

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ДЛЯ
СКЛАДСЬКОГО ОБЛІКУ
SOFTWARE IMPLEMENTATION OF AN AUTOMATED SYSTEM FOR
WAREHOUSE ACCOUNTING

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Івченко А.С.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Івченку Артуру Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Програмна реалізація автоматизованої системи для складського обліку

Software Implementation of an Automated System for Tracking the Warehouse Inventory

керівник проекту (роботи) Гофман Євгеній Олександрович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проекту (роботи) травень 2021 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Гофман Є.О., доцент		
Нормоконтролер	Липовець М.В., асистент		

7. Дата видачі завдання “ 1 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 2
4	Розробка програми	7-9 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	10 тиждень	Розділ 4
8	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
9	Захист роботи	12 тиждень	

Студент(ка)

(підпис)

Івченко А.С.

(прізвище та ініціали)

Керівник проекту (роботи)

(підпис)

Гофман Є.О.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної дипломної роботи бакалавра: 89 с., 44 рис., 4 табл., 4 дод., 21 джерело.

СКЛАДСЬКИЙ ОБЛІК, МОНІТОРИНГ, АВТОМАТИЗАЦІЯ, ПРОДАЖІ, РОЗРОБКА, АНАЛІЗ ВИМОГ, ПРОЕКТУВАННЯ, ВЕБ-ЗАСТОСУНОК.

Об'єкт дослідження – процес розробки та підтримки автоматизованої системи для складського обліку.

Предмет дослідження – методи розробки та підтримки автоматизованої системи для складського обліку.

Метою роботи є програмна реалізація автоматизованої системи для складського обліку.

Матеріали, методи та технічні засоби: мова програмування PHP, система керування вмістом Joomla!, система керування базою даних MySQL.

Результати. Розроблено автоматизованої системи для складського обліку, що представляє собою клієнт-серверний веб-застосунок, який реалізує механізми ведення складського обліку.

Практична цінність роботи полягає в розробці клієнт-серверного кроссплатформного веб-застосунку ведення складського обліку: ведення статистики, формування та надання звітності та відслідковування грошових транзакцій за товари.

Висновки. Розроблено програмне забезпечення, що реалізує автоматизовану систему для складського обліку.

Галузь використання – корпоративні співробітники структурних підрозділів компанії, покупці.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 89 pages, 44 figures, 4 tables, 4 appendixes, 21 sources.

WAREHOUSE ACCOUNTING, MONITORING, AUTOMATION, SALES, DEVELOPMENT, REQUIREMENTS ANALYSIS, DESIGN, WEB APPLICATION.

The object of research is the process of developing and maintaining an automated system for warehouse accounting.

The subject of the research is methods of developing and maintaining an automated system for warehouse accounting.

The purpose of this work is to programmatically implement an automated system for warehouse accounting.

Materials, methods and technical tools: PHP programming language, Joomla content management system!, MySQL database management system.

Results. An automated system for warehouse accounting has been developed, which is a client-server web application that implements warehouse accounting mechanisms.

The practical value of the work lies in the development of a client-server cross-platform web application for warehouse accounting: maintaining statistics, generating and providing reports, and tracking monetary transactions for goods.

Conclusions. Software has been developed that implements an automated system for warehouse accounting.

Scope of use – corporate employees of the company's structural divisions, buyers.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	10
1.1 Складський облік	10
1.1.1 Види складського обліку	10
1.1.2 Автоматизація обліку товарів	11
1.1.3 Переваги автоматизації складського обліку	12
1.2 Дослідження існуючих аналогів	14
1.2.1 Google Таблиці	14
1.2.2 Система автоматизації складського обліку «ABM WMS»	15
1.2.3 Програма «Класс365»	17
1.2.4 Порівняння існуючих аналогів	18
1.3 Постановка завдання	19
1.3.1 Проектування функціонального навантаження системи	20
1.3.2 Функціональність системи	22
1.3.3 Вимоги до інтерфейсу	22
1.3.4 Не функціональні вимоги до продукту	22
1.3.5 Усвідомлення середовища користувача	23
1.3.6 Оцінка надійності, продуктивності і підтримки	23
1.3 Висновки за розділом 1	24
2 Розробка архітектури програми	25
2.1 Вибір між фреймворком та CMS	25
2.1.1 Вибір CMS для розробки системи	28
2.2 Загальна структура додатку	31
2.3 Модулі CMS для розробки застосунку	32
2.4 Висновки за розділом 2	34
3 Основні рішення щодо реалізації компонентів системи	35
3.1 Проектування системи на основі предметної області	35

3.2	Висновки за розділом 3	46
4	Експлуатація, тестування та експериментальне дослідження програми	47
4.1	Опис застосування	47
4.2	Умови виконання програми	47
4.3	Інструкція по експлуатації програми.....	48
4.3.1	Звернення до програми.....	48
4.3.2	Вхідні й вихідні дані.....	48
4.4	Методика та результати тестування	49
	Висновки.....	53
	Перелік джерел посилання	55
	Додаток А Технічне завдання.....	58
	Додаток Б Опис програми	64
	Додаток В Текст програми	69
	Додаток Д Слайди презентації	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ОС – операційна система

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ПП – програмний продукт;

СКБД – система керування базами даних;

ТЗ – технічне завдання;

CMS – система управління вмістом;

GUI – графічний інтерфейс користувача;

MVC – Model-view-controller;

UML – Unified Modeling Language.

ВСТУП

Компанії, діяльність яких пов'язана з торгівлею або виробництвом, мають справу з тимчасовим зберіганням товару або продукції до моменту реалізації. Всі запаси компанії знаходяться на складах. І питання управління запасами на підприємстві один з найбільш важливих, тому багато керівників задаються питанням, автоматизувати управління складом чи ні.

Завдяки автоматизації обліку на складі помітно знижується кількість помилок, які роблять в процесі роботи співробітники підприємства.

Слід досить чітко розуміти, що використання складської програми для бізнесу надає цілий ряд переваг:

- збільшення середнього чека;
- оптимізація асортименту товарів за рахунок аналізу прибутковості товарів;
- виключення шахрайства працівників з товарами і цінами;
- додаткове залучення клієнтів дисконтними картками;
- виключення податкових штрафів за рахунок чіткого товарного обліку;
- оптимізація кількості персоналу;
- легкість аналізу фінансової та управлінської звітності;
- відсутність прив'язки керівника до фізичного робочого місця;
- автоматичне оновлення програми, що не вимагає втручання підприємця.

Саме тому актуальною є розробка нової автоматизованої системи складського обліку, що надає можливість віддалено та колективно проводити моніторинг та аудит складського наповнення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Складський облік

Складським обліком називається процес контролю за рухом товарів [1]. Він включає в себе:

- замовлення постачальникам;
- приймання;
- зберігання, розподіл;
- відпустка на торгову точку-роздрібний або інтернет-магазин;
- продаж кінцевому покупцеві.

Чим прозоріше і доступніше вся процедура, тим ефективніше працює склад і бізнес в цілому [1].

1.1.1 Види складського обліку

Залежно від сфери діяльності можна застосовувати 4 основних види складського обліку [2]:

1) ручний облік товарів. Підходить для підприємців та організацій з невеликим асортиментом і низьким числом угод за одиницю часу. Наприклад невелика лісопилка продає 3 види бруса. У день робиться 10-15 відвантажень покупцям. Все це цілком можна фіксувати в робочому зошиті комірника;

2) облік в текстових редакторах або таблиці Excel. Більш просунутий, але теж далеко не ідеальний варіант. У такі програми можна занести трохи більше інформації, але робити це теж доведеться вручну. Крім цього, Excel може виконувати нескладні математичні розрахунки в автоматичному режимі [1], [2];

3) автоматизований облік товарів на складі. Реалізується за допомогою спеціальних програм-товарооблікових систем. У цьому варіанті все відбувається в автоматичному режимі-потрібно тільки створити первинну базу і робити позначки про рух;

4) комплексна автоматизація обліку. Контролює не тільки склади, а й торгові площі і вітрини магазинів, продажі, закупівлі і все інше. Також реалізується за допомогою програм для автоматизації обліку.

1.1.2 Автоматизація обліку товарів

За складним терміном ховається дуже проста суть: автоматизація потрібна для того, щоб упорядкувати роботу складу і звести до мінімуму участь людини [3-5]. Все робить спеціальна програма для автоматизації обліку. Хороші системи вміють наступне [3-5]:

- контролювати складські залишки в режимі реального часу. Ви завжди точно знаєте, скільки і які товари є в наявності. Ситуація, коли раптово закінчується певна позиція, виключена;
- відстежувати всі рухи товарів. Відвантаження в магазин, міграція між складами, переміщення-все знаходиться під контролем системи;
- фіксувати продажі. Будь-яка позиція списується із залишків в той момент, коли продавець пробиває касовий чек;
- стежити за співробітниками. Вона аналізує ефективність менеджерів, продавців, комірників та інших осіб, що беруть участь у процесі;
- сигналізувати про наявність товару на складі. Якщо щось закінчується, система вчасно попередить про це, і ви встигнете зробити замовлення у постачальника;
- формувати бухгалтерські звіти та документи для постачальників. Самостійно і в автоматичному режимі. Вам залишається тільки їх роздрукувати і підписати, якщо це необхідно;
- створювати технологічні карти для підприємств громадського харчування;
- розробляти та впроваджувати програми лояльності для постійних покупців;
- контролювати продажі та формувати аналітику для керівника;

- працювати з декількома складами, магазинами і торговими точками. Доступ до програми можуть мати кілька співробітників одночасно;
- вивантажувати дані з інших програм і форматів. Вбити вручну кілька сотень позицій-довго і складно. Простіше вивантажити інформацію в систему їх Excel-таблиць або інших і баз даних [3-5].

На цьому список можливостей товарооблікових програм не обмежується. Вони можуть відстежувати закупівельні ціни постачальників, збирати замовлення з декількох складів, контролювати гроші, закладені в товарообігу і багато іншого [6].

1.1.3 Переваги автоматизації складського обліку

Облік складських запасів-це завжди робота з великим обсягом даних. Автоматизація ж обліку дозволяє економити час, гроші і людський ресурс підприємства. Зараз представлений великий вибір програм для управління складськими запасами [6].

Знати про реальний залишок товарів на складі. Хороший деталізований облік складських запасів дозволяє визначати оборот продукції за різними критеріями і проводити аналіз продажів [6], [7]. Програма складського обліку дозволяє зробити склад прозорим, вона надає всю інформацію про складські запаси – вид товару, кількість, дату закупівлі, термін зберігання та інше [7].

Ніяких втрат і ризиків. Автоматизація усуває проблеми зайвих трудовитрат, економить час на ручний облік і формування документації. Товар, який зберігається на складі, сам по собі ризик, і чим більше товару, тим більше ризик зазнати збитків. Все залежить від виду товару [6-8].

Якщо товар з встановленим терміном придатності (продукти, косметика або ліки) програма сама вчасно його виявляє, а менеджери підприємства повинні подбати про своєчасну реалізацію цієї продукції. При великих обсягах одного виду товару існує ризик, що товар втратить свою

актуальність, це призведе або до втрати вкладених коштів, або до отримання низького доходу [6-8].

Всіх перерахованих вище проблем можна уникнути при регулярному формуванні та аналізі різноманітних звітів [6-8]:

- про залишки товару;
- про надходження товару на склад;
- по руху товару;
- відомості обороту товару;
- про неліквідні товари.

Контролювати персонал [6-8]. Усунути розкрадання. Зниження обсягу розкрадань спостерігається після впровадження автоматизації обліку складу в поєднанні з системою інвентаризації. Облік руху номенклатури складу виявляє відповідального за нестачу, до якого застосовуються штрафні санкції [6-8].

Крім контролю діяльності складських працівників для підвищення продуктивності праці і поліпшення виробничої атмосфери створюється система матеріальної мотивації колективу [6-8].

Підвищити мобільність [6-8]. Складський переїзд-без накладок. Важлива причина для ведення автоматизованого обліку складу – це забезпечення мобільності. Автоматизація складу дозволяє працювати не тільки з одним складом, а паралельно з декількома. Система розбиває склад на кілька відділів, і кожному відділу задаються незалежні параметри [6-8].

При переїзді з одного на інший склад використовується можливість програми підтримки параметрів адресного зберігання товарів. Автоматизація також виконує завдання прийому і розміщення товарів на склад, що сприяє швидкому переїзду і налагодженню роботи вже на новому місці [6-8].

Оптимізувати складське господарство [6-8]. Наступна причина, по якій слід автоматизувати облік складу – це оптимізація складського господарства, при якій будуть досягнуті постійні продажі при мінімальних запасах на складі [6-8].

Оптимізація здійснюється проведенням операцій, таких як збір даних, побудови моделей, прогнозування та планування. Автоматизація обліку дозволяє максимально швидко і точно виконати всі ці операції і підвищити точність планування закупівель[6-8].

1.2 Дослідження існуючих аналогів

Все більше підприємців схильється до вибору онлайн-програм для автоматизації бізнесу [7-11]. Наприклад, більшість програм настільки прості у використанні, що немає необхідності відправляти персонал на додаткове навчання [9], [11]. Крім того, не потрібно витрачати величезні кошти на покупку дорогого ліцензійного програмного забезпечення (ПЗ), адже користуватися програмою можна безкоштовно і необмежено. Якщо ж вам знадобитися доступ до більш широкого функціоналу, Ви зможете підключити професійний тариф.

1.2.1 Google Таблиці

Google Таблиці – це програма для роботи з електронними таблицями, включена в безкоштовний набір веб-редакторів Google Docs, пропонується Google [9]. Застосунок сумісний з форматами файлів Microsoft Excel. Додаток дозволяє користувачам створювати і редагувати файли онлайн, співпрацюючи з іншими користувачами в режимі реального часу. Зміни відстежуються користувачем з історією змін, що представляє зміни [9]. Позиція редактора підсвічується певним для редактора кольором і курсором, а система дозволів регулює, що можуть робити користувачі. В оновленнях з'явилися функції з використанням машинного навчання, включаючи «дослідження», що пропонує відповіді на питання на природній мові в електронній таблиці [9].

Система складського обліку засновані на Google Таблицях звісно реалізують мінімальний набір функцій, проте зазвичай і цього достатньо. Зазвичай такі системи дозволяють вести склад облік [9]:

- враховувати прихід товару;
- видачу товару;
- переміщення товару;
- списання товару;
- контролювати наявність залишків;
- контролювати всі обороти по товару використовуючи картку складського обліку.

Такий підхід підійде замовникам, кому необхідно вести складський облік матеріалів, потрібна автоматизація складського обліку, облік торгівлі, облік товарів [9].

На рисунку 1.1 наведено приклад таких систем обліку на базі Google Таблиць.

PROJECT TITLE		[Project's title]					COMPANY NAME		[Company's name]				
PROJECT MANAGER		[Project Manager's name]					DATE		3/12/18				

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	PHASE ONE									
							WEEK 1		WEEK 2		WEEK 3		WEEK 4		WEEK 5	
							M	T	W	R	F	M	T	W	R	F
1	Project Conception and Initiation															
1.1	Project Charter	Leon W	3/12/18	3/15/18	3	100%										
1.1.1	Project Charter Revisions	Kylie R	3/15/18	3/16/18	1	100%										
1.2	Research	Pete S	3/15/18	3/21/18	6	90%										
1.3	Projections	Steve L	3/16/18	3/22/18	6	40%										
1.4	Stakeholders	Allen W	3/18/18	3/22/18	4	70%										
1.5	Guidelines	Malik M	3/19/18	3/22/18	3	60%										
1.6	Project Initiation	Malik M	3/23/18	3/23/18	0	50%										
2	Project Definition and Planning															
2.1	Scope and Goal Setting	Steve L	3/24/18	3/28/18	4	22%										
2.2	Budget	Allen W	3/25/18	4/2/18	3	16%										
2.3	Communication Plan	Malik M			0	0%										
2.4	Risk Management	Malik M			0	0%										
3	Project Conception and Initiation															
3.1	Status and Tracking	Pete S			0	0%										
3.2	KPIs	Leon W			0	0%										
3.2.1	Monitoring	Kylie R			0	0%										
3.2.2	Forecasts	Kylie R			0	0%										
3.3	Project Updates	Pete S			0	0%										

Рисунок 1.1 – Приклад складського обліку на основі Google Таблиць

1.2.2 Система автоматизації складського обліку «ABM WMS»

Автоматизація складського обліку в хмарній системі ABM WMS – це комплексне логістичне рішення для управління складами різної величини і типів [10].

