Вот переработанная программа с графическим интерфейсом (Windows-стиль) на **PyQt5**, которая:

* Работает с **Git**
* Синхронизируется с **GitHub**
* Загружает файлы на **Яндекс.Диск** и **Google Drive**
* Имеет настройки для исключения папок

**📁 Файловая структура**

Copy

Download

RepoManager/

│

├── main.py # Основной код

├── config.json # Настройки (создаётся автоматически)

├── requirements.txt # Зависимости

└── icons/ # Иконки (опционально)

**🚀 Код программы (main.py)**

python

Copy

Download

import os

import json

import subprocess

import shutil

from pathlib import Path

from PyQt5.QtWidgets import (

QApplication, QMainWindow, QVBoxLayout, QWidget, QPushButton,

QLabel, QLineEdit, QTextEdit, QListWidget, QFileDialog, QMessageBox

)

from PyQt5.QtCore import Qt

class RepoManager(QMainWindow):

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

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

self.setWindowTitle("Repo Manager")

self.setGeometry(300, 300, 600, 500)

self.config\_file = "config.json"

self.config = self.load\_config()

self.init\_ui()

def init\_ui(self):

*# Главный виджет*

central\_widget = QWidget()

self.setCentralWidget(central\_widget)

layout = QVBoxLayout()

*# Поля ввода*

self.repo\_path = QLineEdit(self.config.get("repo\_path", ""))

self.github\_repo = QLineEdit(self.config.get("github\_repo", ""))

self.yandex\_token = QLineEdit(self.config.get("yandex\_token", ""))

self.google\_drive\_path = QLineEdit(self.config.get("google\_drive\_path", ""))

*# Кнопки*

self.btn\_select\_repo = QPushButton("Выбрать папку")

self.btn\_init\_git = QPushButton("Инициализировать Git")

self.btn\_sync\_github = QPushButton("Синхронизировать с GitHub")

self.btn\_backup\_yandex = QPushButton("Резервная копия на Яндекс.Диск")

self.btn\_backup\_google = QPushButton("Резервная копия на Google Drive")

self.btn\_save\_config = QPushButton("Сохранить настройки")

*# Список исключаемых папок*

self.exclude\_list = QListWidget()

self.exclude\_list.addItems(self.config.get("exclude\_folders", []))

*# Лог*

self.log = QTextEdit()

self.log.setReadOnly(True)

*# Добавление элементов в layout*

layout.addWidget(QLabel("Путь к репозиторию:"))

layout.addWidget(self.repo\_path)

layout.addWidget(self.btn\_select\_repo)

layout.addWidget(QLabel("GitHub репозиторий (URL):"))

layout.addWidget(self.github\_repo)

layout.addWidget(QLabel("Токен Яндекс.Диска:"))

layout.addWidget(self.yandex\_token)

layout.addWidget(QLabel("Путь к Google Drive:"))

layout.addWidget(self.google\_drive\_path)

layout.addWidget(QLabel("Исключаемые папки:"))

layout.addWidget(self.exclude\_list)

layout.addWidget(self.btn\_init\_git)

layout.addWidget(self.btn\_sync\_github)

layout.addWidget(self.btn\_backup\_yandex)

layout.addWidget(self.btn\_backup\_google)

layout.addWidget(self.btn\_save\_config)

layout.addWidget(QLabel("Лог:"))

layout.addWidget(self.log)

central\_widget.setLayout(layout)

*# Обработчики событий*

self.btn\_select\_repo.clicked.connect(self.select\_repo)

self.btn\_init\_git.clicked.connect(self.init\_git\_repo)

self.btn\_sync\_github.clicked.connect(self.sync\_with\_github)

self.btn\_backup\_yandex.clicked.connect(self.backup\_to\_yandex\_disk)

self.btn\_backup\_google.clicked.connect(self.backup\_to\_google\_drive)

self.btn\_save\_config.clicked.connect(self.save\_config)

def load\_config(self):

if os.path.exists(self.config\_file):

with open(self.config\_file, "r") as f:

return json.load(f)

return {

"repo\_path": "",

"github\_repo": "",

"yandex\_token": "",

"google\_drive\_path": "",

"exclude\_folders": []

