Вбудована вкладка з допомогою для користувача і рекомендаціями щодо використання, роблять роботу з програмним забезпеченням дуже простим. Оскільки Weclapp задуманий як «хмарна система», витрати на впровадження відсутні. Також

виключаються витрати на обслуговування, так як усі оновлення безкоштовні. Завдяки модульній конструкції, ви платите тільки за те, що насправді потрібно вашому підприємству.

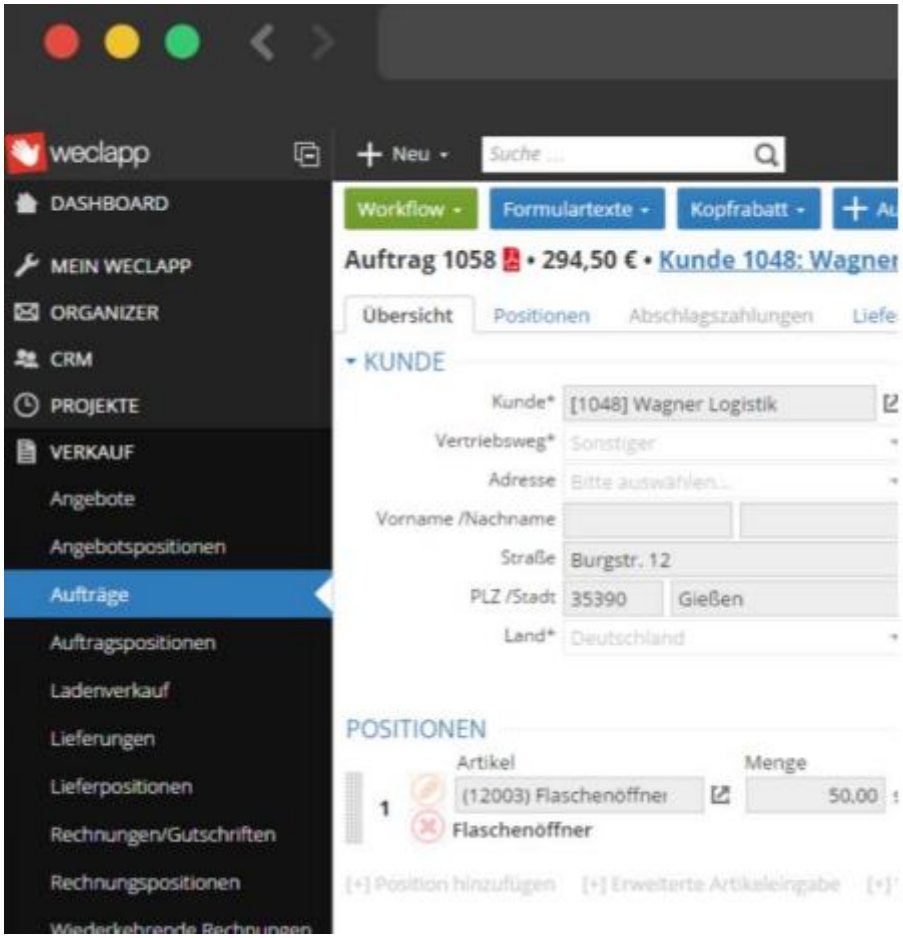


Рис. 7 - Програмне забезпечення "Weclapp"

WeClapp Actindo об'єднує всі процеси торгівлі, доставки, складування і логістики в одну централізовану рішення API First Platform. Значно покращений час завантаження, навігації та панелей моніторингу якщо порівнювати с іншими системами. Нова платформа Actindo Core 1 з функцією масштабування забезпечує оптимальну продуктивність для майбутнього зростання. Рішення Actindo IQS 2.0 забезпечує ефективне управління окремими процесами доставки і маршрутизації, включаючи нові засоби управління MDE на базі Android. Значно простіше інтегрувати додаткові канали поширення (тобто додатки для платформ Android або

IOS) через підхід API-first. Також, завдяки готовими до установки рішеннями в Apple Store, користувач може розширити Actindo бізнес-додатками і компонентами для кожного відділу і галузі.

РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМА ДЛЯ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ

3.1. Модель предметної галузі

Модель предметної галузі – це абстрактна модель предметної галузі бізнесу. Таким чином, основним об’єктом нашого дослідження є саме склад, який напряму пов’язаний з певними видами бізнесу. Проаналізувавши предметну галузь, було побудовано модель, яка представляє предметну галузь «Складський облік» (рис. 8).

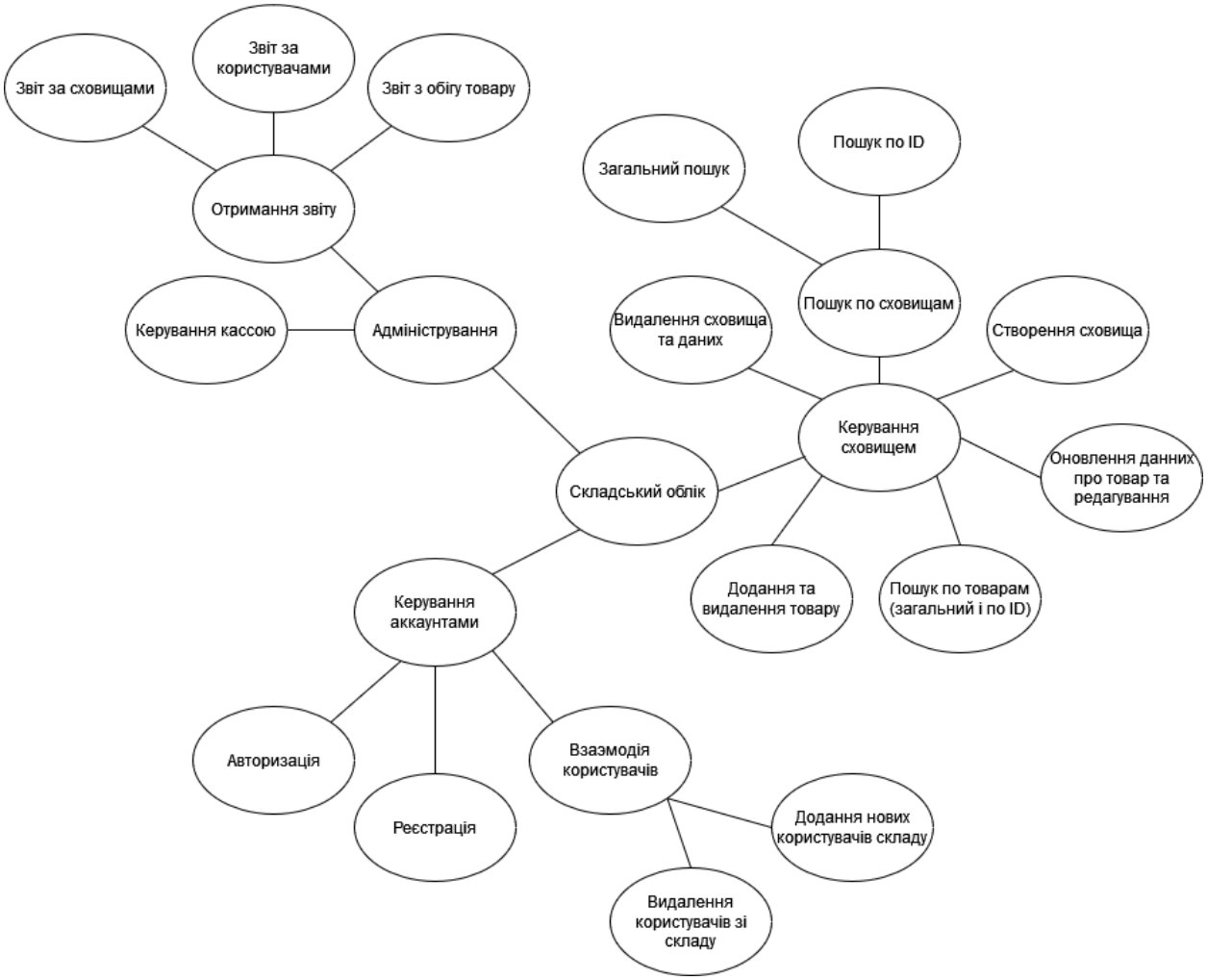


Рис. 8 - Діаграма предметної галузі

Якщо подивитись на діаграму, можна побачити що функція «Складський облік» знаходиться безпосередньо в центрі всієї схеми. Це так, тому що було саме вона була визначена основною для бізнесу. Від неї походять інші функції, задовольняючи потреби користувача, а саме: «Керування сховищем», «Керування аккаунтами» і «Адміністрування».

«Керування сховищем» визначає та відповідає за наступний спектр підфункцій:

- Створення та видалення сховища (складу);
- Пошук по сховищам (загальний пошук і пошук по ID);
- Додання та видалення товару зі складу;
- Пошук по товарам (загальний пошук і пошук по ID);
- Оновлення даних товару (редагування).

«Керування аккаунтами» визначає спектр підфункцій, які відповідають за взаємодію між користувачами одного складу, а також користувачів в системі в цілому. Включає наступні пункти:

- Авторизація;
- Реєстрація;
- Додання та видалення користувачів визначеного складу.

Функція «Адміністрування» є найважливішою саме для керівника або ж власника складу, тобто для бізнесу. З першого розділу ми знаємо про необхідність документування та звітування в області складського обліку, таким чином враховуючи необхідні умови, було виявлено наступні важливі підфункції:

- Керування касою (відповідає за швидке віднімання одиниць товарів у випадку необхідності касового контролю – буде корисним для такого виду бізнесу, як звичайний магазин);
- Створення звіту (за існуючими сховищами, за користувачами конкретного складу та загального звіту з обігу товару).