Автоматизація роботи складу починається з організації коректного обліку. Хмарна система ABM WMS дозволить завжди бути в курсі наявності і рухів товарів, швидко комплектувати і відвантажувати продукцію [10].

Автоматизація складу з програмою ABM WMS підвищує якість бізнес-процесів складу, покращує фінансові показники компанії, не вимагає витрат на дороге програмне забезпечення для складського комплексу, так як є хмарним і надається в оренду [10].

Автоматизація управління складом на базі ABM WMS підходить для компаній різних типів бізнесу [10]:

- виробничі підприємства (склади сировини, склади готової продукції);
- логістичні центри;
- дистриб'юторські та роздрібні компанії;
- фармацевтичні компанії;
- склади відповідального зберігання.

Хмарна програма для автоматизації складу ABM WMS призначена для адресного зберігання, автоматизації приймання та розміщення, автоматизації процесу інвентаризації, білінгу складських операцій, планування, аналізу та контролю роботи складу [10].

На рисунку 1.2 наведено приклад використання програми ABM WMS.

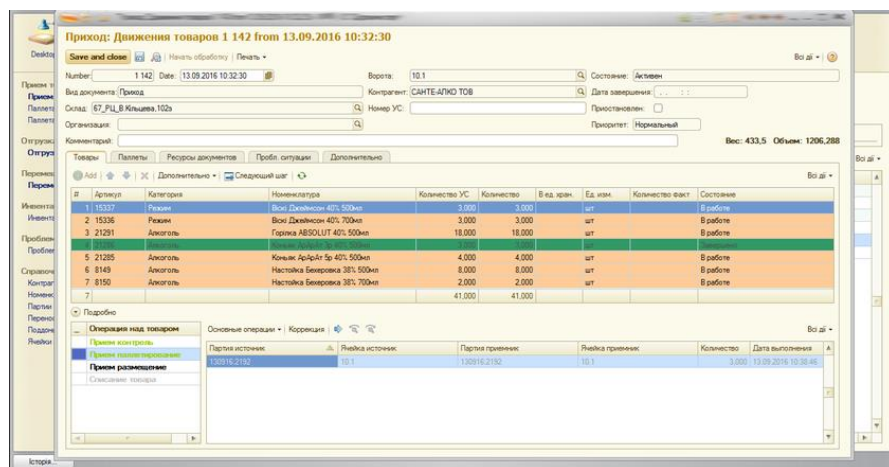


Рисунок 1.2 – Приклад використання програми ABM WMS

1.2.3 Програма «Класс365»

Розробники засвідчують, що «Класс365» настільки проста у використанні, що немає необхідності відправляти персонал на додаткове навчання [11].

Переваги роботи в онлайн-програмі Клас365 [11]:

- доступ до програми з будь-якого пристрою, де є вихід в інтернет;
- звітність в 1 клік по будь-якому напрямку роботи;
- надійний захист даних від моменту реєстрації і протягом всієї роботи;
- гнучка система налаштувань під завдання вашого бізнесу;
- можливість працювати в єдиній системі з віддалених один від одного офісів;
- для роботи з програмою не потрібно проходити навчання.

Почати використання програми дозволить вивести автоматизацію на новий рівень, що вкрай важливо під час конкурування, адже автоматизація – це потужна конкурентна перевага, яка допоможе вам зайняти лідируючі позиції в своєму сегменті ринку [11].

На рисунку 1.3 наведено приклад використання програми «Класс365».

№	Номер	Автор	Организация	Договор	Контрагент	Сумма	Проведен	Статус
1	35	Иванов	Система-А	Договор № 1 от 10.05.2012	ОАО «БЗТО»	1 500.00	✓	Новое
2	13	Иванов	Система-А	Договор № 1 от 2012-07-13	ОАО «График Прода»	24.78	✓	На рассмотрении
3	12	Иванов	Система-А	Договор № 1 от 2012-07-13	ОАО «БЗТО»	39731.35		Отменен
4	24	Иванов	Система-А	Договор № 1 от 2012-07-18	ОАО «БЗТО»	-106.20	✓	На рассмотрении
5	17	Иванов	Система-А	Договор № 1 от 2012-07-18	ОАО «БЗТО»	0.00		На рассмотрении
6	18	Иванов	Система-А	Договор № 1 от 2012-07-18	ОАО «БЗТО»	-106.20	✓	Новое
7	20	Иванов	Система-А	Договор № 1 от 2012-07-18	ОАО «БЗТО»	0.00	✓	Успешно принято

Всего 7 документов на сумму 36 012.67

Рисунок 1.3 – Приклад використання програми «Класс365»

1.2.4 Порівняння існуючих аналогів

Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (табл. 1.1). У таблиці будуть порівнюватися попередні аналоги із системою, що розробляється – «Склад2.0». Серед критеріїв виділимо наступні:

- засоби для складських обліку та аудиту – чи наявні вбудовані засоби саме для складських аудиту та обліку, що спростить використання, налаштування та розшире коло функціональних можливостей;
- встановлення додаткового ПЗ – чи є потреба у встановленні додаткового ПЗ окрім розгортання серверної частини застосунку;
- складність підтримки – складність підтримки системи;
- простота інтерфейсів взаємодії – рівень зрозумілості та простоти інтерфейсів для взаємодії користувача із системою. Цей критерій оцінює загалом систему та її зрозумілість, не тільки графічний інтерфейс користувача (GUI).

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій порівняння	Google Таблиці	ABM WMS	Класс365	Склад2.0
Засоби для складських обліку та аудиту	Відсутні	Присутні	Присутні	Присутні
Встановлення додаткового ПЗ	Не потрібно	Необхідно	Не потрібно	Не потрібно
Складність підтримки	Низький рівень	Високий рівень	Середній рівень	Середній рівень
Простота інтерфейсів взаємодії	Високий рівень	Середній рівень	Низький рівень	Високий рівень

Проводячи порівняння було отримано достатньо цікаві результати. Так нібито найпримітивніший спосіб ведення складського обліку – використання Google Таблиць – засвідчив звісно низький рівень вбудованих механізмів для складського обліку, проте виявився найзрозумілішим та легким в підтримці. Спеціалізовані системи: ABM WMS та Класс365 продемонстрували достатньо широкий спектр функцій проте виявилися складними в підтримці, адже вимагали встановлення додаткового ПЗ на кожен робочу станцію (саме у випадку ABM WMS) або були надлишково громіздкими та мали низький рівень зрозумілості для користувача (система Класс365).

Саме тому актуальною залишається задача розробки нової автоматизованої системи складського обліку із чіткою та розподіленою архітектурою, зрозумілим та комфортним користувацьким інтерфейсом та необхідним, але не надлишковим, функціоналом для обліку.

1.3 Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Метою є розробка нової автоматизованої системи складського обліку «Склад2.0». Система має дозволяти:

- більш ефективно використовувати нові комп'ютерні технології автоматизації;
- перенести обробку даних ближче до клієнта та знизити завантаження мережі і зменшити тривалість транзакцій;
- полегшити використання GUI, які стали доступні на потужних сучасних робочих станціях.

Система повинна зачіпати всі аспекти, пов'язані з рухом товару на склад і зі складу. Також повинна мати простий і зручний інтерфейс.

Розглянемо вимоги до розроблюваної системи за їх типом [12], [13].

1.3.1Проектування функціонального навантаження системи

Перш ніж визначити функціональні вимоги до системи. Спроекуємо функціональне навантаження системи за певними кроками [14].

По-перше, визначаємо завдання [14]. Вибір системи товарообліку залежить від специфіки бізнесу. Для невеликого магазину з одним складом підійде простенька безкоштовна програма. Мережевій торгівлі з великою кількістю точок потрібно щось поскладніше. Перед тим, як вибирати програму, складіть наступний список [14]:

- кількість магазинів;
- кількість складів;
- число постачальників товарів;
- загальний товарообіг на тиждень, місяць, рік;
- число співробітників, які будуть мати доступ до програми одночасно;
- набір необхідних функцій, наприклад: програми лояльності, складання технологічних карт, підтримка додаткового обладнання (електронних ваг, банківських терміналів і так далі).

По-друге, визначаємо бюджет покупки [14]. За ціною готових рішень немає – все впирається в особливості і масштаби бізнесу. Вартість використання Програм товарообліку починається з нуля рублів за безкоштовні або пробні версії. Просунуті продукти для великих мережевиків можуть коштувати десятки тисяч рублів на місяць [14].

По-третє, проектуємо товарооблікову програму [14]. Функціонал багатьох аналогів багато в чому схожий, відмінності будуть в інших моментах:

- зручність інтерфейсу. Не всі програми однаково зручні для використання. Вибирайте продукт з інтуїтивно зрозумілим меню, не перевантаженим зайвим функціоналом;
- наявність додаткових опцій. Будь-яка фіча веде за собою збільшення вартості продукту. Якщо половина можливостей вам не потрібна, таку програму купувати не має сенсу;
- технічна підтримка. Чим оперативніше і якісніше вона працює, тим краще. Перевагу варто віддавати програмам, що здійснюють супровід клієнтів в режимі 24/7/365;
- вартість. Вибирайте максимально просунуту платформу, яка знаходиться в рамках бюджету;
- стабільність роботи. Деякі системи, пропоновані на ринку софту, відверто сируваті. Віддавайте перевагу перевіреним і зарекомендували себе продуктам.

По-четверте, підключасмо до товарооблікової системи [14]. Найпростіша частина завдання з автоматизації. Так потрібно [14]:

- зареєструвати особистий кабінет на сайті розробника платформи. Для цього знадобиться адреса електронної пошти та номер телефону. По ходу справи потрібно придумати логін і пароль;
- вибрати відповідний тариф, що відповідає вашим вимогам;
- вибрати і оплатити Додаткове обладнання: сканери штрих-коду, онлайн-каси, електронні ваги та інше;
- оплатити вартість тарифу. Зазвичай, якщо оплатити послугу на кілька місяців вперед, виходить вигідніше;
- завантажити в систему базу товарів з таблиці Excel, 1С або інших джерел. Можна зробити це вручну за допомогою сканера штрих-кодів або завантажити типову базу;
- запустити програму, підключити і налаштувати обладнання, почати роботу. Далі потрібно тільки вчасно оплачувати вартість послуги.

Нічого складного в автоматизації складського обліку немає. Це набагато простіше, ніж рахувати складські залишки вручну, контролювати персонал і самостійно фіксувати рух товарів [14].

1.3.2 Функціональність системи

Межі системи: система повинна працювати постійно без збою.

Програмний продукт (ПП) повинен виконувати наступні функції:

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;
- можливість оформлення замовлення;
- можливість оплати готівковим / безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення;
- можливість друку.

1.3.3 Вимоги до інтерфейсу

Для забезпечення ергономічності, зручності та зрозумілості інтерфейс розроблюваного веб-застосунку повинен бути зручним та інтуїтивно-зрозумілим.

1.3.4 Не функціональні вимоги до продукту

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- серверна операційна системи (ОС);
- встановлений та розгорнутий веб-сервер;
- інтерпретатор PHP5 та PHP7;
- система керування базами даних (СКБД) MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- персональний комп'ютер (ПК) із встановленою будь-якою користувацькою ОС;
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;
- встановлений веб-браузер.

1.3.5 Усвідомлення середовища користувача

Для користувачів будуть висуватися наступні вимоги:

- систему використовуватимуть люди, що працюють на фірмах типу склад або магазин-склад;
- рівень освіти повинен бути середньо-спеціальним;
- звичайні користувачі, які не мають спеціальної комп'ютерної освіти, повинні володіти комп'ютером на початковому рівні;
- користувачі не обов'язково повинні мати досвід роботи з даними програмами.

1.3.6 Оцінка надійності, продуктивності і підтримки

Для забезпечення достатньої надійності та продуктивності система має відповідати певним вимогам:

- система повинна працювати постійно без збоїв;

- займатися підтримкою системи повинен адміністратор та модератор;
- кожен користувач системи повинен мати ідентифікований логін та пароль;
- бажано щоб програма працювала без додаткового налаштування;
- дана система безкоштовна в межах компанії.

1.3 Висновки за розділом 1

Виконано огляд проблеми автоматизації процесів складського обліку. Виділено основні шляхи для направи для розроблення нового ПП у вигляді веб-застосунку.

Розробка та підтримка автоматизованої системи складського обліку «Склад2.0» має надати систему, що забезпечує можливості зручних:

- використання нових комп'ютерних технологій автоматизації;
- перенесення обробки даних ближче до клієнта та зниження завантаження мережі і зменшити тривалість транзакцій;
- полегшення використання GUI, які стали доступні на потужних сучасних робочих станціях.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

У попередньому розділі було проведено аналіз та дослідження існуючих аналогів ПП для автоматизації складського обліку. Виходячи з проведеного аналізу, було сформульовано основні можливості, якими повинна оперувати система, що проектується.

Що стосується технічних та програмних можливостей, то система повинна забезпечувати можливість швидко здійснювати моніторинг та перевірку складського обліку.

Даний розділ містить опис питань, які стосуються безпосередньо структури системи, що розробляється, вибору мови програмування, системи керування вмістом (CMS), середовища проектування, а також визначаються основні та специфічні архітектурні рішення, які має система.

2.1 Вибір між фреймворком та CMS

По суті, будь-який сайт або веб-застосунок можна розробляти за допомогою одного з трьох підходів [15], [16]:

- взяти систему «із коробки» управління сайтом і адаптувати її під завдання;
- взяти фреймворк і на його основі реалізувати необхідний функціонал;
- написати систему з нуля, використовуючи чисту мову програмування.

Розробку з нуля варто відразу відкинути, так як цей підхід може бути правильним тільки в тому випадку, якщо створення проекту – це основне завдання компанії, а ресурсів під це завдання виділено дуже багато [15], [16]. Хороші проекти з нуля пишуться дуже довго, хоча цей підхід дозволяє створювати дуже серйозні рішення [15], [16].

Тому розглянемо тільки створення сайту на базі платформ [15], [16].

Розробка на CMS – найбільш правильний підхід, якщо проект досить типовий. Тобто в CMS вже є всі потрібні модулі, а ті процеси, які вбудовані в CMS, майже повністю відповідають очікуванням [15], [16].

Розробка на фреймворку – найбільш правильний підхід, якщо бачення проекту не особливо вкладається в можливості і процеси присутніх на ринку CMS. Тобто тоді, коли в разі використання CMS її доведеться істотно допрацьовувати [15], [16].

По суті, вибір варто робити на підставі економічної складової розробки. Нижче наведені деякі критерії, які дозволять прийняти обґрунтоване рішення з цього питання [15], [16].

Функціонал, який вбудований в CMS, дорожче і довше реалізовувати на фреймворку, а розробка складного функціоналу на CMS або переписування базових процесів CMS коштує дорожче і займає більше часу, ніж та ж робота виконана відразу на фреймворку [15], [16].

Домогтися від складного проекту на CMS високої швидкості роботи коштує дорожче, ніж зробити це на фреймворку. Аналогічно йдуть справи і з масштабуванням. Тобто при високих вимогах до стійкості до навантажень, продуктивності або до відмовостійкості вибирайте рішення на фреймворках (або закладайте вартість робіт з оптимізації CMS в бюджет проекту) [15], [16].

Запуск першої пілотної (неповної) версії проекту на CMS завжди швидше, ніж запуск аналогічної версії на фреймворку. Якщо проект складний, а терміни запуску «горять», то краще або випускати «пілот» на CMS, а потім його затратно допрацьовувати або паралельно з цим розробляти рішення на фреймворку, або розставляти пріоритети між термінами розробки і складністю проекту (або відмовлятися від складного функціоналу, або збільшувати терміни) [15], [16].

CMS значно прискорює розробку простого шаблонного сайту. У сайту відразу готова адміністративна панель і її не треба писати окремо, на відміну від розробки на фреймворку. Однак це швидкість створення сайту

досягається за рахунок шаблонності, обмеженості або зайвої універсальності CMS [15], [16].

При бажанні розширити функціонал, додати щось таке, можливо, доведеться прямо воювати з CMS або йти на якісь компроміси. У випадку з фреймворком, таких проблем немає. Так розробку йде більше часу, проте будь-яка задумка здійсненна без будь-яких милиць [15], [16].

У фреймворках часто тільки самий необхідні функції і хороший, налагоджений, відрефакторений код. У CMS, особливо, якщо вона передбачає сторонні модулі, можна зустріти нісенітницю в коді і навіть шкідливий код, який може, наприклад додавати посилання на ліві сайти в твій проект [15], [16].

