

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА

Навчально-науковий інститут комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Інформаційних управляючих систем та технологій
(повна назва кафедри)

"Допущений до захисту"
Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.
" ____ " _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"
(освітньо-кваліфікаційний рівень)

на тему **Системний аналіз процесів диспетчеризації та розробка
АРМ диспетчера служби міського таксі**

Виконав: студент 4 курсу, групи 4145
напряму підготовки (спеціальності)
124 – "Системний аналіз"
(шифр і назва напряму підготовки, спеціальності)

Михелєв О.І.
(прізвище та ініціали)

Керівник Книрік К.О.
(прізвище та ініціали)

Рецензент Беркунський Є.Ю.
(прізвище та ініціали)

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА**

Інститут, факультет Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра Інформаційних управляючих систем та технологій

Рівень вищої освіти перший (бакалаврський)

Спеціальність 124 – "Системний аналіз"

Гарант освітньої програми

к.т.н, доц. НУК Приходько К.С.

" ____ " _____ 2022 р.

***ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ***
на здобуття ступеня вищої освіти "бакалавр"

Михелєву Олександрю Ігоровичу

1. Тема проекту (роботи) Системний аналіз процесів диспетчеризації та розробка АРМ диспетчера служби міського таксі

керівник проекту (роботи) Михелєв Ігор Леонідович,

Затверджені наказом ректора № 269-уч від "18" травня 2022 року.

2. Строк подання студентом проекту (роботи) 20.06.2020 року.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- Титульний аркуш, завдання на дипломну роботу, анотація (українською, російською, англійською), зміст, перелік умовних означень, символів, одиниць та термінів (при необхідності).

- Вступ.
- Аналіз предметної області і постановка задачі.
- Розробка проекту автоматизованої системи.
- Реалізація проекту автоматизованої системи (технічна документація по проекту).

- Розділ з охорони праці.

- Висновки.

- Список використаної літератури.

- Додатки: загальний опис підсистеми, керівництво користувача, керівництво адміністратора, програма і методика випробувань, графічні інтерфейси, код програми.

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

- 5.1 Організаційна структура служби таксі.
- 5.2 Структура інформаційної системи служби таксі.
- 5.3 Модель інформаційної системи.
- 5.4 Діаграма варіантів використання підсистеми.
- 5.5 Фізична модель бази даних.
- 5.6 Приклади екранних форм підсистеми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Книрік К.О.	10.03.22	31.04.22
2.	Книрік К.О.	10.03.22	29.05.22
3.	Книрік К.О.	10.03.22	06.06.22
4.	Книрік К.О.	10.03.22	13.06.22

7. Дата видачі завдання 10.03.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вивчення і опис предметної області	31.04.22	
2.	Розробка концепції ІС	31.04.22	
3.	Розробка проектних рішень по ІС	29.05.22	
4.	Реалізація проекту, розробка робочої документації	06.06.22	
5.	Вирішення питань з охорони праці	13.06.22	
6.	Виконання графічної частини роботи	15.06.22	
7.	Оформлення пояснювальної записки	15.06.22	
8.	Подання роботи на кафедральний захист	17.06.20	
9.	Подання роботи рецензенту	17.06.20	
10.	Подання роботи на державний захист	20.06.20	

Студент _____ Михелєв О.І.

Керівник проекту (роботи) _____ Книрік К.О.

Анотація

Дипломна робота спрямована на набуття навичок у розробці та проектуванні підсистем. Розробляється підсистема "Диспетчер" інформаційної системи служби таксі.

Робота викладена на 85 сторінках, містить 7 рисунків, 4 додатка та список використаної літератури з 7 найменувань.

Робота виконана українською мовою.

Abstract

Graduate work focused on the acquisition of skills in the development and design of subsystems. Developed subsystem "Manager" service taxi information system.

The work is set out on page 85, contains 7 figures, 4 applications, and the bibliography of 7 items.

The work carried out in the Ukrainian language.

ЗМІСТ

ВСТУП	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ.....	10
1.1 Формування вимог до підсистеми "Диспетчер" ІС служби таксі	10
1.1.1 Опис інформаційної системи служби таксі «Чайка».....	10
1.1.2 Формування вимог підсистеми «Диспетчер» інформаційної системи служби таксі «Чайка».....	11
1.2 Аналіз вимог до підсистеми "Диспетчер" інформаційної системи служби таксі та постановка задачі	14
1.3 Розробка концепції підсистеми "Диспетчер".....	19
2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПО СИСТЕМІ ТА ЇЇ ЧАСТИНАХ...23	
2.1 Загальносистемні рішення.....	23
2.1.1 Загальний опис системи	23
2.1.2 Функції, що реалізує система.....	24
2.1.3 Схема функціональної структури.....	25
2.1.4 Захист системи від несанкціонованого доступу.....	29
2.2 Рішення з організаційного забезпечення.....	30
2.3 Рішення з інформаційного забезпечення.....	31
2.3.1 Вхідні дані системи	31
2.3.2 Вихідні дані системи.....	33
2.3.3 Логічна та фізична структура бази даних	33
2.4 Рішення з технічного забезпечення.....	36
2.5 Рішення з програмного забезпечення.....	37
2.5.1 Структура програмного забезпечення.....	37
2.5.2 Функції частин програмного забезпечення.....	39
3 РОЗРОБКА РОБОЧОЇ ДОКУМЕНТАЦІЇ.....	41
4. ОХОРОНА ПРАЦІ.....	45
4.1 Аналіз небезпечних і шкідливих виробничих факторів, характерних для приміщення лабораторії.....	46
4.2 Розрахунок системи освітлення для приміщення лабораторії.....	49
4.3 Розробка заходів щодо забезпечення сприятливих умов праці	

при роботі з персональним комп'ютером.....	51
Висновки.....	55
Список використаної літератури.....	56
Додаток А – Технічне завдання.....	57
Додаток Б – Інструкція з експлуатації.....	62
Додаток В – Технічний опис.....	72
Додаток Г – Код програми.....	75

ВСТУП

Застосування комп'ютерів пройшло шлях від суто наукових розрахунків до масового управління, від роботи над окремими змінними і файлами до зберігання і обробки величезних масивів інформації. І зараз ми можемо спостерігати один з революційних етапів цього шляху - масове впровадження обробки просторової інформації. І, безсумнівно, головну роль у цій справі відіграють географічні інформаційні системи (ГІС).

В принципі, застосування ГІС можна знайти практично скрізь. Але найбільшу віддачу можна отримати там, де основне значення має просторовий характер інформації. Для геології, екології, картографії, військової справи ситуація очевидна - тут ГІС використовується вже давно й дуже успішно. В транспортну сферу ці системи почали приходити тільки в наші дні. Таке "відставання" обумовлено декількома факторами. Застосування ГІС на транспорті вимагає наявності в цих системах певної функціональності, яка не вимагається в інших областях застосування ГІС. Ну і звичайно, свою роль відіграє суто виробничий, більш консервативний характер транспортної сфери.

На транспорті ГІС можуть використовуватися для вирішення трьох груп завдань:

- Управління інфраструктурою та її розвиток;
- Управління парком рухомих засобів та логістика;
- Управління рухом.

Нижче зазначені напрямки застосування ГІС, де ці системи вже довели свою ефективність:

Залізниця: управління об'єктами інфраструктури (енергопостачання, шляхове господарство, сигналізація і зв'язок), стеження за поїздами та вантажами, аналіз вантажопотоків, моніторинг та реагування на надзвичайні ситуації, інформування пасажирів, оцінка ризиків, планування розвитку мережі, розподіл коштів на ремонт та розвиток.

Автошляхи: планування (спільний аналіз транспортного навантаження та стану дорожнього полотна), моніторинг за рухом, відстеження особистого автотранспорту.

Міський пасажирський транспорт: планування і аналіз маршрутної мережі, диспетчеризація, стеження за рухомим складом, ув'язка розкладів з іншими видами транспорту, опису обладнання на зупинках і кінцевих пунктах, підтримка експлуатації систем енергопостачання, сигналізації і зв'язку, складання та аналіз звітів по ДТП.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Формування вимог до підсистеми "Диспетчер" ІС служби таксі

1.1.1 Опис інформаційної системи служби таксі «Чайка»

Робота таксопарку здійснюється наступним чином: кожен водій, заступаючи на зміну, зв'язується з диспетчером, який вносить його в картку тих, що працюють в даний момент. Відомості картки відображаються на екрані. Водії тримають зворотний зв'язок. Замовлення надходять за телефоном до диспетчера, він записує необхідні дані в базу замовлень. Дату і час надходження замовлення диспетчер вводить сам або вони вводяться автоматично. Клієнт може відразу у диспетчера дізнатися вартість замовлення, і тільки потім підтвердити замовлення. Потім зі списку вільних водіїв вибирається той, хто буде виконувати замовлення (якщо в машинах вбудовані маяки вибирає того, хто ближче до місця), або водій сам відповідає, що прийме замовлення. Після підтвердження водієм диспетчер ставить водієві статус "Зайнятий", точніше це відбувається автоматично, коли статус замовлення "Виконується", тому що у бланку кожного замовлення фіксується виконавець. Після виконання замовлення водій по звітує перед диспетчером, той ставить замовленням статус "Виконано" і у водія змінюється статус на "Вільний". У разі, коли замовлення скасовується, причина скасування вноситься до системи. Диспетчер вносить ім'я того водія, який мав виконувати скасоване замовлення в спеціальний список, який регламентує з чиєї вини замовлення відмінено. Наприклад: водій запізнився на час призначений клієнтом, і внаслідок послуги таксі стали неактуальними, в такому випадку водій пише пояснювальну записку, але це не рятує його від стягнень. Може бути так, що клієнт просто передумав, але не попередив диспетчера, в такому випадку водій все одно пише пояснювальну записку, але ніяких стягнень до нього не застосовується. У цих випадках водій сам оповіщає диспетчера про те, що замовлення анульоване. Ще може бути випадок, коли клієнт передумав, але попередив диспетчера, той зв'язується з водієм і перенаправляє його на інше замовлення. У всіх ви-

падках статус замовлення ставитися "Скасовано". Після кожної зміни диспетчер формує звіт, який показує, скільки замовлень надійшло, скільки виконано, скільки було скасовано, і їх загальну вартість. Ці звіти надходять до адміністратора, на підставі цього адміністратор формує загальний звіт (зведений звіт) за певну дату. В кінці місяця адміністратор, користуючись своїми звітами, формує звіт за підсумками місяця і відправляє його вищому керівництву таксопарку.

Проблеми, які можуть виникнути при здійсненні роботи диспетчера такі:

- Втрата даних диспетчером, або неправильне їхнє трактування.
- Вибір не оптимального маршруту слідування таксистів.
- Некоректна форма подання звітів і т.д.

Шляхи вирішення цих проблем можуть бути наступні:

- Використання стандартних форм для заповнення даних і введення / виведення звітів,
- Використання систем ГІС і математичних формул для визначення оптимальних маршрутів з пункту в пункт і їх вартості
- Створення за допомогою картографічних засобів переліку районів і вулиць, з метою виключення помилок диспетчера.

Інформаційна система служби таксі «Чайка», що розробляється, буде мати структуру, яка зображена на рисунку 1.1.

1.1.2 Формування вимог підсистеми «Диспетчер» інформаційної системи служби таксі «Чайка»

Характеристика підсистеми «Диспетчер» інформаційної системи служби таксі «Чайка»

Проектуючи інформаційну систему служби таксі слід сформулювати вимоги, яким повинна відповідати дана система.

- Перш за все система повинна бути безпечною в умовах її використання. Під безпечною слід розуміти, що система повинна відповідати всім вимо-

гам безпечного використання на об'єкті де вона встановлена, а також при впливі зовнішніх и внутрішніх факторів не приводити к погіршенню стану та роботи системи чи до неможливості її функціонування.

- Система повинна бути універсальною і гнучкою. Універсальна система може застосовуватись у різних областях. Так, наприклад, застосування інформаційної системи служби таксі може бути в нагоді як на звичайних та маршрутних таксі, так і для служб швидкої допомоги та та інших екстрених служб. Наприклад, на маршрутному таксі система може допомогти диспетчеру. Завдяки цій системі особа, яка контролює рейси та маршрути таксі, може в режимі реального часу спостерігати за місцеположенням рухомого складу та віддавати розпорядження водіям про можливі негаразди. Під гнучкістю слід розуміти можливість системи адаптуватись до нових умов, тобто система повинна швидко та без перебоїв адаптуватися до умов, в яких вона використовується. Основним гарантом гнучкості системи є стандартизація проектних рішень, а також продумана та прорахована архітектура інформаційної системи.

В даній дипломній роботі об'єкт автоматизації - це підприємство Таксі «Чайка» міста Одеси. У власності підприємства власний автопарк, який складається з різного транспорту, такого як: легкові автомобілі, вантажні автомобілі, спец техніка, пасажирські мікроавтобуси.

Для підключення інтернет використовується технологія Ethernet. Інтернет підключення PPPoE (Point-to-Point Protocol over Ethernet) - мережевий протокол канального рівня передачі кадрів PPP через Ethernet. Швидкість підключення 90 Мбіт/сек.

Опис існуючої інформаційної системи підприємства таксі «Чайка» та її недоліки.

В основі роботи диспетчерів таксі основним інструментом кожного працівника є персональний комп'ютер, підключений до мережі Інтернет.

На підприємстві, для якого розробляється представлена система всі робочі станції підключені по топології зірка.

Контроль та моніторинг над транспортною системою здійснюється в «паперовій» формі (друкуються маршрутні листи для водіїв), а також у формі обміну повідомленнями через мережу мобільного зв'язку.

Після того, як водій залишає територію підприємства, контроль над ним припиняється. Виробничі витрати на обслуговування автопарку дуже великі. Основні затрати підприємство несе на амортизацію та пальне. Після того, коли над водієм транспортного засобу втрачається контроль, водій може використовувати автотранспорт у своїх власних цілях. Завдяки цьому в автомобілі з'являється додатковий пробіг за який підприємство несе збитки (втрачається пальне). Також може бути ситуація коли водій перебуваючи в рейсі, замість того щоб виконувати службові обов'язки, може зупинитись і займатися своїми справами. В такому випадку підприємство оплачує неіснуючі відпрацьовані години. Інших різних випадків нецільового використання транспорту може бути багато.

Обґрунтування необхідності вдосконалення діючої ІС

Існуюча система контролю транспорту дуже примітивна, і не дає повної картини моніторингу управління наземного транспорту. На сьогоднішній день існує багато способів підвищення точності визначення місцеположення об'єктів використовуючи пристрої GPS. Найточніші дані дають пристрої оснащені спеціальними фільтрами, які забезпечують більш точне визначення на нерівному рельєфі і гірській місцевості, проте в умовах міста це не є дуже важливим. Можливості сучасних мобільних пристроїв на платформі Android, що містять вбудований приймач GPS є достатніми для функціонування системи в умовах міста.

На сьогоднішній день, стала актуальна проблема контролю транспорту на підприємстві. Сьогодні, коли ціни на пальне стрімко летять вгору, керівникам треба піклуватися про те, щоб співробітники підприємства, не витрачали даремно пальне та робочий час.

Перед диспетчером виникають наступні проблеми:

- Яким чином оптимізувати маршрут транспорту, або як виявити «лівий» рейс, коли підприємство не знає в дійсності, де знаходився автомобіль?

- Як визначити місцезнаходження автомобіля в інтервалі між відвіданням кінцевих точок маршруту, які відзначені в замовленні?

- Як визначити, чи дійсно автомобіль затримався до часу прибуття, бо стояв, наприклад, кілька годин у пробці?

Вдосконалення існуючої інформаційної системи дозволить диспетчеру служби таксі в режимі реального часу визначати поточне місце розташування об'єкту моніторингу, та при необхідності переглянути архів маршруту транспортного засобу.

Побудова звітів про рух транспортного засобу, дає можливість диспетчеру аналізувати пересування транспорту і робити висновки про розподіл і доцільності використання автомобіля.

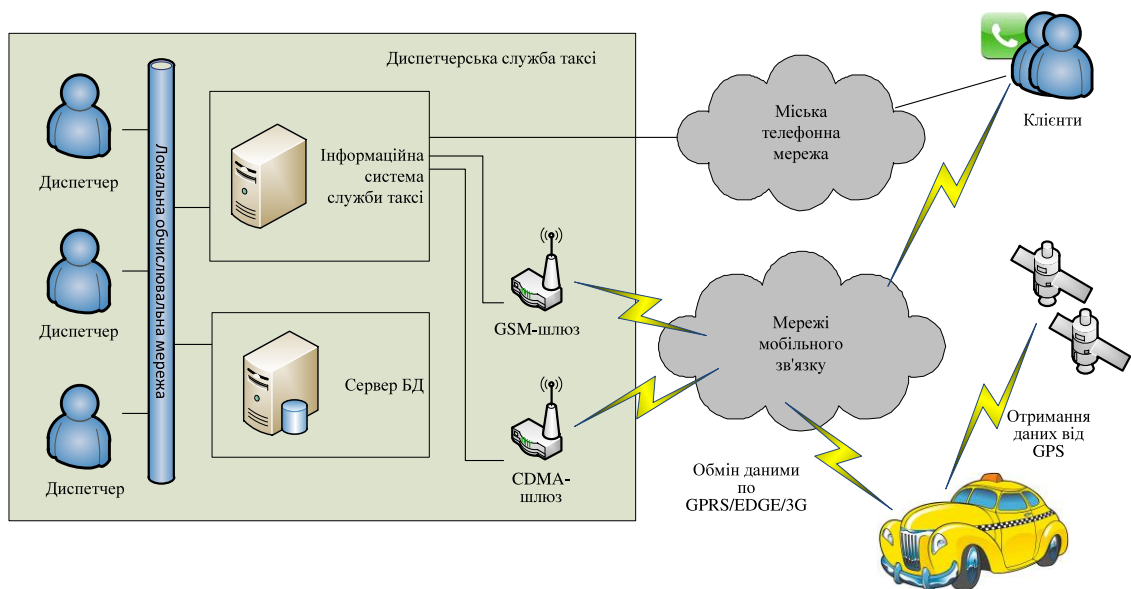


Рисунок 1.1 –Структура інформаційної системи служби таксі

1.2 Аналіз вимог до підсистеми "Диспетчер" інформаційної системи служби таксі та постановка задачі

Основними вимогами до інформаційної системи служби таксі в цілому і підсистеми «Диспетчер» є:

1. Надійність.
2. Захист від несанкціонованого доступу до ІС.

