Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Осіпова Антона Олеговича*
(ПІБ)

академічної групи *122-18ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка інформаційної системи автоматизації обліку діяльності транспортного підприємства з перевезення вантажів*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Гуліна І.Г.*			
розділів:				
спеціальний	*доц. Гуліна І.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18ск-1* ***Осіпова Антона Олеговича***
(група) (прізвище та ініціали)

тема кваліфікаційної роботи ***Розробка інформаційної системи автоматизації обліку діяльності транспортного підприємства з перевезення вантажів***

затверджена наказом ректора НТУ «ДП» від « 07 » 06. 2021 р. № 317-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *доц. Гуліна І.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Осіпов А.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *15.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 70 с., 38 рис., 3 дод., 24 джерела.

Об'єкт розробки: інформаційна система автоматизації обліку діяльності транспортного підприємства з перевезення вантажів.

Мета кваліфікаційної роботи: підвищення ефективності роботи з обліку діяльності транспортного підприємства з перевезення вантажів за рахунок розробки та використанню відповідної інформаційної системи.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування підсистеми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає в автоматизації процесса обліку діяльності транспортного підприємства з перевезення вантажів, зборі інформації у загальну базу даних та її систематизацію.

Актуальність підсистеми маршрутизації запитів визначається великим попитом на подібні розробки.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ТРАНСПОРТНА КОМПАНІЯ, MYSQL, БАЗА ДАНИХ.

ABSTRACT

Explanatory note: 70 pp., 38 figs., 3 apps., 24 sources.

Object of development: information system of automation of the account of activity of the transport enterprise on transportation of freights.

The purpose of the qualification work: to increase the efficiency of accounting for the activities of the transport company for the transportation of goods through the development and use of an appropriate information system.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the subsystem, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and application load, describes the program.

In the economic section, the complexity of the developed information subsystem is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The practical significance lies in the automation of the process of accounting for the activities of the transport company for the transportation of goods, the collection of information into a common database and its systematization.

The relevance of the query routing subsystem is determined by the high demand for such developments.

Keywords: INFORMATION SYSTEM, TRANSPORT COMPANY, MYSQL, DATABASE.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних;

ЕОМ - електронно-обчислювальна машина;

ІС - інформаційна система;

АІС - автоматизована інформаційна система;

ПЗ - програмне забезпечення;

СУБД - система управління базами даних;

ІТ - інформаційні технології.

ВСТУП

На сьогоднішній день сучасна комп'ютерна техніка, та і не тільки комп'ютерна, бере неабияку участь у повсякденному і сучасному житті, адже важко уявити сучасних людей без підтримки електронних пристроїв у житті. Всі операції збереження, редагування, оформлення даних, побудова таблиць та діаграм, все це передбачає застосування комп'ютерів. Все своє життя людина так чи інакше пов'язує з отриманням, накопичуванням і обробкою інформації, яка поступає до людина весь час.

Тому, відразу після появи комп'ютерів увагу на них звернули бізнесмени. Як правило, в цивільному бізнесі не потрібні великі розрахунки за винятком таких галузей як, наприклад, авіа - або автомобілебудування. У більш поширених видах цивільного бізнесу (банківська справа, біржові операції, системи резервування квитків або місць в готелях) основною проблемою завжди були обсяги інформації, яку необхідно збирати, надійно зберігати та оперативно обробляти. Поява інформаційних систем, основним призначенням яких є вирішення зазначеної проблеми, стало відповіддю комп'ютерної індустрії на вимоги світу бізнесу.

Звичайно, залежно від конкретної області застосування, інформаційні системи можуть сильно відрізнятися за своїми функціями, архітектурою, реалізацією. Однак можна виділити, принаймні, дві властивості, які є загальними для всіх інформаційних систем. По-перше, будь-яка інформаційна система призначена для збору, зберігання і обробки інформації. Тому в основі будь-якої інформаційної системи лежить середа зберігання і доступу до даних. Середа повинна забезпечувати рівень надійності зберігання та ефективність доступу, які відповідають області застосування інформаційної системи .

По-друге, інформаційні системи орієнтуються на кінцевого користувача, наприклад, на будівельну кампанію. Такі користувачі можуть бути дуже далекі від світу комп'ютерів. Для них термінал, персональний комп'ютер або робоча станція являють собою всього лише знаряддя їх власної професійної діяльності.

Тому інформаційна система повинна мати простий, зручний, легкий для розуміння інтерфейс, який повинен надати кінцевому користувачу всі необхідні для його роботи функції , але в той же час не дати йому можливість виконувати будь - які зайві дії.

Автоматизована інформаційна система – це система, що реалізує інформаційну технологію у сфері управління за спільної роботи управлінського персоналу та комплексу технічних засобів. Вона призначена для автоматизованого збирання, реєстрації, збереження, пошуку, оброблення та видачі інформації за запитами користувачів.

За деякими оцінками приблизно 75% всіх програм для роботи з базами даних є програми для прийняття рішень. Мета створення сучасних програм для прийняття рішень – заміна друкованих звітів інформацією, відображеною на екрані термінала. При створенні подібних програм, призначених для роботи з певною, вже існуючою, реляційною базою даних, нема необхідності хвилюватись про розробку самої бази даних та її архітектури, забезпечення цілісності даних та проблемах колективного доступу.

Ця база даних повинна працювати під управлінням системи керування базою даних Microsoft SQL Server 2012. Вибір СКБД обумовлений тим, що даний програмний комплекс – це сучасна комерційна СКБД, що розповсюджується компанією Microsoft та підтримує мову формування запитів – Transact-SQL.

Сьогодні найбільш популярними середовищами розробки програмного забезпечення (IDE), що підтримують мову C++ є Microsoft Visual Studio та Embarcadero RAD Studio XE8 C++ Builder. Обидві ці системи дозволяють швидко створювати програмне забезпечення з підтримкою новітніх стандартів та напрямів у програмуванні (у тому числі на мові C++).

Для реалізацій поставленої задачі було обрано C++ Builder – програмний продукт від Embarcadero Technologies, інструмент швидкої розробки додатків (RAD), інтегроване середовище програмування (IDE), система, використовувана програмістами для розробки програмного забезпечення на мові програмування C++. Це середовище, на відміну від Microsoft Visual Studio, містить більшу

кількість «готових» компонентів та вбудованих класів, що, безумовно, спростить та пришвидшить процес реалізації програмного забезпечення. C + + Builder об'єднує в собі комплекс об'єктних бібліотек (STL, VCL, CLX, MFC та ін.), компілятор, відладчик, редактор коду, вбудований редактор інтерфейсу, який значно спрощує розробку програмного забезпечення і багато інших компонентів.

Призначення програми, що строюється – перетворення вихідних даних у корисну інформацію. Головне завдання розробника полягає в наданні користувачу простої та зрозумілої програмної системи, для отримання та обробки потрібних даних. Така система значно спростить роботу персоналу, зменшить кількість похибок, що створить позитивний економічний ефект.

У рамках даної кваліфікаційної роботи були поставлені такі завдання:

а) Розглянути теоретичну основу розробки бази даних;

б) Спроектувати базу даних з реляційною структурою;

в) Створити ненадлишкову базу даних з контролем цілісності даних;

г) Розробити автоматизовану програмну систему для роботи з цією базою даних.

Таким чином, в сучасних умовах, коли з’явилися швидкодіючі комп’ютери та високий розвиток програмного забезпечення в проектуванні баз даних, стає доцільним переведення інформаційної бази підприємства в програмну сферу.

Завдання даної кваліфікаційної роботи та об'єкт розробки безпосередньо пов'язані з спеціальністю 122 «Комп’ютерні науки» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Предметною галуззю бази даних, що буде розроблятися, є транспортне підприємство, що спеціалізується на перевезенні вантажу.

Для перевезень торговельних вантажів на короткі відстані найчастіше використовується вантажний автомобільний транспорт, який забезпечує перевезення товарів від станцій залізниць, зі складів виробничих і сільськогосподарських підприємств на склади оптових підприємств і з цих складів в об'єкти роздрібної та дрібно-роздрібної торговельної мережі, для переміщення товарів між магазинами і для виїзної торгівлі. Часто для таких перевезень використовується транспорт спеціалізованих автотранспортних підприємств, які належать до категорії транспорту загального користування. Ці підприємства виконують централізовані перевезення вантажів, забезпечують транспортно-експедиційне обслуговування вантажовідправників і вантажоодержувачів.

Для перевезень торговельних вантажів підприємствами транспорту загального користування необхідно додержувати заведеного порядку їх планування та організації, який базується на укладанні між перевізниками та замовниками договорів про перевезення вантажів автомобільним транспортом.

Основними документами, які оформляються на перевезення вантажів автотранспортом, є товарно-транспортні накладні та подорожні листи вантажного автомобіля. Надаючи послуги перевезення, підприємство-перевізник отримує винагороду за перевезення вантажу – це дохід від надання послуг перевезення.

База даних буде виконувати функції автоматизації безпосередньо основного процесу перевезення, ведення та формування документообігу компанії. Постійно оновлюються і вводяться нові дані в списки робітників,

транспортних засобів, клієнтів та договорів. Ведеться облік товарно-транспортних відомостей: дата укладення договору, товар, кількість товару, код робітника, код клієнта, код транспортного засобу, маршрут перевезення.

1.2. Призначення розробки та галузь застосування

Призначенням розробки та галуззю застосування системи, що розробляється, є автоматизація обліку діяльності транспортного підприємства з перевезення вантажів.

Основними документами, які оформляються на перевезення вантажів автотранспортом, є товарно-транспортні накладні та подорожні листи вантажного автомобіля. Надаючи послуги перевезення, підприємство-перевізник отримує винагороду за перевезення вантажу – це дохід від надання послуг перевезення.

База даних виконує функції автоматизації безпосередньо основного процесу перевезення, ведення та формування документообігу компанії. Постійно оновлюються і вводяться нові дані в списки робітників, транспортних засобів, клієнтів та договорів. Ведеться облік товарно-транспортних відомостей: дата укладення договору, товар, кількість товару, код робітника, код клієнта, код транспортного засобу, маршрут перевезення.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма спеціальності 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;

- завдання на кваліфікаційну роботу на тему: «Розробка інформаційної системи автоматизації обліку діяльності транспортного підприємства з перевезення вантажів».

1.4. Постановка завдання

Необхідно провести аналіз предметної галузі та постановку задачі по розробці програмного рішення для обробки даних будівельної кампанії.

Перш за все на етапі загальносистемного проєктування необхідно визначити концепцію майбутньої системи, яка визначатиме подальший хід проєктування вимог до функціональних характеристик.

Проектування розпочинається з визначення цільової аудиторії, на яку має бути розрахований програмний засіб, що розробляється. Враховуючи наявні технічні можливості, та тенденції IT-ринку, слід звернути увагу на цільову аудиторію підприємств, що займається перевезеннями та поставкою продукції.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмні вимоги - властивості, які повинні бути належним чином представлені для вирішення конкретних практичних завдань. Процес розробки вимог до програмного забезпечення призначений для визначення функцій, які виконуватиме програмне забезпечення, що розробляється, а також обмежень, які накладаються на функціональні можливості і розробку бази даних.

Програмні вимоги поділяються на функціональні та не функціональні вимоги:

- Функціональні вимоги – це вимоги до програмного забезпечення, які визначають, що система повинна робити та описують внутрішню роботу

системи, її поведінку: калькулювання даних, маніпулювання даними, обробка даних та інші специфічні функції, які має виконувати система;

- Не функціональні вимоги – це вимоги до програмного забезпечення, які визначають якою система повинна бути та задають критерії для оцінки якості роботи програмного забезпечення;

Мета розробки даної програмної інформаційної системи – створення бази даних «Автоматизація обліку діяльності транспортного підприємства з перевезення вантажу». Програмна система повинна мати зручний функціонал для управління, корегування та виведення результатів. Візуальне оформлення повинно бути інтуїтивно зрозумілим. Програмне забезпечення бази даних «Автоматизація обліку діяльності транспортного підприємства з перевезення вантажу» передбачає реалізацію програмної системи з виконанням заданих умов та функціональних можливостей програмного додатку.

Функціональні вимоги:

- Можливість формування списку робітників;
- Можливість формування списку автомобільних транспортів;
- Можливість формування списку клієнтів;
- Можливість формування списку заключення договорів;
- Можливість формування списку товарно-транспортної відомості.

Не функціональні вимоги:

Вимоги до програмного забезпечення системи:

Наявність програмного середовища Microsoft SQL Server Management Studio. Програмний додаток платформи повинен працювати на базі 32- і 64-разрядних операційних систем Windows.

Вимоги до технічного забезпечення системи:

В комплекс технічних засобів повинні входити такі елементи:

- автоматизовані робочі станції;
- джерело безперебійного живлення;
- Мишка, клавіатура; монітор для відображення та перегляду інформації.

Користувачами бази даних є адміністратор та будь-який користувач.

Адміністратор має можливість:

1. Формування списків клієнтів, робітників, транспорту, договорів, товарно-транспортних відомостей.
2. Редагування інформації у базу даних.
3. Заповнення таблиць.
4. Переглядати інформацію у таблицях.

