*Отчет по лабораторной работе№3 предоставил: Нуранов Айдар, 931920гр.*

**1.Задача.**

Задача

Необходимо разработать приложение печати графиков в pdf формате по предложенной архитектуре ПО, в которую требуется внести изменения с целью снижения связности архитектуры. Используя IOC контейнер- внедряем зависимости. Графики различаются типом графика, стилем графика и данными.

Различия графиков:

1. Исходные входные данные определяются пользователем и характеризуются парой [значение, дата]. Данные могут храниться в БД SQLite или же могут быть представлены в json файле.
2. Визуальный стиль. График может быть цветным или черно-белым.
3. Тип графика. График может быть представлен в виде столбчатой (bar) или круговой(pie) диаграмм.

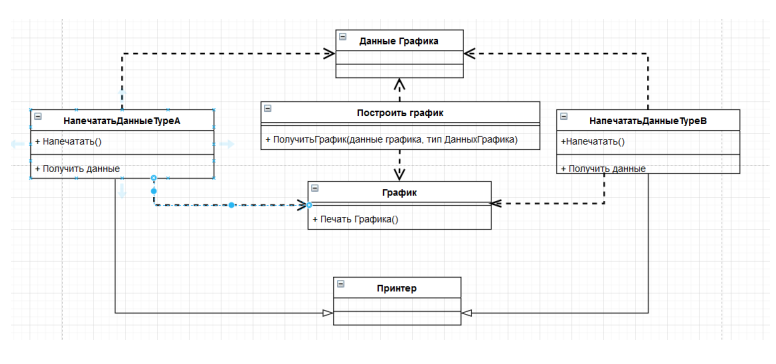
Общие требования к GUI:

1. Загружаем данные, путем выбора нужного файла. Данные в ПО не отображаем, отображаем только график, построенный относительно считанных данных.
2. При печати в pdf выбираем место сохранения графика.

**2.Решение.**

Решение

Для решения поставленной задачи необходимо применить предложенную архитектуру. Ее можно представить в виде следующей Uml-диаграммы



*IOC*

Также необходимо внедрить представленную архитектуру IOС контейнер. Что же это такое? IOC (Inversion of Control) – это важный принцип ООП реализующий инверсию управления. Данный принцип отвечает за уменьшение связности между компонентами системы. Приложение зависит от некоторой абстракции, а реализация скрыта, и в любой момент может быть заменена другой.

IOC отвечает за:

1. Фактическое создание конкретного объекта
2. Выбор конструктора и возврат ссылочного типа
3. Перед вызовом конструктора - получение экземпляров любых зависимостей

*MVC ("модель-вид-контроллер"):*

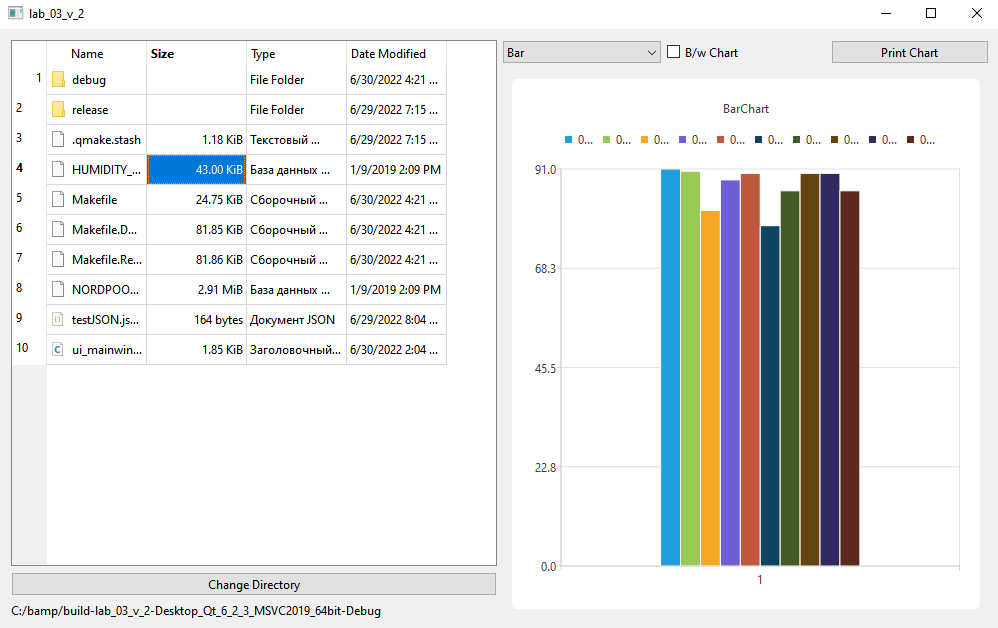
При реализации программы использовалась схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер - таким образом, что модификация каждого компонента может осуществляться независимо.