3. Простий і зручний інтерфейс з можливістю віддаленого доступу
4. Проста установка без втручання в режим роботи транспорту.
5. Багатокористувальницька робота ІС.
6. Точність отриманих даних.
7. Гнучкість
8. Використання загальноприйнятих стандартів при розробці.

Аналізуючи вимоги можна підвести підсумки стосовно зазначених вище вимог.

Надійність – властивість технічних об'єктів зберігати у часі у встановлених межах значення всіх параметрів, необхідних для виконання технічних (технологічних та ін.) функцій в заданих режимах і умовах застосування. Під технічними об'єктами розуміють пристрої, прилади, механізми, машини, комплекси обладнання, будівельні конструкції і споруди, технологічні операції і процеси, системи зв'язку, інформаційні системи, автоматизовані системи управління технологічними процесами тощо.

Для надійності інформаційної системи служби таксі слід дотримуватись деяких вимог при використанні.

1. Для того, щоб система функціонувала потрібно безперебійне живлення 5-12 вольт. Для уникнення випадків коли автотранспорт повністю знеструмлюється можна додатково встановити штатний акумулятор на 12 вольт 9 Ah та перетворювач напруги на 5 вольт для роботи мобільних пристроїв.

2. Для визначення поточного місця розташування об'єкту використовується GPS модуль. Для отримання координат потрібен чіткий сигнал GPS від супутнику. Якість сигналу дуже залежить від ландшафту місцевості, при нерівній місцевості, чи будь яких інших перегород сигнал від супутнику може відбиватись і тим самим спотворюватись та надавати неточні дані про місце розташування.

3. Для передачі даних про місце розташування до серверу використовується канал пакетної передачі даних GPRS/EDGE/3G. Для безперебійного online спостереження потрібен стабільний сигнал GSM.

4. Наявність на рахунку мобільного оператора коштів для передачі даних по GPRS каналу.

Захист від несанкціонованого доступу - це запобігання або істотне утруднення несанкціонованого доступу.

На сьогоднішній день в операційних системах існують уразливі місця через які можуть отримувати доступ люди які не повинні його мати.

Для забезпечення надійності даних інформаційної системи на сервері потрібно організувати підсистему реєстрації та обліку користувачів, а при необхідності встановити програмний мережевий екран (firewall) та антивірусне програмне забезпечення. Крім того, мережевий екран дозволить гнучко налаштувати доступ до серверу.

Простий і зручний інтерфейс з можливістю віддаленого доступу.

Для інформаційної системи одною з головних умов – є простий і зручний інтерфейс користувача. Для забезпечення цих умов інтерфейс повинен бути графічним. Графічний інтерфейс (ГІ) – це різновид користувацького інтерфейсу, у якому елементи інтерфейсу (меню, кнопки, значки, списки і т. п.), представлені користувачеві на дисплеї, виконані у вигляді графічних зображень. Для того щоб інформаційною системою змогли користуватися люди з різним рівнем знань у комп'ютерній техніці, ключові елементи графічного інтерфейсу повинні бути інтуїтивно зрозумілі та мати текстове супроводження яке дозволяє користувачу легко орієнтуватись при використанні ІС. Елементи ГІ повинні підказувати користувачу наступний його крок, для цього на ключових елементах потрібно зробити акцент «підсвітити» елемент ГІ.

Проста установка без втручання в режим роботи транспорту.

При масштабному впровадженні інформаційної системи служби таксі система повинна дуже швидко і легко монтуватися на транспортний засіб.

Багатокористувальницька робота ІС.

Для того, щоб в системі змогли працювати два або більше користувача, доступ до бази даних повинен мати розмежування прав користувачів. Для цього на сервері БД потрібно зробити механізм аутентифікації. Та для кож-

ного користувача встановити різні права доступу до таблиць БД у відповідності з тим, що потрібно користувачу.

Точність отриманих даних.

До 2000 року точність отримання даних за допомогою GPS була дуже великою проблемою, тому що супутники передавали спеціальну похибку.

Спеціальна похибка - це штучна фальсифікація часу в сигналі переданому супутником. Для цивільних GPS приймачів ця похибка вела до менш точного визначенню координат (до 100м), рисунок 1.2. Спеціальна похибка була включена для забезпечення безпеки, для того щоб терористи не могли точно направити ракети використовуючи GPS.

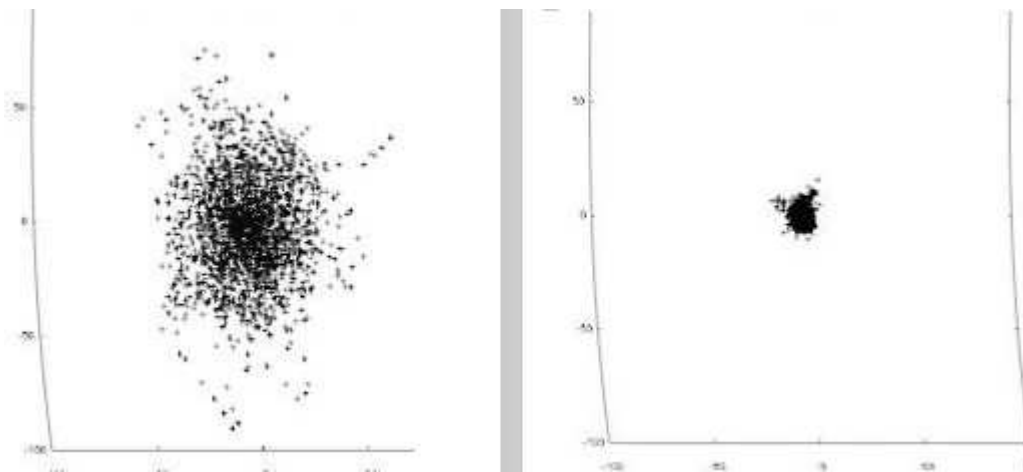


Рисунок 1.2 Точність отримання даних з спеціальною похибкою та без неї

Іншим фактором який впливає на точність – це геометрія супутників. Розташування супутників на небі суттєво впливає на точність. Чим більше супутники розподілені на небі тим більш точні будуть отриманні дані.

Вплив відображення сигналів також дуже впливає на точність. У міській місцевості, сигнал від супутників може спотворюватись відбиваючись від стін споруд тим самим збільшуючи час надходження сигналу.

В атмосфері землі, а саме іоносфері та тропосфері зменшується швидкість сигналу. Через це дані які надходять до GPS приймача також отримують невелику похибку, рисунок 1.3.



Рисунок 1.3 Схема спотворенню сигналу при проходженні крізь атмосферу.

Для зменшення цього ефекту у сучасних пристроях GPS використовуються спеціальні фільтри, які враховують похибку іоносфери і тропосфери тим самим збільшуючи точність отримання даних.

Останнє що впливає на точність це неточність годинника пристрою GPS. На супутнику функціонують атомний годинник, який є дуже точним.

Укупі всі фактори дають наступну похибку

Ефект Іоносфери ± 5 метрів

Зміна орбіти супутника ± 2.5 метра

Похибка годинника супутника ± 2 метри

Вплив відображення сигналів ± 1 метр

Ефект тропосфери ± 0.5 метра

Похибки обчислення та округлення ± 1 метр

В сумі це дає похибку у 12 метрів. Але при використанні спеціальних фільтрів, які враховують ці похибки, точність отриманих даних сягає $\pm 3-5$ метрів.

Гнучкість. Система повинна бути гнучкою щоб пристосовуватись до будь яких поставлених на неї задач. При необхідності до неї можна приєднати датчики такі як:

1. Датчик рівню палива.
2. Датчик витрат палива.
3. Пристрій дистанційного старту та зупинки двигуна

4. Датчик температури
5. Датчик підрахунку пасажирів

Датчики дозволяють отримати додаткову інформацію про об'єкти моніторингу.

Використання загальноприйнятих стандартів при розробці.

Використовуючи загальноприйняті стандарти при розробці інформаційної системи, дозволяє легко модернізувати існуючу систему, а в разі коли модернізацією займається інша людина, їй буде легше зрозуміти принцип дії всіх модулів. В разі несправності людина, яка буде займатися ремонтом чи виявленням неполадки, буде краще орієнтуватись у принципіальній схемі підключення пристрою.

1.3 Розробка концепції підсистеми "Диспетчер"

Виходячи із аналізу вимог можна виділити декілька основних компонентів майбутньої підсистеми:

1. Серверна частина (з серверним програмним забезпеченням)
2. Система управління базами даних
3. Клієнтська частина (з набором клієнтського програмного забезпечення)
4. Технічні засоби.

Далі ми детально розглянемо окремо кожні компоненти.

Серверна частина.

Перш за все буде використовуватись архітектура клієнт-сервер. Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосувань і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

1. Набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

2. Набір клієнтів, які використовують сервіси, що надаються серверами;

3. Мережа, яка забезпечує взаємодію між клієнтами та серверами.

На сервері повинно бути встановлена серверна операційна система Windows Server 2003 чи Windows Server 2008.

Система управління базами даних.

Також на сервері повинно бути встановлена розподілена система управління базами даних (СУБД). Клієнт-серверна СУБД розташовується на сервері разом з БД і здійснює доступ до БД безпосередньо, в монопольному режимі. Всі клієнтські запити на обробку даних обробляються клієнт-серверною СУБД централізовано. Прикладом найвідоміших СУБД які можуть бути встановлені у якості сервера баз даних на сервері - це: Firebird, Interbase, MS SQL, MySQL.

Клієнтська частина представляє собою будь яке робоче місце з встановленими на нього операційною системою Windows не нижче версії XP або Linux, MAC OS. Робоче місце повинно мати доступ до мережі Інтернет з каналом доступу не менше 3 мбіт/сек. Мати монітор з роздільною здатністю не менше 1024*768.

Програмне забезпечення, яке повинно обов'язково встановлено на робочих місцях це веб-браузери: MS Internet Explorer версії не менше 8.0, Google Chrome, Mozilla FireFox, Opera, Safari. Додатково встановлений Flash player версії не менше 11.0, розроблене програмне забезпечення.

Технічні засоби. До технічних засобів належать всі апаратні засоби які потрібні для нормального функціонування інформаційної системи. До них належить:

1. Апаратне забезпечення клієнтського робочого місця
2. Апаратне забезпечення серверної частини
3. Апаратне забезпечення яке встановлюється безпосередньо на об'єкті моніторингу.

Для комфортної роботи клієнтська частина повинна мати такі технічні характеристики:

1. Процесор – intel Core 2 duo 2 Ghz і більше
2. Оперативна пам'ять – 2 Gb
3. Відеокарта – не менше 256mb
4. Мережева карта - 100 мбіт
5. Місце на жорсткому диску не менше 6 гБ.
6. Мишка
7. Дисковий привід DVD
8. Клавіатура
9. Монітор з роздільною здатністю 1024*768 і більше.

Для забезпечення рекомендованих системних вимог на серверній частині, його потрібно обладнати наступними комп'ютерними комплектуючими:

1. Процесор – intel Core 2 duo 2,5 Ghz і більше
2. Оперативна пам'ять – 3 Gb
3. Відеокарта – не менше 256mb
4. Мережева карта - 100 мбіт
5. Місце на жорсткому диску не менше 80 гБ.
6. Дисковий привід DVD
7. Мишка або трекбол
8. Клавіатура
9. Монітор з роздільною здатністю 1024*768 і більше.

Апаратне забезпечення яке встановлюється на об'єкті моніторингу може складатись з різних апаратних комплексів в залежності від задачі яку потрібно вирішувати інформаційній системі.

Основний компонент який встановлюється на транспортному засобі – це GPS модуль, який виконує головну роль у цій інформаційній системі.

На сьогоднішній день існує багато варіантів і розробок різних компаній, які реалізували пристрій для GPS моніторингу, проте через поширення достатньо дешевих мобільних пристроїв, що містять приймач GPS та операцій-

ну систему Android v2.3 або більш нову, було прийнято рішення про оснащення рухомого складу саме ними.

Такі пристрої мають декілька привабливих особливостей.

1. Невелика ціна. Головна перевага цього рішення.
2. Наявність вбудованого мікрофону для дистанційного прослуховування.

Але є і багато суттєвих недоліків в даному рішенні, такі як:

1. Слабка відмовостійкість. При перепадах напруги та сильних вібраціях пристрій може самовільно відключатись, або вийти з ладу.
2. Відсутність фільтру який фільтрує похибки GPS при проходженні сигналу крізь атмосферу.

Таким чином, розглянувши компоненти побудови інформаційної системи можна визначити концепцію підсистеми "Диспетчер".

ПЗ Серверу – Windows Server 2003\2008.

ПЗ клієнту – Windows XP\Vista\7, ПЗ objecttracker.

Технічні засоби – мобільний пристрій з GPS приймачем.

2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПО СИСТЕМІ ТА ЇЇ ЧАСТИНАХ

2.1 Загальносистемні рішення

2.1.1 Загальний опис системи

Підсистема "Диспетчер" інформаційної системи служби таксі призначена для автоматизованого контролю транспортних засобів, які працюють у режимі звичайного, або маршрутного таксі. Інформація, яку отримують спеціалісти підприємства з даної інформаційної системи не тільки дозволяє отримувати оперативну інформацію про поточне географічне положення транспортного засобу для контролю виконання спланованого маршруту, але й, на основі отриманих даних, оперативно змінювати маршрути, коли це є необхідним за різних обставин.

Для реалізації такого контролю було використано пристрої, що передають за допомогою мережі Інтернет дані з транспортного засобу до комп'ютеру, який їх обробляє.

На етапі проектування інформаційної системи була розроблена модель, яка відображує її частини та зв'язки між цими частинами. Дану модель можна бачити на рис. 2.1.

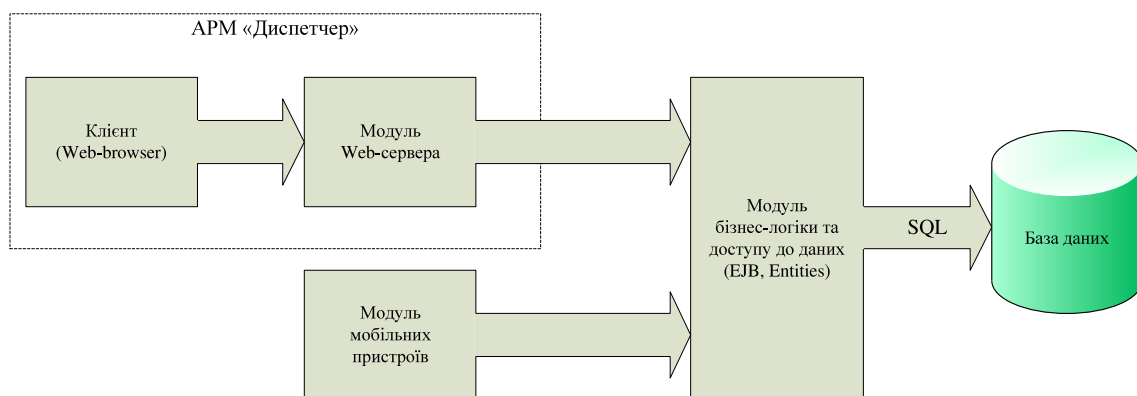


Рис. 2.1 Модель інформаційної системи

Для функціонування інформаційна система використовує засоби операційної системи комп'ютера, на якому вона виконується. Саме засоби операційної системи дозволяють побудувати зв'язки між окремими частинами,

як це показано на рис. 2.1, а також забезпечити отримання інформації, що надходить з мобільних пристроїв через мережу Інтернет, підсистемою "Диспетчер". В свою чергу підсистема візуалізації даних використовує стандартні засоби операційної системи (бібліотеки) для побудови інтерфейсу користувача.

Додатково було використано засоби Інтернет-сервісу «Google» для відображення поточного географічного положення об'єктів, за допомогою якого можливо вбудовувати карту у вікно підсистеми візуалізації даних.

2.1.2 Функції, що реалізує система

Оскільки інформаційна система поділяється на декілька частин, то розглянемо функції окремо кожної з них.

Підсистема "Диспетчер" є першою ланкою потоку даних від пристрою, що їх формує, до користувача, який переглядає їх. Функціями підсистеми є отримання за допомогою мережі Інтернет даних від пристрою та додавання їх до бази даних шляхом надсилання запиту до СУБД. Також однією з функцій даної підсистеми є перевірка надісланих даних на коректність за допомогою вирахування контрольної суми. Якщо контрольна сума не співпадає, дані вважатимуться хибними та не будуть занесені до бази даних.

Функції СУБД даної системи є досить стандартними, оскільки використовується універсальний продукт для обробки даних MySQL. Ця СУБД приймає запити та формує відповіді на них, а також контролює коректність надісланих даних.

Підсистема візуалізації даних має необхідний комплекс функцій для забезпечення візуального спостереження за станом транспортних засобів та надання статистичної інформації користувачу, а також має декілька сервісних функцій для зручності використання інтерфейсу користувача:

- обмін даними з базою даних;
- перевірка введених даних для входу в систему;

- відображення карти, на якій показано поточне місцезнаходження транспортного засобу, а також може бути показано маршрут за обраний період часу;
- відображення таблиць з поточними даними, що надійшли від транспортних засобів та журналу подій;
- побудова графіків зміни швидкості та рівня пального;
- зміна кольорової теми інтерфейсу користувача.

