

**Колледж космического машиностроения и технологий**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине МДК.02.02. Технология разработки и защиты баз данных

**РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ЦВЕТОЧНЫМ МАГАЗИНОМ «РОМАШКА»**

Пояснительная записка

КП.09.02.03.23.13ПЗ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся группы | | П1-19 | Новоселова В.А. | |
| Руководитель курсового проекта | | | Стрельников С.Д. | |
| Результат защиты |  | | |  |

г. Королёв, 2023 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 5](#_Toc131116586)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc131116587)

[1.1. Описание предметной области 6](#_Toc131116588)

[1.2. Описание существующей разработки 6](#_Toc131116589)

[2. ПРОЕКТНАЯ ЧАСТЬ 9](#_Toc131116590)

[2.1. Обоснование инструментов разработки 9](#_Toc131116591)

[2.2. Проектирование сценария работы проекта 11](#_Toc131116592)

[2.3. Разработка диаграммы классов 12](#_Toc131116593)

[2.4. Разработка ER-диаграммы баз данных 13](#_Toc131116594)

[2.5. Описание модулей проекта 13](#_Toc131116595)

[2.6. Тестирование проекта 14](#_Toc131116596)

[2.7. Руководство оператора 15](#_Toc131116597)

[ЗАКЛЮЧЕНИЕ 23](#_Toc131116598)

[Список используемых источников 24](#_Toc131116599)

[Приложение 1. 25](#_Toc131116600)

# **ВВЕДЕНИЕ**

Тема курсового проекта – «Разработка системы управления цветочным магазином «Ромашка»».

Актуальность данного курсового проекта заключается в том, что он позволяет сократить время, используемое на учет цветочной продукции и запись заказов клиентов.

Целью данного курсового проекта является разработка клиентской части десктопного приложения, которое будет взаимодействовать с базой данных.

В первой части будет рассмотрена предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты и модули, которые были разработаны и описано руководство оператора.

В заключительной части будет и заключение по курсовому проекту.

# **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **Описание предметной области**

Целью данной работы является разработка десктопного приложения для электронного учета цветочной продукции, которая имеется на складе, а также ведения заказов клиентов. В данной ситуации предлагается следующее решение поставленной задачи:

Десктопное приложение в функционал которого входит:

* Возможность записи заказов клиентов;
* Управление базой данных через визуальный интерфейс;
* Фильтр, определяющий нужное количество закупаемых цветов, для выполнения заказов;
* Составление продаваемых цветочных букетов;
* Подсчет прибыли.

## **Описание существующей разработки**

В этом разделе рассмотрены некоторые уже имеющиеся программы по данной тематике.

Практически все из них предоставляют похожий функционал:

* Управление базой данных;
* Запись заказов клиентов;
* База данных цветов.

Примеры программ:

* Florapoint

Florapoint – программный комплекс учета продаж в цветочном бизнесе: CRM, склад, аналитика. Визуальный интерфейс показан на рисунке 1.

Преимущества:

* Удобство в использовании;
* Есть пробная бесплатная версия.

Недостатки:

* Полный функционал программы открывается при помощи платной ежемесячной подписке;
* Платные интеграции;
* Низкая оптимизации скорость работы программы.

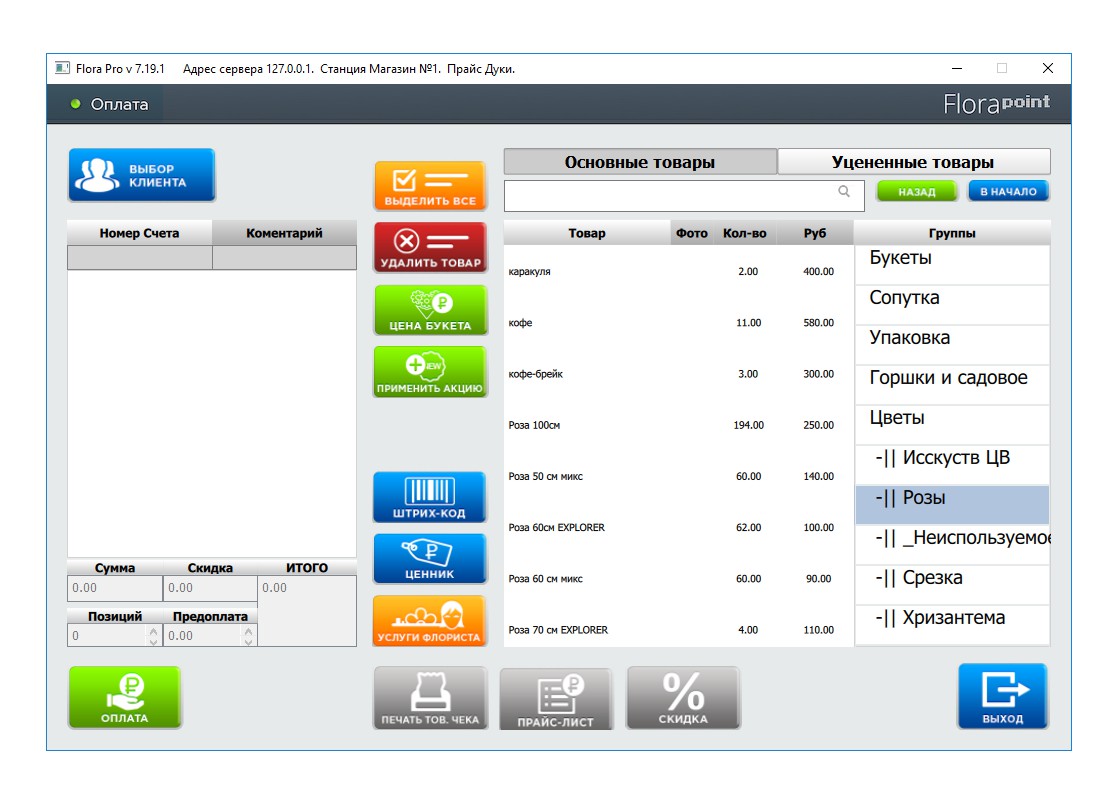


Рисунок 1. Интерфейс программы «Florapoint»

* Posiflora

Posiflora – программа позволяет автоматизировать работу цветочных салонов, интернет‑магазинов, цветочных супермаркетов, маленьких цветочных магазинов и сетей. Вы сможете управлять клиентской базой и системой лояльности, вести складской учёт, контролировать доставку, снизить расходы и повысить продажи. Визуальный интерфейс показан на рисунке 2.

Преимущества:

* Удобство в использовании;
* Есть стандартная бесплатная версия.

Недостатки:

* Полный функционал программы открывается при помощи платной ежемесячной подписке;
* В определенные дни у программы может быть сбой.

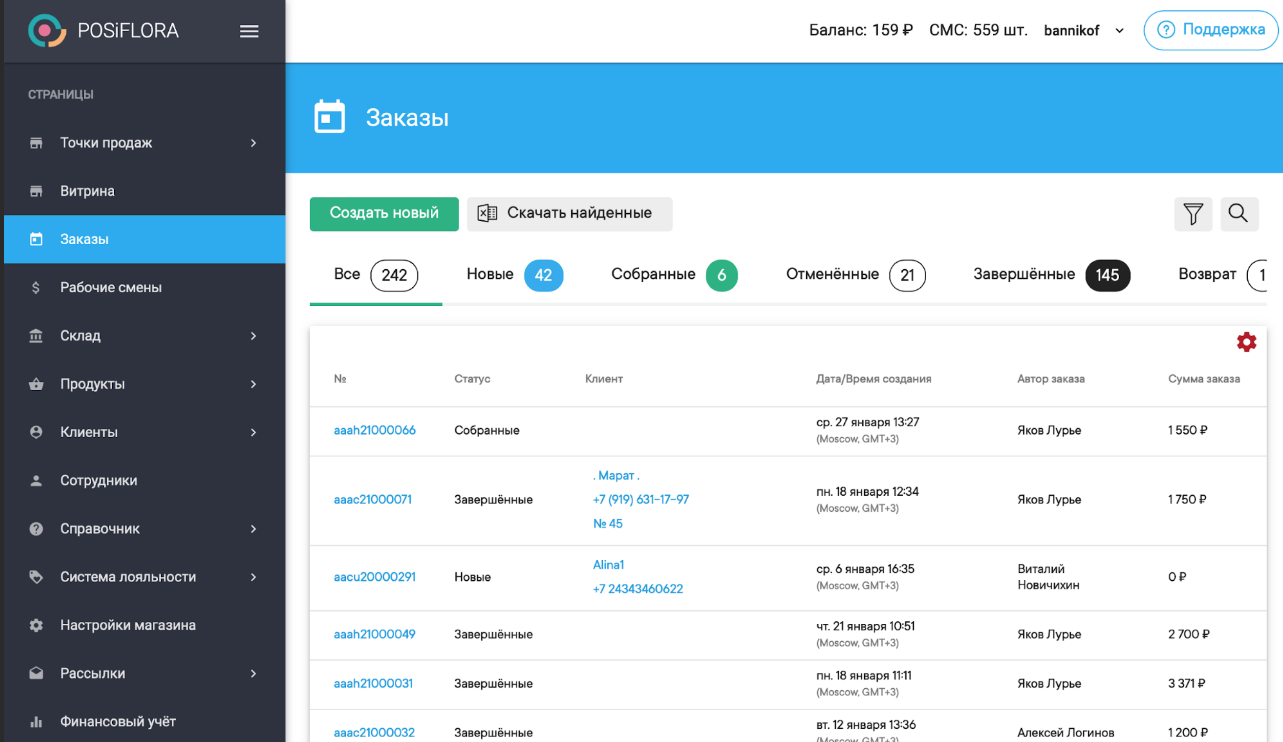


Рисунок 2. Интерфейс программы «Posiflora» 1

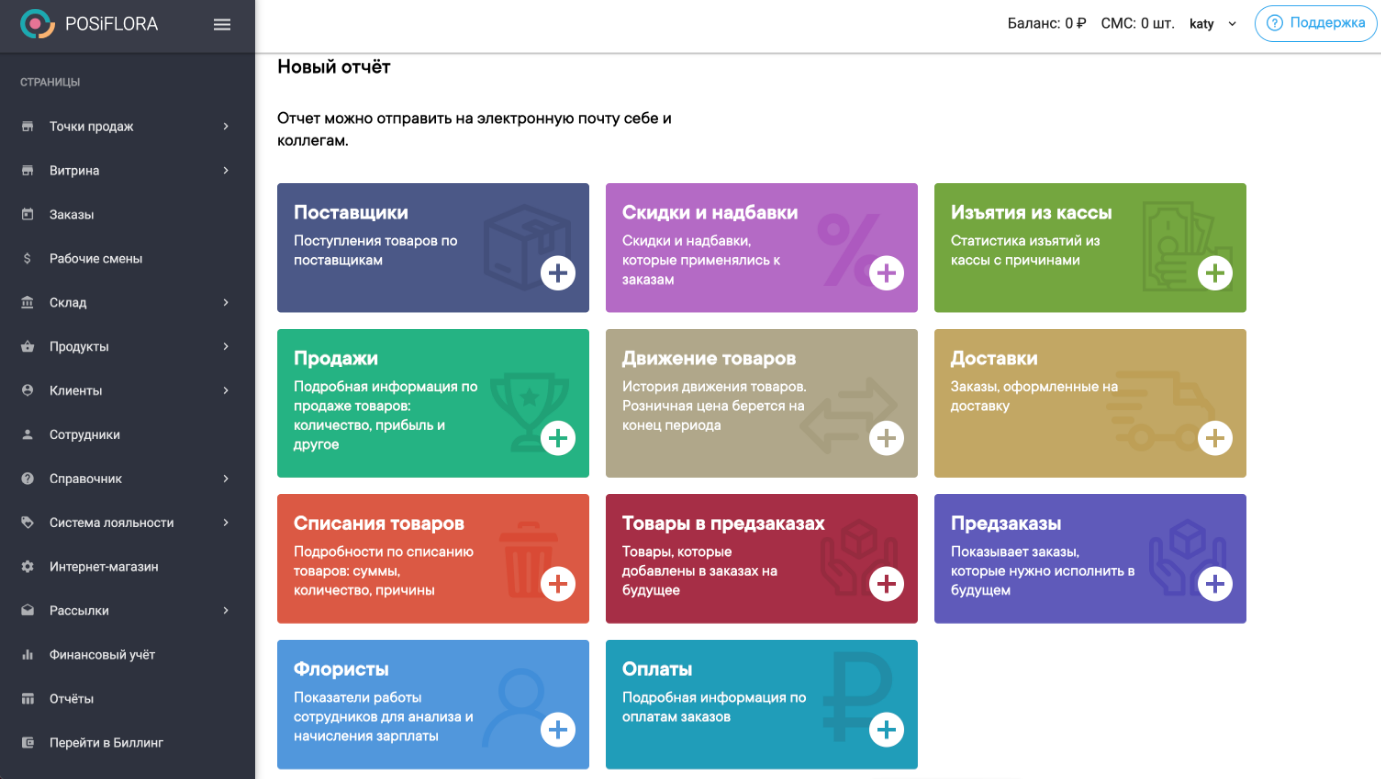


Рисунок 3. Интерфейс программы «Posiflora» 2

# **ПРОЕКТНАЯ ЧАСТЬ**

## **Обоснование инструментов разработки**

Для создания проекта, основным инструментом был выбран PyCharm: the Python IDE, потому что он является одним из лучших инструментов для разработки на Python. В качестве СУБД был выбран SQLite.

Язык Python был выбран по следующим факторам:

* Понятность кода – простое зрительное восприятие программ, написанных на этом языке.
* Наличие мощного фреймворка для разработки клиентских приложений;
* Наличие подробной документации.

Сравнение Python с другими языками программирования представлено в таблице 1.