Ведучи розробку за допомогою фреймворку, розробник сам будуватиме взаємодію користувача з функціями і даними і сам вирішує як це буде виводитися в браузер. У CMS це все вирішили за тебе. Ймовірно, це і непогано, адже якщо у розробника недостатньо досвіду, то з фреймворком або чистою мовою можна розробити таке, що в підсумку CMS виявиться дуже хорошим варіантом [15], [16].

Використовуючи CMS, розробник прив'язується до структури, створеної розробниками, але в шаблонних рішеннях економимо час, а в нешаблонних, найчастіше, животіємо на форумах в пошуках відповідей [15], [16].

А фреймворк дає повну свободу дій. За розробника написали основу, фундамент, а далі бери і твори. Але для якісної розробки на фреймворку необхідно володіти достатнім рівнем, щоб не створювати відвертої непотребщини або, що ще гірше, дірявого продукту [15], [16].

Для полегшення вибору проведемо порівняння використання фреймворку та CMS під потреби замовника [15], [16].

Результати порівняння представимо в табличному вигляді – таблиця 2.1. У якості критеріїв виділемо:

- розробка представлень – швидкість та простота розробки основних сторінок веб-застосунку;
- розробка адміністративної панелі – швидкість та простота розробки адміністративної панелі для адміністрування веб-застосунку;
- підключення платіжних систем – наявність плагінів та інших механізмів для підключення платіжних систем;
- простота та швидкість розробки – загальна швидкість розробки проекту;
- простота підтримки – простота подальшої підтримки проекту.

Таблиця 2.1 – Порівняння використання фреймворку та CMS під завдання замовника

	Фреймворк	CMS
Розробка представлень	+	++
Розробка адміністративної панелі	+	++
Підключення платіжних систем	++	++
Простота та швидкість розробки	+	++
Простота підтримки	+	+

Виходячи з порівняння розробка з використанням CMS буде більш швидкою, не буде потребувати додаткових витрат ресурсів часу на розробку адміністративної панелі. Більш того така розробка буде більш простою в подальшій підтримці, а проблеми можуть бути пов'язані виключно із певним переорієнтуванням системи, в тому випадку якщо буде потрібен певний надлишковий функціонал, що не відповідає початковій задумці.

2.1.1 Вибір CMS для розробки системи

Joomla! – CMS написана на мовах PHP і JavaScript, що використовує в якості сховища бази даних СКБД MySQL або інші стандартні промислові реляційні СКБД [17]. Є вільним ПЗ.

CMS Joomla! включає в себе мінімальний набір інструментів при початковій установці, який доповнюється в міру необхідності. Це знижує захащення адміністративної панелі непотрібними елементами, а також знижує навантаження на сервер і економить місце на хостингу [17].

Joomla! дозволяє відображати інтерфейс фронтальної та адміністративної частини на будь-якій мові. Каталог розширень містить безліч мовних пакетів, які встановлюються штатними засобами адміністрування [17].

Структурно система являє собою ядро, до якого приєднуються компоненти. Компоненти можуть складатися за функціональним призначенням з модуля, плагіна та/або шаблону [17]:

- модулі-елементи, що відображаються на сторінці користувачів у вигляді окремих блоків.
- плагіни-елементи, що відображаються на адміністративній панелі інструментів і надають додаткову функціональність.
- шаблони-елемент, що відповідає за уніфіковане оформлення всього сайту.

В ядро Joomla! Входить [17]:

- блок (функціональність визначається HTML-кодом всередині нього) і його різновиди з розширеними можливостями управління з адміністративної панелі: банери (модулі для розміщення реклами), контакти (блок для розміщення контактної інформації та форми зворотного зв'язку), Стрічка новин (RSS підписка);
- матеріал (стаття) з мітками (ключові слова) і категоріями (папками) для навігації користувачів, редактором для зручного оформлення матеріалу.

Не входять в ядро вільні сторонні розширення [17]:

- плагін для резервного копіювання (backup) і відновлення, наприклад, Akeeba Backup.

Створення власного розширення здійснюється за допомогою написання плагіна на мові PHP з використанням для взаємодії API Joomla! [17].

У таблиці 2.2 наведено порівняння CMS за критеріями. Серед характеристик для порівняння виділімо:

- безпека та продуктивність – безпека та продуктивність веб-застосунків розроблених з використанням тієї або іншої CMS;
- можливості самостійного конфігурування – можливість самостійного налаштування проекту під потреби;
- зручність використання – загальний критерій для визначення зручності маніпулювання веб-застосунком через CMS, що важливо для подальшої підтримки та адміністрування системою;
- вбудовані модулі – наявність та кількість вбудованих базових та безкоштовних модулів для розширення функціональності веб-застосунку та його подальшого модифікування.

Як видно із порівняння всі CMS надають широку палітру для розробки веб-застосунків. А загальним слабким місцем є безпека та продуктивність, проте це вирішується використанням лише офіційних розширень, що пройшли перевірку. Також Joomla! Із самого початку надає більшу можливість для адаптації розробки за рахунок більшої кількості вбудованих модулів, що поставляються офіційно. Тому вибір був зроблено саме на користь Joomla!.

Таблиця 2.2 – Порівняння CMS

	Joomla!	WordPress	OpenCart
Безпека та продуктивність	+	+	+
Можливості самостійного конфігурування	++	++	+
Зручність використання	++	++	+
Вбудовані модулі	++	+	+

2.2 Загальна структура додатку

Термін «клієнт-серверна архітектура» – збірне поняття, що складається з двох взаємодоповнюючих компонентів: сервера і, власне, клієнта [18], [19].

Клієнт – локальний комп'ютер на стороні віртуального користувача, який виконує відправку запиту до сервера для можливості надання даних або виконання певної групи системних дій [18], [19].

Сервер – дуже потужний комп'ютер або спеціальне системне обладнання, яке призначається для вирішення певного кола завдань по процесу виконання програмних кодів [18], [19]. Він виконує роботи сервісного обслуговування за клієнтськими запитами, надає користувачам доступ до певних системних ресурсів, зберігає дані або БД [18], [19].

Особливості такої моделі полягають в тому, що користувач відправляє певний запит на сервер, де той системно обробляється і кінцевий результат відсилається клієнту. У можливості сервера входить одночасне обслуговування відразу декількох клієнтів [18], [19].

Якщо одночасно надходить більше одного запиту, то такі запити встановлюються в певну чергу і сервером виконуються по черзі. Часом запити можуть мати свої власні пріоритети [18], [19]. Частина запитів з більш високими пріоритетами будуть постійно виконуватися в першочерговому порядку [18], [19].

Параметри, які можуть реалізуватися на стороні сервера [18], [19]:

- зберігання, захист і доступ до даних;
- робота з вступниками клієнтськими запитами;
- процес відправки відповіді клієнту.

Параметри, які можуть реалізуватися на стороні клієнта:

- майданчик з надання користувальницького графічного інтерфейсу;
- формулювання запиту до сервера і його подальша відправка;

– отримання підсумків запиту і відправка додаткової групи команд (запити на додавання, оновлення інформації, видалення групи даних).

Архітектура системи клієнт-сервер формулює принципи віртуального спілкування між локальними комп'ютерами, а всі правила і принципи взаємодії знаходяться всередині протоколу [18], [19].

Мережевий протокол – це особливий набір правил, на підставі якого виконується точна взаємодія між комп'ютерами всередині віртуальної мережі [18], [19].

Застосунок «Склад2.0» буде мати клієнт-серверну архітектуру, що представлена на рисунку 2.1.

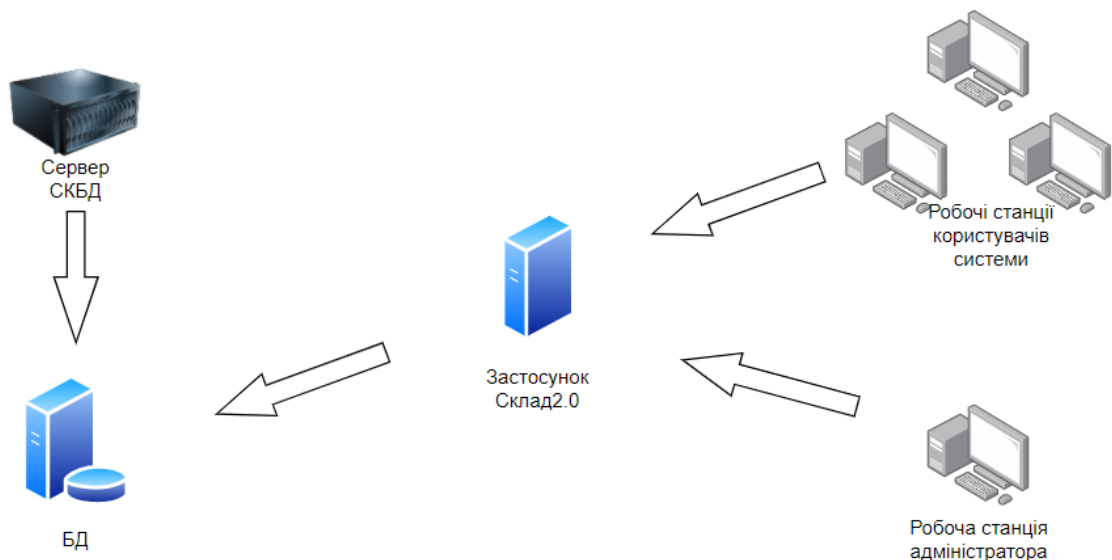


Рисунок 2.1 – Архітектура застосунку «Склад2.0»

2.3 Модулі CMS для розробки застосунку

Модулі CMS, що будуть використовуватися під час розробки [17] наведені в таблиці 2.3.

Таблиця 2.3 – Модулі Joomla!, що будуть використовуватися

Модуль	Мета використання
mod_product_categories	модуль категорій товарів. З його допомогою можна відобразити категорії товарів.
mod_productsroller	модуль прокрутки товарів.
mod_virtuemart_allinone	модуль «все в одному». Може відображати останні, вибірки, або випадкові товари з використанням вкладок.
mod_virtuemart_cart	модуль міні – кошика. Відображає міні – кошик для клієнта з посиланням на сторінку кошика.
mod_virtuemart_currencies	модуль валют. Відображає спадний список валют, які доступні/приймаються у вашому магазині.
mod_virtuemart_featuredprod	модуль рекомендованих товарів. Може показувати рекомендовані товари. Вони позначаються, як «особливі».
mod_virtuemart_latestprod	модуль останніх товарів. Показує нові товари у вашому магазині.
mod_virtuemart_login	модуль входу. Показує форму авторизації для користувачів VirtueMart.
mod_virtuemart_randomprod	модуль випадкових товарів. Відображає випадкові товари з усього магазину або з обраної категорії.
mod_virtuemart_search	модуль пошуку. Відображає форму пошуку товарів.
mod_virtuemart_topten	вибірка продуктів. Відображає ТОП кращих товарів Вашого магазину.
search Bot	виконує пошук по таблиці товарів і поставляє результати пошуку.
productSnapshot Mambot / Plugin	показує посилання на зміст пунктів зображення товару / ціна / додати в кошик.
onPay	платіжний модуль дозволяє приймати WebMoney, пластикові картки VISA і MasterCard, а також безліч інших інтернет – валют.
virtuemart	стандартний модуль для звітності буде використовуватися для роботи бухгалтера. Для роботи Логіста буде використовуватися стандартний модуль в virtuemart «Склад» окрема сторінка, на якій можна подивитися залишки всіх товарів на складі.
AcyMailing	компонент для створення Email розсилок на Joomla.

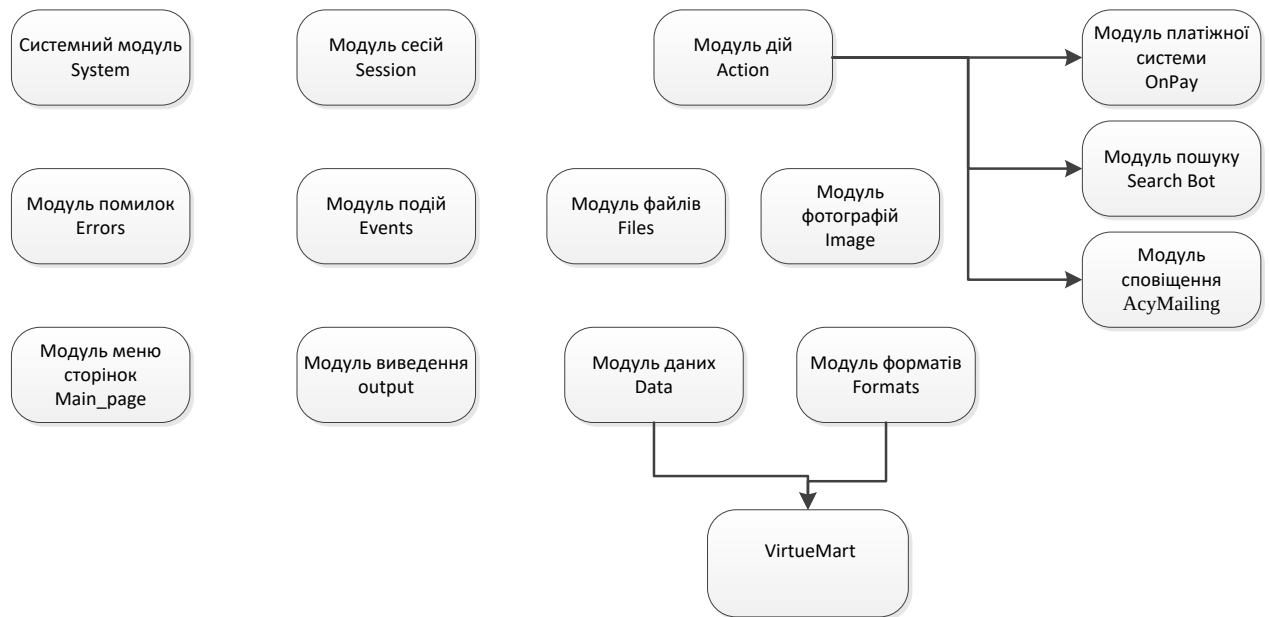


Рисунок 2.1 – Модульне представлення веб-застосунку

2.4 Висновки за розділом 2

Таким чином, архітектура програмного засобу дозволяє вирішувати задачі, що були поставлені перед системою, пов'язані із внутрішнім аудитом. Засоби, що використовуються, забезпечують реалізацію функцій програми.

Для програмної реалізації можливостей ПЗ, що реалізує функції складського обліку, виділених під час аналізу у попередньому розділі, проведено вибір засобів та обґрунтовано структуру системи, що проектується.

Аналізуючи обрану структуру ПЗ, можна зробити висновок, що вона є лаконічною, але, разом з тим, досить працездатною, використовує ресурси ПК раціонально.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Проектування системи на основі предметної області

Unified Modeling Language (UML) – уніфікована мова моделювання, що використовується у парадигмі об’єктно-орієнтованого програмування та є невід’ємною частиною уніфікованого процесу розроблення ПЗ [20], [21]. UML заснована на діаграмах, що дають можливість представити систему у такому вигляді, щоб її можна було легко перевести у програмний код.

Діаграми класів – статичні структурні діаграми, що описують структуру системи, демонструють класи систем, їх атрибути, методи і залежності між класами [20], [21].

Діаграма класів, розроблена для системи «Склад2.0» представлена на рисунок 3.1.

