

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБЛІКУ РОБОТИ
АВТОТРАНСПОРТНОГО ПІДПРИЄМСТВА
SOFTWARE FOR KEEPING TRACK OF A TRUCKING
COMPANY PERFORMANCE

Виконав: студент(ка) 4 курсу, групи КНТ-219сп
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Гришков О.В.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Коцур М.І.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гришкова Олексія Віталійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення обліку роботи автотранспортного підприємства. Software for Keeping Track of a Trucking Company Performance

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

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 7 червня 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання “ 25 ” квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис) Гришков О.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Гофман Є.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
94 с., 36 рис., 4 табл., 4 дод., 24 джерела.

АВТОТРАНСПОРТНЕ ПІДПРИЄМСТВО, БАЗА ДАНИХ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТРАНСПОРТ, C# , JAVA

Об'єкт дослідження – процес розробки програмного забезпечення для автотранспортних підприємств.

Предмет дослідження – програмні засоби обліку роботи автотранспортного підприємства.

Метою роботи є програмна реалізація процесу обліку роботи автотранспортного підприємства.

Матеріали, методи та технічні засоби: мова програмування Java, середовище розробки, система керування базами даних C#.

Результати. Розроблено програмні засоби для обліку роботи автотранспортного підприємства.

Практична цінність роботи полягає у розробці програмного забезпечення для автоматизації обліку діяльності автотранспортного підприємства.

Висновки. Розроблено програмне забезпечення для автоматизації обліку діяльності автотранспортного підприємства.

Галузь використання – автотранспортні підприємства.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 36 figures, 4 tables, 4 appendixes, 24 sources.

MOTOR VEHICLE ENTERPRISE, DATABASE, SOFTWARE, TRANSPORT, CACHE, JAVA

The object of research is the process of software development for trucking companies.

The subject of the research is software tools for accounting the work of a motor transport enterprise.

The purpose of this work is the program implementation of the process of accounting for the work of the trucking company.

Materials, methods and technical tools: Java programming language, development environment, Cache database management system.

Results. The software tools for accounting the work of the motor transport enterprise have been developed.

The practical value of the work lies in the development of software to automate the accounting of the motor transport enterprise.

Conclusions. The software for automation of the account of activity of the motor transport enterprise is developed.

Scope of use – trucking companies.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Основні поняття про діяльність автотранспортних підприємств.....	10
1.2 Інформаційні потоки та облікова інформація, що використовується при автоматизації діяльності АТП	11
1.3 Програмне забезпечення автотранспортних підприємств	12
1.4 Аналіз програмних засобів обліку діяльності автотранспортних підприємств	14
1.5 Висновки за розділом 1	25
2 Матеріали і методи	26
2.1 Вибір мови програмування та інших технологій розробки	26
2.2 Вибір засобів роботи з базою даних	31
2.3 Висновки за розділом 2	34
3 Опис програми	35
3.1 Основні рішення щодо подання даних системи	35
3.2 Основні розроблені алгоритми.....	36
3.3 Основні рішення щодо модульного подання системи.....	37
3.4 База даних програмного забезпечення обліку роботи автотранспортного підприємства	39
3.5 Опис інтерфейсу користувача програмного забезпечення обліку роботи автотранспортного підприємства.....	40
3.6 Висновки за розділом 3	41
4 Експлуатація, тестування та експериментальне дослідження програми.....	42
4.1 Призначення й умови застосування програми	42
4.2 Характеристики програми	43
4.3 Інструкція по експлуатації програми.....	43
4.3.1 Звернення до програми.....	43

4.3.2 Вхідні й вихідні дані	43
4.3.3 Повідомлення.....	44
4.4 Тестування та результати роботи програмного забезпечення обліку роботи автотранспортного підприємства.....	44
4.5 Висновки за розділом 4	49
Висновки	50
Перелік джерел посилання	52
Додаток А Технічне завдання	54
Додаток Б Опис програми	58
Додаток В Текст програми	61
Додаток Д Слайди презентації	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

АТП – автотранспортне підприємство;

БД – база даних;

ПЗ – програмне забезпечення;

ПП – програмний проект;

ОС – операційна система;

СКБД – системама керування базами даних;

СР – середовище розробки;

ТЗ – транспортний засіб.

ВСТУП

Важливим завданням організації і планування діяльності автотранспортних підприємств є раціональне поєднання і використання наявних ресурсів з метою одержання максимального прибутку при забезпеченні процесу перевезення вантажів [1]-[5].

Тому актуальною є розробка програмного забезпечення для автоматизації процесу обліку діяльності автотранспортного підприємства (АТП), що зменшити витрати на утримання штату працівників та пришвидшити процес обробки інформації.

Отже, метою роботи є програмна реалізація процесу обліку роботи автотранспортного підприємства. Тобто необхідно створити програмне забезпечення, що надає працівникам АТП можливість вносити інформацію про своїх співробітників (водіїв) до бази даних та її автоматичної обробки для підрахунку прибутку.

Для досягнення поставленої мети в роботі необхідно розв'язати такі задачі:

- виконати аналіз предметної області стосовно процесу розробки програмного забезпечення для автотранспортних підприємств;
- виконати проєктування програмного забезпечення для обліку роботи автотранспортного підприємства;
- створити програмне забезпечення для обліку роботи автотранспортного підприємства;
- виконати тестування розробленого програмного забезпечення обліку роботи автотранспортного підприємства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття про діяльність автотранспортних підприємств

Автотранспортне підприємство (АТП) – це вид комплексного транспортного підприємства, що займається перевезенням вантажів і пасажирів, зберіганням, ремонтом і обслуговуванням рухомого складу, постачанням витратних матеріалів та запасних частин [1], [3], [5].

Автотранспортна компанія спеціалізується на наданні певних видів продукції та послуг, пов'язаних з переміщенням вантажів і пасажирів, а також діяльності, пов'язаної з цим процесом або компенсує його [1]-[5]. Залежно від виду виконуваної транспортної роботи АТП поділяють на такі види [1]-[5]:

- вантажні – перевезення;
- перевезення пасажирів на автобусах та перевезення пасажирів на автомобілі;
- сумісні перевезення пасажирів і різного вантажу;
- АТП, що спеціалізується на реалізації окремих видів дорожньо-транспортних робіт (сміттєвози, дорожньо-будівельна техніка, пожежні автомобілі, машини швидкої допомоги тощо).

Виробничий процес АТП включає основне, допоміжне та обслуговуюче виробництво [1]-[5]. Основним виробництвом є транспортування, що важливо для АТП [1]-[5]. Таке виробництво потребує обслуговування та побічних робіт. Допоміжне виробництво АТП – це ряд виробничих процесів для забезпечення технічної готовності рухомого складу, що використовується в основному виробництві [1]-[5]. Обслуговуюче виробництво не виробляє реальної продукції, а забезпечує первинне та допоміжне виробництво, інформаційні послуги, технічне обслуговування, контроль якості ремонтів, а також забезпечення допоміжного виробництва енергоресурсів [1]-[5].

1.2 Інформаційні потоки та облікова інформація, що використовується при автоматизації діяльності АТП

У структурі АТП адміністративна, планова, господарська, бухгалтерська, інженерно-технічна, експлуатаційна служби створюють власні інформаційні потоки, які можуть змінювати стан і напрямок вхідного потоку та отримувати необхідну інформацію та дані [1]-[5]. Інформацію, що циркулює в системі АТП, можна класифікувати за її функціональною та прагматичною природою (за використанням) [1]-[5]. Розвиваючи нові зовнішні ринки транспортних послуг та співпрацюючи з клієнтами в інших країнах, важливо знати стан макросередовища – доступ і відповідність комерційно-правовим, нормативним та обмежуючим вимогам загалом та транспорту зокрема, стійкість, природний клімат, демографічні та інші фактори [1]-[5].

Спеціальні інформаційні блоки описують стан інформаційного потоку в зовнішньому середовищі АТП та забезпечують набір даних інформаційним потоком, необхідним для визначення функціонування об'єктів зовнішнього середовища АТП, включаючи замовників, постачальників, посередників, конкурентів і населення [1]-[5]. Ці об'єкти є джерелами інформації і формують інформаційний потік у зовнішньому середовищі АТП [1]-[5]. Це складна система взаємопов'язаних потоків інформації про макро та середнє середовища, яка визначає АТП [1]-[5].

Фактично з точки зору організації інформаційного потоку всі компоненти внутрішнього та зовнішнього середовища АТП є об'єктами-носіями ознак, які можуть зацікавити АТП у вирішенні тих чи інших завдань [1]-[5]. Наприклад, керівництво АТП може бути зацікавлено в зборі інформації про навколишнє середовище (кількість потенційних клієнтів, переваги, потреби та очікування) для оцінки ринкового потенціалу нового регіону, включаючи потенційних покупців, які можуть заповнити пакет замовлень судноплавних компаній, можливе замовлення транспортних послуг;

конкуренти, що працюють на цьому ринку). Систематизація запропонованого потоку інформації веде до виявлення місця та характеру особливих обставин діяльності АТП, які необхідно вчасно вирішити для вибору ефективної стратегії управління [1]-[5].

Інформаційний потік може створюватися і керуватися компанією АТП самостійно [1]-[5]. В інформаційному просторі компанії АТП потік інформації із зовнішнього середовища є структурно однорідним і односпрямованим. Потік інформації з внутрішнього середовища компанії АТП є однорідним, багатограним і пов'язаним з конкретними гравцями транспортного ринку або державними органами та регуляторами [1]-[5]. Ми розглядаємо однорідність або неоднорідність інформації з точки зору її використання: технічний, нормативно-правовий, адміністративний, організаційний, технологічний, економічний, комерційний, юридичний тощо. Важливо розрізняти окремі та контрольовані інформаційні потоки [1]-[5]. Окремі інформаційні потоки містять інформацію, безпосередньо пов'язану з проблемою. Потік інформації моніторингу спрямований на моніторинг довгострокових тенденцій розвитку об'єкта АТП. Наприклад, на транспортний ринок активно виходять сильні конкуренти, які раніше працювали на інших ринках [1]-[5]. Його поява створює проблемну ситуацію для компанії з довгою історією на цьому ринку. Таким чином, при розробці програмного забезпечення обліку роботи автотранспортного підприємства важливо виділити суттєві інформаційні потоки для конкретного підприємства [1]-[5].

1.3 Програмне забезпечення автотранспортних підприємств

Виділяють три типи систем автоматизації підприємств автомобільного транспорту: FMS-, TMS- та GPS-системи [1]-[5]. Система FMS – система управління для перегляду рейсів у базі даних, розрахунку витрати палива, обліку використання транспортних засобів, обліку інвентаризації та технічного обслуговування, обліку праці водіїв, обліку заробітної плати,

планування технічного обслуговування та обліку витрат [1]-[5]. Більшість систем цієї категорії розроблено на платформі 1С. Це пояснюється тим, що програмні продукти та прикладні послуги мають низьку вартість. Система TMS – це система керування транспортними засобами, що поєднує в собі замовлення, автоматизацію транспортної логістики, оптимальне планування маршруту та розрахунок транспортних витрат. Система відстеження GPS – це програмне забезпечення, яке компанія купує при покупці пристрою GPS. Це програмне забезпечення визначає положення автомобіля, паливні канали, роботу двигуна тощо, залежно від встановлених датчиків [1]-[5].

Програмні застосунки для транспортних компаній належать до локальних систем через їх відносно невелику функціональність і можливість роботи з простими ієрархічними системами керування базами даних (СКБД) [1]-[5]. Роль таких систем полягає в наданні інформації та автоматизації роботи функціональних підрозділів АТП, основними з яких є: прийом транспортних замовлень, техніко-економічне планування замовлень (вибір оптимального маршруту), оперативне управління транспортним процесом (контроль конкретних водіїв, транспортних засобів, транспортних засобів на лінії, робота в дорозі), внесення динамічних змін до запланованого маршруту при відхиленні транспортного процесу від плану; ведення обліку роботи водіїв, пробігу транспортних засобів, робочого часу, використання паливно-мастильних матеріалів та рахунків за виконані замовлення; закриття рахунків із замовниками, ведення обліку основних засобів АТП та амортизації з урахуванням заробітної плати водіїв, слюсарів, інженерів та допоміжного персоналу, аналіз основних показників діяльності підприємства та транспортних витрат; ведення господарського обліку нашийників (запчастини, шини, комплектуючі акумуляторів); планування робіт з технічного обслуговування та ремонту транспортних засобів [1]-[5]. Підсистема управління транспортом АТП (або оперативна служба) є основною виробничою одиницею, яка функціонально інтегрує техніко-економічне

планування та оперативне управління підсистемами транспортного процесу. Обов'язки з оперативного обслуговування включають [1]-[5]:

- управління вантажами;
- забезпечення виконання існуючого бізнес-плану АТП;
- забезпечення найбільш ефективного використання транспортних контейнерів (вагонів) на необхідному рівні якості перевезень;
- забезпечення безпеки руху транспортних засобів на лінії;
- аналіз вантажопотоків та аналіз клієнтів та транспортного ринку;
- укладення договори з замовниками;
- планування оптимальних під'їзних шляхів, моніторинг під'їзних шляхів, пунктів навантаження та розвантаження;
- вивчення можливості підвищення рівня механізації та автоматизації ремонту транспортних засобів;
- планування чергування та підготовка завдання для водіїв.

Типова структура технічного обслуговування може відрізнятися залежно від кількості та типу транспортних засобів, а також типу АТП.

1.4 Аналіз програмних засобів обліку діяльності автотранспортних підприємств

Проаналізуємо різні програмні застосунки, що використовуються для обліку діяльності автотранспортних підприємств.

Програма Omnitracс [6] – це розроблене в США програмне забезпечення, що використовується для автоматизації АТП (рис. 1.1). Має управлінську об'єктивну зорієнтованість [6].

Перевагами програмного забезпечення Omnitracс є такі [6]:

- інтерфейс дуже зручний для водія;
- легке та функціональне налаштування;
- галузева універсальність;
- висока захищеність інформації;

- можливість відстеження транспортних засобів (рис. 1.2);
- простота використання звітів на смартфоні.

[Driver Log Editor](#)

[Unassigned Vehicle Activity](#)

[Reports](#)

[Operation Summary](#)

Report Selector

[Generate Report](#)

1 of 1 [Next](#)

[Go Back to the Parent Report](#)

Operation Summary Report by Driver

From: 12/29/2015 To: 01/04/2016

Organization: Multiple

Driver: All

	Total			Safety					Stop				PTO		
Driver	Fuel Economy (mpg)	Distance (mi)	Fuel (gal)	Over Speed Distance (mi)	% of Over Speed Distance	RPM % High	Hard Brakes Count	High Speed mi (Brake)	Total Stop Time (hh:mm:ss)	Avg. Stop Time (hh:mm:ss)	Idle Time (hh:mm:ss)	% of Idle Time	Idle Fuel (gal)	PTO Time (hh:mm:ss)	PTO Fuel (gal)
CHARLOTTE															
BETH BADE (12245)	6.6	311.8	138.79	96.1	10.5	0.0	10	6.3	27:09:38	00:21:44	00:48:05	3.0	0.45	00:00:00	0.00
	8.4	259.9	30.88	0.0	0.0	0.6	0	25.4	05:31:29	00:23:41	00:11:05	3.3	0.20	00:00:00	0.00
ENDERLIN ALBERT (14562)	6.4	799.8	125.71	0.0	0.0	0.3	0	3.6	27:29:20	00:32:59	00:20:10	1.2	0.19	00:00:00	0.00
	7.0	858.9	122.38	0.0	0.0	0.0	0	10.5	28:21:35	00:41:30	00:34:52	2.0	0.44	00:00:00	0.00
	8.1	732.1	90.02	340.2	46.5	4.5	2	11.0	18:47:49	00:28:12	01:54:54	10.2	0.45	00:00:00	0.00
	7.2	354.8	48.83	3.1	0.9	1.2	0	2.2	17:06:07	00:36:39	00:21:23	2.1	0.21	00:00:00	0.00
Total for 4 Drivers(s)	7.0	3897.3	554.81	439.4	11.3	1.0	12	4.1	124:26:08	00:30:06	04:10:29	3.4	1.54	00:00:00	0.00

Copyright © 2015 XRD Corporation. All Rights Reserved.

Рисунок 1.1 – Інтерфейс програми Omnitrac [6]

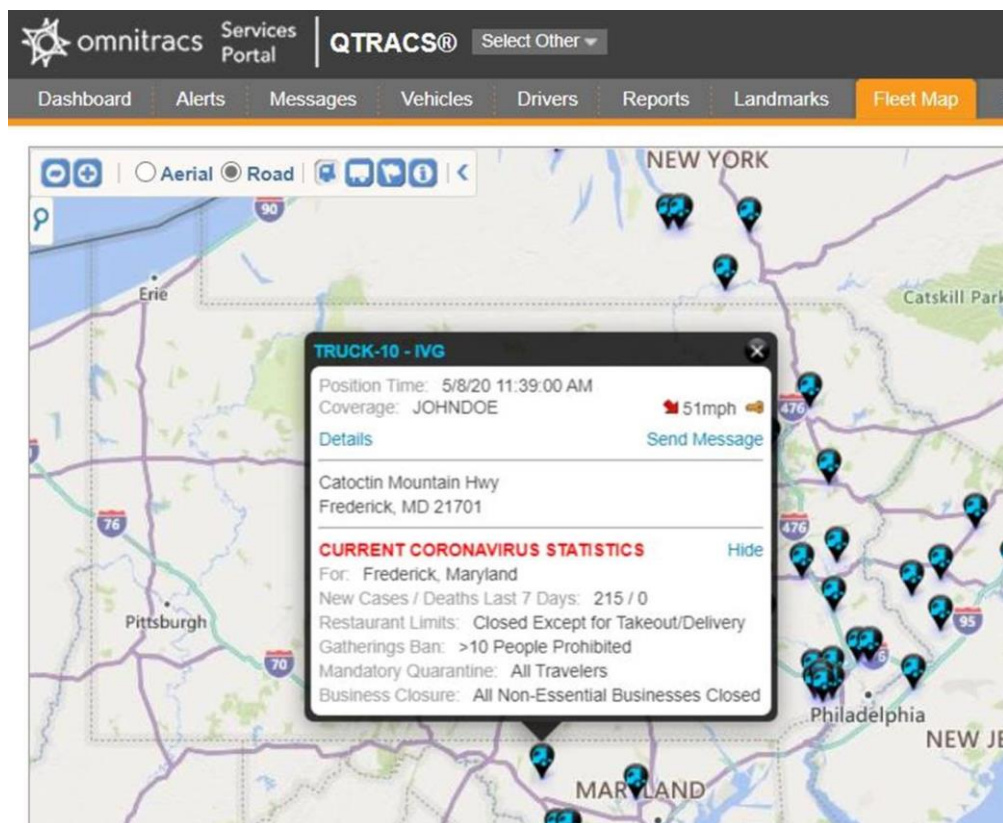


Рисунок 1.2 – Застосування програми Omnitrac
для відстеження транспортних засобів [6]

Недоліками програмного забезпечення Omnitrac є такі: відділи продажів та бухгалтерського обліку повільно працюють; проблеми з техпідтримкою.

Програмне забезпечення GPS Trackit використовується для автоматизації АТП. Наявна комплексна автоматизація та окремі елементи управління (рис. 1.3) [7].