Таблица 1. сравнения языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **C++** | **С#** | **Python** |
| Читабельность | 5 | 7 | 8 |
| Простота | 4 | 6 | 8 |
| GUI | 9 | 8 | 7 |
| Скорость разработки | 5 | 6 | 8 |
| Знание языка | 5 | 3 | 8 |
| Итого баллов | 28 | 30 | 39 |

Причины, по которым был выбран данный инструмент и СУБД:

**PyCharm**

PyCharm – это интегрированная среда разработки (IDE) для языка программирования Python, разработанная компанией JetBrains. Она предоставляет широкий набор инструментов для разработки, отладки, тестирования и управления проектами на Python. PyCharm поддерживает множество популярных фреймворков, таких как Django, Flask, PyQt5 и другие.

Включает в себя:

* Редактор кода;
* Инструменты рефакторинга кода;
* Интеграцию с системой контроля версий Git.

PyCharm делает разработку максимально продуктивной благодаря функциям автодополнения и анализа кода, мгновенной подсветке ошибок и быстрым исправлениям. Автоматические рефакторинги помогают эффективно редактировать код, а удобная навигация позволяет мгновенно перемещаться по проекту.

Возможности форматирования и настройки стиля кода помогают писать понятный код, который легко поддерживать.

**SQLite**

SQLite - это легковесная реляционная система управления базами данных, которая не требует отдельного сервера и может быть использована в качестве встроенной базы данных в приложениях. SQLite может быть использована для хранения небольших объемов данных, таких как настройки приложения, и для создания более сложных приложений, таких как браузеры и мобильные приложения.

**Преимущества SQLite**

1. Легковесность: SQLite является очень легкой базой данных, которая может быть использована в качестве встроенной базы данных в приложениях.
2. Нет необходимости в отдельном сервере: SQLite не требует отдельного сервера для работы, что делает его очень удобным для использования в небольших приложениях.
3. Транзакционность: SQLite поддерживает транзакции, что позволяет обеспечить целостность данных и защиту от ошибок.
4. Индексирование: SQLite поддерживает индексирование, что позволяет быстро находить нужные данные.
5. Ограничения целостности: SQLite поддерживает ограничения целостности, что позволяет обеспечить правильное хранение данных и защиту от ошибок.
6. Бесплатность: SQLite является бесплатной базой данных, что делает его доступным для использования в любых приложениях.

## **Проектирование сценария работы проекта**

В данном разделе приведен сценарий использования программы пользователем приложения (Рисунок 4).

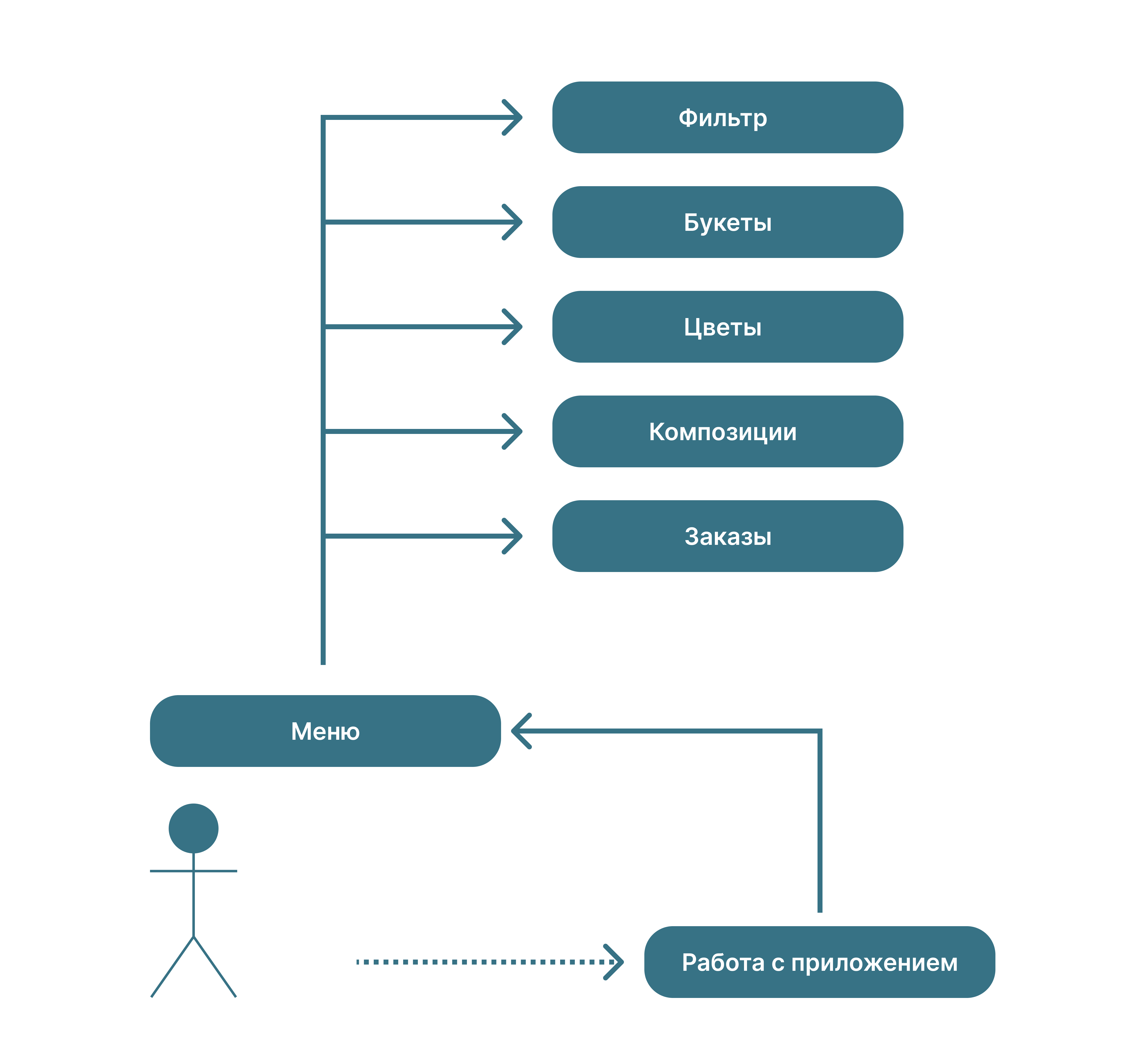


Рисунок 4. Сценарий использования

Пользователь после запуска приложения может выполнить следующие действия:

1. Управление заказами клиентов, добавление новых;
2. Управление букетами;
3. Добавление/удаление/редактирование цветов;
4. Составление цветочных композиций;
5. Фильтрация по определенным цветам и периоду.

При выходе из программы, она завершает свою работу.

## **Разработка диаграммы классов**

В данном разделе представлены все классы, использующиеся в проекте, а также их отношения между собой (Рисунок 5).

1. Main: входная точка;
2. OrderSystem: класс содержит набор функций, импортируемых модулей, реализующих логику приложения, взаимодействие с БД;
   1. Load\_data\_composition - загрузка данных с таблицы composition из БД sql Db.db. Срабатывает при нажатии на кнопку «Загрузить» на вкладке «Композиции»;
   2. Load\_data\_flowers\_flavors - загрузка данных с таблицы flowers и flavors из БД sql Db.db. Срабатывает при нажатии на кнопку «Загрузить» на вкладке «Цветы» и «Букеты»;
   3. Add\_new\_row – добавление новой строки. Срабатывает при нажатии на кнопку «Добавить»;
   4. Delete\_row – удаление выбранной строки. Срабатывает при нажатии на кнопку «Удалить»;
   5. Save\_data\_flowers\_flavors – сохранение данных из таблицы flowers или flavors в таблицу sql. Срабатывает при нажатии на кнопку «Сохранить» на вкладке «Цветы» или «Букеты»;
   6. Load\_data\_order – загрузка данных с таблицы orders из БД sql Db.db. Срабатывает при нажатии на кнопку «Загрузить» на вкладке «Заказы»;
   7. List\_flowers – формирование всего списка цветов для добавления в comboBox на вкладке «Фильтр».
3. DateEdit: класс переопределяет родительский класс QDateEdit, для назначения кастомных стилей;
   1. Date\_edit\_style – переменная, хранящая в себе стили CSS для виджета QDateEdit.
4. ComboPickFlavor: класс переопределяет родительский класс QComboBox, для назначения кастомных стилей;
   1. Combo\_style – переменная, хранящая в себе стили CSS для виджета QComboBox.
5. Singleton: класс создает паттерн singleton, не позволяя создавать более 1 экземпляра приложения;
   1. Getinstance – функция-обертка, реализующая паттерн singleton.
6. UiMainWindow: класс инициализирует виджеты и GUI приложения, производит их первоначальную настройку.
   1. Setup\_ui – функция, создающая необходимые виджеты и стили CSS к ним.
7. Database: класс инициализирует базу данных, если она была не создана, наполняет ее шаблонными данными, содержит набор функцию для взаимодействия с БД.
   1. Db – путь к файлу БД;
   2. Connection – представляет соединение с БД через обращение к переменной «Db»;
   3. Tbl\_data – составляет и выполняет запрос на получение данных из таблицы, получаемой в аргументах;
   4. Save\_data – составляет и выполняет запрос на сохранение данных таблицы, получаемой в аргументах;
   5. Last\_id – возвращает номер последнего ID из таблицы, получаемой в аргументах;
   6. List\_flowers – составляет и выполняет запрос на взятие суммарного количество цветов по взятым параметрам;
   7. All\_money – получение итоговой прибыли;
   8. Get\_flowers – полученгие всего списка цветов.

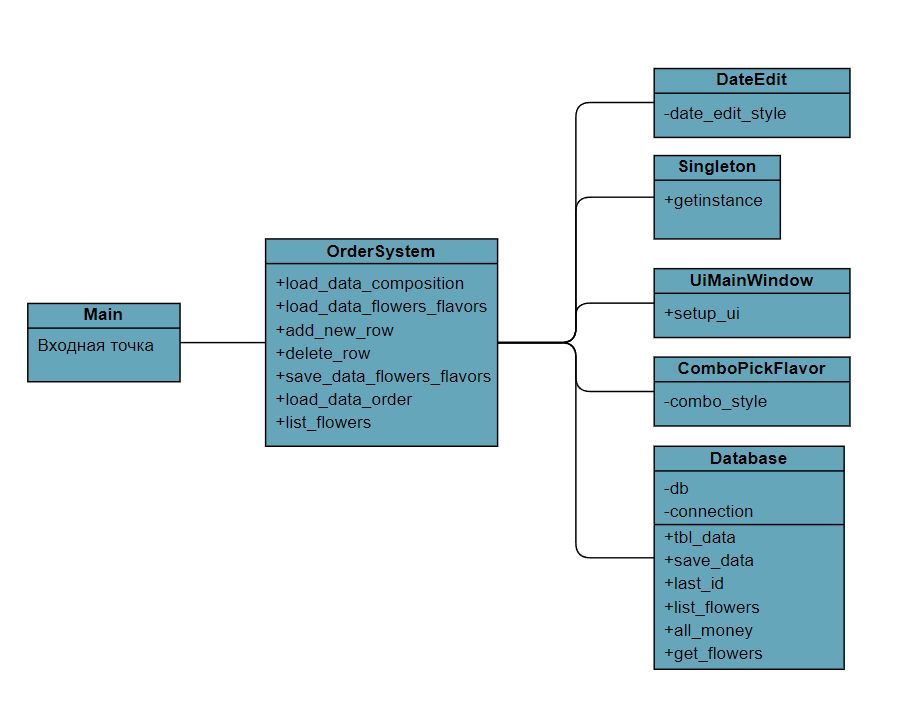


Рисунок 5. Диаграмма классов

## **Разработка ER-диаграммы баз данных**

В приложении для хранения всей нужной информации была создана база данных. На рисунке 6 представлена ER-модель Базы данных. Так же ниже описана структура таблиц и назначение полей.

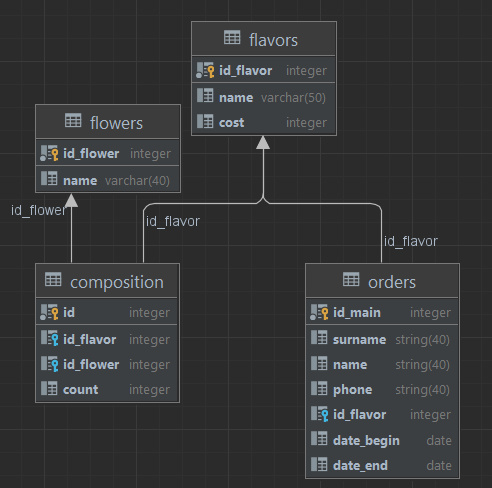


Рисунок 6. ER-модель

База данных состоит из 4 таблиц:

1. Flavors – таблица для хранения наименований букетов и их цены;
   1. Id\_flavor – первичный ключ;
   2. Name – наименование цветочного букета;
   3. Cost – цена букета.
2. Flowers – таблица для хранения наименований цветов;
   1. Id\_flower – первичный ключ;
   2. Name – наименование цветка.
3. Composition – таблица цветочных композиций;
   1. Id – первичный ключ;
   2. Id\_flavor – внешний ключ на таблицу «Flavors»;
   3. Id\_flower – внешний ключ на таблицу «Flowers»;
   4. Count – количество цветка.
4. Orders – таблица заказов клиентов.
   1. Id\_main – первичный ключ;
   2. Surname – фамилия клиента;
   3. Name – имя клиента;
   4. Phone – номер телефона книента;
   5. Id\_flavor – внешний ключ на таблицу «Flavors»;
   6. Date\_begin – дата оформления заказа;
   7. Date\_end – дата выдачи заказа.

## **Описание модулей проекта**

Помимо стартового модуля, в проект входят и другие модули с фрагментами, и интерфейсные модули.

Также в стартовый модуль входит код, который запускает саму программу при ее запуске. Полный код всех модулей находится в «Приложение 1».

async\_init() – данный метод запустится при старте программы, как входная точка в асинхронность;

load\_data() – этот метод вызывается для загрузки данных из БД в определенную таблицу;

add\_new\_row() – данный метод вызывается для добавления новой строки в определенную таблицу;

delete\_row() – данный метод вызывается для удаление строки в определенной таблице;

save\_data() – данный метод берёт сохраняет данные редактируемой таблицы.