3.2. Модель прецедентів

Автоматизовану систему складського обліку бізнес може використовувати як окремо для складського обліку, а також у сумісності з касовим апаратом (зчитувальником штрих-коду). Зроблено це було на основі аналізу аналогів програми та виявлення потреби бізнесу мати як мінімум функцію швидкого «віднімання» кількості товару. Таким чином, ми матимемо кілька варіантів використання ПЗ. Давайте розглянемо перший (рис. 9):

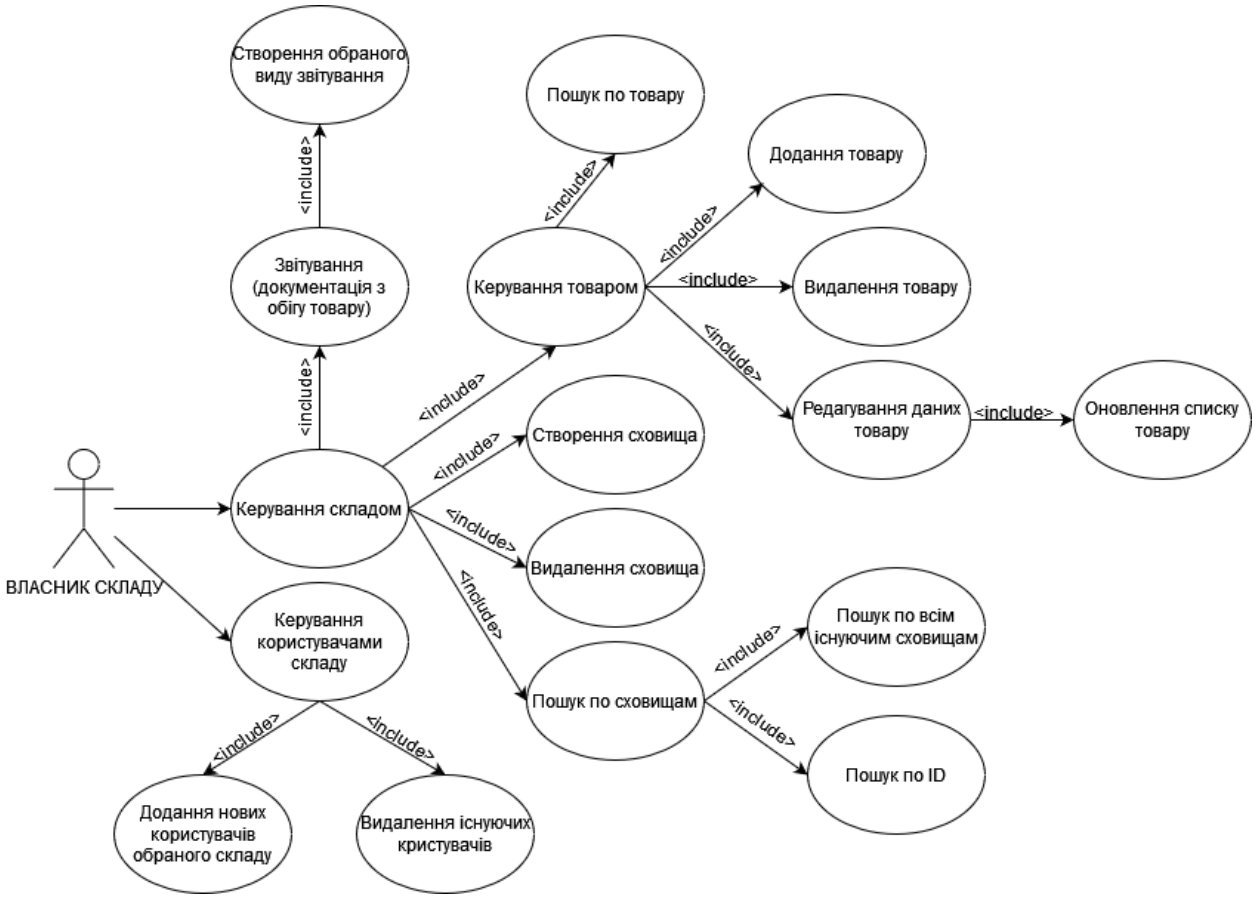


Рис. 9 – Модель прецедентів №1 – Керівник складу

Актором тут є саме власник складу, який може виконувати певні дії за допомогою API. Варто додати що власнику складу надається найбільш широкий спектр можливостей. На діаграмі ви можете побачити основні функції, які буде виконувати розробляємий сервіс.

Наступною буде діаграма, яка зображає можливості звичайного користувача складу (рис. 10). Це може бути як працівник магазину, так і співвласник складу. Як можна побачити, єдина різниця між власником складу та користувачем є відсутність керування користувачами складу серед можливих випадків використання користувача.

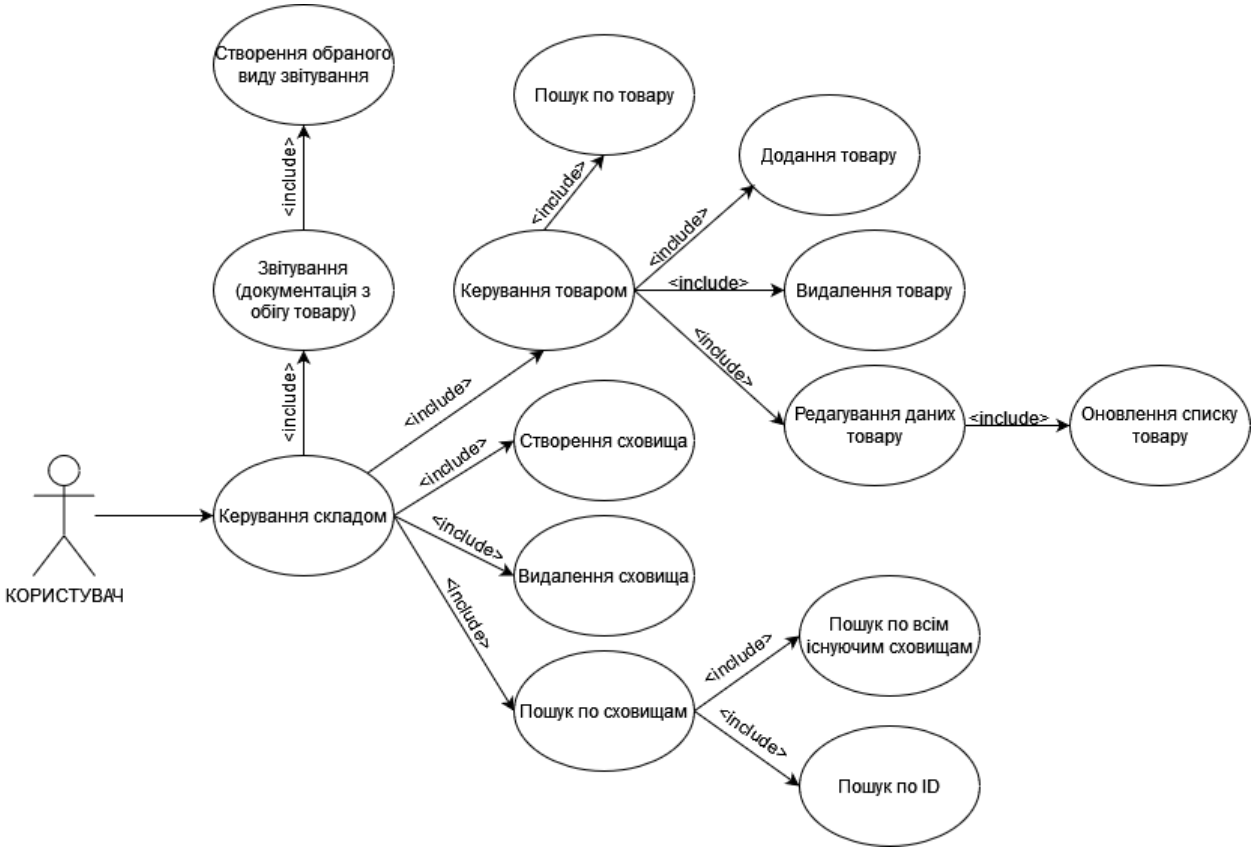


Рис. 10 – Модель прецедентів №2 – Користувач

Останнім актором в системі можна представити звичайного касира, адже навіть у маленьких магазинах наявність касового апарату є необхідною. На діаграмі (рис. 11) можна побачити можливості касиру:

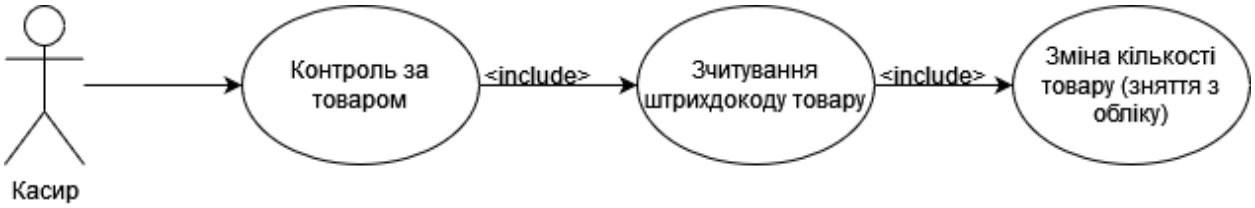


Рис. 11 - Модель прецедентів №3 – Касир

3.3. Нефункціональні вимоги. Бачення з контекстною діаграмою

Для того щоб описати зовнішні властивості системи використаємо контекстну діаграму (рис. 12), яка дозволяє описати систему на рівні «чорного ящику».

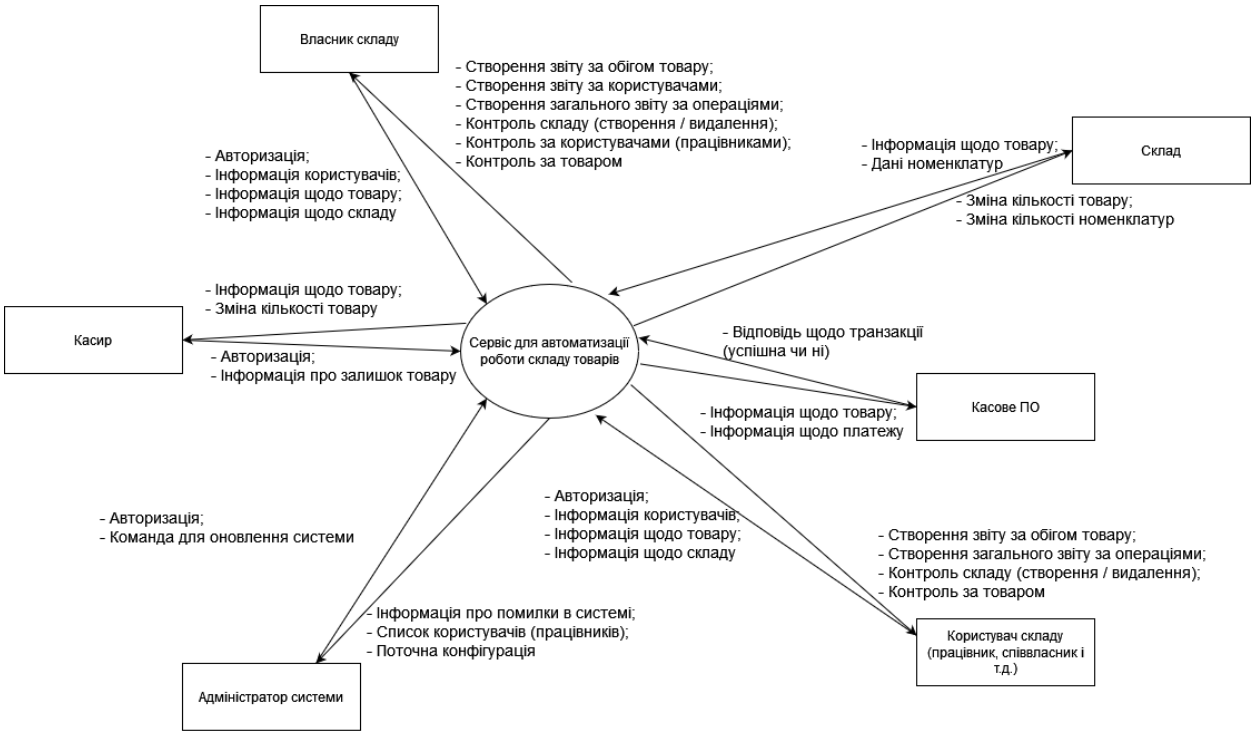


Рис. 12 - Контекстна діаграма

На діаграмі зображено проектувальний об’єкт – сервіс для автоматизації роботи складу. Елементами, які взаємодіють із проектувальним об’єктом є певні групи користувачів та суміжні системи, наприклад: власник складу, адміністратор системи, касове ПЗ, склад та інші, зображені на діаграмі (рис. 12). Також можна побачити вхідні та вихідні потоки даних для кожного елементу системи.