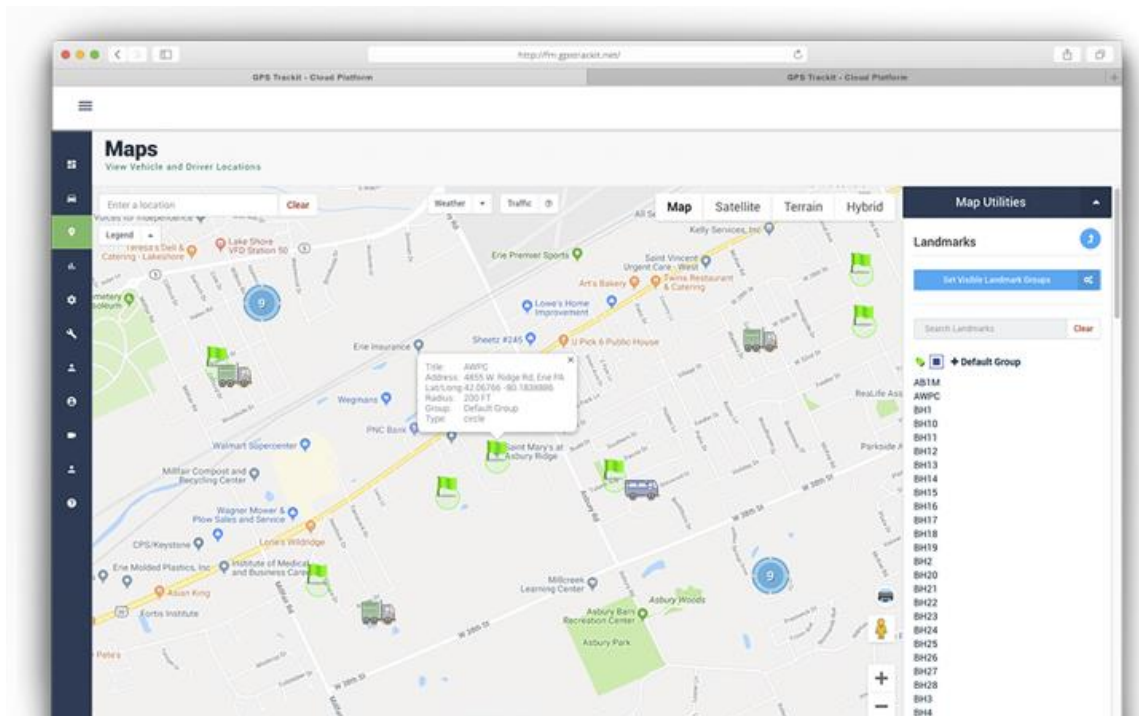


Рисунок 1.3 – Програмне забезпечення GPS Trackit [7]

Перевагами програмного забезпечення GPS Trackit є такі [7]:

- наявність служби звітів і відстеження;
- накладення погоди точніше, ніж будь-який радар;
- можливість відстежувати вантажівки та вантаж;
- двосторонній обмін повідомленнями;
- оптимізація маршруту;
- годинник водія та настроювані форми.

Недоліками програмного забезпечення GPS Trackit є такі [7]:

- проблеми з оплатою рахунків;
- іноді замовлення пристроїв онлайн буває складним.

Програмне забезпечення EROAD – розроблене використовується для автоматизації АТП, має управлінську об'єктивну зорієнтованість (рис. 1.4) [8].

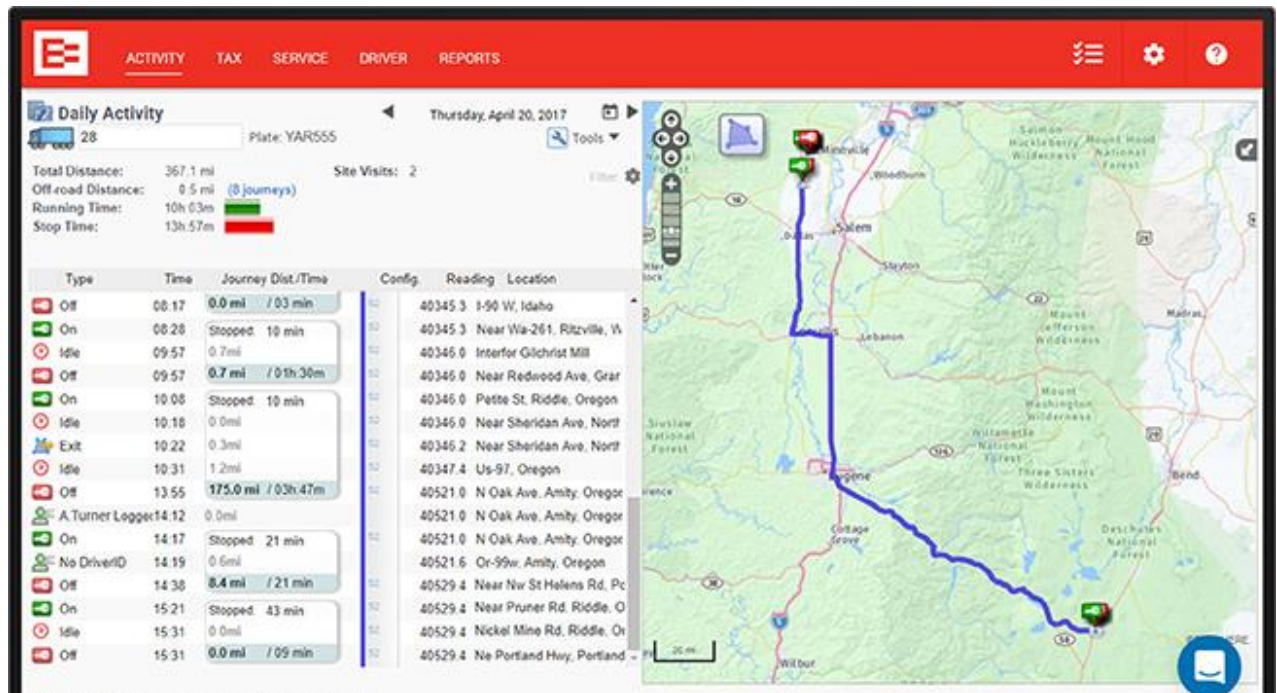


Рисунок 1.4 – Програмне забезпечення EROAD [8]

Переваги програмного забезпечення EROAD [8]:

- відстеження в реальному часі;
- звіти про діяльність, управління податку на паливо та інструменти для оптимізації обслуговування та контролю витрат на паливо;
- відеореєстратори і відстеження причепів.

Недоліки програмного забезпечення EROAD: затримка прийому до Ehub, неможливість створювати звіти IFTA та відсутність можливості шукати BOL [8].

Програмне забезпечення Tailwind TMS (рис. 1.5) створено для транспортних компаній, вантажних брокерських компаній і тих, хто робить і те, і інше [9].

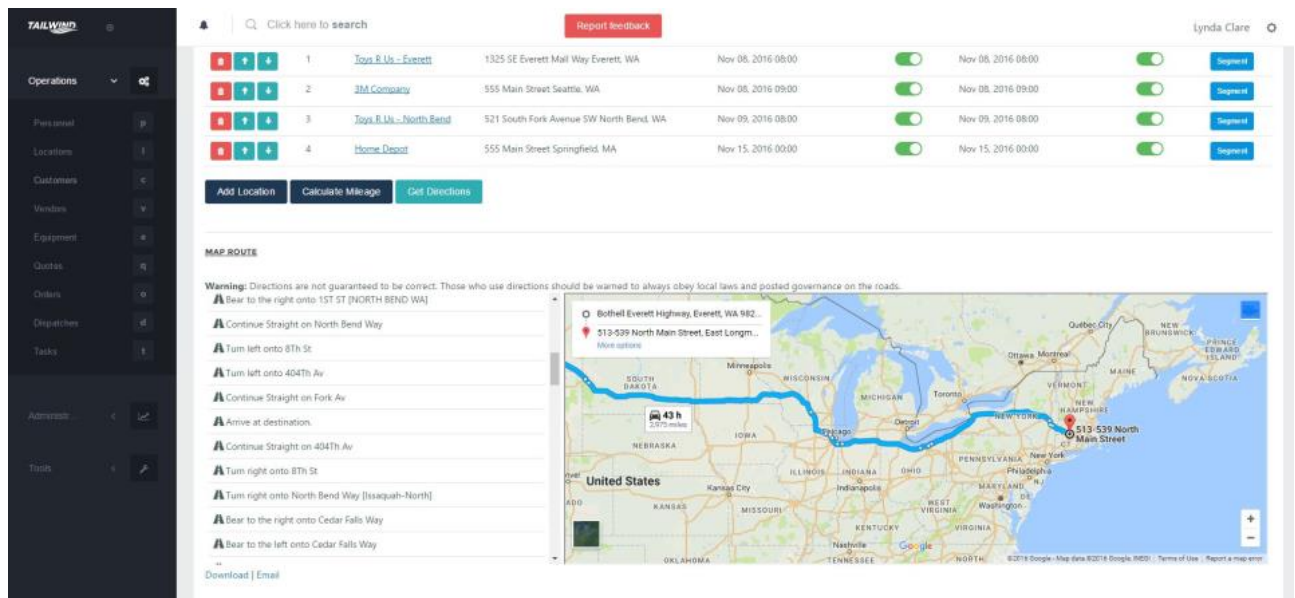


Рисунок 1.5 – Програмне забезпечення Tailwind TMS [9]

Переваги програмного забезпечення Tailwind TMS: програма допомагає керувати операціями, клієнтами, диспетчеризацією, адміністратором і включає безкоштовний додаток для захоплення POD для всіх водіїв; зручність розрахунків [9].

Недоліки програмного забезпечення Tailwind TMS: складність та незручність інтерфейсу; іноді працює дуже повільно; довгий час виправлення помилок [9].

Програмне забезпечення Fullbay розроблене для можливості використання з метою обліку технічного обслуговування вантажівок та обладнання (рис. 1.6) [10].

Переваги програмного забезпечення Fullbay [10]:

- відстеження технічного обслуговування вантажівок та обладнання;
- забезпечення безпеки транспортних засобів;
- зберігання повної історії ремонтів;
- зручний портал для клієнтів дозволяє підтримувати легкий безконтактний зв'язок із клієнтами та надає їм можливість побачити прогрес і дозволити ремонт;
- зручна система бухгалтерського обліку.

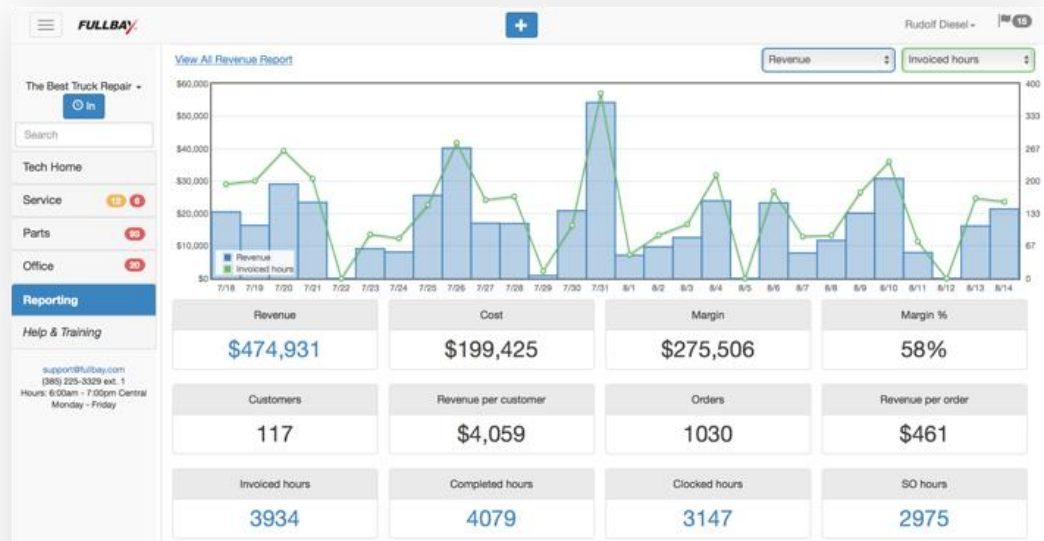


Рисунок 1.6 – Програмне забезпечення Fullbay [10]

Недоліки програмного забезпечення Fullbay [10]:

- висока вартість;
- довгі завантаження;
- повільність системи.

Програмне забезпечення Switchboard розроблене програмне забезпечення, що використовується для автоматизації АТП в США та Канаді для спрощення операцій з управління автопарком (рис. 1.7) [11].

Переваги програмного забезпечення Switchboard [11]:

- спрощує операції з управління автопарком;
- відстеження активів/транспортних засобів;
- простий у використанні ELD; керування водіями.

Недоліки програмного забезпечення Switchboard [11]:

- проблеми з серверами та помилками на стороні клієнта;
- погана техпідтримка та ігнорування клієнтів;
- неможливо обрізати або редагувати відеопотік в режимі реального часу.

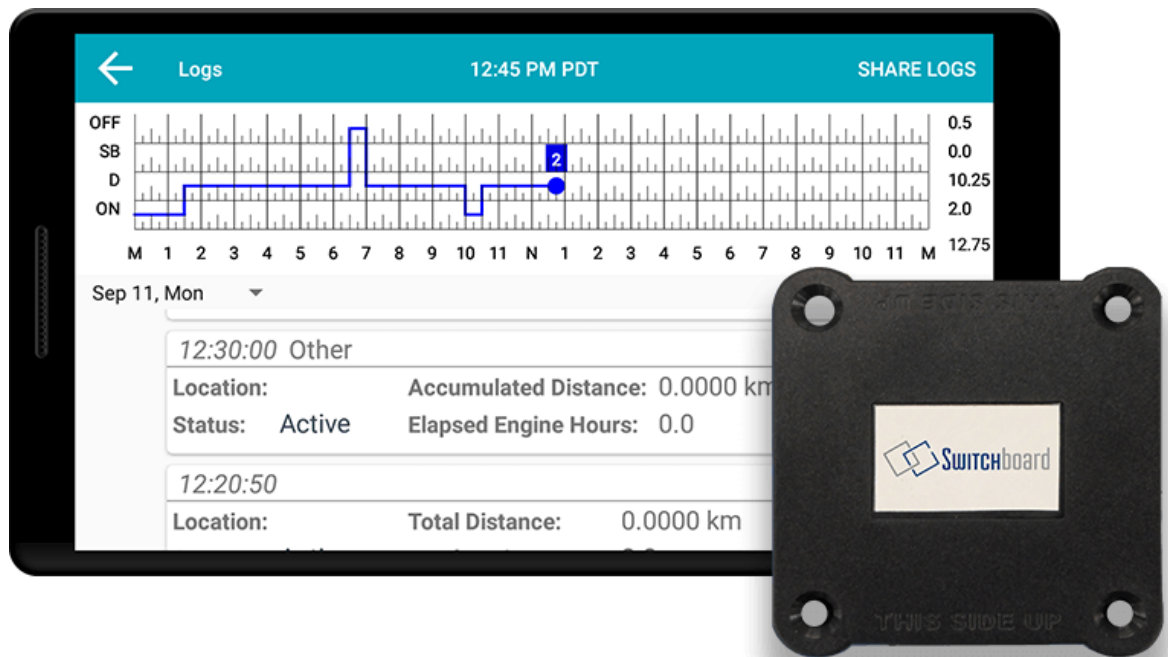


Рисунок 1.7 – Програмне забезпечення Switchboard [11]

Програмне забезпечення Track-POD це рішення для керування вантажівками, яке оптимізує та оцифровує всю логістичну операцію (рис.1.8) [12].

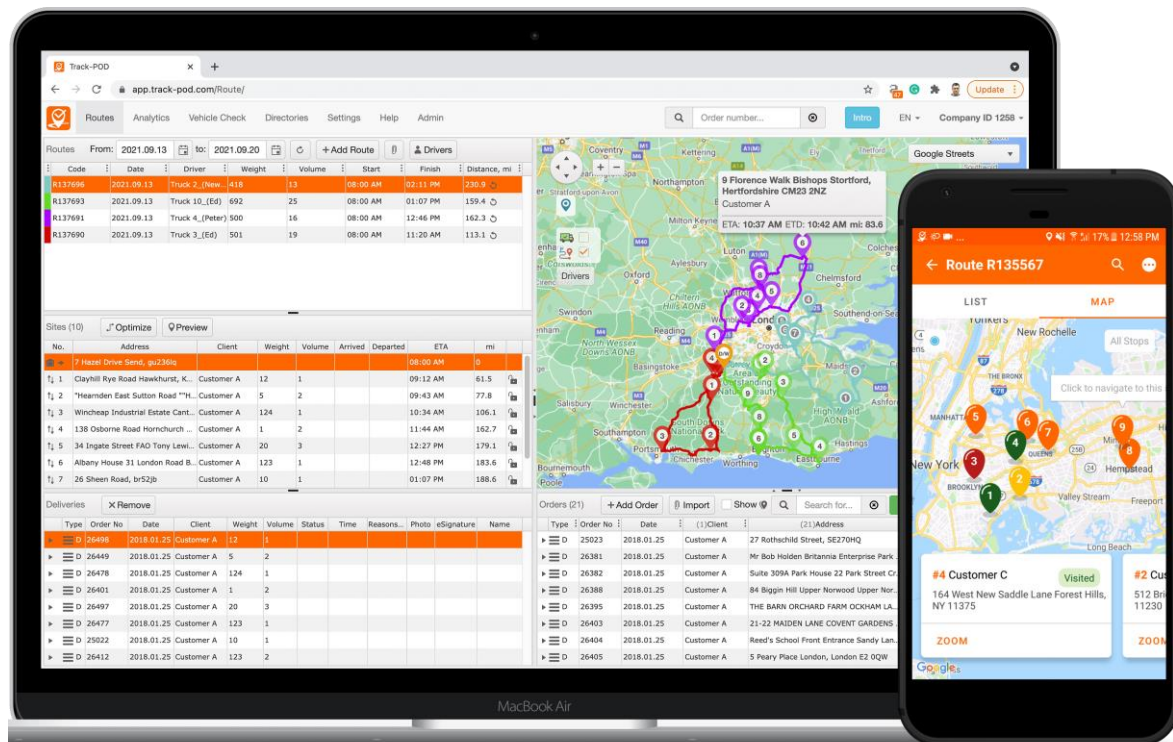


Рисунок 1.8 – Програмне забезпечення Track-POD [12]

Переваги програмного забезпечення Track-POD [12]:

- планування маршрутів на основі штучного інтелекту й оптимізації автопарку;
- відстеження водіїв у реальному часі та повністю безпаперового та безконтактного підтвердження доставки;
- керування операціями доставки на основі вантажівки будь-якого розміру.

Недоліки програмного забезпечення Track-POD [12]:

- клієнти не можуть залишати відгуки через додаток;
- проблеми з техпідтримкою.

Програмне забезпечення Broker PRO комплексне транспортне програмне забезпечення для вантажовідправників і вантажних брокерів для керування їх повсякденними операціями (рис. 1.9) [13].

The screenshot displays the Broker PRO software interface for a shipment from Decatur Tire Company. The top navigation bar includes links for Loads, Customers, Carriers, Rate Quotes, Accounting, Documents, and Help. The main header shows the load number #12413 and the carrier #4557.

Shipper Information:

- Company: YRC Trucking-Calgas
- Phone: [Enter phone number]
- Extension: [Enter extension]
- Location PoC: [Enter point of contact]
- Customer PoC: -- No Contact Selected --
- Address 1: 4559 Main St.
- Fax: [Enter fax number]
- Address 2: [Enter address]
- Quantity: [Enter quantity]
- Type: [Select Type]
- City: Dallas
- Weight: [Enter weight]
- State/Province: TX (Texas)
- Value: [Enter value]
- Zip Code: 75001
- Pickup Date: 08/19/2019
- Pickup #: [Enter pickup number]
- Pickup Time: [Enter time]
- Appointment: []
- Notes: [Enter shipper notes]

Carrier Information:

- Carrier: [Enter Carrier]
- Dispatcher: -- No Dispatcher Selected --
- Trailer #: [Enter carrier trailer number]
- Driver Cell #: [Enter carrier driver number]
- Driver Email: [Enter the driver's email address]

Carrier List:

- MARIA DELCARMEN GARCIA () Rating: 0/10
- Doug Andrus Distributing Lic (MC123310) Rating: 7/10
- D & G TRANSPORT INC () Rating: 10/10
- Bee Trucking, LLC (MC197008) Rating: 8/10
- STAR TRANSPORTATION COMPANY () Rating: 0/10
- OLD DOMINION (TX) () Rating: 0/10
- 1 St Choice Trucking Inc (MC569221) Rating: 9/10
- CMT Logistics, LLC (MC913143) Rating: 9/10
- MEGA FLEET CORP (MC499871) Rating: 8/10
- Advocate Us Africa Lic (MC910710) Rating: 8/10
- Hub Freight Usa Inc (MC758346) Rating: 8/10
- Denny Transport, Inc. (MC123133) Rating: 7/10
- 168 Trucking Inc (MC711278) Rating: 7/10

Cost Summary:

Cost/Unit	Suggested	Gross
\$ 1,250.00	\$ 324.50	\$1,250.00
Cost/Stop		
\$ 0.00		\$0.00
FS Rate	Suggested	
\$ 0.29		\$0.00
Total Costs		\$1,250.00
Discount		\$0.00
Total		\$1,250.00
Driver Advance		\$ 0.00
Balance Due		\$1,250.00

Рисунок 1.9 – Програмне забезпечення Broker PRO [13]

Переваги програмного забезпечення Broker PRO [13]:

- простий у використанні;
- зручна та швидка техпідтримка, гнучка система налаштування контрольних дзвінків та статусу;
- зручні функції для водіїв;
- наявність масової розсилки електронних листів.

