**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ профессиональное**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**моСКОВСКОЙ ОБЛАСТИ  
«КРАСНОГОРСКИЙ КОЛЛЕДЖ»**

СПЕЦИАЛЬНОСТЬ 09.02.03

«ПРОГРАММИРОВАНИЕ В КОМПЬЮТЕРНЫХ СИСТЕМАХ»

**ДИПЛОМНЫЙ ПРОЕКТ**

**НА ТЕМУ**

**«**Разработка базы данных «Репетиционная студия» на языке Python»

Студент \_\_\_\_\_\_/ Семенцов-Огиевский А.М. /

Руководитель \_\_\_\_\_\_\_\_\_\_/ Зибанов А.А. /

Допущен к защите:

Зам.директора по УР

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ И.М. Рыжова /

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

г. Красногорск

2017 г.

Красногорск

2010г

2010г.

2010 г.

|  |  |
| --- | --- |
| ОГЛАВЛЕНИЕ  **ВВЕДЕНИЕ** | 5 |
| **Глава 1. Основание для разработки дипломного проекта.** | **6** |
| 1.1. Цель и назначение разработки проекта. | **6** |
| 1.2. Постановка задачи. | **7** |
| 1.3. Описание предметной области | **7** |
| **Глава 2. Разработка проекта программного продукта.** | **9** |
| 2.1. Предлагаемый алгоритм выполнения проекта. | **9** |
| 2.2. Обоснование требований к оборудованию, программным средствам и языку программирования для реализации проекта. | **10** |
| 2.3. Предлагаемые этапы проектирования и методы контроля законченных этапов проекта. | **12** |
| **Глава 3. Практическая часть.** | **13** |
| 3.1. Разработка алгоритма и описание компонентов (составных частей) программного комплекса. Общие сведения. | **13** |
| 3.2. Описание логической структуры информационного компонента и программного комплекса. Входные данные. Выходные данные. | **18** |
| 3.3. Функциональное назначение и порядок взаимодействия программных и информационных компонентов. | **20** |
| 3.4. Разработка программного продукта. | **26** |
| 3.5.Создание интерфейса программного продукта. Написание программного кода по разработанному алгоритму. | **50** |
| 3.6.Создание демонстрационных исходных данных, позволяющих протестировать разрабатываемый программный комплекс во всех режимах. | 55 |

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

**ДП 09.02.03.17.19 ПЗ**

Разраб.

**Семенцов-Огиевский**

Провер.

Зибанов А.А.

Реценз.

**Толстов А.А.**

Н. Контр.

**Шутов А.И.**

Утверд.

**Рыжова И.М.**

Разработка базы данных «Репетиционная студия» на языке Python

Лит.

Листов

65

**ГРУППА 46Пд-13К**

|  |  |
| --- | --- |
| ОГЛАВЛЕНИЕ  Изм.  Лист  № докум.  Подпись  Дата  Лист  4  **ДП 09.02.03.17.19 ПЗ** |  |
| 3.7.Написание инструкций:  - оператору по инсталляции программного комплекса;  - оператору по работе с программным комплексом и действий в случае сбойных ситуаций; | **57** |
| **Глава 4. Техника безопасности при работе с программным продуктом** | **62** |
| Заключение. | **63** |
| Список источников и литературы. | **64** |
| Приложение. Код программы. | **66** |

# ВВЕДЕНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

**ДП 09.02.03.17.19 ПЗ**

Дипломное проектирование является заключительным этапом обучения в Красногорском колледже. В рамках выполнения работы мне предстоит показать большинство навыков, полученных за время обучения. А именно: проектирование баз данных, разработка алгоритмов программы, создание удобного, интуитивно понятного интерфейса пользователя, разработка собственного компонента, создание файла справочной системы и другие.

Актуальность темы дипломного проекта заключается в том, в реализации данного проекта, студент использовал язык программирования Python3 и кроссплатформенный инструментарий для создания программного обеспечения QT5. С помощью их он реализовал приложение для учета и введения базы данных с SQLite3, для начинающей репетиционной базы «9/10». В данном проекте реализованы все требования и пожелания предъявленными заказчиком.

Дипломный проект состоит из 4 глав.

В первой главе необходимо обосновать требования к разработке дипломного проекта: цель и назначение проекта, постановка задачи, описание предметной области. Данная стадия работы важна тем, что именно решение задач предметной области является основополагающим фактором, который предопределит конечный вид программы.

Во второй главе нужно описать предлагаемый алгоритм выполнения проекта, обосновать требования к оборудованию, программным средствам и языку программирования для реализации дипломного проекта, а также предполагаемые этапы проектирования и методы контроля законченных этапов проекта.

Третья глава является практической частью. В ней отображён непосредственно процесс разработки программного продукта, разработка интерфейса.

Четвёртая глава диплома посвящена технике безопасности при работе с вычислительной техникой.

**Глава 1. Основание для разработки дипломного проекта.**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

**ДП 09.02.03.17.19 ПЗ**

Основанием для разработки данного дипломного проекта является задание разработать автоматизированную базу данных «Репетиционная студия» на языке Python, выбранную мною из списка предложенных заданий для разработки диплома. Задание утверждено руководителем практики и заместителем директора по учебной работе Красногорского колледжа.

* 1. **Цель и назначение разработки проекта.**

Целью выполнения дипломного проекта должна стать демонстрация приемной комиссии:

- приобретенных мною знаний в объеме профессиональных модулей (ПМ):

1. ПМ.01 – Разработка программных модулей программного обеспечения для компьютерных систем.
2. ПМ.01.02 - Раздел 2. ПМ. Разработка и эксплуатация баз данных

- профессиональных компетенций (ПК), указанных в п. 5.2 стандарта по специальности 09.02.03 (230115) “Программирование в компьютерных системах”, утвержденному приказом Министра образования и науки Российской Федерации N 804 от 28 июля 2014 г.,

- моей готовности к деятельности, указанной в п. 4.3 стандарта.

Назначением данного дипломного проекта является практическая реализация приобретенных знаний в виде разработанного программного комплекса и сопроводительной документации к нему в виде настоящего документа, который позволит приемной комиссии оценить уровень приобретенных мною знаний, профессиональных компетенций и готовности к профессиональной деятельности.

* 1. **Постановка задачи**

В рамках дипломного проекта требуется разработать программный комплекс и документацию к нему, соответствующие выбранной теме «Разработка базы данных «Репетиционная студия» на языке Python».

* 1. **Описание предметной области**

В настоящее время большинство репетиционных баз ведут учет используя таблицы Microsoft Excel, поэтому от сотрудников требуется умение работать с данной программой, а так как она не позволяет полноценно упростить и автоматизировать учет, от сотрудников требуется хорошая внимательность. Таким образом, возникает проблема с ошибками в работе с Microsoft Excel из-за невнимательности сотрудников.

Требуется разработать информационную систему для автоматизации учета аренды помещения и оборудования для репетиционной базы. Система должна предусматривать режим учета оплаты репетиции, в котором будет учитываться почасовая оплата помещения и арендованное оборудование, а также режим записи групп на репетицию. В системе должна хранится информация о группах, в виде двух полей:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

**ДП 09.02.03.17.19 ПЗ**

- Наименование группы

- Данные о членах группы

В системе должны храниться информация, о оборудование которая репетиционная база предлагает в аренду, для этого должна быть таблица, хранящая следующие поля:

- Наименование оборудования

- Стоимость оборудования

Репетиционная база должна записывать группы на репетиции, для этого должна существовать таблица записей имеющая следующие поля:

- Наименование группы

- Дата репетиции

- Время начала репетиции

- Время конца репетиции

- Информация о заказчике

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

**ДП 09.02.03.17.19 ПЗ**

Так же должна существовать таблица учета, хранящая в себе следующие поля:

- Наименование группы

- Дата репетиции

- Время начала репетиции

- Время конца репетиции

- Наименования арендованного оборудования

- Стоимость репетиции

Для работы с данными, система должна хранить информацию о пользователях, ограничивающую им уровень доступа:

- Логин

- Пароль

- ФИО

- Номер телефона

- Статус пользователя.

Система должна быть проста в использование и автоматизирована, что бы сотрудники репетиционной студии могли без ошибочно и просто работать с базой данных.

**Глава 2. Разработка проекта программного продукта**

**2.1. Предлагаемый алгоритм выполнения проекта**

Предлагаемая программа реализует автоматизированную систему контроля учета и ведение базы данных для репетиционной базы. Сотрудник должен входить в систему под своим именем и взаимодействовать с ней.

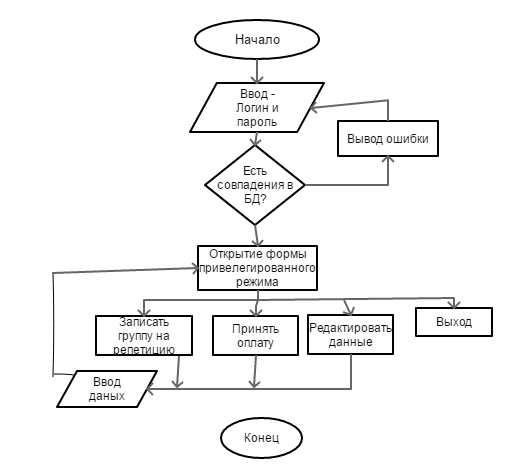


Рисунок 1. Основной алгоритм работы приложения.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

9

**ДП 09.02.03.17.19 ПЗ**

**2.2. Обоснование требований к оборудованию, программным средствам и языку программирования для реализации проекта**

**Требований к оборудованию**

Программа рассчитана на любого пользователя, обладающего базовыми навыками работы с компьютером.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

**ДП 09.02.03.17.19 ПЗ**

Минимальные технические требования к ЭВМ, необходимой для нормального функционирования программы:

1) Процессор: Intel Core i3, 2Ггц

2) Оперативная память: 4 ГБ

3) Видеоадаптер.

4) Клавиатура, мышь.

Рекомендуемые технические требования:

1) Процессор Intel Core i3 и выше.

2) Оперативная память: 4 ГБ и выше.

3) Видеоадаптер.

4) Клавиатура, мышь.

**Выбор языка программирования**

Для разработки программного продукта я выбрал среду разработки JetBrains PyCharm Community Edition с использованием языка программирования Python3.

Python –  высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Преимущества языка программирования Python:

* Простота в написание кода
* Быстродействие
* Читаемость кода

Так же вместе с языком программирования Python будет использоваться кроссплатформенный инструментарий для создания программного обеспечения QT5, в реализации библиотеки PyQt5 для Python3.

Qt позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

**ДП 09.02.03.17.19 ПЗ**

**2.3. Предлагаемые этапы проектирования и методы контроля законченных этапов проекта**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

12

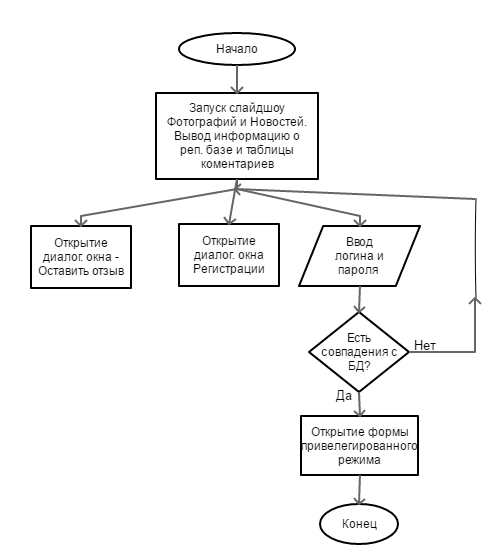
**ДП 09.02.03.17.19 ПЗ**

|  |  |  |  |
| --- | --- | --- | --- |
| № Этапа | Наименование работ | Чем заканчивается работа | Срок исполнения, начало окончание |
| 1. | Анализ задания на дипломный проект и подготовка раздела «Введение» | Написание раздела «Введение» | 27.04.17-28.04.17 |
| 2. | Подготовка раздела «Цель и назначение разработки проекта» | Написание раздела «Цель и назначение разработки проекта» | 28.04.17-29.04.17 |
| 3. | Подготовка раздела «Постановка задачи» | Написание раздела «Постановка задачи» | 29.04.17-30.04.17 |
| 4. | Подготовка раздела «Описание предметной области» | Написание раздела «Описание предметной области» | 01.05.17-04.05.17 |
| 5. | Подготовка раздела «Разработка алгоритма программного комплекса» | Написание раздела «Разработка алгоритма программного комплекса» | 04.05.17-06.05.17 |
| 6. | Подготовка раздела «Описание логической структуры программного комплекса» | Написание раздела «Описание логической структуры программного комплекса» | 06.05.17-08.05.17 |
| 7. | Подготовка раздела «Функциональное назначение программных и информационных компонентов» | Написание раздела «Функциональное назначение программных и информационных компонентов» | 08.05.17-10.05.17 |
| 8. | Подготовка раздела «Разработка программного продукта» | Написание раздела «Разработка программного продукта» | 10.05.17-20.05.17 |
| 9. | Подготовка раздела «Создание интерфейса ПП. Создание демонстрационных исходных данных. Написание инструкций» | Написание раздела «Создание интерфейса ПП. Создание демонстрационных исходных данных. Написание инструкций» | 20.05.17-28.05.17 |
| 10. | Тестирование программы | Готовая рабочая программа | 29.05.17-01.06.17 |
| 11. | Подготовка разделов «Заключение» и «Список литературы» | Наличие готовых разделов | 01.06.17-05.06.17 |

Таблица 1. Стадии и этапы проектирования.

**Глава 3. Практическая часть**

**3.1. Разработка алгоритма и описание компонентов (составных частей) программного комплекса. Общие сведения. Разработка алгоритма**



Изм.

Лист

№ докум.

Подпись

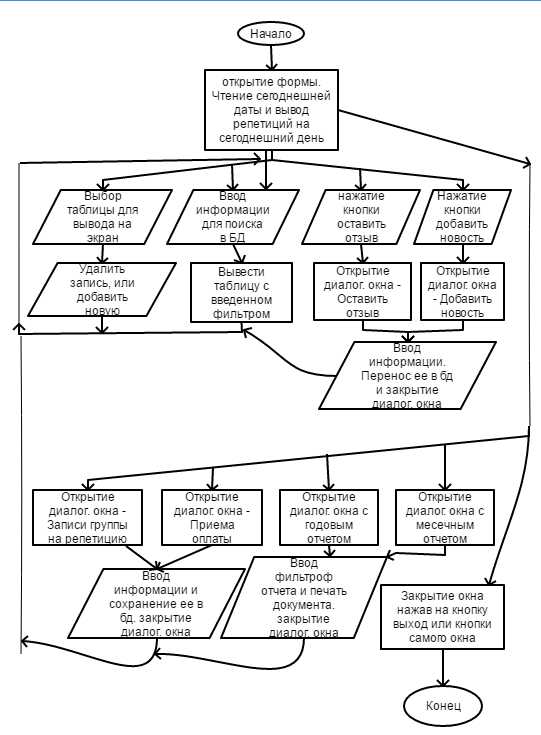
Дата

Лист

13

**ДП 09.02.03.17.19 ПЗ**

Рисунок 2. Алгоритм работы главной формы.



Изм.

Лист

№ докум.

Подпись

Дата

Лист

14

**ДП 09.02.03.17.19 ПЗ**

Рисунок 3. Алгоритм работы формы привилегированного режима.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

15

**ДП 09.02.03.17.19 ПЗ**

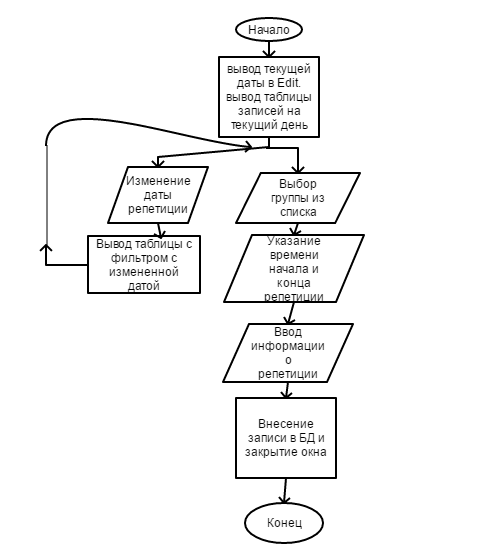


Рисунок 4. Алгоритм работы диалог. окна – Запись.

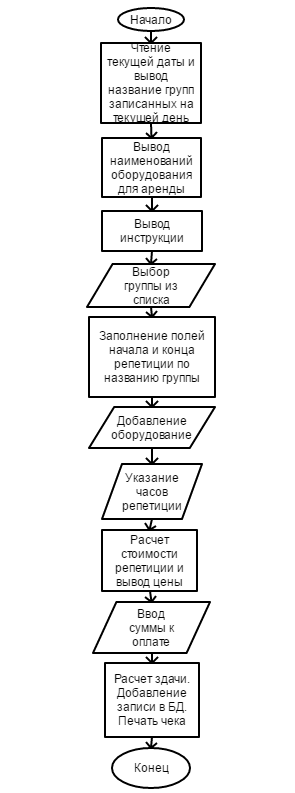


Рисунок 5. Алгоритм работы диалог. окна приема оплаты.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

**ДП 09.02.03.17.19 ПЗ**

**Описание программы**

Данная программа предназначена для ведения учета и базы данных. С помощи нее, сотрудник должен записывать группы на репетиции и принимать оплату, после ее прохождения.

Программа реализована в среде разработки JetBrains PyCharm Community Edition.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

**ДП 09.02.03.17.19 ПЗ**

**Общие сведения**

Программное обеспечение: Microsoft Windows 7 и выше…

Среда разработки: JetBrains PyCharm Community Edition.

Стадия разработки программного продукта включает в себя разработку интерфейса, разработку базы данных и написание программного кода. Все действия я выполняю параллельно, постепенно дополняя.

Программный продукт состоит из:

1) Интерфейса пользователя

2) Программного кода

3) Базы данных

4) Отчетов

**3.2. Описание логической структуры информационного компонента и программного комплекса. Входные данные. Выходные данные.**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

18

**ДП 09.02.03.17.19 ПЗ**

### Описание логической структуры

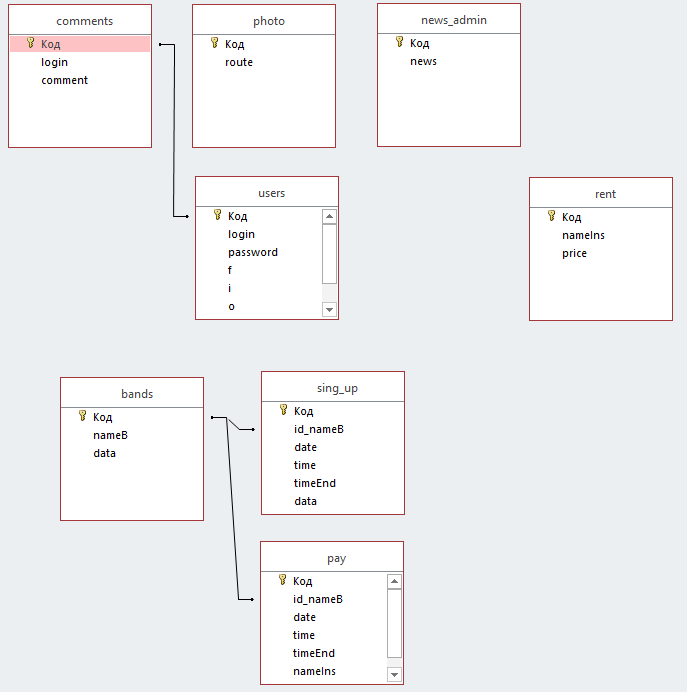


Схема 1. ER-диаграмма.

**Входные данные**

Входными данными являются:

1) Список пользователей

2) Список групп

3) Список оборудования

**Выходные данные**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

Выходными данными являются:

1) Оплата репетиций

2) Записи на репетиции

3) Годовой отчет

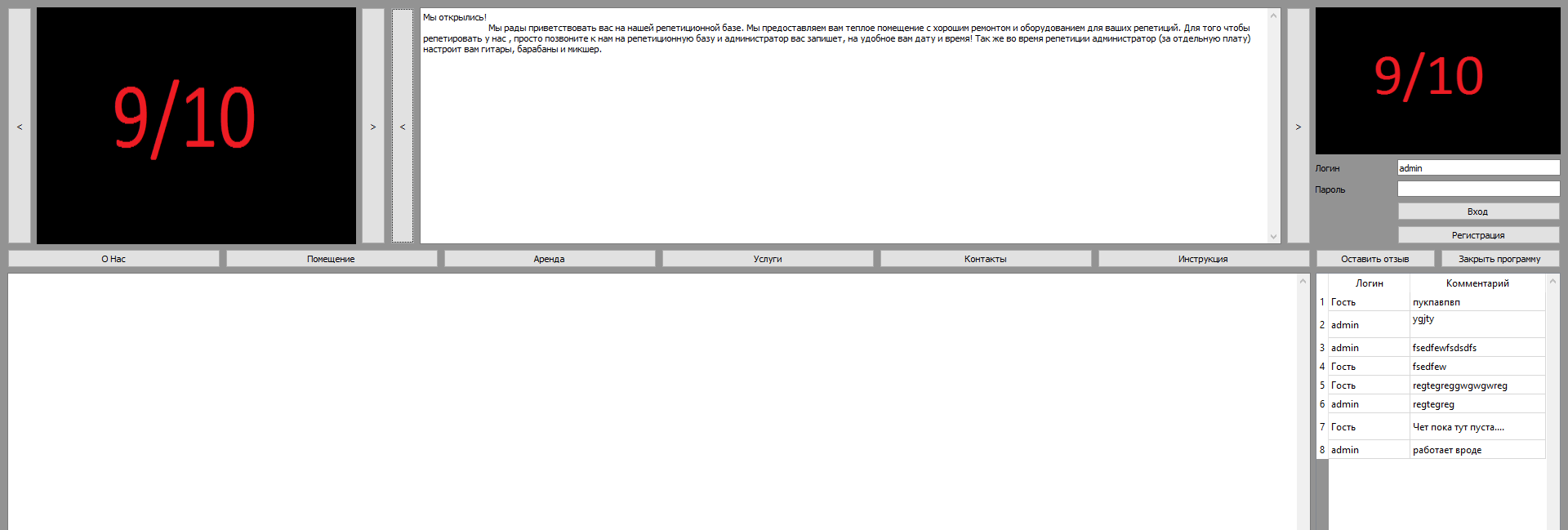
4) Месячный отчет

5) Чек об оплате

**3.3. Функциональное назначение и порядок взаимодействия программных и информационных компонентов**

**ДП 09.02.03.17.19 ПЗ**

1) Компонент Главное меню



Изм.

