Файл ***main.py***отвечает за запуск программы, а также за импортирование необходимых библиотек.

Файл ***new\_note.py*** отвечает за присваивание каждому виджету вспомогательного окна определённых функций.

Файл ***my\_date.py***содержит описание даты (входных данных) в удобном для работы с данными формате.

Файл ***my\_time.py*** содержит описание времени (входных данных) в удобном для работы с данными формате.

Файл ***note.py***содержит описание объекта, а именно – описания задачи. Описание представлено классом ***Note(object).***

Файл ***note\_loader.py***отвечает за загрузку данных из базы данных.

Файл ***note\_comparator.py***отвечает за проверку корректно заданного времени каждой задачи, т.е. осуществляет проверку того, нет ли перекрытия по времени.

Файлы ***gui.py, newNote.py*** созданы автоматическим генератором из xml разметки. Файл ***gui.py*** описывает отрисовку главного (родительского) окна, а ***newNote.py*** – окна, отвечающего за добавление задачи.

Файл ***main.py***

Библиотеки

Модуль ***json*** выполняет всю работу по преобразованию данных из формата Json в значения Python и обратно.

В модуле ***sys*** содержатся функции и константы для взаимодействия с интерпретатором Python.

Модуль ***traceback*** предоставляет стандартный интерфейс для извлечения, форматирования и печати трассировок стека программ Python.

Controllers.new\_note

utility.my\_date

gui.gui

Модуль ***QtCore*** содержит ядро с неграфической функциональностью. Этот модуль используется для работы с временем, файлами, папками, различными типами файлов, потоками, адресами URL, MIME-типами и процессами.

***QtGui*** содержит классы для интеграции систем окон, обработки событий, 2D-графики, базовой обработки изображений, шрифтов и текста.

Модуль ***QtWidgets*** содержит классы, которые обеспечивают набор UI-элементов для создания классических пользовательских интерфейсов. Виджет ***Qwidget*** является базовым классом для всех объектов интерфейса пользователя в PyQt5. ***QcheckBox*** – это виджет, который имеет два состояния: вкл. И выкл. Это квадратик с меткой. Как правило, чекбоксы используют, чтобы представить функции приложения, которые могут быть включены или выключены. **Qapplication** – приложение, с которого начинается построение интерфейса.

Qt

MyTime

Note

NoteComparator

NoteLoader

Файл ***my\_date.py***

Библиотеки и виджеты

Модуль ***datetime*** предоставляет классы для обработки времени и даты разными способами.

***QCalendarWidget*** предоставляет виджет помесячного календаря. Он позволяет пользователю выбирать дату простым и интуитивным путём.

Класс ***MyDate***

Методы данного класса позволяют нам описать входные данные в удобном для работы с ними формате.

С помощью метода ***\_\_init\_\_()*** присвоим значения ***None*** атрибутам класса по умолчанию. Если в конструктор передан параметр ***date***, не являющийся ***None***, то ***self.set\_qt\_date(date)*** устанавливает дату в зависимости от того, что было передано в параметре ***date,*** т.е. позволяет разделить поступающие данные формата ***QCalendar*** на составляющие.

Метод ***set\_date()*** принимает 3 аргумента (день, месяц, год) и записывает их в соответствующие поля данного класса.

Метод ***set\_qt\_date()*** позволяет привести данные к строке. Это осуществляется с помощью встроенной функции ***date.toString().*** Далее создаётся массив ***params***, элементами которого являются день недели ***(day\_of\_week\_name)***, месяц ***(month)***, день ***(day)***, год ***(year).*** С помощью встроенного модуля ***datetime.date()*** мы получаем номер недели

Метод  ***\_\_to\_digit()*** позволяет более быстро осуществить сравнение дат.

Метод ***\_\_eq\_\_()*** переопределяет оператор ***“=”***.

Метод ***\_\_ lt\_\_()*** переопределяет оператор ***“<“.***

Метод ***\_\_ le\_\_()*** переопределяет оператор ***“<=”***.

Метод ***\_\_ ne\_\_()*** переопределяет оператор, обратный оператору ***“=”***.

Метод ***\_\_ gt\_\_()*** переопределяет оператор ***“>”***.

Метод ***\_\_ ge\_\_()*** переопределяет оператор ***“>=”***.

Метод ***\_\_str\_\_()*** возвращает дату в формате “день.месяц”.

Методы ***get\_number\_of\_week(), get\_number\_of\_day(), get\_day(), get\_month\_number(), get\_year(), get\_month\_name(), get\_name\_of\_day\_in\_week()*** позволяют получить соответствующие значения для более удобной работы с датой.

Файл ***my\_time.py***

Класс ***MyTime***

Методы данного класса позволяют нам описать входные данные в удобном для работы с ними формате.

С помощью метода ***\_\_init\_\_()*** присвоим значения ***None*** атрибутам класса по умолчанию. то с помощью ***s.split(':')*** разделим её на два числа по сепаратору ***“:”*** и присвоим атрибутам класса ***\_\_minute*** и ***\_\_hour*** соответствующие значения.

Метод ***set\_from\_string()*** осуществляет проверку на корректность заданной строки.

Метод ***\_\_to\_minutes()*** позволяет переводить заданное время в минуты.

Методы ***hour()*** и ***minute()*** позволяют получить соответствующие значения для более удобной работы.

Метод ***\_\_eq\_\_()*** переопределяет оператор ***“=”***.

Метод ***\_\_ lt\_\_()*** переопределяет оператор ***“<“.***

Метод ***\_\_ le\_\_()*** переопределяет оператор ***“<=”***.

Метод ***\_\_ ne\_\_()*** переопределяет оператор, обратный оператору ***“=”***.

Метод ***\_\_ gt\_\_()*** переопределяет оператор ***“>”***.

Метод ***\_\_ ge\_\_()*** переопределяет оператор ***“>=”***.

Метод ***\_\_str\_\_()*** возвращает дату в формате “часы:минуты”.

Зачем переопределять операторы и где это используется

Файл ***note\_loader.py***

Библиотеки и виджеты

utility.my\_date

mysql.connector

Класс ***NoteLoader***

Данный класс позволяет загружать заметки из базы данных.

Метод ***\_\_init\_\_()*** позволяет создать соединение с базой данных с помощью дополнительных параметров соединения. Также здесь создаётся ***cursor***, являющийся специальным объектом, который делает запросы и получает их результаты.

Метод ***upload()*** получает в качестве параметра заметку (***new\_note***). Далее формируется строка ***request***, в которой лежит текст запроса. Данный запрос добавляет в таблицу ***notes*** в базе данных ***organiser*** значения ***day, month, year, start\_time, end\_time, title, description,*** ***week\_number, day\_of\_week***. Т.к. вначале значения этих полей не известны, то они прописываются как ***%s***, строка. Таким образом, в базе данных всё хранится в виде строк. ***cursor.execute():*** в метод ***execute()*** передаётся строка ***request*** и результат метода ***to\_mysql\_list()***, выполненного на объекте ***new\_note***. Таким образом, метод ***execute()*** поставит элементы из списка, который возвращает функция ***to\_mysql\_list()***, на место ***%s.*** В следующей строке кода регистрируется изменение в базе данных.

Метод ***load\_notes\_by\_week\_number()*** получает на вход дату типа MyDate и возвращет двумерный массив, который представляет собой 7 массивов для каждого дня недели, в каждом из которых содержится массив заметок для каждого дня. Формируется строка ***request***, представляющая запрос на заметки с указанным годом и указанным номером недели. Надо как-то описать цикл…….

Надо как-то описать два цикла…….

Файл ***note.py***