2.1.3 Схема функціональної структури

Підсистема "Диспетчер" використовується безпосередньо співробітниками підприємства. Для наглядного представлення варіантів використання системи зазвичай створюються спеціальні діаграми, що відображують окремо кожну функцію системи та взаємодію цих функцій з користувачем. Для побудови такої діаграми застосовують UML-моделювання.

UML (англ. Unified Modeling Language - Уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, званої UML-моделлю. UML був створений для визначення, візуалізації, проектування та документування в основному програмних систем. За допомогою даного засобу можна створювати різноманітні діаграми, але для нашої задачі на етапі проектування буде використаний один вид – діаграма варіантів використання (Use case diagram).

На рис. 2.2 представлена діаграма варіантів використання інформаційної системи користувачами підприємства. Відповідно до зображеної діаграми нижче у таблицях 2.1-2.6 наведено специфікація кожного з варіанту використання.

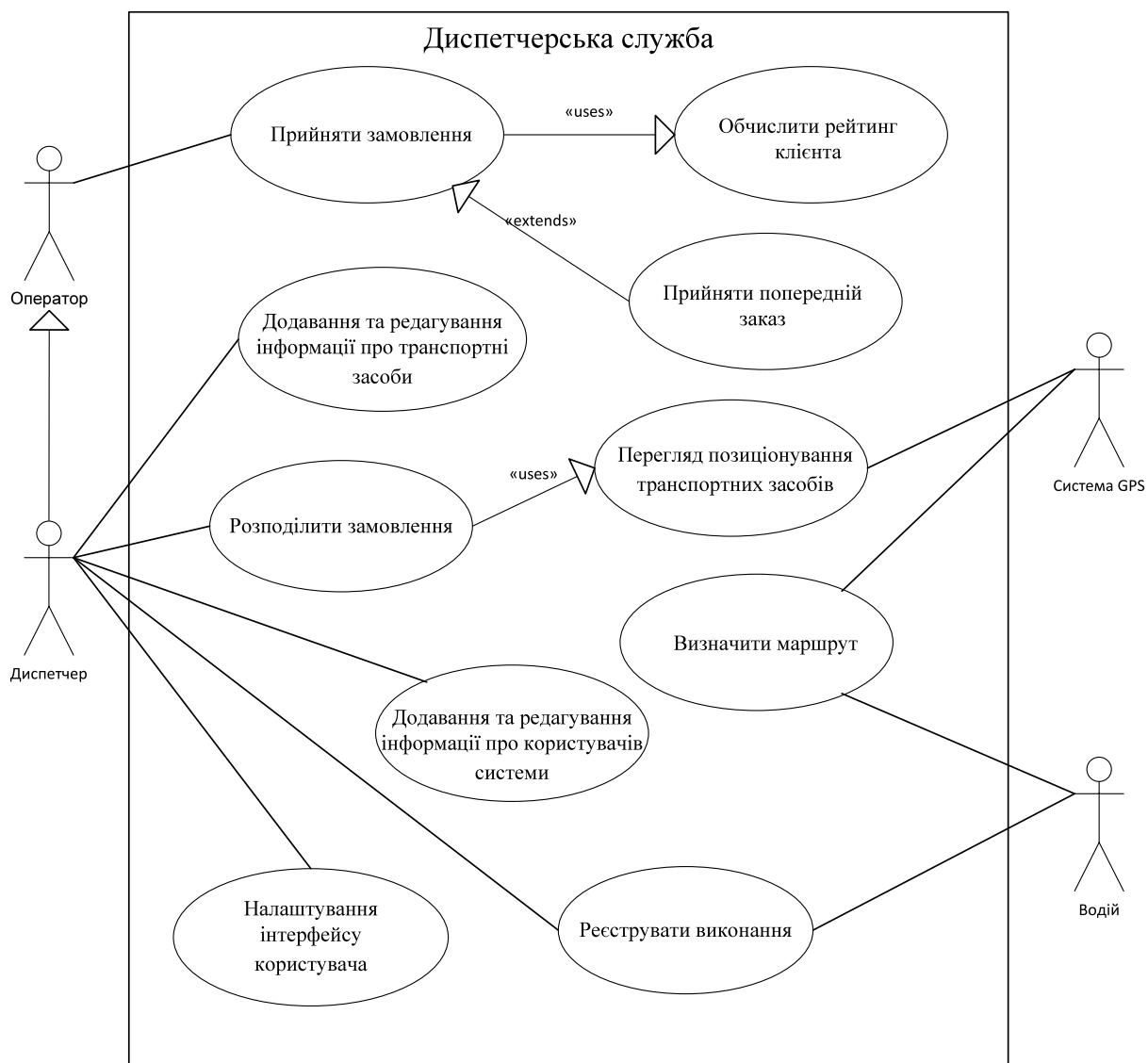


Рис. 2.2 UML-діаграма варіантів використання.

Таблиця 2.1

Варіант використання «Додавання та редагування інформації про транспортні засоби»

Контекст використання	Використовується для внесення до бази даних змін щодо транспортних засобів підприємства
Діюча особа	Користувач з правами адміністратора
Передумова	Користувач з правами адміністратора системи отримав нові дані про транспортний засіб та бажає внести їх до таблиці БД, що містить ці дані
Тригер	Натискання на елемент «Додати» або «Редагувати» кон-

	текстного меню таблиці з даними про транспортні засоби
Сценарій	Користувач з правами адміністратора входить у систему, у таблиці, що містить відповідні дані, обирає зі списку транспортних засобів необхідний, визиває контекстне меню таблиці, обирає пункт меню «Редагувати» або «Додати», у вікні внесення змін до запису таблиці редагує необхідні поля, та натискає кнопку «ОК» для підтвердження внесених змін
Постумова	Дані про транспортний засіб заносяться до бази даних для зберігання та подальшої їх використання

Таблиця 2.2

Варіант використання «Додавання та редагування інформації про користувачів системи»

Контекст використання	Використовується для внесення до бази даних змін щодо користувачів інформаційної системи
Діюча особа	Користувач з правами адміністратора
Передумова	Користувач з правами адміністратора системи отримав нові дані про користувача системи та бажає внести їх до таблиці, що містить ці дані. Наприклад, необхідно додати нового користувача системи або змінити пароль існуючого.
Тригер	Натискання на елемент «Додати» або «Редагувати» контекстного меню таблиці з даними користувачів системи
Сценарій	Користувач з правами адміністратора входить у систему, натискає на кнопку «Користувачі», у вікні з таблицею, що містить відповідні дані, обирає зі списку користувача (якщо необхідно, наприклад, змінити його пароль або інші дані), визиває контекстне меню таблиці, обирає

	пункт меню «Редагувати» або «Додати», у вікні редагування даних користувача для внесення змін до запису таблиці змінює необхідні поля, та натискає кнопку «ОК» для підтвердження внесених змін. Якщо необхідно додати нового користувача, для відкриття контекстного меню можна натискати на будь-якому записі таблиці
Постумова	Дані користувача заносяться до бази даних для зберігання та подальшої їх використання

Таблиця 2.3

Варіант використання «Налаштування інтерфейсу користувача»

Контекст використання	Використовується для встановлення кольорової теми інтерфейсу
Діюча особа	Користувач
Передумова	Користувач бажає змінити тему інтерфейсу
Тригер	Натискання на пункт списку доступних тем
Сценарій	Користувач входить у систему, натискає на випадаючий список з переліком тем, обирає та натискає на необхідний пункт переліку
Постумова	Змінюється кольорова тема інтерфейсу користувача

Таблиця 2.4

Варіант використання «Перегляд позиціонування транспортних засобів»

Контекст використання	Використовується для відстеження поточного місцезнаходження транспортних засобів на карті
Діюча особа	Користувач
Передумова	Користувачу необхідно отримати дані про поточне позиціонування транспортних засобів

Тригер	Вхід до системи
Сценарій	Користувач входить до системи та обирає необхідний масштаб карти
Постумова	Отримано інформацію щодо поточного позиціонування транспортних засобів

Таблиця 2.5

Варіант «Перегляд маршрутів»

Контекст використання	Використовується для відстеження маршруту, який пройшов транспортний засіб за певний період часу
Діюча особа	Користувач
Передумова	Користувачу необхідно отримати дані про пройдений маршрут транспортного засобу
Тригер	Натискання на кнопку відображення маршруту
Сценарій	Користувач входить у систему, обирає необхідний масштаб карти, натискає на кнопку відображення маршруту панелі управління
Постумова	Отримано інформацію щодо пройденого маршруту обраного транспортного засобу

2.1.4 Захист системи від несанкціонованого доступу

Об'єктом захисту в інформаційній системі є інформація з обмеженим доступом, яка циркулює та зберігається у вигляді даних, команд, повідомлень, що мають певну обмеженість і цінність як для її власника, так і для потенційного порушника технічного захисту інформації. Порушник – користувач, який здійснює несанкціонований доступ до інформації

Загроза несанкціонованого доступу – це подія, що кваліфікується як факт спроби порушника вчинити несанкціоновані дії стосовно будь-якої частини інформації в інформаційній системі.

Потенційні загрози несанкціонованого доступу до інформації в інформаційних системах поділяють на цілеспрямовані (умисні) та випадкові. Умисні загрози можуть маскуватися під випадкові шляхом довгочасної масованої атаки несанкціонованими запитами або комп'ютерними вірусами. Випадковими загрозами є неуважність співробітників підприємства.

Для запобігання доступу до даних інформаційної системи особами, які не мають на це повноважень були розроблені заходи щодо захисту системи від несанкціонованого доступу. Організаційні заходи щодо забезпечення захисту технічного забезпечення покладаються повністю на керівництво підприємства. Наприклад, комп'ютер, який містить на своїх носіях базу даних, має бути встановлений у важкодоступному для потенційних злоумисників місці. Технічне забезпечення функціонування системи захищаються спеціальними засобами операційної системи, а також за допомогою спеціалізованого програмного забезпечення (наприклад, антивірус).

Стосовно програмного забезпечення самої системи для забезпечення захисту від несанкціонованого доступу було створено окремий перелік користувачів та розмежування прав доступу користувачів. Неможливо отримати дані інформаційної системи без пройденої аутентифікації на етапі її запуску.

2.2 Рішення з організаційного забезпечення

Використання інформаційної системи для автоматизації диспетчерської служби таксі дозволяє повністю відмовитися від використання рації для таксі.

Робота водіїв ведеться з використанням спеціального програмного забезпечення таксі gprs (Java-додаток), що встановлюється на звичайний мобільний телефон або комунікатор.

Виклик приймається будь-яким каналом зв'язку, обробляється оператором та надходить до диспетчера таксі. У диспетчера на екрані монітора відкривається картка замовлення з уже заповненим номером телефону. У разі якщо власник телефонного номера раніше звертався в службу таксі, у формі

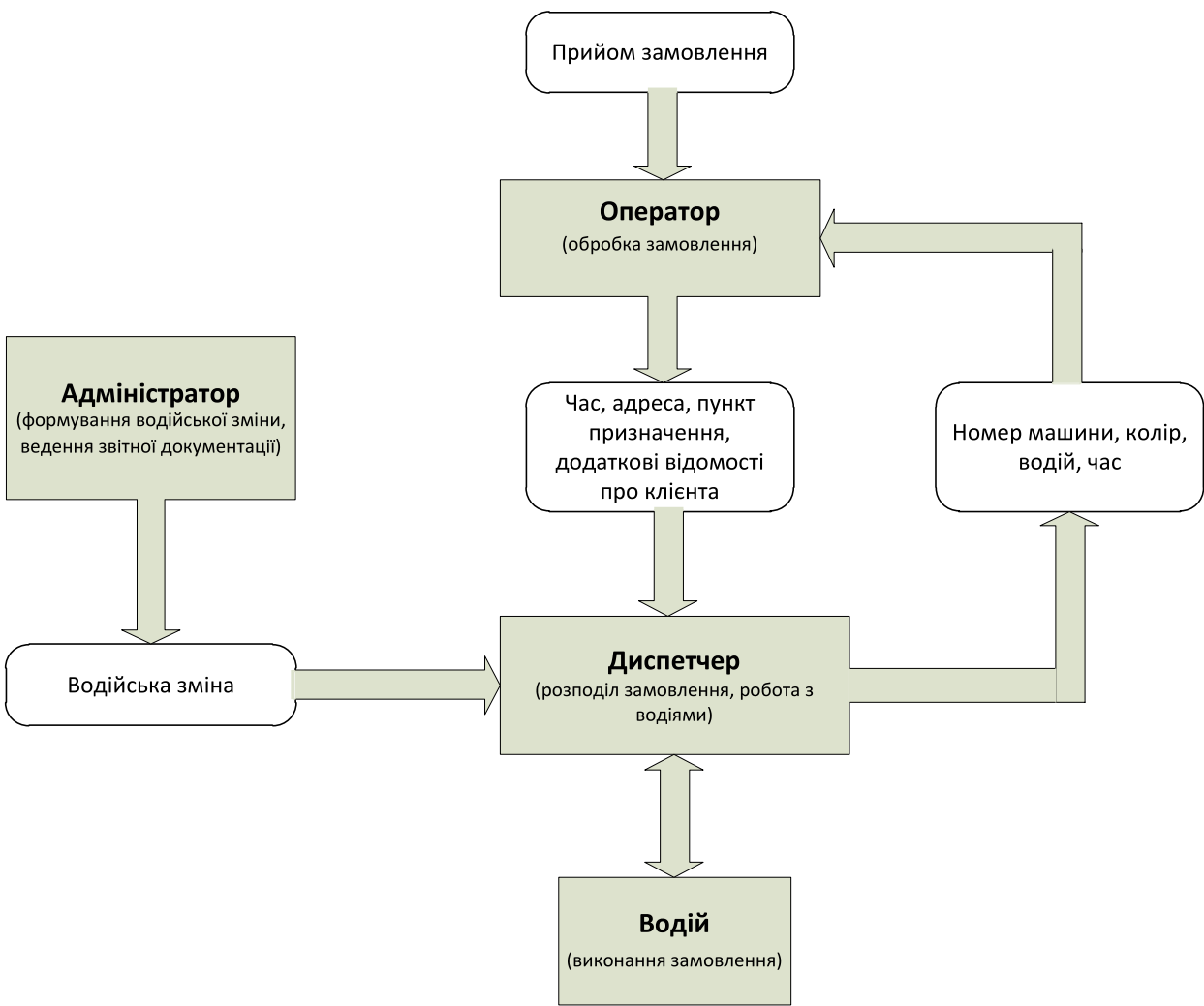
замовлення будуть заповнені всі поля з інформацією про останні поїздки клієнта.

Диспетчер заповнює поля форми замовлення і відправляє замовлення на сервер. Водій натискає на клавішу готовності виконати замовлення, якщо замовлення його цікавить, і вказує час під'їзду до клієнта.

Інформаційна підсистема "Диспетчер" автоматично додзвонюється до клієнта і повідомляє про час під'їзду і номері машини. Інформація до клієнта може доводитися і за допомогою відправки SMS. Водій отримує підтвердження замовлення й виїжджає до клієнта.

Рис. 2.3. Організаційна структура служби таксі

2.3 Рішення з інформаційного забезпечення



2.3.1 Вхідні дані системи

Вхідні дані отримуються з двох джерел:

- інформація, яку передає пристрій, що знаходиться безпосередньо у цьому транспортному засобі;
- інформація, яка вводиться користувачем системи для зручності перегляду даних, отриманих з пристроїв.

З пристрою система отримує дані про поточний стан та положення транспортного засобу:

- унікальний код транспортного засобу;
- кількісний показник залишку пального;
- вісім дискретних входів загального призначення;
- стан вимикача аварійної сигналізації
- стан датчика кнопки виклику;
- стан підключення зовнішнього живлення;
- наявність у пристрої батареї (автономного живлення);
- стан роботи GPS-модуля;
- наявність карти пам'яті у пристрої;
- поточна виміряна швидкість;
- географічна довгота;
- географічна широта.

При надходженні інформації з пристрою також фіксується час виміру. У разі неотримання інформації від транспортного засобу упродовж визначеного проміжку часу вважається, що зв'язок з пристроєм було втрачено. Ці два фактори також є вхідними даними системи.

Від користувача система отримує наступні вхідні дані:

- унікальний код транспортного засобу;
- марка та модель транспортного засобу;
- номер реєстрації транспортного засобу;
- опис транспортного засобу;
- найменування груп транспортних засобів;
- опис груп транспортних засобів;
- найменування типів подій;
- унікальні коди типів подій;

- опис типів подій;
- імена користувачів системи;
- паролі користувачів системи;
- тип користувача.

Всі вхідні дані потребують збереження у сховищі даних, яке забезпечує одна з частин інформаційної системи – СУБД.

2.3.2 Вихідні дані системи

Вихідними даними інформаційної системи є:

- відображене на карті поточне положення транспортних засобів;
- відображення на карті маршруту транспортного засобу за обраний проміжок часу;
- відображений у таблицях стан зв'язку з транспортним засобом та його основні показники;
- відображений у таблиці журнал повідомлень;
- графік швидкості та використання пального за обраний проміжок часу.

2.3.3 Логічна та фізична структура бази даних

Для обробки вхідних даних виникає необхідність їх накопичення та зберігання. Зазвичай для таких задач створюється база даних, яка має певну структуру, що відповідає принципам побудови обраному типу банку даних. У нашому випадку – реляційна база даних. Вхідні дані розподіляються на окремі таблиці, які відповідають принципам нормалізації реляційних баз даних.