Лист

№ докум.

Подпись

Дата

Лист

20

**ДП 09.02.03.17.19 ПЗ**

Рисунок 6. Главная форма.

Главная форма осуществляет открытие двух диалоговых окон и переход в форму привилегированного режима. Ее объявляет класс Ui\_MainWindow, реализованный в модуле main\_windows.py. При нажатии на кнопку «Оставить отзыв» выводится диалоговое окно, и при нажатии кнопки «Регистрация» выводится диалоговое окно, а при правильном вводе логина и пароля, и последующем нажатие на кнопку «Вход», открывается форма привилегированного режима.

2) Компонент диалоговое окно «Оставить отзыв».

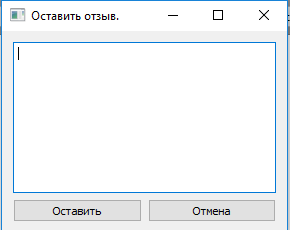


Рисунок 7. Ддиалоговое окно «Оставить отзыв».

Данный компонент позволяет добавить запись в базу данных в таблицу «comments».

3) Компонент диалоговое окно «Регистрация»

Изм.

Лист

№ докум.

Подпись

Дата

Лист

21

**ДП 09.02.03.17.19 ПЗ**

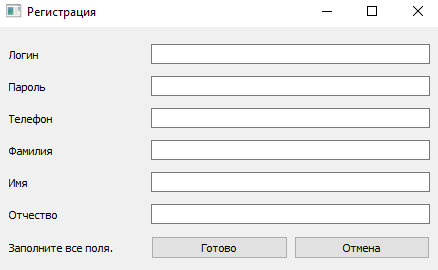


Рисунок 8. Диалоговое окно «Регистрация».

Компонент «Регистрация», позволяет зарегистрировать в системе сотрудника, для дальнейшего входа в систему.

4) Компонент Привилегированный режим.

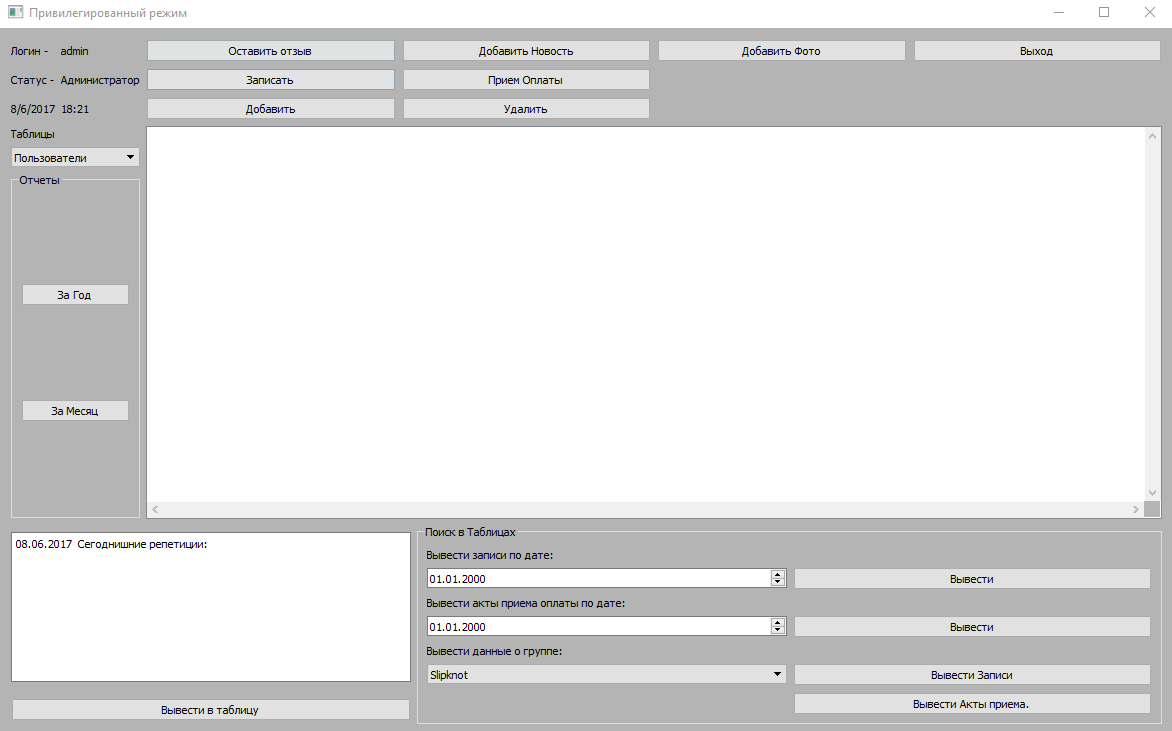


Рисунок 9. Форма Привилегированного режима.

Данный компонент позволяет пользователю производить манипуляции с базой данных. Так же данная форма осуществляет переход в 7 диалоговых окон, одно из которых описано выше («Оставить отзыв»).

5) Компонент диалоговое окно «Оставить новость»

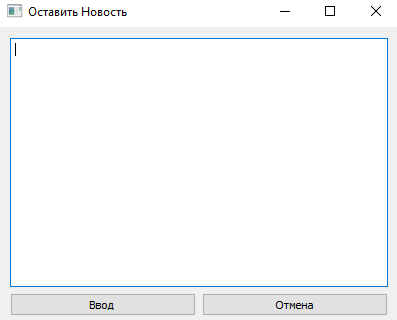


Рисунок 10. Диалоговое окно «Оставить отзыв».

Данный компонент дает возможность добавить запись в таблицу «news\_admin», для дальнейшего вывода ее на главной форме.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

22

**ДП 09.02.03.17.19 ПЗ**

6) Компонент диалоговое окно «Добавить фото»

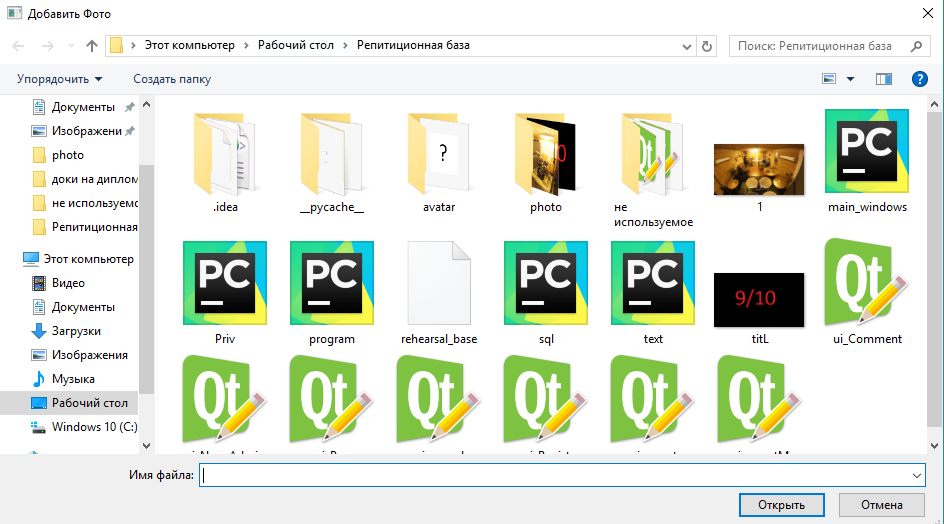


Рисунок 11. Диалоговое окно «Добавить фото»

Данный компонент реализует стандартное диалоговое окно поиска файлов, с помощью которого можно добавить фотографию в папку «Photo» в каталоге программы, для дальнейшего вывода ее на главной форме.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

23

**ДП 09.02.03.17.19 ПЗ**

7) Компонент диалоговое окно «Запись на репетицию»

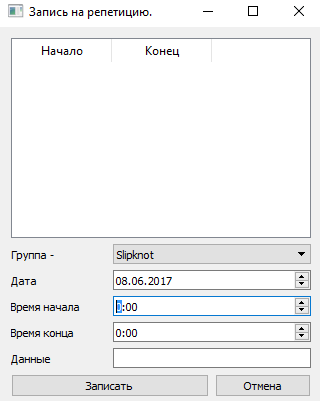


Рисунок 12. Диалоговое окно «Запись на репетицию»

Этот компонент позволяет записывать группы на репетицию. Для этого указывается дата репетиции, а в таблице выводится занятые промежутки времени на этот день. После выбирается непосредственно название группы и заполняются остальные поля.

8) Компонент диалоговое окно «Прием оплаты»

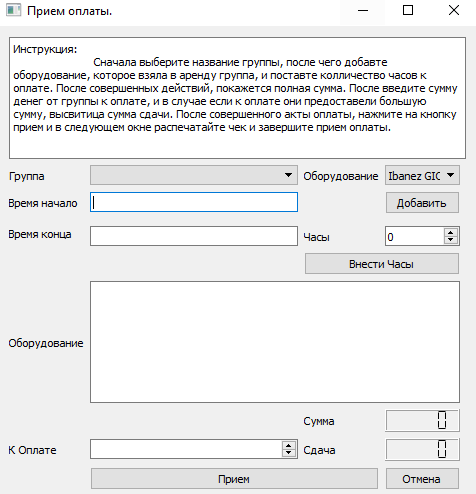


Рисунок 13. Диалоговое окно «Прием оплаты»

Этот компонент позволяет принимать оплату за репетиции. В элемент списка добавляются названия групп, которые записаны на текущую дату. При выборе группы поля «Время начало» и «Время конца» заполняются автоматически. После мы выбираем оборудование из списка и добавляем его в арендованное оборудование, а после указываем количество часов (эти данные не вводятся автоматически по просьбе заказчика). После мы вводим количество средств внесенными группой. После окончания оформления, печатается чек об оплате и добавляется запись в таблицу базы данных.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

24

**ДП 09.02.03.17.19 ПЗ**

9) Компонент диалоговое окно «Отчет»

Изм.

Лист

№ докум.

Подпись

Дата

Лист

25

**ДП 09.02.03.17.19 ПЗ**

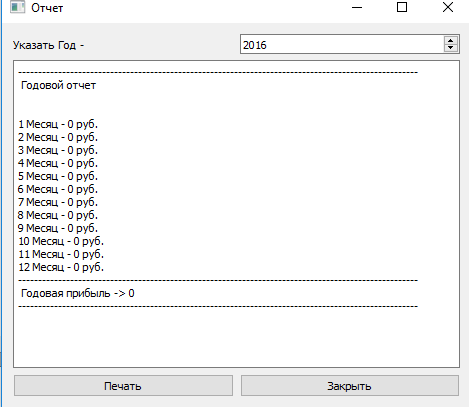


Рисунок 14. Диалоговое окно «Отчет»

Этот компонент выводит годовой отчет на экран. Мы можем указать за какой год нам нужен отчет и распечатать его на принтере.

10) Компонент диалоговое окно «Отчет»

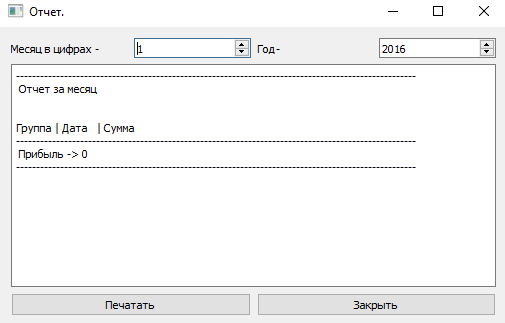


Рисунок 15. Диалоговое окно «Отчет»

Этот компонент позволяет вывести месячный отчет на экран. Мы можем указать за какой месяц и год должен выводиться отчет, а также распечатать его на принтере.

**3.4. Разработка программного продукта**

Разработку программного продукта я начал с создания базы данных. У СУБД SQLite3 нет официальной программы для создания баз данных. В данной программе будем использовать СУБД в виде библиотеки для языка Python3. Для создания базы данных мы будем выполнять следующие действия -

1) Открыть JetBrains PyCharm Community

2) Создать модуль с расширением «.py»

3) Прописываем следующий код:

**class** SqlCom():  
 **def** \_\_init\_\_(self):  
 self.app = QtWidgets.QApplication(sys.argv)  
 self.con = QtSql.QSqlDatabase.addDatabase(**'QSQLITE'**)  
 self.con.setDatabaseName(**'rehearsal\_base'**)  
 self.con.open()  
 self.query = QtSql.QSqlQuery()  
 **if 'news\_admin' not in** self.con.tables():  
 self.query.exec(**"create table news\_admin(ID integer not null primary key autoincrement, news text) "**)  
 self.query.prepare(**"insert into news\_admin values(null,?)"**)  
 self.query.addBindValue(**'Проба ураааааа сработало!!!!'**)  
 self.query.exec\_()  
 **if 'photo' not in** self.con.tables():  
 self.query.exec(**"create table photo(ID integer not null primary key autoincrement, route text)"**)  
 self.query.prepare(**"insert into photo values(null,?)"**)  
 self.query.addBindValue(**'photo/1.jpg'**)  
 self.query.exec\_()  
 self.query.prepare(**"insert into photo values(null,?)"**)  
 self.query.addBindValue(**'photo/2.jpg'**)  
 self.query.exec\_()  
 **if 'comments' not in** self.con.tables():  
 self.query.exec(**"create table comments(ID integer not null primary key autoincrement, login text, comment text)"**)  
 self.query.prepare(**"insert into comments values(null,?,?)"**)  
 self.query.addBindValue(**'admin'**)  
 self.query.addBindValue(**'работает вроде'**)  
 self.query.exec\_()  
 self.query.prepare(**"insert into comments values(null,?,?)"**)  
 self.query.addBindValue(**'Гость'**)  
 self.query.addBindValue(**'Чет пока тут пуста....'**)  
 self.query.exec\_()  
 **if 'users' not in** self.con.tables():  
 self.query.exec(**"create table users(ID integer not null primary key autoincrement, login text, password text,f text,i text,o text, phone integer, status text)"**)  
 self.query.prepare(**"insert into users values(null,?,?,?,?,?,?,?)"**)  
 self.query.addBindValue(**'admin'**)  
 self.query.addBindValue(**'7291'**)  
 self.query.addBindValue(**'Семенцов'**)  
 self.query.addBindValue(**'Алексей'**)  
 self.query.addBindValue(**'Михайлович'**)  
 self.query.addBindValue(**'8998888888'**)  
 self.query.addBindValue(**'Администратор'**)  
 self.query.exec\_()  
 **if 'bands' not in** self.con.tables():  
 self.query.exec(**"create table bands(ID integer not null primary key autoincrement, nameB text, data text)"**)  
 self.query.prepare(**"insert into bands values(null,?,?)"**)  
 self.query.addBindValue(**'Slipknot'**)  
 self.query.addBindValue(**'Кори Тейлор - 89995674354'**)  
 self.query.exec\_()  
 **if 'rent' not in** self.con.tables():  
 self.query.exec(**"create table rent(ID integer not null primary key autoincrement, nameIns text, price integer)"**)  
 self.query.prepare(**"insert into rent values(null,?,?)"**)  
 self.query.addBindValue(**'Ibanez GIO'**)  
 self.query.addBindValue(**'100'**)  
 self.query.exec\_()  
 **if 'sing\_up' not in** self.con.tables():  
 self.query.exec(**"create table sing\_up(ID integer not null primary key autoincrement, id\_nameB INTEGER REFERENCES bands (ID) ON DELETE CASCADE ON UPDATE CASCADE, date integer , time text, timeEnd text, data text)"**)  
 self.query.prepare(**"insert into sing\_up values(null,?,?,?,?,?)"**)  
 self.query.addBindValue(**'1'**)  
 self.query.addBindValue(**'05.06.2017'**)  
 self.query.addBindValue(**'12.00'**)  
 self.query.addBindValue(**'11.00'**)  
 self.query.addBindValue(**'Мик Томсон 89993216548'**)  
 self.query.exec\_()  
 **if 'pay' not in** self.con.tables():  
 self.query.exec(**"create table pay(ID integer not null primary key autoincrement, id\_nameB INTEGER REFERENCES bands (ID) ON DELETE CASCADE ON UPDATE CASCADE, date text, time text,timeEnd text,nameIns text ,price integer)"**)  
 self.query.prepare(**"insert into pay values(null,?,?,?,?,?,?)"**)  
 self.query.addBindValue(**'1'**)  
 self.query.addBindValue(**'05.06.2017'**)  
 self.query.addBindValue(**'12.00'**)  
 self.query.addBindValue(**'13.00'**)  
 self.query.addBindValue(**'Ibanez GIO'**)  
 self.query.addBindValue(**'1500'**)  
 self.query.exec\_()  
 self.stm = QtSql.QSqlTableModel()  
 self.stmR = QtSql.QSqlRelationalTableModel()

Изм.

Лист

№ докум.

Подпись

Дата

Лист

26

**ДП 09.02.03.17.19 ПЗ**

Этот код создает класс SqlCom, параметрами которого являются подключение к базе данных, маркер для выполнения SQL запросов и макет для вывода таблиц. Так же он создает базу данных с таблицами и начальными записями, в случае если их не существует. Рассмотрим создания таблиц подробнее на примере таблицы «news\_admin»:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

27

**ДП 09.02.03.17.19 ПЗ**

Она создается и в нее вносится стартовая информация с помощью следующего кода -

**if 'news\_admin' not in** self.con.tables():  
 self.query.exec(**"create table news\_admin(ID integer not null primary key autoincrement, news text) "**)  
 self.query.prepare(**"insert into news\_admin values(null,?)"**)  
 self.query.addBindValue(**'Проба ураааааа сработало!!!!'**)  
 self.query.exec\_()

Создание таблицы происходит с помощью SQL запроса «Create Table» переданному маркеру для выполнения, в нем мы указываем название полей и указывает тип данных. (SQLite3 поддерживает только 4 типа данных: text, integer, blob и real. Так же он не поддерживает связи таблиц на уровне базы данных. При написании кода мы все равно связываем таблицы на уровне базы данных, так как в будущих обновлениях Python и SQLite возможны добавления такой возможность.) После, с помощью SQL запроса «Insert Into» переданному маркеру, мы добавляем информацию в созданную нами таблицу с помощью встроенного в маркер метода «addBindValue», а после выполняем запрос с помощью метода «exec\_».

Изм.

Лист

№ докум.

Подпись

Дата

Лист

28

**ДП 09.02.03.17.19 ПЗ**

После создания базы данных и ее таблиц, мы должны реализовать методы класса SqlCom, для дальнейшей работы программы с БД. Для этого создадим следующие методы:

1. Метод STMR.

Данный метод реализует создание макета таблицы базы данных, с учетом связывания таблиц. Для этого напишем следующий код:

**def** STMR(self,table,filt=**''**,us=**''**):  
 self.stmR.setEditStrategy(QtSql.QSqlTableModel.OnFieldChange)  
 self.stmR.setTable(table)  
 **if** filt !=**''**:  
 print(**"'"**+filt+**"'"**+**'='**+**"'"**+us+**"'"**)  
 self.stmR.setFilter(filt+**'='**+**"'"**+us+**"'"**)  
 self.stmR.setRelation(1, QtSql.QSqlRelation(**'bands'**,**'ID'**,**'nameB'**))  
 self.stmR.select()  
 **return** self.stmR

В этот метод передаются параметры: название таблицы, название столбца и собственно информация для поиска. После выполнения кода возвращается созданный макет.

1. Метод STM.

Данный метод реализует создание макета таблицы базы данных, без учета связывания таблиц. Для этого напишем следующий код:

**def** STM(self,table):  
 self.stm.setTable(table)  
 self.stm.setEditStrategy(QtSql.QSqlTableModel.OnFieldChange)  
 self.stm.select()  
 **return** self.stm

В этот метод передается параметр название таблицы. После выполнения кода возвращается созданный макет.

1. Метод Delete.

Данный метод реализует удаление записей из таблиц БД по условию указанными параметрами метода. Для этого напишем следующий код:

**def** Delete(self,tabl,name, date):  
 self.query.exec(**"delete from "** + tabl+**' where id\_nameB='**+**"'"**+str(name)+**"'"**+**' and date='**+**"'"**+date+**"'"**)

В этот метод передаются параметры: название таблицы, название столбца для поиска и информация для условия.

