**ПРИДНІПРОВСЬКА ДЕРЖАВНА АКАДЕМІЯ БУДІВНИЦТВА ТА АРХІТЕКТУРИ**

Факультет інформаційних технологій та механічної інженерії

(повне найменування інституту, фікультету)

Кафедра Комп’ютерних наук, інформаційних технологій та прикладної математики

(повна назва кафедри)

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему Розробка програми ведення обліку автозаправної станції

Виконав: здобувач вищої освіти,   
Бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ступінь вищої освіти)

спеціальності

122 Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва спеціальності)

освітньої програми

Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(вид та назва ОП)

групи КН-17

Розумовський Владислав\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Керівник Ілля Ільєв\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Рецензент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Оцінка захисту кваліфікаційної роботи (проекту)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(сума балів, оцінка ЄТКС, оцінка за національною шкалою)

Секретар ЕК \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ім’я та прізвище)

Дніпро – 2021

**ПРИДНІПРОВСЬКА ДЕРЖАВНА АКАДЕМІЯ БУДІВНИЦТВА ТА АРХІТЕКТУРИ**

Інститут, факультет Інформаційних технологій та механічної інженерії\_\_\_\_\_\_\_

Кафедра Комп’ютерних наук, інформаційних технологій та прикладної математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рівень вищої освіти Бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

Спеціальність Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

Освітня програма Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри \_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

**ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Розумовського Владислава\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

1. Тема роботи Розробка програми ведення обліку автозаправної станції\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Ілля Ільєв, к.т.н., доцент кафедри комп’ютерних наук, інформаційних технологій та прикладної математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я, прізвище, ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “08” лютого 2021 року №48-КС  
2. Строк подання роботи 14 червня 2021 року\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Ім’я, прізвище та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 01 лютого 2021 року \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної  роботи | Строк виконання етапів роботи | Примітка |
| 1 | Огляд літератури | 05.02.2021 – 22.03.2021 |  |
| 2 | Вивчення задач | 23.03.2021 – 16.04.2021 |  |
| 3 | Проектування парковки | 17.04.2021 – 04.05.2021 |  |
| 4 | Створення анімації | 05.05.2021 – 10.05.2021 |  |
| 5 | Оформлення пояснювальної записки | 11.05.2021 – 30.05.2021 |  |

**Здобувач вищої освіти \_\_\_\_** Розумовський Владислав

(підпис) (ім’я та прізвище)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Ілля Ільєв\_\_\_\_\_

(підпис) (ім’я та прізвище)

РЕФЕРАТ

Темою кваліфікаційної роботи є «Розробка програми ведення обліку автозаправної станції».

Кваліфікаційна робота: 44 сторінок, 24 рисунків, 13 таблиць, 5 літературних джерел, 1 додаток.

Об`єкт дослідження — автоматизація автозаправної станції.

Мета роботи — дослідити принципи автоматизації поставок та продажу палива на АЗС та розробити програмний додаток, що демонстрував би автоматизовану роботу автозаправної станції.

Характеристика програмних засобів реалізації проекту 4 сторінки.

Проектування та реалізація програмного продукту 17 сторінок. Докладна інформація про розроблення програми.

Розробка інструкції користувача 4 сторінки. Докладна інформація про експлуатацію програми.

Охорона праці 6 сторінок. Аналіз та розрахунок захисту від шкідливих та небезпечних факторів.

Економічна частина 4 сторінки. Економічний розрахунок вартості програмного продукту.

Методика дослідження – були досліджені методи автоматизації поставок/продажу, розробка СУБД для заповнення обліку.

Одержані наслідки – в результаті кваліфікаційної роботи була створено програмний додаток, який автоматизує облік поставок та продажу палива на АЗС.

Ключові слова: *СУБД, автоматизування, автозаправна станція, SQLite, С++, Qt.*

ЗМІСТ

[1.1 Опис середовища та мови програмування розробки програмного продукту 6](#_Toc75110124)

[1.2 Опис використаної СУБД в кваліфікаційному проекті 7](#_Toc75110125)

[3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 10](#_Toc75110126)

[3.1 Опис структури бази даних 10](#_Toc75110127)

[3.2 Проектування інтерфейсу програми 12](#_Toc75110128)

[3.4 Структурна схема взаємодії складових програми 23](#_Toc75110129)

[3.5 Контроль коректності вхідних та вихідних даних 24](#_Toc75110130)

[4 РОЗРОБКА ІНСТРУКЦІЇ КОРИСТУВАЧА 27](#_Toc75110132)

[5 ОХОРОНА ПРАЦІ 34](#_Toc75110133)

[5.1Шкідливі та небезпечні чинники під час розробки автоматизованого робочого місця 34](#_Toc75110134)

[5.2 Інженерно-технічні заходи охорони праці 35](#_Toc75110135)

[5.3 Пожежна профілактика 38](#_Toc75110136)

[6 ЕКОНОМІЧНИЙ РОЗДІЛ 40](#_Toc75110137)

[ВИСНОВКИ 44](#_Toc75110138)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 45](#_Toc75110139)

Додаток 47

ВСТУП

В нашій країні працює безліч заправних станцій для автомобілів. Ними користуються мільйони автомобілів, які заправляються паливом на цих АЗС. Це мільярди доларів прибутку. Але для того що б уникнути втрати через не точний обліку поставок та продажу, потрібно розробити програму для обліку постачання та продажу палива на АЗС.

У зв’язку з цим, була б актуальна програма, яка допомагає вести облік палива, їх постачання та продаж. Програма повинна бути зрозуміла та проста у використанні для любої людини.

В рамках даного кваліфікаційного проекту потрібно організувати базу даних для обліку палива, його постачання та продаж, розробити програмний додаток, який дозволяє працювати з розробленою базою даних.

Програма повинна виконувати такі функції по веденню бази даних:

* ведення обліку палива;
* ведення обліку надходження та продаж палива;
* перегляд інформації про паливо у базі даних;
* сортування таблиці «Паливо» по всім полям;
* сортування таблиці «Постачання» по всім полям;
* сортування таблиці «Продаж» по всім полям;
* фільтрування записів у таблиці «Паливо» за: назвою, ціною;
* фільтрування записів у таблиці «Постачання» за: датою надходження, постачальником та кількістю;
* фільтрування записів у таблиці «Продаж» за: датою продажу, ціною та кількістю;

Крім того, програма повинна перевіряти усі введенні дані на коректність та видавати повідомлення у випадку помилкового вводу.

Таблиця «Паливо» містить у собі дані про назву палива, його вартість та кількість, тип палива.

Таблиця «Постачання» містить у собі дані про паливо, що постачається, його кількість, постачальника та дату постачання.

Таблиця «Продаж» містить у собі дані про паливо, що постачається, його кількість, постачальника та дату постачання.

Таблиця «Вид палива» містить у собі дані про види палива.

Рекомендовані вимоги до конфігурації персонального комп’ютеру:

* тип процесора Intel Pentium 1400 MHz або краще;
* обсяг оперативного запам'ятовуючого пристрою 512 Мб і більше
* обсяг вільного місця на жорсткому диску 100 Мб і більше
* oпераційна систем Microsoft Windows 7, 8, 10

1 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

# 1.1 Опис середовища та мови програмування розробки програмного продукту

C ++ - компільований, статично типізований мова програмування загального призначення.

Синтаксис C ++ успадкований від мови C. Одним з принципів розробки було збереження сумісності з C. Проте, C ++ не є в строгому сенсі надбезліччю C; безліч програм, які можуть однаково успішно транслюватися як компіляторами C, так і компіляторами C ++, досить велике, але не включає всі можливі програми на C.

Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в себе поширені контейнери і алгоритми, введення-виведення, регулярні вирази, підтримку багатопоточності і інші можливості. C ++ поєднує властивості як високорівневих, так і низькорівневих мов [1]. У порівнянні з його попередником - мовою C, - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування. [2]

Стандарт C ++ складається з двох основних частин: опис ядра мови і опис стандартної бібліотеки.

Перший час мови розвивався поза формальних рамок, спонтанно, у міру вставали перед ним завдань. Розвитку мови супутніх розвитку крос-компілятора cfront. Нововведення в рецензії крос-компілятора. C ++ не веде. Лише в 1998 році став стандартизованим.

С ++ підтримує як коментарі в стилі C (/ \* коментар \* /), так і однорядкові (// вся решта рядків є коментарем), де // підміню головного коментаря, а не подальший символ нового рядка, (або еквівалентним йому позначенням ??), вважається закінченням коментаря. Плюс цього коментаря в тому, що його не обов'язково закінчувати, тобто позначати закінчення коментарів.

Специфікатор вбудований для функцій. Функція, певна всередині тіла класу, є inline за замовчуванням. Даний специфікатор є підказкою компілятору і може вбудувати функції в код замість її безпосереднього виклику.

Кваліфікатори const і volatile. На відміну від С, де C ++ змінна з кваліфікаторів const повинна бути инициализирована. volatile використовується в описі змінних і інформує компілятор, що значення може бути змінено способом, який компілятор не в змозі відстежити. Для змінних, оголошених волатильних, компілятор не повинен застосовувати засоби, наприклад, що поміщають її в регістр, або покладаються на незмінність значення змінних в проміжку між двома присвоювання її значення. У многоядерной системі волатільна допомогу уникати бар'єрів пам'яті 2-го типу[3].

Windows Forms - інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код - класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C #, С ++, так і на VB.Net, J # і ін.

Додаток Windows Forms є подієво-орієнтоване додатком, підтримуване Microsoft .NET Framework. На відміну від пакетних програм, велика частина часу витрачається на очікування від користувача будь-яких дій, як, наприклад, введення тексту в текстове поле або кліка мишкою по кнопці[4].

# 1.2 Опис використаної СУБД в кваліфікаційному проекті

SQLite - компактна вбудована реляційна база даних. Вихідний код бібліотеки переданий в суспільне надбання. У 2005 році проект отримав нагороду Google-O'Reilly Open Source Awards.

SQLite підтримує динамічне тіпізірованія даних. Можливі типи полів: INTEGER, REAL, TEXT, BLOB.

Слово «вбудовується» (embedded) означає, що SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якої програма компонується, і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID-функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті поставки йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі движка можливо використовувати SQLite як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.  
Більш ранні версії SQLite були спроектовані без будь-яких обмежень, єдиною умовою було те, щоб база даних вміщувалася в пам'яті, в якій всі обчислення проводилися за допомогою 32-розрядних цілих чисел. Це створювало певні проблеми. Через те, що верхні межі були визначені і відповідно належним чином протестовані, часто виявлялися помилки при використанні SQLite в досить екстремальних умовах. Тому в нових версіях SQLite були введені межі, які тепер перевіряються разом із загальним набором тестів[5].