Недоліки програмного забезпечення Broker PRO [13]:

- баги у системі пошуку вантажу;
- нав'язування клієнтам непотрібних функцій.

Програма 1С: Підприємство 8 використовується для автоматизації обліку на АТП та організаціях, а також в автотранспортних підрозділах самих різних підприємств (рис. 1.10) [14].

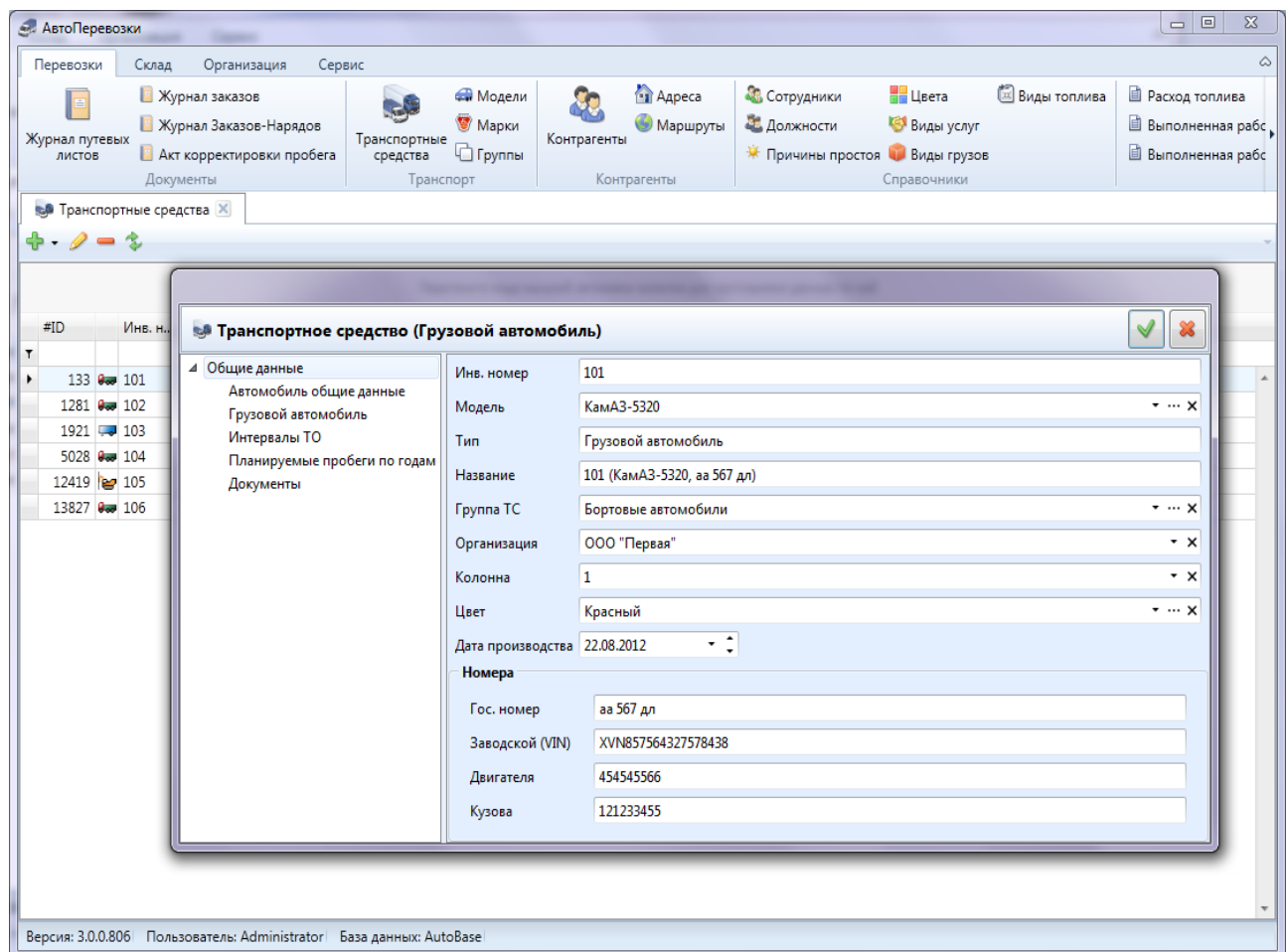


Рисунок 1.10 – Програмне забезпечення 1С: Підприємство 8 [14]

Дане ПЗ є досить зручне, перевагами цього ПЗ є такі [14]:

- довідник моделей транспортних засобів(більше ніж 500 моделей);
- звіт з технічного обслуговування та ремонту транспортних засобів;
- звіт наданих послуг та виконаних робіт;
- настройка тарифів за автоперевезення;
- об'єднання інформації у бази даних.
- розподіл прав доступу до інформації для користувача на рівні
- об'єктів та обмеження можливостей роботи в програмі;
- ефективна праця и надійне збереження інформації при роботі сотень користувачів завдяки використанню платформи «1С:Підприємство 8.2»;
- інтеграція з іншими системами.

Використовуються й інші програми для автоматизації діяльності АТП, зокрема програмне забезпечення Транспортна компанія 4.5, яке представляє собою програму для ведення справ на підприємствах зв'язаних з грузоперевезеннями, автоперевезеннями.

Надає можливість вести перелік клінтів транспортної компанії, автоматизувати прийом и супровід замовлень клінтів, контролювати платежі та заборгованості, оформлювати та контролювати перевезення та багато іншого.

Серед особливостей можна відмітити: швидкий доступ до БД, моментальний пошук даних по введеним критеріям, можливість мережевого доступу.

Процес автоматизації включає реєстрацію всіх необхідних даних про кожне перевезення, контроль виконання перевезень, формування та роздрук документів, перегляд звітів.

Таким чином, у наш час на програмному ринку широко представлені численні програми для роботи у транспортних організаціях, сервісних центрах, автомагазинах тощо. Порівняльний аналіз поширеного програмного забезпечення для АТП наведено у таблиці 1.1.

Таблиця 1.1 – Програмне забезпечення для обліку роботи автотранспортного підприємства

ПЗ	Розміщення	Виписка рахунків	Амортизація	Облік палива	Генератор звітів	GPS
Omnitracs	Windows/Linux Android/IOS	€	€	€	-	€
GPS Trackit	Cloud/SaaS/ Web-Based Android/IOS	€	-	€	-	€
EROAD	Cloud/SaaS/ Web-Based Android/IOS	€	€	-	€	-
Tailwind TMS	Cloud/SaaS/ Web-Based Windows/Linux / Mac Android/IOS	€	€	-	€	€
Fullbay	Cloud/SaaS/ Web-Based	€	€	€	€	-
Switchboard	Cloud/SaaS/ Web-Based Windows/Linux / Mac Android/IOS	€	-	€	€	€
Track-POD	Cloud/SaaS/ Web-Based Windows/Linux / Mac Android/IOS	-	-	€	€	-
Broker PRO	Cloud/SaaS/ Web-Based Windows/Linux / Mac Android/IOS	€	€	€	-	€

Таким чином, наявність недоліків існуючого програмного забезпечення обліку роботи автотранспортного підприємства обумовлює доцільність розробки ПЗ, більш зручного для використання. Пропонується, щоб розроблювана програма дозволяла виконувати такі функції:

- надавати можливість вносити та видаляти дані про водіїв та їхні автівки;
- перегляд рейсів;
- перегляд доступних автомобілей;
- підрахунок прибутку компанії;
- надавати можливість клієнтам залишати замовлення.

1.5 Висновки за розділом 1

Проаналізовано основні поняття про діяльність автотранспортних підприємств. Визначено, що автотранспортне підприємство являє собою вид комплексного транспортного підприємства, що займається перевезенням вантажів і пасажирів, зберіганням, ремонтом і обслуговуванням рухомого складу, постачанням витратних матеріалів та запасних частин. Проаналізовано інформаційні потоки та облікову інформацію, що використовується при автоматизації діяльності АТП.

Досліджено та проаналізовано програмне забезпечення автотранспортних підприємств обліку діяльності автотранспортних підприємств.

Визначено, що наявність недоліків існуючого програмного забезпечення обліку роботи автотранспортного підприємства обумовлює доцільність розробки ПЗ, більш зручного для використання.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування та інших технологій розробки

Програмне забезпечення обліку роботи автотранспортного підприємства доцільно розробити за допомогою мови програмування Java [15]-[19].

Створення мови Java є одним з найважливіших кроків у розвитку середовища програмування за останні роки. Java була потрібна, щоб зробити якісний стрибок у створенні інтерактивних продуктів для Інтернету [15]-[19]. У технологію мови Java інтегровано три ключові елементи, що відрізняє її від усіх існуючих мов програмування [15]-[19].

Java надає власні реалізації для невеликих, надійних, динамічних програм, які не залежать від комп'ютерної платформи, і активних мережеских програм, вбудованих у веб-сайт [15]-[19]. Змінювати та розгортати програми Java так само легко, як і будь-який HTML-документ. Java поєднує простий і знайомий синтаксис з надійним, зручним середовищем розробки, яке розкриває силу об'єктно-орієнтованої розробки додатків [15]-[19]. Це дозволяє широкому колу програмістів швидко створювати нові програми та нові програми [15]-[19]. Java надає багатий набір об'єктів для явного абстрагування багатьох системних функцій, які програміст використовує для роботи з вікнами, мережами, входами та входами [15]-[19]. Ключовою особливістю цих класів є те, що вони забезпечують широкий спектр незалежних від платформи системних інтерфейсів [15]-[19]. Синтаксис мови в основному походить від C і C++. На офіційній практиці програми Java компілюються в байт-коди, які інтерпретуються віртуальною машиною для даної платформи. Oracle надає компілятори Java та віртуальні машини Java, які відповідають специфікаціям JavaCommunityProcess за ліцензією GNUGeneralPublicLicense [15]-[19].

Ця мова запозичила багато синтаксису з C і C++ [15]-[19]. Зокрема, він заснований на об'єктній моделі C++, але змінений [15]-[19]. Це виключає можливість виникнення деяких конфліктних ситуацій, які можуть бути

викликані помилками програміста, і спрощує процес розробки об'єктно-орієнтованої програми [15]-[19]. Ряд операцій, які програмістам необхідно виконати на C/C++, довірено віртуальній машині. Перш за все, Java була розроблена як незалежна від платформи мова, тому для низькорівневого обладнання мало можливостей. За потреби java дозволяє завантажувати підпрограми, написані іншими мовами програмування [15]-[19]. Java вплинула на розвиток Microsoft J++ [15]-[19]. Робота над J++ була припинена через заяву SunMicrosystems про те, що мова програмування була змінена на Java. Потім на новій платформі Microsoft випустила версію J#, щоб полегшити міграцію програмістів .NET, J++ або Java на нову платформу [15]-[19]. З часом нова мова програмування C# захопила більшість Java і стала основною мовою платформи. J# востаннє було включено в Microsoft VisualStudio 2005. Мова сценаріїв JavaScript схожа за назвою та синтаксисом до Java, але не пов'язана з Java [15]-[19].

Одним із фундаментальних принципів розробки Java був захист від несанкціонованого доступу [15]-[19]. Програми Java викликають глобальні функції і не мають доступу до жодних системних ресурсів; це забезпечує рівень безпеки, якого Java не має в інших мовах. Для програм Java цей рівень безпеки забезпечує віртуальна машина Java, встановлена в операційній системі [15]-[19]. Хоча об'єктні моделі в Java прості й легко масштабуються, звичайні типи даних Java не є об'єктами, що підвищують продуктивність [15]-[19]. WorldWideWeb висунув Java на передній план програмування, а Java різко змінила обличчя Інтернету і навіть розширила діапазон об'єктів, які можна було розгорнути у кіберпросторі [15]-[19]. Новіші програми - програми завантажуються з віддаленого сервера і можуть працювати динамічно, тобто без втручання користувача [15]-[19]. До Java цей метод не був прийнятним з точки зору безпеки та переносимості [15]-[19]. Архітектура програм має ряд штучних обмежень, які роблять їх повністю безпечними. Перш за все, Java є мовою опису, а область ресурсів програми Java називається віртуальною

машиною Java (VJM), яка може керувати роботою програми та захищати систему від несприятливих впливів. Крім того, мова Java має додаткові обмеження, які не дозволяють програмі бути «троянським конем» [15]-[19]. Зокрема, аплет Java не може отримати доступ до внутрішнього жорсткого диска. Під час такої спроби виникає аварійна ситуація [15]-[19].

Java успадкувала потужний механізм об'єктно-орієнтованого програмування від C++ [15]-[19]. Оскільки Java була розроблена як «задушлива» і не повинна була бути сумісною з попередніми версіями, розробники мали повну свободу думки [15]-[19]. В результаті виникає відкритий і прагматичний підхід до об'єкта. Вільно поєднуючи ідеї, реалізовані протягом останніх кількох десятиліть, Java зуміла знайти баланс між парадигмою «все є об'єктом» і прагматичним підходом [15]-[19].

Дизайн об'єкта Java простий і масштабований і значно прискорює обробку, оскільки зберігає цільові типи як необ'єктні дані [15]-[19]. У Java є базові класи, які містять основні зведення в реальному світі, які ваші інструменти повинні декодувати [15]-[19]. Популярність Java заснована на абстрактних класах, які роблять її незалежною від платформи мовою [15]-[19]. Насправді, багато архітектурних рішень, прийняті під час створення Java, були мотивовані бажанням забезпечити синтаксис, подібний до C і C++ [15]-[19]. Java використовує майже ті самі вимоги для оголошення змінних, передачі параметрів, декларацій та моніторингу потоків виконання коду [15]-[19]. Java додає всі хороші можливості C++, але усуває останні недоліки [15]-[19]. Вказівники або теги в пам'яті є найпотужнішими та найнебезпечнішими функціями C++. Більшість сучасних помилок коду виникають через несправний покажчик [15]-[19]. Наприклад, одна з найпоширеніших помилок — обчислити розмір масиву в одиницях і розбити вміст осередків пам'яті, що знаходяться за ним [15]-[19].

Хоча ідентифікатори об'єктів Java реалізовані як покажчики, вони не можуть працювати з ними безпосередньо [15]-[19]. Ви не можете перетворити ціле число в покажчик або отримати доступ до адреси пам'яті [15]-[19]. Однак ви можете створювати не лише програми на Java, а й консольні програми, програми GUI, сервлети та JSP [15]-[19]. Незабаром після появи технології сервлетів розробники зіткнулися з такою проблемою: щоб динамічно згенерувати HTML-сторінку за допомогою сервлета, потрібно вбудувати HTML-код у сам сервер [15]-[19]. У цьому випадку HTML-код сторінки змішується з Java-кодом (логіка програми змішується з зовнішнім виглядом веб-сторінки), що ще більше ускладнює роботу програміста та веб-дизайнера. Для вирішення цієї проблеми була розроблена технологія JavaServerPages (JSP). Це дозволяє вам вставляти код Java в HTML-код веб-сайту [15]-[19]. Коли ви вперше переходите на сторінку jsp, її код автоматично перетворюється на сервлет. Після цього під час наступного звернення веб-сервер викликає перетворений сервлет, а не сторінку jsp [15]-[19]. Під час внесення змін до сторінки jsp веб-сервер виявляє, що сторінка змінилася, і знову оновлює відповідний сервлет. Сервлети і JSP поєднують кілька інших технологій Java, і цей комплекс називається Java 2 EnterpriseEdition (J2EE). Таким чином, сервери додатків Java надходили від різних компаній, таких як Sun (IBM, BEA, IONA, Borland). Кожна компанія впроваджує технологію J2EE на власний сервер додатків, але всі вони відповідають специфікаціям Sun. Раніше Sun пропонувала безкоштовний стандартний сервер для програмного забезпечення Javaweb під назвою JServ. Після випуску технології J2EE весь код було передано в ApacheSoftwareFoundation, а назву продукту було змінено на Tomcat. Tomcat наразі є стандартною реалізацією сервера Javaweb і входить до групи проекту Apache під назвою Jakarta [15]-[19].

Багатоплатформний характер веб-середовища висуває дуже високі вимоги до надійності додатків [15]-[19]. Як результат, розвиток Java став пріоритетом можливості створювати відмовостійкі програми [15]-[19]. Java

звільняє програмістів від занепокоєння щодо багатьох поширених причин програмних помилок. Як згадувалося раніше, Java є строго типізованою мовою програмування [15]-[19]. Також за «збір сміття» відповідає система виконання Java, яка автоматично звільняє динамічно виділену пам'ять. Звичайно, це знижує ефективність коду, але дозволяє уникнути поширених помилок, якщо програміст забуває звільнити виділену пам'ять або, навпаки, звільняє використану пам'ять [15]-[19]. Java підтримує об'єктно-орієнтовану обробку винятків, як C++. Однак, на відміну від Java C++, обробка винятків є обов'язковою [15]-[19]. Іншими словами, неможливо скомпілювати програму, яка відкриває файл, не розглянувши можливу помилку «файл не знайдено». Добре написана програма Java може сама виправити всі запуснені помилки [15]-[19].

Java була розроблена для задоволення потреб створення інтерактивних мережеских додатків. Для цього Java підтримує багатозадачне програмування, що дозволяє легко розробляти програми, які запускають кілька процесів одночасно [15]-[19]. Продуктивність програми Java елегантна, але в той же час вона базується на високоорганізованому рішенні синхронізації багатьох процесів, що дозволяє створити високоефективну інтерактивну систему. Java дозволяє писати код для одночасного виконання різних функцій, і не забувайте стежити за тим, що і коли відбувається [15]-[19]. Java використовує найелегантніший метод, який коли-небудь був розроблений, щоб вирішити проблему синхронізації процесів, що дозволяє створити красиву інтерактивну систему [15]-[19]. Прості у використанні дочірні процеси Java дозволяють застосовувати конкретну поведінку до програми, не турбуючись про створення загального циклу [15]-[19].

У таблиці 2.1 наведено результати аналізу різних мов програмування для розробки програмного забезпечення обліку роботи автотранспортного підприємства.

Таблиця 2.1 – Порівняння мов програмування для програмної реалізації обліку роботи автотранспортного підприємства

Критерій порівняння мов програмування	Мова програмування		
	Java	C#	PHP
Зручність використання для створення застосунку обліку роботи автотранспортного підприємства	+	+—	+—
Наявність засобів для об'єктно-орієнтованого програмування	+	+	+—
Засоби захисту інформації	+	+—	+—
Обробка виняткових ситуацій	+	+—	+—
Зручність роботи з базами даних	+	+	+—

Таким чином для програмної реалізації системи обліку роботи автотранспортного підприємства обрано мову програмування Java.

2.2 Вибір засобів роботи з базою даних

Як відзначено у першому розділі, у програмному забезпеченні обліку роботи автотранспортного підприємства необхідно забезпечити можливість обліку інформації про водіїв та їхні автівки, а також збереження та обробку іншої інформації. Тому при створенні програмного забезпечення необхідно використовувати засоби роботи з базою даних.