1. Метод Insert.

Этот метод реализует добавление информации в таблицы базы данных. Для этого напишем следующий код:

**def** Insert(self,tabl,data1,data2=**''**,data3=**''**,data4=**''**,data5=**''**,data6=**''**,data7=**''**):  
 **if** data2 != **''**:  
 **if** data3 != **''**:  
 **if** data4 != **''**:  
 **if** data5 !=**''**:  
 **if** data6 !=**''**:  
 **if** data7 != **''**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?,?,?,?,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.addBindValue(data3)  
 self.query.addBindValue(data4)  
 self.query.addBindValue(data5)  
 self.query.addBindValue(data6)  
 self.query.addBindValue(data7)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?,?,?,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.addBindValue(data3)  
 self.query.addBindValue(data4)  
 self.query.addBindValue(data5)  
 self.query.addBindValue(data6)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?,?,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.addBindValue(data3)  
 self.query.addBindValue(data4)  
 self.query.addBindValue(data5)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.addBindValue(data3)  
 self.query.addBindValue(data4)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.addBindValue(data3)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?,?)"**)  
 self.query.addBindValue(data1)  
 self.query.addBindValue(data2)  
 self.query.exec\_()  
 self.query.finish()  
 **else**:  
 self.query.prepare(**"insert into "** + tabl + **" values(null,?)"**)  
 self.query.addBindValue(data1)  
 self.query.exec\_()  
 self.query.finish()

Изм.

Лист

№ докум.

Подпись

Дата

Лист

29

**ДП 09.02.03.17.19 ПЗ**

В этот метод передаются параметры: название таблицы и значения для заполнения таблицы.

1. Метод SelectMaxID.

Этот метод считает количество строк в таблице, указанной в полученных параметрах метода. Для этого напишем следующий код:

**def** SelectMaxID(self,table):  
 self.query.exec(**"select count(1) from "** + table)  
 lst = 0  
 **if** self.query.isActive():  
 self.query.first()  
 lst=self.query.value(**'count(1)'**)  
 self.query.next()  
 **return** lst

Метод возвращает количество строк в таблице.

1. Метод SelectOne.

Метод возвращает значение столбца из таблицы в виде списка, так же с помощью параметров метода, метод выполняется по условию. Напишем следующий код:

**def** SelectOne(self, table, column,columnS=**''**,line=**''**):  
 **if** columnS == **''**:  
 self.query.exec(**"select \* from "**+table)  
 lst=[]  
 **if** self.query.isActive():  
 self.query.first()  
 **while** self.query.isValid():  
 lst.append(self.query.value(column))  
 self.query.next()  
 **return** lst  
 **else**:  
 line=str(line)  
 self.query.exec(**"select \* from "** + table+**' where '**+columnS+**'='**+line)  
 lst = []  
 **if** self.query.isActive():  
 self.query.first()  
 **while** self.query.isValid():  
 lst.append(self.query.value(column))  
 self.query.next()  
 **return** lst

Метод возвращает список с полученными значениями.

Далее мы приступаем к разработки самого приложения.

С помощью Python3 и стандартной его библиотеки tkinter очень сложно создавать интерфейс приложения, так как не существует программ дизайнеров для создания окон. Поэтому создание интерфейса происходит с помощью написания кода. В этом проекте мы будем использовать кроссплатформенный инструмент по созданию программного обеспечения Qt, в его вариации PyQt5 для работы с Python.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

30

**ДП 09.02.03.17.19 ПЗ**

Для создания окон мы воспользуемся программой QT Designer. Открываем программу и создаем окно типа main windows.

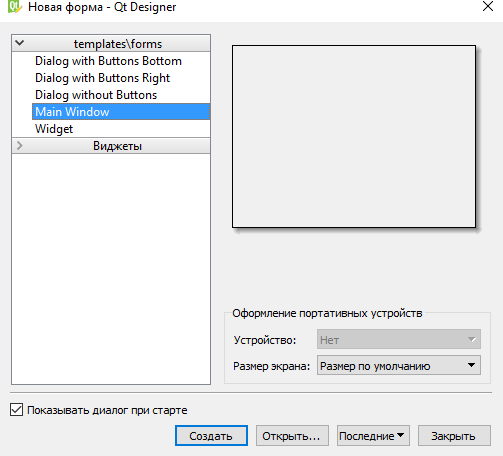


Рисунок 16. Создание нового окна в QT Designer.

Далее мы перетаскиваем компоненты, которые мы будем использовать. После настраиваем их с помощью инструментов и сохраняем.

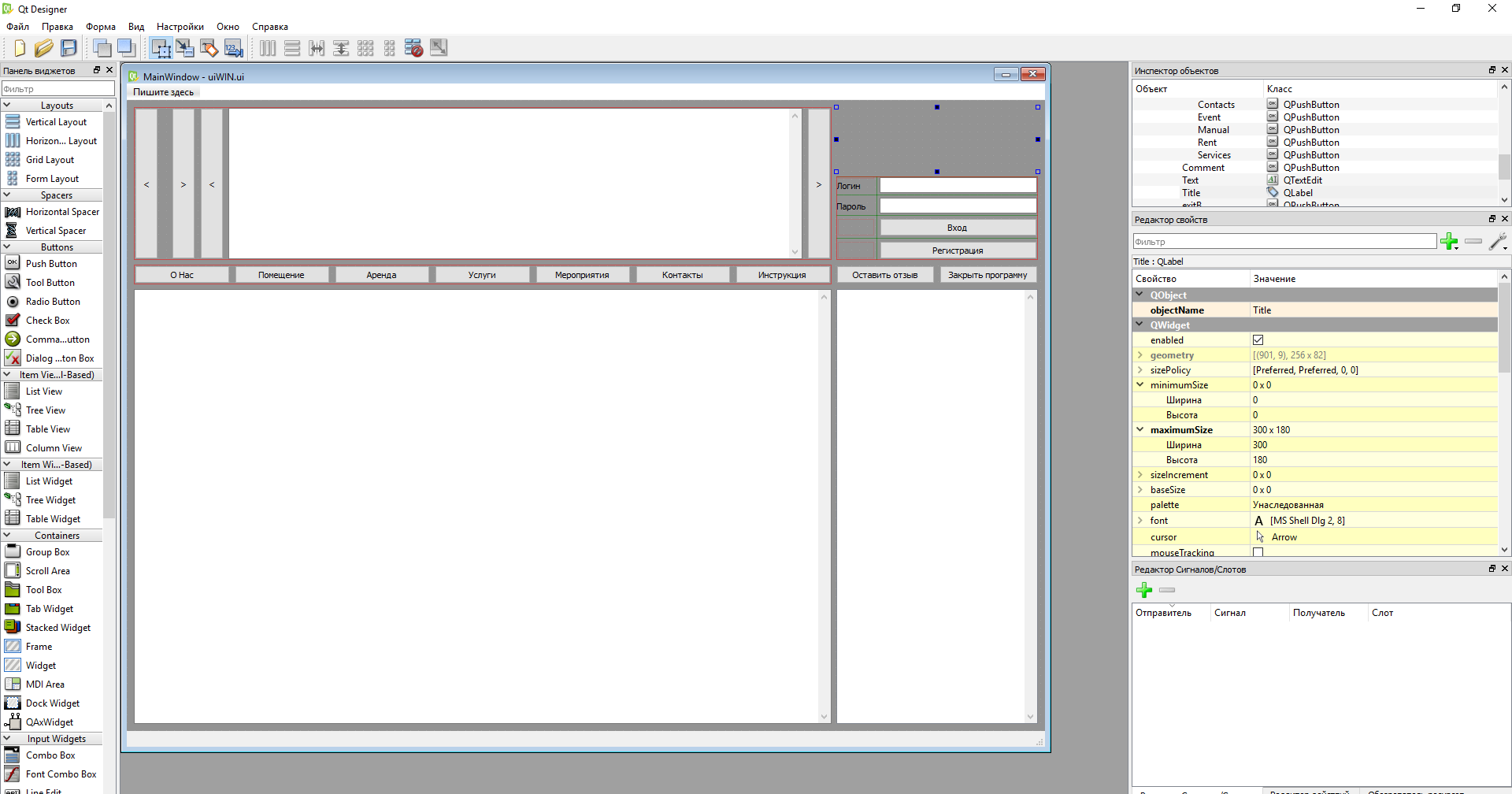


Рисунок 17. Расстановка компонентов.

Далее созданный файл мы переведем в код языка программирования Python. Для этого откроем командную строку и используем команду pyuic5.bat.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

31

**ДП 09.02.03.17.19 ПЗ**

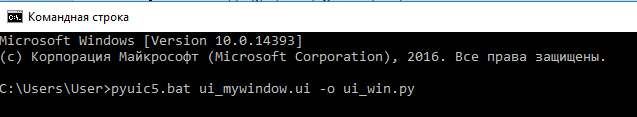


Рисунок 18. Перевод окна в модуль Python.

После созданный файл импортируем в нашу программу. Чтобы работать с созданной формой в нашей программе, создаем класс для работы с этой формой, где в параметрах указываем переменные, перенастраиваем окно и подключаем методы к кнопкам. Для этого пишем следующий код:

**class** MyWindow(QtWidgets.QMainWindow, main\_windows.Ui\_MainWindow):  
 **def** \_\_init\_\_(self, parent=**None**):  
 super(MyWindow, self).\_\_init\_\_()  
 self.setupUi(self)  
 self.sql = sql.SqlCom()  
 *# переменные для работы кнопок* self.photos = self.sql.SelectOne(**'photo'**, **'route'**)  
 self.idNews = 1  
 self.idMax = self.sql.SelectMaxID(**'news\_admin'**)  
 self.news = self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews)  
 self.News.setText(self.news[0])  
 self.Photo.setPixmap(QtGui.QPixmap(self.photos[0]))  
 self.counP = 0  
 self.posP = -1  
 **for** i **in** self.photos:  
 self.posP += 1  
 *# вывод таблицы комментариев* self.TableCommentUp()  
 *# Кнопки* self.Comment.clicked.connect(self.CommentF)  
 self.com = CommentForm(self.sql, MyWindow)  
 self.Register.clicked.connect(self.RegistF)  
 self.reg = RegistForm(self.sql)  
 self.us = PrivilegForm(self.sql, user)  
 self.About.clicked.connect(self.AboutB)  
 self.Cabinet.clicked.connect(self.CabinetB)  
 self.Rent.clicked.connect(self.RentB)  
 self.Services.clicked.connect(self.ServicesB)  
 self.Contacts.clicked.connect(self.ContactsB)  
 self.Manual.clicked.connect(self.ManualB)

Далее мы создаем методы этого окна. Начнем с компонента Text. При нажатии кнопок в него вносится текст. Для этого напишем следующие методы.

**def** ManualB(self):  
 self.Text.setHtml(text.e)  
  
**def** ContactsB(self):  
 self.Text.setHtml(text.d)  
  
**def** ServicesB(self):  
 self.Text.setHtml(text.c)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

32

**ДП 09.02.03.17.19 ПЗ**

**def** CabinetB(self):  
 self.Text.setHtml(text.b)  
  
**def** AboutB(self):  
 self.Text.setHtml(text.a)

**def** RentB(self):  
 self.Text.clear()  
 self.Text.insertPlainText(**'У нас есть следующее оборудование для аренды - \n\n'**)  
 **for** i **in** self.sql.SelectOne(**'rent'**,**'nameIns'**):  
 self.Text.insertPlainText(i+**' - '**+str(self.sql.SelectOne(**'rent'**,**'price'**,**'nameIns'**,**"'"**+i+**"'"**)[0])+**'\n'**)  
 self.Text.insertPlainText(**'\n Оборудование арендуется на все время репетиции, не зависимо от количества часов.'**)  
   
 В последнем методе реализуется запрос к базе данных, через созданный ранее метод «Select One». Сначала выводится фиксированная информация, а после выполнения запроса добавляется оборудование с ценой. Далее создадим метод вывода таблицы комментариев на экран:

**def** TableCommentUp(self):  
 self.sgm = QtSql.QSqlQueryModel(parent=self.tableC)  
 self.sgm.setQuery(**"select \* from comments order by ID desc"**)  
 self.sgm.setHeaderData(1, QtCore.Qt.Horizontal, **'Логин'**)  
 self.sgm.setHeaderData(2, QtCore.Qt.Horizontal, **'Комментарий'**)  
 self.tableC.setModel(self.sgm)  
 self.tableC.hideColumn(0)  
 self.tableC.resizeRowsToContents()  
 self.tableC.setColumnWidth(2, 166)

Тут обращение к методам класса SqlCom не происходит, из-за особенностей запроса макета. Далее опишем метод открытия диалоговых окон:

**def** RegistF(self):  
 self.reg.show()  
  
**def** CommentF(self):  
 self.com.show()

После этого создадим метод перехода в форму привилегированного режима, для этого напишем следующий код:

**def** EnterUs(self):  
 log = self.Login.text()  
 pas = self.Password.text()  
 self.query = QtSql.QSqlQuery()  
 self.query.exec(  
 **"select login,status from users where login="** + **"'"** + log + **"'"** + **" and password="** + **"'"** + pas + **"'"**)  
 **if** self.query.isActive():  
 self.query.first()  
 **if** self.query.isValid():  
 user.append(self.query.value(**'login'**))

Изм.

Лист

№ докум.

Подпись

Дата

Лист

33

**ДП 09.02.03.17.19 ПЗ**

user.append(self.query.value(**'status'**))  
 self.query.finish()  
 self.Password.clear()  
 **if** user[1]!=**'Не активен'**:  
 self.us.show()  
 self.us.start()  
 **else**:  
 user.clear()  
 self.Login.setText(**'Не Активирован'**)  
 **else**:  
 self.Login.setText(**'error'**)

Тут программа читает введенный пользователем логин и пароль, и ищет совпадения с таблицей «users». Если совпадение есть, то проверяется статус пользователя, если пользователь не активирован, то переход не происходит, а в edit логина выводится соответствующее сообщение.

Далее пропишем методы переключения фотографий и новостей. В них происходят запросы к таблицам базы данных, и выводятся записи по столбцам «ID». Пишем код:

**def** NewsB(self):  
 self.idNews -= 1  
 **if** self.idNews < 1:  
 self.idNews = self.idMax  
 self.News.setText(self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews)[0])  
 **else**:  
 self.News.setText(self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews)[0])  
  
**def** NewsN(self):   
 self.idNews += 1  
 **if not** self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews):  
 self.idNews = 1  
 self.News.setText(self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews)[0])  
 **else**:  
 self.News.setText(self.sql.SelectOne(**'news\_admin'**, **'news'**, **'ID'**, self.idNews)[0])

Изм.

Лист

№ докум.

Подпись

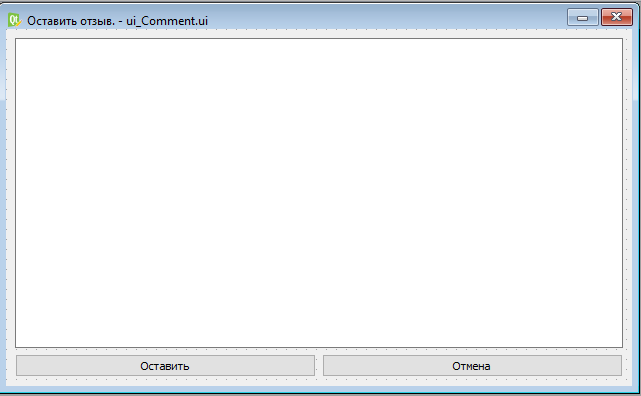
Дата

Лист

34

**ДП 09.02.03.17.19 ПЗ**

На этом создание главного окна законченно. Далее опишем диалоговое окно «Оставить отзыв», для этого снова воспользуемся QT Designer.



Изм.

Лист

№ докум.

Подпись

Дата

Лист

35

**ДП 09.02.03.17.19 ПЗ**

Рисунок 19. Создание диалог. окна «Оставить отзыв».

Далее создадим класс CommentForm и опишем его параметры и метод. Для этого пишем:

**class** CommentForm(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(CommentForm, self).\_\_init\_\_()  
 uic.loadUi(**"ui\_Comment.ui"**, self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.CommentEnter.clicked.connect(self.ComEn)  
 self.CommentExit.clicked.connect(self.close)  
 self.sql = sql  
  
 **def** ComEn(self):  
 **if** len(user) == 0:  
 self.sql.Insert(**'comments'**, **'Гость'**, self.textEdit.toPlainText())  
 window.TableCommentUp()  
 self.close()  
 **else**:  
 self.sql.Insert(**'comments'**, user[0], self.textEdit.toPlainText())  
 window.TableCommentUp()  
 self.close()

В единственном методе класса происходит проверка, в случае если пользователь вошел в систему, то комментарий записывается в таблицу с припиской логина пользователя, а иначе с припиской «Гость». Далее создадим диалоговое окно Регистрации.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

36

**ДП 09.02.03.17.19 ПЗ**

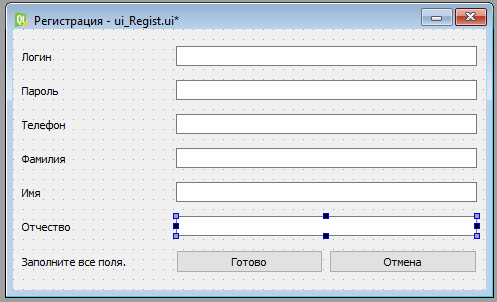


Рисунок 20. Создание диалог. окна «Регистрация».

Далее создадим класс RegistForm и опишем его параметры и методы.

**class** RegistForm(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(RegistForm, self).\_\_init\_\_()  
 uic.loadUi(**"ui\_Regist.ui"**, self)  
 self.sql = sql  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.Enter.clicked.connect(self.EnterReg)  
 self.Cancel.clicked.connect(self.close)  
  
 **def** EnterReg(self):  
 **if** (  
 self.login.text() **and** self.password.text() **and** self.f.text() **and** self.i.text() **and** self.o.text() **and** self.phone.text()) != **''**:  
 self.sql.Insert(**'users'**, self.login.text(), self.password.text(), self.f.text(), self.i.text(),  
 self.o.text(), self.phone.text(), **'Не активен'**)  
 self.close()  
 **else**:  
 self.ErrorL.setText(**'Все поля должны быть заполнены.'**)

В методе класса считываются введенные пользователем данные и заносятся с помощью метода Insert класса SqlCom в базу данных. В случае если поля небыли заполнены в Label ErrorL выводится ошибка.

Далее создадим окно привилегированного режима для этого снова воспользуемся QT Designer.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

37

**ДП 09.02.03.17.19 ПЗ**

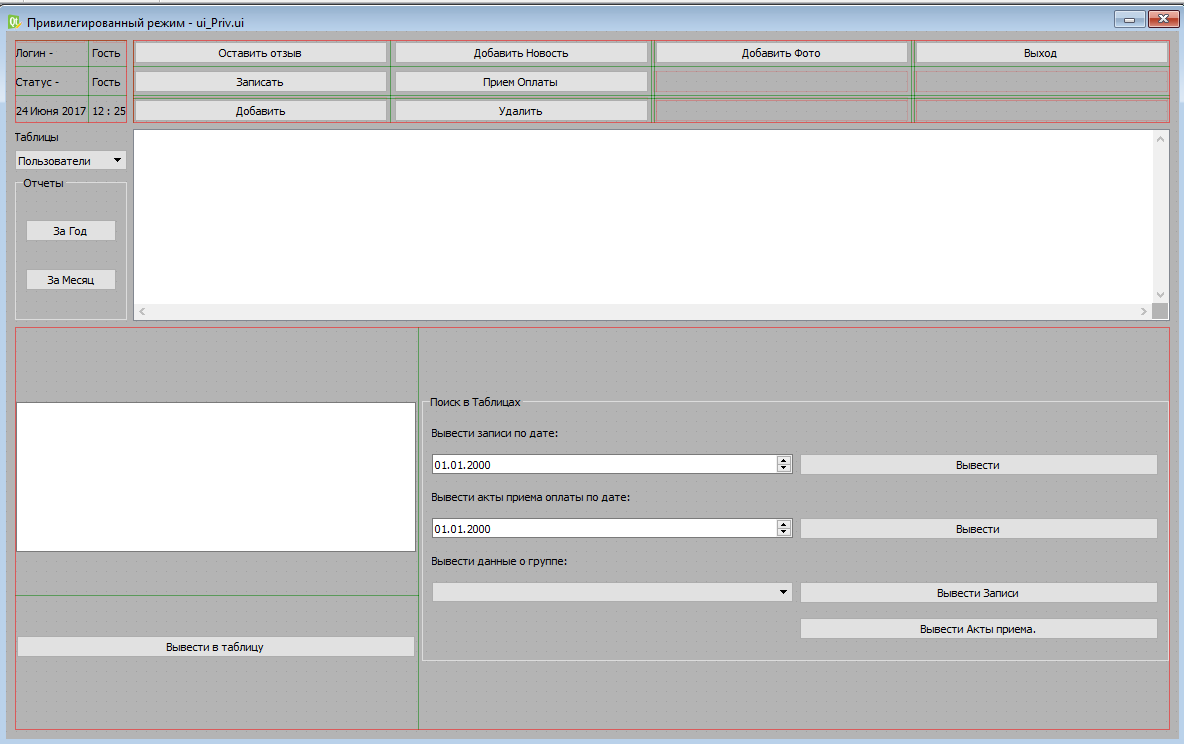


Рисунок 21. Создание окна Привилегированного режима.

