

Детектор Плагіату v. 2961 - Звіт оригінальності: 21.05.2026 11:17:29

Перевірений документ: Кваліфікаційна робота_Мєдведєв А.В..docx Ліцензія: ВОЛОДИМИР МАТІЄВСЬКИЙ_License2

Тип пошуку: Пошук Перепису Знайдено мову: Uk
Тип перевірки: Інтернет-перевірка
ТЕЕ і кодування: DocX n/a

Детальний аналіз документа документа:

Співвідношення:



Графік розподілу:



Джерела плагіату: 7

2%	1212	1.	https://uk.wikipedia.org/wiki/Проектування_бази_даних
1%	901	2.	http://fit.knu.ua/wp-content/uploads/2020/03/Алгоритми-та-методи-обробки-великих-ма...
0,1%	34	3.	https://vstup.osvita.ua/y2024/r14/97/1302952/

Дані оброблених ресурсів: 231 - ОК / 19 - Помилок

Важливі примітки:

Wikipedia:	Google Books:	Сервіси платних робіт:	Античіт:
Знайдена Wiki!	[не знайдено]	[не знайдено]	[не знайдено]

Звіт проти обману UACE:

1. Статус: Аналізатор Увімкнений Нормалізатор Увімкнений схожість символів встановлено 100%
2. Виявлений відсоток забруднення UniCode: 3,8% з обмеженням: 4%
3. Документ не нормалізовано: відсотків не досягнуто 5%
4. Усі підозрілі символи будуть позначені фіолетовим кольором: Abcd...
5. Знайдено невидимі символи: 0
Рекомендація з оцінки: Ніяких особливих дій не потрібно. Документ в порядку.
Статистика алфавіту та аналіз символів:

 Активні посилання (URL-адреси, витягнуті з документа):

URL не знайдені

 Виключено:

URL не знайдені

 Включено:

URL не знайдені

 Детальний аналіз документу:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДЕРЖАВНИЙ ЗАКЛАД

	Знайдено посилань: 0,01%	id: 1
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА”		
	Знайдено плагіату: 0,04%	id: 2
http://fit.knu.ua/wp-content/uploads/2020/03/Алгоритми-та-математики-та-інформатики-Прізвище-Ім'я-По-батькові-НАЗВА-РОБОТИ-кваліфікаційна-робота-здобувача-вищої-освіти-першого-(бакалаврського)-рівня-освітньої-програми		
	Знайдено посилань: 0,01%	id: 3
« Комп'ютерні науки та інформаційні технології »		
	Знайдено плагіату: 0,06%	id: 4
http://fit.knu.ua/wp-content/uploads/2020/03/Алгоритми-та-математики-та-інформатики-Прізвище-Ім'я-По-батькові-НАЗВА-РОБОТИ-кваліфікаційна-робота-здобувача-вищої-освіти-першого-(бакалаврського)-рівня-освітньої-програми		
	Знайдено посилань: 0,01%	id: 5
„Луганський національний університет імені Тараса Шевченка”		
	Знайдено плагіату: 0,03%	id: 6
http://fit.knu.ua/wp-content/uploads/2020/03/Алгоритми-та-математики-та-інформатики-Прізвище-Ім'я-По-батькові-НАЗВА-РОБОТИ-кваліфікаційна-робота-здобувача-вищої-освіти-першого-(бакалаврського)-рівня-освітньої-програми		
	Знайдено посилань: 0,03%	id: 7
“Комп'ютерні науки” (код, назва) Галузь знань 12 “Інформаційні технології” (код, назва) ЗАТВЕРДЖУЮ Завідувач кафедри МТ Ім'я ПРИЗВИЩЕ (підпис)(ім'я, прізвище) “ ”		
202 р. ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ Прізвище Ім'я По батькові (прізвище, ім'я, по батькові) 1. Тема Керівник кваліфікаційної роботи (прізвище, ініціали, науковий ступінь, вчене звання) затверджена наказом по університету від		
	Знайдено посилань: 0%	id: 8
“ ”		
202 року № 2. Строк подання здобувачем вищої освіти проекту 3. Вихідні дані до роботи (проекту) (визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки) 4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) (визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ) Консультанти розділів проекту (роботи) Розділ Прізвище, ініціали та посада консультанта Підпис, дата завдання видав завдання прийняв Дата видачі завдання "" 202_ року .КАЛЕНДАРНИЙ ПЛАН № з/п Назва етапів дипломного проекту (роботи) Строк виконання етапів проекту (роботи) Примітка 1. Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника. до 1 лютого 2. Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень. до 20 березня 3. Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником. до 1 квітня 4. Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання. до 15 квітня 5. Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи. до 30 квітня 6. Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації. до 15 травня 7. Попередній захист роботи на кафедрі до 30 травня 8. Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії за 10 днів до державної атестації 9. Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом. за 5 днів до державної атестації Здобувач вищої освіти Керівник проекту (роботи) підпис Ім'я ПРИЗВИЩЕ (ім'я, прізвище) Ім'я ПРИЗВИЩЕ підпис (ім'я, прізвище) АНОТАЦІЯ Тема: Спеціальність:		

Кваліфікаційна робота містить: с., рис., табл., додат., джерела. Об'єкт дослідження – Предмет дослідження – Мета роботи – Результати роботи. Висновки. Ключові слова. [ABSTRACT](#) [Ivanov, I.](#) [Theme: Specialty: Institution:](#), 202 [Qualification work contains: pages, figures, tables, appendices, sources.](#) [Object of research: Subject of research: Purpose of the study: . Results of research. Conclusions. Keywords.](#) [ALGORITHM, ONTOLOGY, DATA ARRAYS, INTELLECTUAL SYSTEM](#). ЗМІСТ ВСТУП7 1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ8 1.1. Огляд літератури та етапи розвитку наукової думки8 1.2. Аналіз відомих технічних рішень та технологічних процесів9 1.3. Аналіз відомих технологічних процесів12 1.4. Огляд ринку програмних рішень13 1.5. Специфікація вимог до проєктованої системи14 2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ СИСТЕМИ16 2.1. Аналіз предметної області та вибір технологій реалізації16 2.2. Функціональне та статичне моделювання системи21 2.3. Моделювання процесів взаємодії компонентів26 2.4. Проєктування інфологічної та даталогічної моделей бази даних29 2.4.1. Інфологічна модель бази даних29 2.4.2. Даталогічна модель бази даних31 3 РЕАЛІЗАЦІЯ, БЕЗПЕКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ35 3.1. Модель реалізації, якість ПЗ та опис процесів тестування35 3.1.1. Тестування та верифікація можливостей системи35 3.1.2. Інструкція з розгортання та експлуатації системи38 3.2. Аналіз безпеки програмного забезпечення та захист від несанкціонованого доступу до даних40 ВИСНОВОК42 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ43 Додаток44 ВСТУП Актуальність теми. Сучасний етап розвитку будівельної індустрії, характеризується стрімким розширенням номенклатури матеріальних ресурсів. Разом із цим, суттєво підвищуються вимоги до швидкості складських процесів. Ефективне управління запасами, вимагає безперервного та точного контролю за рухом усіх матеріалів. Проте використання застарілих або ручних методів обліку часто призводить до серйозних логістичних помилок.

 AI generated text detected: ChatGPT 5.3, : 63,96%

id: 9

Через це виникають проблеми з дефіцитом товарів, що тягне за собою простой на майданчиках та фінансові втрати. Аналіз існуючих великих [ERP](#)-системами, таких як 1С або [SAP](#), показує їхню високу вартість та складність в інтеграції. На противагу їм, розробка локальних полегшених систем, є значно вигіднішою та простішою. Такі системи не потребують високих апаратних потужностей від серверного обладнання підприємства. Саме тому, створення гнучкої клієнт-серверної автоматизованої системи, на базі мови [PHP](#) та [MySQL](#), є раціональним рішенням. Система забезпечує моніторинг залишків у реальному часі. Мета і завдання дослідження. Метою роботи є створення автоматизованої системи обліку, для ефективного функціонування будівельного підприємства. Проєкт спрямований на підвищення якості складського контролю, автоматизацію руху товарів та оперативне запобігання дефіциту номенклатури. Для досягнення поставленої мети, було вирішено низку науково-практичних завдань. Спочатку виконано детальний аналіз предметної області будівельного складу. Після цього теоретично обґрунтовано вибір клієнт-серверної архітектури та відповідного технологічного стеку розробки. На етапі проєктування побудовано комплекс [UML](#)-діаграм, а також створено концептуальну та фізичну моделі реляційної бази даних. Фінальним кроком стала програмна реалізація модулів системи та їхнє функціональне тестування за методом чорної скриньки. Об'єкт дослідження є процес складського обліку, моніторингу та логістичного контролю за рухом матеріальних ресурсів на будівельному підприємстві. Предмет дослідження є методи, алгоритми та програмні засоби автоматизації обліку номенклатури.