Класс QFileSystemModel предоставляет модель данных для локальной файловой системы

QFileSystemModel \*fileModel; //файловая модель для таблицы выбора файла

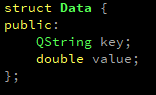
Представление использовали табличное

QTableView \*tableFileView; //таблица выбора файла

Визуальный интерфейс программы выглядит следующим образом. 

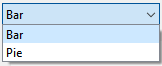
Описание: Слева с помощью кнопки “Change directory” пользователь имеет возможность выбрать папку с файлами только с расширениями .sqlite или .json. Выбранные данные в приложении отображаются в табличном представлении. Слот **changeDirectory**() организует открытие данной директории и её представление. При нажатии на определенный файл - справа отображается график на основе этих данных, этим занимается слот - **fileSelection**(const QItemSelection &selected, const QItemSelection &deselected) (анализирует все эти данные, учитывает тип графика и стиль, и перерисовывает график, отображая его). Обработка и получение данных в нужном виде [значение, дата] происходит при помощи функций ***getData*** (QString path\_)- там мы проверяем бд или json на успешное открытие и далее с помощью select запроса, в случае бд, или при помощи получения списка ключей, в случае json, собираем данные. Далее проходя по всем данным в цикле формируем, и возвращаем данные вида (QList<Data>);

Где Data это:

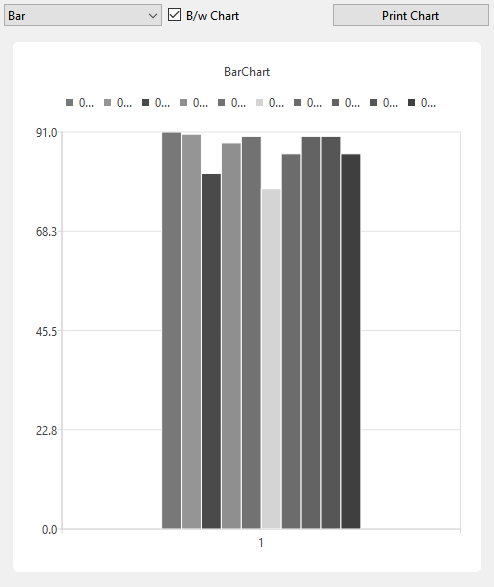


Из выпадающего списка вверху можно выбрать тип диаграммы Pie или Bar

За данный выпадающий список отвечает слот **changeChartType**(); в нем находим подходящий тип графика, связываем интерфейс (IOC) с конкретной реализацией с помощью функции, строящей график ТИПграфикаChart (BarChart, PieChart). Если выбор типа изменился- перерисовываем график функцией drawChart()



Нажимая на отметку чекбокса, график станет черно-белого цвета. Это организовано с помощью слота **colorSwap**(); в котором проверяется наличие нажатой отметки checkState() и далее график вызывает функцию ***recreateChart*** (QList<Data>, bool isBWEnabled) = 0;. Аргумент isBWEnabled отвечает за цвет графика если аргумент true, то график будет черно-белым.