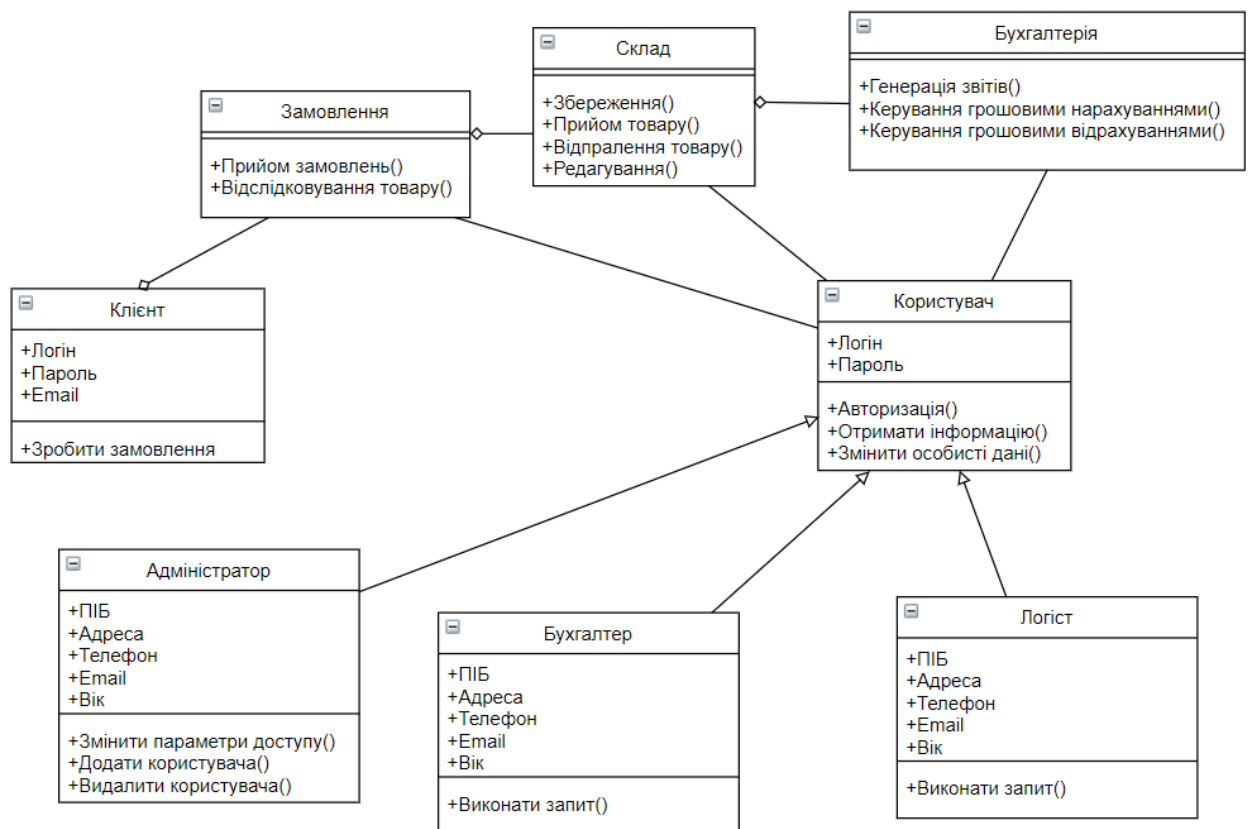


Рисунок 3.1 – Діаграма класів системи

Діаграми пакетів – структурні діаграми, основним змістом яких є пакети і відносини між ними [20], [21]. Ці діаграми служать, в першу чергу, для організації елементів в групі за якими-небудь ознаками з метою спрощення структури і організації роботи з моделлю системи [20], [21].

Система «Склад2.0» складається і 5 компонентів, для роботи з різними відділами компанії, що представлено на рисунку 3.2.

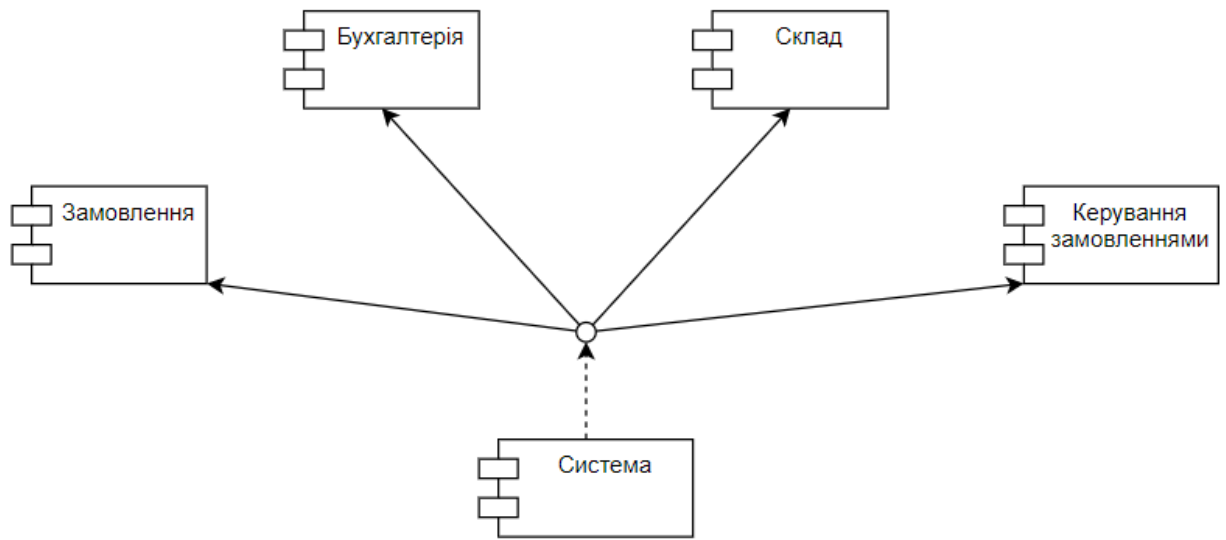


Рисунок 3.2 – Діаграма компонентів системи

Діаграма компонентів – статична структурна діаграма, що показує розбиття програмної системи на структурні компоненти і зв'язки (залежності) між компонентами [20], [21]. В якості фізичних компонент можуть виступати файли, бібліотеки, модулі, файли, пакети тощо [20], [21].

На рисунку 3.3 показано, що система складається з 3 пакетів:

- пакет, який відповідає за роботу системи;
- пакет, який відповідає за роботу співробітників компанії;
- пакет для роботи з клієнтами.

Перший пакет є основним у цій діаграмі, а два інших структурно від нього залежать.

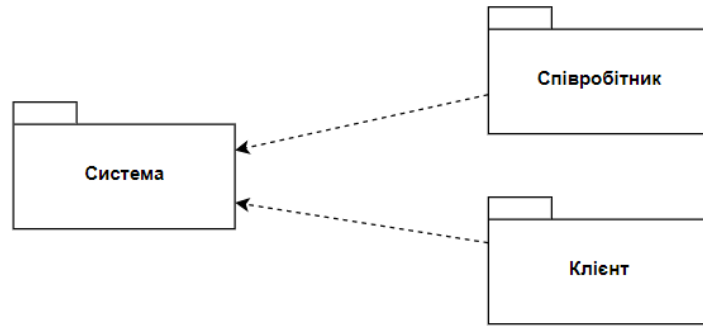


Рисунок 3.3 – Діаграма пакетів системи

Діаграми прецедентів застосовуються для моделювання виду системи з точки зору прецедентів (або варіантів використання) [20], [21]. Даний вид діаграм використовується для моделювання динамічних аспектів системи, так само, як і діаграми станів і діяльності [20], [21].

На рисунках 3.4–3.8 зображені діаграми прецедентів для автоматизованої системи «Склад2.0». Для них були наведені базові та альтернативні потоки виконання.

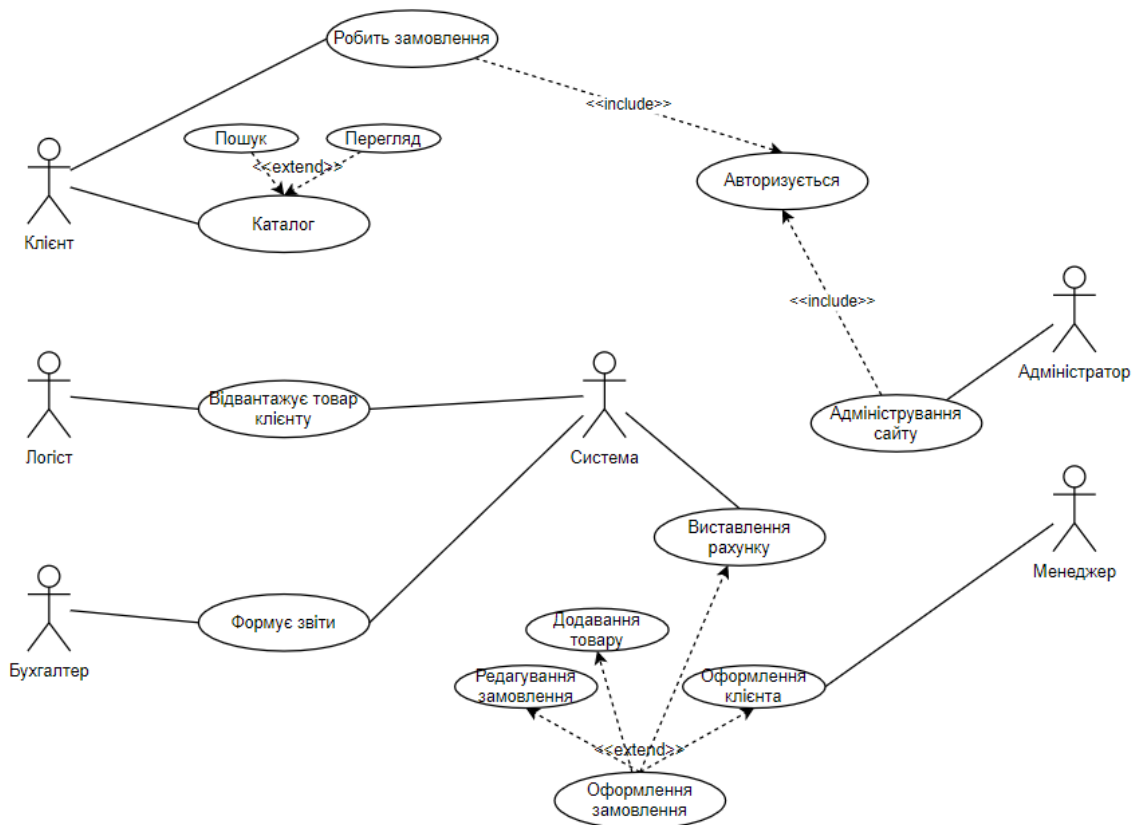


Рисунок 3.4 – Діаграма прецедентів «Робота з системою»

Базовий потік:

- завантажити систему;
- ввести ім'я;
- ввести прізвище;
- ввести e-mail;
- ввести логін;
- ввести пароль;
- ввести підтвердження пароля;
- підтвердження реєстрації;
- збереження даних в БД.

Альтернативні потоки.

Якщо введений e-mail не існує:

- вивести повідомлення про помилку в введеному e-mail.

Якщо введений пароль не збігається з підтвердженням:

- вивести повідомлення про розбіжність пароля.

Якщо введений логін вже існує:

- вивести повідомлення про те, що такий логін вже зареєстрований.

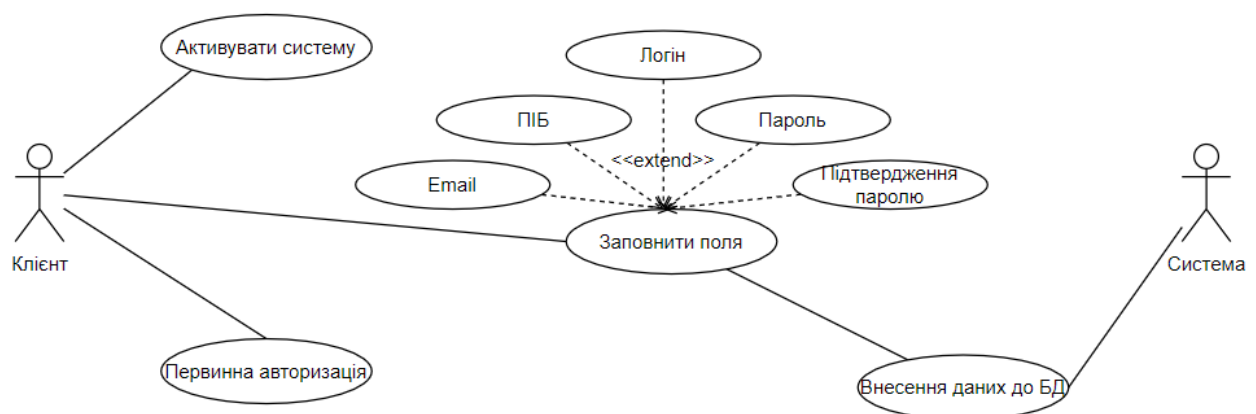


Рисунок 3.5 – Діаграма прецедентів «Реєстрація»

Базовий потік:

- завантажити систему;
- ввести логін;
- ввести пароль;
- підтвердити рішення про авторизацію;
- перевірка даних в БД.

Альтернативні потоки.

Якщо введений логін не існує:

- вивести повідомлення про помилку в введеному логіні.

Якщо введений пароль не існує:

- вивести повідомлення про помилку введення пароля.

А також (і) у разі необхідності:

- запропонувати відправити пароль на e-mail.

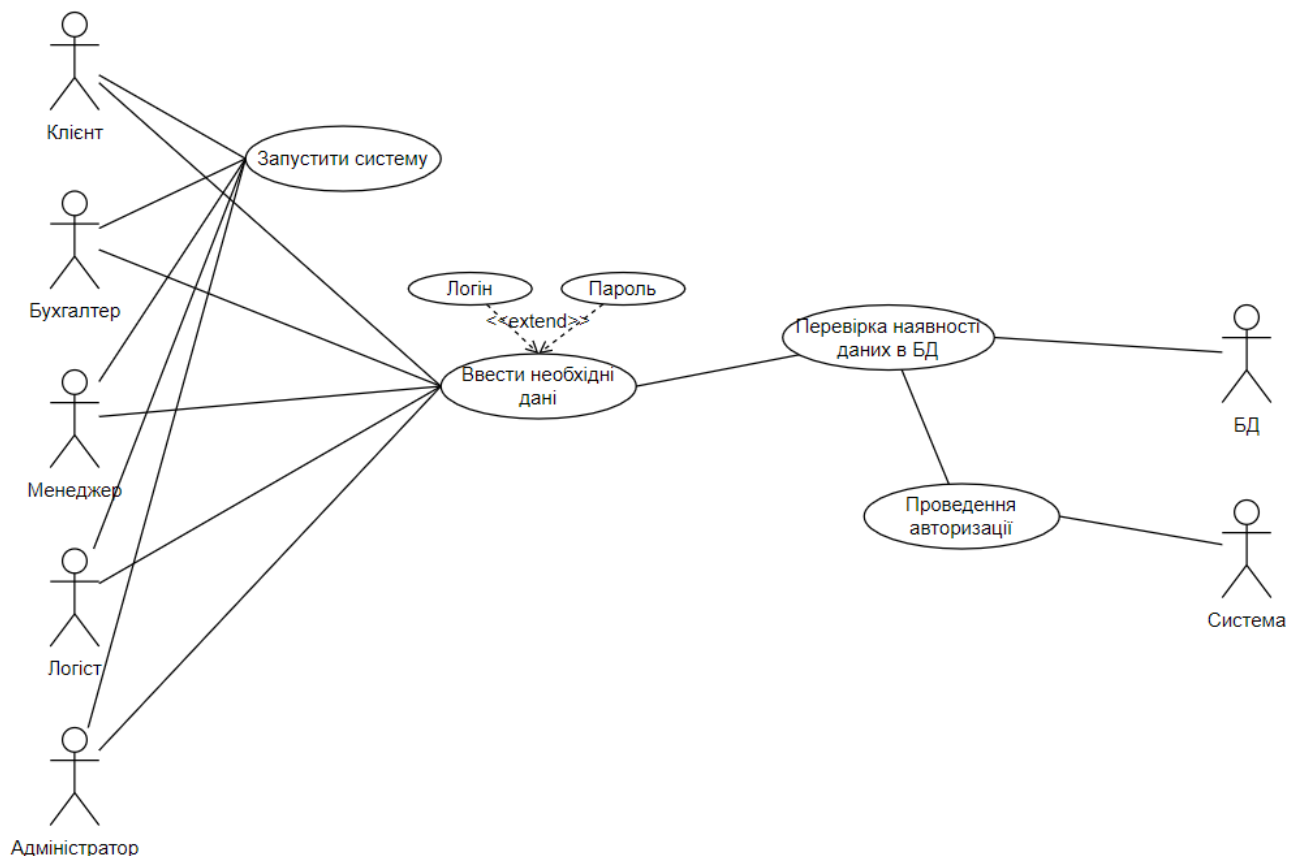


Рисунок 3.6 – Діаграма прецедентів «Авторизація»

Базовий потік:

- завантажити систему;

- введення значення в поле пошуку;
- натискання кнопки «Пошук»;
- пошук збігів в БД;
- висновок результатів.

Альтернативні потоки.

