Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Проект інформаційної веб-системи для управління збутом продукції

Виконав(ла): студент(ка) 4 курсу, групи СНс-42
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Матвіюк Я.А. (прізвище та ініціали)

Керівник (підпис) Приймак М.В. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Кареліна О.В. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«25» січня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Матвіюк Ярослав Анатолійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Проект інформаційної веб-системи для управління збутом продукції**

Керівник роботи **д.т.н., проф. Приймак М.В.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*02*» *березня* 2021 року № *4/7-171*

2. Термін подання студентом завершеної роботи 18 червня 2021 р.

3. Вихідні дані до роботи Опис предметної області

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ ; 1 Потсановка задачі 1.1 Стан проблеми обліку постачань підприємством 1.2Поняття інформаційної системи 1.3 Аналіз відомих автоматизованих систем 1.4 Розвиток Інтернет технологій та їх використання для розробки 1.5 Постановка задачі створення автоматизованої довідково-інформаційної системи 1.6Аналіз технічного завдання та етапи створення автоматизованої довідково-інформаційної системи 2 Проектування системи 2.1 Вибір логічної моделі даних 2.2 Вибір концептуальної моделі 2.3 Побудова моделі 2.4 Аналіз алгоритмів роботи з базою даних 2.5 Аналіз і вибір програмних засобів розробки автоматизованої довідково-інформаційної системи 2.6 Опис загальної структури автоматизованої довідково-інформаційної системи 2.7 Тестування програмного продукту 3 Безпека життєдіяльності та основи охорони праці 3.1 Загальні вимоги законодавства з охорони праці в галузі інформаційних технологій 3.2 Землетруси і порядок дії населення при них; Перелік посилань; Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема. 2. ER-діаграма бази даних. 3. Функціональна схема 4. Структура додатку.
5. Алгоритм роботи користувача 6 Алгоритм формування рахунку-фактури. 7. Алгоритм формування Замовлень 8 – 9 Фрагменти інтерфейсу користувача. 10. Висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доц.		

7. Дата видачі завдання 25 січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.01.21-27.01.21	*Виконано*
2.	Підбір джерел по темі роботи	28.01.21 – 01.04.21	*Виконано*
3.	Оформлення першого розділу	15.04.2021	*Виконано*
4.	Оформлення другого розділу	30.04.2021	*Виконано*
5.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи охорони праці»	15.05.2021	*Виконано*
6.	Оформлення кваліфікаційної роботи	07.06.2021	*Виконано*
7.	Перевірка на плагіат	07.06.2021	*Виконано*
8.	Нормоконтроль	15.06.2021	*Виконано*
9.	Попередній захист кваліфікаційної роботи	18.06.2021	*Виконано*
10.	Захист кваліфікаційної роботи	24.06.2021	

Студент _____ Матвіюк Я.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Приймак М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Проект інформаційної веб-системи для управління збутом продукції // Кваліфікаційна робота бакалавра // Матвіюк Ярослав Анатолійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНс-42 // Тернопіль, 2021 // с. – , рис. – , табл. – , кресл. – , додат. – , бібліогр. – .

Ключові слова: облік, постачальники, споживачі, інформаційна система, інтернет-технологія, клієнт, сервер, інтерпретатор, модуль, апаратне забезпечення, програмне забезпечення, база даних.

При виконанні кваліфікаційної роботи подана характеристика завдань та проведений аналіз існуючих способів розв’язку задачі, побудовані інформаційна модель та схема технологічного процесу обробки інформації. На основі аналізу зроблена постановка задачі проекту, обґрунтовані проектні рішення.

Також робота включає проектування структури системи, подається опис спроектованої бази даних, алгоритму розв'язку задачі та інтерфейсу користувача; описано розробку схеми взаємозв'язку програмних модулів, програмну реалізацію та блок-схеми програмних модулів.

Інформаційна система, що розробляється, відрізнятиметься від аналогічного програмного забезпечення можливістю застосування на сучасній електронно-обчислювальній техніці, зручним інтерфейсом, низькою вартістю, можливістю його використання на будь-якому підприємстві.

ANNOTATION

Information web-system design for products sales management // Qualification work of the educational level "Bachelor" // Matviyuk Yaroslav // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CHc-42 // Ternopil, 2021 // p. – , fig. – , references – , posters – , applications – .

Keywords: accounting, suppliers, consumers, information system, Internet technology, client, server, interpreter, module, hardware, software, database.

During the qualification work preparation, the characteristic of tasks is given and the analysis of existing ways of the decision of a problem is carried out, the information model and the scheme of technological process of information processing are constructed. On the basis of the analysis the statement of the problem of the project is made, the design decisions are substantiated.

The work also includes the design of the system structure, a description of the designed database, the algorithm for solving the problem and the user interface; describes the development of the scheme of interconnection of software modules, software implementation and block diagram of software modules.

The developed information system will differ from the similar software by a possibility of application on modern electronic computer equipment, the convenient interface, low cost, a possibility of its use at any enterprise.

ЗМІСТ

ВСТУП

В даний час зберігання, пошук і доступ до інформації стали важливим явищем не тільки для людей, тісно пов'язаних з діяльністю у сфері IT, але і входить в роботу звичайних службовців, студентів і т.д., допомагаючи їм скоротити часові, матеріальні та інші затрати на опрацювання інформаційних ресурсів.

В даний час практично всі підприємства будь-якої форми власності самостійно займаються пошуком підприємств-постачальників та покупців виробленого товару, а також в міру необхідності організацією постачань.

При здійсненні постачань на підприємство та при продажу вироблених товарів (наданні послуг) здійснюється опрацювання та подальше зберігання великих обсягів даних. Цей процес передбачає:

– оформлення документів і контроль за кожною операцією руху товарів по складі;

– контроль за оприбуткуванням товарів та списанням ресурсів;

– коректне оформлення списання товарів;

– автоматизоване відслідковування мінімальних залишків товарів та сповіщення про необхідність поповнення складу.

У зв'язку з цим для надійного функціонування системи збуту необхідно вести їх систематичний і безперервний облік, а отже автоматизація даної ланки діяльності підприємства є актуальною задачею.

Тому метою даної кваліфікаційної роботи є розробка інформаційної системи обліку і контролю постачань та збуту з використанням Інтернет-технологій.

Інформаційна система, що розробляється, відрізнятиметься від аналогічного програмного забезпечення можливістю застосування на сучасній електронно-обчислювальній техніці, зручним інтерфейсом, низькою вартістю, можливістю його використання на будь-якому підприємстві.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Стан проблеми обліку постачань підприємством

При здійсненні постачань підприємства виробники продукції виробничо-технічного призначення вступають у договірні відносини з підприємствами споживачами (покупцями) як постачальники укладають прямі договори з підприємствами споживачами для збуту продукції і комплексного постачання підприємств-замовників.

Договори про постачання необхідно укладати своєчасно. У них вказуються умови постачання товарів, їх кількість, асортимент, якість, комплектність і терміни постачання. Крім того, в договорах передбачені ціни на товари, загальна сума, порядок розрахунків, платіжні і відвантажувальні реквізити постачальника і одержувача продукції. Договори підлягають обов'язковому виконанню по всім вказаним в них пунктам. Порушення термінів договорів і зобов'язань вабить відповідальність, передбачену “Положенням про постачання продукції виробничо-технічного призначення” і “Особливими умовами постачання.”

Контроль за виконанням договорів здійснюють товарні відділи.

Раціональна організація приймання продукції від постачальників має важливе значення для своєчасного, повного, комплексного постачання підприємств сировиною, матеріалами, паливом, інструментами, устаткуванням і іншими засобами виробництва.

Правильне приймання і оформлення документами товарів, що поступили

Є надійному основою збереженню товарно-матеріальних цінностей.

Загальний порядок приймання товарно-матеріальних цінностей встановлений “Положенням про постачання продукції виробничо-технічного призначення”. Порядок і терміни приймання товарно-матеріальних цінностей в певній кількості і якості, оформлення актів приймання і пред'явлення претензій визначено інструкцією про порядок приймання продукції. Особливості

приймання окремих видів продукції визначаються в ГОСТах, технічних умовах, Особливих умовах постачання і договорах постачання, що передбачають особливі порядки приймання продукції при постачаннях.

У підприємствах будь-яких форм власності поставками та їх оформленням займаються певні працівники з використанням спеціалізованого програмного забезпечення. Розробка такого програмного забезпечення є актуальною задачею, бо, по-перше, воно швидко застаріває, а, по друге, не відповідає вимогам сучасних апаратно-програмних платформ. Відсутність такого ПЗ значно збільшує трудомісткість процесів обліку і знижує продуктивність праці. Розроблюваний продукт призначений для вирішення цих задач.

1.2 Поняття інформаційної системи

Інформаційна система (Information system) – сукупність технічних засобів разом з організаційними для опрацювання інформації з метою забезпечення інформаційних потреб користувачів. Таке визначення є дуже загальним і підлягає уточненню.

Інформаційні системи на сьогодні охоплюють всі сфери життєіяльності людини та виробництва у суспільстві. Це природна потреба в обміні даними. І щоразу їх стає все більше, а вимоги до швидкості такого обміну також ростуть. Тому звичайні паперові документи все більше виконують архівну функцію чи відображають формальний бік процесів в економіці та соціумі. Найдавнішими, якщо можна так сказати, інформаційними системами є бібліотеки та архіви.

Але навіть у цих закладах сьогодні використання комп'ютеризованих ІС є необхідністю, адже саме ці ІС дозволяють вести облік фондів, відслідковувати їх рух, виконувати операції по їх супроводу, колекціонуванню, пошуку і т.д. Звичайно, це означає, що робота людини буде автоматизована і продуктивність працівників такому разі виросте в рази.

1.3 Аналіз відомих автоматизованих систем

Вітчизняна IT-сфера в нинішніх умовах значною мірою займається створенням програмних продуктів д я автоматизації різноманітних бізнес-процесів, тобто розробкою ІС як для закордонних замовників (в основному), так і на внутрішній ринок. Відповідно до типів об'єктів автоматизації та управління, обсягів даних , кількості користувачів сучасні ІС можуть сильно відрізнятись. На сьогодні не існує загально прийнятої класифікації інформаційних систем, але можна виділити ряд ознак, за якими всі ІС можна віднести до певної групи:

– сфера діяльності та рівень діяльності: це масштабу держави чи адміністративної одиниці, на рівні галузі, підприємства чи корпорації, на рівні певного технологічного процесу;

– автоматизація певного типу діяльності: пошукові, довідкові, системи підтримки прийняття рішень, системи штучного інтелекту;

– рівень централізації обробки: централізовані, розподілені, системи колективного доступу;

– ступінь інтеграції функцій ІС: багаторівневі від керівництва до низової ланки працівників, на певному рівні управління чи прийняття рішень.

На державному рівні відомі ІС для управління на рівні Кабінету міністрів, Президента України, міністреств, Національного банку України.

До територіальних автоматизованих інформаційних систем належать ІС для управління адміністративно-територіальною одиницею країни: область, район, місто, ОТГ. Як правило, такі системи інтегровані у загальнодержавні ІС для подачі звітів та отримання доступу до потрібних даних.

Галузеві ІС об'єктом управління мають підприємство чи організацію, установу. Як правило – це галузеві ІС рівня міністерств чи департаментів, державних агенцій тощо. Сюди ж належать і ІС не тільки дл япрромисловості, але і науки.

ІС підприємства (АСУП) чи об'єднання (АСУВ) – це системи для регулярного вирішення задач управління підприємством та автоматизації його бізнес-процесів. Прикладом таких систем є АСУ ТНТУ ім. Івана Пулюя.

Якщо говорити про ІС для прийняття рішень, то тут можна виділити СППР та системи на основі штучного інтелекту. Причому, розрізняють різін способи реалізації таких систем:

1. Системи типу "запитання-відповідь", які в діалоговому режимі виконують запити користувача.

