ТЕХНІЧНИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ

Циклова комісія програмних систем і комплексів

**КУРСОВА РОБОТА**

з дисципліни:

**«Об’єктно-орієнтоване програмування»**

на тему: **«Розробка системи обліку автозапчастин магазину «AutoDrive»»**

Студента 3 курсу групи КН-321 спеціальності 122 «Комп’ютерні науки»

Демчишин М. М.

(прізвище та ініціали)

Керівник: викладач   Слободян  Р.О.         
Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_ Оцінка: ECTS \_\_\_\_

Члени комісії:\_\_\_\_\_\_\_\_\_\_\_ Р.О. Слободян

(підпис)

\_\_\_\_\_\_\_\_\_ Г.Я. Марціяш

(підпис)

м. Тернопіль – 2019

ЗМІСТ

[ВСТУП 4](#_Toc26434668)

[1 ТЕХНІЧНЕ ЗАВДАННЯ 5](#_Toc26434669)

[1.1 Найменування та область застосування 5](#_Toc26434670)

[1.2 Підстави для розробки 5](#_Toc26434671)

[1.3 Призначення розробки 5](#_Toc26434672)

[1.4 Вимоги до програми чи програмного виробу](#_Toc26434673) 6

[1.5 Вимоги до програмної документації 9](#_Toc26434674)

[1.6 Техніко-економічні показники 10](#_Toc26434675)

[1.7 Стадії та етапи розробки 10](#_Toc26434676)

[1.8 Порядок контролю та прийому 11](#_Toc26434677)

[2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ 12](#_Toc26434678)

[2.1 Розробка загальної структури і варіантів використання програми 12](#_Toc26434679)

[2.2 Розробка системи класів 15](#_Toc26434680)

[2.3 Розробка методів 15](#_Toc26434681)

[2.4 Проєктування і опис інтерфейсу користувача 16](#_Toc26434682)

[2.5 Опис файлової структури програми 18](#_Toc26434683)

[2.6 Опис структури бази даних програми 19](#_Toc26434684)

[3 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 21](#_Toc26434685)

[ВИСНОВКИ 25](#_Toc26434686)

[ПЕРЕЛІК ПОСИЛАНЬ 26](#_Toc26434687)

[Додаток А Діаграма варіантів використання 27](#_Toc26434688)

[Додаток Б Лістинг файлу «program.h» 28](#_Toc26434689)

[Додаток В Лістинг файлу «program.cpp» 32](#_Toc26434690)

[Додаток Г Компакт-диск із програмним продуктом 40](#_Toc26434691)

ВСТУП

Сучасні технології програмування переживають період бурхливого розвитку. Причинами цього явища є зростання потужностей комп’ютерної техніки, виникнення все нових і нових сфер застосування комп’ютерів і потреба у різноманітному програмному забезпеченні.

Основним стало питання розробки виразної структури програми, придатної до легкої модифікації, вільної від помилок, стійкої до змін. Об’єктно-орієнтоване програмування (ООП) – це технологія програмування, яка ґрунтується на понятті класів, об’єктів та успадкуванні елементів базових класів похідними класами. Основу ООП складають чотири основні концепції:[інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))та абстракція. Об’єктно-орієнтоване програмування засноване на представленні програми у вигляді сукупності об'єктів, кожен з яких є реалізацією певного класу. Об'єкт являє собою елемент програми. Програмний код і дані структуруються так, щоб імітувалося поведінку фактично існуючих об'єктів. Внутрішній вміст об'єкта захищається від зовнішнього світу за допомогою інкапсуляції. Успадкування дозволяє виділити властивості, методи і події одного об'єкту і приписати їх для іншого об'єкта. Завдяки поліморфізму можна використовувати різні об'єкти і по-різному реалізовані функції під одним ім'ям. Абстракція дозволяє надавати об'єкту характеристик, які відрізняють його від усіх інших об'єктів, чітко визначаючи його концептуальні кордони.

Основним завданням об’єктно-орієнтованого підходу є забезпечення способу структурування програми та керування складними взаємозв’язками між великою кількістю компонентів системи.

Метою курсової роботи є розробка програмного комплексу з використанням об'єктно-орієнтованого підходу і користувальницьких класів для оптимізації процесу обліку автозапчастин. У даній роботі була виконана розробка програми на мові програмування С++ з використанням описаних вище властивостей об’єктно-орієнтованого програмування. Середовище розробки утилітаQT Creator.

# ТЕХНІЧНЕ ЗАВДАННЯ

## Найменування та область застосування

Найменування програми – «AutoDrive».

Область застосування програми – ведення обліку автозапчастин магазину власником та перегляд наявності товарів користувачем.

## Підстави для розробки

Підставами для проведення розробки є індивідуальне завдання на курсову роботу з дисципліни «Об’єктно-орієнтоване програмування».

Найменування теми курсової роботи – «Розробка системи обліку автозапчастин для магазину «AutoDrive»».

Замовник – керівник курсової роботи, викладач Слободян Руслан Олесійович.

Виконавець ­– студент групи КН-321 Демчишин Максим Миколайович.

## Призначення розробки

Функціональне призначення розробки програмного продукту – контроль та облік автомобільних запчастин які є в наявності, додавання та зміна даних про кількість певного товару продавцем або власником, зручність перегляду потрібних товарів покупцем.

Підвищення ефективності роботи продавця чи іншої керуючої особи, яка веде облік товарів магазину «AutoDrive».