Якщо не знайдено жодного товару, що задовольняє пошук:

- вивести повідомлення про відсутність товарів, що задовольняють запитам пошуку.

Якщо необхідно встановити більш детальну інформацію для пошуку:

- перейти до фільтрації результатів.

Якщо товар відсутній на складі:

- вивести повідомлення про тимчасову відсутність певного товару.

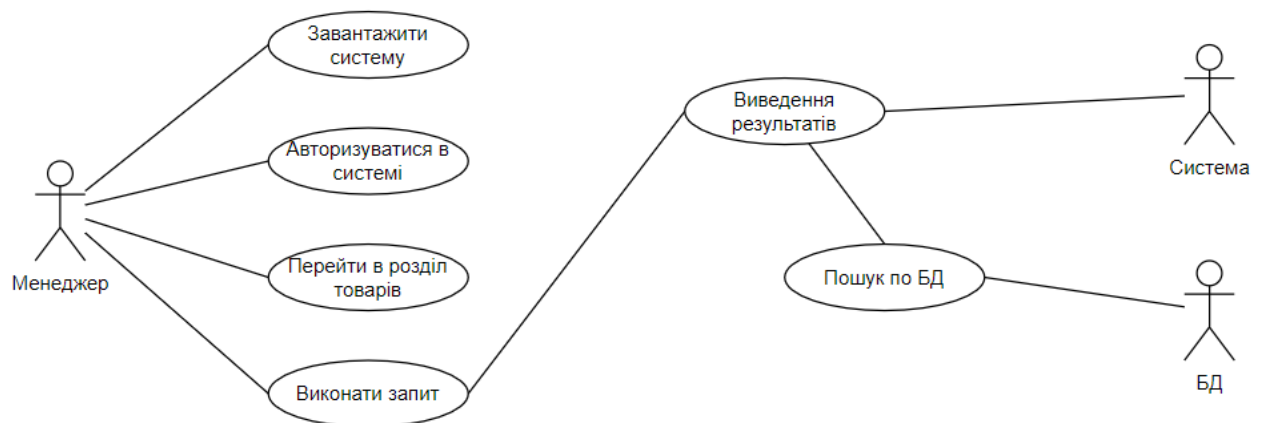


Рисунок 3.7 – Діаграма прецедентів «Пошук товару»

Передумова:

- авторизація виконується під логіном і паролем менеджера.

Базовий потік:

- авторизація;
- перейти в розділ «Продажі»;
- вибрати – «Перегляд статистики»;
- формування статистики;
- висновок статистики.

Альтернативні потоки.

Якщо немає жодного продажу»

- вивести повідомлення про відсутність продажів

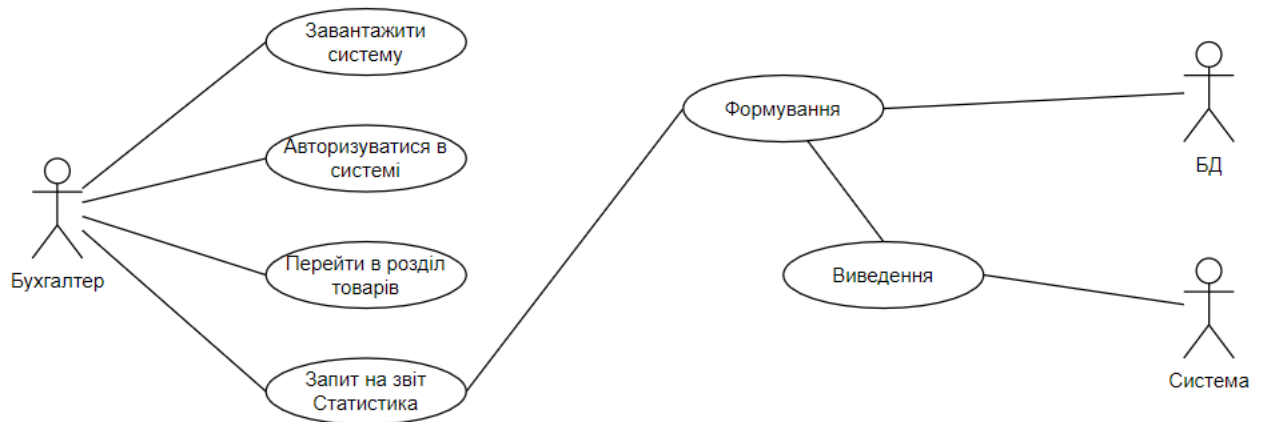


Рисунок 3.8 – Діаграма прецедентів «Переглянути статистику продажів»

Базовий потік:

- завантажити систему;
- авторизуватися в системі;
- переглянути товари;
- вибрати товари;
- підтвердити введені дані;
- вибрати спосіб оплати;
- перевірка введених даних;
- провести оплату;
- обробка замовлення.

Альтернативні потоки.

Якщо не всі необхідні поля заповнені:

- вивести повідомлення про помилку в введених значеннях

Діаграми станів зображують всілякі стани, в яких може перебувати конкретний об'єкт, а також зміни стану об'єкта, які відбуваються в результаті впливу деяких подій на цей об'єкт [20], [21].

На рисунку 3.9 зображена діаграма станів для веб-застосунку автоматизована система складного обліку. На ній показано переходи між станами, коли здійснюється операція пошуку.

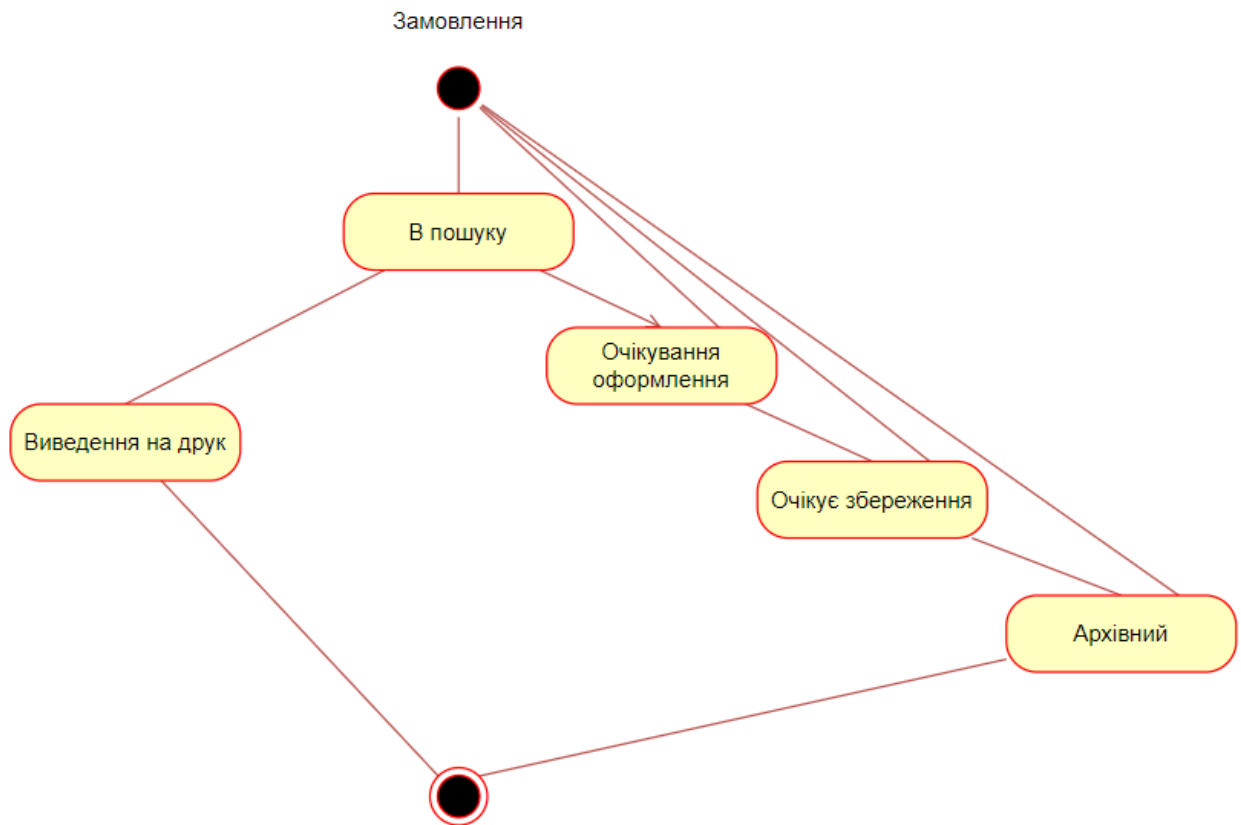


Рисунок 3.9 – Діаграма станів

Діаграма діяльності показує переходи між видами діяльності. Модель видів діяльності може представляти в графічній формі потік подій для прецеденту. Діаграма діяльності для твору замовлення показана на рисунках 3.10.

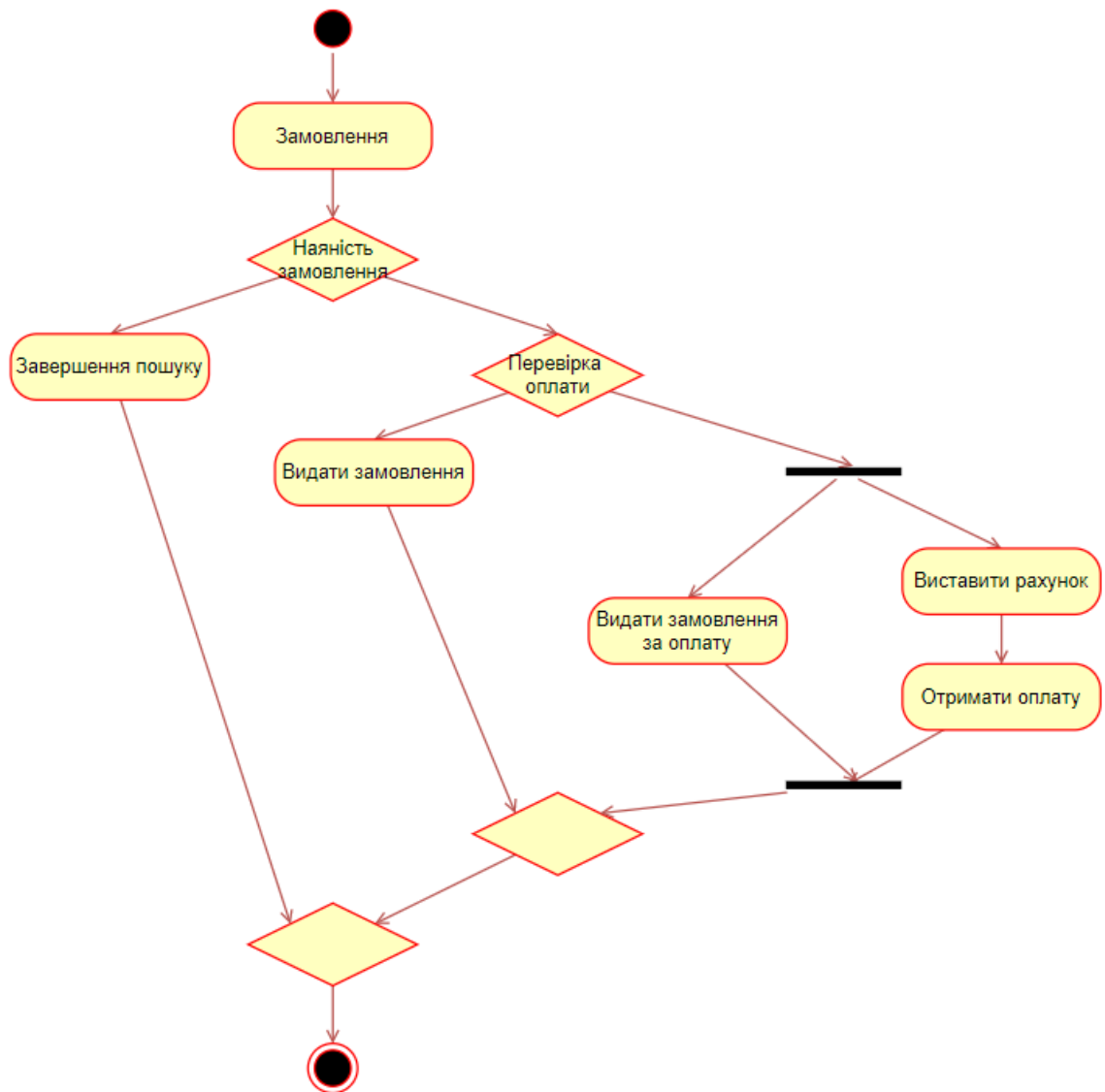


Рисунок 3.10 – Діаграма діяльності

Діаграма кооперації призначена для специфікації структурних аспектів взаємодії [20], [21]. Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, що беруть участь в цій взаємодії [20], [21].

На рисунках 3.11-3.19 зображені діаграми кооперацій, показано взаємодію користувача та системи коли виконуються операції: авторизація, реєстрації, оформлення замовлення, редагування замовлення, Пошук товару, оформлення звіту, адміністрування системи.