# 3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

# 3.1 Опис структури бази даних

Для реалізації бази даних була обрана СУБД SQLite. Доцільно було б поділити базу на чотири таблиці. Опис таблиць приведено нижче у таблицях 3.1 – 3.4.

Таблиця 3.1 – Перелік даних про паливо «Gas»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис | Тип | Ключ |
| ID | ID Замовлення | Лічильник | \* |
| NameGas | Назва палива | Текстовий |  |
| price | Ціна | Дійсний |  |
| IDType | ID Вида палива | Цілий |  |
| count | Кількість | Цілий |  |

Таблиця 3.2 – Перелік даних проданого палива «Coming»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис | Тип | Ключ |
| ID | ID | Лічильник | \* |
| IDGas | ID палива | Цілий |  |
| Price | Ціна | Дійсний |  |
| DateSell | Дата продажу | Дата |  |
| count | Кількість | Цілий |  |

Таблиця 3.3 – Перелік даних надходження палива «Sell»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис | Тип | Ключ |
| ID | ID | Лічильник | \* |
| IDGas | ID палива | Цілий |  |
| DateComing | Дата прийому | Дата |  |
| provider | Поставник | Текстовий |  |
| count | Кількість | Цілий |  |

Таблиця 3.4 – Перелік видів палива «TypeGas»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис | Тип | Ключ |
| ID | ID | Лічильник | \* |
| NameType | Назва типу | Цілий |  |

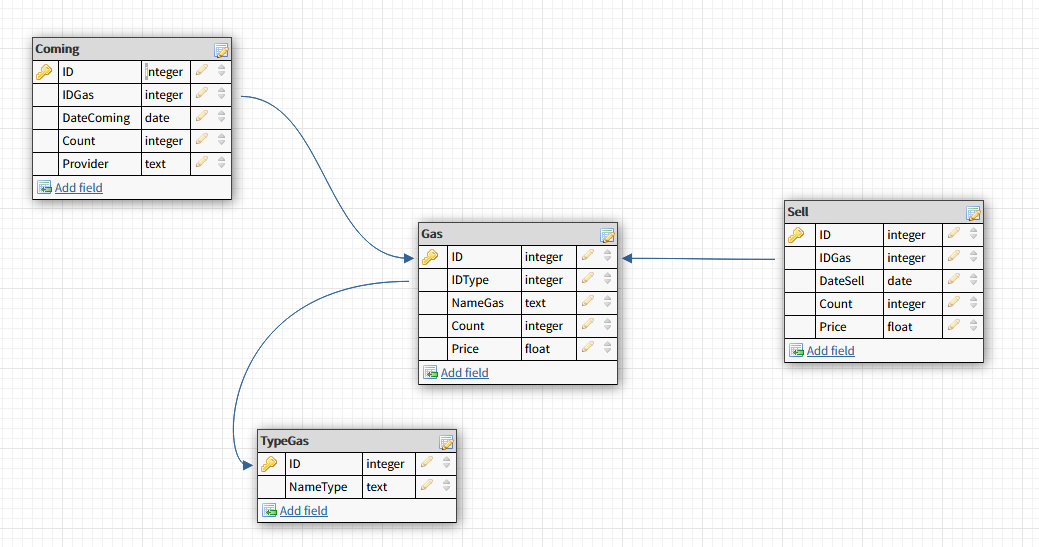


Рисунок 3.1 – Схема зв’язків між таблицями

Поля таблиць, які містять ID не відображається в ході виконання програми, так як служать для зв’язування таблиць за ключовими полями та не несуть змістовного значення для користувача.

# 3.2 Проектування інтерфейсу програми

До складу інтерфейсу програми входять сім форм, призначення яких приведено в таблиці 3.5.

Таблиця 3.5 – Призначення форм

|  |  |
| --- | --- |
| Назва | Призначення |
| MainForm | Головна форма, що містить таблицю «Паливо», «Надходженн» та «Продажа» з даними, сортування, фільтрація таблиць. |
| Gas | Додавання нового запису до таблиці «Паливо» та редагування обраного запису. Викликається з головного меню. |
| Coming | Додавання нового запису до таблиці «Надходження» та редагування обраного запису. Викликається з головного меню. |
| Sell | Додавання нового запису до таблиці «Продаж» та редагування обраного запису. Викликається з головного меню. |
| DirectForm | Форма, що містить таблицю «Тип палива». |
| About | Показує вікно «Про програму». |
| Controller | Модуль, що виконує роль контролера у програмі. |

Згідно з постановкою задачі було розроблено інтерфейс програми, який описано нижче.

На формі MainForm, що зображено на рисунку 3.1 розміщено такі компоненти:FindFuelComingText, SellContext, FindFuelEdit, FirstPriceSell, FindFuelEdit, TypeFuelBox, ComingBox, FindSellBox, StartFind, Searchbutton, SecondPriceSell, ProviderText, FuelText, mainMenu, SearchPanel, toolStrip1, FuelContext, ComingContext, FindFuelBox, toolStrip1, TabControl, FuelGrid, ComingGrid, SellGrid, та безліч компонентів типу label.

Компонент TabControl призначений для перегляду даних таблиць. Незалежно одна від одної, сторінки мають такі назви: «Паливо», «Постачання» та «Продаж», які зображено на рисунках 3.1, 3.2, 3.3 відповідно. Він містить три сторінки для кожної таблиці відповідно.

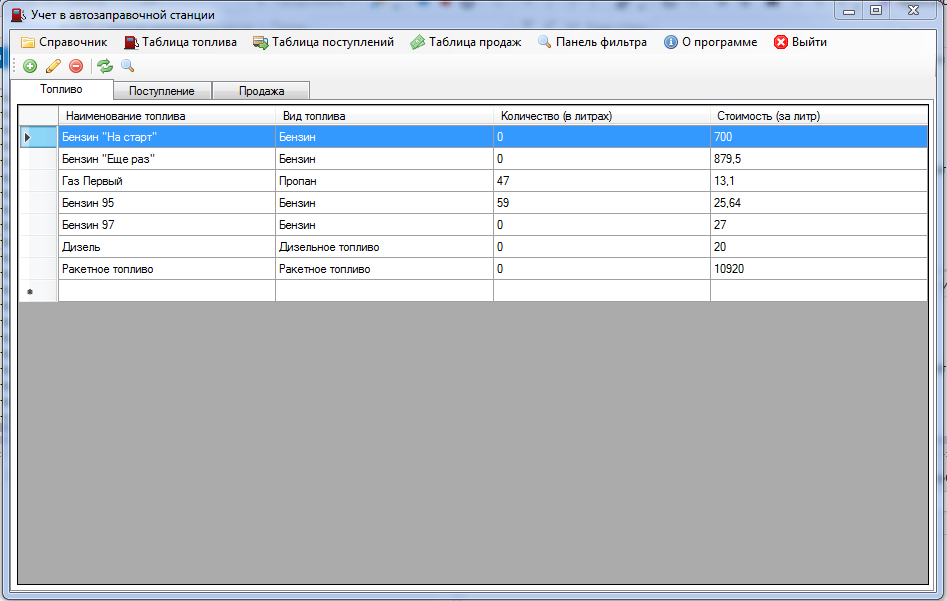


Рисунок 3.1 – Вигляд форми «Головне меню» сторінка перша

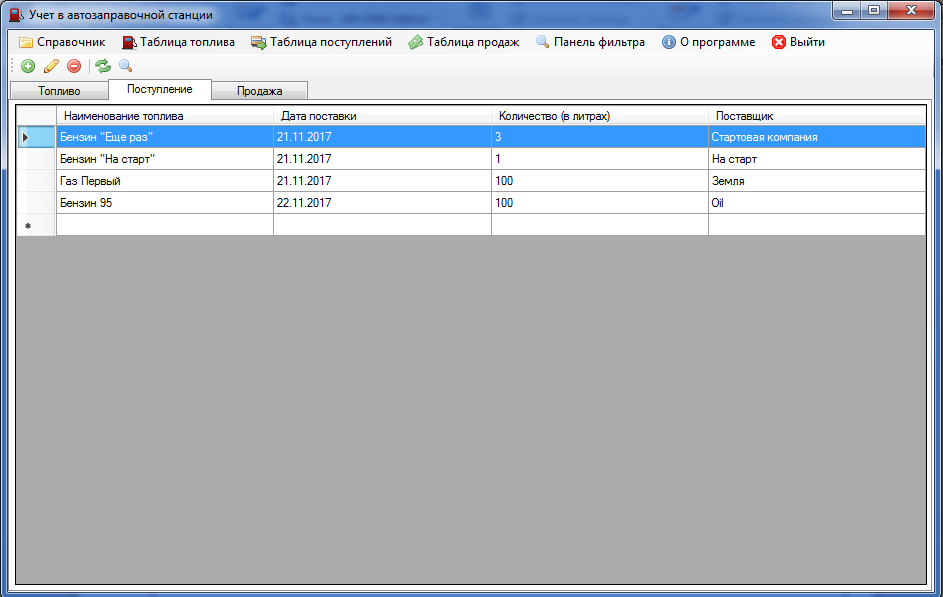


Рисунок 3.2 – Вигляд форми «Головне меню» сторінка друга

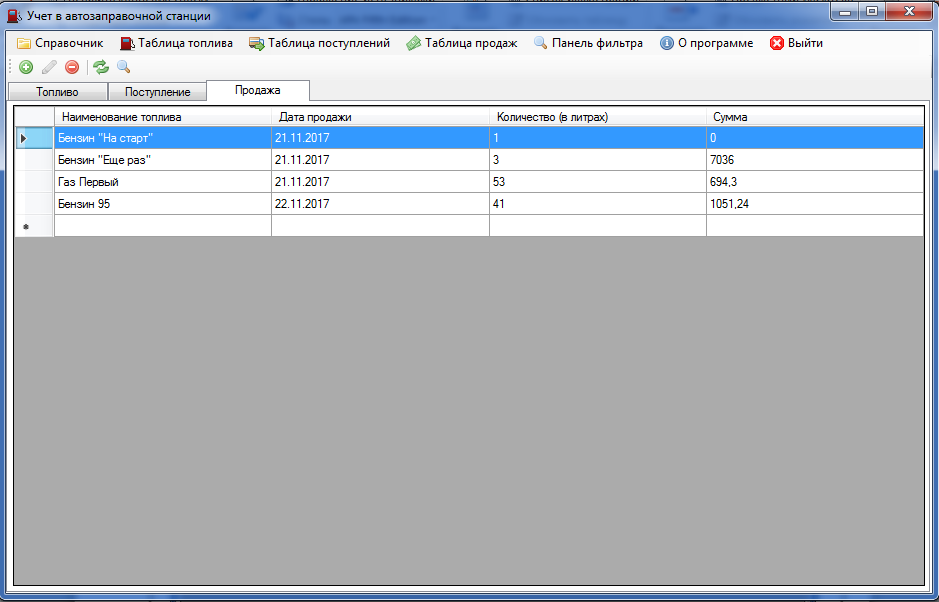


Рисунок 3.3 – Вигляд форми «Головне меню» сторінка третя

FuelText, TypeFuelBox призначені для фільтрування записів за назвою та видом палива у таблиці «Паливо». Усі два компонентів можуть працювати як окремо, так и в парі між собою.