Дослідження також охоплює фіксацію складських транзакцій приходу й видачі та моніторинг критичного рівня дефіциту будівельних матеріалів. Практичне значення отриманих результатів. Результатом роботи став готовий до безпосереднього впровадження автоматизована [autoblik](#). Система має низькі апаратні вимоги, завдяки чому її можна розгорнути на наявному комп'ютерному парку, за допомогою локального сервера [XAMPP](#). Впровадження системи дозволяє повністю автоматизувати робочі місця комірників та адміністраторів. Програма забезпечує швидкий пошук товарів та ведення електронного журналу операцій. Візуальне маркування критичних залишків, мінімізує ймовірність помилок обліку, що суттєво оптимізує щоденну діяльність складу будівельного підприємства. Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків та списку використаних джерел. У першому розділі проведено дослідження предметної області та проаналізовано процеси складського обліку. Другий розділ присвячено обґрунтуванню вибору мови [PHP](#) і реляційної СКБД [MySQL](#) для реалізації проєкту. Також, присвячено комплексному архітектурному проєктуванню

системи, за допомогою діаграм [UML](#). Додатково у ньому представлено концептуальну та фізичну схеми бази даних, розгорнутої в [phpMyAdmin](#). У третьому розділі детально описано програмну реалізацію модулів та наведено інструкцію з їхнього розгортання. Окрему увагу приділено аналізу безпеки коду, захисту від [SQL](#)-ін'єкцій та функціональному тестуванню.

АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ Огляд літератури та етапи розвитку наукової думки Поняття системи автоматизації обліку матеріальних ресурсів, поєднує програмні засоби, бази даних та методики, що забезпечують безперервний збір, реєстрацію, обробку та передачу інформації про наявність і рух ресурсів усередині підприємства. С. В. Івахненко, автор книжки

” Знайдено посилань: **0,01%**

id: **10**

«Інформаційні технології в організації бухгалтерського обліку та аудиту»

). Ця робота розкриває трансформацію традиційного рахівництва у високотехнологічний процес, де автоматизація виступає, як фундамент для побудови цілісної інформаційної системи підприємства. Система автоматизації обліку матеріальних ресурсів розглядається, як одна з підсистем загальної автоматизованої інформаційної системи підприємства. Особлива увага приділяється зміні архітектури контролю, яка за умов автоматизації стає запобіжною та вшитою безпосередньо в програмні алгоритми. Автор описує методику

” Знайдено посилань: **0%**

id: **11**

“аудиту навколо комп'ютера”

та

” Знайдено посилань: **0%**

id: **12**

“через комп'ютер”,

пояснюючи, як автоматизовані системи забезпечують цілісність даних через розмежування прав доступу, логування дій користувачів та автоматичну верифікацію транзакцій [1]. Впровадження такої системи дозволяє мінімізувати помилки, пов'язані з

” Знайдено посилань: **0%**

id: **13**

"людським фактором",

та значно прискорює проведення інвентаризації завдяки використанню технологій штрих-кодування, якщо система це підтримує. Еволюція наукових підходів до автоматизації обліку, пройшла тривалий шлях. Вона трансформувалася від простої фіксації господарських операцій, до створення складних інтелектуальних систем управління.. На початкових етапах розвитку, наукова думка базувалася на дескриптивно-обліковому підході. Автоматизація розглядалася лише як засіб механізації рутинної праці та підвищення точності цифрових реєстрів, порівняно з паперовими журналами.

 AI generated text detected: ChatGPT 5.3, : 54,07%

id: **14**

Чарльз Бахман та Едгар Кодд, відомі за розробки в галузі мережевих та реляційних моделей даних. Завдяки їх роботам, стало можливо вперше структурувати цифрові записи та впорядковувати реєстри, що стало основою для першої хвилі автоматизації. Згодом наукова парадигма змістилася у бік прогностично-ресурсного підходу. Зумовлено це появою концепцій планування потреб у матеріалах. Джозеф Орліцкі та Джордже Плосслор розглядали матеріальні запаси не як статичні одиниці зберігання, а як динамічні потоки. Вони потребують математичного моделювання та синхронізації з виробничими циклами.

Це дозволило науці вийти за межі простого обліку та перейти до розробки методологій оптимізації, де головною метою стало мінімально достатнє забезпечення підприємства ресурсами для безперебійної роботи. Наступним важливим етапом став стратегічно-інтеграційний підхід. Підхід розширив системи до рівня управління всім підприємством. Лі Вайлі, який у 1990 році ввів у науковий обіг термін [Enterprise Resource Planning](#). Він довів, що автоматизація лише складських чи виробничих процесів є недостатньою, і запропонував модель повної інтеграції ресурсів[2]. Сучасний стан наукової думки характеризується переходом до когнітивно-екосистемного підходу, що базується на принципах Індустрії 4.0. Сьогодні науковий інтерес зосереджений навколо концепції

” Знайдено посилань: **0%**

id: **15**

«цифрових двійників»

та інтелектуальних систем, здатних до самостійного прийняття рішень. Замість жорстких алгоритмів, дослідники пропонують моделі, що використовують штучний інтелект та хмарні технології. Вони прогнозують ризики та адаптацію до ринкових змін у реальному часі [3]. Аналіз відомих технічних рішень та технологічних процесів В сьогодення, існує така система, як [Enterprise Resource Planning](#). Це багатофункціональний комплекс програмного забезпечення, який створює єдине інформаційне середовище для компанії. Він дозволяє автоматизувати та управляти всіма ключовими аспектами бізнесу. Система охоплює широкий спектр процесів: від бухгалтерського обліку та закупівель до складської логістики, управління проектами та ризиками. На відміну від розрізнених програмних рішень, система базується на принципі централізації даних. Завдяки чому, забезпечує миттєвий доступ до актуальних показників для всіх підрозділів компанії в режимі реального часу. Завдяки інтеграції фінансових, людських та виробничих ресурсів у межах однієї платформи[4]. До систем автоматизації, існували системи планування. [Material requirements planning](#) – це процес планування матеріальних потреб підприємства. Він вимагає беззастережної точності, високого контролю та постійних оновлень, а також обліку величезної кількості даних. Будь-яка помилка в плануванні може призвести до збитків: простою потужностей або транспорту, перевантаження складу, втрати замовлення або контракту з клієнтом тощо. Аби планувати управління ресурсами вручну, підприємству потрібен цілий відділ з десятками фахівців. У доцифрову епоху це так і було: менеджери й аналітики витрачали безліч часу, аби отримати приблизні прогнози та хоч трохи знизити ризик помилок та втрат. Система [Material requirements planning II](#), на відміну від минулої системи, дозволяє планувати потреби підприємства не тільки в матеріалах, але у всіх виробничих ресурсах, вести облік замовлень, планувати завантаження виробничих потужностей, виробничі витрати, моделювати хід виробництва, вести його облік, планувати випуск готових виробів, оперативно коригувати плани і виробничі завдання. Необхідність впровадження спеціалізованих технічних рішень для систем автоматизації обліку, зумовлена насамперед критичним зростанням обсягів даних та вимогами до швидкості обробки замовлень, з якими людський чинник вже не здатний впоратися самотужки[5]. У сучасній логістиці будь-яка затримка у передачі інформації між фізичним переміщенням товару та його відображенням у цифровій базі призводить до виникнення помилок, затоварення або, навпаки, дефіциту позицій. Автоматизація, розглядається як інженерна необхідність, що дозволяє перетворити статичне сховище на динамічну екосистему, де кожен об'єкт має свій цифровий двійник[3]. Ефективність такої системи безпосередньо залежить від безшовної синхронізації, де апаратні засоби виступають у ролі сенсорів, а програмне забезпечення – у ролі мозку. Без належного технічного базису автоматизація залишається лише поверхневим оцифруванням паперових журналів, що не вирішує проблему масштабованості. Апаратна складова автоматизації складу базується на технологіях швидкого розпізнавання об'єктів та мобільності персоналу, де ключову роль відіграють засоби ідентифікації. Основним інструментом залишається штрих-кодування у форматах [1D](#) та [2D](#), проте все більшої популярності набуває технологія [RFID](#), яка дозволяє зчитувати дані про сотні одиниць товару одночасно без прямої видимості мітки. Це фізичне обладнання доповнюється терміналами збору даних, що являють собою захищені мобільні комп'ютери зі вбудованими сканерами. Вони стають основним робочим місцем комірника, забезпечуючи передачу інформації в систему в реальному часі через [Wi-Fi](#) мережі. Окрім цього, на великих об'єктах впроваджуються системи [Pick-to-Light](#), де світлові індикатори на стелажах підказують працівнику потрібну комірку, що мінімізує вплив людського фактора на швидкість збору замовлень[4]. Програмний рівень автоматизації зосереджений навколо систем управління складом, відомих як [Warehouse Management System](#) та [Enterprise Resource Planning](#). На відміну від облікових платформ загального призначення, [Warehouse Management System](#) фокусується на операційній логіці та динамічному керуванні простором. Основною функцією такого програмного забезпечення є підтримка адресного зберігання, де кожне місце на складі має свій унікальний цифровий індекс.