Як засіб для створення та роботи з базою даних про діяльність АТП доцільно обрати Cache [20]-[24].

InterSystems Cache — це високопродуктивна база даних, яка підтримує транзакційні програми по всьому світу [20]-[24]. Вона використовується для картографування Чумацького Шляху з мільярдами зірок, обробки мільярдів бізнес-транзакцій щодня та керування інтелектуальними мережами [20]-[24].

InterSystems Cache надає клієнтам необхідні програми, дозволяючи їм зберігати, використовувати та одночасно аналізувати транзакційні та історичні дані в будь-якій формі [20]-[24]. Високошвидкісний SQL однаковий для всіх моделей даних. Кеш забезпечує стабільно високу продуктивність в умовах постійно зростаючих вимог до пропускну здатності та даних. InterSystems Cache спрощує процес розробки та розгортання додатків [20]-[24]. Різноманітність високодоступних опцій, включаючи оптимальний метод дзеркального відображення, робить програми надійними та легкими для обладнання [20]-[24]. Гнучка та адаптивна модель безпеки забезпечує легкий доступ до управління доступом до ресурсів. Завдяки своїй простій архітектурі та надійному системному адмініструванню для запуску бази даних Cache потрібно менше адміністраторів, ніж інші рішення постачальника, а в деяких випадках не має окремого адміністратора [20]-[24]. Відкрите середовище для зміни бізнес-логіки та розробки інтерфейсів користувача для мобільних пристроїв і веб-браузерів може прискорити розробку та зробити інноваційні бізнес-процеси більш гнучкими [20]-[24]. Дані кешу можна моделювати та зберігати у вигляді таблиці, об'єкта або багатовимірного масиву (ієрархії). Доступ до даних будь-якої моделі здійснюється безперешкодно, і вони не повинні збігатися з моделлю, що часто погіршує продуктивність [20]-[24].

Вбудована підтримка динамічних об'єктів (таких як XML і JSON) у Cache забезпечує ефективне спілкування та прискорює розробку веб-додатків. SQL — це універсальна мова для пошуку даних у Cache, яка підтримується всіма моделями даних [20]-[24]. Завдяки оптимізованій архітектурі Cache SQL-запити виконуються набагато швидше, ніж інші технології баз даних [20]-[24]. Cache SQL підтримує стандартні, бітові та бітові індекси, які використовуються для даних транзакцій у реальному часі [20]-[24]. Протокол розподіленого кешу (ЕСР) значно покращує масштабованість і продуктивність розподілених програм, дозволяючи керувати кількома мережами серверів як єдиним сховищем даних [20]-[24]. Технологія дзеркального відображення бази даних кешу не вимагає великих інвестицій в обладнання, послуги підтримки, ліцензії на операційну систему або сховище [20]-[24]. Крім того, дзеркала бази

даних Cache дуже прості у створенні та обслуговуванні, а це означає, що їх використання знижує адміністративні витрати. Надійно захищайте дані диска та оброблені дані без шкоди для продуктивності програми. InterSystems Atelier – це нове середовище розробки на базі платформи Eclipse, яке дозволяє модифікувати бізнес-логіку та розробляти інтерфейс користувача для мобільних пристроїв і веб-браузерів [20]-[24]. Це середовище забезпечує доступ до основних інструментів і технологій і прискорює розробку додатків [20]-[24].

У таблиці 2.2 наведено порівняння засобів роботи з базами даних для розробки програмного забезпечення обліку роботи автотранспортного підприємства.

Таблиця 2.2 – Порівняння засобів роботи з базами даних для розробки програмного забезпечення обліку роботи автотранспортного підприємства

Критерій порівняння засобів роботи з базами даних	Системи управління базами даних		
	InterSystems Cache	MS Access	MySQL
Наявність високопродуктивних засобів створення запитів	+	+—	+—
Зручність для створення ПЗ обліку роботи автотранспортного підприємства	+	+—	+—
Робота з розподіленими системами	+	+—	+
Зручність обслуговування дзеркал баз даних	+	+—	+
Надійність захисту інформації	+	+—	+

Таким чином для розробки програмного забезпечення обліку роботи автотранспортного підприємства систему управління базами даних InterSystems Cache.

2.3 Висновки за розділом 2

Для створення програмного забезпечення обліку роботи автотранспортного підприємства було обрано мову програмування Java, яка є досить зручною для розробки програмних систем подібного виду та надає власні засоби для реалізації невеликих, надійних, динамічних програм, які не залежать від комп'ютерної платформи, і активних мережових програм, вбудованих у веб-сайт.

Як засіб для створення та роботи з базою даних про діяльність АТП доцільно обрати Cache, що дозволить забезпечити можливість обліку інформації про водіїв та їхні автівки, а також збереження та обробку іншої інформації.

3 ОПИС ПРОГРАМИ

3.1 Основні рішення щодо подання даних системи

Клас CacheDB відповідає за те, аби підключитися до бази даних та зчитати, додати, оновити, видалити дані.

Поля та методи класу CacheDB наведені у табл. 3.1.

Таблиця 3.1 – Поля та методи класу CacheDB

Поля та методи класу CacheDB	Опис
Connection myConn	З'єднання з БД
List<Transport> list	Список усіх вільних ТЗ
List<Worker> lstW	Список усіх вільних водіїв
List<Voyage>listV	Список усіх активних рейсів
Transport appropriTrans	Машина, яка відповідає умовам рейсу
CacheDB()	Конструктор класу CacheDB
void close(Statement myStmt, ResultSet myRs)	Замкнення з'єднання
void close(Connection myConn, Statement myStmt, ResultSet myRs)	Замкнення з'єднання
addTransport(String govNumber, String	Додати транспорт до БД
Voyage convertRowToVoyage(ResultSet rs)	Конвертує з ResultSet до Voyage
void deleteTransport(int id)	Видалити ТЗ з БД
void addWorker(String firstName, String name, String lastName, String _salary, String _experience)	Додати водія до БД
void deleteWorker(int id)	Видалити водія з БД
void searchFreeTransport(int capacity, int volume)	Пошук в БД вільних ТЗ

Продовження таблиці 3.1

Поля та методи класу CacheDB	Опис
Transport convertRowToTransport(ResultSet rs)	Конвертує з ResultSet до Transport
int getListSize()	Повертає розмір списку з автомобілями
void appropriateTransport()	Вибір з усіх ТЗ найбільш відповідного
void updateTransport()	Зміна стану ТЗ в БД
setVoyage(String _FIO, String _capacity, String _volume, String _from, String _to, String _distance, double _money)	Занести рейс о БД
void searchFreeWorker()	Пошук вільного водія в БД та занесення його до списку listW
Worker convertRowToWorker(ResultSet rs)	Конвертує з ResultSet до Worker
void updateWorker(int workerID)	Зміна стану працівника в БД
void GetActiveVoyage()	Додавання активних рейсів з БД до списку listV
List<Voyage> getList()	Повертає список рейсів
void changeVoyage(int id)	Зміна стану рейсу

3.2 Основні розроблені алгоритми

Основними алгоритмами роботи розробленого курсового проекту є алгоритми, що працюють з базою даних, як наприклад занесення, оновлення, даних до БД, видалення даних. Також в програмі реалізований алгоритм автоматичного підрахунку вартості рейсу враховуючи вартість автомобіля та витрату пального та алгоритм вибору найбільш оптимального виду транспорту для перевезення вантажу.

3.3 Основні рішення щодо модульного подання системи

Система розподілена на класи з даними і функціями, які використовуються цими даними та виконують свої задачі для роботи системи.

Проект включає в себе такі файли:

- CacheDB.java – файл, що містить оголошення класу CacheDB, який призначений для зв'язку з базою даних;
- Transport.java – файл, що зберігає структуру ТЗ;
- Worker.java – файл, що зберігає структуру водія;
- Voyage.java – файл, що зберігає структуру рейсу;
- AdminForm.java – файл, що призначений для відображення форми для адміна, за допомогою якої адмін може додавати автомобілі, водіїв та видаляти автомобілі та водіїв;
- ClientForm.java – файл, що створює графічний інтерфейс для клієнта, аби той вніс рейс до бази даних замовлень;
- ClientTransactionForm.java – файл, що призначений для відображення графічного інтерфейсу оплати рейсу;
- DispatcherForm.java – файл виводу даних про рейс та видалення рейсів, які завершилися;
- PreEnter.java – файл, що відображає форму для вибору перед входом до програми;
- WorkerForm.java – файл, що відображає форму входу для працівника підприємства;
- About.java – файл, що реалізує інтерфейс для виводу таблиці вільних автомобілей;
- Transport.cls – файл, що зберігає структуру об'єкта автомобіль у базі даних БД;
- Worker.cls – файл, що зберігає структуру об'єкта водій в БД;
- Voyage.cls – файл, що зберігає структуру об'єкта рейс в БД.

На рис. 3.1 наведено структуру модулів розробленого програмного забезпечення обліку роботи автотранспортного підприємства.

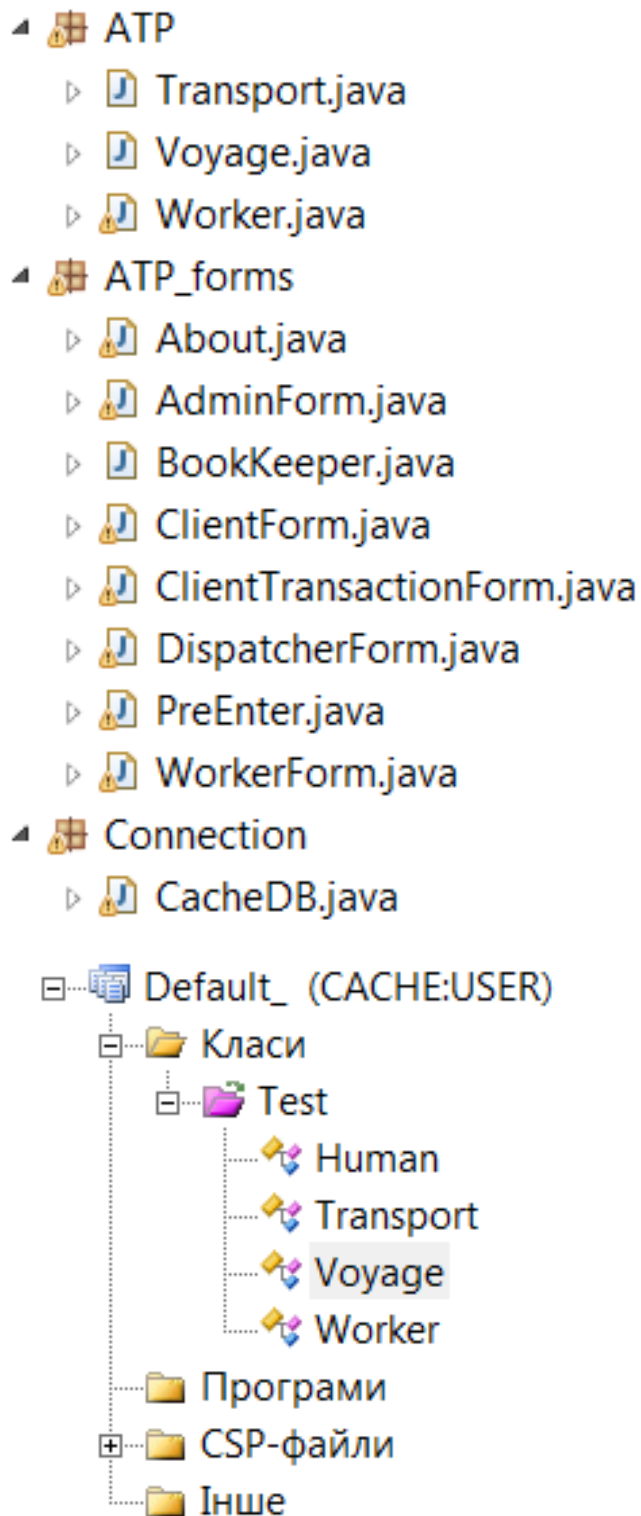


Рисунок 3.1 – Структура модулів програмного забезпечення обліку роботи автотранспортного підприємства

3.4 База даних програмного забезпечення обліку роботи автотранспортного підприємства

Для забезпечення можливості обліку інформації про водіїв та їхні автівки, а також збереження та обробку іншої інформації розроблено базу даних.

ER-діаграма БД зображена на рис 3.2.

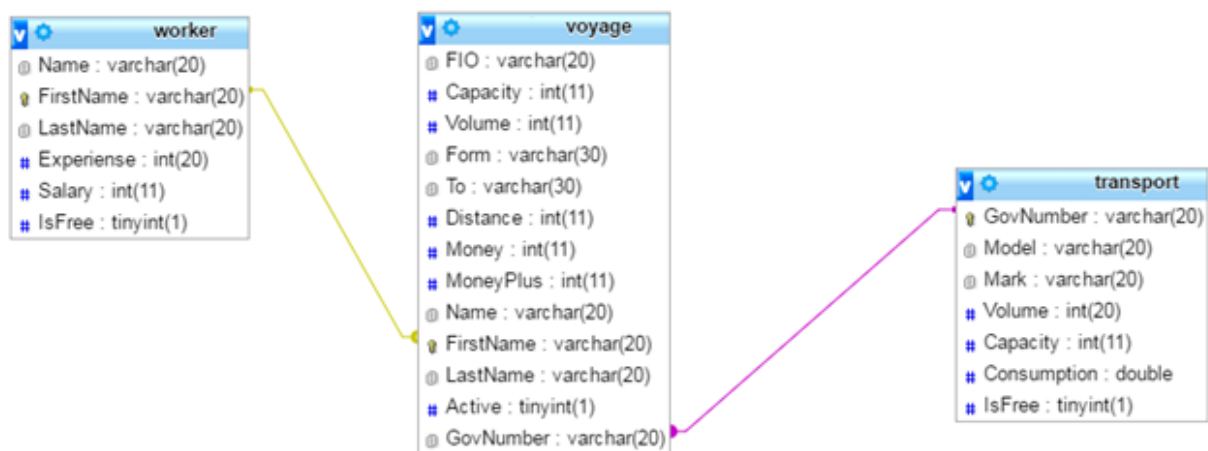


Рисунок 3.2 – ER-діаграма БД

Розроблений програмний продукт є клієнт-серверним програмним забезпеченням. Клієнтом виступає замовник або працівник автотранспортного підприємства, а саме: адміністратор або диспетчер. Сервер забезпечує взаємодію між клієнтом та базою даних за допомогою SQL-запитів, наприклад запит "update Test.Transport set IsFree = ? where ID=?" оновлює стан певного автомобілю в БД.

3.5 Опис інтерфейсу користувача програмного забезпечення обліку роботи автотранспортного підприємства

Інтерфейс користувача програмного забезпечення, розробленого з бізнес-розрахунками автомобільного бізнесу, являє собою елементи та компоненти, які можуть сприймати ефект взаємодії клієнта з програмним забезпеченням електронного магазину. Такий інструмент користувацького інтерфейсу засобу відображення інформації використовується для відображення інформації, формату та коду в програмі розрахунку роботи, розробленій судноплавною компанією; командний режим і мова інтерфейсу користувача; обладнання та технології введення даних; розмови, взаємодії та транзакції між користувачами та комп'ютерами; відгуки клієнтів; підтримка рішень з конкретних питань; інструкції з використання програми [7] - [21].

При розробці інтерфейсу користувача бухгалтерського програмного забезпечення враховується принцип «чим простіше, тим краще», екран має бути просто інформативним, корисним для користувачів, текст, контактна інформація та шрифти повинні бути простими. ти читаєш. Графічні елементи чіткі, вражаючі та швидко завантажуються [11-12]. Коли людське око сканує сторінку зверху вниз, ключ знаходиться вгорі ліворуч. Тому найважливіша інформація в цьому місті включена в розроблену програму [7] - [21]. Для полегшення роботи користувача програмне забезпечення обліку інструментів не використовує велику кількість авангардних і складних композицій над екраном. Він використовує компоненти інтерфейсу, які дозволяють клієнтам легко знаходити потрібну інформацію та полегшують пошук інформації про машину або працівника, якого вони хочуть [7] - [21]. Дизайн програмного забезпечення бухгалтерського обліку автомобільної компанії виконано в ніжних тонах.

На рис. 3.3 зображено приклад екранної форми програми, де відображено повідомлення користувачу.

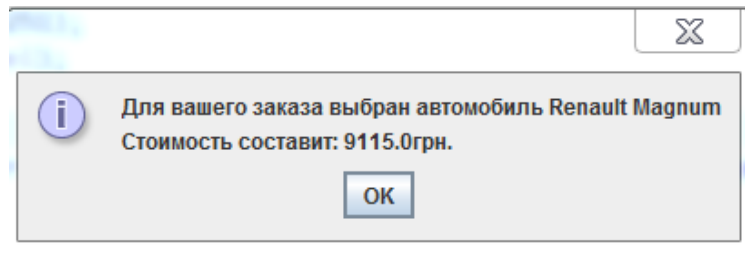


Рисунок 3.3 – Повідомлення користувачу

3.6 Висновки за розділом 3

В цьому розділі описано основні рішення щодо подання даних у програмному забезпеченні обліку роботи автотранспортного підприємства. Запропоновано структуру класів програмного забезпечення обліку роботи автотранспортного підприємства. Розроблено та наведено схему бази даних програми.

Надано опис інтерфейсу користувача програмного забезпечення обліку роботи автотранспортного підприємства. При розробці інтерфейсу програми були використані такі візуальні компоненти Swing: JTable, JLabel, JTextField, JButton, JPasswordField.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програмний продукт призначений для надання клієнту можливості замовити вантажівку та працівникам відстежувати грузи та зберігати дані про працівників та автомобілі. Програма призначена для того, щоб експлуатуватися на комп'ютерах на АТП.

Програмне забезпечення обліку роботи автотранспортного підприємства забезпечує виконання функцій:

- надання можливості клієнту робити замовлення;
- надання можливості диспетчеру відстежувати рейси, прибуток та завершувати рейси;
- авторизація зареєстрованих користувачів;
- можливість пошуку автівок за критеріями;
- надання можливості адміну додавати, видаляти співробітників та автомобілі;
- оформлення автівок для здійснення рейсів.

Для експлуатації програмного забезпечення обліку роботи автотранспортного підприємства необхідно таке апаратне забезпечення:

- процесор 2.2 ГГц або вище;
- оперативна пам'ять 2048 МБ;
- наявність вільного місця на жорсткому диску не менше 400 мегабайт (МБ);
- монітор для взаємодії з користувачами.

Також повинна бути у наявності операційна система сімейства Windows.

Функціональні характеристики розробленого програмного забезпечення обліку роботи автотранспортного підприємства відповідають технічному завданню, наведеному у додатку А. Опис програми обліку роботи

автотранспортного підприємства та текст програми обліку роботи автотранспортного підприємства в наведені у додатках Б та В.

4.2 Характеристики програми

Програма обліку роботи автотранспортного підприємства розроблена за допомогою мови програмування Java. Для забезпечення можливостей роботи з даними автотранспортного підприємства, зокрема функції створення, редагування та видалення інформації про наявні автотранспортні засоби тощо, використовуються засоби системи керування базами даних Cache.

Час роботи програми обліку роботи автотранспортного підприємства залежить від обсягу бази даних про автомобілі та іншої інформації АТП, конфігурації комп'ютера.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програми обліку роботи автотранспортного підприємства необхідно встановити JRE та запустити файл програми з розширенням .jar. Після запуску програми обліку роботи автотранспортного підприємства на екрані з'явиться форма входу в програму. Завершення роботи з програмою обліку роботи автотранспортного підприємства відбувається після закриття усіх активних вікон.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програми обліку роботи автотранспортного підприємства є дані, які вводить користувач, коли робить замовлення, а саме: маса вантажу, об'єм вантажу, адреса відправлення, адреса відвантаження, відстань.

Вихідними даними програми обліку роботи автотранспортного підприємства є інформація про автомобіль, який відповідає заданим вимогам клієнта та вартість замовлення.

