Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и анализа данных**

**Допускаю к защите**

Руководитель:      \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

А.С. Тюрнев

**Промышленное производство.  
Заказчики и статусы, график поставок, платежи/задолженности.**

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Языки программирования»

1.009.00.00 ПЗ

|  |  |  |
| --- | --- | --- |
| Разработал студент группы ИБб-20-2 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | В.М. Жиленков |
|  | подпись |  |
| Нормоконтроль | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | А.А. Маринов |
|  | подпись |  |
| Курсовая работа защищена с оценкой | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | | дата защиты курсовой работы |

Иркутск 2021 г.

Содержание

[Введение. 3](#_Toc89964081)

[1. Структуризация данных. 4](#_Toc89964082)

[1.1. Базы данных. 4](#_Toc89964083)

[1.2. Structure Query Language. 5](#_Toc89964084)

[1.3 Системы управления базами данных. 6](#_Toc89964085)

[2. Разработка программы. 7](#_Toc89964086)

[2.1 Начало разработки. 7](#_Toc89964087)

[2.2. Подключение базы данных. 8](#_Toc89964088)

[2.3. Создание дизайна интерфейса. 11](#_Toc89964089)

[2.4. Создание логики программы. 13](#_Toc89964090)

[2.5. Добавление визуализации для графика поставок. 19](#_Toc89964091)

[Заключение 21](#_Toc89964092)

[Список используемых источников 22](#_Toc89964093)

[Приложение 1 – Исходный код файла с функциями SQL запросов к БД. 23](#_Toc89964094)

[Приложение 2 – Исходный код файла дизайна главного окна. 24](#_Toc89964095)

[Приложение 3 – Исходный код первого диалогового окна. 28](#_Toc89964096)

[Приложение 4 –Исходный код второго диалогового окна. 31](#_Toc89964097)

[Приложение 5 – Исходный код дополнения к календарю. 35](#_Toc89964098)

# Введение.

На сегодняшний момент в любой сфере жизнедеятельности человека возникает нужда в хранении и обработке информации, в основном это различные организации банки, учебные заведения, государственные учреждения и так далее. Но не везде и не всегда этому уделяется должное внимание, что может крайне негативно сказаться на работе таких предприятий. В данной работе речь пойдёт про создание приложения для выполнения вышеперечисленных задач. Актуальность создания такого приложений может показаться незначительной, так как на данный момент существует большое количество приложений для решения поставленных задач. Но так может показаться только на первый взгляд, так как в основном такие приложения разрабатываются для универсального использования и могут нести в себе огромные критические ошибки, что может крайне негативно сказаться на информационной безопасности предприятия. В данной работе будет рассмотрен процесс создания программы на подобии системы управления базой данных (СУБД), на примере предприятия промышленного производства.

Цель – создать программу, для предприятия промышленного производства, выполняющее задачи хранения и обработки информации.

Задачи:

1. Рассмотреть теоретическую часть баз данных.
2. Рассмотреть теоретическую часть систем управления базами данных.
3. Разобрать алгоритм работы приложения.

# 1. Структуризация данных.

# 1.1. Базы данных.

Как такового общепринятого определения БД не существует и поиск определения для БД привёл лишь к субъективному мнению других авторов. В своей сущности база данных (БД) представляет собой массив данных структурированных по определённым правилам. История возникновения и развития технологий баз данных может рассматриваться как в широком, так и в узком аспекте.

В широком смысле понятие истории баз данных обобщается до истории любых средств, с помощью которых человечество хранило и обрабатывало данные. В таком контексте упоминаются, например, средства учёта царской казны и налогов в древнем Шумере, узелковая письменность инков, клинописи, содержащие документы Ассирийского царства и т. п. Следует помнить, что недостатком этого подхода является размывание понятия «база данных» и фактическое его слияние с понятиями “архив” и даже “письменность”.

История баз данных в узком смысле рассматривает базы данных в традиционном (современном) понимании. Эта история начинается с 1955 года, когда появилось программируемое оборудование обработки записей. Программное обеспечение этого времени поддерживало модель обработки записей на основе файлов. Для хранения данных использовались перфокарты.

Оперативные сетевые базы данных появились в середине 1960-х. Операции над оперативными базами данных обрабатывались в интерактивном режиме с помощью терминалов. Простые индексно-последовательные организации записей быстро развились к более мощной модели записей, ориентированной на наборы.

Следующий важный этап связан с появлением в начале 1970-х реляционной модели данных, благодаря работам Эдгара Кодда. Работы Кодда открыли путь к тесной связи прикладной технологии баз данных с математикой и логикой.

Базы данных можно классифицируют по разным признакам, например, по модели данных, по среде постоянного хранения, по содержимому, и так далее. Данные классификации, в свою очередь разделяют на виды БД. [1]

# 1.2. Structure Query Language.

Structure Query Language (SQL) – это язык структурированных запросов, который позволяет эффективно сохранять данные, искать их части, извлекать из базы и удалять. Нужен, в первую очередь, для оптимизации и обслуживания баз данных, хотя на этом сфера его применения не заканчивается. Язык появился в 1974 г. и используется по всему миру. [2]

Язык является универсальным, имеет четкую определенную структуру и устоявшиеся стандарты. Отлично работает с БД даже при огромных объемах данных.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился – обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) – и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные системы управления базами данных (СУБД), а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Создание БД с помощью SQL на сегодняшний день является устаревшим методом, так как работа с такой базой данных возможно только при помощи терминала и специальных команд, использующих синтаксис SQL. Поэтому создаться приложения и программы для более удобного и понятного взаимодействия с базами данных. Такие программы называют СУБД.

# 1.3 Системы управления базами данных.

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных. [3]

СУБД – комплекс программ, позволяющих создать БД и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Сегодня СУБД используется повсеместно, так как являются достаточно простыми в освоении для рядовых пользователей.

Основные функции СУБД:

* Управление данными во внешней памяти.
* Управление данными в оперативной памяти с использованием  
  дискового кэша.
* Журнализация изменений, резервное копирование  
  и восстановление базы данных после сбоев.
* Поддержка языков БД (язык определения данных, язык манипулирования данными).

# 2. Разработка программы.

# 2.1 Начало разработки.

Для создания программы отвечающей нашим требованиям стоит изначально определиться, что именно нужно для разработки такой программы. Во-первых, нужно подключить базу данных, для этой задачи будем использовать модуль SQLite3, так как программирование осуществляется на языке программирования python, и модуль SQLite3 часть стандартного пакета этого языка программирования, тем самым исключаются возможные проблемы с установкой и настройкой другого модуля. Во-вторых, нужно как-то выводить работу программы пользователю, для этого нужен модуль, с помощью которого будет создан интерфейс для более удобной работы с программой нежели использование терминала. В этом нам поможет модуль PyQt5. Выбор данного модуля обусловлен тем, что он имеет множество плюсов по сравнению с другими модулями, например, более современный дизайн, огромное количество настраиваемых элементов дизайна, встроенная программа по созданию интерфейса и тому подобное.

# 2.2. Подключение базы данных.

Перво-наперво разберёмся с подключением базы данных, для этого используем стороннюю программу (DB\_Browser), которая поможет создать базу данных для первичного тестирования. Далее создаём отдельный файл с функциями, который в дальнейшем будем использовать как модуль, это будет удобнее, чем использовать один файл для всего. [4]

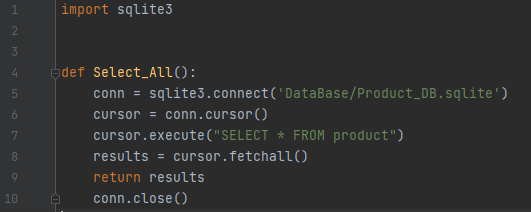
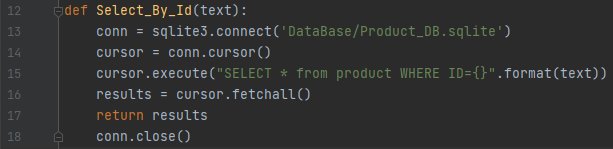


Рисунок 1. Функция, которая возвращает всю таблицу БД.