## **Тестирование проекта**

В этом разделе будут продемонстрированы результаты тестирования программы.

**Тест. Запуск приложения**

Действия: запустить приложение (Рисунок 7).

Ожидаемый результат: отображение окна «Заказы».

Результат теста:



Рисунок 7. Главное меню «Заказы»

В этом разделе показано умение применять средства отладки. Так как я работаю в PyChram, отладчик уже встроен в программу.

Для проверки была искусственно создана ошибка в коде. Вот что показал отладчик (Рисунок 8,9).

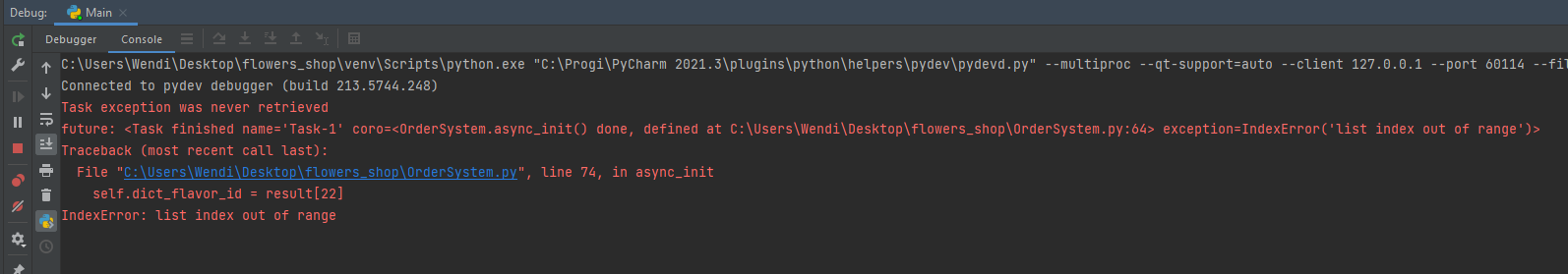


Рисунок 8. Ошибка в коде

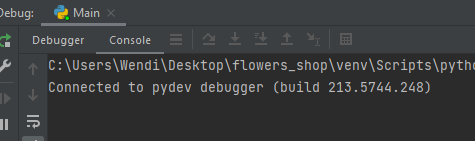


Рисунок 9. Исправление ошибки

**Тест. Срабатывание кнопки «Добавить»**

Действия: добавление новой строки в таблице, посредством кнопки «Добавить».

Ожидаемый результат: отображение новой пустой строки в таблице.

Результат теста:

Программа была запущена с помощью встроенного отладчика PyCharm – Debugger, чтобы отследить возможные ошибки. На вкладке «Заказы» была нажата кнопка «Добавить» (Рисунок 10).



Рисунок 10. Тест добавления новой строки

После этого была добавлена новая строка в таблицу. Следовательно, функция add\_new\_row\_orders сработала корректно (Рисунок 11).



Рисунок 11. Тест с добавленной строкой

В консоли отладчика не было выявлено никаких ошибок (Рисунок 12).

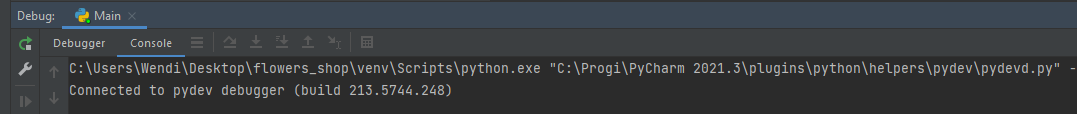


Рисунок 12. Добавление строки - консоль

**Тест. Отображение таблиц на вкладке «Композиции» из списка**

Действия: открыть список таблик на вкладке «Композиции», выбрать букет «Салют».

Ожидаемый результат: отображение выбранной таблицы с загруженными данными.

Результат теста:

Программа была запущена с помощью встроенного отладчика PyCharm – Debugger, чтобы отследить возможные ошибки. Далее, на вкладке «Композиции» был открыт список букетов (Рисунок 13).



Рисунок 13. Выбор таблицы на вкладке «Композиции»

После этого был выбран букет «Салют». Программа загрузила корректные данные в таблицу (Рисунок 14).



Рисунок 14. Загрузка таблицы букета «Салют»

В консоли отладчика не было выявлено никаких ошибок (Рисунок 15).

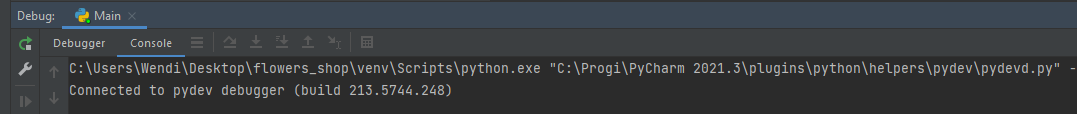


Рисунок 15. Добавление строки - консоль

## **Руководство оператора**

В данном программном документе приведено руководство оператора по применению и эксплуатации программы «Chamomile.exe».

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1)1), ГОСТ 19.103-77 [[2]](#footnote-2)2), ГОСТ 19.104-78\* [[3]](#footnote-3)3), ГОСТ 19.105-78\* [[4]](#footnote-4)4), ГОСТ 19.106-78\* [[5]](#footnote-5)5), ГОСТ 19.505-79\* [[6]](#footnote-6)6), ГОСТ 19.604-78\* [[7]](#footnote-7)7)).

Далее на страницах 20-26 приведено содержание разработанного документа «Руководство оператора» на спроектированное приложение.

1. **Назначение программы**
   1. **Функциональное назначение программы**

Специальное программное обеспечение «Сhamomile.exe» используется для управления цветочным магазином «Ромашка».

* 1. **Эксплуатационное назначение программы**

Специальное программное обеспечение «Сhamomile.exe» может эксплуатироваться в учебных заведениях.

* 1. **Состав функций**
     1. **Функция загрузки данных из БД.**

Эта функция позволяет загрузить данные с базы данных в выбранную таблицу.

* + 1. **Функция сохранения данных в БД.**

Эта функция позволяет пользователю сохранить данные в таблице после изменения данных в ней.

* + 1. **Функция добавления новой строки в таблицу.**

Эта функция позволяет добавить новую строку в выбранную таблицу.

* + 1. **Функция удаления строки в таблице.**

Эта функция позволяет удалить строку из выбранной таблице.

* + 1. **Функция фильтрации по выбранному цветку.**

Эта функция позволяет отфильтровать по выбранному цветку и увидеть в каких букетах он используется и в каком количестве.

1. **Условия выполнения программы**
   1. **Минимальный состав аппаратных средств**

* ОС: Windows 10 и выше;
* Процессор: 32-битные и 64-битные: ARM, MIPS, x86. Тактовая частота не ниже 1,8 ГГц;
* ОЗУ: 2 ГБ;
* Место на жестком диске: 10 МБ;
* Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0.;
* Дисплей: 720 x 1600.
  1. **Минимальный состав программных средств**

Дополнительные программные средства не требуются.

* 1. **Требование к пользователю**

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

1. **Выполнение программы**
   1. **Загрузка и запуск программы**

Запустите программу «Сhamomile.exe», после чего откроется окно пролога (Рисунок 16).



Рисунок 16. Окно приложения при запуске

* 1. **Выполнение программы**
     1. **Выполнение функции добавления и сохранения нового заказа**

Выберите кнопку «Добавить», после этого заполните все поля, кроме «ID» и «Цена букета» (после сохранения данные автоматически заполняться в этих ячейках) (Рисунок 17).



Рисунок 17. Окно «Заказы»

После того как мы заполним данные и нажмём кнопку «Сохранить», при корректном заполнении всех полей, данные будут сохранены в БД, нижняя строка статуса проинформирует об успешном сохранении данных (Рисунок 18).

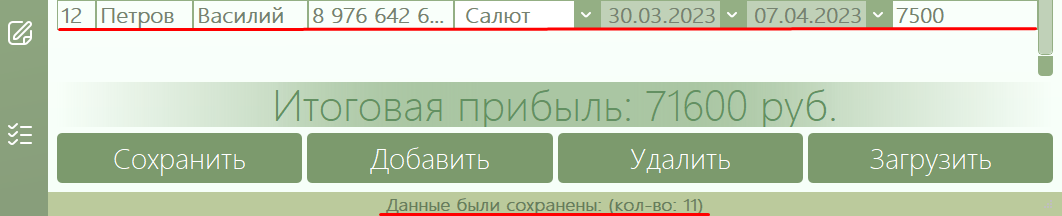


Рисунок 18. Сохранение данных

* + 1. **Выполнение функции удаления**

Перейдите на вкладку с нужной вам таблицей, после этого выберите строку для удаления (Рисунок 19).



Рисунок 19. Вкладка «Композиция»

После того, как вы выбрали нужную строку, нажмите на кнопку «Удалить». Строка будет удалена из визуальной таблицы и БД. Статусная строка в нижней части приложения проинформирует об успешном удалении (Рисунок 20).



Рисунок 20. Удаление строки

* + 1. **Выполнение функции загрузки данных**

Перейдите на вкладку с нужной вам таблицей, после этого измените любые ячейки (Рисунок 21, 22).

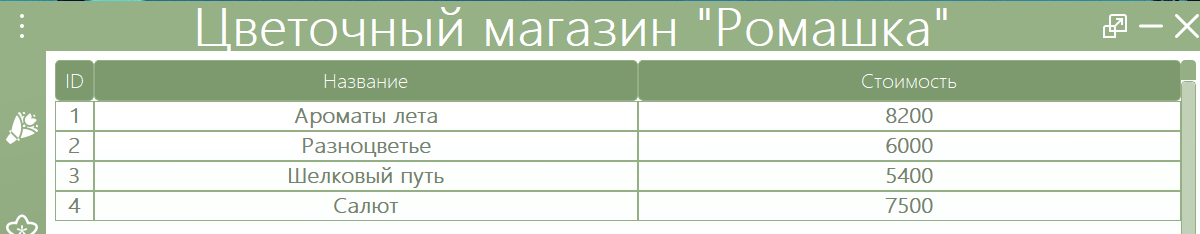


Рисунок 21. Таблица букетов до изменений

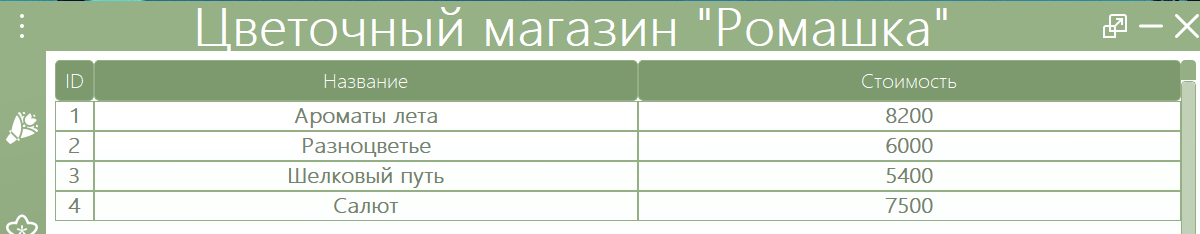


Рисунок 22. Таблица букетов после изменений

После изменения данных в таблице, без сохранения данных нажмите кнопку «Загрузить». Будут загружены данные, до несохраненных изменений, что видно на рисунке 23.

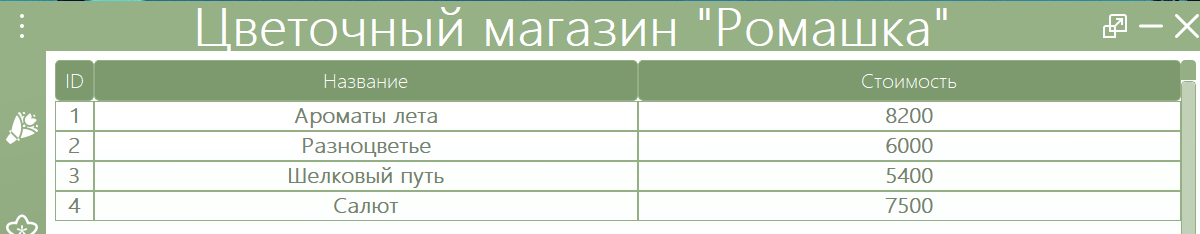


Рисунок 23. Загрузка старых данных без сохранения

* + 1. **Выполнение функции фильтрации**

Перейдите на вкладку «Фильтр». Далее выберите цветок и период, за который должна производиться фильтрация (Рисунок 24).

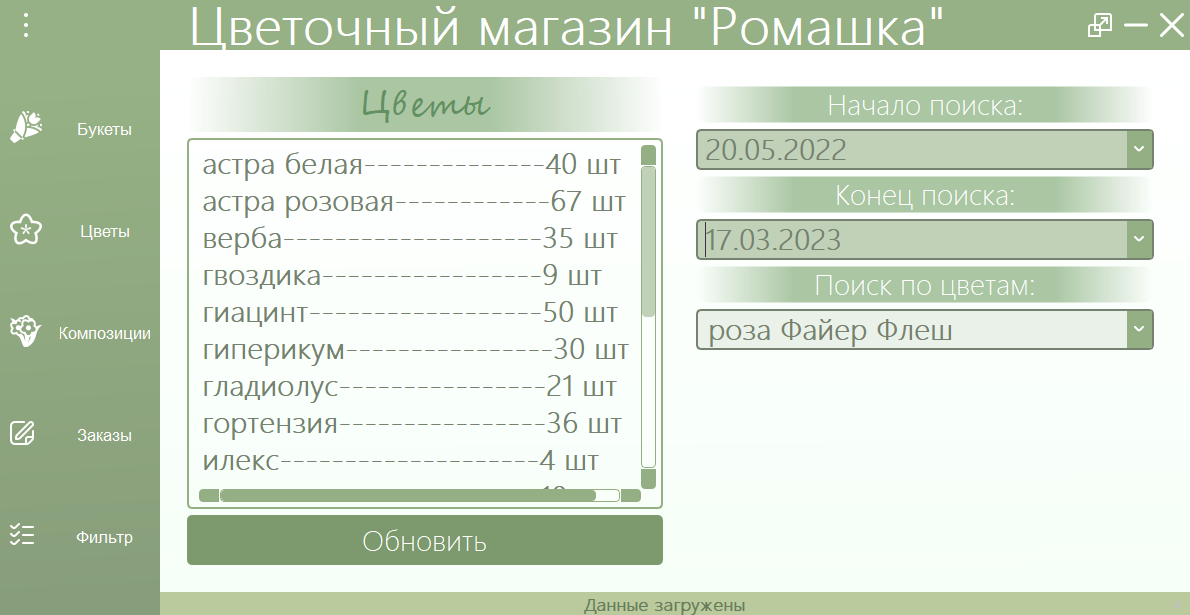


Рисунок 24. Вкладка «Фильтр»

После этого нажмите кнопку «Обновить». В списке будут показаны букеты, в которых используется выбранный цветок и количество цветков, необходимое для дальнейшей закупки, чтобы выполнить все действующие заказы из таблицы на вкладе «Заказы» (Рисунок 25).

