**Рынк Даниил Витальевич группа 931921**

1. Постановка задачи

Разработка приложения печати графиков.

Исходные входные  данные

***Исходные данные для печати*** соответствуют некоторому типу, который определятся пользователем. Данные определенного типа могут отображаться конкретным графиком, который ориентирован на этот тип данных.

Примеры данных.

1. Данные характеризуются парой **[значение, дата]**, хранятся в БД SQLite(архив с файлами прилагается).

2. Данные представлены JSON файлом. Формат данных [**значение , дата].**

**Дано:** предложен начальный вариант архитектуры ПО, в которую требуется внести изменения с целью снижения связности архитектуры. Используется принцип внедрения зависимости. Реализация внедрения зависимости с помощью IOC контейнера.

При разработке архитектуры учесть

Возможность добавления новых графиков (графики отличаются видом и данными

Изменение визуального стиля графиков (цветной, черно белый).

**Общие требования к GUI**

Загружаем данные, путем выбора нужного файла. Данные в ПО не отображаем, отображаем только график, построенный относительно считанных данных.

При печати в pdf выбираем место сохранения графика.

**Использование предложенной реализации IOC контейнера на с++**

Необходимо разобраться в предложенной реализации IOC контейнера.

Код сопроводить соответствующими объяснениями.

1. Предполагаемое решение

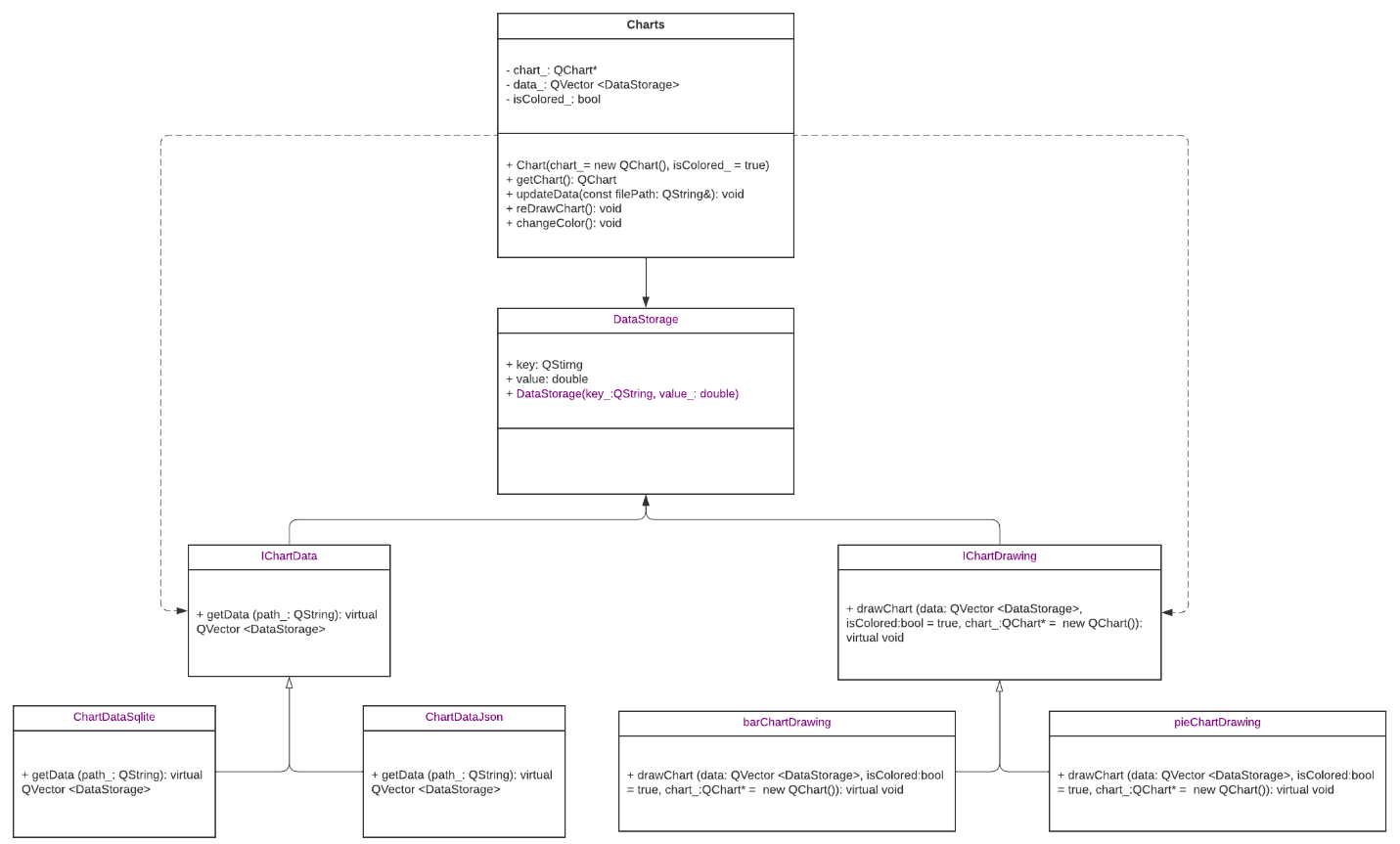
Для реализации нашего интерфейса я использовал следующее:  
- Для общей работы с данными их хранение, отображение и взаимодействия я использовал концепцию **MVC**. Эта концепция состоит из 3 типов объектов. **Model** (модель) – в которой хранятся все данные нашей программы. В моём случае это **QFileSystemModel** с помощью которого мы получали необходимый нам путь к файлу.