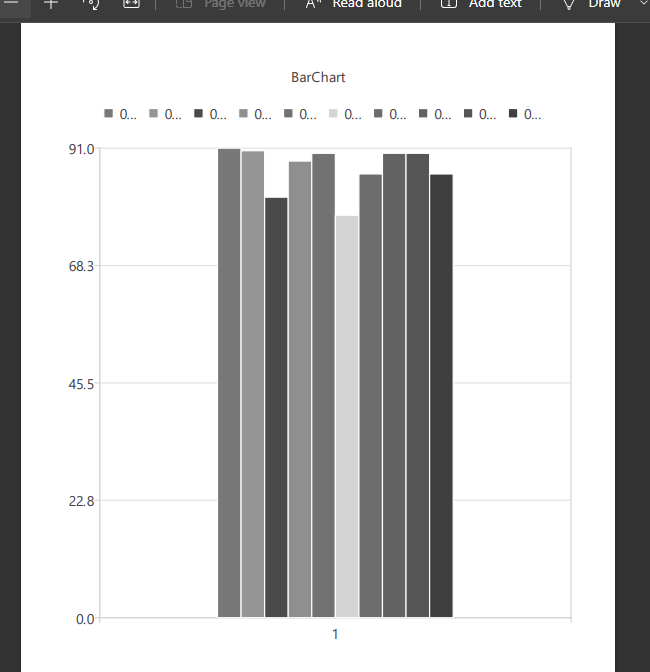


Печать графика организована в слоте **printChart**() там мы определяем путь к файлу, сохраняем в формате pdf (QPdfWriter\* pdfWriter = new QPdfWriter (fileNames.first() + ".pdf")) и отрисовываем график, используя его представление, с помощью инструмента QPainter

QPainter painter(*pdfWriter*);

chartView->render(*&painter*);

Вот в таком виде сохраняются данные в pdf с выбранным именем

Между частями окна существует плавающий виджет-разделитель QSplitter

**3. Код программы.**

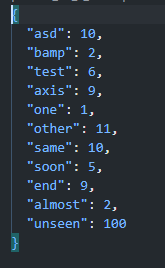
<https://github.com/nuranai/lab_03_v_2>

Разрабатывались такие файлы как, charts.cpp, charts.h, chartdata.cpp, chartdata.h, iocConainer.h, mainwindow.cpp, mainwindow.h, main.cpp

1. **Инструкция пользователя.**

При запуске оконного приложения пользователю необходимо нажать на Choose directory и выбрать папку с файлами sqlite или json.Данная папка отобразиться справа. Далее нажать на конкретный файл и тогда слева появится график ее анализирующий. Сверху можно выбрать другой тип графика pie- круговой, bar-столбчатый и поменять цвет с Цветного на черно-белый отметив чекбокс B/w Chart. Печать графика производиться при нажатии на кнопку Print chart, выбираем путь сохранения файла.

Примеры файла json: использует конструкцию ключ-значение



Примеры файлов sqlite со следующими названиями приведены в папке проекта:

BLOOD\_SUGAR

HUMIDITY\_MOSCOW

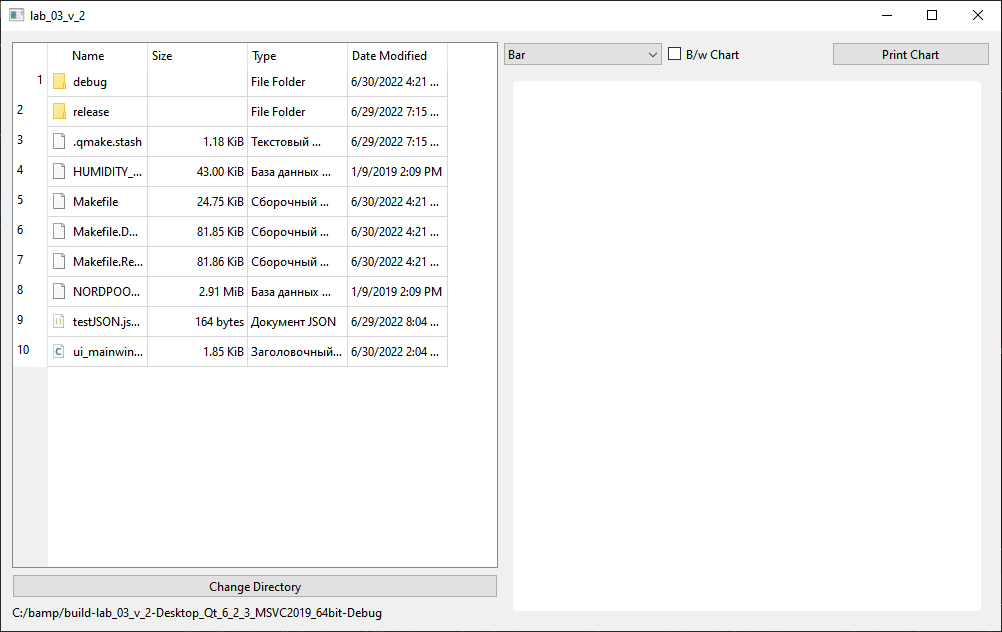
NORDPOOL\_PRICES

PRICES\_NATURAL\_GAS\_USD

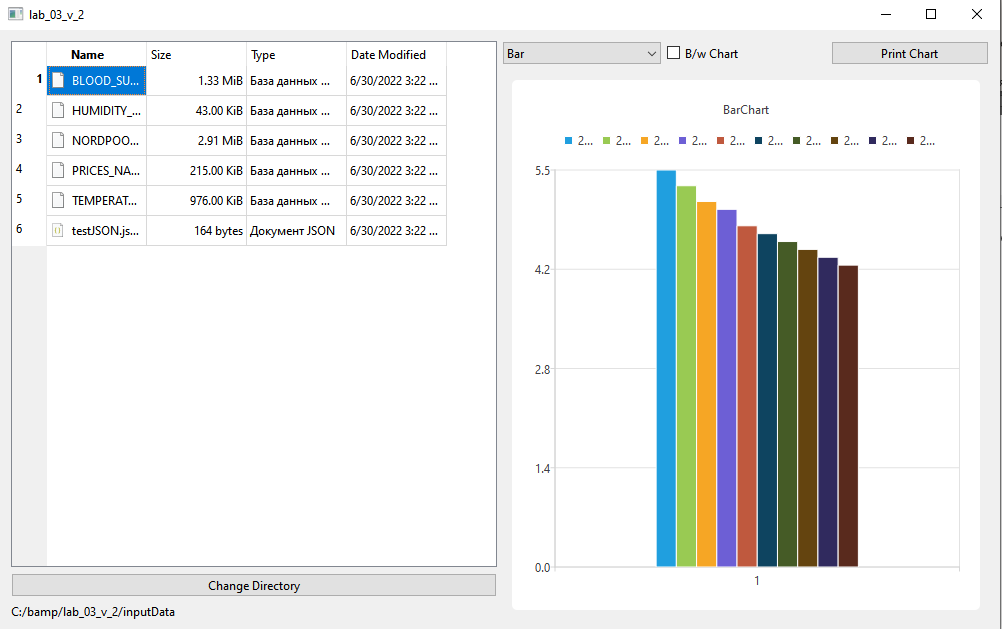