Наприклад, можна побачити що для касира вихідними потоками даних до системи є авторизація (надання особистих логіну та паролю), а також інформація

про товар. Вхідними потоками є інформація про товар (наприклад, після зчитування штрих-коду) та фактичне «віднімання» проданого товару.

3.4. Модель проектування

3.4.1. Проектування діяльності. Діаграма діяльності

На наступних діаграмах можна побачити логіку роботи API на прикладі таких функцій: авторизація (рис. 13), створення віртуального сховища (рис. 14), додавання товару.

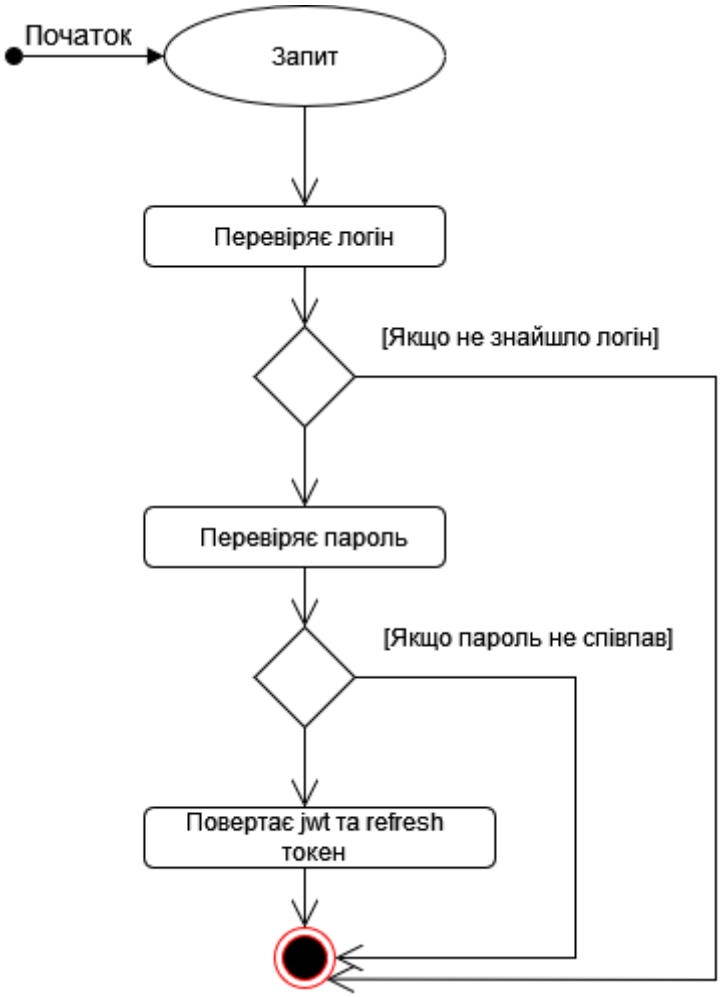


Рис. 13 – Діаграма діяльності для функції «Авторизація»

Як бачимо, логіка роботи API є досить лінійною та на даному етапі розробки не має циклу. Але це не заважає додати його на подальший етапах розробки інтерфейсу користувача. Наприклад, у випадку невдалої авторизації буде розгалуження: або ж спробувати знову, або ж використати функцію «Відновити пароль».

Рис. 14 - Діаграма діяльності для функції «Створення віртуального сховища (складу)»

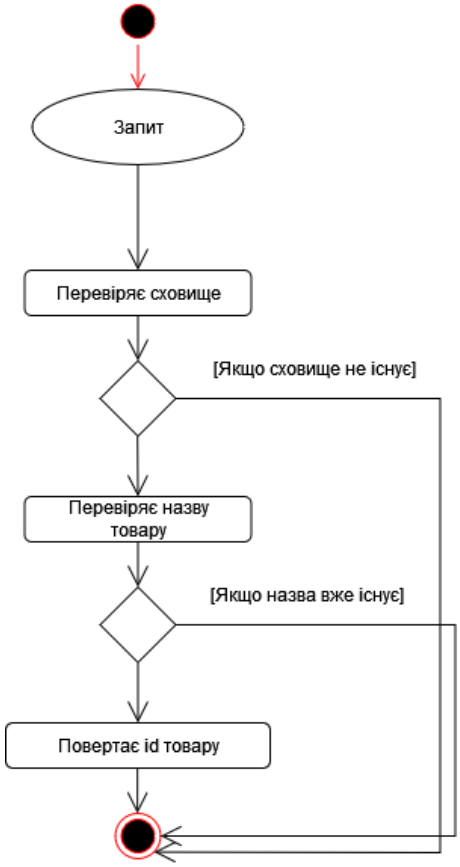


Рис. 15 - Діаграма діяльності для функції «Додавання товару»

3.4.2. Проектування послідовності викликів методів та взаємодії об'єктів. Діаграми послідовностей

Внутрішню логіку роботи програми було зображено на діаграмах послідовностей для наступних функцій: додавання товару (рис. 16), видалення товару (рис. 17), пошук товару (рис. 18).

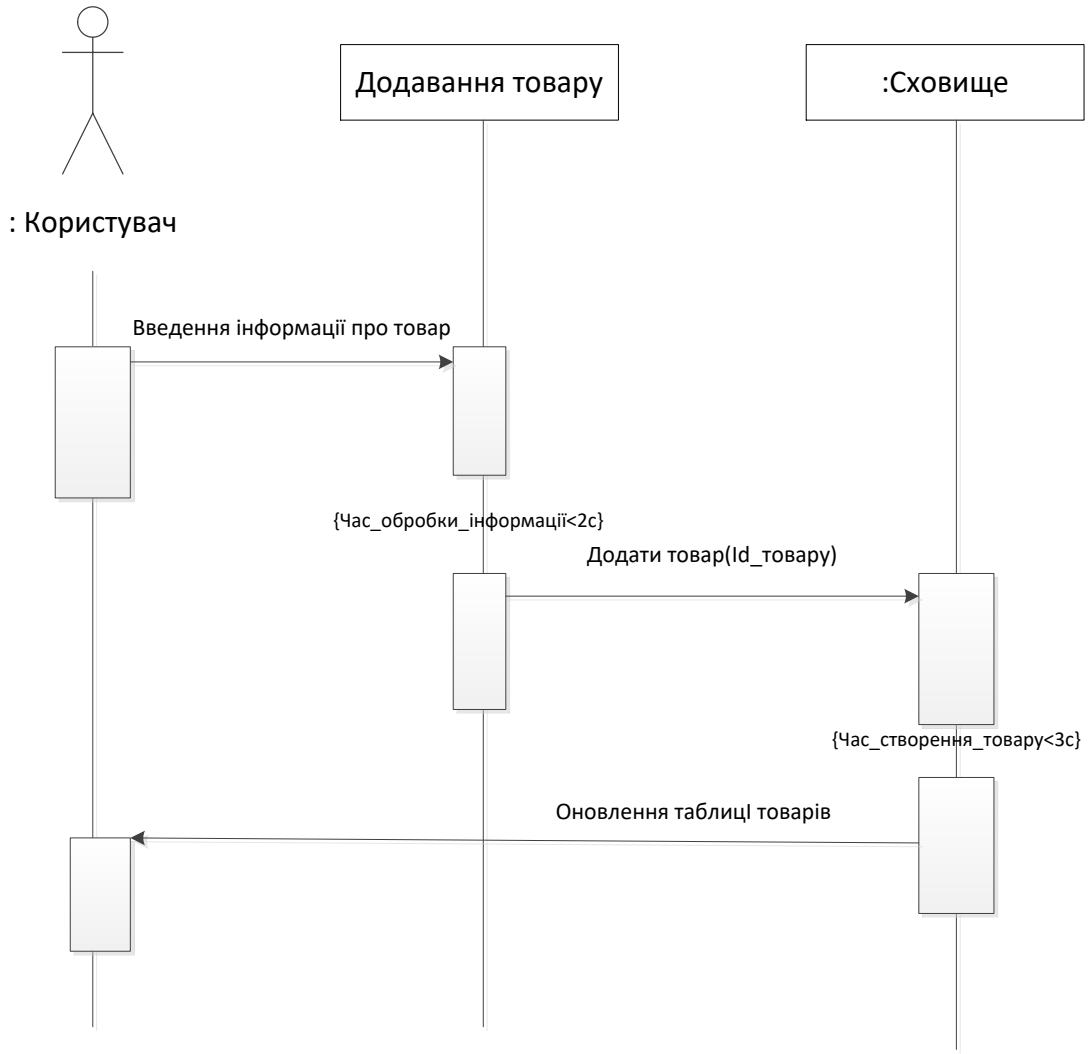


Рис. 16 – Діаграма послідовності для функції "Додавання товару"

Як можна побачити на діаграмах, середній час обробки інформації на етапі передачі потоків даних складає приблизно 2 секунди. Підвищити пропускної здатність веб-сервісу в майбутньому зможуть кращі веб-хостинги, визначити потреби у покращені пропускної здатності допоможе стресове тестування.

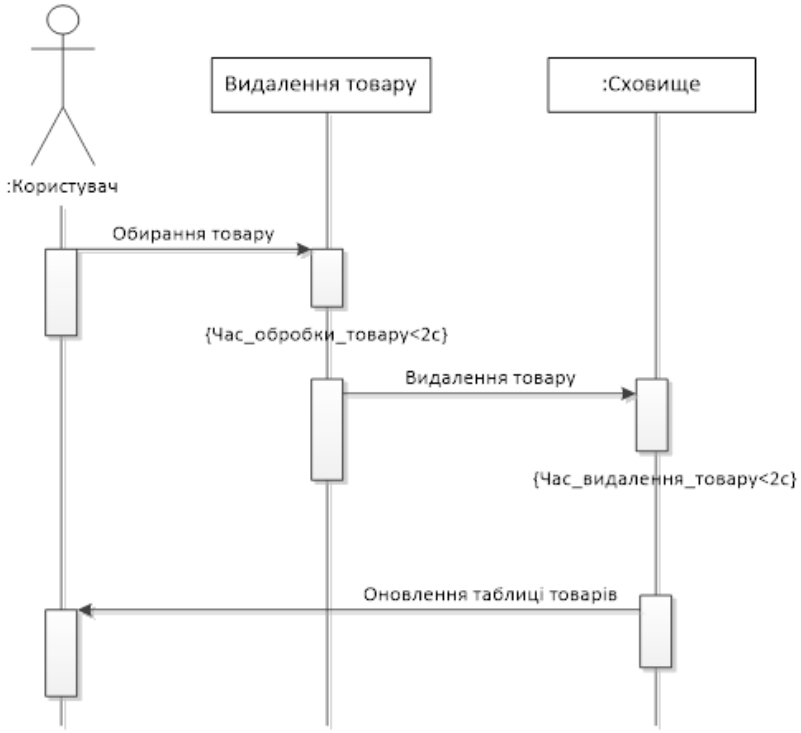


Рис. 17 – Діаграма послідовності для функції "Видалення товару"