- Далее идёт **View** (представление) – оно отображает данные в нужном нам виде на экране. В нашем случае **QTableView.** С помощью Модельных индексов (QModelIndexList), полученных из Модели, мы можем получать доступ к нашим данным.

- Также последним элементом MVC является **Controller** (Контроллер) – он отвечает за пользовательский интерфейс и обеспечивает связь между пользователем и системой. В нашем случае контроллер уже вшит в представление.  
- Именно через наше представление я определял поиск файлов (также благодаря пути к файлу), с помощью QItemSelectionModel отслеживал выбранные элементы и в общем представлял в виде таблицы файлы.

- Для отображения Диаграмм я создал класс **Charts**, который содержал в себе элемент **QChart** который и описывает полностью диаграммы (легенды, серии и оси), а также элемент **QVector <DataStorage>,** содержащий в себе данные считанные с файлов в формате (Ключ, Значение) и флаг отвечающий за цвет диаграммы. В общем этот класс у меня отвечал полностью за построение Диаграмм, смену у них цвета, обновление у них данных из новых файлов и перерисовку Диаграмм по новым данным.  
- Но помимо отрисовки графика есть ещё возможность выбрать его тип. Это реализовано с помощью интерфейса **IChartDrawing** , который реализован с помощью IOC контейнера используя Инверсию Зависимости (DI). То есть в самом интерфейса у нас есть виртуальная функция отрисовки Диаграммы, но в ней изначально не задан тип диаграммы. А затем в нижестоящих реализациях **barChartDrawing** и **pieChartDrawing** мы отрисовываем конкретный тип и передаём его на наш интерфейс. Тем самым мы не зависим от этих реализаций, а зависим только от верхнестоящего интерфейса. И если нам понадобится мы сможем дописать ещё один класс реализующий другой тип данных никак не взаимодействую с основным интерфейсом.  
- Такая же методика применена и в чтение данных. Реализован интерфейс **IChartData,** также на IOC контейнере используя Инверсию Зависимости (DI). От него наследуются **ChartDataSqlite** и **ChartDataJson**, которые позволяют получать данные из баз соответствующих форматов. И если нам понадобится реализовать чтение нового формата, то мы сможем это спокойной сделать опять-таки не взаимодействие с основным интерфейсом. То есть у нас Класс верхнего уровня не завсит от классов нижних => Инверсия зависимости.ч   
- И самое последнее что ещё понадобилось для реализации это **QChartView –** виджет, который помогает нам отображать наши диаграммы на экране.  
  
Далее с использованием этих элементов, я собираю интерфейс, подключаю кнопки и взаимодействие с ними. Описываю выбор директорий с помощью **QFileDialog,** отдельное окно-диалог для выбора папки. А также вывод графика в формате PDF с помощью **QPdfWriter**.

**Получившаяся UML-диаграмма.**(В репозитории есть ещё 2 разновидности.)



1. Коды программ

Все файлы находятся в репозитории: [https://github.com/rynkdesign/TRPO\_lab](https://github.com/rynkdesign/TRPO_lab1)3

А именно:

charts.h – взаимодействие с графиками

charts.cpp  
data.h – взаимодействие с данными для отображения графика

data.cpp

ioccontainer.h – IOC контейнер

mainwindow.h - виджет главного окна  
mainwindow.cpp

main.cpp

1. Инструкция пользователя

Интерфейс программы состоит из 2 полей.

Левое – выбор файла для отображения графика.

Правое – само отображение графика.

При запуске правая часть будет пустая (т.к. график мы ещё не выбрали), а в левой части будет файлы корневой папки.

С помощью кнопки “Открыть папку” у нас есть возможность выбрать нужную нам папку, в которой хранятся наши базы данных.

Выбрав папку в левой части у нас, появится файлы, находящиеся в этой папке. (Их имя, размер, тип и дата изменения).

Кликнув левой кнопкой мыши по файлу тип которого поддерживается программой (на данный момент это “.json” и “.sqlite”) в правой части у нас отобразится график на основе этой базы данных.

В верхней части правого экрана интерфейса у нас есть меню взаимодействия с графиком.

Оно состоит из:

- Надписи: “Выберите тип диаграммы”

- Выпадающим списком: “BarChart” или “PieChart”

- Чекбоксом “Черно-белый график”

- Кнопкой “Печать графика”

При нажатии на выпадающий список, у нас появится возможность выбрать тип отображаемого графика. На данный момент поддерживается 2 типа “BarChart” – Вертикальный график и “PieChart” – Круговой.