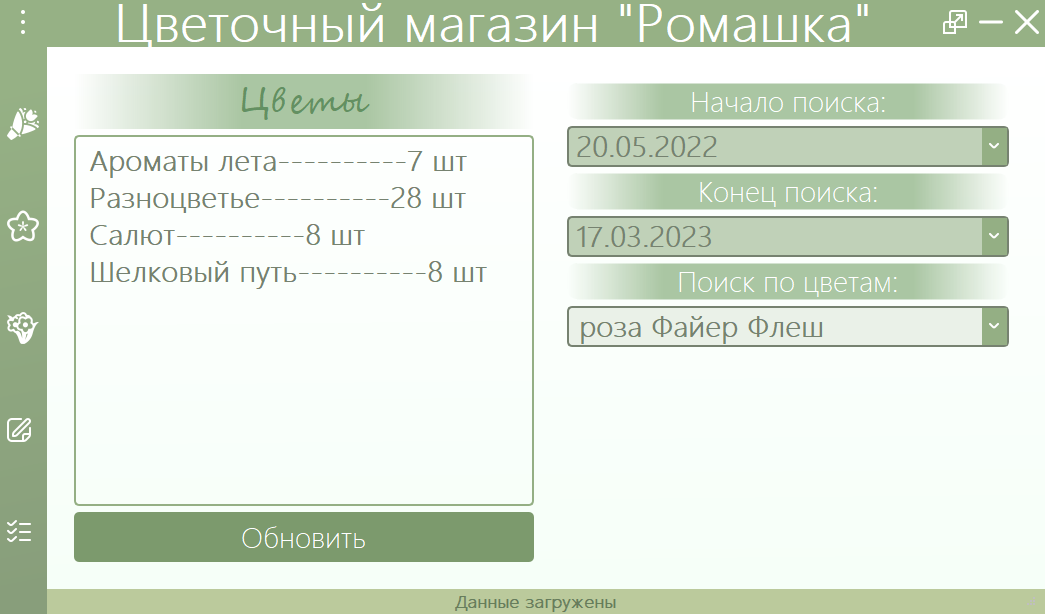


Рисунок 25. Отфильтрованные данные

* 1. **Завершение работы программы**

Чтобы завершить работу программы необходимо нажать на «крестик» в правой верхней части экрана.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта была написана программа «chamomile.exe» для упрощения и удобства учета цветочной продукции на складе и оформления заказов клиентов в цветочном магазине «Ромашка».

В ходе работы были проанализированы: предметная область, существующие разработки, получены навыки по созданию десктопных приложений и работе с БД.

Также планируется продолжение работы над данным проектом с целью расширения возможностей и удобства приложения для пользователей.

# **Список используемых источников**

1. Интернет-ресурс «Pythonist.ru» [Электронный ресурс] // <https://pythonist-ru.turbopages.org/pythonist.ru/s/rukovodstvo-po-pyqt5/> (дата обращения: 02.02.2023).
2. Лутц, М. Изучаем Python, 5-е издание / пер. с англ. СПБ.: ООО “Диалектика”, 2019. — 832 с.: ил. — Парал, тит. англ.
3. Маккинни У. Python и анализ данных: Первичная обработка данных с применением pandas, NumPy и Jupiter / пер. с англ. А. А. Слинкина. 3-е изд. – М.: МК Пресс, 2023. – 536 с.
4. Интернет-ресурс «WebforMyself» [Электронный ресурс] // <https://webformyself.com/rukovodstvo-po-pyqt-python-gui-designer/> (дата обращения: 15.01.2023);
5. Свейгарт Э. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих. 2-е изд / пер. с англ. - М.: ООО «И.Д. Вильямс», 2021. – 592 с.: ил. — Парал, тит. англ.
6. Доусон М. Программируем на Python / пер. с англ. - М.: ООО Издательство «Питер», 2022. – 416 с.: ил.
7. Бейдер Д. Чистый Python. Тонкости программирования для профи / пер. с англ. - М.: ООО Издательство «Питер», 2022. – 288 с.
8. Любанович Б. Простой Python. Современный стиль программирования / пер. с англ. - М.: ООО Издательство «Питер», 2021. – 592 с.

**Приложение 1.**

**Листинг 1. «Main.py»**

from PyQt5 import QtWidgets

from OrderSystem import OrderSystem

import asyncio

import sys

from asyncqt import QEventLoop

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

loop = QEventLoop(app)

asyncio.set\_event\_loop(loop)

ui = OrderSystem(app)

with loop:

sys.exit(loop.run\_forever())

**Листинг 2. «Singleton.py»**

from functools import wraps

class Singleton:

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

self.instances = {}

def \_\_call\_\_(self, ui\_class):

@wraps(ui\_class)

def getinstance(\*args, \*\*kwargs):

if ui\_class not in self.instances:

self.instances[ui\_class] = ui\_class(\*args, \*\*kwargs)

return self.instances[ui\_class]

return getinstance

**Листинг 3. «OrderSystem.py»**

from PyQt5 import QtWidgets

from PyQt5 import QtCore

from PyQt5.QtCore import QDate

from PyQt5.QtCore import Qt, QPropertyAnimation

from PyQt5.QtWidgets import QMainWindow

import asyncio

import SqlData

from GuiApp import \*

from asyncqt import asyncSlot

from Singleton import Singleton

@Singleton()

class OrderSystem(QMainWindow):

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

self.app = app

QMainWindow.\_\_init\_\_(self)

self.ui = UiMainWindow(self)

self.ui.setup\_ui()

self.tables = {1: ["flowers", self.ui.tbl\_flowers], 2: ["flavors", self.ui.tbl\_flavors],

3: ["composition", self.ui.tbl\_composition]}

self.db = SqlData.ex\_db

self.window\_size = 0

self.all\_money = ""

self.dict\_flavor\_id = {}

self.dict\_flower\_id = {}

self.widgets\_mas = []

self.composition\_widgets = []

self.flowers = []

self.min\_date, self.max\_date = ("", "")

self.one\_row\_flag = True

self.last\_row = 0

self.animation = None

self.click\_position = None

self.setWindowFlags(Qt.WindowType.FramelessWindowHint)

QtWidgets.QSizeGrip(self.ui.size\_grip)

self.ui.main\_header.mouseMoveEvent = self.move\_window

self.ui.btn\_toggle.clicked.connect(lambda: self.slide\_left\_menu())

self.ui.stacked\_widget.setCurrentWidget(self.ui.orders\_page)

self.settings\_ui\_btns()

self.async\_init()

self.show()

def move\_window(self, e):

if not self.isMaximized():

if e.buttons() == Qt.LeftButton:

self.move(self.pos() + e.globalPos() - self.click\_position)

self.click\_position = e.globalPos()

e.accept()

@asyncSlot()

async def async\_init(self):

"""

Точка входа в асинхронность

"""

result = await asyncio.gather(self.db.min\_max\_dates(), self.db.get\_flowers(),

self.db.flavor\_flowers\_id(["id\_flavor", "flavors"]),

self.db.flavor\_flowers\_id(["id\_flower", "flowers"]))

self.min\_date, self.max\_date = result[0][0], result[0][1]

self.flowers = result[1]

self.dict\_flavor\_id = result[2]

self.dict\_flower\_id = result[3]

self.ui.date\_begin\_flowers.setDate(QDate.fromString(self.min\_date, "yyyy-MM-dd"))

self.ui.date\_end\_flowers.setDate(QDate.fromString(self.max\_date, "yyyy-MM-dd"))

self.ui.combo\_flowers.addItems(self.flowers)

self.ui.combo\_tbls.addItems(self.dict\_flavor\_id)

await self.load\_data\_order()

await self.load\_data\_flowers\_flavors(self.tables[1])

await self.load\_data\_flowers\_flavors(self.tables[2])

await self.load\_data\_composition()

await self.clicked\_btn()

await self.list\_flowers()

async def clicked\_btn(self):

self.ui.btn\_load.clicked.connect(self.load\_data\_order)

self.ui.btn\_add.clicked.connect(self.add\_new\_row\_orders)

self.ui.btn\_del.clicked.connect(self.delete\_row)

self.ui.btn\_save.clicked.connect(self.save\_data\_order)

self.ui.btn\_products.clicked.connect(self.list\_flowers)

self.ui.btn\_load\_flower.clicked.connect(lambda: self.load\_data\_flowers\_flavors(self.tables[1]))

self.ui.btn\_add\_flower.clicked.connect(lambda: self.add\_new\_row(self.tables[1]))

self.ui.btn\_del\_flower.clicked.connect(lambda: self.delete\_row\_flowers\_flavors(self.tables[1]))

self.ui.btn\_save\_flower.clicked.connect(lambda: self.save\_data\_flowers\_flavors(self.tables[1]))

self.ui.btn\_load\_flavor.clicked.connect(lambda: self.load\_data\_flowers\_flavors(self.tables[2]))

self.ui.btn\_add\_flavor.clicked.connect(lambda: self.add\_new\_row(self.tables[2]))

self.ui.btn\_del\_flavor.clicked.connect(lambda: self.delete\_row\_flowers\_flavors(self.tables[2]))

self.ui.btn\_save\_flavor.clicked.connect(lambda: self.save\_data\_flowers\_flavors(self.tables[2]))

self.ui.btn\_load\_composition.clicked.connect(lambda: self.load\_data\_composition())

self.ui.btn\_add\_composition.clicked.connect(lambda: self.add\_new\_row(self.tables[3]))

self.ui.btn\_del\_composition.clicked.connect(lambda: self.delete\_row\_flowers\_flavors(self.tables[3]))

self.ui.btn\_save\_composition.clicked.connect(lambda: self.save\_data\_flowers\_flavors(self.tables[3]))

def mousePressEvent(self, event):

self.click\_position = event.globalPos()

def slide\_left\_menu(self):

width = self.ui.left\_side\_menu.width()

if width == 50:

new\_width = 160

else:

new\_width = 50

self.animation = QPropertyAnimation(self.ui.left\_side\_menu, b"minimumWidth")

self.animation.setDuration(250)

self.animation.setStartValue(width)

self.animation.setEndValue(new\_width)

self.animation.setEasingCurve(QtCore.QEasingCurve.InOutQuart)

self.animation.start()

def settings\_ui\_btns(self):

self.ui.btn\_min.clicked.connect(lambda: self.showMinimized())

self.ui.btn\_restore.clicked.connect(lambda: self.restore\_maximize\_win())

self.ui.btn\_close.clicked.connect(lambda: self.close())

self.ui.btn\_flowers\_menu.clicked.connect(

lambda: self.ui.stacked\_widget.setCurrentWidget(self.ui.flowers\_page)) # !!!!!!! flowers\_page

self.ui.btn\_flavors\_menu.clicked.connect(

lambda: self.ui.stacked\_widget.setCurrentWidget(self.ui.flavor\_page)) # !!!!!!! flavor\_page

self.ui.btn\_composition\_menu.clicked.connect(

lambda: self.ui.stacked\_widget.setCurrentWidget(self.ui.composition\_page)) # !!!!!!! composition\_page

self.ui.btn\_orders\_menu.clicked.connect(

lambda: self.ui.stacked\_widget.setCurrentWidget(self.ui.orders\_page))

self.ui.btn\_products\_menu.clicked.connect(

lambda: self.ui.stacked\_widget.setCurrentWidget(self.ui.products\_page))

self.ui.combo\_tbls.activated.connect(self.on\_combobox\_changed)

def restore\_maximize\_win(self):

win\_status = self.window\_size

if win\_status == 0:

self.window\_size = 1

self.showMaximized()

else:

self.window\_size = 0

self.showNormal()

def on\_combobox\_changed(self):

self.load\_data\_composition()