Користувач, в свою чергу, має дозвіл лише на:

1. перегляд даних товарно-транспортних відомостей.
2. перегляд бази клієнтів.
3. перегляд автопарку.
4. перегляд вільних водіїв.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– семантичний та синтаксичний контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

Як складовий елемент системи, повязаної із торгівельною діяльністю, підсистема маршрутизації запитів повинна мати наступні характеристики:

– час відновлення після збою - 5 хв;

– час відновлення після відмови одного з елементів підсистеми не повинно перевищувати половини операційного банківського дня;

– вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;

– забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови підсистеми.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми необхідно, щоб обчислювальна машина, на якій буде функціонувати веб-орієнтована підсистема, відповідала наступним вимогам:

– процесор класу Intel Xeon з тактовою частотою не менш 2.4 ГГц;
– не менше 2 GB оперативної пам'яті;
– рідкокристалічний монітор з діагоналлю не менше 17 ";
– 20 Гб вільного місця на жорсткому диску;
– доступ до мережі Internet;
– клавіатура;
– маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати система, відповідало вимозі: робота під управлінням операційної системи сімейства Windows (7, 8, 10). Система є невимогливою до складу програмних засобів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Мета розробки даної інформаційної системи – створення бази даних «Автоматизація обліку діяльності транспортного підприємства з перевезення вантажу». Система має зручний функціонал для управління, корегування та виведення результатів. Візуальне оформлення інтуїтивно зрозумілє. Програмне забезпечення бази даних «Автоматизація обліку діяльності транспортного підприємства з перевезення вантажу» передбачає реалізацію програмної системи з виконанням заданих умов та функціональних можливостей програмного додатку:

- можливість формування списку робітників;
- можливість формування списку автомобільних транспортів;
- можливість формування списку клієнтів;
- можливість формування списку заключення договорів;
- можливість формування списку товарно-транспортної відомості.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної гаузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці системи специфічні математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Зберігання даних в інформаційних системах можна організувати наступним чином:

• набір файлів на диску;

• реляційна база даних;

• нереляційних база даних (ієрархічні СУБД і ін.).

Найбільш універсальним, масштабується і простим для підтримки є використання в системі, що розробляється реляційної СУБД.

Зараз існує багато технологій розробки програмного забезпечення. Основними є такі технології:

- структурне програмування ;
- програмування “знизу вгору”;
- програмування “згори вниз”;
- модульне програмування;
- об’єктно-орієнтоване програмування.

При виконані постановки задачі використовуються наступні технології програмування:

- структурне програмування;
- модульне програмування;
- об’єктно-орієнтоване програмування;
- візуальне програмування.

Універсальна мова моделювання (Unified Modelling Language або UML) - це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему. У Umbrello UML Modeller реалізовано підтримку таких типів діаграм:

Діаграми випадків використання описують взаємозв’язки і залежності між групою випадків використання і акторами, що беруть участь у процесі.

Важливо зауважити, що діаграми випадків використання не призначено для показу компонування, вони не можуть описати внутрішню структуру системи. Діаграми випадків використання призначено для полегшення обміну інформацією між майбутніми користувачами системи і замовником, вони особливо корисні для визначення переліку можливостей, які повинна мати система. За діаграмами випадків використання можна сказати, що система має робити, але не те, як вона досягає потрібних результатів, для останнього ці діаграми просто не придатні.

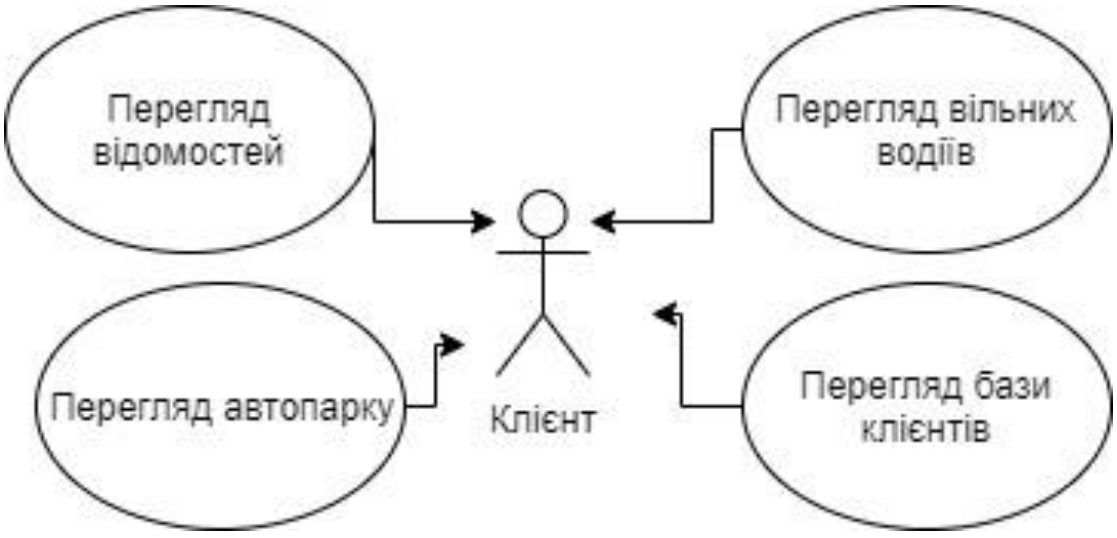


Рис. 2.1. Діаграма варіантів використання для користувача

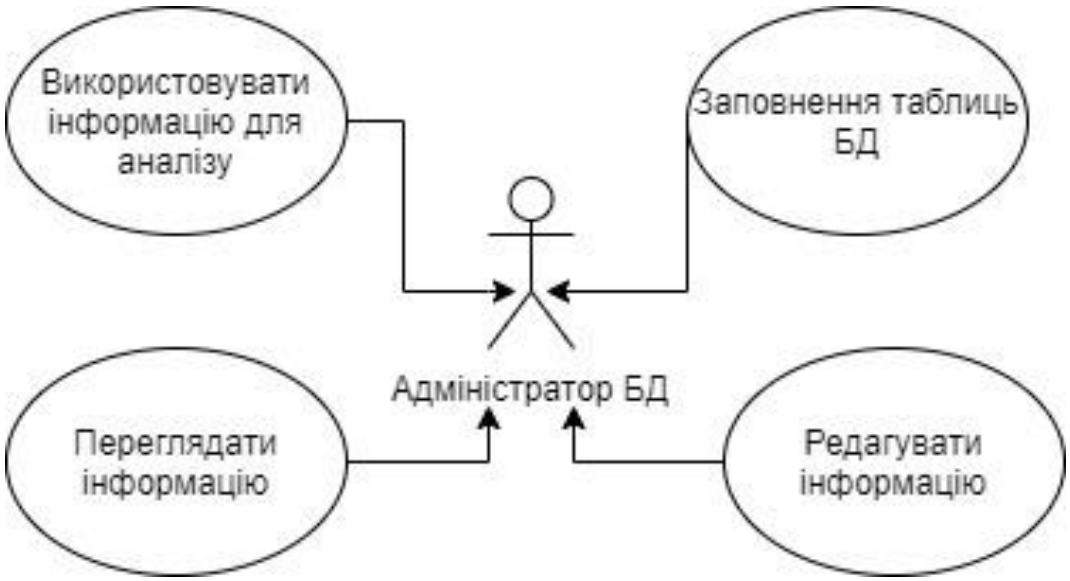


Рис. 2.2. Діаграма варіантів використання для адміністратора

На діаграмах станів зображають різні стани об’єкта під час його існування і стимули, які призводять до переходу об’єкта з одного стану у інший.

На діаграмах стану об’єкти розглядаються як машини станів або скінченні автомати, які можуть перебувати у одному зі станів скінченного набору станів, і які можуть змінювати цей стан через вплив одного зі стимулів зі скінченного набору стимулів. Наприклад, об’єкт типу Сервер мережі може перебувати у одному з таких станів протягом існування:

- Готовність
- Очікування
- Робота
- Зупинка

а подіями, які можуть спричинити зміну стану об’єкта можуть бути

- Створення об’єкта
- Об’єкт отримує повідомлення «очікувати»
- Клієнт надсилає запит на з’єднання мережею
- Клієнт перериває запит
- Запит виконано і перервано
- Об’єкт отримує повідомлення «зупинка»

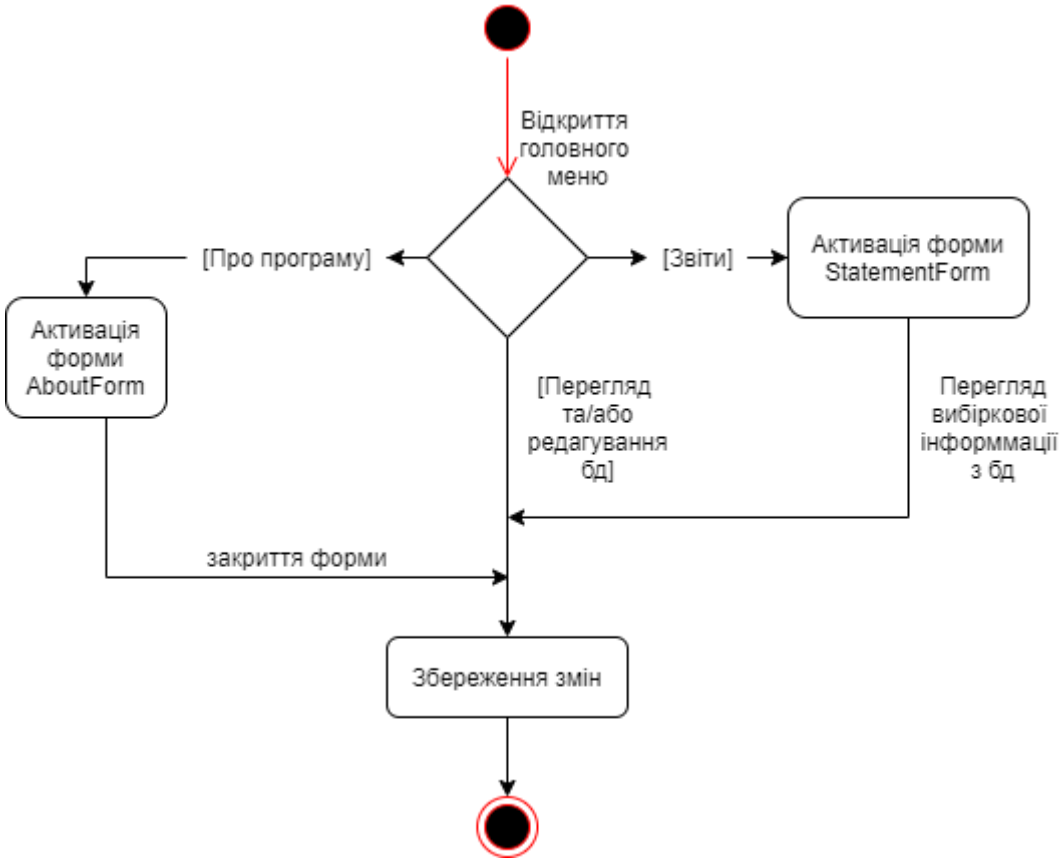


Рис. 2.3. Приведено діаграму станів

Для реалізації системи обрано архітектуру "клієнт-сервер". На сервері зберігається база даних і серверна частина програмного забезпечення системи, до якої входять збережені процедури, індекси, тригери. Клієнт містить клієнтську частину програмного забезпечення, а саме реалізацію інтерфейсу користувача. На рис. 2.6. приведена діаграма розгортання системи

Рис. 2.4. Діаграма розгортання системи

Фізична модель бази даних - це опис конкретної реалізації бази даних, що розміщується в зовнішній пам'яті. Фізична модель описує базові відносини,

визначає організацію файлів і склад індексів, застосовуваних для забезпечення ефективного доступу до даних, а також регламентує всі відповідні обмеження цілісності та заходи захисту. Якщо логічна модель бази даних являла собою інформаційну структуру проектованої системи в залежності від моделі даних, але незалежно від виду програмного забезпечення, за допомогою якого система буде реалізована, то фізична модель даних залежить від технічних і програмних засобів, які будуть використовуватись. Для реалізації бази даних була використана реляційна СКБД SQL Server, були створені необхідні індекси та виконано проектування базових відношень згідно мови DDL обраної СКБД з врахуванням обмежень цілісності.