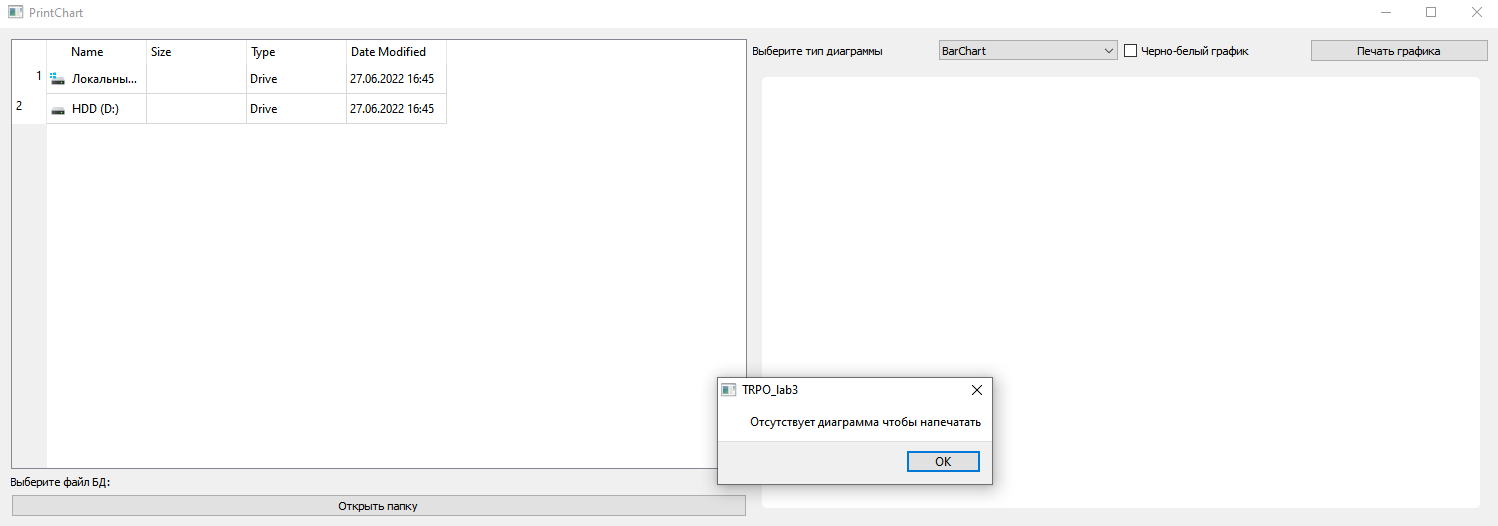
При нажатии на чекбокс “Черно-белый график” наш график (Не важно “BarChart” или “PieChart”) перекрасится в черно-белые цвета.

При нажатии на кнопку “Печать графика” программа предложит вам выбрать папку куда вы хотите сохранить ваш график и его имя (Подписывать “.pdf” не нужно). Затем файл формата PDF с вашим графиком появится в выбранной вами папке.

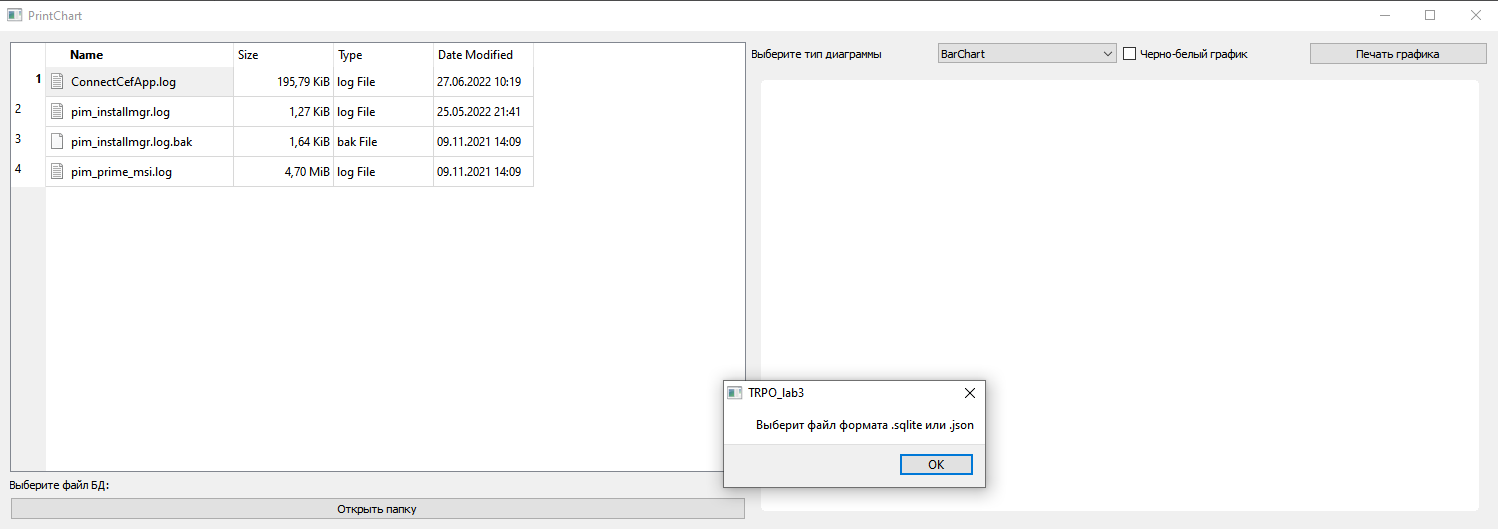
1. Тестирование

Все возможные отработки программы, скрины + описание. (Ошибки, правильные выводы, другие и тд)