В сформованій структурі бази даних основними таблицями є:

- «status» – зберігає поточні дані про стан транспортного засобу;
- «zakaz» – журнал замовлень таксі;
- «car» – основні дані транспортних засобів;
- «cartype» – групи, на які поділяються всі транспортні засоби;

- «tarif» – вартість проїзду 1 км;
- «client» – дані о замовниках;
- «dispatcher» – дані о диспетчерах.

Зберігання інформації у базі даних за допомогою комп'ютера вимагає їх чіткої типізації. Тобто коли створюються таблиці бази даних будь-якої з існуючих СУБД, кожне поле таблиці повинно бути описане наступними параметрами:

- ім'я (латинськими літерами без пробілів);
- тип даних, відповідно до якого будуть зберігатись дані;
- розмірність поля (наприклад, кількість літер у символьному полі).

СУБД MySQL надає доволі велику кількість типів даних для створення полів таблиць. Розглянувши можливі варіанти та обравши підходящі, відповідно до логічної структури, було розроблено фізичну структуру бази даних, що зображено на рис. 2.4.

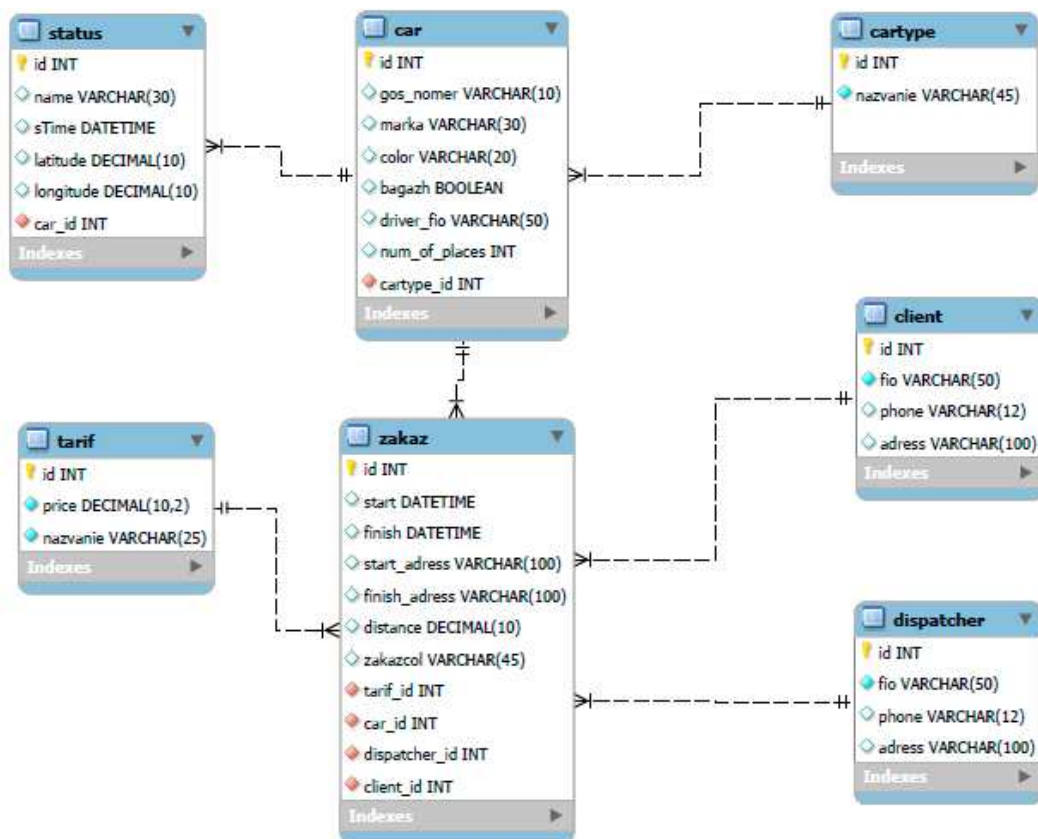


Рис. 2.4. Фізична структура бази даних.

Для створення таблиць були використанні наступні типи даних полів:

- INTEGER – цілі числа з діапазоном значень від -2147483648 до 2147483647;
- BOOLEAN – булево значення 0 –false, 1 – true;
- DECIMAL – дробне число, що зберігається у вигляді рядка;
- DATETIME – комбінація дати та часу;
- VARCHAR (n) – символний рядок, де «n» – кількість символів.

Для зв'язку таблиць використовуються поля «ID» таблиці, на яку відбувається посилання, та відповідне поле таблиці з якої відбувається посилання, яке зазвичай іменується по назві таблиці, на яку посилається, та назві поля цієї таблиці. Наприклад, якщо відбувається зв'язок з таблицею «CARS» по полю «ID», то відповідне поле у таблиці, яку необхідно зв'язати з цією, буде мати назву «CAR_ID». При цьому типи даних цих двох полів повинні бути однаковими.

Після створення структури бази даних за допомогою спеціальних засобів, що надає СУБД MySQL, було створено скрипт для відновлення цієї структури у випадку перевстановлення програмних засобів СУБД або встановленні на іншому сервері. Текст даного скрипту представлено у додатку В.

2.4 Рішення з технічного забезпечення

Функціонування інформаційної системи забезпечується насамперед сукупністю технічних (апаратних) засобів:

- пристрій, що отримує дані з датчиків транспортного засобу та передає їх на сервер за допомогою мережі Інтернет;
- сервер, що забезпечує роботу серверної частини програмного комплексу, який отримує інформацію про транспортні засоби та зберігає їх у базі даних, а також забезпечує можливість підключення до цієї бази даних клієнтів;
- робоча станція (клієнт), що забезпечує функціонування клієнтської частини програмного комплексу, який надає користувачу можливість перегляду отриманих від транспортних засобів та збережених на сервері даних;
- канали зв'язку всіх перелічених вище складових.

Головна частина програмного комплексу підсистеми "Диспетчер" виконується на сервері, яким може бути звичайний персональний комп'ютер з наступними мінімальними характеристиками:

- ЦП з тактовою частотою 2 ГГц;
- ОЗП об'ємом 2 гігабайт;
- ПЗП об'ємом 120 гігабайт;
- Мережева карта 100 Мбіт/с.

Для забезпечення захисту від загублення даних внаслідок нестабільного електропостачання рекомендується використовувати джерело безпе-

ребійного живлення, через яке сервер буде під'єднаний до електромережі. Параметри джерела безперебійного живлення обираються відповідно до потужності комп'ютера, з яким він використовується.

Робоча станція (клієнт) використовується для виконання програми візуалізації даних, отриманих з пристроїв транспортних засобів. Таким технічним засобом може виступати стаціонарний персональний комп'ютер, ноутбук, нетбук тощо. Для функціонування клієнтської частини програмного комплексу інформаційної системи необхідно дотримати наступні мінімальні системні вимоги:

- ЦП з тактовою частотою 1.5 ГГц;
- ОЗП об'ємом 1 гігабайт;
- ПЗП об'ємом 60 гігабайт;
- Мережева карта 100 Мбіт/с.

Всі розглянуті основні вузли обмінюються даними між собою за допомогою каналів зв'язку. У наш час в основному каналами зв'язку є бездротові мережі. Для зв'язку пристроїв, які знаходяться у транспортних засобах, з сервером використовується GPRS-інтернет. Зв'язок робочої станції з сервером може відбуватись за допомогою локальної мережі або за допомогою мережі Інтернет.

Підключення сервера до мережі Інтернет може відбуватись різними шляхами: виділена лінія (використовується дротовий зв'язок), телефонна лінія (ADSL-модем), мобільний зв'язок (3G-модем) тощо.

Для розподілення каналу зв'язку підключення до сервера використовується мережевий концентратор. Це є необхідним лише у тому випадку, коли робочих станцій, яким необхідно отримувати дані по локальній мережі, декілька.

2.5 Рішення з програмного забезпечення

2.5.1 Структура програмного забезпечення

Частини підсистеми "Диспетчер" реалізовано як окремі прикладні програми. Відповідно до описаної у п. 2.1.1 структури були розроблені програми підсистеми візуалізації даних та підсистеми моніторингу даних. Кожна з цих

програм має модульну структуру, тобто поділяється на окремі логічно завершені частини.

Програмний комплекс підсистеми складається з наступних частин:

- «Program» – програма головного модулю;
- «cSQL» – програма модулю доступу до бази даних;
- «cServer» – програма модулю моніторингу та обробки даних;
- «MySql.Data.dll» – динамічна бібліотека, що забезпечує зв'язок з

MySQL.

Програмний комплекс підсистеми візуалізації даних має наступні частини:

- «Program» – програма головного модулю;
- «cSQL» – програма модулю доступу до бази даних;
- «cClient» – програма модулю створення головного вікна;
- «cMap» – програма модулю роботи з картами Google;
- «cLogin» – програма модулю аутентифікації;
- «cEditCars» – програма модулю створення вікна редагування да-

них транспортних засобів;

- «cGraph» – програма модулю побудови графіків;
- «MySql.Data.dll» – динамічна бібліотека, що забезпечує зв'язок з

MySQL.

Програмний код був розроблений за допомогою середовища розробки програмного забезпечення Netbeans який задає свої правила щодо проектування програмного продукту. Кожен з окремих модулів програми може міститись у одному або декількох файлах у залежності від набору його функцій. Якщо однією з функцій модулю є побудова вікна, то при створенні модулю автоматично створюється три файли. Наведемо приклад модулю серверної частини програмного комплексу інформаційної системи («cServer»). Він зберігається у трьох файлах:

- «cServer.java» містить програмний код функцій обробки даних та обробки подій, отриманих в наслідок дій користувача;
- «Design.java» містить програмний код побудови елементів вікна;

- «cServer.xml» містить ресурси вікна у форматі XML.

Всі модулі містять у собі окремий клас, або частину класу, який розподілено між декількох модулів.

2.5.2 Функції частин програмного забезпечення

Далі розглянемо функції модулів підсистем, які описані у п. 2.5.1.

Як можна бачити зі структур обох підсистем, вони мають дві однакові частини – «cSQL» та «MySQL.Data.dll». Ці частини використовуються для підключення до бази даних та обміну з нею даними. Динамічна бібліотека «MySQL.Data.dll» виконує функції доступу до платформи баз даних на низькому рівні. Модуль «cSQL» виконує функції формування SQL-запитів для додавання, редагування та отримання даних таблиць до бази даних, а також приймання відповідей від неї, використовуючи бібліотеку «MySQL.Data.dll».

Також є ще один модуль, який присутній в обох підсистемах – «Program». Це головний модуль програми, який містить функції ініціалізації та підключення основних модулів програми. В ньому реалізовано функцію, яка є головною точкою входу для програми.

Програма підсистеми "Диспетчер" крім описаних вище модулів має тільки один модуль – «cServer». Даний модуль виконує наступні функції:

- прослуховування порту даних мережевого протоколу та формування рядку байт отриманих даних;
- розрахунок та перевірка контрольної суми отриманих даних;
- розбір рядку отриманих даних та виклик функції формування запиту для їх занесення до бази даних;
- контроль зв'язку з кожним транспортним засобом, за яким необхідно здійснювати моніторинг;
- обробка подій отриманих внаслідок дій з іконкою, що відображується на панелі задач під час роботи підсистеми.

Розглянемо функції модулів програми підсистеми візуалізації даних.

Модуль «cClient»:

- формування головного вікна та обробка подій отриманих внаслідок дій користувача над елементами головного вікна;
- формування запиту на отримання інформації з таблиць бази даних;
- оновлення поточних даних по таймеру;
- виклик вікна відображення графіків;
- виклик вікна додавання та редагування даних транспортних засобів.

Модуль «сMap»:

- підключення до сервісу «Google Map» за допомогою наданого його розробниками API;
- формування запитів для відображення об'єктів на карті.

Модуль «сLogin»:

- формування вікна входу до системи;
- перевірка коректності введених даних користувача.

Модуль «сEditCars»:

- формування вікна додавання та редагування даних транспортних засобів, а також обробка подій, отриманих внаслідок дій користувача;
- перевірка коректності ведених даних;
- формування запиту на внесення введеної інформації до бази даних.

Модуль «сGraph»:

- формування вікна відображення графіків;
- формування запиту на отримання даних, необхідних для побудови графіків;
- побудова (малювання) графіків на основі отриманих даних.

3 РОЗРОБКА РОБОЧОЇ ДОКУМЕНТАЦІЇ

Робоча документація – сукупність документів, що містять відомості, необхідні для розробки, виготовлення, супроводу та експлуатації програм. Робоча документація є одним з видів технічної документації.

Комплекс державних стандартів, що встановлюють взаємопов'язані правила розробки, оформлення та обігу програм і програмної документації називається «Єдина система програмної документації» (ЄСПД).

Види документів, що входять до робочої документації, та їх зміст наведені в ГОСТ 19.101-77. Зазвичай використовуються наступні види документів:

- Специфікація – містить інформацію щодо складу програми та документації на неї. Виконується на стадії робочого проекту. Є обов'язковим документом для програмного забезпечення і тих програмних модулів, які можуть мати самостійне застосування;
- Відомість власників оригіналів (код виду документа - 05) – містить перелік підприємств, які зберігають оригінали програмних документів. Виконується на стадії робочого проекту. Необхідність складання документа вирішується на етапі затвердження технічного завдання (за згодою);
- Текст програми (код виду документа - 12) – містить запис програми з необхідними коментарями. Виконується на стадії робочого проекту. Необхідність – за погодженням;
- Опис програми (код виду документа - 13) – містить відомості про логічну структуру та функціонування програми. Виконується на стадії робочого проекту. Необхідність – за погодженням;
- Програма та методика випробувань (код виду документа - 51) – містить вимоги, що підлягають перевірці при випробуваннях програми, а також порядок і методи їх контролю. Виконується на стадії робочого проекту. Необхідність – за погодженням;
- Технічне завдання – містить призначення та область застосування програми, технічні, техніко-економічні та спеціальні вимоги, що ставляться до програми, необхідні стадії і терміни розробки, види випробувань. Ви-

конується на стадії технічного завдання і є обов'язковим документом для комплексу. Необхідність складання технічних завдань на компоненти визначається за погодженням із замовником;

- Записка пояснення (код виду документа - 81) – містить схему алгоритму, загальний опис алгоритму і/або функціонування програми, а також обґрунтування прийнятих технічних та техніко-економічних рішень. Виконується на стадії ескізного та технічного проєктів. Необхідність – за погодженням;

- Експлуатаційні документи – містить відомості для забезпечення функціонування та експлуатації програми. Виконуються на стадії робочого проєкту. Необхідність – за погодженням для кожного документа окремо. Види експлуатаційних документів:

- формуляр (код виду документа - 30) – містить основні характеристики програми, комплектність і відомості про експлуатацію програми;

- опис застосування (код виду документа - 31) – містить відомості про призначення програми, області застосування, застосовуваних методах, класі вирішуваних задач, обмеження для застосування, мінімальної конфігурації технічних засобів;

- керівництво системного програміста (код виду документа - 32) – містить відомості для перевірки, забезпечення функціонування та налаштування програми на умови конкретного застосування;

- керівництво програміста (код виду документа - 32) – містить відомості, необхідні для експлуатації програми;

- керівництво оператора (код виду документа - 33) – містить відомості для забезпечення процедури спілкування оператора з обчислювальною системою в процесі виконання програми;

- опис мови (код виду документа - 35) – містить опис синтаксису і семантики мови програмування;

- керівництво з технічного обслуговування (код виду документа - 46) – містить відомості для застосування тестових і діагностичних програм при обслуговуванні технічних засобів.

Склад застосовуваних видів документів визначається розробником документів залежно від стадій розробки технологічної документації та типу виробництва.

Розробка документації здійснюється за правилами ГОСТ. У таблиці 3.1 наведено номери та опис документів ГОСТ щодо розробки програмно-апаратних комплексів.

Таблиця 3.1

Документ	Опис
ГОСТ 19.001-77	Загальні положення
ГОСТ 19.002-80	Схеми алгоритмів і програм. Правила виконання
ГОСТ 19.003-80	Схеми алгоритмів і програм. Позначення умовні графічні
ГОСТ 19.004-80	Терміни та визначення
ГОСТ 19.005-85	Р-схеми алгоритмів і програм. Позначення умовні графічні та правила виконання
ГОСТ 19.101-77	Види програм і програмних документів
ГОСТ 19.102-77	Стадії розробки
ГОСТ 19.103-77	Позначення програм і програмних документів
ГОСТ 19.104-78	Основні написи
ГОСТ 19 105-78	Загальні вимоги до програмних документів
ГОСТ 19.106-78	Вимоги до програмних документів, виконаним друкованим способом
ГОСТ 19.201-78	Технічне завдання. Вимоги до змісту та оформлення
ГОСТ 19.202-78	Специфікація. Вимоги до змісту та оформлення
ГОСТ 19.301-79	Програма і методика випробувань. Вимоги до змісту та оформлення
ГОСТ 19.401-78	Текст програми. Вимоги до змісту та оформлення
ГОСТ 19.402-78	Опис програми. Вимоги до змісту та оформлення
ГОСТ 19.403-79	Відомість власників оригіналів
ГОСТ 19.404-79	Пояснювальна записка. Вимоги до змісту та оформлення

ГОСТ 19.501-78	Формуляр. Вимоги до змісту та оформлення
ГОСТ 19.502-78	Опис застосування. Вимоги до змісту та оформлення
ГОСТ 19.503-79	Керівництво системного програміста. Вимоги до змісту та оформлення
ГОСТ 19.504-79	Керівництво програміста. Вимоги до змісту та оформлення
ГОСТ 19.505-79	Керівництво оператора. Вимоги до змісту та оформлення
ГОСТ 19.506-79	Опис мови. Вимоги до змісту та оформлення
ГОСТ 19.507-79	Відомість експлуатаційних документів
ГОСТ 19.508-79	Керівництво з технічного обслуговування. Вимоги до змісту та оформлення
ГОСТ 19.601-78	Загальні правила дублювання, обліку та зберігання
ГОСТ 19.602-78	Правила дублювання, обліку та зберігання програмних документів, виконаних друкарським способом
ГОСТ 19.603-78	Загальні правила внесення змін
ГОСТ 19.604-78	Правила внесення змін у програмні документи, виконані друкарським способом
ГОСТ 19.701-90	Схеми алгоритмів, програм, даних і систем. Умовні позначення і правила виконання

Відповідно до приведених документів стандартизації було розроблено декілька документів щодо розробки інформаційної системи, які наведено у додатках:

- Технічне завдання (Додаток А);
- Керівництво користувача (Додаток Б);
- Текст програми підсистеми моніторингу даних (Додаток Г);

4 ОХОРОНА ПРАЦІ

Охорона праці - система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних, лікувально-профілактичних заходів і засобів забезпечення безпеки, збереження здоров'я і працездатності людини в процесі праці.