Теперь снова в командной строке воспользуемся командой pyuic5.bat, для перевода созданного окна в модуль языка программирования Python. После этого создадим класс PrivilegForm в нашей программе. И в следующем коде опишем параметры окна:

**class** PrivilegForm(QtWidgets.QWidget, Priv.Ui\_FormPriv):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(PrivilegForm, self).\_\_init\_\_()  
 self.sql = sql  
 self.setupUi(self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.Comment.clicked.connect(self.UsComment)  
 self.com = **''** self.Exit.clicked.connect(self.close)  
 self.NewsAdminB.clicked.connect(self.newsAdminText)  
 self.newsAd = NewsAdminForm(self.sql, PrivilegForm)  
 self.Box.activated.connect(self.tableUp)  
 self.addB.clicked.connect(self.addBut)  
 self.deleteB.clicked.connect(self.delBut)  
 self.newPhoto.clicked.connect(self.newPh)  
 self.time()  
 self.RecordB.clicked.connect(self.addRec)  
 self.record = RecordBand(self.sql)  
 self.pay = PayForm(self.sql)  
 self.PayB.clicked.connect(self.addPay)  
 self.DayB.clicked.connect(self.FilterDay)  
 self.EnterRecB.clicked.connect(self.EntRec)  
 self.EnterPayB.clicked.connect(self.EntPay)  
 self.EnterRec.clicked.connect(self.EntRecD)  
 self.EnterPay.clicked.connect(self.EntPayD)  
 self.reportY = ReportYear(self.sql)  
 self.yearB.clicked.connect(self.year)  
 self.reportM = ReportMount(self.sql)  
 self.mountB.clicked.connect(self.mount)

Далее опишем методы открытия диалоговых окон: годовой отчет, месячный отчет, запись, приемка, создать новость и оставить отзыв. Опишем эти методы:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

38

**ДП 09.02.03.17.19 ПЗ**

**def** mount(self):  
 self.reportM.show()  
 self.reportM.start()  
  
**def** year(self):  
 self.reportY.show()  
 self.reportY.start()

**def** addPay(self):  
 self.pay.show()  
 self.pay.Start()  
  
**def** addRec(self):  
 self.record.show()  
 self.record.addBand()

**def** UsComment(self):  
 window.CommentF()

**def** newsAdminText(self):  
 self.newsAd.show()

Эти методы открывают диалоговые окна, а методах mount, year и addPay вызываются методы start описанные в классах этих окон, а в методе addRec метод addBand который так же описан в классе этого окна.

Следующие четыре метода которые мы должны создать идентичны. С помощью них осуществляется поиск информации в таблицах базы данных, с помощью методов класса SqlCom STMR и STM.

**def** EntPayD(self):  
 self.Table.setModel(self.sql.STMR(**'pay'**, **'date'**, self.PayDate.text()))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
  
**def** EntRecD(self):  
 self.Table.setModel(self.sql.STMR(**'sing\_up'**, **'date'**, self.RecDate.text()))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
  
**def** EntPay(self):  
 self.Table.setModel(self.sql.STMR(**'pay'**, **'id\_nameB'**, str(  
 self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**, **"'"** + self.BandBox.currentText() + **"'"**)[0])))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
  
**def** EntRec(self):  
 self.Table.setModel(self.sql.STMR(**'sing\_up'**, **'id\_nameB'**, str(  
 self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**, **"'"** + self.BandBox.currentText() + **"'"**)[0])))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

39

**ДП 09.02.03.17.19 ПЗ**

Особенностью этих методов является создание макета таблицы вызванных методом STMR класса SqlCom. Далее опишем вывод таблицы «sing\_up» с фильтрацией по текущему дню:

**def** FilterDay(self):  
 self.Table.setModel(self.sql.STMR(**'sing\_up'**, **'date'**, datetime.datetime.now().strftime(**"%d.%m.%Y"**)))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)

Здесь с помощью библиотеки datatime мы создаем макет таблицы с фильтрацией по текущему дню. Следующий метод, который мы опишем это вывод в компоненты типа Label, текущую дату и время:

**def** time(self):  
 dtime = datetime.datetime.now()  
 self.DateL.setText(str(dtime.day) + **'/'** + str(dtime.month) + **'/'** + str(dtime.year))  
 self.TimeL.setText(str(dtime.hour) + **':'** + str(dtime.minute))

Здесь как и в прошлом методе используется библиотека datatime, с помощью которой мы вносим в компоненты DataL и TimeL текущую дату и время. Далее создадим диалоговое окно «Добавление фотографий». Для этого напишем код:

**def** newPh(self):  
 file = QtWidgets.QFileDialog.getOpenFileName(caption=**'Добавить Фото'**)  
 **if** file[0] != **''**:  
 shutil.copy(file[0], **"photo"**)  
 self.sql.Insert(**'photo'**, os.path.basename(file[0]))  
 os.path.basename(file[0])

Здесь мы воспользуемся классом QFileDialog встроенным в библиотеку PyQt5, для создания диалогового окна типа поиска фалов. Полученный файл он программа копирует в папку photo в директории приложения, а название картинки добавляет в базу данных.

Далее опишем кнопки «Добавить» и «Удалить». Они должны удалять и

удалять записи в таблице базы данных.

**def** delBut(self):  
 self.sql.stmR.removeRow(self.Table.currentIndex().row())  
 self.sql.stmR.select()  
 self.sql.stm.removeRow(self.Table.currentIndex().row())  
 self.sql.stm.select()  
  
**def** addBut(self):  
 self.sql.stm.insertRow(self.sql.stm.rowCount())  
 self.sql.stmR.insertRow(self.sql.stmR.rowCount())

Изм.

Лист

№ докум.

Подпись

Дата

Лист

40

**ДП 09.02.03.17.19 ПЗ**

В этих методах используются и макет со связями, и обычный макет. Вызов их обоих сразу не нарушает работу приложения, так как в один момент используется только один из них, а второй не подключен к таблице. Далее опишем метод вывода таблиц баз данных на экран с помощью компонента Table.

**def** tableUp(self):  
 **if** self.Box.currentText() == **'Группы'**:  
 self.Table.setModel(self.sql.STM(**'bands'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Пользователи'**:  
 self.Table.setModel(self.sql.STM(**'users'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Коментарии'**:  
 self.Table.setModel(self.sql.STM(**'comments'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Аренда'**:  
 self.Table.setModel(self.sql.STM(**'rent'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Новости'**:  
 self.Table.setModel(self.sql.STM(**'news\_admin'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Фотографии'**:  
 self.Table.setModel(self.sql.STM(**'photo'**))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)  
 **if** self.Box.currentText() == **'Записи'**:  
 self.Table.setModel(self.sql.STMR(**'sing\_up'**))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(**True**)

**if** self.Box.currentText() == **'Приемка'**:  
 self.Table.setModel(self.sql.STMR(**'pay'**))  
 self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))  
 self.Table.hideColumn(0)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)  
 self.Table.setSortingEnabled(**True**)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

41

**ДП 09.02.03.17.19 ПЗ**

Этот метод вызывается каждый раз, когда пользователь выбирает из списка Box название таблицы. Метод считывает выбранный элемент их списка и выводит соответствующую таблицу в Table.

При закрытии окна должен происходить полноценный выход из системы. Для этого мы должны очистить глобальную переменную user. Опишем метод срабатывающий при закрытии окна.

**def** closeEvent(self, QCloseEvent):  
 self.close()  
 **global** user  
 user = []

Так же мы должны описать метод, который вносит стартовые значения в данное окно.

**def** start(self):  
 self.Login.setText(user[0])  
 self.Status.setText(user[1])  
 self.Box.clear()  
 **if** user[1]==**'Администратор'**:  
 self.Box.addItem(**"Пользователи"**)  
 self.Box.addItem(**"Фотографии"**)  
 self.Box.addItem(**"Новости"**)  
 self.Box.addItem(**"Аренда"**)  
 self.Box.addItem(**"Приемка"**)  
 self.Box.addItem(**"Записи"**)  
 self.Box.addItem(**"Группы"**)  
 self.Box.addItem(**"Коментарии"**)  
 self.DayT.clear()  
 self.DayT.insertPlainText(datetime.datetime.now().strftime(**"%d.%m.%Y"**) + **' Сегоднишние репетиции: \n'**)  
 **for** i **in** self.sql.SelectOne(**'sing\_up'**, **'id\_nameB'**, **'date'**,  
 **"'"** + datetime.datetime.now().strftime(**"%d.%m.%Y"**) + **"'"**):  
 self.DayT.insertPlainText(self.sql.SelectOne(**'bands'**, **'nameB'**, **'ID'**, i)[0] + **'\n'**)  
 self.BandBox.addItems(self.sql.SelectOne(**'bands'**, **'nameB'**))

Этот метод вносит в компоненты Login и Status, значения, полученные при входе пользователем в систему, а после вносит в список таблиц, таблицы который позволяет добавить уровень доступа. Так же он добавляет в компонент DayT текущую дату и названия групп, которые записаны на текущий день.

Теперь создадим диалоговое окно «Запись на репетицию» с помощью Qt Designer.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

42

**ДП 09.02.03.17.19 ПЗ**

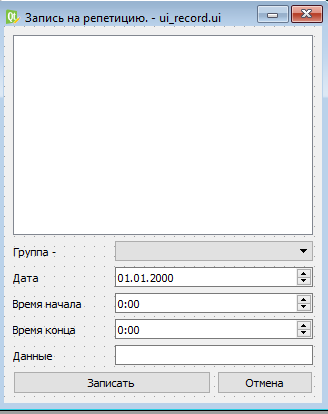


Рисунок 22. Создание диалогового окна «Запись на репетицию»

Далее мы создадим класс RecordBand, где опишем его параметры и методы.

**class** RecordBand(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(RecordBand, self).\_\_init\_\_()  
 self.sql = sql  
 uic.loadUi(**"ui\_record.ui"**, self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.Cancel.clicked.connect(self.close)  
 self.Record.clicked.connect(self.addRecord)  
 self.Date.dateChanged.connect(self.selectTable)

Теперь опишем методы этого класса. Первый из них будет метод вывода таблицы «sing\_up» с фильтром по дате, указанным в компоненте Date, и с выводом только столбцов time и timeEnd.

**def** selectTable(self):  
 self.sgm = QtSql.QSqlQueryModel(parent=self.Table)  
 self.sgm.setQuery(**"select time,timeEnd from sing\_up where date='"** + self.Date.text() + **"'"**)  
 self.sgm.setHeaderData(0, QtCore.Qt.Horizontal, **'Начало'**)  
 self.sgm.setHeaderData(1, QtCore.Qt.Horizontal, **'Конец'**)  
 self.Table.setModel(self.sgm)  
 self.Table.resizeRowsToContents()  
 self.Table.setColumnWidth(2, 166)

Далее создадим метод добавления записи в таблицу «sing\_up», информацией взятых с компонентов диалогового окна.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

43

**ДП 09.02.03.17.19 ПЗ**

**def** addRecord(self):  
 band = self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**, **"'"** + self.Band.currentText() + **"'"**)[0]  
 self.sql.Insert(**'sing\_up'**, band, self.Date.text(), self.Time.text(), self.TimeEnd.text(), self.Data.text())  
 self.close()

Добавим метод заполнения компонента Band списком названий групп из таблицы «bands».

**def** addBand(self):  
 self.Band.clear()  
 self.Band.addItems(self.sql.SelectOne(**'bands'**, **'nameB'**))  
 datePc = datetime.datetime.now()  
 self.Date.setDate(datePc.date())  
 self.selectTable()

Теперь создадим диалоговое окно «Прием оплаты», для этого снова воспользуемся Qt Designer.

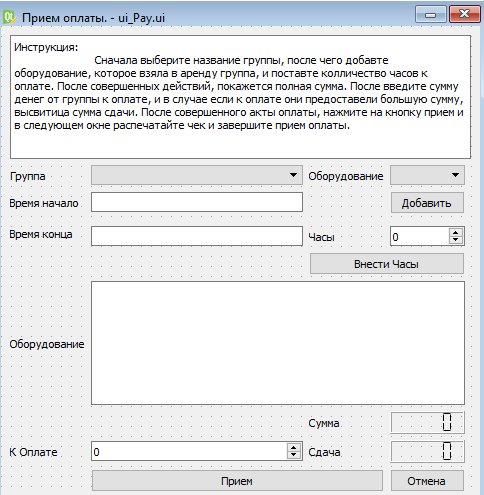


Рисунок 23. Создание диалогового окна «Прием оплаты».

Теперь создадим класс PayForm и опишем его параметры.

**class** PayForm(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(PayForm, self).\_\_init\_\_()  
 uic.loadUi(**"ui\_Pay.ui"**, self)  
 self.sql = sql  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.IntPay = 0  
 self.IntZ = 0  
 self.Cancel.clicked.connect(self.close)  
 self.Band.activated.connect(self.TimeTimeEnd)  
 self.Pay.setDecMode()  
 self.addRent.clicked.connect(self.addR)  
 self.addHour.clicked.connect(self.addH)  
 self.PayBand.editingFinished.connect(self.ReturnSum)  
 self.PayB.clicked.connect(self.addPay)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

44

**ДП 09.02.03.17.19 ПЗ**

Далее перейдем к описанию его методов. В начале добавим метод добавление записи в таблицу «pay» и печать чека об оплате репетиции.

**def** addPay(self):  
 self.sql.Insert(**'pay'**, self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**, **"'"** + self.Band.currentText() + **"'"**)[0],  
 datetime.datetime.now().strftime(**"%d.%m.%Y"**), self.Time.text(), self.TimeEnd.text(),  
 self.RentT.toPlainText(), self.Pay.intValue())  
 txt = QtGui.QTextDocument()  
 txt.setPlainText(  
 **'-'** \* 100 + **"\n Репетиционная студия '9/10'"** + **"\n Спасибо что провели у нас репетицию "** + self.Band.currentText() + **"\n"** + **'-'** \* 100 +  
 **'\n'** + str(datetime.datetime.now()) +  
 **'\n К оплате - \n Продолжительность репетиции - '** + self.Hour.text() + **' час/часов \n'** + **'(Стоимость часа - 200 руб)\n'** +  
 **'Арендованно оборудование: \n'** + self.RentT.toPlainText() + **'\n'** + **'-'** \* 100 +  
 **'\n Итог - '** + str(self.Pay.intValue()) + **'\n Внесено -'** + self.PayBand.text() + **'\n Сдача - '** + str(  
 self.Return.intValue()) +  
 **'\n Спасибо '** + self.Band.currentText() + **', приходите еще. \n'** + **'-'** \* 100)  
 pri = QtPrintSupport.QPrinter()  
 dia = QtPrintSupport.QPrintDialog(pri)  
 dia.exec()  
 txt.print\_(pri)  
 self.sql.Delete(**'sing\_up'**, self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**, **"'"** + self.Band.currentText() + **"'"**)[0],  
 datetime.datetime.now().strftime(**"%d.%m.%Y"**))  
 self.close()

Особенностью этого метода является использование класса библиотеки PyQt5 QPrinter. С помощью него мы отправляем созданный нами документ на печать. Далее опишем метод, который выполняет действия при открытии окна, такие как очистка компонентов, заполнение компонента Band списком названий групп, записанных на текущей день и заполнение компонента Rent списком названий оборудования.

**def** Start(self):   
 self.addHour.setEnabled(**True**)  
 self.addRent.setEnabled(**True**)  
 self.Pay.display(0)  
 self.Return.display(0)  
 self.Band.clear()  
 self.Rent.clear()  
 **for** i **in** self.sql.SelectOne(**'sing\_up'**, **'id\_nameB'**, **'date'**,  
 **"'"** + datetime.datetime.now().strftime(**"%d.%m.%Y"**) + **"'"**):  
 self.Band.addItems(self.sql.SelectOne(**'bands'**, **'nameB'**, **'ID'**, i))  
 self.Rent.addItems(self.sql.SelectOne(**'rent'**, **'nameIns'**))  
 self.Time.clear()  
 self.TimeEnd.clear()  
 self.RentT.clear()  
 self.PayBand.clear()  
 self.Hour.setValue(0)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

45

**ДП 09.02.03.17.19 ПЗ**

Далее опишем метод, который будет выполнятся при выборе названия группы из компонента Band и вносить в компоненты Time и TimeEnd время начала и конца репетиции, взятые из таблицы «sing\_up» по фильтру даты и названия группы.

**def** TimeTimeEnd(self):  
 self.Time.setText(self.sql.SelectOne(**'sing\_up'**, **'time'**, **'id\_nameB'**, self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**,  
 **"'"** + self.Band.currentText() + **"'"**)[  
 0])[0])  
 self.TimeEnd.setText(self.sql.SelectOne(**'sing\_up'**, **'timeEnd'**, **'id\_nameB'**,  
 self.sql.SelectOne(**'bands'**, **'ID'**, **'nameB'**,  
 **"'"** + self.Band.currentText() + **"'"**)[0])[0])

Теперь опишем метод добавления оборудования, который будет вносить название оборудования в компонент RentT и прибавлять их стоимость к общей сумме.

**def** addR(self):  
 self.RentT.insertPlainText(self.Rent.currentText() + **', '**)  
 self.Pay.display(self.sql.SelectOne(**'rent'**, **'price'**, **'nameIns'**, **"'"** + self.Rent.currentText() + **"'"**)[0])  
 self.Rent.removeItem(self.Rent.currentIndex())  
 **if** self.Rent.count() == 0:  
 self.addRent.setEnabled(**False**)

Далее метод, который будет добавлять к общей сумме стоимость часов, указанных в компоненте Hour.

**def** addH(self):  
 self.Pay.display(int(self.Hour.text()) \* 200)  
 self.addHour.setEnabled(**False**)

Последний метод, который мы должны описать, это метод, который должен срабатывать, при заполнении компонента PayBand. Он вычитает из этого компонента общую сумму, и передает остаток в компонент Return.

**def** ReturnSum(self):  
 self.Return.display(int(self.PayBand.text()) - self.Pay.intValue())

Далее создадим диалоговое окно «Оставить новость» с помощью Qt Designer.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

46

**ДП 09.02.03.17.19 ПЗ**

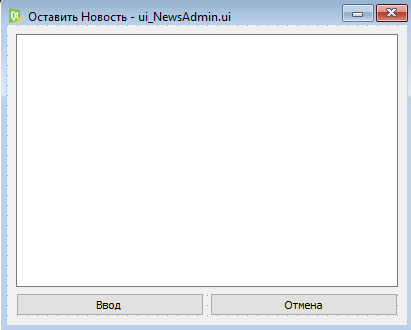


Рисунок 24. Создание диалогового окна «Оставить новость»

Далее создадим класс NewsAdminForm, и опишем его параметры.

**class** NewsAdminForm(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(QtWidgets.QWidget, self).\_\_init\_\_()  
 self.sql = sql  
 uic.loadUi(**"ui\_NewsAdmin.ui"**, self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.CancelB.clicked.connect(self.close)  
 self.EnterB.clicked.connect(self.enterNews)

Теперь опишем его единственный метод, который должен добавлять запись в таблицу «news\_admin».

**def** enterNews(self):  
 self.sql.Insert(**'news\_admin'**, self.NewsT.toPlainText())  
 self.close()

Теперь создадим диалоговое окно «Годовой отчет», снова воспользовавшись Qt Designer.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

47

**ДП 09.02.03.17.19 ПЗ**

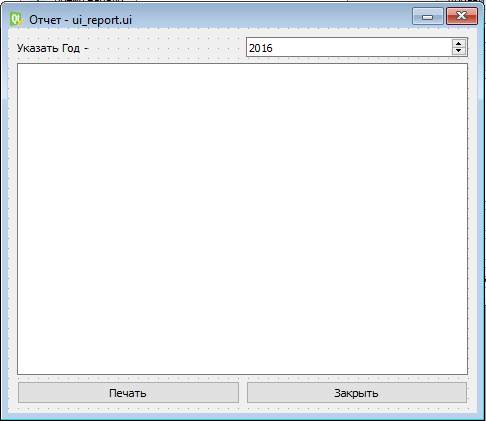


Рисунок 25. Создание диалогового окна «Годовой отчет»

Теперь создадим класс ReportYear и опишем его параметры.

**class** ReportYear(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(ReportYear, self).\_\_init\_\_()  
 uic.loadUi(**"ui\_report.ui"**, self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.sql = sql  
 self.report = QtGui.QTextDocument()  
 self.YearEdit.editingFinished.connect(self.start)  
 self.Close.clicked.connect(self.close)  
 self.Print.clicked.connect(self.PrintR)

Далее опишем его методы. Первый метод должен создавать отчет по году указанном в компоненте YearEdit и выводить его в компонент Text.