Рассмотрим рисунок 1, на первой строке происходит подключение модуля SQLite3, для работы с БД. Далее идёт первая функция. Разберём за что отвечает эта функция. При вызове данной функции происходит инициализация переменной **conn**,которая будет отвечать за подключение к базе данных, далее идёт инициализация переменной **cursor**, данная переменная в свою очередь отвечает за создание объекта, при помощи которого будет происходить обращение к базе данных. Теперь уже происходит обращение к базе данных с использованием стандартного синтаксиса SQL. Разберём это обращение. Команда SELECT извлекает записи или запись, из заданной таблицы или нескольких таблиц. Знак звёздочки в нашем случае указывает на то, что нужно извлечь все записи. Команда FROM указывает на то, откуда нужно производить заданные ранее действия, в нашем случае это таблица под название “product”. Следом идёт инициализация переменной, в которую передаётся то, что было получено в процессе SQL запроса. Далее функция возвращает результат своей работы, а уже затем происходит закрытие подключения к базе данных. Данная функция нужна для получения данных из всей таблицы, чуть позже это будет использоваться для отображения содержания БД в программе. [4]

 Рисунок 2. Функция, которая возвращает определённую запись в таблице БД.

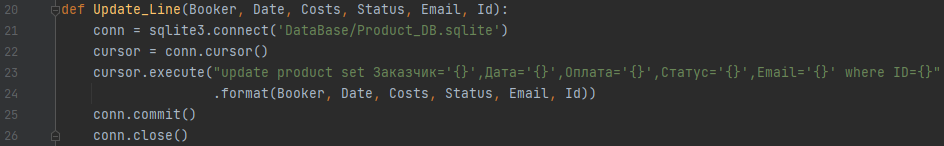
Рассмотрим вторую функцию в данном модул (см. Рис.2). В данной функции на вход приходит определённое число, на основе которого проиходит SQL запрос к БД, команда WHERE на то, какие именно данные должна вернуть БД. Эта функция отвичает за вывод информации в определённой строке БД. Описывать функцию также подробно не имеет смысла, так как по сравнению с прошлой функцией почти ничего не изменилось.

Рисунок 3. Функция, обновляющая значения определённой записи в таблице БД.

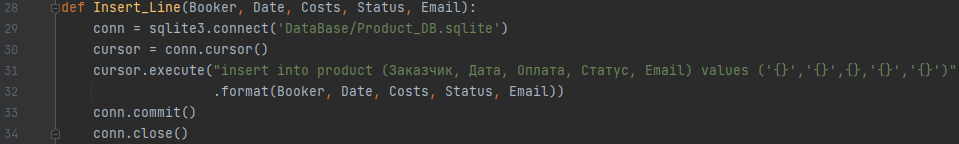
Рассмотрим третью функцию (см. Рис.3). Здесь картина уже более интересная по сравнению с прошлыми функциями. Функция принимает на вход значения, которые передаются в SQL запрос. В свою очередь данный запрос обновляет данные записанные в таблице БД под определённым Id, и в конце перед закрытием соединения с БД происходит сохранение изменений в БД. Данная функция будет работать в связке со второй функцией.

Рисунок 4. Функция, которая добавляет запись на основе полученных значений.

Рассмотрим четвёртую функцию (см. Рис.4). Функция принимает значения, которые будут добавлены в БД. Далее происходит SQL запрос, который передаёт базе данных, что нужно создать новую запись в БД с полученными значениями. Данная функция отвечает за создание новой строки на основе входных данных.

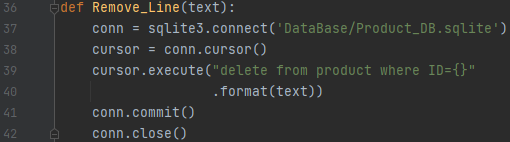


Рисунок 5. Функция, которая удаляет определённую запись таблицы БД.

На рисунке 5 показана пятая функция, отвечающая за удаление записи, Id которой функция получает как входное значение.

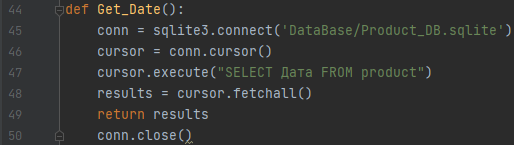


Рисунок 6. Функция, возвращающая все дата, найденные в таблице БД.

Рассмотрим рисунок 6, на нём продемонстрирована последняя функция, с помощью которой из таблицы БД будем получать все даты.

# 2.3. Создание дизайна интерфейса.

Создание интерфейса в PyQt5 можно разделить на два этапа, первый – создание дизайна, второй – добавление логики в дизайн. Для начала разберёмся с дизайном. В PyQt5 для создания дизайна можно использовать программу, которая идёт вместе с данным модулем, QtDesigner. [5]

