**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Кафедра програмних засобів

**КУРСОВИЙ ПРОЕКТ**

**(РОБОТА)**

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

(назва дисципліни)

на тему: *«Розпорядок заходів»*

Студентів *2* курсу *КНТ-117* групи

Напряму підготовки *Програмна інженерія*

Спеціальності *Інженерія програмного*  забезпечення

. Нескоромний І.М.

(прізвище та ініціали)

. Кузнєцов Т.В.

(прізвище та ініціали)

. Таряник М.В.

(прізвище та ініціали)

. Івченко А.С.

(прізвище та ініціали)

Керівник *к.т.н., доцент Табунщик Г.В*.

(посада, вченне звання, науковий ступінь, призвіще та ініціали)

Національна шкала  Кількість балів: Оцінка: ECTS

Члени комісії . Табунщик Г.В

(підпис) (прізвище та ініціали)

. .. Каплієнко Т.І.

(підпис) (прізвище та ініціали)

. .. Миронова Н.О.

(підпис) (прізвище та ініціали)

м.Запоріжжя  
2018 рік

Зміст

[Реферат 6](#_Toc533616233)

[вступ 7](#_Toc533616234)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8](#_Toc533616235)

[1 аналіз предметної області 9](#_Toc533616236)

[1.1 Огляд існуючих методів вирішення завдання 9](#_Toc533616237)

[1.1.1 Програма Wunderlist 10](#_Toc533616238)

[1.1.2 Програма EventLook 11](#_Toc533616239)

[1.1.3 Програма Google Now 12](#_Toc533616240)

[1.2 Постановка завдання роботи 13](#_Toc533616241)

[2 Аналіз програмних засобів 15](#_Toc533616242)

[2.1 Огляд особливостей мови програмування 15](#_Toc533616243)

[2.2 Огляд особливостей обраного компілятору 16](#_Toc533616244)

[2.3 Класи візуального інтерфейсу 17](#_Toc533616245)

[2.4 Висновки з розділу 19](#_Toc533616246)

[3 Основні рішення з реалізації компонентів системи 20](#_Toc533616247)

[3.1 Основні розроблені алгоритми 20](#_Toc533616248)

[3.1.1 Створення годинника у реальному часі 20](#_Toc533616249)

[3.1.2 Сортування планів заходу за (id, часом початку заходів, часом кінця заходу, за алфавітним порядком планів) 21](#_Toc533616250)

[3.1.3 Привітання при вдалому підключенні до БД, і музичне повідомлення про помилку при невдалому підключенні 21](#_Toc533616251)

[3.1.4 Зчитування даних з БД, та занесення їх до combobox 21](#_Toc533616252)

[3.1.5 Зчитування даних з БД про плани заходів, та занесення їх до таблиці 21](#_Toc533616253)

[3.1.6 Зчитування даних з БД про заходи, та занесення їх до таблиці 22](#_Toc533616254)

[3.1.7 Підготовка таблиці для внесення змін в неї 22](#_Toc533616255)

[3.2 Ієрархія розроблених класів 23](#_Toc533616256)

[3.3 Основні рішення щодо розробки інтерфейсу користувача 31](#_Toc533616257)

[3.3.1 Загальна інформація 31](#_Toc533616258)

[3.3.2 Вікно «TimeTask - Головне вікно» 32](#_Toc533616259)

[3.3.3 Вікно «TimeTask – Про заходи» 33](#_Toc533616260)

[3.3.4 Вікно «TimeTask – Додати захід» 34](#_Toc533616261)

[3.3.5 Вікно «TimeTask – Редагування заходу» 35](#_Toc533616262)

[3.3.6 Вікно «TimeTask – Додати план» 35](#_Toc533616263)

[3.3.7 Вікно «TimeTask – Редагування плану» 36](#_Toc533616264)

[3.3.8 Вікно «TimeTask – Видалення планів» 37](#_Toc533616265)

[3.3.9 Вікно «TimeTask – Видалення заходів» 37](#_Toc533616266)

[3.3.10 Вікно «TimeTask – Про програму» 38](#_Toc533616267)

[3.4 Основні рішення щодо збереження даних 39](#_Toc533616268)

[3.4.1 Основні рішення щодо працювання з таблицями 39](#_Toc533616269)

[4 Керівництво програміста 42](#_Toc533616270)

[4.1 Призначення та умови застосування програми 42](#_Toc533616271)

[4.2 Характеристики програми 42](#_Toc533616272)

[4.3 Звертання до програми 43](#_Toc533616273)

[4.4 Вхідні та вихідні дані 43](#_Toc533616274)

[4.5 Повідомлення 44](#_Toc533616275)

[5 ІНСТРУКЦІЯ КОРИСТУВАЧА 46](#_Toc533616276)

[5.1 Призначення програми 46](#_Toc533616277)

[5.2 Умови виконання програми 46](#_Toc533616278)

[5.3 Як запустити програму 46](#_Toc533616279)

[5.4 Виконання програми 47](#_Toc533616280)

[5.5 Повідомлення користувачу 51](#_Toc533616281)

[ВИСНОВКИ 53](#_Toc533616282)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 54](#_Toc533616283)

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Кафедра *програмних засобів*

Дисципліна *Об’єктно-орієнтоване програмування*

Спеціальність *Інженерія програмного забезпечення*

Курс *2* Група *КНТ-117* Семестр *III*

**ЗАВДАННЯ**

**на курсовий проект (роботу) студентів**

*Нескоромний І.М., Кузнєцов Т.В., ТаряникМ.В., Івченко А.С..*

1.Тема проекту (роботи): *«Розпорядок заходів »*

2. Термін здачі студентом закінченого проекту (роботи): *26 грудня 2018*

3.Вихідні дані до проекту: *Реалізувати помічника для планування часу*

вхідні дані: назва заходу, назва місця проведення, адреса заходу, кількість людей, дата початку, дата кінця.

4.ЗМІСТ розрахунково-пояснювальної записки (перелік питань, що їх належить розроби-

ти*):* *1 Аналіз предметної області*

*2 Аналіз програмних засобів*

*3 Основні рішення з реалізації компонентів системи*

*4 Посібник програміста*

*5 Інструкція користувача*

*Висновки,*

*Додаток А Текст програми,*

*Додаток Б Інтерфейс програми,*

*Додаток В Презентація*

5.Перелік графічного матеріалу: *Слайди презентації*

6. Дата видачі завдання: *12 вересня 2018*

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № пор. | Назва етапів курсового проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
| 1. | Аналіз індивідуального завдання. | 1-2 тиждень |  |
| 2. | Аналіз програмних засобів,що будуть використовуватися в роботі. | 3-4 тиждень |  |
| 3. | Аналіз структур даних, що необхідно використати в курсовій роботі. | 4-5 тиждень |  |
| 4. | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-6 тиждень |  |
| 5. | Оформлення відповідних пунктів пояснювальної записки. | 4 тиждень | Розділи 1,2 ПЗ |
| 6. | Проміжний контроль. | 8 тиждень |  |
| 7. | Аналіз вимог до апаратних засобів. | 9 тиждень |  |
| 8. | Розробка програмного забезпечення. | 10-15 тиждень |  |
| 9. | Оформлення відповідних пунктів пояснювальної записки. | 9-16 тиждень | Розділи 3,4,5 ПЗ |
| 10. | Захист курсової роботи. | 17 тиждень |  |

Студент. . Кузнєцов Т.В.

(підпис) (прізвище, ім’я, по батькові)

Студент. . Таряник М.В.

(підпис) (прізвище, ім’я, по батькові)

Студент. . Нескоромний І.М.

(підпис) (прізвище, ім’я, по батькові)

Студент. . Івченко А.С.

(підпис) (прізвище, ім’я, по батькові)

Керівник. . Табунщик Г.В

(підпис) (прізвище, ім’я, по батькові)

“\_\_\_\_” . 2018р.

Реферат

ПЗ: 105 стор. 23 рис., 15 табл., 3 додатки, 10 джерел.

Метою даного курсового проекту є розробка програмного забезпечення для створення розкладу заходів, що зберігає інформацію в базі даних.

Було виконано аналіз предметної області, розглянуто аналогічні існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування С++ та середа розробки QT. У якості комплілятора було обрано MinGW.

Були розроблені такі класи форм, даних і таблиць та описано їх методи:

AboutWindow, AddPlans, AddWindow, ChangePlans, ChangeWindow,  
ConnectionDB, DeletePlan, DeleteWindow, JSONDB, MainWindow,Messages, ShowWindow,Sounds,UserTime.

При виконані курсового проекту було розглянуто особливості мови програмування С++ , компілятору, а також зроблено огляд стандартного бібліотеки шаблоних класів. Також приділяється акцент розробці інтерфейсу користувача, яких було розробелений за допомогою бібліотеки візуальних компонентів QT, та файловому вводу-виводу та обробці виняткових ситуацій.

STL, ВІЗУАЛЬНИЙ ІНТЕРФЕЙС, КЛАС, СИСТЕМА ОБЛІКУ

вступ

Актуальність роботи полягає у важливості мистецтва керування своїм часом у сучасному ритмі життя.

Метою даного курсового проекту є розробка програмного забезпечення- помічника для планування часу.

Згідно з темою була поставлена мета: розробити програму за допомогою мови програмування С++ та фреймворка Qt, що реалізує программу «Розпорядок заходів».

* + згідно з метою були поставлені наступні завдання:
  + проаналізувати предметну область;
  + розробити відповідні структури даних;
  + створити графічний інтерфейс;
  + розробити програму;
  + провести тестування;
  + оформити пояснювальну записку.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД - база даних

ООП - Об'єктно-орієнтоване програмування

ОС - Операційна система

ПЗ - Програмне забезпечення

GUI - Graphical User Interface (графічний інтерфейс користувача)

ISO - International Organization for Standardization

JSON - JavaScript Object Notation

QGV - об’єкт класу View (наслідник QGraphicsView)

Qt - (Сute – к'ют) кросплатформна бібліотека C++ класів для створення графічних програм

STL - Standart Template Library

1. аналіз предметної області

В даному розділі розглянуто існуючі методи вирішення завдання та огляд програмних засобів, метою яких є надання користувачам можливості контролювати свій час.

* 1. Огляд існуючих методів вирішення завдання

Програма розкладу заходів дуже корисна для людей, які хочуть контролювати свій вільний та робочий час за допомогою тайм-менеджменту. Одними з найбільш поширених програм є саме розклад заходів. Знати в який день,в якому місці,о котрій годині розпочинається те чи інше шоу програми. За допомогою електронного розкладу,можна забути про паперові буклети, або ж про стенді при вході. Тож для того щоб допомогти користувачам програми “розклад заходів”, необхідно створити програмне забезпечення з корисними функціями та зі зручним інтерфейсом.

Таким чином, програма “розклад заходів” допоможе зберегти багато часу користувачу,який зможе дивитися які заходи проходять у той чи інший день,також протягом якого часу проходе той чи інший захід,а головне сам розклад усіх планів заходу.

Огляд існуючих планувальників:

* + 1. Програма Wunderlist

Компанія-видавець: Wunderlist.

Офіційний сайт програми: https://www.wunderlist.com.

Wunderlist - це додаток, призначений для створення спільних списків справ, завдяки яким ви можете більш точно координувати свої дії з колегами, друзями і родичами.

Переваги даного програмного продукту:

* + сортування задач (за датою створення, датою виконання, за алфавітом, за виконавцем, за рівнем важливості);
  + функція пошуку справ;
  + гарячі клавіші для швидкого доступу до функцій;
  + можна групувати задачі за темами;
  + можливість обрати фон програми;
  + є всі умови для проектної роботи: за допомогою електронної пошти можна долучати людей до створеного списку.

Робота програми продемонстрована на рисунку 1.1.

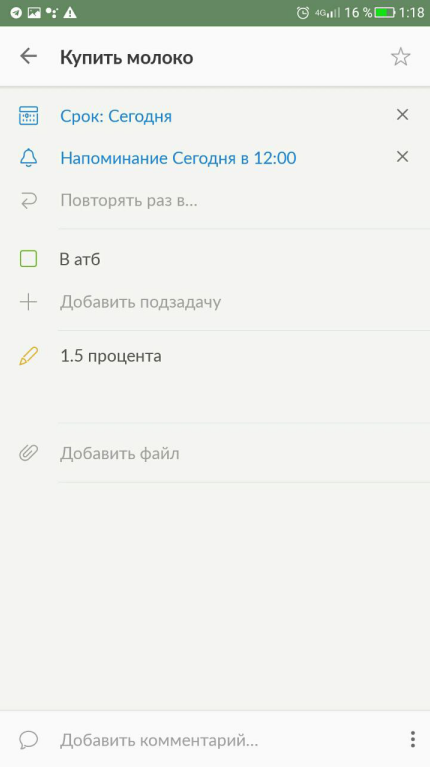


Рис. 1.1 – Приклад роботи програми

* + 1. Програма EventLook

Компанія-видавець: [Mobile Dimension LLC](https://play.google.com/store/apps/developer?id=Mobile+Dimension+LLC).

Офіційний сайт програми: https://eventlookapp.ru.

EventLook - це персональний гід по заходам і конференціям.

Всі дані про захід в вашому телефоні. Додаток дозволяє вам спланувати свій маршрут по необхідним сесіям і не забути про важливі виступи. Ви зможете отримати повну інформацію про місце та дату проведення заходу, розклад, список учасників і доповідачів. За допомогою програми ви також зможете дати вашу оцінку виступам.

Переваги даного програмного продукту:

* + простий і зрозумілий інтерфейс;
  + перегляд даних щодо заходу в режимі оффлайн без підключення до мережі інтернет;
  + дані про місце і час проведення заходу;
  + перегляд загального розкладу заходів і створення свого власного з урахуванням ваших уподобань;
  + можливість голосування та оцінювання виступів;
  + перегляд схематичної карти майданчикиів конференції.

Робота програми продемонстрована на рисунку 1.2.

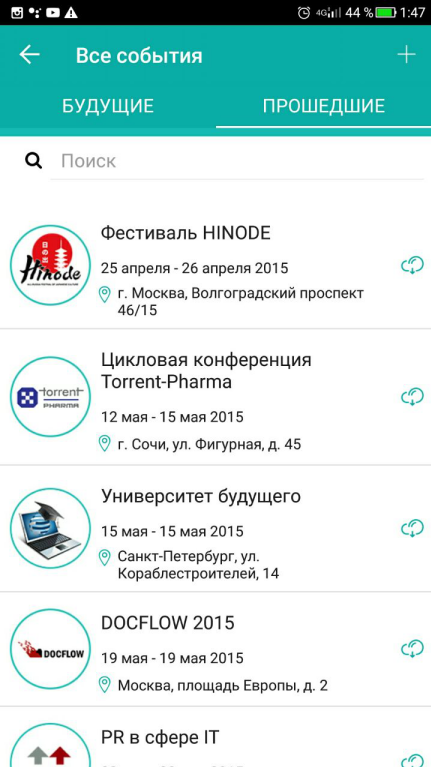


Рис. 1.2 – Приклад роботи програми

* + 1. Програма Google Now

Компанія-видавець: [Google LLC](https://play.google.com/store/apps/dev?id=5700313618786177705).

Офіційний сайт програми: https://www.google.com/intl/ru/landing/now.

Google Now - це інтелектуальний голосовий помічник, який обробляє складні запити і використовує вашу поведінку, щоб передбачати, яка інформація вам знадобиться, ще до того, як ви попросите про це. Google Now також повністю інтегрований в екосистему додатків Google, що дозволяє вам призначати заходи в Google Календарі, вносити замітки і багато іншого.

Переваги даного програмного продукту:

* + Google Now дає вам більше часу, щоб зосередитися на своїй роботі, адже він надає вам потрібну інформацію, не змушуючи вас копатися в пошуковій видачі;
  + Google Now завжди готовий допомогти, хочете ви зарезервувати столик в ресторані або намагаєтеся згадати, де припаркували свій автомобіль;
  + додаток автоматично адаптується до вашої поведінки і забезпечує вам унікальний і повністю настроюється досвід, який можна змінити в будь-який момент.

Робота програми продемонстрована на рисунку 1.3.

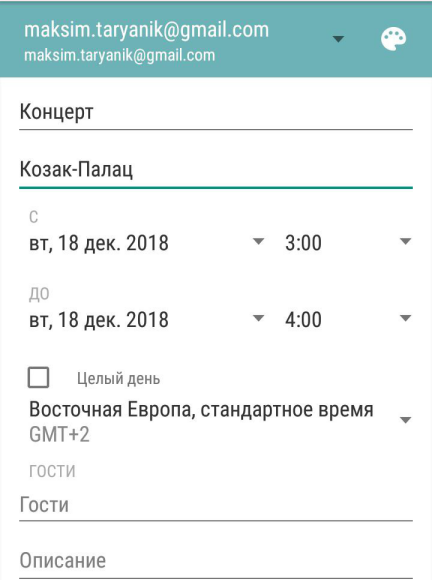


Рис. 1.3 – Приклад роботи програми

* 1. Постановка завдання роботи

Проаналізувавши продукти-аналоги, наведені у пункті 1.2., можна прийти до висновку, що запропоновані на ринку програмні продукти можуть бути застосовані, в основному, для мобільних телефонів. Якщо ж метою буде тестування і розвиток швидкості знаходження цих об'єктів у певному порядку на комп'ютері за допомогою миші – доцільно буде використовувати саме програмний продукт, що написаний для Windows або Linux.

Задача цієї програми: надати користувачу можливість створювати заходи та виконувати над ними різні дії.

Функції, що будуть реалізовані системою:

* + можливість створення заходу та плану до нього;
  + можливість перегляду інформафії про захід та його план;
  + можливість видалення заходу;
  + можливість редагування заходу;
  + можливість видалення плану заходу;
  + можливість редагування плану заходу.

1. Аналіз програмних засобів

В даному розділі розглянуто основні особливості програмних засобів, за допомогою яких реалізовано курсовий проект.

* 1. Огляд особливостей мови програмування

C++ - мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор.

С++ має синтаксис, заснований на синтаксисі С. Нововведеннями С++ порівняно з С є:

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

С++ додає до С об'єктно-орієнтовані можливості. Він вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляцію, успадкування і поліморфізм.

* 1. Огляд особливостей обраного компілятору

Qt Creator — інтегроване середовище розробки, призначене для створення крос-платформових застосунків з використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в якій використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками. Qt Creator може використовувати GCC або Microsoft VC++ як компілятор і GDB як зневаджувач. Для Windows версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW.

Особливості цього середовища:

* + вбудований редактор форм (Qt Designer) і довідкова система (Qt Assistant);
  + контекстно-залежна система допомоги;
  + розширюваність плагінами;
  + є графічний фронтенд для GDB;
  + підтримка зневадження за допомогою CDB;
  + для створення проектів використовується qmake;
  + узагальнене підсвічування синтаксису, підтримується велика кількість мов програмування і розмітки;
  + можливість редагувати етапи складання проекту;
  + підтримка розробки на мовах C/C++, JavaScript, QML.
  1. Класи візуального інтерфейсу

У роботі було використано такі класи Qt, як:

* + QMainWindow;
  + QApplication;
  + QTime;
  + QString;
  + QMessageBox;
  + QTimer;

У наступних кількох абзацах надано короткий опис кожного з цих класів.

Клас QMainWindow надає можливість для створення головного вікна програми. Головне вікно надає структуру для створення призначеного для користувача інтерфейсу додатку. Qt має QMainWindow і пов'язані з ним класи для управління головним вікном. QMainWindow має свій власний макет, до якого ви можете додати QToolBar'и, QDockWidget'и, QMenuBar, і QStatusBar. Компоновщик має центральну область, яка може бути зайнята будь-яким виджетом.

Клас QApplication управляє потоком управління додатком GUI і основними настройками. QApplication спеціалізує на з деякою функціональністю, необхідної для додатків, заснованих на QWidget. Він забезпечує спеціальну ініціалізацію, завершення.Для будь-якої програми з графічним інтерфейсом з використанням Qt, точно наявний один об'єкт QApplication, незалежно від того, чи має додаток 0, 1, 2 або більше вікон, в будь-який момент часу.

Деякі додатки з графічним інтерфейсом забезпечують спеціальний пакетний режим IE. приймаючи аргументи командного рядка для виконання завдань без ручного втручання. У такому режимі без графічного інтерфейсу, часто буває достатньо класу QCoreApplication, щоб уникнути надмірно ініціалізації ресурсів, необхідних для графічного призначеного для користувача інтерфейсу.

Для роботи з часом бібліотека Qt надає клас QTime. Як і у випадку з об'єктами дати, з об'єктами часу можна проводити операції порівняння ==,! =, <, <=,> Або> =. Об'єкти часу здатні зберігати час з точністю до мілісекунд. У конструктор класу QTime передаються чотири параметри. Перший параметр задає годинник, другий - хвилини, третій - секунди, а четвертий - мілісекунди. Третій і четвертий параметри можна опустити, за замовчуванням вони дорівнюють нулю. Клас QString представляє собою рядок символів Unicode.

QString зберігає рядок 16-бітних QChars, де кожен QChar відповідає одному символу Unicode 4.0. (Символи Unicode з кодовими значеннями вище 65535 зберігаються за допомогою сурогатних пар, тобто два послідовних QChars.)

Unicode є міжнародним стандартом який підтримує більшість пишучих систем, що використовуються сьогодні. Це є надбудовою US-ASCII (ANSI X3.4-1986) і Latin-1 (ISO 8859-1), і все символи US-ASCII / Latin-1 доступні в тих же самих кодових позиціях.

Клас QMessageBox надає цілу серію статичних методів, за допомогою яких можна створювати вікна повідомлень. Ці методи надають підтримку для трьох рівнів:

* + інформаційного,
  + застережливого,
  + критичного.

Тип вікна вибирається залежно від обставин. Вікна можуть містити до трьох кнопок.

Це дуже зручно, так як не потрібно писати додаткового коду для реалізації виведення повідомлення. Можна застосовувати такі вікна для налагоджувальних цілей - вивести необхідну інформацію і призупинити виконання програми.

Для ліквідації цих незручностей Qt надає клас таймера QTimer, що є безпосередньою спадкоємницею класу QObject. Щоб запустити таймер, потрібно створити об'єкт класу QTimer, а потім викликати метод start (). У параметрі методу передається значення інтервалу запуску в мілісекундах.

* 1. Висновки з розділу

У другому розділі було описано особливості мови програмування, галузі розробки та обраного компілятору. Вибрана мова програмування має багато переваг, серед яких:

* + є відкритою ISO-стандартизованою мовою;
  + це компільована мова;
  + є сильно типізованою мовою без суворого контролю типів;
  + підтримує як статичну так і динамічну типізацію;
  + є мультипарадигмовою мовою;
  + є портативним;
  + сумісний з С.

Окрім цього у розділі були зазначені та описані класи бібліотеки Qt, що зображують переваги вибраної галузі розробки та були використані у ході розробки програми: QMainWindow, QApplication, QMessageBox, QTimer, QTime, QString. Вибрані класи забезпечують зручну розробку програми , спрощують процес її написання та роблять зручним інтерфейс.

1. **Основні рішення з реалізації компонентів системи**
   1. Основні розроблені алгоритми

У програмі були розроблені наступні алгоритми:

* + Створення годинника у реальному часі;
  + Сортування планів заходу за(id,часом початку заходів, часом кінця заходу, за алфавітним порядком планів);
  + Привітання при вдалому підключенні до БД, і музичне повідомлення про помилку при невдалому підключенні;
  + Зчитування даних з БД, та занесення їх до combobox:
  + Зчитування даних з БД про плани заходів, та занесення їх до таблиці;
  + Зчитування даних з БД про заходи, та занесення їх до таблиці;
  + Підготовка таблиці для внесення змін в неї.
    1. ***Створення годинника у реальному часі***

Алгоритм створення годинника у реальному часі допомагає користувачам спостерігати теперішню дату та час, це допоможе користувачу у плануванні часу на теперішній день.

* + 1. Сортування планів заходу за (id, часом початку заходів, часом кінця заходу, за алфавітним порядком планів)

Для більш швидкого пошуку формації у таблицях використовується інтегрований у середовищі розробки QT функції для сортування TableView. У таблицях ми можемо сортувати за часом додавання планів, за назвою,за датою та ін.

* + 1. Привітання при вдалому підключенні до БД, і музичне повідомлення про помилку при невдалому підключенні

При вдалому підключені до Бази даних відбувається привітання зі словом “Hello”, якщо ж підключення до бази невдале, грає повідомлення про невдале підключення “You are not connected to the database”.

* + 1. Зчитування даних з БД, та занесення їх до combobox

Алгоритм, який заносить дані за Бази даних до combobox.