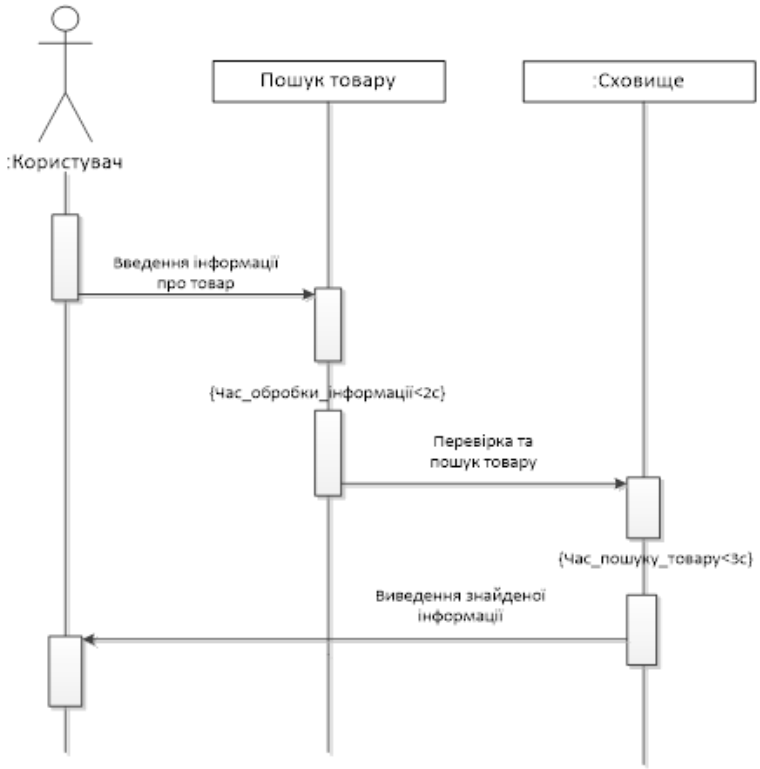


Рис. 18 – Діаграма послідовності для функції "Пошук товару"

3.4.3. Проектування послідовності - діаграми станів

У розділі будуть представлені діаграми станів для декількох функцій, пов’язаних безпосередньо з керуванням товару (рис. 19, 20, 21).

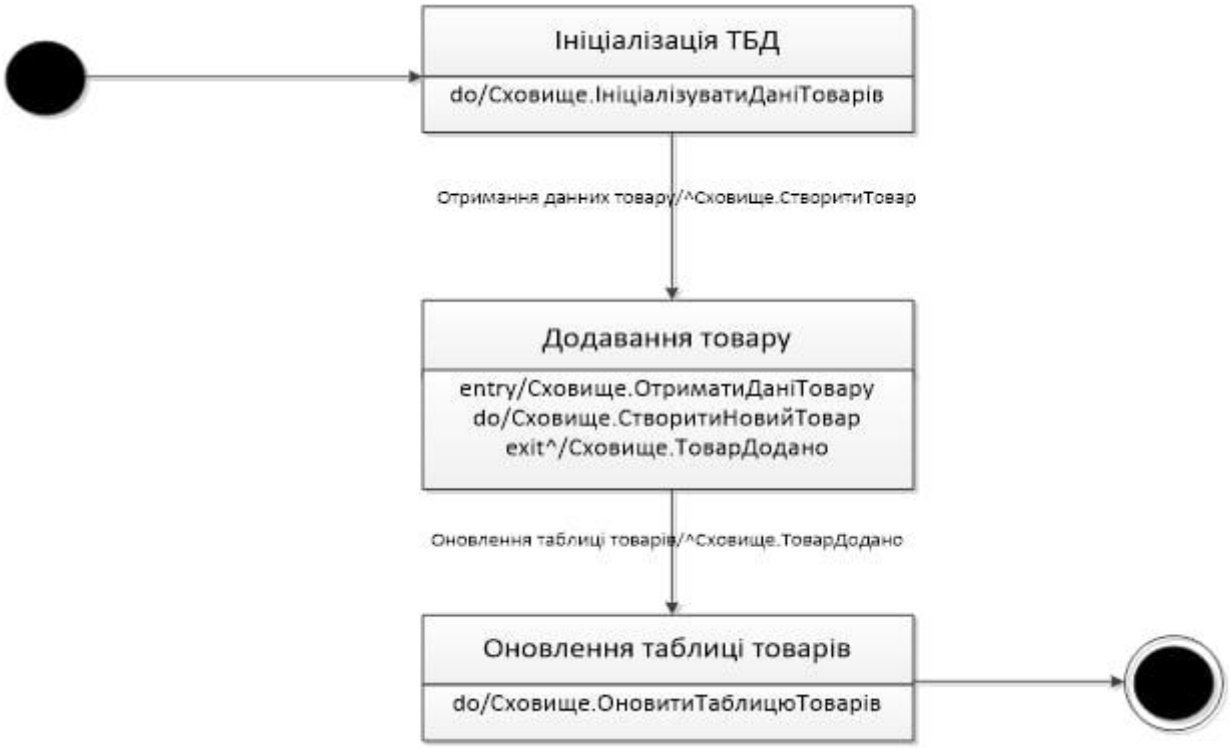


Рис. 19 – Діаграма стану для функції "Додавання товару"

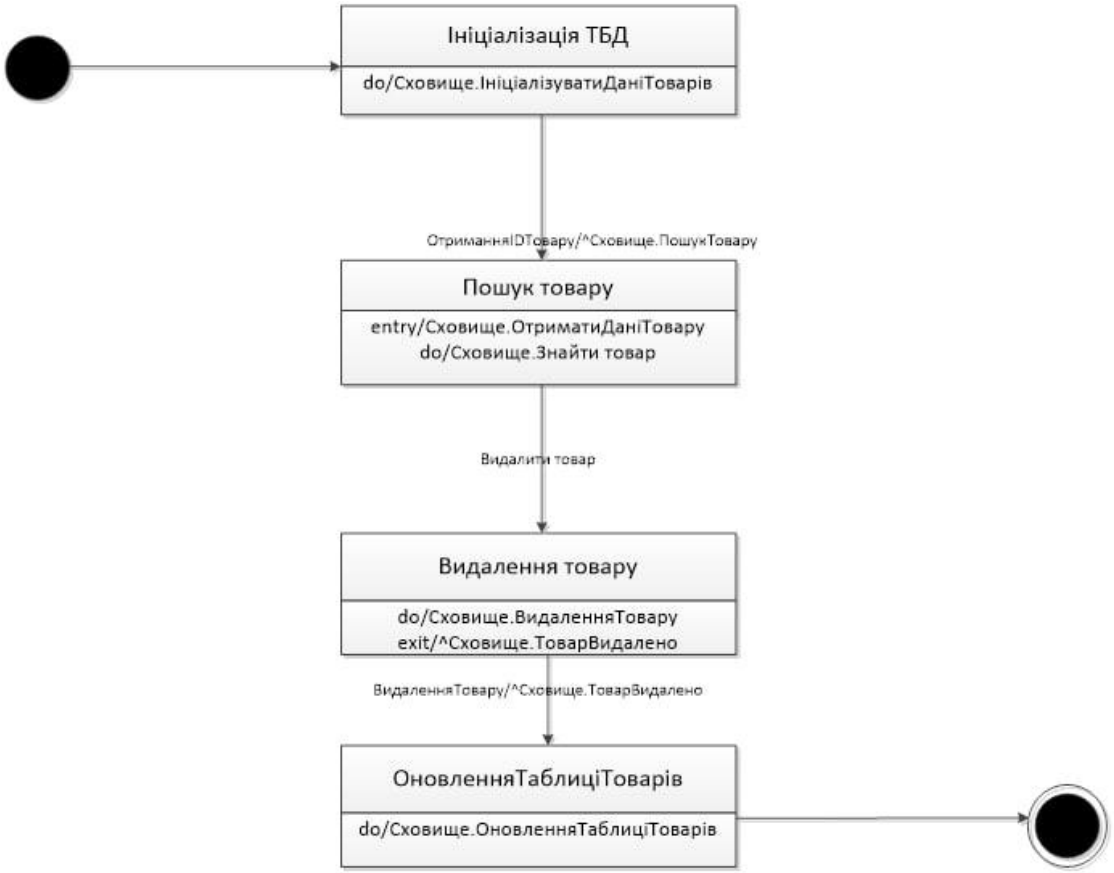


Рис. 20 – Діаграма стану для функції "Видалення товару"

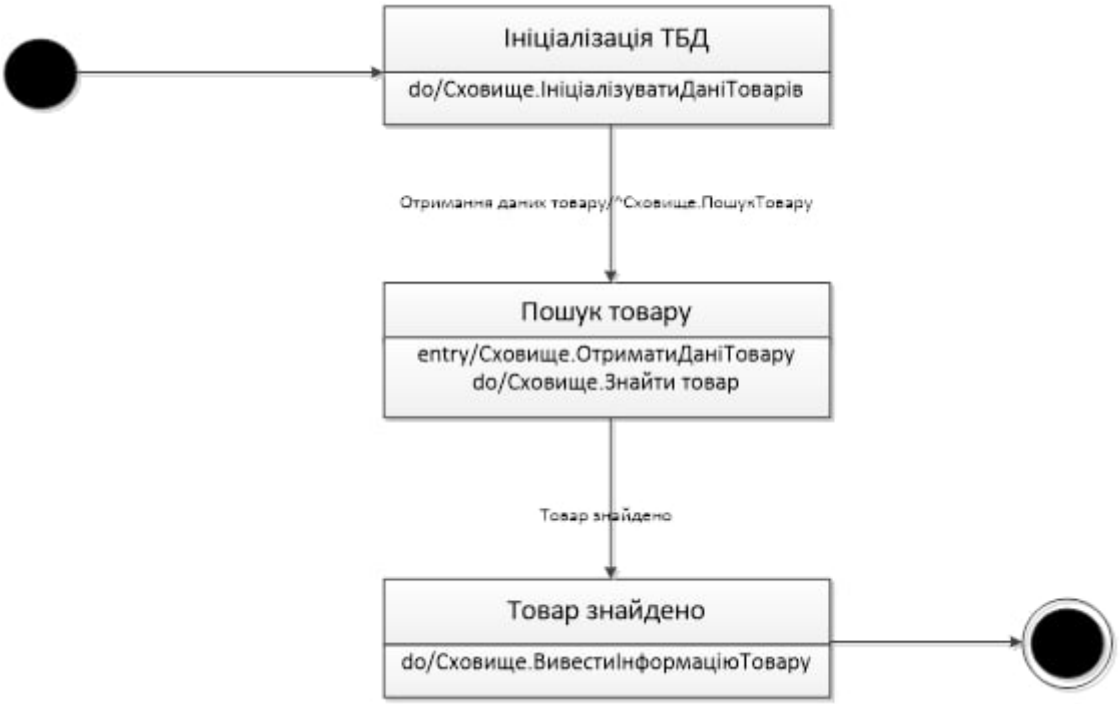


Рис. 21 – Діаграма стану для функції "Пошук товару"

3.4.4. Діаграми комунікацій (кооперацій)

Нижче відображені діаграми комунікацій для декількох функцій, пов’язаних з керуванням товару (рис. 22, 23, 24).

