­­­­­ Государственное бюджетное общеобразовательное учреждение города Москвы "Школа № 1518"

Проектная работа по теме:

**«Мой мини фоторедактор»**

*Выполнила ученица:*

*9 «Д» класса*

*Душейко Дарья Васильевна*

*e-mail: osk1609@gmail.com*

Город Москва,

2020 г.

**Введение**

***Актуальность темы:***

В современном мире социальные сети играют очень важную роль в нашей жизни. Одним из самых популярных сетей является Instagram, где люди обмениваются различными фотографиями и видео. Для того, чтобы привлечь внимание других пользователей они делают контент более ярким и красочным. В этом людям помогают фоторедакторы.

***Цель работы:***

Создать свой личный фоторедактор собственными руками, с помощью которого можно было бы не только изменять фотографии и картинки, но и делать их интересными для того, чтобы делиться с другими людьми.

***Задачи практической работы:***

* создать программу с ярким, интересным и красивым дизайном;
* создать программу с множеством различных функций;
* создать программу с удобным и понятным интерфейсом в использовании для любого возраста;
* применить в программе как можно больше технологий;
* изучить и понять как работают различные функции языка питона;

Объектом практической работы является создание программы для редактирования фотографий.

***Основными методами*** практической работы является программирование.

Работа состоит из введения, двух глав и заключения.

**Глава 1. Суть проекта.**

* 1. **Фильтры.**

Для обозначения месторасположения фильтров был создан **label\_2,** которому присвоили название: «FILLTERS». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я попыталась реализовать как можно больше различных фильтров. Общее количество – 23, + оригинальный.

1. **Сold**

Фильтр оставляет на картинке только холодные цвета (зеленый, синий, голубой, фиолетовый). За этот фильтр отвечает функция **run2(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение красного цвета значений RGB их цвета на ноль.

Для осуществления этого фильтра также была создана кнопка **pushButton\_2,** которой я присвоила зелёный цвет.

1. **Lime**

Фильтр изменяет цвета на картинке, преобразуя их в лаймовые, фиолетовые и изумрудные. За этот фильтр отвечает функция **run3(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем местами значение красного и зелёного цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_3,** которой я присвоила ярко-зелёный цвет.

1. **Light**

Фильтр изменяет цвета на картинке, преобразуя их в более светлые. За этот фильтр отвечает функция **run4(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение значений RGB их цвета более белое (прибавляем 100).

Для осуществления этого фильтра также была создана кнопка **pushButton\_4,** которой я присвоила кремовый цвет.

1. **Dark**

Фильтр изменяет цвета на картинке, преобразуя их в более тёмный. За этот фильтр отвечает функция **run5(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение значений RGB их цвета более чёрное (вычитаем 100).

Для осуществления этого фильтра также была создана кнопка **pushButton\_5,** которой я присвоила ярко-бордовый цвет.

1. **Green -> Pink**

Фильтр изменяет цвета на картинке, изменяя зелёный цвет в розовый. За этот фильтр отвечает функция **run6(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы проверяем, является ли цвет зелёным. Если является, то мы меняем цвет пикселя на розовый.

Для осуществления этого фильтра также была создана кнопка **pushButton\_6,** которой я присвоила ярко-розовый цвет.

1. **Black and Wight**

Фильтр изменяет картинку с цветной на черно-белую. За этот фильтр отвечает функция **run7(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем их цвета на сумму значений RGB делённую на 3.

Для осуществления этого фильтра также была создана кнопка **pushButton\_7,** которой я присвоила белый цвет.

1. **Orange**

Фильтр изменяет цвета на картинке, преобразуя их в оранжевый, фиолетовые и синий. За этот фильтр отвечает функция **run1(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем местами значение красного и синего цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton,** которой я присвоила ярко-фиолетовый цвет.

1. **Soft green**

Фильтр изменяет цвета на картинке, преобразуя их в различные оттенки зеленый. За этот фильтр отвечает функция **run11(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем присваиваем значениям красного и синего цветов значений RGB 130.

Для осуществления этого фильтра также была создана кнопка **pushButton\_11,** которой я присвоила ярко-изумрудный цвет.