**def** start(self):  
 self.Text.clear()  
 self.report.clear()  
 self.Text.insertPlainText(**'-'** \* 100 + **'\n Годовой отчет \n \n \n'**)  
 pr = 0  
 **for** i **in** range(13):  
 **if** i != 0:  
 sum = 0  
 **if** i < 10:  
 self.sql.query.exec(**"select \* from pay where date like '\_\_.0"** + str(i) + **"."** + str(self.YearEdit.text()) + **"'"**)  
 **else**:  
 self.sql.query.exec(  
 **"select \* from pay where date like '\_\_."** + str(i) + **"."** + str(self.YearEdit.text()) + **"'"**)  
 lst1 = []

**if** self.sql.query.isActive():  
 self.sql.query.first()  
 **while** self.sql.query.isValid():  
 lst1.append(self.sql.query.value(**'price'**))  
 self.sql.query.next()  
 **for** f **in** lst1:  
 sum += f  
 pr += sum  
 self.Text.insertPlainText(str(i) + **' Месяц - '** + str(sum) + **' руб. \n'**)  
 self.Text.insertPlainText(**'-'** \* 100 + **'\n Годовая прибыль -> '** + str(pr) + **'\n'** + **'-'** \* 100)  
 self.report.setPlainText(self.Text.toPlainText())

Изм.

Лист

№ докум.

Подпись

Дата

Лист

48

**ДП 09.02.03.17.19 ПЗ**

Теперь опишем метод, который будет выполнять печать созданного отчета.

**def** PrintR(self):  
 pri = QtPrintSupport.QPrinter()  
 dia = QtPrintSupport.QPrintDialog(pri)  
 dia.exec()  
 self.report.print\_(pri)

Далее создадим последнее диалоговое окно «Месячный отчет», снова воспользовавшись Qt Designer.

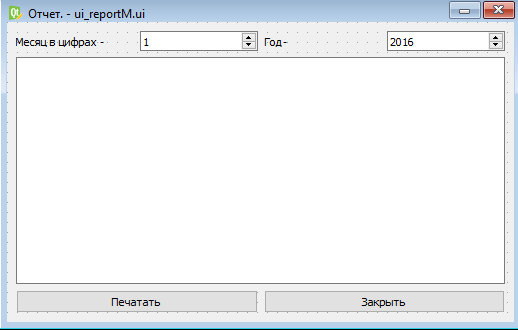


Рисунок 26. Создание диалогового окна «Месячный отчет»

Далее мы создадим класс ReportMount и внесем в него параметры

**class** ReportMount(QtWidgets.QWidget):  
 **def** \_\_init\_\_(self, sql, parent=**None**):  
 super(ReportMount, self).\_\_init\_\_()  
 uic.loadUi(**"ui\_reportM.ui"**, self)  
 self.setWindowModality(QtCore.Qt.ApplicationModal)  
 self.sql = sql  
 self.report = QtGui.QTextDocument()  
 self.Mount.editingFinished.connect(self.start)  
 self.Year.editingFinished.connect(self.start)  
 self.Close.clicked.connect(self.close)

self.Print.clicked.connect(self.PrintR)

Теперь опишем методы этого класса. Первый метод, будет создавать месячный отчет по указанному месяцу в компоненте Mount и по году указанному в компоненте Year.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

49

**ДП 09.02.03.17.19 ПЗ**

self.Text.clear()  
self.report.clear()  
self.Text.insertPlainText(**'-'** \* 100 + **'\n Отчет за месяц \n \n \n'** + **'Группа | Дата | Сумма \n'**)  
pr = 0  
lst = []  
**if** int(self.Mount.text()) < 10:  
 self.sql.query.exec(  
 **"select \* from pay where date like '\_\_.0"** + str(self.Mount.text()) + **"."** + str(self.Year.text()) + **"'"**)  
**else**:  
 self.sql.query.exec(  
 **"select \* from pay where date like '\_\_."** + str(self.Mount.text()) + **"."** + str(self.Year.text()) + **"'"**)  
**if** self.sql.query.isActive():  
 self.sql.query.first()  
 **while** self.sql.query.isValid():  
 lst.append(self.sql.query.value(**'id\_nameB'**))  
 lst.append(self.sql.query.value(**'date'**))  
 lst.append(self.sql.query.value(**'price'**))  
 self.sql.query.next()  
con = 0  
**for** i **in** lst:  
 **if** con == 0:  
 self.Text.insertPlainText(self.sql.SelectOne(**'bands'**, **'nameB'**, **'ID'**, **"'"** + str(i) + **"'"**)[0] + **' '**)  
 con += 1  
 **elif** con == 1:  
 self.Text.insertPlainText(str(i) + **' '**)  
 con += 1  
 **elif** con == 2:  
 pr += i  
 self.Text.insertPlainText(str(i) + **'\n'**)  
 con = 0  
self.Text.insertPlainText(**'-'** \* 100 + **'\n Прибыль -> '** + str(pr) + **'\n'** + **'-'** \* 100)  
self.report.setPlainText(self.Text.toPlainText())

Далее опишем метод печать созданного документа.

**def** PrintR(self):  
 pri = QtPrintSupport.QPrinter()  
 dia = QtPrintSupport.QPrintDialog(pri)  
 dia.exec()  
 self.report.print\_(pri)

Теперь для полноценной работы приложения создадим второй поток, который каждые 7 секунд будет переключать фотографии и новости в главном окне, а также изменять текущую дату и время в окне привилегированного режима. Для этого создадим класс Thread и опишем его параметры и метод работы.

**class** Thread(QtCore.QThread):  
 sig = QtCore.pyqtSignal()  
  
 **def** \_\_init\_\_(self, parent=**None**):  
 QtCore.QThread.\_\_init\_\_(self, parent)  
  
 **def** run(self):  
 **while True**:  
 time.sleep(10)  
 self.sig.emit()

**3.5. Создание интерфейса программного продукта. Написание программного кода по разработанному алгоритму**

Для создания интерфейса я пользовался следующими компонентами для форм.

1. На форме **MyWindow**:

QLineEdit, QLabel, QPushButton, QTextEdit и QTableView;



Рисунок 27. Расположение объектов на форме MyWindow.

2) На форме **PrivilegForm**:

QPushButton, QLineEdit, QTextEdit, QLabel, QGroupBox, QCombpBox, QDateEdit, QTableView;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

50

**ДП 09.02.03.17.19 ПЗ**

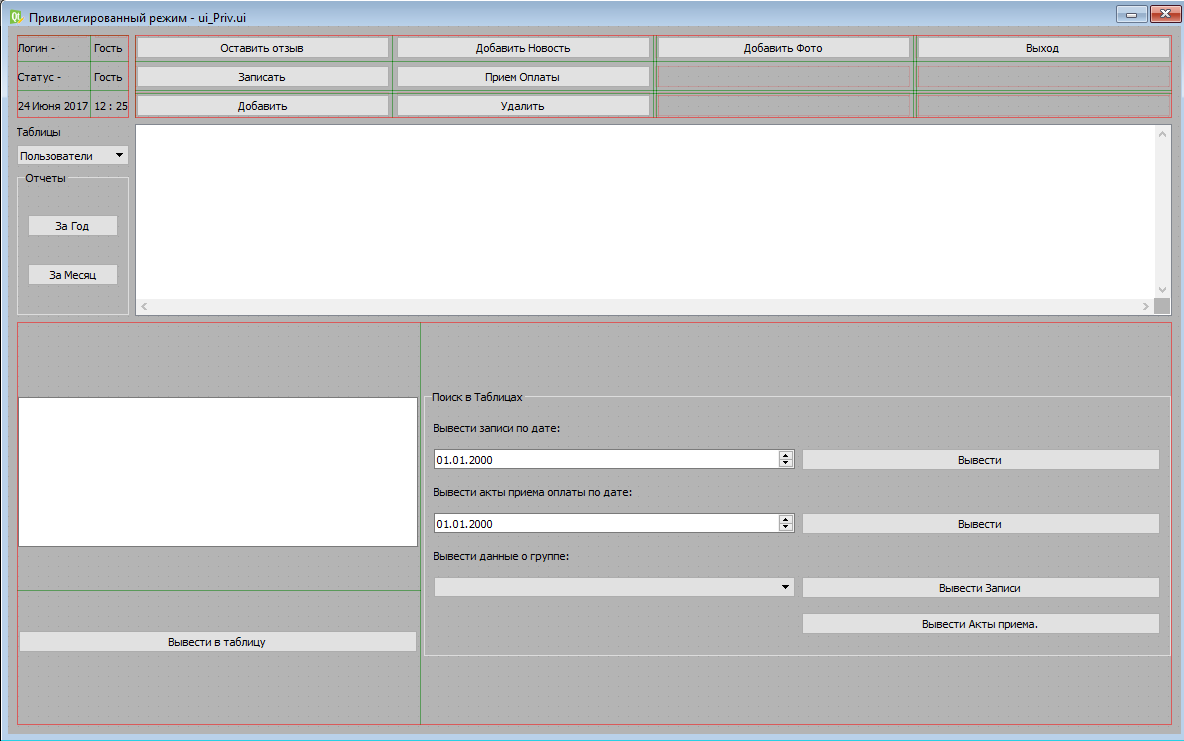


Рисунок 28. Расположение объектов на форме **PrivilegForm**.

3) В диалоговом окне RegistForm:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

51

**ДП 09.02.03.17.19 ПЗ**

QLineEdit, QLabel, QPushButton;

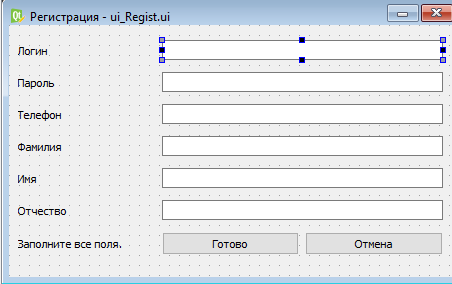


Рисунок 29. Расположение объектов в диалоговом окне RegistForm.

4) В диалоговом окне RecordBand:

QTableView, QLabel, QLineEdit, QDateEdit, QTimeEdit, QPushButton;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

52

**ДП 09.02.03.17.19 ПЗ**

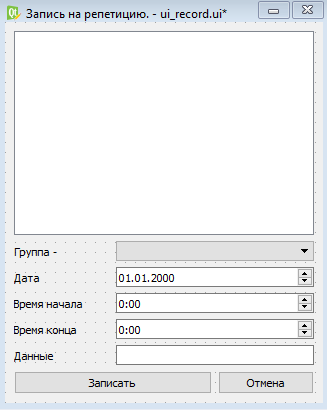


Рисунок 30. Расположение объектов на форме RecordBand.

5) В диалоговом окне PayForm:

QTextEdit, QComboBox, QPushButton, QSpinBox, QLCDNumber, QLabel;

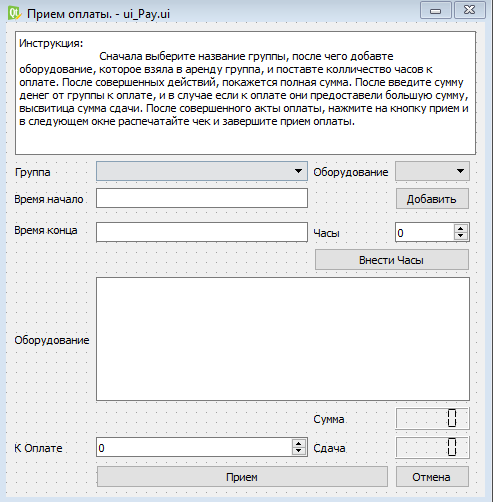


Рисунок 31. Расположение объектов в диалоговом окне PayForm.

6) В диалоговом окне ReportYear:

QLabel, QSpinBox, QTextEdit, QPushButton;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

53

**ДП 09.02.03.17.19 ПЗ**

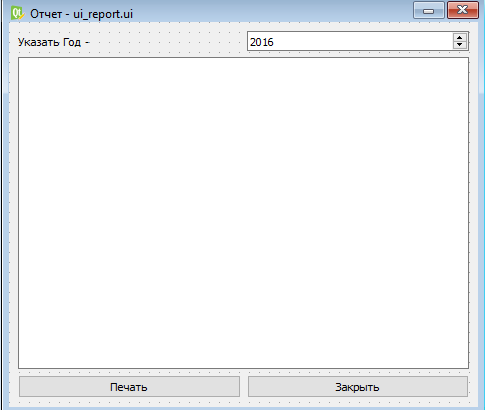
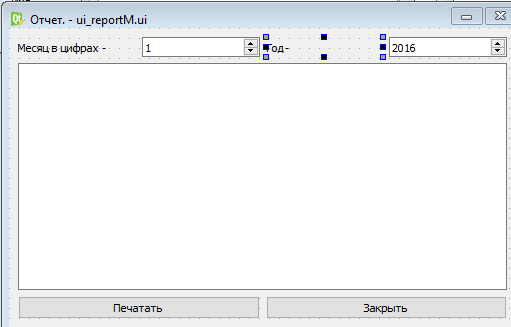


Рисунок 32. Расположение объектов в диалоговом окне ReportYear.

7) В диалоговом окне ReportMount:

QPushButton, QSpinBox, QTextEdit, QLabel;



Изм.

Лист

№ докум.

Подпись

Дата

Лист

54

**ДП 09.02.03.17.19 ПЗ**

Рисунок 33. Расположение объектов в диалоговом окне ReportMount.

7) В диалоговом окне NewsAdmin:

QTextEdit, QPushButton;

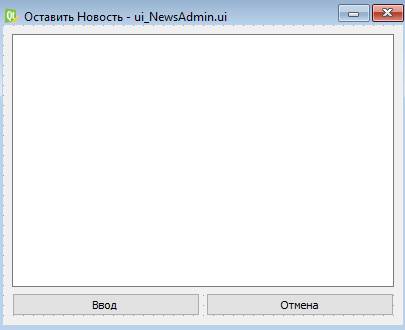


Рисунок 34. Расположение объектов на форме NewsAdmin.

8) В диалоговом окне CommentForm:

QTextEdit, QPushButton;

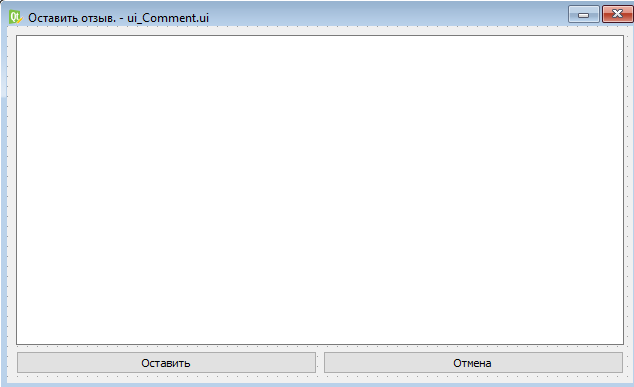


Рисунок 35. Расположение объектов на форме CommentForm.

Изм.

Лист

№ докум.

Подпись

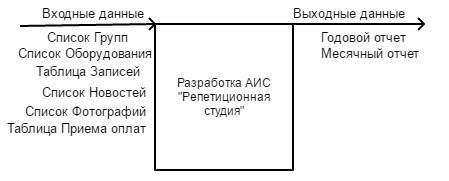
Дата

Лист

55

**ДП 09.02.03.17.19 ПЗ**

**3.6.Создание демонстрационных исходных данных, позволяющих протестировать разрабатываемый программный комплекс во всех режимах**



1) Группы.



Рисунок 36. Список Групп.

2) Записи



Рисунок 37. Список Записей.

1. Оборудование

Изм.

Лист

№ докум.

Подпись

Дата

Лист

**ДП 09.02.03.17.19 ПЗ**



Рисунок 38. Список оборудования.

4) Новости



Рисунок 39. Список новостей.

5) Фотографии



Рисунок 40. Список фотографий.

1. Прием оплаты

Изм.

Лист

№ докум.

Подпись

Дата

Лист

56

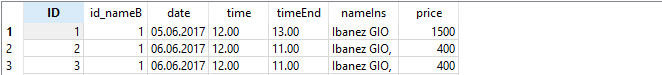


Рисунок 41. Таблица прием оплаты.

**3.7. Написание инструкций:**

**Оператору по инсталляции программного комплекса;**

1. Открыть установочный носитель

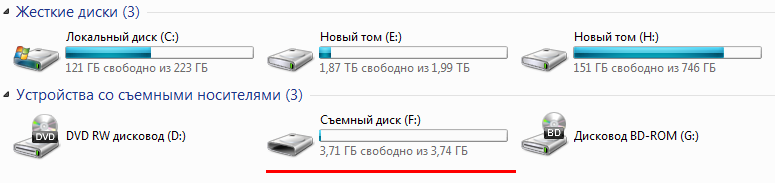
****

Рисунок 42. Открытие установочного носителя.

1. Скопировать папку с программой на жесткий диск компьютера
2. Открыть папку, после этого в папке «программное обеспечение» установть на компьютер Python и PyQt5, запустив соответствующие установочные файлы.
3. В папке с программой запускаем файла «program.py», через IDLE Python.

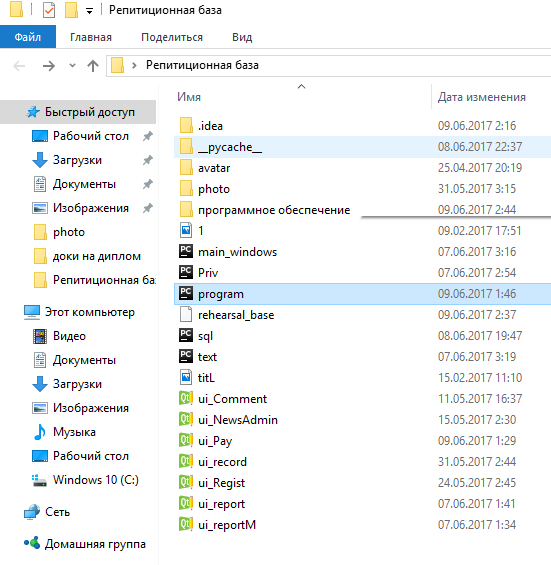


Рисунок 43. Каталог программы.

После запуска мы видем главное окно программы. В нем можно посмотреть слайдшоу фотографий и новостей. Почитать информацию о репетиционной базе, нажав на соответствующие кнопки. Оставить отзыв, нажав на соответствующую кнопку.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

57

**ДП 09.02.03.17.19 ПЗ**



Рисунок 44 Главная форма.

Далее чтобы работать с программой вы должны пройти процедуру регистрации, нажав на соответствующую кнопку. Здесь вы должны заполнить все поля и нажать на кнопку «Зарегистрировать». После чего вы можете войти в систему.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

58

**ДП 09.02.03.17.19 ПЗ**

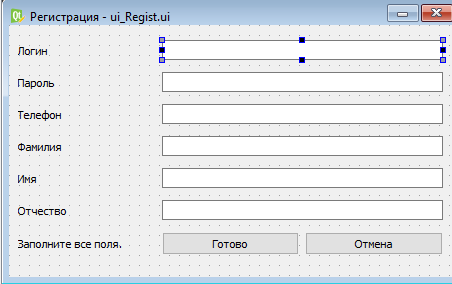


Рисунок 45. Диалоговое окно регистрации.

Для того чтобы войти в систему, вы должны ввести свой логин и пароль, в соответствующие поля в главном окне и нажать ввод. В случае если вас не активировала администрация, то в поле Логин вы увидете надпись – Не активирован.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

59

**ДП 09.02.03.17.19 ПЗ**

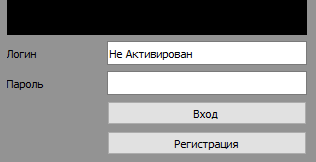
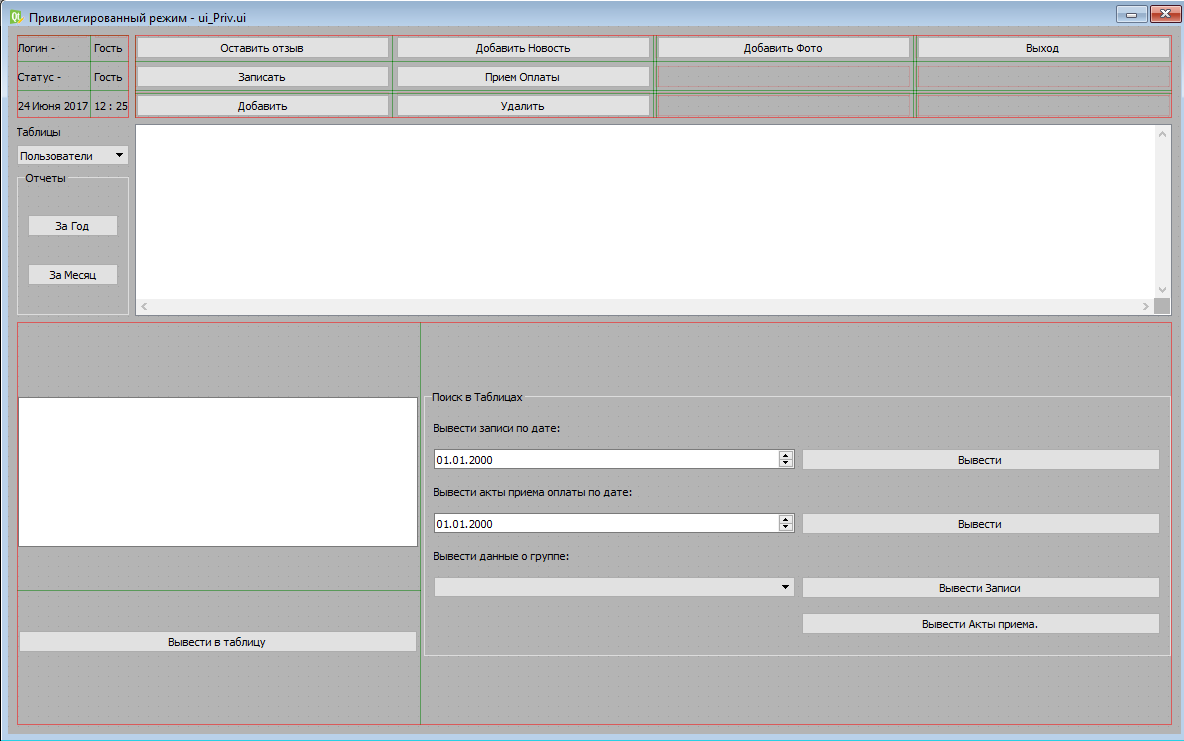


Рисунок 46. Вывод ошибки «Не активированный пользователь»

После активации вы заходите в привилегированный режим. Здесь вы можете просматривать таблицы выбрав их в списке таблиц, произвадить поиск по таблицам, добавлять и удалять записи в таблицах.

