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 програмного забезпечення для автоматизації обліку розрахунків плати за гуртожиток»**

Студента  3  курсу групи ОПК-321 напряму підготовки 6.050101 «Комп’ютерні науки» спеціальності 5.05010101 «Обслуговування програмних систем і комплексів»

Нікіфоров О.М.

(прізвище та ініціали)

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

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

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

(підпис)

\_\_\_\_\_\_\_\_\_ Р.В. Ділай

(підпис)

м. Тернопіль – 2017

ЗМІСТ

[ВСТУП 5](#_Toc505888218)

[1 ТЕХНІЧНЕ ЗАВДАННЯ 6](#_Toc505888219)

[1.1 Найменування та область застосування 6](#_Toc505888220)

[1.2 Підстави для розробки 6](#_Toc505888221)

[1.3 Призначення розробки 6](#_Toc505888222)

[1.4 Вимоги до програми чи програмного виробу 7](#_Toc505888223)

[1.5 Вимоги до програмної документації 9](#_Toc505888224)

[1.6 Техніко-економічні показники 9](#_Toc505888225)

[1.7 Стадії та етапи розробки 9](#_Toc505888226)

[1.8 Порядок контролю та прийому 10](#_Toc505888227)

[2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ 11](#_Toc505888228)

[2.1 Розробка загальної структури і варіантів використання програми 11](#_Toc505888229)

[2.2 Розробка системи класів 14](#_Toc505888230)

[2.3 Розробка методів 16](#_Toc505888231)

[2.4 Проектування і опис інтерфейсу користувача 18](#_Toc505888232)

[2.5 Опис файлової структури програми 19](#_Toc505888233)

[3 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ 21](#_Toc505888234)

[ВИСНОВКИ 28](#_Toc505888235)

[ПЕРЕЛІК ПОСИЛАНЬ 29](#_Toc505888236)

[Додаток А Лістинг файлу «main.cpp» 30](#_Toc505888237)

[Додаток Б Лістинг файлу «mainwindow.h» 31](#_Toc505888238)

[Додаток В Лістинг файлу «mainwindow.cpp» 33](#_Toc505888239)

[Додаток Г Лістинг файлу «adddialog.h» 40](#_Toc505888240)

[Додаток Д Лістинг файлу «adddialog.cpp» 41](#_Toc505888241)

[Додаток Е Лістинг файлу «taker.h» 48](#_Toc505888242)

[Додаток Є Лістинг файлу «taker.cpp» 49](#_Toc505888243)

[Додаток Ж Лістинг файлу «database.h» 50](#_Toc505888244)

[Додаток З Лістинг файлу «database.cpp» 51](#_Toc505888245)

[Додаток И CD-диск із програмним продуктом 52](#_Toc505888246)

ВСТУП

Інформаційні технології є найбільш важливою складовою процесу використання інформаційних ресурсів суспільства. До теперішнього часу вони пройшли кілька еволюційних етапів, зміни в яких визначалися рівнем розвитку науково-технічного прогресу, появою все більш досконалих засобів обробки інформації.

У цій роботі розробляються інформаційна система обліку розрахунків плати за гуртожиток. Актуальность даної роботи пов'язана з розвитком науково-технічного прогресу, який вимагає застосування інноваційних комп'ютерних технологій. Сучасний науково-технічний прогрес немислимий без широкого впровадження і використання обчислювальної техніки у всіх сферах життя.

Сьогодні на основі засобів обчислювальної техніки розробляються і впроваджуються різні автоматизовані інформаційні системи. Успіх у впровадженні цих систем багато в чому залежить від фахівців з системотехніки, які знають методику аналізу і проектування цих систем, можливості обчислювальної техніки і які володіють математичними методами, які використовуються при постановці і вирішенні завдань.

Метою даної роботи є розробка автоматизованої інформаційної системи «Облік розрахунків плати за гуртожиток» для автоматизації завдання обліку реалізації.

# ТЕХНІЧНЕ ЗАВДАННЯ

## Найменування та область застосування

Найменування програми – «Розробка програмного забезпечення для автоматизації обліку розрахунків плати за гуртожиток».

Стисла назва - «Rent Room».

Область застосування програми – ведення обліку оплати за гуртожиток.

## Підстави для розробки

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

Найменування теми курсової роботи – «Розробка програмного забезпечення для автоматизації обліку розрахунків плати за гуртожиток».

Виконавець – студент групи ОПК-321 Нікіфоров Олег Михайлович

## Призначення розробки

Підвищення ефективності праці роботи секретарів чи іншої керуючої особи, яка виконує облік проживаючих у гуртожитку, керування списками кімнат, і прийняття оплати за кімнату, являється експлуатаційним призначенням даної розробки.

Відповідно функціональне призначення – оптимізація і спрощення реєстрації, обліку проживаючих у гуртожитку і ведення обліку про оплату знятих кімнат.

## Вимоги до програми чи програмного виробу

### Вимоги до функціональних характеристик

Програма для автоматизації роботи секретарів відділень вищого навчального закладу повинна забезпечити оптимізацію і спрощення ведення таких форм обліку:

* Перелік кімнат. Тут міститься номери кімнат, типи кімнат, та П.І.П жильця та датта до якої оплачена кімната якщо такий є. Таким чином, це таблиця (див.табл. 1.1) з 4 стовпців і певної кількості рядків, яка відповідає кількості номерів кімнат.

Таблиця 1.1 – Зразок таблиці груп для програми Rent Room

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Тип кімнати | Прізвище Ім’я Побатькові жильця | Проплачена до |
| 1 | Економ | Чепіль Олег Романович | 07.02.2018 |
| 2 | Стандарт | Нікіфоров Олег Миколайович | 09.05.2018 |

* Перегляд списку боржників. Тут містяться дані про всіх жильців які не оплатили прожиття на текущий місяць. Таким чином, це таблиця (див. табл. 1.2) з 3 стовпців і певної кількості рядків, яка відповідає кількості боржників.

Таблиця 1.2 – Зразок таблиці студентів для програми Student Control

|  |  |  |
| --- | --- | --- |
| Прізвище Ім’я Побатькові Боржника | Номер кімнати | Проплачена до |
| Чепіль Олег Романович | 1 | 07.02.2018 |
| Штокало Володимир Ярославович | 4 | 15.01.2018 |

* Добавлення жильця в кімнату. Це форма в якій повинна вводитись інформація про жильця кімнати, а саме: П.І.П, номер телефону, номер паспорту.
* Звільнити кімнату. Функція яка дозволяє видалити інформацію про текущого жильця кімнати, після чого кімната буде вважатися пустою й доступною до заселення.
* Внести оплату. Це функція, яка по вибору необхідної кімнати дозволяє збільшити число до якого проплачена кімната на 1 місяць.
* Перегляд інформації про жильця кімнати. Це функція яка після вибору необхідної кімнати дозволяє, у вигляді повідомлення, вивести інформацію про текущого жильця кімнати, якщо вибрана пуста кімната буде виведено інформацію про те, що вибрана пуста кімната.