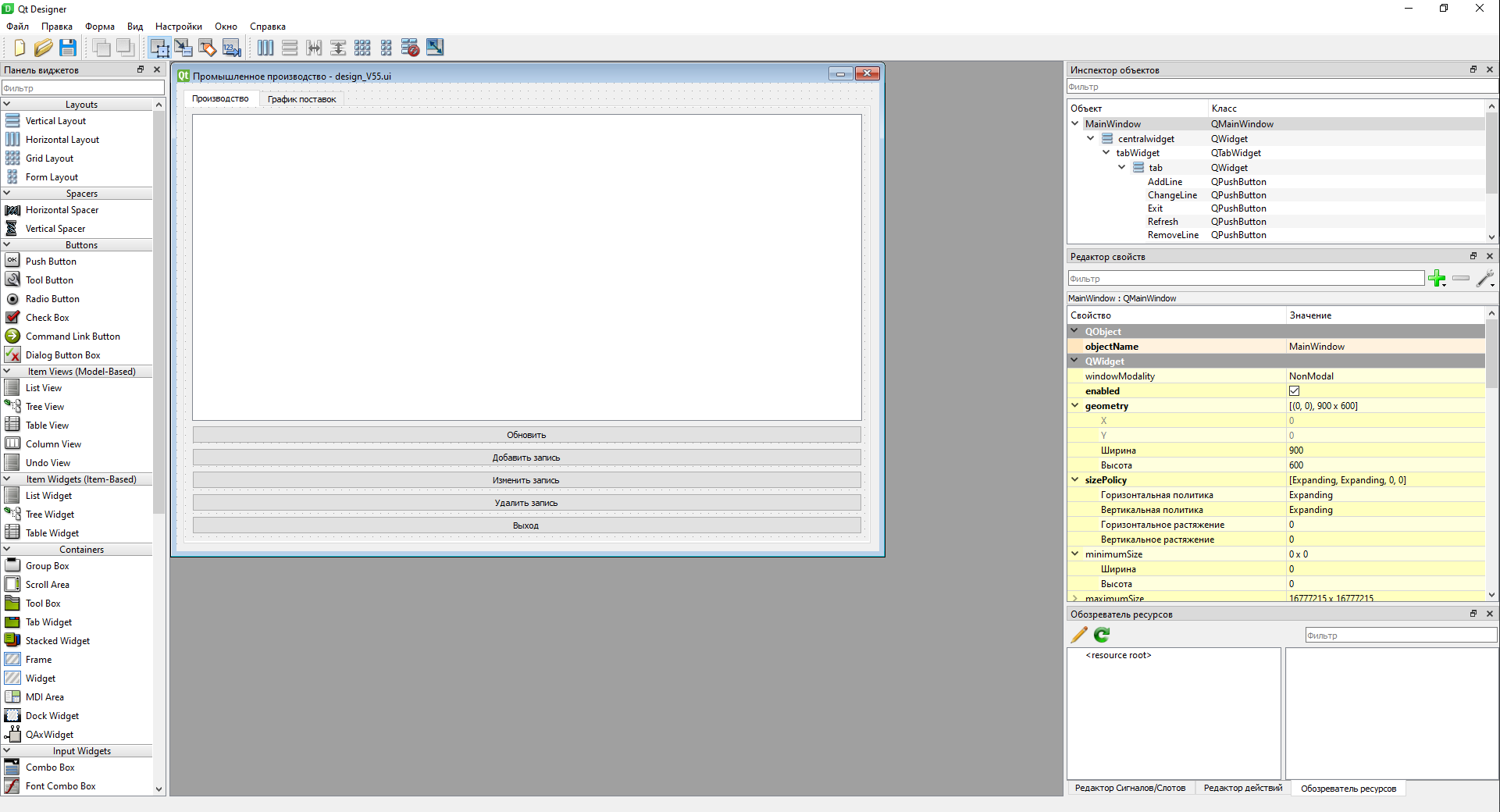


Рисунок 7. QtDesigner, Первая вкладка.

На рисунке 7 можно увидеть интерфейс данной программы, а также созданный нами бедующий дизайн главного окна. Главное окно состоит из следующих элементов:

* Виджет с двумя вкладками (первая вкладка предназначена для взаимодействия с БД, вторая вкладка отвечает за график поставок). (См. Рис. 7)
* Первая таблица для просмотра содержимого БД. (См. Рис. 7)
* Четыре кнопки, предназначенные для взаимодействия с содержимым БД. (См. Рис. 7)
* Кнопка, предназначенная для взаимодействия с программой. (См. Рис. 7)
* Календарь, предназначен для отображения дат поставок. (См. Рис. 7.1)
* Вторая таблица, предназначенная для более удобного просмотра графика поставок. (См. Рис. 7.1)

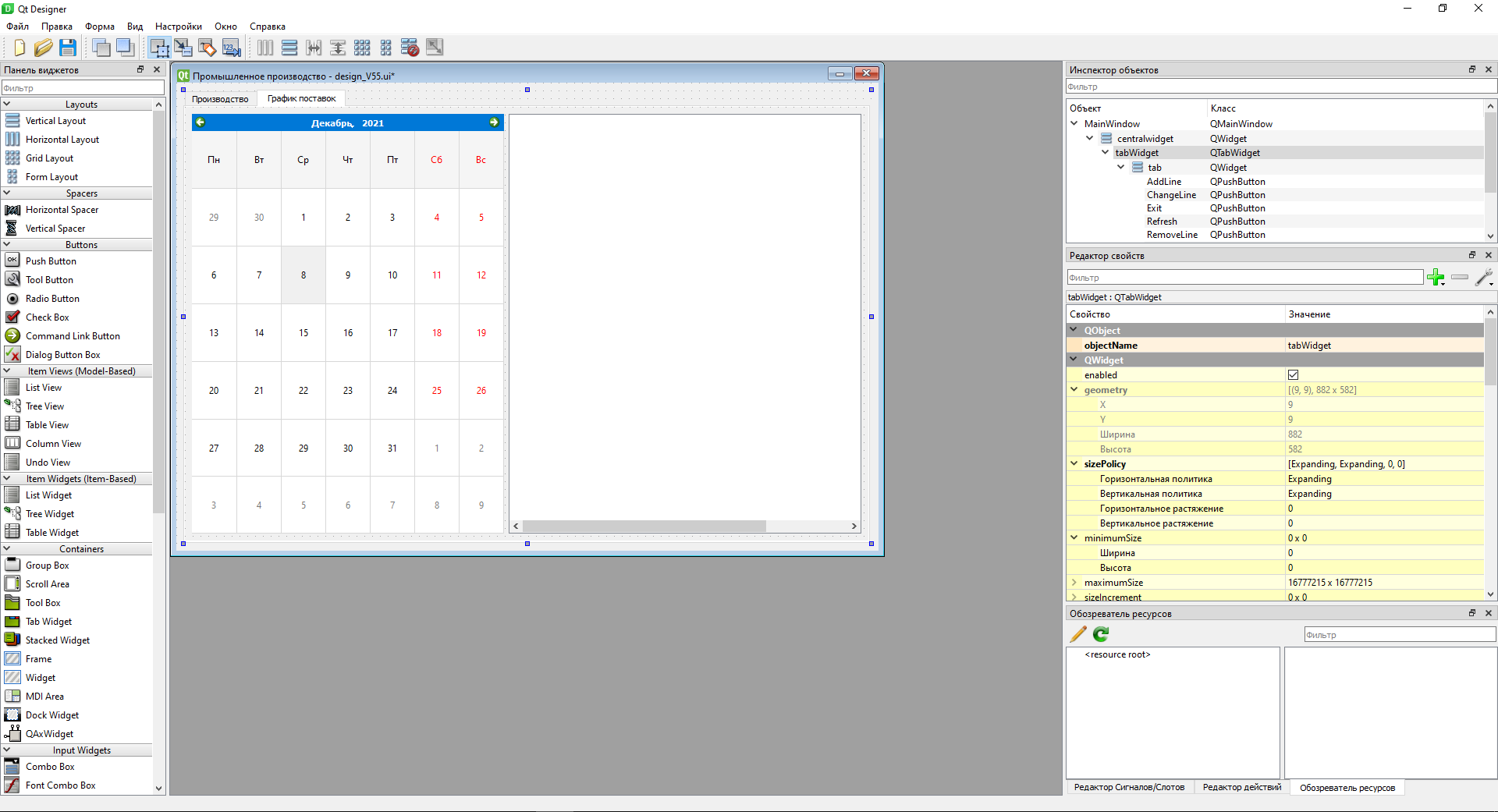
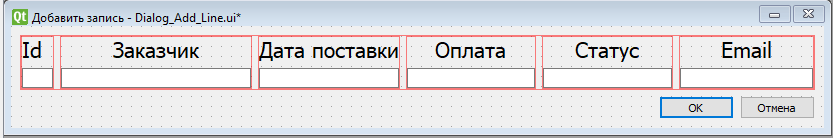


Рисунок 7.1. QtDesigner, Вторая вкладка.

Так же нужно создать ещё два дизайна для диалоговых окон, которые, будут отвечать за такие взаимодействие с содержимым БД, как добавление записи (См. Рис. 7.2) и изменение записи (См. Рис. 7.3).



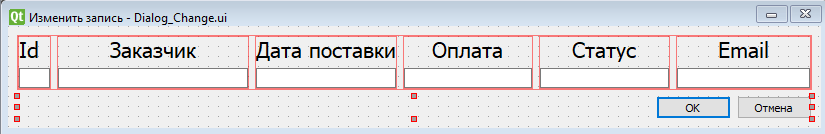
Рисунок 7.2. QtDesigner, добавление записи.

Рисунок 7.3. QtDesigner, изменение записи.

Создание и использование этих окон необходимо, так как встроенные в PyQt5 виджеты очень ограничены в использовании.

После того, как был создан дизайн, его следует конвертировать в python код, для того, чтобы его можно было использовать. Для этого в комплекте с PyQt5, как и в случае с QtDesigner, идёт скрипт, с помощью которого можно конвертировать дизайн из ui (Расширение, используемое QtDesigner) в py. Для конвертации в терминале нужно написать следующую команду: pyuic5 “Имя\_файла”.ui -o “Имя\_файла”.py. После этого, дизайн уже можно использовать как модуль и задать логику программы.