* + 1. Зчитування даних з БД про плани заходів, та занесення їх до таблиці

Зчитування даних з БД про плани заходів, та занесення їх до таблиці.

* + 1. Зчитування даних з БД про заходи, та занесення їх до таблиці

Зчитування даних з БД про заходи, та занесення їх до таблиці.

* + 1. Підготовка таблиці для внесення змін в неї

Алгоритм, який змінює данні в таблиці та заносить їх до змінної, для відправки даних до бази.

* 1. Ієрархія розроблених класів

В ході роботи над програмою курсового проекту було розроблено класи: AboutWindow, AddPlans, AddWindow, ChangePlans, ChangeWindow,  
ConnectionDB, DeletePlan, DeleteWindow, JSONDB, MainWindow,Messages, ShowWindow,Sounds,UserTime

Клас MainWindow містить інформацію про авторів програми «TimeTask». Дані та методи класу наведені в табл. 3.1

Таблиця 3.1 – Дані та методи класу MainWindow

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit MainWindow(QWidget \*parent = 0) | Конструктор |
| ~MainWindow() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_clicked() | Відкриття вікна «Показати заходи» |
| void on\_pushButton\_2\_clicked() | Відкриття вікна «Додати захід» |
| void on\_pushButton\_3\_clicked() | Відкриття вікна «Видалити захід» |
| void on\_pushButton\_4\_clicked() | Відкриття вікна «Редагувати плани заходів» |
| void on\_action\_7\_triggered() | Розгортає розділ «?» у верхній панелі меню |
| void on\_pushButton\_5\_clicked() | Відкриття вікна «Додати плани заходів» |

|  |  |
| --- | --- |
| void on\_pushButton\_6\_clicked() | Відкриття вікна «Видалити плани заходів» |
| void on\_pushButton\_7\_clicked() | Відкриття вікна «Редагування заходів» |
| void timeManager() | Показує реальний час |
| ***private:*** | |
| Ui::MainWindow \*ui | покажчик на форму "MainWindow" |
| QSqlDatabase db | змінна для підключення до бази даних |

Продовження таблиці 3.1

Клас AboutWindow містить інформацію про авторів програми «TimeTask». Дані та методи класу наведені в табл. 3.2.

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit AboutWindow(QWidget \*parent = 0) | Конструктор |
| ~AboutWindow() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_clicked() | Закриває вікно "О программе" |
| ***private:*** | |
| Ui::AboutWindow \*ui | покажчик на форму "AboutWindow" |

Таблиця 3.2 – Дані та методи класу AboutWindow

Клас AddWindow реалізує додавання нових заходів програми «TimeTask». Дані та методи класу наведені в табл. 3.3.

Таблиця 3.3 – Дані та методи класу AddWindow

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit AddWindow(QWidget \*parent = 0) | Конструктор |
| ~AddWindow() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_2\_clicked() | Закриття вікна |
| void on\_pushButton\_clicked() | Додавання нового заходу |
| ***private:*** | |
| Ui::AddWindow \*ui | покажчик на форму "AddWindow" |
| QSqlDatabase db | змінна для підключення до бази даних |

Клас AddPlans реалізує додавання нових планів до заходу у програмі «TimeTask». Дані та методи класу наведені в табл. 3.4.

Таблиця 3.4 – Дані та методи класу AddPlans

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit AddWindow(QWidget \*parent = 0) | Конструктор |
| ~AddWindow() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_clicked() | Закриття вікна |
| void on\_pushButton\_2\_clicked() | Додавання нового плану заходу |

|  |  |
| --- | --- |
| ***private:*** | |
| Ui::AddPlans \*ui | Покажчик на форму "AddPlans" |
| QSqlDatabase db | Змінна для підключення до бази даних |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |

Продовження таблиці 3.4

Клас ChangeWindow реалізує редагування існуючих заходів у програмі «TimeTask». Дані та методи класу наведені в табл. 3.5.

Таблиця 3.5 – Дані та методи класу ChangeWindow

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit changeWindow(QWidget \*parent = 0) | Конструктор |
| ~changeWindow() | Деструктор |
| ***private slots:*** | |
| void on\_comboBox\_currentTextChanged(const QString &arg1) | Розгортає список з існуючими заходами |
| void on\_pushButton\_clicked() | Закриття вікна |
| void on\_pushButton\_2\_clicked() | Додавання змін до заходу |
| ***private:*** | |
| Ui::changeWindow \*ui | Покажчик на форму "ChangeWindow" |
| QSqlDatabase db | Змінна для підключення до бази даних |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |
| QSqlTableModel \*plans |

Клас ChangePlans реалізує редагування плану існуючого заходу у програмі «TimeTask». Дані та методи класу наведені в табл. 3.6.

Таблиця 3.6 – Дані та методи класу ChangePlans

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit ChangePlans(QWidget \*parent = 0) | Конструктор |
| ~ChangePlans() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_clicked() | Закриття вікна |
| void on\_pushButton\_2\_clicked() | Додавання змін до плану заходу |
| void on\_comboBox\_currentTextChanged(const QString &arg1) | Розгортає список з існуючими заходами |
| void updateComboBox() | Змінені дані в таблиці |
| ***private:*** | |
| Ui::ChangePlans \*ui | Покажчик на форму "ChangePlans" |
| QSqlDatabase db | Змінна для підключення до бази даних |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |
| QSqlTableModel \*plans |

Клас DeleteWindow реалізує видалення існуючого заходу у програмі «TimeTask». Дані та методи класу наведені в табл. 3.7.

Таблиця 3.7 – Дані та методи класу DeleteWindow

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit DeleteWindow(QWidget \*parent = 0) | Конструктор |
| ~DeleteWindow() | Деструктор |

Продовження таблиці 3.7

|  |  |
| --- | --- |
| ***private slots:*** | |
| void on\_pushButton\_2\_clicked() | Видалення вибраного заходу |
| void on\_pushButton\_clicked() | Закриття вікна |
| void updateComboBox() | Змінені дані в таблиці |
| ***private:*** | |
| Ui::DeleteWindow \*ui | Покажчик на форму "DeleteWindow" |
| QSqlDatabase db | Змінна для підключення до бази даних |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |

Клас DeletePlan реалізує видалення плану існуючого заходу у програмі «TimeTask». Дані та методи класу наведені в табл. 3.8.

Таблиця 3.8 – Дані та методи класу DeletePlan

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| Explicit DeletePlan(QWidget \*parent = 0) | Конструктор |
| ~DeletePlan() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_2\_clicked() | Видалення вибраного плану заходу |
| void on\_pushButton\_clicked() | Закриття вікна |
| void updateEvent() | Змінені дані в таблиці |
| void on\_comboBox\_currentTextChanged(const QString &arg1) | Розгортає список з існуючими заходами |
| ***private:*** | |
| Ui::deletePlan \*ui | Покажчик на форму "DeletePlan" |
| QSqlDatabase db | Змінна для підключення до бази даних |

Продовження таблиці 3.8

|  |  |
| --- | --- |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |
| QSqlTableModel \*plans |

Клас connectionDB реалізує з’єднання з базою даних MySQL. Дані та методи класу наведені в табл. 3.9.

Таблиця 3.9 – Дані та методи класу connectionDB

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| connectionDB::connect() | Підключення до бази даних MySQL |
| ***private:*** | |
| QSqlDatabase db | Змінна для підключення до бази даних |

Клас jsondb реалізує з’єднання з базою даних JSON. Дані та методи класу наведені в табл. 3.10.

Таблиця 3.10 – Дані та методи класу jsondb

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| jsondb(); | Конструктор |
| insert(QString insert); | Функція для занесення в json формат |
| QString parse(); | Функція для парсинга з json формату в звичайний |

Клас ShowWindowреалізує показ заходів, записаних в програмі «TimeTask». Дані та методи класу наведені в табл. 3.11.

Таблиця 3.11 – Дані та методи класу ShowWindow

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| explicit ShowWindow(QWidget \*parent = 0) | Конструктор |
| ~ShowWindow() | Деструктор |
| ***private slots:*** | |
| void on\_pushButton\_clicked() | закриває вікно |
| void on\_pushButton\_2\_clicked() | підтверджує дату початку заходу |
| void on\_pushButton\_4\_clicked() | підтверджує захід |
| ***private:*** | |
| Ui::ShowWindow \*ui | Покажчик на форму "ShowWindow" |
| QSqlDatabase db | Змінна для підключення до бази даних |
| QSqlTableModel \*model | Змінна для створення редагованої моделі даних для однієї таблиці бази даних |
| QSqlTableModel \*arrays |
| QSqlTableModel \*plans |
| QSqlTableModel \*all |

Клас soundsреалізує cупроводження звуків при успішному або невдалому процесі в ході роботи з програмою «TimeTask». Дані та методи класу наведені в табл. 3.12.

Таблиця 3.12 – Дані та методи класу sounds

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| sounds(); | Конструктор |
| succecful() | Звукове повідомлення при успішному ході роботи з програмою |
| error() | Звукове повідомлення при помилці у ході роботи з програмою |

Клас usertimeреалізує поточну дату, яку можна побачити в головному меню програми. «TimeTask». Дані та методи класу наведені в табл. 3.13.

Таблиця 3.13 – Дані та методи класу usertime

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| ***public:*** | |
| *usertime();* | Конструктор |
| QString timeEvent() | Відображає теперешню дату |

* 1. Основні рішення щодо розробки інтерфейсу користувача
     1. Загальна інформація

Програма «TimeTask» містить у собі 9 вікон. Дизайн всіх вікон виконаний в одному стилі. Основні кольори елементів програми – оранжевий та чорний, було обрано відповідно до кольорів які містяться у логотипі програми.

* + 1. **Вікно «TimeTask - Головне вікно»**

Вікно «TimeTask - Головне вікно» містить у собі: рядок меню, логотип програми, 7 кнопок та панель відображення поточних дати та часу (рис. 3.1)

Натискання на кожну кнопку викликає відповідне вікно (табл. 3.14)

Таблиця 3.14 – кнопки та дії кнопок.

|  |  |  |
| --- | --- | --- |
| № | Кнопка | Вікно |
| 1 | «Показати заходи» | «TimeTask - Про заходи» |
| 2 | «Додати захід» | «TimeTask - Додати захід» |
| 3 | «Редагування заходу» | «TimeTask - Додати захід» |
| 4 | «Додати плани заходів» | «TimeTask - Додати план» |
| 5 | «Редагувати плани заходів» | «TimeTask - Редагування плану» |
| 6 | «Видалити плани заходів» | «TimeTask - Видалення планів» |
| 7 | «Видалити захід» | «TimeTask - Видалення заходів» |

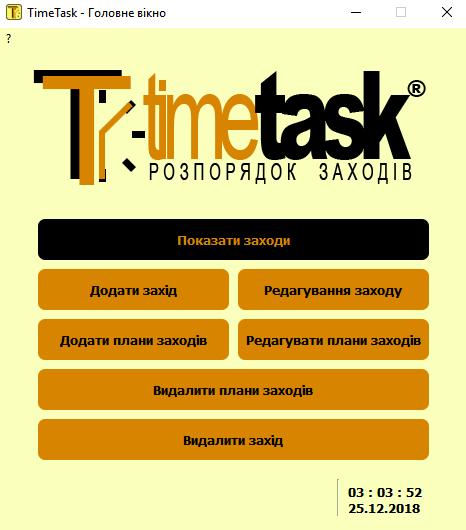


Рисунок 3.1 – Вікно «TimeTask - Головне вікно»

* + 1. Вікно «TimeTask – Про заходи»

Вікно «TimeTask – Про заходи» містить у собі: 3 кнопки, 2 елементи вибору та 2 поля виводу інфорації (рис. 3.2).

Натискання на кожну кнопку викликає певну дію. Кнопка «ОК» підставляє в елемент вибору заходів, ті заходи, які мають ту ж початкову дату, що міститься в елементі вибору початкової дати. Кнопка «Вибрати захід» відображає інформацію про захід, який обраний у полі вибору заходу. Інформація виводиться у поля виводу інформації про захід та план цього заходу. Кнопка «Закрити» закриває дане вікно.

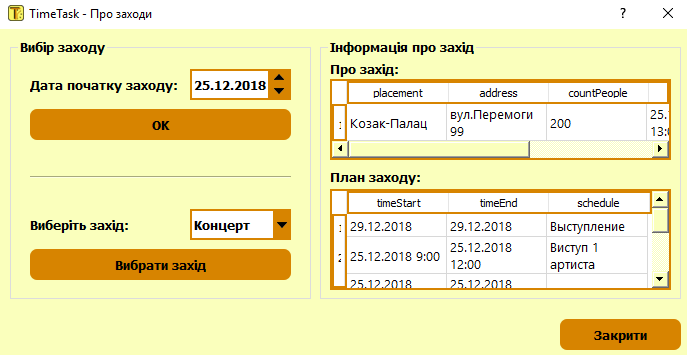


Рисунок 3.2 – Вікно «TimeTask – Про заходи»

* + 1. ***Вікно «TimeTask – Додати захід»***

Вікно «TimeTask – Додати захід» містить у собі: 2 кнопки, 2 елементи вибору та 4 текстових поля вводу інформації (рис. 4.3).

В кожне текстове поле вводиться відповідна інформація. В елементи вибору інформації вводиться початкова та кінцева дата заходу. Натискання на кожну кнопку викликає певну дію. Кнопка «Додати» додає захід в базу даних, з інформацією, яка введена вище. Кнопка «Закрити» закриває дане вікно.

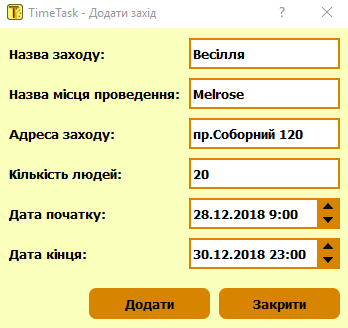


Рисунок 3.3 – Вікно «TimeTask – Додати захід»

* + 1. Вікно «TimeTask – Редагування заходу»

Вікно «TimeTask – Редагування заходу» містить у собі: 2 кнопки, елемент вибору та таблицю для редагування інформації (рис. 3.4).

В елементі вибору інформації обирається захід, інформацію про який потрібно редагувати. В таблиці для редагування інформації можна відредагувати інформацію про захід. Натискання на кожну кнопку викликає певну дію. Кнопка «Відредагувати» редагує інформацію про захід, відповідно до введеної вище інформації та заносить її до бази даних. Кнопка «Закрити» закриває дане вікно.

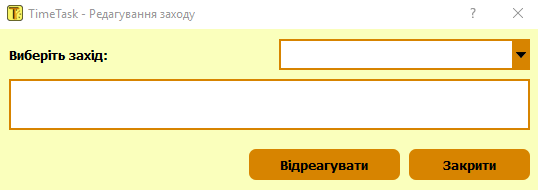


Рисунок 3.4 – Вікно «TimeTask – Редагування заходу»

* + 1. Вікно «TimeTask – Додати план»

Вікно «TimeTask – Додати план» містить у собі: 2 кнопки, 3 елемент вибору та текстове поле вводу інформації (рис. 3.5).

В елементах вибору інформації обираються захід, початкова та кінцева дати пункту плану.В текстове поле вводиться назва пункту плану.Кнопка «Додати план в захід» додає пункт плану до відповідно обраного заходу.Кнопка «Закрити» закриває дане вікно.

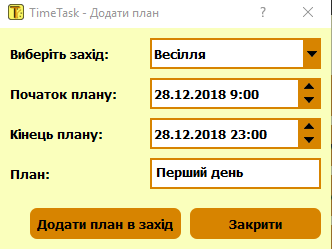


Рисунок 3.5 – Вікно «TimeTask – Додати план»

* + 1. Вікно «TimeTask – Редагування плану»

Вікно «TimeTask – Редагування плану» містить у собі: 2 кнопки, елемент вибору, таблицю для редагування інформації (рис. 3.6).

В елементі вибору інформації обирається захід, план якого потрібно редагувати. В таблиці для редагування інформації можна відредагувати інформацію про план заходу. Натискання на кожну кнопку викликає певну дію. Кнопка «Відредагувати» редагує інформацію про план заходу, відповідно до введеної вище інформації та заносить її до бази даних. Кнопка «Закрити» закриває дане вікно.

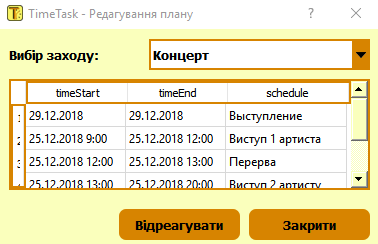


Рисунок 3.6 – Вікно «TimeTask – Редагування плану»

* + 1. Вікно «TimeTask – Видалення планів»

Вікно «TimeTask – Видалення планів» містить у собі: 2 кнопки, елемент вибору, таблицю для редагування інформації та текстове поле вводу інформації (рис. 3.7).

В елементі вибору інформації обирається захід, план якого потрібно видалити. В таблиці для редагування інформації можна дізнатися ID пункту плану, який необхідно видалити. В текстове поле вводиться ID пункту плану. Натискання на кожну кнопку викликає певну дію. Кнопка «Видалити» видаляє обраний пункт плану з бази даних. Кнопка «Закрити» закриває дане вікно.

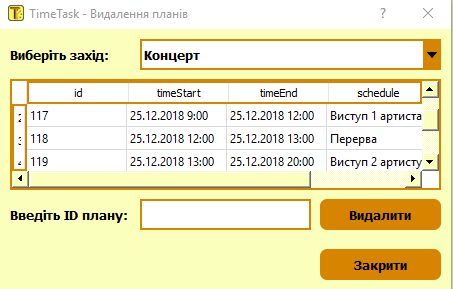


Рисунок 3.7 – Вікно «TimeTask – Видалення планів»

* + 1. Вікно «TimeTask – Видалення заходів»

Вікно «TimeTask – Видалення заходів» містить у собі: 2 кнопки та елемент вибору (рис. 3.8).

В елементі вибору інформації обирається захід, який потрібно видалити. Натискання на кожну кнопку викликає певну дію. Кнопка «Видалити» видаляє обраний захід з бази даних. Кнопка «Закрити» закриває дане вікно.

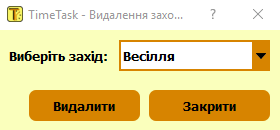


Рисунок 3.8 – Вікно «TimeTask – Видалення заходів»

* + 1. Вікно «TimeTask – Про програму»

Вікно «TimeTask – Про програму» містить у собі: кнопку, коротку текстову інформацію про програму та альтернативний логотип програми (рис. 3.9).

Кнопка «Закрити» закриває дане вікно.

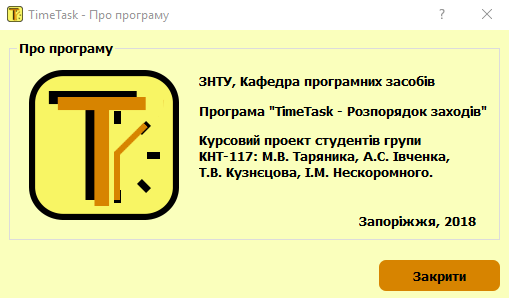


Рисунок 3.9 – Вікно «TimeTask – Про програму»

* 1. Основні рішення щодо збереження даних

Для роботи з даними були використані такі бібліотеки: QtSql/QSqlDatabase, QtSql/QSqlQuery, QtSql/QSqlTableMode, QJsonObject, QJsonDocument, QJsonValue, QByteArray. БД у роботі використовується для збереження та відправлення даних, які були підготовленні раніше. JSON використовується для збереження часа входу користувача в програму.

* + 1. Основні рішення щодо працювання з таблицями

Для роботи з БД використовуються такі QSL-запити:

* + "INSERT INTO ім’я\_таблиці (поле1, поле2, …, полеN) VALUES (:значення1, :значення2, …, :значенняN)" – параметричний запит на вставку в таблицю у перераховані поля перерахованих значень;
  + "SELECT значення\_поля1, значення\_поля2, …, значення\_поляN FROM ім’я\_таблиці" – запит на отримання значень перерахованих полів з таблиці;
  + "UPDATE <Ім'я таблиці> SET <Ім'я поля> = <Значення>,..., <Ім'я поля> = <Значення> [ WHERE <Критерій відбору>];" – запит на внесення змін в таблицю;
  + "CREATE TABLE назва (перелік полів та їх типу)" – запит на створення таблиці в БД.

Для того, щоб вказати передати параметр використовується метод QSqlQuery::exec(QString), у якому аргумент – це параметр у запиті.

З'єднання з БД відбувається наступним чином:

Скріншот з’єднання за базою данних можно побачити на рис. 3.10

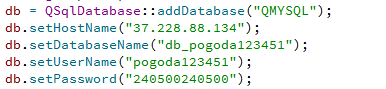


Рисунок 3.10 – Скріншот з’єднання з базою даних MYSQL

Запит для внесення даних до бази даних зображений на рис. 3.11

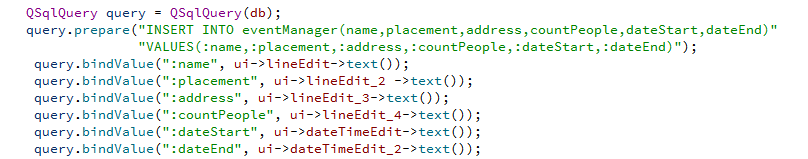


Рисунок 3.11– Скріншот запиту для внесення даних до БД у програмі TimeTask

Запит на виведення даних з бази даних до нашої змінної, приклад зображено на рис. 3.12

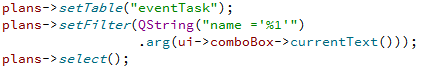


Рисунок 3.12 – Скріншот запиту для виведення даних з БД у програмі TimeTask

Запит на видалення даних з бази даних зображений на рис. 3.13



Рисунок 3.13 – Скріншот запиту для видалення даних з БД у програмі TimeTask

Скріншот роботи з БД зображений на рис 3.14

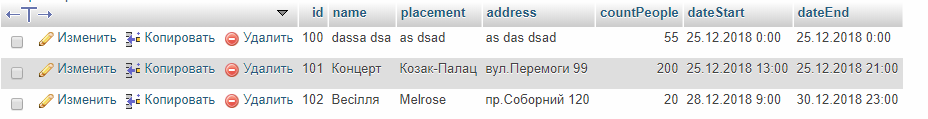


Рисунок 3.14 – Скріншот результату роботи з БД у програмі TimeTask

Занесення даних в JSON формат, приклад зображено на рис. 3.15

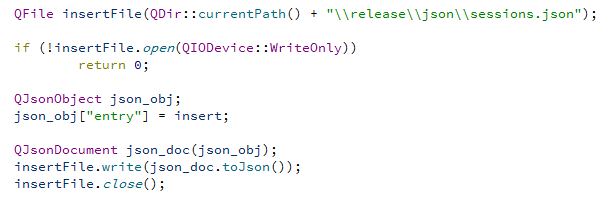


Рисунок 3.15 – Скріншот занесення даних в JSON формат у програмі TimeTask

Вилучення даних з JSON формату, приклад зображено на рис. 3.15

 Рисунок 3.15 – Скріншот вилучення даних з JSON у програмі TimeTask

1. **Керівництво програміста**
   1. Призначення та умови застосування програми

Призначення програми – забезпечення системи обліку споруд та будівель організації.

Умовою застосування для коректної роботи є використання останніх версій П3, що необхідно для роботи програми. Для отримання повного функціоналу цілком достатній коректне введення даних у таблицю.

* 1. Характеристики програми

Програма виконана за допомогою мови програмування високого рівня С++ в середовищі розробки Qt Creator 5.3. Проект (рис. 4.1) містить класи, їх реалізацію, файли ресурсів та проекту.

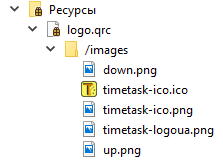
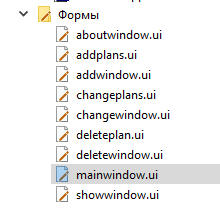
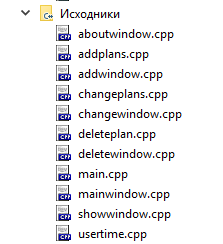
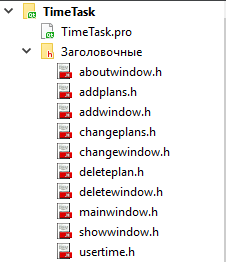


Рисунок 4.1 – Структура проекту

* 1. Звертання до програми

Для звернення, по-перше, необхідно розархівувати папку з програмою та розташувати її у каталог, до якого має доступ оператор. По-друге, для запуску програми у папці TimeTask (рис.4.2) треба запустити файл TimeTask.pro, дочекатися запуску проекту. По-третє, натиснути кнопку «Сборка» і програма готова до роботи.

