**Детальный план разработки проекта**

1. Архитектура проекта

Проект будет состоять из двух основных модулей (пакетов Python):

image\_processor\_app/ # Корневая директория проекта

│

├── main.py # Точка входа в приложение

│

├── core/ # Модуль обработки и работы с изображениями (Core)

│ ├── \_\_init\_\_.py

│ ├── image\_processor.py # Основной класс для обработки изображений

│ └── image\_utils.py # Вспомогательные утилиты (валидация, файловые операции)

│

└── gui/ # Модуль взаимодействия с пользователем (GUI)

├── \_\_init\_\_.py

├── main\_window.py # Главное окно приложения

├── project\_widgets.py # Виджеты для каждого типа проекта

└── styles.py # Стили для интерфейса (опционально)

2. Реализация модулей

a. Модуль обработки и работы с изображениями (core/)

Этот модуль будет содержать класс `ImageProcessor` со всеми необходимыми методами:

python

# core/image\_processor.py

from PIL import Image

import os

from datetime import datetime

class ImageProcessor:

@staticmethod

def resize\_image(input\_path, output\_path, new\_size):

"""Изменение размера изображения (Проект 1)"""

with Image.open(input\_path) as img:

resized\_img = img.resize(new\_size)

resized\_img.save(output\_path)

return True

@staticmethod

def rotate\_image(input\_path, output\_path, degrees):

"""Поворот изображения (Проект 1)"""

with Image.open(input\_path) as img:

rotated\_img = img.rotate(degrees, expand=True)

rotated\_img.save(output\_path)

return True

@staticmethod

def convert\_format(input\_path, output\_path, new\_format):

"""Изменение расширения файла (Проект 2)"""

with Image.open(input\_path) as img:

# Конвертируем в RGB если нужно (для JPG)

if new\_format.upper() == 'JPG' and img.mode in ('RGBA', 'P'):

img = img.convert('RGB')

img.save(output\_path, format=new\_format)

return True

@staticmethod

def get\_image\_info(input\_path):

"""Получение информации об изображении (Проект 3)"""

with Image.open(input\_path) as img:

file\_stats = os.stat(input\_path)

return {

'size\_bytes': file\_stats.st\_size,

'dimensions': img.size,

'format': img.format,

'mode': img.mode,

'creation\_date': datetime.fromtimestamp(file\_stats.st\_ctime)

}

@staticmethod

def merge\_images(input\_path1, input\_path2, output\_path, direction='horizontal'):

"""Склеивание изображений (Проект 4)"""

with Image.open(input\_path1) as img1, Image.open(input\_path2) as img2:

if direction == 'horizontal':

# Горизонтальное склеивание

total\_width = img1.width + img2.width

max\_height = max(img1.height, img2.height)

new\_img = Image.new('RGB', (total\_width, max\_height))

new\_img.paste(img1, (0, 0))

new\_img.paste(img2, (img1.width, 0))

else:

# Вертикальное склеивание

total\_height = img1.height + img2.height

max\_width = max(img1.width, img2.width)

new\_img = Image.new('RGB', (max\_width, total\_height))

new\_img.paste(img1, (0, 0))

new\_img.paste(img2, (0, img1.height))

new\_img.save(output\_path)

return True

@staticmethod

def convert\_to\_grayscale(input\_path, output\_path):

"""Изменение цветовой гаммы в черно-белое (Проект 5)"""

with Image.open(input\_path) as img:

grayscale\_img = img.convert('L')

grayscale\_img.save(output\_path)

return True

@staticmethod

def move\_image(input\_path, output\_folder):

"""Перемещение изображения в другие папки (Проекты 2, 5)"""

filename = os.path.basename(input\_path)

output\_path = os.path.join(output\_folder, filename)

os.rename(input\_path, output\_path)

return output\_path

@staticmethod

def rename\_image(input\_path, new\_name):

"""Переименование изображения (Проект 3)"""

directory = os.path.dirname(input\_path)

file\_extension = os.path.splitext(input\_path)[1]

new\_path = os.path.join(directory, new\_name + file\_extension)

os.rename(input\_path, new\_path)

return new\_path

b. Модуль взаимодействия с пользователем (gui/)

Главное окно приложения с вкладками для каждого проекта:

python

# gui/main\_window.py

from PyQt5.QtWidgets import (QMainWindow, QTabWidget, QVBoxLayout,

QWidget, QMessageBox, QFileDialog)

from .project\_widgets import (ResizeRotateWidget, FormatConvertWidget,

InfoRenameWidget, MergeWidget, GrayscaleWidget)

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Image Processing Application")

self.setGeometry(100, 100, 800, 600)

self.init\_ui()

def init\_ui(self):

# Создаем виджет с вкладками

self.tabs = QTabWidget()

# Добавляем вкладки для каждого проекта

self.tabs.addTab(ResizeRotateWidget(), "Проект 1 - Изменение размера/поворот")

self.tabs.addTab(FormatConvertWidget(), "Проект 2 - Конвертация/перемещение")

self.tabs.addTab(InfoRenameWidget(), "Проект 3 - Информация/переименование")

self.tabs.addTab(MergeWidget(), "Проект 4 - Склеивание изображений")

self.tabs.addTab(GrayscaleWidget(), "Проект 5 - Ч/Б конвертация")

self.setCentralWidget(self.tabs)

def show\_success\_message(self, message):