# 2.4. Создание логики программы.

В первую очередь нужно подключить модули, которые будут использоваться в программе. (См. Рис. 8)

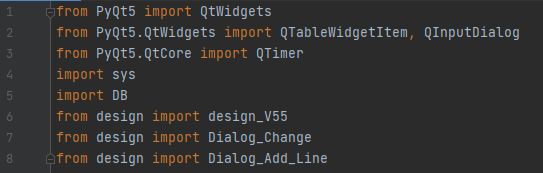


Рисунок 8. Подключение модулей.

Рассмотрим рисунок 8 подробнее. Первые три модуля это некоторые дополнения из PyQt5, с помощью которых будут реализованы несколько функций. Далее идёт подключение модуля с функциями связи с БД, и наконец подключение файлов с дизайном.

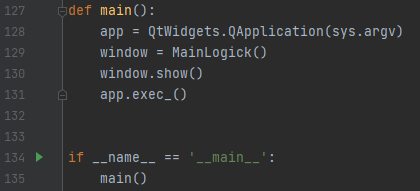
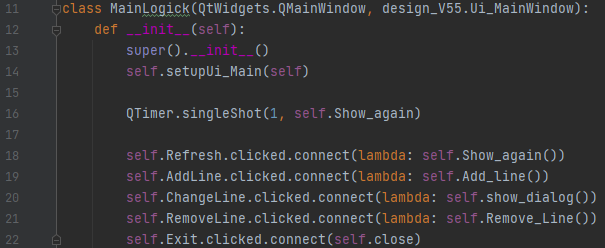
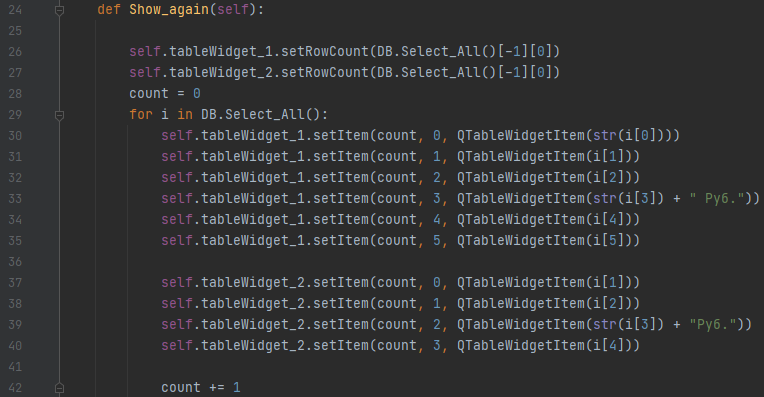


Рисунок 8.1. Запуск программы.

Далее следует рассмотреть конец программы (Рис.8), так как, хоть данная часть кода и стоит в конце, выполнение этой части идёт сразу после подключения модулей. Сначала идёт инициализация функции **main**, в этой функции создаётся переменная, которая используя модуль **QtWidgets** создаёт новый экземпляр **QApplication**, которому передаётся аргументы командной строки. Далее в переменную **Window** передаётся класс, в котором описана логика главного окна. Затем, эта переменная получает команду “Показать окно”, и потом происходит закрытие программы. Хоть в коде и написано, что нужно закрыть программу, этого не произойдёт, тогда может показаться, что эта строка не нужно, но это не так, потому что, если убрать эту строчку кода, то программа запустит окно и сразу же закроет его, другими словами, эта строка нужна для того, чтобы программа закрывала программу только по команде пользователя. Последние две строчки кода нужны для того чтобы приложение не запускалось если его импортируют.

Рисунок 9. Главное окно.

Рассмотрим следующую часть кода (Рис. 9). В этой части кода описывается логика главного окна. В класс с логикой главного окна передаться параметр QtWidgets, который отвечает за создание главного окна, и, созданный нами, модуль с дизайном для главного окна. Далее используется функция, называемая конструктором классов. Затем идёт строка, без которой невозможно бы было обращаться к переменным, которые были инициализированы в модуле дизайна. Следующая строка нужна для инициализации нашего дизайна. Далее идёт строка, без которой вывод содержимого БД нужно было бы получать только по нажатию кнопки, а с помощью данной функции модуля PyQt5 при запуске программы сразу запускается функция отвечающая за отображение содержимого БД. Следующие 5 строк кода задают логику действия при нажатии на кнопки. Следует обратить внимание на последнюю строку данной части кода, так как если для предыдущих кнопок в данном классе описаны свои функции, то для последней кнопки она не нужна, потому что последняя кнопка закрывает программу и достаточно передать действие, для оставшихся же четырёх кнопок, как и было сказано раньше, заданы свои функции, которые рассмотрим далее.

Рисунок 9.1. Функция для кнопки обновления.

Рассмотрим функцию для первой кнопки, “Обновление”. Сначала задаётся количество строк для таблицы, это нужно из-за того, что таблица PyQt5 отрисовалась с правильным количеством строк, без этого при выполнении следующего кода таблица будет пустой. Стоит упомянуть, что получаемые результаты запросов от БД имеют вид кортежа в списке из-за этого следует правильно выбирать элементы, например, в этой части берётся последнее значение первого столбца, в котором содержится нумерация записей. Далее задаётся переменную “счётчик”, для изменения строки, в которую будем передавать значение получение от функции запроса к БД. В следующей строке начинаем цикл в списке, а затем для каждого элемента таблиц задаём нужное значение. Рассмотрим на примере первой таблицы. Здесь, для первого элемента таблицы, находящегося на позиции 0,0 задаётся значение из БД и так далее, затем в конце цикла “счётчик” увеличивается на единицу и всё начинается заново, только теперь для следующей строки.

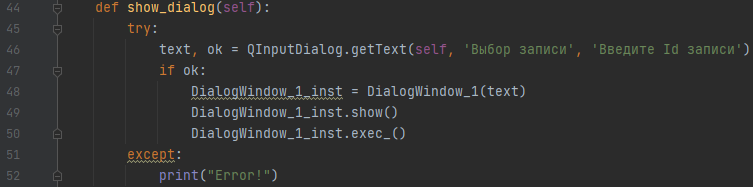


Рисунок 9.2. Функция изменения записи.

На рисунке 9.2 показана функция для кнопки изменения записи. В этой функции инициализируются две переменные которые принимаю значение вызываемого виджета, в атрибуты которого записывается сначала название окна, а затем надпись, создание двух переменных связанно с особенностью видждета, она заключается в том, что при нажатии кнопки “Ок”, первая переменная получает значение введённых пользователем данных, а вторая переменная получает логическое значение “Истина”. Затем ели пользователь нажимает кнопку “Ок”, то программа выполняет следующий блок кода, создаётся новая переменная, которая получает значение класса первого, созданного нами ранее, диалогового окна, в который передаётся значение, введённое пользователем. Далее идёт запуск нового окна, как это было ранее с главным окном в функции **main**. Данная функция обёрнута в конструкцию исключения, это сделано из-за того, что при вводе в текстовое поле виджета номера не существующего в таблице или чего-то кроме номера записи (например, “Привет”), программа закрывается с критической ошибкой.

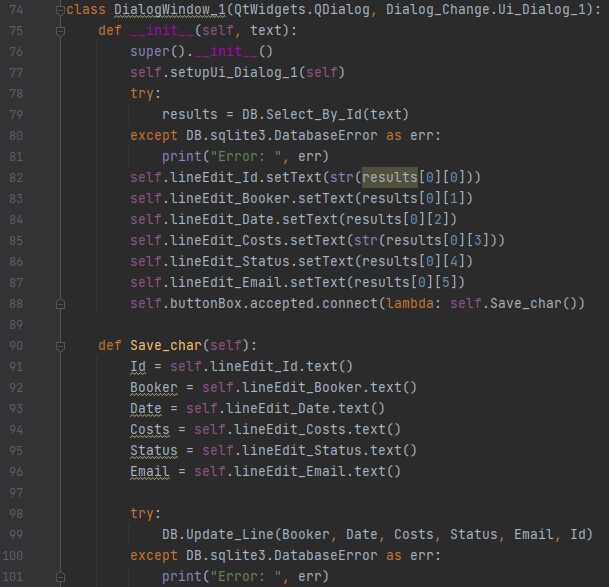


Рисунок 10. Класс первого диалогового окна.