## Вимоги до програми чи програмного виробу

### Вимоги до функціональних характеристик

Функціональність  – це сукупність властивостей, які визначають спроможність програмного забезпечення (ПЗ) виконувати в заданому середовищі упорядковану послідовність дій для задоволення споживчих властивостей, замовлених користувачем, відповідно до вимог обробки і загальносистемних засобів. Основні атрибути функціональності ПЗ наведено у таблиці 1.1.

Таблиця 1.1 – Атрибути функціональності програмного забезпечення

|  |  |
| --- | --- |
| Назва | Характеристика |
| Функціональна повнота | Показує ступінь достатності основних функцій для вирішення спеціальних завдань відповідно до призначення ПЗ |
| Правильність | Показує, як забезпечується досягнення правильних та погоджених результатів |
| Інтероперабельність або сумісність | Вказує на спроможність ПЗ взаємодіяти з іншими системами і середовищами |
| Захищеність | Вказує на можливість запобігати несанкціонованому доступу до програм і даних |
| Узгодженість | Вказує на відповідність заданим стандартам, угодам, правилам, законам і розпорядженням |

Програма системи обліку автозапчастин магазину «AutoDrive», яка створена щоб оптимізувати і спростити облік та пошук потрібних наявних товарів, повинна забезпечити наступні функції:

* створення списку товарів по назві;
* створення списку товарів по артикулу;
* додавання нового товару до списку;
* кількість товару;
* видалення товару зі списку;
* зміна характеристик товару;
* авторизація та реєстрація користувача;
* пошук у списку товару за назвою або артикулом;
* збереження списку товарів у файл та отримання списку товарів із файлу;
* зберігання даних в БД.

Для введення даних про кожний із товарів на складі в магазині використаємо структуру, яка буде містити наступні елементи:

Рисунок 1.1 – Схема структури введення даних про товар

### Вимоги до часових характеристик

Програма повинна виконувати всі запити менше ніж за 1.5 секунди.

### Вимоги до надійності

Надійність – це множина атрибутів, які вказують на спроможність програмного забезпечення коректно перетворювати вхідні дані на результати. Зниження надійності ПЗ відбувається внаслідок помилок у вимогах, проектуванні і виконанні. У таблиці 1.2 наведено основні атрибути надійності програми.

Таблиця 1.2 – Атрибути надійності програмного забезпечення

|  |  |
| --- | --- |
| Назва | Характеристика |
| Безвідмовність | Визначає частоту відмов внаслідок наявності помилок у програмному забезпеченні |
| Стійкість до помилок | Вказує на забезпечення спроможності виконувати функції в аномальних умовах (збої апаратури, помилки в даних та інтерфейсах, порушення в діях оператора тощо) |
| Відновлюваність | Вказує на спроможність програми до перезапуску для повторного виконання й відновлення даних після відмов |
| Узгодженість | Показує відповідність діючим стандартам, угодам, правилам, законам і розпорядженням |

Надійне функціонування програми повинне виконувати необхідні функції за певних умов і забезпечити перевірку правильності вхідних даних та недопустимість помилок.

У разі поганої компетентності користувача, для зручності застосування і достовірності, як основного критерію захищеності, буде виведено помилки які він допустив при роботі з програмою. Це забезпечить високу надійність та зручне користування з нею.

### Умови до експлуатації

Для експлуатації розробленого програмного продукту потрібен ПК та базові навички користувача для роботи з ним, також необхідне з’єднання з БД при роботі з програмою.

### Вимоги до складу і параметрів технічних засобів

Для роботи з програмою потрібні такі технічні засоби: ПК, монітор, клавіатура, мишка.

Мінімальні технічні характеристики ПК:

* процесор Intel Pentium Dual Core;
* ОЗУ 1 ГБ;
* Обсяг вільної дискової пам’яті – 2 ГБ;
* Windows XP.

### Вимоги до інформаційної і програмної сумісності

Коди розробленого програмного продукту написані на мові C++. В якості середовища розробки потрібне використання QTCreator не нижче версії 5.12.2. Системні програмні засоби і утиліти, які буде використовувати програма, повинні забезпечуватись встановленою операційною системою сімейства Windows версії не нижче Windows XP. Вимоги до захисту інформації і програми не пред’являються.

## Вимоги до програмної документації

Після закінчення розробки програмного забезпечення потрібно підготувати наступну документацію:

– інструкція інсталяції та запуску програми;

– відомості про можливості програми;

* інструкція з експлуатації програми.

## Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться. Орієнтовне число використань розробленої програми протягом року складає 200–250 раз.

## Стадії та етапи розробки

Розробка програми системи обліку автозапчастин магазину «AutoDrive» повинна мати такі стадії та етапи:

* [Аналіз вимог](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3)

На цій стадії аналізуються вимоги до розробки програмного засобу. Аналіз включає в себе вив­чення пот­реб і цілей замовника, кла­сифікація та пе­рет­во­рен­ня його ви­мог до сис­те­ми, апа­рату­ри і ПЗ.

* [Проєктування програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)

Ця стадія включає в себе декомпозицію програми на функціона­льні компоненти для незалежного і одночасного їхнього виконання, розподіл компонентів у середовищі функціонування і їх взаємодія між собою, забезпечення яко­сті і живучості системи тощо.

* [Програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)

Стадія програмування включає в собі написання коду програми згідно поставлених вимог.

* [Тестування програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F)

При тестуванні програмного забезпечення виявляють помилки в роботі програми шляхом спостереження за його роботою в штучно створених ситуаціях.

## Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

− діючу програму, яка повністю відповідає даному технічному завданню;

­− вихідний програмний код, записаний разом із програмою на оптичний носій інформації.