### Вимоги до надійності

Надійне функціонування програми повинне забезпечуватись наявністю блоку перевірки правильності введених даних. Це збереже від помилок, що виникають при виконанні рутинної роботи користувачем.

Також необхідно забезпечити достатню стійкість програми до некваліфікованих дій користувача, що досягається супроводом будь-яких операцій в програмі чіткими і зрозумілими повідомленнями

### Умови до експлуатації

Розроблений програмний продукт по потребуватиме яких-небудь видів обслуговування, для його використання достатньо буде навиків користувача ПК.

### Вимоги до функціональних характеристик

В склад технічних засобів повинні входити: монітор (діагоналлю не менше 15”), клавіатура, мишка, IBM-сумісний персональний комп’ютер із такими мінімальними характеристиками:

* процесор – Intel Pentium;
* оперативна пам’ять - 1 ГБ;
* обсяг дискової пам’яті – 2 ГБ.

### Вимоги до інформаційної і програмної сумісності

Вихідні коди програми повинні бути реалізовані мовою С++. В якості середовища розробки програми повинне використовуватись середовище Qt Creator 4.4. Дані повинні зберігатися у базі даних.

Системні програмні засоби і утиліти, які буде використовувати програма, повинні забезпечуватись операційною системою сімейства Windows версії не нижче Windows 10.

Вимоги до захисту інформації і програми не пред’являються.

## Вимоги до програмної документації

По закінченню розробки програмного забезпечення потрібно підготувати таку документацію:

* інструкція інсталяції програми;
* загальні відомості про можливості програми;
* інструкція з експлуатації.

## Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться.

Приблизне число використань розробленої програми в рік – 10000-13000 раз.

## Стадії та етапи розробки

Розробка програми обліку розрахунків плати за гуртожиток буде мати такі стадії:

* Аналіз вимог.

Метою аналізу є максимально повний опис поставленої задачі. Усі дані, що надходять потрібно проаналізувати і систематизувати, важливо також врахувати всі технічні обмеження, які можуть виникнути на стороні замовника. Підсумком даного етапу має стати створення докладної специфікації, що відповідає всім вимогам замовника. Також слід звернути увагу і на інші чинники, які можуть ускладнювати процес розробки.

* Проектування.

На етапі проектування використовуються такі прийоми розробки класів як успадкування, композиція, наповнення. Результатом етапу проектування є створення ієрархії класів та їх опис.

* Розробка і програмування.

Програмування передбачає чотири основні стадії:

1) розробка алгоритмів – фактично, створення логіки роботи програми;

2) написання вихідного коду;

3) компіляція – перетворення в машинний код;

4) тестування і налагодження – юніт-тестування.

## 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 ПРОЕКТУ

## Розробка загальної структури і варіантів використання програми

В даному підрозділі пояснювальної записки описується такий етап розробки програми, як аналіз. Метою аналізу є максимально повний опис поставленої задачі, відповідно до технічного завдання.

Під час проходження етапу аналізу спочатку було визначено такі 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?

В даному випадку з програмою Rent Room має працювати один користувач – власник чи інша керуюча особа. Таким чином, один і той самий користувач вводить інформацію, здійснює облік і переглядає їх у різних видах.

Наступне, що потрібно було з'ясувати, це групу дій, які ініціюватимуть діючий суб'єкт. Виходячи із технічного завдання, діючий суб'єкт буде виконувати такі дії:

почати роботу з програмою;

переглянути список кімнат;

переглянути список боржників;

добавити жильця;

звільнити кімнату;

прийняти оплату;

переглянути інформацію про жильця;

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

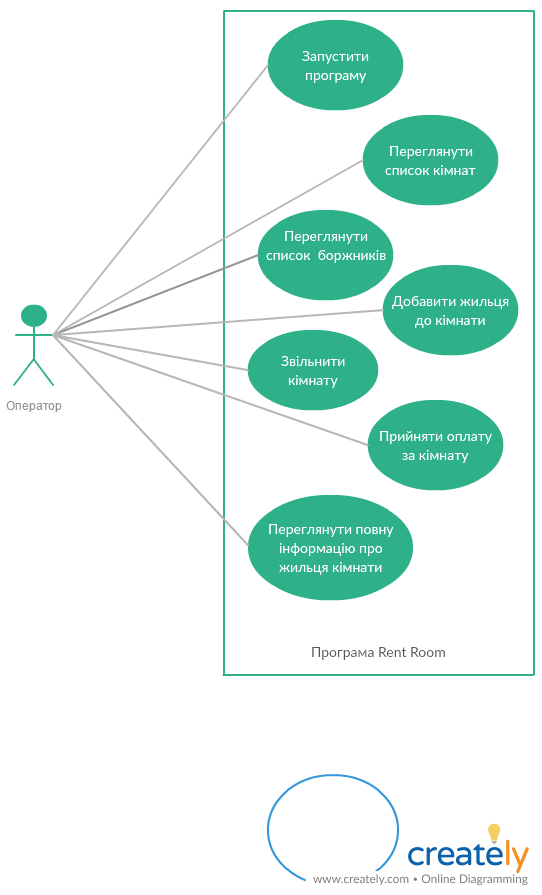


Рисунок 2.1 – Діаграма варіантів використання для програми Student Control

Наступний крок етапу аналізу це опис варіантів використання, де необхідно детально описати всі варіанти використання.

**Запустити програму.** Ця дія, здавалося б, є дуже очевидною для того, щоб про неї зовсім не згадувати, але все ж таки... Коли запускається програма, на екран повинно виводитися вікно авторизації, з якого користувач може авторизуватись і потрапити в головне меню, або зареєструватись. Це може називатися екраном інтерфейсу користувача.

**Переглянути список кімнат**: На головному вікні програми повинно вивести, у вигляді таблиці, список кімнат і інформацію про заселеного в нього жильця якщо такий присутній.

**Переглянути список боржників**: На головному вікні програми повинно вивести, у вигляді таблиці, список боржників з номерм кімнати, а також даттою до якої її було оплачено.

**Добавити нового жильця:** сценарій 1. Після вибору необхідної кімнати, якщо вибрана кімната вже зайнята то данна функція реалізована у вигляді кнопки буде не активною.

**Добавити нового жильця:** сценарій 2. Після вибору необхідної кімнати, якщо вибрана кімната пуста то данна функція реалізована у вигляді кнопки буде доступною і виведе форму для заповнення необхідних данних.

**Звільнити кімнату:** сценарій 1. Після вибору необхідної кімнати, якщо вибрана кімната пуста то данна функція реалізована у вигляді кнопки буде не активною.

**Звільнити кімнату:** сценарій 2. Після вибору необхідної кімнати, якщо вибрана кімната зайнята то данна функція реалізована у вигляді кнопки буде доступною, а після її виклику користувача перепитають про правильність його дій, якщо все вірно то данні текущого жильця буде видалено і кімната получить статус пустої.