Рассмотрим рисунок 10, на котором показан класс диалогового окна, вызванный нами ранее, через встроенное диалоговое окно, в функции изменения записи. Сначала всё так же, как и в начале класса главного окна, затем вызываем функцию, описанную ранее, она возвращает всю строку, по Id введённому ранее пользователем, и передаёт значение элементов строки в изменяемые поля для того, чтобы пользователь мог изменить элементы БД. Затем, при нажатии пользователем на кнопку “Ок”, запускается функция, в которой все данные которые были изменены, или возможно остались неизменны записываются в переменные. Далее, с помощью функции происходит изменение записей в БД и сохранение. Можно заметить, что при вызове функций SQL запросов используются исключающие конструкции, это нужно для того, что пользователь может допустить ошибку и ввести неправильные данные (Например, в строку с оплатой попадёт не числовое значение, а буквенное), если убрать данную конструкцию, то приложение закроется с критической ошибкой.

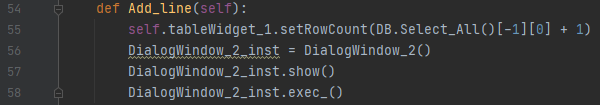


Рисунок 9.3. Функция добавления записи.

Вернёмся к основному классу и рассмотрим функцию добавления новой записи (Рис. 9.3), эта функция достаточно простая, сначала добавляем дополнительную строку, а затем по аналогии с предыдущей функцией вызываем диалоговое окно, которое рассмотрим далее.

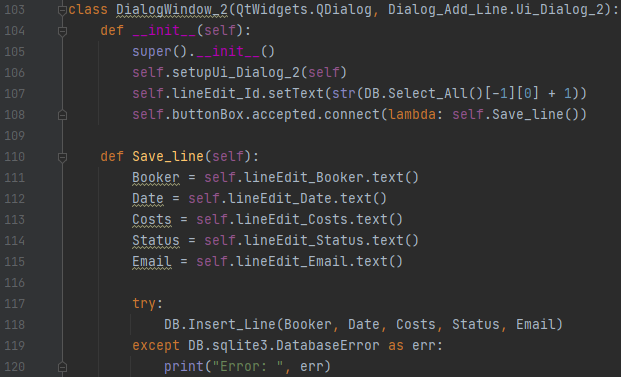


Рисунок 11. Класс второго Диалогового окна.

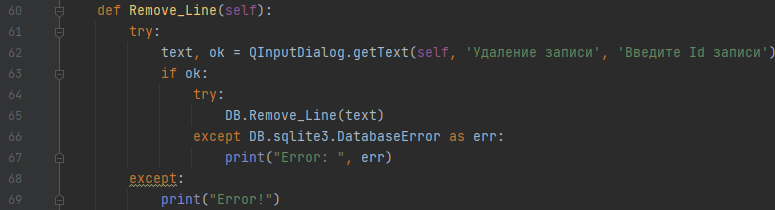
По аналогии с первым диалоговым окном, второе начинается похожим образом, однако здесь идёт добавление единицы к последнему Id, это используется для правильного отображения нумерации строк. Далее, как и в первом диалоговом окне считываются введённые значения и передаются в функцию SQL запроса, однако в этот раз не нужно выводить значения строки.

Рисунок 9.4. Функция удаления записи.

И наконец последняя функция, которая является достаточно простой по сравнению с предыдущими, потому что, хоть она и похожа на функцию изменения записи, в данной функции не приходиться вызывать дополнительное диалоговое и достаточно воспользоваться встроенным диалоговым окном, а также не возникает никаких критических ошибок при вводе нестандартных значений.

На этом основная структура программы заканчивается, но осталось сделать график поставок, так как сейчас на вкладке “График поставок” находится календарь, на котором отмечена только сегодняшняя дата, и краткая таблица, дублирующая записи в БД для удобства.

# 2.5. Добавление визуализации для графика поставок.

Для реализации данной задумки создадим ещё один модуль для нашей программы, в котором будет прописана логика отрисовки для определённых дат в календаре.



Рисунок 12. Подключение модулей для календаря.

Сначала следует подключит уже известные модули, кроме QDate, этот модуль преобразует дату в объект, который сможет быть читаемым отрисовщиком календаря.

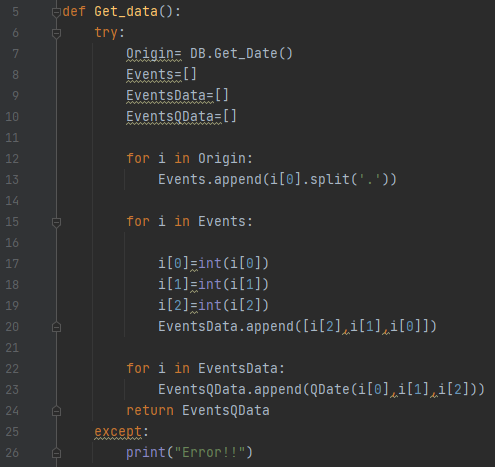


Рисунок 12.1. Функция преобразования даты.

На рисунке 12.1 показана функция, которую приходиться использовать, для получения корректных данных. Это происходит из-за того, что всем привычный формат даты (день.месяц.год) не подходит для виджета календаря, ему нужен формат год.месяц.день, и для преобразования формата даты в корректный для календаря, используется данная функция, в которой принимаются все записанные в БД значения дат, а затем их приходиться сначала доставать из кортежа, при этом разбивая на отдельные числа, и только уже потом, попутно меняя их тип из строки в целые числа, записывать их в корректном виде в список.

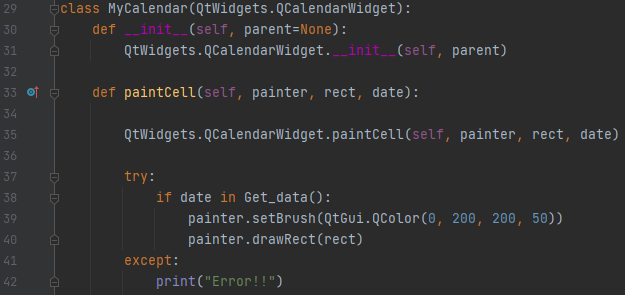


Рисунок 12.2. Класс дополнения к календарю.

Далее идёт класс дополнения к календарю, в котором инициализируется виджет календаря и запускается встроенная функция отрисовщика, которая проверяет, если дата корректна, то данная дата в календаре помечается. Таким образом календарь преображается и становиться более удобны в использовании (См. Рис. 12.3).

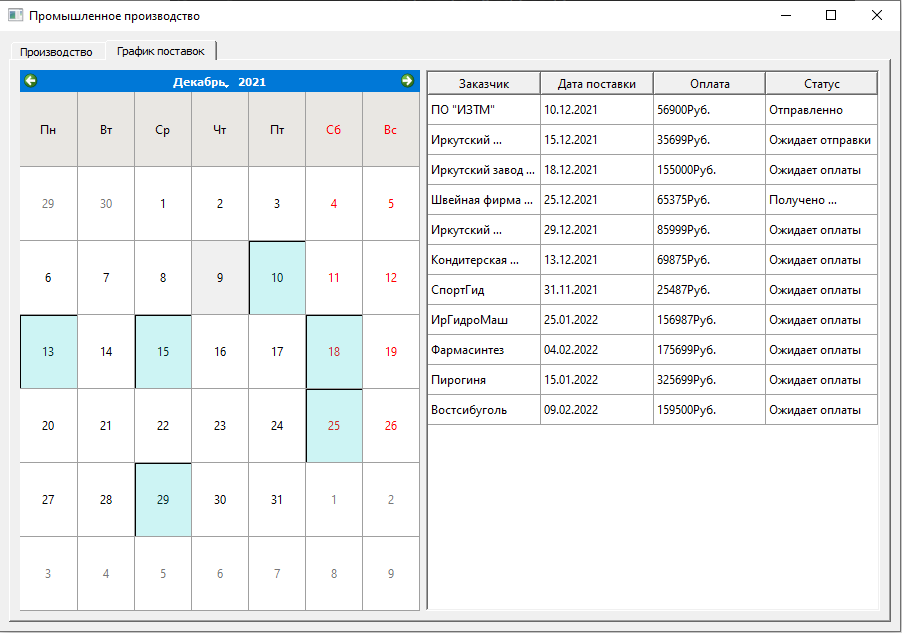


Рисунок 12.3. Итоговый вид календаря.