Прийом програмного забезпечення повинен відбуватися перед комісією з двох чоловік (один з яких – Замовник) у такій послідовності:

− доповідь Виконавця про виконану роботу;

−­­­­­­­ демонстрація Виконавцем роботи програми;

− контрольні випробовування роботи програми;

− відповіді на запитання і зауваження комісії.

# РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

## Розробка загальної структури і варіантів використання програми

Під час проходження етапу аналізу спочатку було визначено наступні питання:

– Хто буде діючими суб'єктами?

– Хто вводитиме інформацію?

– Хто запрошуватиме?

– Чи буде хто-небудь ще взаємодіяти з програмою?

– Чи буде сама програма взаємодіяти з іншими програмами?

З програмою може взаємодіяти 2 особи:

* продавець;
* покупець.

Наступне, що потрібно було з'ясувати, це групу дій, які ініціюватимуть діючі суб'єкт. Виходячи із технічного завдання, діючі суб'єкти – продавець та покупець.

Продавець буде виконувати такі дії:

– запустити програму;

* виконати авторизацію;
* вивести таблицю товарів;
* додати новий товар;
* редагувати товар;
* видалити товар.

Покупець буде виконувати такі дії:

– запустити програму;

– виконати авторизацію;

– вивести таблицю товарів;

– переглянути доступні товари.

Таким чином, діаграму варіантів використання, яка отримується внаслідок наведеного переліку дій, можна подати так, як показано на рисунку 2.1.

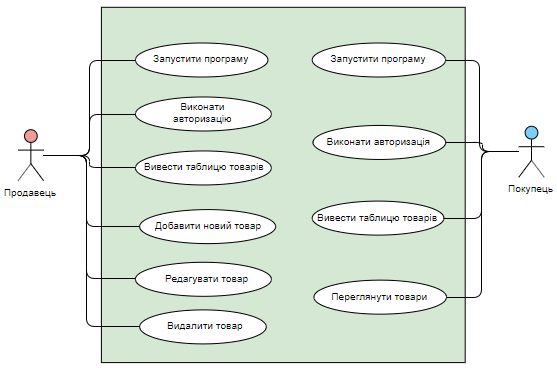


Рисунок 2.1 – Діаграма варіантів використання програми

Наступним кроком аналізу програми є опис варіантів використання, де необхідно описати всі варіанти використання.

1. Запустити програму – після запуску програми користувач зможе авторизуватись за зареєструватись.

2. Виконати авторизацію – якщо користувач має обліковий запис, ввівши логін та пароль, може увійти, якщо ж ні, то може зареєструватись.

3. Вивести таблицю товарів – виводиться вікно таблиці з назвами товарів.

4. Додати новий товар – користувач натискає кнопку «Додати товар», після чого вводить характеристики товару (назва, кількість тощо).

5. Редагувати товар – подвійним натисканням ЛКМ продавець може змінити характеристики товару.

6. Видалити товар – користувач виділяє поле з товаром який бажає видалити та натискає кнопку «Видалити товар», після чого поле з товаром видаляється.

Також на етапі аналізу для моделювання варіантів використання доцільно розробити UML-діаграми дій. Цей тип діаграм демонструє керівні потоки від одних дій до інших (рисунок 2.2).



Рисунок 2.2 – Діаграма дій«Авторизація»

## Розробка системи класів

У програмі «AutoDrive» є 6 основних класів які показані на рисунку 2.3.

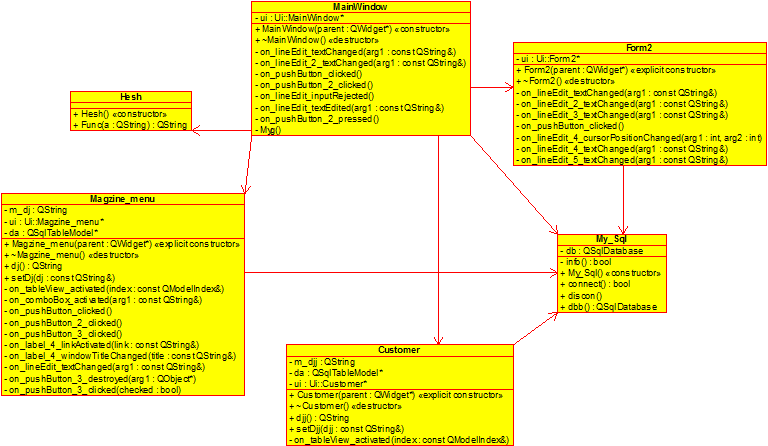


Рисунок 2.3 – UML діаграма класів

Клас «MainWindow» – головне меню програми.

Клас «Hesh» – використаний для хешування паролів.

Клас «MySql» – використаний для роботи з MySql.

Клас «Form2» – використаний для реєстрації.

Клас «Magazine\_menu» – використаний для виведення таблиці з можливістю вносити зміни.

Клас «Customer» – використаний для виведення таблиці з товарами.

## Розробка методів

void MainWindow::on\_pushButton\_clicked() – виконує авторизацію;

void MainWindow::on\_pushButton\_2\_clicked() – посилає на форму реєстрації;

void Form2::on\_pushButton\_clicked() – виконує реєстрацію;

void Magzine\_menu::setDj(const QString &dj) – приймає значення Myg() і передає його в Lable;

void Magzine\_menu::on\_pushButton\_clicked() – додає нове поле;

void Magzine\_menu::on\_pushButton\_2\_clicked() – обновляє таблицю;

void Magzine\_menu::on\_pushButton\_3\_clicked() – видаляє запис, який було виділено;

Клас Hesh містить такий метод:

* Func повертає хеш.