4.3.3 Повідомлення

Розроблена програма обліку роботи автотранспортного підприємства забезпечує функціонування у різних операційних системах. У випадку виникнення виняткових ситуацій, наприклад, коли користувач програми обліку роботи автотранспортного підприємства задає замість чисельних вхідних даних текстові, програма видає відповідні повідомлення про необхідність введення даних у коректній формі.

4.4 Тестування та результати роботи програмного забезпечення обліку роботи автотранспортного підприємства

Для запуску програми необхідно встановити JRE, після чого запустити відповідний файл АТР.jar на виконання.

Після цього з'явиться головна форма програми (рис. 4.1). Для того аби почати користування програмою необхідно вибрати тип користувача: «Сотрудник» чи «Клієнт».

Після натискання на кнопку «Сотрудник» перед користувачем з'явиться нова форма, де можна увійти як співробітник. Для цього необхідно ввести логін та пароль. Після того, як користувач зайшов у програми, відкривається форма залежно від статусу користувача (рис. 4.2).



Рисунок 4.1 – Форма входу у програму

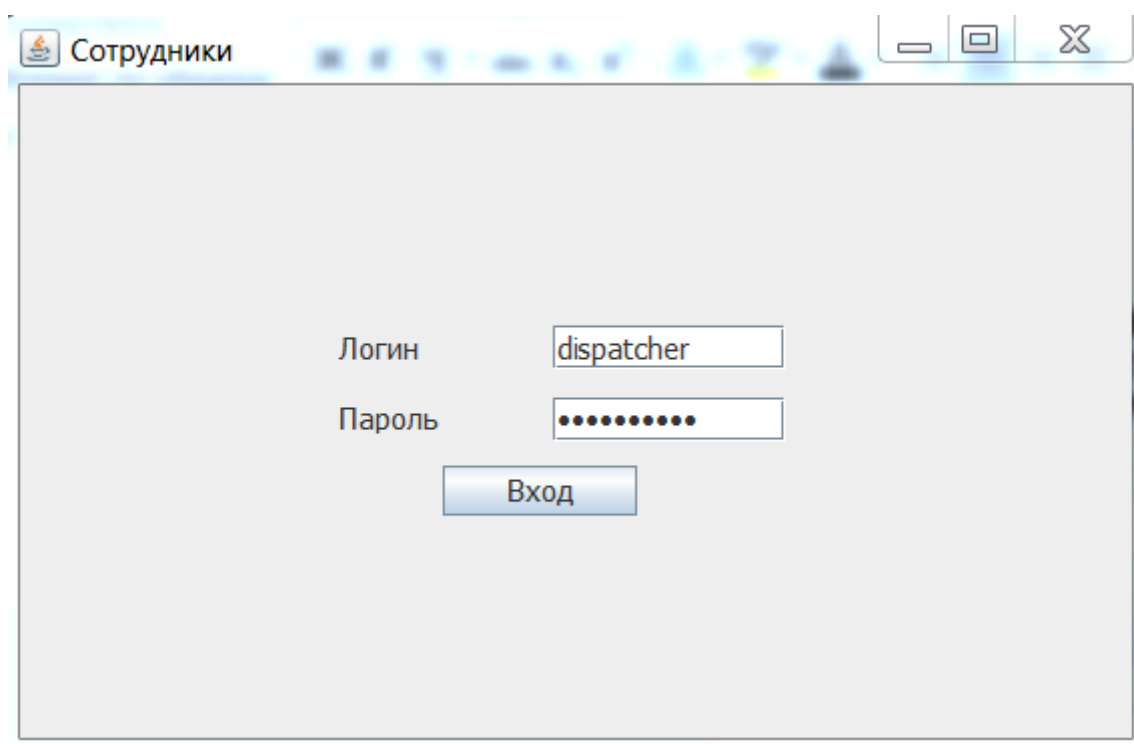


Рисунок 4.2 – Вікно входу для співробітника

Якщо користувач увійшов у програму як диспетчер (пароль: “dispatcher”), то з'являється форма, де є можливість продивитися усі рейси та завершити ті, які вже завершилися (рис. 4.3).

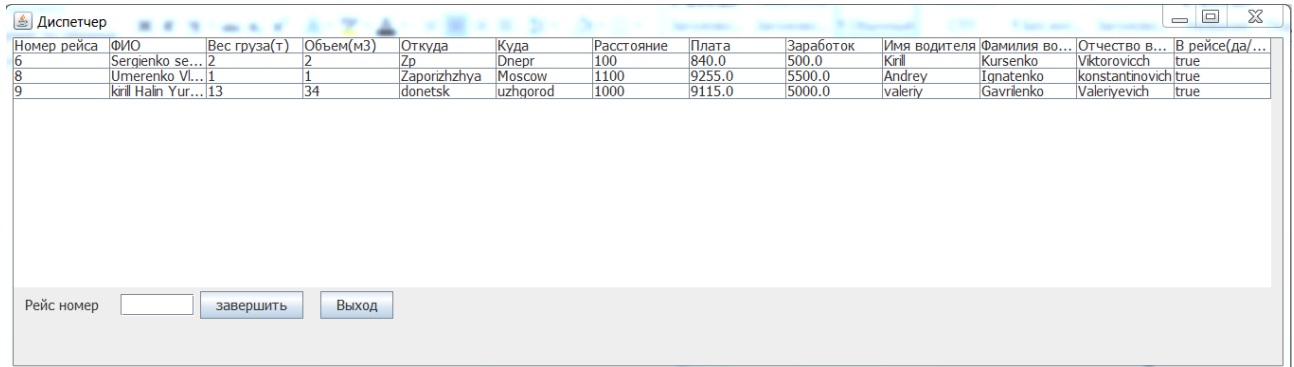


Рисунок 4.3 – Вікно для диспетчера

Якщо користувач увійшов у програму як адмін (пароль: “admin”), то відкривається форма, де адміністратор може додати, видалити автомобілі або водіїв (рис. 4.4, рис. 4.5).

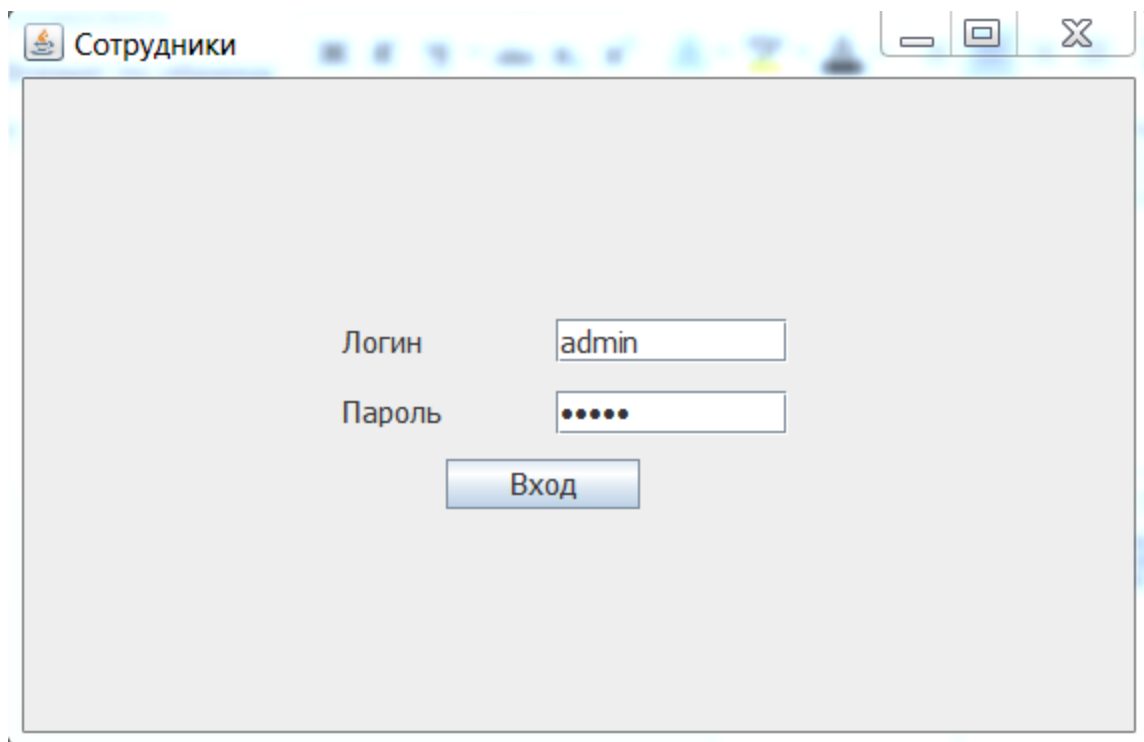


Рисунок 4.4 – Вікно входу для адміністратора

The screenshot shows a window titled 'Админ' (Admin) with a standard Windows-style title bar. The window contains three main sections:

- Добавить машину** (Add vehicle): A form with input fields for 'Гос. номер' (State number), 'Грузоподъемность' (Load capacity) with a unit 'т.' (tons), 'Объем' (Volume) with a unit 'м3', 'Марка' (Brand), 'Модель' (Model), 'Стоимость' (Cost) with a unit '\$', and 'Расход' (Consumption) with a unit 'л/100км.' (liters/100km). Below the fields is a button labeled 'Добавить машину'.
- Добавить работника** (Add employee): A form with input fields for 'Фамилия' (Surname), 'Имя' (Name), 'Отчество' (Patronymic), 'Стаж' (Experience), and 'Оклад' (Salary) with a unit 'грн/км' (hryvnia/km). Below the fields is a button labeled 'Добавить работника'.
- Удалить машину из базы данных** (Delete vehicle from database): A form with an input field and a button labeled 'Удалить'.
- Удалить работника из базы данных** (Delete employee from database): A form with an input field and a button labeled 'Удалить'.
- Освободить автомобиль** (Release vehicle): A form with an input field and a button labeled 'Исполнить'.

At the bottom right of the window is a button labeled 'Выход' (Exit).

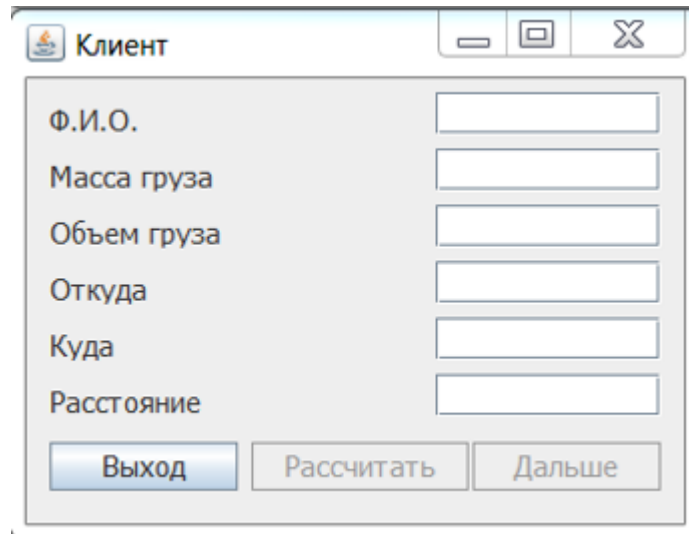
Рисунок 4.5 – Вікно адміністратора

Після натискання кнопки «Клиент», користувач може бачити дві форми. На першій (рис. 4.6) відображено таблицю, де виводяться вільні автомобілі, на другій (рис. 4.7) – поля, які необхідно заповнити, щоб замовити рейс. Якщо умови, що надаються автотранспортним підприємством, задовольняють користувача, то він може продовжити та «оплатити» рейс (рис. 4.8).



ID	Гос.номер	Объем	Грузоподъемн...	Марка	Модель	Свободен(да/...
3	213f	26	10	Maz	223	true
4	qw1451r	40	25	KAMAZ	356	true
5	rty442	120	22	Scania	Superspace	true
6	4547hq	26	3	Foton	334	true
7	4554t	38	7	Mercedes	Atego 815	true
8	348re	69	20	Mercedes	Actros	true
9	774t	79	20	DAF	335	true
11	3325y	30	30	Ural	420	true

Рисунок 4.6 – Таблица доступних автомобілів



Клиент

Ф.И.О.

Масса груза

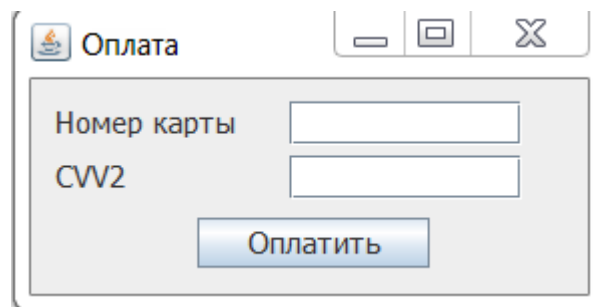
Объем груза

Откуда

Куда

Расстояние

Рисунок 4.7 – Вікно клієнта



Оплата

Номер карты

CVV2

Рисунок 4.8 – Вікно для сплати рейсу

Для завершения выполнения программы на будь-якому етапі користувачу необхідно натиснути червоний хрестик в правому верхньому куту екрана.

У ході роботи з програмою користувач отримує повідомлення перед оформленням рейсу, на якому виводиться вартість замовлення та автомобіль. Приклади повідомлення на рис. 4.9.

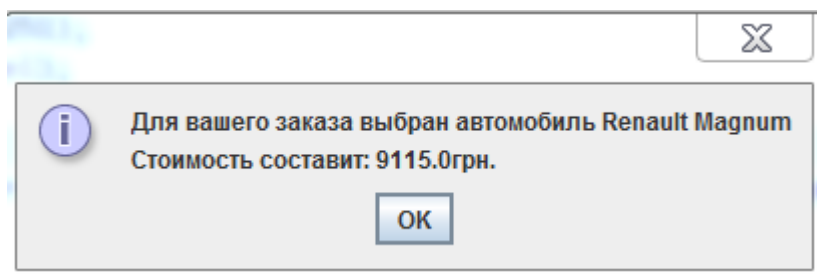


Рисунок 4.9 – Приклад повідомлення користувачу

Таким чином, за результатами тестування виявлено, що розроблене програмне забезпечення обліку роботи автотранспортного підприємства відповідає технічному завданню.

4.5 Висновки за розділом 4

Описано та досліджено програмне забезпечення обліку роботи автотранспортного підприємства. Для оцінки якості роботи програмного забезпечення обліку роботи автотранспортного підприємства було проведено його тестування. За результатами тестування виявлено, що розроблене програмне забезпечення обліку роботи автотранспортного підприємства відповідає технічному завданню.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для автотранспортних підприємств.

Проаналізовано основні поняття про діяльність автотранспортних підприємств. Визначено, що автотранспортне підприємство являє собою вид комплексного транспортного підприємства, що займається перевезенням вантажів і пасажирів, зберіганням, ремонтом і обслуговуванням рухомого складу, постачанням витратних матеріалів та запасних частин. Проаналізовано інформаційні потоки та облікову інформацію, що використовується при автоматизації діяльності АТП.

Досліджено та проаналізовано програмне забезпечення автотранспортних підприємств обліку діяльності автотранспортних підприємств. Визначено, що наявність недоліків існуючого програмного забезпечення обліку роботи автотранспортного підприємства обумовлює доцільність розробки ПЗ, більш зручного для використання.

Для створення програмного забезпечення обліку роботи автотранспортного підприємства було обрано мову програмування Java, яка є досить зручною для розробки програмних систем подібного виду та надає власні засоби для реалізації невеликих, надійних, динамічних програм, які не залежать від комп'ютерної платформи, і активних мережевих програм, вбудованих у веб-сайт.

Як засіб для створення та роботи з базою даних про діяльність АТП обрано Cache, що дозволяє забезпечити можливість обліку інформації про водіїв та їхні автівки, а також збереження та обробку іншої інформації.

Описано основні рішення щодо подання даних у програмному забезпеченні обліку роботи автотранспортного підприємства. Запропоновано структуру класів програмного забезпечення обліку роботи автотранспортного підприємства. Розроблено та наведено схему бази даних програми.

Надано опис інтерфейсу користувача програмного забезпечення обліку роботи автотранспортного підприємства. При розробці інтерфейсу програми були використані різні візуальні компоненти.

Описано та досліджено програмне забезпечення обліку роботи автотранспортного підприємства. Для оцінки якості роботи програмного забезпечення обліку роботи автотранспортного підприємства було проведено його тестування. За результатами тестування виявлено, що розроблене програмне забезпечення обліку роботи автотранспортного підприємства відповідає технічному завданню.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гриньов А. В. Конкуренстоспроможність автотранспортного підприємства на ринку міжнародних вантажних перевезень / А. В. Гриньов, Н. В. Пономарьова, Н. М. Пономарьова. – Харків : ХНАДУ, 2009. – 116 с.
2. Редзюк А.М. Автомобільний транспорт України: стан, проблеми, перспективи розвитку / [А.М. Редзюк, П.П. Волков, С.Ю. Дмитренко та ін.]. – Київ : ДП "ДержавтотрансНДІпроект", 2005. – 400 с.
3. Бідняк М. Н. Виробничі системи на транспорті: теорія і практика / М. Н. Бідняк, В. В. Біліченко. – Вінниця : УНІВЕРСУМ, 2006. – 176 с.
4. Крикавський Є.В. Логістичне управління / Є.В. Крикавський. – Л.: Вид-во НУ «Львівська політехніка», 2005. – 683 с.
5. Гужва В.М. Інформаційні системи і технології на підприємствах: навч. посібник / В.М. Гужва. – К.: КНЕУ, 2014. – 400 с.
6. Omnitracs [Electronic resource]. – Access mode : <https://www.omnitracs.com/>.
7. Trackit [Electronic resource]. – Access mode : <https://gpstrackit.com/>.
8. EROAD [Electronic resource]. – Access mode : <https://www.eroad.com/>.
9. Tailwind TMS [Electronic resource]. – Access mode : <https://www.tailwindtransportationsoftware.com/>.
10. Fullbay [Electronic resource]. – Access mode : <https://www.fullbay.com/>.
11. Switchboard [Electronic resource]. – Access mode : <https://onswitchboard.com/>.
12. Track-POD [Electronic resource]. – Access mode : <https://www.track-pod.com/ru/>.
13. Broker PRO [Electronic resource]. – Access mode : <https://brokerpro.com/>.

14. 1C: Підприємство 8 [Електрон. ресурс]. – Режим доступу : <https://www.softcom.ua/ua/1c/v8/>.
15. java [Electronic resource]. – Access mode : <https://www.java.com/>.
16. Самоучитель по Java с нуля [Электрон. ресурс]. – Режим доступа: <https://vertex-academy.com/tutorials/ru/samouchitel-po-java-s-nulya/>.
17. Сравнение JavaScript фреймворков Vue.js, React и Angular [Электрон. ресурс]. – Режим доступа: <https://mkdev.me/posts/sravnenie-javascript-freymvorkov-vue-js-react-i-angular-2019>.
18. Этан Браун Изучаем JavaScript: руководство по созданию современных вебсайтов [Текст] / Этан Браун. – М.: Диалектика-Вильямс, 2016. – 368 с. – ISBN 978-5-9908463-9-5.
19. Блох Д. Java. Эффективное программирование / Д. Блох. – М.: Лори, 2020. – 464 с.
20. Гайдаржи В.І. Об'єктно-реляційна СУБД Cache. Багатовимірний сервер даних і способи реалізації бізнес логіки засобами вбудованої мови Cache Objectscript : навчальний посібник / В.І. Гайдаржи, І.Ю. Михайлова. – К. : Освіта України, 2015. – 312 с.
21. Михайлова І.Ю. Об'єктно-реляційна СКБД Cache / І.Ю. Михайлова, В.І. Гайдаржи. – К. : Освіта України, 2016 – 404 с.
22. Крёнке Д. Теория и практика построения баз данных / Д. Крёнке. – СПб. : Питер, 2003. – 800 с.
23. Дейт К. Дж. Введение в системы базы данных : пер. с англ. – 6-е изд. / К. Дж. Дейт. – К. : Диалектика, 1998. – 784 с.
24. Intersystems Cache [Electronic resource]. – Access mode : <https://www.intersystems.com>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення обліку роботи автотранспортного підприємства може використовуватися для вирішення задач, пов'язаних з діяльністю підприємств, що займаються діяльністю з перевезення вантажів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення обліку роботи автотранспортного підприємства», затверджене наказом Національного університету «Запорізька політехніка» № 102 від 27 квітня 2022 р.