1. **Coral**

Фильтр изменяет цвета на картинке, преобразуя их в различные оттенки красного. За этот фильтр отвечает функция **run11(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем присваиваем значениям зелёного и синего цветов значений RGB 130.

Для осуществления этого фильтра также была создана кнопка **pushButton\_16,** которой я присвоила ярко-коралловый цвет.

1. **Ocher**

Фильтр изменяет цвета на картинке, преобразуя их в охровые, фиолетовые оттенки. За этот фильтр отвечает функция **run17(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем присваиваем значениям красного на 180 и зелёного на 160 цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_17,** которой я присвоила охровый цвет.

1. **Negative**

Фильтр изменяет цвета на картинке, преобразуя их в полностью противоположные им. За этот фильтр отвечает функция **run12(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем присваиваем значениям нового красного на 255 минус значение старого красного, нового зелёного на 255 минус значение старого зелёного и нового синего на 255 минус значение старого синего цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_12,** которой я присвоила голубо-фиолетовый цвет.

1. **Flower**

Фильтр изменяет цвета на картинке, преобразуя их в изумрудные, синие и красные. За этот фильтр отвечает функция **run10(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем местами значение зелёного и синего цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_10,** которой я присвоила сиреневый цвет.

1. **Witch**

Фильтр изменяет цвета на картинке, преобразуя их в салатовый, голубой цвета. За этот фильтр отвечает функция **run13(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем местами все три значение зелёного, красного и синего цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_13,** которой я присвоила болотный цвет.

1. **Watermelon**

Фильтр изменяет цвета на картинке, преобразуя их в розовый, изумрудный, жёлтый цвета. За этот фильтр отвечает функция **run14(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы меняем местами все три значение зелёного, красного и синего цветов значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_14,** которой я присвоила красно-оранжевый цвет.

1. **Blueberry**

Фильтр изменяет цвета на картинке, преобразуя их в синий, розовый, фиолетовый, сиреневый, голубой цвета. За этот фильтр отвечает функция **run13(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение зелёного цвета значений RGB на ноль.

Для осуществления этого фильтра также была создана кнопка **pushButton\_18,** которой я присвоила светло-фиолетовый цвет.

1. **Rowan**

Фильтр изменяет цвета на картинке, преобразуя их в зелёный, красный, жёлтый цвета. За этот фильтр отвечает функция **run15(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение синего цвета значений RGB на ноль.

Для осуществления этого фильтра также была создана кнопка **pushButton\_15,** которой я присвоила оранжевый цвет.

1. **Mountain**

Фильтр изменяет цвета на картинке, преобразуя их в голубой, розовый, жёлтый цвета. За этот фильтр отвечает функция **run23(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение зелёного цвета значений RGB на 180.

Для осуществления этого фильтра также была создана кнопка **pushButton\_23,** которой я присвоила бирюзовый цвет.

1. **Melon**

Фильтр изменяет цвета на картинке, преобразуя их в персиковый, болотный, темно изумрудный цвета. За этот фильтр отвечает функция **run22(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение зелёного на 180, синего на 10 и значение нового красного меняем на значение 30 плюс старого красного цвета значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_22,** которой я присвоила изумрудный цвет.

1. **Yellow**

Фильтр изменяет цвета на картинке, преобразуя их в различные оттенки жёлтого. За этот фильтр отвечает функция **run21(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение зелёного на 246 и значение красного на значение 246 значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_21,** которой я присвоила ярко-жёлтого цвет.

1. **Sky**

Фильтр изменяет цвета на картинке, преобразуя их в фиолетовый, сиреневый, красный цвета. За этот фильтр отвечает функция **run24(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение красного на 180 значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_24,** которой я присвоила темно-бежевый цвет.

1. **Night**

Фильтр изменяет цвета на картинке, преобразуя их в жёлтый, чёрный, синий и серый цвета. За этот фильтр отвечает функция **run20(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение красного и зелёного на значение синего, а значение синего цвета заменяем на значение зелёного значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_20,** которой я присвоила ярко-синего цвет.

