Акназаров Тимур Русланович, ИПМКН гр. 931921

Часть 1.

1.Постановка задачи

Необходимо разработать приложение печати графиков.
Исходные данные для печати соответствуют некоторому типу, который будет определяться 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 c++

Необходимо разобраться в предложенной реализации IOC контейнера.

Код сопроводить соответствующими объяснениями.

2. Предполагаемое решение

Интерфейс реализован следующим образом:

Была использована концепция MVC для общей работы с данными и их хранением, отображением. Данная концепция состоит из трех компонентов, а именно:

- Модель (Model), в которой непосредственно хранятся все данные приложения (в моем случае используется QFileSystemModel, которая позволяет получать необходимый путь к файлу.)
- Представление (View), которое отображает наши данные в необходимом виде на экране пользователя (в моем случае QTableView, с помощью QModelIndexList, полученных из модели, есть возможность получать доступ к данным).
- Контроллер (Controller), который отвечает за сам интерфейс, предоставляющий возможность пользователю взаимодействовать с приложением (В моем случае контроллер уже реализован в представлении)

С помощью QItemSelectionModel отслеживаются выбранные элементы, а также файлы представляются в виде таблицы.

Для отображение диаграмм был создан класс Charts, содержащий в себе элемент QChart, описывающий сами диаграммы. Также был создан элемент QVector <DataStorage>, который содержит в себе флаг, отвечающий за цвет диаграммы и уже считанные данные с файлов в необходимом виде.

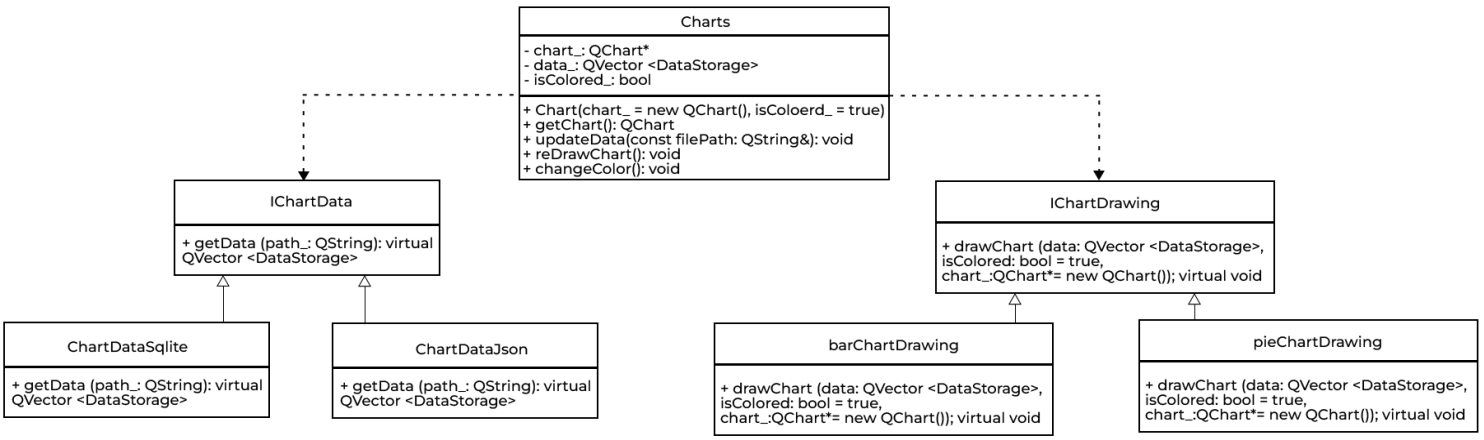
Помимо отрисовки графика имеется возможность выбирать его тип. Такая возможность реализована с помощью интерфейса IChartDrawing, который в свою очередь реализован с помощью IOC контейнера, используя инверсию зависимости. Иными словами, в интерфейсе есть виртуальная функция отрисовки диаграммы, у которой изначально не задан тип. Ниже в реализациях barChartDrawing и pieChartDrawing отрисовывается необходимый тип далее передаваемый в интерфейс программы. Таким образом, при необходимости, имеется возможность дописать дополнительный класс, реализующий другой тип данных, не взаимодействуя с основным интерфейсом.