**Прийняти оплату:** сценарій 1. Після вибору необхідної кімнати, якщо вибрана кімната пуста то данна функція реалізована у вигляді кнопки буде не активною.

**Прийняти оплату:** сценарій 2. Після вибору необхідної кімнати, якщо вибрана кімната зайнята то данна функція реалізована у вигляді кнопки буде доступною, а після її виклику дата до якої проплачена вибрана кімната збільшиться на 1 місяць.

**Переглянути інформацію про жильця кімнати:** сценарій 1. Після подвійного натискання на вибрану кімнату, якщо вона пуста буде виведено повідомлення про те, що вибрана кімната являється пустою.

**Переглянути інформацію про жильця кімнати:** сценарій 2. Після подвійного натискання на вибрану кімнату, якщо вона зайнята буде виведено повідомлення з інформацією про жильця яка була введена при реєстрації.

## Розробка системи класів

Наступний етап розробки програмного забезпечення – етап проектування. Даний етап поділяється на логічне та фізичне проектування. Логічне полягає у розробці структури класів, коли визначаються поля для збереження складових об'єктів, алгоритми методів, що описують поведінку об'єктів. Насамперед необхідно передбачити класи, з яких буде складатися програма. Для цього спочатку проаналізуємо перелік іменників з опису варіантів використання, потім внесемо деякі їх уточнення, визначимо їх атрибути, перейдемо від дієслів до повідомлень, і, на завершення, розробимо діаграми класів і послідовностей.

**Аналіз переліку іменників з опису варіантів використання.** Розглянемо перелік усіх іменників, які візьмемо з опису варіантів використання:

1. Користувач;
2. Список кімнат;
3. Список боржників;
4. База даних;
5. Кнопка;
6. Дата до якої проплачена вибрана кімната;
7. Повідомлення;
8. Головне вікно програми;
9. Форма додавання жильця;
10. П.І.П майбутнього жильця;
11. Номер телефону;
12. Номер паспорта;
13. Необхідна інформація жильця;
14. Вікно перегляду повної інформації про студента;

**Уточнення переліку іменників.** З різних причин багато іменників не зможуть стати класами. Давайте проведемо відбір тільки тих іменників, які можуть претендувати на те, щоб бути класом.

Було виписано багато різних іменників наприклад: П.І.П, номер телефону, номер паспорта. Дуже аналогічні назви притаманні жильцю, тому узагальнивши можна зробити клас КЛІЄНТ. Наступним буде створений клас БАЗА ДАНИХ який буде містити підключення до сервера на якому буде зберігатись вся інформація. Наступні 2 класів це класи вікон, які містять інформацію про усі функції вікон і іх можливості . Отже, складемо перелік класів, які було тільки що уточнено:

1. Головне вікно программи;
2. База данних;
3. Вікно добавлення жильця;
4. Клієнт;

**Визначення** **атрибутів**. Багато іменників, яким відмовлено в реєстрації як кандидатами у класи, будуть потенційними кандидатами в атрибути (компонентні дані) класів. Наприклад, в класі "Клієнт" будуть такі атрибути: П.І.П, номер телефону, номер паспорту. Більшість атрибутів можуть бути визначені так само.

**Перехід від дієслів до повідомлень.** Тепер подивимося, що нам дають варіанти використання для з'ясування того, якими повідомленнями обмінюватимуться класи. Оскільки повідомлення – це, по суті справи, є викликом методу в об'єкті, то визначення повідомлень зводиться до визначення методів класу, що приймає те або інше повідомлення. Як і у випадку з іменниками, далеко не кожне дієслово стає кандидатом у повідомлення. Деякі з них, замість прийняття даних від користувачів, зв'язуються з такими операціями, як виведення інформації на екран, або з якими-небудь ще діями.

Розглянемо опис варіанта використання функції «Вивести повну інформацію про жильця»: «Програма виводить на екран інформацію про персону яка наразі проживає у вибранній кімнаті, інформація про нього складається з 3 полів: прізвище ім’я по батькові, номер телефону, номер паспорту».

Під словом «програма» ми насправді маємо на увазі вікно інтерфейсу користувача, отже, слово «виводить» означає, що об'єкт «Вікно інтерфейсу користувача» посилає повідомлення об'єкту «Клієнт» (тобто викликає його метод). У повідомленні має міститися вказівка вивести самого себе на екран. Даний метод назвемо Display().

Дієслово «складається» не належить ні до якого повідомлення. Він просто приблизно визначає склад рядка об'єкта «База даних».

Розглянемо опис варіанту використання «Добавити нового жильця»: «У вікні має відобразитися повідомлення, у якому програма просить користувача ввести всі необхідні данні. Ця інформація повинна заноситися в таблицю. Тут, екран інтерфейсу користувача повинен послати повідомлення класу «Екран введення жильців», наказуючи йому вивести себе і отримати дані від користувача. Це повідомлення може бути викликом методу класу з іменем, наприклад addtoDatabase().

Усі інші варіанти використання були проаналізовані аналогічним чином, це дозволило створити основу для зв'язування класів, тобто дозволило виконати фізичне проектування.

**Побудова діаграм класів**. Знаючи, які класи буде включено в розроблювану програму і як вони пов'язані між собою, можна побудувати діаграму класів, яка зображуватиме взаємозв’язки між класами. Дану діаграму зображено на рисунку 2.3.

## Розробка методів

В даному підрозділі описується наступний етап розробки програмного забезпечення – еволюцію, що представляє собою етап послідовної реалізації та підключення класів до проекту. Етап починається із створення основного інтерфейсу проекту, бібліотеки класів, описуються тип поля класу, розробляються алгоритми методів, уточнюються вимоги до програми та вносяться певні зміни. Одержаний прототип продукту тестується й налагоджується. В програмі використовуються контейнери Qt, що допомагають організувати роботу із записами таблиць програми.

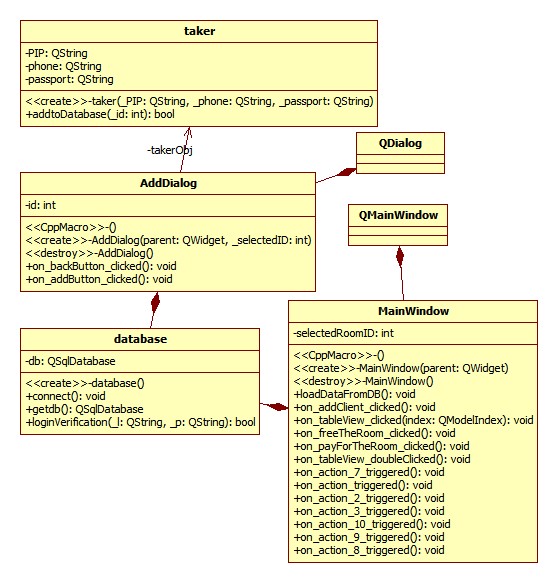


Рисунок 2.3 – Діаграма класів для програми

У всіх агрегатах було прийнято рішення зберігати покажчики замість самих об'єктів, щоб уникнути зайвого копіювання даних у пам'яті. Зберігання самих об'єктів необхідно застосовувати в тих випадках, коли об'єктів мало і вони невеликі.