Для створення бази даних створено наступний сценарій:

а) Створення бази даних “Транспортне підприємство ”;

```
create database Транспортна_компанія
```

б) Використання бази даних “Транспортне підприємство”;

```
use Транспортна_компанія
```

в) Створення таблиці “ Клієнт”;

```
create table Клієнт
(
ClientID int PRIMARY KEY not Null,
Назва nvarchar (40) not Null,
Тип nvarchar (20) not Null
)
Go
```

г) Створення таблиці “Транспорт”;

```
create table Транспорт
(
AutoID int PRIMARY KEY not Null,
Категорія nvarchar (4) not Null,
Швидкість int not Null,
[Грузопід’ємність] int not Null
```

)
Go

д) Створення таблиці “Робітник ”;

```
create table Робітник
(
WorcerID int PRIMARY KEY not Null,
ПІБ nvarchar (60) not Null,
Вік int not Null,
Посада nvarchar (30) not Null,
Статус nvarchar (1) not Null
)
Go
```

е) Створення таблиці “Договір ”;

```
create table Договір
(
ContractID int PRIMARY KEY not Null,
ClientID int not Null,
ДатаП date not Null,
ДатаЗ date not Null,
Вартість int not Null
)
Go
```

ж) Створення таблиці “Відомість ”;

```
create table Відомість
(
NacladID int PRIMARY KEY not Null,
ClientID int not Null,
AutoID int not Null,
```

```
WorcerID int not Null,
Маршрут nvarchar(100) not Null,
Товар nvarchar(60) not Null,
Кількість int not Null,
[Дата укладення] date not Null
)
Go
```

Для предметної галузі транспортне підприємство були сформовані запити на вибірку з бази даних на виконання відповідних функції.

а) Переглядати відомості про договори, укладені із вказаним клієнтом:

```
select
    Договір.ДатаП, Договір.ДатаЗ, Договір.Вартість, Клієнт.Назва
from
    Договір, Клієнт
where
    Клієнт.ClientID = Договір.ClientID and
    Договір.ClientID = 1
Go
```

б) Переглядати відомості про незадіяних на даний момент співробітників:

```
select
    Робітник.ПІБ, Робітник.Посада, Робітник.Статус
from
    Робітник
where
    Робітник.Статус = 'в'
go
```

в) Переглядати відомості на попит на різні типи автотранспортних засобів:

```
select
Транспорт.AutoID as ID_транспорта,
```

```
Транспорт.[Грузопід’ємність], Транспорт.Категорія,
Транспорт.Швидкість,
 COUNT (Відомість.AutoID) as Кількість_виїздів
from
    Відомість, Транспорт
where
     Транспорт.Категорія = 'C' and
    Відомість.AutoID = Транспорт.AutoID
group by
Транспорт.[Грузопід’ємність], Транспорт.Категорія,
Транспорт.Швидкість, Транспорт.AutoID
Go
```

г) Переглядати вказану товарно-транспортну накладну:

```
Select
Відомість.Маршрут, Відомість.Товар, Відомість.Кількість,
Відомість.[Дата укладення], Клієнт.Назва as Клієнт,
    Транспорт.Категорія as Категорія_транспорту, Робітник.ПІБ
from
    Робітник, Клієнт, Транспорт, Відомість
where
    Клієнт.ClientID = Відомість.ClientID and
    Транспорт.AutoID = Відомість.AutoID and
    Робітник.WorcerID = Відомість.WorcerID and
    Відомість.NacladID = 4
Go
```

д) Переглядати зведену відомість про виручку за вказаний період:

```
select
    SUM(Договір.Вартість) as Прибуток
From Договір
Where Договір.ДатаП > '2021.03.03' and
```

Договір.ДатаЗ <= '2021.03.06'

Go

Для розробки програмного додатку «Транспортна компанія» було обрано середу розробки «Embarcadero RAD Studio C++ Builder XE8».

Клієнтська частина являє собою інтерфейс для користувача. Інтерфейс складається з декількох форм.

Доступ до бази даних був організований за допомогою бібліотеки ADO.

ADO (від англ. ActiveX Data Objects — «об'єкти даних ActiveX») — прикладний програмний інтерфейс для доступу до даних, розроблений компанією Microsoft (MS Access, MS SQL Server) і заснований на технології компонентів ActiveX. ADO дозволяє представляти дані з різноманітних джерел: (реляційної СУБД, текстових файлів тощо) в об'єктно-орієнтованому програмуванні виді.

Далі подана коротка характеристика основних компонентів ADO у таблиці 2.1.

Таблиця 2.1.

Характеристика основних компонентів ADO

Компонент	Характеристика
TADOConnection	Використовується для зв’язку з набором даних ADO. Може працювати з декількома компонентами наборів даних як диспетчер виконання їх команд.
TADOTable	Використовується для роботи з однією таблицею. Може з’єднуватися безпосередньо, або через компонент TADOConnection.
TADOQuery	Використовується для роботи з набором даних за допомогою запитів SQL. Може з’єднуватися безпосередньо, або через компонент TADOConnection.

Також використовувалися наступні компоненти зв’язку та відображення даних (таб.2.1):

Таблиця 2.2.

Використані компоненти зв’язку та відображення даних

Компонент	**Характеристика**
DataSource	Забезпечує механізм для зв'язку компонентів доступу до даних з візуальними компонентами, які відображають дані.
DBGrid	Компонент DBGrid відображає набір даних у форматі електронної таблиці.
DBNavigator	Надає користувачеві можливість переглядати набір даних. Навігатор містить кнопки для переходу на першу, останню, попередню і наступну записи, вставки, видалення і редагування запису, скасування та збереження змін, а також поновлення даних.

Далі, на рис. 2.5. приведено схему форми компонентів для роботи з даними ConnectForm.

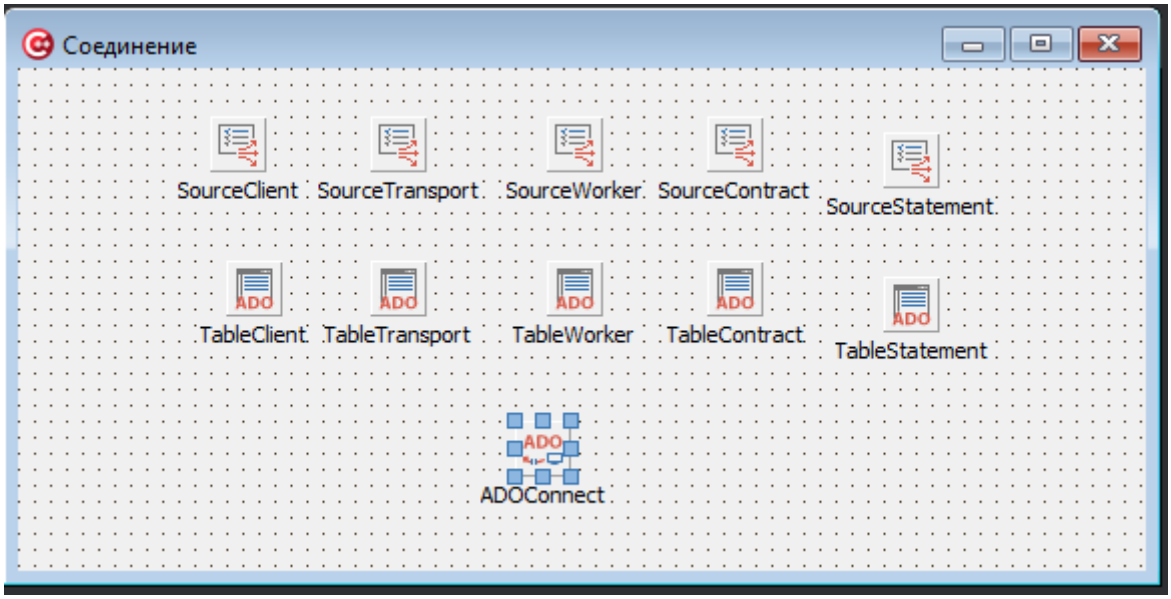


Рис. 2.5. Схема форми ConnectForm

Додаток розроблений на мові С++.

Мова С створена Д.Рітчі на початку 1970-х років для розробки операційної системи UNIX. Має засоби для прямої роботи з пам'яттю. Була задумана як мова системного програмування для заміни асемблера, щоб мати можливість створювати такі ж ефективні і короткі програми, але не залежати від конкретного процесора. Вона є найпопулярнішою мовою для створення системного програмного забезпечення.Однак великий набір операцій і типів даних, сучасне оформлення і висока ступінь машинонезалежності зробили її привабливою мовою програмування загального призначення. Незважаючи на те, що С не розроблялася для новачків, вона активно використовується для навчання програмуванню. Надалі синтаксис мови С став основою для багатьох інших мов. На мові С написана безліч прикладних і системних програм і ряд відомих ОС (зокрема, UNIX).

Подальшим розвитком ідеї алгоритмічної мови стали мови програмування більш загального, не обов'язково алгоритмічного характеру. Як і алгоритмічні мови, такі мови, зрештою, теж націлені на отримання машинних програм, але в багатьох випадках їх тексти допускають певну свободу у виконанні і, як правило, дають лише матеріал для синтезу шуканих алгоритмів, а не самі ці алгоритми. Такими мовами , наприклад, є С++, об'єктно-орієнтоване розширення мови С, створене Б. Страуструпом в 1980 р.Поєднує властивості як високорівневих, так і низькорівневих мов. Заснованана використанні класів і об'єктів.

При створенні мови С++ прагнули зберегти її синтаксис, сумісний з мовою С. Більшість програм, написаних мовою С, справно працюють і з компілятором мови С++. Нововведеннями мови С++ порівняно з мовою С підтримка об'єктно-орієнтованого програмування через класи і об'єкти, підтримка узагальненого програмування через шаблони, доповнення до стандартної бібліотеки, додаткові типи даних, обробка виключень (виключних ситуацій), простори імен, вбудовані функції, перевизначення операторів та імен функцій, посилання та оператори управління вільно розподіленою пам'яттю.

C++ (Сі-плюс-плюс) — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом в AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) 1979 року та початково отримала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові C. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014.

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java.

При створенні C++ прагнули зберегти сумісність з мовою C. Більшість програм на C справно працюватимуть і з компілятором C++. C++ має синтаксис, заснований на синтаксисі C .

Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

В 1998 році мова Сі++ була стандартизована Міжнародною організацією стандартизації під номером 14882:1998 — Мова Програмування Сі++. Поточний

стандарт — C++11, він був прийнятий у 2011 році робочою групою МОС після десятирічної підготовки.

Стандарт Сі++ на 1998 рік складається з двох основних частин: ядра мови і стандартної бібліотеки. Стандартна бібліотека C++ увібрала в себе бібліотеку шаблонів STL, що розроблялася одночасно із стандартом. Зараз назва STL офіційно не вживається, проте в колах програмістів на Сі++ ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандарт Сі++ містить нормативне посилання на стандарт Сі від 1990 року і не визначає самостійно ті функції стандартної бібліотеки, які запозичуються із стандартної бібліотеки Сі.

Поза тим, існує величезна кількість бібліотек Сі++, котрі не входять в стандарт. У програмах на Сі++ можна використовувати багато бібліотек Сі.

2.4. Опис структури системи та алгоритмів її функціонування

Проектування бази даних це процес створення проекту БД, призначеної для підтримки функціонування підприємства і підтримки досягнення його мети.

Проектування баз даних - це ітераційний, багатоетапний процес прийняття обґрунтованих рішень в процесі аналізу інформаційної моделі предметної області, вимог до даних з боку прикладних програмістів і користувачів, синтезу логічних і фізичних структур даних, аналізу та обґрунтування вибору програмних і апаратних засобів. Етапи проектування баз даних пов'язані з багаторівневою організацією даних.

Етапи проектування баз даних:

- Концептуальне проектування.
- Логічне проектування.
- Нормалізація даних.
- Фізичне проектування баз даних.

Концептуальна модель – абстрактна модель, що визначає структуру модельованої системи, властивості її елементів та причинно-наслідкові зв’язки, властиві системі і суттєві для досягнення мети моделювання.

Сутності в базі даних – це будь-які об’єкти в базі даних, які можна виділити виходячи з суті предметної області для якої розробляється база даних.

Основними сутностями розробленої бази даних є:

- Клієнт;
- Робітник;
- Транспорт;
- Накладна;
- Договір;

Опис типів сутностей, їх атрибути та зв’язки приведені у таблицях 2.3-2.5.

Таблиця 2.3.

Опис типів сутностей

Ім’я сутності	**Опис**
Клієнт	Сутність, в якій містяться дані про клієнта
Робітник	Сутність, в якій містяться дані про робітника
Транспорт	Сутність, в якій містяться дані про транспорт
Накладна	Сутність, в якій містяться дані про накладні
Договір	Сутність, в якій містяться дані про договори

Атрибути в базі даних - це іменовані характеристики, що є деякими властивостями сутності.

Таблиця 2.4.