ProviderText, FindFuelComingText призначені для фільтрування записів за постачальником та назвою палива у таблиці «Постачання». Ці два компонента можуть працювати як окремо, так и в парі між собою.

FindFuelEdit, FirstPriceSell, SecondPriceSell призначені для фільтрування записів за назвою та вартістю у таблиці «Продаж». Усі три компонента можуть працювати як окремо, так и в парі між собою.

FuelGrid, ComingGrid, SellGrid – це компоненти, в які завантажуються записи з бази даних. Для роботи над кожною таблицею, створені окремі компоненти типу ContextMenu.

Компонент menuStrip призначений для організації меню переходу між формами та містить такі пункти:

* «Справочник» – перехід до форми «Довідник видів»;
* «Топливо» має підпункти «Добавить», «Редактировать», «Удалить» за допомогою яких, здійснюється маніпулювання над даними у таблиці «Паливо»;
* «Поставка» має підпункти «Добавить», «Редактировать», «Удалить» за допомогою яких, здійснюється маніпулювання над даними у таблиці «Постачання»;
* «Продажа» має підпункти «Добавить», «Редактировать», «Удалить» за допомогою яких, здійснюється маніпулювання над даними у таблиці «Продаж»;
* «Панель фильтра» дозволяє увімкнути та вимкнути панель фільтру;
* «О программе» – перехід до форми «Про програму» ;
* «Выход» – вихід з програми.

Головне меню зображено на рисунку 3.4.

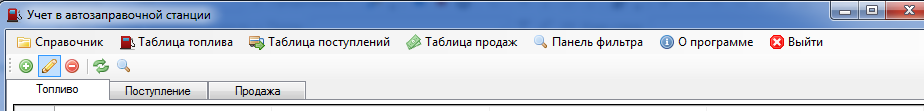


Рисунок 3.4 – Головне меню

На формі «Довідник видів», що зображено на рисунку 3.5 розміщено такі компоненти: FindBox, Findbutton, Refleshbutton, menuStrip1, TypeContext, EditPanel, SearchPanel, TypeFuelGrid, Cancelbutton, Savebutton, TypeBox, TypeNameEdit.

Форма призначена для ведення обліку видів палива, його додавання, редагування, видалення та фільтрування.

Компонент TypeFuelGrid, це таблиця, для відображення видів палива.

Компонент SearchPanel, це контейнер, який містить в собі такі компоненти FindBox, Findbutton, Refleshbutton, за допомогою яких, ведеться фільтрування таблиці за назвою.

Компонент EditPanel, це контейнер, який містить в собі такі компоненти Cancelbutton, Savebutton, TypeBox, TypeNameEdit за допомогою яких, ведеться додавання та редагування видів.

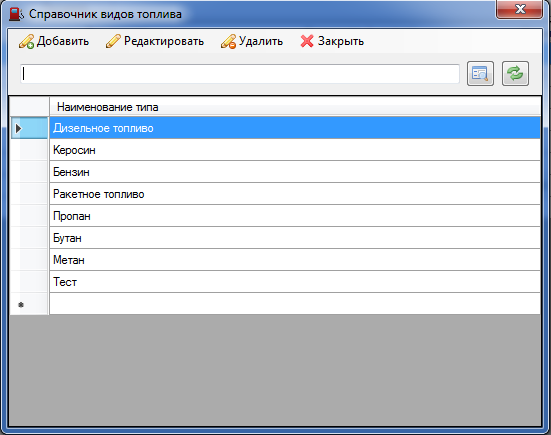


Рисунок 3.5 – Форма довідника видів

На формі «Додавання палива», що зображено на рисунку 3.6 розміщено такі компоненти: FuelBox, Findbutton, TypeBox, NameFuelEdit, PriceEdit, Savebutton, Cancelbutton, label.

Форма «Додавання палива» призначена для додавання та редагування палива.

Компоненти TypeBox, PriceEdit, Findbutton, NameFuelEdit призначені для вводу назви, вартості та виду палива.

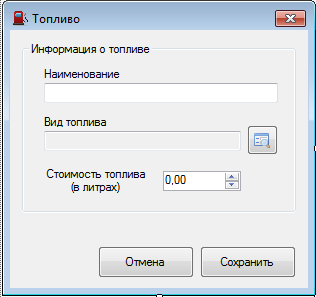


Рисунок 3.6 – Форма додавання палива

На формі «Постачання», що зображено на рисунку 3.7 розміщено такі компоненти: ProviderBox, Savebutton, Cancelbutton, DateEdit, CountEdit, ComingBox, FuelGrid, FuelBox. Форма «Постачання» призначена для додавання запису до таблиці «Постачання» та редагування обраних записів.

Компонент DateEdit призначений для введення дати постачання палива.

Компонент FuelGrid призначений для вибору палива з таблиці.

Компоненти Savebutton, Cancelbutton призначений для збереження та відміни дії.

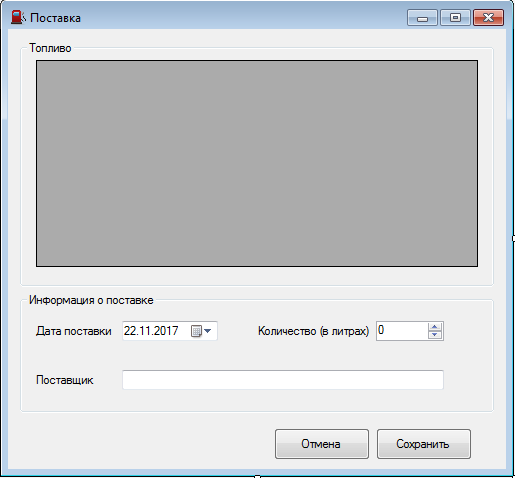


Рисунок 3.7 – Форма «Постачання»

На формі «Продаж», що зображено на рисунку 3.8 розміщено такі компоненти: Cancelbutton, Savebutton, ComingBox, CountEdit, DateEdit, FuelBox, FuelGrid, PriceEdit. Форма «Продаж» призначена для додавання запису до таблиці «Продаж» та редагування записів.

Компонент DateEdit призначений для введення дати продажу палива.

Компонент FuelGrid призначений для зручного вибору палива.

Компонент CountEdit призначений для введення кількості палива.

Компонент PriceEdit призначений для виводу вартості.

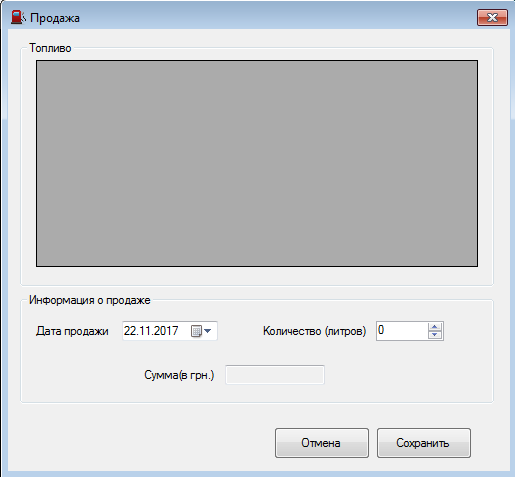


Рисунок 3.8 – Форма «Продаж»

На формі «О программе», що зображено на рисунку 3.9 розміщено такі компоненти: Image, button1 та label. Форма призначена відображення інформації про автора програми.

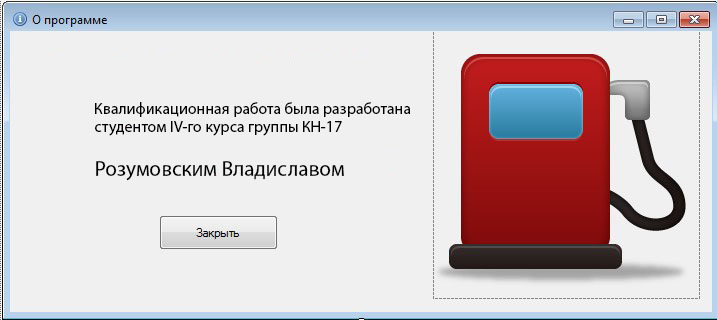


Рисунок 3.9 – Форма «О программе»

На формі «DataModul» не розташовані компоненти, по причині того, що форма виконує роль контролера.