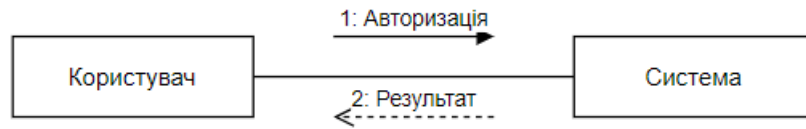


Рисунок 3.11 – Авторизація

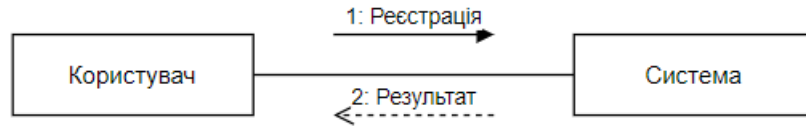


Рисунок 3.12 – Реєстрація

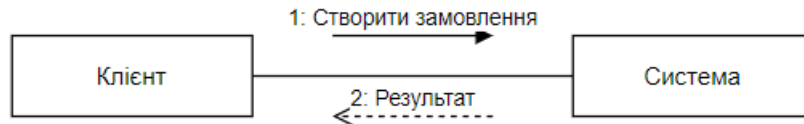


Рисунок 3.13 – Оформлення замовлення

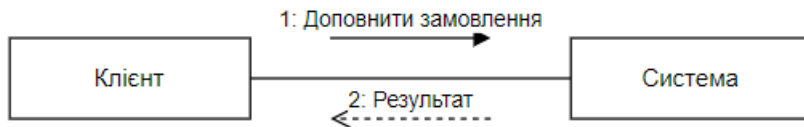


Рисунок 3.14 – Редагування замовлення

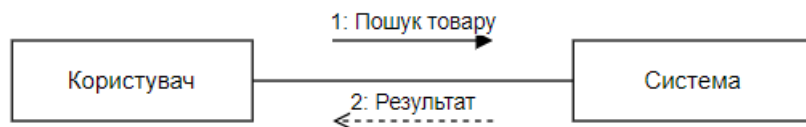


Рисунок 3.15 – Пошук товару

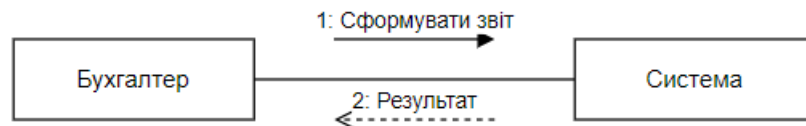


Рисунок 3.16 – Оформлення звітів

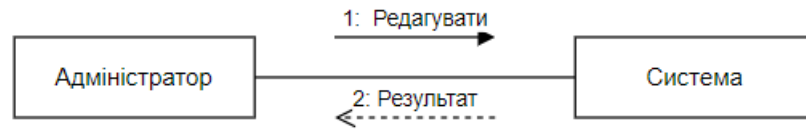


Рисунок 3.17 – Адміністрування системи

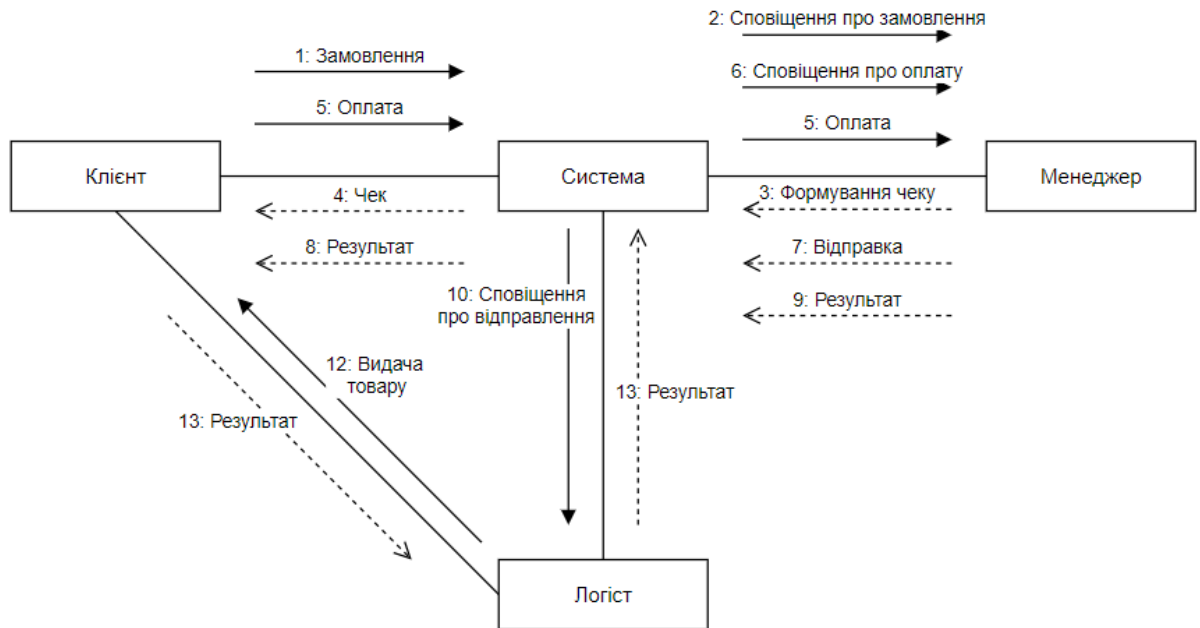


Рисунок 3.18 – Продаж товарів

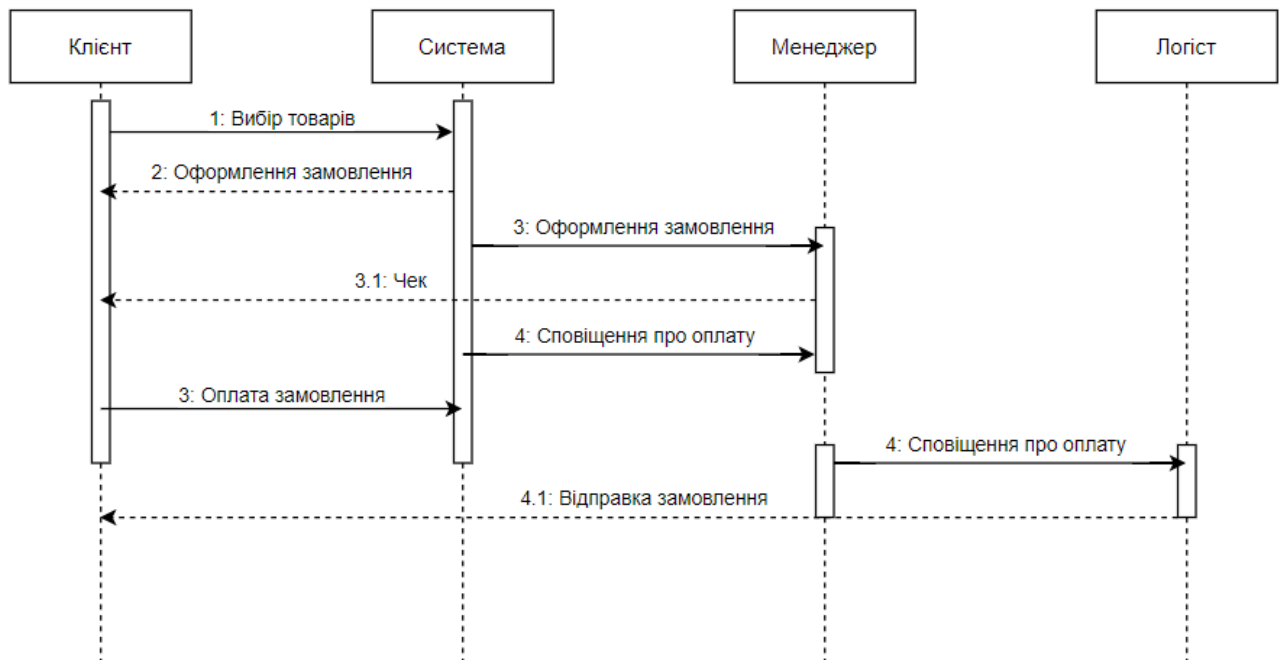


Рисунок 3.19 – Діаграма послідовності «Продаж товарів»

3.2 Висновки за розділом 3

Для програмної реалізації автоматизованої системи складського обліку було проаналізовано сучасні засоби для веб-розробки на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра. Було виявлені необхідні засоби. Отже, для розробки практичної частини роботи обрано:

- 1) мову програмування PHP;
- 2) CMS Joomla!;
- 3) Клієнт-серверну архітектуру.

Проаналізувавши обрану клієнт-серверну архітектуру можна зробити висновок: застосування, що розроблюється є надійним, зручним у використанні, має відкриту структуру, що відкриває широкі можливості по адаптації та масштабуванню.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосування

Тема обрана для проектування «Програмна реалізація автоматизованої системи для складського обліку «Склад2.0».

Для вирішення цілого комплексу завдань зі зберігання даних, забезпечення паралельного доступу до них, відстеження послідовності від моменту замовлення товарів до його видачі зі складу, для захисту цілісності цих даних і буде розроблятися дана система.

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;
- можливість оформлення замовлення;
- можливість оплати готівковим/безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення.

Система працює в режимі реального часу. Запуск програми займає не більше хвилини. Автозбереження здійснюється за допомогою додаткового плагіна «Autosave».

4.2 Умови виконання програми

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- серверна ОС;
- встановлений та розгорнутий веб-сервер;

- інтерпретатор PHP5 та PHP7;
- СКБД MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК із встановленою будь-якою користувацькою ОС;
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;
- встановлений веб-браузер.

4.3 Інструкція по експлуатації програми

4.3.1Звернення до програми

Для звернення до системи необхідно запустити сервер за допомогою іконки Start XAMPP, відкрити браузер, ввести адресу сайту та за необхідністю зареєструватися або авторизуватися на головній сторінці. Після відкриття доступу до системи, можливо переходити за посиланнями на головній сторінці веб-застосунку для виконання відповідних функцій.

4.3.2Вхідні й вихідні дані

Програма не має конкретних вхідних даних для обробки, і ключем до доступу до роботи з нею є комбінація з логіна і пароля користувача чи адміністратора.

Вихідною інформацією є дані, що хоче отримати користувач від системи (наприклад, власний профіль, список покупок, результати пошуку та ін..), систематизована в таблицях бази даних з можливістю фільтрування за допомогою SQL-запитів для представлення користувачеві в табличній формі на сторінках веб-застосунку.

4.4 Методика та результати тестування

На рисунку 4.1 зображено форму авторизації, яку буде бачити користувач при авторизації, а на рисунку 4.2 – при реєстрації.

Склад2.0

Авторизація

Логін

Пароль

Увійти

Зареєструватися

Рисунок 4.1 – Сторінка Авторизації

Склад2.0

Ім'я

Прізвище

Логін

Email

Пароль

Підтвердження паролю

Реєстрація

Рисунок 4.2 – Сторінка Реєстрації

Рисунок 4.3 відображає каталог продукції. На якій розміщується перелік існуючих продуктів компанії, а також інформація про новинки та акції, що проводяться в певний період часу.



Рисунок 4.3 – Каталог, що представлений на складі

На рисунку 4.4 показана форма для оформлення замовлення, а на рисунку 4.5 форма для друку чека для клієнта.

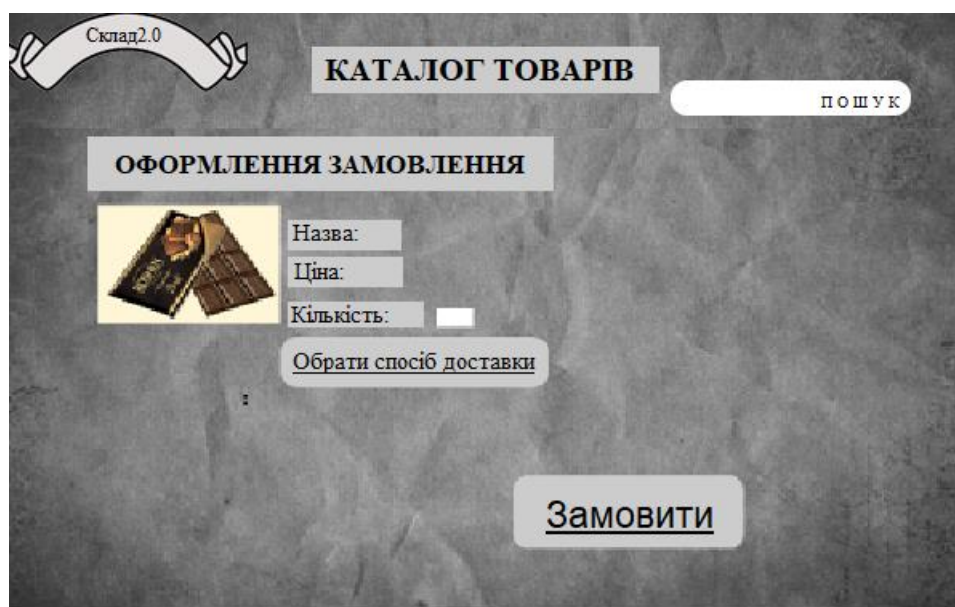


Рисунок 4.4 – Сторінка оформлення замовлення

Склад2.0

ЧЕК

пошук

№ Замовлення

ПІБ Клієнта

Дата

№	Найменування товару	Ціна	Кіл-ть

Усього:

Раздрукувати чек

Сповістити логіста

Рисунок 4.5 – Сторінка із сформованим чеком

Сповіщення логіста або менеджера здійснюється шляхом розсилки поштових повідомлень, рисунок 4.6.

Склад2.0

Email

▼

№ Замовлення
ПІБ клієнта

№ Замовлення
ПІБ клієнта

№ Замовлення
ПІБ клієнта

Рисунок 4.6 – Сторінка сповіщень для логіста

Форма для роботи зі звітами та їх різними видами представлена на рисунку 4.7.

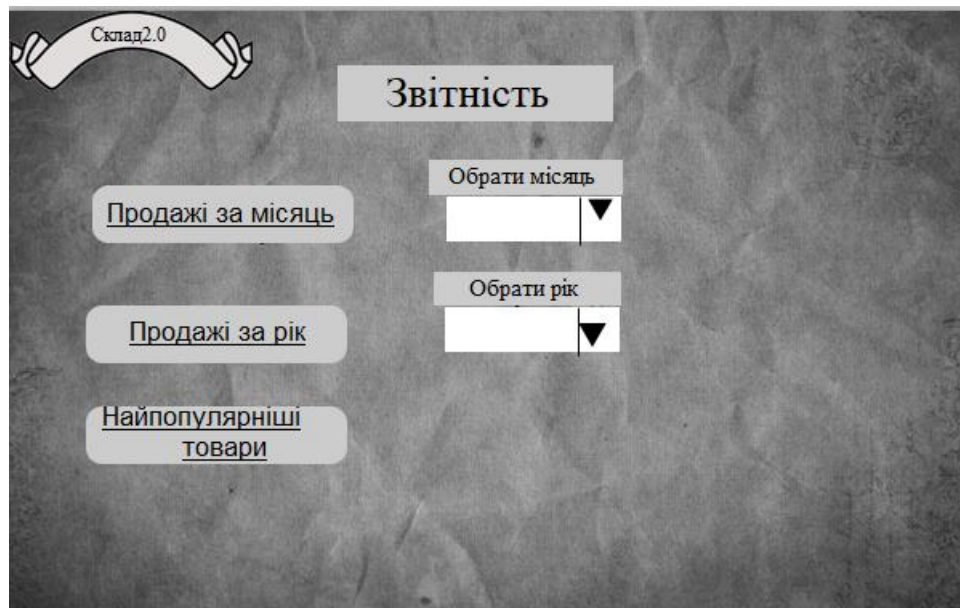


Рисунок 4.7 – Сторінка Звітності

ВИСНОВКИ

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проведено огляд та аналіз методів розробки та підтримки автоматизованих систем складського обліку. Відібрані матеріали були систематизовані, проаналізовані, узагальнені та надані у першому розділі пояснювальної записки.

Для цього було виконано наступні завдання:

- розглянуто проблемну область;
- проаналізовано сучасні найбільш популярні аналоги забезпечення автоматизації складського обліку та аудиту;
- виконано проектування веб-застосунку;
- розроблено веб-застосунок;
- розроблено сценарій та проведено тестування.

Для програмної реалізації автоматизованої системи складського обліку у вигляді веб-застосунку було проаналізовано сучасні засоби для проектування та веб-застосунків на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра. Було виявлені необхідні засоби. Отже, для розробки практичної було використано: мову програмування PHP, CMS Joomla!; систему керування БД MySQL для проектування та розробки бази даних з метою централізованого збереження інформації; веб-сервер XAMPP.

В результаті виконання завдання було отримано веб-застосунок, що відповідає всім вимогам представленим в технічному завданні, зручний для використання та виконує всі необхідні функції, для автоматизації складського обліку. Застосунок виконує наступні функції:

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;

- можливість оформлення замовлення;
- можливість оплати готівковим / безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення;
- можливість друку.

В результаті виконання завдання було отримано систему, що відповідає всім вимогам представленим в технічному завданні, зручна для використання та виконує всі необхідні функції.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бурдюг Н. В. Складський облік запасів: актуальні напрями вдосконалення [Текст] / Н. В. Бурдюг // Збірник наукових праць Черкаського державного технологічного університету. Серія : Економічні науки. – 2010. – Вип. 25(1). – С. 111–114.
2. Жук В. М. Інститут аудиту: сутність та його складові [Текст] / В. М. Жук, К. П. Мельник // Економіка АПК. – 2020. – № 11. – С. 60–73.
3. Гладій І. О. Особливості обліку матеріальних витрат підприємства складського господарства [Текст] / І. О. Гладій // Вісник соціально–економічних досліджень. – 2012. – Вип. 2. – С. 52–57.
4. Харченко Ю. А. Удосконалення інформаційної системи обліку продажів супутніх товарів та наданих послуг в мережах автозаправних комплексів [Текст] / Ю. А. Харченко // Економічний простір. – 2019. – № 143. – С. 183–194.
5. Гринюк С. В. Автоматизована інформаційна система обліку та руху товарів на складі [Текст] / С. В. Гринюк, О. І. Міскевич // Комп'ютерно–інтегровані технології: освіта, наука, виробництво. – 2015. – № 21. – С. 32–36
6. Букало Н. А. Автоматизація обліку в сучасних умовах [Текст] / Н. А. Букало // Збірник наукових праць Черкаського державного технологічного університету. Серія : Економічні науки. – 2013. – Вип. 33(2). – С. 56–59.
7. Бойчук А. А. Автоматизація обліку на підприємствах України [Текст] / А. А. Бойчук, А. М. Мельник // Наука й економіка. – 2016. – Вип. 2. – С. 22–25.
8. Костецький О. Автоматизація обліку та аналізу масових платежів внесків, податків і зборів [Текст] / О. Костецький // Вісник Львівського національного аграрного університету. Серія : Економіка АПК. – 2014. – № 21(1). – С. 326–332.
9. Google Таблиці / [Електронне джерело] – Режим доступу: <https://docs.google.com › spreadsheets › create>

10. Система автоматизації складського обліку «ABM WMS» / [Електронне джерело] – Режим доступу: <https://abmcloud.com/abm-soft/wms/>
11. Програма «Класс365» / [Електронне джерело] – Режим доступу: <https://class365.ru/>
12. Матвієнко О. Вимоги до функціональних можливостей програмних засобів (для бібліотечних процесів) [Текст] / О. Матвієнко // Бібліотечний вісник. – 1996. – № 4. – С. 8–9.
13. Kruglik V. Information system of software distribution / V. Kruglik // Інформ. технології в освіті : зб. наук. пр. – 2011. – Вип. 9. – С. 169–174.
14. Муравський В. Автоматизація інвентаризації в комп'ютерно-комунікаційній формі обліку [Текст] / В. Муравський // Вісник Тернопільського національного економічного університету. – 2017. – Вип. 3. – С. 107–116.
15. CMS або Фреймворк: що обрати для інтернет-магазину / [Електронне джерело] – Режим доступу: <https://turumburum.ua/blog/cms-ili-framework-cho-vybrat-dlya-internet-magazina/>
16. CMS або Фреймворк: що обрати для інтернет-магазину / [Електронне джерело] – Режим доступу: <https://top.studio/stati/cms-ili-framework-cho-vybrat-dlja-internet-magazina/>
17. Joomla! / [Електронне джерело] – Режим доступу: <https://downloads.joomla.org/ru/>
18. Левус Є. В. Аналіз ефективності аспектно-орієнтованої реалізації для забезпечення супроводу системи клієнт-серверної архітектури [Текст] / Є. В. Левус, М. І. Шалак, О. Я. Вітоль // Вісник Національного університету "Львівська політехніка". Серія : Інформаційні системи та мережі. – 2015. – № 829. – С. 177–186.
19. Горбенко Ю. І. Сучасні проблеми централізованих технологій типу «клієнт – сервер» та можливості їх удосконалення на основі децентралізації [Текст] / Ю. І. Горбенко, М. В. Єсіна, Д. В. Мялковський, О.