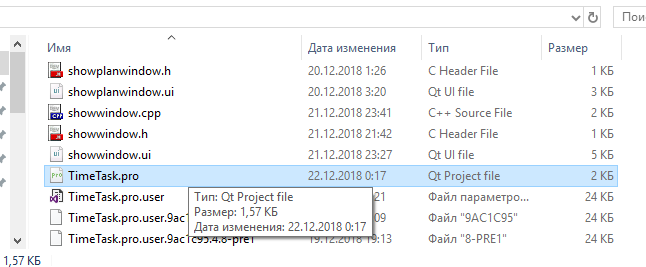


Рисунок 4.2 – Папка з файлами програми

* 1. Вхідні та вихідні дані

Вхідними даними виступають різні набори даних: інформація про захід (найменування, місце проведення, адреса закладу, дата початку заходу, дата закінцення заходу); інформація про план заходу (найменування пункт плану, id пункту плану, дата початку пункту плану, дата закінчення пункту плану). Вихідними даними виступають дані, розміщенні в елементах виводу інформації на вікнах програми.

Також в програмі реалізовано обробка виняткових ситуацій для окремих класів, що оброблюють помилки відкриття вхідних та вихідних файлів.

Так як в нашій програмі ми використовуємо базу даних типу MYSQL то

спочатку ми, приєднуємося до нашого серверу за допомогою IP, логіну і паролю від бази. Після посилання запиту дані записуються до таблиці.

* 1. Повідомлення

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі оператор отримає інформаційні повідомлення щодо деталей помилки.

При роботі можуть бути отримані такі повідомлення:

* помилка заповнення даних в полях вводу інформації, така помилка може виникнути у випадку, якщо в полі вводу інформації дані відсутні або некоректно введені (рис. 4.3). В такому разі необхідно перевірити правильність введення даних в поля введення інфомації.

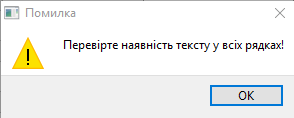


Рисунок 4.3 – Помилка вводу даних

* Помилка підключення до бази даних через посилання до бази некоректного пакету даних (рис. 4.4).

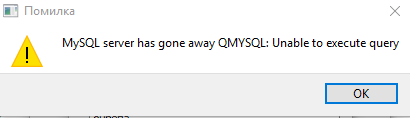


Рисунок 4.4 – Помилка БД

1. **ІНСТРУКЦІЯ КОРИСТУВАЧА**
   1. Призначення програми

Програма приначена для робот з різними наборами інформації, їх оновлення, зміни та зберегання в окремих текстових файлах.

* 1. Умови виконання програми

Програма має коректно працювати за умов:

- запуску останньої версії програми;

- збереження наборів інформації у директорії, в яких є права на читання та запис файлів, що не є заблокованими;

- правильного вводу даних до таблиці, згідно із вимогами.

* 1. Як запустити програму

Для роботи у програмі перш за все необхідно розархівувати теку з програмою та розташувати її у каталог, до якого ви маєте доступ. Для запуску застосунку у папці TimeTask (рис.5.1) треба запустити файл TimeTask.pro, дочекатися запуску проекту та натиснути кнопку «Сборка».Програма готова до роботи.

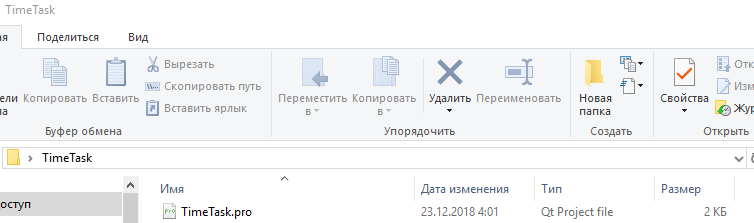


Рисунок 5.1 – Каталог з файлами програми

* 1. Виконання програми

Після того, як було запущено програму, перед користувачем постає стартова форма з кнопками переходу на інші форми. Головна форма зображена на рис. 5.2.

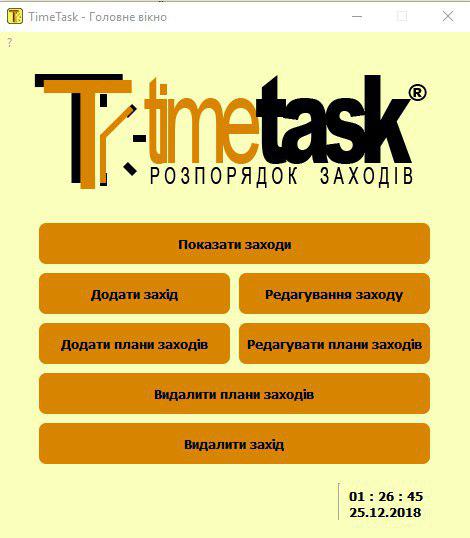


Рисунок 5.2 – Головна форма програми

Після натискання на кнопку «Показати заходи» користувач переходить до дати початку заходу, переліку заходів та планів, які принадлежать до кожного заходу окремо(рис. 5.3).

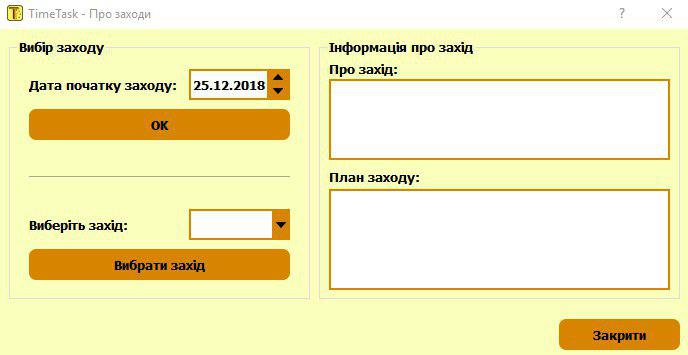


Рисунок 5.3 – Форма показу заходів

Для додавання нового заходу користувач повинен натиснути на кнопку «Додати захід», після чого заповнити отриману форму зображену на рис. 5.4. А щоб додати план заходу, користувач повинен натиснути на кнопку «Додати плани заходів», після чого заповнити отриману форму (рис. 5.5).

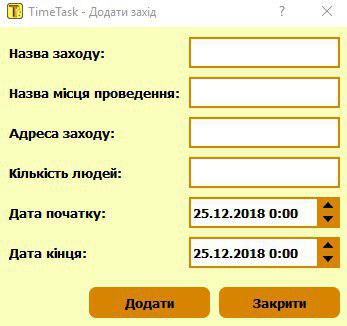


Рисунок 5.4 – Форма додавання заходу

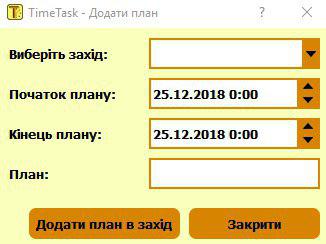


Рисунок 5.5 – Форма додавання плану заходу

Для редагування існуючого заходу користувач повинен натиснути на кнопку «Редагування заходу», після чого отримаємо форму зображену на рис. 5.6. А щоб відредагувати план заходу, користувач повинен натиснути на кнопку «Відредагувати плани заходів», після чого отримаємо форму (рис. 5.7).

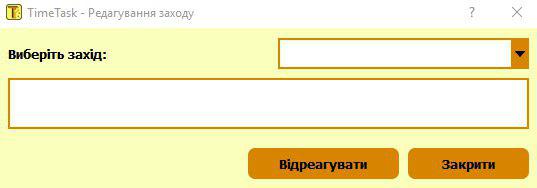


Рисунок 5.6 – Форма редагування заходу

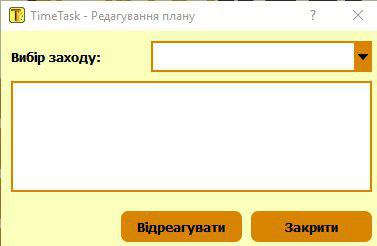


Рисунок 5.7 – Форма редагування планів заходу

Для видалення існуючого заходу користувач повинен натиснути на кнопку «Видалити захід», після чого зробити вибір в отриманій формі зображеної на рис. 5.8. А щоб видалити плани існуючого заходу, користувач повинен натиснути на кнопку «Видалити плани заходів», зробити вибір в отриманій формі (рис. 5.9).

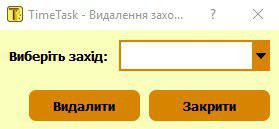


Рисунок 5.8 – Форма видалення заходу

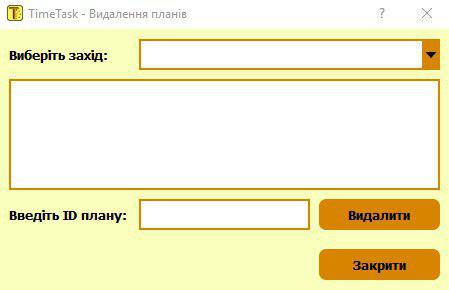


Рисунок 5.9 – Форма видалення плану заходу

* 1. Повідомлення користувачу

В процесі роботи програми можуть виникнути помилки в роботі  
програми. В такому разі користувач отримає інформаційні повідомлення щодо  
деталей помилки.

У системі реалізовано функції винятків, тому усі непередбачені ситуації  
оброблюються та програма адекватно реагує Одне з можливих повідомлень, що  
можуть бути отримані при роботі – помилка при відсутності тексту в рядках (рис. 5.10) . Також, при розірваному з'єднанні з SQL сервер, з’являєтся повідомлення зображенне на рис 5.11.

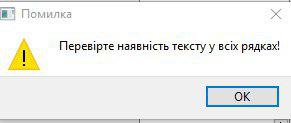


Рисунок 5.10 – Повідомлення при при відсутності тексту в деяких рядках

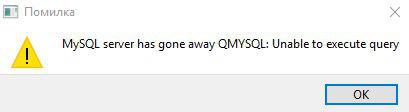


Рисунок 5.11 – Повідомлення при при розірваному з'єднанні з SQL сервер

ВИСНОВКИ

Під час виконання курсового проекту була розроблена «Програма розпорядку заходів «TimeTask»».

Створена програма призначена для користувача у вигляді файлу “TimeTask.exe”.

Також було розглянуто та проаналізовано існуючі програмні засоби, які виконують аналогічні завдання. А саме було розглянуто такі програми, як: «EventLook», «Wanderlist», «GoogleNow». Розгляд даних програм дав можливість грамотно поставити завдання до роботи та приступити до реалізації програмного продукту.

Було вирішено завдання, які ставив курсовий проект:

* + розглянуто та проаналізовано подібні існуючі програмні засоби;
  + поставлено завдання до роботи;
  + розглянуто можливості мови програмування С++, середовища розробки QtCreator;
  + розроблена «Програма розпорядку заходів «TimeTask» »;
  + описано основні рішення з реалізації компонентів системи;
  + реалізовано «Керівництво програміста» згідно з ГОСТом 19.504-79;
  + реалізовано «Інструкція користувача» згідно з ГОСТом 19.503-79.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бланшет Ж. Qt 4: программирование GUI на C++ [Текст] Ж.Бланшет, М.Саммерфильд. – М.:КУДИЦ-ПРЕСС, 2008. – 736 с.
2. Табунщик Г.В. Методичні вказівки до виконання курсової роботи з дисципліни “Об’єктно-орієнтоване програмування” для студентів напряму підготовки 6.050103 «Програмна інженерія» всіх форм навчання. Частина 2. Основні теоретичні відомості [Текст] /Укл.: Г.В. Табунщик, Г.В. Неласа, Н.О. Миронова – Запоріжжя: ЗНТУ, 2010. – 70 с.
3. Шилдт Г. Полный справочник по C++ [Текст] / Г. Шилдт. – М.: «Вильямс». 2007. – 800 с.
4. Просто о Qt. Библиотека контейнеров: [Електронний ресурс]. – Режим доступу: https://habrahabr.ru/post/127870/.
5. Работа с базами данных в Qt: [Електронний ресурс]. – Режим доступу: https://habrahabr.ru/post/51650/.

6. С++: [Електронний ресурс]. – Режим доступу:

https://uk.wikipedia.org/wiki/C%2B%2B.

1. JSON Support in Qt: [Електронний ресурс]. – Режим доступу: http://doc.qt.io/qt-5/json.html.
2. Qt: [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Qt.
3. Qt и базы данных:[Електроннийресурс].–Режим доступу:

https://ru.opensuse.org/QT\_и\_Базы\_данных.

10.Qt Довідкова Документація: [Електронний ресурс]. – Режим доступу: http://qtdocs.narod.ru/4.1.0/doc/html/index.html.

**ДОДАТОК А**

**ТЕКСТ ПРОГРАМИ**

Модуль aboutwindow.h:

#ifndef ABOUTWINDOW\_H

#define ABOUTWINDOW\_H

#include <QDialog>

namespace **Ui** {

class **AboutWindow**;

}

class **AboutWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **AboutWindow**(QWidget \*parent = 0);

~***AboutWindow***();

private slots:

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

private:

Ui::AboutWindow \*ui;

};

#endif // ABOUTWINDOW\_H

Модуль aboutwindow.cpp:

#include "aboutwindow.h"

#include "ui\_aboutwindow.h"

AboutWindow::AboutWindow(QWidget \*parent) :

QDialog(parent),

ui(new Ui::AboutWindow)

{

ui->setupUi(this);

}

AboutWindow::~AboutWindow()

{

delete ui;

}

void AboutWindow::on\_pushButton\_clicked()

{

close();

}

Модуль addplans.h:

#ifndef ADDPLANS\_H

#define ADDPLANS\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **AddPlans**;

}

class **AddPlans** : public QDialog

{

Q\_OBJECT

public:

explicit **AddPlans**(QWidget \*parent = 0);

~***AddPlans***();

private slots:

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

void **on\_pushButton\_2\_clicked**();

private:

Ui::AddPlans \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

};

#endif // ADDPLANS\_H

Модуль addplans.cpp:

#include "addplans.h"

#include "ui\_addplans.h"

#include <QDebug>

#include <QDate>

#include <QTextEdit>

#include <mainwindow.h>

AddPlans::**AddPlans**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::AddPlans)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

model->*setTable*("eventManager");

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

QDate date = QDate::currentDate();

ui->dateTimeEdit->setDate(date);

ui->dateTimeEdit\_2->setDate(date);

}

AddPlans::~***AddPlans***()

{

delete ui;

}

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

{

messages print;

/\* QString enter = "";

QString text = ui->textEdit->toPlainText();

int count = ui->textEdit->toPlainText().length();

bool first = false;

QFile fix(QDir::currentPath() + "\\release\\fix.txt");

fix.open(QIODevice::ReadOnly);

enter = fix.readAll(); \*/

if(ui->textEdit->toPlainText().length() == 0)

{

print.message(0);

return;

}

/\* else

{

for(int i = 0; i<count; i++)

{

if(text.at(i) == enter)

text.remove(i,i+1);

if(i % 40 == 0)

text.insert(i,enter);

}

QMessageBox::information(0,0,text);

} \*/

QSqlQuery query = QSqlQuery(db);

query.prepare("INSERT INTO eventTask(name,timeStart,timeEnd,schedule)"

"VALUES(:name,:timeStart,:timeEnd,:schedule)");

query.bindValue(":name", ui->comboBox->currentText());

query.bindValue(":timeStart", ui->dateTimeEdit->text());

query.bindValue(":timeEnd", ui->dateTimeEdit\_2->text());

query.bindValue(":schedule", ui->textEdit->toPlainText());

try

{

if(!query.exec())

{

qDebug() << query.lastError().text();

print.messageArguments(1,query.lastError().text());

}

else

{

print.messageArguments(0,ui->comboBox->currentText());

ui->textEdit->setText("");

}

} catch(const std::exception& e) { qDebug()<<e.*what*(); }

}

void AddPlans::**on\_pushButton\_2\_clicked**()

{

close();

}

Модуль addwindow.h:

#ifndef ADDWINDOW\_H

#define ADDWINDOW\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **AddWindow**;

}

class **AddWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **AddWindow**(QWidget \*parent = 0);

~***AddWindow***();

private slots:

void **on\_pushButton\_2\_clicked**();

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

private:

Ui::AddWindow \*ui;

QSqlDatabase db;

};

#endif // ADDWINDOW\_H

Модуль addwindow.cpp:

#include "addwindow.h"

#include "ui\_addwindow.h"

AddWindow::**AddWindow**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::AddWindow)

{

ui->setupUi(this);

QDate date = QDate::currentDate();

ui->dateTimeEdit->setDate(date);

ui->dateTimeEdit\_2->setDate(date);

}

AddWindow::~***AddWindow***()

{

delete ui;

}

void AddWindow::**on\_pushButton\_2\_clicked**()

{

close();

}

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

{

messages print;

if(ui->lineEdit->text().length() == 0 || ui->lineEdit\_2->text().length() == 0 ||

ui->lineEdit\_3->text().length() == 0 || ui->lineEdit\_4->text().length() == 0)

{

print.message(1);

return;

}

QSqlQuery query = QSqlQuery(db);

query.prepare("INSERT INTO eventManager(name,placement,address,countPeople,dateStart,dateEnd)"

"VALUES(:name,:placement,:address,:countPeople,:dateStart,:dateEnd)");

query.bindValue(":name", ui->lineEdit->text());

query.bindValue(":placement", ui->lineEdit\_2 ->text());

query.bindValue(":address", ui->lineEdit\_3->text());

query.bindValue(":countPeople", ui->lineEdit\_4->text());

query.bindValue(":dateStart", ui->dateTimeEdit->text());

query.bindValue(":dateEnd", ui->dateTimeEdit\_2->text());

try

{

if(!query.exec())

{

qDebug() << query.lastError().text();

print.messageArguments(1,query.lastError().text());

}

else

{

print.messageArguments(2,ui->lineEdit->text());

ui->lineEdit->setText("");

ui->lineEdit\_2->setText("");

ui->lineEdit\_3->setText("");

ui->lineEdit\_4->setText("");

}

} catch(const std::exception& e) { qDebug()<<e.*what*(); }

}

Модуль changeplans.h:

#ifndef CHANGEPLANS\_H

#define CHANGEPLANS\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **ChangePlans**;

}

class **ChangePlans** : public QDialog

{

Q\_OBJECT

public:

explicit **ChangePlans**(QWidget \*parent = 0);

~***ChangePlans***();

private slots:

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

void **on\_pushButton\_2\_clicked**();

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

void **updateComboBox**();

private:

Ui::ChangePlans \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

QSqlTableModel \*plans;

};

#endif // CHANGEPLANS\_H

Модуль changeplans.cpp:

#include "changeplans.h"

#include "ui\_changeplans.h"

ChangePlans::**ChangePlans**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ChangePlans)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

plans = new QSqlTableModel(this,db);

updateComboBox();

}

ChangePlans::~***ChangePlans***()

{

delete ui;

}

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

{

messages print;

if(plans->submitAll())

print.message(2);

else

print.message(3);

updateComboBox();

}

void ChangePlans::**updateComboBox**()

{

model->*setTable*("eventManager");

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

}

void ChangePlans::**on\_pushButton\_2\_clicked**()

{

close();

}

void ChangePlans::**on\_comboBox\_currentTextChanged**(const QString &arg1)

{

plans->*setTable*("eventTask");

plans->*setEditStrategy*(QSqlTableModel::OnManualSubmit);

plans->*setFilter*(QString("name ='%1'")

.arg(ui->comboBox->currentText()));

plans->*select*();

plans->*removeRows*(0,0);

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

int i =0;

while (i < ui->tableView->verticalHeader()->count())

ui->tableView->verticalHeader()->setSectionResizeMode(i,QHeaderView::ResizeToContents),i++;

ui->tableView->setColumnHidden(0, true);

ui->tableView->setColumnHidden(1, true);

ui->tableView->horizontalHeader()->setStretchLastSection( true );

}

Модуль changewindow.h:

#ifndef CHANGEWINDOW\_H

#define CHANGEWINDOW\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **changeWindow**;

}

class **changeWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **changeWindow**(QWidget \*parent = 0);

~***changeWindow***();

private slots:

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

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

void **on\_pushButton\_2\_clicked**();

private:

Ui::changeWindow \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

QSqlTableModel \*plans;

};

#endif // CHANGEWINDOW\_H

Модуль changewindow.cpp:

#include "changewindow.h"

#include "ui\_changewindow.h"

changeWindow::**changeWindow**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::changeWindow)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

plans = new QSqlTableModel(this,db);

model->*setTable*("eventManager");

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

}

changeWindow::~***changeWindow***()

{

delete ui;

}

void changeWindow::**on\_comboBox\_currentTextChanged**(const QString &arg1)

{

plans->*setTable*("eventManager");

plans->*setEditStrategy*(QSqlTableModel::OnManualSubmit);

plans->*setFilter*(QString("name ='%1'")

.arg(ui->comboBox->currentText()));

plans->*select*();

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

int i =0;

while (i < ui->tableView->verticalHeader()->count())

ui->tableView->verticalHeader()->setSectionResizeMode(i,QHeaderView::ResizeToContents),i++;

ui->tableView->setColumnHidden(0, true);

ui->tableView->setColumnHidden(1, true);

}

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

{

messages print;

if(plans->submitAll())

print.message(2);

else

print.message(3);

}

void changeWindow::**on\_pushButton\_2\_clicked**()

{

close();

}

Модуль connectiondb.h:

#ifndef CONNECTIONDB\_H

#define CONNECTIONDB\_H

#include "mainwindow.h"

class **connectionDB**

{

public:

**connectionDB**();

connect();

private:

QSqlDatabase db;

};

#endif // CONNECTIONDB\_H

Модуль connectiondb.cpp:

#include "connectiondb.h"

#include "mainwindow.h"

#include "sounds.h"

connectionDB::**connectionDB**()

{

}

connectionDB::connect()

{

sounds ply;

messages print;

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

db.setHostName("37.228.88.134");

db.setDatabaseName("db\_pogoda123451");

db.setUserName("pogoda123451");

db.setPassword("240500240500");

if(!db.open())

{

qDebug() << db.lastError().text();

ply.error();

print.message(4);

qApp->exit();

}

else

{

ply.succecful();

qDebug() << "Success!";

}

}

Модуль deleteplan.h:

#ifndef DELETEPLAN\_H

#define DELETEPLAN\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **deletePlan**;

}

class **deletePlan** : public QDialog

{

Q\_OBJECT

public:

explicit **deletePlan**(QWidget \*parent = 0);

~***deletePlan***();

private slots:

void **on\_pushButton\_2\_clicked**();

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

void **updateEvent**();

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

private:

Ui::deletePlan \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

QSqlTableModel \*plans;

};

#endif // DELETEPLAN\_H

Модуль deleteplan.cpp:

#include "deleteplan.h"

#include "ui\_deleteplan.h"

deletePlan::**deletePlan**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::deletePlan)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

plans = new QSqlTableModel(this,db);

model->*setTable*("eventManager");

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

}

deletePlan::~***deletePlan***()

{

delete ui;

}

void deletePlan::**on\_pushButton\_2\_clicked**()

{

close();

}

void deletePlan::**updateEvent**()

{

plans->*setTable*("eventTask");

plans->*setFilter*(QString("name ='%1'")

.arg(ui->comboBox->currentText()));

plans->*select*();

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

int i =0;

while (i < ui->tableView->verticalHeader()->count())

ui->tableView->verticalHeader()->setSectionResizeMode(i,QHeaderView::ResizeToContents),i++;

ui->lineEdit->setText("");

ui->tableView->setColumnHidden(1, true);

ui->tableView->horizontalHeader()->setStretchLastSection( true );

}

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

{

QSqlQuery query = QSqlQuery(db);

messages print;

query.prepare(QString("DELETE FROM eventTask WHERE id ='%1'")

.arg(ui->lineEdit->text()));

try

{

if(!query.exec())

{

qDebug() << query.lastError().text();

print.messageArguments(1,query.lastError().text());

}

else

{

print.messageArguments(3,ui->lineEdit->text());

}

} catch(const std::exception& e) { qDebug()<<e.*what*(); }

updateEvent();

}

void deletePlan::**on\_comboBox\_currentTextChanged**(const QString &arg1)

{

updateEvent();

}

Модуль deletewindow.h:

#ifndef DELETEWINDOW\_H

#define DELETEWINDOW\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **DeleteWindow**;

}

class **DeleteWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **DeleteWindow**(QWidget \*parent = 0);

~***DeleteWindow***();

private slots:

void **on\_pushButton\_2\_clicked**();