2. Системи для розрахунку різноманітних статистичних характеристик, що допомагає потім оператору прийняти рішення на основі отриманої аналітики. Це класичні технології на основі сховищ даних. При цьому від оператора не вимагається глибоке знання математичних моделей для розрахунку статистичних характеристик та прогнозування.

3. Так звані експертні системи, що реалізують алгоритми на основі ШІ та матстатистику для надання базової інформації з прийняття рішення чи вироблення альтернативних рішень і їх порівняння.

1.4 Розвиток Інтернет-технологій та їх використання для розробки

З часів своєї появи в 60-х роках 20-го століття і до сьогодні мережа Інтернет працює на основі стеку протоколів TCP/IP і надає користувача велику кількість послуг та сервісів. Перерахуємо коротко як давніші, так і сучасні сервіси, доступні в Інтернет-і, взявши до уваги, що всі ці сервіси умовно можна поділити на дві групи:

1. Для обміну інформацією.
2. Для доступу до даних в базах даних чи в сховищах даних.

Отже, до першої групи можна віднести такі сервіси**:**

Telnet – протокол для віддаленого доступу. Реалізує підключення до комп'ютера та управління його роботою. На сьогодні застарів з міркувань безпеки, використовується його безпечний аналог SSH.

FTP (File Transfer Protocol) – протокол обміну файлами. Використовується також для розгортання сайтів, керування резервними копіями на віддалених комп'ютерах та хостинг-площадках.

NFS (Network File System) – розподілена файлова система. Протокол дозволяє ствоиити на основі декількох накопичувачів даних на різних комп'ютерах віртуальний диск і працювати з ним, як єдиною файловою системою.

Електронна пошта – практично найстаріша служба в Інтернет, актуальна і широко використовувана і до сьогодні. Дозволяє обмінюватись текстовими повідомленнями та навіть файлами без встановлення каналу зв'язку між абонентами по аналогії до роботи звичайної, традиційної поштової служби.

Крім послуг, перерахованих вище, Internet надає також ряд інших популярних сервісів. Наприклад, електронні словники (Webster), служби електронного перекладу (найпопулярніші сьогоні від Google та Microsoft; системи пошуку інформації та документів в мережі Інтернет (Google, Bing Та інші). Ну і, звичайно, служба всесвітньої павутини WWW – створення та обмін гіпертекстовими документами на основі протоколу HTTP(S). Вона настільки стала популярною, що за цими принципами сьогодні будують клієнт-серверні застосування з використанням протоколу RESTFull API.

Звичайно, що всі послуги в голобальній чи в локальній мережі побудовані за принципом клієнт-сервер. При чому сервером є програма, котра виконується на комп'ютері, підключеному до мережі і перебуває в режимі очікування, щоби дати відповідь на запит клієнта (іншої програми), складеному на певній мові та надісланому по встановленому протоколу прикладного рівня. Типово на одному комп'ютері працює декілька серверів. Наприклад, веб-сервер, сервер баз даних та сервер прикладного програмного рівня для автоматизації бізнес-процесів.

Кількість серверів служби WWW постійно зростає. Для функціонування цього сервісу використовується протокол HTTP, а самі документи складаються з використанням мови розмітки HTML (Hyper Text Markup Language). Для таких документів характерним є наявність гіперпосилань, які власне і утворюють цю павутину документів. На сьогодні розвиток цієї техїноогії передбачає використання також таблиць стилів для відділення контенту від його оформлення, виконання на стороні клієнта в HTML-документі програмних скриптів на мові Java Script чи Type Script, що дозволяє зробити документи динамічними і мінімізувати мережевий трафік, реалізувавши частину бізнес-логіки на стороні клієнта.

Попри свою популярність ця служба має і ряд недоліків. Основний з них те, що сервер, надавши відповідь клієнтові, "забуває про нього. А на прикладному рівні виникає потреба управління сесіями підключення, коли, наприклад, користувач перебуває на ресурсі, що вимагає автентифікації. Тоді для цього використовується механізм cookies разом з програними засобами керування сесіями як на серверній стороні, так і на стороні клієнта.

Крім того, оскільки клієнтським програмним забезпеченням для цієї служби є інтернет-браузери, а на мову HTML існує стандарт, то кожен виробник браузерів реалізує вимоги стандарту власним чином. Через це виникають проблеми міжбраузерної сумісності документів, що є ще одним розділом в тестуванні програмних продуктів, призначених для мережі Інтернет і базованих на протоколі HTTP та мові HTML.

1.5 Постановка задачі створення автоматизованої довідково-інформаційної системи

Будь-яке підприємство, здійснюючи свою діяльність, для отримання продукції від постачальників повинно укласти з останніми договір на постачання продукції. Звично на однойменну продукцію підприємство-замовник укладає

декілька договорів з підприємствами-постачальниками. Потім замовник у міру потреби в певній продукції висилає постачальнику заявку на постачання продукції і одержує від останнього рахунок-фактура, в якому вказано найменування продукції і її відпускну ціну. На підставі цих рахунків підприємство-замовник визначає оптимальну заявку і висилає постачальнику замовлення на постачання продукції. Після отримання замовленої продукції замовник відправляє рахунок в бухгалтерію, яка оплачує його в банку в перебігу терміну, передбаченого договором. Тому для документального забезпечення процесу постачань на підприємство програма повинна створювати (роздруковувати) наступні необхідні документи:

– бланк договору підприємства-замовника з фірмою-постачальником (з вказівкою найменування і юридичних адрес сторін, асортименту продукції для постачань, її кількості і гаданої вартості, а так само умови і терміни дії договору);

– замовлення на постачання необхідної продукції (указується кількість, найменування, номенклатура, терміни постачання).

Також створювана автоматизована система за наявними даними про постачальників і знов одержаним даним повинна визначати оптимальний рахунок-фактуру з точки зору кількість-ціна.

Будь-яке постачання підприємство-замовник зобов'язано сплатити у встановлені договором терміни, тому АС повинна здійснювати підрахунок суми боргу (грошей до виплати) на поточну дату. В додатку А представлена функціональна схема здійснення постачань на підприємство.

Таким чином для розробки автоматизованої довідково-інформаційної системи необхідно виконати наступні завдання:

– реалізація управління доступом до автоматизованої довідково-інформаційної системи;

– створити СУБД для роботи автоматизованої довідково-інформаційної системи;

– розробити довідкову інформацію по автоматизованій довідково-інформаційній системі;

– виконати аналіз і обліку постачань.

1.6 Аналіз технічного завдання та етапи створення автоматизованої довідково-інформаційної системи

Підприємство або фірма, проводячи свою продукцію, потребує постачань сировини від інших підприємств. Але на одну і ту саму сировину у різних виробників-постачальників різна відпускна ціна, тому в цілях зниження собівартості продукції, що випускається, підприємство замовник укладає договори з великою кількістю постачальників і потім висилає постачальникам заявку на постачання продукції з вказівкою типу і її кількості. Оскільки підприємство-замовник при отриманні вантажів так або інакше пов'язано з документами, з документальним оформленням постачань, то проектована автоматизована довідково-інформаційна система повинна створювати всі бланки документів, пов'язаних з постачаннями.

Оскільки всі постачальники висилають замовнику рахунку-фактуру (прейскурант цін на замовлену продукцію), то серед їх множини необхідно визначити найбільш вигідне для підприємства-замовника, як за ціною, так і за якістю, що і повинна виконувати створювана АС.

Оскільки договори з постачальниками вкладаються на певний термін, передбачувана кількість продукції, що поставляється, і на певну суму, то при здійсненні замовлення на постачання продукції, в договорі обмовляється термін, в перебігу якого замовлення повинно бути сплачене, тому необхідно знати суму до оплати на вказане число, як загальну так і по різним постачальникам окремо.

Оскільки всі вище перелічені дії здійснюються впродовж тривалого часу, то при прийнятті рішення про продовження терміну дії договору доцільно приймати до уваги наступні чинники: якість постачань конкретними

постачальниками (зважаючи на виконання термінів здійснення постачань, відповідність номенклатури поставленої продукції замовленої, відсутність або відсоток браку), його терпимість по відношенню до оплати по постачанням. Тому необхідно зберігати всю інформацію про постачання на підприємство, що б надалі її можна було б використовувати.

Автоматизована довідково-інформаційна система “Контроль та облік постачань” використовуватиметься на підприємствах різних форм власності і забезпечуватиме контроль і облік постачань вироблюваних на підприємство. Також при використанні даного ПЗ буде можливість складання звітності про постачання на підприємство, виявлення заборгованості по оплаті постачань. Що розроблюване ПЗ може бути використано як керівником підприємства, для здійснення контролю вироблюваних постачань на підприємство, так і начальниками цехів, для ведення обліку постачань.

1.6.1 Призначення розробки автоматизованої довідково-інформаційної системи

Метою дипломного проектування є розробка і створення програмного продукту “Контроль та облік постачань”. Дане програмне забезпечення призначено для контролю, обліку, автоматизації і систематизації інформації про постачання різного виду продукції на підприємство будь-якої форми власності, що займається будь-яким видом виробництва або діяльності.

Програмний продукт, що розробляється, повинен забезпечувати створення інформаційної бази про здійснені постачання на підприємство, а також здійснювати створення наступних документів :

– бланк договору підприємства замовника з фірмою-постачальником (з вказівкою найменування і юридичних адрес сторін, що беруть участь в договорі, асортименту продукції для постачань, її кількості, гаданої вартості, умови і терміни дії договору);

– заявку на постачання необхідної продукції (указується кількість, найменування, номенклатура, терміни постачання, сума постачання);

– замовлення на постачання.

Комерційна версія програмного продукту дозволить проводити:

– повніший контроль і організацію обліку про постачання на підприємство;

– автоматизувати процес оформлення постачань на підприємство;

– зменшить часові витрати на оформлення документів, пов'язаних з постачаннями;

– обчислювати заборгованість по оплаті здійснених постачань на вказаний період;

– забезпечити користувача системою допомоги як за поняттями наочної області, так і по користуванню програмним продуктом.

Що розробляється автоматизована система повинна буде реалізувати наступні функції:

1. Забезпечення введення даних про постачання на підприємство;

2. Аналіз введеної інформації;

3. Підрахунок заборгованості підприємства за здійснені постачання;

4. Визначати оптимальний рахунок-фактура з точки зору “кількість-ціна”;

5. Проводить друк документації, пов'язаної з організацією постачань (бланк договору, замовлення, заявки).

1.6.2 Вимоги до розробки

Автоматизована довідково-інформаційна система, що розробляється, повинна забезпечувати автоматизований контроль, а так само облік постачань на підприємство (цех цього підприємства), для цього створювана система повинна:

– забезпечувати введення, пов'язаних з постачаннями на підприємство і обробку цих даних;

– створювати звітні документи і документи для організації товаропостачання;

– мати систему допомоги;

– при введенні даних про найменування товарів повинен використовуватися довідник “Номенклатура товарів”;

– створювані документи повинні відповідати галузевим стандартам, прийнятим на підприємстві.

Створювана програма повинна функціонувати, легко інсталюватися, настроюватися і коректно працювати при виконанні наступних вимог:

– наявність операційної системи типу, Windows NT 4.x, Windows 2000, Windwos XP, Windows 2003, Windows Vista/7 і сумісних з ними;

– наявність бази даних MySQL;

– ввід дати обов'язковий у формі маски;

– ввід цифр обов'язковий.

Для забезпечення захисту від несанкціонованого доступу до інформації, пов'язаної з постачаннями на підприємство буде передбачена система паролів при завантаженні програми в оперативну пам'ять. Для забезпечення захисту даних про збої в мережі живлення ПК або аварійному завершенні роботи програми буде передбачений режим авто збереження.