Рисунок 47. Привилегированный режим.

Для того чтобы записать группу на репетицию, нажимаем кнопку «Запись». Открывается диалоговое окно. В таблице указываются промежутки времени занятые другими группами в день указанным в поле «дата». После выбора группы и даты, вводим часы начала и конца репетиции, в свободный промежуток времени и нажимаем кнопку «Записать».

Изм.

Лист

№ докум.

Подпись

Дата

Лист

60

**ДП 09.02.03.17.19 ПЗ**

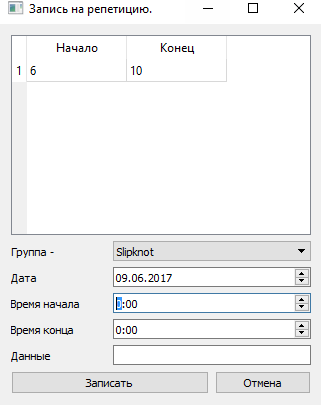


Рисунок 48. Запись группы на репетицию.

Что бы принять оплату у группы в окне привелегированного режима нажмити на кнопку «Приемка». Откроется окно. В списке групп, указанны названия груп репетирующих в этот день. Выберите группу и поля время начала и время конца заполнятся автоматически. После выбераем арендованное оборудование в списке и нажимаем кнопку добавить чтобы внести его в оплату. Далее указываем колличество часов, которое группа репетировала (не заполняется автоматически, для того чтобы на группу наложить штраф или долг), и нажимаем кнопку внести. Далее мы видем на экране сколько стоит прошедшая репетиция, после чего мы вносим сумму предоставленную группой и на экранне появится сумма сдачи. При нажатие кнопки «Прием», запись сохраняется и открывается печать чека об оплате.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

61

**ДП 09.02.03.17.19 ПЗ**

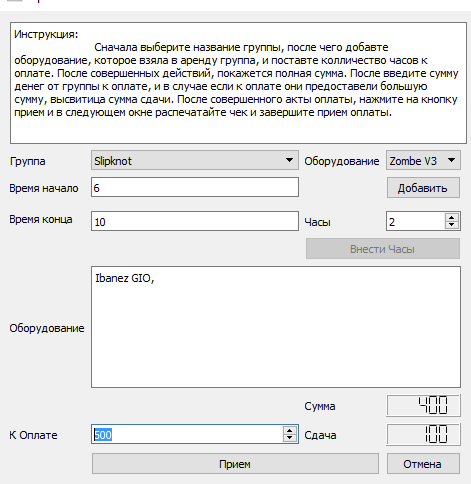


Рисунок 49. Прием оплаты.

Чтобы просмотреть годовой и месячный отчет, нажмите на соответствующие кнопки в привелегированном режиме. Откроются окна, в них указывается год и месяц за который нужно вывести отчет. Отчет автоматически заполнится после ввода параметров. После этого можно нажать кнопку «Печать» и документ отправится на печать.

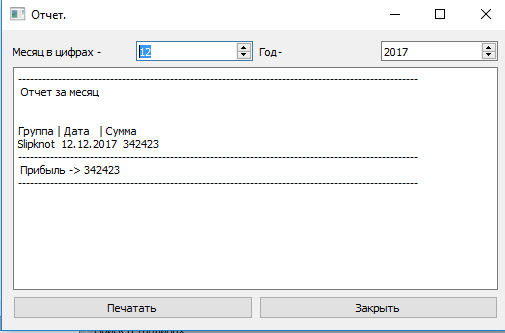


Рисунок 50. Месячный отчет.

**Глава 4. Техника безопасности при работе с программным продуктом**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

62

**ДП 09.02.03.17.19 ПЗ**

1. Разместите монитор так, чтобы его верхняя точка находилась прямо перед вашими глазами или выше, что позволит держать голову прямо, и исключит развитие шейного остеохондроза. Расстояние от монитора до глаз должно быть не меньше 45 см.

2. Стул должен иметь спинку и подлокотники, а так же такую высоту, при которой ноги могут прочно стоять на полу. Идеальным будет приобретение кресла с регулируемой высотой, в таком случае спинка позволит держать спину прямо, подлокотники дадут возможность отдохнуть рукам, правильное положение ног не будет мешать кровообращению в них.

3. Расположение часто используемых вещей не должно приводить к долгому нахождению в какой-либо искривлённой позе.

4. Освещение рабочего места не должно вызывать блики на экране монитора. Нельзя ставить монитор рядом с окном, так чтобы вы одновременно видели и экран и то, что находится за окном.

5. При работе с клавиатурой, угол сгиба руки в локте должен быть прямым (90 градусов).

6. При работе с мышкой кисть должна быть прямой, и лежать на столе как можно дальше от края.

**Заключение**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

63

**ДП 09.02.03.17.19 ПЗ**

В рамках дипломного проекта мной был разработан программный продукт «Автоматизированная информационная система «Репетиционная база». Данная программа упростит работу сотрудников репетиционной базы, сократит время работы, тем самым увеличив производительность труда. При разработке программного продукта был пройден полный цикл программы - от постановки задачи, до сдачи приложения в эксплуатацию.

При разработке программы я старался использовать как можно больше изученных средств, чтобы дать как можно более полную картину своих способностей. Мной были использованы такие приёмы, как: использование указателей, разделение потоков, динамическое выделение памяти, работа со сложными структурами (дерево), отладка, создание собственного компонента, создание базы данных со связями и другие.

Иными словами, за время дипломного проектирования я окончательно закрепил свои полученные за время учёбы и практики знания и теперь оправданно могу считать себя программистом.

**Список источников и литературы**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

64

**ДП 09.02.03.15.19 ПЗ**

1.Астахова И. Ф. и др. - SQL в примерах и задачах. Ч. 1. Мн: Новое знание, 2014. 176с.

2.Прохоренок Н.А., Дронов В.А. – Python3 и PyQt5. - Санкт-Петербург: Издательство «БХВ-Петербург», 2016. 832с.

3.Прохоренок Н.А., Дронов В.А. – Python 3 Самое Необходимое. Санкт-Питербург. Издательство «БХВ-Петербург», 2016, 461с.

4.Саммерфилд М. – Программирование на Python 3. Подробное руководство. – 2009, 608с.

5.Любанович Б. – Простой Python.Современный стиль программирования. – Санкт-Петербург, издательство «Питер», 2016. 476с.

6.Лутц М. – Python. Карманный справочник- Издательский дом «Вильямс» 2015, 313c.

7.Лутц М. – Программирование на Python. Том 1 – Издательство «Символ-Плюс», 2011, 976с.

8.Лутц М. – Изучаем Python – Издательство «Символ-Плюс», 2011, 1249с.

9.Бизли Д. – Подробный Справочник – Издательство «Символ-Плюс», 2010, 794с.

10.Хернандес М, Вьескас Д. – SQL запросы для простых смертных – Издательство «Лори», 2013, 458с.

11.Дунаев В.В. – Базы данных. Язык SQL для студентов – Издательство «БХВ-Петербург», 2006, 288с.

12.Лучано Рамальо – Python. К вершинам мастерства – Издательство «ДМК-Пресс», 2016, 768с.

13.Уэсли Дж. Чан – Python: создание приложений. Библиотека профессионала – Издательство «Вильямс», 2015, 816с.

14.Эл Свейгарт – Автоматизация рутинных задач с помощью Python. Практическое руководство для начинающих – Издательство «Вильямс», 2016, 592с.

15. Майкл МакГрат – Программирование на Python для начинающих- Издательство «Эксмо», 2017, 192с.

16.Лутц М. – Программирование на Python. Том 2. – Издательство «Символ-Плюс», 2011, 992с.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

65

**ДП 09.02.03.17.19 ПЗ**

17.Слаткин Б. – Секреты Python – Издательство «Вильямс», 2016, 272с.

18.Бьюли А. – Изучаем SQL – Издательство «Символ-Плюс», 2007, 312с.

19.Кригель А., Трухнов Б. – SQL Библия пользования – Издательство «Вильямс», 2010, 752с.

20)Рейтц К., Шлюччер Т. – Авто-стопом по Python- Издательство «Питер», 2017, 336с.

**Интернет – источники**

1) https://pythonworld.ru/

2) [https://www.python.org/](http://delphiexpert.ru/)

3) https://python.ru/

4) http://python-3.ru/

5) http://pep8.ru/

6) http://www.sqlite.org/

7) https://www.qt.io/

8) http://qt-doc.ru/

9) https://pypi.python.org/

10) https://pythonspot.com/

11) http://www.cyberforum.ru/

12) http://pyqt.sourceforge.net/

13) http://ru.wikihow.com/

14) http://pythontutor.ru/

15) https://tproger.ru/

**Приложение. Код Программы.**

**Модуль main\_windows.py –**

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'uiWIN.ui'

#

# Created by: PyQt5 UI code generator 5.6

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

# Класс создания окна с помощью метода setupUI и retranslateUi

def setupUi(self, MainWindow):

#Метод установки компонентов и настройка окна

MainWindow.setObjectName("MainWindow")

MainWindow.resize(1166, 842)

palette = QtGui.QPalette()

brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(147, 147, 147))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(147, 147, 147))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(147, 147, 147))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(147, 147, 147))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)

MainWindow.setPalette(palette)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)

self.gridLayout.setObjectName("gridLayout")

self.LayoutNorth = QtWidgets.QHBoxLayout()

self.LayoutNorth.setObjectName("LayoutNorth")

self.Pback = QtWidgets.QPushButton(self.centralwidget)

self.Pback.setMaximumSize(QtCore.QSize(30, 300))

self.Pback.setObjectName("Pback")

self.Pback.clicked.connect(self.PhotoB)

self.LayoutNorth.addWidget(self.Pback)

self.Photo = QtWidgets.QLabel(self.centralwidget)

self.Photo.setMaximumSize(QtCore.QSize(400, 300))

self.Photo.setText("")

self.Photo.setPixmap(QtGui.QPixmap("photo/1.jpg"))

self.Photo.setScaledContents(True)

self.Photo.setObjectName("Photo")

self.LayoutNorth.addWidget(self.Photo)

self.Pnext = QtWidgets.QPushButton(self.centralwidget)

self.Pnext.setMaximumSize(QtCore.QSize(30, 300))

self.Pnext.setObjectName("Pnext")

self.Pnext.clicked.connect(self.PhotoN)

self.LayoutNorth.addWidget(self.Pnext)

self.Nback = QtWidgets.QPushButton(self.centralwidget)

self.Nback.setMaximumSize(QtCore.QSize(30, 300))

self.Nback.setObjectName("Nback")

self.Nback.clicked.connect(self.NewsB)

self.LayoutNorth.addWidget(self.Nback)

self.News = QtWidgets.QTextEdit(self.centralwidget)

self.News.setMaximumSize(QtCore.QSize(16777215, 300))

self.News.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.News.setObjectName("News")

self.LayoutNorth.addWidget(self.News)

self.Nnext = QtWidgets.QPushButton(self.centralwidget)

self.Nnext.setMaximumSize(QtCore.QSize(30, 300))

self.Nnext.setObjectName("Nnext")

self.Nnext.clicked.connect(self.NewsN)

self.LayoutNorth.addWidget(self.Nnext)

self.gridLayout.addLayout(self.LayoutNorth, 0, 0, 2, 1)

self.Title = QtWidgets.QLabel(self.centralwidget)

self.Title.setMaximumSize(QtCore.QSize(300, 180))

self.Title.setText("")

self.Title.setPixmap(QtGui.QPixmap("titL.png"))

self.Title.setScaledContents(True)

self.Title.setWordWrap(False)

self.Title.setObjectName("Title")

self.gridLayout.addWidget(self.Title, 0, 1, 1, 2)

self.LayoutEast = QtWidgets.QGridLayout()

self.LayoutEast.setObjectName("LayoutEast")

self.Llogin = QtWidgets.QLabel(self.centralwidget)

self.Llogin.setMaximumSize(QtCore.QSize(100, 16777215))

self.Llogin.setObjectName("Llogin")

self.LayoutEast.addWidget(self.Llogin, 1, 0, 1, 1)

self.Login = QtWidgets.QLineEdit(self.centralwidget)

self.Login.setMaximumSize(QtCore.QSize(200, 30))

self.Login.setObjectName("Login")

self.LayoutEast.addWidget(self.Login, 1, 1, 1, 1)

self.Lpassword = QtWidgets.QLabel(self.centralwidget)

self.Lpassword.setMaximumSize(QtCore.QSize(100, 16777215))

self.Lpassword.setObjectName("Lpassword")

self.LayoutEast.addWidget(self.Lpassword, 2, 0, 1, 1)

self.Enter = QtWidgets.QPushButton(self.centralwidget)

self.Enter.setMaximumSize(QtCore.QSize(200, 30))

self.Enter.setObjectName("Enter")

self.Enter.clicked.connect(self.EnterUs)

self.LayoutEast.addWidget(self.Enter, 3, 1, 1, 1)

self.Password = QtWidgets.QLineEdit(self.centralwidget)

self.Password.setMaximumSize(QtCore.QSize(200, 30))

self.Password.setObjectName("Password")

self.Password.setEchoMode(QtWidgets.QLineEdit.Password)

self.LayoutEast.addWidget(self.Password, 2, 1, 1, 1)

self.Register = QtWidgets.QPushButton(self.centralwidget)

self.Register.setMaximumSize(QtCore.QSize(200, 30))

self.Register.setObjectName("Register")

self.LayoutEast.addWidget(self.Register, 4, 1, 1, 1)

self.gridLayout.addLayout(self.LayoutEast, 1, 1, 1, 2)

self.LayoutNorth2 = QtWidgets.QHBoxLayout()

self.LayoutNorth2.setObjectName("LayoutNorth2")

self.About = QtWidgets.QPushButton(self.centralwidget)

self.About.setObjectName("About")

self.LayoutNorth2.addWidget(self.About)

self.Cabinet = QtWidgets.QPushButton(self.centralwidget)

self.Cabinet.setObjectName("Cabinet")

self.LayoutNorth2.addWidget(self.Cabinet)

self.Rent = QtWidgets.QPushButton(self.centralwidget)

self.Rent.setObjectName("Rent")

self.LayoutNorth2.addWidget(self.Rent)

self.Services = QtWidgets.QPushButton(self.centralwidget)

self.Services.setObjectName("Services")

self.LayoutNorth2.addWidget(self.Services)

self.Contacts = QtWidgets.QPushButton(self.centralwidget)

self.Contacts.setObjectName("Contacts")

self.LayoutNorth2.addWidget(self.Contacts)

self.Manual = QtWidgets.QPushButton(self.centralwidget)

self.Manual.setMaximumSize(QtCore.QSize(16777215, 16777215))

self.Manual.setObjectName("Manual")

self.LayoutNorth2.addWidget(self.Manual)

self.gridLayout.addLayout(self.LayoutNorth2, 2, 0, 1, 1)

self.Comment = QtWidgets.QPushButton(self.centralwidget)

self.Comment.setMaximumSize(QtCore.QSize(300, 16777215))

self.Comment.setObjectName("Comment")

self.gridLayout.addWidget(self.Comment, 2, 1, 1, 1)

self.exitB = QtWidgets.QPushButton(self.centralwidget)

self.exitB.setObjectName("exitB")

self.exitB.clicked.connect(exit)

self.gridLayout.addWidget(self.exitB, 2, 2, 1, 1)

self.Text = QtWidgets.QTextEdit(self.centralwidget)

self.Text.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.Text.setObjectName("Text")

self.gridLayout.addWidget(self.Text, 3, 0, 1, 1)

self.tableC = QtWidgets.QTableView(self.centralwidget)

self.tableC.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.tableC.setObjectName("tableC")

self.gridLayout.addWidget(self.tableC, 3, 1, 1, 2)

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 1166, 21))

self.menubar.setObjectName("menubar")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

#Метод создания видемых имен и внесения текста

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.Pback.setText(\_translate("MainWindow", "<"))

self.Pnext.setText(\_translate("MainWindow", ">"))

self.Nback.setText(\_translate("MainWindow", "<"))

self.Nnext.setText(\_translate("MainWindow", ">"))

self.Llogin.setText(\_translate("MainWindow", "Логин"))

self.Lpassword.setText(\_translate("MainWindow", "Пароль"))

self.Enter.setText(\_translate("MainWindow", "Вход"))

self.Register.setText(\_translate("MainWindow", "Регистрация"))

self.About.setText(\_translate("MainWindow", "О Нас"))

self.Cabinet.setText(\_translate("MainWindow", "Помещение"))

self.Rent.setText(\_translate("MainWindow", "Аренда"))

self.Services.setText(\_translate("MainWindow", "Услуги"))

self.Contacts.setText(\_translate("MainWindow", "Контакты"))

self.Manual.setText(\_translate("MainWindow", "Инструкция"))

self.Comment.setText(\_translate("MainWindow", "Оставить отзыв"))

self.exitB.setText(\_translate("MainWindow", "Закрыть программу"))

**Модуль Priv.py –**

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'ui\_Priv.ui'

#

# Created by: PyQt5 UI code generator 5.6

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_FormPriv(object):

#Класс создания окна с помощью методов setupUI и retranslateUi

def setupUi(self, FormPriv):

#Метод создания компонентов и внесения параметров окна

FormPriv.setObjectName("FormPriv")

FormPriv.resize(1173, 708)

palette = QtGui.QPalette()

brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(180, 180, 180))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(180, 180, 180))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)

brush = QtGui.QBrush(QtGui.QColor(180, 180, 180))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)

brush = QtGui.QBrush(QtGui.QColor(180, 180, 180))

brush.setStyle(QtCore.Qt.SolidPattern)

palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)

FormPriv.setPalette(palette)

self.gridLayout\_4 = QtWidgets.QGridLayout(FormPriv)

self.gridLayout\_4.setObjectName("gridLayout\_4")

self.gridLayout = QtWidgets.QGridLayout()

self.gridLayout.setObjectName("gridLayout")

self.Login = QtWidgets.QLabel(FormPriv)

self.Login.setMaximumSize(QtCore.QSize(150, 16777215))

self.Login.setObjectName("Login")

self.gridLayout.addWidget(self.Login, 1, 1, 1, 1)

self.Status = QtWidgets.QLabel(FormPriv)

self.Status.setMaximumSize(QtCore.QSize(150, 16777215))

self.Status.setObjectName("Status")

self.gridLayout.addWidget(self.Status, 2, 1, 1, 1)

self.statusL = QtWidgets.QLabel(FormPriv)

self.statusL.setMaximumSize(QtCore.QSize(150, 16777215))

self.statusL.setObjectName("statusL")

self.gridLayout.addWidget(self.statusL, 2, 0, 1, 1)

self.LoginL = QtWidgets.QLabel(FormPriv)

self.LoginL.setMaximumSize(QtCore.QSize(150, 16777215))

self.LoginL.setObjectName("LoginL")

self.gridLayout.addWidget(self.LoginL, 1, 0, 1, 1)

self.DateL = QtWidgets.QLabel(FormPriv)

self.DateL.setObjectName("DateL")

self.gridLayout.addWidget(self.DateL, 3, 0, 1, 1)

self.TimeL = QtWidgets.QLabel(FormPriv)

self.TimeL.setObjectName("TimeL")

self.gridLayout.addWidget(self.TimeL, 3, 1, 1, 1)

self.gridLayout\_4.addLayout(self.gridLayout, 0, 0, 1, 1)

self.gridLayout\_2 = QtWidgets.QGridLayout()