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

void **updateComboBox**();

private:

Ui::DeleteWindow \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

};

#endif // DELETEWINDOW\_H

Модуль deletewindow.cpp:

#include "deletewindow.h"

#include "ui\_deletewindow.h"

DeleteWindow::**DeleteWindow**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::DeleteWindow)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

updateComboBox();

}

DeleteWindow::~***DeleteWindow***()

{

delete ui;

}

void DeleteWindow::**on\_pushButton\_2\_clicked**()

{

close();

}

void DeleteWindow::**updateComboBox**()

{

model->*setTable*("eventManager");

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

}

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

{

QSqlQuery query = QSqlQuery(db);

messages print;

try

{

query.prepare(QString("DELETE FROM eventTask WHERE name ='%1'")

.arg(ui->comboBox->currentText()));

if(!query.exec())

{

qDebug() << query.lastError().text();

}

query.prepare(QString("DELETE FROM eventManager WHERE name ='%1'")

.arg(ui->comboBox->currentText()));

if(!query.exec())

{

qDebug() << query.lastError().text();

print.messageArguments(1,query.lastError().text());

}

else

{

print.messageArguments(4,ui->comboBox->currentText());

}

} catch(const std::exception& e) { qDebug()<<e.*what*(); }

updateComboBox();

}

Модуль jsondb.h:

#ifndef JSONDB\_H

#define JSONDB\_H

#include "qstring.h"

class **jsondb**

{

public:

**jsondb**();

insert(QString insert);

QString **parse**();

};

#endif // JSONDB\_H

Модуль jsondb.cpp:

#include <jsondb.h>

#include <QJsonDocument>

#include <QJsonObject>

#include <QJsonValue>

#include <QString>

#include <QFile>

#include <QMessageBox>

#include <QDir>

jsondb::**jsondb**()

{

}

jsondb::insert(QString insert)

{

QFile insertFile(QDir::currentPath() + "\\release\\json\\sessions.json");

if (!insertFile.*open*(QIODevice::WriteOnly))

return 0;

QJsonObject json\_obj;

json\_obj["entry"] = insert;

QJsonDocument json\_doc(json\_obj);

insertFile.write(json\_doc.toJson());

insertFile.*close*();

}

QString jsondb::**parse**()

{

QByteArray data;

QFile parseFile(QDir::currentPath() + "\\release\\json\\sessions.json");

if (!parseFile.*open*(QIODevice::ReadOnly))

return 0;

data = parseFile.readAll();

QJsonDocument document = QJsonDocument::fromJson(data);

QJsonObject root = document.object();

QJsonValue date = root.value("entry");

parseFile.*close*();

return date.toString();

}

Модуль mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtSql>

#include <QtSql/QSqlDatabase>

#include <QtSql/QSqlQuery>

#include <QtSql/QSqlTableModel>

#include <QDebug>

#include <QMessageBox>

#include <QDateEdit>

#include <QTimer>

#include "messages.h"

namespace **Ui** {

class **MainWindow**;

}

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = 0);

~***MainWindow***();

private slots:

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

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **on\_pushButton\_4\_clicked**();

void **on\_action\_7\_triggered**();

void **on\_pushButton\_5\_clicked**();

void **on\_pushButton\_6\_clicked**();

void **on\_pushButton\_7\_clicked**();

void **timeManager**();

private:

Ui::MainWindow \*ui;

QSqlDatabase db;

};

#endif // MAINWINDOW\_H

Модуль mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "showwindow.h"

#include "addwindow.h"

#include "deletewindow.h"

#include "aboutwindow.h"

#include "addplans.h"

#include "changeplans.h"

#include "deleteplan.h"

#include "changewindow.h"

#include "usertime.h"

#include "connectiondb.h"

#include "messages.h"

#include "jsondb.h"

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

QDateTime dt = QDateTime::currentDateTime();

connectionDB DB;

DB.connect();

QTimer \*timer = new QTimer(this);

connect(timer,SIGNAL(timeout()),this,SLOT(timeManager()));

timer->start(1000);

jsondb json;

json.insert(dt.toString());

}

MainWindow::~***MainWindow***()

{

QDateTime dt = QDateTime::currentDateTime();

jsondb json;

QMessageBox::about(0,"Час користування","Вхід в програму: " + json.parse() + "\n\nВихід з програми: " + dt.toString());

delete ui;

}

void MainWindow::**timeManager**()

{

usertime a;

a.timeEvent();

QString returnText = a.timeEvent();

ui->label\_2->setText(returnText);

}

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

{

ShowWindow window;

window.setModal(true);

window.*exec*();

}

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

{

AddWindow window1;

window1.setModal(true);

window1.*exec*();

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

DeleteWindow window2;

window2.setModal(true);

window2.*exec*();

}

void MainWindow::**on\_pushButton\_4\_clicked**()

{

ChangePlans window3;

window3.setModal(true);

window3.*exec*();

}

void MainWindow::**on\_action\_7\_triggered**()

{

AboutWindow window4;

window4.setModal(true);

window4.*exec*();

}

void MainWindow::**on\_pushButton\_5\_clicked**()

{

AddPlans window5;

window5.setModal(true);

window5.*exec*();

}

void MainWindow::**on\_pushButton\_6\_clicked**()

{

deletePlan window6;

window6.setModal(true);

window6.*exec*();

}

void MainWindow::**on\_pushButton\_7\_clicked**()

{

changeWindow window7;

window7.setModal(true);

window7.*exec*();

}

Модуль messages.h:

#ifndef MESSAGES\_H

#define MESSAGES\_H

#include "qstring.h"

class **messages**

{

public:

**messages**();

message(int printX);

messageArguments(int printX, QString printY);

};

#endif // MESSAGES\_H

Модуль messages.cpp:

#include "messages.h"

#include "mainwindow.h"

messages::**messages**()

{

}

messages::message(int printX)

{

switch (printX) {

case 0:

QMessageBox::critical(0, "Помилка", "Додайте будь-ласка текст до вашого плану!");

break;

case 1:

QMessageBox::critical(0, "Помилка", "Перевірте наявність тексту у всіх рядках!");

break;

case 2:

QMessageBox::information(0, "Успішно", "Зміни внесені в базу даних!");

break;

case 3:

QMessageBox::critical(0, "Помилка", "Зміни не вдалося!");

break;

case 4:

QMessageBox::critical(0, "Критична помилка", "Не вдалося зв'язатися з базою даних!");

break;

default:

break; }

}

messages::messageArguments(int printX, QString printY)

{

switch (printX) {

case 0:

QMessageBox::information(0, "Успішно", "Інформація додана в плани [" + printY + "]");

break;

case 1:

QMessageBox::critical(0, "Помилка", printY);

break;

case 2:

QMessageBox::information(0, "Успішно", "Захід " + printY + " створено!");

break;

case 3:

QMessageBox::information(0, "Успішно", "План [ID=" + printY + "] був видалений!");

break;

case 4:

QMessageBox::information(0, "Успішно", "Захід " + printY + " був видалений!");

break;

default:

break; }

}

Модуль showwindow.h:

#ifndef SHOWWINDOW\_H

#define SHOWWINDOW\_H

#include <QDialog>

#include "mainwindow.h"

namespace **Ui** {

class **ShowWindow**;

}

class **ShowWindow** : public QDialog

{

Q\_OBJECT

public:

explicit **ShowWindow**(QWidget \*parent = 0);

~***ShowWindow***();

private slots:

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

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_4\_clicked**();

private:

Ui::ShowWindow \*ui;

QSqlDatabase db;

QSqlTableModel \*model;

QSqlTableModel \*arrays;

QSqlTableModel \*plans;

QSqlTableModel \*all;

};

#endif // SHOWWINDOW\_H

Модуль showwindow.cpp:

#include "showwindow.h"

#include "ui\_showwindow.h"

ShowWindow::**ShowWindow**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::ShowWindow)

{

ui->setupUi(this);

model = new QSqlTableModel(this,db);

arrays = new QSqlTableModel(this,db);

plans = new QSqlTableModel(this,db);

all = new QSqlTableModel(this,db);

all->*setTable*("eventManager");

all->*select*();

ui->comboBox->setModel(all);

ui->comboBox->setModelColumn(1);

QDate date = QDate::currentDate();

ui->dateEdit->setDate(date);

}

ShowWindow::~***ShowWindow***()

{

delete ui;

}

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

{

close();

}

void ShowWindow::**on\_pushButton\_2\_clicked**()

{

arrays->*setTable*("eventManager");

arrays->*setFilter*(QString("name ='%1'")

.arg(ui->comboBox->currentText()));

arrays->*select*();

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

int i=0;

while (i < ui->tableView->verticalHeader()->count())

ui->tableView->verticalHeader()->setSectionResizeMode(i,QHeaderView::ResizeToContents),i++;

plans->*setTable*("eventTask");

plans->*setFilter*(QString("name ='%1'")

.arg(ui->comboBox->currentText()));

plans->*select*();

ui->tableView\_2->*setModel*(plans);

i=0;

while (i < ui->tableView\_2->verticalHeader()->count())

ui->tableView\_2->verticalHeader()->setSectionResizeMode(i,QHeaderView::ResizeToContents),i++;

ui->tableView->setColumnHidden(0, true);

ui->tableView->setColumnHidden(1, true);

ui->tableView\_2->setColumnHidden(0, true);

ui->tableView\_2->setColumnHidden(1, true);

ui->tableView\_2->horizontalHeader()->setStretchLastSection( true );

}

void ShowWindow::**on\_pushButton\_4\_clicked**()

{

model->*setTable*("eventManager");

model->*setFilter*(QString("dateStart LIKE '%%1%'")

.arg(ui->dateEdit->text()));

model->*select*();

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(1);

}

Модуль sounds.h:

#ifndef SOUNDS\_H

#define SOUNDS\_H

class **sounds**

{

public:

**sounds**();

succecful();

error();

};

#endif // SOUNDS\_H

Модуль sounds.cpp:

#include "sounds.h"

#include "mainwindow.h"

#include <QMediaPlayer>

sounds::**sounds**()

{

}

sounds::succecful()

{

QMediaPlayer\* succesful = new QMediaPlayer();

succesful->setMedia(QUrl::fromLocalFile(QDir::currentPath() + "/sounds/succesful.mp3"));

succesful->play();

}

sounds::error()

{

QMediaPlayer\* error = new QMediaPlayer();

error->setMedia(QUrl::fromLocalFile(QDir::currentPath() + "/sounds/error.mp3"));

error->play();

}

Модуль usertime.h:

#ifndef USERTIME\_H

#define USERTIME\_H

#include "qstring.h"

class **usertime**

{

public:

**usertime**();

QString **timeEvent**();

};

#endif // USERTIME\_H

Модуль usertime.cpp:

#include "usertime.h"

#include "mainwindow.h"

usertime::**usertime**()

{

}

QString usertime::**timeEvent**()

{

QDateTime time = QDateTime::currentDateTime();

QString time\_text = time.toString("hh : mm : ss \n"

"dd.MM.yyyy");

return time\_text;

}

Модуль 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 Б**

**ІНТЕРФЕЙС ПРОГРАМИ**



Рисунок Б.1 – Головне вікно програми

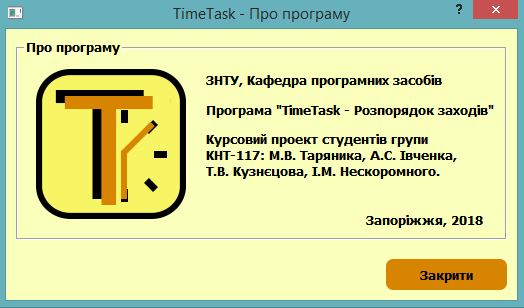


Рисунок Б.2 – Вікно «Про програму»

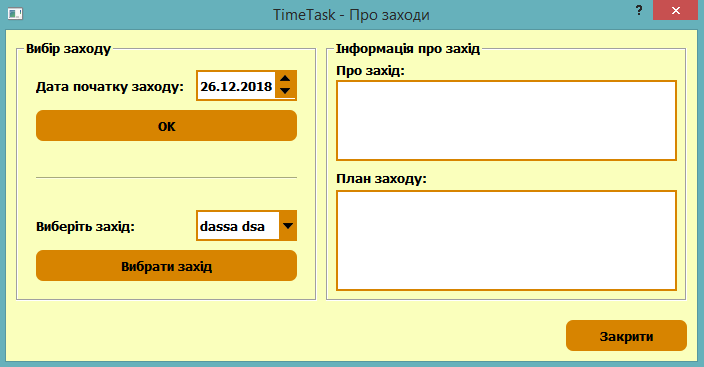


Рисунок Б.3 – Вікно «Про заходи»

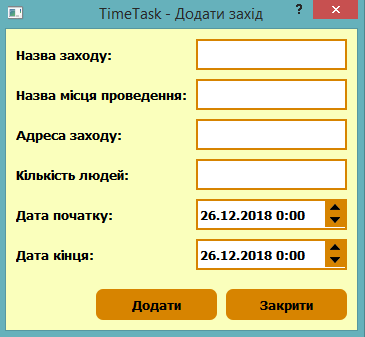


Рисунок Б.3 – Вікно «Додати захід»

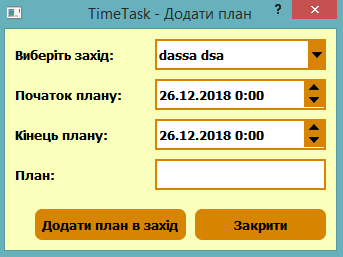


Рисунок Б.4 – Вікно «Додати план»

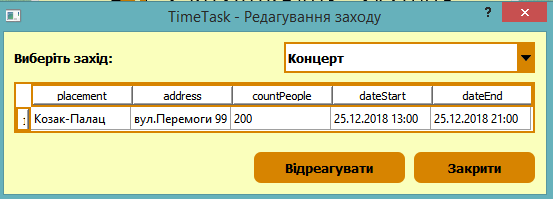


Рисунок Б.5 – Вікно «Редагування заходу»

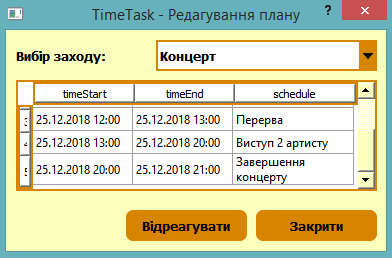


Рисунок Б.6 – Вікно «Редагування плану»

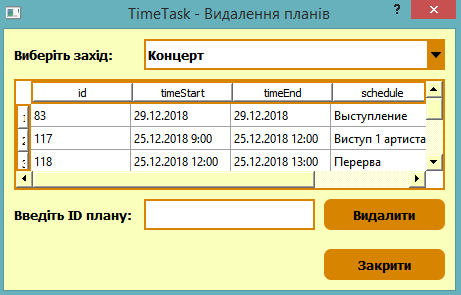


Рисунок Б.7 – Вікно «Видалення планів»

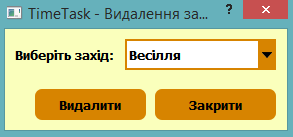


Рисунок Б.8 – Вікно «Видалення заходу»

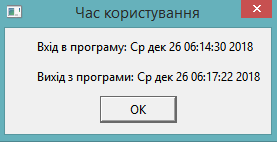


Рисунок Б.9 – Повідомлення «Час користування»