 AI generated text detected: ChatGPT 5.3, : 61,8%

id: 16

Програма самостійно розраховує оптимальні маршрути для персоналу, враховуючи габарити товару, частоту його відвантаження та сумісність різних категорій продукції. Важливою частиною софту є алгоритми черговості, які автоматично вибирають для відвантаження товари з найменшим терміном придатності або ті, що надійшли першими, що критично для складів із продуктами харчування чи фармацевтикою. Технічна інтеграція між “залізом” та софтом забезпечується через проміжні інтерфейси та хмарні

обчислення. Сучасні рішення часто використовують [API](#)-сервіси для синхронізації з кур'єрськими службами та маркетплейсами, що дозволяє автоматично генерувати транспортні накладні та оновлювати залишки в інтернет-магазинах у момент сканування товару на складі. На рівні серверної архітектури перевага надається хмарним [SaaS](#)-моделям, які забезпечують високу швидкість обробки великих масивів даних та надійне резервне копіювання.

Це дозволяє керівнику складу бачити аналітику по залишках та продуктивності кожної зміни з будь-якого пристрою, а самій системі – масштабуватися без необхідності закупівлі додаткових локальних серверів під час пікових навантажень[6]. Аналіз відомих технологічних процесів Технологічний процес обліку матеріальних ресурсів на сучасному підприємстві – це складна послідовність взаємопов'язаних операцій. Вони спрямовані на безперервний контроль руху цінностей від моменту їхнього надходження до кінцевого використання або списання. Первинний етап процесу зазвичай розпочинається з оприбуткування. Ця процедура включає фізичне приймання товарів, а також перевірку відповідності їхньої кількості та якості даним у супровідній документації. Після цього отримані ресурси обов'язково реєструються в системі обліку. На цьому кроці критично важливо вносити інформацію максимально оперативно. Будь-яка затримка даних призводить до розбіжностей у залишках, що суттєво ускладнює планування подальших закупівель. Наступний етап охоплює організацію зберігання та внутрішнього переміщення ресурсів у межах складських приміщень. Цей процес вимагає чіткої структуризації простору. В автоматизованих системах така задача ефективно реалізується через механізми адресного зберігання [7]. Паралельно з цим обов'язковим елементом є проведення регулярних інвентаризацій. Вони дозволяють чітко зіставити фактичну наявність матеріалів із поточними обліковими даними. Технологічно цей процес може бути як повним, так і вибіркоvim. Проте його загальна ефективність критично залежить від швидкості обробки результатів. Сучасні системи мінімізують час нарахувань завдяки автоматичному формуванню актів розбіжностей. Заключна стадія технологічного процесу пов'язана з безпосереднім вибуттям матеріальних ресурсів. Зазвичай це відбувається через їх відпуск у виробництво, продаж чи списання у зв'язку з непридатністю. Кожна така операція супроводжується автоматичним перерахунком залишків на складах. Одночасно з цим програма генерує необхідну вихідну документацію. Особлива увага в загальному технологічному циклі приділяється контролю критичних залишків. Програма в реальному часі сигналізує про необхідність поповнення конкретних позицій, це дозволяє уникнути дефіциту сировини та повністю запобігає раптовій зупинці будівельних чи виробничих процесів. Огляд ринку програмних рішень Сучасний ринок програмного забезпечення для обліку ресурсів представлений широким спектром рішень. Зазвичай їх умовно ділять на комплексні [ERP](#)-системи та спеціалізовані локальні додатки. Серед світових лідерів варто відзначити гігантів класу [ERP](#), таких як [SAP](#) або [Oracle](#), що пропонують потужний інструментарій для управління великими корпораціями. Ці продукти характеризуються глибокою інтеграцією всіх бізнес-процесів. Проте висока вартість впровадження, тривале навчання персоналу та серйозні вимоги до технічних потужностей часто роблять їх недоступними для представників малого та середнього бізнесу. Більш гнучкою та фінансово вигідною альтернативою є модульні системи з відкритим кодом, наприклад [Odoo](#). Вони дозволяють компанії обирати лише потрібні для роботи компоненти, вирізняючись при цьому пластичністю налаштувань та зручним веб-інтерфейсом, який забезпечує високу мобільність користувачів [7]. Однак використання подібних платформ часто вимагає додаткових зусиль з боку розробників для їхньої адаптації під специфічні вимоги українського законодавства та стандарти бухгалтерської звітності, що в підсумку може суттєво збільшити кінцеву вартість усього проєкту.

 AI generated text detected: ChatGPT 5.3, : 59,32%

id: 17

На локальному ринку тривалий час домінували продукти сімейства [Business Automation Software](#). Ці продукти максимально адаптовані до вітчизняних стандартів обліку. Вони забезпечують високу точність фінансових розрахунків та автоматичне формування регламентованої звітності. Архітектура цих рішень часто є надто громіздкою для виконання простих складських операцій, а інтерфейс може здаватися перевантаженим для рядового користувача. Це створює нішу для розробки легковажних, вузькоспеціалізованих систем, які фокусуються на швидкодії, інтуїтивності та забезпеченні базових потреб обліку без надлишкового функціоналу. Специфікація вимог до проєктованої системи Процес формування вимог до проєктованої системи базується на

детальному дослідженні виявлених недоліків існуючих рішень та специфіки предметної області. Основний масив функціональних вимог зосереджений на забезпеченні повного циклу обліку матеріальних цінностей. Це включає можливість створення та редагування карток товарів, фіксацію операцій приходу та видатку, а також автоматичне формування звітів про рух і залишки ресурсів.

Важливою вимогою є реалізація механізмів пошуку та фільтрації даних за різними критеріями, що значно прискорює роботу оператора при великих обсягах інформації. Окрім функціональної складової, критичне значення мають нефункціональні вимоги, які визначають якісні характеристики програмного продукту. Однією з пріоритетних вимог є надійність системи, що передбачає збереження цілісності бази даних навіть у випадку раптового збою живлення або обриву зв'язку з сервером. Продуктивність системи повинна забезпечувати мінімальний час відгуку інтерфейсу, адже це є критичним при інтенсивній роботі на складі. Також висувуються суворі вимоги до безпеки даних. Вони реалізуються через систему авторизації та розмежування прав доступу, де кожен користувач має допуск лише до тих функцій, які необхідні для виконання його посадових обов'язків. Специфікація вимог також охоплює ергономічні аспекти, оскільки інтерфейс системи має бути адаптивним для роботи на різних типах пристроїв, включаючи стаціонарні комп'ютери та мобільні термінали. При цьому технічні вимоги до апаратної частини повинні залишатися мінімальними. Це дозволить експлуатувати систему на існуючому обладнанні підприємства без додаткових витрат на його оновлення. Кінцевий перелік вимог формує фундамент для проєктування архітектури бази даних та вибору технологічного стеку розробки, гарантуючи відповідність майбутнього продукту реальним потребам автоматизації.

МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ СИСТЕМИ

Аналіз предметної області та вибір технологій реалізації

Поняття матеріальних ресурсів підприємства є досить широким, адже воно включає в себе все – від звичайної сировини до готової продукції. Щоб просто перевірити роботу створеної системи та наочно показати її можливості. Для практичного приклад було обрано облік будівельних матеріалів. Ця сфера чудово підходить для демонстрації системи. На будівництві матеріали рухаються дуже швидко, постійно йде прихід та видача, є багато різних одиниць виміру, а сама робота компанії напряму залежить від того, чи є потрібні запаси на складі. Аналіз типового складського господарства будівельного спрямування виявив, що процеси обліку часто ведуться за допомогою застарілих паперових журналів або локальних таблиць [MS Excel](#). Такий підхід має низку суттєвих недоліків: низька оперативність внесення даних про прихід та видачу; регулярне виникнення помилок через людський чинник при ручному перерахунку; відсутність автоматичного сповіщення про критичний дефіцит товарів; складність формування зведеної звітності для аналізу витрат матеріалів на конкретні об'єкти.

Впровадження автоматизованої системи розглядається як інженерна необхідність, що дозволяє перетворити статичне сховище на динамічну екосистему. У процесі проєктування такої системи обліку матеріальних ресурсів першочерговим завданням є побудова надійної та безпечної архітектури взаємодії між сховищем даних та серверною логікою. Для реалізації програмного комплексу було обрано клієнт-серверну модель. У ній основне обчислювальне навантаження та перевірка бізнес-правил перенесені на сторону веб-сервера. Таке архітектурне рішення дозволяє повністю ізолювати базу даних від прямого несанкціонованого доступу з боку клієнта. Воно забезпечує контроль за кожною операцією приходу чи видачі облікових одиниць, а також гарантує стабільну роботу при одночасному підключенні кількох користувачів. Середовищем для розгортання та інтеграції серверних компонентів обрано програмний пакет [XAMPP](#). Він об'єднує веб-сервер [Apache](#), інтерпретатор мови програмування та реляційну систему керування базами даних [MySQL](#). Це дозволяє створити стабільну екосистему для комунікації програмного коду та сховища інформації. Для організації сховища інформації автоматизованої системи, було обрано реляційну систему управління базами даних [MySQL](#). Вона є однією з найпопулярніших та найстабільніших систем у світі. Використовує стандартизовану мову структурованих запитів ([SQL](#)) та поширюється з відкритим вихідним кодом. Платформа зарекомендувала себе: високою швидкодією, кросплатформенністю та можливістю безкоштовного розгортання. Вона дозволяє значно знизити загальну вартість проєктування та впровадження програмного забезпечення на підприємстві. У [MySQL](#), є архітектурна перевага для систем такого класу. Суворі підтримка реляційної моделі та механізмів забезпечення цілісності даних на рівні зв'язків, між таблицями за допомогою зовнішніх ключів ([Foreign Keys](#)). Це дозволяє жорстко пов'язати сутності користувачів, матеріалів,

категорій та транзакцій. Така архітектура, повністю виключає ризик виникнення логічних помилок або утворення ізольованих записів. Окрім цього, [MySQL](#) демонструє високу ефективність під час обробки транзакцій та виконання складних операцій об'єднання таблиць, через оператори [LEFT JOIN](#). Метод є критично важливим для миттєвого відображення поточних залишків на складі та швидкого формування зведених аналітичних звітів за будь-які часові періоди. За логіку автоматизованої системи, виступає серверна мова програмування [PHP](#). [PHP](#) – це поширена мова скриптів із відкритим вихідним кодом, яка спеціально спроектована для веб-розробки і виконується на стороні сервера[9]. [PHP](#) має вбудовану архітектурну підтримку бази даних [MySQL](#) через розширення [mysqli](#) та [PHP Data Objects](#). Це забезпечує мінімальний час відгуку системи під час виконання складних транзакційних запитів. Оскільки система передбачає чітке розмежування прав доступу, адміністратор та робітник), вбудований у [PHP](#) механізм сесій ([session_start\(\)](#), [\\$_SESSION](#)) дозволяє безпечно зберігати стан авторизації, ідентифікувати користувача під час переходів між сторінками та блокувати несанкціонований доступ. Скрипти інтерпретуються безпосередньо на сервері під керуванням веб-сервера [Apache](#) або [Nginx](#). Програма може безперешкодно розгортатися як локально, так і на віддалених серверах під керуванням операційних систем сімейства [Linux](#), не вимагаючи дорогого ліцензійного програмного забезпечення. Процедурний та об'єктно-орієнтований підходи в [PHP](#) дозволяють швидко перехоплювати дані з [HTML](#)-форм через глобальні масиви [\\$_POST](#) та [\\$_GET](#), виконувати їх валідацію, захисне екранування від [SQL](#)-ін'єкцій та динамічно генерувати оновлений інтерфейс для користувача. Серверні [PHP](#)-скрипти виступають у ролі контролерів, які перехоплюють запити від клієнта. Вони здійснюють дезінфекцію та екранування вхідних параметрів для запобігання вразливостям типу [SQL](#)-ін'єкцій[8]. Впроваджена модель розмежування прав доступу, захищає цілісність нормативно-довідкової інформації та обмежує дії персоналу складу відповідно до їхніх посадових обов'язків. Обліковий запис адміністратора влаштований так, що він володіє повним і необмеженим інструментарієм. Він дозволяє додавати нові матеріали, редагувати параметри наявних позицій або видаляти застарілі записи. Також адміністратор може наповнювати базові системні довідники новими категоріями ресурсів та одиницями виміру. На противагу цьому, для облікового запису звичайного робітника, функції модифікації, редагування чи видалення матеріалів повністю заблоковані. Це обмеження надійно реалізовано на рівні сесій сервера, що унеможливорює випадкове чи навмисне псування даних. Функціонал робітника чітко обмежений переглядом стандартного списку матеріальних ресурсів для оперативного моніторингу залишків. Він бачить, які товари є в достатній кількості, а які мають статус дефіциту й потребують закупівлі. Робітник також може переглядати загальну історію транзакцій та безпосередньо реєструвати нові операції приходу чи видачі. Для забезпечення безпосередньої взаємодії між серверними скриптами [PHP](#) та системою управління базами даних, використовується спеціалізоване розширення [MySQLi \(MySQL Improved\)](#). У межах розробки було застосовано процедурний стиль цього розширення за допомогою стандартних функцій. Такий підхід дозволив створити просту, лінійну архітектуру програмного коду для всіх сторінок складу, що суттєво спростило процес його написання, полегшило подальше налагодження системи та пошук можливих помилок. Важливою перевагою інтеграції [MySQLi](#) є підвищений рівень безпеки складських даних та оптимізація швидкодії вебдодатка. Завдяки використанню вбудованої функції [mysqli_real_escape_string\(\)](#) програма в автоматичному режимі екранує службові символи у вхідних запитах користувачів, що надійно захищає базу даних підприємства від хакерських атак типу [SQL](#)-ін'єкцій. Крім того, розширення оптимізоване для роботи з сучасними версіями [MySQL](#). Тому, забезпечується мінімальний час відгуку сервера під час фільтрації великої кількості матеріалів та динамічного формування зведених звітів. Розширення [MySQLi](#) ініціює захищене з'єднання на етапі завантаження сторінок і забезпечує трансляцію динамічних [SQL](#)-запитів. Під час фіксації складських рухів бекенд виконує критично важливу валідацію бізнес-логіки. Перед додаванням операції списання [PHP](#)-скрипт запитує з бази даних поточний залишок обраної позиції та порівнює його із наміченою кількістю видачі. У разі виявлення нестачі система блокує транзакцію та виводить інформативне вікно помилки. Якщо перевірка успішна, дані записуються в журнал транзакцій. Після цього на рівні самої бази даних, [MySQL](#) автоматично спрацьовують тригери, які миттєво оновлюють підсумкову кількість наявного ресурсу в картці матеріалу. Це гарантує синхронність даних і повністю виключає появу помилкових від'ємних залишків на складі. Окремим елементом архітектури розробленої системи, є аналітичний модуль генерації зведеної звітності за довільний проміжок часу. Користувач