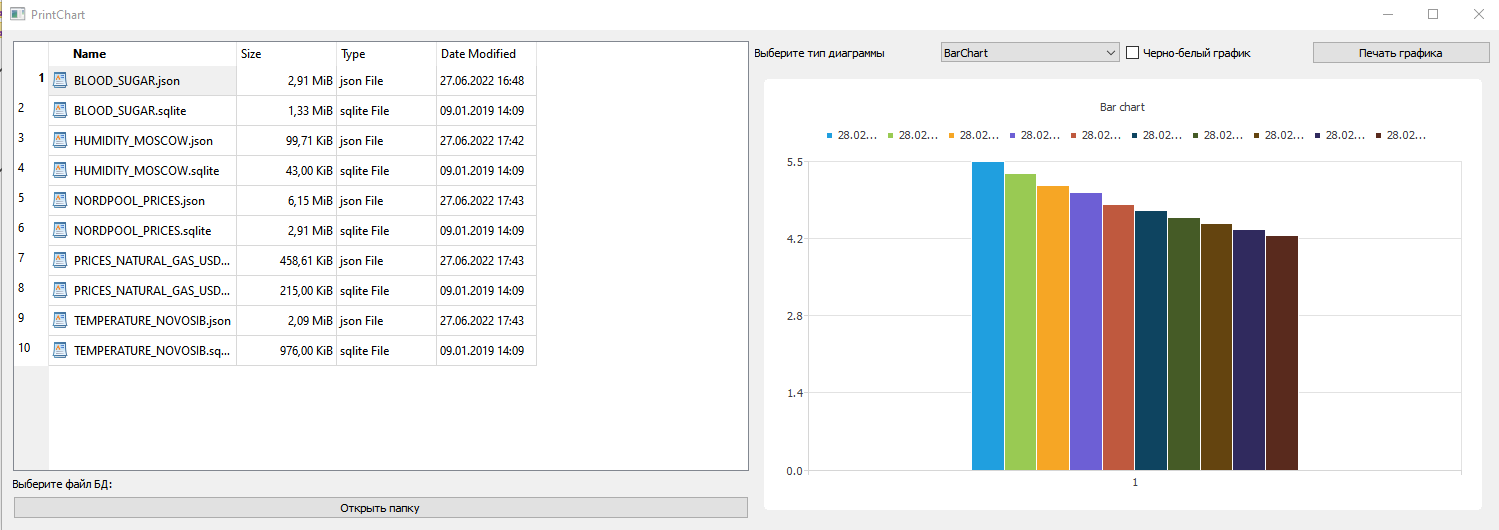
1. Если после запуска программы нажать кнопку “Печать графика”, то появится окно ошибки. Т.к. у нас ещё нет никакого графика для печати.



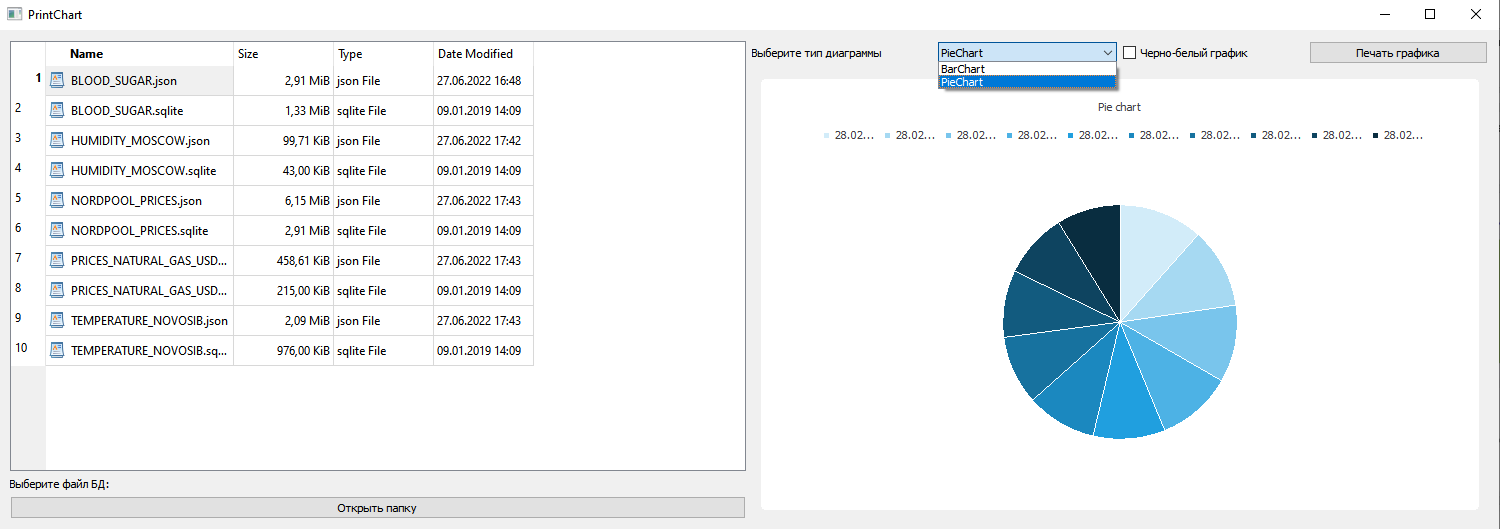
1. Если выбрать файл формата не “.sqlite” или “.json”, то появится окно ошибки. Т.к. у нас реализовано взаимодействие только с такими типами файлов.



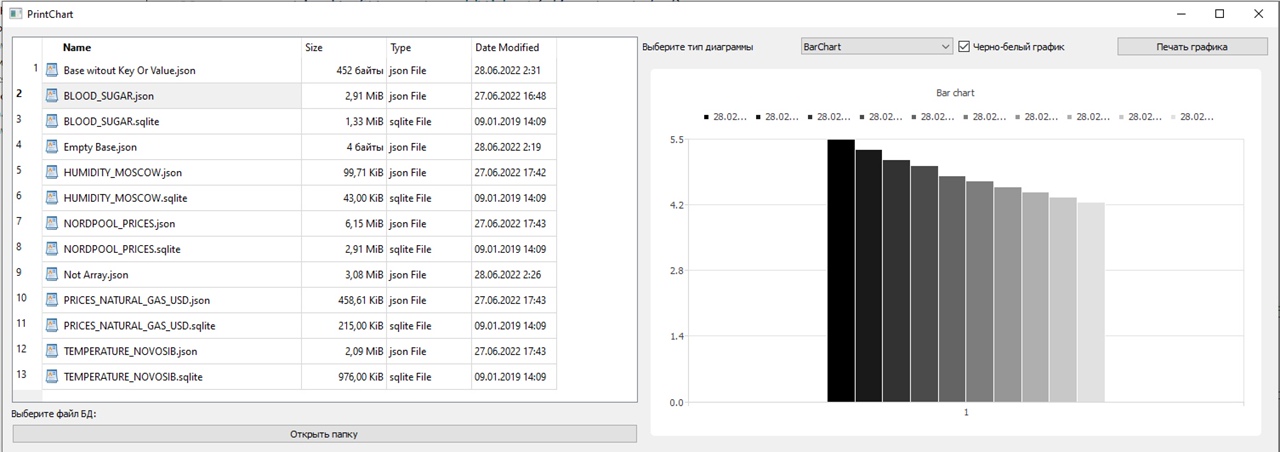
3. Если выбрать файл формата “.sqlite” или “.json”, то по умолчанию выведется вертикальный график показывающий первые 10 данных из базы.