**ДОДАТОК В**

**СКРІНШОТИ ПРЕЗЕНТАЦІЇ**

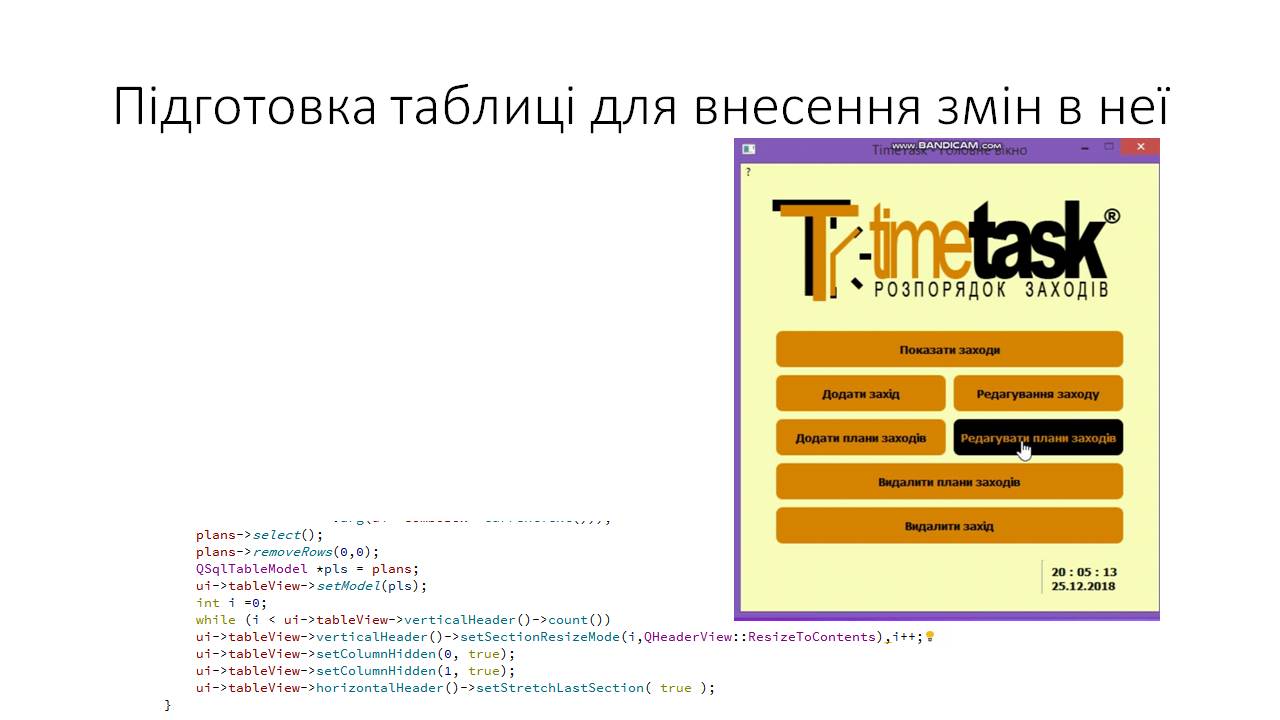
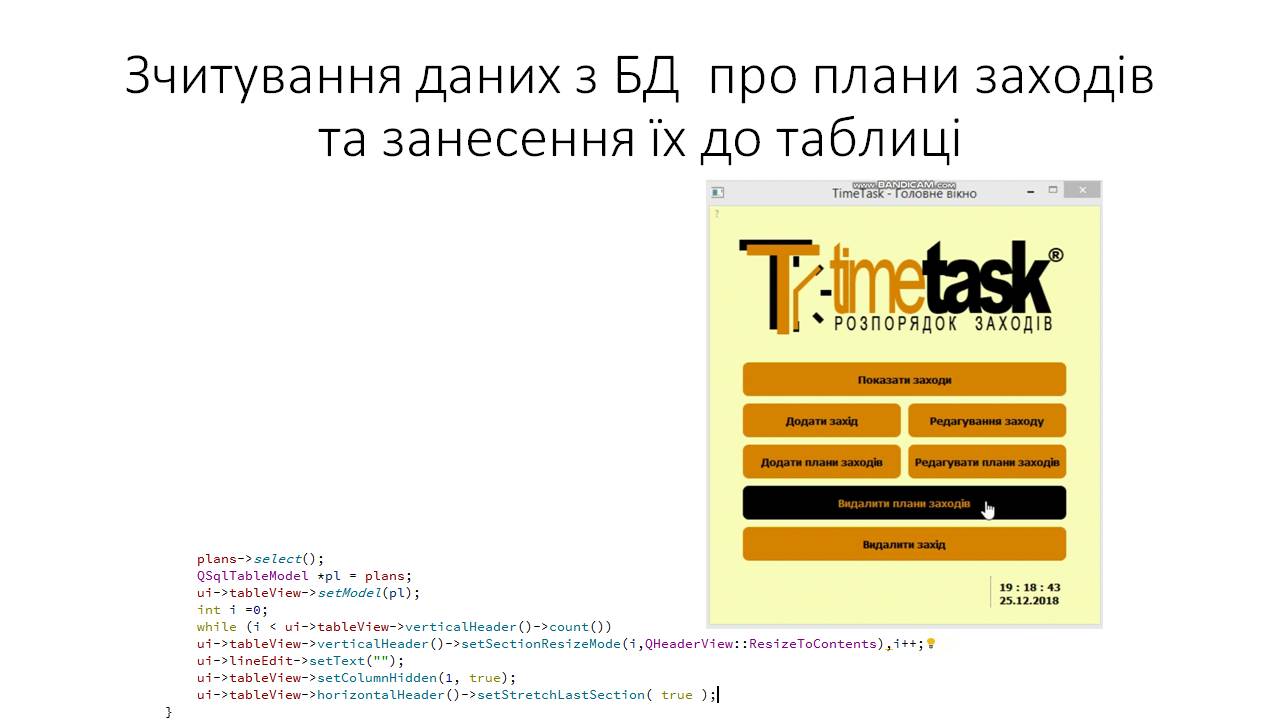
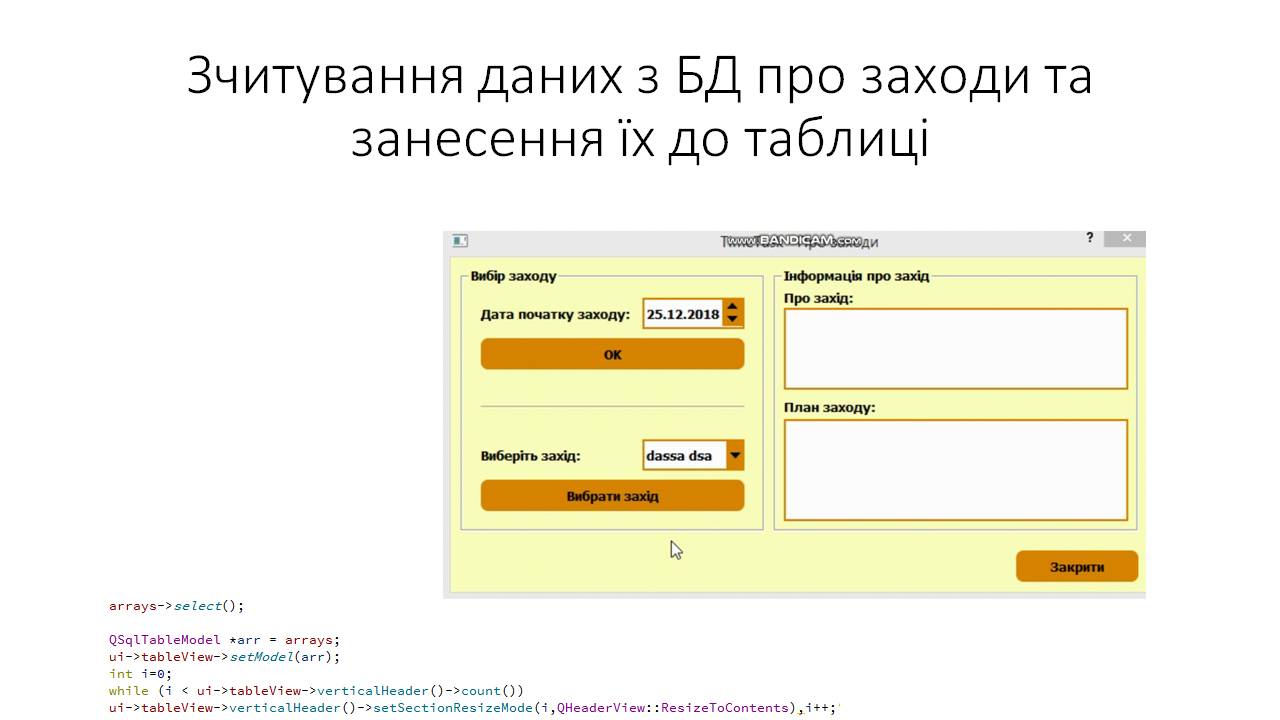
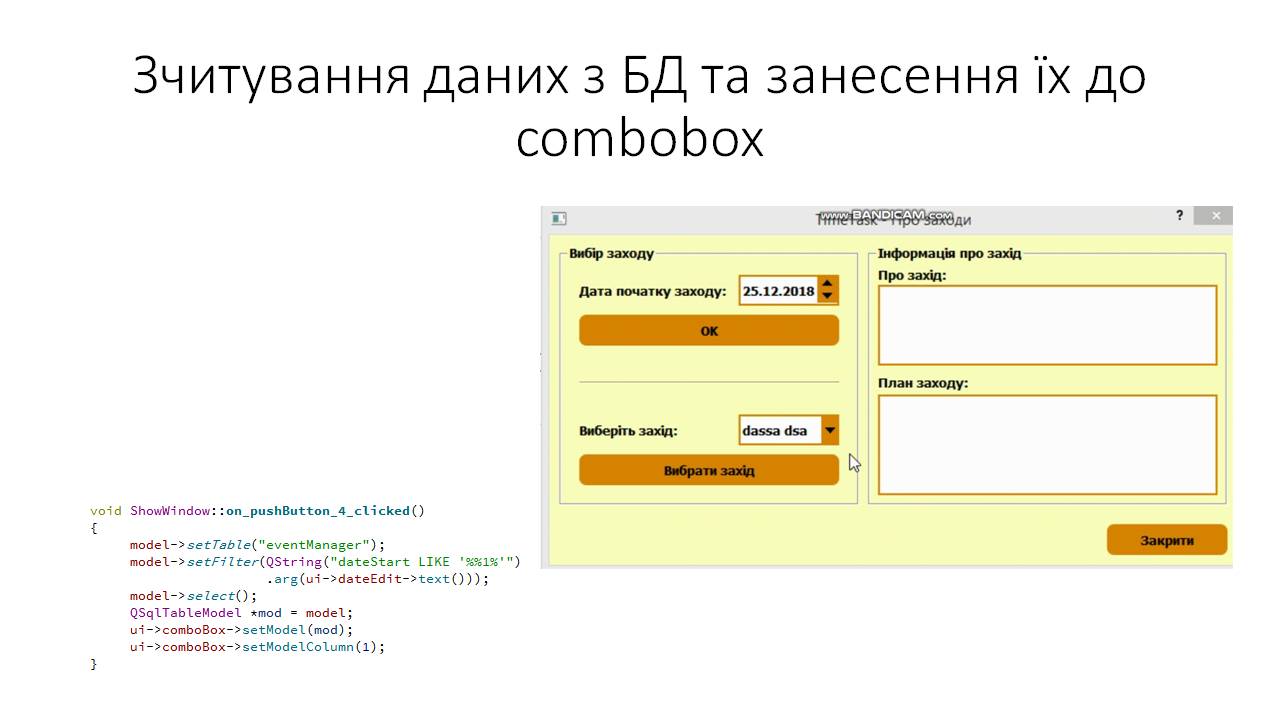
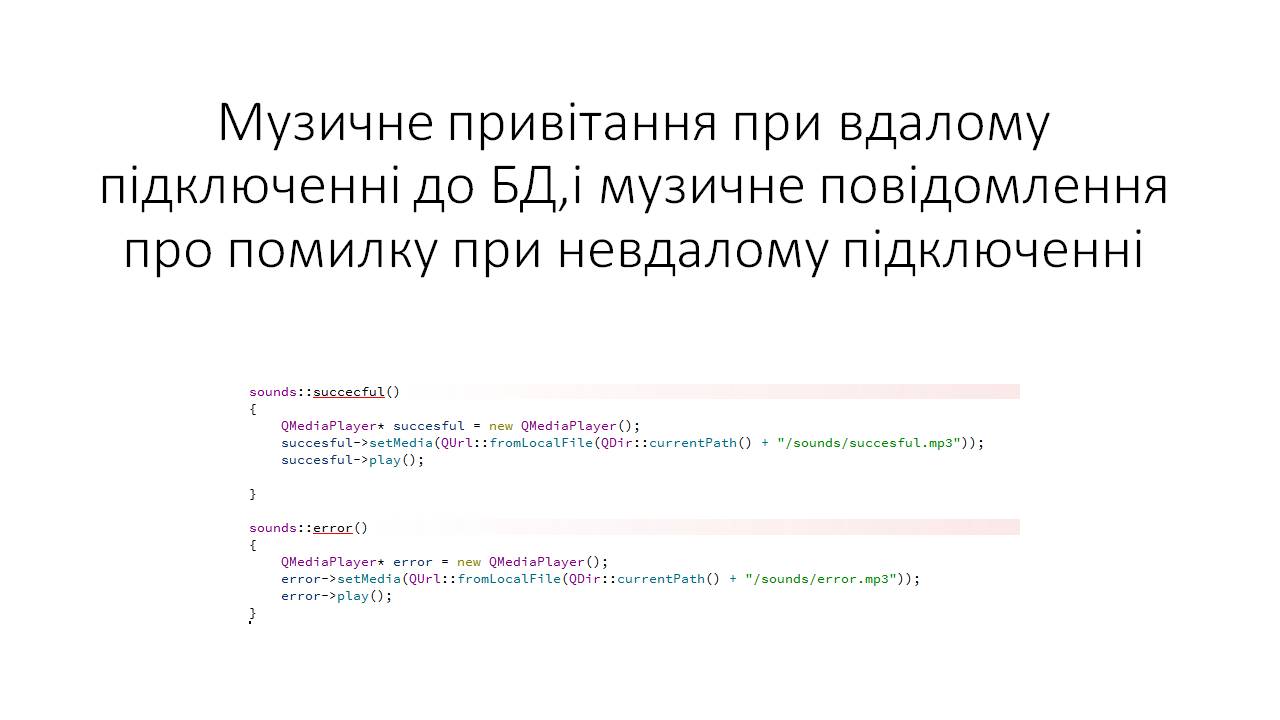
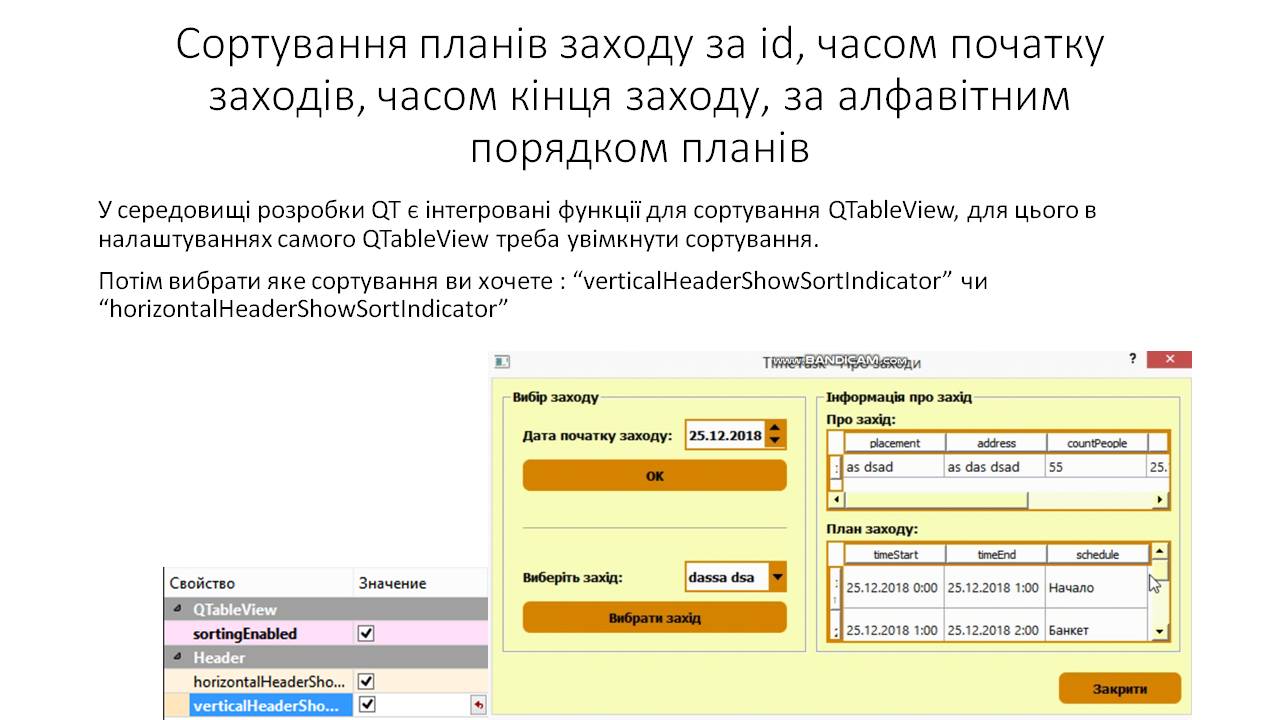
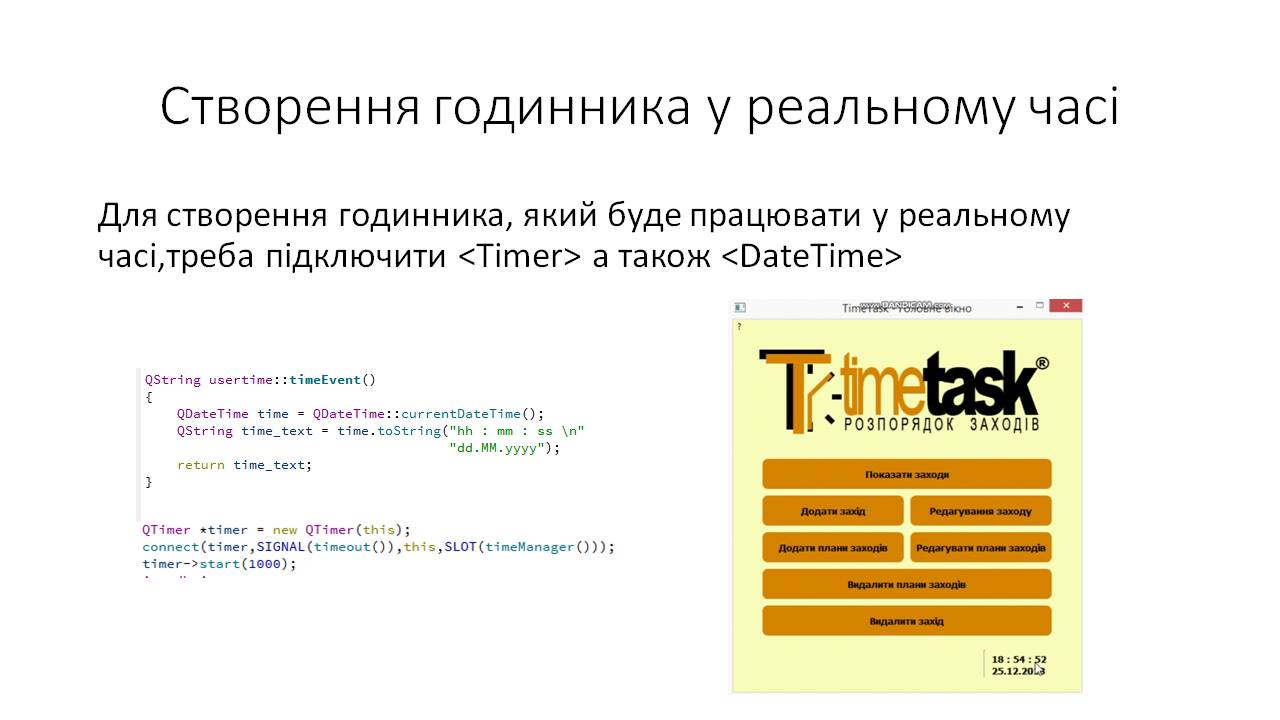
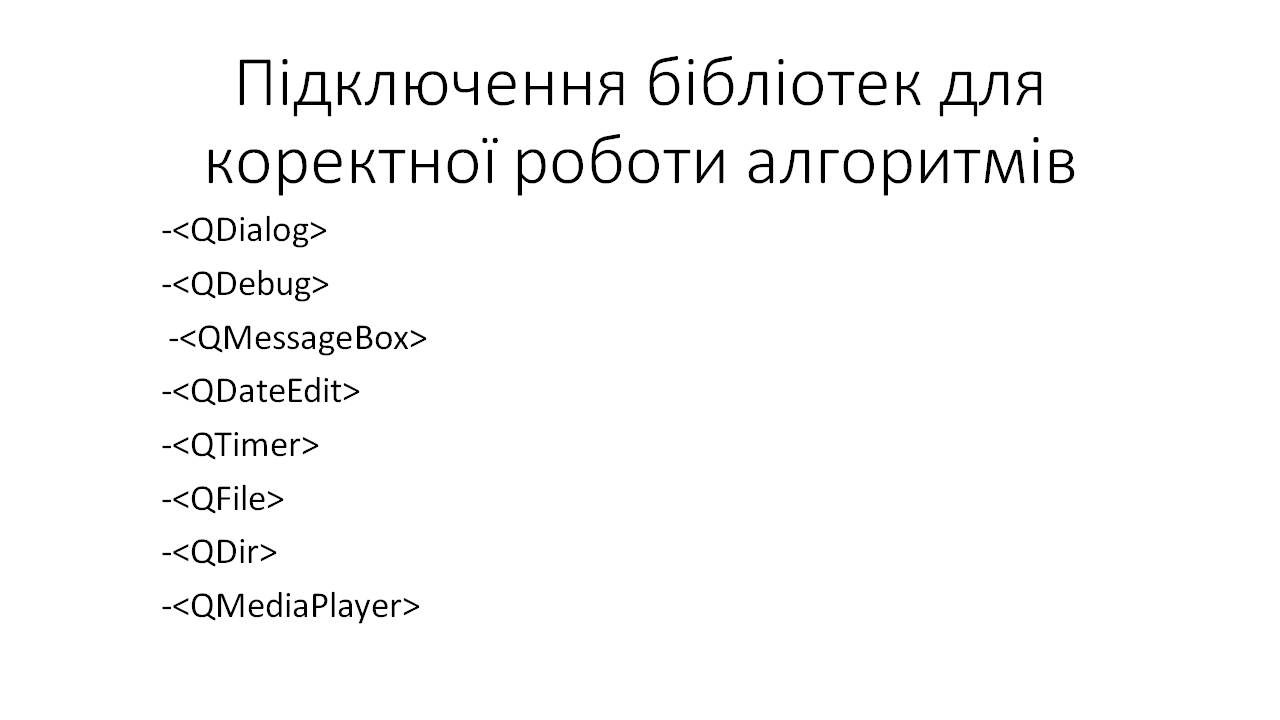
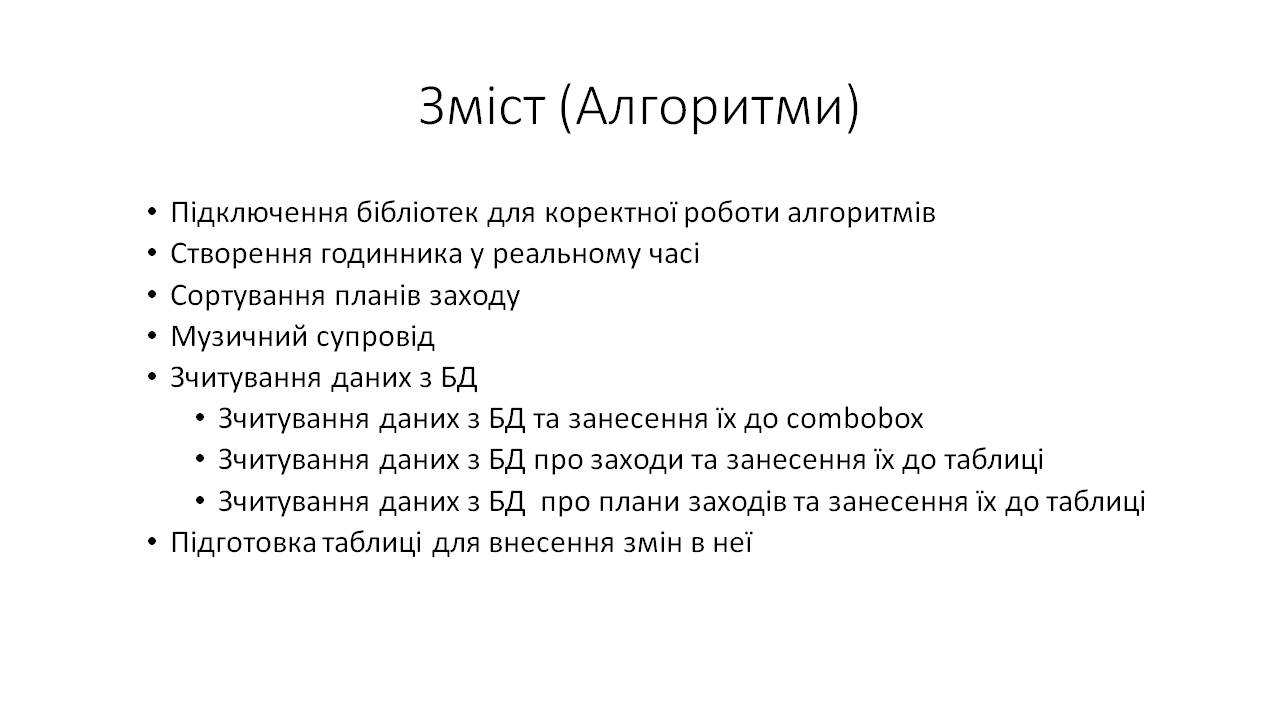
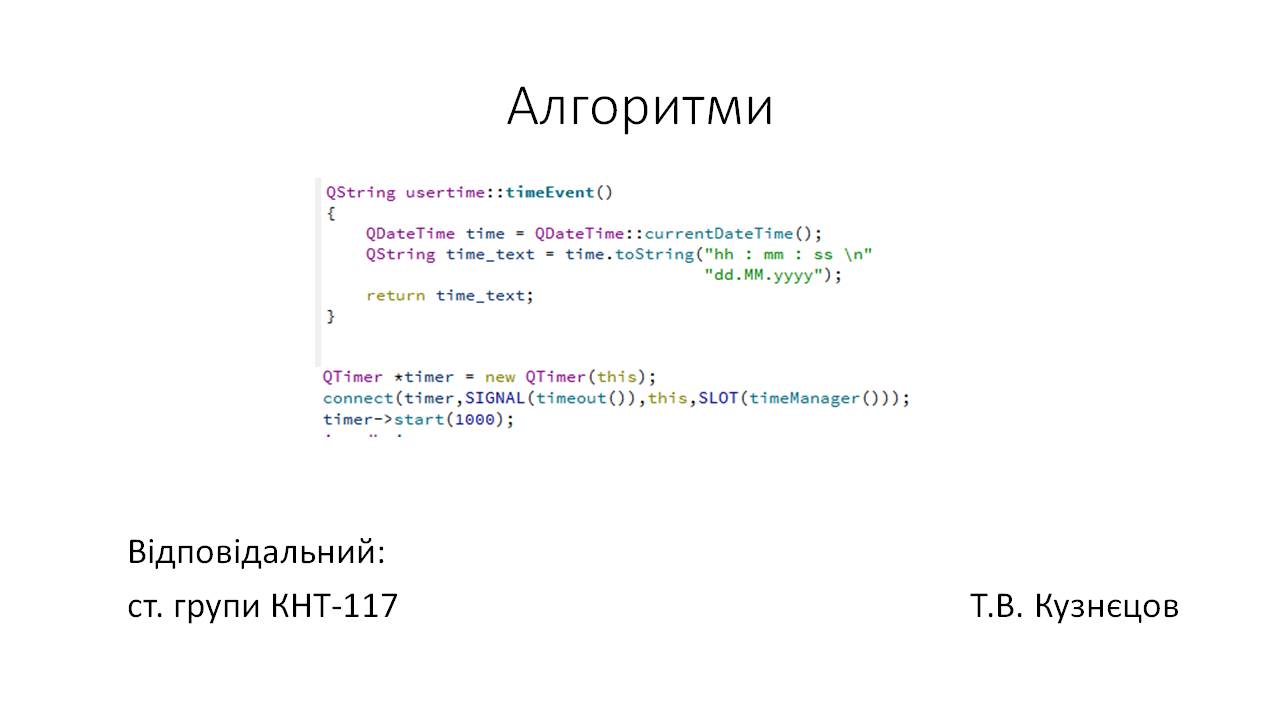
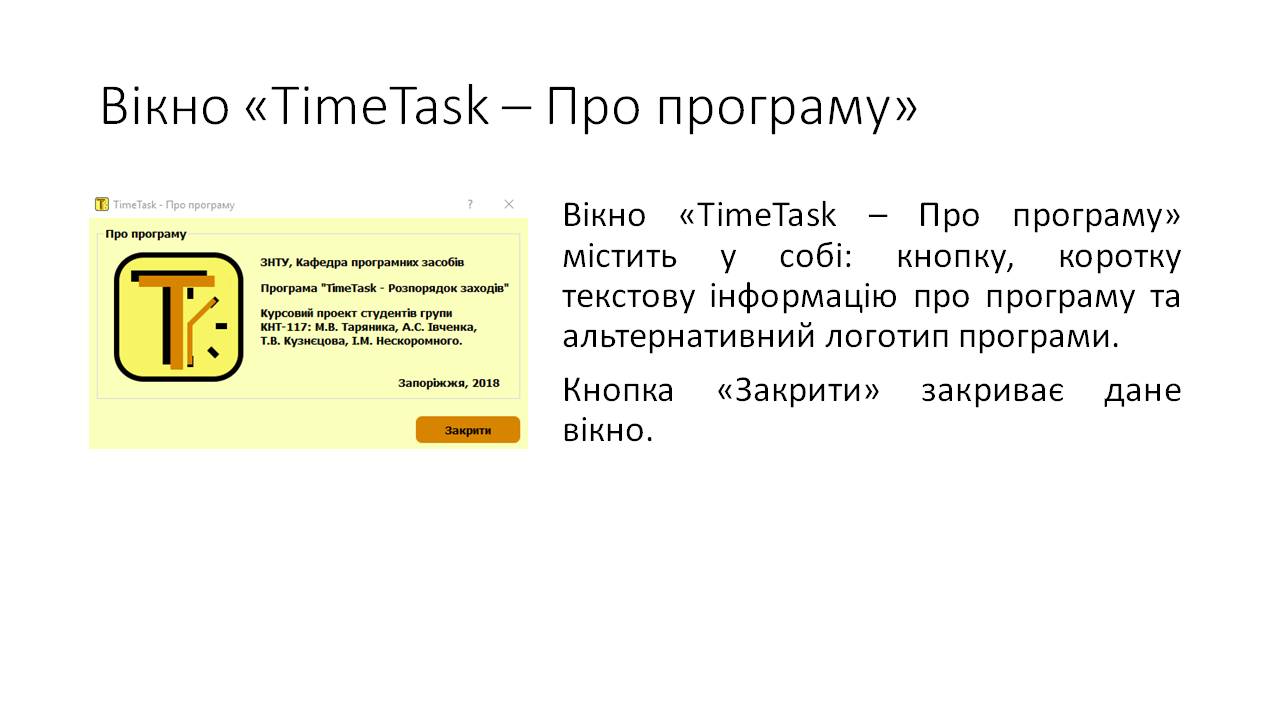