Як уже було зазначено вище, багато атрибутів (методи) для кожного з класів є похідними з тих іменників, які самі не стали класами. Інші атрибути можуть бути виведені з асоціацій в діаграмі класів. Асоціації можуть визначати ті атрибути, які є покажчиками або посиланнями на інші класи. Це пояснюється неможливістю асоціювати щось з чимось, що знаходиться невідомо де.

**Складені значення (агрегати).** Агрегатні зв'язки показані в чотирьох місцях на діаграмі класів. Зазвичай агрегати виявляють ті контейнери, які є атрибутами агрегатного класу (тобто "цілого" класу, що містить "частини").

## Проектування і опис інтерфейсу користувача

Дизайн усіх вікон, які будуть використовуватись в програмі, були розроблені за допомогою Qt Designer (див.рис. 2.1-2.2).

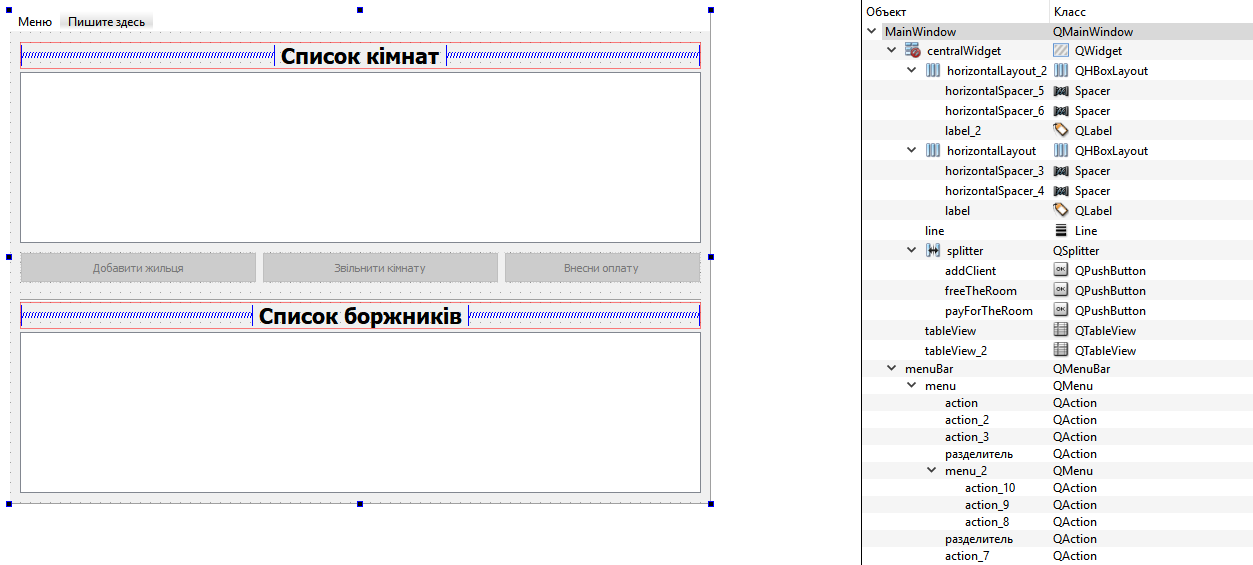


Рисунок 2.1 – Структура головного вікна програми

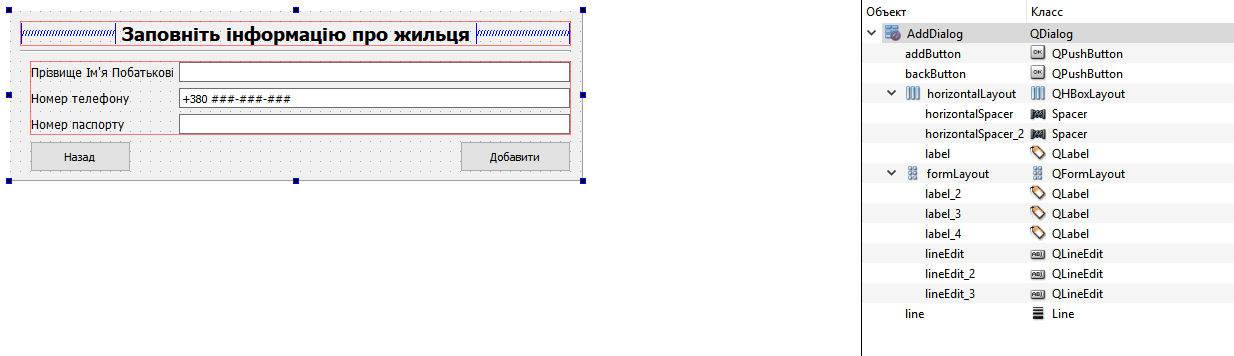


Рисунок 2.2 – Структура вікна додавання жильця

## Опис файлової структури програми

Програма складається з такий файлів:

* «main.cpp» - лістинг даного файла знаходиться в додатоку А. В цьому файлі містисься головна функція програми ;
* «mainwindow.h» - лістинг даного файла знаходиться в додатоку Б;
* «mainwindow.cpp» - лістинг даного файла знаходиться в додатоку В.

У цих 2-х файлах містяться усі методи і атрибути вікна головного меню;

* «adddialog.h» - лістинг даного файла знаходиться в додатоку Г;
* «adddialog.cpp» - лістинг даного файла знаходиться в додатоку Д.

У цих 2-х файлах містяться усі методи і атрибути вікна додавання майбутнього жильця;

* «taker.h» - лістинг даного файла знаходиться в додатоку Е;
* «taker.cpp» - лістинг даного файла знаходиться в додатоку Є.

У цих 2-х файлах містяться усі методи і атрибути класу КЛІЄНТ;

* «database.h» - лістинг даного файла знаходиться в додатоку Ж;
* «database.cpp» - лістинг даного файла знаходиться в додатоку З.

У цих 2-х файлах містяться усі методи і атрибути класу БАЗА ДАНИХ;

# ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ

В ході тестування програми було виявлено і виправлено незначні помилки. Запустивши програму на екрані монітора відображається головне вікно програми, яке представлене на рисунку 3.1. З його допомогою користувач може виконати такі дії:

* Переглянути список кімнат;
* Переглянути список боржників;
* Добавити жильця;
* Звільнити кімнату;
* Внести оплату;
* Переглянути інформацію про жильця кімнати.