КласMy\_Sql містить такі методи:

– info відкриває з’єднання з БД;

– connect перевіряє з’єдання з БД;

* discon закриває з’єднання з БД.

Клас MainWindow містить такий метод:

* Myg передає дані в форму обліку з форми авторизації.

## Проєктування і опис інтерфейсу користувача

У головному вікні програми можна бачити 2 поля для вводу та 2 кнопки: «Авторизація» та «Зареєструватись» (рисунок 2.4).

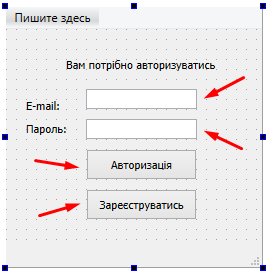


Рисунок 2.4 – Форма авторизації

Вікно реєстрації містить 5 полів для введення інформації про користувача та кнопку «Зареєструватись» (рисунок 2.5).

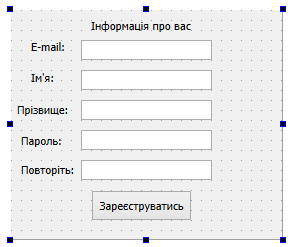


Рисунок 2.5 – Форма реєстрації

Після успішної авторизації «як покупець» відкривається вікно з таблицею товарів (рисунок 2.6)

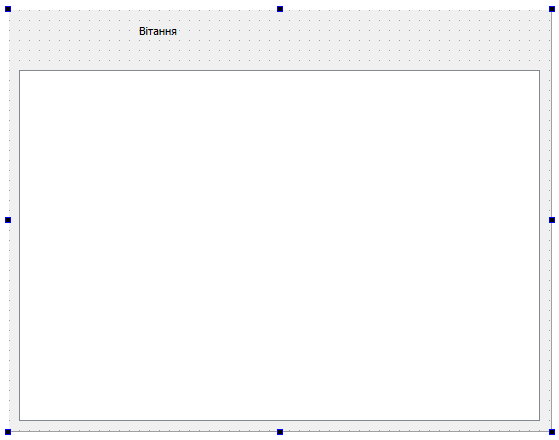


Рисунок 2.6 – Форма з таблицею товарів

Якщо ж виконати вхід від імені адміністратора відкривається вікно з таблицею і можливістю вносити зміни в таблицю за допомогою наступних кнопок: «Добавити товар», «Видалити вибране поле» та «Обновити» (рисунок 2.7).

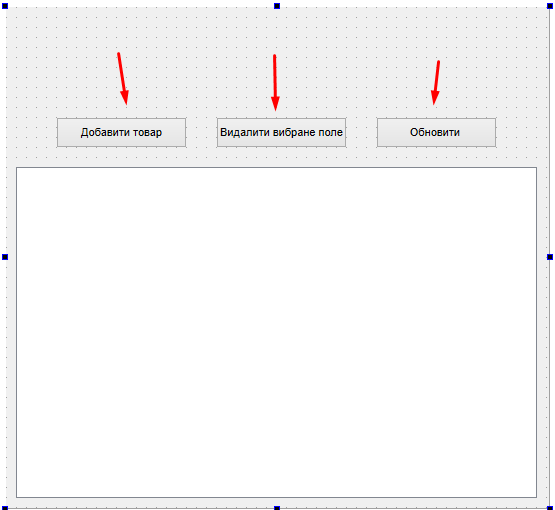


Рисунок 2.7 – Форма з таблицею товарів

## Опис файлової структури програми

Файлова система:

– magazine\_menu.h – заголовковий файл форми з таблицею;

* magazine\_menu.cpp – реалізація форми з таблицею;

– Kursova.pro – QT файл;

– Hesh.h – заголовковий файл класу;

– Hesh.cpp – файл класу хешування;

– MySql.h – заголовковий файл класу;

– MySql.cpp – файл реалізації підключення до БД;

– mainwindow.h – заголовковий файл головної форми;

– mainwindow.cpp – файл реалізацї головної форми;

– main.cpp – головний файл з точкою входу;

– form2.h – заголовковий файл форми;

– form2.cpp – файл реалізації форми;

– customer.h – заголовковий файл форми з таблицею;

– customer.cpp – файл реалізації форми з таблицею.

## Опис структури бази даних програми

В програмі використовується база даних «MySql». Щоб забезпечити стабільну роботу програми потрібні 3 таблиці:

People – таблиця що зберігає дані користувачів (таблиця 2.1).

Таблиця 2.1 – Структура таблиці “People”

|  |  |  |
| --- | --- | --- |
| Name | Lastname | Email |
| Varchar(70) | Varchar(70) | Varchar(70) |

Products – таблиця, що містить дані товарів (таблиця 2.2).

Таблиця 2.2 –Структура таблиці “Product”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Артикул | Авто | Товар | В наявності | Ціна |
| INT | Varchar(70) | Varchar(70) | Varchar(70) | INT |

User – таблиця, що містить дані користувачів для авторизації (таблиця 2.3).

Таблиця 2.3 – структура таблиці “User”

|  |  |  |
| --- | --- | --- |
| ID | Email | Password |
| INT | Varchar(70) | Varchar(70) |

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

Запустивши програму на екрані відображається вікно авторизації (рисунок 3.1). За його допомогою користувач, ввівши дані, може авторизуватись.