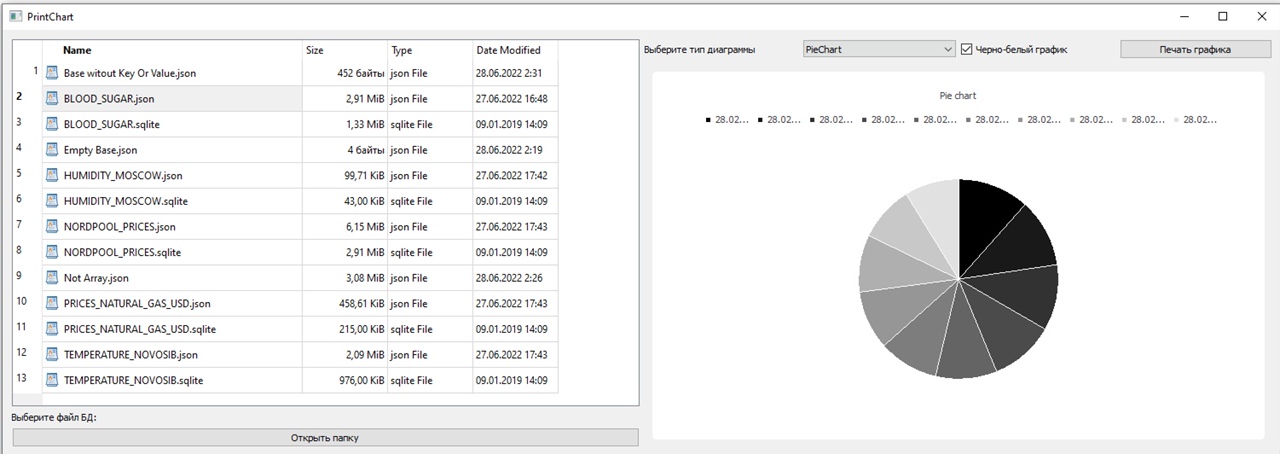


4. Если после выбора формата файла и отрисовки по умолчанию вертикального графика выбрать из выпадающего меню “Выберите тип диаграммы” – “PieChart”, тогда выведется круговой график показывающий первые 10 данных из базы.

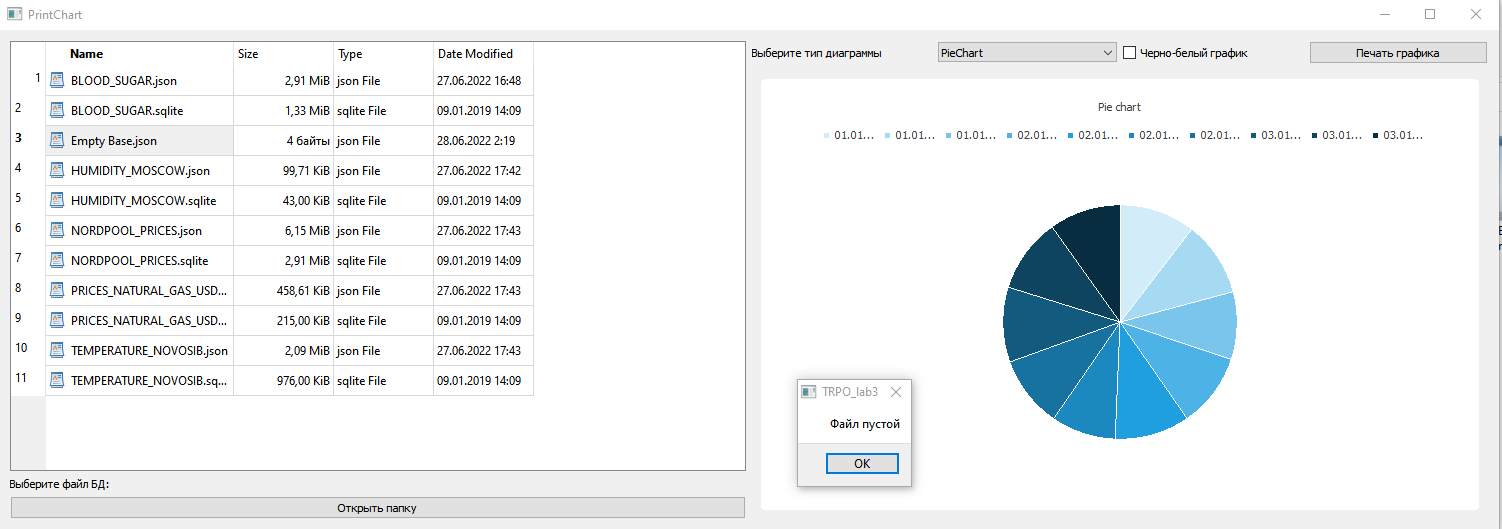


5. Если график уже отрисован (не важно Вертикальный или Круговой) и нажать на Чекбокс “Черно-белый график”, то выведется график в черно-белых цветах.