TEMPERATURE\_NOVOSIB

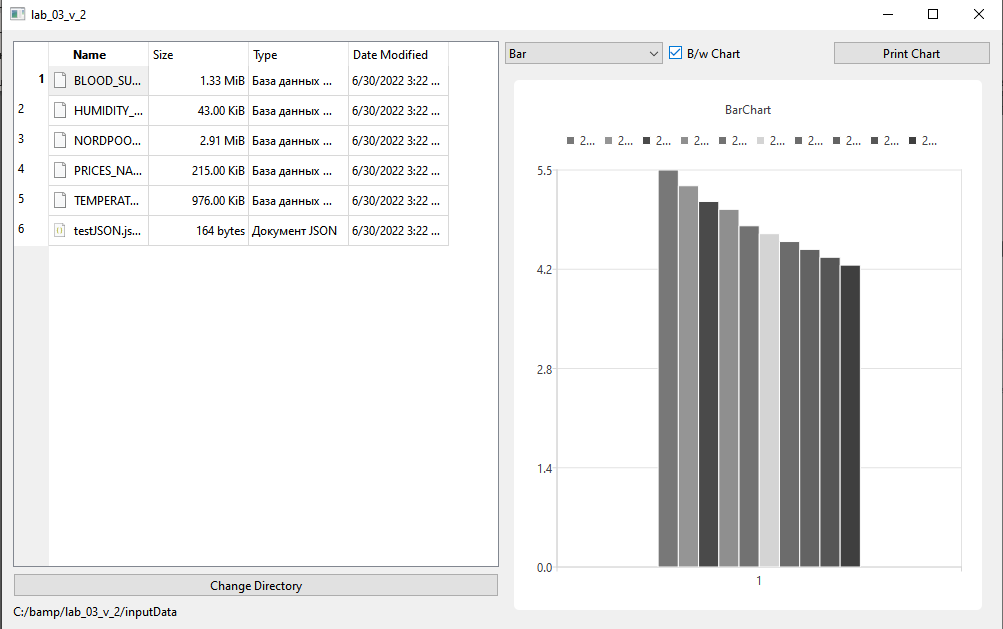
1. **Тестирование.**

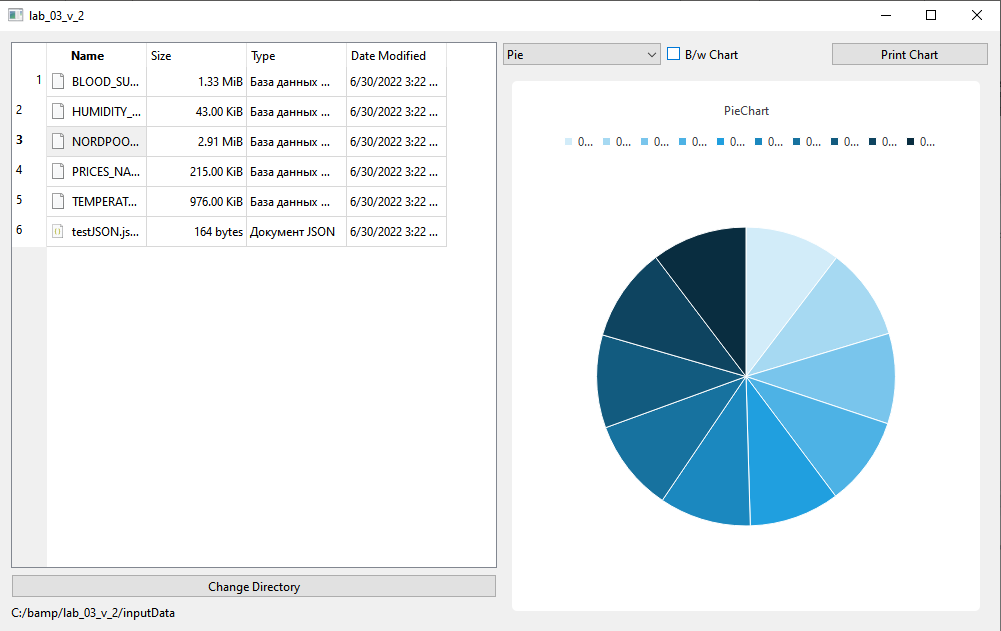
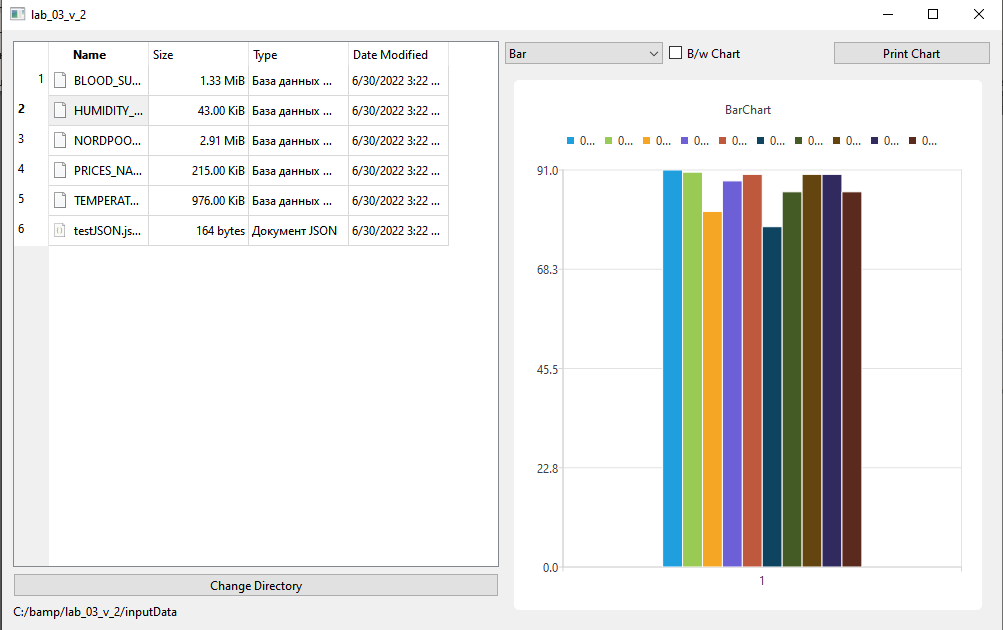