Опис атрибутів сутностей

Ім'я сутності	**Атрибути**	**Опис атрибута**	**Тип**	**Обмеження**
Клієнт	ClientID	Унікальний ідентифікатор сутності (первинний ключ)	Int	(1,1)
	Назва	Назва фірми	nvarchar	40
	Тип	Визначення товаровідправника та товароприймача	nvarchar	20
Робітник	WorcerID	Унікальний ідентифікатор сутності (первинний ключ)	Int	(1,1)
	ПІБ	ПІБ робітника	nvarchar	60
	Вік	Вік робітника	Int	-
	Посада	Посада, яку займає робітник	nvarchar	30
	Статус	Визначення зайнятості робітника	nvarchar	1
Транспорт	AutoID	Унікальний ідентифікатор сутності (первинний ключ)	Int	(1,1)

Ім’я сутності	**Атрибути**	**Опис атрибута**	**Тип**	**Обмеження**
	Категорія	Категорія транспорту за конфігурацією	nvarchar	4
	Швидкість	Швидкість транспорту	Int	-
	Грузопід’ємність	Вага, яку може перевозити транспорт	Int	-
Договір	ContractID	Унікальний ідентифікатор сутності (первинний ключ)	Int	(1,1)
	ClientID	ID клієнта	Int	-
	ДатаП	Дата укладення договору	date	-
	ДатаЗ	Дата завершення договору	date	-
	Вартість	Ціна послуг компанії	Int	-
Відомість	NacladID	Унікальний ідентифікатор сутності (первинний ключ)	Int	(1,1)
	ClientID	ID клієнта	Int	-
	AutoID	ID авто	Int	-
	WorcerID	ID роітника	Int	-

Ім’я сутності	Атрибути	Опис атрибута	Тип	Обмеження
	Маршрут	Маршрут перевозок	nvarchar	100
	Товар	Товар, що перевозиться	nvarchar	60
	Кількість	Кількість замовленого товару	Int	-
	Дата укладення	Дата укладення договору	date	-

Зв’язки між сутностями – це деякі відношення між двома типами сутностей.

База даних, що розробляється, має такі відносини між сутностями:

1. Клієнт може укласти багато договорів, тому зв’язок має тим один до багатьох.
2. Робітник може фігурувати у багатьох відомостях, тому зв’язок має тим один до багатьох.
3. Авто може фігурувати у багатьох відомостях как транспорт для перевозки, тому зв’язок має тим один до багатьох.
4. Відомості містить інформацію про замовлення клієнта, тому зв’язок має тим багатьох до одного.

Таблиця 2.5.

Опис зв‘язків сутностей

Ім’я сутності	Зв’язок	Ім’я сутності	Тип зв’язку
Клієнт	Укладає	Договір	1:N
Робітник	Фігурує	Відомість	1:N
Авто	Фігурує	Відомість	1:N
Відомість	Містить	Клієнт	N:1

Концептуальна схема моделі приведена на рис. 2.6

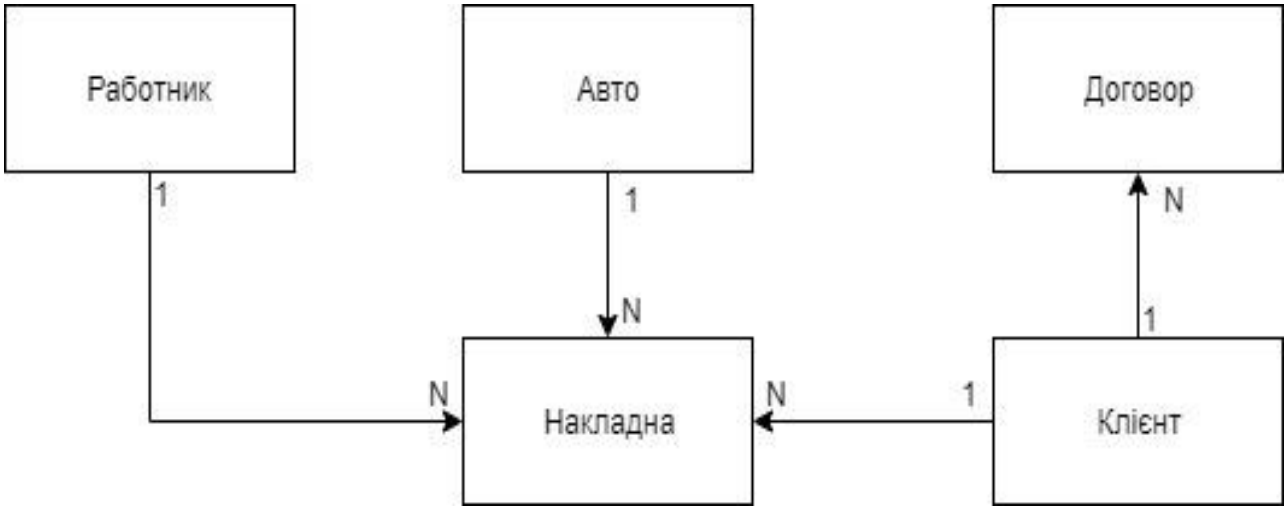


Рис. 2.6. Концептуальна схема моделі

Логічна модель даних – це модель даних конкретної предметної області, виражена незалежно від конкретного продукту керування базами даних або технології зберігання, але в термінах структур даних, таких як реляційні таблиці та колонки. Логічна модель даних включає в себе сутності (таблиці), атрибути(поля) та відношення(ключі), а також використовує бізнес-назви для сутностей і атрибутів, нормалізована до четвертої нормальної форми(нормальна форма застосовна в нормалізації баз даних) та не залежить від платформи та СКБД.

В процесі будування логічної моделі даних для предметної області “Транспортне підприємство” були визначені та створені зв’язки між таблицями(сутностями), встановлені первинні та вторинні ключі, типи даних та їх обмеження.

Логічна модель даних для даної предметної області приведена на рис. 2.7.

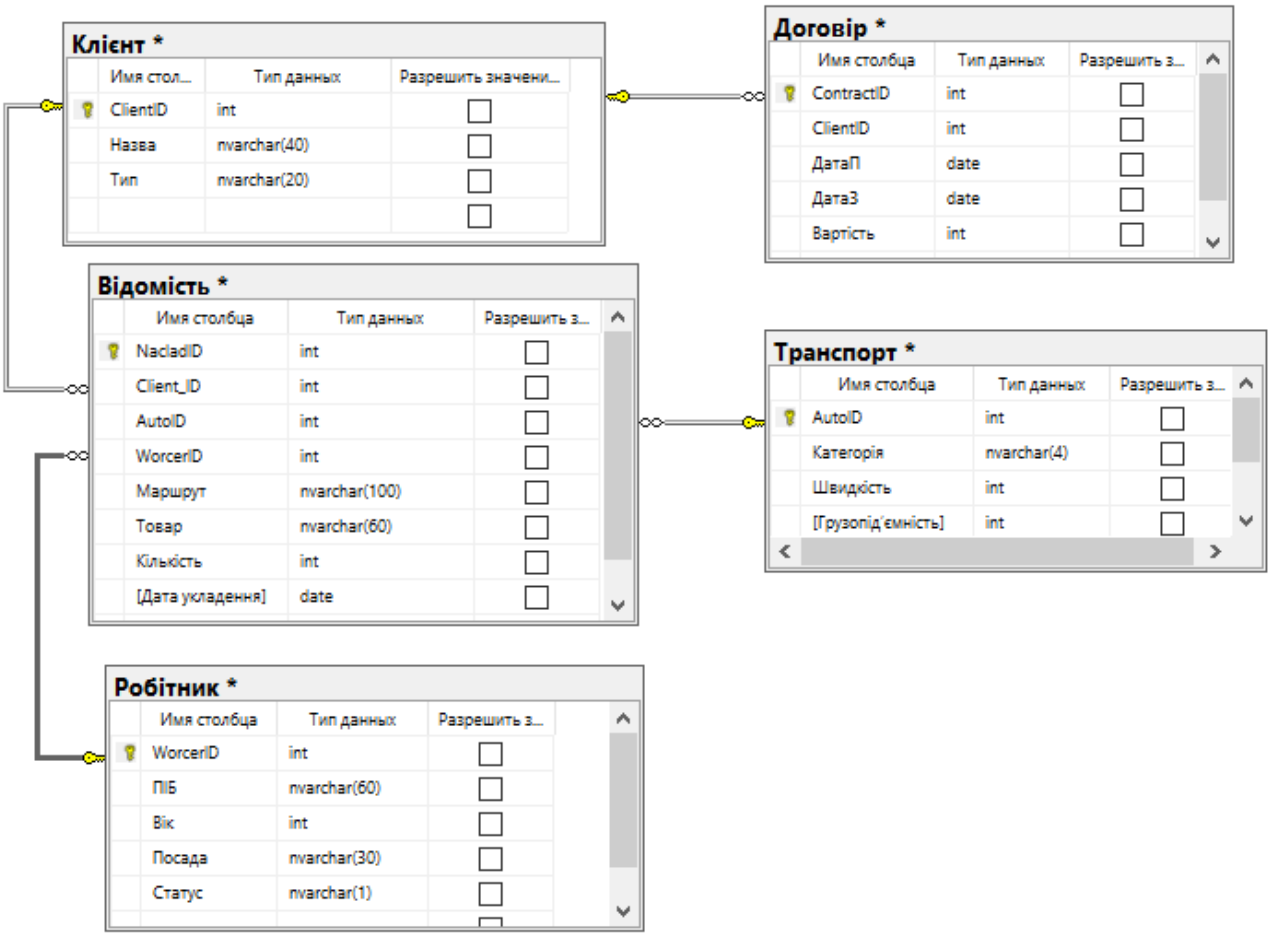


Рис. 2.7. Логічна модель даних

Захист інформації у базі даних є суттєвим фактором ефективного функціонування БД, а особливо у сфері технологій електронного уряду. Він складається із захисту від несанкціонованого та санкціонованого втручання. Перший напрям зв'язаний із забезпеченням безпеки БД, а другий – із забезпеченням цілісності БД. Безпека забезпечується підсистемою безпеки СУБД, яка перевіряє відповідність усіх запитів правилам безпеки, які зберігаються в системному каталозі. СУБД підтримує такі напрями забезпечення безпеки БД:

– Встановлення певного рівня доступу до даних;

– Шифрування даних;

У багатьох СУБД підтримується вибіркове або обов'язкове управління доступом до бази даних. При вибіркову управлінні кожен користувач має доступ

до даних тільки у межах свого зовнішнього представлення. Вибіркове управління складається за таких компонентів:

– Ім’я правила;
– Привілеї;
– Діапазон, до якого застосовується правило;
– Прізвища користувачів, які мають певні привілеї;

При обов’язковому управлінні доступом до бази даних кожен об’єкт має деякий рівень класифікації (таємний, цілком таємний, для службового користування та інші), а кожен користувач має певний рівень допуску за такими ж привілеями, що і на рівні класифікації. При обов’язковому управлінні виконуються такі правила безпеки:

– Користувач має доступ до об’єкту тільки в тому випадку, якщо його рівень більше або дорівнює рівню класифікації об’єкта;
– Користувач може модифікувати об’єкт тільки в тому випадку, якщо його рівень допуску дорівнює рівню класифікації об’єкта;

Цілісність даних стосується забезпечення вірності операцій, які виконує користувач та підтримання непротирічності даних. Виділяють такі обмеження цілісності:

– Обмеження атрибутів (тип і формат поля, завдання діапазону значень, ознака непустого поля та інші);
– Обмеження кортежів;
– Обмеження відношень;
– Обмеження БД;

При розробці бази даних “Транспортне підприємство” були встановлені певні рівні доступу до даних, обране вибіркове управління доступом до бази даних та проведена цілісність даних - були обмежені атрибути, кортежі, відношення та БД.

Проектування архітектури програмного забезпечення (ПЗ) -це процес розроблення, що виконується після етапу аналізу і формування вимог. Задача такого проектування — перетворення вимог до системи у вимоги до ПЗ і

побудова на їхній основі архітектури системи. Побудова архітектури системи здійснюється шляхом визначення цілей системи, її вхідних і вихідних даних, декомпозиції системи на підсистеми, компоненти або модулі та розроблення її загальної структури. Проектування архітектури системи може проводитися різними методами (стандартизованим, об'єктно-орієнтованим, компонентним і ін.), кожний з яких пропонує свій шлях побудови архітектури, а саме, визначення концептуальної, об'єктної й інших моделей за допомогою відповідних конструктивних елементів (блок-схем, графів, структурних діаграм тощо).

Існує декілька основних підходів до проектування програмної системи:

– Стандартизований підхід – регламентує стадії й етапи процесу розробки програмної системи на основі стандарту ГОСТ 34.601–90.

– Об'єктний підхід – дозволяє враховувати аспекти, властиві діючим особам (акторам) системи, створювати сценарії виконання системи тощо. Об'єктний стиль проектування — це декомпозиція майбутньої системи на окремі підсистеми (пакети), визначення функціональних і нефункціональних вимог і об'єктної моделі предметної області. Носіями інтересів, можливостей і дій в системі (або пакеті) є діючі особи — актори.

– Загальносистемний підхід – традиційний неформальний підхід до визначення архітектури системи, її компонентів, способів їхнього подання й об'єднання в систему.

Проектування архітектури системи завершується створенням опису, в якому відображені зафіксовані проектні рішення, логічна і фізична структура системи, а також способи взаємодії об'єктів.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Вхідними даними для роботи системи є заздалегідь створена база даних, що містить перелік інформації про співробітників, автотранспортні засоби, клієнтів, відомості та укладені договори.

Кожен розділ має свій ID. Окрім основної інформації про робітників, база даних також вказує на їх поточний робочий статус та посаду. Оскільки клієнти можуть бути як замовниками, так і відправниками, у базі даних є окремий пункт для уточнення їх статусу. Таблиця автотранспортних засобів дає наглядну інформацію щодо максимальної вантажопідйомності конкретного транспорту і уточнення швидкості для більш продуктивного планування перевезення товару. Загальне призначення відомостей – вказання точного часу, витраченого на перевезення та суму, виплачену замовником. Основна інформація зосереджена у таблиці з договорами: хто є замовником, визначений шлях перевезення товару, назва товару та його кількість.