A.2 Призначення розробки

Програмний продукт обліку роботи автотранспортного підприємства призначений для організації обліку діяльності підприємств, що займаються діяльністю з перевезення вантажів. Розроблена програма надає клієнту можливості замовити вантажівку та працівникам відстежувати грузи та зберігати дані про працівників та автомобілі.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення обліку роботи автотранспортного підприємства повинно забезпечувати виконання таких функцій:

- надання можливості клієнту робити замовлення;
- надання можливості диспетчеру відстежувати рейси, прибуток та завершувати рейси;
- авторизація зареєстрованих користувачів;

- можливість пошуку автівок за критеріями;
- надання можливості адміну додавати, видаляти співробітників та автомобілі;
- оформлення автівок для здійснення рейсів.

A.3.2 Вимоги до інтерфейсу програми

Програмне забезпечення обліку роботи автотранспортного підприємства повинно забезпечувати можливість відображення інформації за допомогою у зручному для користувача вигляді. Взаємодія користувача з програмою обліку роботи автотранспортного підприємства можлива за допомогою екранних форм.

A.3.3 Вимоги до надійності

Програмне забезпечення обліку роботи автотранспортного підприємства повинно обробляти виняткові ситуації, зокрема введення користувачем некоректних даних.

A.3.4 Умови експлуатації

Експлуатація програмного забезпечення обліку роботи автотранспортного підприємства повинна здійснюватися відповідно до документації на розроблене програмне забезпечення обліку роботи автотранспортного підприємства. Програмне забезпечення обліку роботи автотранспортного підприємства може бути записана як на носії інформації.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації Програмне забезпечення обліку роботи

автотранспортного підприємства необхідні такі засоби:

- процесор 2.2 ГГц або вище;
- оперативна пам'ять 2048 МБ;
- наявність вільного місця на жорсткому диску не менше 400 мегабайт (МБ);
- монітор для взаємодії з користувачами.

Також повинна бути у наявності операційна система сімейства Windows.

А.3.6 Вимоги до маркування і пакування

Програмне забезпечення обліку роботи автотранспортного підприємства може поставлятися на різних носіях. На упакуванні повинна бути назва програми «Програмне забезпечення обліку роботи автотранспортного підприємства».

А.4 Вхідні дані до роботи

Вхідними даними до програми обліку роботи автотранспортного підприємства є дані, які вводить користувач, коли робить замовлення, а саме: маса вантажу, об'єм вантажу, адреса відправлення, адреса відвантаження, відстань.

Вихідними даними програми обліку роботи автотранспортного підприємства є інформація про автомобіль, який відповідає заданим вимогам клієнта та вартість замовлення.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програмне забезпечення обліку роботи автотранспортного підприємства розроблено за допомогою мови програмування Java.

Б.2 Функціональне призначення

Програмний продукт призначений для надання клієнту можливості замовити вантажівку та працівникам відстежувати грузи та зберігати дані про працівників та автомобілі. Програма призначена для того, щоб експлуатуватися на комп'ютерах на АТП.

Програмне забезпечення обліку роботи автотранспортного підприємства забезпечує виконання функцій:

- надання можливості клієнту робити замовлення;
- надання можливості диспетчеру відстежувати рейси, прибуток та завершувати рейси;
- авторизація зареєстрованих користувачів;
- можливість пошуку автівок за критеріями;
- надання можливості адміну додавати, видаляти співробітників та автомобілі;
- оформлення автівок для здійснення рейсів.

Б.3 Опис логічної структури

Програмне забезпечення обліку роботи автотранспортного підприємства представляє застосунок, що складається з різних компонентів та програмних файлів-модулів.

Для реалізації програмного забезпечення обліку роботи автотранспортного підприємства було створено програмні модулі.

Б.4 Використані технічні засоби

Для експлуатації програмного забезпечення обліку роботи автотранспортного підприємства необхідно таке апаратне забезпечення:

- процесор 2.2 ГГц або вище;
- оперативна пам'ять 2048 МБ;
- наявність вільного місця на жорсткому диску не менше 400 мегабайт (МБ);
- монітор для взаємодії з користувачами.

Б.5 Виклик і завантаження

Для запуску програми обліку роботи автотранспортного підприємства необхідно встановити JRE та запустити файл програми з розширенням .jar. Після запуску програми обліку роботи автотранспортного підприємства на екрані з'явиться форма входу в програму. Завершення роботи з програмою обліку роботи автотранспортного підприємства відбувається після закриття усіх активних вікон.

Б.6 Вхідні та вихідні дані

Вхідними даними до програми обліку роботи автотранспортного підприємства є дані, які вводить користувач, коли робить замовлення, а саме: маса вантажу, об'єм вантажу, адреса відправлення, адреса відвантаження, відстань.

Вихідними даними програми обліку роботи автотранспортного підприємства є інформація про автомобіль, який відповідає заданим вимогам клієнта та вартість замовлення.

ДОДАТОК В
Текст програми

```

package Connection;

import java.io.FileInputStream;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.Date;
import java.sql.*;
import java.io.*;

import com.intersys.jdbc.*;

import ATP.Transport;
import ATP.Voyage;
import ATP.Worker;

public class CacheDB {

    private Connection myConn;
    List<Transport> list;
    List<Worker> listW;
    List<Voyage> listV;
    List<Transport> listT;
    //private String appropriateGovNumber;
    Transport approprTrans;

    public CacheDB() throws Exception{

        Class.forName("com.intersys.jdbc.CacheDriver");
        Properties props = new Properties();
        props.load(new FileInputStream("demo.properties"));
        String user=props.getProperty("user");
        String password=props.getProperty("password");
        String dburl=props.getProperty("dburl");
        CacheDataSource ds = new CacheDataSource();
        ds.setURL(dburl);
        ds.setUser(user);
        ds.setPassword(password);
        myConn = ds.getConnection();
        System.out.println("Db connection succesful to:" + dburl);
    }

    private void close(Statement myStmt, ResultSet myRs) throws SQLException {
        close(null, myStmt, myRs);
    }

    private static void close(Connection myConn, Statement myStmt, ResultSet myRs)
        throws SQLException {

        if (myRs != null) {
            myRs.close();
        }

        if (myStmt != null) {
        }

        if (myConn != null) {
            myConn.close();
        }
    }

    public void addTransport(String govNumber, String _weight, String _volume, String mark,
String model, String _cost, String _consumption)
    {
        PreparedStatement prStmn=null;
        ResultSet rs=null;
        int capacity, volume, cost, consumption;
        capacity = Integer.parseInt(_weight);
        volume = Integer.parseInt(_volume);
        cost = Integer.parseInt(_cost);
        consumption = Integer.parseInt(_consumption);
        try{
            prStmn=myConn.prepareStatement("insert
Test.Transport(GovNumber,Capacity, Volume, Mark, Model, Cost, Consumption, IsFree)
values(?,?,?,?,?,?,?,?)");
            prStmn.setString(1, govNumber);
            prStmn.setInt(2, capacity);
            prStmn.setInt(3, volume);
            prStmn.setString(4, mark);

```

```

        prStmn.setString(5, model);
        prStmn.setInt(6, cost);
        prStmn.setInt(7, consumption);
        prStmn.setBoolean(8, true);
        prStmn.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }finally{
        try {
            close(prStmn,rs);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

public void deleteTransport(int id) throws SQLException
{
    PreparedStatement prStmn=null;
    ResultSet rs=null;
    try{
        prStmn=myConn.prepareStatement("delete from Test.Transport where
ID=?");

        prStmn.setInt(1, id);
        prStmn.executeUpdate();
    }finally{
        close(prStmn,rs);
    }
}

public void addWorker(String firstName, String name, String lastName, String _salary,
String _experience)
{
    PreparedStatement prStmn=null;
    ResultSet rs=null;
    int salary, experience;
    salary = Integer.parseInt(_salary);
    experience = Integer.parseInt(_experience);
    try{
        prStmn=myConn.prepareStatement("insert into Test.Worker( Name,
FirstName, LastName, Salary, Experience, IsFree) values(?,?,?,?,?,?)");
        prStmn.setString(1, name);
        prStmn.setString(2, firstName);
        prStmn.setString(3, lastName);
        prStmn.setInt(4, salary);
        prStmn.setInt(5, experience);
        prStmn.setBoolean(6, true);
        prStmn.executeUpdate();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }finally{
        try {
            close(prStmn,rs);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

public void deleteWorker(int id) throws SQLException
{
    PreparedStatement prStmn=null;
    ResultSet rs=null;
    try{
        prStmn=myConn.prepareStatement("delete from Test.Worker where ID=?");
        prStmn.setInt(1, id);
        prStmn.executeUpdate();
    }finally{
        close(prStmn,rs);
    }
}

public void searchFreeTransport(int capacity, int volume) throws Exception
{
    SearchTransport(capacity, volume);
    System.out.println("list size: " + list.size());
}

```

```

        appropriateTransport();
    }

    public void SearchTransport(int capacity, int volume) throws Exception {
        list = new ArrayList<Transport>();
        PreparedStatement myStmn = null;
        ResultSet rs = null;
        try {
            myStmn = myConn.prepareStatement("select * from Test.Transport where Capacity >=
? and IsFree = ? and Volume >= ?");
            myStmn.setInt(1, capacity);
            myStmn.setBoolean(2, true);
            myStmn.setInt(3, volume);
            rs = myStmn.executeQuery();
            while(rs.next()) {
                Transport tempTransport = convertRowToTransport(rs);
                list.add(tempTransport);
            }
        } finally {
            close(myStmn, rs);
        }
    }

    public Transport convertRowToTransport(ResultSet rs) throws SQLException {
        {
            int ID = rs.getInt("ID");
            String govNumber = rs.getString("GovNumber");
            int capacity = rs.getInt("Capacity");
            int volume = rs.getInt("Volume");
            String mark = rs.getString("Mark");
            String model = rs.getString("Model");
            int cost = rs.getInt("Cost");
            int consumption = rs.getInt("Consumption");
            boolean isFree = rs.getBoolean("IsFree");

            Transport tempTransport = new Transport(ID, govNumber, capacity, volume, mark,
model, cost, consumption, isFree);

            return tempTransport;
        }

        public int getListSize()
        {
            int size = list.size();
            return size;
        }

        public void appropriateTransport() throws SQLException
        {
            int minCapacity = 400;
            for(int i = 0; i < list.size(); i++)
            {
                if(list.get(i).getCapacity() < minCapacity)
                {
                    minCapacity = list.get(i).getCapacity();
                    //appropriateGovNumber = list.get(i).getGovNumber();
                    apprprTrans = new Transport(list.get(i).getID(),
list.get(i).getGovNumber(), list.get(i).getCapacity(), list.get(i).getVolume(),
list.get(i).getMark(), list.get(i).getModel(), list.get(i).getCost(),
list.get(i).getConsumption(), list.get(i).getisFree());
                }
            }
            if(getListSize() >= 1)
            {
                System.out.println(apprprTrans.getID() + " " + apprprTrans.getMark()
+ " " + apprprTrans.getModel());
                //updateTransport();
            }
        }

        public void updateTransport() throws SQLException
        {
            PreparedStatement prStmn = null;
            ResultSet rs = null;
            try {
                prStmn = myConn.prepareStatement("update Test.Transport set IsFree = ? where
ID=?");
                prStmn.setBoolean(1, false);
                prStmn.setInt(2, apprprTrans.getID());
            }
        }
    }

```

```

        prStmn.execute();
    }finally{
        close(prStmn,rs);
    }
}

public String getMark()
{
    return apprprTrans.getMark();
}

public String getModel()
{
    return apprprTrans.getModel();
}

public int getCost()
{
    return apprprTrans.getCost();
}

public double getConsumption()
{
    return apprprTrans.getConsumption();
}

public void setVoyage(String _FIO, String _capacity, String _volume, String _from, String
_to, String _distance, double _money) throws SQLException
{
    searchFreeWorker();
    int workerID = listW.get(0).getID();
    updateWorker(workerID);
    PreparedStatement prStmn=null;
    ResultSet rs=null;
    int capacity = Integer.parseInt(_capacity);
    int volume = Integer.parseInt(_volume);
    int distance = Integer.parseInt(_distance);
    double moneyPlus = distance * 5;

    try{
        prStmn=myConn.prepareStatement("insert into Test.Voyage (FIO,
Capacity, Volume, From, To, Distance, Money, MoneyPlus, Name, FirstName, LastName, Active,
GovNumber) values(?,?,?,?,?,?,?,?,?,?,?,?,?)");
        prStmn.setString(1, _FIO);
        prStmn.setInt(2, capacity);
        prStmn.setInt(3, volume);
        prStmn.setString(4, _from);
        prStmn.setString(5, _to);
        prStmn.setInt(6, distance);
        prStmn.setDouble(7, _money);
        prStmn.setDouble(8, moneyPlus);
        prStmn.setString(9, listW.get(0).getName());
        prStmn.setString(10, listW.get(0).getFirstName());
        prStmn.setString(11, listW.get(0).getLastName());
        prStmn.setBoolean(12, true);
        prStmn.setString(13, apprprTrans.getGovNumber());
        prStmn.executeUpdate();
    }finally{
        close(prStmn,rs);
    }
    System.out.println( apprprTrans.getGovNumber());
}

public void searchFreeWorker() throws SQLException
{
    listW = new ArrayList<Worker>();
    PreparedStatement myStmn= null;
    ResultSet rs=null;
    try{
        myStmn=myConn.prepareStatement("select * from Test.Worker where IsFree = ?");
        myStmn.setBoolean(1, true);
        rs=myStmn.executeQuery();
        while(rs.next()){
            Worker tempWorker=convertRowToWorker(rs);
            listW.add(tempWorker);
        }
    }
    finally{
        close(myStmn,rs);
    }
}

```

```

    }

    public Worker convertRowToWorker(ResultSet rs) throws SQLException
    {
        int ID = rs.getInt("ID");
        String name = rs.getString("Name");
        String firstName = rs.getString("FirstName");
        String lastName = rs.getString("LastName");
        int experience = rs.getInt("Experience");
        int salary = rs.getInt("Salary");
        boolean isFree = rs.getBoolean("IsFree");

        Worker tempWorker = new Worker(ID, name, firstName, lastName, experience,
salary, isFree);

        return tempWorker;
    }

    public void updateWorker(int workerID) throws SQLException
    {
        PreparedStatement prStmn=null;
        ResultSet rs=null;
        try{
            prStmn=myConn.prepareStatement("update Test.Worker set IsFree = ? where
ID=?");

            prStmn.setBoolean(1, false);
            prStmn.setInt(2, workerID);
            prStmn.execute();
        }finally{
            close(prStmn,rs);
        }
    }

    public void GetActiveVoyage()throws Exception{
        listV = new ArrayList<>();
        PreparedStatement myStmn=null;
        ResultSet myRs=null;
        try{
            myStmn=myConn.prepareStatement("select * from Test.Voyage where
Active=?");

            myStmn.setBoolean(1, true);
            myRs=myStmn.executeQuery();
            while(myRs.next()){
                Voyage tempVoyage=convertRowToVoyage(myRs);
                listV.add(tempVoyage);
            }
        }finally{
            close(myStmn,myRs);
        }
    }

    public Voyage convertRowToVoyage(ResultSet rs) throws SQLException
    {
        int ID = rs.getInt("ID");
        String FIO = rs.getString("FIO");
        int capacity = rs.getInt("Capacity");
        int volume = rs.getInt("Volume");
        String from = rs.getString("From");
        String to = rs.getString("To");
        int distance = rs.getInt("Distance");
        double money = rs.getDouble("Money");
        double moneyPlus = rs.getDouble("MoneyPlus");
        String name = rs.getString("Name");
        String firstName = rs.getString("FirstName");
        String lastName = rs.getString("LastName");
        boolean active = rs.getBoolean("Active");
        String govNumber = rs.getString("GovNumber");

        Voyage tempVoyage = new Voyage(ID,FIO, capacity, volume, from, to, distance,
money, moneyPlus, name, firstName, lastName, active, govNumber);

        return tempVoyage;
    }

    public List<Voyage> getList()
    {
        return listV;
    }
}

```

```

public void changeVoyage(int id) throws SQLException
{
    PreparedStatement prStmn=null;
    ResultSet rs=null;

    for(int i=0; i<listV.size(); i++)
    {
        if(listV.get(i).getID() == id)
            id = i;
    }

    try{
        prStmn=myConn.prepareStatement("update Test.Worker      set  IsFree = ? where
FirstName=? and Name=?");
        prStmn.setBoolean(1, true);
        prStmn.setString(2, listV.get(id).getFirstName());
        prStmn.setString(3, listV.get(id).getName());
        prStmn.execute();
    }finally{
        close(prStmn,rs);
    }
    try
    {
        prStmn=myConn.prepareStatement("update Test.Transport set IsFree = ?
where GovNumber=?");
        prStmn.setBoolean(1, true);
        prStmn.setString(2, listV.get(id).getGovNumber());
        prStmn.execute();
    }finally{
        close(prStmn,rs);
    }
    try
    {
        prStmn=myConn.prepareStatement("update Test.Voyage set Active=? where
ID=?");
        prStmn.setBoolean(1, false);
        prStmn.setInt(2, listV.get(id).getID());
        prStmn.execute();
    }finally{
        close(prStmn,rs);
    }
}

public void updateTransId(String id) throws SQLException
{
    PreparedStatement prStmn=null;
    ResultSet rs=null;
    int id1 = Integer.parseInt(id);
    try{
        prStmn=myConn.prepareStatement("update Test.Transport  set IsFree = ? where ID
= ?");
        prStmn.setBoolean(1, true);
        prStmn.setInt(2, id1);
        prStmn.execute();
    }finally{
        close(prStmn,rs);
    }
}

public void getAllTransport() throws SQLException
{
    listT = new ArrayList<Transport>();
    PreparedStatement myStmn= null;
    ResultSet rs=null;
    try{

        myStmn=myConn.prepareStatement("select * from Test.Transport where IsFree = ?
");
        myStmn.setBoolean(1, true);
        rs=myStmn.executeQuery();
        while(rs.next()){
            Transport tempTransport=convertRowToTransport(rs);
            listT.add(tempTransport);
        }
    }finally{
        close(myStmn,rs);
    }
}

```

```

        public List<Transport> getListTransport()
        {
            return listT;
        }
    }
}

```

```
package ATP;
```

```

public class Transport {

    private int ID;
    private String govNumber;
    private int capacity;
    private int volume;
    private String mark;
    private String model;
    private int cost;
    private double consumption;
    private boolean isFree;

    public Transport(int ID, String govNumber, int capacity, int volume, String mark, String
model, int cost, double consumption, boolean isfree)
    {
        this.ID = ID;
        this.govNumber = govNumber;
        this.capacity = capacity;
        this.volume = volume;
        this.mark = mark;
        this.model = model;
        this.cost = cost;
        this.consumption = consumption;
        this.isFree = isfree;
    }

    public int getID()
    {
        return ID;
    }

    public String getGovNumber()
    {
        return govNumber;
    }

    public int getCapacity()
    {
        return capacity;
    }

    public int getVolume()
    {
        return volume;
    }

    public String getMark()
    {
        return mark;
    }

    public String getModel()
    {
        return model;
    }

    public int getCost()
    {
        return cost;
    }

    public double getConsumption()
    {
        return consumption;
    }

    public boolean getisFree()
    {

```

```

        return isFree;
    }
}

```

```
package ATP;
```

```

public class Worker {

    private int ID;
    private String Name;
    private String FirstName;
    private String LastName;
    private int Experience;
    private int Salary;
    private boolean IsFree;

    public Worker(int ID, String name, String firstName, String lastname, int experience, int
salary, boolean isFree)
    {
        this.ID = ID;
        this.Name = name;
        this.FirstName = firstName;
        this.LastName = lastname;
        this.Experience = experience;
        this.Salary = salary;
        this.IsFree = isFree;
    }

    public String getName()
    {
        return Name;
    }

    public String getFirstName()
    {
        return FirstName;
    }

    public String getLastName()
    {
        return LastName;
    }

    public int getID()
    {
        return ID;
    }
}