@asyncSlot()

async def load\_data\_composition(self):

self.dict\_flower\_id = await self.db.flavor\_flowers\_id(["id\_flower", "flowers"])

cur\_tbl = self.ui.tbl\_composition

combo\_item = self.ui.combo\_tbls.currentText()

data\_composition = await self.db.composition\_data(self.dict\_flavor\_id[combo\_item])

cur\_tbl.setRowCount(0)

self.composition\_widgets.clear()

for row\_number, row\_data in enumerate(data\_composition):

cur\_tbl.insertRow(row\_number)

for col\_number, col\_data in enumerate(row\_data):

if col\_number == 1:

self.composition\_widgets.append(ComboPickFlavor(self, self.dict\_flower\_id))

cur\_tbl.setCellWidget(row\_number, col\_number, self.composition\_widgets[row\_number])

self.composition\_widgets[row\_number].setCurrentText(col\_data)

item = QtWidgets.QTableWidgetItem(str(col\_data))

item.setTextAlignment(Qt.AlignHCenter)

cur\_tbl.setItem(row\_number, col\_number, item)

self.ui.lbl\_info\_tbl.setText("Данные загружены")

self.one\_row\_flag = True

@asyncSlot()

async def load\_data\_flowers\_flavors(self, tbl):

cur\_tbl = tbl[1]

data\_flowers = await self.db.flowers\_flavors\_data(tbl[0])

cur\_tbl.setRowCount(0)

for row\_number, row\_data in enumerate(data\_flowers):

cur\_tbl.insertRow(row\_number)

for col\_number, col\_data in enumerate(row\_data):

item = QtWidgets.QTableWidgetItem(str(col\_data))

item.setTextAlignment(Qt.AlignHCenter)

cur\_tbl.setItem(row\_number, col\_number, item)

self.ui.lbl\_info\_tbl.setText("Данные загружены")

self.one\_row\_flag = True

@asyncSlot()

async def add\_new\_row(self, tbl):

cur\_tbl = tbl[1]

if self.one\_row\_flag:

row\_position = cur\_tbl.rowCount()

self.last\_row = (await self.db.last\_id(tbl[0]))[0]

res = str((await self.db.last\_id(tbl[0]))[0] + 1)

cur\_tbl.insertRow(row\_position)

cur\_tbl.setItem(row\_position, 0, QtWidgets.QTableWidgetItem(res))

if tbl[0] == "composition":

self.composition\_widgets.append(ComboPickFlavor(self, self.dict\_flower\_id))

cur\_tbl.setCellWidget(row\_position, 1, self.composition\_widgets[-1])

self.one\_row\_flag = False

else:

self.ui.lbl\_info\_tbl.setText('Сохраните таблицу')

@asyncSlot()

async def delete\_row\_flowers\_flavors(self, tbl):

if tbl[1].rowCount() > 0 and tbl[1].currentRow() != -1:

current\_row = tbl[1].currentRow()

if tbl[0] == "composition":

del self.composition\_widgets[current\_row]

tbl[1].removeRow(current\_row)

self.ui.lbl\_info\_tbl.setText(f'Строка {current\_row + 1} была успешно удалена')

@asyncSlot()

async def save\_data\_flowers\_flavors(self, tbl):

try:

data = []

for row in range(tbl[1].rowCount()):

data.append([])

if not tbl[1].item(row, 0).text().isdigit():

self.ui.lbl\_info\_tbl.setText('ID должен быть числом')

raise Exception

data[row].append(tbl[1].item(row, 0).text())

if tbl[0] == "flowers" or tbl[0] == "flavors":

data[row].append(tbl[1].item(row, 1).text())

if tbl[0] == "flavors":

data[row].append(tbl[1].item(row, 2).text())

elif tbl[0] == "composition":

data[row].append(self.dict\_flavor\_id[self.ui.combo\_tbls.currentText()])

data[row].append(self.dict\_flower\_id[self.composition\_widgets[row].currentText()])

data[row].append(tbl[1].item(row, 2).text())

save\_rows\_count = await self.db.save\_data(data, tbl[0])

if tbl[0] == "flowers":

await self.load\_data\_composition()

self.flowers = await self.db.get\_flowers()

self.ui.combo\_flowers.clear()

self.ui.combo\_flowers.addItems(self.flowers)

elif tbl[0] == "flavors":

await self.load\_data\_order()

self.dict\_flavor\_id = await self.db.flavor\_flowers\_id(["id\_flavor", "flavors"])

self.ui.combo\_tbls.clear()

self.ui.combo\_tbls.addItems(self.dict\_flavor\_id)

await self.load\_data\_composition()

self.ui.lbl\_info\_tbl.setText(f"Данные были сохранены: (кол-во: {save\_rows\_count})")

self.one\_row\_flag = True

except AttributeError:

self.ui.lbl\_info\_tbl.setText('Заполните все поля корректно')

@asyncSlot()

async def load\_data\_order(self):

self.dict\_flavor\_id = await self.db.flavor\_flowers\_id(["id\_flavor", "flavors"])

self.widgets\_mas.clear()

data\_orders = await self.db.orders\_data()

all\_money = await self.db.all\_money()

all\_money = f"Итоговая прибыль: {all\_money[0]} руб."

self.ui.lbl\_cost.setText(all\_money)

self.ui.tbl.setRowCount(0)

for row\_number, row\_data in enumerate(data\_orders):

self.ui.tbl.insertRow(row\_number)

self.widgets\_mas.append([ComboPickFlavor(self, self.dict\_flavor\_id), DateEdit(self),

DateEdit(self)])

for col\_number, col\_data in enumerate(row\_data):

if col\_number not in [4, 5, 6]:

self.ui.tbl.setItem(row\_number, col\_number, QtWidgets.QTableWidgetItem(str(col\_data)))

elif col\_number == 4:

self.ui.tbl.setCellWidget(row\_number, col\_number, self.widgets\_mas[row\_number][0])

self.widgets\_mas[row\_number][0].setCurrentText(col\_data)

elif col\_number == 5:

date = QDate.fromString(col\_data, "yyyy-MM-dd")

self.ui.tbl.setCellWidget(row\_number, col\_number, self.widgets\_mas[row\_number][1])

self.widgets\_mas[row\_number][1].setDate(QDate(date))

elif col\_number == 6:

date = QDate.fromString(col\_data, "yyyy-MM-dd")

self.ui.tbl.setCellWidget(row\_number, col\_number, self.widgets\_mas[row\_number][2])

self.widgets\_mas[row\_number][2].setDate(QDate(date))

self.ui.lbl\_info\_tbl.setText("Данные загружены")

self.one\_row\_flag = True

@asyncSlot()

async def list\_flowers(self):

self.min\_date = self.ui.date\_begin\_flowers.date().toPyDate().strftime('%Y-%m-%d')

self.max\_date = self.ui.date\_end\_flowers.date().toPyDate().strftime('%Y-%m-%d')

self.ui.list\_flowers.clear()

result\_list = await self.db.list\_flowers(self.ui.combo\_flowers.currentText(), self.min\_date, self.max\_date)

self.ui.list\_flowers.addItems(result\_list)

@asyncSlot()

async def save\_data\_order(self): try:

data = []

for row in range(self.ui.tbl.rowCount()):

data.append([])

two\_date = [self.widgets\_mas[row][1].date().toPyDate().strftime('%Y-%m-%d'),

self.widgets\_mas[row][2].date().toPyDate().strftime('%Y-%m-%d')]

if not self.ui.tbl.item(row, 0).text().isdigit():

self.ui.lbl\_info\_tbl.setText('ID должен быть числом')

raise Exception

if not two\_date[0] <= two\_date[1]:

self.ui.lbl\_info\_tbl.setText('Первая дата не может быть больше второй')

raise Exception

data[row].append(self.ui.tbl.item(row, 0).text())

data[row].append(self.ui.tbl.item(row, 1).text())

data[row].append(self.ui.tbl.item(row, 2).text())

data[row].append(self.ui.tbl.item(row, 3).text())

data[row].append(self.dict\_flavor\_id[self.widgets\_mas[row][0].currentText()])

data[row].append(two\_date[0])

data[row].append(two\_date[1])

save\_rows\_count = await self.db.save\_data\_order(data)

self.ui.lbl\_info\_tbl.setText(f"Данные были сохранены: (кол-во: {save\_rows\_count})")

self.one\_row\_flag = True

except AttributeError:

self.ui.lbl\_info\_tbl.setText('Заполните все поля корректно')

@asyncSlot()

async def add\_new\_row\_orders(self):

if self.one\_row\_flag:

row\_position = self.ui.tbl.rowCount()

res = str((await self.db.last\_id\_orders())[0] + 1)

self.ui.tbl.insertRow(row\_position)

self.widgets\_mas.append([ComboPickFlavor(self, self.dict\_flavor\_id), DateEdit(self), DateEdit(self)])

self.ui.tbl.setItem(row\_position, 0, QtWidgets.QTableWidgetItem(res))

self.ui.tbl.setCellWidget(row\_position, 4, self.widgets\_mas[row\_position][0])

self.ui.tbl.setCellWidget(row\_position, 5, self.widgets\_mas[row\_position][1])

self.ui.tbl.setCellWidget(row\_position, 6, self.widgets\_mas[row\_position][2])

self.ui.tbl.setItem(row\_position, 7, QtWidgets.QTableWidgetItem("-"))

self.one\_row\_flag = False

else:

self.ui.lbl\_info\_tbl.setText('Сохраните таблицу')

def delete\_row(self):

if self.ui.tbl.rowCount() > 0 and self.ui.tbl.currentRow() != -1:

current\_row = self.ui.tbl.currentRow()

self.ui.tbl.removeRow(current\_row)

del self.widgets\_mas[current\_row]

self.ui.lbl\_info\_tbl.setText(f'Строка {current\_row + 1} была успешно удалена')

**Листинг 4. «Database.py»**

import sqlite3 as sql

import aiosqlite

import os

class Database:

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

self.max\_date = ""

self.min\_date = ""

self.db = "./SqlData/Db.db"

self.connection = sql.connect(self.db)

self.cur = self.connection.cursor()

sql\_files\_list = sorted(os.listdir("./SqlData/sql\_files/create\_tables/"))

for i in sql\_files\_list:

with open("./SqlData/sql\_files/create\_tables/" + i, 'r') as sql\_file:

sql\_script = sql\_file.read()

try:

self.cur.executescript(sql\_script)

self.connection.commit()

except sql.Error as e:

self.error = f"Ошибка выполнения запроса {e} Имя файла: {sql\_file.name}" # убрать

if self.connection:

self.connection.close()

async def flowers\_flavors\_data(self, tbl\_name):

"""Составляет и выполняет запрос на взятие данных с таблицы flowers и flavors"""

query = "SELECT \* FROM {}".format(tbl\_name)

async with aiosqlite.connect(self.db) as db:

async with db.execute(query) as cursor:

return await cursor.fetchall()

async def composition\_data(self, id\_flavor):

"""Составляет и выполняет запрос на взятие данных с таблицы composition"""

query = "SELECT id, flowers.name, count FROM composition JOIN flowers USING(id\_flower) WHERE id\_flavor={} ".format(id\_flavor)

async with aiosqlite.connect(self.db) as db:

async with db.execute(query) as cursor:

return await cursor.fetchall()

async def save\_data(self, data, name\_tbl):

"""Составляет и выполняет запрос на сохранение данных в таблицу"""

if name\_tbl == "flowers":

query = "INSERT INTO {} VALUES (?, ?)".format(name\_tbl)

elif name\_tbl == "composition":

query = "INSERT INTO {} VALUES (?, ?, ?, ?)".format(name\_tbl)

else:

query = "INSERT INTO {} VALUES (?, ?, ?)".format(name\_tbl)

async with aiosqlite.connect(self.db) as db:

async with db.cursor() as cursor:

if name\_tbl == "composition":

del\_query = "DELETE FROM {} WHERE id\_flavor = {}".format(name\_tbl, data[0][1])

else:

del\_query = "DELETE FROM {}".format(name\_tbl)

if name\_tbl == "flavors":

id\_flavors = [i[0] for i in data]

id\_flavors = ", ".join(id\_flavors)

del\_query1 = "DELETE FROM composition WHERE id\_flavor NOT IN ({})".format(id\_flavors)

await cursor.execute(del\_query1) # удаление старых данных с таблицы sql

await cursor.execute(del\_query) # удаление старых данных с таблицы sql

await cursor.executemany(query, data) # вставка новых данных в таблицу sql

await db.commit()

return cursor.rowcount

async def last\_id(self, tbl\_name):

"""Составляет и выполняет запрос получение последнего id из таблицы"""

query = "SELECT \* FROM {} ORDER BY 1 DESC LIMIT 1".format(tbl\_name)

async with aiosqlite.connect(self.db) as db:

async with db.execute(query) as cursor:

return await cursor.fetchone()

async def orders\_data(self):

"""Составляет и выполняет запрос на взятие данных с таблицы orders"""

query = "SELECT id\_main, surname, orders.name, phone, flavors.name, date\_begin, date\_end, cost " \

"FROM orders INNER JOIN flavors USING (id\_flavor)"

async with aiosqlite.connect(self.db) as db:

async with db.execute(query) as cursor:

return await cursor.fetchall()

async def save\_data\_order(self, data):

"""Составляет и выполняет запрос на сохранение данных в таблицу orders"""

query = """INSERT INTO orders (id\_main, surname, name, phone, id\_flavor, date\_begin, date\_end)

VALUES (?, ?, ?, ?, ?, ?, ?)"""

async with aiosqlite.connect(self.db) as db:

async with db.cursor() as cursor:

await cursor.execute("DELETE FROM orders") # удаление старых данных с таблицы sql

await cursor.executemany(query, data) # вставка новых данных в таблицу sql

await db.commit()

return cursor.rowcount

async def list\_flowers(self, flower, min\_date, max\_date):

"""Составляет и выполняет запрос на взятие суммарного количество цветов по взятым параметрам"""

list\_flowers = []

data = [min\_date, max\_date]

if flower == "Все цветы":

with open("./SqlData/sql\_files/queries/get\_list\_flowers.sql", 'r') as sql\_file:

query = sql\_file.read()

else:

with open("./SqlData/sql\_files/queries/get\_list\_flowers\_filter.sql", 'r') as sql\_file:

query = sql\_file.read()

data.append(flower)

length\_data = len(data)

async with aiosqlite.connect(self.db) as db:

async with db.execute(query, data) as cursor:

async for i in cursor:

if length\_data == 3:

result\_str = f"{i[0]}{'-' \* (25 - len(i[1]))}{i[2]} шт"

else:

result\_str = f"{i[0]}{'-' \* (25 - len(i[0]))}{i[1]} шт"

list\_flowers.append(result\_str)

return list\_flowers

async def all\_money(self):

"""Составляет и выполняет запрос на получение суммарной прибыли"""

with open("./SqlData/sql\_files/queries/get\_all\_money.sql", 'r') as sql\_file:

async with aiosqlite.connect(self.db) as db:

async with db.execute(sql\_file.read()) as cursor:

return await cursor.fetchone()

async def min\_max\_dates(self):