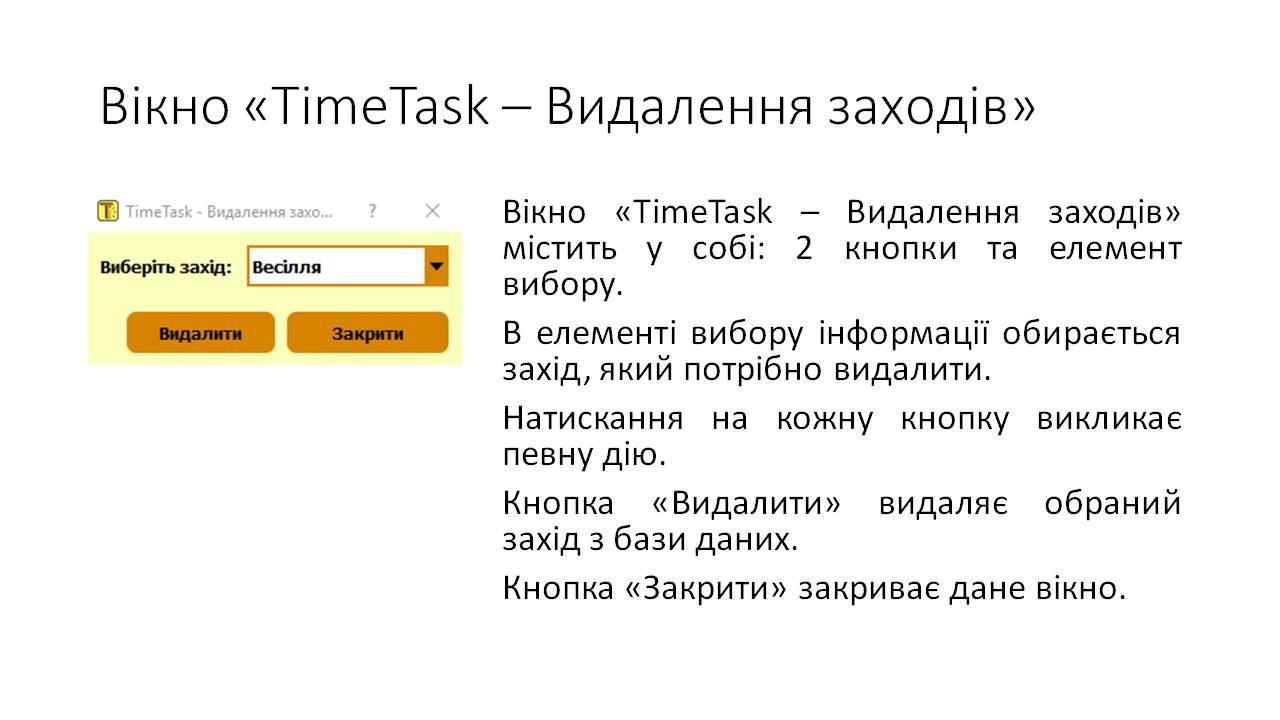
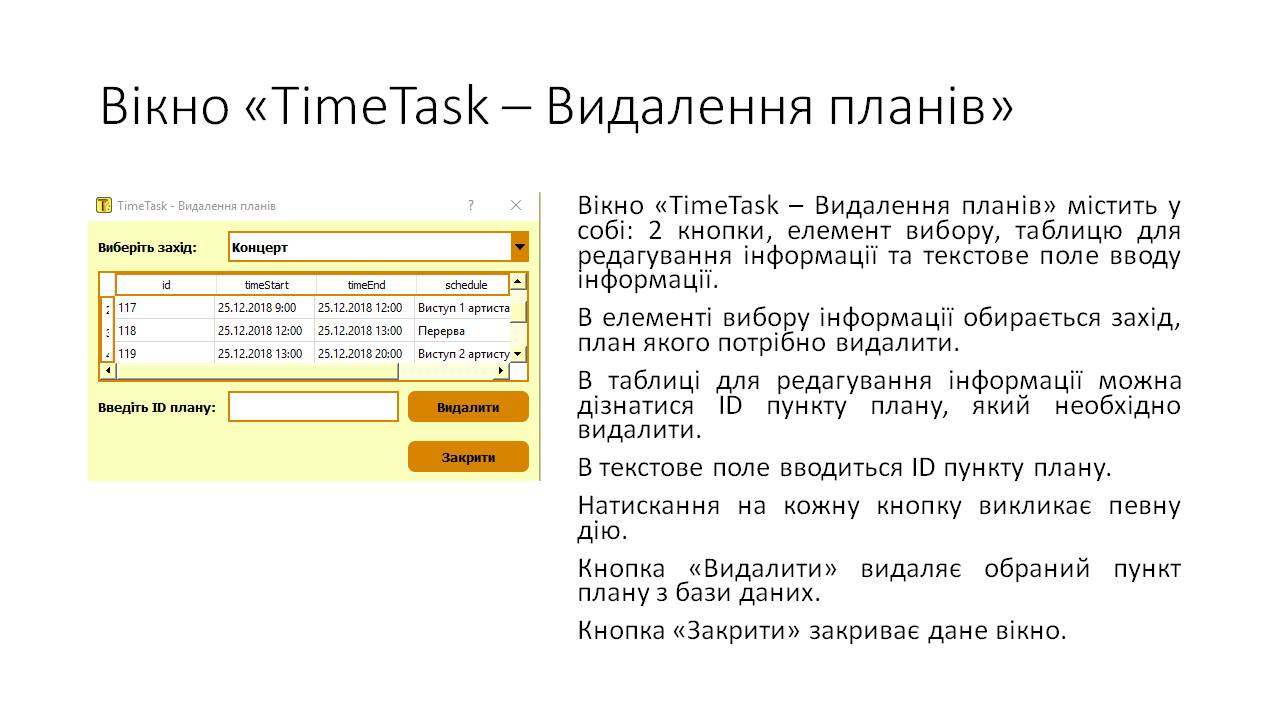
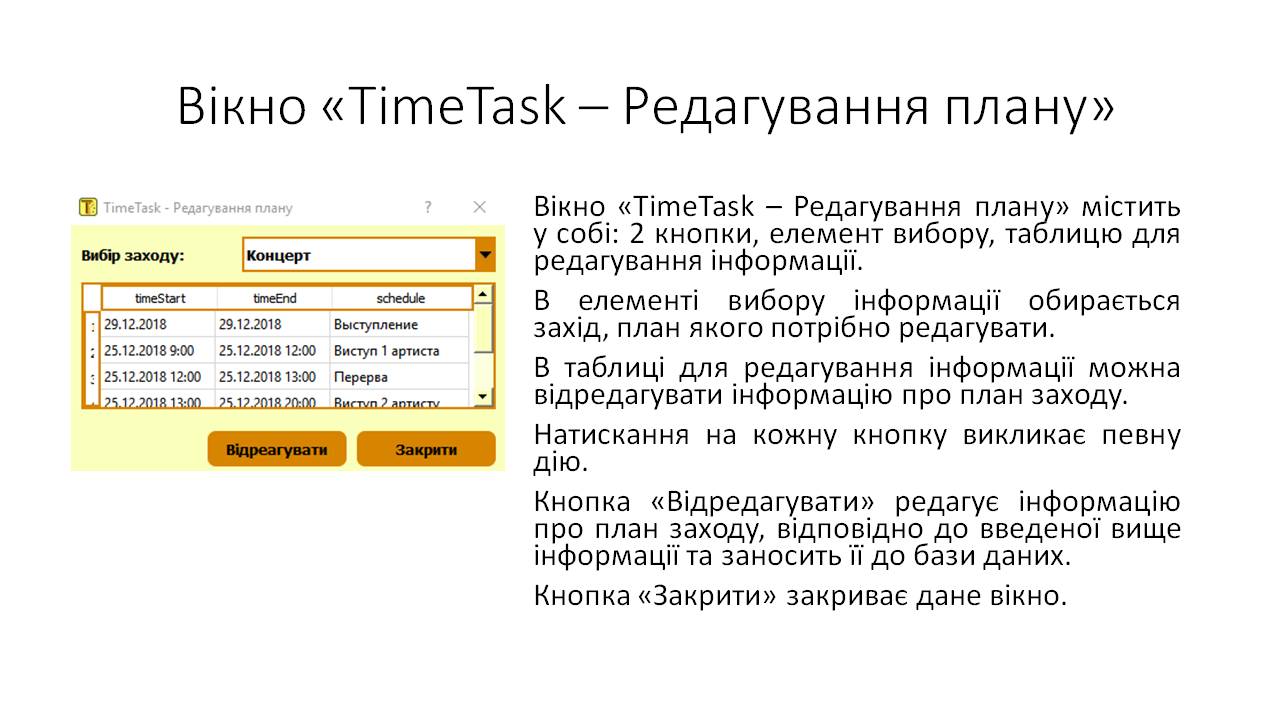
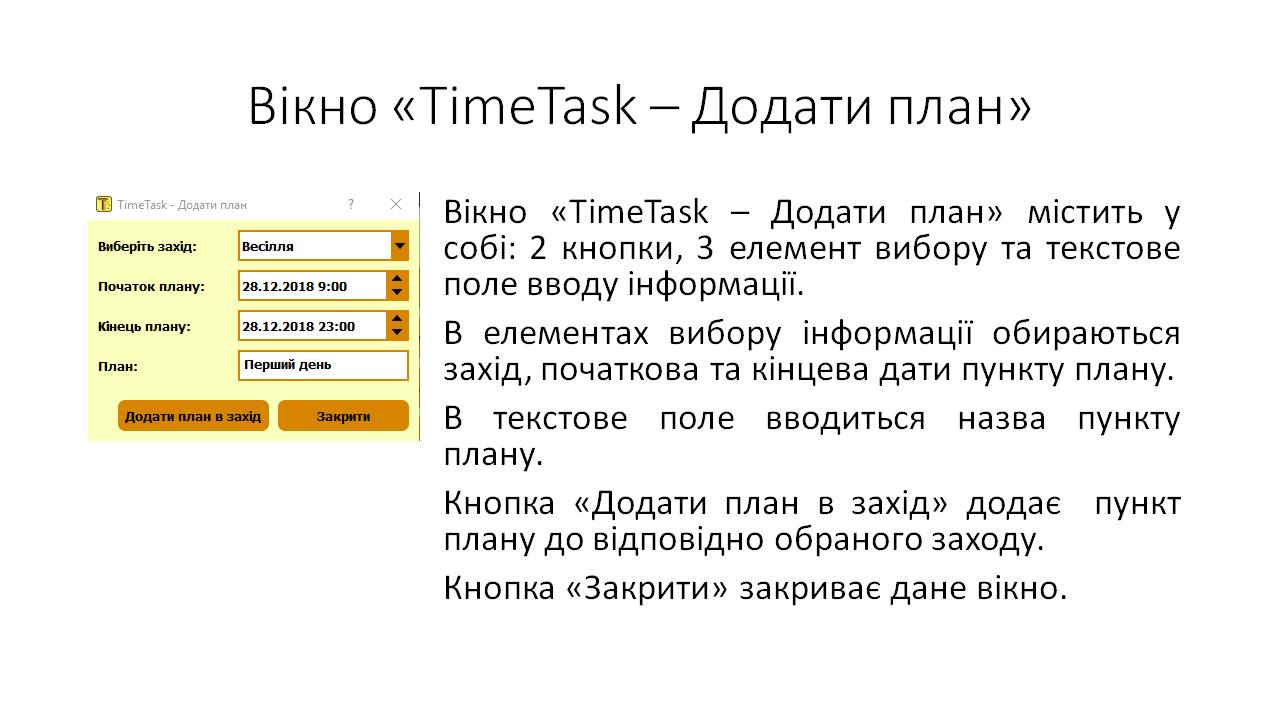
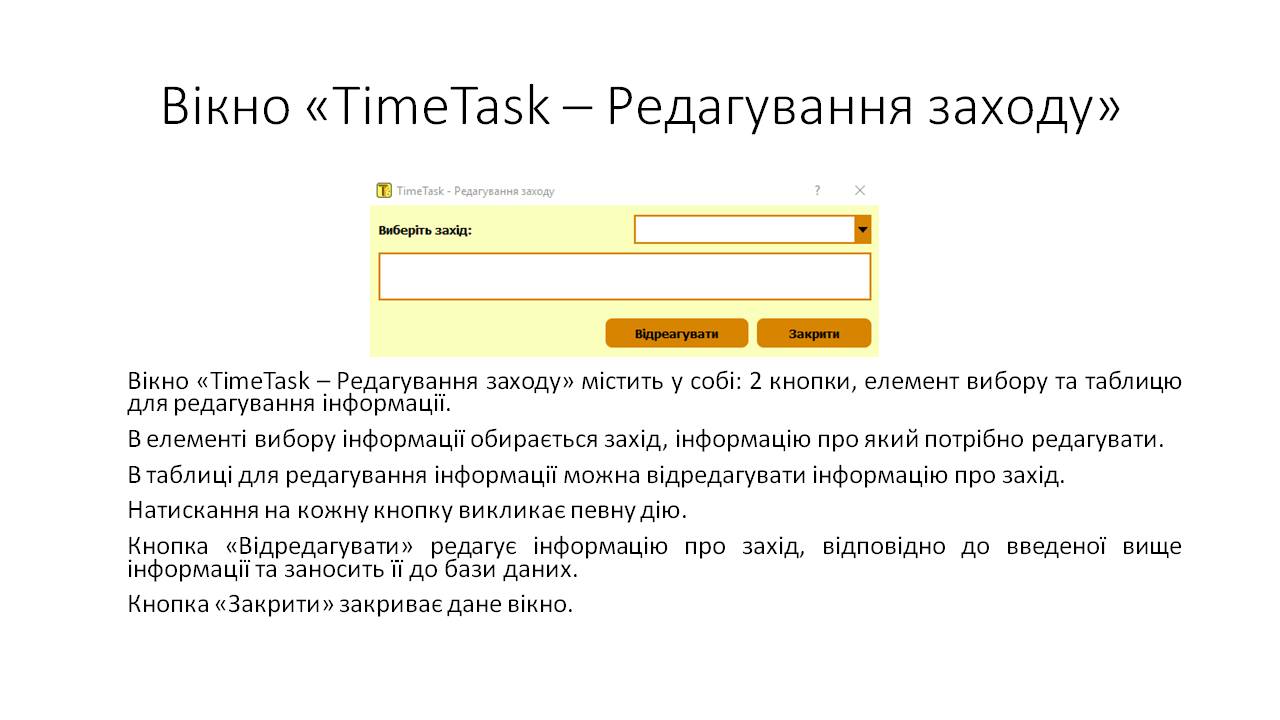
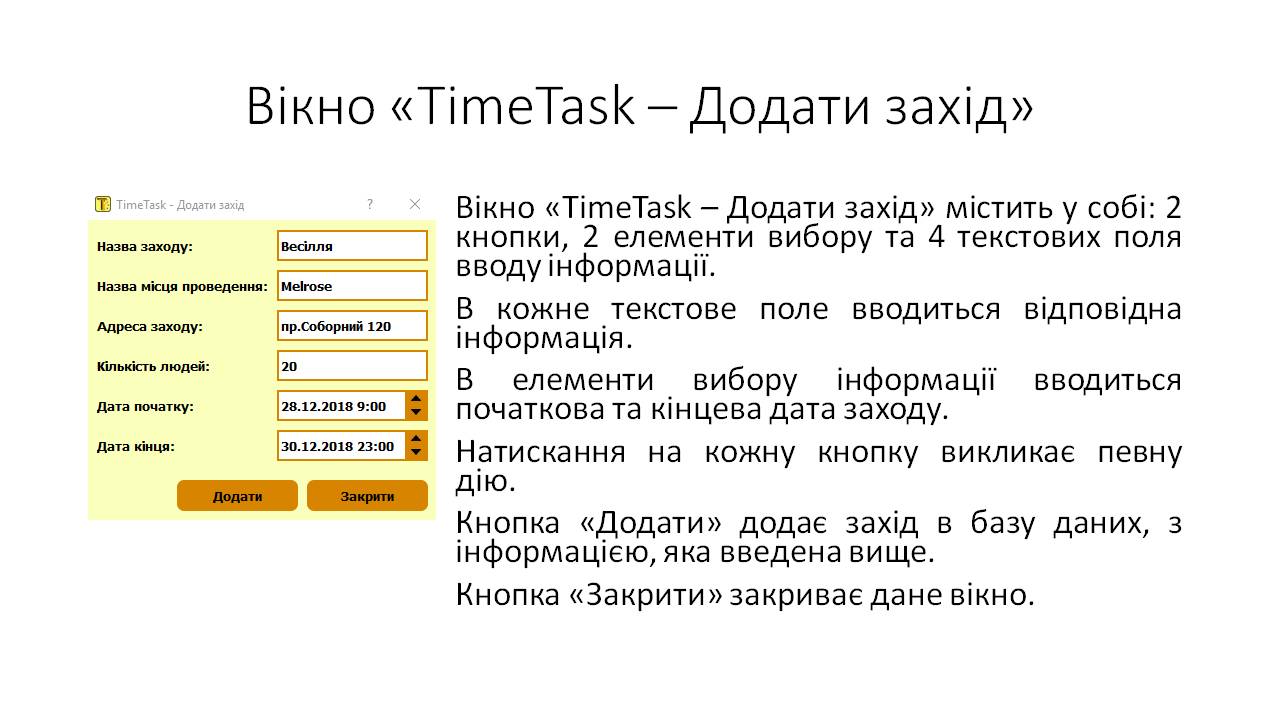
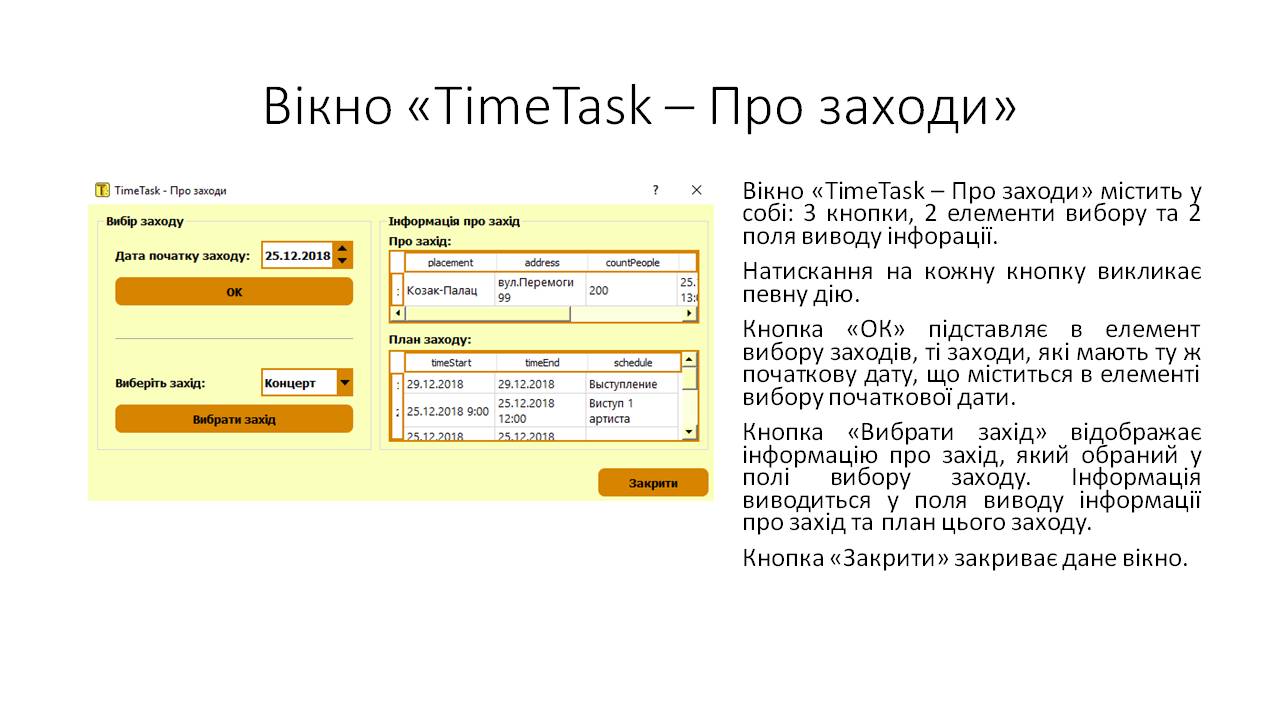
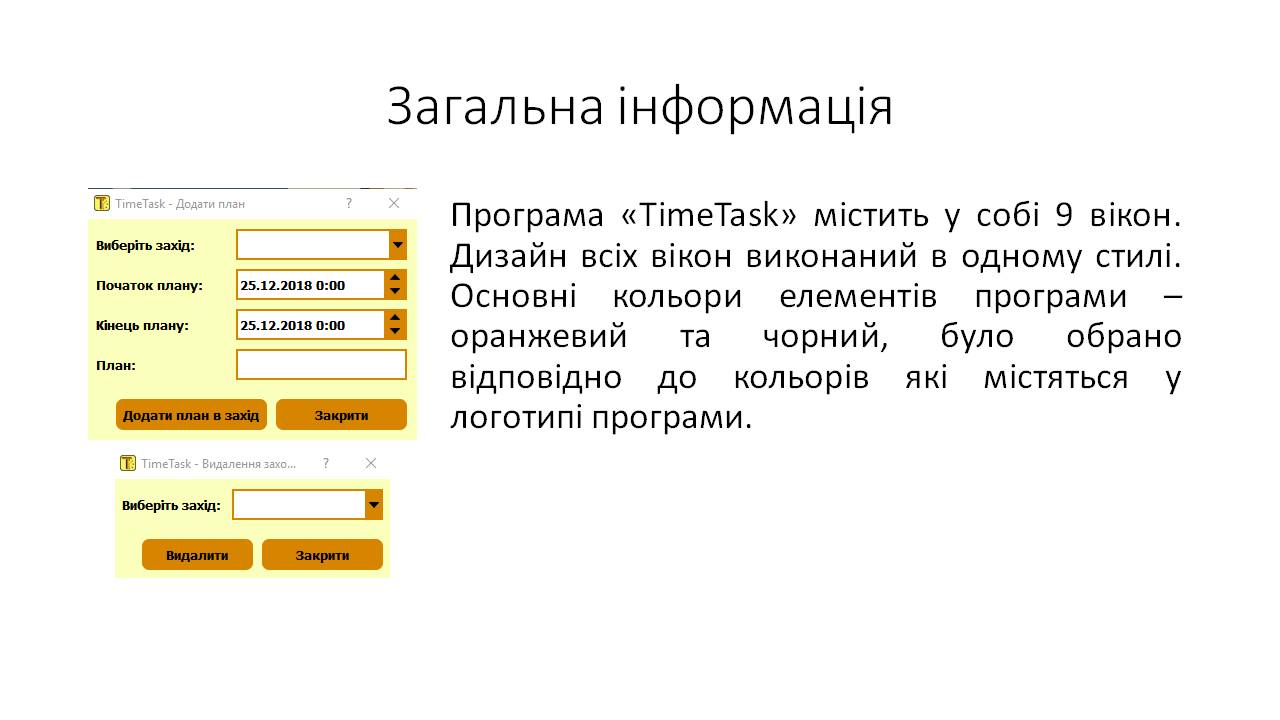
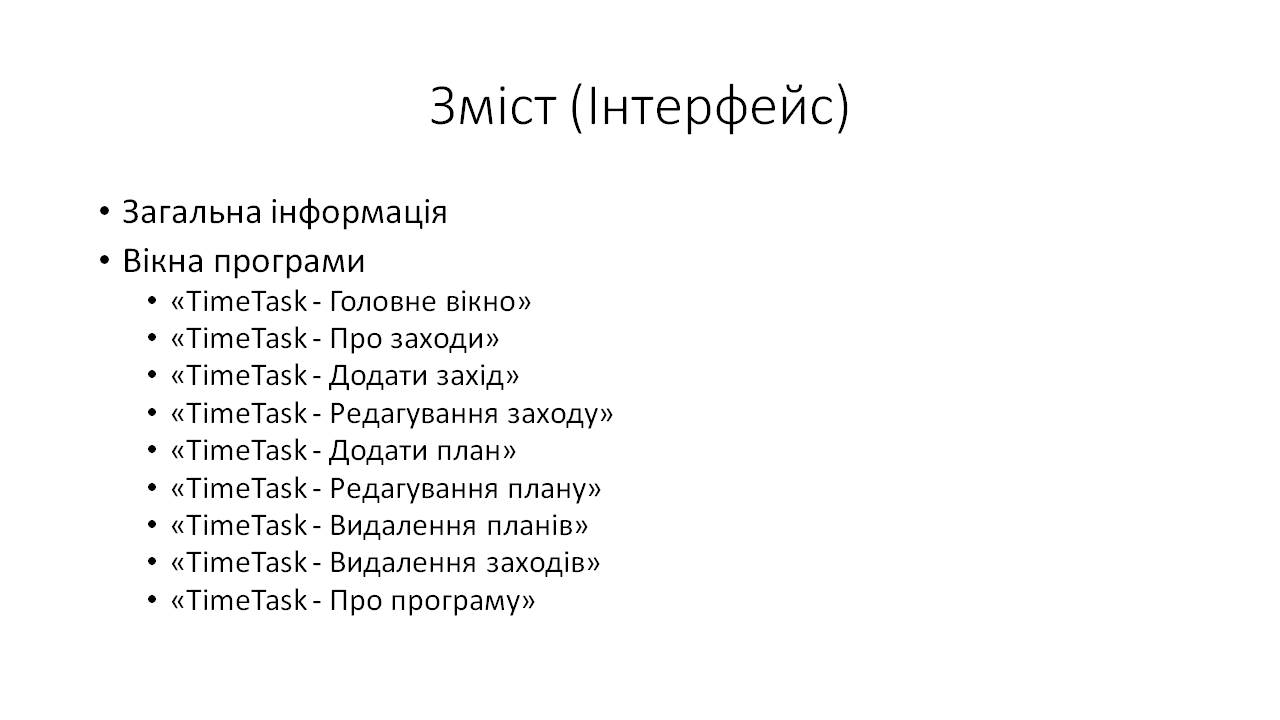
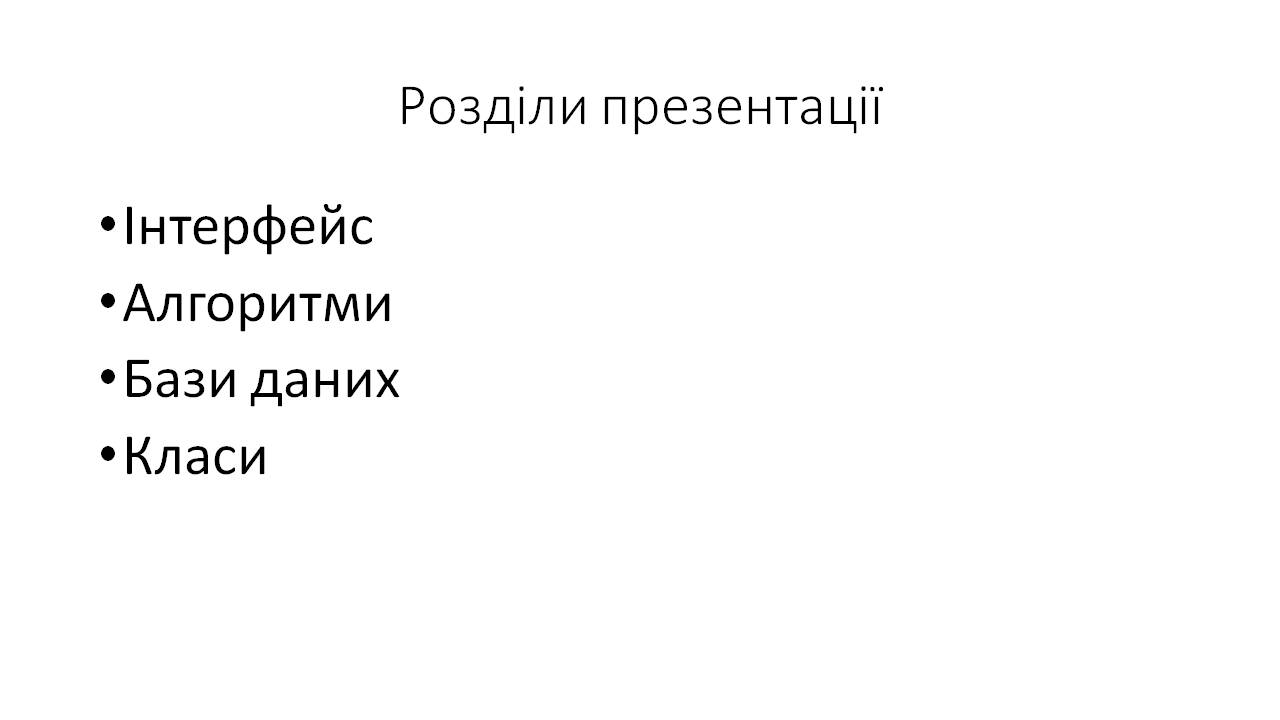
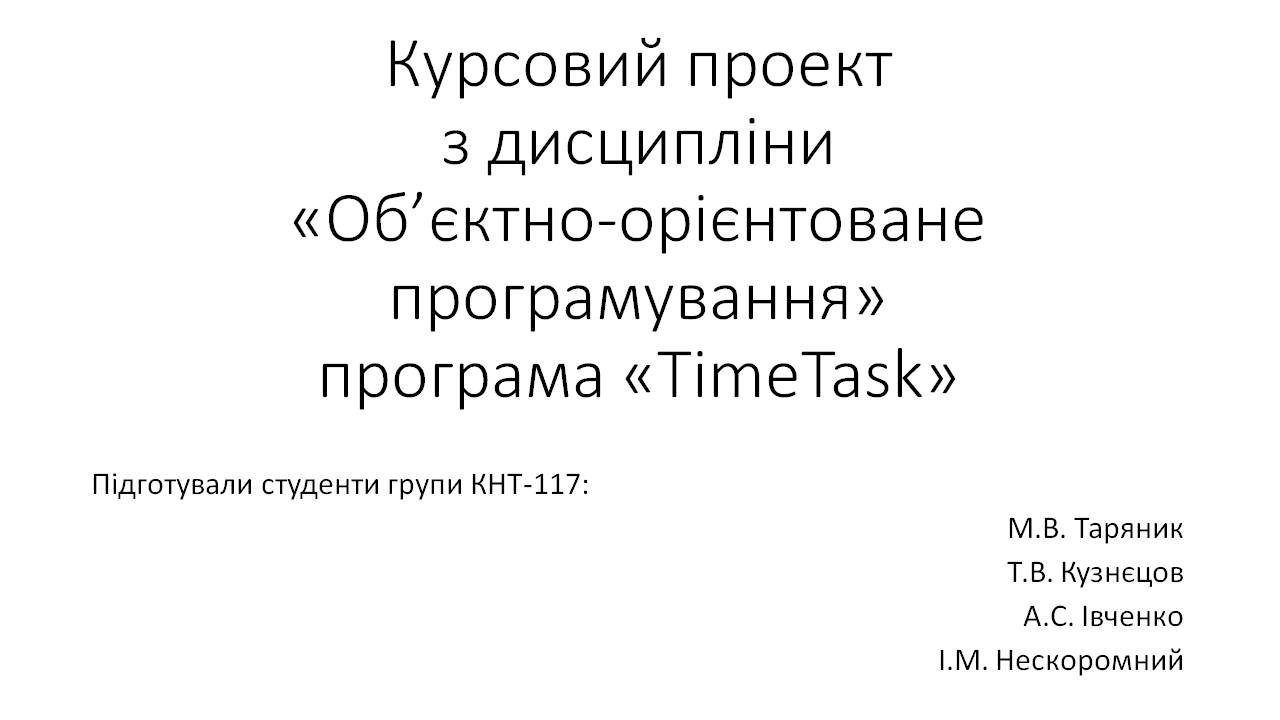
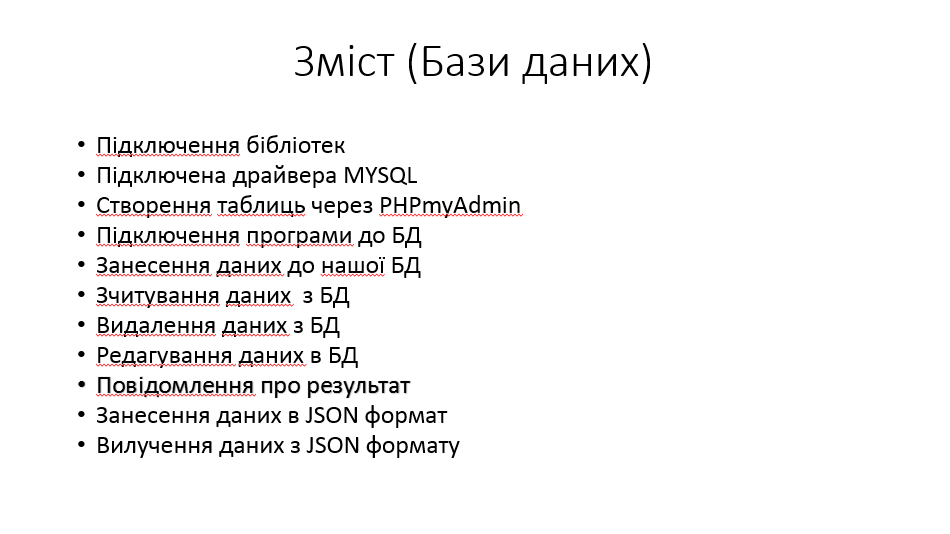
****