1.6.3 Плановані показники ефективності

В результаті розробки системи планується досягти певних результатів від її викоистання:

– зменшення часу на облік операцій поставки товарів;

– автоматизоване управління поставками;

– зберігання відомостей про всі поставки за довгий період часу для можливості розрахунку аналітики і планування роботи компанії;

– контроль за оплатою проданих товарів та ведення фінансової звітності.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Вибір логічної моделі даних

2.1.1 Ієрархічна модель даних

Ієрархічна модель даних є ієрархією у вигляді дерева. Дана модель даних базується на сегменті, який є сукупністю полів, що характеризують даний сегмент. Сегменти розрізняються по типу, а кожен тип характеризується фіксованою довжиною і конкретним розбиттям на поля даних. Два зв'язані сегменти, розташованих на суміжних рівнях називаються початковим (вищого рівня) і породженим (нижчого). У ієрархічній структурі є сегмент, який не має результатного і називається головним або кореневим. У цьому сегменті звично розташовується ідентифікатор об'єкту, властивості якого розкриваються в сегментах другого і нижчих рівнів ієрархії.

Для реалізації даної моделі на фізичному рівні використовується ряд стандартних методів розміщення даних на пристроях, що запам'ятовують, які можуть розміщувати сегменти наступними ієрархічними способами доступу: послідовний, індексний-послідовний, прямій, індексний-прямій. Відповідно до способів розміщення сегментів встановлюється порядок доступу до них. Встановлений порядок доступу до сегментів обумовлює процедурна мови запитів і вимагає від користувача знання шляхів доступу до даним, що проходять ро гілкам дерева ієрархічного запису. Що є одним з недоліків даної моделі. Як інші недоліки можна відзначити наступні:

– складність реалізації “багато до багатьох”, що вимагає надмірності даних на фізичному рівні, що приведе до небажаного і не виправданого збільшення БД;

– вимога підвищеної коректності до операції видалення, оскільки видалення початкового сегменту спричиняє за собою видалення породжених;

– доступ до будь-якого породженого сегменту можливий тільки через результатний, що збільшує час відповіді а запит до БД.

У зв'язку з тим, що ієрархічна модель володіє великою кількістю недоліків вона не буде застосуються для моделювання ASYS, що розробляється.

2.1.2 Сіткова модель даних

Сітка – більш загальна структура порівняно з ієрархією. Вузлами мережі є окремі екземпляри запису. Вузли запису є одиницею доступу до БД. Оскільки окремий вузол може мати декілька безпосередньо старших вузлів, так само, як і декілька безпосередньо підлеглих, то дана структура забезпечує пряме представлення відношення “багато до багатьох”. Для зв'язку між записами-вузлами існує зв’язуючий запис, всі екземпляри якого поміщаються в ланцюжок для зв'язку двох екземплярів.

Основною конструкцією сіткової моделі даних є набір. Для кожного типу набору, визначуваного в схемі, повинен бути вказаний певний тип запису власника набору, а так само довільне число типів запису членів набору. Кожен екземпляр набору складається з одного екземпляра-власника і одного або більше екземплярів записів-членів.

Кожен екземпляр запису-набору представляє ієрархічні зв'язки між екземпляром запису-власника і відповідними екземплярами записів-членів. Це є наслідком того обмеження, що жоден екземпляр запису-члена з набору не може належати більш, ніж одному екземпляру набору. Спосіб, яким кожен екземпляр запису власника зв'язується з відповідними екземплярами записів-членів, визначається в схемі сітки. Одним із способів організації таких зв'язків є встановлення ланцюжка вказівників, що виходять з екземпляра запису-власника, записів-членів, і які проходять через всі екземпляри, а потім повертаються назад до екземпляра запису-власника, що забезпечує високу швидкість обробки запитів.

Головний недолік сіткової моделі полягає в складності структур пам'яті. Користувач повинен знати, які ланцюжки існують і які відсутні. В результаті мова запитів процедурна і вимагає навиків програмування.

2.1.3 Реляційна модель даних

Реляційна модель набір нормалізованих відношень, які складаються з кортежів, кожен з яких – набір значень, що відповідають обмеженням домену атрибуту. Домен – це множина допустимих значень атрибуту, з якої значення для даного атрибуту беруться для запису. Тобто в основі реляційної моделі є звичайні двовимірні таблиці, де на перетині кожного рядка і стовпця знаходиться одне неподільне значення. Кожен кортеж унікальний в рамках свого відношення, а порядок опрацювання атрибутів не має значення. Декілька (щонайменше один) атрибутів служать для ідентифікації кортежів і називаються ключами. Виділяють первинні ключі (ідентифікують кортежі) і потенційні (можуть виконувати функцію ідентифікації кортежів).

Основний недолік реляційної моделі даних пов'язується з низькою продуктивністю реляційної СУБД. Але розробка сучасних СУБД таких як, ORACLE, InterBase, MySQL і ін. дозволила подолати і цей недолік.

Достоїнства реляційної моделі можна розділити на дві групи:

1) достоїнства для користувача:

– реляційна БД є набором даних у вигляді двовимірних таблиць, які представляють користувачеві звичні сутності з його предметної області;

– реалізація запитів користувача повністю покладена на СКБД;

– мова запитів SQL набагато простіша за мови запитів до інших баз даних, орієнтованих на використання розробниками, а не звичанийми користувачами;

2) достоїнства обробки даних реляційної БД:

– зв'язки між таблицями реляційної бази даних відображають бізнес-процеси, в яких задіяні зв'язані типи сутностей;

– зв'язки підтримуються засобами СКБД для збереження цілісності даних під час їх опрацювання;

– виконання операцій з'єднання відношень дозволяють виконати вибірку інформації за один запит більше, ніж з одного відношення;

– керування доступом на основі ідентифікаторів користувачів, їхніх ролей та наданих привілеїв;

– файлові структури даних реляційної бази даних набагато простіше реалізувати, ніж для інших моделей даних;

– реляційна модель даних допускає зміну без написання нового програмного коду при умові збереження елементів моделі, на котрі зав'язаний існуючий програмний код. Допускається використання віртуальних відношень для забезпечення нового рівня абстрагування прикладної програми від моделі даних.

Тому для написання прикладної програми в кваліфікаційній роботі вибираємо реляційну СКБД.

2.2 Вибір концептуальної моделі

Модель “сутність-зв'язок” відповідно до своєї назви містить набір типів сутностей, що відображають поняття предметної області, та зв'язки, що відображають бізнес-процеси, в котрих задіяні ці типи сутностей. Найкращим засобом представлення цієї моделі є ER-діаграма. Кожен тип сутності характеризується набором атрибутів, означених на своїх доменах. Зв'язки характеризуються кількістю екземплярів кожного типу сутності і бувають трьох типів: “один до одного”, “один до багатьох”, “багато до багатьох”. Реляційна модель реалізує перші два типи зв'язків. Для утворення тертього типу вводять проміжний тип сутності і декомпозиції зв'язку "багато до багатьох" на два з кардинальністю "один до багатьох".

2.3 Побудова моделі

2.3.1 Виділення сутностей

Сутність “постачальник” є основною сутністю моделі, що розробляється. З постачальником укладається договір, на підставі якого ведеться подальша діяльності підприємства з постачання, відправлення заявки постачальникам, отримання від них рахунку-фактури, відправлення замовлення на постачання, отримання товару, оплата постачання. Як ключ для даної сутності вводиться атрибут № Постачальника.

Всі сутності, їх атрибути і ключі представлені в таблиці 2.1.

Таблиця 2.1 – Виділення сутностей

Назва сутності	Атрибут	Ключ
Договір	№договору, дата договору, сума договору, термін дії.	№ договору
Постачальник	№постачальника, найменування постачальника, адреса, телефон.	№ Постачальника
Асортимент товарів	№товару, найменування товару.	№ Товару
Заявка	№Заявки, ассортимент заявки, номер договору, дата заявки.	№ Заявки
Замовлення	№Заказу, №Договору, ассортимент замовлення, дата замовлення, номер рахунку.	№ Заказу
Рахунок-фактура	№Рахунку, ассортимент рахунку, ціна за одиницю товару, сума рахунку.	№ Рахунку

Виділення зв'язків між сутностями здійснюється на підставі аналізу предеметної області. Всі виділені зв'язки представлені на рисунку 2.1.

Виконавши аналіз сутностей і зв'язків між ними, побудуємо логічну модель у вигляді відношень (рисунок 2.2)

2.4 Аналіз алгоритмів роботи з базою даних

Система управління розробленої БД використовує реляційний підхід для побудови бази даних. Подібні системи засновані на реляційній моделі даних. Застосування реляційної моделі даних обумовлено використанням реляційної алгебри і відповідних алгоритмів і операцій для виконання дій над даними. Використання алгоритмів реляційної алгебри дозволяє забезпечити високу продуктивність роботи з базою даних.

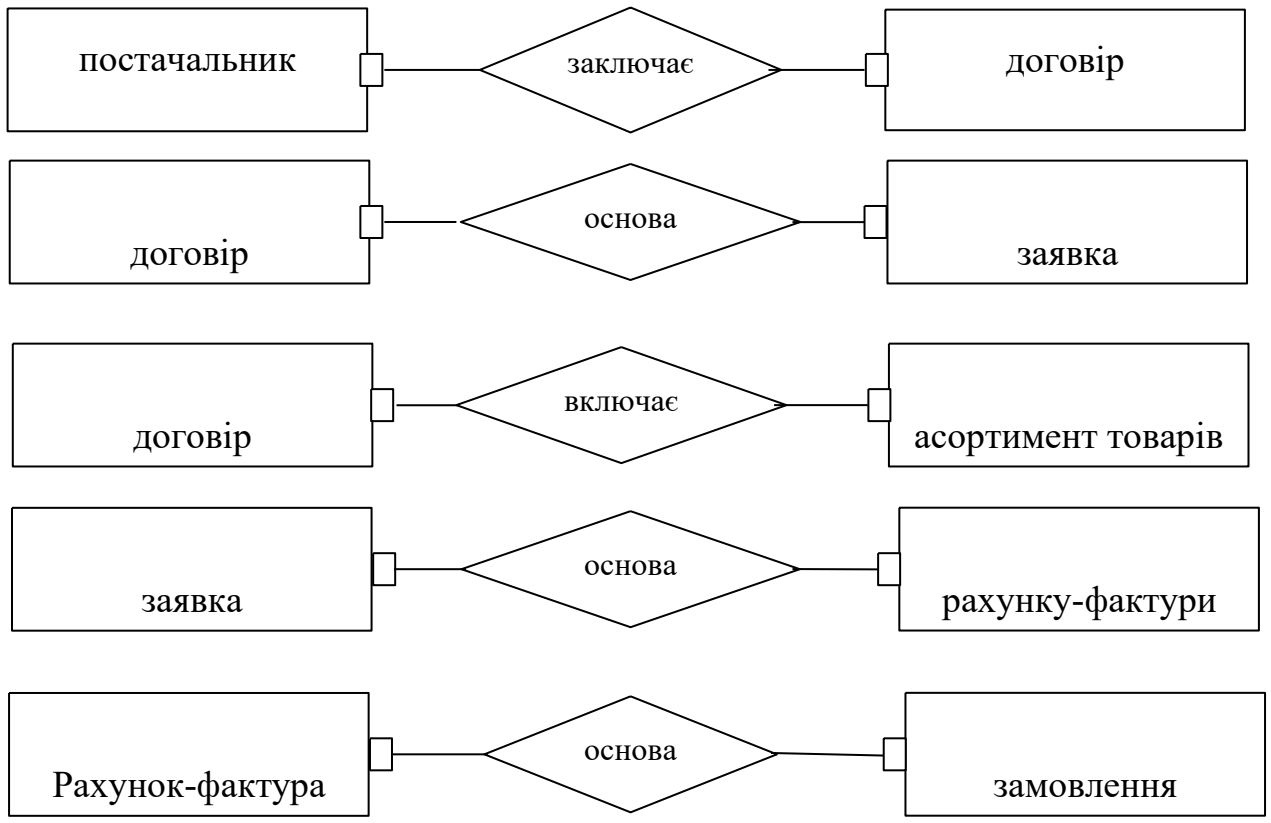

Рисунок 2.1 – Зв'язки між сутностями

Основні операції реляційної алгебри були вперше запропоновані Коддом. Він довів, що запити, що формулюються за допомогою мови числення, можуть

бути сформульовані в мовах реляційної алгебри і навпаки, тобто запити представлені за допомогою мови реляційної алгебри можуть бути використані для виконання запитів до розробленої БД. В додатку Г приведені запити до БД.