"""Составляет и выполняет запрос на получение ранней и поздней дат"""

async with aiosqlite.connect(self.db) as db:

async with db.cursor() as cur:

await cur.execute("SELECT MIN(date\_begin), MAX(date\_begin) FROM orders")

return await cur.fetchone()

async def get\_flowers(self):

"""Составляет и выполняет запрос на получение всего спискса цветов"""

async with aiosqlite.connect(self.db) as db:

async with db.cursor() as cur:

await cur.execute("SELECT name FROM flowers")

return [" ".join(x) for x in await cur.fetchall()]

async def flavor\_flowers\_id(self, query\_data):

"""Выполняет запрос на получение названий букетов либо цветов и формирует из данных словарь"""

query = "SELECT {}, name FROM {}".format(query\_data[0], query\_data[1])

dict\_id = {}

async with aiosqlite.connect(self.db) as db:

async with db.execute(query) as cursor:

async for i in cursor:

dict\_id[i[1]] = i[0]

return dict\_id

async def last\_id\_orders(self):

"""Составляет и выполняет запрос получение последнего id из таблицы orders"""

async with aiosqlite.connect(self.db) as db:

async with db.execute("SELECT id\_main FROM orders ORDER BY id\_main DESC LIMIT 1") as cursor:

return await cursor.fetchone()

**Листинг 5. «ComboPickFlavor.py»**

from PyQt5.QtWidgets import QComboBox

class ComboPickFlavor(QComboBox):

def \_\_init\_\_(self, parent, id\_flavors):

super().\_\_init\_\_(parent)

self.addItems(list(id\_flavors.keys()))

self.setStyleSheet("""QComboBox {

font: 15pt "Microsoft YaHei UI Light";

font-weight: bold;

background-color: #fff;

border: none;

}""")

**Листинг 6. «DateEdit.py»**

from datetime import datetime

from PyQt5.QtCore import QDate

from PyQt5.QtWidgets import QDateEdit

class DateEdit(QDateEdit):

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

super().\_\_init\_\_(parent)

self.setCalendarPopup(True)

self.setDisplayFormat("dd.MM.yyyy")

self.setDate(QDate(datetime.now().date()))

self.setStyleSheet("""

font: 15pt "Microsoft YaHei UI Light";

font-weight: bold;

background-color: #C0D1B8;

border: none;

border-radius: 0px;

""")

**Листинг 7. «UiMainWindow.py»**

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtGui import QIcon

from PyQt5.QtCore import Qt

class UiMainWindow(object):

def \_\_init\_\_(self, main\_window):

self.main\_window = main\_window

self.styles = QtWidgets.QWidget(self.main\_window)

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

self.main\_body = QtWidgets.QFrame(self.styles)

self.main\_header = QtWidgets.QFrame(self.styles)

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.main\_header)

self.tittle\_bar\_container = QtWidgets.QFrame(self.main\_header)

self.lbl\_title = QtWidgets.QLabel(self.main\_header)

self.btns\_navigation = QtWidgets.QFrame(self.main\_header)

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout(self.tittle\_bar\_container)

self.left\_menu\_toggle = QtWidgets.QFrame(self.tittle\_bar\_container)

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout(self.left\_menu\_toggle)

self.btn\_toggle = QtWidgets.QPushButton(self.left\_menu\_toggle)

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout(self.btns\_navigation)

self.btn\_restore = QtWidgets.QPushButton(self.btns\_navigation)

self.btn\_min = QtWidgets.QPushButton(self.btns\_navigation)

self.btn\_close = QtWidgets.QPushButton(self.btns\_navigation)

self.horizontalLayout = QtWidgets.QHBoxLayout(self.main\_body)

self.left\_side\_menu = QtWidgets.QFrame(self.main\_body)

self.central\_body = QtWidgets.QFrame(self.main\_body)

self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.left\_side\_menu)

self.btn\_flowers\_menu = QtWidgets.QPushButton(self.left\_side\_menu)

self.btn\_flavors\_menu = QtWidgets.QPushButton(self.left\_side\_menu)

self.btn\_composition\_menu = QtWidgets.QPushButton(self.left\_side\_menu)

self.btn\_orders\_menu = QtWidgets.QPushButton(self.left\_side\_menu)

self.btn\_products\_menu = QtWidgets.QPushButton(self.left\_side\_menu)

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.central\_body)

self.middle\_body = QtWidgets.QFrame(self.central\_body)

self.main\_footer = QtWidgets.QFrame(self.central\_body)

self.horizontalLayout\_7 = QtWidgets.QHBoxLayout(self.main\_footer)

self.lbl\_info\_tbl = QtWidgets.QLabel(self.main\_footer)

self.size\_grip = QtWidgets.QFrame(self.main\_footer)

self.horizontalLayout\_6 = QtWidgets.QHBoxLayout(self.middle\_body)

self.stacked\_widget = QtWidgets.QStackedWidget(self.middle\_body)

self.flowers\_page = QtWidgets.QWidget()

self.verticalLayout\_11 = QtWidgets.QVBoxLayout(self.flowers\_page)

self.tbl\_flowers = QtWidgets.QTableWidget(self.flowers\_page)

self.btns\_table\_flowers = QtWidgets.QFrame(self.flowers\_page)

self.horizontalLayout\_11 = QtWidgets.QHBoxLayout(self.btns\_table\_flowers)

self.btn\_add\_flower = QtWidgets.QPushButton(self.btns\_table\_flowers)

self.btn\_del\_flower = QtWidgets.QPushButton(self.btns\_table\_flowers)

self.btn\_save\_flower = QtWidgets.QPushButton(self.btns\_table\_flowers)

self.btn\_load\_flower = QtWidgets.QPushButton(self.btns\_table\_flowers)

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!

# start --------page flavors------------

self.flavor\_page = QtWidgets.QWidget()

self.verticalLayout\_12 = QtWidgets.QVBoxLayout(self.flavor\_page)

self.tbl\_flavors = QtWidgets.QTableWidget(self.flavor\_page)

self.btns\_table\_flavors = QtWidgets.QFrame(self.flavor\_page)

self.horizontalLayout\_12 = QtWidgets.QHBoxLayout(self.btns\_table\_flavors)

self.btn\_add\_flavor = QtWidgets.QPushButton(self.btns\_table\_flavors)

self.btn\_del\_flavor = QtWidgets.QPushButton(self.btns\_table\_flavors)

self.btn\_save\_flavor = QtWidgets.QPushButton(self.btns\_table\_flavors)

self.btn\_load\_flavor = QtWidgets.QPushButton(self.btns\_table\_flavors)

# end --------page flavors------------

# start --------page composition------------

self.composition\_page = QtWidgets.QWidget()

self.verticalLayout\_13 = QtWidgets.QVBoxLayout(self.composition\_page)

self.combo\_frame = QtWidgets.QFrame(self.composition\_page)

self.horizontalLayout\_14 = QtWidgets.QVBoxLayout(self.combo\_frame)

self.combo\_lbl = QtWidgets.QLabel(self.combo\_frame)

self.combo\_lbl.setStyleSheet("""background-color: qlineargradient(spread:reflect, x1:0.5, y1:0.5, x2:1,

y2:0.5, stop:0.271186 rgba(170, 198, 163, 255),

stop:1 rgba(255, 255, 255, 0)); font-size: 35px; color: #627957; """)

self.combo\_lbl.setAlignment(Qt.AlignCenter)

self.combo\_tbls = QtWidgets.QComboBox(self.combo\_frame)

self.combo\_tbls.setStyleSheet("background-color: rgb(234, 241, 232);")

self.tbl\_composition = QtWidgets.QTableWidget(self.composition\_page)

self.btns\_table\_composition = QtWidgets.QFrame(self.composition\_page)

self.horizontalLayout\_13 = QtWidgets.QHBoxLayout(self.btns\_table\_composition)

self.btn\_add\_composition = QtWidgets.QPushButton(self.btns\_table\_composition)

self.btn\_del\_composition = QtWidgets.QPushButton(self.btns\_table\_composition)

self.btn\_save\_composition = QtWidgets.QPushButton(self.btns\_table\_composition)

self.btn\_load\_composition = QtWidgets.QPushButton(self.btns\_table\_composition)

# end --------page composition------------

self.orders\_page = QtWidgets.QWidget()

self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.orders\_page)

self.orders\_cost = QtWidgets.QFrame(self.orders\_page)

self.tbl = QtWidgets.QTableWidget(self.orders\_page)

self.btns\_table = QtWidgets.QFrame(self.orders\_page)

self.horizontalLayout\_9 = QtWidgets.QHBoxLayout(self.orders\_cost)

self.lbl\_cost = QtWidgets.QLabel(self.orders\_cost)

self.horizontalLayout\_8 = QtWidgets.QHBoxLayout(self.btns\_table)

self.btn\_add = QtWidgets.QPushButton(self.btns\_table)

self.btn\_del = QtWidgets.QPushButton(self.btns\_table)

self.btn\_save = QtWidgets.QPushButton(self.btns\_table)

self.btn\_load = QtWidgets.QPushButton(self.btns\_table)

self.products\_page = QtWidgets.QWidget()

self.verticalLayout\_6 = QtWidgets.QVBoxLayout(self.products\_page)

self.body\_products = QtWidgets.QFrame(self.products\_page)

self.horizontalLayout\_10 = QtWidgets.QHBoxLayout(self.body\_products)

self.left\_body = QtWidgets.QFrame(self.body\_products)

self.verticalLayout\_8 = QtWidgets.QVBoxLayout(self.left\_body)

self.lbl\_flowers = QtWidgets.QLabel(self.left\_body)

self.list\_flowers = QtWidgets.QListWidget(self.left\_body)

self.btn\_products = QtWidgets.QPushButton(self.left\_body)

self.right\_body = QtWidgets.QFrame(self.body\_products)

self.verticalLayout\_9 = QtWidgets.QVBoxLayout(self.right\_body)

self.sort\_flowers = QtWidgets.QFrame(self.right\_body)

self.empty\_frame = QtWidgets.QFrame(self.right\_body)

self.verticalLayout\_7 = QtWidgets.QVBoxLayout(self.sort\_flowers)

self.lbl\_date\_begin = QtWidgets.QLabel(self.sort\_flowers)

self.date\_begin\_flowers = QtWidgets.QDateEdit(self.sort\_flowers)

self.lbl\_date\_end = QtWidgets.QLabel(self.sort\_flowers)

self.date\_end\_flowers = QtWidgets.QDateEdit(self.sort\_flowers)

self.lbl\_combo\_flowers = QtWidgets.QLabel(self.sort\_flowers)

self.combo\_flowers = QtWidgets.QComboBox(self.sort\_flowers)

def setup\_ui(self):

"""

Инициализация и настройка виджетов и макетов

"""

self.main\_window.setObjectName("MainWindow")

self.main\_window.resize(1300, 600)

self.main\_window.setStyleSheet(open('./GuiApp/ui.stylesheet').read())

self.main\_window.setCentralWidget(self.styles)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setSpacing(0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.horizontalLayout.addWidget(self.left\_side\_menu)

self.horizontalLayout.addWidget(self.central\_body)

self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_2.setSpacing(0)

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

self.horizontalLayout\_2.addWidget(self.tittle\_bar\_container, 0, Qt.AlignLeft)

self.horizontalLayout\_2.addWidget(self.lbl\_title)

self.horizontalLayout\_2.addWidget(self.btns\_navigation)

self.horizontalLayout\_3.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_3.setSpacing(0)

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

self.horizontalLayout\_3.addWidget(self.btn\_restore)

self.horizontalLayout\_3.addWidget(self.btn\_min)

self.horizontalLayout\_3.addWidget(self.btn\_close)

self.horizontalLayout\_4.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_4.setSpacing(0)

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

self.horizontalLayout\_4.addWidget(self.btn\_toggle)

self.horizontalLayout\_5.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_5.setSpacing(0)

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

self.horizontalLayout\_5.addWidget(self.left\_menu\_toggle, 0, Qt.AlignLeft | Qt.AlignVCenter)

self.horizontalLayout\_6.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_6.setSpacing(0)

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

self.horizontalLayout\_6.addWidget(self.stacked\_widget)

self.horizontalLayout\_7.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_7.setSpacing(0)

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

self.horizontalLayout\_7.addWidget(self.lbl\_info\_tbl, 0, Qt.AlignVCenter)

self.horizontalLayout\_7.addWidget(self.size\_grip, 0, Qt.AlignRight | Qt.AlignBottom)

self.horizontalLayout\_8.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_8.setSpacing(5)

self.horizontalLayout\_8.addWidget(self.btn\_save)

self.horizontalLayout\_8.setObjectName("horizontalLayout\_8")

self.horizontalLayout\_8.addWidget(self.btn\_add)

self.horizontalLayout\_8.addWidget(self.btn\_del)

self.horizontalLayout\_8.addWidget(self.btn\_load)

self.horizontalLayout\_9.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_9.setSpacing(0)

self.horizontalLayout\_9.setObjectName("horizontalLayout\_9")

self.horizontalLayout\_9.addWidget(self.lbl\_cost, 0, Qt.AlignHCenter | Qt.AlignVCenter)

self.horizontalLayout\_10.setObjectName("horizontalLayout\_10")

self.horizontalLayout\_10.addWidget(self.left\_body)

self.horizontalLayout\_10.addWidget(self.right\_body)

# !!!!!!!!!!!!!!!

self.horizontalLayout\_11.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_11.setSpacing(5)

self.horizontalLayout\_11.addWidget(self.btn\_save\_flower)

self.horizontalLayout\_11.setObjectName("horizontalLayout\_11")

self.horizontalLayout\_11.addWidget(self.btn\_add\_flower)

self.horizontalLayout\_11.addWidget(self.btn\_del\_flower)

self.horizontalLayout\_11.addWidget(self.btn\_load\_flower)