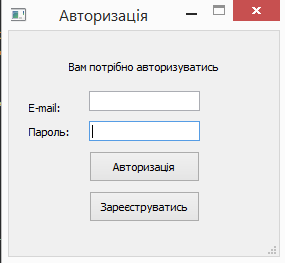


Рисунок 3.1 – Вікно авторизації

При відсутності облікового запису користувач має змогу створити його натиснувши кнопку «Зареєструватись» і ввести дані (рисунок 3.2)

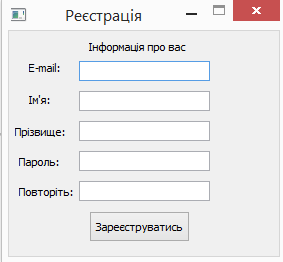


Рисунок 3.2 – Вікно реєстрації

Після авторизації як адміністратор відкриється вікно за таблицею (рисунок 3.3), де можливо додати товар, видалити його та обновити дані таблиці.

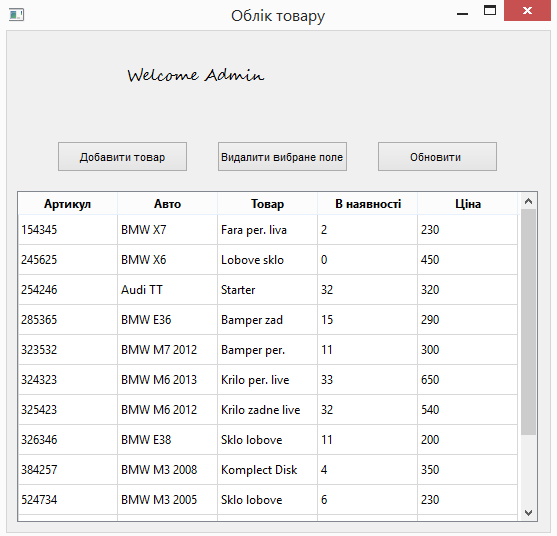


Рисунок 3.3 – Вікно з таблицею товарів

Або ж, авторизовуючись як покупець відкривається вікно з таблицею без можливості вносити зміни (рисунок 3.4).

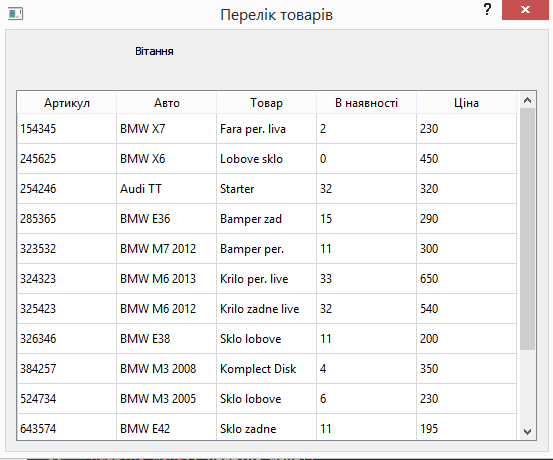


Рисунок 3.4 – Вікно з таблицею товарів

**Програма була також протестована на помилки:**

При неправильному підключенні до БД виводиться помилка на екран (рисунок 3.5).

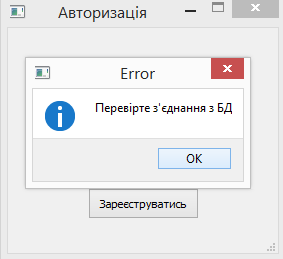


Рисунок 3.5 – Помилка з’єднання з БД

При некоректному введені даних буде виведена помилка з повідомленням яке нагадає користувачу про некоректні дані (рисунок 3.6).

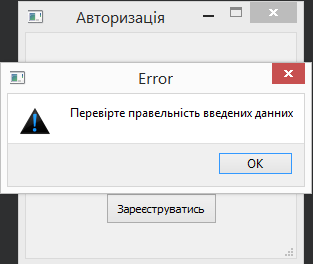


Рисунок 3.6 – Помилка правильності даних

Коли користувач не заповнює всі поля на екрані буде виведена помилка (рисунок 3.7).

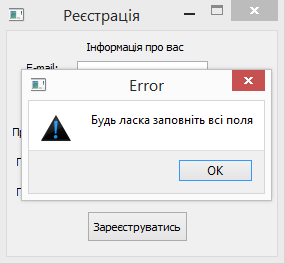


Рисунок 3.7 – Помилка «Заповніть всі поля»

Якщо при реєстрації користувач вкаже вже зареєстрований E-mail буде виведено помилку на екран (рисунок 3.8).

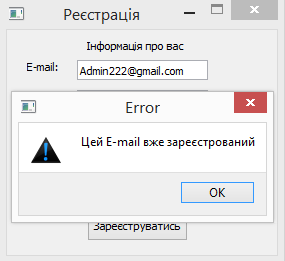


Рисунок 3.8 – Помилка «Цей E-mail вже зареєстрований»

ВИСНОВКИ

В даній курсовій роботі було описано етапи розробки програми «AutoDrive», яка призначена для обліку автозапчастин продавцем, чи іншою керуючою особою і для перегляду товарів покупцем. Розроблений програмний продукт відповідає усім поставленим вимогам технічного завдання.

Для розробки даного програмного забезпечення використовувались принципи і концепції об’єктно-орієнтованого програмування.

Під час виконання завдання курсової роботи мною було закріплено і поглиблено знання здобуті на заняттях з дисципліни «Об’єктно-орієнтоване програмування», більш детально вивчено базові конструкції мови програмування С++ з використанням середовища QT Creator.

Виконана курсова робота надала мені можливість розвинути вміння використовувати основні принципи об’єктно-орієнтованого програмування, отримати практичний досвід працювати з об'єктами, навчитися використовувати класи QTCreator та застосовувати ієрархію класів.