С. Акользіна, В. А. Пономарь // Радиотехника. – 2019. – Вып. 198. – С. 131–145.

20. Императивное программирование и объектно–ориентированное моделирование: Java, UML, OCL [Текст] : учеб. пособие для студентов высш. учеб. заведений / А. Ф. Верлань [и др.]. – Одесса : Экология, 2013. – 430 с.

21. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем [Текст] : навч. посіб. / С. Д. Постіл ; Ун–т держ. фіск. служби України. – Ірпінь : Ун–т держ. фіск. служби України, 2019. – 321 с.

ДОДАТОК А
Технічне завдання

Вступ

Назва програми – Програмна реалізація автоматизованої системи для складського обліку «Склад2.0». Система призначена для збільшення прибутку компанії, що займається продажем кондитерських виробів, тобто для максимального збільшення обсягів продажу.

A.1 Підстава для розробки

Підставою для розробки слугує завдання на випускну роботу на тему «Програмна реалізація автоматизованої системи для складського обліку», затверджене наказом Національного університету «Запорізька політехніка» № 103 від 30 березня 2021 р.

A.2 Призначення розробки

Завданням автоматизованої системи складського обліку є вирішення завдання в більш швидкому просуванні товарів на ринок, збільшення обсягу послуг, що надаються клієнтам, більш оперативному відстеженні тенденцій розвитку ринку, загальному зменшенні витрат. Системою зможуть користуватися тільки зареєстровані відвідувачі інтернет-магазину.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

ПП повинен виконувати наступні функції:

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;

- можливість оформлення замовлення;
- можливість оплати готівковим / безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення;
- можливість друку.

A.3.2 Вимоги до надійності

Вся інформація про користувачів системи має зберігатися в окремій БД, з функцією періодичного авто зберігання та резервного копіювання.

A.3.3 Умови експлуатації

Програма може зберігатися та переноситися на будь-яких існуючих носіях інформації.

Експлуатація програми здійснюється згідно експлуатаційної документації на розроблену програму, яка відповідає стандартам і містить інформацію, необхідну для її освоєння та експлуатації.

A.3.4 Вимоги до складу та параметрів технічний засобів

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- серверна ОС;
- встановлений та розгорнутий веб-сервер;
- інтерпретатор PHP5 та PHP7;
- СКБД MySQL.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК із встановленою будь-якою користувацькою ОС;
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;
- встановлений веб-браузер.

A.3.5 Вимоги до маркування і пакування

Програма може бути записана на переносний носій або розповсюджуватися через хмарні сховища. На впакуванні повинна бути назва програми – «Автоматизована система складського обліку «Склад2.0».

A.3.6 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання аналогічні вимогам, пропонованим до носія, на якому записана програма.

A.3.7 Вимоги до програмної документації

Програмний продукт повинен поставлятися у наступному складі:

- технічне завдання;
- опис програми;
- текст програми.

A.4 Стадії та етапи розробки

Виділяють такі етапи в розробці програмного забезпечення:

а) збирання та аналіз вимог. Проводиться збір вимог до програмного забезпечення, їх систематизація, документування, аналіз, виявлення протиріч, неповноти, рішення конфліктів в процесі розробки програмного забезпечення;

б) розробка ТЗ. На основі раніше зібраних вимог формується вихідний документ (ТЗ) на проектування програмного забезпечення. Описується всі вимоги до створюваного продукту;

в) створення логотипу та дизайн основних сторінок;

г) налаштування БД;

д) розробка структури. Дозволяє оцінити інформативність та компоновку елементів системи на початковому етапі розробки;

е) підключення адміністративного інтерфейсу. Має містити повне управління настройками системи, окремих модулів та контентом сайту;

ж) робота над програмним кодом. Розробка основного функціоналу системи;

з) наповнення та тестування. Професійні тестувальники знаходять «баги» у системі та фіксують їх. Заповнюється базовий «контент», щоб сайт мав привабливіший вигляд для перших користувачів;

и) розміщення та відкритий доступ, технічна підтримка. Соціальна мережа розміщується на сайті унтернет-магазину та стає доступною для всіх користувачів;

к) оформлення документації.

A.5 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до складу та змісту робіт, наведених у Технічному завданні, і відповідно до умов контракту.

Звіт про проведenu роботу направляється Замовнику на паперовому носії в 2-х екземплярах, підписаних належним чином, а також на електронному носії.

Вимоги до оформлення проміжних звітів і підсумкового звіту узгоджуються з Виконавцем при підписанні контракту.

Одночасно з підсумковим звітом про виконання робіт Виконавець надає Замовнику підписаний зі сторони Виконавця Акт здачі-приймання виконаної роботи.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Назва програмного продукту – «Автоматизована система складського обліку «Склад2.0».

ПЗ було розроблено з використанням мови програмування PHP та CMS Joomla!.

Б.2 Функціональне призначення

ПП повинен виконувати наступні функції:

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;
- можливість оформлення замовлення;
- можливість оплати готівковим / безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення;
- можливість друку.

Б.3 Опис логічної структури

Для побудови архітектурного каркаса додатки використовують схему проектування Model-view-controller (MVC). Це схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, користувацький інтерфейс і взаємодія з користувачем розділені на три окремих компонента.

Логіка відображення – відповідає за збір даних від користувачів в різних формах і обробку їх відповідно до специфікації.

Логіка управління даними – обробляє дані, отримані від користувача для передачі на наступний рівень.

Бізнес-логіка-сукупність правил, принципів, залежностей поведінки об'єктів предметної області. Це реалізація предметної області в інформаційній системі.

Особливості розробки архітектури веб-застосунки на основі предметної області:

- можливість більш докладного опису логіки взаємодії об'єктів предметної області;
- можливість гнучкої зміни логіки роботи програми;
- відсутність надлишкового коду в кінцевій реалізації;
- поділ логіки відображення, логіки управління даними та бізнес-логіки.

Недоліки розробки архітектури веб-застосунки на основі предметної області:

- велика трудомісткість при створенні проекту;
- більш жорсткі вимоги до кваліфікації розробника.

На рисунку Б.1 показано, як виглядає архітектура веб-застосунку системи «Склад2.0» на основі предметної області.

Б.4 Використані технічні засоби

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- ОС – будь-яка з лінійки Windows Server;
- веб-сервер – HTTP-сервер Apache для Unix і Windows, сервер на кшталт XAMPP;
- інтерпретатор PHP5 та PHP7;

– MySQL-сервер із правами створення та вилучення користувачів БД (або база SQLite з аналогічними правами).

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- ПК із встановленою будь-якою користувацькою ОС;
- процесор із тактовою частотою від 1,5 GHz;
- обсяг оперативної пам'яті – не менше 3GB;
- підключення до мережі Internet;
- встановлений веб-браузер.

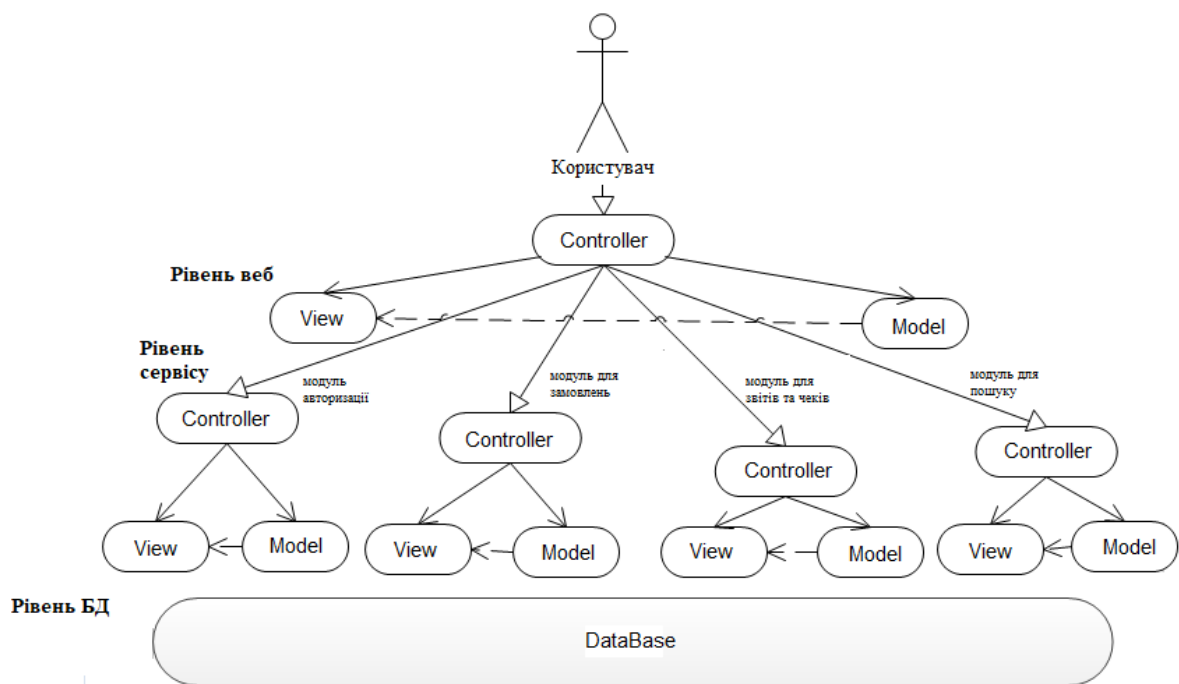


Рисунок Б.1 – Схема системи

Б.5 Виклик і завантаження

На наданому для хостингу веб-вузла на MySQL-сервері необхідно створити БД.

Необхідно отримати логін і пароль користувача БД з правами адміністратора, дисковий простір для БД, вказати адресу головної сторінки графічного інтерфейсу користувача index.php.

Використовуючи phpMyAdmin «Експорт», вибравши шлях до файлу дампу БД і вказавши кодування, перенести БД на новий сервер, або скопіювати папку з базою даних у директорію розміщення даних баз даних MySQL-сервера.

Для завантаження сайту і взаємодії з ним достатньо вказати в адресному рядку браузера sklad.ua.

Б.6 Вхідні та вихідні дані

Вхідними даними є інформація про користувача: логін та пароль людини, що запрошує доступ до системи. Вхідна інформація зберігається у текстовій формі в таблицях бази даних на сервері. Також до вхідної інформації можна віднести всі дані, що відносяться до певного користувача.

Вихідною інформацією є дані, що хоче отримати користувач від системи (наприклад, власний профіль, замовлення, результати пошуку та ін.), систематизована в таблицях бази даних з можливістю фільтрування за допомогою SQL-запитів для представлення користувачеві в табличній формі на сторінках веб-застосунку.

ДОДАТОК В
Текст програми

```

<?php

if (!defined('HOMEPAHT')) {
    exit('Errrrr!!! There NO direct script accss allowed');
}

class Library_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }

    public function getList() {
        $query = $this->db->query("SELECT * FROM items
WHERE deleted='0' ORDER BY name ASC");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                $items[] = $item;
            }

        }

        return $items;
    }

    public function getPackList() {
        $query = $this->db->query("SELECT * FROM Pack
WHERE deleted='0' ORDER BY name ASC");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                $items[] = $item;
            }

        }

        return $items;
    }

    public function additem($name, $info) {

```

```

        $this->db->query("INSERT INTO items (name,
information) VALUES ('$name', '$info')");
    }

    public function updateitem($id, $name, $info) {
        $this->db->query("UPDATE items SET name='$name',
information='$info' WHERE id='$id'");
    }

    public function getitem($id) {
        $query = $this->db->query("SELECT * FROM items
WHERE id='$id'");
        $item = array();
        if ($query->num_rows() > 0) {
            foreach ($query->result() as $row) {
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
            }
        }

        return $item;
    }

    public function delitem($id) {
        $this->db->query("UPDATE items SET deleted='1'
WHERE id='$id'");
    }

    public function getinformations() {
        $query = $this->db->query("SELECT DISTINCT
information FROM items WHERE deleted=0");
        $items = array();
        if ($query->num_rows() > 0) {
            foreach ($query->result() as $row) {
                $item['info'] = $row->information;
                $items[] = $item;
            }
        }

        return $items;
    }
}

<?php

if (!defined('HOMEPATH')) {
    exit('Errrrr!!! There NO direct script access allowed');
}

class Packing_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }
}

```

```

    }

    public function getList() {
        $query = $this->db->query("SELECT * FROM Pack
WHERE deleted='0'");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                if ($this->checkPackingNull($row->id)) {
                    $item['null'] = 1;
                } else {
                    $item['null'] = 0;
                }
                $items[] = $item;
            }
        }

        return $items;
    }

    public function addPacking($name, $info) {
        $this->db->query("INSERT INTO Pack (name,
information) VALUES ('$name', '$info')");
    }

    public function updatePacking($id, $name, $info) {
        $this->db->query("UPDATE Pack SET name='$name',
information='$info' WHERE id='$id'");
    }

    public function getPacking($id) {
        $query = $this->db->query("SELECT * FROM Pack
WHERE id='$id'");
        $item = array();
        if ($query->num_rows() > 0) {
            foreach ($query->result() as $row) {
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
            }
        }

        return $item;
    }

    public function delPacking($id) {

```

```

        if ($this->checkPackingNull($id)) {
            $this->db->query("UPDATE Pack SET deleted='1'
WHERE id='$id'");
            return true;
        } else {
            return false;
        }
    }

    public function checkPackingNull($id) {
        $q = "SELECT "
            . "items.id AS id, "
            . "store.sum AS store_sum, "
            . "itemion.sum AS itemion_sum, "
            . "onway.sum AS onway_sum "
            . "FROM items "
            . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum, id_packing FROM `log` WHERE (action=2 OR
action=3 OR action=4 OR action=6) AND id_packing=$id "
            . "GROUP BY 1 ORDER BY 2) AS store "
            . "ON items.id=store.id_item "
            . "LEFT JOIN (SELECT id_item,
id_packing, sum(amount) AS sum FROM `log` WHERE (action=5 OR
action=6) AND id_packing=$id "
            . "GROUP BY 1) AS onway "
            . "ON items.id=onway.id_item "
            . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum, id_packing FROM `log` WHERE (action=1) AND
id_packing=$id "
            . "GROUP BY 1 ORDER BY 2) AS itemion "
            . "ON items.id=itemion.id_item "
            . "WHERE items.deleted='0' ";

        $query = $this->db->query($q);

        $items = array();

        if ($query->num_rows() > 0) {
            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['store_sum'] = intval($row-
>store_sum);
                $item['onway_sum'] = intval(0 - $row-
>onway_sum);
                $item['itemion_sum'] = intval($row-
>itemion_sum);
                $items[] = $item;
            }
        }

        $totals = array();

```

```

        foreach ($items as $item) {
            // $totals[$item['id']] = $item['store_sum'] +
            $item['itemion_sum'] - $item['onway_sum'];
            if ($item['onway_sum'] != 0) {
                return false;
            }
        }
        return true;
    }
}
<?php

if (!defined('HOMEPATH')) {
    exit('Errrrr!!! There NO direct script access allowed');
}

class Order_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }

    public function getList() {
        $query = $this->db->query("SELECT * FROM orders
WHERE deleted='0'");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                $items[] = $item;
            }
        }

        return $items;
    }

    public function addOrder($name, $info) {
        $this->db->query("INSERT INTO orders (name,
information) VALUES ('$name', '$info')");
    }

    public function updateOrder($id, $name, $info) {
        $this->db->query("UPDATE orders SET name='$name',
information='$info' WHERE id='$id'");
    }
}