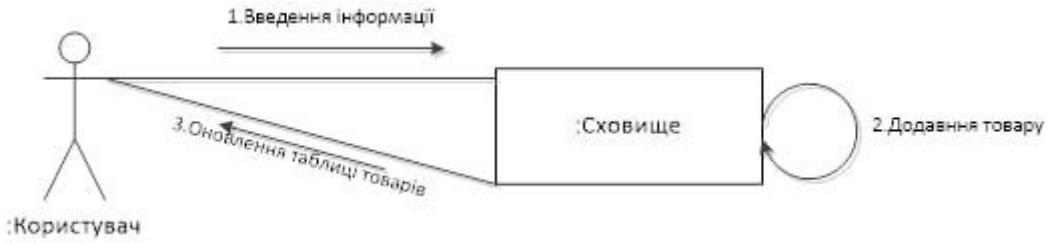


Рис. 22 – Діаграма комунікації для функції "Додавання товару"

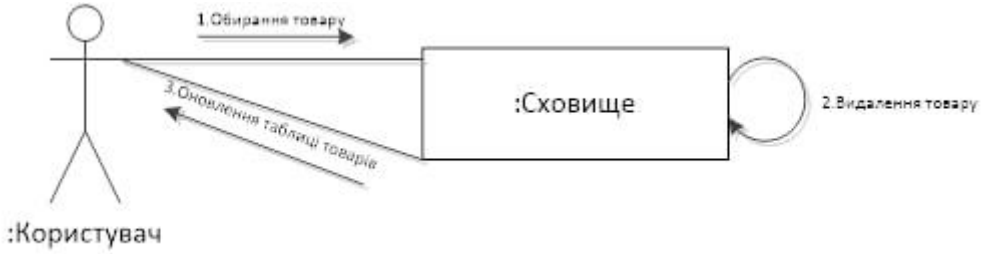


Рис. 23 - Діаграма комунікації для функції "Видалення товару"

Рис. 24 - Діаграма комунікації для функції "Пошук товару"

3.4.5. Діаграма класів

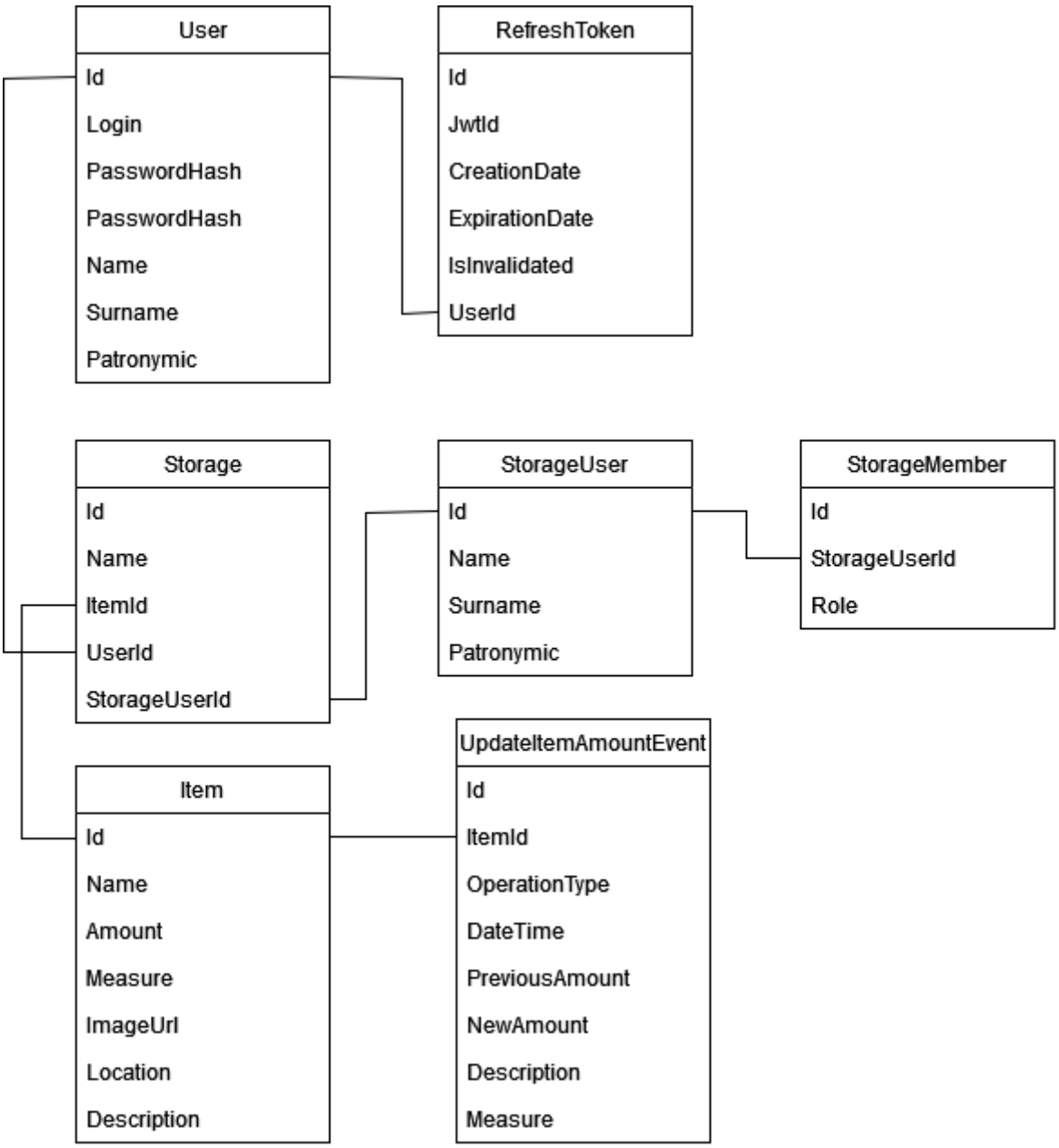


Рис. 25 - Діаграма класів

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ СКЛАДСЬКОГО ОБЛІКУ

4.1. Параметризовані інтерфейси

Параметризований інтерфейс – це загальний або каркасний інтерфейс, який має формальні параметри, які будуть замінені одним або кількома іменами класів або іменами інтерфейсів. Коли він розширюється шляхом заміни конкретних імен класів чи інтерфейсів як фактичних параметрів, створюється інтерфейс, який функціонує як непараметризований інтерфейс. Параметрування — це ще один спосіб узагальнити інтерфейс, щоб його можна було використовувати з широким спектром об’єктів дизайну. Таким чином параметрування інтерфейсу нічим не відрізняється від параметрування модуля чи функції. На скріншоті (рис. 25) можна побачити клас IItemsService який власно і є параметризованим інтерфейсом.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Threading.Tasks;
4   using EasyStorage.BusinessLogic.Dtos.Items;
5
6   namespace EasyStorage.BusinessLogic.Services.Contracts
7   {
        Ссылок: 6
8       public interface IItemsService
9       {
            Ссылок: 2
10          Task<Guid> Add(string storageId, CreateItemDto itemDto);
            Ссылок: 2
11          Task<ItemDto> Get(string storageId,string itemId);
            Ссылок: 2
12          Task<IReadOnlyCollection<ItemDto>> GetAll(string storageId);
            Ссылок: 2
13          Task Delete(string storageId, string itemId);
            Ссылок: 2
14          Task Update(string storageId, string itemId, ItemDto dto);
            Ссылок: 2
15          Task UpdateItemAmount(Guid itemId, CreateUpdateItemAmountEventDto dto);
16      }
17  }
```

Рис. 26 – Клас IItemsService (Параметризований інтерфейс)

4.2. Параметризовані класи

Універсальні класи інкапсулюють операції, які не належать до конкретного типу даних. Універсальні класи найчастіше використовуються для роботи з колекціями, такими як пов'язані списки, хеш-таблиці, стеки, черги, дерева і т. д. Такі операції, як додавання та видалення елементів колекції, по суті виконуються однаково, незалежно від типу даних, що зберігаються. Прикладом такого класу є клас User (рис. 26).

```
namespace EasyStorage.DataAccess.Entities.Identity
{
    Ссылок: 19
    public class User : DocumentBase
    {
        Ссылок: 3
        public string Login { get; set; }
        Ссылок: 2
        public string PasswordHash { get; set; }
        Ссылок: 0
        public string Name { get; set; }
        Ссылок: 0
        public string Surname { get; set; }
        Ссылок: 0
        public string Patronymic { get; set; }
    }
}
```

Рис. 27 - Клас User (Параметризований клас)

Як правило, при створенні універсального класу спочатку визначається конкретний клас, після чого його типи по черзі замінюються параметрами типів доти, доки не буде досягнуто необхідного балансу між ступенем узагальнення та зручністю роботи.

Класи, які є універсальними, тобто конкретні класи, можуть успадковуватися від закритих сконструйованих базових класів. Спадкування від аналогічних відкритих класів або параметрів типу неможливо, оскільки під час виконання

клієнтський код не може надати аргумент типу, необхідний для створення екземпляра базового класу.

Універсальні класи, що успадковуються від відкритих сконструйованих типів, повинні надавати аргументи типу для будь-яких параметрів типу базового класу, які не використовуються разом із спадковим класом.

4.3. Параметризовані методи

Компілятор може визначити параметри типу на основі аргументів методу, які ви передаєте; він не може визначити параметри типу лише з обмеження або значення, що повертається. Тому висновок типу не працює з методами, які не мають параметрів. Виведення типу відбувається під час компіляції до того, як компілятор спробує вирішити перевантажені сигнатури методів. Компілятор застосовує логіку виведення типу до всіх загальних методів, які мають однакову назву. На етапі вирішення перевантаження компілятор включає лише ті загальні методи, для яких виведення типу вдалося. Реалізацію можна побачити на прикладі методу для створення віртуального сховища (рис. 27).

```
Ссылок: 2
public async Task<string> Add(CreateStorageDto storageDto)
{
    Storage storage = _mapper.Map<CreateStorageDto, Storage>(storageDto);

    User user = await _usersRepository.GetById(_jwtTokenReader.UserId);

    if (user == null)
    {
        throw new NotFoundException("UserNotFound");
    }

    StorageMember storageMember = new StorageMember()
    {
        StorageUser = _mapper.Map<User, StorageUser>(user),
        Role = Role.Owner
    };

    storage.StorageMembers.Add(storageMember);

    await _storagesRepository.Insert(storage);

    return storage.Id;
}
```

Рис. 28 - Метод для створення віртуального сховища

4.4. Використання шаблонів проектування

Шаблонний метод — це поведінковий шаблон проектування, який дозволяє визначити скелет алгоритму в базовому класі та дозволити підкласам замінити кроки, не змінюючи загальну структуру алгоритму. Шаблон Template Method досить поширений у фреймворках C#. Розробники часто використовують його для надання користувачам фреймворків простих засобів розширення стандартної функціональності за допомогою успадкування.

Метод шаблону можна розпізнати поведінковими методами, які вже мають поведінку «за замовчуванням», визначену базовим класом. Класи та об'єкти, що беруть участь у цьому шаблоні, включають абстрактний клас та конкретний клас.

Абстрактний клас визначає абстрактні примітивні операції, які визначають конкретні підкласи для реалізації кроків алгоритму, реалізує метод шаблону, що

визначає скелет алгоритму. Метод шаблону викликає примітивні операції, а також операції, визначені в абстрактному класі або операції інших об’єктів.

Окрім цього в шаблоні приймає участь конкретний клас, який реалізує примітивні операції для виконання специфічних для підкласів кроків алгоритму. Реалізацію можна побачити на шаблоні проектування Builder (рис. 28) та шаблон Repository (рис. 29).