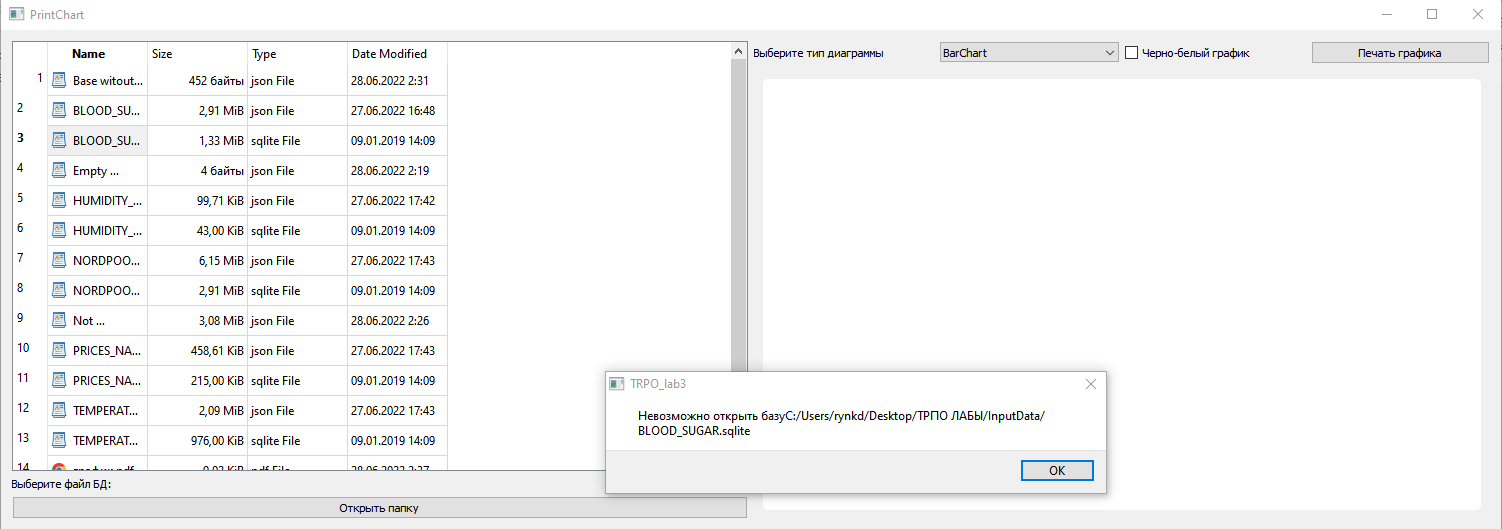




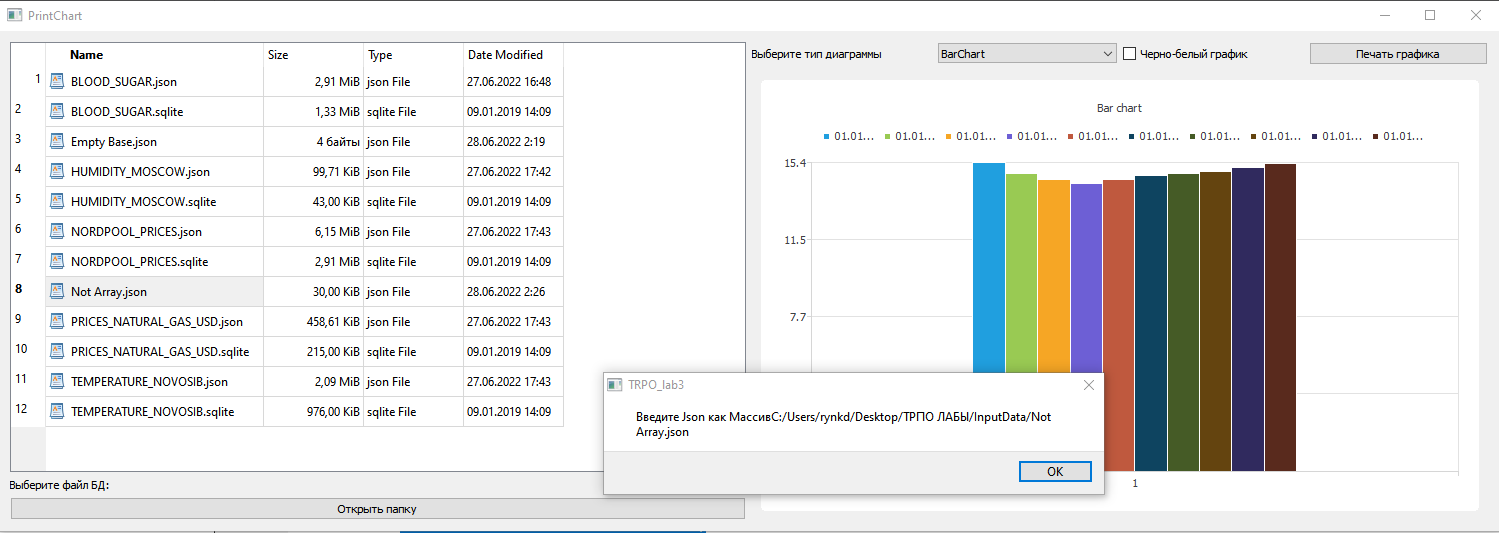
6. Если выбранная база формата “.sqlite” или “.json” оформлена как база, но не содержит в себе элементов (пустая база), то появится окно ошибки, сообщающий, что Файл пустой.