```

```

        public function getOrder($id) {
            $query = $this->db->query("SELECT * FROM orders
WHERE id='$id'");
            $item = array();
            if ($query->num_rows() > 0) {
                foreach ($query->result() as $row) {
                    $item['id'] = $row->id;
                    $item['name'] = $row->name;
                    $item['info'] = $row->information;
                }
            }

            return $item;
        }

        public function delOrder($id) {
            $this->db->query("UPDATE orders SET deleted='1'
WHERE id='$id'");
        }
    }
<?php

if (!defined('HOMEPATH')) {
    exit('Errrrr!!! There NO direct script access allowed');
}

class Library_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }

    public function getList() {
        $query = $this->db->query("SELECT * FROM items
WHERE deleted='0' ORDER BY name ASC");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                $items[] = $item;
            }
        }

        return $items;
    }
}

```

```

        public function getPackList() {
            $query = $this->db->query("SELECT * FROM Pack
WHERE deleted='0' ORDER BY name ASC");

            $items = array();

            if ($query->num_rows() > 0) {

                foreach ($query->result() as $row) {
                    $item = array();
                    $item['id'] = $row->id;
                    $item['name'] = $row->name;
                    $item['info'] = $row->information;
                    $items[] = $item;
                }
            }

            return $items;
        }

        public function additem($name, $info) {
            $this->db->query("INSERT INTO items (name,
information) VALUES ('$name', '$info')");
        }

        public function updateitem($id, $name, $info) {
            $this->db->query("UPDATE items SET name='$name',
information='$info' WHERE id='$id'");
        }

        public function getitem($id) {
            $query = $this->db->query("SELECT * FROM items
WHERE id='$id'");
            $item = array();
            if ($query->num_rows() > 0) {
                foreach ($query->result() as $row) {
                    $item['id'] = $row->id;
                    $item['name'] = $row->name;
                    $item['info'] = $row->information;
                }
            }

            return $item;
        }

        public function delitem($id) {
            $this->db->query("UPDATE items SET deleted='1'
WHERE id='$id'");
        }

        public function getinformations() {
            $query = $this->db->query("SELECT DISTINCT
information FROM items WHERE deleted=0");

```

```

        $items = array();
        if ($query->num_rows() > 0) {
            foreach ($query->result() as $row) {
                $item['info'] = $row->information;
                $items[] = $item;
            }
        }
        return $items;
    }

}
<?php

if (!defined('HOMEPATH')) {
    exit('No direct script access allowed');
}

class Log_model extends CI_Model {

    function __construct() {
        parent::__construct();
    }

    public function getList($group) {
        $query = $this->db->query("SELECT "
            . "items.id AS id, "
            . "items.name AS name, "
            . "items.information AS information, "
            . "counter.sum AS sum "
            . "FROM items LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE action=$group OR action=5
GROUP BY 1 ORDER BY 2) AS counter "
            . "ON items.id=counter.id_item WHERE
items.deleted='0' ORDER BY name ASC");

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                if ($row->sum != '') {
                    $item['sum'] = $row->sum;
                } else {
                    $item['sum'] = 0;
                }
                $items[] = $item;
            }
        }
    }
}

```

```

        return $items;
    }

    // $ngroup is negation group (the one with minuses)
    public function getOrderList($group, $ngroup = 5) {
        $q = "SELECT "
            . "items.id AS id, "
            . "items.name AS name, "
            . "items.information AS information, "
            . "counter.sum AS sum "
            . "FROM items "
            . "JOIN (SELECT id_item, id_order,
sum(amount) AS sum, max(id) AS id_log FROM `log` WHERE
action=$group OR action=$ngroup GROUP BY 1) AS counter "
            . "ON items.id=counter.id_item "
            . "WHERE items.deleted='0' AND
counter.sum<>0 ORDER BY id_log info";

        $query = $this->db->query($q);

        $items = array();

        if ($query->num_rows() > 0) {

            foreach ($query->result() as $row) {
                $item = array();
                $item['id'] = $row->id;
                $item['name'] = $row->name;
                $item['info'] = $row->information;
                if ($row->sum != '') {
                    $item['sum'] = $row->sum;
                } else {
                    $item['sum'] = 0;
                }
                $items[] = $item;
            }
        }

        return $items;
    }

    public function getPackingList($group, $ngroup = 0) {

        $q = "SELECT "
            . "items.id AS id, "
            . "items.name AS name, "
            . "items.information AS information, "
            . "packings.id AS packingid, "
            . "packings.name AS packing, "
            . "counter.sum AS sum "
            . "FROM items "

```

```

        . "JOIN (SELECT id_item, id_packing,
sum(amount) AS sum, max(id) AS id_log FROM `log` WHERE
action=$group OR action=$ngroup GROUP BY 1, 2) AS counter "
        . "ON items.id=counter.id_item "
        . "JOIN packings "
        . "ON counter.id_packing = packings.id "
        . "WHERE items.deleted='0' AND
counter.sum<>0 ORDER BY id_log info";

```

```
$query = $this->db->query($q);
```

```
$items = array();
```

```
if ($query->num_rows() > 0) {
```

```

    foreach ($query->result() as $row) {
        $item = array();
        $item['id'] = $row->id;
        $item['name'] = $row->name;
        $item['info'] = $row->information;
        if ($row->sum != '') {
            $item['sum'] = $row->sum;
        } else {
            $item['sum'] = 0;
        }
        $item['packingid'] = $row->packingid;
        $item['packing'] = $row->packing;
        $items[] = $item;
    }
}

```

```
return $items;
```

```
}
```

```

public function addAction($id_item, $amount, $action,
$id_packing = 0, $id_order = 0) {
    $id_user = $this->session->userdata('user_id');
    $this->db->query("INSERT INTO log (id_user,
action, amount, id_item, id_packing, id_order) "
        . "VALUES ('$id_user', '$action',
'$amount', '$id_item', '$id_packing', '$id_order')");
}

```

```

public function getStatusList() {
    $query = $this->db->query("SELECT "
        . "items.id AS id, "
        . "items.name AS name, "
        . "items.information AS information, "
        . "counter.sum AS sum "
        . "FROM items JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` GROUP BY 1 ORDER BY 2) AS counter
"

```

```

        . "ON items.id=counter.id_item WHERE
items.deleted='0' ORDER BY id");

    $items = array();

    if ($query->num_rows() > 0) {

        foreach ($query->result() as $row) {
            $item = array();
            $item['id'] = $row->id;
            $item['name'] = $row->name;
            $item['info'] = $row->information;
            $item['sum'] = $row->sum;
            $items[] = $item;
        }
    }

    return $items;
}

public function getNewStatusList() {
    $q = "SELECT "
        . "items.id AS id, "
        . "items.name AS name, "
        . "items.information AS information, "
        . "store.sum AS store_sum, "
        . "itemion.sum AS itemion_sum, "
        . "onway.sum AS onway_sum "
        . "FROM items "
        . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE action=2 OR action=3 OR
action=4 OR action=6 "
        . "GROUP BY 1 ORDER BY 2) AS store "
        . "ON items.id=store.id_item "
        . "LEFT JOIN (SELECT id_item,
id_packing, sum(amount) AS sum FROM `log` WHERE action=5 OR
action=6 "
        . "GROUP BY 1) AS onway "
        . "ON items.id=onway.id_item "
        . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE action=1 OR action=5 "
        . "GROUP BY 1 ORDER BY 2) AS itemion "
        . "ON items.id=itemion.id_item "
        . "WHERE items.deleted='0' ORDER BY name
ASC";

    $query = $this->db->query($q);

    $items = array();

    if ($query->num_rows() > 0) {

        foreach ($query->result() as $row) {

```

```

        $item = array();
        $item['id'] = $row->id;
        $item['name'] = $row->name;
        $item['info'] = $row->information;
        $item['store_sum'] = intval($row-
>store_sum);
        $item['onway_sum'] = intval(0 - $row-
>onway_sum);
        $item['itemion_sum'] = intval($row-
>itemion_sum);
        $items[] = $item;
    }
}

return $items;
}

public function getSearchList($start, $end, $item) {

    $starts = date("Y-m-d H:i:s", strtotime($start));
    $ends = date("Y-m-d H:i:s", strtotime($end . ' + 1
day')));

    $qand = "AND (date BETWEEN '$starts' AND '$ends')
";

    $q = "SELECT "
        . "items.id AS id, "
        . "items.name AS name, "
        . "items.information AS information, "
        . "store.sum AS store_sum, "
        . "itemion.sum AS itemion_sum, "
        . "given.sum AS given_sum, "
        . "onway.sum AS onway_sum "
        . "FROM items "
        . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE action=2 OR action=3 OR
action=4 OR action=6 "
        . "GROUP BY 1 ORDER BY 2) AS store "
        . "ON items.id=store.id_item "
        . "LEFT JOIN (SELECT id_item,
id_packing, sum(amount) AS sum FROM `log` WHERE action=5 OR
action=6 "
        . "GROUP BY 1) AS onway "
        . "ON items.id=onway.id_item "
        . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE action=1 OR action=5 "
        . "GROUP BY 1 ORDER BY 2) AS itemion "
        . "ON items.id=itemion.id_item "
        . "LEFT JOIN (SELECT id_item,
sum(amount) AS sum FROM `log` WHERE (action=4) $qand "

```

ДОДАТОК Д
Слайди презентації



Рисунок Д.1 – Слайд 1

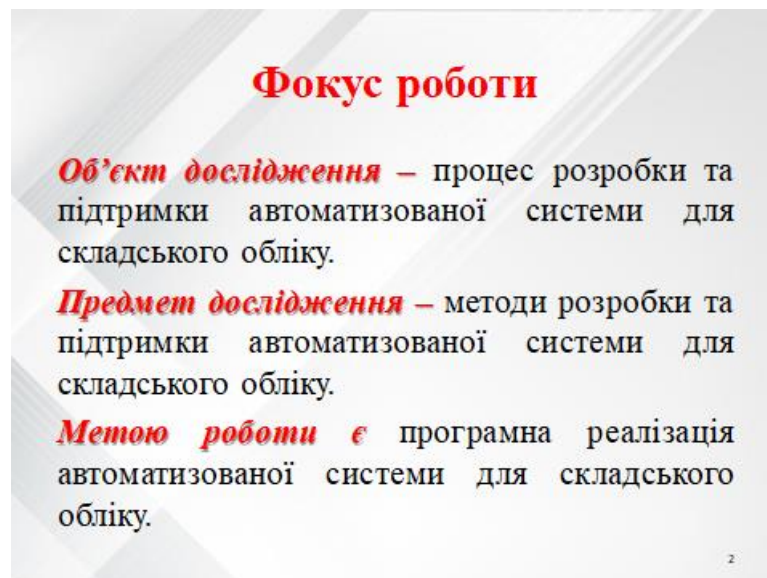


Рисунок Д.2 – Слайд 2

Завдання роботи

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.
Метою є розробка нової автоматизованої системи складського обліку «Склад2.0».

Система має дозволяти:

- можливість реєстрації нових користувачів;
- авторизація користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;
- можливість оформлення замовлення;
- можливість оплати готівковим / безготівковим розрахунком;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення;
- можливість друку.

3

Рисунок Д.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	Google Таблиці	ABM WMS	Класс365	Склад2.0
Засоби для складських обліку та аудиту	Відсутні	Присутні	Присутні	Присутні
Встановлення додаткового ПЗ	Не потрібно	Необхідно	Не потрібно	Не потрібно
Складність підтримки	Низький рівень	Високий рівень	Середній рівень	Середній рівень
Простота інтерфейсів взаємодії	Високий рівень	Середній рівень	Низький рівень	Високий рівень

4

Рисунок Д.4 – Слайд 4

Вибір апаратних платформ для розробки системи

	Фреймворк	CMS
Розробка представлень	+	++
Розробка адміністративної панелі	+	++
Підключення платіжних систем	++	++
Простота та швидкість розробки	+	++
Простота підтримки	+	+

5

Рисунок Д.5 – Слайд 5

Вибір CMS для розробки

	Joomla!	WordPress	OpenCart
Безпека та продуктивність	+	+	+
Можливості самостійного конфігурування	++	++	+
Зручність використання	++	++	+
Вбудовані модулі	++	+	+

6

Рисунок Д.6 – Слайд 6

Архітектурна побудова застосунку

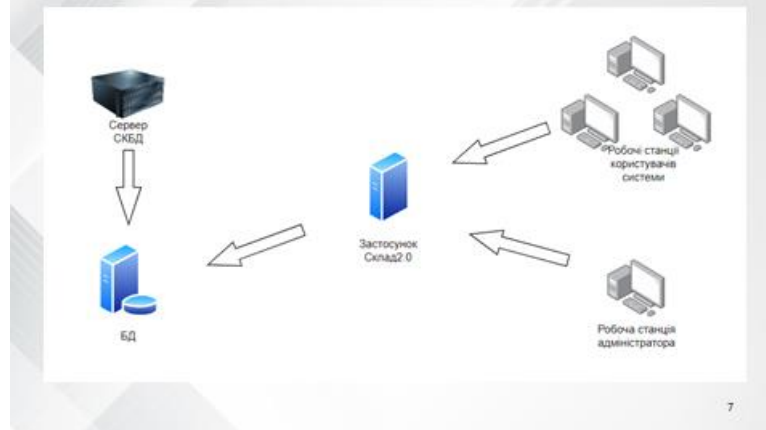


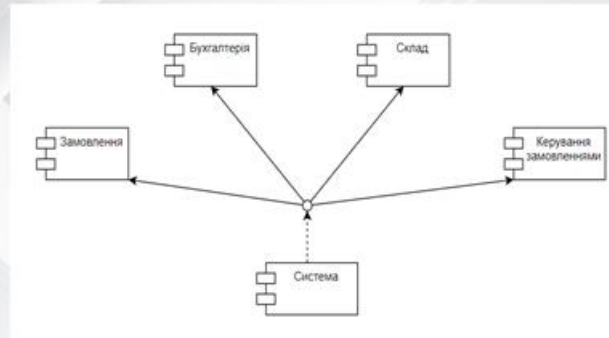
Рисунок Д.7 – Слайд 7

Діаграма класів



Рисунок Д.8 – Слайд 8

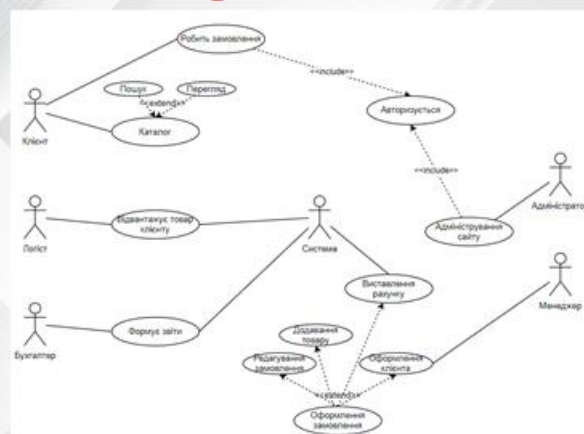
Діаграма компонентів системи



9

Рисунок Д.9 – Слайд 9

Логіка взаємодії – діаграма прецедентів



10

Рисунок Д.10 – Слайд 10



Рисунок Д.11 – Слайд 11



Рисунок Д.12 – Слайд 12

Висновки

В ході виконання даної дипломної кваліфікаційної роботи бакалавра було проведено опіяд та аналіз методів розробки та підтримки автоматизованих систем складського обліку. Відібрані матеріали були систематизовані, проаналізовані та узагальнені.

Для програмної реалізації автоматизованої системи складського обліку у вигляді веб-застосунку було проаналізовано сучасні засоби для проектування та веб-застосунків на основі виявлених особливостей та архітектури застосунку, що розробляється в рамках даної дипломної кваліфікаційної роботи бакалавра.

В результаті виконання завдання було отримано веб-застосунок, що відповідає всім вимогам представленим в технічному завданні:

- можливість реєстрації/авторизації користувачів;
- можливість перегляду і пошуку товарів в каталозі;
- можливість додавання товарів до кошика;
- можливість оформлення замовлення;
- можливість доставки товарів;
- ведення звітності продажів;
- можливість ведення складського обліку;
- можливість дистанційного оповіщення.

Усі завдання випускної дипломної роботи бакалавра повністю виконано.

13

Рисунок Д.13 – Слайд 13