```

```
package ATP;
```

```

public class Voyage {

    private int ID;
    private String FIO;
    private int capacity;
    private int volume;
    private String from;
    private String to;
    private int distance;
    private double money;
    private double moneyPlus;
    private String name;
    private String firstName;
    private String lastName;
    private boolean active;
    private String govNumber;

    public Voyage(int ID,String FIO, int capacity, int volume, String from, String to, int
distance, double money, double moneyPlus, String name, String firstName, String lastName, boolean
active, String govNumber)
    {
        this.ID = ID;
        this.FIO = FIO;
        this.capacity = capacity;
        this.volume = volume;
        this.from = from;
    }
}

```

```

        this.to= to;
        this.distance = distance;
        this.money = money;
        this.moneyPlus = moneyPlus;
        this.name = name;
        this.firstName = firstName;
        this.lastName = lastName;
        this.active = active;
        this.govNumber = govNumber;
    }

    public int getID(){return ID;}

    public String getFIO(){return FIO;}

    public int getCapacity(){return capacity;}

    public int getVolume(){return volume;}

    public String getFrom(){return from;}

    public String getTo(){return to;}

    public int getDistance(){return distance;}

    public double getMoney(){return money;}

    public double getMoneyPlus(){return moneyPlus;}

    public String getName(){return name;}

    public String getFirstName(){return firstName;}

    public String getLastName(){return lastName;}

    public boolean getActive(){return active;}

    public String getGovNumber(){return govNumber;}
}

```

```
package ATP_forms;
```

```
import java.awt.EventQueue;
import java.awt.Font;
import java.util.List;
```

```
import javax.swing.JFrame;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
```

```
import ATP.Transport;
import ATP.Voyage;
import Connection.CacheDB;
```

```
import java.awt.BorderLayout;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import java.awt.Window.Type;
```

```
public class About {
```

```

    private JFrame About;
    //private JTable Jtable;
    private List<Transport> list;
    CacheDB db;
    private DefaultTableModel tableModel;

```



```

// The 0 argument is
number rows.

        table.setEnabled(true);
        table.setFont(new Font("Tahoma", Font.PLAIN, 15));
        table.setFillViewportHeight(true);
        table.setModel(tableModel);

        loadDataTotable();
    }

    public void loadDataTotable()
    {
        try {
            db.getAllTransport();
            list = db.getListTransport();
            //System.out.println(list.size());
            Object []title = {"ID", "Гос.номер", "Объем", "Грузоподъемность",
"Марка", "Модель", "Свободен (да/нет)"};
            tableModel.addRow(title);

            for (int i = 0; i < list.size(); i++){
                Object[] data = {
                    list.get(i).getID(), list.get(i).getGovNumber(),
list.get(i).getVolume(), list.get(i).getCapacity(), list.get(i).getMark(), list.get(i).getModel(),
list.get(i).getisFree()

                };

                tableModel.addRow(data);
            }
            //System.out.println(list.get(0).getFIO());

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

package ATP_forms;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTabbedPane;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Container;

import javax.swing.JLayeredPane;
import javax.swing.JPanel;
import javax.swing.JComponent;
import javax.swing.JDesktopPane;
import javax.swing.JFormattedTextField;
import javax.swing.JLabel;
import java.awt.Font;

```

```

import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Locale;

import javax.swing.JTextField;

import com.intersys.classes.ResultSet;

import javax.swing.JButton;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import Connection.CacheDB;

public class AdminForm {

    private JFrame frame;
    private JTextField govNumber;
    private JTextField weight;
    private JTextField volume;
    private JTextField firstName;
    private JTextField name;
    private JTextField lastName;
    private JTextField experience;
    private JTextField salary;
    private CacheDB db;
    private JTextField mark;
    private JTextField model;
    private JTextField carId;
    private JTextField workerId;
    private JTextField cost;
    private JTextField consumptioon;
    private JTextField transId;

    /**
     * Launch the application.
     */
    public void newScreen() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    AdminForm window = new AdminForm();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public AdminForm() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize(){
        try {
            db = new CacheDB();
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

        frame = new JFrame();
        frame.setTitle("\u0410\u0434\u043C\u0438\u043D\u0438\u043C");
        frame.setBounds(100, 100, 386, 780);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblNewLabel = new JLabel("\u0418\u043C\u044F \u0438\u043C\u0435\u043D\u0438\u0442\u0435\u043B\u044F");

```

```

lblNewLabel.setBounds(12, 44, 103, 16);
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(lblNewLabel);

JLabel label = new JLabel("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C
\u043C\u0430\u043D\u0448\u043D\u0443");
label.setBounds(12, 13, 161, 16);
label.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label);

JLabel label_1 = new
JLabel("\u0413\u0440\u0443\u0437\u043E\u043F\u043E\u0434\u044A\u0435\u043C\u043D\u043E\u0441\u0442\u0442
\u044C");
label_1.setBounds(12, 73, 140, 16);
label_1.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_1);

JLabel label_2 = new JLabel("\u041E\u0431\u044A\u0435\u043C");
label_2.setBounds(12, 102, 76, 16);
label_2.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_2);

govNumber = new JTextField();
govNumber.setBounds(176, 42, 116, 22);
frame.getContentPane().add(govNumber);
govNumber.setColumns(10);

weight = new JTextField();
weight.setBounds(176, 71, 115, 22);
weight.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent event) {
        char vchar = event.getKeyChar();
        if(!Character.isDigit(vchar))
        {
            event.consume();
        }
    }
});
frame.getContentPane().add(weight);
weight.setColumns(10);

volume = new JTextField();
volume.setBounds(176, 100, 116, 22);
volume.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent event) {
        char vchar = event.getKeyChar();
        if(!Character.isDigit(vchar))
        {
            event.consume();
        }
    }
});
frame.getContentPane().add(volume);
volume.setColumns(10);

JLabel label_3 = new JLabel("\u0442.");
label_3.setBounds(303, 74, 103, 16);
label_3.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_3);

JLabel label_4 = new JLabel("\u043C3");
label_4.setBounds(304, 103, 103, 16);
label_4.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_4);

JButton addTransport = new
JButton("\u0414\u0431\u0431\u0430\u0432\u0438\u0442\u044C \u043C\u0430\u0448\u043D\u0443\u0443");
addTransport.setBounds(75, 249, 192, 25);
addTransport.addActionListener(new ActionListener() { //addCar
    public void actionPerformed(ActionEvent e) {
        db.addTransport(govNumber.getText(), weight.getText(),
volume.getText(), mark.getText(), model.getText(), cost.getText(), consumptioon.getText());
    }
});
addTransport.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(addTransport);

```

```

JLabel label_5 = new JLabel("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C\u0440\u0430\u0431\u043E\u0442\u043D\u0438\u043A\u0430");
label_5.setBounds(12, 287, 161, 16);
label_5.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_5);

JLabel label_6 = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u0444");
label_6.setBounds(12, 316, 103, 16);
label_6.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_6);

JLabel label_7 = new JLabel("\u0418\u043C\u0444");
label_7.setBounds(12, 345, 103, 16);
label_7.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_7);

JLabel label_8 = new
JLabel("\u041E\u0442\u0447\u0435\u0441\u0442\u0432\u043E");
label_8.setBounds(12, 374, 103, 16);
label_8.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_8);

JLabel label_9 = new JLabel("\u0421\u0442\u0430\u0436");
label_9.setBounds(12, 403, 103, 16);
label_9.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_9);

JLabel label_10 = new JLabel("\u041E\u043A\u043B\u0430\u0434");
label_10.setBounds(12, 432, 103, 16);
label_10.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_10);

firstName = new JTextField();
firstName.setBounds(176, 310, 116, 22);
frame.getContentPane().add(firstName);
firstName.setColumns(10);

name = new JTextField();
name.setBounds(176, 339, 116, 22);
frame.getContentPane().add(name);
name.setColumns(10);

lastName = new JTextField();
lastName.setBounds(176, 368, 116, 22);
frame.getContentPane().add(lastName);
lastName.setColumns(10);

experience = new JTextField();
experience.setBounds(176, 397, 116, 22);
frame.getContentPane().add(experience);
experience.setColumns(10);

salary = new JTextField();
salary.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent event) {
        char vchar = event.getKeyChar();
        if(!Character.isDigit(vchar))
        {
            event.consume();
        }
    }
});
salary.setBounds(176, 426, 116, 22);
frame.getContentPane().add(salary);
salary.setColumns(10);

JButton addWorker = new
JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C\u0440\u0440\u0430\u0431\u043E\u0442\u043D\u0438\u0438\u043A\u0430");
addWorker.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        db.addWorker(firstName.getText(), name.getText(),
lastName.getText(), salary.getText(), experience.getText());
    }
});
addWorker.setBounds(75, 457, 192, 25);
addWorker.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(addWorker);

```

```

JButton button = new JButton("\u0412\u044B\u0445\u043E\u0434");
button.setBounds(221, 702, 97, 25);
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        PreEnter preEnter = new PreEnter();
        preEnter.newScreen();
        frame.hide();
    }
});
button.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(button);

JLabel label_11 = new JLabel("\u041C\u0430\u0440\u0430\u0430");
label_11.setBounds(12, 131, 76, 16);
label_11.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(label_11);

JLabel modellabel = new JLabel("\u041C\u043E\u0434\u0435\u0433\u0435\u043B\u044C");
modellabel.setBounds(12, 160, 76, 16);
modellabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
frame.getContentPane().add(modellabel);

mark = new JTextField();
mark.setBounds(176, 129, 116, 22);
frame.getContentPane().add(mark);
mark.setColumns(10);

model = new JTextField();
model.setBounds(176, 158, 116, 22);
frame.getContentPane().add(model);
model.setColumns(10);

JLabel label_12 = new JLabel("\u0423\u0434\u0430\u0430\u043B\u0438\u0442\u044C
\u043C\u0430\u0448\u043D\u0443 \u0438\u0438\u0437 \u0438\u0430\u0430\u0437\u044B
\u0434\u0430\u0430\u043D\u044B\u0445");
label_12.setFont(new Font("Tahoma", Font.PLAIN, 15));
label_12.setBounds(12, 502, 238, 16);
frame.getContentPane().add(label_12);

carId = new JTextField();
carId.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent event) {
        char vchar = event.getKeyChar();
        if(!Character.isDigit(vchar))
        {
            event.consume();
        }
    }
});
carId.setBounds(12, 531, 116, 22);
frame.getContentPane().add(carId);
carId.setColumns(10);

JButton button_1 = new JButton("\u0423\u0434\u0430\u0430\u043B\u0438\u0442\u044C");
button_1.addActionListener(new ActionListener() { //delete car
    public void actionPerformed(ActionEvent e) {
        String id = carId.getText();
        System.out.println("carId = " + id);
        try {
            db.updateTransId(id);
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
});
button_1.setBounds(140, 531, 97, 25);
frame.getContentPane().add(button_1);

JLabel label_13 = new JLabel("\u0423\u0434\u0430\u0430\u043B\u0438\u0442\u044C
\u0440\u0430\u0442\u0438\u043D\u0438\u0430\u0430 \u0438\u0438\u0437 \u0438\u0430\u0430\u0437\u044B
\u0434\u0430\u0430\u043D\u044B\u0445");
label_13.setFont(new Font("Tahoma", Font.PLAIN, 15));
label_13.setBounds(12, 566, 261, 16);
frame.getContentPane().add(label_13);

JButton button_2 = new JButton("\u0423\u0434\u0430\u0430\u043B\u0438\u0442\u044C");
button_2.addActionListener(new ActionListener() {

```

```

        public void actionPerformed(ActionEvent e) {
            int id = Integer.parseInt(workerId.getText());
            try {
                db.deleteWorker(id);
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
    });
    button_2.setBounds(140, 595, 97, 25);
    frame.getContentPane().add(button_2);

    workerId = new JTextField();
    workerId.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent event) {
            char vchar = event.getKeyChar();
            if(!Character.isDigit(vchar))
            {
                event.consume();
            }
        }
    });
    workerId.setBounds(12, 595, 116, 22);
    frame.getContentPane().add(workerId);
    workerId.setColumns(10);

    JLabel label_14 = new
JLabel("\u0421\u0442\u043E\u0438\u043C\u0441\u0441\u0442\u044C");
    label_14.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_14.setBounds(12, 189, 76, 16);
    frame.getContentPane().add(label_14);

    JLabel label_15 = new JLabel("\u0420\u0430\u0441\u0445\u043E\u0434");
    label_15.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_15.setBounds(12, 218, 76, 16);
    frame.getContentPane().add(label_15);

    cost = new JTextField();
    cost.setBounds(176, 187, 116, 22);
    frame.getContentPane().add(cost);
    cost.setColumns(10);

    consumptioon = new JTextField();
    consumptioon.setBounds(176, 216, 116, 22);
    frame.getContentPane().add(consumptioon);
    consumptioon.setColumns(10);

    JLabel label_16 = new JLabel("$");
    label_16.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_16.setBounds(303, 189, 15, 16);
    frame.getContentPane().add(label_16);

    JLabel label_17 = new JLabel("\u0431/100\u043A\u043C.");
    label_17.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_17.setBounds(297, 222, 76, 16);
    frame.getContentPane().add(label_17);

    JLabel label_18 = new JLabel("\u0433\u0440\u0434/\u043A\u043C");
    label_18.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_18.setBounds(297, 432, 103, 16);
    frame.getContentPane().add(label_18);

    transId = new JTextField();
    transId.setBounds(12, 659, 116, 22);
    frame.getContentPane().add(transId);
    transId.setColumns(10);

    JLabel lblNewLabel_1 = new
JLabel("\u041E\u0441\u0432\u043E\u0431\u043E\u0434\u0438\u0442\u044C
\u0430\u0432\u0442\u043E\u043C\u043E\u0431\u0438\u043B\u044C");
    lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 15));
    lblNewLabel_1.setBounds(12, 630, 181, 16);
    frame.getContentPane().add(lblNewLabel_1);

    JButton setIdTrans = new
JButton("\u0418\u0441\u043F\u043E\u0434\u0434\u0438\u0442\u044C");
    setIdTrans.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {

```

```

        //int id = Integer.parseInt(setIdTrans.getText());
        try {
            db.updateTransId(transId.getText());
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
});
setIdTrans.setBounds(140, 658, 97, 25);
frame.getContentPane().add(setIdTrans);
}

package ATP_forms;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import Connection.CacheDB;

import java.awt.event.InputMethodListener;
import java.awt.event.InputMethodEvent;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.SQLException;

public class ClientForm {

    private JFrame frame;
    private JTextField fio;
    private JTextField weight;
    private JTextField volume;
    private JTextField from;
    private JTextField to;
    private JTextField distance;
    private JButton calc;
    private boolean bool1, bool2, bool3;
    private CacheDB db;

    /**
     * Launch the application.
     */
    public void newScreen() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    ClientForm window = new ClientForm();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the application.
     */
    public ClientForm() {
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
        try {

```

```

        db = new CacheDB();
    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    frame = new JFrame();
    frame.setTitle("\u041A\u043B\u0438\u0435\u043D\u0442");
    frame.setBounds(100, 100, 357, 274);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    bool1 = false;
    bool2 = false;
    bool3 = false;

    JButton exit = new JButton("\u0412\u044B\u0445\u043E\u0434"); //выход
    exit.setFont(new Font("Tahoma", Font.PLAIN, 15));
    exit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            PreEnter preEnter = new PreEnter();
            preEnter.newScreen();
            frame.hide();
        }
    });
    exit.setBounds(12, 187, 97, 25);
    frame.getContentPane().add(exit);

    JLabel label = new JLabel("\u0424.\u0418.\u0415.");
    label.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label.setBounds(12, 13, 56, 16);
    frame.getContentPane().add(label);

    JLabel label_1 = new JLabel("\u041C\u0430\u0441\u0441\u0430\u0437\u0430\u0440\u0430\u0443\u0430");
    label_1.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_1.setBounds(12, 42, 97, 16);
    frame.getContentPane().add(label_1);

    JLabel label_2 = new JLabel("\u0415\u0431\u0440\u0435\u043C\u0430\u0437\u0430\u0440\u0430\u0443\u0430");
    label_2.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_2.setBounds(12, 71, 97, 16);
    frame.getContentPane().add(label_2);

    JLabel label_3 = new JLabel("\u0415\u0442\u0430\u0443\u0430\u0443\u0430\u0430\u0430");
    label_3.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_3.setBounds(12, 100, 97, 16);
    frame.getContentPane().add(label_3);

    JLabel label_4 = new JLabel("\u0410\u0443\u0443\u0430\u0430\u0430");
    label_4.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_4.setBounds(12, 129, 97, 16);
    frame.getContentPane().add(label_4);

    JLabel label_5 = new
JLabel("\u0420\u0430\u0441\u0441\u0442\u0435\u0444\u043D\u0438\u0435");
    label_5.setFont(new Font("Tahoma", Font.PLAIN, 15));
    label_5.setBounds(12, 158, 97, 16);
    frame.getContentPane().add(label_5);

    fio = new JTextField();
    fio.setBounds(211, 7, 116, 22);
    frame.getContentPane().add(fio);
    fio.setColumns(10);

    weight = new JTextField();
    weight.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent event) {
            char vchar = event.getKeyChar();
            if(!Character.isDigit(vchar))
            {
                event.consume();
            }
        }
    });
    weight.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent e) {

```

```

        bool1 = true;
        if(bool1 && bool2 && bool3){
            calc.setEnabled(true);
        }
    });
    weight.setBounds(211, 36, 116, 22);
    frame.getContentPane().add(weight);
    weight.setColumns(10);

    volume = new JTextField();
    volume.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent event) {
            char vchar = event.getKeyChar();
            if(!Character.isDigit(vchar))
            {
                event.consume();
            }
        }
    });
    volume.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent e) {
            bool2 = true;
            if(bool1 && bool2 && bool3){
                calc.setEnabled(true);
            }
        }
    });
    volume.setBounds(211, 65, 116, 22);
    frame.getContentPane().add(volume);
    volume.setColumns(10);

    from = new JTextField();
    from.setBounds(211, 94, 116, 22);
    frame.getContentPane().add(from);
    from.setColumns(10);

    to = new JTextField();
    to.setBounds(211, 123, 116, 22);
    frame.getContentPane().add(to);
    to.setColumns(10);

    JButton next = new JButton("\u0414\u0430\u043B\u044C\u0448\u0435");
    next.setEnabled(false);
    next.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            try {
                db.updateTransport();

                double calc = calc();
                db.setVoyage(fio.getText(), weight.getText(),
volume.getText(), from.getText(), to.getText(), distance.getText(), calc );

                ClientTransactionForm transaction = new
ClientTransactionForm();

                transaction.newScreen();
                frame.hide();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });
    next.setFont(new Font("Tahoma", Font.PLAIN, 15));
    next.setBounds(230, 187, 97, 25);
    frame.getContentPane().add(next);

    calc = new
JButton("\u0420\u0430\u0441\u0447\u0438\u0442\u0430\u0442\u0430\u0442\u044C");
    calc.setEnabled(false);
    calc.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) { //calc
            int capacity = Integer.parseInt(weight.getText());
            int volumeInt = Integer.parseInt(volume.getText());
            try {