}

def save\_config(self):

self.config = {

"repo\_path": self.repo\_path.text(),

"github\_repo": self.github\_repo.text(),

"yandex\_token": self.yandex\_token.text(),

"google\_drive\_path": self.google\_drive\_path.text(),

"exclude\_folders": [self.exclude\_list.item(i).text() for i in range(self.exclude\_list.count())]

}

with open(self.config\_file, "w") as f:

json.dump(self.config, f, indent=4)

self.log\_message("Настройки сохранены!")

def select\_repo(self):

folder = QFileDialog.getExistingDirectory(self, "Выберите папку репозитория")

if folder:

self.repo\_path.setText(folder)

def log\_message(self, message):

self.log.append(message)

def run\_command(self, command, cwd=None):

try:

result = subprocess.run(

command,

cwd=cwd or self.config["repo\_path"],

stdout=subprocess.PIPE,

stderr=subprocess.PIPE,

text=True,

shell=True

)

if result.returncode == 0:

self.log\_message(result.stdout)

return True

else:

self.log\_message(f"Ошибка: {result.stderr}")

return False

except Exception as e:

self.log\_message(f"Исключение: {str(e)}")

return False

def init\_git\_repo(self):

if not self.config["repo\_path"]:

QMessageBox.warning(self, "Ошибка", "Укажите путь к репозиторию!")

return

if os.path.exists(os.path.join(self.config["repo\_path"], ".git")):

self.log\_message("Git репозиторий уже инициализирован.")

return

if self.run\_command("git init"):

self.log\_message("Git репозиторий успешно создан!")

def sync\_with\_github(self):

if not self.config["github\_repo"]:

QMessageBox.warning(self, "Ошибка", "Укажите URL репозитория GitHub!")

return

commands = [

"git remote add origin " + self.config["github\_repo"],

"git add .",

'git commit -m "Auto commit from Repo Manager"',

"git push -u origin master"

]

for cmd in commands:

if not self.run\_command(cmd):

break

def backup\_to\_yandex\_disk(self):

if not self.config["yandex\_token"]:

QMessageBox.warning(self, "Ошибка", "Укажите токен Яндекс.Диска!")

return

*# Здесь должна быть реализация загрузки через API Яндекс.Диска*

self.log\_message("Загрузка на Яндекс.Диск... (реализуйте этот метод)")

def backup\_to\_google\_drive(self):

if not self.config["google\_drive\_path"]:

QMessageBox.warning(self, "Ошибка", "Укажите путь к Google Drive!")

return

*# Копирование файлов*

try:

dest = os.path.join(self.config["google\_drive\_path"], os.path.basename(self.config["repo\_path"]))

if os.path.exists(dest):

shutil.rmtree(dest)

shutil.copytree(self.config["repo\_path"], dest)

self.log\_message(f"Резервная копия создана в {dest}")

except Exception as e:

self.log\_message(f"Ошибка: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication([])

window = RepoManager()

window.show()

app.exec\_()

**📦 Зависимости (requirements.txt)**

Copy

Download

PyQt5==5.15.9

gitpython==3.1.40

requests==2.31.0

Установите их командой:

bash

Copy

Download

pip install -r requirements.txt

**🖼️ Как это выглядит?**

Программа имеет:

* Поля для ввода настроек
* Кнопки для действий (Git, GitHub, Яндекс.Диск, Google Drive)
* Лог операций
* Сохранение конфигурации в config.json

**🔧 Что нужно доработать?**

1. **API Яндекс.Диска**: Реализуйте метод backup\_to\_yandex\_disk() с использованием вашего OAuth-токена.
2. **Google Drive API**: Добавьте загрузку через API вместо простого копирования.
3. **Дизайн**: Можно улучшить интерфейс с помощью Qt Designer.

**▶️ Запуск**

bash

Copy

Download

python main.py

Теперь у вас есть программа с графическим интерфейсом для управления репозиториями!