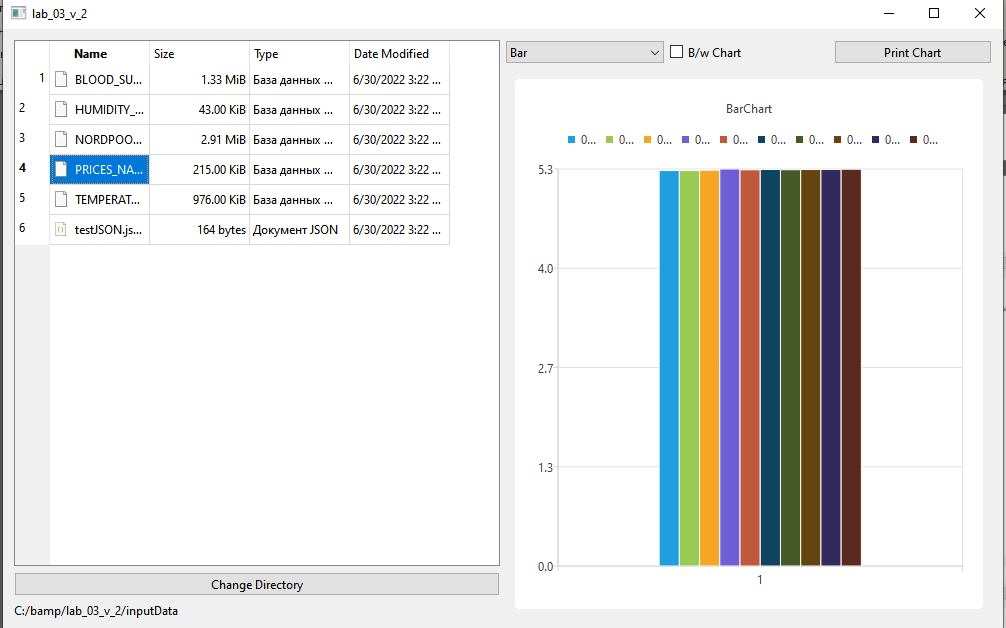
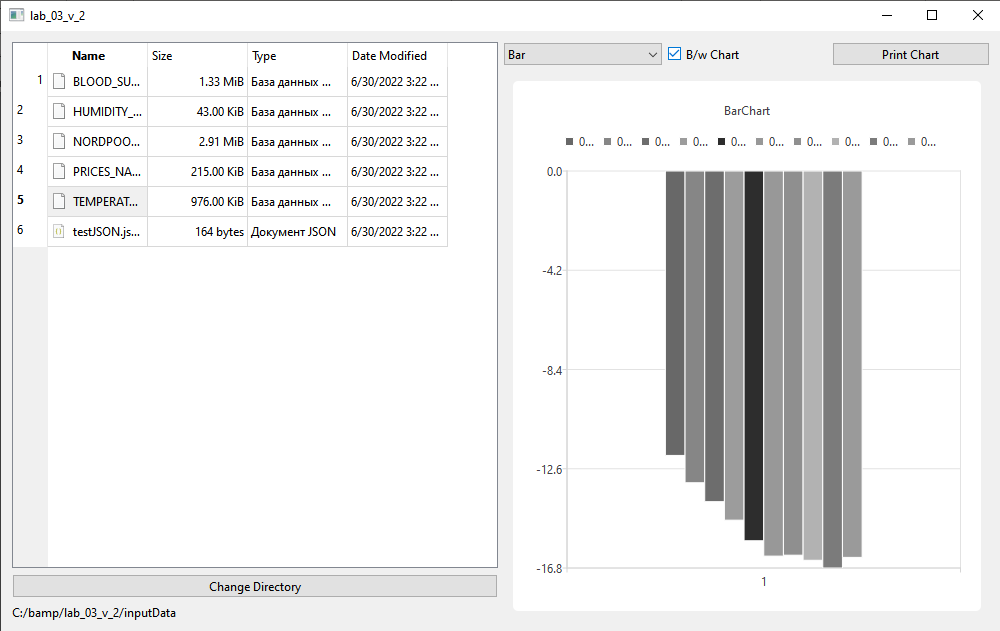
Вид окна изначально. Необходимо выбрать директорию