Для побудови логічної моделі даних використовувалося CASE-засіб ERWin, який дозволяє проектувати реляційні моделі даних як на логічному рівні (ER-діаграми), так і на фізичному (проектування таблиць БД).

Логічна модель даних представлена у вигляді ER-діаграми на рисунку 2.2.

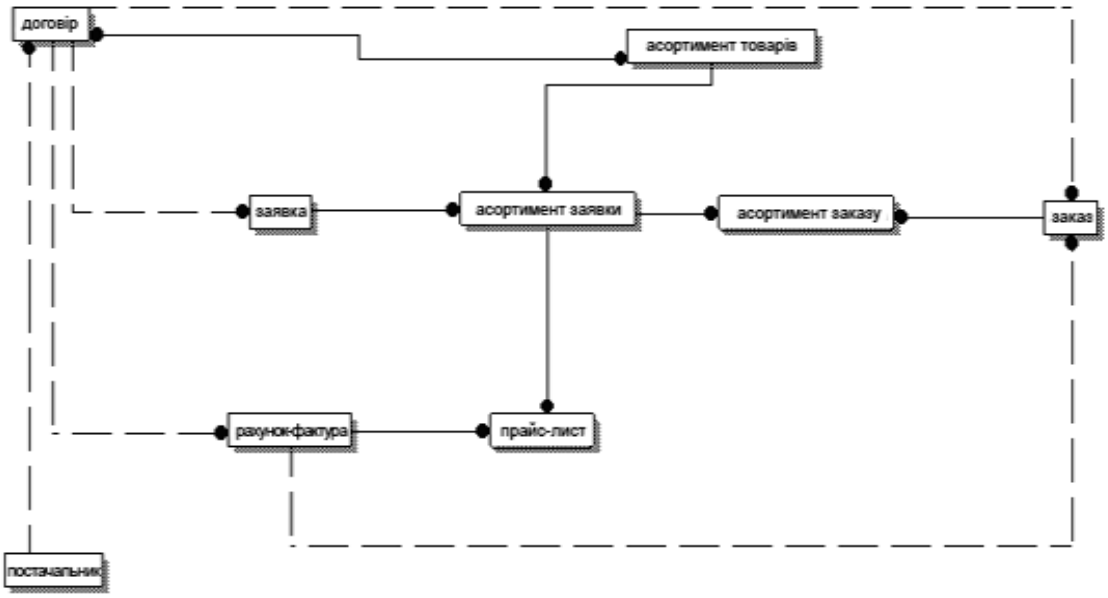

Рисунок 2.2 – ER-діаграма моделі даних автоматизованої довідково-інформаційної системи “Контроль та облік збуту”

Розглянемо чотири операції над відношеннями:

- вибірка;
- проекція;
- теоретико-множинне об’єднання;
- з’єднання.

Вибірка (selected_on – піддані селекції по) зменшує кількість рядків в таблиці, і її можна представити як результат розрізання таблиці по горизонталі і видалення непотрібних кортежів. Формально селекція записується так:

R selected_on [<предикат>] { синтаксис мови запитів (SQL)}

Тут <предикат> – це логічний вираз, який може містити порівняння значень одних атрибутів із значеннями інших в тому ж кортежі або з константами. В результаті зберігаються тільки рядки, що задовільняють <предикат>.

Операція вибірки відповідає програмам, які вибирають записи з файлів і друкують ці записи. Проте умови відбору можуть відноситься тільки до окремо взятих записів. Наприклад, неможливо вибрати запис, виходячи з того, що значення якого-небудь його поля рівне або більше, ніж значення цього поля в попередньому записі. Насправді майже неможливо змоделювати поведінку автомата з кінцевим числом станів, який змінює свій стан для кожного запису, змінюючи тим самим критерії відбору для наступного запису.

Проекція (projected_to – спроектоване на) зменшує кількість стовпців в таблиці; дану операцію можна уявити собі як розрізання по вертикалі. Назва операції має своїм джерелом поняття проекції множини точок N-мірного простору в простір з меншою кількістю вимірів. Наприклад, в результаті проекції множини точок площини (Х,У) на вісь Х виходить множина точок, розташованих на цій осі. На жаль, значення проекцій деяких “точок” можуть співпадати; це відбудеться у тому випадку, коли проекція видалить стовпець, що входить в ключ, так що частини двох “укорочених” кортежів, що залишилися, можуть бути ідентичними. Тоді доведеться видалити дублікати і тим самим зменшити кількість рядків, тобто розмір БД. Якщо хоч би один з можливих ключів при виконанні проекції залишиться незачепленим, то дублікатів не буде.

Формально проекція записується таким чином:

R projected_to <ім’я-атрибуту>{, <ім’я-атрибуту>}

Де список <імен-атрибутів> означає імена стовпців, що зберігаються.

Операція проекції відповідає програмі відбору дещо іншого роду, ніж операція вибірки, а саме, вона друкує певні поля з кожного запису. Видалення дублікатів зазвичай досягається в результаті сортування записів по необхідних

полях, після чого записи пропускаються до тих пір, поки не зміниться значення поля. На практиці при одному перегляді файлу операція проекції як правило відбувається з операцією вибірки.

Теоретико-множинне об’єднання (union) є двохопреандною операцією; ця операція аналогічна операції об'єднання множин. Це можливо лише у тому випадку, коли обидві таблиці мсумісні по об'єднанню, тобто мають набори атрибутів , попарно означені на однакових доменах. Такі таблиці називають “сумісними по об’єднанню”. Всі дублікати рядків повинні бути видалені з відношення-результату. Дана операція аналогічна об’єднанню множин в алгебрі, але вона є додатковою по відношенню до обмеження, оскільки є можливість відновити відношення шляхом об’єднання двох взаємодоповнюючих результатів операції вибірки.

З’єднання (joined_to – з’єднання з) має два операнди; воно визначене для будь-яких двох таблиць. Якщо ці дві таблиці не мають стовпців із співпадаючими іменами, то з’єднання поводиться, як декартовий добуток, сполучаючи кожен рядок першої таблиці по черзі з кожним рядком другої таблиці. Якщо імена всіх стовпців цих двох таблиць співпадають, то з’єднання поводиться як теоретико-множинний перетин, і створює таблицю, що складається з тих рядків, які зустрічаються в кожній з даних двох таблиць (така таблиця може бути і порожньою, аналогічно порожній множині). Якщо у двох таблиць-операндів співпадають лише деякі імена стовпців, то в результаті з’єднання виходить таблиця, що містить всі імена стовпців першої таблиці, а також всі ті імена стовпців другої таблиці, які не зустрілися в першій. Рядки результату вибираються з першої таблиці, а додаткові значення конкатенуються (приєднуються) з тих рядків другої таблиці, у яких значення в загальних стовпцях співпадають. До деякої міри з’єднання є доповненням проекції, якщо здійснити проекцію “початкового” відношення так, щоб вийшов набір відношень, кожне з яких зберігає первинний ключ результуючого, то з’єднання

цього відношення відновить результуюче за додаткової умови, що кожен стовпець початкового відношення зустрічається хоч би в одній з проекцій.

При формулюванні запитів операція з'єднання є вирішальною, якщо в запиті використовується більше, ніж одне відношення. Як правило, для формування запиту використовується з'єднання декількох таблиць, а потім вибірка необхідних рядків, і, нарешті, проекція на необхідні стовпці при друці.

Операція з'єднання більше всього відповідає операції "селективної вибірки", при виконанні якої список ключів представлений у вигляді записів у файлі транзакцій, і потрібно вибрати або записати у вихідний файл відповідні записи з основного файлу. Ключі у файлі транзакцій можуть співпадати, наприклад, із стороннім ключем в основному файлі або ж з частиною первинного ключа, і в цих випадках для кожного запису у файлі транзакцій може бути вибране декілька записів з основного файлу. Таким чином, використовується з'єднання як узагальнений перетин.

Алгоритми, які виконують вище перелічені операції, реалізуються на рівні системи управління базою даних. Їх зміст формується на основі визначень цих операцій. Для їх реалізації використовуються або стандартні функції мови програмування, або формується SQL-запит.

2.5 Аналіз і вибір засобів розробки системи

Сучасні засоби розробки програмного забезпечення характеризуються рядом характеристик, відповідно до яких розробник може вибрати потрібний інструмент та створити потрібний програмний продукт з мінімальними затратами часу та інших ресурсів. Всі засоби розробки дозволяють:

– створити елементи інтерфейсу користувача та скопонувати їх на основі базових наборів класів для цих елементів;

– окремо реалізувати кожен бізнес процес у вигляді окремого компоненту на налагодити взаємодію між цими компонентами;

– створювати засоби програмного доступу до баз даних і самі бази даних також;

– використовувати механізм обробки виключних ситуацій для підвищення надійності створюваного програмного продукту.

Сучасні засоби розробки, як правило, володіють такими параметрами:

– такі системи розробки фактично є CASE-засобами створення програмного коду та баз даних;

– наявність візуальних компонентів побудови програми, інтерфейсу користувача та елементів доступу до баз даних і мереж;

– візуальне представлення даних при підключенні до БД;

– інтеграція з системами контролю версій при командній роботі над проектом.

Таким чином, основна відмінність засобів розробки ПЗ полягає призначенні якої саме області вони найкраще підходять.

Виходячи з приведених вимог, виділимо наступні характеристики засобів розробки програмного забезпечення:

– наявність досвіду роботи з запропонованими інструментами створення ПЗ;

– вимоги до ресурсів комп'ютера;

– програмна платформа для даного інструменту;

– доступність;

– час на розробку програмного забезпечення, підтримка автоматичної генерації програмного коду.

Тому для вибору як мови програмування, так і середовища розробки будемо користуватись переліком характеристик, показаних в таблиці 2.2

Таблиця 2.2 – Результати вибору мови програмування

Засіб розробки / Характеристика засобів розробки	PHP	Perl	ASP	Delphi
Наявність досвіду розробки з використанням даного програмного продукту;	8	6	4	4
Вимоги по ресурсам;	7	6	6	5
Підтримка операційної системи;	8	8	8	7
Наочність розробки інтерфейсу;	9	7	8	5
Можливості роботи, що надаються, з базами даних;	8	6	4	7
Швидкість роботи розробленого програмного забезпечення;	6	7	8	7
Обробка виняткових ситуацій;	8	8	8	6
Час створення розробленого програмного забезпечення;	9	6	5	7
Зручність експлуатації;	7	8	8	7
Всього:	70	62	60	56

В результаті проведеного аналізу виявлено, що для задачі створення програмного продукту відповідно до завдання кваліфікаційної роботи є мова PHP.

Основні відомості про PHP.

PHP – (офіційно "PHP: Hypertext Preprocessor") – платформо-незалежна, що виконується на сервері, HTML зв'язана мова скриптів.

Місця вставки коду починаються і закінчуються спеціально певними тегами. В інакшому випадку PHP видаватиме помилку і скрипт не виконуватиметься.

PHP краще від Javascript тим, що PHP виконується на сервері, а Javascript – на машині клієнта. Наприклад, ніхто не зможе подивитися код PHP скрипта без бажання розробника – видно лише результат його роботи на відміну від "досяжного" JavaScript.

WWW сервер можна налаштувати таким чином, що будь-яка WWW сторінка з розширенням *.html оброблятиметься PHP процесором і навіть не буде відомо PHP скрипт.

PHP здатний робити все, що роблять інші CGI програми. PHP скрипти можуть збирати і виконувати опрацювання дані, передані із веб-форм, динамічно генерувати вміст веб-сторінок, працювати з файлами "cookies", з сесіями і т.п.