# Заключение

В ходе проделанной работы была создана программа для промышленного производства, которая выполняет функции хранения и обработки информации. Хоть разработанная программа и отличается от похожих программ, что существуют сейчас, но имеет свои преимущества по отношению к таким программам, например, открытый код, это означает что любой может убедиться в том, что делает данная программа, не несёт ли она в себе какой-либо вредоносный код или не отправляет ли она данные предприятия третьим лицам. Большим плюсом так же является специализация программы именно под данную сферу деятельности человека, что несёт за собой плюс для конечного пользователя, так как не нужно разбираться в нагромождении всяких иконок и вкладок в интерфейсе.

При написании данного курсового проекта были получены навыки работы с объектно-ориентированным программированием, работы с различными модулями python, а также, были получены базовые знания SQL синтаксиса для работы с БД.

# Список используемых источников

1. И.П. Карпова БАЗЫ ДАННЫХ [Электронный ресурс]: М., Питер: 2013. – 240 с. – Режим доступа:  
<https://publications.hse.ru/mirror/pubs/share/direct/259052819> (дата обращения: 29.11.2021).

2. Памятка/шпаргалка по SQL [Электронный ресурс; Режим доступа: https://habr.com/ru/post/564390/] (дата обращения 26.11.2021);

3. Система управления базами данных [Электронный ресурс; Режим доступа: https://ru.wikipedia.org/wiki/Система\_управления\_базами\_данных] (дата обращения: 30.11.2021).

4. Документация по Sqlite3 [Электронный ресурс; Режим доступа: https://docs.python.org/3/library/sqlite3.html] (дата обращения: 03.11.2021).

5. Документация по PyQt5 [Электронный ресурс; Режим доступа: https://doc.qt.io/qtforpython/] (дата обращения: 03.11.2021).

# Приложение 1 ­– Исходный код файла с функциями SQL запросов к БД.

import sqlite3

def Select\_All():

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("SELECT \* FROM product")

results = cursor.fetchall()

return results

conn.close()

def Select\_By\_Id(text):

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("SELECT \* from product WHERE ID={}".format(text))

results = cursor.fetchall()

return results

conn.close()

def Update\_Line(Booker, Date, Costs, Status, Email, Id):

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("update product set Заказчик='{}',Дата='{}',Оплата='{}',Статус='{}',Email='{}' where ID={}"

.format(Booker, Date, Costs, Status, Email, Id))

conn.commit()

conn.close()

def Insert\_Line(Booker, Date, Costs, Status, Email):

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("insert into product (Заказчик, Дата, Оплата, Статус, Email) values ('{}','{}',{},'{}','{}')"

.format(Booker, Date, Costs, Status, Email))

conn.commit()

conn.close()

def Remove\_Line(text):

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("delete from product where ID={}"

.format(text))

conn.commit()

conn.close()

def Get\_Date():

conn = sqlite3.connect('DataBase/Product\_DB.sqlite')

cursor = conn.cursor()

cursor.execute("SELECT Дата FROM product")

results = cursor.fetchall()

return results

conn.close()

# Приложение 2 ­– Исходный код файла дизайна главного окна.

from PyQt5 import QtCore, QtGui, QtWidgets

import Calendar

class Ui\_MainWindow(object):

def setupUi\_Main(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.setEnabled(True)

MainWindow.resize(900, 600)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())

MainWindow.setSizePolicy(sizePolicy)

MainWindow.setBaseSize(QtCore.QSize(800, 600))

MainWindow.setAnimated(True)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

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

self.verticalLayout.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)

self.verticalLayout.setObjectName("verticalLayout")

self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)

sizePolicy=QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.tabWidget.sizePolicy().hasHeightForWidth())

self.tabWidget.setSizePolicy(sizePolicy)

self.tabWidget.setObjectName("tabWidget")

self.tab = QtWidgets.QWidget()

self.tab.setAutoFillBackground(True)

self.tab.setObjectName("tab")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.tab)

self.verticalLayout\_2.setSizeConstraint(QtWidgets.QLayout.SetMaximumSize)

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

self.tableWidget\_1 = QtWidgets.QTableWidget(self.tab)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.tableWidget\_1.sizePolicy().hasHeightForWidth())

self.tableWidget\_1.setSizePolicy(sizePolicy)

self.tableWidget\_1.setTabletTracking(False)

self.tableWidget\_1.setContextMenuPolicy(QtCore.Qt.NoContextMenu)

self.tableWidget\_1.setLineWidth(1)

self.tableWidget\_1.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tableWidget\_1.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.tableWidget\_1.setDragEnabled(False)

self.tableWidget\_1.setColumnCount(6)

self.tableWidget\_1.setObjectName("tableWidget\_1")

self.tableWidget\_1.setRowCount(0)

self.tableWidget\_1.horizontalHeader().setVisible(False)

self.tableWidget\_1.horizontalHeader().setCascadingSectionResizes  
(True)

self.tableWidget\_1.horizontalHeader().setDefaultSectionSize(128)

self.tableWidget\_1.horizontalHeader().setHighlightSections(True)

self.tableWidget\_1.horizontalHeader().setMinimumSectionSize(40)

self.tableWidget\_1.horizontalHeader().setStretchLastSection(False)

self.tableWidget\_1.verticalHeader().setVisible(False)

self.tableWidget\_1.verticalHeader().setCascadingSectionResizes(True)

self.tableWidget\_1.verticalHeader().setHighlightSections(True)

self.tableWidget\_1.verticalHeader().setSortIndicatorShown(False)

self.verticalLayout\_2.addWidget(self.tableWidget\_1)

self.Refresh = QtWidgets.QPushButton(self.tab)

self.Refresh.setObjectName("Refresh")

self.verticalLayout\_2.addWidget(self.Refresh)

self.AddLine = QtWidgets.QPushButton(self.tab)

self.AddLine.setObjectName("AddLine")

self.verticalLayout\_2.addWidget(self.AddLine)

self.ChangeLine = QtWidgets.QPushButton(self.tab)

self.ChangeLine.setObjectName("ChangeLine")

self.verticalLayout\_2.addWidget(self.ChangeLine)

self.RemoveLine = QtWidgets.QPushButton(self.tab)

self.RemoveLine.setObjectName("RemoveLine")

self.verticalLayout\_2.addWidget(self.RemoveLine)

self.Exit = QtWidgets.QPushButton(self.tab)

self.Exit.setObjectName("Exit")

self.verticalLayout\_2.addWidget(self.Exit)

self.tabWidget.addTab(self.tab, "")

self.tab\_2 = QtWidgets.QWidget()

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.tab\_2.sizePolicy().hasHeightForWidth())

self.tab\_2.setSizePolicy(sizePolicy)

self.tab\_2.setAutoFillBackground(True)