```
Ссылок: 0
public void ConfigureServices(IServiceCollection services)
{
    services.AddDatabaseContext(_configuration)
        .AddApplicationServices(_configuration)
        .AddPersistenceServices()
        .AddMapper()
        .AddFriendlyJwt()
        .AddFluentValidation();
```

Рис. 29 – Шаблон проектування Builder

```
namespace EasyStorage.DataAccess.Repositories
{
    Ссылок: 2
    public class UsersRepository : MongoRepositoryBase<User>, IUsersRepository
    {
        Ссылок: 0
        public UsersRepository(IMongoDbContext mongoDbContext) : base(mongoDbContext.Users)
        {

        }

        Ссылок: 2
        public async Task<User> GetByLogin(string login)
        {
            return await Collection.AsQueryable()
                .Where(user => user.Login == login)
                .SingleOrDefaultAsync();
        }

        Ссылок: 2
        public async Task<bool> IsLoginExist(string login)
        {
            return await Collection.AsQueryable()
                .AnyAsync(user => user.Login == login);
        }
    }
}
```

Рис. 30 – Шаблон проектування Repository

4.5. Використання JSON серіалізації

Простір імен System.Text.Json містить усі точки входу та основні типи. Простір імен System.Text.Json.Serialization містить атрибути та API для розширених сценаріїв та налаштувань, характерних для серіалізації та десеріалізації.

Клас StoregesController.cs (рис. 29) працює наступним чином, що при поверненні відповіді клієнту об’єкт автоматично серіалізується.

```
[HttpGet]
Ссылок: 0
public async Task<IActionResult> GetAllStorage()
{
    IReadOnlyCollection<StorageDto> dtos =  await _storagesService.GetAll();

    return Ok(dtos);
}
```

Рис. 31 – Клас StoregesController.cs

4.6. Використання обробки виключень

На фрагменті коду класу IdentityService.cs (рис. представлена обробка виключення якщо користувач намагається зареєструвати акаунт з існуючим логіном.

```
Ссылок: 2
public async Task<AuthenticationResult> Register(RegisterUserDto registerUserDto)
{
    if (await _usersRepository.IsLoginExist(registerUserDto.Login))
    {
        throw new ValidationException("RegisterLoginBusy");
    }
```

Рис. 32 – Клас IdentityService.cs (Обробка виключення)

4.7. Інверсія залежностей. Впровадження залежностей

Принцип інверсії залежностей DIP стверджує, що модулі високого рівня не повинні залежати від модулів низького рівня; обидва мають залежати від абстракцій. Абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій. Під час написання програмного забезпечення надзвичайно поширено реалізувати його так, що кожен модуль або метод конкретно посилається на своїх співробітників, які роблять те ж саме. Цей тип програмування, як правило, не має достатніх шарів абстракції, і це призводить до дуже тісно пов'язаної системи, оскільки кожен модуль безпосередньо посилається на модулі нижнього рівня.

.NET підтримує шаблон проектування програмного забезпечення для впровадження залежностей (DI), який є технікою досягнення інверсії керування (IoC – Inversion of Control) між класами та їх залежностями. Ін'єкція залежностей у .NET є вбудованою частиною фреймворка разом із конфігурацією, веденням журналу та шаблоном параметрів. Залежність – це об'єкт, від якого залежить інший об'єкт.

Розглянемо фрагмент коду, на якому відображено підключення залежностей (рис. 32).

```
ссылка: 1
public static IServiceCollection AddApplicationServices(this IServiceCollection services, IConfiguration configuration)
{
    services.AddScoped<IIdentityService, IdentityService>();
    services.AddScoped<IStoragesService, StoragesService>();
    services.AddScoped<IItemsService, ItemsService>();
    services.AddScoped<IReportsService, ReportsService>();
    AddCustomSettings<IdentityOptions>(services, configuration);

    return services;
}
```

Рис. 33 - Підключення залежностей (dependency injection)

4.8. Використання бібліотек

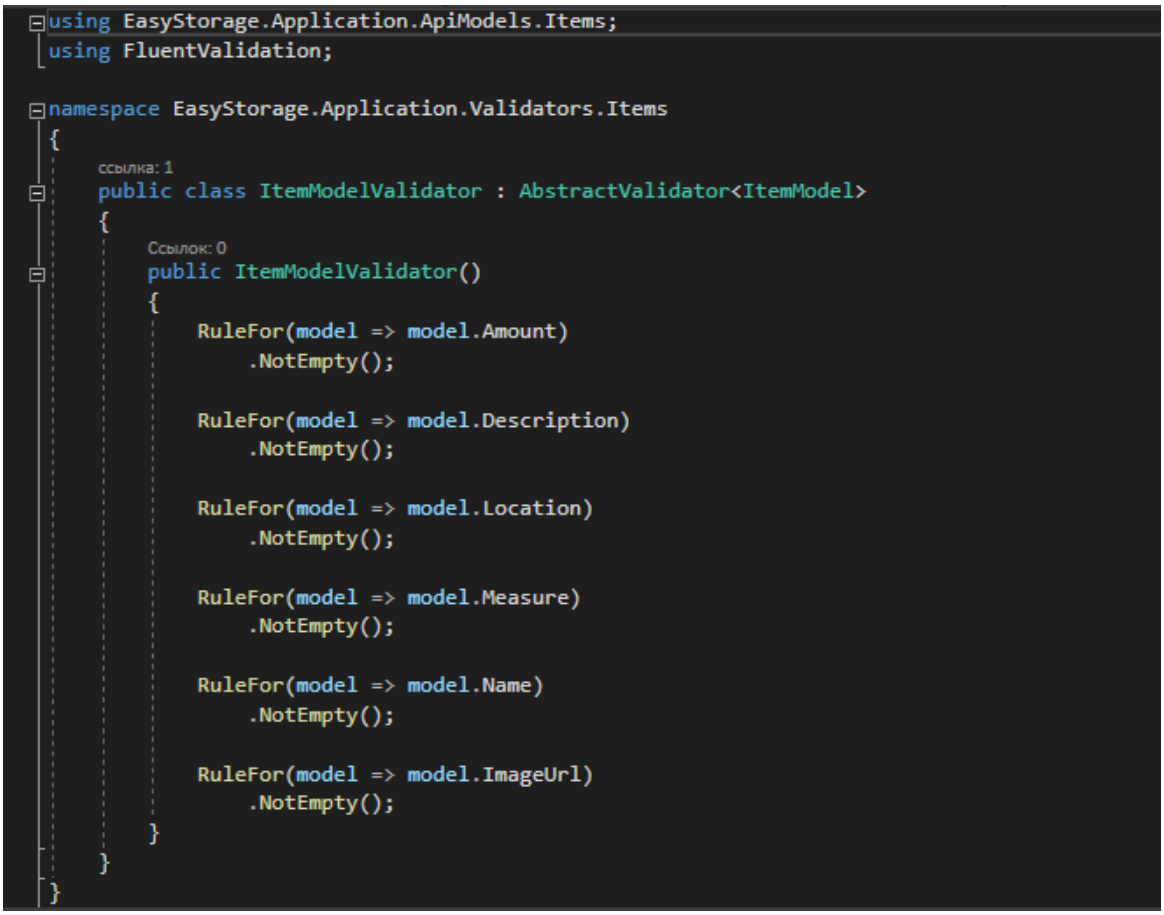
Однією з використаних бібліотек є AutoMapper. AutoMapper – це бібліотека, створена для вирішення проблеми позбавлення скорочення коду, який поєднував один об’єкт з іншим. AutoMapper – це картограф об'єкт-об'єкт. Відображення об’єкт-об’єкт працює шляхом перетворення вхідного об’єкта одного типу у вихідний об’єкт іншого типу. Що робить AutoMapper цікавим, так це тим, що він надає деякі цікаві конвенції, які допомагають спростити роботу, пов’язану з тим, як зіставити тип A з типом B. Поки тип B відповідає встановленій конвенції AutoMapper, для відображення двох типів потрібна майже нульова конфігурація.

Метод забезпечує просту конфігурацію типів, а також просте тестування відображень. На фрагменті коду класу ItemsProfile.cs (рис. 33) можна побачити як було підключено AutoMapper.

```
using AutoMapper;
using EasyStorage.Application.ApiModels.Items;
using EasyStorage.BusinessLogic.Dtos.Items;

namespace EasyStorage.Application.Mapping
{
    ссылка: 1
    public class ItemsProfile : Profile
    {
        Ссылок: 0
        public ItemsProfile()
        {
            CreateMap<ItemModel, CreateItemDto>();

            CreateMap<ItemModel, ItemDto>();

            CreateMap<CreateUpdateItemAmountEventModel, CreateUpdateItemAmountEventDto>();

            CreateMap<OperationTypeModel, OperationTypeDto>();
        }
    }
}
```

Рис. 34 – Правило AutoMapper

Також використовується бібліотека для валідації FluentValidator, на скріншоті нижче можна побачити код підключення (рис. 34):



```
using EasyStorage.Application.ApiModels.Items;
using FluentValidation;

namespace EasyStorage.Application.Validators.Items
{
    ссылка: 1
    public class ItemModelValidator : AbstractValidator<ItemModel>
    {
        Ссылок: 0
        public ItemModelValidator()
        {
            RuleFor(model => model.Amount)
                .NotEmpty();

            RuleFor(model => model.Description)
                .NotEmpty();

            RuleFor(model => model.Location)
                .NotEmpty();

            RuleFor(model => model.Measure)
                .NotEmpty();

            RuleFor(model => model.Name)
                .NotEmpty();

            RuleFor(model => model.ImageUrl)
                .NotEmpty();
        }
    }
}
```

Рис. 35 – Правила валідації

4.9. Підключення до бази даних

```
namespace EasyStorage.DataAccess.DataContext
{
    Ссылок: 2
    public class MongoDbContext : IMongoDbContext
    {
        private readonly MongoClient _mongoClient;
        private readonly IMongoDatabase _database;

        Ссылок: 3
        public IMongoCollection<User> Users { get; }
        Ссылок: 3
        public IMongoCollection<RefreshToken> RefreshTokens { get; }
        Ссылок: 3
        public IMongoCollection<Storage> Storages { get; }
        Ссылок: 2
        public IMongoCollection<Item> Items { get; }
        Ссылок: 3
        public IMongoCollection<UpdateItemAmountEvent> UpdateItemAmountEvents { get; }