Але, мабуть, найсильніша і найбільш розвинена сторона PHP – це робота з базами даних. Написання скрипта, який взаємодіє з базою даних, – завдання дуже нескладне, якщо робити це на PHP. У PHP добре розвинена підтримка різних мережевих протоколів прикладного рівня для роботи з різними сервісами Інтернет.

2.6 Опис загальної структури автоматизованої довідково-інформаційної системи

Схема функціонування системи відображена на рисунку 2.3.

В склад автоматизованої довідково-інформаційної системи входить: користувач, інтерфейс користувача, трансляція, інтерфейс БД, сама база даних, СКБД, формування документів.

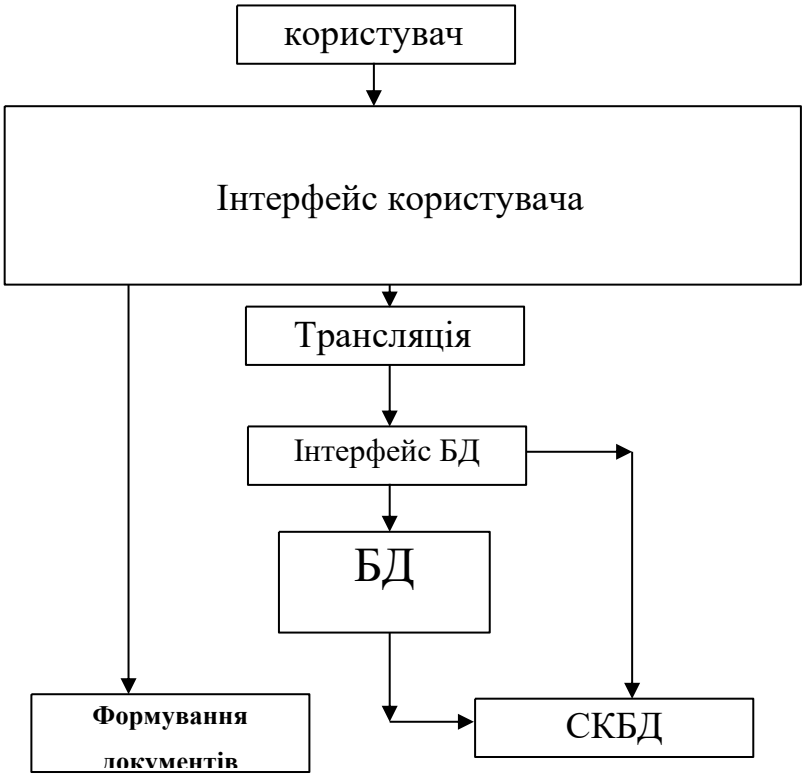

Рисунок 2.3 – Схема функціонування системи

2.6.1 Опис інтерфейсу

Після запуску адресу http://127.0.0.1/ на моніторі з'являється головне меню (рисунок 1.4).

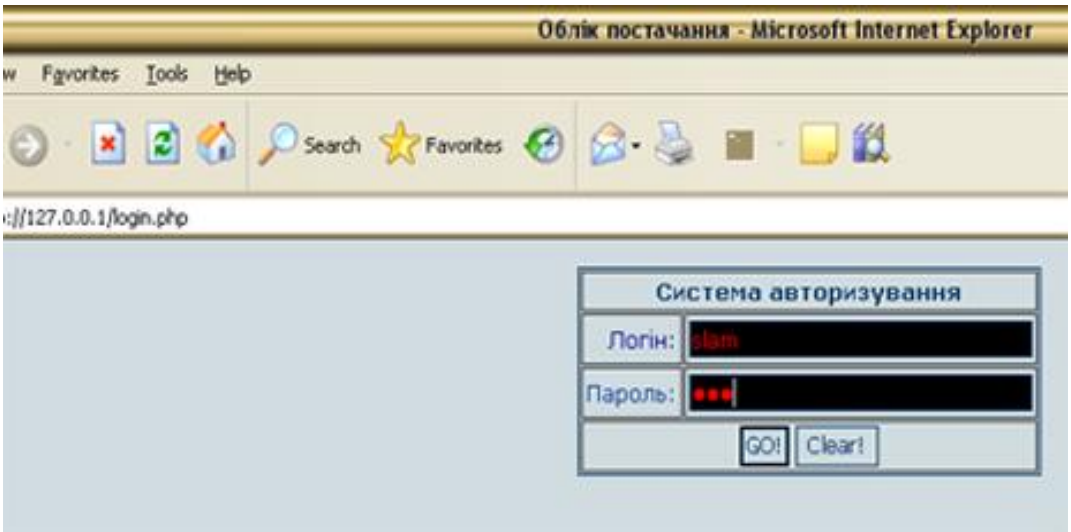

Рисунок 2.4 – Головне меню

Для початку роботи з програмою необхідно ввести логін і пароль оператора і клацнути по команді меню після чого відбудеться з'єднання з БД

Логін – Slam. Пароль доступу – 123.

У випадку, якщо з'єднання пройшло успішно, то користувач допускається до роботи з автоматизованої довідково-інформаційної системи.

Робоче вікно автоматизованої довідково-інформаційної системи подано на рисунку 2.5

Робота з договорами включає:

– роботу з постачальниками;

– роботу з договорами;

– роботу з товарами;

– роботу з укладеними договорами;

– роботу з асортиментом договорів.

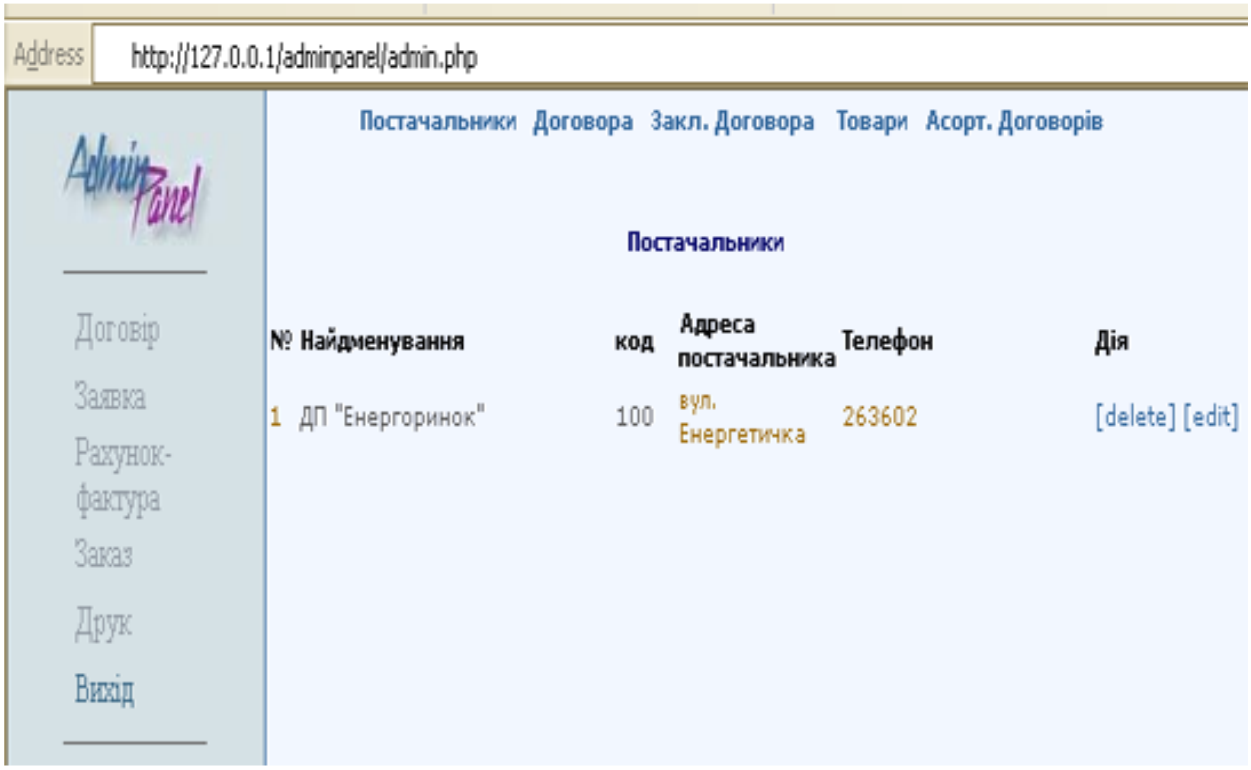

Рисунок 2.5 – Робоча область а системи

Договір укладається підприємством-замовником з підприємством-постачальником на постачання певного виду і асортименту продукції. З одним постачальником може бути укладено декілька договорів. Атрибутами договору є наступні поля: номер договору, код постачальника, дата договору, сума договору, термін дії договору. Всі атрибути, окрім терміну дії договору є обов'язковими для заповнення. На підставі договору проводиться подальша діяльність по постачанню підприємством. Вона полягає в:

- роботі із заявками;
- роботі з рахівницями;
- роботі із замовленнями.

Для автоматизації використання автоматизованої довідково-інформаційної системи “Контроль та облік збуту” реалізована можливість друку бланків документів договору, заявки, замовлення.

2.6.2 Робота з таблицями бази даних

Додавання нового договору здійснюється шляхом вибору відповідної сторінки і введенні тексту в поля-атрибути таблиці. Додавання за умови, що для договору, що додається, відомий постачальник.

Редагування відбувається при натисненні на посилання Edit на вибраному записі. Відбувається автоматична зміна всіх полів інших таблиць пов'язаних з номером редагованого договору. Ця зміна необхідне для підтримки посилальної цілісності в БД.

Для видалення певного договору необхідно клацнути лівою кнопкою миші на посиланні Delete навпроти договору, що видаляється. Автоматично віддаляться всі записи пов'язані з договором, що видаляється (заявки, рахунки-фактури, замовлення).

Робота з партнерами полягає в додаванні нового постачальника чи покупця, його атрибутів, видаленні постачальника (покупця), редагуванні

атрибутів постачальника: код постачальника (для кожного постачальника код унікальний), найменування постачальника (покупця), адреса і телефон. Всі атрибути, окрім телефону є обов'язковими для заповнення, у разі їх незаповнення виникає помилка.

Додавання постачальника (покупця) проводиться таким чином: користувач вибирає відповідну таблицю і заповнює атрибути.

Для редагування таблиці “постачальники” потрібно вибрати запис для редагування, нажати на посилання Edit навпроти і відредактувати необхідну інформацію. Змінені атрибути постачальника автоматично змінюються в інших таблицях.

Видалення запису “постачальник” відбувається шляхом клацання мишею на посиланні Delete навпроти запису, що видаляється. При цьому потрібен запит на підтвердження видалення запису.

Таблиця “товари” є довідник товарів, які поставляються на підприємство. Атрибути цієї таблиці містять унікальний код для кожного товару і найменування товару. При укладенні кожного нового договору необхідно заповнити таблицю асортимент договору.

Додавання нового запису в таблицю здійснюється шляхом введення інформації про товар в рядки таблиці товари. Редагування – посилання Edit навпроти редагованого рядку.

Видалення – Посилання Delete .

Робота з таблицею про укладені договори для користувача обмежена, оскільки даними для її заповнення служать раніше заповнені таблиці (договір, постачальник).

Робота з асортиментом договорів полягає в додаванні, редагуванні і видаленні найменування товару або товарів, які постачальник зобов'язується поставити замовнику на підставі переліку товарів, що поставляються, указуваному в укладеному договорі. Вищезгадані операції проводяться аналогічно операціям в роботі з договорами.

Веб-сторінка “заявка” містить таблицю для роботи з замовленнями на основі укладених договорів. В цій таблиці наявні всі їх характеритстики: номер, договір, дата створення заявки. Всі поля обов'язкові ля заповнення. Номер договору один ув'язнених, інакше генерується виключна ситуація.

Користувач має можливість виконувати операції додавання, витирання та редагування записів таблиці.