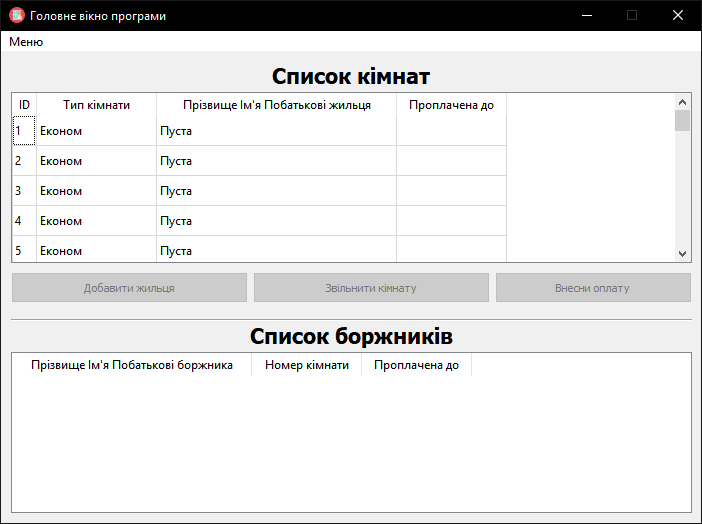


Рисунок 3.1 – Головне вікно програми

Не активні на данний момент кнопки стануть активними, тілкь відповідні, після вибору кімнати з таблиці (див.рис. 3.2).

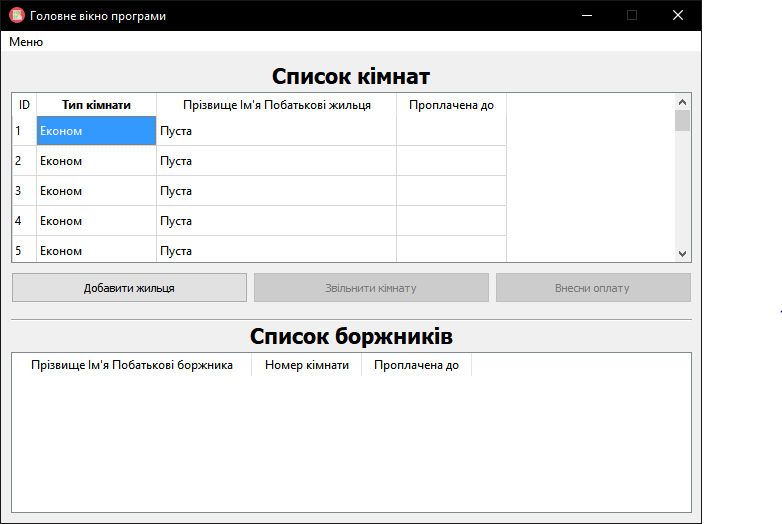


Рисунок 3.2 – Зміна активності кнопок при виборі не зайнятої кімнати

Щоби кнопки “Звільнити кімнату” і “ Внести оплату” стали активними, необхідно вибрати зайняту кімнату (див.рис. 3.4).

Кнопка “Добавити жильця” виклече форму у вигляді діалогового вікна для заповнення необхідних данних (див.рис. 3.3). З відси користувач уже може добавити жильця до попередньо вибранної кімнати, або повернутися до головного меню.

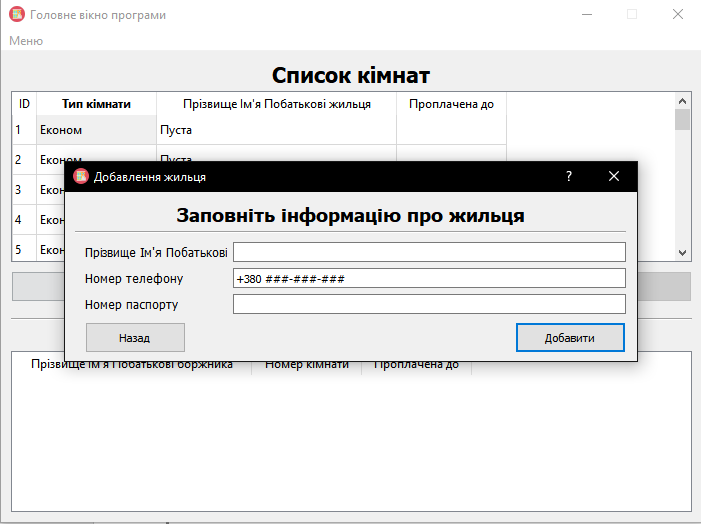


Рисунок 3.3 – Діалогове вікно добавлення жильця

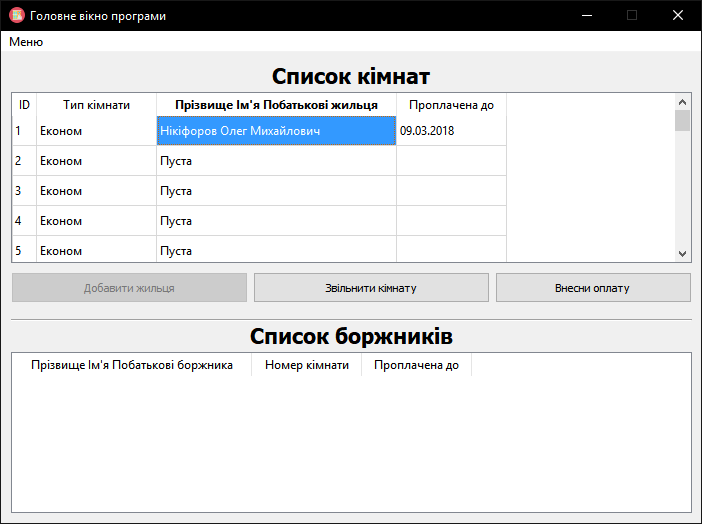


Рисунок 3.4 – Зміна активності кнопок при виборі зайнятої кімнати

Також можна переглянути інформацію, про жилця вибранної кімнати, завдяки подвійному кліку по необхідному полю таблиці. Інформацію, або повідомлення про вибір пустої кімнати буде виведено у вигляді повідомлення (див.рис. 3.5 – 3.6).

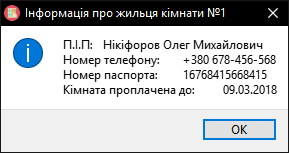


Рисунок 3.5 – Повідомлення з інформацією про жильця кімнати

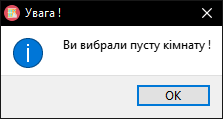


Рисунок 3.6 – Повідомлення про вибір пустої кімнати

Усі функції є також продубльовані у меню головного вікна, разом з довідкою і функцією виходу з програми (див.рис. 3.7), за для зручності у використанні.