Аналогичная методика была применена в чтении данных. Был реализован интерфейс IChartData (тоже на IOC контейнере, используя инверсию зависимости), от которого наследуются ChartDataSqlite и ChartDataJson, которые предоставляют возможность получать данные из баз в соответствующих форматах. Таким же образом, если появится необходимость реализовать чтение нового формата, у нас есть возможность справиться с этой задачей, не взаимодействуя с основным интерфейсом. Иными словами, у нас есть класс верхнего уровня, не зависящий от классов нижнего уровня (Инверсия зависимости).

Последнее, что понадобилось это QChartView, виджет, который помогает отображать диаграммы на экране пользователя.

С использованием вышеописанных элементов был воедино собран интерфейс, подключены необходимые кнопки и все взаимодействия с ними. Был описан выбор директорий с помощью QFileDialog (диалоговое окно выбора папки), а также вывод графика в pdf формате, с помощью QPdfWriter.

UML-диаграмма:



3. Коды программ

Файлы с кодом программы:

1. charts.h – взаимодействие с графиками
2. charts.cpp
3. data.h – взаимодействие с данными для отображения графика
4. data.cpp
5. ioccontainer.h – IOC контейнер
6. mainwindow.h - виджет главного окна
7. mainwindow.cpp

8. main.cpp

Находятся на репозитории github.com/timuraknazarov/trpo_lab_2

4. Инструкция пользователя

Интерфейс программы разделен на две части:

- Левая половина интерфейса это выбор файла для отображения графика.
- Правая половина интерфейса это отображение графика по выбранному файлу.

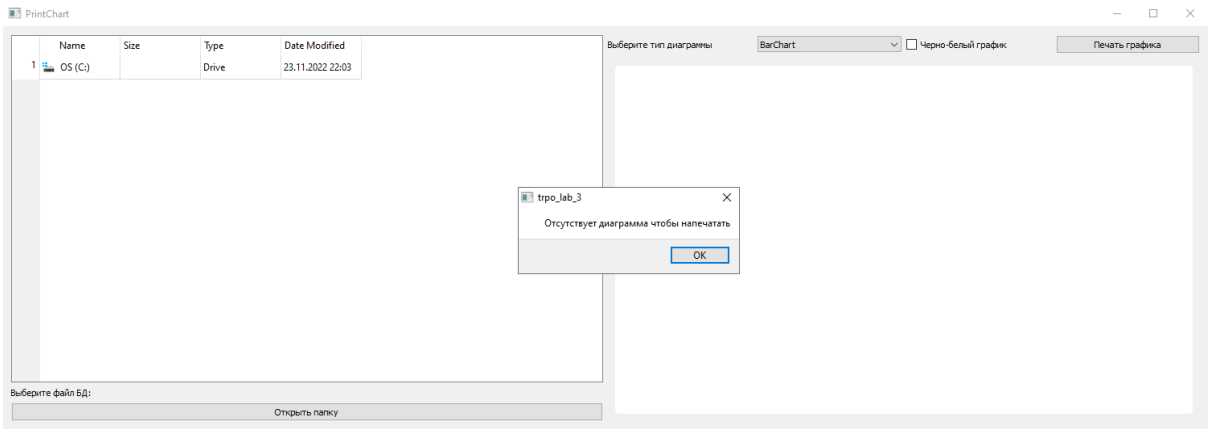
При запуске программы, правая часть интерфейса будет пустая, так как еще не выбран файл для отображения графике. В левой части будут файлы, находящиеся в корневой папке программы.

При помощи кнопки “открыть папку”, можно выбрать необходимую папку, в которой хранятся нужные базы данных. После выбора папки в левой части интерфейса появляются файлы, находящиеся в выбранной папке (а также их имя, размер, тип и дата изменения).