        Ссылок: 0
        public MongoDbContext(IOptions<MongoDbOptions> mongoDbOptions)
        {
            _mongoClient = new MongoClient(mongoDbOptions.Value.ConnectionUrl);
            _database = _mongoClient.GetDatabase(mongoDbOptions.Value.DatabaseName);
            Users = _database.GetCollection<User>(mongoDbOptions.Value.Users);
            RefreshTokens = _database.GetCollection<RefreshToken>(mongoDbOptions.Value.RefreshTokens);
            Storages = _database.GetCollection<Storage>(mongoDbOptions.Value.Storages);
            Items = _database.GetCollection<Item>(mongoDbOptions.Value.Items);
            UpdateItemAmountEvents = _database.GetCollection<UpdateItemAmountEvent>(mongoDbOptions.Value.UpdateItemAmountE
        }
```

Рис. 36 - Клас MongoDbContext, підключення до БД

```
},
"MongoDbOptions": {
  "ConnectionUrl": "mongodb://localhost:27017",
  "DatabaseName": "EasyStorageDb",
  "Users": "Users",
  "RefreshTokens": "RefreshTokens",
  "Storages": "Storages",
  "Items": "Items",
  "UpdateItemAmountEvents": "UpdateItemAmountEvents"
},
```

Рис. 37 - Параметри підключення БД

4.10. Використання DTO

DTO (Data Transfer objects) — це контейнер даних для переміщення даних між шарами. Їх також називають об’єктами передачі. DTO використовується лише

для передачі даних і не містить жодної бізнес-логіки. У них є лише прості сетери та геттери.

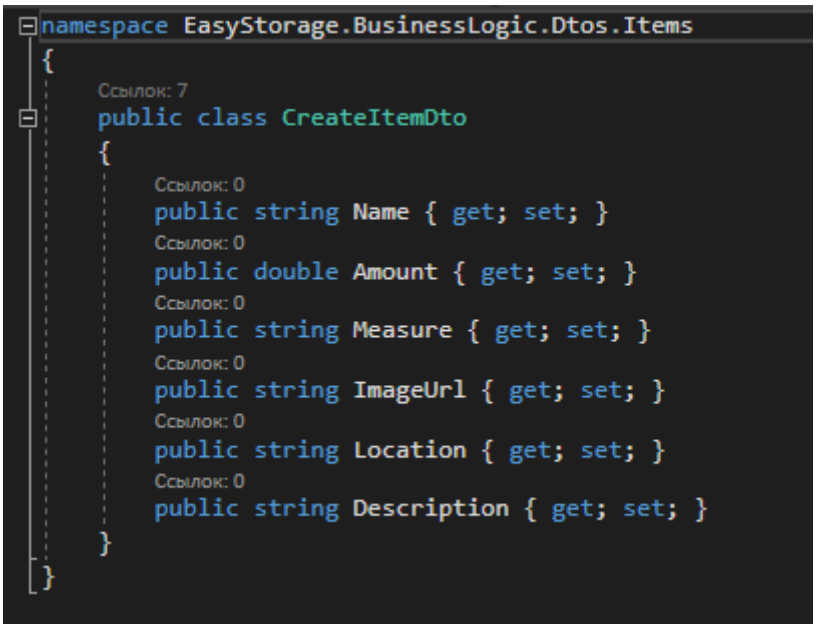


Рис. 38 - Використання DTO в проекті

4.11. Використання JWT-токену

JWT – це токен, який може генерувати лише сервер і може містити корисне навантаження даних. Корисне навантаження JWT може містити такі речі, як UserID або Email. Дані, які передаються за допомогою JWT між сторонами, мають цифровий підпис, тому їх можна легко перевірити та довіряти їм.

Першим кроком є налаштування автентифікації на основі JWT у нашому проекті. Для цього нам потрібно зареєструвати схему автентифікації JWT. Другим кроком є створення, а третім налаштування роботи веб-токену.

```
private GeneratedTokenInfo CreateAccessToken(User user)
{
    TimeSpan lifeTime = TimeSpan.FromMinutes(_identityOptions.JwtTokenExpiresMinutes);
    string secret = _identityOptions.Secret;

    return new JwtTokenBuilder(lifeTime, secret)
        .WithAudience(_identityOptions.Audience)
        .WithIssuer(_identityOptions.Issuer)
        .WithUserIdPayloadData(user.Id.ToString())
        .WithPayloadData("Login", user.Login)
        .Build();
}
```

Рис. 39 - Створення JWT-токену

```
"IdentityOptions": {
  "Secret": "huiQcYqrYXegWl1H2RoHL9JavuIyZLWq",
  "Issuer": "http://localhost:5000",
  "Audience": "http://localhost:5000",
  "JwtTokenExpiresMinutes": 30,
  "RefreshTokenExpiresMonth": 3
},
```

Рис. 40 - Налаштування JWT-токену

4.12. Використання Refresh Token

Якщо ми використовуємо маркер доступу протягом тривалого часу, є ймовірність, що хакер може вкрасти наш токен і використати його неправильно. Тому використовувати маркер доступу протягом тривалого періоду не дуже безпечно.

Токени оновлення (Refresh tokens) — це тип маркерів, які можна використовувати для отримання нових маркерів доступу. Коли термін дії маркерів доступу закінчується, ми можемо використовувати маркери оновлення, щоб отримати новий маркер доступу від контролера аутентифікації. Термін життя маркера оновлення зазвичай набагато довший в порівнянні з терміном життя маркера доступу.

Було встановлено короткий термін дії токена доступу. Таким чином, навіть маркер доступу, який може використати потенційний хакер, отримує доступ лише на короткий період. Розроблений API видає маркер оновлення разом із маркером доступу у відповідь на запит на вхід. Щоразу, коли термін дії маркера доступу закінчується, ми можемо отримати новий маркер доступу за допомогою маркера оновлення.

Щоразу, коли користувач входить у програму, використовуючи дійсні облікові дані, ми оновлюватимемо маркер оновлення та час закінчення терміну дії маркера в таблиці користувачів у базі даних Identity. Після закінчення терміну дії маркера доступу, якщо користувач знову спробує отримати захищений ресурс із програми, він видасть 401 неавторизовану помилку.