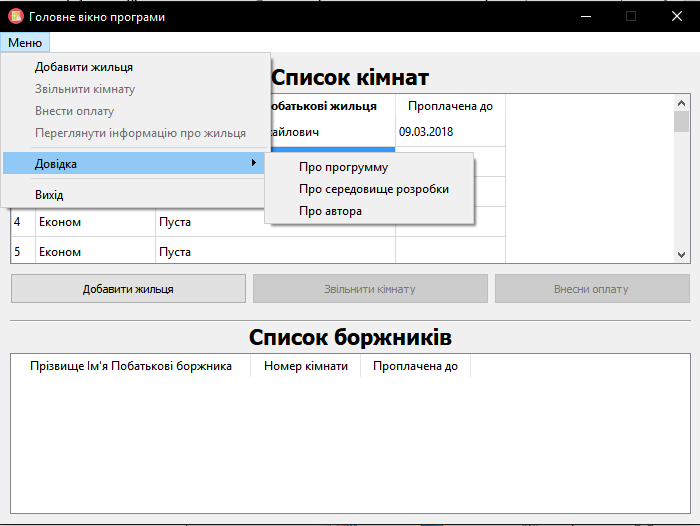


Рисунок 3.7 – Відображення функцій меню програми

Також усі важливі дії користувача супроводжуються повідомленнями які перевіряють правильність дій користувача (див.рис. 3.8).

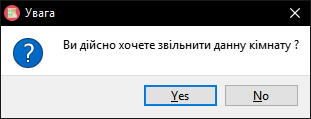


Рисунок 3.8 – Вікно - діалог перевірки дій користувача

ВИСНОВКИ

В даній курсовій роботі було описано етапи розробки програми Rent Room, яка призначена для обліку оплати за гуртожиток. Розроблена програма відповідає усім поставленим вимогам технічного завдання.

Під час виконання курсової роботи було закріплено і поглиблено знання, одержанні за час вивчення предмету «Об’єктно-орієнтоване програмування», більш детально вивчено мову програмування С++ й отримано практичні навички розробки пограмного продукту.

Також в даній роботі було використане середовище Qt, що дозволило швидше розробити програмне забезпечення, за допомогою стандартних бібліотек фреймворку Qt.

ПЕРЕЛІК ПОСИЛАНЬ

1. Грицюк Ю.І. Об’єктно-орієнтоване програмування мовою С++ [Текст]: навчальний посібник / Ю.І. Грицюк, Т.Є. Рак. – Львів: Вид-во Львівського ДУ БЖД, 2011. – 404 с.
2. Build with Qt [Електронний ресурс]. – Електорон. дан. – Режим доступу: <https://www1.qt.io/built-with-qt/>. – Get started with Qt.
3. Qt Creator Manual [Електронний ресурс]. – Електорон. дан. – Режим доступу: <http://doc.qt.io/qtcreator/index.html> – Qt Documentation.

Додаток А  
Лістинг файлу «main.cpp»

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.setFixedSize(w.size());

w.show();

**return** a.exec();

}

Додаток Б  
Лістинг файлу «mainwindow.h»

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "database.h"

#include "adddialog.h"

#include <QMessageBox>

#include <QDate>

**namespace** Ui {

**class** MainWindow;

}

**class** MainWindow : **public** QMainWindow

{

Q\_OBJECT

**public**:

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

~***MainWindow***();

*void* **loadDataFromDB**();

*void* **ViewInfo**();

**private** **slots**:

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

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

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

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

*void* **on\_tableView\_doubleClicked**();

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

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

*void* **on\_action\_2\_triggered**();

*void* **on\_action\_3\_triggered**();

*void* **on\_action\_10\_triggered**();

*void* **on\_action\_9\_triggered**();

*void* **on\_action\_8\_triggered**();

*void* **on\_tableView\_2\_doubleClicked**(**const** QModelIndex &index);

*void* **on\_action\_4\_triggered**();

**private**:

Ui::MainWindow \*ui;

database db;

*int* selectedRoomID;

};

#endif **//** **MAINWINDOW\_H**

Додаток В  
Лістинг файлу «mainwindow.cpp»

#include "mainwindow.h"

#include "ui\_mainwindow.h"

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

QMainWindow(parent),

ui(**new** Ui::MainWindow)

{

ui->setupUi(**this**);

db.connect();

loadDataFromDB();

}

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

{

**delete** ui;

}

*void* MainWindow::**loadDataFromDB**()

{

QSqlQueryModel \*model = **new** QSqlQueryModel();

QSqlQuery query = QSqlQuery(db.getdb());

query.prepare("select `apartments`.`id`, `apartments`.`type`, `apartments`.`PIP`, `apartments`.`paidTo` from `apartments`;");

query.exec();

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

model->setQuery(query);

model->*setHeaderData*(0, Qt::Horizontal, QObject::tr("ID"));

model->*setHeaderData*(1, Qt::Horizontal, QObject::tr("Тип кімнати"));

model->*setHeaderData*(2, Qt::Horizontal, QObject::tr("Прізвище Ім'я Побатькові жильця"));

model->*setHeaderData*(3, Qt::Horizontal, QObject::tr("Проплачена до"));

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

ui->tableView->setColumnWidth(0,25);

ui->tableView->setColumnWidth(1,120);

ui->tableView->setColumnWidth(2,240);

ui->tableView->setColumnWidth(3,110);

//////////////////////////////////////////////

model = **new** QSqlQueryModel();

query.prepare("select `apartments`.`PIP`, `apartments`.`id`, `apartments`.`paidTo` from `apartments` where paidTo <= :date;");

query.bindValue(":date",QDate::currentDate().addDays(1));

query.exec();

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

model->setQuery(query);

model->*setHeaderData*(0, Qt::Horizontal, QObject::tr("Прізвище Ім'я Побатькові боржника"));

model->*setHeaderData*(1, Qt::Horizontal, QObject::tr("Номер кімнати"));

model->*setHeaderData*(2, Qt::Horizontal, QObject::tr("Проплачена до"));

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

ui->tableView\_2->setColumnWidth(0,240);

ui->tableView\_2->setColumnWidth(1,110);

ui->tableView\_2->setColumnWidth(2,110);

}

*void* MainWindow::**ViewInfo**()

{

QDate date;

QString PIP, phone, passport;

QSqlQuery query = QSqlQuery(db.getdb());

query.prepare("select `apartments`.`PIP`, `apartments`.`phone`, `apartments`.`passport`, `apartments`.`paidTo` from `apartments` where `id` = :id");

query.bindValue(":id",selectedRoomID);

query.exec();

**while**(query.next()){

PIP = query.value(0).toString();

phone = query.value(1).toString();

passport = query.value(2).toString();

date = query.value(3).toDate();

}

**if** (!date.isNull())

QMessageBox::information(**this**,"Інформація про жильця кімнати №"+QString::number(selectedRoomID),"П.І.П:\t"+PIP+"\nНомер телефону:\t"+phone+"\nНомер паспорта:\t"+passport+"\nКімната проплачена до:\t"+date.toString("dd.MM.yyyy"));

**else** QMessageBox::information(**this**,"Увага !","Ви вибрали пусту кімнату !");

}

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

{

AddDialog \*addForm = **new** AddDialog(0,selectedRoomID);

addForm->setFixedSize(addForm->size());

addForm->*exec*();

loadDataFromDB();