Навички, отримані мною під час виконання курсової роботи, необхідні для подальшого удосконалення професійних здібностей програміста, збільшення рівня кваліфікації. Адже практичне застосування теоретичних знань є однією з найбільш важливих частин підготовки майбутнього фахівця.

ПЕРЕЛІК ПОСИЛАНЬ

1. Грицюк Ю. І. Об’єктно-орієнтоване програмування мовою С++ : навч. посіб. / Ю. І. Грицюк, Т. Є. Рак . – Львів : вид-во Львівського ДУ БЖД, 2011. – 404 с.
2. Вайсфельд М. Объектно-ориентированное мышление / Мэтт Вайсфельд. – СПб. : Питер, 2014. – 304 с.
3. Программирование на языке С++ в среде Qt Creator / Алексеев Е. Р., Злобин Г. Г., Костюк Д. А. и др. – Москва : ALT Linux, 2015. – 448 с.
4. BuildwithQt. URL: https://www1.qt.io/built-with-qt (дата звернення: 15.12.2019).
5. EVILEG : вебсайт.URL: https://evileg.com/uk/knowledge/qt (дата звернення: 15.12.2019).

Додаток А

Лістинг файлу «customer.h»

#ifndef CUSTOMER\_H

#define CUSTOMER\_H

#include"mainwindow.h"

#include <QDialog>

*namespace***Ui** {

*class***Customer**;

}

*class***Customer** : *public* QDialog

{

Q\_OBJECT

*public*:

*explicit***Customer**(QWidget \*parent = *nullptr*);

~***Customer***(); QString **djj**() *const* { *return* m\_djj; }

void **setDjj**(*const* QString& djj);

*private* slots:

void **on\_tableView\_activated**(*const* QModelIndex &index);

*private*:

QString m\_djj;

QSqlTableModel \*da;

Ui::Customer \*ui;

};

#endif *//CUSTOMER\_H*

Додаток Б

**Лістинг файлу «Customer.cpp»**

#include "customer.h"

#include "ui\_customer.h"

#include"mainwindow.h"

#include"mysql.h"

Customer::**Customer**(QWidget \*parent) :

QDialog(parent),

ui(*new* Ui::Customer)

{

ui->setupUi(*this*);

ui->setupUi(*this*);

setWindowTitle("Облік продуктів");

My\_Sql a;

a.connect();

da = *new* QSqlTableModel(*this*,a.dbb());

da->*setTable*("products");

da->*select*();

da->*setEditStrategy*(QSqlTableModel::*OnFieldChange*);

ui->tableView->*setModel*(da);

ui->tableView->horizontalHeader()->*setVisible*(*false*);

ui->tableView->verticalHeader()->*setVisible*(*false*);

ui->tableView->setEditTriggers(QAbstractItemView::*NoEditTriggers*);

}

Customer::~***Customer***()

{

*delete* ui;

}

void Customer::**setDjj**(*const* QString& djj){

m\_djj=djj;

ui->label\_4->setText(m\_djj);

}

void Customer::**on\_tableView\_activated**(*const* QModelIndex &index)

{

}

Додаток В

**Лістинг файлу «form2.h»**

#ifndef FORM2\_H

#define FORM2\_H

#include <QWidget>

#include"mainwindow.h"

*namespace***Ui** {

*class***Form2**;

}

*class***Form2** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit***Form2**(QWidget \*parent = *nullptr*);

~***Form2***();

*private* slots:

void **on\_lineEdit\_textChanged**(*const* QString &arg1);

void **on\_lineEdit\_2\_textChanged**(*const* QString &arg1);

void **on\_lineEdit\_3\_textChanged**(*const* QString &arg1);

void **on\_pushButton\_clicked**();

void **on\_lineEdit\_4\_cursorPositionChanged**(int arg1, int arg2);

void **on\_lineEdit\_4\_textChanged**(*const* QString &arg1);

void **on\_lineEdit\_5\_textChanged**(*const* QString &arg1);

*private*:

Ui::Form2 \*ui;

};

#endif *//FORM2\_H*

Додаток Г

**Лістинг файлу «form2.cpp»**

#include "form2.h"

#include "ui\_form2.h"

#include"mainwindow.h"

#include"mysql.h"

#include"hesh.h"

Form2::**Form2**(QWidget \*parent) :

QWidget(parent),

ui(*new* Ui::Form2)

{

ui->setupUi(*this*);

window()->setFixedSize(270,225);

*this*->setWindowTitle("Реєстрація");

}

Form2::~***Form2***()

{

*delete* ui;

}

void Form2::**on\_lineEdit\_textChanged**(*const* QString &arg1)

{

}

void Form2::**on\_lineEdit\_2\_textChanged**(*const* QString &arg1)

{

}

void Form2::**on\_lineEdit\_3\_textChanged**(*const* QString &arg1)

{

}

void Form2::**on\_pushButton\_clicked**()

{

My\_Sql \*sql\_1=*new* My\_Sql;

Hesh heshing;

QString inp\_log,name,lname;

QString inp\_pass;

*if*(ui->lineEdit\_2!=*nullptr*&&ui->lineEdit->text()!=*nullptr*&&ui->lineEdit\_3!=*nullptr*){

*if*(ui->lineEdit\_2->text()==ui->lineEdit\_3->text()&&(ui->lineEdit->text().endsWith("gmail.com")

||ui->lineEdit->text().endsWith("mail.ru"))&& ui->lineEdit\_4!=*nullptr*&&ui->lineEdit\_5!=*nullptr* ){

inp\_log=ui->lineEdit->text();