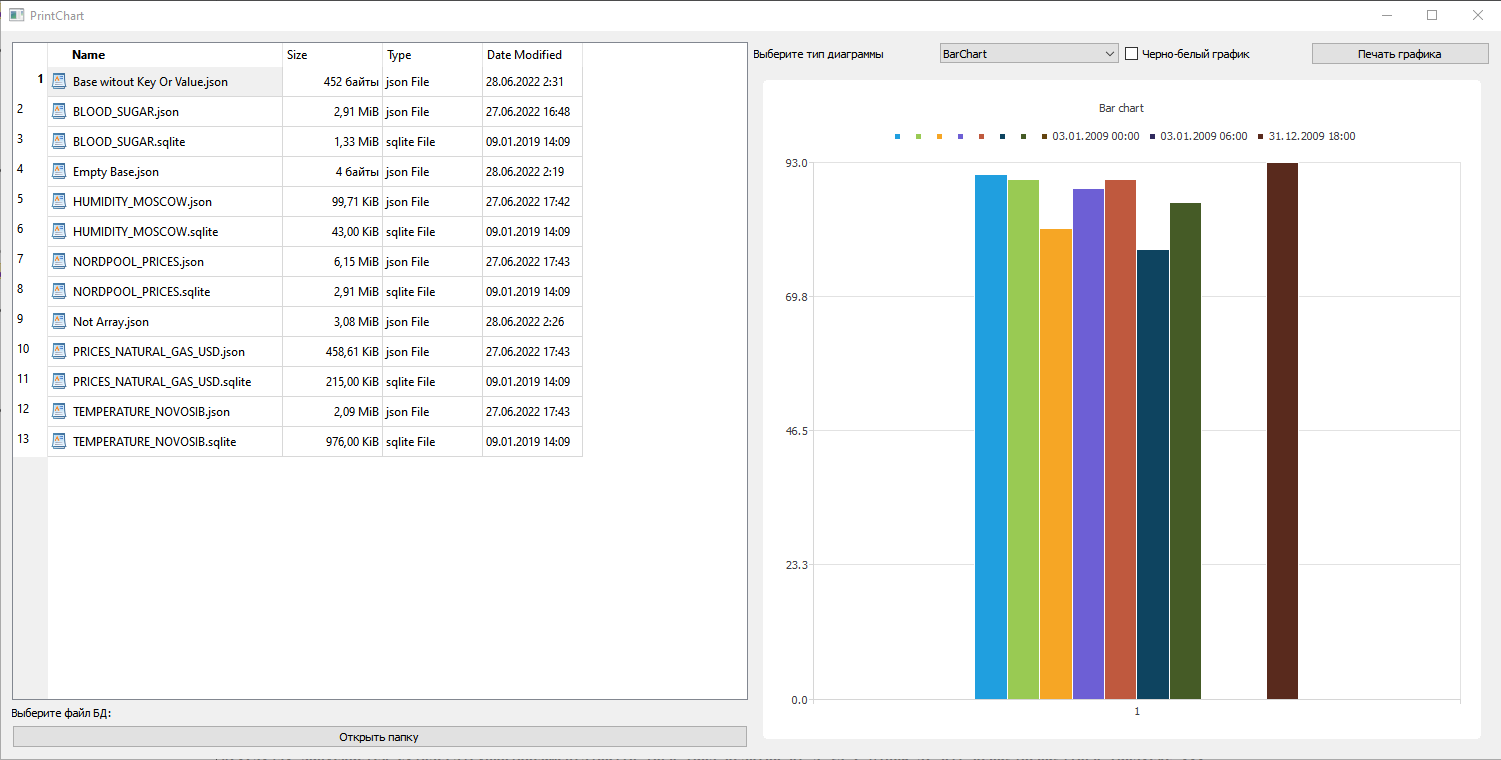
7. Если по каким-то причинам база не открылась, то появится окно ошибки, также показывающее путь базы, которая не откралась.