1. **Sweet**

Фильтр изменяет цвета на картинке, преобразуя их в белого, розового, голубого цвета. За этот фильтр отвечает функция **run19(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение нового зелёного цвета заменяем на значение старого зелёного плюс 50, а значение нового красного цвета заменяем на значение старого красного плюс 50 значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_19,** которой я присвоила светло-розового цвет.

1. **Red**

Фильтр изменяет цвета на картинке, преобразуя их в красного, розового, голубого цвета. За этот фильтр отвечает функция **run25(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение зелёного на значение синего, а значение нового красного цвета заменяем на значение старого красного плюс 30 значений RGB.

Для осуществления этого фильтра также была создана кнопка **pushButton\_25,** которой я присвоила ярко-красного цвет.

1. **Original**

Открывает оригинал картинки отвечает функция **run8(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение.

Для осуществления была создана кнопка **pushButton\_8,** которой я присвоила ярко-красного цвет.

**1.2. Поворот изображения.**

Для обозначения месторасположения фильтров был создан **label\_4,** которому присвоили название: «TURNS». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я реализовала три поворота.

1. **900 0**

Функция поворачивает картинку на 90 градусов в право. За это отвечает функция **changeTitle(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Также используется класс **QcheckBox** библиотеки **PyQt5.** Это позволяет нам узнать, хочет пользователь перевернуть картинку или нет (если галочка стоит, то картинка переворачивается, если не стоит, то не переворачивается).

Для осуществления этой функции была создана кнопка **checkbox**.

1. **180 0**

Функция поворачивает картинку на 180 градусов в право. За это отвечает функция **changeTitle\_2(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Также используется класс **QcheckBox** библиотеки **PyQt5.** Это позволяет нам узнать, хочет пользователь перевернуть картинку или нет (если галочка стоит, то картинка переворачивается, если не стоит, то не переворачивается).

Для осуществления этой функции была создана кнопка **checkbox\_2**.

1. **270 0**

Функция поворачивает картинку на 270 градусов в право. За это отвечает функция **changeTitle\_3(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Также используется класс **QcheckBox** библиотеки **PyQt5.** Это позволяет нам узнать, хочет пользователь перевернуть картинку или нет (если галочка стоит, то картинка переворачивается, если не стоит, то не переворачивается).

Для осуществления этой функции была создана кнопка **checkbox\_3**.

* 1. **Рамки для изображения.**

Для обозначения месторасположения рамок был создан **label\_13,** которому присвоили название: «PHOTO FRAMES». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я реализовала два вида рамок, но особенность этой функции является заключается в том, что используя две рамки одновременно, мы получаем рамки со всех сторон.

1. **RIGHT and LEFT**

Данная рамка добавляет белые полосы с правой и левой сторон. За эту функцию отвечает функция **run26(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение крайних 30 столбцов на белый цвет.

Для осуществления этого фильтра также была создана кнопка **pushButton\_26,** которой я присвоила голубой цвет.

1. **ABOVE and BOTTOM**

Данная рамка добавляет белые полосы снизу и сверху картинки. За эту функцию отвечает функция **run27(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение крайних 30 колонок на белый цвет.

Для осуществления этого фильтра также была создана кнопка **pushButton\_27,** которой я присвоила темно-голубой цвет.

**1.4. Настройка прозрачности изображения.**

Для обозначения места, где можно изменить прозрачность был создан **label\_17,** которому присвоили название: «TRANSPARENCY». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я соединила изменение значение прозрачности картинки с изменением положения бегунка на слайдере. Совмещение делалось с помощью класса **QVBoxLayout** (названный **vbox**)библиотеки **PyQt5.QtWidgets.** Слайдер создавался с помощью класса **QSlider** той же библиотеки. Также был создан **lcd** класса **QLCDNumber** библиотеки **PyQt5.QtWidgets,** который отображал значение прозрачности в процентах. Фоновый цвет является светло-зелёным.

За эту функцию отвечает функция **valueChanged(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Затем, проходя по каждому пикселю с помощью цикла **for**, мы изменяем значение их цветов по формуле NEWCOLOR = OLDCOLOR \* (1 - prosent) + 211 \* prosent.