Як відомо, цілком безпечних і нешкідливих виробництв не існує, тому основною задачею охорони праці є зведення до мінімуму імовірність поразки чи захворювання працюючого з одночасним забезпеченням комфорту при максимальній продуктивності праці.

Комплекс організаційних і технічних заходів і засобів, що запобігають нещасні випадки, що виникають при впливі на працюючого небезпечних виробничих факторів, називається технікою безпеки. Виробнича санітарія містить у собі комплекс організаційних, гігієнічних і санітарно-технічних заходів, що запобігають виробничі шкідливості.

Турбота про охорону праці являється справою державної важливості. 14 жовтня 1992 року був прийнятий Закон України "Про охорону праці", у якому закріплені найважливіші положення в області охорони праці, і який поширюється на всіх суб'єктів підприємницької діяльності незалежно від форм власності. Основними пунктами в цьому законі є: стаття про гарантію прав громадян на охорону праці, стаття про відшкодування моральної шкоди, статті про охорону праці жінок, неповнолітніх, інвалідів.

У Законі "Про охорону праці" зазначені органи, покликані здійснювати нагляд і контроль за дотриманням законодавства про працю і правил по охороні праці. До них відносяться спеціально уповноважені на те державні органи й інспекції, що не залежать у своїй діяльності від адміністрації підприємств, установ, організацій і їхніх вищих органів.

Державне керування органами охорони праці здійснюють: Кабінет Міністрів України, державний комітет України по нагляду за охороною праці, міністерство й інші централізовані органи, місцеві ради народних депутатів і їхні виконавчі комітети. Державний нагляд за дотриманням законодавства й інших нормативних актів по охороні праці здійснюють: державний комітет

України по нагляду за охороною праці, державний комітет України по атомній і радіаційній безпеці, органи державного пожежного нагляду, органи і служби санітарно епідемічного нагляду. А трудові колективи в особі своїх уповноважених і профспілки здійснюють суспільний контроль за охороною праці.

Вищий нагляд за точним виконанням законів про працю міністерствами і відомствами, підприємствами, установами, громадянами і посадовими особами покладений на Генерального Прокурора України.

Велике значення в створенні здорових і безпечних умов праці має стандартизація. Вона дозволяє вжити діючих заходів по підвищенню технічного рівня й упорядкуванню розробки нормативно-технічної документації по безпеці праці. З 1972 року створюється Система стандартів по безпеці праці (ССБТ), що являє собою комплекс великого числа взаємозалежних стандартів, спрямованих на забезпечення безпеки праці. Ця система встановлює загальні вимоги і норми по видах небезпечних і шкідливих виробничих факторів, загальні вимоги безпеки до виробничого устаткування і процесів, вимоги до засобів захисту працюючих, методи оцінки безпеки праці.

4.1 Аналіз небезпечних і шкідливих виробничих факторів, характерних для приміщення лабораторії

Реальні виробничі умови характеризуються, як правило, наявністю деяких небезпечних і шкідливих виробничих факторів .

До небезпечних виробничих факторів відносяться фактори, вплив яких приводить до травми, фактори, що приводять до професійного захворювання працюючого. По природній дії небезпечні і шкідливі фактори за Дст 120.003-74 діляться на:

- фізичні;
- хімічні;
- біологічні;
- психологічні.

Впровадження обчислювальної техніки в усі сфери народного господарства привело до значного розширення парку ПЕОМ. У зв'язку з цим вини-

кла проблема організації робочих місць користувачів ПЕОМ відповідно до вимог нормативів охорони праці.

Робота з персональним комп'ютером зв'язана з низкою шкідливих і небезпечних факторів, що можуть впливати на організм людини.

За даними Всесвітньої організації охорони здоров'я, професійна діяльність зв'язана з роботою на ПЕОМ може в окремих випадках приводити до порушень з боку зорового аналізатора, кістково-м'язовим (змушена поза) порушенням, нервово-емоційною напругою при роботі, шкірним захворюванням і ін. Високий відсоток (92%) складають скарги на зорове стомлення. Дослідження показали, що професія користувача ЕОМ являє собою модель розумової праці, що виконується в одноманітній позі в умовах обмеження загальної м'язової активності при рухливості рук, при високій напрузі зорових функцій і нервово-емоційній напрузі в умовах впливу різноманітних фізичних факторів. Дослідження умов праці показали, що на більшості підприємств користувачі ЕОМ працюють при несприятливому мікрокліматі, зниженій вологості повітря, нераціональному освітленні. Підвищений шум на робочому місці, унаслідок роботи комп'ютерів і принтерів, шкідливо впливає на органи слуху і на весь організм людини через центральну нервову систему, викликаючи швидку стомлюваність, головний біль, безсоння.

Під час роботи за персональним комп'ютером, організм людини піддається електромагнітному випромінюванню завдяки тому, що переважна більшість моніторів мають основний пристрій для відображення - електронно-променеву трубку (ЕПТ), хоча більшість сучасних моніторів задовольняють міжнародному стандарту MPR II, чи навіть ще більш строгому шведському стандарту TCO-92, що регламентують низькочастотне електромагнітне випромінювання, випромінюване моніторами на ЕПТ.

Вплив магнітних полів на людину, що працює з персональним комп'ютером, залежить від режиму і характеру роботи, від конструктивних особливостей випромінюючих пристроїв (в основному від моніторів), розташування робочого місця, технічного стану устаткування.

Спектр випромінювання монітора характеризується наявністю рентгенівських, ультрафіолетових, інфрачервоних і інших електромагнітних коливань. Небезпека рентгенівських частини і інших випромінювань більшістю учених визнається пренебрежливо малою, їхній рівень досить невеликий і майже цілком поглинається речовиною екрана. Найбільш важка ситуація зв'язана з полями випромінювань дуже низьких частот, що здатні викликати біологічні ефекти при впливі на живі організми. Виявлено, що електромагнітні полюси з частотою 60 Гц можуть ініціювати біологічні зрушення (аж до порушення синтезу ДНК) у клітках людини. Небезпека їхнього впливу не слабкає при зниженні інтенсивності, деякі полюси діють на клітки тіла тільки при малих інтенсивностях чи на конкретних частотах. Американські вчені пояснили цей факт тим, що перемінне електромагнітне поле, що робить коливання з частотою 60 Гц, втягує в аналогічні коливання молекули будь-якого типу, незалежно від того, чи знаходяться вони в тілі чи людини в його мозку. Унаслідок цього змінюється активність ферментів і клітинного імунітету, подібні процеси відбуваються при виникненні пухлин.

Технічні характеристики дисплеїв (розрішаюча здатність, яскравість, контрастність, частота відновлення чи мелькання) у випадку, якщо на них не звертають уваги при виборі пристрою чи неправильно встановлюють, можуть у край негативно позначатися на зорі, викликати астенію, чи швидку стомлюваність очей, запалення очей, головні болі і двоїння в очах.

У зв'язку з використанням клавіатури можуть виникнути захворювання рук, спини, пліч і шиї.

Отже, підводячи підсумок, при роботі із ЕОМ можна виділити наступні фактори:

фізичні - низькочастотне електромагнітне випромінювання моніторів, яскраві відблиски на екрані, підвищена яскравість світла монітора, недостатня освітленість робочого місця, запиленість повітря поблизу екрана дисплея через його електризацію, підвищення рівня шуму, підвищена напруженість електричних і магнітних полів, підвищення рівня статичної електрики, небезпека поразки електричним струмом;

психофізіологічні - тривалі статичні навантаження м'язів рук, шиї, спини, зап'ясть, пліч, ніг, постійна напруга зору, нервово-емоційні перевантаження (монотонність виконуваних дій, розумова перенапруга).

Установлено наступні санітарні норми і правила :

- мікроклімат у робочій зоні приміщення (до 2 м над рівнем підлоги) для легкої роботи (витрати енергії до 150 ккал/година);
- температура 17-22 С;
- відносна вологість не більш 75%;
- швидкість руху повітря не більш 0.3 м/с.;
- припустимі коефіцієнти пульсації штучного освітлення в залежності від системи освітлення і характеру виконуваної роботи не повинні перевищувати 10-20%;
- освітленість при комбінованому освітленні 150-400 лк, при загальному освітленні - 200 лк, коефіцієнт запасу - 1.5 (для люмінесцентних ламп);
- рівень шуму - 50 дБА в кімнаті програмістів і до 65 дБА в машинному залі;
- гранично припустима напруженість електричної складової електромагнітного поля 10 В/м, магнітної складовий - 0.3 А/м;
- гранично припустима напруженість електростатичного поля при тривалому впливі на людину на робочому місці (до 1 години) - 60 кв/м.

4.2 Розрахунок системи освітлення для приміщення лабораторії

Рациональне освітлення приміщень - один з найбільш важливих факторів, від яких залежить ефективність трудової діяльності людини. Гарне освітлення необхідне для виконання більшості задач оператора. Для того, щоб спланувати раціональну систему освітлення, враховується специфіка робочого завдання, для якого створюється система освітлення, швидкість і точність, з яким це робоче завдання повинне виконуватися, тривалість його виконання і різні зміни в умовах виконання робочих операцій.

Розрахунок освітленості будемо робити для приміщення, що має наступні характеристики:

- довжина приміщення 16 м;

- ширина приміщення 6 м;
- висота 4 м;
- число вікон 3;
- кількість робочих місць 3.

У приміщенні, де знаходиться робоче місце оператора використовується змішане освітлення, тобто сполучення природного і штучного освітлення. У якості природного - бічне освітлення через вікна. Штучне освітлення використовується при недостатнім природному освітленні. У даному приміщенні використовується загальне штучне освітлення.

Розрахунок його здійснюється по методу світлового потоку з урахуванням потоку, відбитого від стін і стелі. Нормами для даних робіт установлена необхідна освітленість робочого місця $E_H=300\text{лк}$ (середня точність роботи з розрізнення деталей розміром від 1 до 10 мм).

Загальний світловий потік визначається по формулі:

$$F_{\text{общ}} = \frac{E_H \times S \times z_1 \times z_2}{V}, (1)$$

де E_H - нормована освітленість ($E_H=300\text{лк}$);

S - площа приміщення;

Z_1 - коефіцієнт, що враховує старіння ламп і забруднення світильників ($z_1=1.5$);

Z_2 - коефіцієнт, що враховує нерівномірність висвітлення приміщення ($z_2=1.1$);

V - коефіцієнт використання світлового потоку; визначається в залежності від коефіцієнтів відображення від стін, стелі, робочих поверхонь, типів світильників і геометрії приміщення.

Площа приміщення: $S = A \times B = 16 \times 6 = 96 \text{ м}^2$.

Виберемо з таблиці коефіцієнт використання світлового потоку по наступним даним:

- коефіцієнт відображення побіленої стелі $R_{\text{П}}=70\%$;
- коефіцієнт відображення від стін, пофарбованих у світлу фарбу $R_{\text{СТ}}=50\%$;

- коефіцієнт відображення від підлоги, покритого лінолеумом темного кольору $R_p=10\%$;

- індекс приміщення:

$$i = \frac{A \times B}{h \times (A + B)} = \frac{16 \times 6}{4 \times (16 + 6)} = 1,1 \quad (2)$$

Знайдений коефіцієнт $V=0,34$.

По формулі визначаємо загальний світловий потік:

$$F_{\text{общ}} = \frac{300 \times 96 \times 1,1 \times 1,5}{0,34} = 139764 \text{ лм.} \quad (3)$$

Для організації загального штучного освітлення виберемо лампи типу ЛБ40. Люмінесцентні лампи мають ряд переваг перед лампами накаливання: їхній спектр ближче до природного; вони мають велику економічність (більше світловіддача) і термін служби (у 10-12 разів). Поряд з цим мають і недоліки: їхня робота супроводжується іноді шумом; гірше працюють при низьких температурах; їх не можна застосовувати у вибухонебезпечних приміщеннях; мають малу інерційність.

Для нашого приміщення люмінесцентні лампи підходять.

Світловий потік однієї лампи ЛБ40 складає не менш $F_L=2810$ лм. Число N ламп, необхідних для організації загального освітлення визначається по формулі:

$$N = \frac{F_{\text{общ}}}{F_L} = \frac{139764}{2810} = 50 \quad (4)$$

Як світильники вибираємо ПВЛ-1, 2x40 Вт.

Таким чином, щоб забезпечити світловий потік $F_{\text{общ}}=139764$ лм треба використовувати 25 світильників по 2 лампи ЛБ40 у кожному.

Електрична потужність однієї лампи ЛБ40 $W_L=40$ Вт.

Потужність всієї освітлювальної системи

$$W_{\text{общ}} = W_L * N = 40 * 50 = 2000 \text{ Вт.}$$

4.3 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером

Сьогодні фахівці в області ергономіки вже зрозуміли, що не можна знайти ідеальне положення, у якому можна перебувати і працювати протягом усього робочого дня. Для більшості людей комфортабельним робочим місцем повинне бути таке, котре можна пристосувати не менш чим для двох позицій, при цьому положення крісла, монітора повинні щораз відповідати виконуваній роботі і звичкам. Багато хто вважають, що для роботи на комп'ютері більше підходять вертикальне і злегка похиле положення. Можливо, щоб крісло було злегка нахилене вперед.

Дисплей. Положення тіла звичайно відповідає напрямку погляду; дисплеї, розташовані занадто низько чи під неправильним кутом, є основними причинами сутулості. Відстань від дисплея до очей може варіюватися в залежності від характеру виконуваної роботи - 40-70 см. При роботі з текстом відстань від екрана до очей повинна лише небагато перевищувати відстань між книгою й очима і складати 40-45 см. Необхідно давати відпочинок очам і час від часу їх просто закривати. Не можна допускати, щоб очі увесь час були сфокусовані на одній відстані, працюючи з текстом, рекомендується установлювати великий шрифт. Дуже важливо, щоб екран монітора не мерехтів. Частота регенерації зображення повинна бути не менш 72 Гц, тому що при більш низькій частоті помітне мерехтіння, хоча люди з особливо чутливим зором можуть помітити його і при більш високій частоті. Їм доведеться підібрати графічну плату і монітор з частотою регенерації 85 Гц і вище.

Рекомендується встановлювати на екран монітора спеціальні скляні поляризаційні фільтри з заземленням. Необхідно уникати того, щоб термінал був звернений екраном убік вікна, оскільки інтенсивна освітленість області зору може затопити потоками світла очі і розмити зображення на сітківці. Якщо приходить сидіти поруч з вікном, то треба розташуватися під прямим кутом до нього, причому екран дисплея повинний бути перпендикулярний шибці - цим виключаються відблиски на екрані.

Для зниження впливу низькочастотного електромагнітного випромінювання монітора на ЕЛТ рекомендується вибирати монітор, що задовольняє стандарту MPR II, чи TCO 92-95. У протилежному випадку необхідно установи-

ти на екран захисний фільтр, найбільш сучасні моделі, затримують до 99 % електромагнітного випромінювання.

Форма спинки крісла повинна повторювати форму спини працюючого. Крісло необхідно установити на такій висоті, щоб не почувався тиск на куприк (крісло розташоване занадто низько) на стегна (крісло розташоване занадто високо). Фахівці з ергономіки, вважали, що кут між стегнами і хребтом повинний скласти 90°, однак недавно проведені дослідження показали, що більшість людей переважно сидять трохи відкинувшись.

Клавіатуру необхідно установити так, щоб не треба було до неї тягтися і нахилитися вперед. При зміні положення тіла (наприклад, з вертикального на похиле) обов'язково треба перемінити положення клавіатури і дисплея. Може виявитися корисною регульована підставка клавіатури, завдяки якій можна без усякої напруги працювати з маніпулятором "миша". Можна поставити клавіатуру і на коліна. Руки при роботі повинні бути прямими в зап'ястях і зігнуті в ліктях приблизно під прямим кутом. Пальці також повинні бути злегка зігнуті. Удари по клавішах не повинні бути занадто сильними.

Зручна висота столу особливо важлива в тому випадку, коли на ньому розташовується клавіатура. Якщо в клавіатури немає підставки, а висоту столу не можна змінити (і він занадто високий), то треба вище підняти сидіння крісла, а під ноги підставити лавочку: чи що-небудь інше. Якщо стіл занадто низький, необхідно підкласти що-небудь під його ніжки.

Якщо при роботі часто приходиться дивитися на документи, треба установити підставку з оригіналом документа в одній площині і на одній, висоті з екраном. Якщо треба частіше дивитися на оригінал, чим на екран, необхідно повернути крісло й екран таким чином, щоб оригінал розміщався прямо перед очима, а комп'ютер ледве збоку.

Щогодини необхідно робити перерву в роботі. У цей час рекомендується робити вправи для зап'ясть. Нижче описаний можливий комплекс вправ.

- Покласти руку на край столу долонею вниз. Взявшись, за пальці, іншою рукою відвести кисть назад і утримувати протягом 5 секунд. Повтори-

ти для іншої руки.