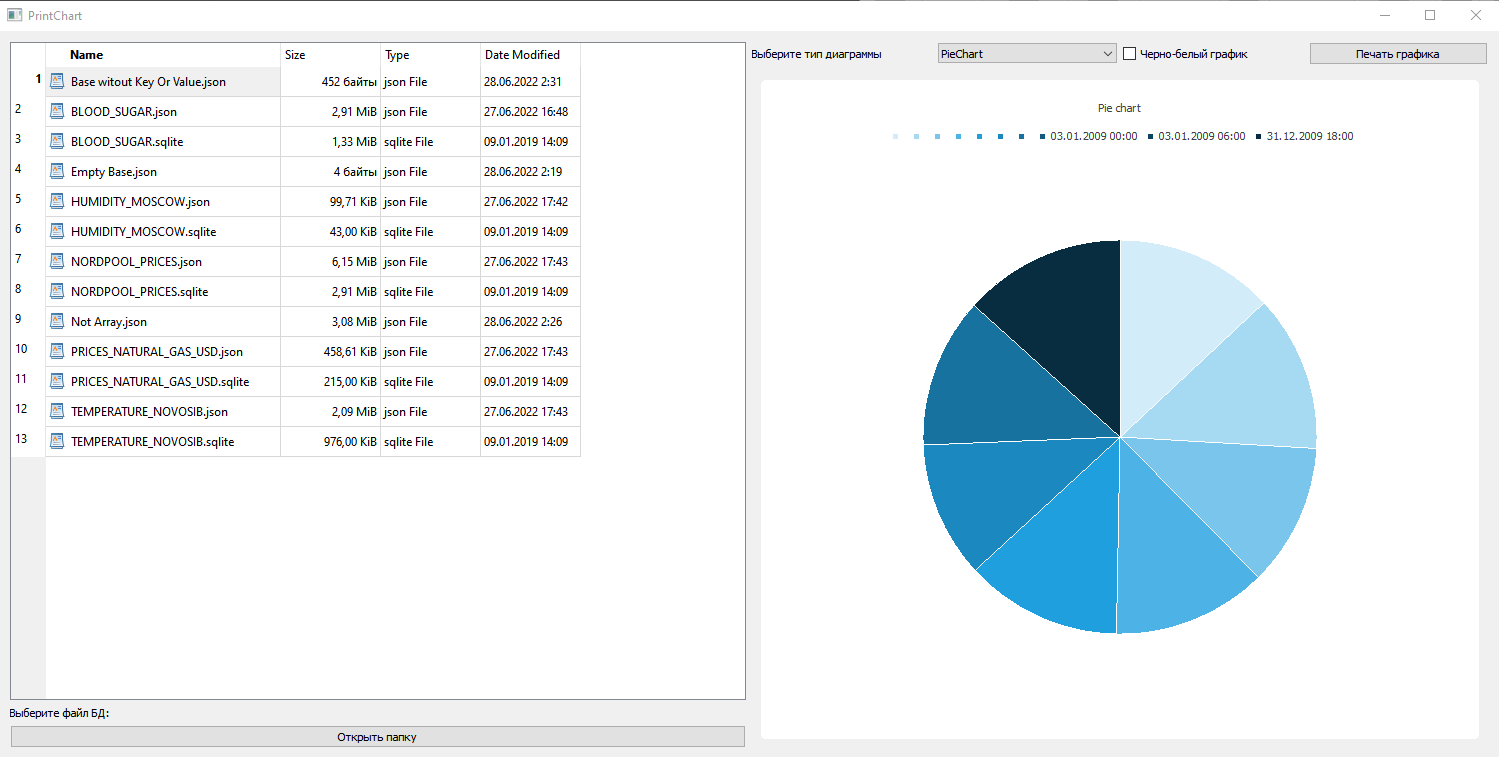
8. Если база формата “.json” реализована не как Массив. (Будет начинаться с { и заканчиваться }, вместо [ и ] соответственно, тогда программа выдаст ошибку.



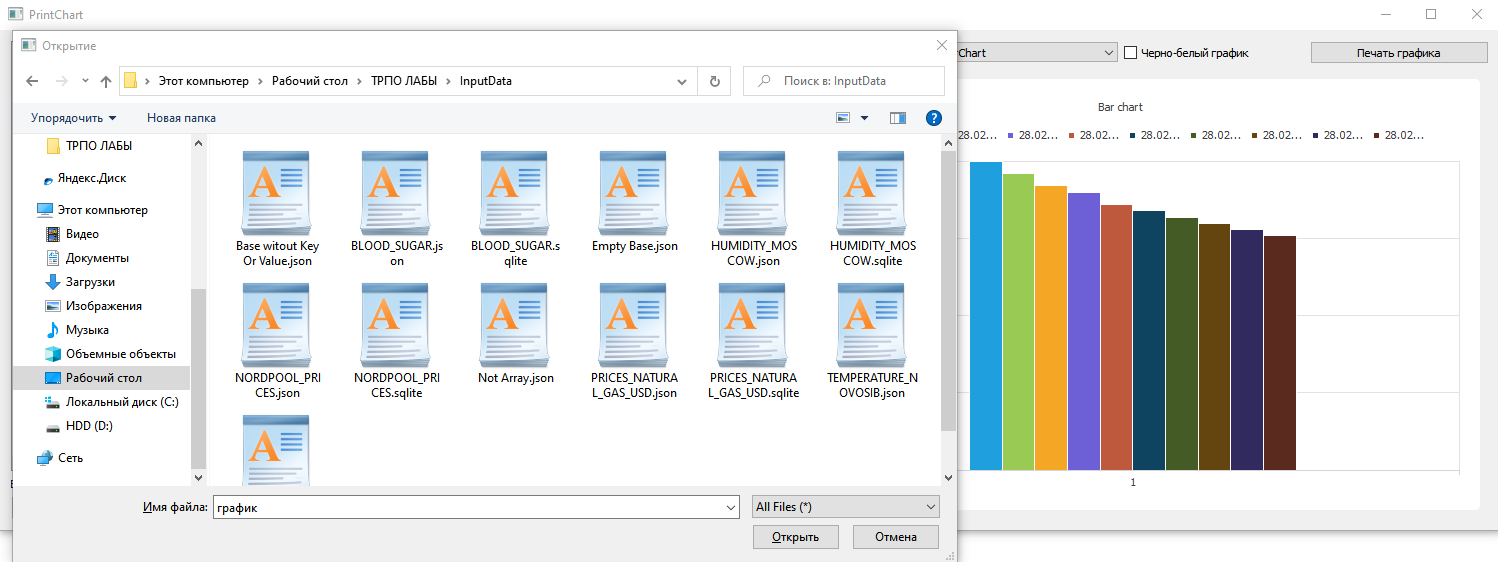
9. Если в базе отсутствуют какие-то данные (в нашем случае key или value), тогда программа выведет график вот так. Она пропустит в легенде несуществующие key и пропустит столбцы где пропущены value.



Тоже самое и на круговой диаграмме



10. Если мы правильно выбрали файл и у нас появился график, мы можем вывести его, нажав кнопку “Печать графика”. Появится окно сохранения, где нужно указать имя и путь для сохранения. (Указывать формат не нужно).



После чего файл сохранится в формате “.pdf”