3.3 Опис складових частин програми

Програма містить 7 модулі, їх опис приведено у таблиці 3.5

Таблиця 3.6 – Вміст модулів проекту

|  |  |
| --- | --- |
| Назва | Перелік обробників та функцій |
| MainForm | searchToolStripMenuItem\_Click, RefleshFuel, RefleshComing, RefleshSell, MainForm\_Load, DirectItem\_Click, AddGasItem\_Click, EditGasItem\_Click, AboutItem\_Click, ExitItem\_Click, AddComingItem\_Click, EditComingItem\_Click, AddSellItem\_Click, EditSellItem\_Click, DeleteFuelItem\_Click, Addtool\_Click, Edittool\_Click, Deletetool\_Click, Refleshtool\_Click, TabControl\_SelectedIndexChanged, DeleteComingItem\_Click, DeleteSellItem\_Click, Searchbutton\_Click, FirstPriceMed\_KeyPress |
| DirectType | RefleshType, Cancelbutton\_Click, AddItem\_Click, EditItem\_Click CloseItem\_Click DeleteItem\_Click DirectType\_Load Refleshbutton\_Click, FindBox\_KeyPress, Savebutton\_Click, ChoiseItem\_Click, TypeFuelGrid\_CellClick, Findbutton\_Click, TypeFuelGrid\_CellDoubleClick |
| EditFuel | Cancelbutton\_Click, Savebutton\_Click, PriceEdit\_Leave, FuelEdit\_Load, Findbutton\_Click |
| Coming | Cancelbutton\_Click, Savebutton\_Click, ComingEdit\_Load, FuelGrid\_CellClick |
| Sell | SellEdit\_Load, Cancelbutton\_Click, Savebutton\_Click, FuelGrid\_CellClick, CountEdit\_Leave |
| About | Closebutton\_Click |
| DataModul | OpenConnection, CloseConnection, fillDataTable, SQLrequest, SetIndex, GetIndex, CheckCount, StringToInt, VerifyKey, CommaToDot, GetField, FillBox, SetType, CleanType, GetIDType, GetNameType, SetFuel, CleanFuel, GetCountFuel, SetIDFuelType, GetIDFuelType, SetPriceFuel, GetPriceFuel, SetComingFuel, CleanComingFuel, GetIDComingFuel, SetIDFuelC, GetIDFuelC, GetCountComing, SetCountComing, GetDateComing, GetProvider, SetSell, ClearSell, GetIDSellFuel, GetIDFuelS, SetIDFuelS, GetCountSell, SetCountSell, GetDateSell, GetPrice |

Більш детально призначення підпрограм головної форми описано нижче.

Таблиця 3.7 – Призначення підпрограм форми MyForm

|  |  |
| --- | --- |
| Назва | Призначення |
| searchToolStripMenuItem\_Click | Включення та виключення панелі фільтру |
| RefleshFuel | Оновлення таблиці палива |
| RefleshComing | Оновлення таблиці постачання |
| RefleshSell | Оновлення таблиці продаж |
| MainForm\_Load | Завантаження параметрів, при завантаженні форми |
| DirectItem\_Click | Перехід до форми довідників |
| AddGasItem\_Click | Перехід до додавання палива |
| EditGasItem\_Click | Перехід до редагування палива |
| AboutItem\_Click | Перехід до форми про програму |
| ExitItem\_Click | Вихід з програми |
| AddComingItem\_Click | Перехід до додавання постачання |
| EditComingItem\_Click | Перехід до редагування продажу |
| AddSellItem\_Click | Перехід до додавання продажу |
| DeleteFuelItem\_Click | Видалення запису палива |
| FirstPrice\_KeyPress | Заборона введення символів |
| TabControl\_SelectedIndexChanged | При переходжені між сторінками, зміна параметрів фільтру |
| DeleteComingItem\_Click | Видалення запису постачання |
| DeleteSellItem\_Click | Видалення запису продаж |
| Searchbutton\_Click | Запуск фільтру |

Таблиця 3.8 – Призначення підпрограм форми DirectType

|  |  |
| --- | --- |
| Назва | Призначення |
| RefleshType | Оновлення таблиці видів палива |
| Cancelbutton\_Click | Відміна додавання |
| AddItem\_Click | Встановлення режиму додавання |
| EditItem\_Click | Встановлення режиму редагування |
| CloseItem\_Click | Закриття форми |
| DeleteItem\_Click | Видалення запису з таблиці видів |
| DirectType\_Load | Завантаження параметрів форми |
| Refleshbutton\_Click | Оновлення таблиці видів палива |
| FindBox\_KeyPress | Заборонна введення чисел |
| Savebutton\_Click | Збереження запису |
| ChoiseItem\_Click | Вибір запису при режимі перегляду |
| TypeFuelGrid\_CellClick | Вибір запису |
| Findbutton\_Click | Старт пошуку |
| TypeFuelGrid\_CellDoubleClick | Вибір запису при режимі перегляду |

Таблиця 3.9 – Призначення підпрограм форми FuelEdit

|  |  |
| --- | --- |
| Назва | Призначення |
| Cancelbutton\_Click | Відміна додавання |
| Savebutton\_Click, | Збереження запису |
| PriceEdit\_Leave | Заміна коми на точку |
| FuelEdit\_Load | Завантаження параметрів, при завантаженні форми |
| Findbutton\_Click | Перехід до форми видів палива |

Таблиця 3.10 – Призначення підпрограм форми Coming

|  |  |
| --- | --- |
| Назва | Призначення |
| Cancelbutton\_Click | Відміна додавання |
| Savebutton\_Click | Збереження запису |
| ComingEdit\_Load | Завантаження параметрів, при завантаженні форми |
| FuelGrid\_CellClick | Вибір запису з таблиці |

Таблиця 3.11 – Призначення підпрограм форми Sell

|  |  |
| --- | --- |
| Назва | Призначення |
| SellEdit\_Load | Завантаження параметрів, при завантаженні форми |
| Cancelbutton\_Click | Відміна додавання |
| Savebutton\_Click | Збереження запису |
| FuelGrid\_CellClick | Вибір запису з таблиці |
| CountEdit\_Leave | Розрахунок суми продажу |

Таблиця 3.12 – Призначення підпрограм форми About

|  |  |
| --- | --- |
| Назва | Призначення |
| Closebutton\_Click | Закриття форми |

Таблиця 3.13 – Призначення підпрограм форми DataModul

|  |  |
| --- | --- |
| Назва | Призначення |
| OpenConnection | Підключення до БД |
| CloseConnection | Закриття підключення до БД |
| fillDataTable | Завантаження таблиці з БД |
| SQLrequest | Виконання запиту |
| SetIndex та GetIndex | Встановлення та взяття індексу таблиці |
| CheckCount | Перевірка на наявність у БД |
| StringToInt | Перетворення з строки до числа |
| VerifyKey | Перевірка введених значень |
| CommaToDot | Перетворення коми в точку |
| GetField | Взяття тексту з запису таблиці |
| FillBox | Завантаження видів палива |
| SetType | Записати інформацію виду до контролера |
| CleanType | Очистка інформацію виду у контролері |
| SetFuel | Записати інформацію палива до контролера |
| CleanFuel | Очистка інформацію палива у контролері |
| SetComingFuel | Записати інформацію постачання до контролера |
| CleanComingFuel | Очистка інформацію постачання у контролері |

# 3.4 Структурна схема взаємодії складових програми

З головної форми через пункти головного меню викликаються такі форми:

* «DirectType»;
* «FuelEdit»
* «Coming»;
* «Sell»;
* «About».
* «DataModul»

Схема взаємодії складових часин програми приведена на рисунку 3.9.

MainForm

DirectType

Coming

FuelEdit

About

Sell

Controller

Рисунок 3.10 – Схема взаємодії складових частин програми

# 3.5 Контроль коректності вхідних та вихідних даних

Контроль коректності даних, це болюча тема усіх програм, оскільки виникають при неправильному введенні даних, які приводять до помилок в роботі програми.

При проектування програми були усунені деякі помилки, одну з них було приведено на рисунку 3.11.

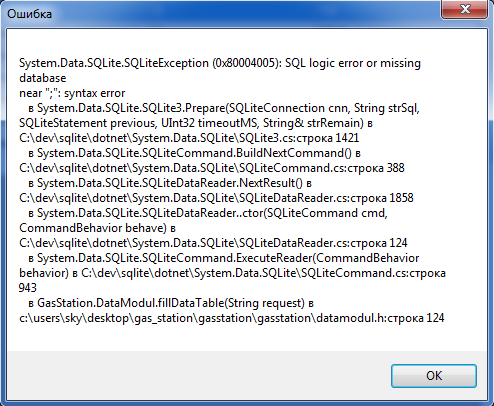


Рисунок 3.11 – Помилка при неправильно сформованому запиті

Ця помилка була усунена за допомогою відстеження місця, де була помилка і виправлення її.

Для захисту від не коректного вводу інформації у таблиці, було розроблено багато перевірок. Приклади зображено на рисунках 3.11 - 3.13.

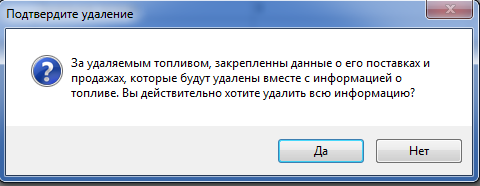


Рисунок 3.12 – Вікно підтвердження при спробі видалити запис за яким ведеться облік постачання та продажу

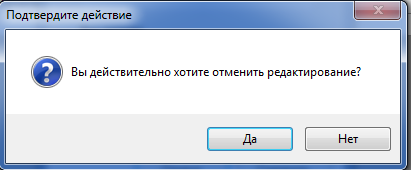


Рисунок 3.13 – Підтвердження відміни додавання

# 

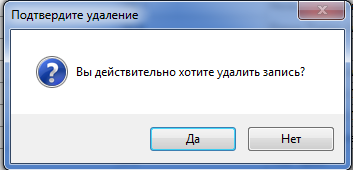


Рисунок 3.14 – Вікно підтвердження видалення

# 4 РОЗРОБКА ІНСТРУКЦІЇ КОРИСТУВАЧА

Для запуску програми потрібно двічі клацнути на файл GasStation.exe. Після цього з’явиться головне вікно програми, що зображено на рисунку 4.1



Рисунок 4.1 – Головне вікно програми

Для комфортності, програма може бути розвернута на весь екран. Головне меню програми розбито на три сторінки. Перша для роботи з паливом, друга для роботи з постачанням палива, третя для роботи з продажами палива. Для перегляду таблиці, потрібно прокрутити колесо мишки або використати скролл-бар у правій частині таблиці.

Вверху програми розташоване меню, яке поділене на сім пунктів. Перший пункт меню дозволяє відкрити форму «Довідник». Другий, третій та четвертий пункти меню дозволяють робити різні операції над таблицями «Паливо», «Постачання» та «Продаж» такі, як додавання нового запису, редагування запису, видалення запису. П’ятий пункт дозволяє відобразити чи сховати панель фільтру. Шостий пункт меню дозволяє відкрити форму «О программе» ,а сьомий пункт – вийти з програми.

Якщо клацнути правою кнопкою миші по таблиці, з’явиться контекстне меню для роботи з таблицею. Контекстне меню дозволяє робити ті ж самі операції над таблицями, що і в звичайному меню.

У верхній часті форми, нижче меню, розташована інструментальна панель, яка дублює функції меню та контекстного меню. А також дозволяє оновити дані в таблицях.

На кожній сторінці розташовані таблиці.

На першій сторінці розташована таблиця для відображення палива. На другій сторінці розташована таблиця для відображення поставок палива. Приклад форми зображений на рисунку 4.2.