- Злегка упертися рукою в стіл і на 5 секунд напружити пальці в зап'ястя. Повторити іншою рукою.

- Сильно зжати пальці , а потім розпрямити їх. Виконання приведених рекомендацій дозволяє скоротити вплив шкідливих, виробничих: факторів на організм людини.

ВИСНОВКИ

В результаті виконання даної дипломної роботи була розроблена підсистема "Диспетчер" інформаційної системи служби таксі. Для цього була досліджена існуюча організаційна та інформаційна структура підприємства "Чайка" м.Одеса. Знайдені недоліки в існуючій системі, та запропонована концепція та модель нової системи.

Було розроблене технічне завдання та проектні рішення які включають:

- рішення з інформаційного забезпечення;
- рішення з організаційного забезпечення;
- рішення з програмного забезпечення;
- рішення з технічного забезпечення.

У рамках цих проектних рішень був інтегрований апаратний GPS\GSM модуль на транспортний засіб, розроблене програмне забезпечення і технічна документація до інформаційної системи.

Обраний GPS\GSM модуль дозволяє додатково підключити датчики, які дозволяють отримати додаткову інформацію про транспортний засіб.

Дану інформаційну систему можна використовувати як частину протигінних заходів власного автомобіля. Якщо транспортний засіб було викрадено, на протязі деякого часу можна визначити його місце розташування.

Крім того було виконано дослідження з охорони праці.

З урахуванням усього вищенаведеного можна сказати, що цілі, поставлені на початку даної дипломної роботи були повністю виконані.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

- 1 Методичні вказівки до виконання дипломного проектування зі спеціальності 7.080401 "Інформаційні управляючі системи та технології" / К.В. Кошкін, С.Б.Приходько, С.В. Суслов, В.І. Тимофєєв. – Миколаїв: УДМТУ, 2002. – 32 с.
- 2 Гарсиа-Молина, Г., Системы баз данных. Полный курс / Ульман Д. Д., Уидом Д., - М.: «Вильямс», 2004. – 1083 с.
- 3 Михайлюк В.О., Ізотов В.І., Штейн П.В. Методичні вказівки до виконання розділу з питань безпеки життєдіяльності та цивільної оборони у дипломних роботах (проектах) студентів. – Миколаїв: НУК, 2008. – 56 с.
- 4 ГОСТ 12.0.003-74. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация. – Введ 01.01.1976.
- 5 Виробниче освітлення та його розрахунок: Навчальний посібник. – Миколаїв: УДМТУ, 2001. – 84 с.
- 6 Кириллов, В.В., Основы проектирования реляционных баз данных. Учебное пособие/ Кириллов, В.В. - СПб.: ИТМО, 1994. - 90 с.
- 7 Петрова, А.М., GPS: Все, что Вы хотели знать, но боялись спросить. Мобильные коммуникации. / Петрова, А.М. - М.: Бук-Пресс и К, 2005. – 352с.

ДОДАТОК А. Технічне завдання

1. Загальні відомості

1.1. Найменування системи

Повне найменування системи: Інформаційна система управління та моніторингу наземного транспорту.

Скорочене найменування системи: ObjectTracker

Далі, в технічному завданні (ТЗ), до системи можуть застосовуватись наступні назви та терміни: система, інформаційна система, ObjectTracker.

1.2. Підстави для проведення робіт

Завдання на дипломний проект.

1.3. Планові строки початку та закінчення робіт

Плановий строк початку робіт: 01.11.2012

Плановий строк закінчення робіт: 30.12.2012

Строк закінчення робіт може бути подовжений у випадку внесення змін до технічного завдання на розробку.

2. Призначення та мета створення системи

2.1. Призначення системи

Інформаційна система призначена для відстеження місцезнаходження та стану наземного транспорту, який належить підприємству, що використовує дану систему. Система призначена насамперед для спеціалістів, що займаються логістикою.

2.2. Мета створення системи

Метою створення системи є автоматизація процесу моніторингу наземного транспорту, що дозволяє оперативно отримувати всю необхідну інформацію про транспортні засоби, які належать підприємству. Можливість отримувати таку інформацію оперативно надає змогу вчасного коректування маршруту спеціалістом-логістом, що може значно зменшити витрати часу та коштів підприємства.

3. Характеристика об'єктів автоматизації

Об'єктами автоматизації є транспортні засоби підприємства, з яких отримуються дані про їх місцезнаходження та стан важливих показників.

Кожен транспортний засіб обладнується спеціальним пристроєм, до якого підключаються необхідні датчики контролю та інші необхідні компоненти, інформація про які буде передаватись на комп'ютер спостерігача:

- зовнішнє живлення;
- датчик рівня палива;
- карта пам'яті;
- аварійна сигналізація;
- карта мобільного оператора.

Також цей пристрій має батарею для автономного живлення, кнопку виклику та GPS-модуль. Для передачі даних використовується GPRS-інтернет, до якого пристрій підключається після необхідних налаштувань автоматично.

Кожний пристрій має можливість програмування індивідуального номеру транспортного засобу для його ідентифікації при отриманні даних спостерігачем.

4. Вимоги до програмного забезпечення

4.1. Загальні вимоги

Для виконання поставленої задачі необхідно розробити програмне забезпечення, що буде надавати змогу безперервного моніторингу даних, які надходять від транспортних засобів.

Також необхідно створити зручний інтерфейс користувача, за допомогою якого спеціалісти підприємства, яким необхідно отримувати інформацію щодо транспортних засобів, мали б змогу переглядати ці дані, а також вносити зміни в інформацію про транспортні засоби (додавати нові та редагувати існуючі).

Розроблене програмне забезпечення повинно мати змогу забезпечити одночасну роботу декількох користувачів незалежно від їх місцезнаходження.

Програмний продукт повинен мати захист від несанкціонованого доступу та розподілення прав доступу користувачів на дві групи: адміністратори та оператори. Останні не повинні мати прав на здійснення змін та мають обмежені права доступу до даних.

Програмний продукт повинен поставлятись у вигляді прикладної програми для виконання під управлінням операційної системи Microsoft Windows.

4.2. Вимоги до функцій, що повинні виконуватись системою

Система повинна виконувати наступні функції:

- здійснення безперервного моніторингу даних, що надходять від пристроїв, встановлених у транспортних засобах;
- здійснення перевірки коректності надісланих даних;
- збереження прийнятої інформації у банку даних;
- при виникненні помилок чи важливих подій, інформація про них повинна бути занесена до окремого журналу (такими є загублення зв'язку з пристроєм та увімкнення двигуна транспортного засобу);
- здійснення захисту від несанкціонованого входу в систему (надання можливості введення імені та паролю користувача, перевірка введених даних);
- надання можливості користувачам переглядати отриману інформацію;
- місцезнаходження транспортних засобів повинно бути відображено на карті, яку можна при необхідності масштабувати;
- надання можливості користувачам з правами адміністратора робити зміни в інформацію про транспортні засоби;
- побудова графіків швидкості та витрати пального за встановлений користувачем проміжок часу;
- відображення маршруту руху транспортного засобу за встановлений користувачем проміжок часу.

5. Склад та зміст робіт по створенню системи

Роботи по створенню системи виконуються в три етапи:

- Проектування, узгодження функції, що виконуються системою (тривалість – 1 тиждень)
- Розробка програмного коду (тривалість – 4 тижня)
- Тестування та відладка системи (тривалість – 3 тижня)

6. Порядок контролю та приймання системи

6.1. Вимоги до випробувань

Для перевірки роботи розробленої системи необхідно мати встановлений у транспортний засіб пристрій з картою мобільного оператора з налаштованим підключенням до мережі Інтернет, комп'ютер з налаштованою програмою, яка приймає дані, зберігає у базі даних, та програми, яка відображує ці дані.

До бази даних вводиться інформація про транспортний засіб (головне – ідентифікатор), на якому встановлено пристрій, до якого також заноситься ідентичний ідентифікатор. Двигун транспортного засобу вимкнено.

При вході у програму повинно бути відображено на карті місцезнаходження транспортного засобу, яке відомо під час випробовувань.

Далі необхідно змінити місцезнаходження транспортного засобу шляхом його самостійного пересування. При цьому у журналу подій повинний з'явитись запис про увімкнення двигуна та змінитись положення, що відображено на карті. Якщо побудувати графік, повинна відображатись крива зміни швидкості під час руху, а також крива зміни рівня пального. При побудові маршруту руху повинно бути відображено на карті лінію пройденого транспортним засобом шляху з початкової до кінцевої точки.

6.2. Загальні вимоги до приймання робіт

Система може вважатись прийнятою при відсутності виключних ситуацій під час заключної стадії випробувань.

7. Вимоги до документування

На момент приймання роботи повинна бути створена інструкція користувача, яка матиме наступну інформацію:

- загальний опис системи;
- вимоги до технічних засобів для використання програмного забезпечення системи;
- опис функцій, які виконує система;
- опис інтерфейсу користувача;
- порядок встановлення та налаштувань програмного забезпечення.

ДОДАТОК Б. Інструкція користувача

1. Загальний опис системи

Інформаційна система управління та моніторингу наземного транспорту призначена для відстеження місцезнаходження транспорту підприємства та спостереження за показниками датчиків, що розміщуються у цих транспортних засобах (наприклад, датчик рівня пального, спідометр тощо). Інтерфейс користувача інформаційної системи надає зручні функції моніторингу такі як побудова маршрутів руху транспортних засобів, побудова графіків швидкості та використання пального.

Програмне забезпечення інформаційної системи поставляється у вигляді двох окремих частин: серверна (програма моніторингу даних) та клієнтська (програма візуалізації даних).

Програмний комплекс використовує для збереження інформації СУБД MySQL.

Для функціонування серверної частини програмного комплексу інформаційної системи необхідно виконати певні системні вимоги до персонального комп'ютеру, на якому вона виконується:

- ЦП з тактовою частотою 2 ГГц;
- ОЗП об'ємом 2 Гігабайт;
- ПЗП об'ємом 120 Гігабайт;
- Мережева карта 100 Мбіт/с;
- Операційна система MS Windows версії 6 та вище;
- Підключення до мережі Інтернет.

2. Встановлення програмного забезпечення системи

Встановлення інформаційної системи поділяється на два етапи. Першим етапом є встановлення СУБД MySQL, яка виконує в системі роль сховища інформації. Для встановлення даного програмного продукту необхідно запустити файл програми інсталяції «mysql-5.5.23-win32.exe», який йде у комплекті поставки разом з іншими файлами інформаційної системи.

Після запуску програми інсталяції СУБД MySQL з'явиться вікно привітання, яке можна бачити на рис. 2.1.

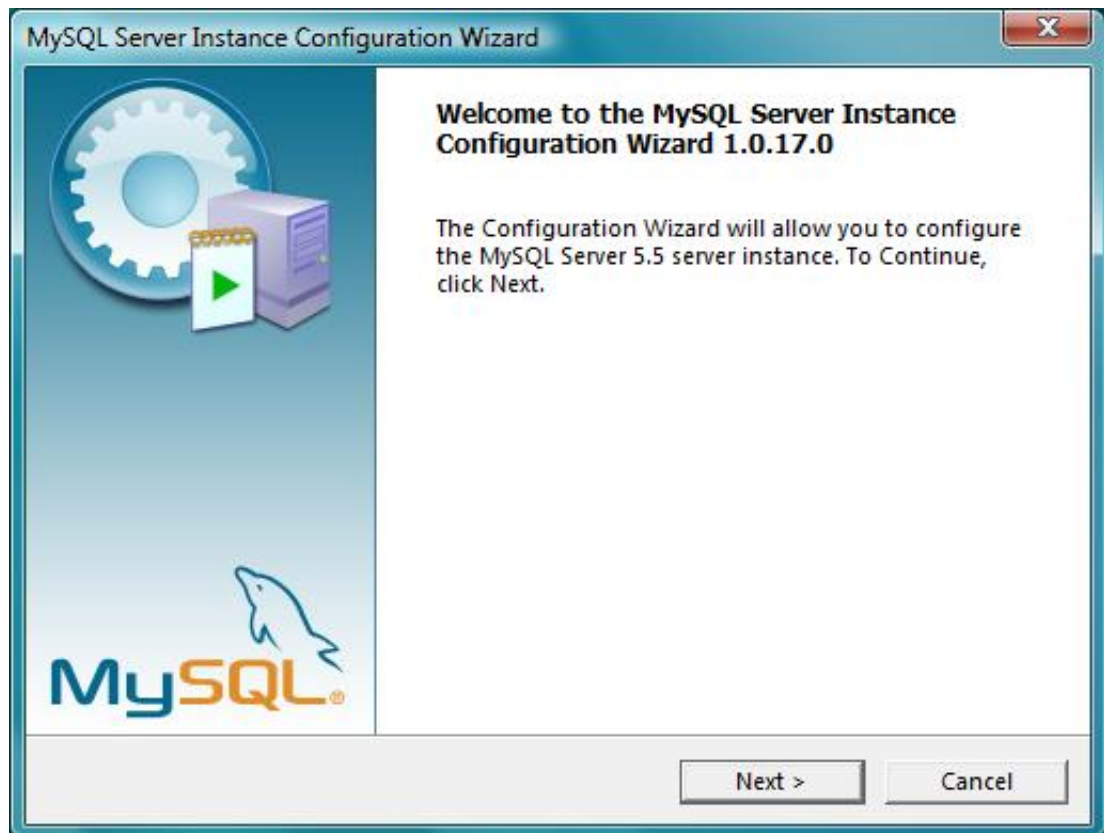


Рис. 2.1. Вікно привітання інсталятора СУБД MySQL.

Натискаючи кнопку «Next >» запускаємо процес інсталяції СУБД. Після встановлення даного програмного продукту автоматично запуститься майстер налаштувань сервера баз даних MySQL. Натискаємо також кнопку «Next >» доки не відкриється сторінка параметрів, яка зображена на рис. 2.2.



Рис. 2.2. Вікно налаштувань параметрів MySQL.

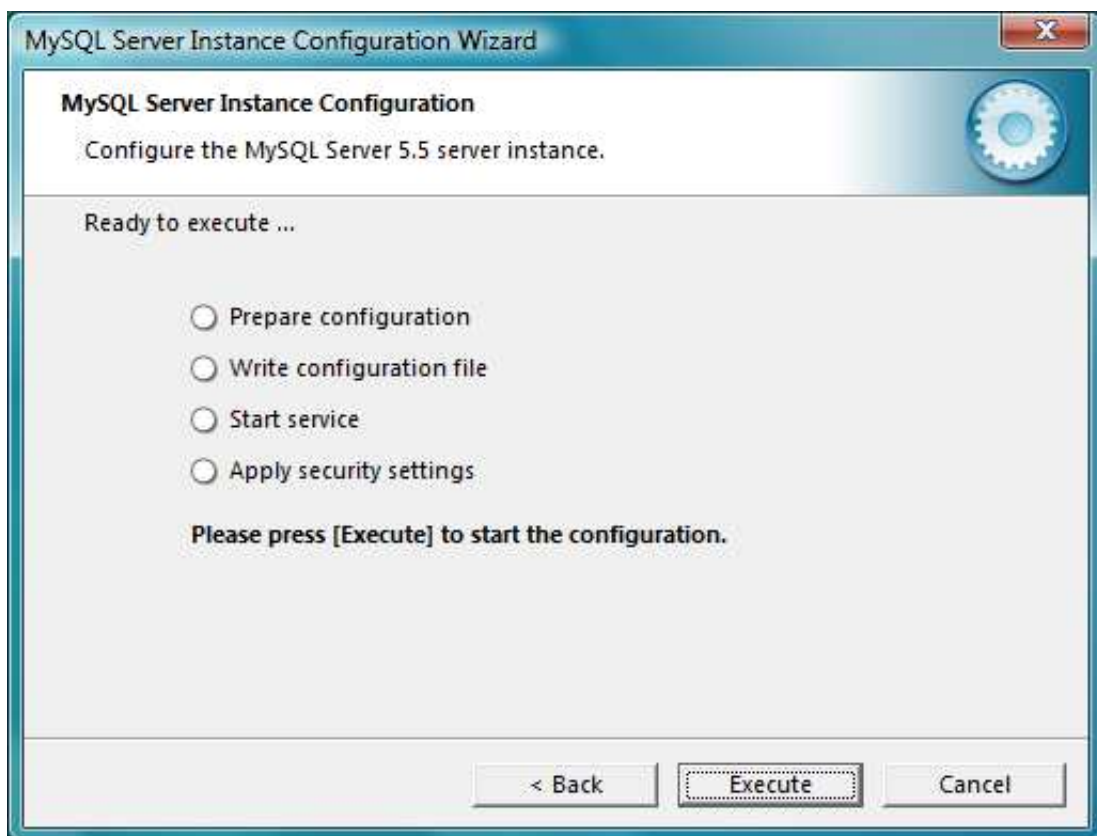


Рис. 2.3. Підсумкове вікно інсталятора СУБД Firebird.

На цій сторінці необхідно зробити налаштування так, як показано на рис 2.2: у поля вводу «New root password» та «Confirm» ввести пароль «masterkey», а також відмітити «Enable root access from remote machines».

Після зроблених змін треба натиснути кнопку «Next >», щоб перейти до підсумкової сторінки, яка показана на рис. 2.3, на якій необхідно натиснути кнопку «Execute» для застосування налаштувань. По завершенню налаштувань серверу баз даних майстер покаже фінальне вікно, яке можна закрити та починати працювати з СУБД.

Для створення бази даних інформаційної системи необхідно виконати SQL-скрипт з файлу, який також входить у комплект поставки інформаційної системи, та має назву «ObjTrack.sql». Для цього можна використати засоби, які надає встановлений продукт MySQL. Необхідно завантажити консоль СУБД та виконати запит, що міститься у файлі «ObjTrack.sql». Як результат виконання запиту буде створено таблиці бази даних, з якими працює серверна та клієнтська частина програмного забезпечення системи.