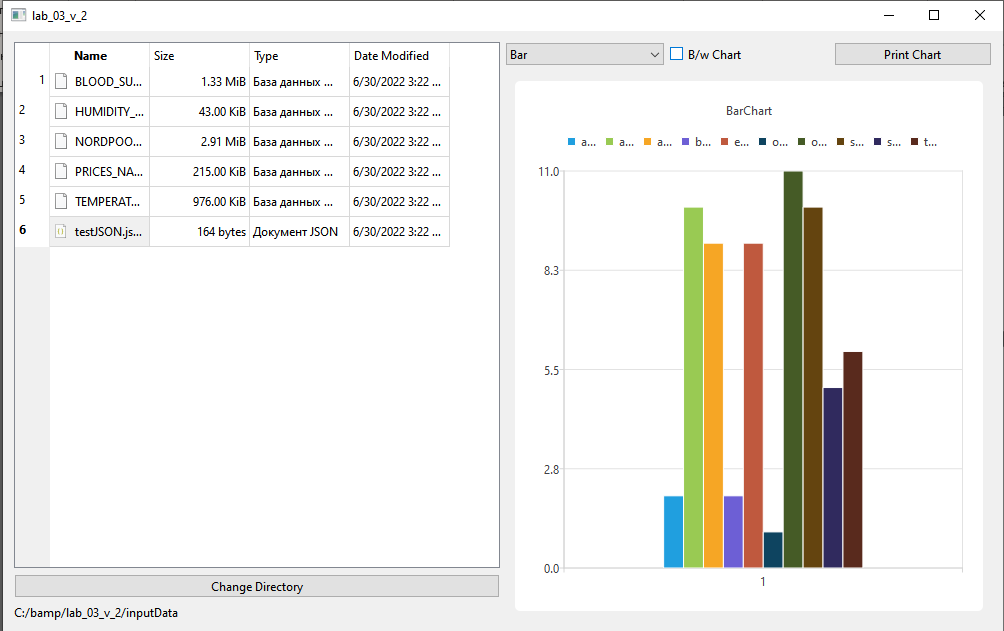
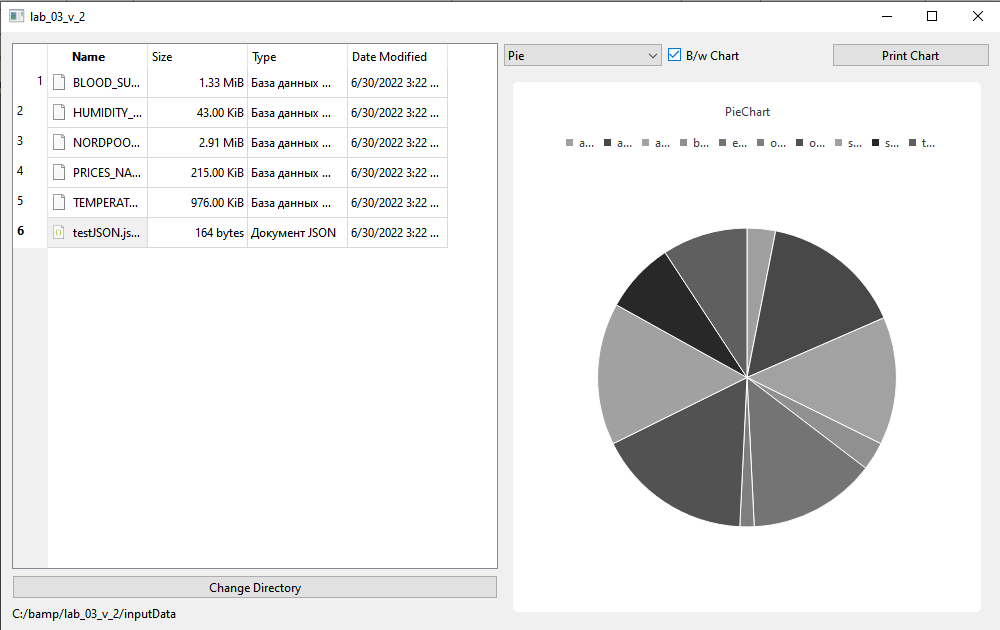