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

self.horizontalLayout = QtWidgets.QHBoxLayout(self.tab\_2)

self.horizontalLayout.setSizeConstraint(QtWidgets.QLayout.SetNoConstraint)

self.horizontalLayout.setObjectName("horizontalLayout")

self.Calendar = Calendar.MyCalendar(self.tab\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.Calendar.sizePolicy().hasHeightForWidth())

self.Calendar.setSizePolicy(sizePolicy)

self.Calendar.setMinimumSize(QtCore.QSize(400, 0))

self.Calendar.setGridVisible(True)

self.Calendar.setVerticalHeaderFormat(QtWidgets.QCalendarWidget.NoVerticalHeader)

self.Calendar.setNavigationBarVisible(True)

self.Calendar.setObjectName("Calendar")

self.horizontalLayout.addWidget(self.Calendar)

self.tableWidget\_2 = QtWidgets.QTableWidget(self.tab\_2)

sizePolicy=QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.tableWidget\_2.sizePolicy().hasHeightForWidth())

self.tableWidget\_2.setSizePolicy(sizePolicy)

self.tableWidget\_2.setTabletTracking(False)

self.tableWidget\_2.setContextMenuPolicy  
(QtCore.Qt.DefaultContextMenu)

self.tableWidget\_2.setLineWidth(1)

self.tableWidget\_2.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tableWidget\_2.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.tableWidget\_2.setDragEnabled(False)

self.tableWidget\_2.setColumnCount(4)

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

self.tableWidget\_2.setRowCount(0)

self.tableWidget\_2.horizontalHeader().setVisible(False)

self.tableWidget\_2.horizontalHeader().setCascadingSectionResizes  
(False)

self.tableWidget\_2.horizontalHeader().setDefaultSectionSize(128)

self.tableWidget\_2.horizontalHeader().setHighlightSections(False)

self.tableWidget\_2.horizontalHeader().setSortIndicatorShown(False)

self.tableWidget\_2.horizontalHeader().setStretchLastSection(False)

self.tableWidget\_2.verticalHeader().setVisible(False)

self.tableWidget\_2.verticalHeader().setCascadingSectionResizes(True)

self.tableWidget\_2.verticalHeader().setHighlightSections(True)

self.tableWidget\_2.verticalHeader().setSortIndicatorShown(False)

self.horizontalLayout.addWidget(self.tableWidget\_2)

self.tabWidget.addTab(self.tab\_2, "")

self.verticalLayout.addWidget(self.tabWidget)

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

self.tabWidget.setCurrentIndex(0)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

# Устанавливаем фиксированый размер для колонок

self.tableWidget\_1.setHorizontalHeaderLabels(["Id", "Заказчик", "Дата поставки", "Оплата", "Статус", "Email"]) # задаём названия колонок

self.tableWidget\_2.setHorizontalHeaderLabels(["Заказчик", "Дата поставки", "Оплата", "Статус"]) # задаём названия колонок

self.tableWidget\_1.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch) # !!!

self.tableWidget\_1.horizontalHeader().setDefaultSectionSize(40)

self.tableWidget\_1.horizontalHeader().setSectionResizeMode(0, QtWidgets.QHeaderView.Fixed)

self.tableWidget\_2.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch) # !!!

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

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

self.Refresh.setText(\_translate("MainWindow", "Обновить"))

self.AddLine.setText(\_translate("MainWindow", "Добавить запись"))

self.ChangeLine.setText(\_translate("MainWindow", "Изменить запись"))

self.RemoveLine.setText(\_translate("MainWindow", "Удалить запись"))

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

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate("MainWindow", "Производство"))

self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate("MainWindow", "График поставок"))

# Приложение 3 – Исходный код первого диалогового окна.

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog\_1(object):

def setupUi\_Dialog\_1(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(742, 102)

self.verticalLayout\_7 = QtWidgets.QVBoxLayout(Dialog)

self.verticalLayout\_7.setObjectName("verticalLayout\_7")

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.label\_Id = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Id.setFont(font)

self.label\_Id.setFrameShape(QtWidgets.QFrame.NoFrame)

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

self.verticalLayout.addWidget(self.label\_Id)

self.lineEdit\_Id = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Id.setMinimumSize(QtCore.QSize(31, 20))

self.lineEdit\_Id.setMaximumSize(QtCore.QSize(31, 20))

self.lineEdit\_Id.setText("")

self.lineEdit\_Id.setFrame(True)

self.lineEdit\_Id.setAlignment(QtCore.Qt.AlignCenter)

self.lineEdit\_Id.setReadOnly(True)

self.lineEdit\_Id.setObjectName("lineEdit\_Id")

self.verticalLayout.addWidget(self.lineEdit\_Id)

self.horizontalLayout.addLayout(self.verticalLayout)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

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

self.label\_Booker = QtWidgets.QLabel(Dialog)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_Booker.sizePolicy().hasHeightForWidth())

self.label\_Booker.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Booker.setFont(font)

self.label\_Booker.setFrameShape(QtWidgets.QFrame.NoFrame)

self.label\_Booker.setTextFormat(QtCore.Qt.AutoText)

self.label\_Booker.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_2.addWidget(self.label\_Booker)

self.lineEdit\_Booker = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Booker.setMinimumSize(QtCore.QSize(190, 0))

self.lineEdit\_Booker.setText("")

self.lineEdit\_Booker.setObjectName("lineEdit\_Booker")

self.verticalLayout\_2.addWidget(self.lineEdit\_Booker)

self.horizontalLayout.addLayout(self.verticalLayout\_2)

self.verticalLayout\_3 = QtWidgets.QVBoxLayout()

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

self.label\_Date = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Date.setFont(font)

self.label\_Date.setContextMenuPolicy(QtCore.Qt.CustomContextMenu)

self.label\_Date.setLayoutDirection(QtCore.Qt.RightToLeft)

self.label\_Date.setAutoFillBackground(False)

self.label\_Date.setInputMethodHints(QtCore.Qt.ImhNone)

self.label\_Date.setFrameShape(QtWidgets.QFrame.NoFrame)

self.label\_Date.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_3.addWidget(self.label\_Date)

self.lineEdit\_Date = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Date.setMinimumSize(QtCore.QSize(62, 0))

self.lineEdit\_Date.setText("")

self.lineEdit\_Date.setObjectName("lineEdit\_Date")

self.verticalLayout\_3.addWidget(self.lineEdit\_Date)

self.horizontalLayout.addLayout(self.verticalLayout\_3)

self.verticalLayout\_4 = QtWidgets.QVBoxLayout()

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

self.label\_Costs = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Costs.setFont(font)

self.label\_Costs.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_4.addWidget(self.label\_Costs)

self.lineEdit\_Costs = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Costs.setText("")

self.lineEdit\_Costs.setObjectName("lineEdit\_Costs")

self.verticalLayout\_4.addWidget(self.lineEdit\_Costs)

self.horizontalLayout.addLayout(self.verticalLayout\_4)

self.verticalLayout\_5 = QtWidgets.QVBoxLayout()