Встановлення серверної та клієнтської програми відбувається звичайним копіюванням каталогу «OT_Server» та «OT_Client» комплекту поставки програмного забезпечення на ПЗП сервера та клієнта відповідно. Для постійного запуску програми при включенні сервера необхідно занести ярлик на файл «OT_Server.exe» у папку автозапуску операційної системи. По завершенню встановлення серверної програми та серверу баз даних необхідно перезавантажити сервер. Після завантаження операційної системи на сервері можна буде бачити на панелі задач іконку програми моніторингу даних, що зображено на рис. 2.4.

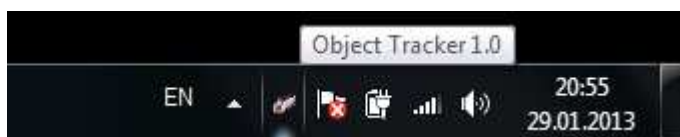


Рис. 2.4. Відображення іконки програми на панелі задач.

3. Запуск клієнтської програми. Вікно входу в систему.

Для початку роботи з інформаційною системою необхідно увійти в папку, яку було скопійовано при встановленні, та завантажити файл «OT_Client.exe».

На першому етапі запуску системи відображується вікно входу до системи, яке показано на рис. 3.1.

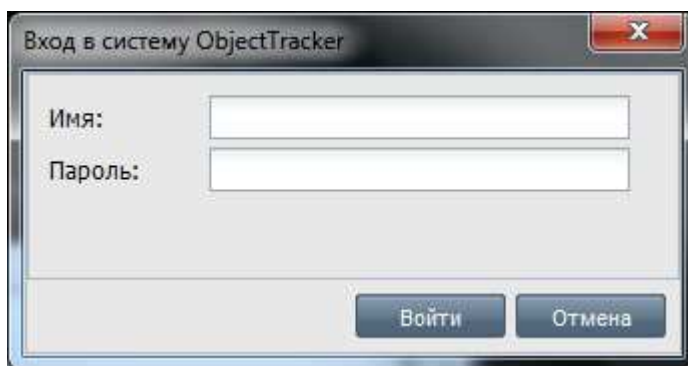


Рис. 3.1. Вікно входу до системи.

В даному вікні необхідно обрати ім'я користувача системи та вказати його пароль у відповідному полі для вводу, після чого натиснути кнопку «ОК». Якщо пароль обраного користувача було введено правильно, дане вікно зачиниться та відкриється головне вікно системи. В противному випадку буде видано повідомлення про невірно введений пароль, та необхідно буде повторити введення паролю.

4. Головне вікно клієнтської програми

Після успішного входу до системи з'являється головне вікно, яке показано на рис. 4.1.

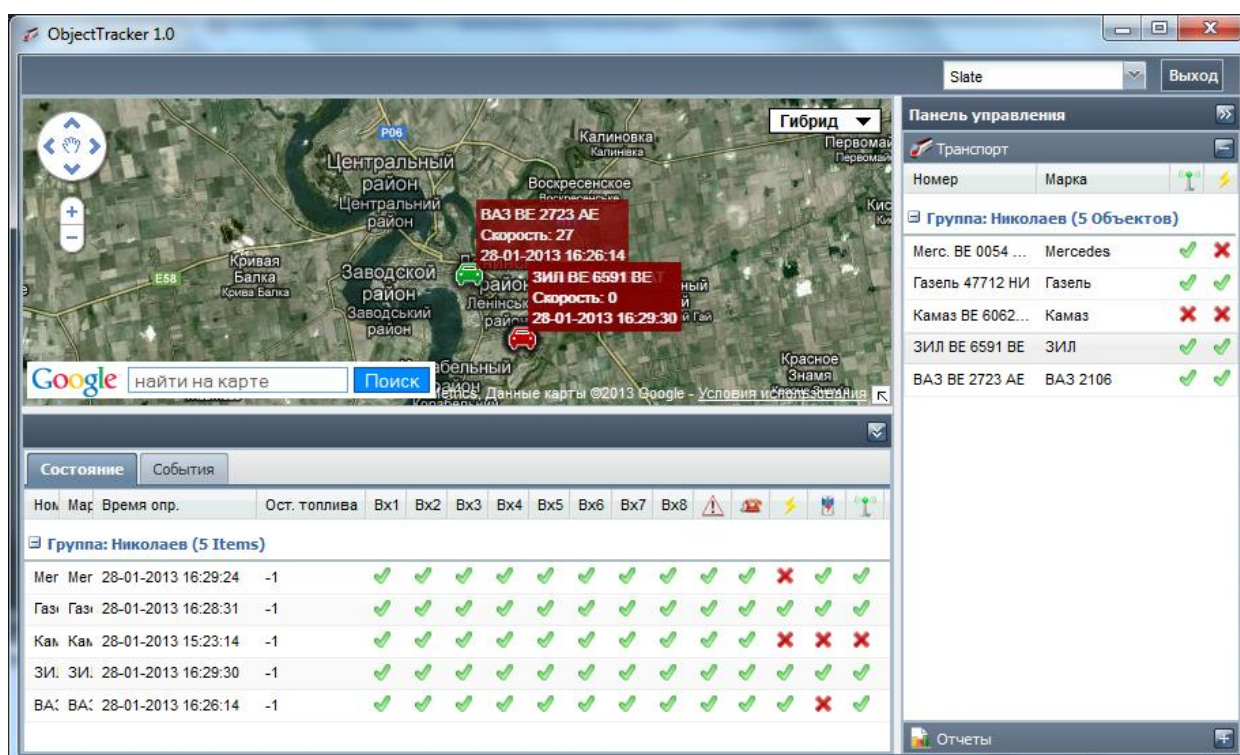


Рис. 4.1. Головне вікно системи.

Головне вікно містить наступні елементи:

- кнопку для виходу з програми;
- випадаючий список кольорових тем інтерфейсу;
- область відображення карти;
- панель журналів (під картою);
- бокову панель управління;

При натисканні на кнопку виходу з програми головне вікно закривається та відбувається завершення роботи клієнтської програми.

Для налаштування кольорової теми інтерфейсу необхідно натиснути на кнопку праворуч від назви теми випадаючого списку та у показаному переліку тем натиснути на одну з них. На рис. 4.2. показано змінення кольорової теми інтерфейсу.

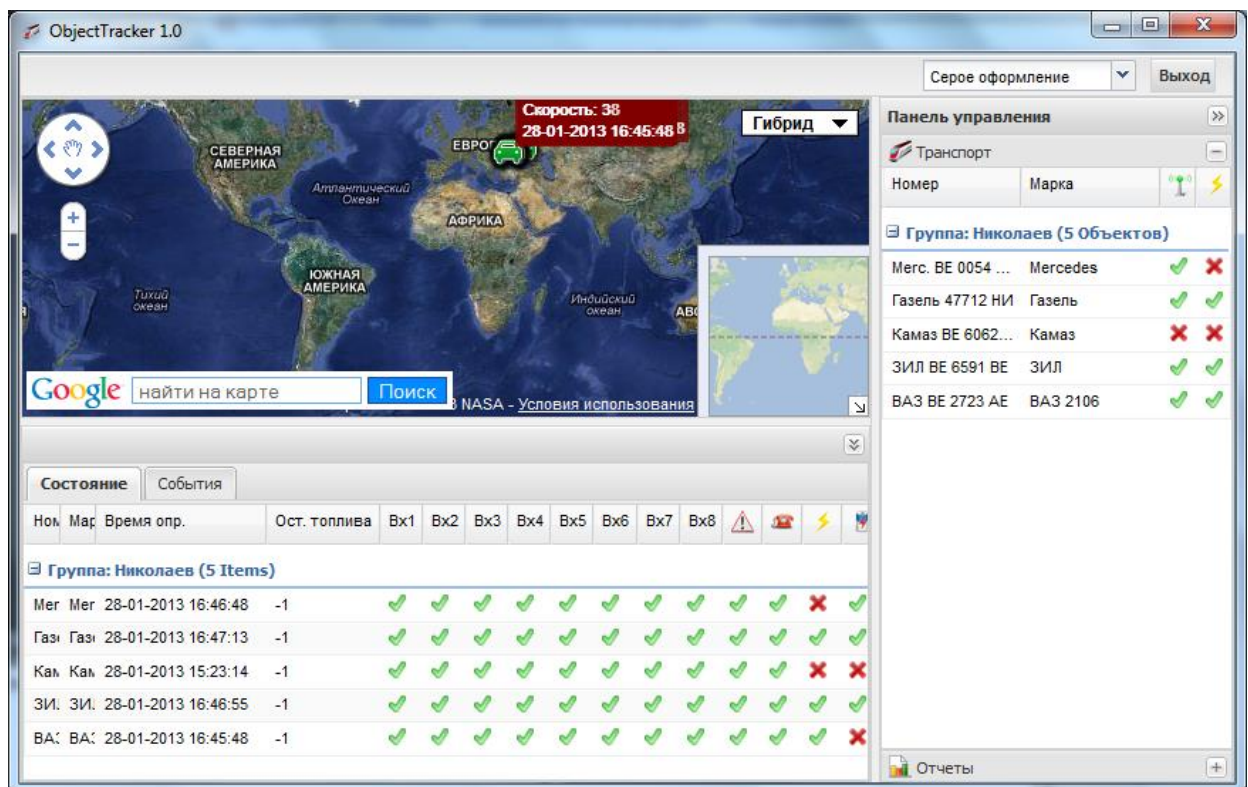


Рис. 4.2. Головне вікно зі зміненою кольоровою темою інтерфейсу.

Область відображення карти містить карту, на якій розміщується поточне місцезнаходження всіх транспортних засобів, з яким є зв'язок у момент перегляду.

Панель журналів містить дві вкладки: «Стан» та «Події». На першій вкладці розміщено таблицю з переліком транспортних засобів та показників їх датчиків (див. п. 5). Інша вкладка містить журнал подій (див. п. 6).

Бокова панель управління надає доступ до перегляду переліку транспортних засобів та отримання статистичної інформації.

5. Таблиця поточного стану транспортних засобів

На першій вкладці панелі журналів міститься таблиця поточного стану транспортних засобів, яка містить наступну інформацію:

- номер транспортного засобу;
- марка та модель транспортного засобу;
- час останнього визначення показників;
- залишок палива;
- стан 8-и дискретних входів;
- стан аварійної сигналізації;
- стан кнопки виклику;
- стан увімкнення зовнішнього живлення;
- стан батареї для автономного живлення;
- наявність зв'язку з транспортним засобом.

Всі показники крім перших чотирьох з переліку є дискретними, тобто можуть приймати лише два значення – «Так» чи «Ні». У таблиці стан «Так» зображується у вигляді зеленої галочки, а протилежний стан – у вигляді червоного хрестика.

6. Журнал подій

Журнал подій міститься на вкладці «Події» панелі журналів головного вікна програми (рис. 6.1).

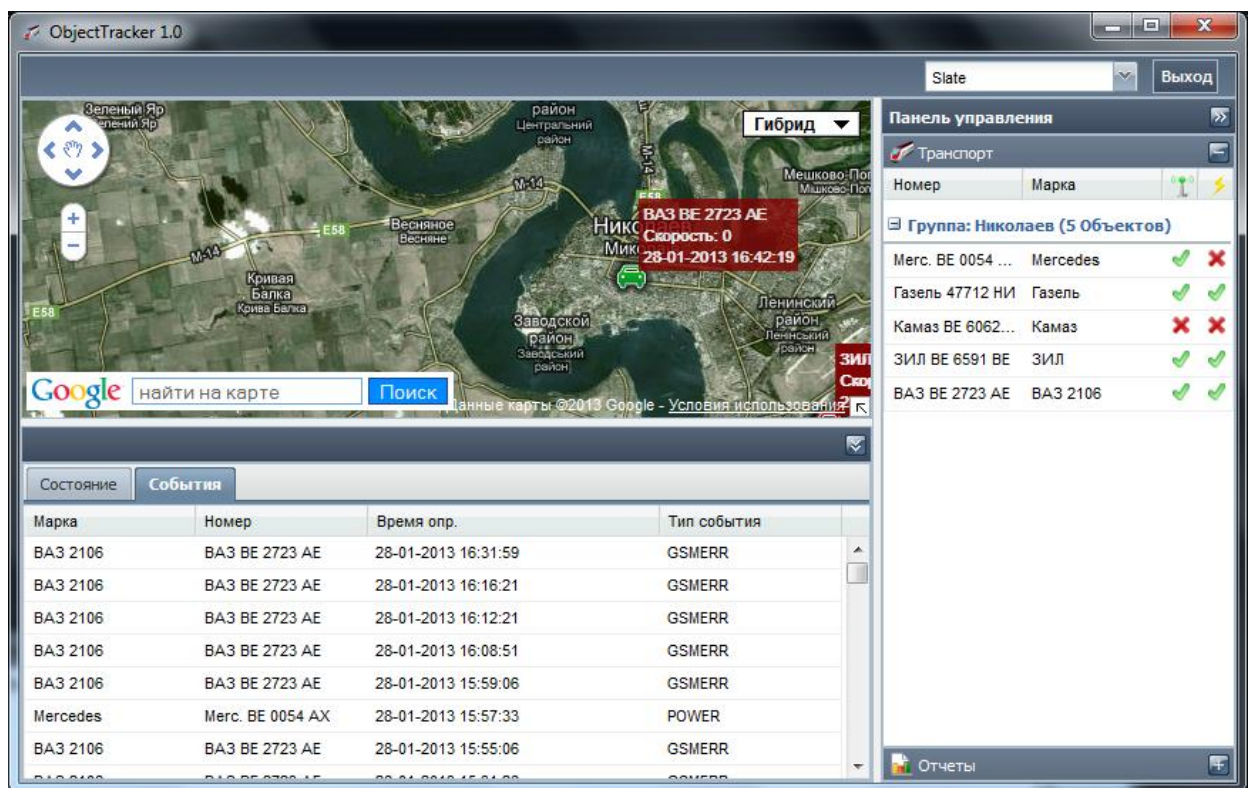


Рис. 6.1. Головне вікно з відображенням вкладки «Події».

Даний журнал містить наступні відомості:

- марка та модель транспортного засобу;
- номер транспортного засобу;
- час визначення події;
- тип події;

На момент розробки першої версії програмного забезпечення було визначено лише дві події, які необхідно було фіксувати у журналі: «GSMERR» – виникає при загубленні зв'язку з транспортним засобом, та «POWER» – виникає при увімкненні двигуна транспортного засобу.

7. Перегляд маршруту руху транспортного засобу

Для перегляду маршруту руху транспортного засобу необхідно зробити декілька кроків:

- 1) перейти на вкладку «Звіти» панелі управління головного вікна;
- 2) натиснути кнопку «Вибрати транспорт для звіту»;
- 3) у списку транспортних засобів обрати необхідний;
- 4) встановити початок та кінець періоду, за який необхідно переглянути маршрут;

5) натиснути на кнопку із зображенням точок, сполучених лініями.

У результаті виконання цих дій буде показано точки, через які проходив транспортний засіб. При наведенні на одну з них буде показано детальну інформацію із зазначенням часу та швидкості у цій точці. Описаний результат та елементи інтерфейсу можна бачити нарис. 7.1.

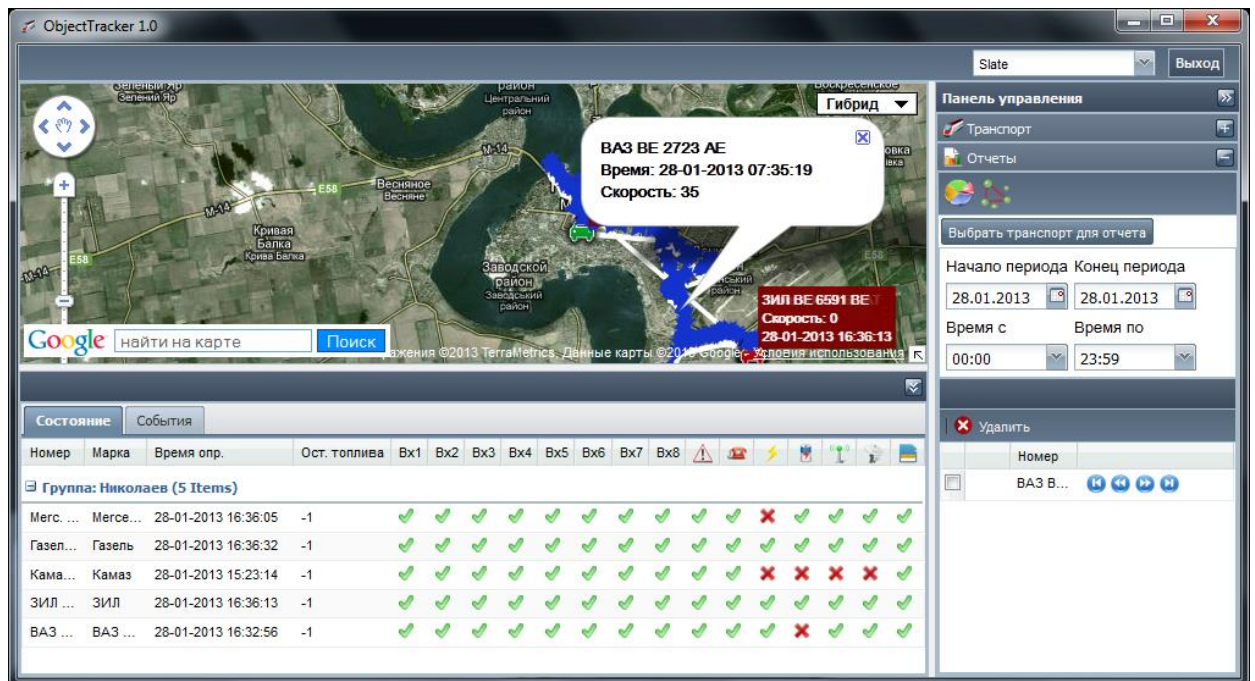


Рис. 7.1. Головне вікно з відображенням маршруту на карті.