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.Exit = QtWidgets.QPushButton(FormPriv)

self.Exit.setObjectName("Exit")

self.gridLayout\_2.addWidget(self.Exit, 0, 6, 1, 1)

self.NewsAdminB = QtWidgets.QPushButton(FormPriv)

self.NewsAdminB.setObjectName("NewsAdminB")

self.gridLayout\_2.addWidget(self.NewsAdminB, 0, 2, 1, 1)

self.Comment = QtWidgets.QPushButton(FormPriv)

self.Comment.setObjectName("Comment")

self.gridLayout\_2.addWidget(self.Comment, 0, 1, 1, 1)

self.newPhoto = QtWidgets.QPushButton(FormPriv)

self.newPhoto.setObjectName("newPhoto")

self.gridLayout\_2.addWidget(self.newPhoto, 0, 4, 1, 1)

self.deleteB = QtWidgets.QPushButton(FormPriv)

self.deleteB.setEnabled(True)

self.deleteB.setAutoDefault(False)

self.deleteB.setDefault(False)

self.deleteB.setFlat(False)

self.deleteB.setObjectName("deleteB")

self.gridLayout\_2.addWidget(self.deleteB, 3, 2, 1, 1)

self.addB = QtWidgets.QPushButton(FormPriv)

self.addB.setObjectName("addB")

self.gridLayout\_2.addWidget(self.addB, 3, 1, 1, 1)

self.PayB = QtWidgets.QPushButton(FormPriv)

self.PayB.setObjectName("PayB")

self.gridLayout\_2.addWidget(self.PayB, 1, 2, 1, 1)

self.RecordB = QtWidgets.QPushButton(FormPriv)

self.RecordB.setObjectName("RecordB")

self.gridLayout\_2.addWidget(self.RecordB, 1, 1, 1, 1)

self.gridLayout\_4.addLayout(self.gridLayout\_2, 0, 1, 1, 1)

self.Table = QtWidgets.QTableView(FormPriv)

self.Table.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.Table.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.Table.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustIgnored)

self.Table.setObjectName("Table")

self.gridLayout\_4.addWidget(self.Table, 2, 1, 4, 1)

self.gridLayout\_3 = QtWidgets.QGridLayout()

self.gridLayout\_3.setObjectName("gridLayout\_3")

self.DayT = QtWidgets.QTextEdit(FormPriv)

self.DayT.setMinimumSize(QtCore.QSize(150, 150))

self.DayT.setMaximumSize(QtCore.QSize(400, 150))

self.DayT.setObjectName("DayT")

self.gridLayout\_3.addWidget(self.DayT, 0, 0, 1, 1)

self.DayB = QtWidgets.QPushButton(FormPriv)

self.DayB.setObjectName("DayB")

self.gridLayout\_3.addWidget(self.DayB, 1, 0, 1, 1)

self.TabSea = QtWidgets.QGroupBox(FormPriv)

self.TabSea.setObjectName("TabSea")

self.gridLayout\_5 = QtWidgets.QGridLayout(self.TabSea)

self.gridLayout\_5.setObjectName("gridLayout\_5")

self.PayDate = QtWidgets.QDateEdit(self.TabSea)

self.PayDate.setObjectName("PayDate")

self.gridLayout\_5.addWidget(self.PayDate, 4, 0, 1, 1)

self.label\_2 = QtWidgets.QLabel(self.TabSea)

self.label\_2.setMaximumSize(QtCore.QSize(16777215, 15))

self.label\_2.setObjectName("label\_2")

self.gridLayout\_5.addWidget(self.label\_2, 3, 0, 1, 2)

self.label = QtWidgets.QLabel(self.TabSea)

self.label.setMaximumSize(QtCore.QSize(16777215, 15))

self.label.setObjectName("label")

self.gridLayout\_5.addWidget(self.label, 0, 0, 1, 2)

self.RecDate = QtWidgets.QDateEdit(self.TabSea)

self.RecDate.setObjectName("RecDate")

self.gridLayout\_5.addWidget(self.RecDate, 1, 0, 1, 1)

self.BandBox = QtWidgets.QComboBox(self.TabSea)

self.BandBox.setObjectName("BandBox")

self.gridLayout\_5.addWidget(self.BandBox, 7, 0, 1, 1)

self.EnterRecB = QtWidgets.QPushButton(self.TabSea)

self.EnterRecB.setObjectName("EnterRecB")

self.gridLayout\_5.addWidget(self.EnterRecB, 7, 1, 1, 1)

self.EnterPayB = QtWidgets.QPushButton(self.TabSea)

self.EnterPayB.setObjectName("EnterPayB")

self.gridLayout\_5.addWidget(self.EnterPayB, 8, 1, 1, 1)

self.label\_4 = QtWidgets.QLabel(self.TabSea)

self.label\_4.setMaximumSize(QtCore.QSize(16777215, 15))

self.label\_4.setObjectName("label\_4")

self.gridLayout\_5.addWidget(self.label\_4, 6, 0, 1, 2)

self.EnterPay = QtWidgets.QPushButton(self.TabSea)

self.EnterPay.setObjectName("EnterPay")

self.gridLayout\_5.addWidget(self.EnterPay, 4, 1, 1, 1)

self.EnterRec = QtWidgets.QPushButton(self.TabSea)

self.EnterRec.setObjectName("EnterRec")

self.gridLayout\_5.addWidget(self.EnterRec, 1, 1, 1, 1)

self.gridLayout\_3.addWidget(self.TabSea, 0, 1, 2, 1)

self.gridLayout\_4.addLayout(self.gridLayout\_3, 6, 0, 1, 2)

self.Box = QtWidgets.QComboBox(FormPriv)

self.Box.setEditable(False)

self.Box.setMinimumContentsLength(0)

self.Box.setDuplicatesEnabled(False)

self.Box.setObjectName("Box")

self.gridLayout\_4.addWidget(self.Box, 3, 0, 1, 1)

self.label\_3 = QtWidgets.QLabel(FormPriv)

self.label\_3.setMaximumSize(QtCore.QSize(16777215, 15))

self.label\_3.setObjectName("label\_3")

self.gridLayout\_4.addWidget(self.label\_3, 2, 0, 1, 1)

self.ReportBox = QtWidgets.QGroupBox(FormPriv)

self.ReportBox.setObjectName("ReportBox")

self.verticalLayout = QtWidgets.QVBoxLayout(self.ReportBox)

self.verticalLayout.setObjectName("verticalLayout")

self.yearB = QtWidgets.QPushButton(self.ReportBox)

self.yearB.setObjectName("yearB")

self.verticalLayout.addWidget(self.yearB)

self.mountB = QtWidgets.QPushButton(self.ReportBox)

self.mountB.setObjectName("mountB")

self.verticalLayout.addWidget(self.mountB)

self.gridLayout\_4.addWidget(self.ReportBox, 4, 0, 2, 1)

self.retranslateUi(FormPriv)

QtCore.QMetaObject.connectSlotsByName(FormPriv)

def retranslateUi(self, FormPriv):

#Метод объявления видимых имен компонентов и внесение в них текст

\_translate = QtCore.QCoreApplication.translate

FormPriv.setWindowTitle(\_translate("FormPriv", "Привилегированный режим"))

self.Login.setText(\_translate("FormPriv", "Гость"))

self.Status.setText(\_translate("FormPriv", "Гость"))

self.statusL.setText(\_translate("FormPriv", "Статус -"))

self.LoginL.setText(\_translate("FormPriv", "Логин -"))

self.DateL.setText(\_translate("FormPriv", "24 Июня 2017"))

self.TimeL.setText(\_translate("FormPriv", "12 : 25"))

self.Exit.setText(\_translate("FormPriv", "Выход"))

self.NewsAdminB.setText(\_translate("FormPriv", "Добавить Новость"))

self.Comment.setText(\_translate("FormPriv", "Оставить отзыв"))

self.newPhoto.setText(\_translate("FormPriv", "Добавить Фото"))

self.deleteB.setText(\_translate("FormPriv", "Удалить"))

self.addB.setText(\_translate("FormPriv", "Добавить"))

self.PayB.setText(\_translate("FormPriv", "Прием Оплаты"))

self.RecordB.setText(\_translate("FormPriv", "Записать"))

self.DayB.setText(\_translate("FormPriv", "Вывести в таблицу"))

self.TabSea.setTitle(\_translate("FormPriv", "Поиск в Таблицах"))

self.label\_2.setText(\_translate("FormPriv", "Вывести акты приема оплаты по дате:"))

self.label.setText(\_translate("FormPriv", "Вывести записи по дате:"))

self.EnterRecB.setText(\_translate("FormPriv", "Вывести Записи"))

self.EnterPayB.setText(\_translate("FormPriv", "Вывести Акты приема."))

self.label\_4.setText(\_translate("FormPriv", "Вывести данные о группе:"))

self.EnterPay.setText(\_translate("FormPriv", "Вывести"))

self.EnterRec.setText(\_translate("FormPriv", "Вывести"))

self.label\_3.setText(\_translate("FormPriv", "Таблицы "))

self.ReportBox.setTitle(\_translate("FormPriv", "Отчеты"))

self.yearB.setText(\_translate("FormPriv", "За Год"))

self.mountB.setText(\_translate("FormPriv", "За Месяц"))

**Модуль program.py –**

#Импорт системных библиотек

import sys, os

#Подключение библиотеки PyQt5 и используемые в ней модули

from PyQt5 import QtWidgets, QtCore, uic, QtGui, QtSql, QtPrintSupport

#Импорт библиотеки работы с файлами

import shutil

#Импорт модулей создания окна

import main\_windows, Priv

#импорт модулей базы данных и модуля с текстом для вывода в форме

import sql, text

#импорт библиотек для работы с датой и временем

import time, datetime

class Thread(QtCore.QThread):

#Класс подключения второго потока

sig = QtCore.pyqtSignal() # Создание сигнала

def \_\_init\_\_(self, parent=None):

# параметры класса наследуются из классса QThread

QtCore.QThread.\_\_init\_\_(self, parent)

def run(self):

#Метод запускающий цикл второго потока

while True:

time.sleep(10)

self.sig.emit()

#

class RegistForm(QtWidgets.QWidget):

#Класс создания окна регистрации

def \_\_init\_\_(self, sql, parent=None):

#Параметры и компоненты окна

super(RegistForm, self).\_\_init\_\_()

uic.loadUi("ui\_Regist.ui", self)

self.sql = sql

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.Enter.clicked.connect(self.EnterReg)

self.Cancel.clicked.connect(self.close)

def EnterReg(self):

#Метод внесения в таблицу регистрации записи

if (

self.login.text() and self.password.text() and self.f.text() and self.i.text() and self.o.text() and self.phone.text()) != '':

self.sql.Insert('users', self.login.text(), self.password.text(), self.f.text(), self.i.text(),

self.o.text(), self.phone.text(), 'Не активен')

self.close()

else:

self.ErrorL.setText('Все поля должны быть заполнены.')

#

class MyWindow(QtWidgets.QMainWindow, main\_windows.Ui\_MainWindow):

#Класс создание главной формы

def \_\_init\_\_(self, parent=None):

#Метод определения параметров и компонентов окна

super(MyWindow, self).\_\_init\_\_()

self.setupUi(self)

self.sql = sql.SqlCom()

# переменные для работы кнопок

self.photos = self.sql.SelectOne('photo', 'route')

self.idNews = 1

self.idMax = self.sql.SelectMaxID('news\_admin')

self.news = self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews)

self.News.setText(self.news[0])

self.Photo.setPixmap(QtGui.QPixmap(self.photos[0]))

self.counP = 0

self.posP = -1

for i in self.photos:

self.posP += 1

# вывод таблицы комментариев

self.TableCommentUp()

# Кнопки

self.Comment.clicked.connect(self.CommentF)

self.com = CommentForm(self.sql, MyWindow)

self.Register.clicked.connect(self.RegistF)

self.reg = RegistForm(self.sql)

self.us = PrivilegForm(self.sql, user)

self.About.clicked.connect(self.AboutB)

self.Cabinet.clicked.connect(self.CabinetB)

self.Rent.clicked.connect(self.RentB)

self.Services.clicked.connect(self.ServicesB)

self.Contacts.clicked.connect(self.ContactsB)

self.Manual.clicked.connect(self.ManualB)

def ManualB(self):

self.Text.setHtml(text.e)

def ContactsB(self):

self.Text.setHtml(text.d)

def ServicesB(self):

self.Text.setHtml(text.c)

def RentB(self):

self.Text.clear()

self.Text.insertPlainText('У нас есть следующее оборудование для ареннды - \n\n')

for i in self.sql.SelectOne('rent','nameIns'):

self.Text.insertPlainText(i+' - '+str(self.sql.SelectOne('rent','price','nameIns',"'"+i+"'")[0])+'\n')

self.Text.insertPlainText('\n Оборудование арендуется на все время репетиции, не зависимо от колличества часов.')

def CabinetB(self):

#Метод внесения текста в виджет текст

self.Text.setHtml(text.b)

def AboutB(self):

#Метод аналогичен предыдущему

self.Text.setHtml(text.a)

def TableCommentUp(self):

#Метод создания модели таблицы БД и вывода ее в виджет table

self.sgm = QtSql.QSqlQueryModel(parent=self.tableC)

self.sgm.setQuery("select \* from comments order by ID desc")

self.sgm.setHeaderData(1, QtCore.Qt.Horizontal, 'Логин')

self.sgm.setHeaderData(2, QtCore.Qt.Horizontal, 'Комментарий')

self.tableC.setModel(self.sgm)

self.tableC.hideColumn(0)

self.tableC.resizeRowsToContents()

self.tableC.setColumnWidth(2, 166)

def RegistF(self):

#Метод открытия окна регистрации

self.reg.show()

self.reg.login.clear()

self.reg.password.clear()

self.reg.phone.clear()

self.reg.f.clear()

self.reg.i.clear()

self.reg.o.clear()

def CommentF(self):

#Метод открытия окна оставить комментарий

self.com.show()

self.com.textEdit.clear()

def EnterUs(self):

# Метод перехода в форму привилегированного режима с учетом введения логина и пароля.

log = self.Login.text()

pas = self.Password.text()

self.query = QtSql.QSqlQuery()

self.query.exec(

"select login,status from users where login=" + "'" + log + "'" + " and password=" + "'" + pas + "'")

if self.query.isActive():

self.query.first()

if self.query.isValid():

user.append(self.query.value('login'))

user.append(self.query.value('status'))

self.query.finish()

self.Password.clear()

if user[1]!='Не активен':

self.us.show()

self.us.start()

else:

user.clear()

self.Login.setText('Не Активирован')

else:

self.Login.setText('error')

def PhotoB(self):

#Метод переключения фотографии назад

self.counP -= 1

if self.counP < 0:

self.counP = self.posP

self.Photo.setPixmap(QtGui.QPixmap(self.photos[self.counP]))

def PhotoN(self):

#Метод переключения фотографии вперед

self.counP += 1

if self.counP > self.posP:

self.counP = 0

self.Photo.setPixmap(QtGui.QPixmap(self.photos[self.counP]))

def NewsB(self):

#Метод переключения метода назад

self.idNews -= 1

if self.idNews < 1:

self.idNews = self.idMax

self.News.setText(self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews)[0])

else:

self.News.setText(self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews)[0])

def NewsN(self):

#Метод переключения новости вперед

self.idNews += 1

if not self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews):

self.idNews = 1

self.News.setText(self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews)[0])

else:

self.News.setText(self.sql.SelectOne('news\_admin', 'news', 'ID', self.idNews)[0])

#

class RecordBand(QtWidgets.QWidget):

#Класс создания окна записи группы

def \_\_init\_\_(self, sql, parent=None):

#Метод внесения параметров и компонентов окна

super(RecordBand, self).\_\_init\_\_()

self.sql = sql

uic.loadUi("ui\_record.ui", self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.Cancel.clicked.connect(self.close)

self.Record.clicked.connect(self.addRecord)

self.Date.dateChanged.connect(self.selectTable)

def selectTable(self):

#Метод создания модели таблицы и вывода ее в виджет Table

self.sgm = QtSql.QSqlQueryModel(parent=self.Table)

self.sgm.setQuery("select time,timeEnd from sing\_up where date='" + self.Date.text() + "'")

self.sgm.setHeaderData(0, QtCore.Qt.Horizontal, 'Начало')

self.sgm.setHeaderData(1, QtCore.Qt.Horizontal, 'Конец')

self.Table.setModel(self.sgm)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

def addRecord(self):

#Метод добавления записи в таблицу записей на репетицию

band = self.sql.SelectOne('bands', 'ID', 'nameB', "'" + self.Band.currentText() + "'")[0]

self.sql.Insert('sing\_up', band, self.Date.text(), self.Time.text(), self.TimeEnd.text(), self.Data.text())

self.close()

def addBand(self):

#Метод внесения названий групп в список

self.Band.clear()

self.Band.addItems(self.sql.SelectOne('bands', 'nameB'))

datePc = datetime.datetime.now()

self.Date.setDate(datePc.date())

self.selectTable()

#

class PayForm(QtWidgets.QWidget):

#Класс создания окна оплаты

def \_\_init\_\_(self, sql, parent=None):

#Объявление параметров и компонентов окна

super(PayForm, self).\_\_init\_\_()

uic.loadUi("ui\_Pay.ui", self)

self.sql = sql

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.IntPay = 0

self.IntZ = 0

self.Cancel.clicked.connect(self.close)

self.Band.activated.connect(self.TimeTimeEnd)

self.Pay.setDecMode()

self.addRent.clicked.connect(self.addR)

self.addHour.clicked.connect(self.addH)

self.PayBand.editingFinished.connect(self.ReturnSum)

self.PayB.clicked.connect(self.addPay)

def addPay(self):

#Мктод добавление записи в таблицу оплаты

self.sql.Insert('pay', self.sql.SelectOne('bands', 'ID', 'nameB', "'" + self.Band.currentText() + "'")[0],

datetime.datetime.now().strftime("%d.%m.%Y"), self.Time.text(), self.TimeEnd.text(),

self.RentT.toPlainText(), self.Pay.intValue())

txt = QtGui.QTextDocument()

txt.setPlainText(

'-' \* 100 + "\n Репетиционная студия '9/10'" + "\n Спасибо что провели у нас репетицию " + self.Band.currentText() + "\n" + '-' \* 100 +

'\n' + str(datetime.datetime.now()) +

'\n К оплате - \n Продолжительность репетиции - ' + self.Hour.text() + ' час/часов \n' + '(Стоимость часа - 200 руб)\n' +

'Арендованно оборудование: \n' + self.RentT.toPlainText() + '\n' + '-' \* 100 +

'\n Итог - ' + str(self.Pay.intValue()) + '\n Внесено -' + self.PayBand.text() + '\n Сдача - ' + str(

self.Return.intValue()) +

'\n Спасибо ' + self.Band.currentText() + ', приходите еще. \n' + '-' \* 100)

pri = QtPrintSupport.QPrinter()

dia = QtPrintSupport.QPrintDialog(pri)

dia.exec()

txt.print\_(pri)

self.sql.Delete('sing\_up', self.sql.SelectOne('bands', 'ID', 'nameB', "'" + self.Band.currentText() + "'")[0],

datetime.datetime.now().strftime("%d.%m.%Y"))

self.close()

def ReturnSum(self):

#Метод определения здачи

self.Return.display(int(self.PayBand.text()) - self.Pay.intValue())

def addH(self):

#Метод вносящий стоимость часов в оплату

self.Pay.display(int(self.Hour.text()) \* 200)

self.addHour.setEnabled(False)

def addR(self):

#Метод вносящий обородувание в RentT и вносящий их стоимость в оплату

self.RentT.insertPlainText(self.Rent.currentText() + ', ')

self.Pay.display(self.sql.SelectOne('rent', 'price', 'nameIns', "'" + self.Rent.currentText() + "'")[0])

self.Rent.removeItem(self.Rent.currentIndex())

if self.Rent.count() == 0:

self.addRent.setEnabled(False)

def TimeTimeEnd(self):

#Метод заполняющий поля время начала и время конца репетиции

self.Time.setText(self.sql.SelectOne('sing\_up', 'time', 'id\_nameB', self.sql.SelectOne('bands', 'ID', 'nameB',

"'" + self.Band.currentText() + "'")[

0])[0])

self.TimeEnd.setText(self.sql.SelectOne('sing\_up', 'timeEnd', 'id\_nameB',

self.sql.SelectOne('bands', 'ID', 'nameB',

"'" + self.Band.currentText() + "'")[0])[0])

def Start(self):

#Метод выполняющийся при запуске окна выполняющий функцию первоночальных параметров

self.addHour.setEnabled(True)

self.addRent.setEnabled(True)

self.Pay.display(0)

self.Return.display(0)

self.Band.clear()

self.Rent.clear()

for i in self.sql.SelectOne('sing\_up', 'id\_nameB', 'date',

"'" + datetime.datetime.now().strftime("%d.%m.%Y") + "'"):

self.Band.addItems(self.sql.SelectOne('bands', 'nameB', 'ID', i))

self.Rent.addItems(self.sql.SelectOne('rent', 'nameIns'))

self.Time.clear()

self.TimeEnd.clear()

self.RentT.clear()

self.PayBand.clear()

self.Hour.setValue(0)

#

class ReportYear(QtWidgets.QWidget):

#класс создания окна годового отчета

def \_\_init\_\_(self, sql, parent=None):

#метод определения параметров и компонентов

super(ReportYear, self).\_\_init\_\_()

uic.loadUi("ui\_report.ui", self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.sql = sql

self.report = QtGui.QTextDocument()

self.YearEdit.editingFinished.connect(self.start)

self.Close.clicked.connect(self.close)

self.Print.clicked.connect(self.PrintR)

def PrintR(self):

#Метод печати отчета

pri = QtPrintSupport.QPrinter()

dia = QtPrintSupport.QPrintDialog(pri)

dia.exec()

self.report.print\_(pri)

def start(self):

# Метод Выполняющийся при открытии окна и изменении даты отчета, который заполняет сам отчет

self.Text.clear()

self.report.clear()

self.Text.insertPlainText('-' \* 100 + '\n Годовой отчет \n \n \n')

pr = 0

for i in range(13):

if i != 0:

sum = 0

if i < 10:

self.sql.query.exec("select \* from pay where date like '\_\_.0" + str(i) + "." + str(self.YearEdit.text()) + "'")

else:

self.sql.query.exec(

"select \* from pay where date like '\_\_." + str(i) + "." + str(self.YearEdit.text()) + "'")

lst1 = []

if self.sql.query.isActive():

self.sql.query.first()

while self.sql.query.isValid():

lst1.append(self.sql.query.value('price'))

self.sql.query.next()

for f in lst1:

sum += f

pr += sum

self.Text.insertPlainText(str(i) + ' Месяц - ' + str(sum) + ' руб. \n')

self.Text.insertPlainText('-' \* 100 + '\n Годовая прибыль -> ' + str(pr) + '\n' + '-' \* 100)

self.report.setPlainText(self.Text.toPlainText())