має можливість обрати через інтерфейс початкову та кінцеву дати. Після цього серверний [PHP](#)-код транслює до бази даних складний оптимізований [SQL](#)-запит. У ньому використовується умовне об'єднання та групування даних за ідентифікаторами ресурсів. Система автоматично аналізує весь масив накопичених транзакцій за вказаний період та самостійно сумує показники. У результаті, виводиться зведена таблиця з двома окремими стовпчиками для кожного матеріалу. Вони відображають сумарну кількість усього надходження та сумарну кількість усієї видачі за обраний проміжок часу. Цей звіт містить інтегровану функцію швидкого виклику друку. Вона дозволяє миттєво перетворити цифри на екрані на офіційний паперовий документ або зберегти його як цифровий [PDF](#)-файл для звітності. Зовнішній вигляд розробленої автоматизованої системи та взаємодія з персоналом складу, базуються на стандартах клієнтської розмітки [HTML](#). У даному проєкті, використовуються виключно як інструмент доставки розробленого функціоналу до кінцевого користувача. При цьому в інтерфейсі повністю відсутнє зайве декоративне навантаження. Елементи [HTML](#) відповідають за структуру кожної сторінки та генерацію інтерактивних компонентів. До них відносяться форми реєстрації рухів, випадючі списки вибору номенклатури з прив'язкою до одиниць виміру та календарі для фільтрації дат. Стилзація засобами [CSS](#) застосовується суто для забезпечення правильної та чіткої геометрії інтерфейсу. Вона відповідає за компактне розташування таблиць залишків, пошукових рядків та кнопок керування. Такий підхід робіт щоденну роботу оператора максимально ергономічною та швидкою. Функціональне та статичне моделювання системи

Проект розробки будь-якого сучасного програмного забезпечення вимагає чіткої графічної формалізації його архітектури. Для цього традиційно застосовують уніфіковану мову моделювання [UML](#). Вона дозволяє наочно представити логіку взаємодії користувачів із функціональними модулями створюваного додатка. Першим важливим етапом є побудова моделі прецедентів, яка описує систему з позиції зовнішнього спостерігача. На рис. 2.1 представлено структуру розробленої діаграми варіантів використання автоматизованої системи обліку матеріальних ресурсів. Рис. 2.1. Діаграма варіантів використання автоматизованої системи

Головними графічними елементами побудованої моделі виступають актори, варіанти використання та межі системи. Прямокутник у центрі схеми окреслює рамки проєктованої системи та містить внутрішні функціональні блоки. З двох сторін від меж системи розташовані актори, які уособлюють різні ролі користувачів. Вони мають чітко розмежовані права доступу до бази даних програми. Ліворуч на діаграмі знаходиться актор Комірник(Робітник). Він виступає основним оператором складських процесів на підприємстві. Цей користувач безпосередньо взаємодіє з поточним рухом матеріалів у реальному часі. Для нього на схемі відкриті чотири ключові варіанти використання: Авторизуватися в системі – обов'язковий крок для ідентифікації користувача та отримання доступу до робочого простору; Переглянути поточні залишки – функція оперативного моніторингу наявності будівельних матеріалів на складах; Оформити прихід/видачу матеріалів – прецедент для реєстрації нових складських транзакцій та проведення накладних; Сформувати зведений звіт – аналітичний модуль для перегляду підсумків руху товарів за обраний проміжок часу. Праворуч на схемі розміщено актора Адміністратор. Він володіє максимальним рівнем привілеїв і повністю контролює технічний стан системи. Окрім стандартних функцій авторизації, перегляду залишків, реєстрації накладних та генерації звітів, адміністратор одноосібно підключений до таких унікальних прецедентів: Керувати довідниками (товари, категорії) – модуль для наповнення каталогу номенклатури, який дозволяє додавати, редагувати або видаляти картки товарів; Керувати користувачами та ролями – інструмент створення нових облікових записів працівників і налаштування безпеки. Логічні зв'язки між окремими операціями на схемі чітко специфіковані за допомогою стандартних стереотипів мови [UML](#). Зв'язок типу [include](#) (включення) спрямований від процесу оформлення приходу та видачі до перегляду поточних залишків. Це показує, що система обов'язково виконує перевірку наявних обсягів сировини перед фіксацією будь-якої видачі на об'єкт будівництва. Зв'язок типу [extend](#) (розширення) підключений від прецеденту. Це підкреслює, що друк на папір або збереження у цифровий файл є додатковою необов'язковою опцією. Вона запускається виключно за бажанням користувача під час аналізу звітних даних. Наступним кроком логічного проектування системи є побудова її статичної моделі. Для цього розробляється діаграма класів, яка відображає архітектуру об'єктно-орієнтованого програмного коду вебдодатка та визначає структуру сутностей обміну даними. На відміну від суто фізичної схеми бази даних, кожна сутність на цій діаграмі містить не лише набір атрибутів, а й методи для безпосередньої обробки інформації та реалізації бізнес-логіки. На рис. 2.2

представлено структуру розробленої діаграми класів для автоматизованої системи обліку матеріальних ресурсів, зорієнтовану по вертикалі для забезпечення оптимального розміщення на сторінці документа. Рис. 2.2. Діаграма класів автоматизованої системи Архітектура програмного коду бекенд-частини додатка складається з шести взаємопов'язаних класів, які повністю покривають функціонал усіх модулів системи: **User** (Користувач системи) – відповідає за збереження реєстраційних даних співробітників (**login**, **password**) та їхніх рольових пріоритетів (**admin**, **worker**). Методи класу **authorize()** та **checkSession()** забезпечують автентифікацію, контроль активних сесій та розмежування прав доступу до інтерфейсу. **Material** (Матеріал) – базовий клас для управління номенклатурою складу. Крім ідентифікаторів та текстових назв, він містить атрибути фінансового та кількісного обліку: **price** (ціна), **min_stock** (мінімальний ліміт) та **current_quantity** (поточний залишок). Клас реалізує операції за допомогою методів створення, оновлення та видалення записів, а також містить логіку **checkDeficit()** для миттєвого виявлення дефіцитних позицій. **Transaction** (Транзакція) – журнал фіксації складських рухів матеріалів. Він оперує типами операцій **IN** (надходження) та **OUT** (видача), фіксуючи точну кількість ресурсу та мітку часу **date_time**. Методи класу забезпечують додавання нових операцій, контроль залишків на складі (**validateStock()**) та інтелектуальну обробку пошукових запитів користувача, включаючи нормалізацію дат та типів операцій. **Category** (Категорія) – логічний довідник для класифікації будівельних матеріалів. Окрім створення нових категорій, клас реалізує критично важливий метод **checkLinkedMaterials()**. Він виконує каскадну перевірку цілісності даних і блокує видачу помилкових ізольованих записів, якщо до категорії досі прив'язані активні товари на складі. **Unit** (Одиниця виміру) – ізольований довідник для збереження систем вимірювання кількості матеріальних ресурсів (штуки, кілограми, метри тощо). **Report** (Звіт) – обчислювальний аналітичний клас. Він не має власної фізичної таблиці у базі даних, а створюється динамічно в пам'яті комп'ютера. Атрибути класу фіксують часові межі звітності, а методи **generateSummaryReport()** та **printReport()** відповідають за агрегування даних за допомогою складних **SQL**-запитів та подальше виведення документа на друк у форматі накладної. Зв'язки між компонентами на діаграмі чітко розділені за типом взаємодії. Постійні структурні зв'язки зображені суцільними лініями з відповідними індексами кратності (один до багатьох). Вони показують, що один користувач може зареєструвати безліч транзакцій, а одна категорія чи одиниця виміру може охоплювати велику кількість матеріалів. Тимчасові зв'язки зображені пунктирними стрілками, спрямованими від класу **Report** до сутностей **Transaction**, **Material** та **Unit**. Це наочно демонструє, що аналітичний модуль є вторинним і повністю залежить від працездатності та структури першоджерел даних, з яких він акумулює підсумкову інформацію про рух складських запасів. Моделювання процесів взаємодії компонентів Для детального аналізу динамічної поведінки системи та визначення точного алгоритму взаємодії між об'єктами. У часі застосовується моделювання процесів взаємодії компонентів програмного забезпечення. Цей етап проєктування дозволяє простежити, як саме розподіляються функції та інформаційні потоки між інтерфейсною частиною додатка, серверними скриптами обробки бізнес-логіки та таблицями бази даних під час виконання конкретного сценарію роботи. Першим кроком у межах цього підрозділу є побудова діаграми послідовності. Вона фокусується на часовій упорядкованості повідомлень, що передаються між компонентами системи протягом життєвого циклу одного процесу. На рис. 2.3 представлено розроблену діаграму послідовності для базового процесу перегляду складських запасів та виконання інтелектуального пошуку матеріалів із автоматичним контролем дефіциту номенклатурних позицій. Рис. 2. 3. Діаграма послідовності процесу перегляду та пошуку матеріалів На побудованій діаграмі відображено чотири ключові об'єкти, які взаємодіють між собою: Актор (Користувач) – комірник або адміністратор, який ініціює запит; Граничний об'єкт інтерфейсу (модуль **dashboard.php**) – відображає візуальну частину програми в браузері; Керуючий об'єкт бізнес-логіки (Серверний обробник **PHP**) – виконує аналіз та обробку даних на серверній стороні; Сутність збереження даних (база даних **MySQL**) – відповідає за фізичне збереження та вибірку інформації. Процес взаємодії компонентів виконується за чітким лінійним алгоритмом у хронологічному порядку зверху вниз. Спочатку користувач вводить назву необхідного будівельного матеріалу або ключове слово статусу, у пошукове поле інтерфейсу сторінки **dashboard.php** та натискає кнопку