8. Вікно побудови графіку швидкості та використання пального

Для побудови даного графіка необхідно виконати всі ті ж самі дії, що й для перегляду маршруту (див. п. 7), окрім останньої, якою є натискання на кнопку відображення маршруту. Поряд з нею розміщено кнопку з зображенням діаграми, при натисканні на яку відбувається побудова графіків та відкривається вікно для його відображення, що зображено на рис. 8.1.

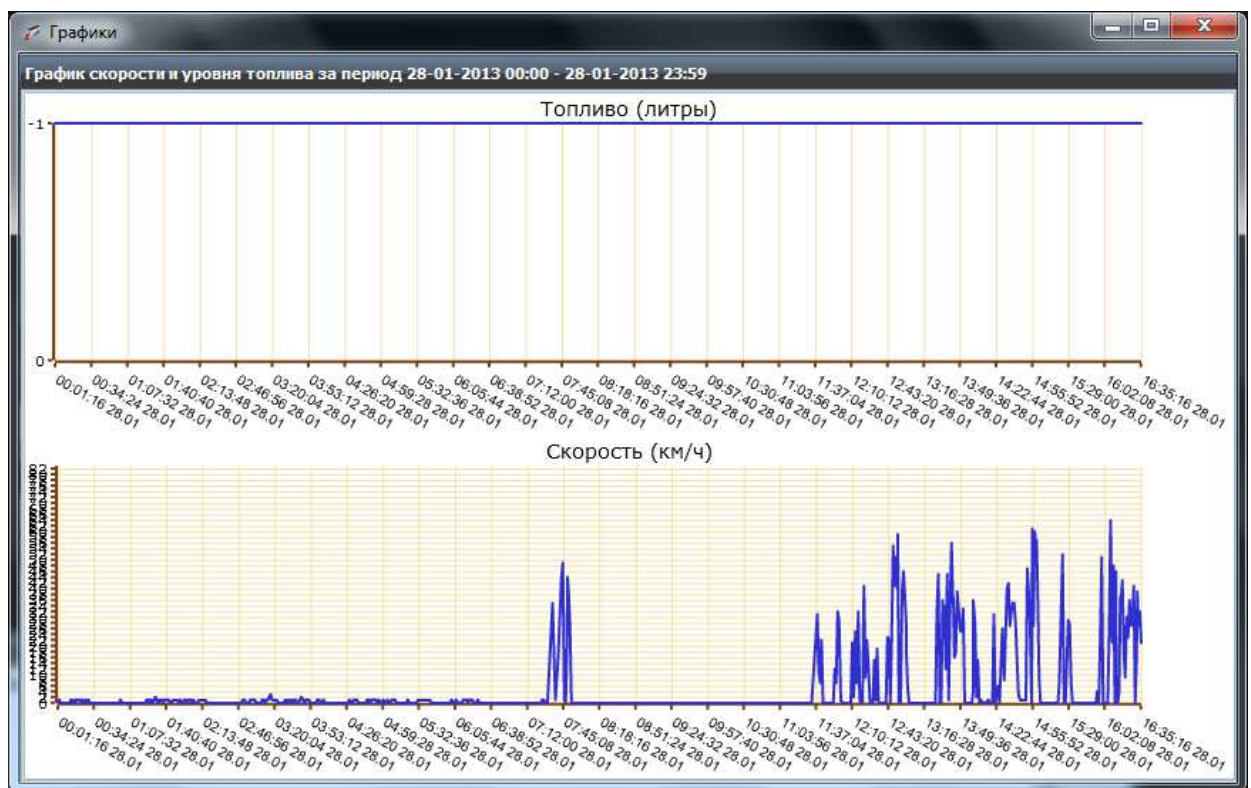


Рис. 8.1. Графік швидкості та використаного пального за обраний проміжок часу.

Як можна бачити з представленого зображення, датчик рівня палива не був підключений на момент спостерігання, тому на графіку використаного пального зображено пряму лінію. Крива зміни швидкості побудована відповідно отриманим даним у кожний з моментів часу за обраний проміжок.

ДОДАТОК В. Технічний опис Скрипт для створення бази даних

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `taxi` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci ;
USE `taxi` ;

-- -----
-- Table `taxi`.`cartype`
-- -----
CREATE TABLE IF NOT EXISTS `taxi`.`cartype` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `nazvanie` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB;

-- -----
-- Table `taxi`.`car`
-- -----
CREATE TABLE IF NOT EXISTS `taxi`.`car` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `gos_nomer` VARCHAR(10) NULL ,
  `marka` VARCHAR(30) NULL ,
  `color` VARCHAR(20) NULL ,
  `bagazh` TINYINT(1) NULL ,
  `driver_fio` VARCHAR(50) NULL ,
  `num_of_places` INT NULL ,
  `cartype_id` INT NOT NULL ,
  PRIMARY KEY (`id`) ,
  INDEX `fk_car_cartype1_idx` (`cartype_id` ASC) ,
  CONSTRAINT `fk_car_cartype1`
    FOREIGN KEY (`cartype_id`)
      REFERENCES `taxi`.`cartype` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----
-- Table `taxi`.`status`
-- -----
CREATE TABLE IF NOT EXISTS `taxi`.`status` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(30) NULL ,
  `sTime` DATETIME NULL ,
  `latitude` DECIMAL(10) NULL ,
  `longitude` DECIMAL(10) NULL ,
  `car_id` INT NOT NULL ,
```

```

PRIMARY KEY (`id`) ,
INDEX `fk_status_car1_idx` (`car_id` ASC) ,
CONSTRAINT `fk_status_car1`
    FOREIGN KEY (`car_id`)
    REFERENCES `taxi`.`car` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-----
-- Table `taxi`.`dispatcher`
-----
CREATE TABLE IF NOT EXISTS `taxi`.`dispatcher` (
    `id` INT NOT NULL AUTO_INCREMENT ,
    `fio` VARCHAR(50) NOT NULL ,
    `phone` VARCHAR(12) NULL ,
    `adress` VARCHAR(100) NULL ,
    PRIMARY KEY (`id`) )
ENGINE = InnoDB;

```

```

-----
-- Table `taxi`.`client`
-----
CREATE TABLE IF NOT EXISTS `taxi`.`client` (
    `id` INT NOT NULL AUTO_INCREMENT ,
    `fio` VARCHAR(50) NOT NULL ,
    `phone` VARCHAR(12) NULL ,
    `adress` VARCHAR(100) NULL ,
    PRIMARY KEY (`id`) )
ENGINE = InnoDB;

```

```

-----
-- Table `taxi`.`tarif`
-----
CREATE TABLE IF NOT EXISTS `taxi`.`tarif` (
    `id` INT NOT NULL AUTO_INCREMENT ,
    `price` DECIMAL(10,2) NOT NULL ,
    `nazvanie` VARCHAR(25) NOT NULL ,
    PRIMARY KEY (`id`) )
ENGINE = InnoDB;

```

```

-----
-- Table `taxi`.`zakaz`
-----
CREATE TABLE IF NOT EXISTS `taxi`.`zakaz` (
    `id` INT NOT NULL AUTO_INCREMENT ,
    `start` DATETIME NULL ,
    `finish` DATETIME NULL ,
    `start_adress` VARCHAR(100) NULL ,
    `finish_adress` VARCHAR(100) NULL ,
    `distance` DECIMAL(10) NULL ,

```

```

`zakazcol` VARCHAR(45) NULL ,
`tarif_id` INT NOT NULL ,
`car_id` INT NOT NULL ,
`dispatcher_id` INT NOT NULL ,
`client_id` INT NOT NULL ,
PRIMARY KEY (`id`) ,
INDEX `fk_zakaz_tarif_idx` (`tarif_id` ASC) ,
INDEX `fk_zakaz_car1_idx` (`car_id` ASC) ,
INDEX `fk_zakaz_dispatcher1_idx` (`dispatcher_id` ASC) ,
INDEX `fk_zakaz_client1_idx` (`client_id` ASC) ,
CONSTRAINT `fk_zakaz_tarif`
    FOREIGN KEY (`tarif_id` )
    REFERENCES `taxi`.`tarif` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_zakaz_car1`
    FOREIGN KEY (`car_id` )
    REFERENCES `taxi`.`car` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_zakaz_dispatcher1`
    FOREIGN KEY (`dispatcher_id` )
    REFERENCES `taxi`.`dispatcher` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_zakaz_client1`
    FOREIGN KEY (`client_id` )
    REFERENCES `taxi`.`client` (`id` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `taxi` ;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```


ДОДАТОК Г. Код програми

Текст програми підсистеми моніторингу даних

Файл Car.java

```
package entities;

import java.io.Serializable;
import java.util.List;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;

/**
 *
 * @author KKO
 */
@Entity
@Table(name = "car")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Car.findAll", query = "SELECT c FROM Car c"),
    @NamedQuery(name = "Car.findById", query = "SELECT c FROM Car c WHERE c.id = :id"),
    @NamedQuery(name = "Car.findByGosNomer", query = "SELECT c FROM Car c WHERE c.gosNomer = :gosNomer"),
    @NamedQuery(name = "Car.findByMarka", query = "SELECT c FROM Car c WHERE c.marka = :marka"),
    @NamedQuery(name = "Car.findByColor", query = "SELECT c FROM Car c WHERE c.color = :color"),
    @NamedQuery(name = "Car.findByBagazh", query = "SELECT c FROM Car c WHERE c.bagazh = :bagazh"),
    @NamedQuery(name = "Car.findByDriverFio", query = "SELECT c FROM Car c WHERE c.driverFio = :driverFio"),
    @NamedQuery(name = "Car.findByNumOfPlaces", query = "SELECT c FROM Car c WHERE c.numOfPlaces = :numOfPlaces"))
public class Car implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```

@Basic(optional = false)
@Column(name = "id")
private Integer id;
@Size(max = 10)
@Column(name = "gos_nomer")
private String gosNomer;
@Size(max = 30)
@Column(name = "marka")
private String marka;
@Size(max = 20)
@Column(name = "color")
private String color;
@Column(name = "bagazh")
private Boolean bagazh;
@Size(max = 50)
@Column(name = "driver_fio")
private String driverFio;
@Column(name = "num_of_places")
private Integer numPlaces;
@JoinColumn(name = "cartype_id", referencedColumnName =
"id")
@ManyToOne(optional = false)
private Cartype cartypeId;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "carId")
private List<Zakaz> zakazList;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "carId")
private List<Status> statusList;

public Car() {
}

public Car(Integer id) {
    this.id = id;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getGosNomer() {
    return gosNomer;
}

public void setGosNomer(String gosNomer) {
    this.gosNomer = gosNomer;
}

public String getMarka() {
    return marka;
}

```

```

public void setMarka(String marka) {
    this.marka = marka;
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    this.color = color;
}

public Boolean getBagazh() {
    return bagazh;
}

public void setBagazh(Boolean bagazh) {
    this.bagazh = bagazh;
}

public String getDriverFio() {
    return driverFio;
}

public void setDriverFio(String driverFio) {
    this.driverFio = driverFio;
}

public Integer getNumOfPlaces() {
    return numOfPlaces;
}

public void setNumOfPlaces(Integer numOfPlaces) {
    this.numOfPlaces = numOfPlaces;
}

public Cartype getCartypeId() {
    return cartypeId;
}

public void setCartypeId(Cartype cartypeId) {
    this.cartypeId = cartypeId;
}

@XmlTransient
public List<Zakaz> getZakazList() {
    return zakazList;
}

public void setZakazList(List<Zakaz> zakazList) {
    this.zakazList = zakazList;
}

@XmlTransient
public List<Status> getStatusList() {

```

```

        return statusList;
    }

    public void setStatusList(List<Status> statusList) {
        this.statusList = statusList;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case
the id fields are not set
        if (!(object instanceof Car)) {
            return false;
        }
        Car other = (Car) object;
        if ((this.id == null && other.id != null) || (this.id !=
null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "entities.Car[ id=" + id + " ]";
    }
}

```

Файл Zakaz.java

```

package entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;

```

```

import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author KKO
 */
@Entity
@Table(name = "zakaz")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Zakaz.findAll", query = "SELECT z FROM Zakaz z"),
    @NamedQuery(name = "Zakaz.findById", query = "SELECT z FROM Zakaz z WHERE z.id = :id"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.start = :start"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.finish = :finish"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.startAdress = :startAdress"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.finishAdress = :finishAdress"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.distance = :distance"),
    @NamedQuery(name = "Zakaz.findByName", query = "SELECT z FROM Zakaz z WHERE z.zakazcol = :zakazcol"))
public class Zakaz implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Column(name = "start")
    @Temporal(TemporalType.TIMESTAMP)
    private Date start;
    @Column(name = "finish")
    @Temporal(TemporalType.TIMESTAMP)
    private Date finish;
    @Size(max = 100)
    @Column(name = "start_adress")
    private String startAdress;
    @Size(max = 100)
    @Column(name = "finish_adress")
    private String finishAdress;
    @Column(name = "distance")
    private Long distance;
    @Size(max = 45)
    @Column(name = "zakazcol")
    private String zakazcol;
    @JoinColumn(name = "client_id", referencedColumnName = "id")
    @ManyToOne(optional = false)
    private Client clientId;

```

```

    @JoinColumn(name = "dispatcher_id", referencedColumnName =
"id")
    @ManyToOne(optional = false)
    private Dispatcher dispatcherId;
    @JoinColumn(name = "car_id", referencedColumnName = "id")
    @ManyToOne(optional = false)
    private Car carId;
    @JoinColumn(name = "tarif_id", referencedColumnName = "id")
    @ManyToOne(optional = false)
    private Tarif tarifId;

    public Zakaz() {
    }

    public Zakaz(Integer id) {
        this.id = id;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Date getStart() {
        return start;
    }

    public void setStart(Date start) {
        this.start = start;
    }

    public Date getFinish() {
        return finish;
    }

    public void setFinish(Date finish) {
        this.finish = finish;
    }

    public String getStartAdress() {
        return startAdress;
    }

    public void setStartAdress(String startAdress) {
        this.startAdress = startAdress;
    }

    public String getFinishAdress() {
        return finishAdress;
    }

    public void setFinishAdress(String finishAdress) {

```

```

        this.finishAdress = finishAdress;
    }

    public Long getDistance() {
        return distance;
    }

    public void setDistance(Long distance) {
        this.distance = distance;
    }

    public String getZakazcol() {
        return zakazcol;
    }

    public void setZakazcol(String zakazcol) {
        this.zakazcol = zakazcol;
    }

    public Client getClientId() {
        return clientId;
    }

    public void setClientId(Client clientId) {
        this.clientId = clientId;
    }

    public Dispatcher getDispatcherId() {
        return dispatcherId;
    }

    public void setDispatcherId(Dispatcher dispatcherId) {
        this.dispatcherId = dispatcherId;
    }

    public Car getCarId() {
        return carId;
    }

    public void setCarId(Car carId) {
        this.carId = carId;
    }

    public Tarif getTarifId() {
        return tarifId;
    }

    public void setTarifId(Tarif tarifId) {
        this.tarifId = tarifId;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
    }

```

```

        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case
the id fields are not set
        if (!(object instanceof Zakaz)) {
            return false;
        }
        Zakaz other = (Zakaz) object;
        if ((this.id == null && other.id != null) || (this.id !=
null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "entities.Zakaz[ id=" + id + " ]";
    }
}

```

Файл Status.java

```

package entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author KKO
 */
@Entity
@Table(name = "status")
@XmlRootElement
@NamedQueries({

```



```

    @NamedQuery(name = "Status.findAll", query = "SELECT s FROM
Status s"),
    @NamedQuery(name = "Status.findById", query = "SELECT s FROM
Status s WHERE s.id = :id"),
    @NamedQuery(name = "Status.findByName", query = "SELECT s
FROM Status s WHERE s.name = :name"),
    @NamedQuery(name = "Status.findBySTime", query = "SELECT s
FROM Status s WHERE s.sTime = :sTime"),
    @NamedQuery(name = "Status.findByLatitude", query = "SELECT
s FROM Status s WHERE s.latitude = :latitude"),
    @NamedQuery(name = "Status.findByLongitude", query = "SELECT
s FROM Status s WHERE s.longitude = :longitude"))
public class Status implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Size(max = 30)
    @Column(name = "name")
    private String name;
    @Column(name = "sTime")
    @Temporal(TemporalType.TIMESTAMP)
    private Date sTime;
    @Column(name = "latitude")
    private Long latitude;
    @Column(name = "longitude")
    private Long longitude;
    @JoinColumn(name = "car_id", referencedColumnName = "id")
    @ManyToOne(optional = false)
    private Car carId;

    public Status() {
    }

    public Status(Integer id) {
        this.id = id;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

```

public Date getSTime() {
    return sTime;
}

public void setSTime(Date sTime) {
    this.sTime = sTime;
}

public Long getLatitude() {
    return latitude;
}

public void setLatitude(Long latitude) {
    this.latitude = latitude;
}

public Long getLongitude() {
    return longitude;
}

public void setLongitude(Long longitude) {
    this.longitude = longitude;
}

public Car getCarId() {
    return carId;
}

public void setCarId(Car carId) {
    this.carId = carId;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case
the id fields are not set
    if (!(object instanceof Status)) {
        return false;
    }
    Status other = (Status) object;
    if ((this.id == null && other.id != null) || (this.id !=
null && !this.id.equals(other.id))) {
        return false;
    }
    return true;
}

```

```
@Override
public String toString() {
    return "entities.Status[ id=" + id + " ]";
}

}
```