Рисунок В.2 – Скріншот слайду №2

Рисунок В.1 – Скріншот слайду №1

Рисунок В.3 – Скріншот слайду №3

Рисунок В.4 – Скріншот слайду №4

Рисунок В.6 – Скріншот слайду №6

Рисунок В.5 – Скріншот слайду №5

Рисунок В.8 – Скріншот слайду №8

Рисунок В.7 – Скріншот слайду №7

Рисунок В.9 – Скріншот слайду №9

Рисунок В.10 – Скріншот слайду №10

Рисунок В.11 – Скріншот слайду №11

Рисунок В.12 – Скріншот слайду №12

Рисунок В.13 – Скріншот слайду №13

Рисунок В.14 – Скріншот слайду №14

Рисунок В.15 – Скріншот слайду №15

Рисунок В.16 – Скріншот слайду №16

Рисунок В.17 – Скріншот слайду №17

Рисунок В.18 – Скріншот слайду №18

Рисунок В.20 – Скріншот слайду №20

Рисунок В.19 – Скріншот слайду №19

Рисунок В.21 – Скріншот слайду №21

Рисунок В.22 – Скріншот слайду №22

Рисунок В.24 – Скріншот слайду №24

Рисунок В.23 – Скріншот слайду №23

Рисунок В.25 – Скріншот слайду №25

Рисунок В.26 – Скріншот слайду №26

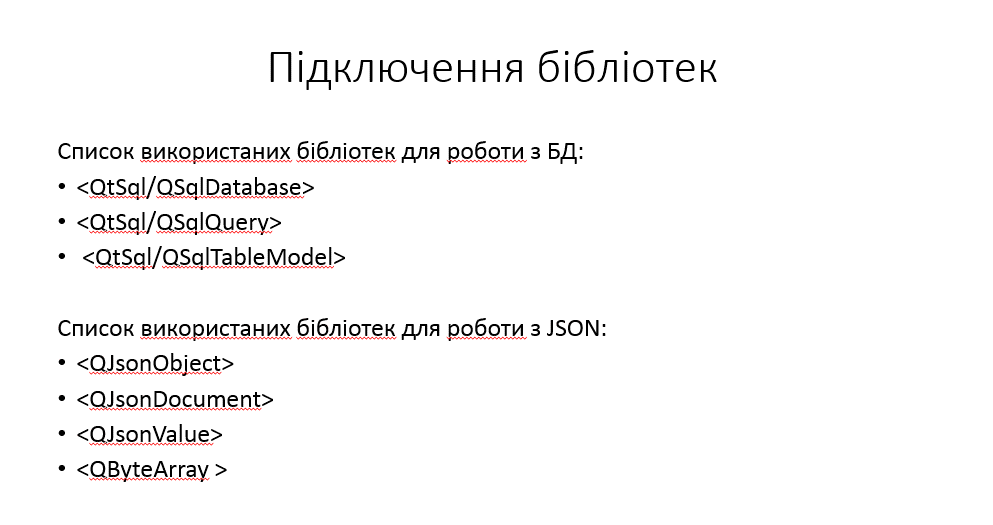


Рисунок В.27 – Скріншот слайду №27

Рисунок В.27 – Скріншот слайду №27

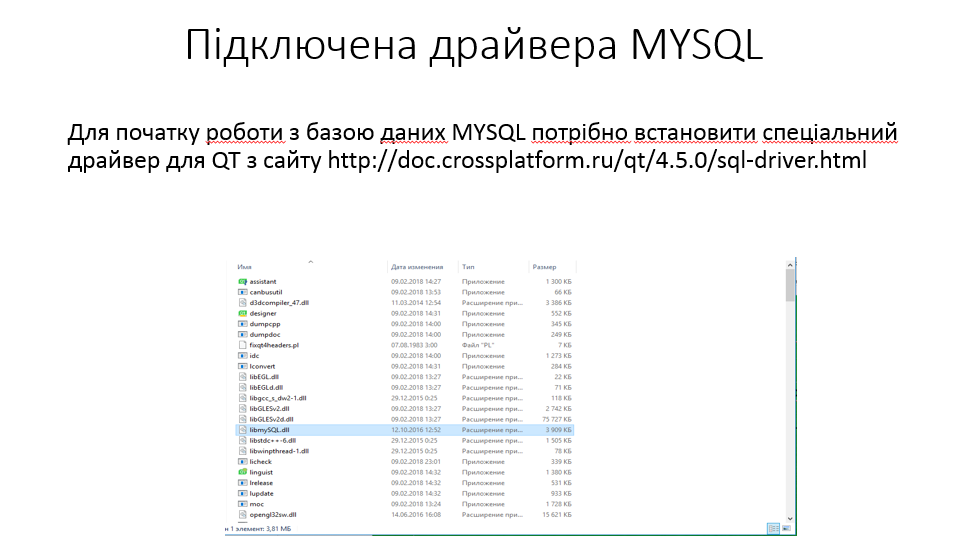


Рисунок В.28 – Скріншот слайду №28

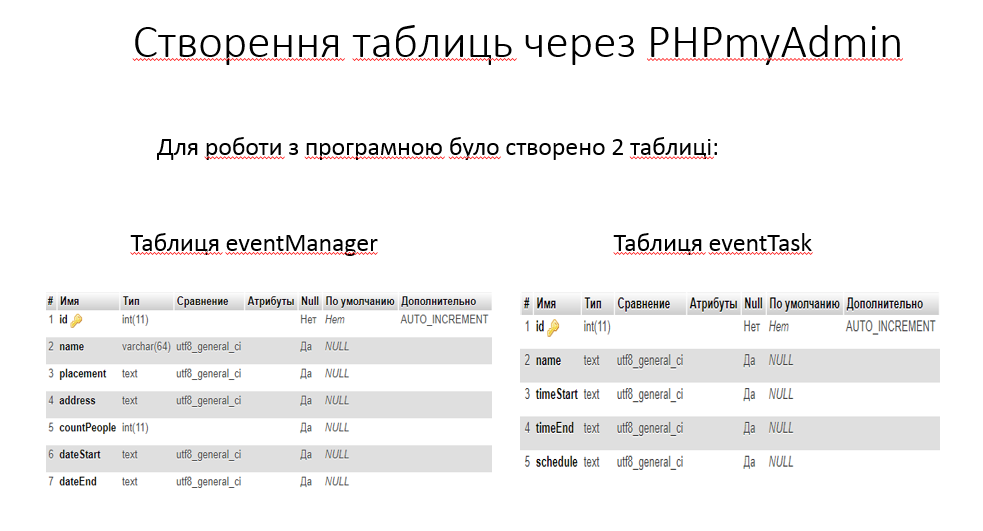
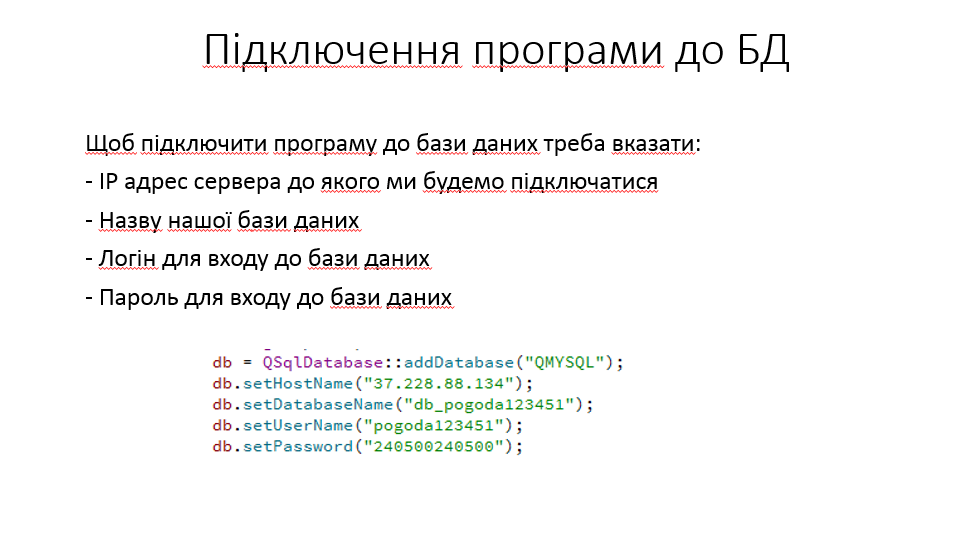