Граничний об'єкт веб-інтерфейсу перехоплює цю подію та надсилає [HTTP POST](#)-запит із параметром пошуку на сервер. Серверний обробник [PHP](#) першочергово ініціює внутрішню перевірку активної сесії користувача та валідує його рольові привілеї для забезпечення безпеки системи. Скрипт виконує фільтрацію та екранування вхідного запиту від потенційних ін'єкцій за допомогою функції [mysqli_real_escape_string\(\)](#). Програма формує реляційний [SQL](#)-запит із операторами [LEFT JOIN](#) для динамічного об'єднання даних із трьох таблиць ([materials](#), [categories](#), [units](#)) та передає його на виконання до бази даних. Система керування базами даних [MySQL](#) здійснює пошук у фізичних таблицях, формує результат вибірки та повертає масив рядків назад до обробника [PHP](#). На серверній стороні запускається алгоритм логічного порівняння поточних залишків матеріалів із їхнім мінімально допустимим критичним порогом. На основі результатів перевірки, [PHP](#)-скрипт динамічно генерує [HTML](#)-код сторінки, маркуючи позиції, де виявлено дефіцит, відповідним колірним стилем. Оновлений інтерфейс повертається у браузер користувача у вигляді структурованої таблиці з актуальним станом складських запасів підприємства. Далі слідує перетворення хронологічної послідовності повідомлень, у структурну мережу взаємодії компонентів. Для цього будується діаграма кооперації. На відміну від діаграми послідовності, вона акцентує увагу не на часі, а на геометрії та архітектурних зв'язках між об'єктами системи, відображаючи їхню структурну зв'язність. На рис. 2.4 представлено розроблену діаграму кооперації для аналогічного процесу складського обліку – перегляду запасів та інтелектуального пошуку матеріалів. Рис. 2.4. Діаграма кооперації компонентів системи під час пошуку матеріалів

Особливістю побудованої діаграми кооперації є відсутність вертикальних ліній життя об'єктів та блоків активації. Усі учасники процесу представлені у вигляді канонічних компонентів архітектури, а суцільні лінії між ними визначають наявність прямих каналів обміну інформацією. Черговість і логіка виконання всіх операцій фіксується за допомогою чіткої цифрової нумерації векторів повідомлень. Компонент користувача взаємодіє виключно з граничним інтерфейсом сторінки [dashboard.php](#) (повідомлення 1 та 9). Користувач не має і не може мати прямого доступу до серверних скриптів чи бази даних. Граничний об'єкт інтерфейсу виступає посередником, який передає клієнтські ініціативи на серверний обробник [PHP](#) через [HTTP](#)-протокол (повідомлення 2) і приймає назад згенерований результат для рендерингу в браузері (повідомлення 8). Центральний серверний процесор [PHP](#), акумулює в собі всю внутрішню бізнес-логіку додатка. Повідомлення 3, 4 та 7 спрямовані на самого себе, що вказує на виконання локальних обчислень: перевірку авторизаційних сесій, безпечне екранування текстових рядків та математичний розрахунок залишків на предмет дефіциту. Взаємодія із базою даних [MySQL](#) (повідомлення 5 та 6) здійснюється строго через серверний шар [PHP](#) за допомогою реляційних запитів. Така архітектура моделі взаємодії забезпечує високий рівень безпеки складських даних, ізолює критично важливу інформацію від прямого втручання з боку клієнтської частини та гарантує стабільність функціонування автоматизованої системи в цілому. Проектування інфологічної та даталогічної моделей бази даних

 **Знайдено плагіату: 0,04%** https://uk.wikipedia.org/wiki/Проектування_бази_даних id: 19

модель бази даних Створення ефективного інформаційного забезпечення розпочинається з побудови інфологічної моделі. Модель є первинним концептуальним представленням предметної області. Інфологічна модель розробляється на основі методології

 **Знайдено посилань: 0%** id: 20

«сутність-зв'язок»

 **Знайдено плагіату: 0,08%** https://uk.wikipedia.org/wiki/Проектування_бази_даних id: 21

([Entity-Relationship Model](#)). Її головна мета – відобразити логічну структуру інформаційних об'єктів, повністю абстрагуючись від технічних обмежень, конкретних типів даних комп'ютера. Це дозволяє зафіксувати склад сутностей системи та встановити коректну кратність зв'язків між ними на етапі аналізу. На рис. 2.5 представлено розроблену інфологічну модель автоматизованої системи обліку матеріальних ресурсів. Рис

2.5. Інфологічна модель бази даних Концептуальний простір бази даних, складається з п'яти взаємопов'язаних сутностей. Кожна сутність виконує чітко визначену роль: Користувачі ([Users](#)) – сутність, яка акумулює дані про персонал, що має право доступу до системи. Вона включає такі логічні атрибути, як унікальний ідентифікатор ([ID](#)), логін, хеш пароля для безпечної автентифікації та роль користувача в системі, яка визначає його

рівень привілеїв (Адміністратор або Комірник). Матеріали (**Materials**) – центральна номенклатурна сутність обліку. Вона містить інформацію про будівельні матеріали, що зберігаються на складі. Присутні такі атрибути: найменування ресурсу, ціна за одиницю, мінімально допустимий поріг запасу для контролю дефіциту та поточний фактичний залишок матеріалу. Транзакції (**Transactions**) – динамічна сутність, яка виконує функцію електронного журналу складського руху. Вона фіксує кожну операцію приходу або видачі матеріалів, зберігаючи інформацію про тип операції, кількість ресурсу, що переміщується, та точну дату й час проведення операції. Категорії (**Categories**) – довідник, призначений для логічного групування матеріалів за їхнім типом. Одиниці виміру (**Units**) – довідник, який містить перелік метрик для кількісного відображення ресурсів. Усі зв'язки в системі побудовані за класичним реляційним типом

” Знайдено посилань: 0%

id: 22

«один до багатьох»,

що забезпечує гнучкість та цілісність даних. Користувачі – Транзакції (один до багатьох): Один конкретний користувач системи може зареєструвати безліч складських операцій протягом своєї робочої зміни. Проте кожна окрема транзакція в журналі жорстко закріплена лише за одним відповідальним оператором, який її створив. Матеріали – Транзакції (один до багатьох): Один і той самий будівельний матеріал може фігурувати у безлічі різних транзакцій, як надходження на склад, так і видачі в роботу. Але кожна окрема позиція в журналі операцій, стосується лише одного конкретного найменування матеріалу. Категорії – Матеріали (один до багатьох): Одна логічна категорія може об'єднувати велику кількість різних матеріалів. Але кожен окремих матеріал може належати строго до однієї категорії. Одиниці виміру – Матеріали (один до багатьох): Одна одиниця виміру, може застосовуватися до безлічі різних номенклатурних позицій, але кожен конкретний матеріал вимірюється лише однією фіксованою метрикою. Даталогічна

🚫 Знайдено плагіату: 0,11% https://uk.wikipedia.org/wiki/Проектування_бази_даних id: 23

модель бази даних Даталогічна модель або фізична схема, відображає безпосереднє технічне втілення концептуальної архітектури засобами обраної системи управління базами даних. На цьому, етапі абстрактні сутності предметної області трансформуються у реальні реляційні таблиці. Логічні атрибути набувають строгих фізичних типів даних із фіксованою довжиною, а зв'язки реалізуються на рівні ядра бази даних, за допомогою механізмів первинних та зовнішніх ключів. На рис 2.6. представлено даталогічну модель бази даних автоматизованої системи. Рис