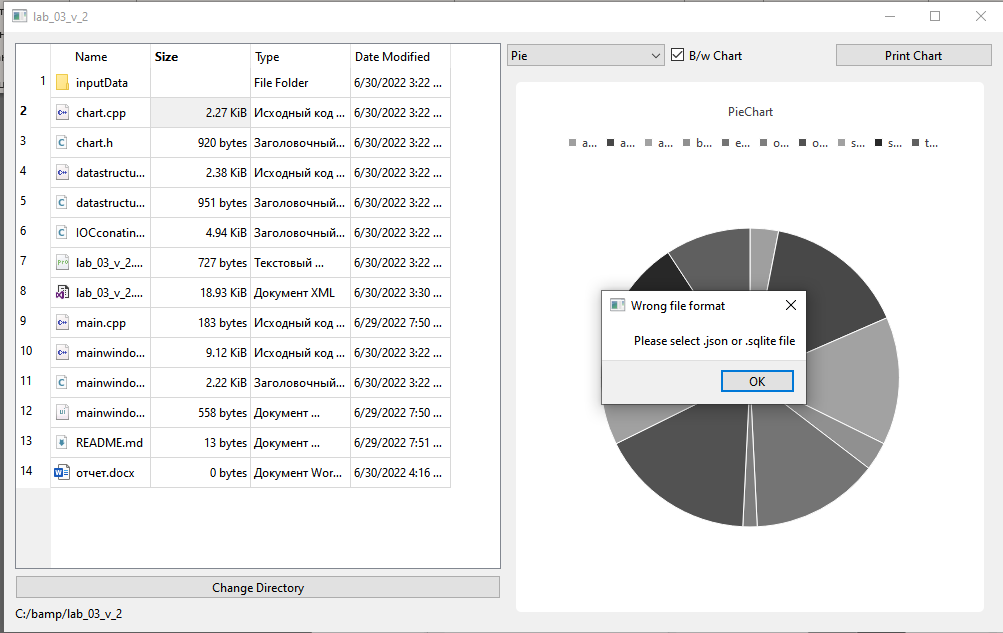
Выбрали директорию с предложенными файлами sqlite. Изначально график типа bar и цветной

Затем файл бд оставили тот же, но изменили цвет на ч/б

ниже скрины построения графиков на основе других файлов

   Построим график на основе данных из json

 При выборе файла с расширением не sqlite и не json программа сообщает об это. Например срр

Функционирование кнопки печати