```

```

        db.searchFreeTransport(capacity, volumeInt);
        if(db.getListSize() >= 1)
        {
            JOptionPane.showMessageDialog(null,"Для
вашего заказа выбран автомобиль " + db.getMark() + " " + db.getModel() + "\nСтоимость составит: "
+ calc() + "грн.", null, JOptionPane.INFORMATION_MESSAGE);
            next.setEnabled(true);
        }
        if(db.getListSize() < 1)
            JOptionPane.showMessageDialog(null,"На
данный момент на базе нету подходящего вида транспорта чтобы принять ваш заказ.", null,
JOptionPane.INFORMATION_MESSAGE);

        calc.setEnabled(false);
    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}

});
calc.setFont(new Font("Tahoma", Font.PLAIN, 15));
calc.setBounds(116, 187, 112, 25);
frame.getContentPane().add(calc);

distance = new JTextField();
distance.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent event) {
        char vchar = event.getKeyChar();
        if(!Character.isDigit(vchar))
        {
            event.consume();
        }
    }
});
distance.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent arg0) {
        bool3 = true;
        if(bool1 && bool2 && bool3){
            calc.setEnabled(true);
        }
    }
});

distance.setBounds(211, 152, 116, 22);
frame.getContentPane().add(distance);
distance.setColumns(10);

}
public double calc()
{
    int distanceInt = Integer.parseInt(distance.getText());
    int weightInt = Integer.parseInt(weight.getText());
    double money = distanceInt*5 + 0.01*db.getCost()*weightInt/10 +
distanceInt*(db.getConsumption()/100)*20;
    return money;
}
}

package ATP_forms;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import javax.swing.JTextField;

import Connection.CacheDB;

import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

```

```

public class ClientTransactionForm {

    private JFrame frame;
    private JTextField textField;
    private JTextField textField_1;

    /**
     * Launch the application.
     */
    public void newScreen() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    ClientTransactionForm window = new
ClientTransactionForm();

                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public ClientTransactionForm() {
        initialize();
    }

    private void initialize() {
        frame = new JFrame();
        frame.setTitle("\u0418\u043F\u043B\u0442\u0430");
        frame.setBounds(100, 100, 296, 152);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblNewLabel = new JLabel("\u041D\u043E\u043C\u0435\u0440\u0430\u0430\u0440\u0442\u044B");
        lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNewLabel.setBounds(12, 13, 105, 16);
        frame.getContentPane().add(lblNewLabel);

        JLabel lblCvv = new JLabel("CVV2");
        lblCvv.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblCvv.setBounds(12, 40, 105, 16);
        frame.getContentPane().add(lblCvv);

        textField = new JTextField();
        textField.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent event) {
                char vchar = event.getKeyChar();
                if(!Character.isDigit(vchar))
                {
                    event.consume();
                }
            }
        });
        textField.setFont(new Font("Tahoma", Font.PLAIN, 15));
        textField.setBounds(129, 11, 116, 22);
        frame.getContentPane().add(textField);
        textField.setColumns(10);

        textField_1 = new JTextField();
        textField_1.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent event) {
                char vchar = event.getKeyChar();
                if(!Character.isDigit(vchar))
                {
                    event.consume();
                }
            }
        });
        textField_1.setFont(new Font("Tahoma", Font.PLAIN, 15));
        textField_1.setColumns(10);
        textField_1.setBounds(129, 38, 116, 22);
        frame.getContentPane().add(textField_1);

        JButton btnNewButton = new
JButton("\u0418\u043F\u043B\u0442\u0430\u0442\u0438\u0442\u044C");
        btnNewButton.addActionListener(new ActionListener() {

```

```

        public void actionPerformed(ActionEvent e) {
            if(textField.getText().equals("") || textField_1.getText().equals(""))
                JOptionPane.showMessageDialog(null,"Заполните все поля",
null, JOptionPane.INFORMATION_MESSAGE);
            else{
                PreEnter enter = new PreEnter();
                enter.newScreen();
                frame.hide();
            }
        }
    });
    btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 15));
    btnNewButton.setBounds(83, 69, 116, 25);
    frame.getContentPane().add(btnNewButton);
}
}

```

```
package ATP_forms;
```

```
import java.awt.EventQueue;
import java.util.List;
```

```
import javax.swing.JFrame;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
```

```
import ATP.*;
import Connection.CacheDB;
import javax.swing.JButton;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import java.awt.Font;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
```

```
public class DispatcherForm {
```

```
    private JFrame frame;
    private JTable table;
    private List<Voyage> list;
    CacheDB db;
    private DefaultTableModel tableModel;
    private JLabel lblNewLabel;
    private JTextField endVoyage;
```

```
    /**
```

```
     * Launch the application.
```

```
     */
```

```
    public void newScreen() {
```

```
        EventQueue.invokeLater(new Runnable() {
```

```
            public void run() {
```

```
                try {
```

```
                    DispatcherForm window = new DispatcherForm();
```

```
                    window.frame.setVisible(true);
```

```
                } catch (Exception e) {
```

```
                    e.printStackTrace();
```

```
                }
```

```
            }
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Create the application.
```

```
     */
```

```

public DispatcherForm() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    try {
        db = new CachedB();
    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    frame = new JFrame();

    frame.setTitle("\u0414\u0438\u0441\u043F\u0435\u0442\u0447\u0435\u0440");
    frame.setBounds(100, 100, 1509, 389);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    String col[] = {"FIO", "Capacity", "Volume", "From", "To", "Distance",
        "Money", "Заработок", "Имя водителя", "Фамилия водителя", "Отчество водителя", "В
        рейсе (да/нет)", "Гос. номер"};

    tableModel = new DefaultTableModel(col, 0);
    // The 0 argument is
    number rows.

    table = new JTable();
    table.setEnabled(false);
    table.setFont(new Font("Tahoma", Font.PLAIN, 15));
    table.setFillViewportHeight(true);
    table.setModel(tableModel);
    //table.getColumnModel().getColumn(0).setPreferredWidth(70);

    JButton end = new
    JButton("\u0437\u0430\u0432\u0440\u0448\u0438\u0442\u044C");
    end.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            int id = Integer.parseInt(endVoyage.getText());
            try {
                db.changeVoyage(id);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    });
    end.setFont(new Font("Tahoma", Font.PLAIN, 15));

    lblNewLabel = new JLabel("\u0420\u0435\u0439\u0441
    \u0434\u0435\u043C\u0435\u043C\u0440");
    lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));

    endVoyage = new JTextField();
    endVoyage.addKeyListener(new KeyAdapter() {
        @Override
        public void keyTyped(KeyEvent event) {
            char vchar = event.getKeyChar();
            if (!Character.isDigit(vchar))
            {
                event.consume();
            }
        }
    });
    endVoyage.setFont(new Font("Tahoma", Font.PLAIN, 15));
    endVoyage.setColumns(10);

    JButton exit = new JButton("\u0412\u044B\u0445\u043E\u0443\u0434\u0438\u0442");
    exit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
            PreEnter preEnter = new PreEnter();

```

```

        preEnter.newScreen();
        frame.hide();
    }
});
exit.setFont(new Font("Tahoma", Font.PLAIN, 15));
GroupLayout groupLayout = new GroupLayout(frame.getContentPane());
groupLayout.setHorizontalGroup(
    groupLayout.createParallelGroup(Alignment.LEADING)
        .addGroup(groupLayout.createSequentialGroup()

.addGroup(groupLayout.createParallelGroup(Alignment.LEADING)

.addGroup(groupLayout.createSequentialGroup()

        .addContainerGap()
        .addComponent(lblNewLabel)
        .addGap(18)
        .addComponent(endVoyage,
GroupLayout.PREFERRED_SIZE, 78, GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(ComponentPlacement.RELATED)

        .addComponent(end)
        .addGap(18)
        .addComponent(exit))
        .addComponent(table,
GroupLayout.DEFAULT_SIZE, 1479, Short.MAX_VALUE)
        .addContainerGap()
);
groupLayout.setVerticalGroup(
    groupLayout.createParallelGroup(Alignment.LEADING)
        .addGroup(groupLayout.createSequentialGroup()
            .addComponent(table,
GroupLayout.PREFERRED_SIZE, 262, GroupLayout.PREFERRED_SIZE)
            .addGap(4)

        .addGroup(groupLayout.createParallelGroup(Alignment.BASELINE)
            .addComponent(lblNewLabel)
            .addComponent(endVoyage,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
            .addComponent(end)
            .addComponent(exit))
);
frame.getContentPane().setLayout(groupLayout);

loadDataTotable();
table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
}

public void loadDataTotable()
{
    try {
        db.GetActiveVoyage();
        list = db.getList();
        //System.out.println(list.size());
        Object []title = {"Номер рейса", "ФИО", "Вес
груза(т)", "Объем(м3)", "Откуда", "Куда", "Расстояние", "Плата", "Заработок", "Имя
водителя", "Фамилия водителя", "Отчество водителя", "В рейсе(да/нет)", "Гос. номер"};
        tableModel.addRow(title);

        for (int i = 0; i < list.size(); i++){
            Object[] data = {
                list.get(i).getID(), list.get(i).getFIO(),
list.get(i).getCapacity(), list.get(i).getVolume(), list.get(i).getFrom(),
list.get(i).getTo(), list.get(i).getDistance(), list.get(i).getMoney(),
list.get(i).getMoneyPlus(), list.get(i).getName(), list.get(i).getFirstName(),
list.get(i).getLastName(), list.get(i).getActive(), list.get(i).getGovNumber()

            };

            tableModel.addRow(data);
        }
    }
}

```

```
        //System.out.println(list.get(0).getFIO());  
    } catch (Exception e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Дипломна кваліфікаційна робота

**Програмне забезпечення обліку роботи
автотранспортного підприємства**

Виконав

ст. гр. КНТ-219сп

О. В. Гришков

Керівник

к. т. н., доцент

Є. О. Гофман

Рисунок Д.1 – Слайд 1

Об’єктом дослідження є процес розробки програмного забезпечення для автотранспортних підприємств.

Предмет дослідження – програмні засоби обліку роботи автотранспортного підприємства.

Мета роботи – програмна реалізація процесу обліку роботи автотранспортного підприємства.

Рисунок Д.2 – Слайд 2

Задачі роботи:

- виконати аналіз предметної області стосовно процесу розробки програмного забезпечення для автотранспортних підприємств;
- виконати проектування програмного забезпечення для обліку роботи автотранспортного підприємства;
- створити програмне забезпечення для обліку роботи автотранспортного підприємства;
- виконати тестування розробленого програмного забезпечення обліку роботи автотранспортного підприємства.

3

Рисунок Д.3 – Слайд 3

Програмні засоби для обліку роботи автотранспортного підприємства

ПЗ	Розміщення	Виписка рахунків	Амортизація	Облік палива	Генератор звітів	GPS
OmniTracs	Windows/Linux Android/iOS	є	є	є	-	є
GPS Trackit	Cloud/SaaS/ Web-Based Android/iOS	є	-	є	-	є
EROAD	Cloud/SaaS/ Web-Based Android/iOS	є	є	-	є	-
Tailwind TMS	Cloud/SaaS/ Web-Based Windows/Linux/ Mac Android/iOS	є	є	-	є	є
Fullbay	Cloud/SaaS/ Web-Based	є	є	є	є	-
Switchboard	Cloud/SaaS/ Web-Based Windows/Linux/ Mac Android/iOS	є	-	є	є	є
Track-POD	Cloud/SaaS/ Web-Based Windows/Linux/ Mac Android/iOS	-	-	є	є	-
Broker PRO	Cloud/SaaS/ Web-Based Windows/Linux/ Mac Android/iOS	є	є	є	-	є

4

Рисунок Д.4 – Слайд 4

Вибір мови програмування

Критерії порівняння	Мова програмування		
	Java	C#	PHP
Зручність використання для створення застосунку обліку роботи автотранспортного підприємства	+	+–	+–
Наявність засобів для об'єктно-орієнтованого програмування	+	+	+–
Засоби захисту інформації	+	+–	+–
Обробка виняткових ситуацій	+	+–	+–
Зручність роботи з базами даних	+	+	+–

5

Рисунок Д.5 – Слайд 5

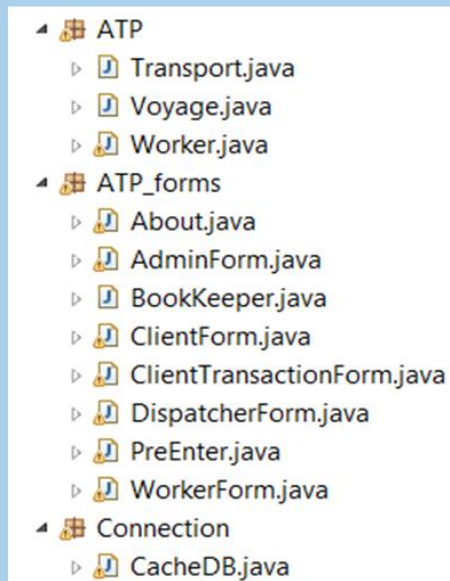
Порівняльний аналіз засобів роботи з базами даних

Критерії порівняння	Системи управління базами даних		
	InterSystems Cache	MS Access	MySQL
Наявність високопродуктивних засобів створення запитів	+	+–	+–
Зручність для створення ПЗ обліку роботи автотранспортного підприємства	+	+–	+–
Робота з розподіленими системами	+	+–	+
Зручність обслуговування дзеркал баз даних	+	+–	+
Надійність захисту інформації	+	+–	+

6

Рисунок Д.6 – Слайд 6

Структура модулів програмного забезпечення обліку роботи автотранспортного підприємства



7

Рисунок Д.7 – Слайд 7

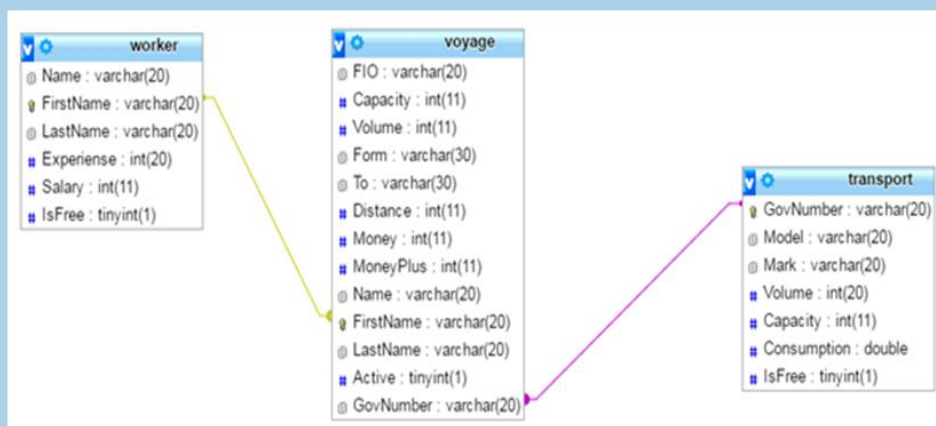
Методи та структури даних програмного забезпечення обліку роботи АТП

Методи та структури даних	Опис
Connection myConn	З'єднання з БД
List<Transport> list	Список усіх вільних ТЗ
List<Worker> lstW	Список усіх вільних водіїв
List<Voyage> listV	Список усіх активних рейсів
Transport appropriTrans	Машинна, яка відповідає умовам рейсу
CacheDB()	Конструктор класу CacheDB
void close(Statement myStmt, ResultSet myRs)	Замкнення з'єднання
void close(Connection myConn, Statement myStmt, ResultSet myRs)	Замкнення з'єднання
addTransport(String govNumber, String	Додати транспорт до БД
Voyage convertRowToVoyage(ResultSet rs)	Конвертує з ResultSet до Voyage
void deleteTransport(int id)	Видалити ТЗ з БД
void addWorker(String firstName, String name, String lastName, String _salary, String _experience)	Додати водія до БД
void deleteWorker(int id)	Видалити водія з БД
void searchFreeTransport(int capacity, int volume)	Пошук в БД вільних ТЗ

8

Рисунок Д.8 – Слайд 8

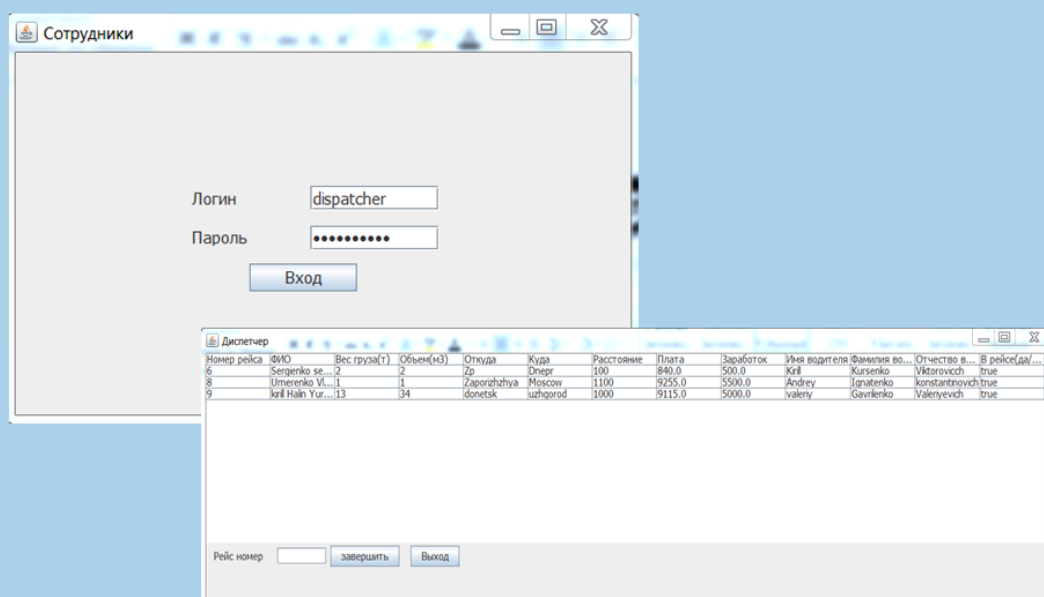
Схема бази даних



9

Рисунок Д.9 – Слайд 9

Приклад роботи з програмою



10

Рисунок Д.10 – Слайд 10

Приклад роботи з програмою

Сотрудники

Логин:

Пароль:

Админ

Добавить машину

Гос. номер:

Грузоподъемность: т.

Объем: м3

Марка:

Модель:

Стоимость: \$

Расход: л/100км.

Добавить работника

Фамилия:

Имя:

Отчество:

Стаж:

Оклад: грн/км

Удалить машину из базы данных

Удалить работника из базы данных

Освободить автомобиль

11

Рисунок Д.11 – Слайд 11

Приклад роботи з програмою

Доступные автомобили

ID	Гос.номер	Объем	Грузоподъемн...	Марка	Модель	Свободен(да/...
3	213f	26	10	Maz	223	true
4	qw1451r	40	25	KAMAZ	356	true
5	rty442	120	22	Scania	Superspace	true
6	4547hq	26	3	Foton	334	true
7	4554t	38	7	Mercedes	Atego 815	true
8	348re	69	20	Mercedes	Actros	true
9	774t	79	20	DAF	335	true
11	3325y	30	30	Ural	420	true

Клиент

Ф.И.О.:

Масса груза:

Объем груза:

Откуда:

Куда:

Расстояние:

12

Рисунок Д.12 – Слайд 12

Висновки

В ході виконання даної дипломної кваліфікаційної роботи було проаналізовано та досліджено процес розробки програмного забезпечення для автотранспортних підприємств.

В процесі розробки вирішено такі завдання:

- аналіз предметної області стосовно процесу розробки програмного забезпечення для автотранспортних підприємств;
- проектування програмного забезпечення для обліку роботи автотранспортного підприємства;
- розробка програмне забезпечення для обліку роботи автотранспортного підприємства;
- тестування розробленого програмного забезпечення обліку роботи автотранспортного підприємства.

13

Рисунок Д.13 – Слайд 13

Висновки

Для створення програмного забезпечення обліку роботи автотранспортного підприємства було обрано мову програмування Java, яка є досить зручною для розробки програмних систем подібного виду. Як засіб для створення та роботи з базою даних про діяльність АТП обрано Cache, що дозволяє забезпечити можливість обліку інформації про водіїв та їхні автівки, а також збереження та обробку іншої інформації.

Запропоновано структуру класів програмного забезпечення обліку роботи автотранспортного підприємства. Розроблено та наведено схему бази даних програми.

Проведені експерименти засвідчили, що програма має достатню працездатність і може використовуватися за призначенням для обліку роботи автотранспортного підприємства.

14

Рисунок Д.14 – Слайд 14