2. 6. Даталогічна модель бази даних Архітектура складається з п'яти оптимізованих таблиць, кожна з яких має сувору типізацію полів для забезпечення максимальної швидкодії та мінімізації витрат дискового простору сервера. Таблиця користувачів (**users**): Призначена для збереження облікових записів і забезпечення безпеки. Поле **id** є первинним ключем із типом **int(11)** та властивістю автоматичного приросту (**AUTO_INCREMENT**). Авторизаційні атрибути містяться у полях **login** (текстовий тип **varchar(50)**) та **password** (**varchar(255)**), що гарантує достатній обсяг пам'яті для збереження паролів. Рольова ознака, реалізована через перелічуваний тип **role**. Фіксований набір значень **enum('admin','worker')**, виключає можливість підміни ролі на етапі **HTTP**-запиту. Таблиця номенклатури складських запасів (**materials**): Виступає головним інформаційним вузлом. Крім первинного ключа **id : int(11)** та текстового найменування **name : varchar(255)**, вона містить два зовнішні ключі – **category_id : int(11)** та **unit_id : int(11)**. Для категорій фінансового та кількісного обліку, таких як ціна (**price**), мінімальний ліміт дефіциту (**min_stock**) та поточна кількість (**current_quantity**), використано точний числовий тип фіксованої довжини **decimal(10,2)**. Використання **decimal** замість типів із плаваючою точкою (**float**, **double**), дозволило повністю запобігти накопиченню похибок під час математичних округлень копійок, чи дробних частин матеріалу. Таблиця електронного журналу операцій (**transactions**): Акумулює динамічні дані про рух ресурсів підприємства. Оперує первинним ключем **id : int(11)** та двома зовнішніми ключами: **material_id : int(11)** вказує на конкретний товар та **user_id : int(11)**, який ідентифікує відповідального оператора. Поле типу операції зафіксоване через **type : enum('IN','OUT')**, що дозволяє зберігати лише два строго визначені стани: надходження (**IN**) або видача (**OUT**). Кількість матеріалу в накладній визначається полем **quantity : decimal(10,2)**. Хронологічна фіксація автоматизована, за допомогою системного типу **date_time : timestamp**, який самостійно записує точний час транзакції в момент додавання рядка. Довідкові таблиці нормалізації

([categories](#) та [units](#)): Розроблені для усунення дублювання текстової інформації в базі даних. Таблиця [categories](#) оперує полем назви групи [name](#) : [varchar](#)(100), а таблиця одиниць виміру [units](#) – полем [name](#) : [varchar](#)(20), для збереження коротких текстових міток. Фізичні зв'язки між таблицями, відображені лініями зв'язку на (рис. 2.6.), налаштовані за допомогою реляційних обмежень. Первинні ключі [id](#) таблиць [users](#), [categories](#), [units](#) та [materials](#) виступають батьківськими полями, на які посилаються відповідні зовнішні ключі ([user_id](#), [category_id](#), [unit_id](#), [material_id](#)). Проектування даталогічної моделі виконано із суворим дотриманням вимог третьої нормальної форми. Створення зовнішніх ключів, на рівні ядра [MySQL](#), гарантує реляційну цілісність бази даних. Система автоматично заблокує спробу видалити користувача або категорію товарів, якщо в журналі транзакцій чи таблиці матеріалів існують активні зв'язані з ними записи, що повністю захищає інформаційне середовище складу від руйнування та логічних збоїв. Окремо слід підкреслити, що для всіх первинних ([Primary Key](#)) та зовнішніх ([Foreign Key](#)) ключів, [MySQL](#) автоматично створюються індекси. У межах розробленої бази даних, це дозволило мінімізувати час виконання операцій об'єднання таблиць за допомогою оператора [LEFT JOIN](#). Завдяки індексуванню полів ідентифікаторів, система не виконує повне сканування таблиць під час кожного пошуку, а здійснює пряму вибірку за адресою запису в пам'яті. Таке рішення гарантує стабільно високу швидкість відгуку системи. Навіть за умов інтенсивного масштабування номенклатури складу та накопичення великої кількості записів у журналі транзакцій.

РЕАЛІЗАЦІЯ, БЕЗПЕКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Модель реалізації, якість ПЗ та опис процесів тестування

Тестування та верифікація можливостей системи Метою етапу тестування, є перевірка відповідності розробленої системи, заявленим функціональним та технічним вимогам, виявлення потенційних помилок у [PHP](#)-скриптів та верифікація надійності взаємодії із [MySQL](#). Для комплексного аналізу якості системи, було застосовано методологію функціонального тестування методом чорної скриньки. Цей підхід дозволяє перевірити коректність роботи інтерфейсу додатка, обробки вхідних даних із [HTML](#)-форм та виконання користувацьких сценаріїв, без необхідності аналізу внутрішньої структури самого коду. Перевірка успішної авторизації користувача. Для перевірки базової захищеності та коректності входу в систему було виконано сценарій успішної автентифікації. У поля форми введення було внесено чинні облікові дані адміністратора системи, які зберігаються в базі даних [autoblik](#). Серверний [PHP](#)-скрипт успішно опрацював запит, згенерував унікальний ідентифікатор сесії та перенаправив користувача на головну робочу панель додатка. На рис. 3.1. представлено інтерфейс системи після успішного проходження авторизації.

Рисунок 3.1. Інтерфейс системи після успішної авторизації користувача

Авторизація із помилкою. З метою перевірки стійкості модуля безпеки до введення некоректних даних, було проведено негативне тестування форми входу. У текстові поля логіну та пароля було навмисно введено випадковий набір символів, яких немає в таблиці [users](#) бази даних. Блок логіки сервера перехопив некоректний запит, заблокував створення сесії та згенерував інтерфейсне попередження для користувача про помилку введення. Візуалізацію захисного спрацьовування системи відображено на рис. 3.2.

Рисунок 3.2. Інтерфейсне повідомлення про помилку автентифікації

Інтелектуальний пошук матеріалу. Для верифікації роботи модуля пошуку та динамічної фільтрації складських запасів у відповідний інтерфейсний рядок, було введено частину назви будівельного матеріалу. Система в автоматичному режимі надіслала асинхронний запит до бази даних, виконала вибірку номенклатури за маскою найменування та миттєво перебудувала таблицю залишків, приховавши невідповідні позиції. Результат виконання цього сценарію пошуку наочно продемонстровано на рис. 3.3.

Рисунок 3.3. Результат роботи модуля інтелектуального пошуку матеріалів

Списання матеріалу в роботу. Наступним кроком було протестовано коректність проведення фінансово-облікових операцій, під час видачі будівельних матеріалів зі складу. Через спеціалізовану форму було створено транзакцію видачі на кількість матеріалу, яка не перевищує його фактичну наявність. Серверний скрипт успішно зафіксував нову операцію в журналі транзакцій та автоматично зменшив поточний баланс ресурсу в таблиці [materials](#). Оновлений стан журналу складського руху після проведення операції наведено на рисунку 3.4.

Рисунок 3. 4. Відображення успішно проведеної транзакції в журналі обліку

Списання з перевищенням наявного залишку. Критично важливим елементом стабільності складського обліку є недопущення утворення від'ємних залишків товарів. Для перевірки захисних алгоритмів було здійснено спробу списання кількості матеріалу, що значно перевищує його реальний залишок на складі. Програма успішно заблокувала надсилання некоректних даних до бази даних, зупинила виконання