**1.5. Настройка размытия изображения.**

Для обозначения места, где можно изменить размытость был создан **label\_6,** которому присвоили название: «BLUR». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я соединила изменение значение размытости картинки с изменением положения бегунка на слайдере. Совмещение делалось с помощью класса **QVBoxLayout** (названный **vbox2**)библиотеки **PyQt5.QtWidgets.** Слайдер создавался с помощью класса **QSlider** той же библиотеки. Также был создан **lcd\_2** класса **QLCDNumber** библиотеки **PyQt5.QtWidgets,** который отображал значение размытости в процентах. Фоновый цвет является ярко-красный.

За эту функцию отвечает функция **valueChanged\_2(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Также используется инструмент **ImageFilter** той же библиотеки, который изменял размытие.

**1.6. Настройка количества цветов на изображении.**

Для обозначения места, где можно изменить размытость был создан **label\_9,** которому присвоили название: «NUMBER OF COLORS». Фоновый цвет меняется в зависимости от выбора пользователя с помощью класса **QColorDialog** библиотеки **PyQt5.QtWidgets.**

В своём проекте я создала отдельное поле для того, чтобы пользователей мог вписать количество цветов, которое он хочет оставить на картинке.библиотеки **PyQt5.QtWidgets.** Слайдер создавался с помощью класса **QSlider** той же библиотеки. Также был создан **lcd\_2** класса **QLCDNumber** библиотеки **PyQt5.QtWidgets,** который отображал значение размытости в процентах. Фоновый цвет является ярко красный.

За эту функцию отвечает функция **valueChanged\_2(),** где используется инструмент **Image** библиотеки **PIL**, с помощью которого мы загружаем изображение. Также используется инструмент **ImageFilter** той же библиотеки, который изменял размытие.

**Глава 2. Особенности проекта.**

Главной особенность моей программы является то, что пользователь может выбрать и отредактировать любую фотографию, которая есть на устройстве. Также любую изменённую фотографию можно сохранить c помощью специальной кнопки.

Для придания фонового цвета проекту, был создан **label\_9,** которому я присвоила светло-голубой цвет.

Также для улучшения дизайна программы были созданы **label\_3 и label\_5,** которые постоянно меняют цвета, в зависимости от цвета последнего пикселя картинки.

Загрузка изображения в **label** осуществлялся с помощью класса **QPixmap** библиотеки **PyQt5.QtWidgets**.

В данной программе используются такие модули, библиотеки, инструменты и классы, как: модуль **sys;** библиотека **PyQt5** с модулем **uic** и классами **QApplication, QMainWindow, QWidget, QPushButton, QPixmap, QLabel, QMainWindow, QColorDialog, QLCDNumber, QSlider** и **QVBoxLayout, Qt;** библиотека **PIL** синструментами **Image, ImageFilter** и **ImageDraw**

**Вывод.**

Я создала свой личный редактор фотографий, который изменяет картинки и фотографии. Он имеет яркий, интересный дизайн. Программа имеет удобный и понятный интерфейс с множеством различных функций, которые выполняют абсолютно разные действия. Также в программе я постаралась использовать как можно больше технологий языка питона.

Но как и любая программа, она нуждается в улучшении и обновлении. В нее можно добавить такие возможности, как создания коллажа, наложение изображений, возможность рисования и добавления текста на картинке.

Данный проект помог мне узнать новые функций и возможности программирования, а также полностью разобраться с их структурой.

**Список источников:**

* [**https://tproger.ru/translations/python-gui-pyqt/**](https://tproger.ru/translations/python-gui-pyqt/)
* [**https://pythonworld.ru/gui/pyqt5-layout.html**](https://pythonworld.ru/gui/pyqt5-layout.html)
* [**https://doc.qt.io/qt-5/qvboxlayout.html**](https://doc.qt.io/qt-5/qvboxlayout.html)
* [**https://doc.qt.io/qt-5/qtmodules.html**](https://doc.qt.io/qt-5/qtmodules.html)
* [**https://doc.qt.io/qt-5/qbuttongroup.html**](https://doc.qt.io/qt-5/qbuttongroup.html)