```
private RefreshToken CreateRefreshToken(string userId, string accessToken)
{
    return new RefreshToken()
    {
        JwtId = Guid.Parse(accessToken),
        CreationDate = DateTime.UtcNow,
        ExpirationDate = DateTime.UtcNow.AddMonths(_identityOptions.RefreshTokenExpiresMonth),
        IsInvalidated = false,
        UserId = userId
    };
}
```

Рис. 41 - Створення Refresh token

ВИСНОВКИ

В процесі виконання роботи було досліджено предметну галузь. Було виявлено основні потреби користувача та створено модель предметної галузі. Також виявлені основні проблеми в області складського обліку, такі як відсутність української мови в програмах-аналогах або дорога вартість ліцензійного програмного забезпечення.

Проаналізовано аналоги на ринку програмних продуктів з автоматизації складського обліку.

Спираючись на ці дані було виявлено необхідні для користувачів функціональності та розроблено модель прецедентів, на якій зображені різні варіанти використання програмного забезпечення.

Розроблена контекстну діаграму для опису системи, а також діаграми діяльності та діаграми послідовностей. Для зображення внутрішньої структури було створено діаграму класів.

Було створено API-сервер та базу даних для обліку товарів на складі, використовуючи різноманітні технології. У процесі розробки було використано сучасні бібліотеки для .NET такі як FluentValidator та AutoMapper. Також використано шаблони проектування Builder, Repository.

В процесі роботи було використано параметризовані інтерфейси, класи та методи, а також методи серіалізації JSON.

У розробленому програмному забезпечені діє підвищений захист даних користувачів за допомогою використання таких технологій як JWT-токени та Refresh Tokens.

ПЕРЕЛІК ПОСИЛАНЬ

1. Чорноус К. В. Магістерська дисертація на тему: «Система управління складським обліком». – Режим доступу: https://ela.kpi.ua/bitstream/123456789/25551/1/Chornous_magistr.pdf
2. Тарасов А. В. Бакалаврська кваліфікаційна робота на тему: «Автоматизація роботи складу підприємства». – Режим доступу: http://eprints.library.odeku.edu.ua/id/eprint/5953/1/Tarasov_Avtomatuzacia_robotu_sklady.pdf
3. A4 Company [Електронний ресурс]: BAS Управління торгівлею – Режим доступу: https://a4.com.ua/ru/bas-upravlenie-torgovlej/
4. Документація Microsoft [Електронний ресурс]: Boolean logical operators (C# reference) – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators
5. Документація Microsoft [Електронний ресурс]: Using domain analysis to model microservices – Режим доступу: https://docs.microsoft.com/uk-ua/azure/architecture/microservices/model/domain-analysis
6. Project VeriPage [Електронний ресурс]: SystemVerilog Interfaces – Режим доступу: https://www.project-veripage.com/interface_4.php
7. Документація Microsoft [Електронний ресурс]: Generic Classes (C# Programming Guide) – Режим доступу: https://docs.microsoft.com/en-gb/dotnet/csharp/programming-guide/generics/generic-classes
8. Документація Microsoft [Електронний ресурс]: Generic Methods (C# Programming Guide) – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/generic-methods
9. Refactoring.Guru [Електронний ресурс]: Template Method in C# – Режим доступу: https://refactoring.guru/design-patterns/template-method/csharp/example
10. Dofactory [Електронний ресурс]: C# Template Method – Режим доступу: https://www.dofactory.com/net/template-method-design-pattern

11. DevIQ [Електронний ресурс]: Dependency Inversion Principle – Режим доступу: https://deviq.com/principles/dependency-inversion-principle
12. Документація Microsoft [Електронний ресурс]: Dependency injection in .NET – Режим доступу: https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection
13. AutoMapper Docs [Електронний ресурс]: Getting Started Guide – Режим доступу: https://docs.automapper.org/en/latest/Getting-started.html
14. Code Project [Електронний ресурс]: Data Transfer Object Design Pattern in C# – Режим доступу: https://cutt.ly/CJMtJkC
15. Medium [Електронний ресурс]: Moshe Binieli «JWT Authentication using C#» – Режим доступу: https://cutt.ly/BJMyhW5
16. C# Corner [Електронний ресурс]: Jignesh Trivedi «JWT Authentication In ASP.NET Core» – Режим доступу: https://www.c-sharpcorner.com/article/jwt-json-web-token-authentication-in-asp-net-core/
17. C# Corner [Електронний ресурс]: Sarathlal Saseendran «JWT Authentication With Refresh Tokens In .NET 6.0» – Режим доступу: https://cutt.ly/cJMy2F7

ДОДАТКИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

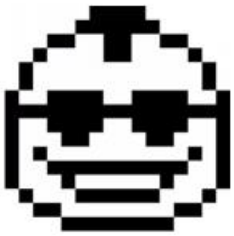
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Створення сервісу для автоматизації роботи складу товарів мовою C#

Виконавець: студент 4 курсу,
групи ПД-43
Воронецький Микита Андрійович
Керівник роботи: ст. викл. Коба
Андрій Борисович

Київ 2022

Аналоги



Аналіз аналогів

	Kataloger	Тріумф V 2.0	BAS Управління торгівлею	Subtotal	WeclApp
Потреби бізнесу	+	+	+	+	+
Інтеграція з іншими ПЗ	-	-	+	+	-
Інтеграція з касою	-	-	+	+	-
Створення звітності	-	+	+	+	+
Безмежні сховища	-	-	-	-	-
Українська мова	-	-	+	-	-

Мета, об’єкт та предмет дослідження

Мета роботи – автоматизація обліку товарів на складі та створення звітності за допомогою API серверу

Об’єкт дослідження – здійснення складського обліку за допомогою сервісів для автоматизації

Предмет дослідження – веб-сервіс для керування складом на основі створеного API-серверу та БД для складського обліку

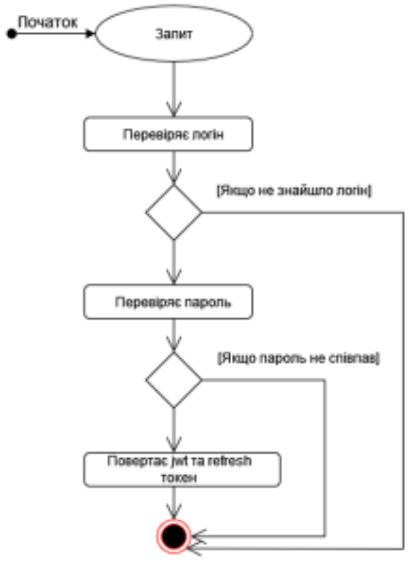
Технічне завдання

1. Реалізувати функціональність для створення віртуального складу (кількість необмежена)
2. Реалізувати функціональність для додавання нових користувачів складу (співвласник або працівник)
3. Реалізувати контроль за обігом товару та функціональність для його додавання, видалення, редагування та пошуку
4. Реалізувати функціональність для створення звітностей різного рівня (за обігом товару, за складами, за користувачами, за внутрішніми процесами)
5. Забезпечити створення звітностей у форматі .csv задля економії обсягу пам’яті пристрою

Засоби реалізації

Сценарії використання

Проектування діяльності

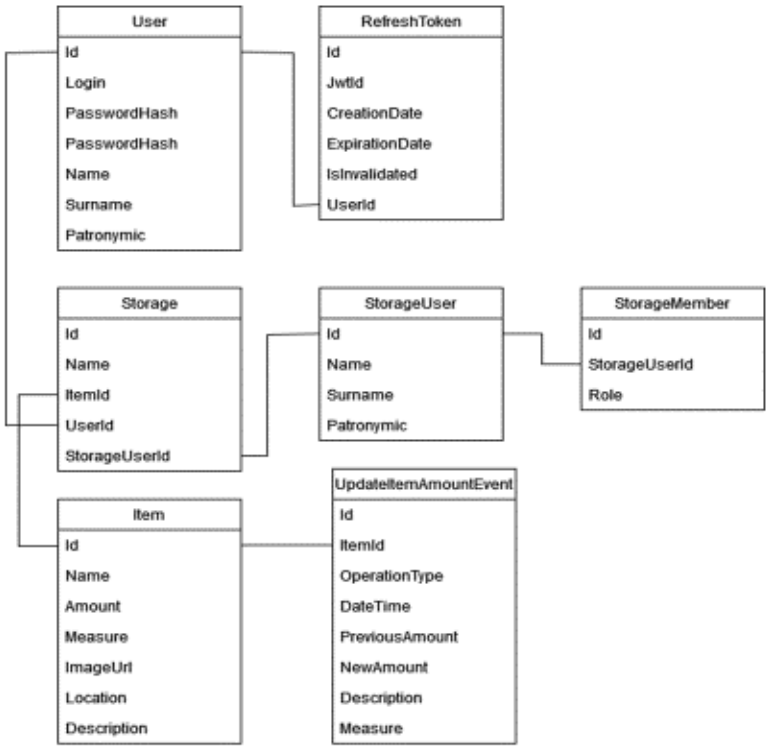


Авторизація

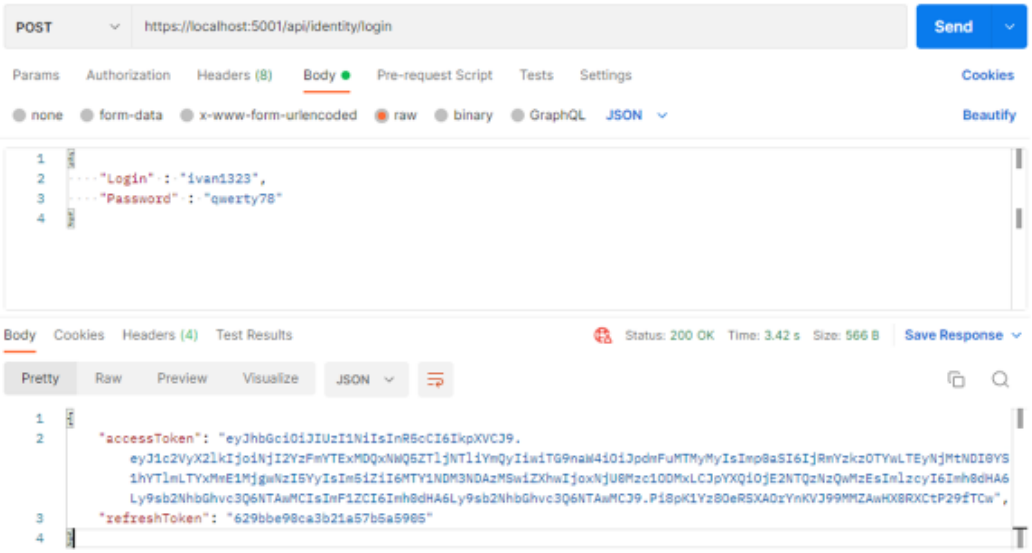
Додавання товару

Створення складу

Діаграма класів

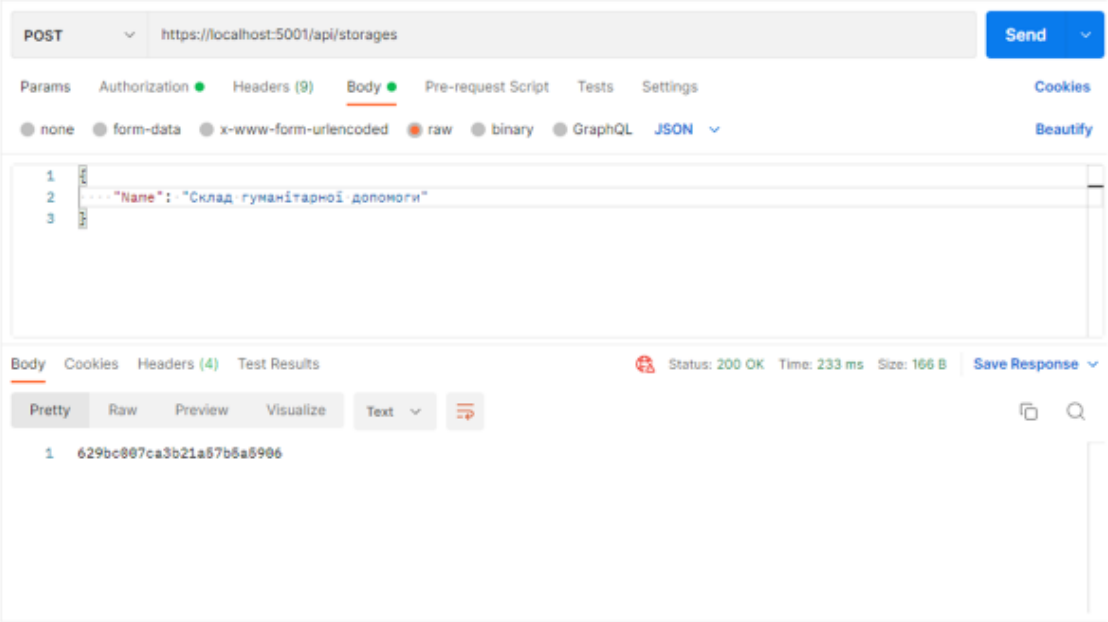


Приклад запиту API



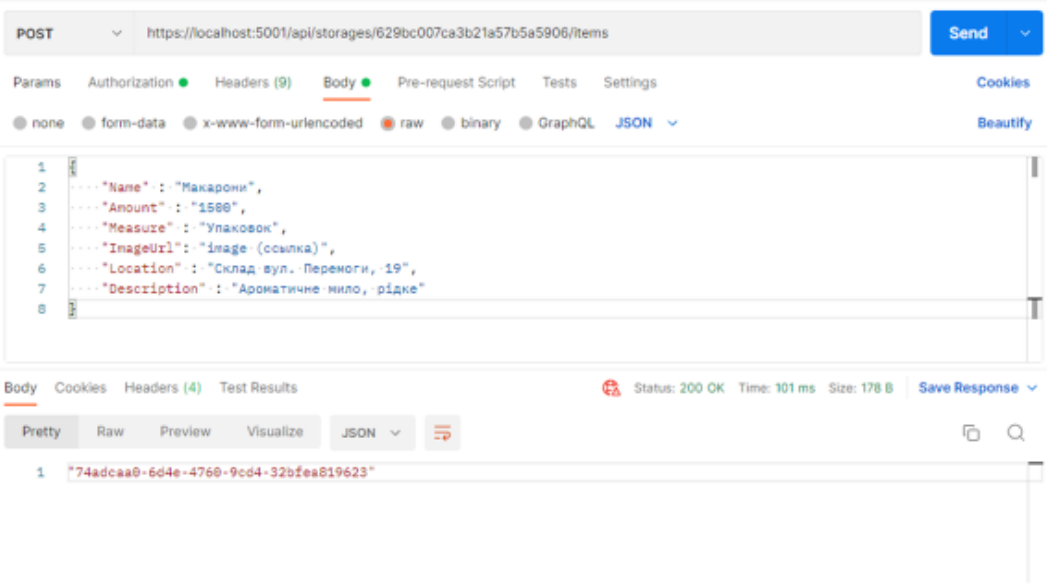
Запит на авторизацію

Приклад запиту API



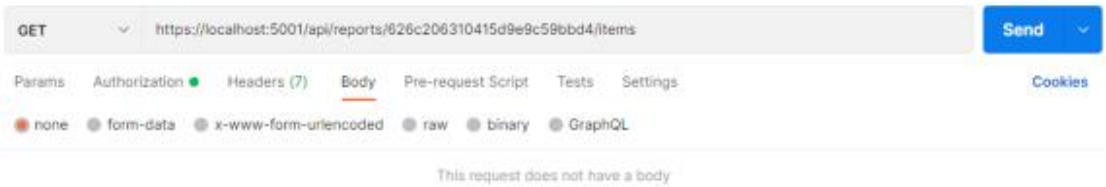
Запит на створення складу

Приклад запиту API



Запит на додавання товару

Приклад запиту API



Запит на створення звіту за залишком товаром

Апробація результатів дослідження

- Воронецький М. А. «АВТОМАТИЗАЦІЯ СКЛАДУ». Робота пройшла апробацію на Науково-технічній конференції «Застосування програмного забезпечення в ІКТ», м. Київ, ДУТ, 20 квітня 2022 року.

Висновки

1. Проаналізовано сферу автоматизації складського обліку, виявлено основні потреби бізнесу.
2. На основі аналізу аналогів було визначено основні функціональні та нефункціональні вимоги. Також доведена актуальність шляхом порівняння.
3. Розроблено API-сервер та БД для автоматизації обліку товарів на складі

Дякую за увагу!