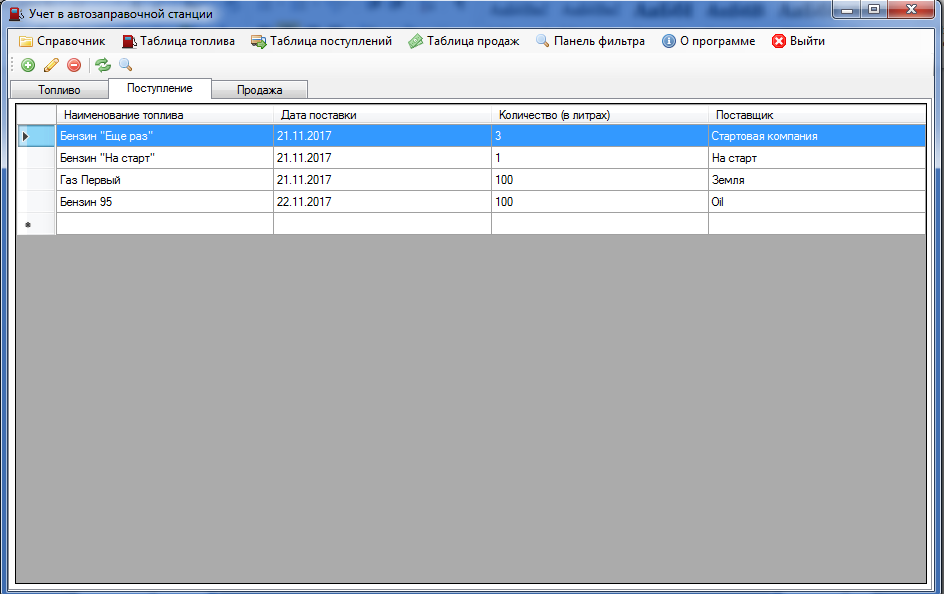


Рисунок 4.2 – Головне вікно програми, друга сторінка

На третій сторінці розташована таблиця продаж палива. Приклад форми зображений на рисунку 4.3.

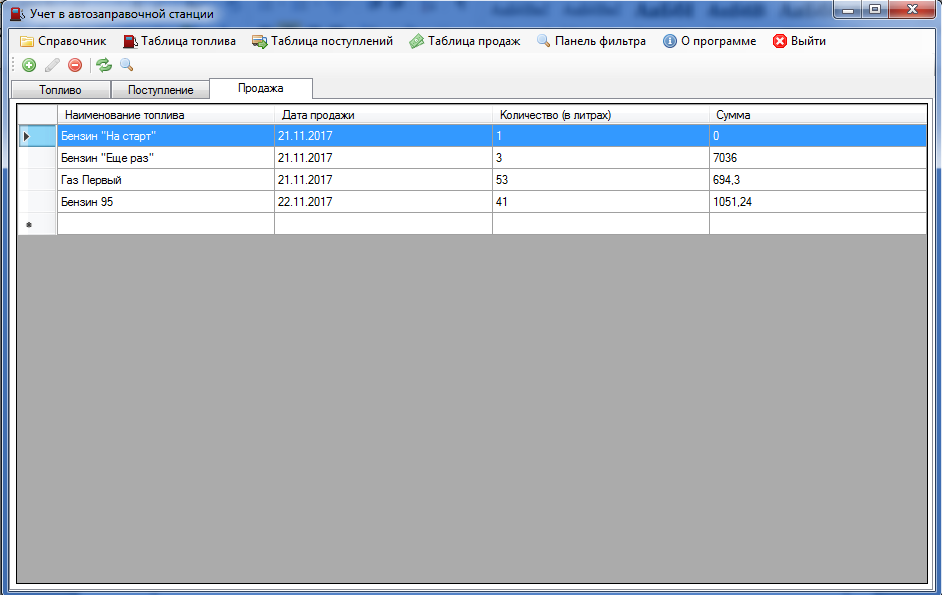


Рисунок 4.3 – Головне вікно програми, третя сторінка

Панель фільтру записів розташована справа на формі. Зазвичай, вона вимкнута, щоб не заважати перегляду таблиці. Для відкриття панелі фільтрів потрібно вибрати пункт меню «Панель фильтров», чи аналогічні дублюючі функції. Кожна таблиця має свої фільтри. Усі фільтри працюють як окрема, так і усі разом. Фільтрація здійснюється після натискання кнопки «Фильтровать». Для вимикання фільтру потрібно натиснути кнопку оновлення в меню інструментів . Приклад фільтру зображений на рисунку 4.4.

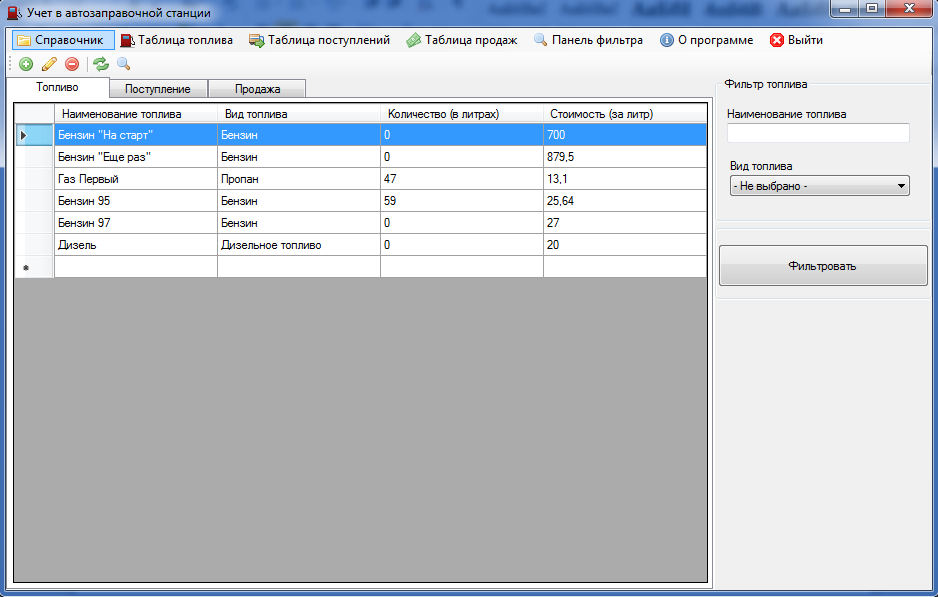


Рисунок 4.4 – Головне вікно програми з панеллю фільтрів

Для того, щоб додати новий вид палива, треба клацнути по пункту меню «Довідники». Після цього з’явиться вікно, яке зображено на рисунку 4.5.

На формі розташована таблиця видів палива, панель фільтру, та меню з пунктами для роботи з таблицею. Для додавання нового запису потрібну обрати «Добавить», після чого з’явиться панель додавання у режимі додавання. Потрібно ввести назву виду, та обрати «Зберегти». Після чого програма почне перевіряти данні. Якщо був знайдений схожий запис, то програма не пропустить такі данні. Редагування працює так само як і додавання. Для видалення потрібно обрати пункт «Удалить».

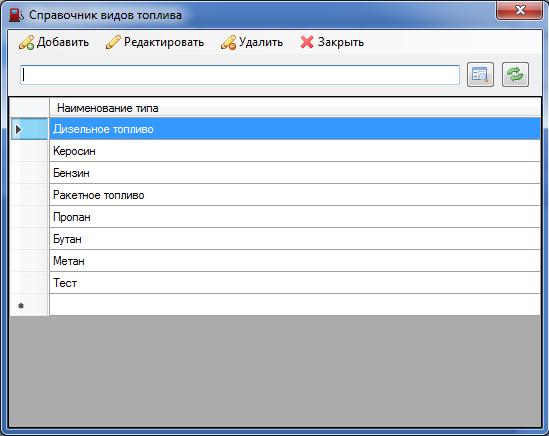


Рисунок 4.5 – Форма «Довідник»

Для того, щоб додати нове паливо, треба клацнути правою кнопкою миші по таблиці та обрати пункт «Добавить». Після цього з’явиться вікно, яке зображено на рисунку 4.6.

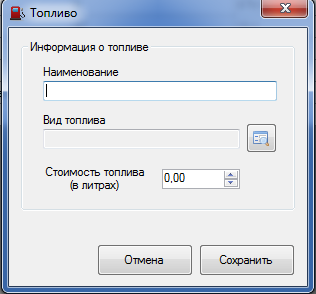


Рисунок 4.6 – Форма додавання палива

Потім ввести в усі поля текст, після чого натиснути кнопку «Сохранить» і запис буде доданий до таблиці. Для виходу чи відміни додавання потрібно натиснути на кнопку «Отменить».

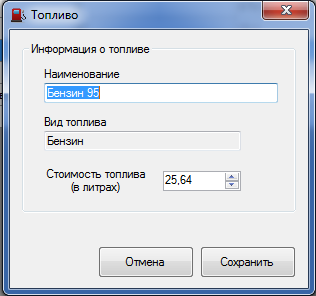


Рисунок 4.7 – Форма редагування палива

Для того, щоб редагувати запис палива, треба клацнути правою кнопкою миші по обраному запису з таблиці та обрати пункт «Редактировать». Після цього з’явиться вікно, яке зображено на рисунку 4.7.

Після чого з’явиться вікно «Додавання запису» (див. рис 4.6) тільки у режимі редагування запису. Працювати з цим вікно так само як із вікном «Додаванням даних нового палива».

Щоб видалити обраний палива, потрібно клацнути правою кнопкою миші по обраному запису з таблиці та обрати пункт «Удалить». Програма запитає в вас підтвердження видалення, після чого запис буде видалений.

Щоб додати інформацію про постачання палива, потрібно перейти на вкладку «Постачання» у головній формі, та за допомогою меню чи клацнувши правою кнопкою миші та обрати пункт «Добавить». Після цього з’явиться вікно, яке зображено на рисунку 4.8.

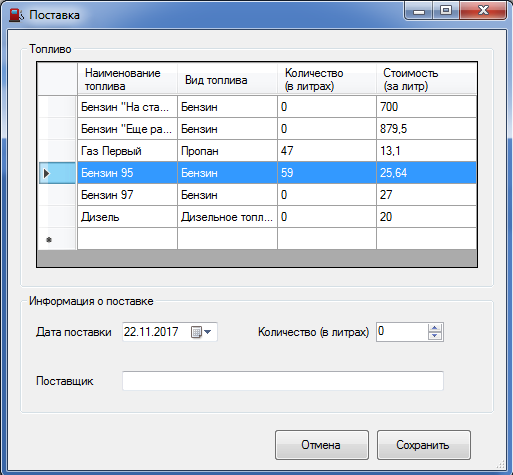


Рисунок 4.8 – Форма додавання інформації про постачання палива

У відкритому вікні потрібно вибрати паливо з таблиці, вказати дату постачання, кількість та вказати постачальника. Щоб зберегти потрібно натиснути «Сохранить», а для того щоб відмінити – «Отменить».