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

self.label\_Status = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Status.setFont(font)

self.label\_Status.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_5.addWidget(self.label\_Status)

self.lineEdit\_Status = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Status.setMinimumSize(QtCore.QSize(115, 0))

self.lineEdit\_Status.setText("")

self.lineEdit\_Status.setObjectName("lineEdit\_Status")

self.verticalLayout\_5.addWidget(self.lineEdit\_Status)

self.horizontalLayout.addLayout(self.verticalLayout\_5)

self.verticalLayout\_6 = QtWidgets.QVBoxLayout()

self.verticalLayout\_6.setObjectName("verticalLayout\_6")

self.label\_Email = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Email.setFont(font)

self.label\_Email.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_6.addWidget(self.label\_Email)

self.lineEdit\_Email = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Email.setMinimumSize(QtCore.QSize(133, 0))

self.lineEdit\_Email.setText("")

self.lineEdit\_Email.setObjectName("lineEdit\_Email")

self.verticalLayout\_6.addWidget(self.lineEdit\_Email)

self.horizontalLayout.addLayout(self.verticalLayout\_6)

self.verticalLayout\_7.addLayout(self.horizontalLayout)

self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)

self.buttonBox.setOrientation(QtCore.Qt.Horizontal)

self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)

self.buttonBox.setCenterButtons(False)

self.buttonBox.setObjectName("buttonBox")

self.verticalLayout\_7.addWidget(self.buttonBox)

self.retranslateUi(Dialog)

self.buttonBox.accepted.connect(Dialog.accept)

self.buttonBox.rejected.connect(Dialog.reject)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Изменить запись"))

self.label\_Id.setText(\_translate("Dialog", "Id"))

self.label\_Booker.setText(\_translate("Dialog", "Заказчик"))

self.label\_Date.setText(\_translate("Dialog", "Дата поставки"))

self.label\_Costs.setText(\_translate("Dialog", "Оплата"))

self.label\_Status.setText(\_translate("Dialog", "Статус"))

self.label\_Email.setText(\_translate("Dialog", "Email"))

# Приложение 4 –Исходный код второго диалогового окна.

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog\_2(object):

def setupUi\_Dialog\_2(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(813, 102)

self.verticalLayout\_7 = QtWidgets.QVBoxLayout(Dialog)

self.verticalLayout\_7.setObjectName("verticalLayout\_7")

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

self.verticalLayout = QtWidgets.QVBoxLayout()

self.verticalLayout.setObjectName("verticalLayout")

self.label\_Id = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Id.setFont(font)

self.label\_Id.setFrameShape(QtWidgets.QFrame.NoFrame)

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

self.verticalLayout.addWidget(self.label\_Id)

self.lineEdit\_Id = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Id.setMinimumSize(QtCore.QSize(31, 20))

self.lineEdit\_Id.setMaximumSize(QtCore.QSize(31, 20))

self.lineEdit\_Id.setText("")

self.lineEdit\_Id.setFrame(True)

self.lineEdit\_Id.setAlignment(QtCore.Qt.AlignCenter)

self.lineEdit\_Id.setReadOnly(True)

self.lineEdit\_Id.setObjectName("lineEdit\_Id")

self.verticalLayout.addWidget(self.lineEdit\_Id)

self.horizontalLayout.addLayout(self.verticalLayout)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout()

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

self.label\_Booker = QtWidgets.QLabel(Dialog)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_Booker.sizePolicy().hasHeightForWidth())

self.label\_Booker.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Booker.setFont(font)

self.label\_Booker.setFrameShape(QtWidgets.QFrame.NoFrame)

self.label\_Booker.setTextFormat(QtCore.Qt.AutoText)

self.label\_Booker.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_2.addWidget(self.label\_Booker)

self.lineEdit\_Booker = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Booker.setMinimumSize(QtCore.QSize(190, 0))

self.lineEdit\_Booker.setText("")

self.lineEdit\_Booker.setObjectName("lineEdit\_Booker")

self.verticalLayout\_2.addWidget(self.lineEdit\_Booker)

self.horizontalLayout.addLayout(self.verticalLayout\_2)

self.verticalLayout\_3 = QtWidgets.QVBoxLayout()

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

self.label\_Date = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Date.setFont(font)

self.label\_Date.setContextMenuPolicy(QtCore.Qt.CustomContextMenu)

self.label\_Date.setLayoutDirection(QtCore.Qt.RightToLeft)

self.label\_Date.setAutoFillBackground(False)

self.label\_Date.setInputMethodHints(QtCore.Qt.ImhNone)

self.label\_Date.setFrameShape(QtWidgets.QFrame.NoFrame)

self.label\_Date.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_3.addWidget(self.label\_Date)

self.lineEdit\_Date = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Date.setMinimumSize(QtCore.QSize(62, 0))

self.lineEdit\_Date.setText("")

self.lineEdit\_Date.setObjectName("lineEdit\_Date")

self.verticalLayout\_3.addWidget(self.lineEdit\_Date)

self.horizontalLayout.addLayout(self.verticalLayout\_3)

self.verticalLayout\_4 = QtWidgets.QVBoxLayout()

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

self.label\_Costs = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Costs.setFont(font)

self.label\_Costs.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_4.addWidget(self.label\_Costs)

self.lineEdit\_Costs = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Costs.setText("")

self.lineEdit\_Costs.setObjectName("lineEdit\_Costs")

self.verticalLayout\_4.addWidget(self.lineEdit\_Costs)

self.horizontalLayout.addLayout(self.verticalLayout\_4)

self.verticalLayout\_5 = QtWidgets.QVBoxLayout()

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

self.label\_Status = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Status.setFont(font)

self.label\_Status.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_5.addWidget(self.label\_Status)

self.lineEdit\_Status = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Status.setMinimumSize(QtCore.QSize(115, 0))

self.lineEdit\_Status.setText("")

self.lineEdit\_Status.setObjectName("lineEdit\_Status")

self.verticalLayout\_5.addWidget(self.lineEdit\_Status)

self.horizontalLayout.addLayout(self.verticalLayout\_5)

self.verticalLayout\_6 = QtWidgets.QVBoxLayout()

self.verticalLayout\_6.setObjectName("verticalLayout\_6")

self.label\_Email = QtWidgets.QLabel(Dialog)

font = QtGui.QFont()

font.setPointSize(16)

self.label\_Email.setFont(font)

self.label\_Email.setAlignment(QtCore.Qt.AlignCenter)

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

self.verticalLayout\_6.addWidget(self.label\_Email)

self.lineEdit\_Email = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_Email.setMinimumSize(QtCore.QSize(133, 0))

self.lineEdit\_Email.setText("")

self.lineEdit\_Email.setObjectName("lineEdit\_Email")

self.verticalLayout\_6.addWidget(self.lineEdit\_Email)

self.horizontalLayout.addLayout(self.verticalLayout\_6)

self.verticalLayout\_7.addLayout(self.horizontalLayout)

self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)

self.buttonBox.setOrientation(QtCore.Qt.Horizontal)

self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)

self.buttonBox.setCenterButtons(False)

self.buttonBox.setObjectName("buttonBox")

self.verticalLayout\_7.addWidget(self.buttonBox)

self.retranslateUi(Dialog)

self.buttonBox.accepted.connect(Dialog.accept)

self.buttonBox.rejected.connect(Dialog.reject)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Добавить запись"))

self.label\_Id.setText(\_translate("Dialog", "Id"))

self.label\_Booker.setText(\_translate("Dialog", "Заказчик"))

self.label\_Date.setText(\_translate("Dialog", "Дата поставки"))

self.label\_Costs.setText(\_translate("Dialog", "Оплата"))

self.label\_Status.setText(\_translate("Dialog", "Статус"))

self.label\_Email.setText(\_translate("Dialog", "Email"))

# Приложение 5 – Исходный код дополнения к календарю.

from PyQt5 import QtGui, QtWidgets

from PyQt5.QtCore import QDate

import DB

def Get\_data():

try:

Origin = DB.Get\_Date()

Events = []

EventsData = []

EventsQData = []

for i in Origin:

Events.append(i[0].split('.'))

for i in Events:

i[0] = int(i[0])

i[1] = int(i[1])

i[2] = int(i[2])

EventsData.append([i[2], i[1], i[0]])

for i in EventsData:

EventsQData.append(QDate(i[0], i[1], i[2]))

return EventsQData

except:

print("Error!!")

class MyCalendar(QtWidgets.QCalendarWidget):

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

QtWidgets.QCalendarWidget.\_\_init\_\_(self, parent)

def paintCell(self, painter, rect, date):

try:

QtWidgets.QCalendarWidget.paintCell(self, painter, rect, date)

if date in Get\_data():

painter.setBrush(QtGui.QColor(0, 200, 200, 50))

painter.drawRect(rect)

except:

print("Error!!")