5. Тестирование

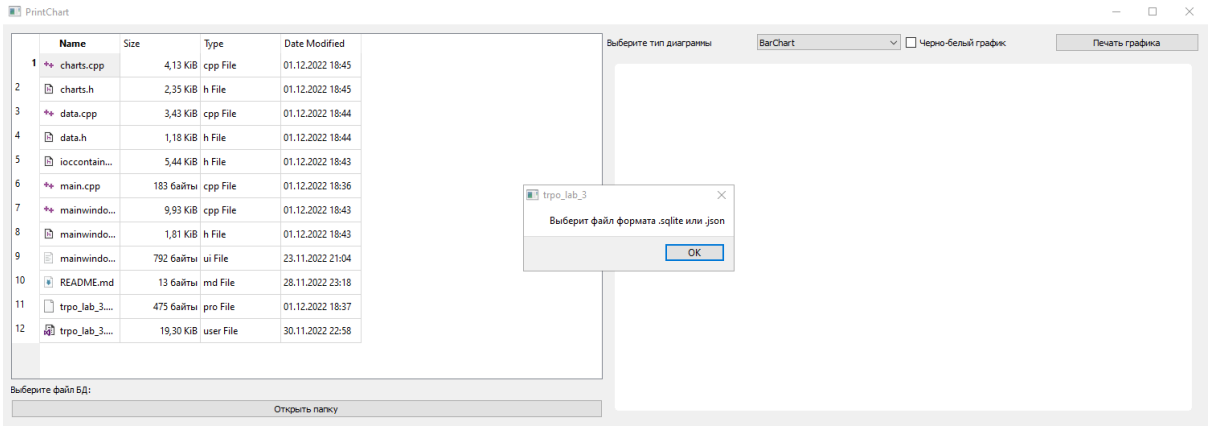
- **Тест 1:**

Если сразу после запуска программы нажать “Печать графика”, появляется окно ошибки, так как еще не выбран файл для вывода графика.



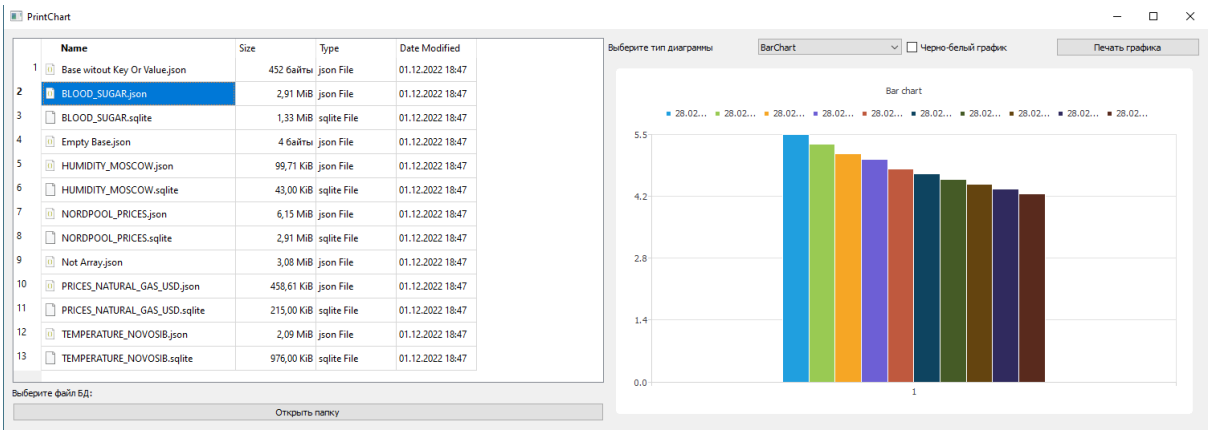
- **Тест 2:**

Если выбрать файл какого либо формата кроме .sqlite или .json, появляется окно ошибки, так как реализовано взаимодействие только с такими типами файлов.



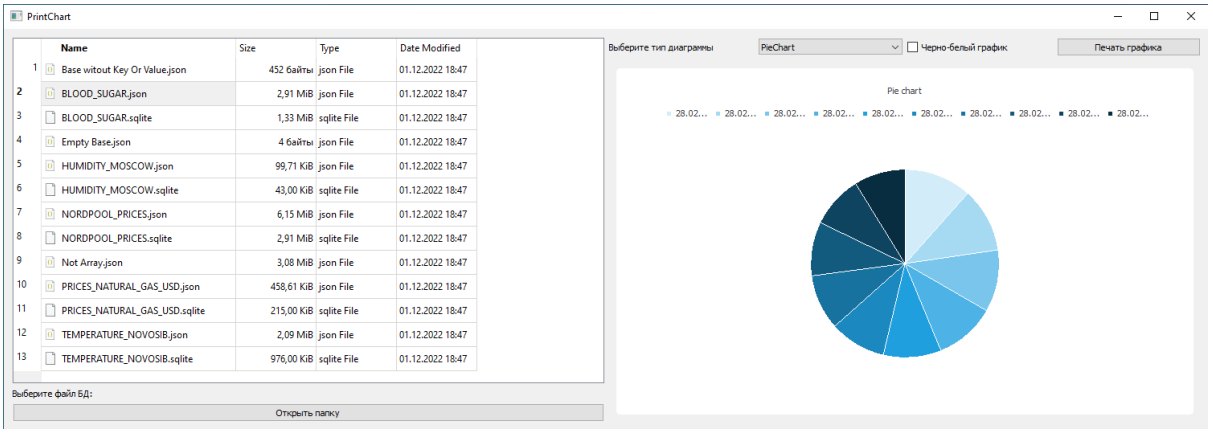
- **Тест 3:**

Если выбрать подходящий файл, то изначально выведется вертикальный график, показывающий первые 10 данных из базы.