Редагування інформації про постачання працює так само, що і додавання інформації. Приклад форми додавання у режимі редагування зображений на рисунку 4.9.

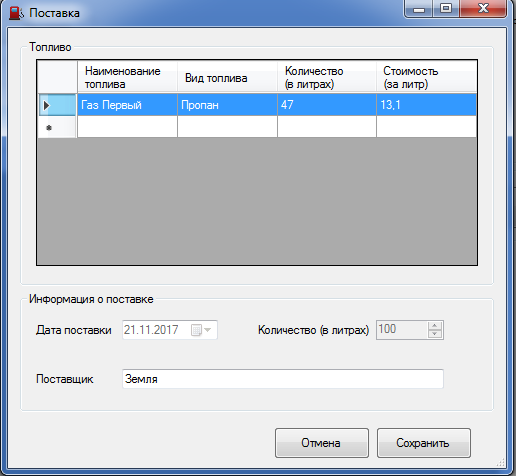


Рисунок 4.9 – Форма додавання у режимі редагування

На відмінну від додавання, при редагуванні неможливо змінити паливо, кількість та дату.

Для додавання інформації про продаж палива, потрібно перейти на останню сторінку та обрати «Добавить» з пункту меню, чи з панелі інструментів, чи за допомогою миші. Приклад форми додавання результату зображений на рисунку 4.10.

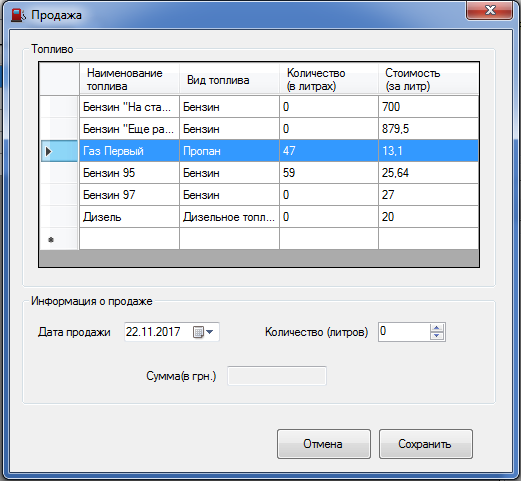


Рисунок 4.10 – Форма додавання інформації про продаж палива

У відкритому вікні потрібно вибрати паливо з таблиці, вказати дату продажу та кількість палива. Вартість продажу розрахується автоматично. Щоб зберегти потрібно натиснути «Сохранить», а для того щоб відмінити – «Отменить».

Редагування інформації про продаж неможливе, по причині, щоб уникнути маніпулювання інформацією про продаж.

5 ОХОРОНА ПРАЦІ

Охорона праці — це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я та працездатності людини в процесі праці.

Відповідно до статті 18 Закону України «Про охорону праці», працівник зобов'язаний «знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з машинами, механізмами, устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту, проходити у встановленому порядку попередні та періодичні медичні огляди» [6].

5.1Шкідливі та небезпечні чинники під час розробки автоматизованого робочого місця

Аналіз шкідливих та небезпечних виробничих факторів проводять згідно з ГОСТ 12.1.003-74. Небезпечними та шкідливими називають фактори, що призводять до раптового погіршення здоров’я людини, або навіть смерті, під час її трудової діяльності.

Згідно з класифікацією ГОСТ 12.1.003-74 до числа небезпечних та шкідливих факторів, що виникають при роботі з ПК слід віднести:

* нервово-психічні перевантаження (розумове перевантаження, монотонність праці, емоційні перевантаження);
* фізичне перевантаження;
* невідповідність параметрів мікроклімату робочої зони санітарним нормам;
* недостатня або надмірна освітленість робочої зони;
* несприятливе забарвлення стін та підлоги, віддзеркалення;
* робота з комп’ютером.

Робоче місце призначено для оператора, що працює з програмною системою обліку студентів коледжу, які навчаються за контрактом. Кількість робочих місць — 1. Робоче місце оснащене ПК, принтером, монітором та іншими периферійними пристроями. Розміщення принтера або інших пристроїв введення-виведення інформації на робочому місці має забезпечувати добру видимість екрану, зручність ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля: по висоті від 900 до 1300 мм, по глибині від 400 до 500 мм. Під матричні принтери потрібно підкладати вібраційні килимки для гасіння вібрації та шуму.

Площа, виділена для одного робочого місця ПК повинна складати не менше 6 м2, а об'єм — не менше 20 м3. Робочі місця з моніторами відносно світлових прорізів повинні розміщуватися так, щоб природне світло падало збоку, переважно зліва.

Режим праці та відпочинку.

При організації праці, пов'язаної з використанням ПК, для збереження здоров'я працівника передбачаються:

* перерви для відпочинку і вживання їжі;
* перерви для відпочинку й особистих потреб;
* додаткові перерви.

5.2 Інженерно-технічні заходи охорони праці

Інженерно-технічні заходи передбачають впровадження колективних заходів забезпечення сприятливих мікрокліматичних та зорових умов праці на робочих місцях, заходи захисту від впливу шкідливих речовин у повітрі робочої зони, від шуму, ультразвуку, вібрації, електромагнітного випромінювання, іонізуючого випромінювання, а також заходи попередження ураження електричним струмом, виникнення пожеж та аварій під час експлуатації технологічного устаткування.

Об'ємно-планувальні рішення будівель і приміщень для роботи з моніторами, ПК, ноутбуками мають відповідати вимогам чинних нормативних актів.

Заборонено розміщувати робочі місця з моніторами, ПК та ноутбуками у підвальних приміщеннях, на цокольних поверхах, поряд з приміщеннями, в яких рівні шуму та вібрації перевищують допустимі значення (поряд з механічними цехами, майстернями тощо), з мокрими виробництвами, з вибухопожежонебезпечними приміщеннями категорій А і Б, а також над такими приміщеннями або під ними.

Приміщення мають бути оснащені природним і штучним освітленням. Приміщення мають бути обладнані системами водяного опалення, кондиціонування або припливно-витяжною вентиляцією. Заземлені конструкції приміщення (батареї опалення, водопровідні труби, кабелі з заземленим відкритим екраном тощо) надійно захищені діелектричними щитками або сітками від випадкового дотику.

Приміщення з моніторами, ПК і ноутбуками оснащують системою пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 од. на кожні 20 м2 площі приміщення. Підходи до засобів пожежогасіння мають бути вільними.

Робочі місця з моніторами, щодо світлових прорізів розміщують так, щоб природне світло падало збоку, переважно зліва.

Екран монітора і клавіатура мають розміщуватися на оптимальний відстані від очей користувача, але не ближче 600 мм з урахуванням розміру алфавітно-цифрових знаків і символів. Відстань від екрана до ока працівника залежить від діагоналі екрана

Уміст шкідливих речовин у повітрі робочої зони не має перевищувати ГДК відповідно до ГОСТ 12.1.005-88 уміст озону не більше 0,1 мг/м3, вміст оксидів азоту — не більше 5 мг/м3. уміст пилу — не більше 4 мг/м3.

Оптимальні величини температури, відносної вологості та швидкості руху повітря для приміщень з моніторами, ПК і ноутбуками наведені у таблиці 5.2.

Таблиця 5.2 – Оптимальні величини температури, відносної вологості та швидкості руху повітря для приміщень з моніторами, ПК і ноутбуками

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Період року | Категорія робіт | Температура повітря, °С | Відносна вологість повітря, % | Швидкість руху повітря, м/с |
| холодний | Легка-1 а | від 22 до 24 | від 40 до 60 | 0,1 |
| Легка-16 | від 21 до 23 | від 40 до 60 | 0,1 |
| теплий | Легка-1а | від 23 до 25 | від 40 до 60 | 0,1 |
| Легка-16 | від 22 до 24 | від 40 до 60 | 0,2 |

Рівні іонізації повітря приміщень під час роботи з моніторами, ПК і ноутбуками наведено у таблиці 5.3.

Таблиця 5.3 – Рівні іонізації повітря приміщень під час роботи з моніторами, ПК і ноутбуками

|  |  |  |
| --- | --- | --- |
| Рівні іонізації повітря | Кількість іонів у 1 см3 повітря | |
| n+ | n- |
| Мінімально необхідні | 400 | 600 |
| Оптимальні | від 1500 до 3000 | від 3000 до 5000 |
| Максимально допустимі | 50000 | 50000 |

Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот представлений в таблиці 5.4.

Таблиця 5.4 – Допустимі рівні звуку, еквівалентні рівні звуку, рівні звукового тиску в октавних смугах частот

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Види трудової діяльності | Рівні звукового тиску, дБ, в октавних смугах із середньо геометричними частотами, Гц | | | | | | | | | Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв |
| 31.5 | 63 | 125 | 250 | 500 | 1000 | 2000 | 3000 | 4000 |
| Програмісти ПК | 86 | 71 | 61 | 54 | 49 | 45 | 42 | 40 | 38 | 50 |
| Оператори в залах оброблення інформації на ПК та оператори комп'ютерного набору | 96 | 83 | 74 | 68 | 63 | 60 | 57 | 55 | 54 | 65 |

Для підтримання оптимальних значень параметрів повітря робочої зони потрібно застосовувати вентиляцію приміщень, кондиціонування повітря, використовувати установки або прилади зволожування та штучної іонізації.

5.3 Пожежна профілактика

Пожежна профілактика - комплекс організаційних і технічних заходів щодо попередження, локалізації та ліквідації пожеж, а також щодо забезпечення безпечної евакуації людей та матеріальних цінностей у разі пожеж.

Пожежна безпека – це такий стан промислового об'єкта, при якому виключається можливість пожежі, а в разі його виникнення попереджається вплив на людей небезпечних факторів і забезпечується захист матеріальних цінностей. Пожежі завдають величезних матеріальних збитків, призводять до травм та загибелі людей, так як супроводжуються виникненням небезпечних чинників, таких як відкритий вогонь, підвищена температура, токсичні речовини, дим, недолік кисню, ушкодження і порушення будівель, споруд, вибухи технічного обладнання тощо. Тому виконання правил пожежної безпеки на підприємствах є обов'язковим для всіх посадових осіб і громадян. Основи пожежної безпеки закладаються на стадії проектування підприємства, будівлі, споруди, планування технологічного процесу, встановлення обладнання, тобто враховується інженерно – технологічними заходами, які представлені в проектах при розробці проектної документації на будівництво, і вимагає суворого дотримання протипожежних правил у процесі експлуатації.   
Пожежна безпека промислових підприємств складається з системи попередження пожеж, системи пожежного захисту і організаційно-технічних заходів.