#

class ReportMount(QtWidgets.QWidget):

#Класс создание окна месячного отчета

def \_\_init\_\_(self, sql, parent=None):

#Метод объявления параметров и компонентов окна

super(ReportMount, self).\_\_init\_\_()

uic.loadUi("ui\_reportM.ui", self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.sql = sql

self.report = QtGui.QTextDocument()

self.Mount.editingFinished.connect(self.start)

self.Year.editingFinished.connect(self.start)

self.Close.clicked.connect(self.close)

self.Print.clicked.connect(self.PrintR)

def PrintR(self):

#Метод печати документа

pri = QtPrintSupport.QPrinter()

dia = QtPrintSupport.QPrintDialog(pri)

dia.exec()

self.report.print\_(pri)

def start(self):

#Метод запускающийся при открытие окна и при изменение месяца и года отчета, который формирует сам отчет

self.Text.clear()

self.report.clear()

self.Text.insertPlainText('-' \* 100 + '\n Отчет за месяц \n \n \n' + 'Группа | Дата | Сумма \n')

pr = 0

lst = []

if int(self.Mount.text()) < 10:

self.sql.query.exec(

"select \* from pay where date like '\_\_.0" + str(self.Mount.text()) + "." + str(self.Year.text()) + "'")

else:

self.sql.query.exec(

"select \* from pay where date like '\_\_." + str(self.Mount.text()) + "." + str(self.Year.text()) + "'")

if self.sql.query.isActive():

self.sql.query.first()

while self.sql.query.isValid():

lst.append(self.sql.query.value('id\_nameB'))

lst.append(self.sql.query.value('date'))

lst.append(self.sql.query.value('price'))

self.sql.query.next()

con = 0

for i in lst:

if con == 0:

self.Text.insertPlainText(self.sql.SelectOne('bands', 'nameB', 'ID', "'" + str(i) + "'")[0] + ' ')

con += 1

elif con == 1:

self.Text.insertPlainText(str(i) + ' ')

con += 1

elif con == 2:

pr += i

self.Text.insertPlainText(str(i) + '\n')

con = 0

self.Text.insertPlainText('-' \* 100 + '\n Прибыль -> ' + str(pr) + '\n' + '-' \* 100)

self.report.setPlainText(self.Text.toPlainText())

#

class PrivilegForm(QtWidgets.QWidget, Priv.Ui\_FormPriv):

#Класс создание окна привилегированного режима

def \_\_init\_\_(self, sql, parent=None):

#Объявление параметров и компонентов окна

super(PrivilegForm, self).\_\_init\_\_()

self.sql = sql

self.setupUi(self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.Comment.clicked.connect(self.UsComment)

self.com = ''

self.Exit.clicked.connect(self.close)

self.NewsAdminB.clicked.connect(self.newsAdminText)

self.newsAd = NewsAdminForm(self.sql, PrivilegForm)

self.Box.activated.connect(self.tableUp)

self.addB.clicked.connect(self.addBut)

self.deleteB.clicked.connect(self.delBut)

self.newPhoto.clicked.connect(self.newPh)

self.time()

self.RecordB.clicked.connect(self.addRec)

self.record = RecordBand(self.sql)

self.pay = PayForm(self.sql)

self.PayB.clicked.connect(self.addPay)

self.DayB.clicked.connect(self.FilterDay)

self.EnterRecB.clicked.connect(self.EntRec)

self.EnterPayB.clicked.connect(self.EntPay)

self.EnterRec.clicked.connect(self.EntRecD)

self.EnterPay.clicked.connect(self.EntPayD)

self.reportY = ReportYear(self.sql)

self.yearB.clicked.connect(self.year)

self.reportM = ReportMount(self.sql)

self.mountB.clicked.connect(self.mount)

def mount(self):

#Модуль открытие окна месячного отчета

self.reportM.show()

self.reportM.start()

def year(self):

#Модуль открытия годового отчета

self.reportY.show()

self.reportY.start()

def EntPayD(self):

#Метод создания модели таблицы оплаты и вывода ее в таблицу по дате указанной в компоненте PayDate для поиска

self.Table.setModel(self.sql.STMR('pay', 'date', self.PayDate.text()))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def EntRecD(self):

#Метод создания модели таблицы записи и вывода ее в таблицу по дате указанной

self.Table.setModel(self.sql.STMR('sing\_up', 'date', self.RecDate.text()))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def EntPay(self):

#Метод создания модели таблицы оплаты и вывода ее в таблицу по группе выбранной в списке компонента BandBox

self.Table.setModel(self.sql.STMR('pay', 'id\_nameB', str(

self.sql.SelectOne('bands', 'ID', 'nameB', "'" + self.BandBox.currentText() + "'")[0])))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def EntRec(self):

#Метод создания модели таблицы записи и вывода ее в таблицу по группе выбранной в списке компонента BandBox

self.Table.setModel(self.sql.STMR('sing\_up', 'id\_nameB', str(

self.sql.SelectOne('bands', 'ID', 'nameB', "'" + self.BandBox.currentText() + "'")[0])))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def FilterDay(self):

#Метод создание модели таблицы записей и вывод ее в виджет table по текущей дате

self.Table.setModel(self.sql.STMR('sing\_up', 'date', datetime.datetime.now().strftime("%d.%m.%Y")))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def addPay(self):

#Метод открытия окна оплаты

self.pay.show()

self.pay.Start()

def addRec(self):

#Метод открытия окна записи

self.record.show()

self.record.addBand()

def time(self):

# Метод занесения текущей даты и времени в виджеты DataL и TimeL

dtime = datetime.datetime.now()

self.DateL.setText(str(dtime.day) + '/' + str(dtime.month) + '/' + str(dtime.year))

self.TimeL.setText(str(dtime.hour) + ':' + str(dtime.minute))

def newPh(self):

#Методдобавления фотографии в каталог программы и занесение ее в таблицу фотографий

file = QtWidgets.QFileDialog.getOpenFileName(caption='Добавить Фото')

if file[0] != '':

shutil.copy(file[0], "photo")

self.sql.Insert('photo', os.path.basename(file[0]))

os.path.basename(file[0])

def delBut(self):

#Метод удаления записи

self.sql.stmR.removeRow(self.Table.currentIndex().row())

self.sql.stmR.select()

self.sql.stm.removeRow(self.Table.currentIndex().row())

self.sql.stm.select()

def addBut(self):

#Метод добавления записи

self.sql.stm.insertRow(self.sql.stm.rowCount())

self.sql.stmR.insertRow(self.sql.stmR.rowCount())

def tableUp(self):

#Метод создания макета таблиц и вывода их в виджет Table, которые выводятся по выбранной таблице в списке Box

if self.Box.currentText() == 'Группы':

self.Table.setModel(self.sql.STM('bands'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Пользователи':

self.Table.setModel(self.sql.STM('users'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Коментарии':

self.Table.setModel(self.sql.STM('comments'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Аренда':

self.Table.setModel(self.sql.STM('rent'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Новости':

self.Table.setModel(self.sql.STM('news\_admin'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Фотографии':

self.Table.setModel(self.sql.STM('photo'))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Записи':

self.Table.setModel(self.sql.STMR('sing\_up'))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

if self.Box.currentText() == 'Приемка':

self.Table.setModel(self.sql.STMR('pay'))

self.Table.setItemDelegateForColumn(1, QtSql.QSqlRelationalDelegate(self.Table))

self.Table.hideColumn(0)

self.Table.resizeRowsToContents()

self.Table.setColumnWidth(2, 166)

self.Table.setSortingEnabled(True)

def closeEvent(self, QCloseEvent):

#Метод который срабатывает при закрытие окна любым способом

self.close()

global user

user = []

def UsComment(self):

#Метод открытия формы коментариев из объекта класса главного окна

window.CommentF()

def start(self):

#Метод который срабатывает при запуске окна, вносящий в виджеты логин и статус пользователя и определяющий по нему уровень доступа

self.Login.setText(user[0])

self.Status.setText(user[1])

self.Box.clear()

if user[1]=='Администратор':

self.Box.addItem("Пользователи")

self.Box.addItem("Фотографии")

self.Box.addItem("Новости")

self.Box.addItem("Аренда")

self.Box.addItem("Приемка")

self.Box.addItem("Записи")

self.Box.addItem("Группы")

self.Box.addItem("Коментарии")

self.DayT.clear()

self.DayT.insertPlainText(datetime.datetime.now().strftime("%d.%m.%Y") + ' Сегоднишние репетиции: \n')

for i in self.sql.SelectOne('sing\_up', 'id\_nameB', 'date',

"'" + datetime.datetime.now().strftime("%d.%m.%Y") + "'"):

self.DayT.insertPlainText(self.sql.SelectOne('bands', 'nameB', 'ID', i)[0] + '\n')

self.BandBox.addItems(self.sql.SelectOne('bands', 'nameB'))

def newsAdminText(self):

#Метод открытие окна добавления новости

self.newsAd.show()

#

class NewsAdminForm(QtWidgets.QWidget):

#Класс создания окна добавления новости

def \_\_init\_\_(self, sql, parent=None):

#Метод вносящий параметры и компоненты окна

super(QtWidgets.QWidget, self).\_\_init\_\_()

self.sql = sql

uic.loadUi("ui\_NewsAdmin.ui", self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.CancelB.clicked.connect(self.close)

self.EnterB.clicked.connect(self.enterNews)

def enterNews(self):

#Метод занесения в таблицу news\_admin базы данных запись

self.sql.Insert('news\_admin', self.NewsT.toPlainText())

self.close()

#

class CommentForm(QtWidgets.QWidget):

#Класс создания окна добавления коментариев

def \_\_init\_\_(self, sql, parent=None):

#Метод объявления параметров и компонентов окна

super(CommentForm, self).\_\_init\_\_()

uic.loadUi("ui\_Comment.ui", self)

self.setWindowModality(QtCore.Qt.ApplicationModal)

self.CommentEnter.clicked.connect(self.ComEn)

self.CommentExit.clicked.connect(self.close)

self.sql = sql

def ComEn(self):

# Метод добавления записи в таблицу комментариев

if len(user) == 0:

self.sql.Insert('comments', 'Гость', self.textEdit.toPlainText())

window.TableCommentUp()

self.close()

else:

self.sql.Insert('comments', user[0], self.textEdit.toPlainText())

window.TableCommentUp()

self.close()

if \_\_name\_\_ == "\_\_main\_\_":

app = QtWidgets.QApplication(sys.argv)

user = []

window = MyWindow(user)

window.showFullScreen()

th = Thread()

th.start()

th.sig.connect(window.NewsN)

th.sig.connect(window.PhotoN)

th.sig.connect(window.us.time)

sys.exit(app.exec\_())

**Модуль sql.py –**

from PyQt5 import QtWidgets, QtSql

import sys

class SqlCom():

# Класс создания объекта для работы с базой данных

def \_\_init\_\_(self):

#Метод объявления компонентов БД и создания таблиц при их несуществовании

self.app = QtWidgets.QApplication(sys.argv)

self.con = QtSql.QSqlDatabase.addDatabase('QSQLITE')

self.con.setDatabaseName('rehearsal\_base')

self.con.open()

self.query = QtSql.QSqlQuery()

if 'news\_admin' not in self.con.tables():

self.query.exec("create table news\_admin(ID integer not null primary key autoincrement, news text) ")

self.query.prepare("insert into news\_admin values(null,?)")

self.query.addBindValue('Проба ураааааа сработало!!!!')

self.query.exec\_()

if 'photo' not in self.con.tables():

self.query.exec("create table photo(ID integer not null primary key autoincrement, route text)")

self.query.prepare("insert into photo values(null,?)")

self.query.addBindValue('photo/1.jpg')

self.query.exec\_()

self.query.prepare("insert into photo values(null,?)")

self.query.addBindValue('photo/2.jpg')

self.query.exec\_()

if 'comments' not in self.con.tables():

self.query.exec("create table comments(ID integer not null primary key autoincrement, login text, comment text)")

self.query.prepare("insert into comments values(null,?,?)")

self.query.addBindValue('admin')

self.query.addBindValue('работает вроде')

self.query.exec\_()

self.query.prepare("insert into comments values(null,?,?)")

self.query.addBindValue('Гость')

self.query.addBindValue('Чет пока тут пуста.... ')

self.query.exec\_()

if 'users' not in self.con.tables():

self.query.exec("create table users(ID integer not null primary key autoincrement, login text, password text,f text,i text,o text, phone integer, status text)")

self.query.prepare("insert into users values(null,?,?,?,?,?,?,?)")

self.query.addBindValue('admin')

self.query.addBindValue('7291')

self.query.addBindValue('Семенцов')

self.query.addBindValue('Алексей')

self.query.addBindValue('Михайлович')

self.query.addBindValue('8998888888')

self.query.addBindValue('Администратор')

self.query.exec\_()

if 'bands' not in self.con.tables():

self.query.exec("create table bands(ID integer not null primary key autoincrement, nameB text, data text)")

self.query.prepare("insert into bands values(null,?,?)")

self.query.addBindValue('Slipknot')

self.query.addBindValue('Кори Тейлор - 89995674354')

self.query.exec\_()

if 'rent' not in self.con.tables():

self.query.exec("create table rent(ID integer not null primary key autoincrement, nameIns text, price integer)")

self.query.prepare("insert into rent values(null,?,?)")

self.query.addBindValue('Ibanez GIO')

self.query.addBindValue('100')

self.query.exec\_()

if 'sing\_up' not in self.con.tables():

self.query.exec("create table sing\_up(ID integer not null primary key autoincrement, id\_nameB INTEGER REFERENCES bands (ID) ON DELETE CASCADE ON UPDATE CASCADE, date integer , time text, timeEnd text, data text)")

self.query.prepare("insert into sing\_up values(null,?,?,?,?,?)")

self.query.addBindValue('1')

self.query.addBindValue('05.06.2017')

self.query.addBindValue('12.00')

self.query.addBindValue('11.00')

self.query.addBindValue('Мик Томсон 89993216548')

self.query.exec\_()

if 'pay' not in self.con.tables():

self.query.exec("create table pay(ID integer not null primary key autoincrement, id\_nameB INTEGER REFERENCES bands (ID) ON DELETE CASCADE ON UPDATE CASCADE, date text, time text,timeEnd text,nameIns text ,price integer)")

self.query.prepare("insert into pay values(null,?,?,?,?,?,?)")

self.query.addBindValue('1')

self.query.addBindValue('05.06.2017')

self.query.addBindValue('12.00')

self.query.addBindValue('13.00')

self.query.addBindValue('Ibanez GIO')

self.query.addBindValue('1500')

self.query.exec\_()

self.stm = QtSql.QSqlTableModel()

self.stmR = QtSql.QSqlRelationalTableModel()

def STMR(self,table,filt='',us=''):

#Метод создания макета с межтабличной связью

self.stmR.setEditStrategy(QtSql.QSqlTableModel.OnFieldChange)

self.stmR.setTable(table)

if filt !='':

print("'"+filt+"'"+'='+"'"+us+"'")

self.stmR.setFilter(filt+'='+"'"+us+"'")

self.stmR.setRelation(1, QtSql.QSqlRelation('bands','ID','nameB'))

self.stmR.select()

return self.stmR

def STM(self,table):

#Метод создания макета

self.stm.setTable(table)

self.stm.setEditStrategy(QtSql.QSqlTableModel.OnFieldChange)

self.stm.select()

return self.stm

def Delete(self,tabl,name, date):

#Метод удаления записи из таблицы

self.query.exec("delete from " + tabl+' where id\_nameB='+"'"+str(name)+"'"+' and date='+"'"+date+"'")

def Insert(self,tabl,data1,data2='',data3='',data4='',data5='',data6='',data7=''):

#Метод добавления записи в таблицу

if data2 != '':

if data3 != '':

if data4 != '':

if data5 !='':

if data6 !='':

if data7 != '':

self.query.prepare("insert into " + tabl + " values(null,?,?,?,?,?,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.addBindValue(data3)

self.query.addBindValue(data4)

self.query.addBindValue(data5)

self.query.addBindValue(data6)

self.query.addBindValue(data7)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?,?,?,?,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.addBindValue(data3)

self.query.addBindValue(data4)

self.query.addBindValue(data5)

self.query.addBindValue(data6)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?,?,?,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.addBindValue(data3)

self.query.addBindValue(data4)

self.query.addBindValue(data5)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?,?,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.addBindValue(data3)

self.query.addBindValue(data4)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.addBindValue(data3)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?,?)")

self.query.addBindValue(data1)

self.query.addBindValue(data2)

self.query.exec\_()

self.query.finish()

else:

self.query.prepare("insert into " + tabl + " values(null,?)")

self.query.addBindValue(data1)

self.query.exec\_()

self.query.finish()

def SelectMaxID(self,table):

#Метод определения колличества строк в таблице

self.query.exec("select count(1) from " + table)

lst = 0

if self.query.isActive():

self.query.first()

lst=self.query.value('count(1)')

self.query.next()

return lst

def SelectOne(self, table, column,columnS='',line=''):

Метод вывода данных из таблицы , с возможностью фильтрации

if columnS == '':

self.query.exec("select \* from "+table)

lst=[]

if self.query.isActive():

self.query.first()

while self.query.isValid():

lst.append(self.query.value(column))

self.query.next()

return lst

else:

line=str(line)

self.query.exec("select \* from " + table+' where '+columnS+'='+line)

lst = []

if self.query.isActive():

self.query.first()

while self.query.isValid():

lst.append(self.query.value(column))

self.query.next()

return lst

**Модуль text.py –**

#Переменная с текстом, которая выводится при нажатии кнопки на главной странице, о нас.

a="""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"> <html><head><meta

name="qrichtext" content="1" /><style type="text/css"> p, li { white-space: pre-wrap; } </style></head><body style="

font-family:'MS Shell Dlg 2'; font-size:8pt; font-weight:400; font-style:normal;"> <p style=" margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><span style="

font-size:8.25pt;">Репитиционная база 9/10.</span></p> <p style="-qt-paragraph-type:empty; margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8.25pt;"><br

/></p> <p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;

text-indent:0px;"><span style=" font-size:8.25pt;"> База работает специально для начинающих музыкантов.

Мыпредоставляем качественное оборудование и помещение за небольшие деньги. Развивайте свое творчество и в дальнейшем

мы поможем вам с организацией концертов и записей альбомов.</span></p><p><img src="1.jpg"></p></body></html> """

#Переменная с текстом - помещение

b="""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"> <html><head><meta

name="qrichtext" content="1" /><style type="text/css"> p, li { white-space: pre-wrap; } </style></head><body style="

font-family:'MS Shell Dlg 2'; font-size:8pt; font-weight:400; font-style:normal;"> <p style=" margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><span style="

font-size:8.25pt;">Репитиционная база 9/10.</span></p> <p style="-qt-paragraph-type:empty; margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8.25pt;"><br

/></p> <p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;

text-indent:0px;"><span style=" font-size:8.25pt;"> База работает специально для начинающих музыкантов.

Мыпредоставляем качественное оборудование и помещение за небольшие деньги. Развивайте свое творчество и в дальнейшем

мы поможем вам с организацией концертов и записей альбомов.</span></p></body></html>"""

#Переменная с текстом – Сервесные услуги

c="""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"> <html><head><meta

name="qrichtext" content="1" /><style type="text/css"> p, li { white-space: pre-wrap; } </style></head><body style="

font-family:'MS Shell Dlg 2'; font-size:8pt; font-weight:400; font-style:normal;"> <p style=" margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><span style="

font-size:8.25pt;">Репитиционная база 9/10.</span></p> <p style="-qt-paragraph-type:empty; margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8.25pt;"><br

/></p> <p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;

text-indent:0px;"><span style=" font-size:8.25pt;"> Здесь перечислены возможные сервестные услуги. </span></p></body></html>"""

#Переменная с текстом - Контакты

d="""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"> <html><head><meta

name="qrichtext" content="1" /><style type="text/css"> p, li { white-space: pre-wrap; } </style></head><body style="

font-family:'MS Shell Dlg 2'; font-size:8pt; font-weight:400; font-style:normal;"> <p style=" margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><span style="

font-size:8.25pt;">Репитиционная база 9/10.</span></p> <p style="-qt-paragraph-type:empty; margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8.25pt;"><br

/></p> <p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;

text-indent:0px;"><span style=" font-size:8.25pt;"> Здесь перечислены контактная информация. </span></p></body></html>"""

#Переменная с текстом – Инстукция

e="""<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd"> <html><head><meta

name="qrichtext" content="1" /><style type="text/css"> p, li { white-space: pre-wrap; } </style></head><body style="

font-family:'MS Shell Dlg 2'; font-size:8pt; font-weight:400; font-style:normal;"> <p style=" margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><span style="

font-size:8.25pt;">Репитиционная база 9/10.</span></p> <p style="-qt-paragraph-type:empty; margin-top:0px;

margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:8.25pt;"><br

/></p> <p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;

text-indent:0px;"><span style=" font-size:8.25pt;"> Здесь написана инструкция к программе. </span></p></body></ht