У подальшому інформація у готовій базі даних може бути відредагована з використанням програми.

Форма MenuM, що відображає головне меню програми, дає можливість переглядати та формувати списки таблиць бази даних. У центральній частині форми знаходиться таблиця та пристрій навігації, які дозволяють переглянути списки робітників або сформувати новий договір між клієнтом та постачальником.

Друга форма для виведення даних – StatementForm. Відповідає за виведення вибіркової інформації з готових таблиць бази даних. Складається з чотирьох сторінок, що містять у собі таблицю для виведення даних, кнопку-активатор та елемент для уточнення вибіркового критерію.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

- ✓ процесор класу Intel ® Xeon з тактовою частотою 2.4GHz;
- ✓ шина даних - 1066MHz,
- ✓ кеш другого рівня - 2048 КБ;

- ✓ оперативна пам’ять 2 x DIMM DDR2-800 1024 Мб;
- ✓ жорсткі диски 3x 250 Гб SATA 2 16 Мб буфер, 7200 RPM;
- ✓ рідкокристалічний монітор з діагоналлю не менше 17 ";
- ✓ доступ до мережі Internet;
- ✓ клавіатура;
- ✓ маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Для реалізації програми використано Microsoft SQL Server.

Microsoft SQL Server представляє собою інтегрований пакет програм, призначений для:

- Створення, розгортання та управління промисловими програмами, які є безпечними, масштабованими та надійними.
- Збільшення продуктивності інформаційних технологій за рахунок зменшення складності побудови, розгортання та управління програмами по роботі з базами даних.
- Розділення даних між платформами, програмами та пристроями для полегшення поєднання внутрішніх та зовнішніх систем.

Платформа даних SQL Server включає наступні інструменти:

- Реляційна база даних: безпечне, надійне, масштабоване, легкодоступне ядро з покращеною продуктивністю і підтримкою структурованих та неструктурованих (XML) даних.
- Replication Services: реплікація даних для розподілених та мобільних програм обробки даних, висока доступність систем, масштабований паралелізм із вторинними сховищами даних та інтеграція з різнорідними системами, включаючи існуючі бази даних Oracle.
- Notification Services: розвинуті можливості повідомлень для розробки і впровадження масштабованих програм, що можуть доставляти персоналізовані

та своєчасні обновлення інформації великій кількості як підключених, так і мобільних пристроїв.

- Integration Services: засоби отримання, перетворення та завантаження інформації для сховищ даних та інтеграції даних в масштабі підприємства.

- Analysis Services: аналітична обробка інформації в реальному часі (OLAP) та технології інтелектуального аналізу даних (Data Mining), які використовуються для швидкого та складного аналізу великих і змішаних наборів даних.

- Reporting Services: засіб для створення, управління і доставки як традиційних паперових звітів, так і інтерактивних, що базуються на Web-технології.

- Інструменти управління: SQL Server включає засоби управління та настроювання баз даних. Стандартні протоколи доступу до даних суттєво зменшують час, необхідний для інтеграції даних SQL Server з існуючими системами. Крім того є вбудована підтримка Web-служб для забезпечення взаємодії з іншими системами та платформами.

- Інструменти розробки: SQL Server надає інтегровані інструменти розробки для ядра бази даних, отримання, трансформації та завантаження даних, OLAP та звітності, які тісно інтегровані з MS Visual Studio для надання наскрізних можливостей розробки інформаційних систем. Кожна головна підсистема SQL Server поставляється із своєю власною об'єктною моделлю та набором API для розширення системи в довільному напрямку, який є унікальним для бізнесу компанії.

Платформа даних SQL Server надає наступні переваги:

- Використання активів даних. SQL Server дозволяє користувачам отримати більше вигоди від їх даних завдяки використанню таких вбудованих функцій як звітність, аналіз та отримання інформації.

- Збільшення продуктивності. Завдяки широким можливостям інтелектуальних ресурсів підприємства та інтеграції з популярними програмами, такими як MS Office, SQL Server надає працівникам інформаційної сфери підприємства важливу та своєчасну інформацію, пристосовану для їх конкретних потреб.

- Зменшення складності інформаційної технології. SQL Server спрощує розробку, впровадження та управління галузями промисловості та аналітичними

програмами, надаючи програмістам гнучке середовище розробки та інтегровані, автоматизовані інструменти управління адміністраторам баз даних.

- Зниження загальної вартості володіння. Інтегрований підхід та фокусування на простоті використання і впровадження приводить до зменшення витрат на реалізацію і підтримку, що сприяє швидкому поверненню інвестицій в бази даних.

Сьогодні є велика кількість середовищ розробки програмного забезпечення. Найпоширініше використовуються наступні середовища розробки програмного забезпечення (IDE):

- Microsoft Visual Studio 2019;
- Embarcadero RAD Studio XE8.

Для розробки програмного додатку «Обліку кадрів підприємства» було обрано середу розробки «Embarcadero RAD Studio C++ Builder XE8», яка має ряд переваг:

- наявність більшої кількості стандартних компонентів, а також велика кількість бібліотек компонент від сторонніх розробників, що розширюють і доповнюють можливості стандартних;
- підтримка технологій ActiveX, OLE, COM, InterNet-технологій;
- простота у використанні, більша швидкість роботи та надійність;
- можливість генерації коду під платформи Win32, Win64;
- орієнтація на «візуальні» методи розробки програм, що дозволяє швидко і якісно спроєктувати і реалізувати стандартний користувальницький інтерфейс;
- перспективність, популярність і широка поширеність середовища розробки у світі.

C++ Builder — програмний продукт, інструмент швидкої розробки додатків (RAD), інтегроване середовище розробки (IDE), система, яка використовується програмістами для розробки програмного забезпечення на мові програмування C++.

Спочатку розроблявся компанією Borland Software, а потім її підрозділом CodeGear, який сьогодні належить компанії Embarcadero Technologies.

C++ Builder об'єднує в собі комплекс об'єктних бібліотек (STL, VCL, CLX, MFC та ін.), компілятор, зневаджувач, редактор коду та багато інших компонентів.

2.6.3. Виклик та завантаження програми

Спосіб виклику програми з відповідного носія даних та умови його завантаження є стандартними для запуску виконуючих файлів при роботі в ОС Windows. Додаткових чи специфічних вимог щодо запуску програми не встановлено, програма не потребує спеціального завантаження та налаштування. Для роботи потрібен ПК чи ноутбук з ОС Windows.

2.6.4. Опис інтерфейсу користувача

Тестування програмного забезпечення - це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись.

Тестування програмного забезпечення - техніка контролю якості, що перевіряє відповідність між реальною та очікуваною поведінкою програми завдяки кінцевому набору тестів, що обираються певним чином.

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування як процес, своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення і забезпечення якості програмного забезпечення, до якого належать усі складові ділового процесу, а не тільки тестування.

Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів — це по суті дослідницький та творчий процес, а не тільки створення і виконання рутинної процедури.

Основними засобами тестування, що використовувались при розробці системи були вбудовані у середовище розробки Embarcadero RAD Studio модулі тестування та лагодження.

Результати пройденого тестування свідчать про достатньо високу стабільність системи. Завдяки системі аналізу виключних ситуацій, навіть при неправильній роботі з системою (наприклад, при невірному введенні даних) вірогідність критичного збою мізерна. Система сама повідомляє користувача про знайдену помилку та надає варіанти її вирішення.

Програма призначена для:

- роботи з базою даних транспортної компанії
- отримання інформації за рахунок обробки даних;

Запуск програми гарантований на операційних системах Microsoft Windows 7 і вище. Про вдалий запуск системи свідчить поява головного вікна програми.

Для запуску програми потрібно відкрити виконуючий файл програми «Dpl.exe». Після запуску програми на екрані з’явиться головне вікно програми (рис.2.8).

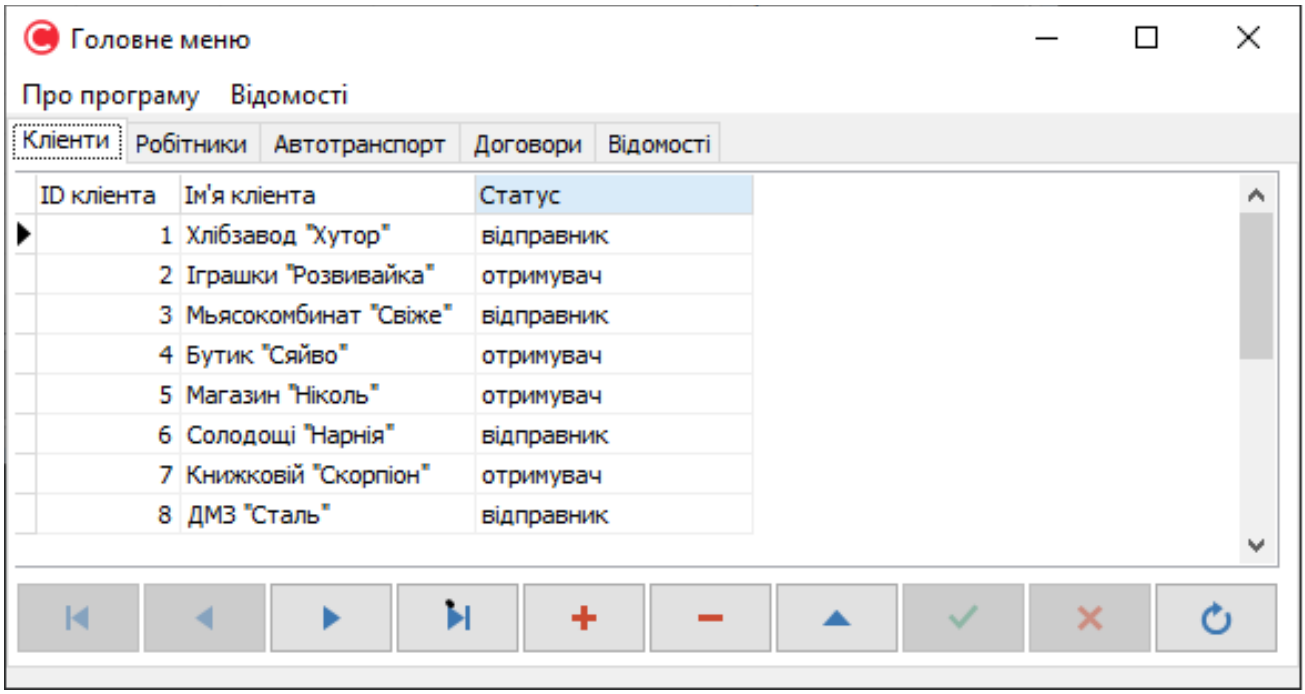
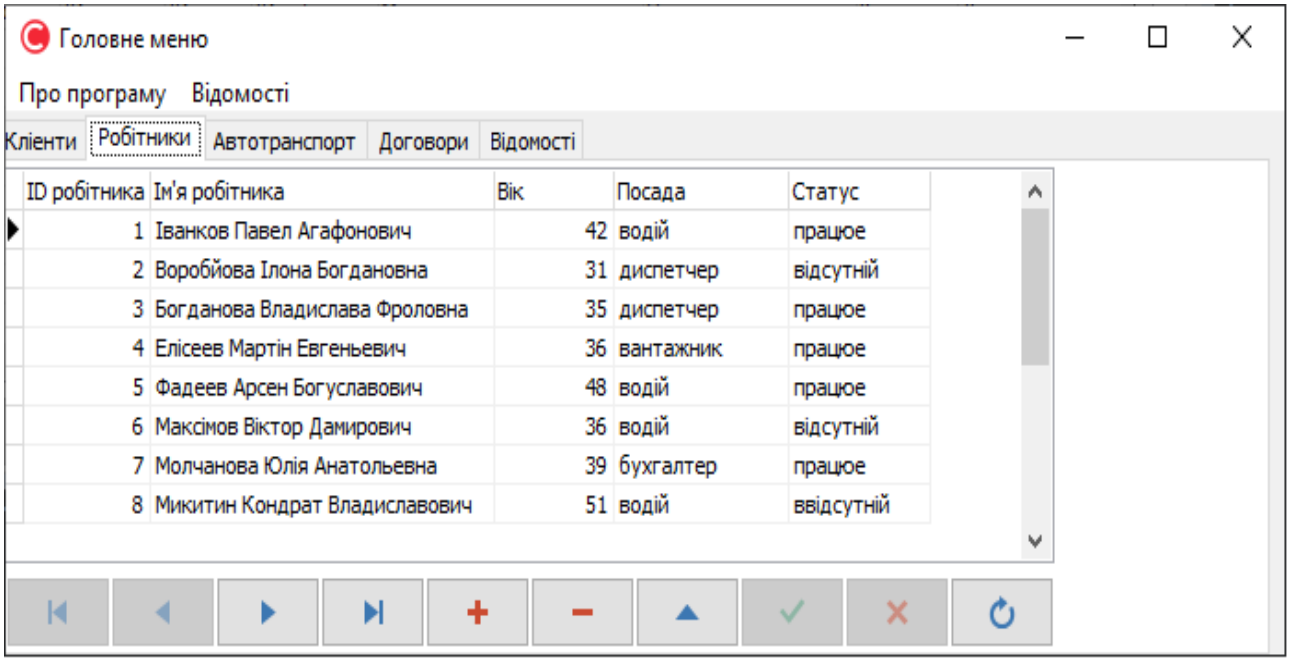