}

*void* MainWindow::**on\_tableView\_clicked**(**const** QModelIndex &index)

{

QDate date;

QModelIndex x = QModelIndex(index.sibling(index.row(),0));

selectedRoomID = x.data().toInt();

x = QModelIndex(index.sibling(index.row(),3));

date = x.data().toDate();

**if** (date.isNull()){

ui->addClient->setEnabled(1); ui->freeTheRoom->setEnabled(0); ui->payForTheRoom->setEnabled(0);

ui->action->setEnabled(1); ui->action\_2->setEnabled(0); ui->action\_3->setEnabled(0); ui->action\_4->setEnabled(0);

} **else**{

ui->addClient->setEnabled(0); ui->freeTheRoom->setEnabled(1); ui->payForTheRoom->setEnabled(1);

ui->action->setEnabled(0); ui->action\_2->setEnabled(1); ui->action\_3->setEnabled(1); ui->action\_4->setEnabled(1);

}

}

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

{

QSqlQuery query = QSqlQuery(db.getdb());

query.prepare("update apartments set `PIP` = 'Пуста', `phone` = NULL, `passport` = NULL, `paidTo` = NULL where `id` = :id");

query.bindValue(":id",selectedRoomID);

QMessageBox::StandardButton reply;

reply = QMessageBox::question(**this**, "Увага","Ви дійсно хочете звільнити данну кімнату ?",

QMessageBox::Yes|QMessageBox::No);

**if** (reply == QMessageBox::Yes) {

**if**(!query.exec()){

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

QMessageBox::warning(**this**,"Помилка !","Сталася невідома помилка !");

} **else** QMessageBox::information(**this**,"Успішно !","Кімнату звільнено успішно !");;

}

loadDataFromDB();

}

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

{

QDate date;

QSqlQuery query = QSqlQuery(db.getdb());

query.prepare("select `apartments`.`paidTo` from `apartments` where `id` = :id");

query.bindValue(":id",selectedRoomID);

query.exec();

**while**(query.next()){

date = query.value(0).toDate();

}

query.prepare("update apartments set `paidTo` = :time where `id` = :id");

query.bindValue(":id",selectedRoomID);

query.bindValue(":time",date.addMonths(1));

QMessageBox::StandardButton reply;

reply = QMessageBox::question(**this**, "Увага","Ви збираєтесь збільшити срок проплаченності на місяць !",

QMessageBox::Yes|QMessageBox::No);

**if** (reply == QMessageBox::Yes) {

**if**(!query.exec()){

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

QMessageBox::warning(**this**,"Помилка !","Сталася невідома помилка !");

} **else** QMessageBox::information(**this**,"Успішно !","Оплату прийнято !");;

}

loadDataFromDB();

}

*void* MainWindow::**on\_tableView\_doubleClicked**()

{

ViewInfo();

}

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

{

QCoreApplication::exit();

}

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

{

on\_addClient\_clicked();

}

*void* MainWindow::**on\_action\_2\_triggered**()

{

on\_freeTheRoom\_clicked();

}

*void* MainWindow::**on\_action\_3\_triggered**()

{

on\_payForTheRoom\_clicked();

}

*void* MainWindow::**on\_action\_10\_triggered**()

{

QMessageBox::information(**this**,"Про програму:","Версія 1.2");

}

*void* MainWindow::**on\_action\_9\_triggered**()

{

QMessageBox::aboutQt(**this**);

}

*void* MainWindow::**on\_action\_8\_triggered**()

{

QMessageBox::information(**this**,"Про автора:","Программа написана Нікіфором Олегом");

}

*void* MainWindow::**on\_tableView\_2\_doubleClicked**(**const** QModelIndex &index)

{

*int* temp;

QModelIndex x = QModelIndex(index.sibling(index.row(),1));

temp = x.data().toInt();

QDate date;

QString PIP, phone, passport;

QSqlQuery query = QSqlQuery(db.getdb());

query.prepare("select `apartments`.`PIP`, `apartments`.`phone`, `apartments`.`passport`, `apartments`.`paidTo` from `apartments` where `id` = :id");

query.bindValue(":id",temp);

query.exec();

**while**(query.next()){

PIP = query.value(0).toString();

phone = query.value(1).toString();

passport = query.value(2).toString();

date = query.value(3).toDate();

}

QMessageBox::information(**this**,"Інформація про боржника кімнати №"+QString::number(temp),"П.І.П:\t"+PIP+"\nНомер телефону:\t"+phone+"\nНомер паспорта:\t"+passport+"\nКімната проплачена до:\t"+date.toString("dd.MM.yyyy"));

}

*void* MainWindow::**on\_action\_4\_triggered**()

{

ViewInfo();

}

Додаток Г  
Лістинг файлу «adddialog.h»

#ifndef ADDDIALOG\_H

#define ADDDIALOG\_H

#include <QDialog>

#include "taker.h"

#include <QMessageBox>

**namespace** Ui {

**class** AddDialog;

}

**class** AddDialog : **public** QDialog

{

Q\_OBJECT

**public**:

**explicit** AddDialog(QWidget \*parent = 0, *int* \_selectedID = 0);

~***AddDialog***();

**private** **slots**:

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

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

**private**:

*int* id;

taker \*takerObj;

Ui::AddDialog \*ui;

};

#endif **//** **ADDDIALOG\_H**

Додаток Д  
Лістинг файлу «adddialog.cpp»

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent, QString \_l, QString \_p, class Widget \*\_prev) :

QMainWindow(parent), prev(\_prev),

ui(new Ui::MainWindow), login(\_l), pass(\_p)

{

ui->setupUi(this);

createUserObj();

setdepartament();

curentID = 0;

*int* a = userobj.getacces();

switch (a) {

case 1:{

ui->pushButton\_2->setEnabled(0); ui->action->setEnabled(0);

ui->pushButton\_3->setEnabled(0); ui->action\_2->setEnabled(0);

ui->pushButton\_5->setEnabled(0);

ui->pushButton\_6->setEnabled(0); ui->action\_6->setEnabled(0);

break;

}

case 2: {

ui->pushButton\_5->setEnabled(0); ui->action->setEnabled(0);

ui->pushButton\_3->setEnabled(0); ui->action\_2->setEnabled(0);

ui->pushButton\_6->setEnabled(0); ui->action\_6->setEnabled(0);

break;

}

case 3 : {

break;

}

default:

QCoreApplication::exit();

}

}

*void* MainWindow::getdata()

{

QSqlQuery query = QSqlQuery(db.getdb());

*int* id;

query.prepare("select `departaments`.`id` from `departaments` where `departaments`.`name` = :name;");

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

query.exec();

while(query.next())

{

id = query.value(0).toInt();

}

/////////////////////////////////

model= new QSqlQueryModel();

query.prepare("select `groups`.`id`, `groups`.`name`, `groups`.`code` from `groups`, `departaments` where `groups`.`departament\_code` = :ID and `departaments`.`name` = :name;");

query.bindValue(":ID",id);