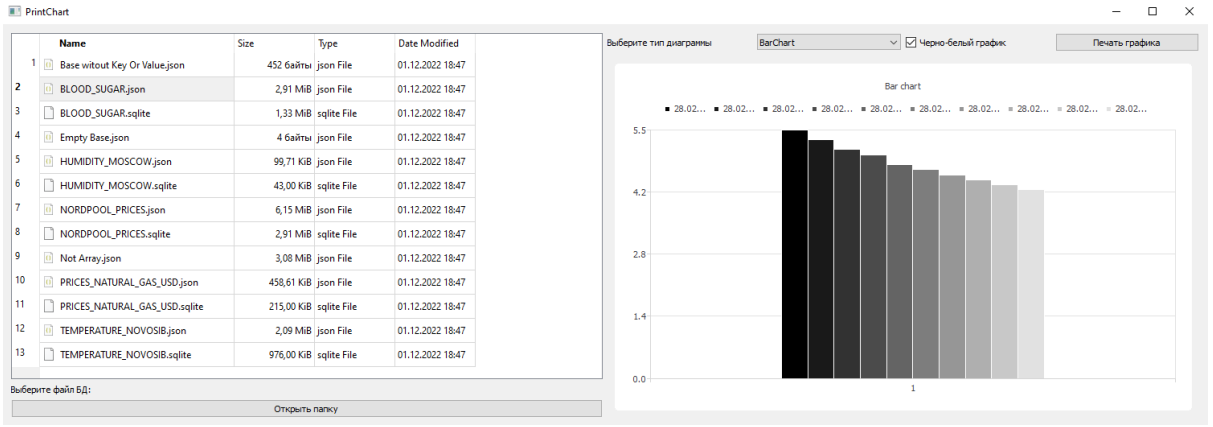
- **Тест 4:**

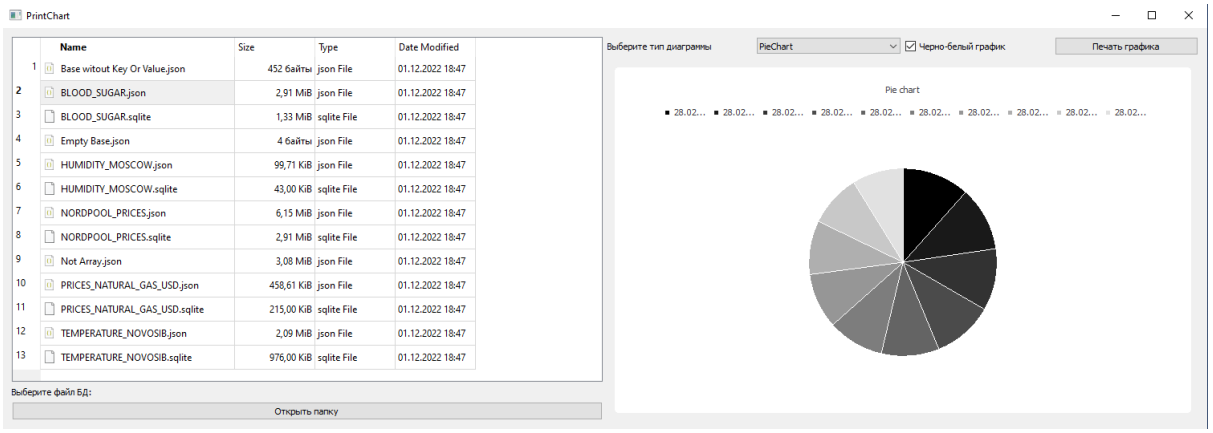
После выбора подходящего файла можно выбрать тип графика из выпадающего меню “Выберите тип диаграммы”. Если выбрать тип “PieChart”, выведется круговой график, показывающий первые 10 данных из базы.