Сторінка “асортимент товарів” виводить таблицю з відомостями про склад поточної заявки, тобто список товарів відповідно до договору. По аналогії до попереднього випадку всі атрибути обов'язкові. Робота з записами здійснюється за допомогою таких же інструментів інтерфейсу користувача, що і при роботі з заявками.

Для роботи з рахунками створена сторінка “рахунок-фактура”. У цій таблиці вказані всі атрибути рахунків, а саме: номер, ідентифікатор заявки, ідентифікатор договору, загальна сума рахунку. Також всі атрибути мають бути заповнені без порожніх значень.

Для опрацювання замовлень пропонується дві сторінки для роботи з окремим замовленням та зі списком всіх замовлень:

Сторінка “друк” служить для формування друкованих звітів по складених договорах, замовленнях, їх оплаті. У додатках приведений програмний код розробленої системи.

2.7 Тестування програмного продукту

Метою тестування програмного продукту та його модулів є встановлення невідповідності між реалізованою в програмному коді бізнес-логікою, з одного боку, та специфікацією вимог (функціональних і нефункціональних), з іншого. Розробка тестів для створеного продукту можна поділити на такі чотири етапи:

1. На основі специфікації вимог були розроблені набори тест-кейсів для кожного пункту вимог з використання позитивних і негативних тестів з

використанням методів виділення зон еквівалентності, перевірки граничних значень.

2. Розроблено і виконано аналіз покриття тестами всіх інструкцій програмного коду, перевірка умов розгалуження, завершення циклів. Тобто використано метод "білого ящика".

3. Перевірено роботу циклів з використанням програмних "заглушок" для максимальної кількості повторів для циклів з лічильником.

4. Виконано планування та виконання тест-кейсів для того, щоби згенерувати та перехопити усі виключні ситуації.

Слід зазначити, що процес компіляції є теж свого роду тестуванням статичного типу, тобто перевірка програмного коду на наявність помилок, неправильно описаних ідентифікаторів тощо.

Після виконання тестів було здійснено виправлення всіх виявлених дефектів, усунено помилки логічного характеру, зациклювання, вихід за межі діапазонів.

Остаточним тестуванням була перевірка працездатності програми з точки зору користувача. Це свого роду аналог альфа та бета-тестування на системному рівні. При цьому з використанням спеціальних інструментів перевірено роботу програмного продукту в специфічних умовах роботи, а саме:

1. Тестування стресових станів, коли програмний продукт працює в режимах з максимальними відповідно до вимог навантаженнями. Генеруються тестові набори даних для цього і імітується підключення певної кількості клієнтів на певний момент часу. Для проведення цих тестів до БД ставилось одночасно до 20 запитів.

2. Тестування об'єму передбачає перевірку коректності роботи програми з великими обсягами даних протягом тривалого часу. Виявляються недоліки роботи з пам'яттю та іншими системними ресурсами.

3. Конфігураційне тестування передбачає перевірку роботи програмного продукту при різних налаштуваннях парамерів програми та при

різних обсягах системних ресурсів. Максимальна кількість тест-кейсів тут дуже велика, але тести були розроблені таким чином, щоби охопити різні режими роботи системи мінімальною кількістю тестів.

4. Тестування безпеки – окремий вид тестування. Який потребує спеціальних знань та інструментів. Для онлайн-систем виконується перевірка коректності аутентифікації, вразливість до ін'єкцій різного роду, крос-браузерні вразливості.

2.8 Висновки за результатами тестування ПЗ

За результатами проведених і описаних вище тестів можна зробити ряд висновків.

1. Розроблений продукт реалізує всі функціональні вимоги відповідно до специфікації.

2. Завдяки механізму керування транзакціями при збоях підключення до бази даних цілісність інформації не порушується.

3. Перевірено роботу і працездатність системи для різних обсягів системних ресурсів.

4. Завдяки використанню фреймворків захисту забезпечено вказаний у вимогах рівень безпеки даних.

5. Програмний продукт зберігає працездатність при підключенні максимальної кількості клієнтів (до 20 запитів одночасно) та з максимальним обсягом бази даних.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Загальні вимоги законодавства з охорони праці в галузі інформаційних технологій

Правову основу охорони праці становить Конституція України як за своїми юридичними особливостями, так і своїми принципами, тобто юридично вираженими об'єктивними закономірностями організації і функції соціально-економічної, політичної, духовної сфер суспільства, правового положення особи.

Конституційні норми, з одного боку, закладають суть безпеки (норми-принципи), а з іншого, – вказують на цілі подальшого розвитку і реалізацію правового забезпечення безпеки життєдіяльності (норми-програми, норми-завдання, норми-зобов'язання).

Реалізація і розвиток основних конституційних положень, які регламентують суспільні правовідносини, безпосередніми суб'єктами яких є особа і держава, здійснюється за допомогою як чинних фундаментальних нормативно-правових актів, так і спеціальних (Кодексів України про працю, Закону "Про охорону навколишнього природного середовища" та ін.)

Поруч з нормативними актами, які прийняті вищим законодавчим органом держави, для встановлення взаємозв'язків, усунення програм, а в ряді випадків і реалізації окремих правових норм або їх елементів, до правової бази безпеки життєдіяльності належать спеціальні акти, розроблені за дорученням виконавчих державних органів усіх рівнів (Кабінет Міністрів, Міністерства, Державні Комітети та ін.).

Так, наприклад, "Положення...", які розвивають Закон України Про охорону праці, діляться на звичайні "Положення" і "Типові положення". Тут держава розподілила питання своєї прерогативи стосовно розробки нормативних

актів і прерогативи своїх повноважень стосовно контролю, "правового простору" у вигляді нормативних актів підприємств.

З іншого боку, формуючи систему "Типових положень" держава на сьогоднішній день ліквідує прогалини в чинному законодавстві, узгоджує взаємозв'язки між суб'єктами правовідносин, створює юридичну базу для удосконалення і розвинення "правового поля" підприємств.

Кожний нормативно-правовий документ має свою структуру, яка визначає собою ідею систематизації відповідно зі своїм рівнем, метою та завданнями. Відповідно до цього в кожному нормативному акті є елементи, що відповідальні за зовнішній його зв'язок і створення передумов для відповідного розвинення за рахунок розробки нижчих нормативно-законодавчих актів. Сама структура нормативного акта формує відповідні внутрішні зв'язки.

Основними систематизуючими ланками нормативних актів безпеки життєдіяльності (які за ієрархією знаходяться нижче законів) є встановлення взаємовідносин в галузі виробництва, в межах дії небезпечного фактора (в тому числі і факторів довкілля), а також відносно управління основних технологій безпеки життєдіяльності (розслідування нещасних випадків, навчання, організації робіт та ін.).

Узагальнюючими ланками систематизації на рівні держави є національна ідея, взаємовідносини в суспільстві, соціально-економічне і політичне становище держави, можливості сприймання і використання законодавчих актів з боку споживачів та ін.

4.2 Землетруси і порядок дії населення при них

Чинники небезпеки землетрусів: руйнування будівельних конструкцій будинків та споруд; руйнування на потенційно небезпечних об'єктах, нафто- та газопроводах; утворення завалів; руйнування систем життєзабезпечення та розлами земної кори. Додатковою небезпекою є повторні поштовхи. Ми

пропонуємо заходи, необхідні для виконання кожній людині, яка може опинитися у можливій зоні землетрусу.

Дії у випадку загрози виникнення землетрусу:

Уважно слухати інформацію про обстановку та інструкції про порядок дій, не користуватись без потреби телефоном.

Зберігати спокій, попередити сусідів, надати допомогу інвалідам, дітям та людям похилого віку. Навчити дітей, як поводитися під час землетрусу. Дізнатися у місцевих органів державної влади та місцевого самоврядування місце збору мешканців для евакуації. Завчасно вирішити, де буде місце зустрічі родини у разі евакуації.

Одягнутись, взяти документи та зібрати найбільш необхідні речі, невеликий запас продуктів харчування на декілька днів, питну воду, медикаменти, кишеньковий ліхтарик.

Від'єднати всі електроприлади від електромережі, вимкнути газ та систему нагрівання.

Поставити на підлогу важчі та великі речі. Закріпити речі, що можуть впасти і спричинити травми. Не ставити ліжко біля вікна з великим склом.

Тримати у зручному місці один або декілька вогнегасників. Тримати шланги для поливу саду підключеними до кранів.

З'ясувати, чи не перебуває житло або місце роботи під загрозою затоплення (у разі руйнування греблі), зсуву або дії іншого стихійного лиха.

Вивести худобу на більш безпечну місцевість.

Дії під час землетрусу:

Зберігати спокій, уникати паніки.

Діяти негайно, як тільки відчуються коливання ґрунту або споруди, головна небезпека, яка загрожує, — це предмети й уламки, що падають.

Швидко залишити будинок та відійти від нього на відкрите місце, якщо ви перебуваєте на першому – другому поверсі.

Негайно залишити кутові кімнати, якщо ви перебуваєте вище другого поверху.

Негайно перейти у більш безпечне місце, якщо ви перебуваєте у приміщенні. Стати в отворі внутрішніх дверей або у кутку кімнати, подалі від вікон і важких предметів.

Не кидатись до сходів або до ліфта, якщо ви знаходитесь у висотній споруді вище п'ятого поверху. Вихід зі споруди найбільш буде заповнений людьми, а ліфти вийдуть з ладу.

Вибігати з будинку швидко, але обережно. Остерігатись уламків, електричних дротів та інших джерел небезпеки.

Віддалитись від високих споруд, шляхопроводів, мостів та ліній електропередач.

Зупинитись, якщо ви їдете автомобілем, відчинити двері та залишатись в автомобілі до припинення коливань.

Перевірити, чи немає поблизу потерпілих, сповістити про них рятувальників та за можливості надати допомогу.

Дії після землетрусу:

Зберігати спокій, заспокоїти дітей та тих, хто дістав психічну травму в результаті землетрусу, оцінити ситуацію.

Допомогти за можливості потерпілим, викликати медичну допомогу тим, хто її потребує.

Переконатись, що житло не зазнало ушкоджень. Бути дуже обережним, може статися раптове обвалення, загрожує небезпека від витоку газу, від ліній електромереж, розбитого скла.

Перевірити зовнішнім оглядом стан мереж електро-, газо- та водопостачання.

Обов'язково кип'ятити питну воду, вона може бути забруднена.

Перевірити, чи немає загрози пожежі.

Не користуватись відкритим вогнем, освітленням, нагрівальними приладами, газовими плитами і не вмикати їх до того часу, доки не будете впевненості, що немає витоку газу.

Не користуватись довго телефоном, окрім як для повідомлення про серйозну небезпеку.

Не поспішати з оглядом міста, не відвідувати зони руйнувань, якщо там не потрібна допомога.

Уникати морського узбережжя, де може виникнути небезпека від морських хвиль, спричинених сейсмічними поштовхами.

Бути готовим до повторних поштовхів. Часто вони призводять до додаткових руйнувань.

Дізнатися у місцевих органів державної влади та місцевого самоврядування адреси організацій, які відповідають за надання допомоги потерпілому населенню.

ВИСНОВОК

В рамках кваліфікаційної роботи була розроблена автоматизована довідково-інформаційна система для контролю та облік постачань готової продукції. В результаті можна зробити такі висновки:

При розробці системи був пройдений повний цикл проектування програми від постановки завдання замовником до здачі програмного забезпечення в експлуатацію.

Розроблена система дозволяє досягти наступних ефектів:

– зменшення часу для обліку поставок;

– автоматизація контролю постачань з формуванням документів;

– можливість зберігання даних про роботу підприємства з можливістю використовувати їх для аналітичного аналізу;

– отримання інформації про оплату за замовлення.

Також під час створення системи були досліджені питання безпеки життєдіяльності та хорони праці розробника ПЗ та запропоновані умови з їх поліпшення.

На підставі можна зробити висновок про те, що розробка системи є доцільною і приноситиме реальну користь при використанні її на підприємстві.

ПЕРЕЛІК ПОСИЛАНЬ

1. Афіцький Сергій Іванович. HTML: навч. посіб.. – Кам'янець-Подільський : Аксіома, 2007. – 80с.