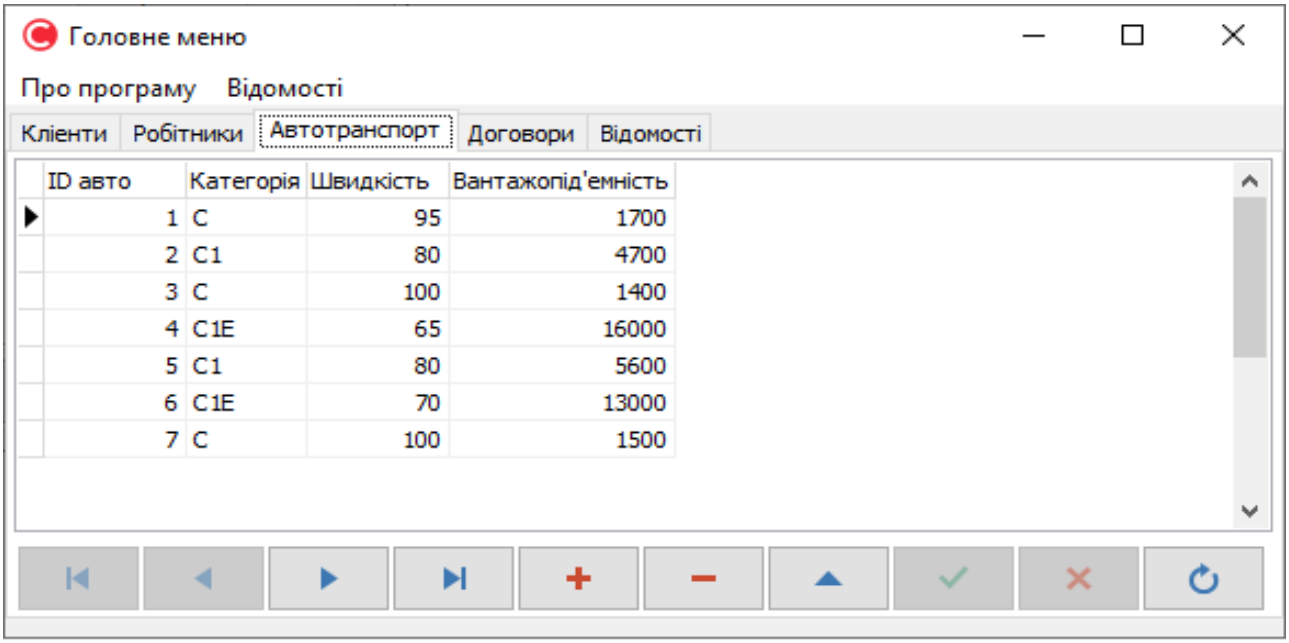
Рис. 2.8. Головна сторінка програми

На рис. 2.9-2.12 знаходяться різні вкладки головного меню, що дають можливість переглядати та редагувати інформацію у базі даних:



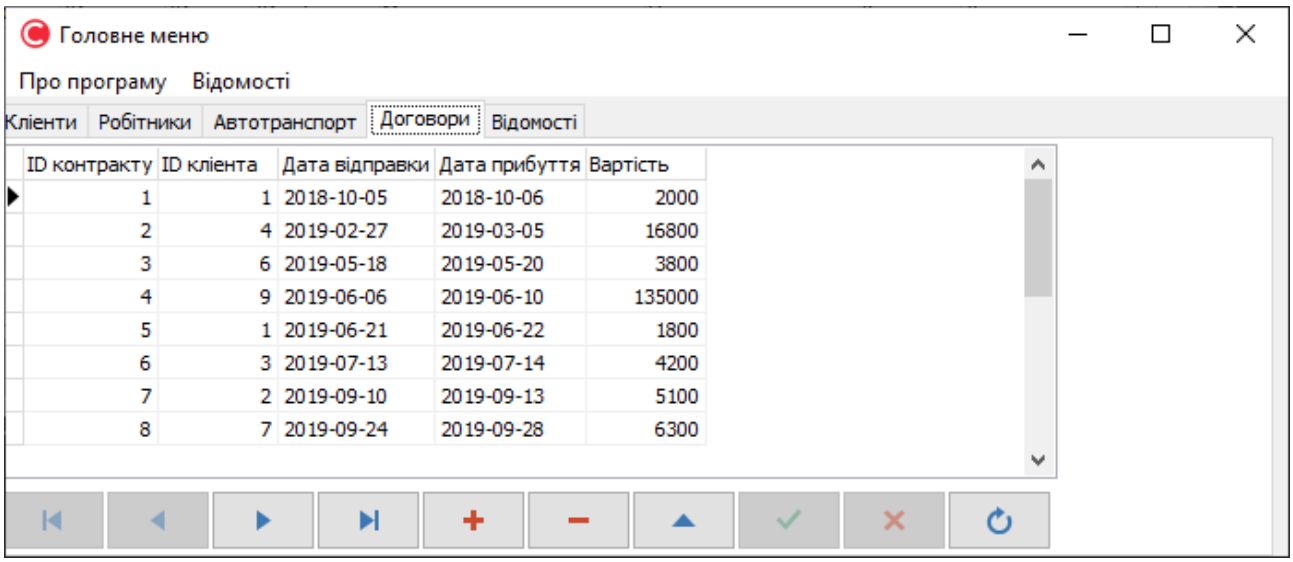
ID робітника	Ім'я робітника	Вік	Посада	Статус
1	Іванков Павел Агафонович	42	водій	працює
2	Воробйова Ілона Богдановна	31	диспетчер	відсутній
3	Богданова Владислава Фроловна	35	диспетчер	працює
4	Елісеев Мартін Евгеньевич	36	вантажник	працює
5	Фадеев Арсен Богуславович	48	водій	працює
6	Максімов Віктор Дамирович	36	водій	відсутній
7	Молчанова Юлія Анатольевна	39	бухгалтер	працює
8	Микитин Кондрат Владиславович	51	водій	ввідсутній

Рис. 2.9. Вкладка Робітник



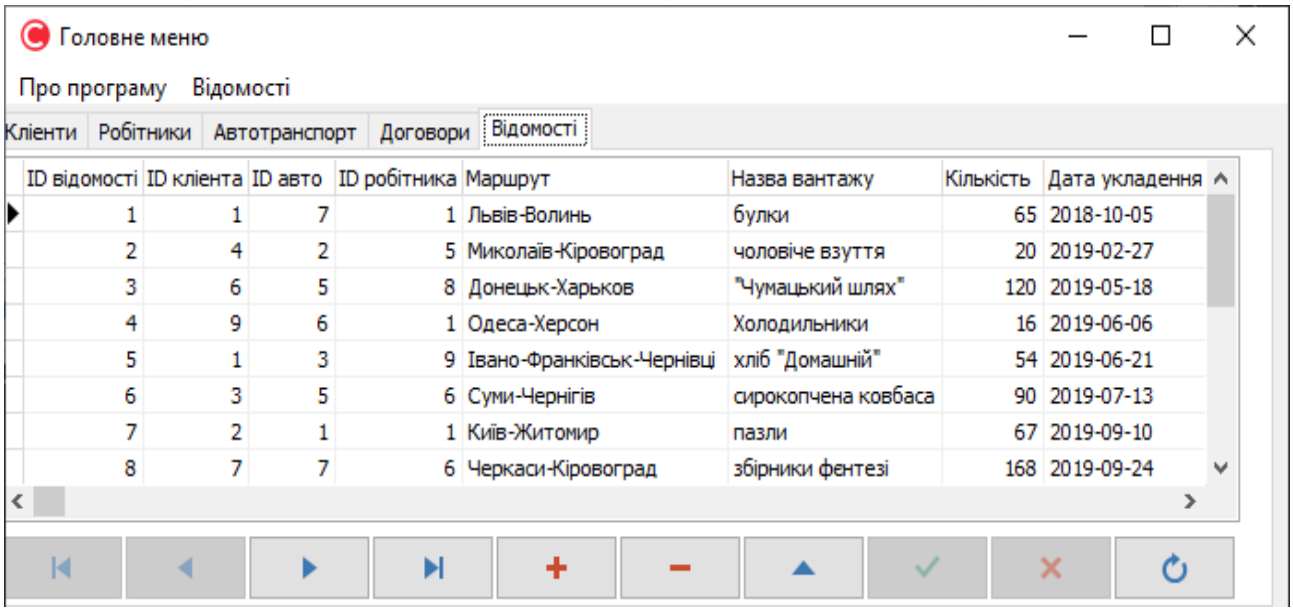
ID авто	Категорія	Швидкість	Вантажопід'ємність
1	C	95	1700
2	C1	80	4700
3	C	100	1400
4	C1E	65	16000
5	C1	80	5600
6	C1E	70	13000
7	C	100	1500

Рис. 2.10. Вкладка Автотранспорт



ID контракту	ID клієнта	Дата відправки	Дата прибуття	Вартість
1	1	2018-10-05	2018-10-06	2000
2	4	2019-02-27	2019-03-05	16800
3	6	2019-05-18	2019-05-20	3800
4	9	2019-06-06	2019-06-10	135000
5	1	2019-06-21	2019-06-22	1800
6	3	2019-07-13	2019-07-14	4200
7	2	2019-09-10	2019-09-13	5100
8	7	2019-09-24	2019-09-28	6300

Рис. 2.11. Вкладка Договори



ID відомості	ID клієнта	ID авто	ID робітника	Маршрут	Назва вантажу	Кількість	Дата укладення
1	1	7	1	Львів-Волинь	булки	65	2018-10-05
2	4	2	5	Миколаїв-Кіровоград	чоловіче взуття	20	2019-02-27
3	6	5	8	Донецьк-Харьков	"Чумацький шлях"	120	2019-05-18
4	9	6	1	Одеса-Херсон	Холодильники	16	2019-06-06
5	1	3	9	Івано-Франківськ-Чернівці	хліб "Домашній"	54	2019-06-21
6	3	5	6	Суми-Чернігів	сирокопчена ковбаса	90	2019-07-13
7	2	1	1	Київ-Житомир	пазли	67	2019-09-10
8	7	7	6	Черкаси-Кіровоград	збірники фентезі	168	2019-09-24

Рис. 2.12. Вкладка Відомості

Результат запитів з технічного завдання, можна переглянути на рис.2.13-2.17:

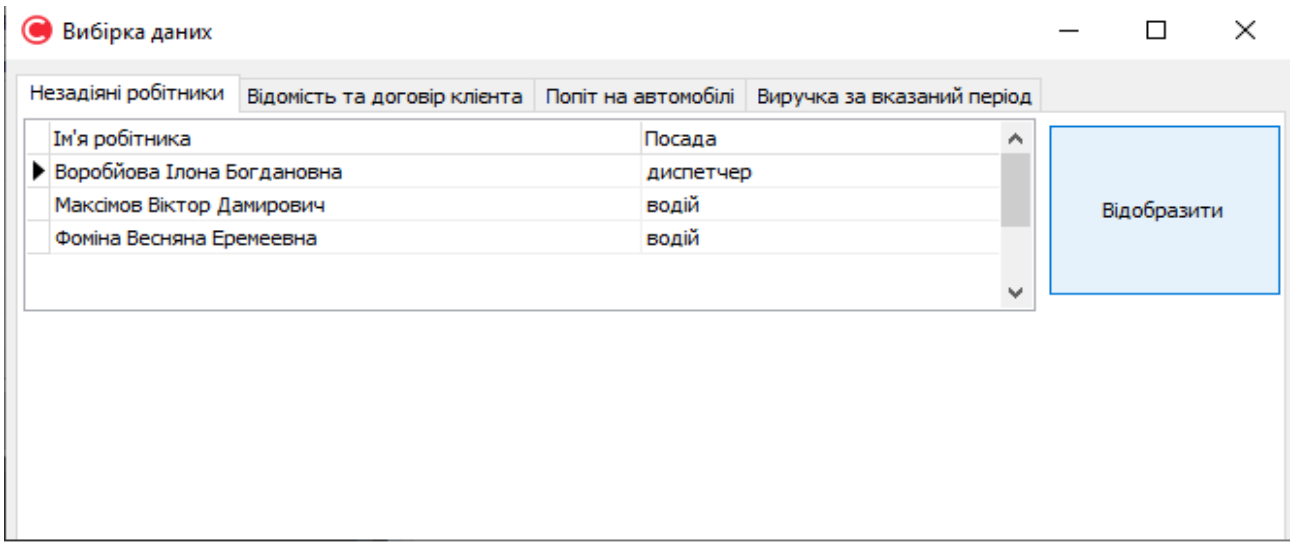


Рис. 2.13. Робітники, що не задіяні зараз у роботі

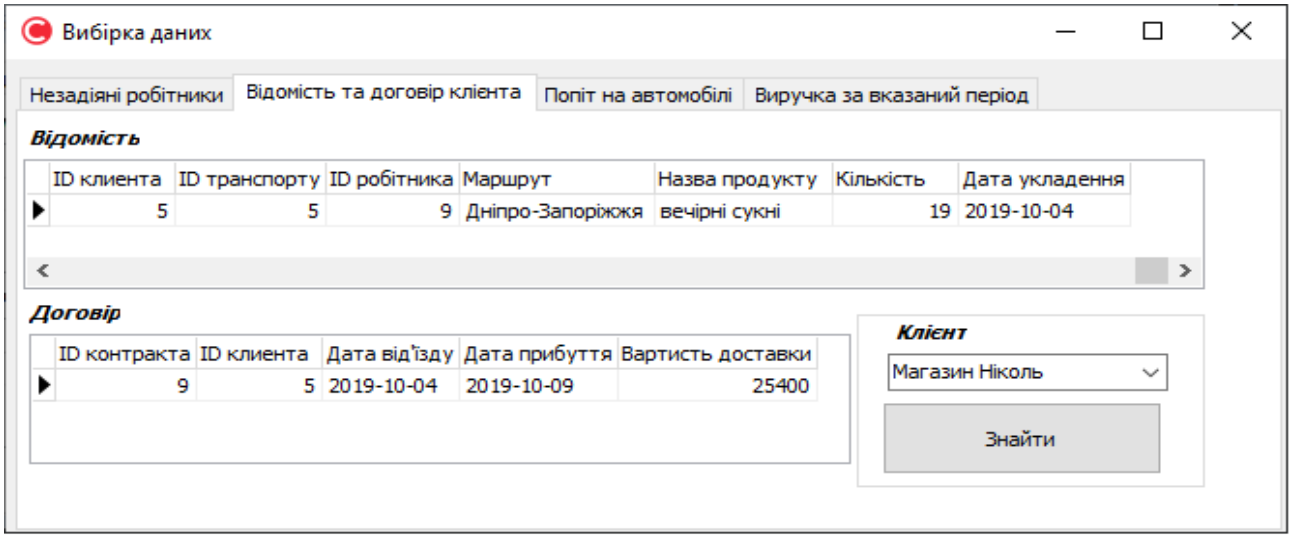


Рисунок 2.14. Договори та накладні пов'язані спільним клієнтом

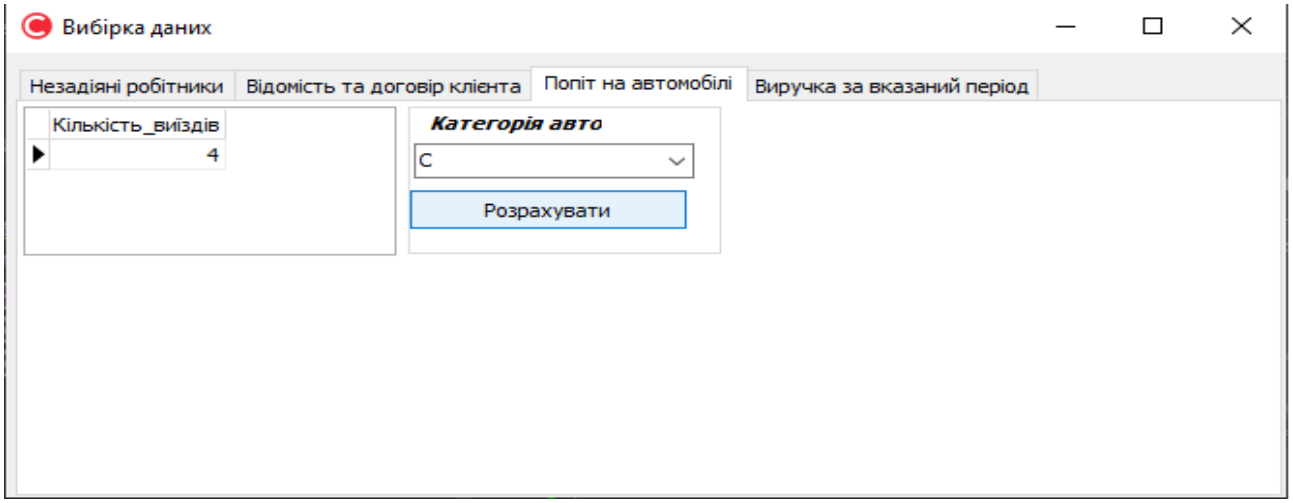


Рис. 2.15. Попит на обраний вид транспорту

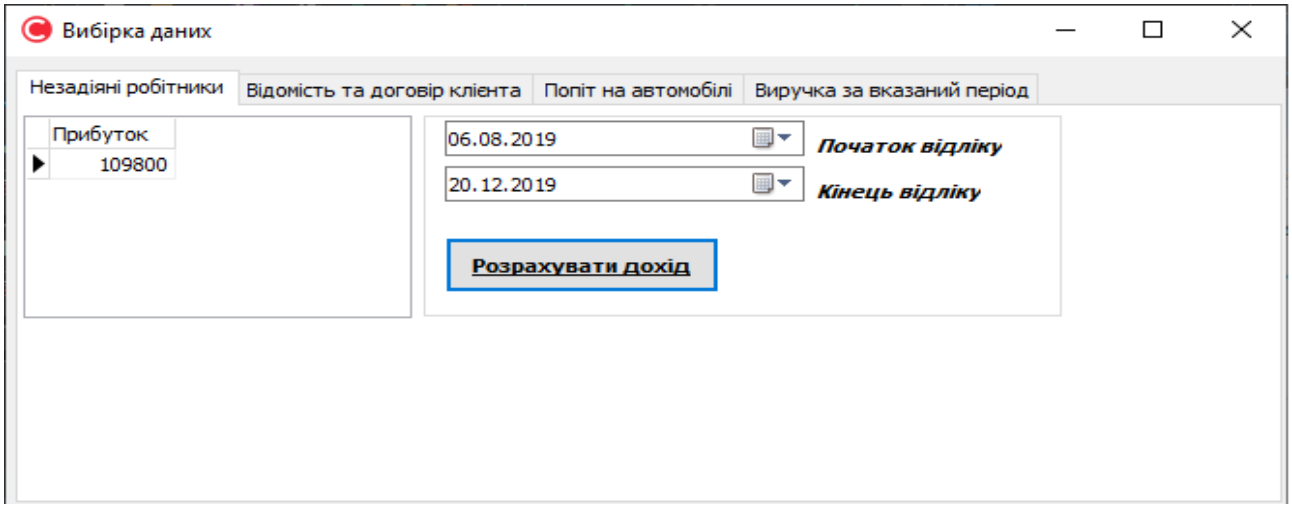


Рис. 2.16. Прибуток за вказаний період часу

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1200;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 60 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1200 \cdot (1 + 0{,}05) = 2016$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2016 \cdot 1{,}2) / (75 \cdot 1{,}2) = 26{,}88 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2016 / (20 \cdot 1{,}2) = 84 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2016 / (25 \cdot 1{,}2) = 67{,}2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2016 / (5 \cdot 1{,}2) = 336 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1,5 \cdot 336 = 504 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 2016 / (18 \cdot 1,2) = 93,33 \text{ людино-годин.}$$

$$t_{до} = 0,75 \cdot 93,33 = 70 \text{ людино-годин.}$$

$$t_{д} = 93,33 + 70 = 163,33 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 26,88 + 84 + 67,2 + 336 + 163,33 = 727,41 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 60 грн / год, отримуємо:

$$З_{ЗП} = 727{,}41 \cdot 60 = 43\,644{,}6 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 336 \cdot 13 = 4368 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 43644{,}6 + 4368 = 48\,012{,}6 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 727{,}41 / 1 \cdot 176 \approx 4 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для автоматизації ведення документації транспортної компанії. Вартість даного програмного забезпечення становить 48 тис. грн. і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 4 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В даній роботі приведена розробка інформаційної системи бази даних «Автоматизація обліку діяльності транспортного підприємства з перевезення вантажу».

База даних має мінімальну надмірність, всі таблиці знаходяться в третій нормальній формі, відсутні аномалії додавання, виключення, модифікації. Також спроектована система задовольняє усім сучасним вимогам ринку.

Система реалізована з використанням архітектури клієнт-сервер. В роботі дано аналіз і вибір засобів розробки програмної системи, описано проєктування бази даних, проектування програми і результати її роботи.

Робота має практичний напрям і може бути використана для автоматизації роботи транспортної компанії та полегшення роботи з документацією.

Також у проекті було визначено трудомісткість розробленої підсистеми 727 люд-год, проведений підрахунок вартості роботи по створенню програми 48012 грн. та розраховано час на його створення 4 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Виейра, Р. Программирование баз данных Microsoft SQL Server 2008. Базовый курс. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2010. – 816 с. : ил – Парал. тит. англ. – 1000 экз. – ISBN 978-5-8459-1612-9 (рус.).
2. Уолтерс, Р. SQL Server 2008. Ускоренный курс для профессионалов. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2010. – 498 с. : ил – Парал. тит. англ. – 1000 экз. – ISBN 978-5-8459-1481-1 (рус.).
3. Найт Б., Пэтел К., Снайдер В., Лофорт Р., Уорт С. Для профессионалов Microsoft SQL Server 2008. Руководство администратора. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2010. – 618 с. : ил – Парал. тит. англ. – 1000 экз. – ISBN 978-5-8459-1594-8 (рус.).
4. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2011. – 624 с. : ил. – Парал. тит. англ. ISBN 978-5-8459-0768-4 (рус.).
5. Шилдт, Герберт. Полный справочник по C++, 4-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2011. – 800 с. : ил. – Парал. тит. англ. ISBN 978-5-8459-0489-8 (рус.).
6. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2007. – 496 с.: ил. ISBN 5-94074-334-X.
7. Ицик Бен-Ган, Диджан Сарка, Рон Талмейдж. Microsoft SQL Server 2012. Создание запросов. Учебный курс Microsoft. : Пер. с англ. – М. : Русская редакция, 2014. –720 с.: ил. – Парал. тит. англ. ISBN 978-5-7502-0432-8 (рус.).
8. Архангельский А.Я. C++ Builder 6. Справочное пособие. Книга 1. Язык C++. – М.: Бином-Пресс, 2002 г. – 544 с.: ил. ISBN 978-5-9518-0229-3.
9. О.М. Томашевський, Г.Г. Цетелик, М.Б. Вітер, В.І. Дубук. Інформаційні технології та моделювання бізнес-процесів [Текст]: навчальний посібник — К.: «Центр Учбової літератури», 2012. — 296 с. – ISBN 978-617-673-003-3.
10. Радченко, Г.И. Распределенные вычислительные системы / Г.И. Радченко. – Челябинск, 2012. – 184 с. ISBN 978-5-89879-198-8.
11. Раскин Д. Интерфейс: новые направления в проєктировании компьютерных систем. — Пер. с англ. — СПб: Символ-Плюс, 2004. — 272 с., ил. ISBN 5-93286-030-8.

12. Коннолли Т., Бегг К. Базы данных. Проєктирование, реализация и сопровождение. Теория и практика. — 3-е изд. — М.: «Вильямс», 2003. — 1436 с. — ISBN 0-201-70857-4.
13. А.Аллан. Клієнтська розробка для професіоналів. Node.js – СПБ.:2017. – 220с.
14. Комисаров Д.А., Станкевич А.Г. Персональный учитель по персональному компьютеру. – М, 2000
15. Angular и TypeScript. Сайтостроение для профессионалов, 2018 р, 2020. - 752 с. Яков Файн, Антон Моисеев.
16. Хэррон, Д. Node.js Разработка серверных веб-приложений на JavaScript / Д. Хэррон. - М.: ДМК, 2018. - 144 с.
17. Белоногов, Г.Г. Автоматизация процессов накопления, поиска и обобщения информации / Г.Г. Белоногов, А.П. Новоселов. - М.: Наука, 2017. - 256 с.
18. Website и Web Application: в чем разница? [Електронний ресурс] - Режим доступу: https://dinarys.com/ru/blog/websitevs.-webapplication
19. Дейт, К.Дж. Введение в системы баз данных / К.Дж. Дейт. - К.: Диалектика; Издание 6-е, 2004. - 784 с.
20. Інькова Н. А. Створення Web-сайтів: Навчально-методичний посібник [Електронний ресурс] / Інькова Н.А., Зайцева Е.А., Кузьміна Н.В., Толстих С.Г. / / Режим доступу: http://club-edu.tambov.ru/methodic/fio/p5.doc
21. Середня заробітна плата програміста у Дніпрі станом на початок 2021 року. [Електронний ресурс] - Режим доступу: https://dou.ua/lenta/articles/salary-report-devs-june-2020/
22. Вартість аренди ноутбуку почасово [Електронний ресурс] - Режим доступу: https://notebooksbu.com/garantiya-3-goda/
23. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун- т». – Д.: НГУ, 2013. – 11 с.

24. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 6.050101 «Комп’ютерні науки / І.М. Удовик, Л.М. Коротенко, О.С. Шевцова. Нац. гірн. ун-т. – Д : НТУ «Дніпровська політехніка» . - 2018. – 65 с.

ДОДАТОК А

КОД ПРОГРАМИ

```
#include <vcl.h>
#pragma hdrstop

#include "About.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TAboutForm *AboutForm;
//---------------------------------------------------------------------------
__fastcall TAboutForm::TAboutForm(TComponent* Owner)
        : TForm(Owner)
{
}

#include <vcl.h>
#pragma hdrstop

#include "Connect.h"
#include "Statement.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TConnectForm *ConnectForm;
//---------------------------------------------------------------------------
__fastcall TConnectForm::TConnectForm(TComponent* Owner)
        : TForm(Owner)
{
}

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "MainMenu.h"
#include "Connect.h"
#include "About.h"
#include "Statement.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "DBAxisGridsEh"
```

```
#pragma link "DBGridEh"
#pragma link "DBGridEhGrouping"
#pragma link "DBGridEhToolCtrls"
#pragma link "DynVarsEh"
#pragma link "EhLibVCL"
#pragma link "GridsEh"
#pragma link "ToolCtrlsEh"
#pragma resource "*.dfm"
TMenuM *MenuM;
//---------------------------------------------------------------------------
__fastcall TMenuM::TMenuM(TComponent* Owner)
        : TForm(Owner)
{

}

void __fastcall TMenuM::N1Click(TObject *Sender)
{
        AboutForm->ShowModal();
}

void __fastcall TMenuM::N2Click(TObject *Sender)
{
        StatementForm->ShowModal();
}
//---------------------------------------------------------------------------


void __fastcall TMenuM::FormClose(TObject *Sender, TCloseAction &Action)
{
if (ConnectForm->ADOConnect->Connected) {
                        ConnectForm->TableClient->Active=false;
                        ConnectForm->TableTransport->Active=false;
                        ConnectForm->TableWorker->Active=false;
                        ConnectForm->TableContract->Active=false;
                        ConnectForm->TableStatement->Active=false;
                        ConnectForm->ADOConnect->Connected = false;
 }
}
//---------------------------------------------------------------------------



//---------------------------------------------------------------------------

#include <vcl.h>
```

```
#pragma hdrstop

#include "Statement.h"
#include "MainMenu.h"
#include "Connect.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TStatementForm *StatementForm;
//---------------------------------------------------------------------------
__fastcall TStatementForm::TStatementForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


void __fastcall TStatementForm::WorkersClick(TObject *Sender)
{   ADOQ->Active=false;
        ADOQ->SQL->Clear();
        ADOQ->SQL->Add("SELECT Worker.NameWorker as [²ì'ÿ ðîá³òíèêà], Worker.Position as
[Ïîñàäà]");
        ADOQ->SQL->Add("From Worker where Worker.Status = 'â³äñóòí³é'");
        ADOQ->Active=true;
}
//---------------------------------------------------------------------------


void __fastcall TStatementForm::ButtonClientClick(TObject *Sender)
{
        ADOQ->Active=false;
        ADOQ2->Active=false;
        ADOQ->SQL->Clear();
        ADOQ2->SQL->Clear();

        ADOQ->SQL->Add("SELECT Contract.ContractID as [ID êîíòðàêòà], Contract.ClientID as
[ID êëèåíòà], ");
        ADOQ->SQL->Add("Contract.DateStart as [Äàòà â³ä'¿çäó], Contract.DateEnd as [Äàòà
ïðèáóòòÿ], ");
        ADOQ->SQL->Add("Contract.Cost as [Âàðòèñòü äîñòàâêè]");
        ADOQ->SQL->Add("From Contract inner join Client");
        ADOQ->SQL->Add("on Contract.ClientID=Client.ClientID");
        ADOQ->SQL->Add("WHERE Client.NameClient=:d");
        ADOQ->Parameters->ParamByName("d")->Value=ClientBox->KeyField;
```

```
        ADOQ2->SQL->Add("SELECT Statement.NacladID as [ID â³äîìîñò³], Statement.ClientID
as [ID êëèåíòà], ");
        ADOQ2->SQL->Add("Statement.AutoID as [ID òðàíñïîðòó], Statement.WorcerID as [ID
ðîá³òíèêà], ");
        ADOQ2->SQL->Add("Statement.Route as [Ìàðøðóò], Statement.NameProduct as [Íàçâà
ïðîäóêòó], ");
        ADOQ2->SQL->Add("Statement.ProductCount as [Ê³ëüê³ñòü], Statement.DateSign as
[Äàòà óêëàäåííÿ], ");
        ADOQ2->SQL->Add("From Statement inner join Client");
        ADOQ2->SQL->Add("on Statement.ClientID=Client.ClientID");
        ADOQ2->SQL->Add("WHERE Client.NameClient=:d");
        ADOQ2->Parameters->ParamByName("d")->Value=ClientBox->KeyField;

        ADOQ->Active=true;
        ADOQ2->Active=true;
}
//---------------------------------------------------------------------------


void __fastcall TStatementForm::ButtonAutoClick(TObject *Sender)
{
        ADOQ->Active=false;
        ADOQ->SQL->Clear();
        ADOQ->SQL->Add("select  COUNT (Statement.AutoID) as Ê³ëüê³ñòü_âè¿çä³â");
        ADOQ->SQL->Add("From from Statement inner join Transport ");
        ADOQ->SQL->Add("on Statement.AutoID=Transport.AutoID");
        ADOQ->SQL->Add("WHERE Transport.Category=:dy");
        ADOQ->Parameters->ParamByName("dy")->Value=AutoBox->KeyField;
        ADOQ->Active=true;
}
//---------------------------------------------------------------------------

void __fastcall TStatementForm::ButtonSumClick(TObject *Sender)
{
        ADOStoredProc1->Close();
  ADOStoredProc1->Parameters->ParamByName("@DataB")->Value = DateB->Date;
  ADOStoredProc1->Parameters->ParamByName("@DataE")->Value = DateE->Date;
  ADOStoredProc1->Open();
}
//---------------------------------------------------------------------------

#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
```

```
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Imaging.pngimage.hpp>
//---------------------------------------------------------------------------
class TAboutForm : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TImage *Image1;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
private:        // User declarations
public:         // User declarations
        __fastcall TAboutForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TAboutForm *AboutForm;
//---------------------------------------------------------------------------
#endif

//---------------------------------------------------------------------------

#ifndef ConnectH
#define ConnectH
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Data.DB.hpp>
#include <FireDAC.Comp.Client.hpp>
#include <FireDAC.Comp.UI.hpp>
#include <FireDAC.Phys.hpp>
#include <FireDAC.Phys.Intf.hpp>
#include <FireDAC.Phys.ODBCBase.hpp>
#include <FireDAC.Stan.Async.hpp>
#include <FireDAC.Stan.Def.hpp>
#include <FireDAC.Stan.Error.hpp>
#include <FireDAC.Stan.Intf.hpp>
#include <FireDAC.Stan.Option.hpp>
#include <FireDAC.Stan.Pool.hpp>
#include <FireDAC.UI.Intf.hpp>
#include <FireDAC.VCLUI.Wait.hpp>
#include <FireDAC.Comp.DataSet.hpp>
#include <FireDAC.DApt.hpp>
```

```
#include <FireDAC.DApt.Intf.hpp>
#include <FireDAC.DatS.hpp>
#include <FireDAC.Stan.Param.hpp>
#include <Data.Win.ADODB.hpp>
//---------------------------------------------------------------------------
class TConnectForm : public TForm
{
__published:   // IDE-managed Components
        TADOConnection *ADOConnect;
        TADOTable *TableClient;
        TADOTable *TableTransport;
        TADOTable *TableWorker;
        TADOTable *TableContract;
        TADOTable *TableStatement;
        TDataSource *SourceClient;
        TDataSource *SourceTransport;
        TDataSource *SourceWorker;
        TDataSource *SourceContract;
        TDataSource *SourceStatement;
private:        // User declarations
public:         // User declarations
        __fastcall TConnectForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TConnectForm *ConnectForm;
//---------------------------------------------------------------------------
#endif

//---------------------------------------------------------------------------

#ifndef MainMenuH
#define MainMenuH
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.Menus.hpp>
#include <Vcl.ComCtrls.hpp>
#include <Vcl.DBCtrls.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Data.DB.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.Grids.hpp>
//---------------------------------------------------------------------------
class TMenuM : public TForm
```

```
{
__published:   // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TTabSheet *TabSheet3;
        TTabSheet *TabSheet4;
        TTabSheet *TabSheet5;
        TDBGrid *DBGrid1;
        TDBGrid *DBGrid2;
        TDBGrid *DBGrid3;
        TDBGrid *DBGrid4;
        TDBGrid *DBGrid5;
        TDBNavigator *DBNavigator1;
        TDBNavigator *DBNavigator3;
        TDBNavigator *DBNavigator2;
        TDBNavigator *DBNavigator4;
        TDBNavigator *DBNavigator5;
        void __fastcall N1Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);



private:        // User declarations
public:         // User declarations
        __fastcall TMenuM(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TMenuM *MenuM;
//---------------------------------------------------------------------------
#endif

//---------------------------------------------------------------------------

#ifndef StatementH
#define StatementH
//---------------------------------------------------------------------------
/*#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
```

```
#include <Data.DB.hpp>
#include <Vcl.ComCtrls.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.Grids.hpp>
#include <Data.Win.ADODB.hpp>
#include <Vcl.DBCtrls.hpp>  */
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ComCtrls.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Data.DB.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.Grids.hpp>
#include <FireDAC.Comp.Client.hpp>
#include <FireDAC.Comp.DataSet.hpp>
#include <FireDAC.DApt.hpp>
#include <FireDAC.DApt.Intf.hpp>
#include <FireDAC.DatS.hpp>
#include <FireDAC.Phys.Intf.hpp>
#include <FireDAC.Stan.Async.hpp>
#include <FireDAC.Stan.Error.hpp>
#include <FireDAC.Stan.Intf.hpp>
#include <FireDAC.Stan.Option.hpp>
#include <FireDAC.Stan.Param.hpp>
#include <Data.Win.ADODB.hpp>
#include <Vcl.DBCtrls.hpp>
//---------------------------------------------------------------------------
class TStatementForm : public TForm
{
__published:   // IDE-managed Components
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TTabSheet *TabSheet3;
        TTabSheet *TabSheet4;
        TButton *Workers;
        TDBGrid *DBGrid1;
        TDataSource *DataSource1;
        TADOQuery *ADOQ;
        TDBGrid *DBGrid2;
        TDBGrid *DBGrid3;
        TLabel *Â³äîì³ñòü;
        TLabel *Label1;
        TADOQuery *ADOQ2;
```

```
        TDataSource *DataSource2;
        TDBGrid *DBGrid4;
        TGroupBox *GroupBox1;
        TLabel *Label4;
        TButton *ButtonClient;
        TGroupBox *GroupBox2;
        TLabel *Label2;
        TButton *ButtonAuto;
        TGroupBox *GroupBox3;
        TDBGrid *DBGrid5;
        TDateTimePicker *DateB;
        TDateTimePicker *DateE;
        TLabel *Label3;
        TLabel *Label5;
        TButton *ButtonSum;
        TADOStoredProc *ADOStoredProc1;
        TDataSource *DataSource3;
        TDBLookupComboBox *ClientBox;
        TDBLookupComboBox *AutoBox;
        void __fastcall WorkersClick(TObject *Sender);
        void __fastcall ButtonClientClick(TObject *Sender);
        void __fastcall ButtonAutoClick(TObject *Sender);
        void __fastcall ButtonSumClick(TObject *Sender);
private:        // User declarations
public:         // User declarations
        __fastcall TStatementForm(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TStatementForm *StatementForm;
//---------------------------------------------------------------------------
#endif

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
//---------------------------------------------------------------------------
USEFORM("MainMenu.cpp", MenuM);
USEFORM("Statement.cpp", StatementForm);
USEFORM("About.cpp", AboutForm);
USEFORM("Connect.cpp", ConnectForm);
//---------------------------------------------------------------------------
int WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
{
        try
```

```
	{
		Application->Initialize();
		Application->MainFormOnTaskBar = true;
		Application->CreateForm(__classid(TMenuM), &MenuM);
		Application->CreateForm(__classid(TConnectForm), &ConnectForm);
		Application->CreateForm(__classid(TAboutForm), &AboutForm);
		Application->CreateForm(__classid(TStatementForm), &StatementForm);
		Application->Run();
	}
	catch (Exception &exception)
	{
		Application->ShowException(&exception);
	}
	catch (...)
	{
		try
		{
			throw Exception("");
		}
		catch (Exception &exception)
		{
			Application->ShowException(&exception);
		}
	}
	return 0;
}
//---------------------------------------------------------------------------
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом Осіпов.doc	Пояснювальна записка . Документ Word.
Диплом Осіпов.pdf	Пояснювальна записка в форматі PDF.
Програма	
Diplom Осіпов.zip	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація Осіпов.ppt	Презентація роботи.