- **Тест 5:**

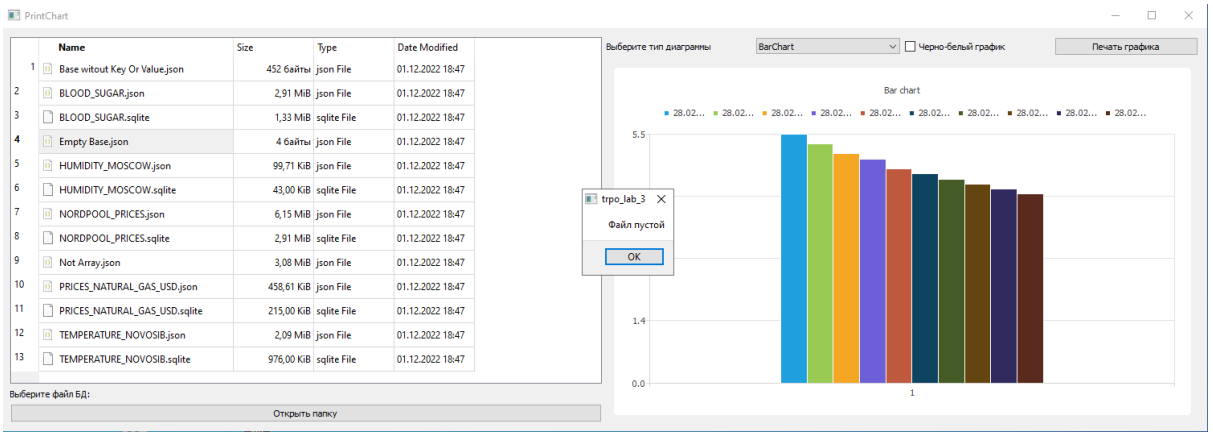
Если график уже отрисован (вне зависимости от его типа), можно поставить флажок “Черно-белый график”, в таком случае график выведется в черно-белых цветах.





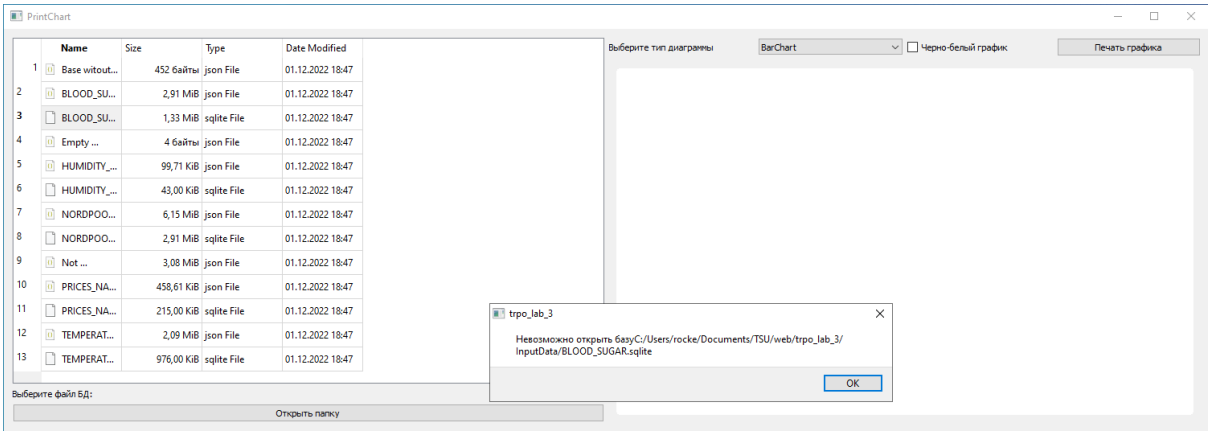
- **Тест 6:**

Если файл формата .sqlite или .json оформлен как база, но не содержит в себе элементов (пустая база), то появится окно ошибки, сообщающее что файл пуст.