2. Буров Євген. Комп’ютерні мережі: 2-ге оновл. і доп. вид. / В. Пасічник (ред.). – Л. : БаК, 2003. – 584с.

3. Дейт К. Дж.. Введение в системы баз данных: Пер.с англ.. – 6.изд. – СПб.: Издательский дом "Вильямс", 2000. – 848с. – (Системное программирование). – На тит.л. места изд.: Москва, Санкт-Петербург, Киев.

4. Жидецький Валерій Цезарійович. Охорона праці користувачів комп'ютерів / Українська академія друкарства. – Л. : Афіша, 2000. – 174с. : іл. – Бібліогр.: с. 169-172.

5. Коннолли Томас, Бегг Каролин. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Р.Г. Имамутдинова (пер.с англ.), К.А. Птицына (пер.с англ.). – 3-е изд. – М. ; СПб. ; К. : Издательский дом "Вильямс", 2003. – 1439с. : рис. – Библиогр.: с. 1395-1426.

6. Котеров Д. В. Самоучитель PHP 4: Язык программирования для создания эффективных Web- приложений. – СПб. : БХВ-Петербург, 2003. – 553с. – (Самоучитель).

7. Мельник Роман Андрійович. Програмування для WEB- та SQL-серверів (PHP та Perl): Навч. посіб. для студ. усіх форм навч. напряму "Комп'ютерні науки" / Національний ун-т "Львівська політехника". – Л. : Видавництво НУ "Львівська політехніка", 2006. – 132с. – Бібліогр.: с. 131.

8. Ніколаєнко О. Ю., Кравченко С. М., Вернигоренко С. А.. Використання HTML та JavaScript для створення Web-документів: Посібник для студ.. – К. : НПУ ім. М. П. Драгоманова, 2003. – 112с. – Бібліогр.: с. 110.

9. Сетевые средства Windows NT: Windows NT Workstation и Windows NT Server версия 3.5 / Елизавета Кароник (отв.ред.), Ольга Кокорева (пер.). – СПб. : BHV - Санкт- Петербург, 1996. – 496с. – (Ресурсы Windows NT).

10. Спортак Марк А., Паппас Франк Ч., Рензинг Эмиль, Мастерс Луис С., Блай Мартин Дж.. Высокопроизводительные сети: Энциклопедия пользователя. – К. : ДиаСофт, 1998. – 421с.

11. Стахнов Алексей А.. Linux: Наиболее полное руководство. – СПб. : БХВ-Петербург, 2003. – 881с.

12. Хант Крейг. Персональные компьютеры в сетях TCP/IP: Руководство администратора сети. – К. : BHV, 1997. – 384с. : ил.

ДОДАТКИ

Додаток А

Текст програмних модулів системи

index.php

```
    <script>
     function add_online()
     {
     if (post.user.value=="")
     {
     alert("Невідомий користувач!");
     return false;
     }
     if (post.password.value=="")
     {
     alert("Неправильний пароль!");
     return false;
     }
     return true;
     }
    </script>
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
    <html>
    <head>
    <link type="text/css" rel="stylesheet" href="cp.css">
    <title>Satellite</title>
    <meta       http-equiv="Content-Type"       content="text/html;
charset=windows-1251">
    </head>

    <body bgcolor="#CDF1D7">
    <table  width="200"  border="0"  cellspacing="0"  cellpadding="0"
align="center" height="100%">
     <tr>
      <td>
    <table  width="200"  border="0"  cellspacing="0"  cellpadding="1"
align="center" bgcolor="#6A9ACA">
    <tr><td>
    <table  width="200"  border="0"  cellspacing="0"  cellpadding="5"
align="center" bgcolor="#D6ECF6">
    <form      action="logon.php"      method="post"      name=post
onsubmit="return add_online();">
    <tr><td    colspan=2    bgcolor="#336699"    align=center><font
color=#ffffff><img                          src="../images/keys.gif"
align=absmiddle> <b>Вхід</b></font></td></tr>
     <tr>
        <td width="30%" align="Left" class=logon><b>Логін:</b></td>
        <td  width="70%"><input  name="user"  type="text"  id="user"
size="20" maxlength="40" class=book-sm></td>
      </tr>
      <tr>
        <td align="left" class=logon><b>Пароль:</b></td>
```

```
        <td><input   name="password"   type="password"   size="20"
maxlength="40" class=book-sm></td>
      </tr>
      <tr>
        <td></td><td><input       type="submit"       name="Submit"
value="Увійти" class=button></td>
      </tr>
     </form>
     </table>
     </td></tr></table>
     </td></tr>
    </table>
    </body>
    </html>
```

logon.php

```
<?
session_start();
$user = $_POST['user'];
$pasw = $_POST['password'];
$auth = 0;

if(!empty($user) && !empty($pasw))
{
  include_once("./include.php");
  $dtable = "admin_user";

  $sql = "SELECT * FROM `admin_user` WHERE `user` = '$user'";
  $res = mysql_query($sql);

  $rows = mysql_num_rows($res);

  if($rows == 1) {

   $row = mysql_fetch_array($res);
   $did     = $row['id'];
   $duser   = $row['user'];
   $dpasw   = $row['password'];
   $dstatus = $row['status'];
   $dipdep  = $row['ipdep'];
   $dip     = $row['ip'];
   $dcom    = $row['comment'];
   $dmail   = $row['mail'];


   if(strtolower($dstatus) == 'yes') {
    $p = crypt($pasw, $dpasw);
    if($p == $dpasw) {
     $auth  = 1;
     $user  = $duser;
     $uid = $did;

     $sid = @session_name().'='.@session_id();
     $_SESSION["uid"] = $uid;
     $_SESSION["user"] = $user;
     $_SESSION["auth"] = $auth;

     echo    "<META    HTTP-EQUIV=\"Refresh\"       CONTENT=\"0;
URL=frame.php?$sid\">";

    }
    else {
    echo "Password Error";
    echo    "<META    HTTP-EQUIV=\"Refresh\"       CONTENT=\"0;
URL=index.php\">";
    }
```

```
        }
        else {
         //echo "Account $duser is not active";
         echo    "<META    HTTP-EQUIV=\"Refresh\"        CONTENT=\"0;
URL=index.php\">";
        }

       }
       else {
        //echo "User unknown";
        echo    "<META    HTTP-EQUIV=\"Refresh\"        CONTENT=\"0;
URL=index.php\">";
       }
    }  else  {  echo  "<META  HTTP-EQUIV=\"Refresh\"    CONTENT=\"0;
URL=index.php\">"; session_destroy();}
    ?>
```

logout.php

```
<?
  session_start();
  session_destroy();
  echo    "<META    HTTP-EQUIV=\"Refresh\"         CONTENT=\"0;
URL=index.php\">";
?>
```

Post.php

```
<?
  session_start();
  session_destroy();
  echo    "<META    HTTP-EQUIV=\"Refresh\"         CONTENT=\"0;
URL=index.php\">";
?>
```

Set.php

```
<?php
$type = $_REQUEST['type'];
$id   = $_REQUEST['id'];

include_once("./include.php");
if($type=='main'){
 $res  =  mysql_query("SELECT  `main_id`  FROM  `main`  WHERE
`default` LIKE '1';") or die(mysql_error());
 $cur = mysql_fetch_row($res);
 $cur = $cur[0];

 if($cur <> $id) {
  mysql_query("UPDATE    `main`    SET    `default`='0'    WHERE
`main_id`='$cur';") or die(mysql_error());
  mysql_query("UPDATE    `main`    SET    `default`='1'    WHERE
`main_id`='$id';") or die(mysql_error());
 }
 else {
  mysql_query("UPDATE    `main`    SET    `default`='1'    WHERE
`main_id`='$id';") or die(mysql_error());
 }
 header("Location: data.php?m=main");
}
if($type=='about'){
 $res  =  mysql_query("SELECT  `id`  FROM  `about_rozdils`  WHERE
`default` LIKE '1';") or die(mysql_error());
 $cur = mysql_fetch_row($res);
 $cur = $cur[0];

 if($cur <> $id) {
  mysql_query("UPDATE  `about_rozdils`  SET  `default`='0'  WHERE
`id`='$cur';") or die(mysql_error());
  mysql_query("UPDATE  `about_rozdils`  SET  `default`='1'  WHERE
`id`='$id';") or die(mysql_error());
 }
```

```
     else {
      mysql_query("UPDATE `about_rozdils` SET `default`='1' WHERE
`id`='$id';") or die(mysql_error());
     }
          header("Location: data.php?m=about");
    }

    ?>
```

Include.php

```
    <?php
    $type = $_REQUEST['type'];
    $id   = $_REQUEST['id'];
    include_once("./include.php");
    if($type=='main'){
     $res  =  mysql_query("SELECT  `main_id`  FROM  `main`  WHERE
`default` LIKE '1';") or die(mysql_error());
     $cur = mysql_fetch_row($res);
     $cur = $cur[0];

     if($cur <> $id) {
      mysql_query("UPDATE   `main`   SET   `default`='0'   WHERE
`main_id`='$cur';") or die(mysql_error());
      mysql_query("UPDATE   `main`   SET   `default`='1'   WHERE
`main_id`='$id';") or die(mysql_error());
     }
     else {
      mysql_query("UPDATE   `main`   SET   `default`='1'   WHERE
`main_id`='$id';") or die(mysql_error());
     }
     header("Location: data.php?m=main");
    }
    if($type=='about'){
     $res  =  mysql_query("SELECT  `id`  FROM  `about_rozdils`  WHERE
`default` LIKE '1';") or die(mysql_error());
     $cur = mysql_fetch_row($res);
     $cur = $cur[0];
    if($cur <> $id) {
      mysql_query("UPDATE `about_rozdils` SET `default`='0' WHERE
`id`='$cur';") or die(mysql_error());
      mysql_query("UPDATE `about_rozdils` SET `default`='1' WHERE
`id`='$id';") or die(mysql_error());
     }
     else {
      mysql_query("UPDATE `about_rozdils` SET `default`='1' WHERE
`id`='$id';") or die(mysql_error());
     }
          header("Location: data.php?m=about");
    }

    ?>
```

frame.php

```
<?
session_start();
$auth = $_SESSION["auth"];
$user = $_SESSION["user"];
if($auth == 1){
$sid = @session_name().'='.@session_id();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<? echo "<title>Адмінчастина сайту Satellite, корстувач:
$user</title>"; ?>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
<frameset rows="*" cols="130,*" framespacing="0"
frameborder="NO" border="0">
  <frame src="menu.php?<?=$sid?>" name="leftFrame"
scrolling="NO" noresize>
  <frame src="data.php?<?=$sid?>" name="mainFrame">
</frameset>
<noframes><body>
</body></noframes>
</html>

<?
 }
 else echo "Not authorised";
?>
```

data.php

```
<?
session_start();
$id   = $_SESSION['uid'];
$user = $_SESSION["user"];
$auth = $_SESSION["auth"];
include_once("./include.php");
$m = $_REQUEST['m'];

$tmp = mysql_fetch_array(mysql_query("SELECT * FROM `admin_user`
WHERE `user` = '$user' AND `id`='$id';")) or die(mysql_error());
$duser   = $tmp['user'];
$dstatus = $tmp['status'];

if($dstatus!='yes') $error = 1;
if(!$error) {
if(!$m) $m = 'about';

if(file_exists('modules/'.$m.".php")) {
        require("modules/".$m.".php");
}
else {
        require("modules/404.php");
}
}
else {
        session_destroy();
        header("Location: index.php");;
}
?>
```