QMessageBox.information(self, "Успех", message)

def show\_error\_message(self, message):

QMessageBox.warning(self, "Ошибка", message)

def get\_save\_filename(self, caption, default\_filter):

return QFileDialog.getSaveFileName(self, caption, filter=default\_filter)

def get\_existing\_directory(self, caption):

return QFileDialog.getExistingDirectory(self, caption)

Пример реализации виджета для Проекта 1:

python

# gui/project\_widgets.py

from PyQt5.QtWidgets import (QWidget, QVBoxLayout, QHBoxLayout, QLabel,

QLineEdit, QPushButton, QSpinBox, QComboBox)

from PyQt5.QtCore import Qt

from core.image\_processor import ImageProcessor

import os

class ResizeRotateWidget(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.init\_ui()

def init\_ui(self):

layout = QVBoxLayout()

# Выбор файла

file\_layout = QHBoxLayout()

file\_layout.addWidget(QLabel("Путь к изображению:"))

self.file\_path = QLineEdit()

file\_layout.addWidget(self.file\_path)

self.browse\_btn = QPushButton("Обзор")

self.browse\_btn.clicked.connect(self.browse\_file)

file\_layout.addWidget(self.browse\_btn)

layout.addLayout(file\_layout)

# Изменение размера

layout.addWidget(QLabel("Изменение размера:"))

size\_layout = QHBoxLayout()

size\_layout.addWidget(QLabel("Ширина:"))

self.width\_spin = QSpinBox()

self.width\_spin.setRange(1, 10000)

self.width\_spin.setValue(800)

size\_layout.addWidget(self.width\_spin)

size\_layout.addWidget(QLabel("Высота:"))

self.height\_spin = QSpinBox()

self.height\_spin.setRange(1, 10000)

self.height\_spin.setValue(600)

size\_layout.addWidget(self.height\_spin)

self.resize\_btn = QPushButton("Изменить размер")

self.resize\_btn.clicked.connect(self.resize\_image)

size\_layout.addWidget(self.resize\_btn)

layout.addLayout(size\_layout)

# Поворот

layout.addWidget(QLabel("Поворот изображения:"))

rotate\_layout = QHBoxLayout()

rotate\_layout.addWidget(QLabel("Градусы:"))

self.rotate\_combo = QComboBox()

self.rotate\_combo.addItems(["90", "180", "270"])

rotate\_layout.addWidget(self.rotate\_combo)

self.rotate\_btn = QPushButton("Повернуть")

self.rotate\_btn.clicked.connect(self.rotate\_image)

rotate\_layout.addWidget(self.rotate\_btn)

layout.addLayout(rotate\_layout)

self.setLayout(layout)

def browse\_file(self):

filename, \_ = QFileDialog.getOpenFileName(

self, "Выберите изображение", "",

"Image Files (\*.png \*.jpg \*.jpeg \*.bmp)"

)

if filename:

self.file\_path.setText(filename)

def resize\_image(self):

try:

input\_path = self.file\_path.text()

if not os.path.exists(input\_path):

self.parent().parent().show\_error\_message("Файл не существует")

return

new\_size = (self.width\_spin.value(), self.height\_spin.value())

output\_path = self.get\_output\_path("resized")

ImageProcessor.resize\_image(input\_path, output\_path, new\_size)

self.parent().parent().show\_success\_message(

f"Изображение успешно изменено и сохранено как:\n{output\_path}"

)

except Exception as e:

self.parent().parent().show\_error\_message(f"Ошибка: {str(e)}")

def rotate\_image(self):

try:

input\_path = self.file\_path.text()

if not os.path.exists(input\_path):

self.parent().parent().show\_error\_message("Файл не существует")

return

degrees = int(self.rotate\_combo.currentText())

output\_path = self.get\_output\_path("rotated")

ImageProcessor.rotate\_image(input\_path, output\_path, degrees)

self.parent().parent().show\_success\_message(

f"Изображение успешно повернуто и сохранено как:\n{output\_path}"

)

except Exception as e:

self.parent().parent().show\_error\_message(f"Ошибка: {str(e)}")

def get\_output\_path(self, prefix):

input\_path = self.file\_path.text()

directory = os.path.dirname(input\_path)

filename = os.path.basename(input\_path)

name, ext = os.path.splitext(filename)

return os.path.join(directory, f"{prefix}\_{name}{ext}")

c. Точка входа в приложение

python

# main.py

import sys

from PyQt5.QtWidgets import QApplication

from gui.main\_window import MainWindow

def main():

app = QApplication(sys.argv)

# Настройка стиля приложения

app.setStyle('Fusion')

window = MainWindow()

window.show()

sys.exit(app.exec\_())

if \_\_name\_\_ == '\_\_main\_\_':

main()

3. Установка зависимостей

Создайте файл requirements.txt:

Pillow>=9.0.0

PyQt5>=5.15.0

Установите зависимости:

bash

pip install -r requirements.txt