# !!!!!!!!!!!!!!!!

# start --------page flavors------------

self.horizontalLayout\_12.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_12.setSpacing(5)

self.horizontalLayout\_12.addWidget(self.btn\_save\_flavor)

self.horizontalLayout\_12.setObjectName("horizontalLayout\_12")

self.horizontalLayout\_12.addWidget(self.btn\_add\_flavor)

self.horizontalLayout\_12.addWidget(self.btn\_del\_flavor)

self.horizontalLayout\_12.addWidget(self.btn\_load\_flavor)

# end --------page flavors------------

# start --------page composition------------

self.horizontalLayout\_13.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_13.setSpacing(5)

self.horizontalLayout\_13.addWidget(self.btn\_save\_composition)

self.horizontalLayout\_13.setObjectName("horizontalLayout\_13")

self.horizontalLayout\_13.addWidget(self.btn\_add\_composition)

self.horizontalLayout\_13.addWidget(self.btn\_del\_composition)

self.horizontalLayout\_13.addWidget(self.btn\_load\_composition)

self.horizontalLayout\_14.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_14.setSpacing(5)

self.horizontalLayout\_14.setObjectName("horizontalLayout\_14")

self.horizontalLayout\_14.addWidget(self.combo\_lbl)

self.horizontalLayout\_14.addWidget(self.combo\_tbls)

# end --------page composition------------

self.verticalLayout.setContentsMargins(0, 0, 0, 0)

self.verticalLayout.setSpacing(0)

self.verticalLayout.setObjectName("verticalLayout")

self.verticalLayout.addWidget(self.main\_header)

self.verticalLayout.addWidget(self.main\_body)

self.verticalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.verticalLayout\_2.setSpacing(0)

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

self.verticalLayout\_2.addWidget(self.middle\_body)

self.verticalLayout\_2.addWidget(self.main\_footer)

self.verticalLayout\_3.setContentsMargins(10, 0, 0, 0)

self.verticalLayout\_3.setSpacing(0)

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

self.verticalLayout\_3.addWidget(self.btn\_flavors\_menu) # !!!!!!! page flavors

self.verticalLayout\_3.addWidget(self.btn\_flowers\_menu) # !!!!!!! page flowers

self.verticalLayout\_3.addWidget(self.btn\_composition\_menu) # !!!!!!! page composition

self.verticalLayout\_3.addWidget(self.btn\_orders\_menu)

self.verticalLayout\_3.addWidget(self.btn\_products\_menu)

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

self.verticalLayout\_5.addWidget(self.tbl)

self.verticalLayout\_5.addWidget(self.orders\_cost)

self.verticalLayout\_5.addWidget(self.btns\_table, 0, Qt.AlignBottom)

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

self.verticalLayout\_6.addWidget(self.body\_products)

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

self.verticalLayout\_7.addWidget(self.lbl\_date\_begin)

self.verticalLayout\_7.addWidget(self.date\_begin\_flowers)

self.verticalLayout\_7.addWidget(self.lbl\_date\_end)

self.verticalLayout\_7.addWidget(self.date\_end\_flowers)

self.verticalLayout\_7.addWidget(self.lbl\_combo\_flowers)

self.verticalLayout\_7.addWidget(self.combo\_flowers)

self.verticalLayout\_8.setObjectName("verticalLayout\_8")

self.verticalLayout\_8.addWidget(self.lbl\_flowers)

self.verticalLayout\_8.addWidget(self.list\_flowers)

self.verticalLayout\_8.addWidget(self.btn\_products)

self.verticalLayout\_9.setObjectName("verticalLayout\_9")

self.verticalLayout\_9.addWidget(self.sort\_flowers)

self.verticalLayout\_9.addWidget(self.empty\_frame)

self.verticalLayout\_11.setObjectName("verticalLayout\_11")

self.verticalLayout\_11.addWidget(self.tbl\_flowers)

self.verticalLayout\_11.addWidget(self.btns\_table\_flowers, 0, Qt.AlignBottom)

# start --------page flavors------------

self.verticalLayout\_12.setObjectName("verticalLayout\_12")

self.verticalLayout\_12.addWidget(self.tbl\_flavors)

self.verticalLayout\_12.addWidget(self.btns\_table\_flavors, 0, Qt.AlignBottom)

# end --------page flavors------------

# start --------page composition------------

self.verticalLayout\_13.setObjectName("verticalLayout\_13")

self.verticalLayout\_13.addWidget(self.combo\_lbl)

self.verticalLayout\_13.addWidget(self.combo\_tbls)

self.verticalLayout\_13.addWidget(self.tbl\_composition)

self.verticalLayout\_13.addWidget(self.btns\_table\_composition, 0, Qt.AlignBottom)

# end --------page composition------------

self.stacked\_widget.addWidget(self.orders\_page)

self.stacked\_widget.addWidget(self.flowers\_page) # !!!!!!! flowers\_page

self.stacked\_widget.addWidget(self.flavor\_page) # !!!!!!! flavor\_page

self.stacked\_widget.addWidget(self.composition\_page) # !!!!!!! composition

self.stacked\_widget.addWidget(self.products\_page)

self.main\_header.setMaximumSize(QtCore.QSize(16777215, 50))

self.main\_header.setFrameShape(QtWidgets.QFrame.NoFrame)

self.main\_header.setFrameShadow(QtWidgets.QFrame.Raised)

self.main\_header.setObjectName("main\_header")

self.tittle\_bar\_container.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.tittle\_bar\_container.setFrameShadow(QtWidgets.QFrame.Raised)

self.tittle\_bar\_container.setObjectName("tittle\_bar\_container")

self.left\_menu\_toggle.setMinimumSize(QtCore.QSize(50, 0))

self.left\_menu\_toggle.setMaximumSize(QtCore.QSize(50, 16777215))

self.left\_menu\_toggle.setFrameShape(QtWidgets.QFrame.NoFrame)

self.left\_menu\_toggle.setFrameShadow(QtWidgets.QFrame.Raised)

self.left\_menu\_toggle.setObjectName("left\_menu\_toggle")

self.btn\_toggle.setObjectName("btn\_toggle")

self.btn\_toggle.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap(":/icons/icons/menu-dots-vertical-free-icon-font.png"),

QIcon.Normal, QIcon.Off)

self.btn\_toggle.setIcon(icon)

self.btn\_toggle.setIconSize(QtCore.QSize(24, 24))

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

self.lbl\_title.setSizePolicy(size\_policy)

self.lbl\_title.setMinimumSize(QtCore.QSize(0, 0))

self.lbl\_title.setAlignment(Qt.AlignCenter)

self.lbl\_title.setObjectName("lbl\_title")

self.lbl\_title.setText('Цветочный магазин "Ромашка"')

self.btns\_navigation.setFrameShape(QtWidgets.QFrame.NoFrame)

self.btns\_navigation.setObjectName("btns\_navigation")

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap(":/icons/icons/resize-free-icon-font.png"), QIcon.Normal, QIcon.Off)

self.btn\_restore.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

self.btn\_restore.setIcon(icon1)

self.btn\_restore.setIconSize(QtCore.QSize(24, 24))

self.btn\_restore.setFlat(True)

self.btn\_restore.setObjectName("btn\_restore")

icon2 = QtGui.QIcon()

icon2.addPixmap(QtGui.QPixmap(":/icons/icons/minus-free-icon-font.png"), QIcon.Normal, QIcon.Off)

self.btn\_min.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

self.btn\_min.setIcon(icon2)

self.btn\_min.setIconSize(QtCore.QSize(24, 24))

self.btn\_min.setFlat(True)

self.btn\_min.setObjectName("btn\_min")

icon3 = QtGui.QIcon()

icon3.addPixmap(QtGui.QPixmap(":/icons/icons/cross-free-icon-font.png"), QIcon.Normal, QIcon.Off)

self.btn\_close.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

self.btn\_close.setIcon(icon3)

self.btn\_close.setIconSize(QtCore.QSize(24, 24))

self.btn\_close.setFlat(True)

self.btn\_close.setObjectName("btn\_close")

self.main\_body.setFrameShape(QtWidgets.QFrame.NoFrame)

self.main\_body.setFrameShadow(QtWidgets.QFrame.Raised)

self.main\_body.setObjectName("main\_body")

self.left\_side\_menu.setMinimumSize(QtCore.QSize(50, 200))

self.left\_side\_menu.setMaximumSize(QtCore.QSize(0, 16777215))

self.left\_side\_menu.setFrameShape(QtWidgets.QFrame.NoFrame)

self.left\_side\_menu.setFrameShadow(QtWidgets.QFrame.Raised)

self.left\_side\_menu.setObjectName("left\_side\_menu")

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!

self.btn\_flowers\_menu.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_flowers\_menu.setObjectName("btn\_flowers\_menu")

self.btn\_flowers\_menu.setText("Цветы")

self.btn\_flowers\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!

# start --------page flavors------------

self.btn\_flavors\_menu.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_flavors\_menu.setObjectName("btn\_flavors\_menu")

self.btn\_flavors\_menu.setText("Букеты")

self.btn\_flavors\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

# end --------page flavors------------

# start --------page composition------------

self.btn\_composition\_menu.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_composition\_menu.setObjectName("btn\_composition\_menu")

self.btn\_composition\_menu.setText("Композиции") self.btn\_composition\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor))

# end --------page composition------------

self.btn\_orders\_menu.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_orders\_menu.setObjectName("btn\_orders\_menu")

self.btn\_orders\_menu.setText("Заказы")

self.btn\_flowers\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # page\_flowers

self.btn\_flavors\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # page\_flavors

self.btn\_composition\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # composition

self.btn\_products\_menu.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_products\_menu.setObjectName("btn\_products\_menu")

self.btn\_products\_menu.setText("Фильтр")

self.btn\_flowers\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # page\_flowers

self.btn\_flavors\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # page\_flavors

self.btn\_composition\_menu.setCursor(QtGui.QCursor(Qt.PointingHandCursor)) # composition

self.central\_body.setFrameShape(QtWidgets.QFrame.NoFrame)

self.central\_body.setFrameShadow(QtWidgets.QFrame.Raised)

self.central\_body.setObjectName("central\_body")

self.middle\_body.setFrameShape(QtWidgets.QFrame.NoFrame)

self.middle\_body.setFrameShadow(QtWidgets.QFrame.Raised)

self.middle\_body.setObjectName("middle\_body")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0)

size\_policy.setHeightForWidth(self.stacked\_widget.sizePolicy().hasHeightForWidth())

self.stacked\_widget.setSizePolicy(size\_policy)

self.stacked\_widget.setStyleSheet("background-color:transparent;")

self.stacked\_widget.setObjectName("stacked\_widget")

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!

self.flowers\_page.setObjectName("flowers\_page")

self.flowers\_page.setStyleSheet("background-color: transparent;")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0)

size\_policy.setHeightForWidth(self.tbl.sizePolicy().hasHeightForWidth())

self.tbl\_flowers.setSizePolicy(size\_policy)

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI Light")

font.setPointSize(12)

font.setBold(True)

font.setItalic(False)

font.setWeight(75)

self.tbl\_flowers.setFont(font)

self.tbl\_flowers.setFrameShape(QtWidgets.QFrame.NoFrame)

self.tbl\_flowers.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.tbl\_flowers.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tbl\_flowers.setAutoScroll(True)

self.tbl\_flowers.setDragDropOverwriteMode(True)

self.tbl\_flowers.setAlternatingRowColors(False)

self.tbl\_flowers.setShowGrid(False)

self.tbl\_flowers.setCornerButtonEnabled(True)

self.tbl\_flowers.setObjectName("tbl\_flowers")

self.tbl\_flowers.setColumnCount(2)

self.tbl\_flowers.setRowCount(0)

self.tbl\_flowers.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem("ID"))

self.tbl\_flowers.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem("Название"))

self.tbl\_flowers.horizontalHeader().setVisible(True)

self.tbl\_flowers.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch)

self.tbl\_flowers.horizontalHeader().setSectionResizeMode(0, QtWidgets.QHeaderView.ResizeToContents)

self.tbl\_flowers.horizontalHeader().setSectionResizeMode(1, QtWidgets.QHeaderView.ResizeToContents)

self.tbl\_flowers.horizontalHeader().setCascadingSectionResizes(True)

self.tbl\_flowers.horizontalHeader().setStretchLastSection(True)

self.tbl\_flowers.verticalHeader().setVisible(False)

self.tbl\_flowers.verticalHeader().setCascadingSectionResizes(False)

self.tbl\_flowers.verticalHeader().setHighlightSections(False)

self.tbl\_flowers.verticalHeader().setSortIndicatorShown(False)

self.btns\_table\_flowers.setSizePolicy(size\_policy)

self.btns\_table\_flowers.setStyleSheet("""QPushButton {

background-color: #7C9A6D;

}

QPushButton:hover {

background-color: rgb(161, 183, 149);

}""")

self.btns\_table\_flowers.setObjectName("btns\_table")

self.btn\_add\_flower.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_add\_flower.setObjectName("btn\_add")

self.btn\_add\_flower.setText("Добавить")

self.btn\_del\_flower.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_del\_flower.setObjectName("btn\_del")

self.btn\_del\_flower.setText("Удалить")

self.btn\_save\_flower.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_save\_flower.setObjectName("btn\_save")

self.btn\_save\_flower.setText("Сохранить")

self.btn\_load\_flower.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_load\_flower.setObjectName("btn\_load")

self.btn\_load\_flower.setText("Загрузить")

# !!!!!!!!!!!!!!!!!!!!!!!!!!!!

# start --------page flavors------------

self.flavor\_page.setObjectName("flavors\_page")

self.flavor\_page.setStyleSheet("background-color: transparent;")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0)

size\_policy.setHeightForWidth(self.tbl.sizePolicy().hasHeightForWidth())

self.tbl\_flavors.setSizePolicy(size\_policy)