cp.css

```
body {
        font-family: verdana;
        font-size : 12px;
        color : #000000;
        margin-left: 5px;
        margin-top: 5px;
        margin-right: 5px;
        margin-bottom: 5px;
}
a{
        text-decoration : none;
        color : #003399;
}
a.no{
        text-decoration : none;
        color : #003399;
}

a:hover{
        text-decoration : none;
        color:#6699CC;
}

table{
        font-family:Verdana;
        font-size : 11px;
        color : Black;
}
img.n{
        border: 1px solid #FFCCCC;
     }

input   {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:495966;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        }
input.admin {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:495966;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width: 495px;
}
textarea.admin{
        font-family: Tahoma;
```

```
        font-size:11px;
        font-style:normal;
        color:495966;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width: 495px;
        height:100px;
        }
textarea.admin2{
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:495966;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width: 495px;
        height:250px;
        }

input.submit
        {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:black;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        }

textarea.book
        {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:black;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width:250px;
        height:150px;
        }
textarea.book2
        {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:black;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width:340px;
        height:150px;
        }
```

```
input.book2
        {
        font-family: Tahoma;
        font-size:11px;
        font-style:normal;
        color:black;
        background: #F7F6F6;
        border: 1px solid #C5CDCF;
        width:200px
                }

input.def
        {
        font-family: Tahoma;
        font-size:10px;
        font-style:normal;
        color:#495966;
        border: 1px solid #C5CDCF;
        width: 200px;
        }
hr {    color: #F0F0F0;

}
```

modules/404.php

```
<center>
<h2>404</h2>
Вказаний файл не знайдено</center>
```

modules/about.php

```
<?PHP
$id  = $_REQUEST['id'];
$cid = $_REQUEST['cid'];
$a   = $_REQUEST['a'];
$isc = $_GET['is_confirmed'];

switch ($a) {
 default:
 case 'list':
      $count = $about_rozdils->getCount();
      $rozdils    =    $about_rozdils->GetList(null,null,null,
null,null);
      $smarty->assign("count", $count);
      $smarty->assign("rozdils", $rozdils);
      $page = "cp/rozdils.html";
      break;
 case 'del':
      $about_rozdils->delete($id);
      $about_info->query("DELETE   FROM   `about_info`   WHERE
`cid`=".$id);
      header("Location: data.php?m=about");
      break;
```

```
     case 'add_rozdil':
          $about_rozdils->insert($_POST);
          header("Location: data.php?m=about");
          break;
     case 'edit':
          $rozdil = $about_rozdils->Get($id,null);
          $info   = $about_info->Get(null, "`cid`='".$id."'");
          $rozdil['name'] = htmlspecialchars($rozdil['name']);
          $smarty->assign("infocount",                $about_info-
>GetListCount("`cid`='".$id."'",null,null,null,null));
          $smarty->assign("rozdil", $rozdil);
          $smarty->assign("info", $info);
          $page = "cp/rozdils_edit.html";
          break;
     case 'add_article':
          $page = "cp/rozdil_article_add.html";
          break;
     case "change_name":
          $about_rozdils->update($_POST,$_POST['id']);
          header("Location: data.php?m=about");
          break;
     case "add":
          $about_info->insert($_POST);
          header("Location: data.php?m=about");
          break;
     case "edit_article":
          $smarty->assign("info", $about_info->Get($id));
          $page = "cp/rozdil_article_edit.html";
          break;
     case "article_edit":
          $about_info->update($_POST,$_POST['id']);
          header("Location: data.php?m=about");
          break;
          case "del_article":
          $about_info->delete($_GET['id']);
          header("Location: ".$_SERVER['HTTP_REFERER']);
          break;
    }
    if (isset($page)) {
     $smarty->assign("include_tpl", $page);
     $smarty->display("cp/index.html");
    }
    ?>
```

modules/logview.php

```
    <?php
    class log extends db {
            var $table = 'log';
            var $idField = 'log_id';
    }

    $log = new log();
```

```
    $MaxOnPage = 40;
    switch ($_REQUEST["a"]) {
        default:
        case 'list':
            if ($_REQUEST["order"])
                $order           =           $_REQUEST["order"].'
'.$_REQUEST['direction'];
            else {
               $_REQUEST["order"] = "log_id";
                $_REQUEST['direction'] = "down";
            }

            $from = $_REQUEST['dc1'];
            $to   = $_REQUEST['dc2'];
            if(!empty($from) && !empty($to)) {
                $smarty->assign("dc1", $_REQUEST['dc1']);
                $smarty->assign("dc2", $_REQUEST['dc2']);
                $from = reformatdate($from, "00:00:00");
                $to   = reformatdate($to, "23:59:59");
                $where = " `post` BETWEEN '".$from."' AND '".$to."'
";
            }
            else $where = null;

            $Logs   =   $log->GetList($where,   $_REQUEST['order'],
$_REQUEST['direction'], $_REQUEST['start'], $MaxOnPage);
            $logListCount       =       $log->GetListCount($where,
$_REQUEST['order'], $_REQUEST['direction']);
            $smarty->assign("Logs", $Logs);
            $smarty->assign("logListCount", $logListCount);
            $smarty->assign("MaxOnPage", $MaxOnPage);
            $page = "../templates/log_list.tpl";

        break;

    }

    if (isset($page)) {
     $smarty->assign("include_tpl", $page);
     $smarty->display("cp/index.html");
    }
    ?>
```

modules/main.php

```
<?PHP
$id  = $_REQUEST['id'];
$cid = $_REQUEST['cid'];
$a   = $_REQUEST['a'];
$isc = $_GET['is_confirmed'];

switch ($a) {
        default:
        case 'list':
                $count = $main->getCount();
                $info  =  $main->GetList(null,"main_id","down",
$_REQUEST['start'], MaxOnPage);
                $smarty->assign("count", $count);
                $smarty->assign("info", $info);
                $smarty->assign("max", MaxOnPage);
                $page = "cp/main_edit.html";
                break;
        case 'del_article':
                $main->delete($id);
                header("Location: data.php?m=main");
                break;
        case 'add_article':
                $page = "cp/main_add.html";
                break;
        case "add":
                $main->insert($_POST);
                header("Location: data.php?m=main");
                break;
        case "edit_article":
                $smarty->assign("info", $main->Get($id));
                $page = "cp/main_article_edit.html";
                break;
        case "article_edit":
                $main->update($_POST, $_POST['main_id']);
                header("Location: data.php?m=main");
                break;
}
if (isset($page)) {
        $smarty->assign("include_tpl", $page);
        $smarty->display("cp/index.html");
}
?>
```

modules/news.php

```
<?PHP
if(empty($_REQUEST['a'])) {
        $_REQUEST['a'] = 'list';
}
switch ($_REQUEST['a']) {
        default:
        case 'list':
```

```
                    $smarty->assign("news",    $news->GetList(null,
$news->idField, "down", $_REQUEST['start'], MaxOnPage));
                    $smarty->assign("newscount",             $news-
>GetListCount(null,null,null,null,null));
                    $smarty->assign("MaxOnPage", MaxOnPage);
                    $smarty->assign("MAX_NEWS",
getVar("MAX_NEWS"));
                    $page = "cp/news_list.html";

            break;
            case 'add_new':
                    $smarty->assign("img_files",
scandir("../images/news/small/"));
                    $page = "cp/news_add.html";
            break;
            case 'insert_new':

if(!empty($HTTP_POST_FILES['user_file']['name'])                  &&
$HTTP_POST_FILES['user_file']['size'] > 0) {
                            $name            =            $upload-
>upload_img($HTTP_POST_FILES['user_file']['tmp_name'],
$HTTP_POST_FILES['user_file']['name'],
"../images/news/big/",$HTTP_POST_FILES["user_file"]["type"]);
                            $name                                  =
createIMG2("../images/news/big/", $name, 300, 300, 0);
                            if   (copy("../images/news/big/".$name,
"../images/news/small/".$name)) {

createIMG2("../images/news/small/", $name, 120, 120, 0);
                            }
                            $_POST['news_img'] = $name;
                    }
                    elseif (!empty($_POST['img_files'])) {
                            $_POST['news_img']                     =
$_POST['img_files'];
                    }
                    $news->insert($_POST);
                    header("Location: data.php?m=news");
            break;
            case 'edit':
                    $new = $news->Get($_GET['id']);
                    $new['news_title']                             =
htmlspecialchars($new['news_title']);
                    $smarty->assign("news", $new);
                    $smarty->assign("img_files",
scandir("../images/news/big/"));
                    $page = "cp/news_edit.html";
            break;
            case 'del':
                    if($_GET['is_confirmed']==1) {
                            $news->delete($_GET['id']);
                    }
```

```
                header("Location: ".$_SERVER['HTTP_REFERER']);
        break;
        case 'del_selected':
                $news->multiDel($_POST['id_box']);
                header("Location: ".$_SERVER['HTTP_REFERER']);
        break;
        case 'update_new':

if(!empty($HTTP_POST_FILES['user_file']['name'])                &&
$HTTP_POST_FILES['user_file']['size'] > 0) {
                        $name            =            $upload-
>upload_img($HTTP_POST_FILES['user_file']['tmp_name'],
$HTTP_POST_FILES['user_file']['name'],      "../images/news/big/",
$HTTP_POST_FILES["user_file"]["type"]);
                        $name                                    =
createIMG2("../images/news/big/", $name, 300, 300, 0);
                        if   (copy("../images/news/big/".$name,
"../images/news/small/".$name)) {

createIMG2("../images/news/small/", $name, 120, 120, 0);
                        }
                        $_POST['news_img'] = $name;
                }
                elseif     (!empty($_POST['img_files'])     &&
$_POST['img_files'] != "spacer.gif") {
                        $_POST['news_img']                       =
$_POST['img_files'];
                }
                if ($_POST['del_current'] == 1) {
                        $_POST['news_img'] = '';
                }
                $news->update($_POST, $_POST['id']);
                header("Location: data.php?m=news");
        break;
        case "update_max_new":
                setVar('MAX_NEWS', $_POST['MAX_NEWS']);
                header("Location: ".$_SERVER['HTTP_REFERER']);
        break;
    }
    if (isset($page)) {
            $smarty->assign("include_tpl", $page);
            $smarty->display("cp/index.html");
    }
    ?>
```

Додаток Б

Тестування АДІС “Контроль та облік зубуту”

№ тесту	Вхідні дані		Очікуваний результат	Результат програми
1	Номер договору	1000	Нормальна робота АДІС	Очікування вводу, дати договору
	Дата договору	04.04.05	Нормальна робота АДІС	Занесення запису в БД
2	Номер договору	1000	Помилка. Ключ повинен бути унікальним	Видача повідомлення про помилку
3	Номер договору	100	Нормальна робота АДІС	Очікування вводу дати договору
	Дата договору	2,2,2	Помилка. Ввід не в формі дати	Видача повідомлення про помилку
4	Сума рахунку	12349	Нормальна робота АДІС	Занесення запису в БД
5	Сума рахунку	0	Помилка. Сума повинна бути більше 0	Видача повідомлення про помилку
6	Оплачено	Так	Нормальна робота АДІС	Занесення запису в БД
7	Оплачено	Ні	Нормальна робота АДІС	Занесення запису в БД
8	Оплачено	Є	Помилка. Значення поля приймає 2 значення. так, ні	Видача повідомлення про помилку

Додаток Г

Приклади запитів до БД

```
SELECT nomer_dogovora, postav.nomer_postav,
dogovor.nomer_postav, naimen_post
FROM    postav, dogovor
WHERE postav.nomer_postav=dogovor.nomer_postav

SELECT select nomer_zajavki, zajavka.nomer_dogovora,
dogovor.nomer_dogovora,naimen_post,postav.nomer_postav,
dogovor.nomer_postav
FROM    from zajavka,dogovor,postav
WHERE (zajavka.nomer_dogovora=dogovor.nomer_dogovora)
               AND
(postav.nomer_postav=dogovor.nomer_postav)

SELECT nomer_zakaza, zakaz.nomer_dogovora,
dogovor.nomer_dogovora,
                naimen_post,postav.nomer_postav,
dogovor.nomer_postav
FROM    zakaz, dogovor, postav
WHERE (zakaz.nomer_dogovora=dogovor.nomer_dogovora)
             AND (postav.nomer_postav=dogovor.nomer_postav)
```