Система попередження пожеж – це комплекс організаційних і технічних засобів, спрямованих на виключення можливості виникнення пожеж, на запобігання утворення горючої і вибухонебезпечного середовища шляхом регламентації вмісту горючих газів, парів і пилу в повітрі, а також виключення можливості виникнення джерел загоряння або вибуху; забезпечення пожежної безпеки технологічних процесів, обладнання, електрообладнання.

Пожежна профілактика є складовою частиною технологічних процесів виробництва, містобудування, планування і забудови сільських населених місць. Організацією профілактики займаються органи пожежного нагляду.

Пожежна профілактика досягається:

* розробкою, впровадженням та контролем за дотриманням пожежних норм і правил;
* веденням конструювання та планування з урахуванням пожежної безпеки створюваних об'єктів;
* вдосконаленням і змістом в готовності протипожежних засобів;
* регулярним проведенням пожежно-технічних обстежень промислових і с / г підприємств, організацій, житлових і громадських будівель;
* пропагандою пожежно-технічних знань серед населення.

6 ЕКОНОМІЧНИЙ РОЗДІЛ

Згідно із завданням кваліфікаційного проекту необхідно визначити собівартість і ціну розробленого проекту. Для виконання розрахунку були використані початкові дані, представлені в таблиці 6.1.

Таблиця 6.1 — Початкові дані для розрахунку

|  |  |  |
| --- | --- | --- |
| Найменування початкових даних | Показник | Джерело отримання |
| 1. Трудомісткість складання додатку | 12 днів (12\*8=96год) | Фактичні витрати часу на розробку програми |
| 2. Місячна ставка програміста | 10000,00 грн | Дані переддипломної практики |
| 3. Кількість годин в місяці | 168 год | Кількість робочих днів - 21 днів |
| 4. Додаткова зарплата | 10% | Дані переддипломної практики |
| 5. Нарахування єдиного соціального внеску (з 01.01 2016 р. — 22%) | 22% | Дані переддипломної практики |
| 6. Загальновиробничі витрати (%) | 10% | Дані переддипломної практики |

Стаття 1. Матеріали

Матеріали для даного проекту не використовуються.

Стаття 2. Електроенергія

Витрати на технологічну енергію (ВЕ) розраховуються за формулою:

|  |  |
| --- | --- |
| , | (6.2) |

де РЕ — ціна за 1 кВт-год, грн;

Wі — середня потужність, що споживається



Стаття 3. Основна заробітна плата:

|  |  |
| --- | --- |
| , | (6.3) |

де  — годинна тарифна ставка оператора, грн.;

 — кількість годин у місяці, приймається 168 годин — вихідні дані.



Визначаємо годинну тарифну ставку програміста:

|  |  |
| --- | --- |
| , | (6.4) |

де  — місячна ставка оператора, грн.



Стаття 4. Додаткова заробітна плата:

|  |  |
| --- | --- |
| , | (6.4) |

Де  — додаткова заробітна плата, грн.;

— відсоток додаткової заробітної плати, приймається 10% — вихідні дані.



Стаття 5. Нарахування єдиного соціального внеску:

|  |  |
| --- | --- |
| , | (6.5) |



Стаття 6. Загальновиробничі витрати:

|  |  |
| --- | --- |
| , | (6.6) |

де  — загальновиробничі витрати, грн.;

 — відсоток загальновиробничих витрат, приймається 10% — вихідні дані.



Виробнича собівартість:

|  |  |
| --- | --- |
|  | (6.7) |



Розрахунок вартості проекту представлено у таблиці 6.2.

Таблиця 6.2 — Планова калькуляція виробничої собівартості на виконання розробки програми

|  |  |
| --- | --- |
| Статті калькуляції | Сума, грн. |
| Стаття 1 Матеріали | 0 |
| Стаття 2 Електроенергія | 41,76 |
| Стаття 3 Основна заробітна плата | 5713,92 |
| Стаття 4 Додаткова заробітна плата | 571,39 |
| Стаття 5 Відрахування в соціальні фонди | 1257,06 |
| Стаття 6 Загальновиробничі витрати | 571,39 | |
| Виробнича собівартість | 8045,36 | |

Висновок: таким чином розрахунок показав, що собівартість розробленого проекту складає 8045,36 грн., при цьому найбільша питома вага складають витрати на оплату праці.

# ВИСНОВКИ

В ході виконання кваліфікаційної роботи був розроблений програмний додаток, що дозволяє виконувати перегляд бази даних, додавання інформації, редагування та видалення, сортування даних, фільтрування даних.

Програмний додаток має досить зрозумілий інтерфейс та містить підказки та перевірки коректності введених даних, що дозволяє працювати з ним людям з мінімальними навиками володіння комп’ютером. Отже цей програмний додаток можна використовувати у автозаправних станціях.

У подальшому цей проект можна буде вдосконалити додавши у нього функцію формування та друкування документу, також удосконалити програмний код таким чином, щоб програма могла працювати швидше з великим об’ємом даних, тобто ще більш оптимізувати програму та поступово перевести її у розряд клієнт-серверних.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шилд Г. Повний довідник по С++, 2006 р. -797с.
2. Страуструп Б. Язык программирования C++ .Пер. с англ. //СПб.; М, 1999. — 560 с.
3. Грицюк Ю. , Рак Т., Програмування мовою С++ ,2011 р. - 292с.
4. Інтернет ресурс: https://docs.microsoft.com/en-us/framework/winforms/index
5. Інтернет ресурс: <https://sites.google.com/site/javatokens/sqlitе>

Додаток

Лістинг програми

Лістинг 1.1 - About.h

#pragma once

namespace GasStation {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для About

/// </summary>

public ref class About : public System::Windows::Forms::Form

{

public:

About(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~About()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

protected:

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Button^ Closebutton;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(About::typeid));

this->label1 = (gcnew System::Windows::Forms::Label());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label3 = (gcnew System::Windows::Forms::Label());

this->Closebutton = (gcnew System::Windows::Forms::Button());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(37, 23);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(313, 24);

this->label1->TabIndex = 0;

this->label1->Text = L"Курсовой проект был разработан";

//

// label2

//

Лістинг 1.2 - ComingEdit.h

#pragma once

#include "DataModul.h"

namespace GasStation {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Сводка для ComingEdit

/// </summary>

public ref class ComingEdit : public System::Windows::Forms::Form

{

public:

DataModul^ DM;

int EditMode;

String^ countCur;

ComingEdit(DataModul^ DM1, int EM)

{

EditMode = EM;

DM = DM1;

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~ComingEdit()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::GroupBox^ FuelBox;

private: System::Windows::Forms::DataGridView^ FuelGrid;

private: System::Windows::Forms::GroupBox^ ComingBox;

private: System::Windows::Forms::NumericUpDown^ CountEdit;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::DateTimePicker^ DateEdit;

private: System::Windows::Forms::Button^ Cancelbutton;

private: System::Windows::Forms::Button^ Savebutton;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ ProviderBox;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(ComingEdit::typeid));

this->FuelBox = (gcnew System::Windows::Forms::GroupBox());

this->FuelGrid = (gcnew System::Windows::Forms::DataGridView());

this->ComingBox = (gcnew System::Windows::Forms::GroupBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->ProviderBox = (gcnew System::Windows::Forms::TextBox());

this->CountEdit = (gcnew System::Windows::Forms::NumericUpDown());

this->label2 = (gcnew System::Windows::Forms::Label());

this->label1 = (gcnew System::Windows::Forms::Label());

this->DateEdit = (gcnew System::Windows::Forms::DateTimePicker());

this->Cancelbutton = (gcnew System::Windows::Forms::Button());

this->Savebutton = (gcnew System::Windows::Forms::Button());

this->FuelBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->FuelGrid))->BeginInit();

this->ComingBox->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->CountEdit))->BeginInit();

this->SuspendLayout();

//

// FuelBox

//

this->FuelBox->Controls->Add(this->FuelGrid);

this->FuelBox->Location = System::Drawing::Point(12, 12);

this->FuelBox->Name = L"FuelBox";

this->FuelBox->Size = System::Drawing::Size(474, 246);

this->FuelBox->TabIndex = 0;

this->FuelBox->TabStop = false;

this->FuelBox->Text = L"Топливо";

//

// FuelGrid

//

this->FuelGrid->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;

this->FuelGrid->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->FuelGrid->Location = System::Drawing::Point(16, 19);

this->FuelGrid->MultiSelect = false;

this->FuelGrid->Name = L"FuelGrid";

this->FuelGrid->ReadOnly = true;

this->FuelGrid->SelectionMode = System::Windows::Forms::DataGridViewSelectionMode::FullRowSelect;

this->FuelGrid->Size = System::Drawing::Size(442, 207);

this->FuelGrid->TabIndex = 1;

this->FuelGrid->CellClick += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &ComingEdit::FuelGrid\_CellClick);

//

// ComingBox

//

this->ComingBox->Controls->Add(this->label3);

this->ComingBox->Controls->Add(this->ProviderBox);

this->ComingBox->Controls->Add(this->CountEdit);

this->ComingBox->Controls->Add(this->label2);

this->ComingBox->Controls->Add(this->label1);

this->ComingBox->Controls->Add(this->DateEdit);

this->ComingBox->Location = System::Drawing::Point(12, 264);

this->ComingBox->Name = L"ComingBox";

this->ComingBox->Size = System::Drawing::Size(474, 120);

this->ComingBox->TabIndex = 1;

this->ComingBox->TabStop = false;

this->ComingBox->Text = L"Информация о поставке";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Location = System::Drawing::Point(13, 80);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(65, 13);

this->label3->TabIndex = 5;

this->label3->Text = L"Поставщик";

//

// ProviderBox

//

this->ProviderBox->Location = System::Drawing::Point(102, 77);

this->ProviderBox->Name = L"ProviderBox";

this->ProviderBox->Size = System::Drawing::Size(322, 20);

this->ProviderBox->TabIndex = 4;