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI Light")

font.setPointSize(12)

font.setBold(True)

font.setItalic(False)

font.setWeight(75)

self.tbl\_flavors.setFont(font)

self.tbl\_flavors.setFrameShape(QtWidgets.QFrame.NoFrame)

self.tbl\_flavors.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.tbl\_flavors.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tbl\_flavors.setAutoScroll(True)

self.tbl\_flavors.setDragDropOverwriteMode(True)

self.tbl\_flavors.setAlternatingRowColors(False)

self.tbl\_flavors.setShowGrid(False)

self.tbl\_flavors.setCornerButtonEnabled(True)

self.tbl\_flavors.setObjectName("tbl\_flowers")

self.tbl\_flavors.setColumnCount(3)

self.tbl\_flavors.setRowCount(0)

self.tbl\_flavors.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem("ID"))

self.tbl\_flavors.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem("Название"))

self.tbl\_flavors.setHorizontalHeaderItem(2, QtWidgets.QTableWidgetItem("Стоимость")

self.tbl\_flavors.horizontalHeader().setVisible(True)

self.tbl\_flavors.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch)

self.tbl\_flavors.horizontalHeader().setSectionResizeMode(0, QtWidgets.QHeaderView.ResizeToContents)

self.tbl\_flavors.horizontalHeader().setCascadingSectionResizes(True)

self.tbl\_flavors.horizontalHeader().setStretchLastSection(True)

self.tbl\_flavors.verticalHeader().setVisible(False)

self.tbl\_flavors.verticalHeader().setCascadingSectionResizes(False)

self.tbl\_flavors.verticalHeader().setHighlightSections(False)

self.tbl\_flavors.verticalHeader().setSortIndicatorShown(False)

self.btns\_table\_flavors.setSizePolicy(size\_policy)

self.btns\_table\_flavors.setStyleSheet("""QPushButton {

background-color: #7C9A6D;

}

QPushButton:hover {

background-color: rgb(161, 183, 149);

}""")

self.btns\_table\_flavors.setObjectName("btns\_table")

self.btn\_add\_flavor.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_add\_flavor.setObjectName("btn\_add")

self.btn\_add\_flavor.setText("Добавить")

self.btn\_del\_flavor.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_del\_flavor.setObjectName("btn\_del")

self.btn\_del\_flavor.setText("Удалить")

self.btn\_save\_flavor.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_save\_flavor.setObjectName("btn\_save")

self.btn\_save\_flavor.setText("Сохранить")

self.btn\_load\_flavor.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_load\_flavor.setObjectName("btn\_load")

self.btn\_load\_flavor.setText("Загрузить")

# end --------page flavors------------

# start --------page page composition------------

self.combo\_lbl.setText("Выберите букет: ")

self.composition\_page.setObjectName("composition\_page")

self.composition\_page.setStyleSheet("background-color: transparent;")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0)

size\_policy.setHeightForWidth(self.tbl.sizePolicy().hasHeightForWidth())

self.tbl\_composition.setSizePolicy(size\_policy)

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI Light")

font.setPointSize(12)

font.setBold(True)

font.setItalic(False)

font.setWeight(75)

self.tbl\_composition.setFont(font)

self.tbl\_composition.setFrameShape(QtWidgets.QFrame.NoFrame)

self.tbl\_composition.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.tbl\_composition.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tbl\_composition.setAutoScroll(True)

self.tbl\_composition.setDragDropOverwriteMode(True)

self.tbl\_composition.setAlternatingRowColors(False)

self.tbl\_composition.setShowGrid(False)

self.tbl\_composition.setCornerButtonEnabled(True)

self.tbl\_composition.setObjectName("tbl\_composition")

self.tbl\_composition.setColumnCount(3)

self.tbl\_composition.setRowCount(0)

self.tbl\_composition.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem("ID"))

self.tbl\_composition.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem("Цветок"))

self.tbl\_composition.setHorizontalHeaderItem(2, QtWidgets.QTableWidgetItem("Количество"))

self.tbl\_composition.horizontalHeader().setVisible(True)

self.tbl\_composition.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch)

self.tbl\_composition.horizontalHeader().setSectionResizeMode(0, QtWidgets.QHeaderView.ResizeToContents)

self.tbl\_composition.horizontalHeader().setCascadingSectionResizes(True)

self.tbl\_composition.horizontalHeader().setStretchLastSection(True)

self.tbl\_composition.verticalHeader().setVisible(False)

self.tbl\_composition.verticalHeader().setCascadingSectionResizes(False)

self.tbl\_composition.verticalHeader().setHighlightSections(False)

self.tbl\_composition.verticalHeader().setSortIndicatorShown(False)

self.btns\_table\_composition.setSizePolicy(size\_policy)

self.btns\_table\_composition.setStyleSheet("""QPushButton {

background-color: #7C9A6D;

}

QPushButton:hover {

background-color: rgb(161, 183, 149);

}""")

self.btns\_table\_composition.setObjectName("btns\_table")

self.btn\_add\_composition.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_add\_composition.setObjectName("btn\_add")

self.btn\_add\_composition.setText("Добавить")

self.btn\_del\_composition.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_del\_composition.setObjectName("btn\_del")

self.btn\_del\_composition.setText("Удалить")

self.btn\_save\_composition.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_save\_composition.setObjectName("btn\_save")

self.btn\_save\_composition.setText("Сохранить")

self.btn\_load\_composition.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_load\_composition.setObjectName("btn\_load")

self.btn\_load\_composition.setText("Загрузить")

# end --------page composition------------

self.orders\_page.setStyleSheet("background-color: transparent;")

self.orders\_page.setObjectName("orders\_page")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0)

size\_policy.setHeightForWidth(self.tbl.sizePolicy().hasHeightForWidth())

self.tbl.setSizePolicy(size\_policy)

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI Light")

font.setPointSize(12)

font.setBold(True)

font.setItalic(False)

font.setWeight(75)

self.tbl.setFont(font)

self.tbl.setFrameShape(QtWidgets.QFrame.NoFrame)

self.tbl.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)

self.tbl.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)

self.tbl.setAutoScroll(True)

self.tbl.setDragDropOverwriteMode(True)

self.tbl.setAlternatingRowColors(False)

self.tbl.setShowGrid(False)

self.tbl.setCornerButtonEnabled(True)

self.tbl.setObjectName("tbl")

self.tbl.setColumnCount(8)

self.tbl.setRowCount(0)

item = QtWidgets.QTableWidgetItem("ID")

self.tbl.setHorizontalHeaderItem(0, item)

item = QtWidgets.QTableWidgetItem("Фамилия")

self.tbl.setHorizontalHeaderItem(1, item)

item = QtWidgets.QTableWidgetItem("Имя")

self.tbl.setHorizontalHeaderItem(2, item)

item = QtWidgets.QTableWidgetItem("Телефон")

self.tbl.setHorizontalHeaderItem(3, item)

item = QtWidgets.QTableWidgetItem("Букет")

self.tbl.setHorizontalHeaderItem(4, item)

item = QtWidgets.QTableWidgetItem("Дата принятия")

self.tbl.setHorizontalHeaderItem(5, item)

item = QtWidgets.QTableWidgetItem("Дата выдачи")

self.tbl.setHorizontalHeaderItem(6, item)

item = QtWidgets.QTableWidgetItem("Цена букетов")

self.tbl.setHorizontalHeaderItem(7, item)

self.tbl.horizontalHeader().setVisible(True)

self.tbl.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch)

self.tbl.horizontalHeader().setSectionResizeMode(0, QtWidgets.QHeaderView.ResizeToContents)

self.tbl.horizontalHeader().setSectionResizeMode(1, QtWidgets.QHeaderView.ResizeToContents)

self.tbl.horizontalHeader().setSectionResizeMode(2, QtWidgets.QHeaderView.ResizeToContents)

self.tbl.horizontalHeader().setCascadingSectionResizes(True)

self.tbl.horizontalHeader().setStretchLastSection(True)

self.tbl.verticalHeader().setVisible(False)

self.tbl.verticalHeader().setCascadingSectionResizes(False)

self.tbl.verticalHeader().setHighlightSections(False)

self.tbl.verticalHeader().setSortIndicatorShown(False)

self.orders\_cost.setMaximumSize(QtCore.QSize(16777215, 45))

self.orders\_cost.setStyleSheet("""background-color: qlineargradient(spread:reflect,

x1:0.5, y1:0.5, x2:1, y2:0.5,

stop:0.271186 rgba(170, 198, 163, 255), stop:1 rgba(255, 255, 255, 0));""")

self.orders\_cost.setObjectName("orders\_cost")

self.lbl\_cost.setObjectName("lbl\_cost")

self.btns\_table.setSizePolicy(size\_policy)

self.btns\_table.setStyleSheet("""QPushButton {

background-color: #7C9A6D;

}

QPushButton:hover {

background-color: rgb(161, 183, 149);

}""")

self.btns\_table.setObjectName("btns\_table")

self.btn\_add.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_add.setObjectName("btn\_add")

self.btn\_add.setText("Добавить")

self.btn\_del.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_del.setObjectName("btn\_del")

self.btn\_del.setText("Удалить")

self.btn\_save.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_save.setObjectName("btn\_save")

self.btn\_save.setText("Сохранить")

self.btn\_load.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_load.setObjectName("btn\_load")

self.btn\_load.setText("Загрузить")

self.products\_page.setStyleSheet("background-color: transparent;")

self.products\_page.setObjectName("products\_page")

self.body\_products.setObjectName("body\_products")

self.left\_body.setObjectName("left\_body")

self.right\_body.setObjectName("right\_body")

self.lbl\_flowers.setStyleSheet("""background-color: qlineargradient(spread:reflect, x1:0.5, y1:0.5, x2:1,

y2:0.5, stop:0.271186 rgba(170, 198, 163, 255),

stop:1 rgba(255, 255, 255, 0));""")

self.lbl\_flowers.setAlignment(Qt.AlignCenter)

self.lbl\_flowers.setObjectName("lbl\_flowers")

self.lbl\_flowers.setText("Цветы")

self.lbl\_flowers.setObjectName("lbl\_flowers")

self.btn\_products.setMinimumSize(QtCore.QSize(100, 50))

self.btn\_products.setText("Обновить")

self.btn\_products.setStyleSheet("""QPushButton{

background-color: #7C9A6D;

}

QPushButton:hover{

background-color: #AAC6A3;

}""")

self.btn\_products.setObjectName("btn\_products")

self.sort\_flowers.setStyleSheet("""QFrame#sort\_flowers QLabel{ background-color:

qlineargradient(spread:reflect, x1:0.5, y1:0.5, x2:1, y2:0.5,

stop:0.568182 rgba(170, 198, 163, 255),

stop:1 rgba(255, 255, 255, 0));}""")

self.sort\_flowers.setObjectName("sort\_flowers")

size\_policy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

size\_policy.setHorizontalStretch(0)

size\_policy.setVerticalStretch(0) size\_policy.setHeightForWidth(self.lbl\_date\_begin.sizePolicy().hasHeightForWidth())

self.lbl\_date\_begin.setSizePolicy(size\_policy)

self.lbl\_date\_begin.setAlignment(Qt.AlignCenter)

self.lbl\_date\_begin.setObjectName("lbl\_date\_begin")

self.lbl\_date\_begin.setText("Начало поиска:")

self.date\_begin\_flowers.setSizePolicy(size\_policy)

self.date\_begin\_flowers.setStyleSheet("background-color: rgb(192, 209, 184);")

self.date\_begin\_flowers.setDateTime(QtCore.QDateTime(QtCore.QDate(2022, 1, 1), QtCore.QTime(0, 0, 0)))

self.date\_begin\_flowers.setCalendarPopup(True)

self.date\_begin\_flowers.setObjectName("date\_begin\_flowers")

self.lbl\_date\_end.setSizePolicy(size\_policy)

self.lbl\_date\_end.setAlignment(Qt.AlignCenter)

self.lbl\_date\_end.setObjectName("lbl\_date\_end")

self.lbl\_date\_end.setText("Конец поиска:")

self.date\_end\_flowers.setSizePolicy(size\_policy)

self.date\_end\_flowers.setStyleSheet("background-color: rgb(192, 209, 184);")

self.date\_end\_flowers.setDateTime(QtCore.QDateTime(QtCore.QDate(2022, 1, 1), QtCore.QTime(0, 0, 0)))

self.date\_end\_flowers.setCalendarPopup(True)

self.date\_end\_flowers.setObjectName("date\_end\_flowers")

self.lbl\_combo\_flowers.setSizePolicy(size\_policy)

self.lbl\_combo\_flowers.setAlignment(Qt.AlignCenter)

self.lbl\_combo\_flowers.setObjectName("lbl\_combo\_flowers")

self.lbl\_combo\_flowers.setText("Поиск по цветам:")

self.combo\_flowers.setSizePolicy(size\_policy)

self.combo\_flowers.setStyleSheet("background-color: rgb(234, 241, 232);")

self.combo\_flowers.setFrame(False)

self.combo\_flowers.setObjectName("combo\_flowers")

self.combo\_flowers.addItem("Все цветы")

self.empty\_frame.setMinimumSize(QtCore.QSize(0, 200))

self.empty\_frame.setObjectName("empty\_frame")

self.main\_footer.setMinimumSize(QtCore.QSize(0, 25))

self.main\_footer.setMaximumSize(QtCore.QSize(16777215, 25))

self.main\_footer.setObjectName("main\_footer")

self.lbl\_info\_tbl.setSizePolicy(size\_policy)

self.lbl\_info\_tbl.setAlignment(Qt.AlignCenter)

self.lbl\_info\_tbl.setObjectName("lbl\_info\_tbl")

self.lbl\_info\_tbl.setStyleSheet("""font: 12pt "Microsoft YaHei UI Light";

font-weight: bold;

color: rgb(103, 125, 92);""")

self.size\_grip.setMinimumSize(QtCore.QSize(20, 20))

self.size\_grip.setMaximumSize(QtCore.QSize(20, 20))

self.size\_grip.setObjectName("size\_grip")

1. 1) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. 2) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. 3) ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. 4) ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. 5) ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. 6) ГОСТ 19.505-79\* ЕСПД. Руководство оператора. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. 7) ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)