*if*(sql\_1->connect()){

QSqlQuery zap,zap\_1;

zap.prepare("SELECT COUNT(\*) FROM user WHERE email=:email");

zap.bindValue(":email",inp\_log);

zap.exec();

int a=0,b=0;

*if*(zap.next()){

a=zap.value(0).toInt();

}

*if*(a==0){

zap.clear();

zap.exec("SELECT MAX(id) FROM user");

*if*(zap.next()){

b=zap.value(0).toInt();

b++;

}

name=ui->lineEdit\_4->text();

lname=ui->lineEdit\_5->text();

zap.clear();inp\_pass=heshing.Func( ui->lineEdit\_2->text());

zap.prepare("INSERT INTO user (id,email,password) VALUES(:id,:email,:password)");

zap.bindValue(":id",b);

zap.bindValue(":email",inp\_log);

zap.bindValue(":password",inp\_pass);

zap\_1.prepare("INSERT INTO people(Name,LastName,email) VALUES(:Name,:lName,:email);");

zap\_1.bindValue(":Name",name);

zap\_1.bindValue(":lName",lname);

zap\_1.bindValue(":email",inp\_log);

*if*(zap.exec()&&zap\_1.exec()){

zap.exec();

*if*(QMessageBox::information(*this*,"Succesful","Тепер ви можете авторизуватись")){

}

}

*else*{

QMessageBox::warning(*this*,"Error","Error");

}

}

*else*{

QMessageBox::warning(*this*,"Error","Цей E-mail вже зареєстрований");

}

}

*else*{

QMessageBox::warning(*this*,"Error","Перевірте підключення до БД");

}

}*else*{

QMessageBox::warning(*this*,"Error","Перевірте правельність введених данних");

}

}

*else*{

QMessageBox::warning(*this*,"Error","Будь ласка заповніть всі поля");

}

sql\_1->discon();

*delete* sql\_1;

}

void Form2::**on\_lineEdit\_4\_cursorPositionChanged**(int arg1, int arg2)

{

}

void Form2::**on\_lineEdit\_4\_textChanged**(*const* QString &arg1)

{

}

void Form2::**on\_lineEdit\_5\_textChanged**(*const* QString &arg1)

{

}

Додаток Ґ

**Лістинг файлу «hash.h»**

#ifndef HESH\_H

#define HESH\_H

#include"mainwindow.h"

*class* Hesh

{

*public*:

Hesh();

QString Func (QString a);

};

#endif *//HESH\_H*

Додаток Д

**Лістинг файлу «hash.cpp»**

#include "hesh.h"

Hesh::**Hesh**()

{

}

QString Hesh::**Func** (QString a){

QByteArray ba = a.toUtf8();

ba=QCryptographicHash::hash(ba,QCryptographicHash::*Sha256*).toHex();

*return* ba; }

Додаток E

**Лістинг файлу «magazine\_menu.h»**

#ifndef MAGZINE\_MENU\_H

#define MAGZINE\_MENU\_H

#include"mainwindow.h"

#include <QWidget>

*namespace* Ui {

*class* Magzine\_menu;

}

*class* Magzine\_menu : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* Magzine\_menu(QWidget \*parent = *nullptr*);

~Magzine\_menu();

QString dj() *const* { *return* m\_dj; }

void setDj(*const* QString& dj);

*privateslots*:

void on\_tableView\_activated(*const* QModelIndex &index);

void on\_comboBox\_activated(*const* QString &arg1);

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_label\_4\_linkActivated(*const* QString &link);

void on\_label\_4\_windowTitleChanged(*const* QString &title);

*private*:QString m\_dj;

Ui::Magzine\_menu \*ui;

QSqlTableModel \*da;

};

#endif *//MAGZINE\_MENU\_H*

Додаток Є

**Лістинг файлу «magazine\_menu.cpp»**

#include "magzine\_menu.h"

#include "ui\_magzine\_menu.h"

#include"mainwindow.h"

#include"mysql.h"

Magzine\_menu::Magzine\_menu(QWidget \*parent) :

QWidget(parent),

ui(*new* Ui::Magzine\_menu)

{

ui->setupUi(*this*);

setWindowTitle("Облік товару");

My\_Sql a;

a.connect();

da = *new* QSqlTableModel(*this*,a.dbb());

da->setTable("products");

da->select();

da->setEditStrategy(QSqlTableModel::OnFieldChange);

ui->tableView->setModel(da);

ui->tableView->horizontalHeader()->setVisible(*false*);

ui->tableView->verticalHeader()->setVisible(*false*);

}

void Magzine\_menu::setDj(*const* QString &dj){

m\_dj = dj;

ui->label\_4->setText(dj);

}

Magzine\_menu::~Magzine\_menu()

{

*delete* ui;

}

void Magzine\_menu::on\_tableView\_activated(*const* QModelIndex &index)

{

}

void Magzine\_menu::on\_comboBox\_activated(*const* QString &arg1)

{

}

void Magzine\_menu::on\_pushButton\_clicked()

{

da->insertRow(da->rowCount());

}

void Magzine\_menu::on\_pushButton\_2\_clicked()

{

da->select();

}

void Magzine\_menu::on\_pushButton\_3\_clicked()

{

int SelectRows=ui->tableView->currentIndex().row();

*if*(SelectRows>=0){

da->removeRow(SelectRows);

}

*else*{

QMessageBox::warning(*this*,"Error","Ні одне поле не було виділено");

}

}

void Magzine\_menu::on\_label\_4\_linkActivated(*const* QString &link)

{

}

void Magzine\_menu::on\_label\_4\_windowTitleChanged(*const* QString &title)