//

// CountEdit

//

this->CountEdit->Location = System::Drawing::Point(356, 28);

this->CountEdit->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 100000000, 0, 0, 0 });

this->CountEdit->Name = L"CountEdit";

this->CountEdit->Size = System::Drawing::Size(68, 20);

this->CountEdit->TabIndex = 3;

this->CountEdit->Leave += gcnew System::EventHandler(this, &ComingEdit::CountEdit\_Leave);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Location = System::Drawing::Point(235, 31);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(118, 13);

this->label2->TabIndex = 2;

this->label2->Text = L"Количество (в литрах)";

//

// label1

//

this->label1->AutoSize = true;

this->label1->Location = System::Drawing::Point(13, 31);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(83, 13);

this->label1->TabIndex = 1;

this->label1->Text = L"Дата поставки";

//

// DateEdit

//

this->DateEdit->Format = System::Windows::Forms::DateTimePickerFormat::Short;

this->DateEdit->Location = System::Drawing::Point(102, 28);

this->DateEdit->Name = L"DateEdit";

this->DateEdit->Size = System::Drawing::Size(96, 20);

this->DateEdit->TabIndex = 0;

//

// Cancelbutton

//

this->Cancelbutton->Location = System::Drawing::Point(266, 399);

this->Cancelbutton->Name = L"Cancelbutton";

this->Cancelbutton->Size = System::Drawing::Size(96, 32);

this->Cancelbutton->TabIndex = 4;

this->Cancelbutton->Text = L"Отмена";

this->Cancelbutton->UseVisualStyleBackColor = true;

this->Cancelbutton->Click += gcnew System::EventHandler(this, &ComingEdit::Cancelbutton\_Click);

//

// Savebutton

//

this->Savebutton->Location = System::Drawing::Point(368, 399);

this->Savebutton->Name = L"Savebutton";

this->Savebutton->Size = System::Drawing::Size(96, 32);

this->Savebutton->TabIndex = 3;

this->Savebutton->Text = L"Сохранить";

this->Savebutton->UseVisualStyleBackColor = true;

this->Savebutton->Click += gcnew System::EventHandler(this, &ComingEdit::Savebutton\_Click);

//

// ComingEdit

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(498, 440);

this->Controls->Add(this->Cancelbutton);

this->Controls->Add(this->Savebutton);

this->Controls->Add(this->ComingBox);

this->Controls->Add(this->FuelBox);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->MaximizeBox = false;

this->MinimizeBox = false;

this->Name = L"ComingEdit";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Поставка";

this->Load += gcnew System::EventHandler(this, &ComingEdit::ComingEdit\_Load);

this->FuelBox->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->FuelGrid))->EndInit();

this->ComingBox->ResumeLayout(false);

this->ComingBox->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->CountEdit))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void Cancelbutton\_Click(System::Object^ sender, System::EventArgs^ e)

{

auto result = MessageBox::Show("Вы действительно хотите отменить редактирование?", "Подтвердите действие", MessageBoxButtons::YesNo, MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

EditMode = 0;

Close();

}

}

private: System::Void Savebutton\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (DM->GetIDFuelC() != "")

{

if (ProviderBox->Text != "")

{

if (CountEdit->Value > 0)

{

String^ request;

//Добавление

if (EditMode == 1)

{

request = "INSERT INTO Coming";

request = request + " (IDGas,DateComing,Count,Provider) ";

request = request + " VALUES('" + DM->GetIDFuelC() + "','" +DateEdit->Text + "','" + CountEdit->Text + "','" + ProviderBox->Text + "');";

DM->SQLrequest(request);

}

//Редактирование

if (EditMode == 2)

{

request = " UPDATE Coming ";

request = request + " SET Provider = '" + ProviderBox->Text + "'";

request = request + " WHERE id = " + DM->GetIDComingFuel() + ";";

DM->SQLrequest(request);

}

if (EditMode == 1)

{

int countGas = DM->StringToInt(countCur);

int count = DM->StringToInt(CountEdit->Text);

count = count + countGas;

request = " UPDATE Gas ";

request = request + " SET Count = '" + count + "'";

request = request + " WHERE id = " + DM->GetIDFuelC() + ";";

DM->SQLrequest(request);

}

DM->CleanComingFuel();

Close();

}

else MessageBox::Show("Количество не может быть 0", "Ошибка ввода");

}

else MessageBox::Show("Не указан поставщик", "Ошибка ввода");

}

else MessageBox::Show("Не выбрано наименование топлива", "Ошибка ввода");

}

private: System::Void ComingEdit\_Load(System::Object^ sender, System::EventArgs^ e)

{

try

{

if (EditMode == 1)

{

if (FuelGrid->Rows->Count > 0) DM->SetIndex(FuelGrid->CurrentCell->RowIndex);

FuelGrid->DataSource = DM->fillDataTable("SELECT Gas.id,Gas.NameGas,TypeGas.NameType,Gas.Count,Gas.Price,Gas.IDType FROM Gas,TypeGas WHERE TypeGas.ID = Gas.IDType;");

if (FuelGrid->Rows->Count > 1)

{

FuelGrid->Rows[DM->GetIndex()]->Selected = true;

FuelGrid->CurrentCell = FuelGrid->Rows[DM->GetIndex()]->Cells[1];

}

FuelGrid->Columns[0]->Visible = false;

FuelGrid->Columns[1]->HeaderText = "Наименование топлива";

FuelGrid->Columns[2]->HeaderText = "Вид топлива";

FuelGrid->Columns[3]->HeaderText = "Количество (в литрах)";

FuelGrid->Columns[4]->HeaderText = "Стоимость (за литр)";

FuelGrid->Columns[5]->Visible = false;

}

if (EditMode == 2)

{

CountEdit->Enabled = false;

DateEdit->Enabled = false;

FuelGrid->DataSource = DM->fillDataTable("SELECT Gas.id,Gas.NameGas,TypeGas.NameType,Gas.Count,Gas.Price,Gas.IDType FROM Gas,TypeGas WHERE TypeGas.ID = Gas.IDType AND Gas.id = '"+DM->GetIDFuelC()+"';");

FuelGrid->Columns[0]->Visible = false;

FuelGrid->Columns[1]->HeaderText = "Наименование топлива";

FuelGrid->Columns[2]->HeaderText = "Вид топлива";

FuelGrid->Columns[3]->HeaderText = "Количество (в литрах)";

FuelGrid->Columns[4]->HeaderText = "Стоимость (за литр)";

FuelGrid->Columns[5]->Visible = false;

ProviderBox->Text = DM->GetProvider();

CountEdit->Text = DM->GetCountComing();

DateEdit->Text = DM->GetDateComing();

}

}

catch (Exception ^e)

{

MessageBox::Show(e->ToString(), "Ошибка");

}

}

private: System::Void FuelGrid\_CellClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e)

{

if (EditMode == 1)

{

DM->SetIndex(FuelGrid->CurrentCell->RowIndex);

DM->SetIDFuelC(FuelGrid->Rows[DM->GetIndex()]->Cells["id"]->Value->ToString());

countCur = FuelGrid->Rows[DM->GetIndex()]->Cells["count"]->Value->ToString();

}

}

private: System::Void CountEdit\_Leave(System::Object^ sender, System::EventArgs^ e)

{

}

};

}

Лістинг 1.3 - DataModul.h

#pragma once

#include <cliext/vector>

namespace GasStation {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Data::SQLite;

using namespace cliext;

/// <summary>

/// Сводка для DataModul

/// </summary>

public ref class DataModul : public System::Windows::Forms::Form

{

//\*\*\*\*\*\*\*\*TypeFuel\*\*\*\*\*\*\*\*

String^ IDType;

String^ NameType;

//\*\*\*\*\*\*\*\*Fuel\*\*\*\*\*\*\*\*\*\*\*\*

String^ IDFuel;

String^ NameFuel;

String^ CountFuel;

String^ IDFuelType;

String^ PriceFuel;

//\*\*\*\*\*\*\*\*Coming\*\*\*\*\*\*\*\*\*\*

String^ IDComingFuel;

String^ IDFuelC;

String^ CountComing;

String^ DateComing;

String^ Provider;

//\*\*\*\*\*\*\*\*Sell\*\*\*\*\*\*\*\*\*\*\*\*

String^ IDSellFuel;

String^ IDFuelS;

String^ CountSell;

String^ DateSell;

String^ Price;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int RowSelect;

public:

SQLiteConnection ^db;

private: System::Windows::Forms::DataGridView^ GridBuf;

public:

String^ fileName = "gasStation.db";

DataModul(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~DataModul()

{

CloseConnection();

if (components)

{

delete components;

}

}

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

this->GridBuf = (gcnew System::Windows::Forms::DataGridView());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->GridBuf))->BeginInit();

this->SuspendLayout();

//

// GridBuf

//

this->GridBuf->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->GridBuf->Location = System::Drawing::Point(130, 184);

this->GridBuf->Name = L"GridBuf";

this->GridBuf->Size = System::Drawing::Size(90, 54);

this->GridBuf->TabIndex = 1;

//

// DataModul

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(263, 174);

this->Controls->Add(this->GridBuf);

this->Name = L"DataModul";

this->Text = L"DataModul";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->GridBuf))->EndInit();

this->ResumeLayout(false);

}

#pragma endregion

public: void OpenConnection()

{

db = gcnew SQLiteConnection();

db->ConnectionString = "Data Source=\"" + fileName + "\"";

db->Open();

}

public: void CloseConnection()

{

db->Close();

}

public: DataTable^ fillDataTable(String^ request)

{

DataTable ^table;

try

{

SQLiteCommand ^cmdSelect = db->CreateCommand();

//Обратите внимание, что SQL запрос оформляем как обычную строчку

cmdSelect->CommandText = request;

SQLiteDataReader ^reader = cmdSelect->ExecuteReader();

DataColumn ^column; //Столбец таблицы

DataRow ^row; //Строка таблицы

//Создаем таблицу данных

table = gcnew DataTable();

//Вектор названий столбцов

vector<String^>^ nameColumns = gcnew vector<String^>();

//Заполним данные о столбцах

for (int i = 0; i < reader->FieldCount; i++) {

nameColumns->push\_back(reader->GetName(i));

column = gcnew DataColumn(nameColumns->at(i), String::typeid);

table->Columns->Add(column);

}

//Пробегаем по каждой записи

while (reader->Read()) {

//Заполняем строчку таблицы

row = table->NewRow();

//В каждой записи пробегаем по всем столбцам

for (int i = 0; i < reader->FieldCount; i++) {

//Добавляем значение столбца в row

row[nameColumns->at(i)] = reader->GetValue(i)->ToString();