транзакції та вивела на екран інформаційне сповіщення про дефіцит ресурсу, що відображено на рис. 3.5. Рисунок 3. 5. Блокування некоректної транзакції при перевищенні залишку Інструкція з розгортання та експлуатації системи Для забезпечення працездатності автоматизованої системи обліку матеріальних ресурсів, на локальному комп'ютері користувача необхідно виконати комплекс дій із розгортання серверного середовища, імпорту бази даних та конфігурації файлів проєкту. Оскільки система розроблена на базі клієнт-серверної архітектури з використанням мови [PHP](#) та [MySQL](#), процес інсталяції та запуску складається з кількох послідовних етапів. Етап 1. Для запуску серверної частини додатка без необхідності купівлі віддаленого хостингу, використовується кросплатформене збірне середовище [XAMPP](#). Процес налаштування сервера виглядає наступним чином: Завантажити та встановити пакет [XAMPP](#) (рекомендовано версію з підтримкою [PHP 8.x](#)) на локальний комп'ютер. Обов'язково при встановленні обрати [MySQL](#) та [phpMyAdmin](#). Запустити панель керування сервера через ярлик [XAMPP Control Panel](#). У вікні панелі керування навпроти модулів [Apache](#) та [MySQL](#) натиснути кнопки [Start](#). Успішний запуск модулів підтверджується зміною кольору їхніх назв на зелений та відображенням виділених системних портів. Етап 2. Локальний вебсервер [Apache](#) опрацьовує лише ті файли, які розташовані у його спеціалізованому кореневому каталозі. Скопіювати папку з усіма вихідними файлами системи. Перейти за шляхом встановлення сервера (за замовчуванням це корінь системного диска [C:\xampp\](#)) та знайти цільову директорію [htdocs](#). Вставити скопійовану папку проєкту в каталог [htdocs](#) та перейменувати її в [autoblik](#). Таким чином, повний фізичний шлях до головного файлу системи матиме вигляд: [C:\xampp\htdocs\autoblik\](#). Етап 3. Для того щоб програма могла взаємодіяти з інформаційним сховищем, необхідно створити структуру таблиць у базі даних. Відкрити будь-який сучасний браузер та у зведеному адресному рядку ввести [URL](#)-адресу локального інструменту керування: [http://localhost/phpmyadmin/](#). У лівому бічному меню натиснути кнопку Створити БД ([New](#)). У полі назви бази даних ввести точне ім'я [autoblik](#), вибрати кодування символів [utf8mb4_general_ci](#) та натиснути кнопку Створити. Перевірити, що новостворена база даних є активною. Зверху, має відображатися шлях: сервер 127.0.0.1 база даних [autoblik](#). Після чого перейти на верхню вкладку навігаційного меню Імпорт ([Import](#)). Натиснути кнопку (Вибрати файл) та обрати файл з базою даних із розширенням [.sql](#). Натиснути кнопку Вперед ([Go](#)) наприкінці сторінки. База даних в автоматичному режимі розгорне всі п'ять реляційних таблиць ([users](#), [materials](#), [transactions](#), [categories](#), [units](#)) та налаштує зв'язки між ними. Етап 4. Після успішного виконання попередніх кроків система повністю готова до безпосередньої експлуатації користувачем. У браузері ввести адресу доступу до проєкту: [http://localhost/autoblik/](#). Веб-сервер автоматично перенаправить користувача на стартову сторінку авторизації. Для входу в систему необхідно ввести в текстові поля форми облікові дані, які вже містяться у базі даних після імпорту: Адміністратор: [admin_daniil](#) (пароль: [admin123](#)); Користувач: [ivan_worker](#) (пароль: [worker456](#)). Після успішної перевірки пароля через [PHP](#)-скрипт сесія стає активною, і користувач отримує повний доступ до робочого простору системи. Аналіз безпеки програмного забезпечення та захист від несанкціонованого доступу до даних Надійна безпека та захист інформації, є умовою функціонування автоматизованих систем. Несанкціоноване втручання або модифікація даних, може призвести до значних фінансових збитків підприємства. У межах розробки системи, було проведено комплексний аналіз потенційних загроз та реалізовано систему захисту, що охоплює автентифікацію користувачів, валідацію вхідних даних та криптографічне перетворення конфіденційної інформації. Першим контуром захисту системи є модуль автентифікації та розмежування прав доступу на основі використання [PHP](#). Безпека мережевого протоколу реалізується шляхом ініціалізації функції [session_start\(\)](#) у заголовочній частині кожного серверного скрипта. Після успішного введення логіну та пароля програма створює на стороні сервера унікальний ідентифікатор сесії, а в суперглобальний масив [\\$_SESSION](#) записується роль користувача ([admin](#) або [worker](#)). Наявність перевірок на кожній внутрішній сторінці системи, повністю виключає можливість несанкціонованого доступу до функцій. Шлях прямого введення [URL](#)-адреси в рядок браузера вхідними користувачами, без активної сесії заборонено. Другим аспектом безпеки, стала автентифікація користувачів. Метод реалізований за допомогою прямого реляційного зіставлення текстових рядків на рівні [SQL](#)-запиту до таблиці [users](#). Водночас, з метою забезпечення перспектив подальшого розвитку, фізична структура таблиці спроектована із урахуванням майбутнього впровадження алгоритмів незворотного хешування. Поле [password](#) має оптимізований тип даних [varchar\(255\)](#), що дозволить під час промислового розгортання системи

безперешкодно інтегрувати серверні функції [password_hash\(\)](#) та [password_verify\(\)](#) для надійного захисту облікових записів персоналу від потенційної компрометації у разі несанкціонованого копіювання бази даних підприємства. У модулі авторизації [auth.php](#) та на головній сторінці, реалізовано практичний захист від атак типу [SQL](#)-ін'єкцій. Для нейтралізації потенційно небезпечних символів у вхідних параметрах форми пошуку ([\\$_POST\['search'\]](#)) та поля логіну ([\\$_POST\['login'\]](#)), застосовано функцію [mysqli_real_escape_string\(\)](#). Цей метод в автоматичному режимі екранує одиночні лапки та інші службові послідовності бази даних, безпосередньо перед формуванням та виконанням реляційного запиту.

 AI generated text detected: ChatGPT 5.3, : 53,72%

id: 24

Програмне рішення перетворює будь-який надісланий зловмисником фрагмент коду на безпечний текстовий рядок, що повністю блокує можливість несанкціонованої модифікації структури [SQL](#)-запитів до таблиць на рівні сервера. ВИСНОВКИ У кваліфікаційній роботі успішно вирішено інженерну задачу з автоматизації складського обліку будівельного підприємства. Актуальність обраної теми, зумовлена необхідністю точного контролю за великою номенклатурою матеріалів у сучасних умовах.

Особливістю об'єкта дослідження, є висока динамічність складських операцій приходу та видачі товарів. Проведений аналіз показав, що ручні методи обліку призводять до логістичних помилок та несвоєчасного виявлення дефіциту. Для вирішення цих проблем було обґрунтовано перехід на клієнт-серверну систему. На етапі проектування розроблено комплекс [UML](#)-діаграм та створено оптимальну структуру реляційної бази даних у середовищі [MySQL](#). Практичною пропозицією роботи, є реалізація автоматизованої системи [autoblik](#). Система успішно пройшла функціональне тестування за методом чорної скриньки та забезпечує суворе розмежування прав доступу. Для впровадження системи рекомендується її розгортання на наявному комп'ютерному парку за допомогою локального сервера [XAMPP](#). Ефект від реалізації проєкту полягає у повній автоматизації робочих місць та ліквідації паперового обліку. Програма забезпечує швидкий інтелектуальний пошук товарів та ведення електронного журналу операцій. Візуальне маркування низьких залишків надійно захищає склад від дефіциту матеріалів, що забезпечує високу економічну ефективність для підприємства. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ Івахненко С. В. Інформаційні технології в організації бухгалтерського обліку та аудиту : навч. посіб. 4-е вид., випр. і доп. Київ : Знання, 2008. 343 с. [Wylie L. ERP: A Vision of the Next-Generation MRP II. Gartner Group Research Report. 1990. Scenario S-100-124.](#) Гриценко С. І., Дейкун В. О. Оптимізація логістичних процесів підприємства в умовах цифрової трансформації (на прикладі [Kuehne + Nagel](#)). Вісник економічної науки України. 2025. № 1 (48). 134-139 с. Плєскач В. Л., Рогушина Ю. В. Інформаційні системи та технології на підприємствах : підручник. Київ : КНИГА, 2011. 512 с. С 41 Основи інформаційних систем: Навч. посібник. – Вид. 2-ге, перероб. і доп. / В. Ф. Ситник, Т. А. Писаревська, Н. В. Єрьоміна, О. С Краєва; За ред. В. Ф. Ситника. – К.: КНЕУ, 2001. – 420 с Муха Т. [Analysis of recovery and optimization strategies for SaaS solutions on shipment tracking efficiency in logistics systems.](#) Менеджмент та підприємництво: тренди розвитку (Запорізький національний університет). 2025. № 33. 73-83 с. Скіцько В. І. Цифрові технології сучасної логістики та управління ланцюгами поставок / В. І. Скіцько. Маркетинг і цифрові технології. 2018. Т. 2, № 3. 48-63 с. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І., Задерейко О. В. Веб-технології : навч. посіб. Одеса : Фенікс, 2017. 320 с. [PHP: Hypertext Preprocessor. Official Documentation. URL: https://www.php.net/manual/en/](#) (дата звернення: 18.04.2026). Додаток

Відмова від відповідальності:

Цей звіт повинен бути правильно інтерпретований та проаналізований кваліфікованою особою, яка несе відповідальність за оцінку!

Будь-яка інформація, наведена у цьому звіті, не є остаточною та підлягає ручному огляду та аналізу. Дотримуйтесь вказівок: [Рекомендації з оцінки](#)

Детектор Плагіату - Право знати автора!  SkyLine LLC