Рисунок В.29 – Скріншот слайду №29

 Рисунок В.30 – Скріншот слайду №30

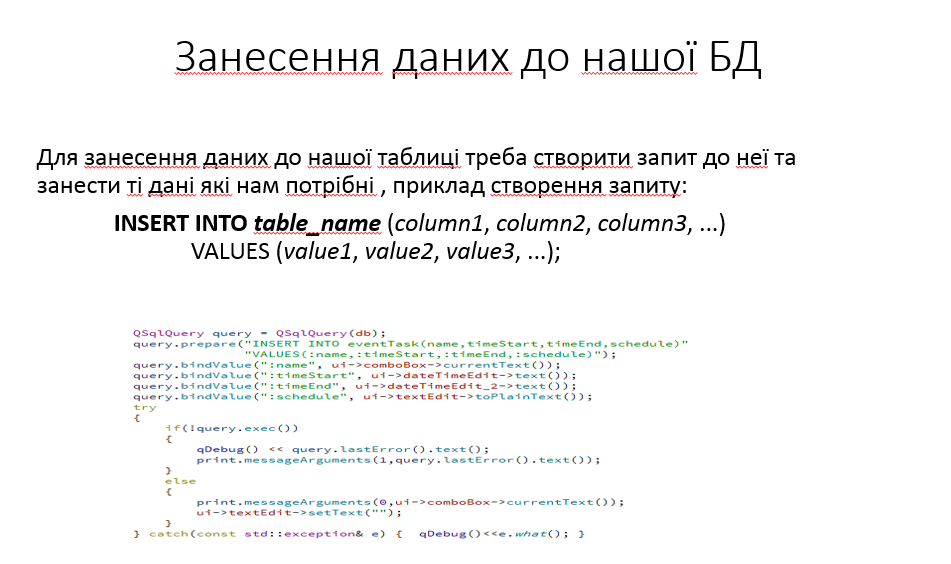


Рисунок В.31 – Скріншот слайду №31

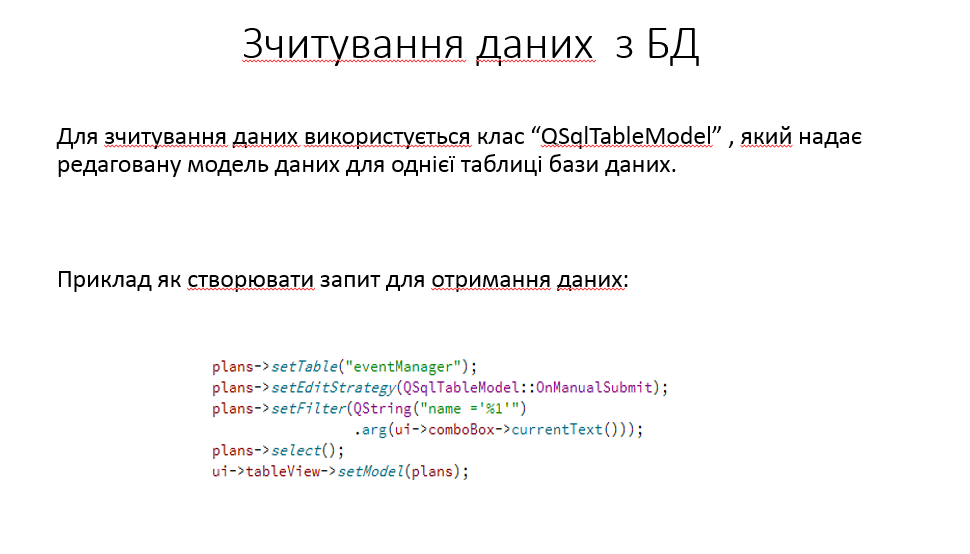


Рисунок В.32 – Скріншот слайду №32

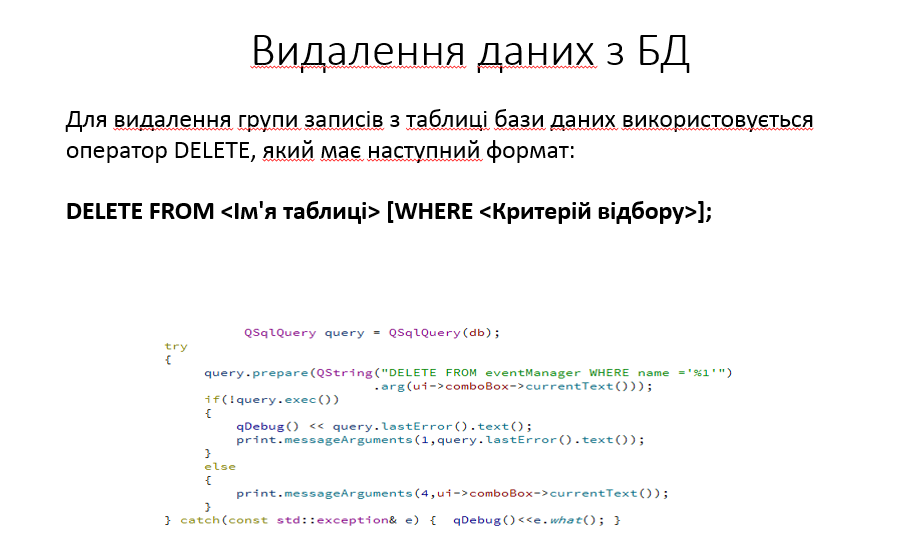


Рисунок В.33 – Скріншот слайду №33

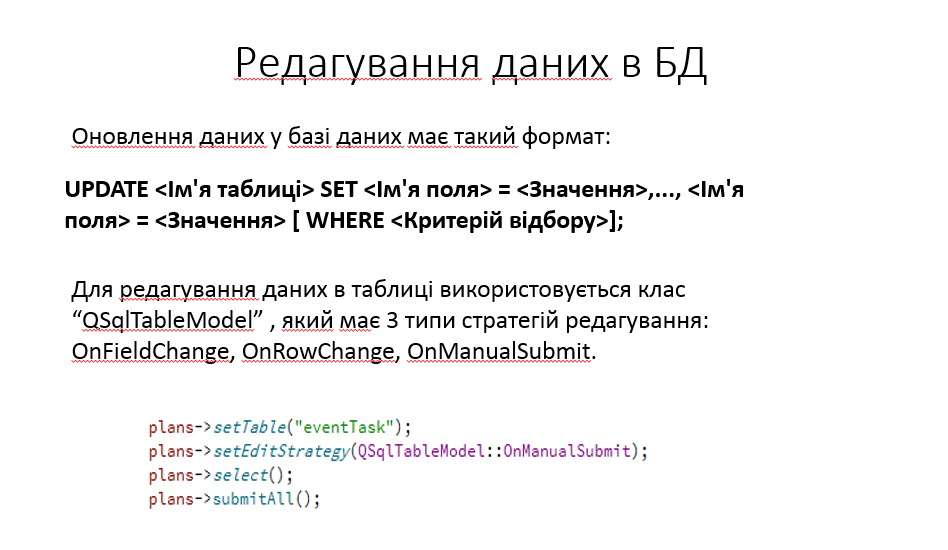


Рисунок В.34 – Скріншот слайду №34

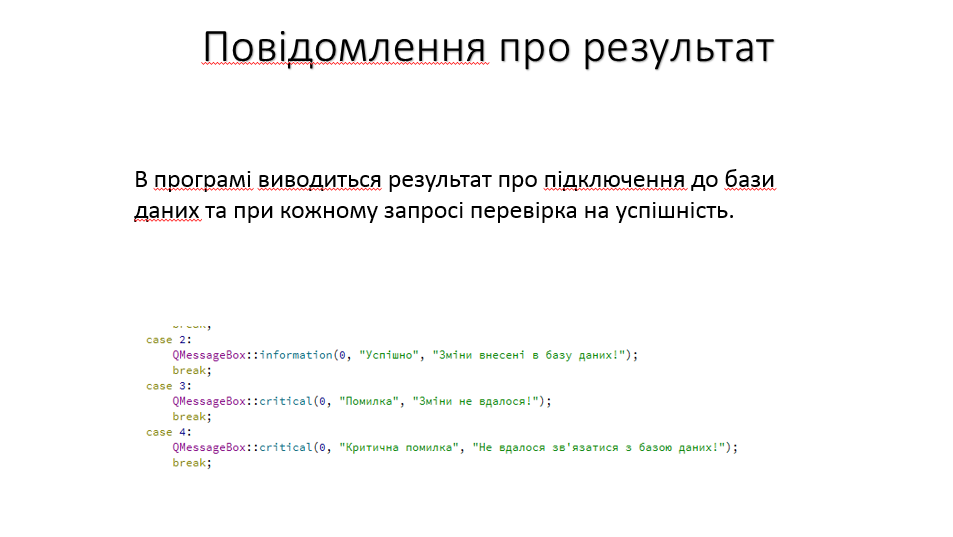


Рисунок В.35 – Скріншот слайду №35

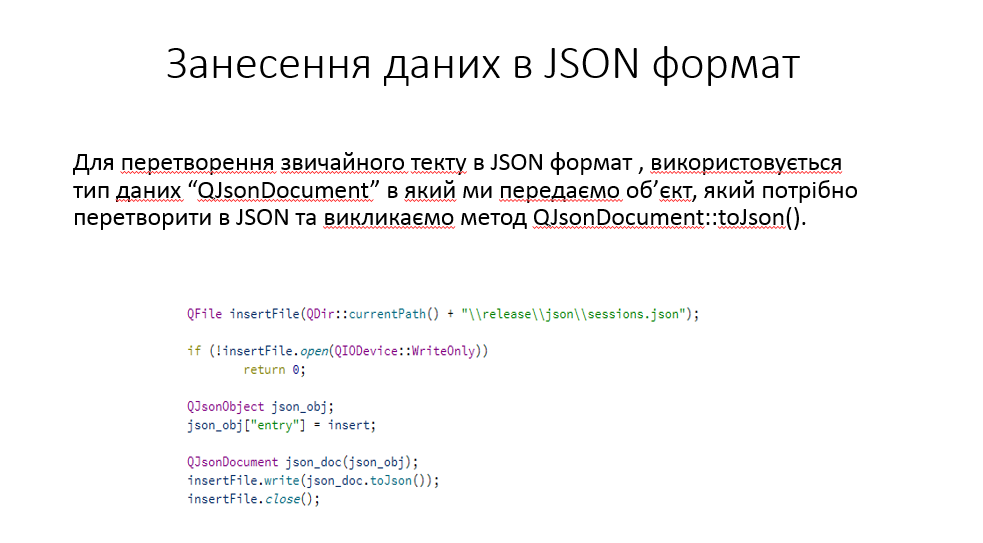


Рисунок В.36 – Скріншот слайду №36

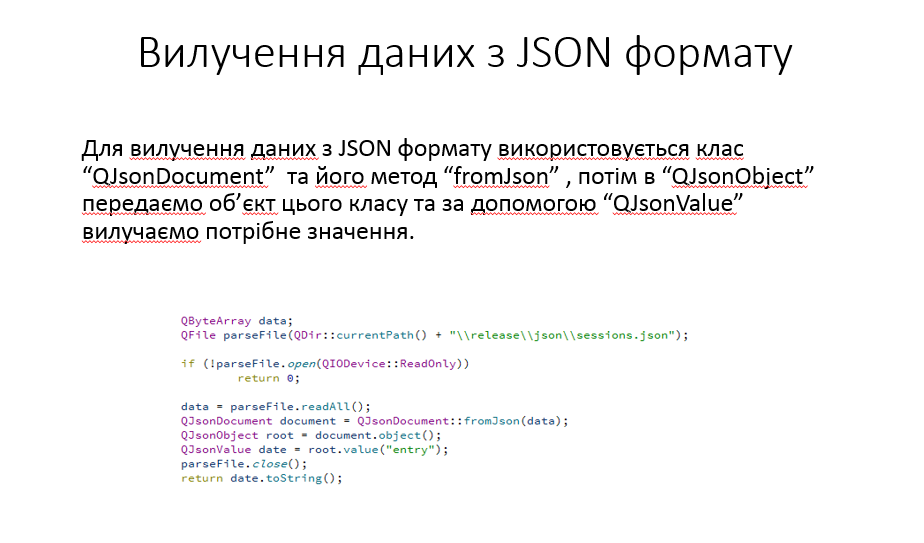


Рисунок В.37 – Скріншот слайду №37



Рисунок В.38 – Скріншот слайду №38

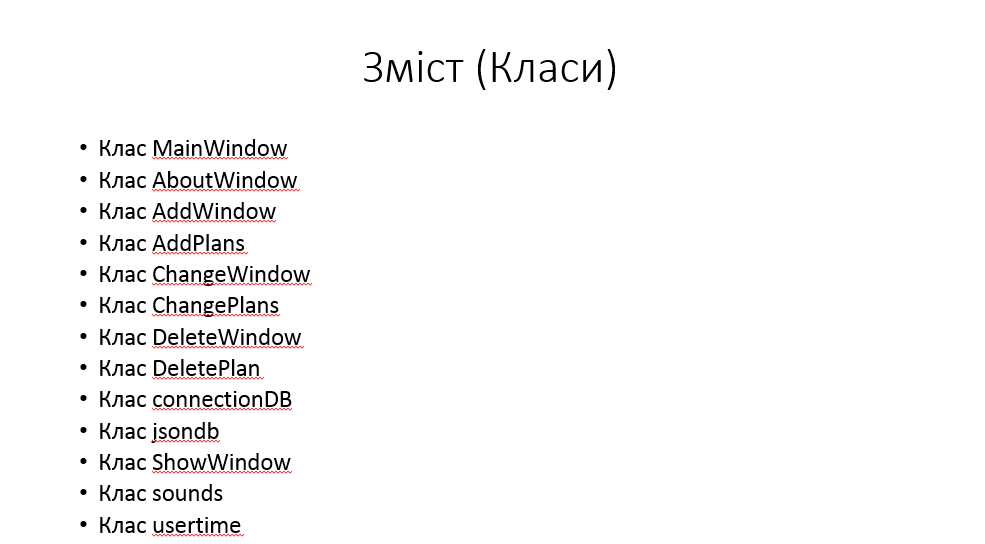


Рисунок В.39 – Скріншот слайду №39

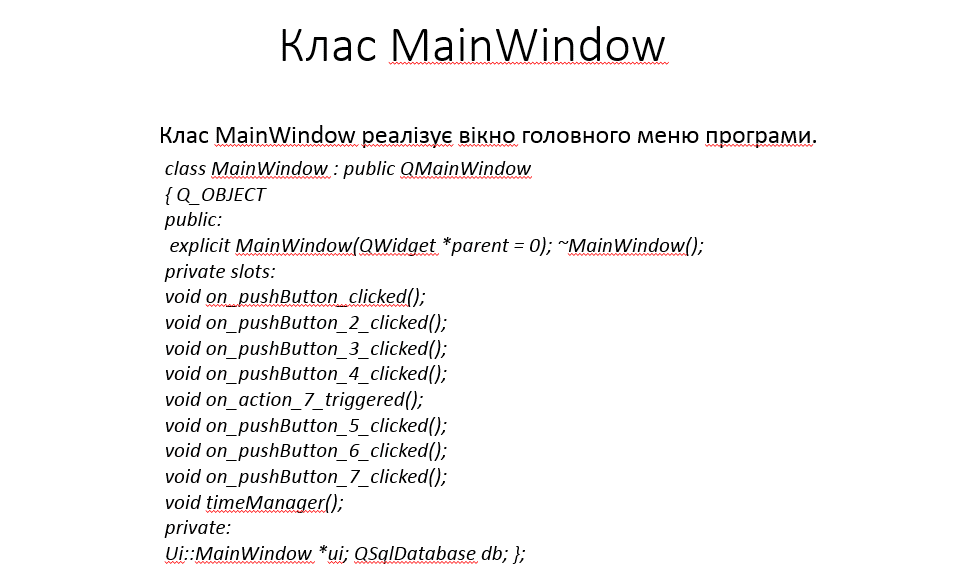


Рисунок В.40 – Скріншот слайду №40

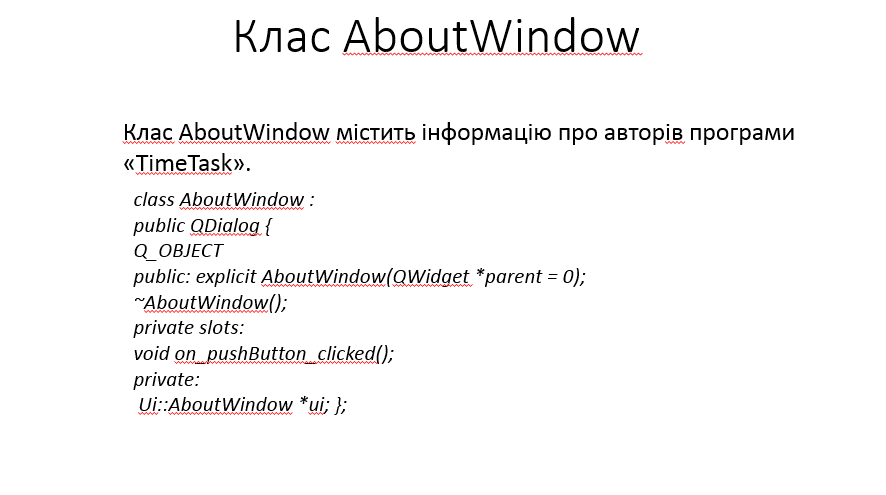


Рисунок В.41 – Скріншот слайду №41

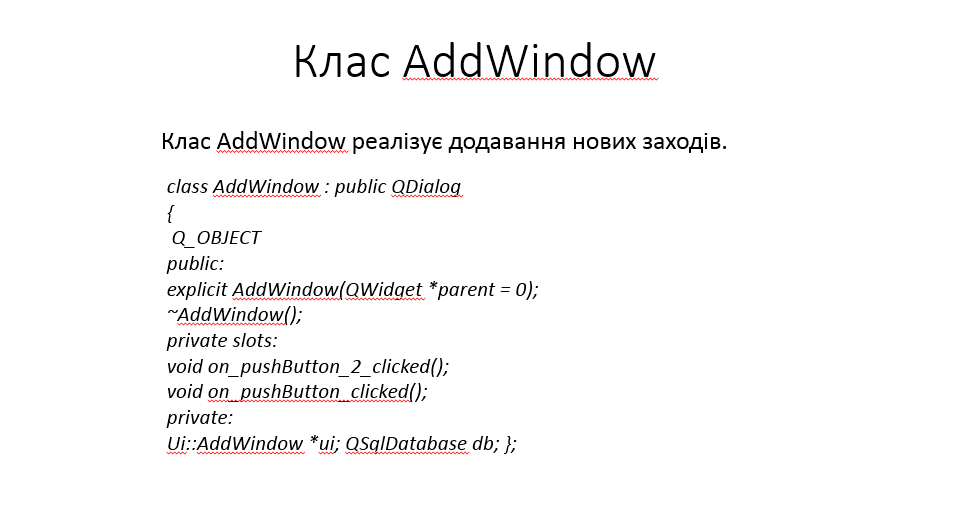


Рисунок В.42 – Скріншот слайду №42

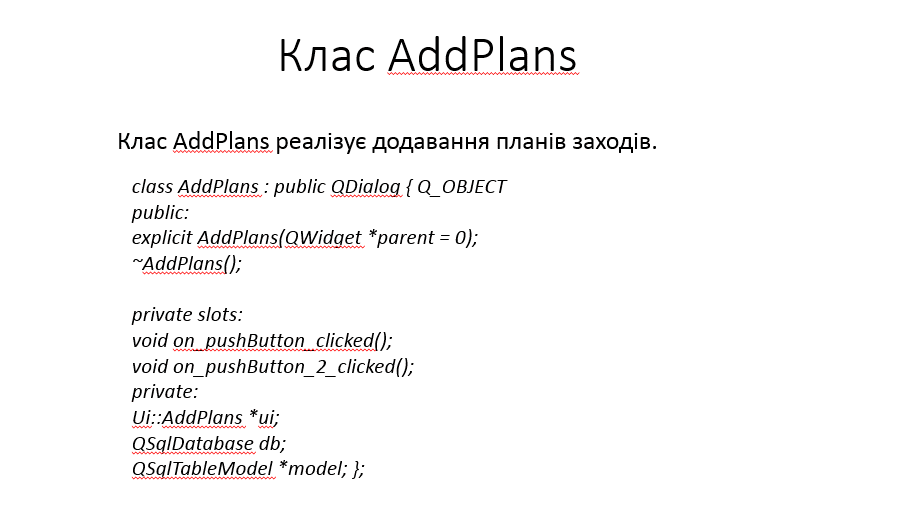


Рисунок В.43– Скріншот слайду №43

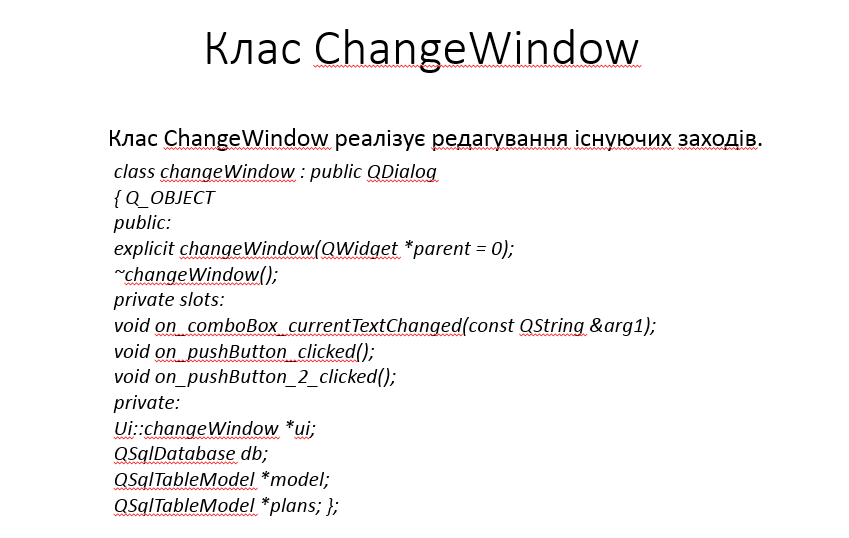


Рисунок В.44– Скріншот слайду №44

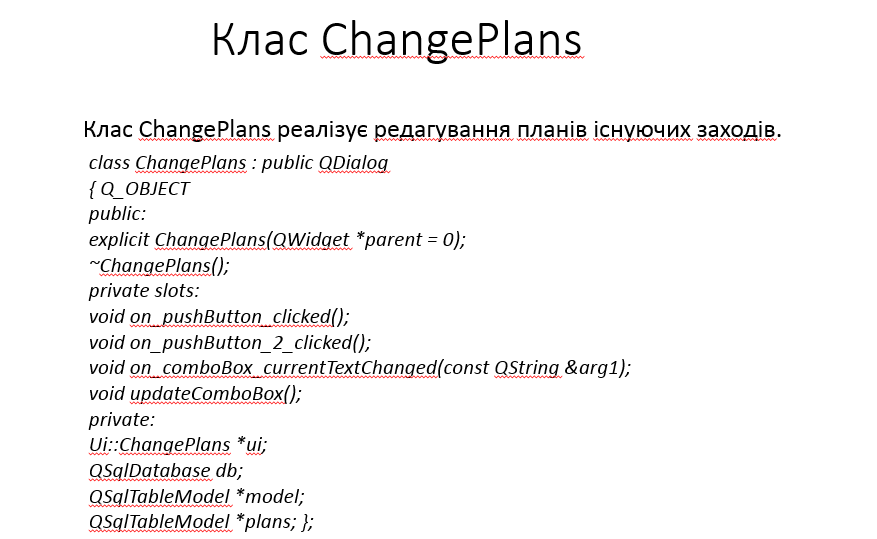


Рисунок В.45– Скріншот слайду №45

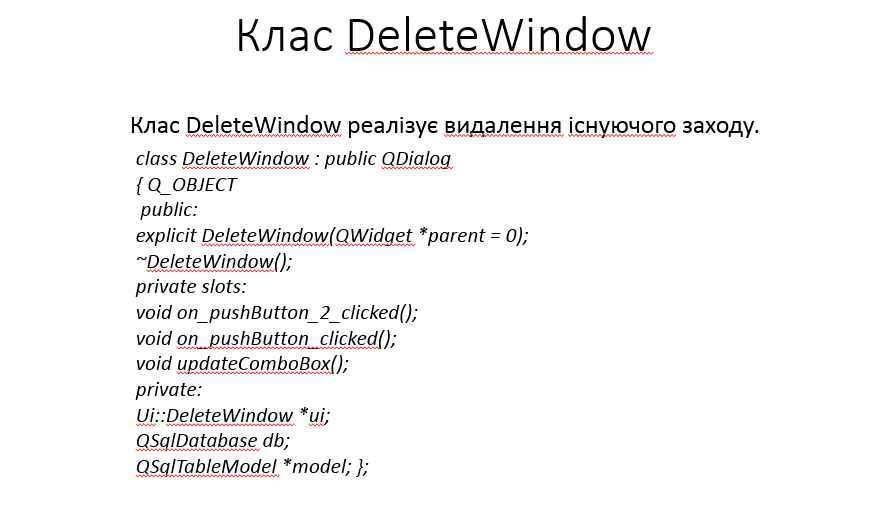


Рисунок В.46– Скріншот слайду №46

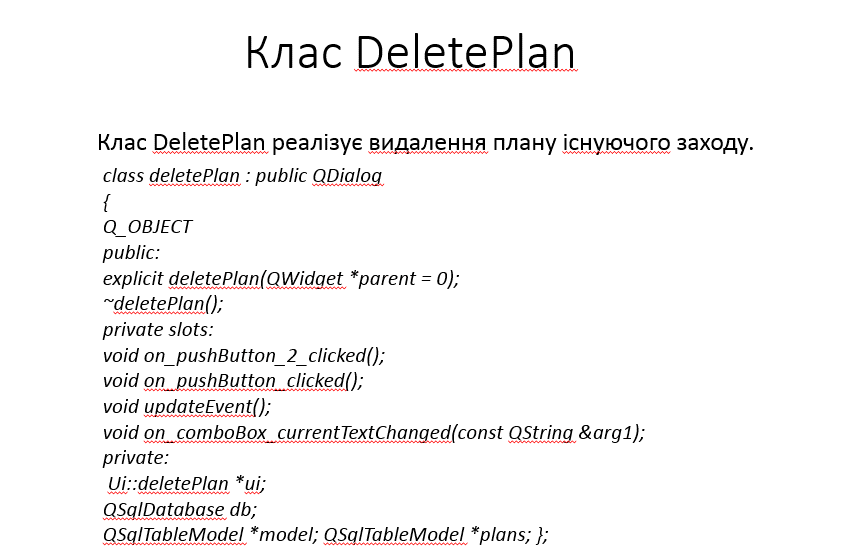


Рисунок В.47– Скріншот слайду №47

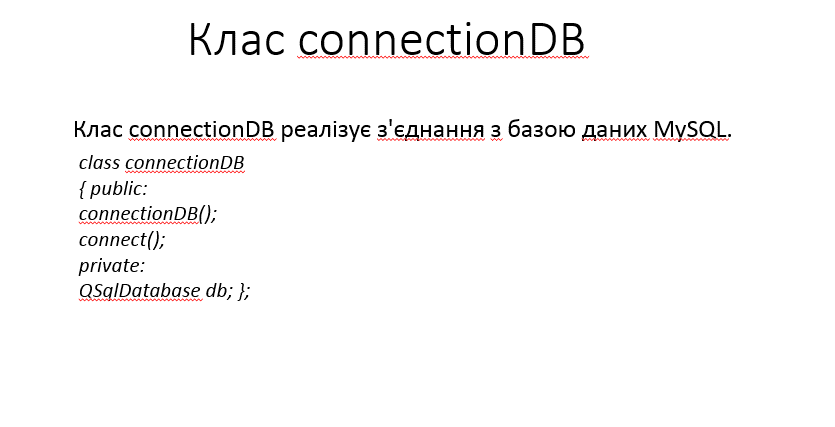


Рисунок В.48– Скріншот слайду №48

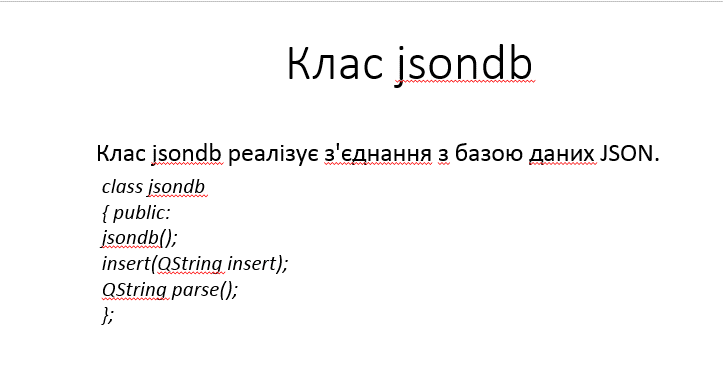


Рисунок В.49– Скріншот слайду №49

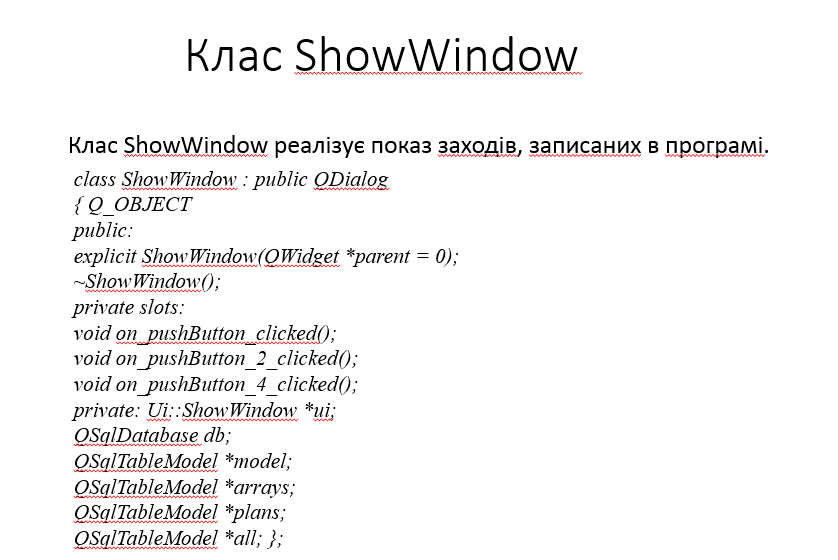


Рисунок В.50– Скріншот слайду №50

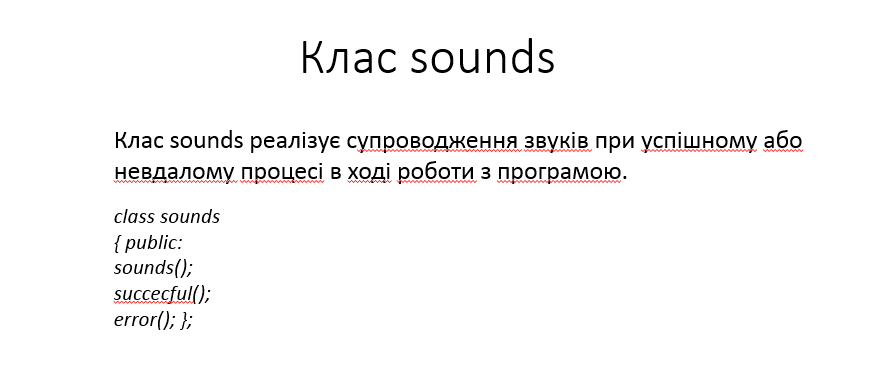


Рисунок В.51– Скріншот слайду №51

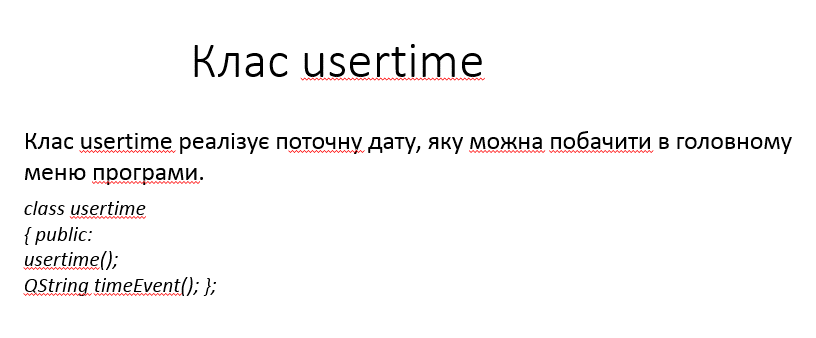


Рисунок В.52– Скріншот слайду №52

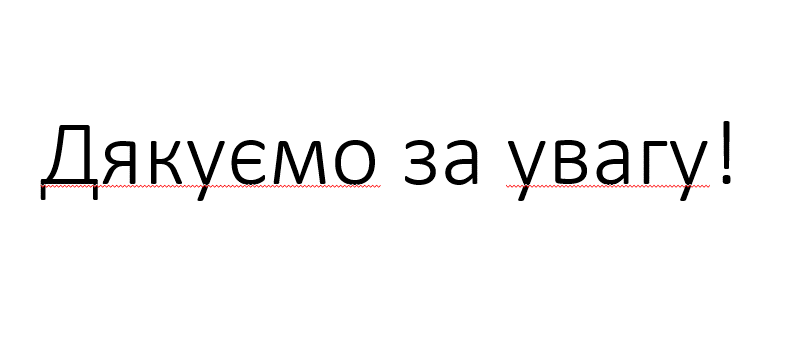


Рисунок В.53– Скріншот слайду №53