{

}

Додаток Ж

**Лістинг файлу «mainwindow.h»**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<QtSql>

#include<QSqlDatabase>

#include<QMessageBox>

#include<QTextEdit>

#include<QLineEdit>

#include<QCryptographicHash>

#include<QByteArray>

#include <QAbstractButton>

#include<qcryptographichash.h>

#include<QHash>

QT\_BEGIN\_NAMESPACE

*namespace* Ui { *class* MainWindow; }

QT\_END\_NAMESPACE

*class* MainWindow : *public* QMainWindow

{

Q\_OBJECT

*public*:

MainWindow(QWidget \*parent = *nullptr* );

~MainWindow();

*privateslots*:

void on\_lineEdit\_textChanged(*const* QString &arg1);

void on\_lineEdit\_2\_textChanged(*const* QString &arg1);

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_lineEdit\_inputRejected();

void on\_lineEdit\_textEdited(*const* QString &arg1);

void on\_pushButton\_2\_pressed();

void Myg();

*private*:

Ui::MainWindow \*ui;

};

#endif *//MAINWINDOW\_H*

Додаток З

**Лістинг файлу «mainwindow.cpp»**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include"form2.h"

#include<conio.h>

#include"mysql.h"

#include"hesh.h"

#include"customer.h"

#include"magzine\_menu.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

setWindowTitle("Aвторизація");

QObject::connect(ui->pushButton,*SIGNAL*(),*this*,*SLOT*(Myg()));

window()->setFixedSize(270,225);

}

*static* My\_Sql sql ;

MainWindow::~MainWindow()

{

*delete* ui;

}

void MainWindow::on\_lineEdit\_textChanged(*const* QString &arg1)

{

ui->lineEdit->setEchoMode(QLineEdit::Password);

}

void MainWindow::on\_lineEdit\_2\_textChanged(*const* QString &arg1)

{

}

void MainWindow::Myg(){

*//Admin111@mail.ru:123*

QString zap=ui->lineEdit\_2->text(),zpp;

QString t;

*if*(zap=="Admin222@gmail.com"){

Magzine\_menu \*fmr = *new* Magzine\_menu(0);

t = "Welcome Admin";

fmr->setDj(t);

fmr->show();

}

*else*{

QString zap=ui->lineEdit\_2->text(),zpp;

sql.connect();

QSqlQuery z;

z.prepare("SELECT Name,LastName FROM test.people WHERE email=:email;");

z.bindValue(":email",zap);

zpp=z.exec();

*while*(z.next())

{

zpp = z.value(0).toString();

}

sql.discon();

QString t;

Customer \*fmr1 = *new* Customer ;

t ="Вітання"+zpp;

fmr1->setDjj(t);

fmr1->show();

}

}

void MainWindow::on\_pushButton\_clicked()

{

Hesh heshing;

QString inp\_pass;

QString zapr;

QString input\_log;

QSqlQuery zap;

*if*(sql.connect()){

*if*(ui->lineEdit->text()!=*nullptr*&&ui->lineEdit\_2->text()!=*nullptr*

&&( ui->lineEdit\_2->text().endsWith("@gmail.com")

|| ui->lineEdit\_2->text().endsWith("@mail.ru"))){

input\_log=ui->lineEdit\_2->text();

inp\_pass=heshing.Func(ui->lineEdit->text());

zap.prepare("SELECT COUNT(\*) FROM user WHERE password=:pass and email=:email");

zap.bindValue(":pass",inp\_pass);

zap.bindValue(":email",input\_log);

zap.exec();

int a=0;

*if*(zap.next()){

a=zap.value(0).toInt();

}

*if*(a!=0){

Myg(); sql.discon();

*this*->hide();

}

*else*{

QMessageBox::information(*this*,"Error","Пароль або логін невірні ");

}

}

*else*{

QMessageBox::warning(*this*,"Error","Перевірте правельність введених данних");

}

}

*else*{

QMessageBox::information(*this*,"Error","Перевірте з'єднання з БД");

} sql.discon();

}

void MainWindow::on\_pushButton\_2\_clicked()

{

Form2 \*f = *new* Form2;

f->show();

*this*->hide();

}

void MainWindow::on\_lineEdit\_inputRejected()

{

}

void MainWindow::on\_lineEdit\_textEdited(*const* QString &arg1)

{

}

void MainWindow::on\_pushButton\_2\_pressed(){

}

Додаток И

**Лістинг файлу «mysql.h»**

#ifndef MYSQL\_H

#define MYSQL\_H

#include"mainwindow.h"

*class* My\_Sql*//MainWindow*

{

QSqlDatabase db=QSqlDatabase::addDatabase("QMYSQL3");

bool info();

*public*:

My\_Sql();

bool connect();

void discon();

QSqlDatabase dbb();

};

#endif *//MYSQL\_H*

Додаток І

**Лістинг файлу «mysql.cpp»**

#include "mysql.h"

My\_Sql::My\_Sql()

{

}

QSqlDatabase My\_Sql::dbb(){

*return* db;

}

bool My\_Sql::info(){

db.setHostName("127.0.0.1");

db.setUserName("root");

db.setPort(3306);

db.setDatabaseName("nn");

*if* (db.open()){

*returntrue*;

}

*else* {

*returnfalse*;

}

}

void My\_Sql::discon(){

db.close();

}

bool My\_Sql::connect(){

*if*( info()){

*returntrue*;

}

*else*{

*returnfalse*;

}

}

Додаток Ї

**Лістинг файлу «main.cpp»**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

*return* a.exec();

}

Додаток Г  
Компакт-диск із програмним продуктом