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

query.exec();

model->setQuery(query);

model->*setHeaderData*(0, Qt::Horizontal, QObject::tr("ID"));

model->*setHeaderData*(1, Qt::Horizontal, QObject::tr("Назва спеціальності"));

model->*setHeaderData*(2, Qt::Horizontal, QObject::tr("Група"));

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

ui->tableView->setColumnWidth(0,25);

ui->tableView->setColumnWidth(1,240);

ui->tableView->setColumnWidth(2,127);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

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

{

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Увага","Ви дійсно хочете вийти з програми ?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes)

QCoreApplication::exit();

}

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

{

this->hide();

addStudent = new addStudentForm(0,this,0,false,0);

addStudent->setFixedSize(addStudent->size());

addStudent->show();

getdata();

}

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

{

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Увага","Ви дійсно хочете змінити користувача ?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes){

this->hide();

prev->show();

prev->clearLines();

}

}

*void* MainWindow::on\_tableView\_doubleClicked(const QModelIndex &index)

{

QModelIndex x = QModelIndex(index.sibling(index.row(),0));

openGroupInfo(x.data().toInt());

}

*void* MainWindow::openGroupInfo(*int* x)

{

infoForm = new studentControl(0, this, x, userobj.getacces());

infoForm->setFixedSize(infoForm->size());

this->hide();

infoForm->show();

}

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

{

addUser = new addUserForm(0,this);

this->hide();

addUser->setFixedSize(addUser->size());

addUser->show();

}

*void* MainWindow::createUserObj()

{

QSqlQuery query = QSqlQuery(db.getdb());

query.exec("select name, sname, tname, sex, bday, phone\_number, adress, type\_of\_acces from users where login='" + login + "' and pass ='" + pass + "';");

*int* acces;

QDate bday;

*bool* sex;

QString name, sname, tname, phone\_number, adress;

while(query.next()){

name = query.value(0).toString();

sname = query.value(1).toString();

tname = query.value(2).toString();

sex = query.value(3).toBool();

bday = query.value(4).toDate();

phone\_number = query.value(5).toString(); adress = query.value(6).toString();

acces = query.value(7).toInt();

}

userobj = user(name, sname, tname, sex, bday, phone\_number, adress, acces, login, pass);

}

*void* MainWindow::setdepartament()

{

QSqlQuery query = QSqlQuery(db.getdb());

model= new QSqlQueryModel();

query.prepare("select `departaments`.`name` from `departaments`;");

query.exec();

model->setQuery(query);

ui->comboBox->setModel(model);

ui->comboBox->setModelColumn(0);

getdata();

}

*void* MainWindow::on\_comboBox\_currentIndexChanged(*int* index)

{

getdata();

}

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

{

QMessageBox::aboutQt(this);

}

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

{

QMessageBox::information(this,"Про автора:","Made by Alex \"ShunA\" Pokydko");

}

*void* MainWindow::on\_action\_5\_triggered()

{

QMessageBox::information(this,"Про програму:","Версія 0.8.9а \nЕтапи розробки можна переглянути на \nhttps://github.com/Shuna322/aloha-shaka-laka/");

}

*void* MainWindow::on\_action\_6\_triggered()

{

on\_pushButton\_5\_clicked();

}

*void* MainWindow::on\_tableView\_clicked(const QModelIndex &index)

{

QModelIndex x = QModelIndex(index.sibling(index.row(),0));

curentID = x.data().toInt();

ui->pushButton\_7->setEnabled(1);

}

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

{

openGroupInfo(curentID);

}

*void* MainWindow::on\_action\_4\_triggered()

{

on\_pushButton\_4\_clicked();

}

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

{

depForm = new departamentContol(0,this);

depForm->setFixedSize(depForm->size());

this->hide();

depForm->show();

}

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

{

groupConForm = new groupControl(0,this);

groupConForm->setFixedSize(groupConForm->size());

this->hide();

groupConForm->show();

}

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

{

on\_pushButton\_3\_clicked();

}

*void* MainWindow::on\_action\_2\_triggered()

{

on\_pushButton\_6\_clicked();

}

*void* MainWindow::on\_action\_3\_triggered()

{

on\_pushButton\_clicked();}

Додаток Е  
Лістинг файлу «taker.h»

#ifndef TAKER\_H

#define TAKER\_H

#include <QString>

#include "database.h"

#include <QDate>

**class** taker

{

QString PIP, phone, passport;

**public**:

taker(QString \_PIP, QString \_phone, QString \_passport);

*bool* **addtoDatabase**(*int* \_id);

};

#endif **//** **TAKER\_H**

Додаток Є  
Лістинг файлу «taker.cpp»

#include "taker.h"

taker::**taker**(QString \_PIP, QString \_phone, QString \_passport):

PIP(\_PIP), phone(\_phone), passport(\_passport)

{

}

*bool* taker::**addtoDatabase**(*int* \_id)

{

database db;

QSqlQuery query = QSqlQuery(db.getdb());

qDebug () << PIP;

query.prepare("update apartments set `PIP` = :pip, `phone` = :phone, `passport` = :passport, `paidTo` = :paidTo where `id` = :id");

query.bindValue(":id",\_id);

query.bindValue(":pip",PIP);

query.bindValue(":phone",phone);

query.bindValue(":passport",passport);

query.bindValue(":paidTo",QDate::currentDate().addMonths(1));

**if**(!query.exec()){

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

**return** 0;

} **else** **return** 1;

}

Додаток Ж  
Лістинг файлу «database.h»

#ifndef DATABASE\_H

#define DATABASE\_H

#include <QWidget>

#include <QString>

#include <Qsql>

#include <QSqlDriver>

#include <QSqlDatabase>

#include <QtSql/QSqlQuery>

#include <QtSql/QSqlDatabase>

#include <QtSql/QSqlTableModel>

#include <QtSql/QSqlError>

#include <QSqlRecord>

#include <QDebug>

#include <QSqlQuery>

**class** database

{

QSqlDatabase db;

**public**:

database();

*void* **connect**();

QSqlDatabase **getdb**();

*bool* **loginVerification**(QString \_l, QString \_p);

};

#endif **//** **DATABASE\_H**

Додаток З  
Лістинг файлу «database.cpp»

#include "database.h"

#include <QMessageBox>

#include <QCoreApplication>

database::**database**()

{

}

*void* database::**connect**()

{

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

db.setHostName("149.202.160.211");

db.setDatabaseName("gyrtogutok");

db.setUserName("gyrtogutok");

db.setPassword("4K9z0J7a");

**if** (!db.open())

QMessageBox(QMessageBox::Critical,"Помилка підключення до бази данних",db.lastError().text()).*exec*();

**else** qDebug() << "До бази даних підключився!";

}

QSqlDatabase database::**getdb**()

{

**return** db;

}

Додаток И  
CD-диск із програмним продуктом