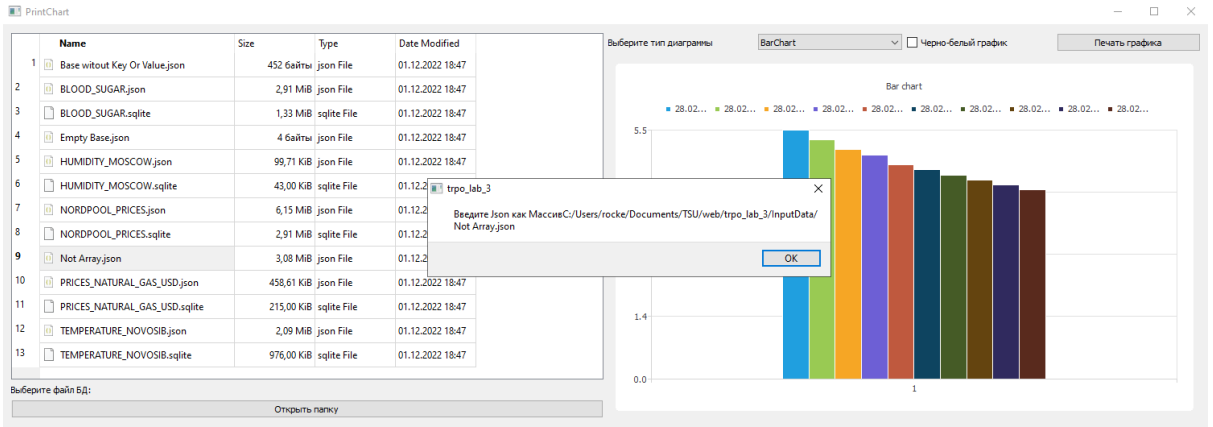
- **Тест 7:**

Если по каким-либо причинам база не открылась, появляется окно ошибки, показывающее путь до файла, который не открылся.



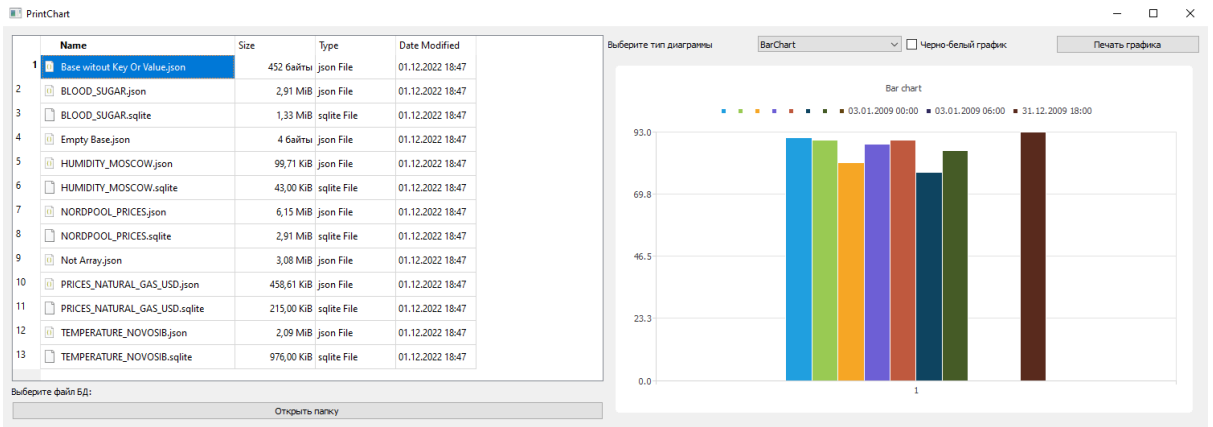
- **Тест 8:**

Если база в формате .json реализована не как массив, появится соответствующее окно ошибки.



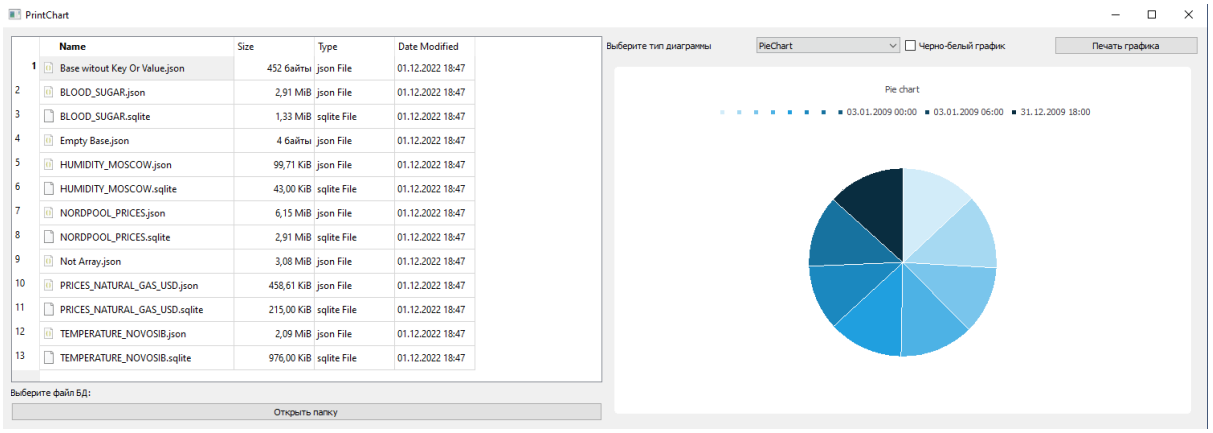
- **Тест 9:**

Если в базе отсутствуют какие-либо данные (в нашем случае key или value), то программа выведет график следующим образом:



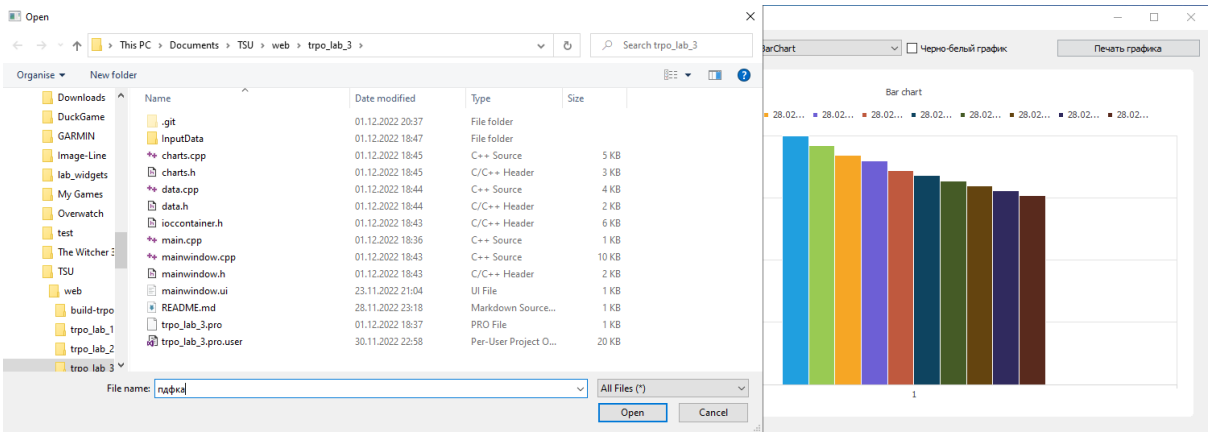
Программа пропускает в легенде несуществующие key, а также, пропускает столбцы с пропущенными value.

Аналогично для круговой диаграммы:



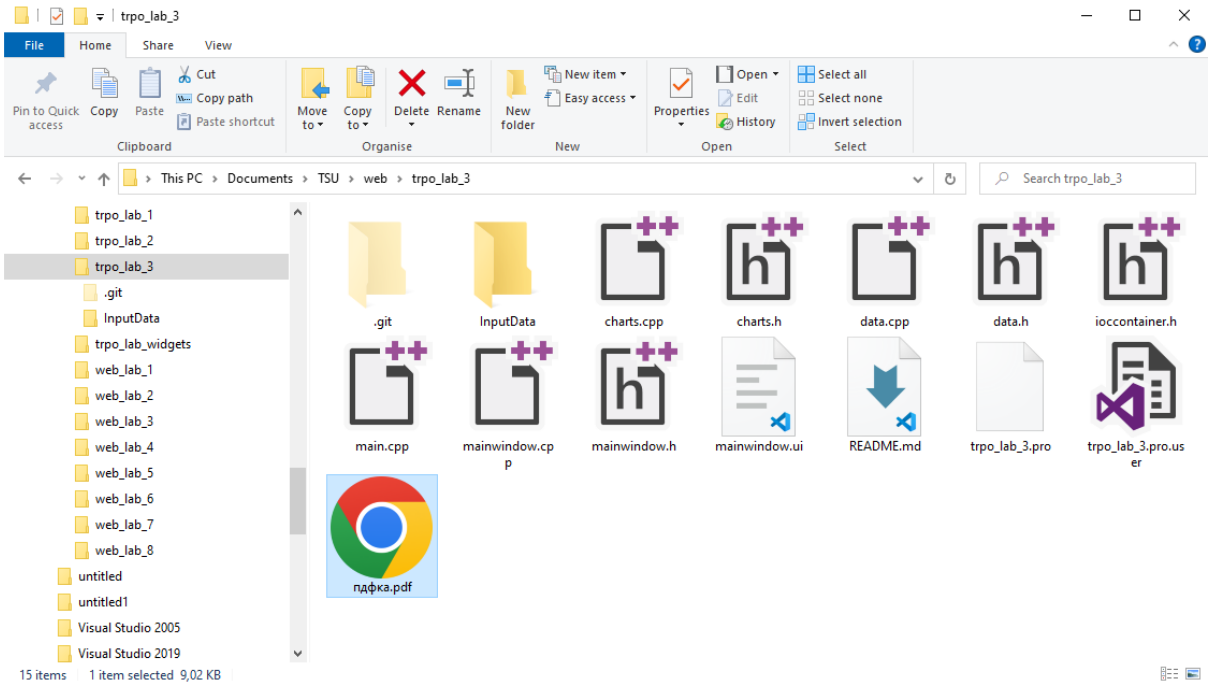
- **Тест 10:**

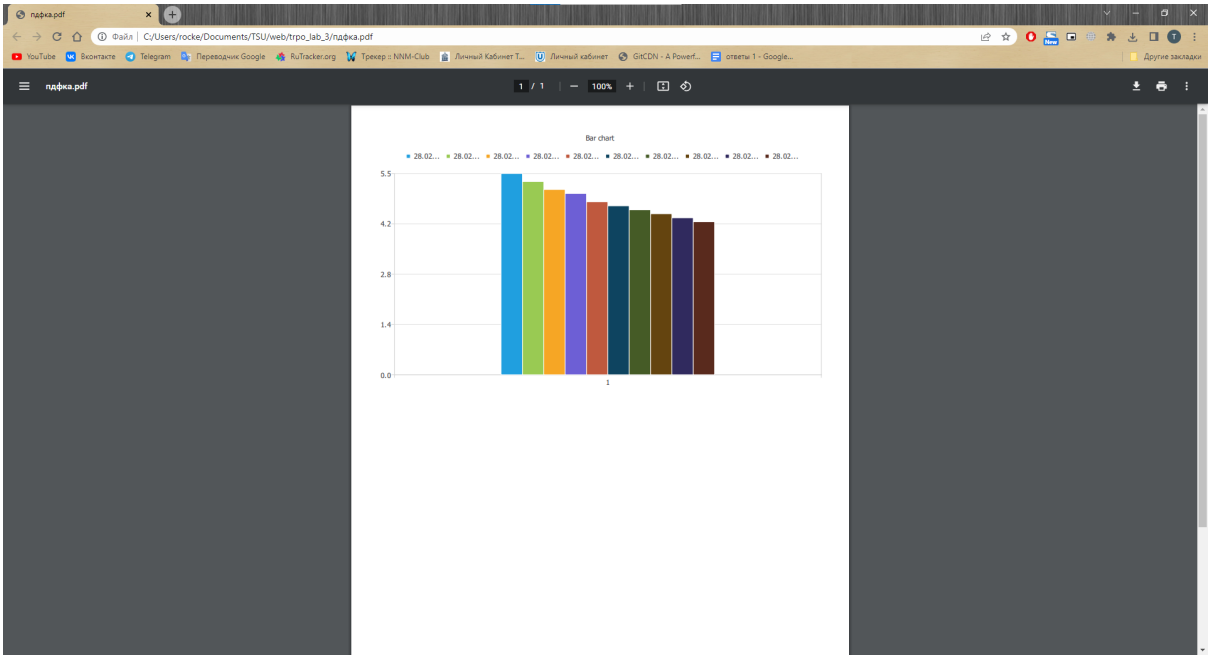
Если мы выбрали файл, и у нас появился график, у нас есть возможность вывести его, нажав на кнопку “Печать графика”. Появится окно для сохранения, где необходимо указать имя файла, и путь для сохранения.



Указывать формат файла не нужно.

После чего по указанному пути создается файл с выбранным именем формате .pdf


trpo_lab_3
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organise
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Documents › TSU › web › trpo_lab_3
Search trpo_lab_3
trpo_lab_1
trpo_lab_2
trpo_lab_3
.git
InputData
trpo_lab_widgets
web_lab_1
web_lab_2
web_lab_3
web_lab_4
web_lab_5
web_lab_6
web_lab_7
web_lab_8
untitled
untitled1
Visual Studio 2005
Visual Studio 2019
.git
InputData
charts.cpp
charts.h
data.cpp
data.h
ioccontainer.h
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
README.md
trpo_lab_3.pro
trpo_lab_3.pro.user
пдфка.pdf
15 items
1 item selected 9,02 KB


пдфка.pdf
C:/Users/rocke/Documents/TSU/web/trpo_lab_3/пдфка.pdf
пдфка.pdf
1 / 1
100%
Bar chart
5.5
4.2
2.8
1.4
0.0
1