|  |  |
| --- | --- |
| http://www.pl130.ru/doc/index/%D0%9B%D0%BE%D0%B3%D0%BE%D1%82%D0%B8%D0%BF22.jpg | Санкт-Петербургское государственное бюджетное  профессиональное образовательное учреждение  "Колледж электроники и приборостроения" |

отчет

**По практической работе**

**по дисциплине МДК 01.01 «Разработка программных модулей»**

**Тема: «PyQt6 исполняемый файл»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22ИТ |  | Липинский К.С. |
| Преподаватель |  | Склярова А. В. |

Санкт-Петербург

2025

# Цель работы:

Создать приложение PyQt6, далее нужно скомпилировать исходный код Python в автономный исполняемый файл. Кроме того, обычно требуется создать установщик, чтобы пользователи могли легко настроить  
ваше приложение.

# Ход работы:

Код созданного приложение в VS Code представлен ниже:

import sys

import os

from PyQt6.QtWidgets import (QApplication, QMainWindow, QPlainTextEdit,

                            QFileDialog, QMessageBox, QMenuBar, QStatusBar)

from PyQt6.QtCore import Qt, QTimer

from PyQt6.QtGui import QAction, QKeySequence, QFont, QIcon

class TextEditor(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.current\_file = None

        self.is\_modified = False

        self.init\_ui()

        self.init\_menus()

        self.init\_statusbar()

        self.connect\_signals()

        self.update\_title()

    def init\_ui(self):

        self.setWindowTitle("Текстовый редактор")

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

        self.text\_edit = QPlainTextEdit()

        self.text\_edit.setFont(QFont("Consolas", 11))

        self.setCentralWidget(self.text\_edit)

    def init\_menus(self):

        menubar = self.menuBar()

        file\_menu = menubar.addMenu('&Файл')

        open\_action = QAction('&Открыть', self)

        open\_action.setShortcut(QKeySequence.StandardKey.Open)

        open\_action.setStatusTip('Открыть существующий файл')

        open\_action.triggered.connect(self.open\_file)

        file\_menu.addAction(open\_action)

        save\_action = QAction('&Сохранить', self)

        save\_action.setShortcut(QKeySequence.StandardKey.Save)

        save\_action.setStatusTip('Сохранить текущий файл')

        save\_action.triggered.connect(self.save\_file)

        file\_menu.addAction(save\_action)

        save\_as\_action = QAction('Сохранить &как...', self)

        save\_as\_action.setShortcut(QKeySequence.StandardKey.SaveAs)

        save\_as\_action.setStatusTip('Сохранить файл с новым именем')

        save\_as\_action.triggered.connect(self.save\_file\_as)

        file\_menu.addAction(save\_as\_action)

        file\_menu.addSeparator()

        new\_action = QAction('&Новый', self)

        new\_action.setShortcut(QKeySequence.StandardKey.New)

        new\_action.setStatusTip('Создать новый документ')

        new\_action.triggered.connect(self.new\_file)

        file\_menu.addAction(new\_action)

        file\_menu.addSeparator()

        close\_action = QAction('&Закрыть', self)

        close\_action.setShortcut(QKeySequence.StandardKey.Quit)

        close\_action.setStatusTip('Закрыть приложение')

        close\_action.triggered.connect(self.close\_application)

        file\_menu.addAction(close\_action)

        edit\_menu = menubar.addMenu('&Правка')

        undo\_action = QAction('&Отменить', self)

        undo\_action.setShortcut(QKeySequence.StandardKey.Undo)

        undo\_action.setStatusTip('Отменить последнее действие')

        undo\_action.triggered.connect(self.text\_edit.undo)

        edit\_menu.addAction(undo\_action)

        redo\_action = QAction('&Повторить', self)

        redo\_action.setShortcut(QKeySequence.StandardKey.Redo)

        redo\_action.setStatusTip('Повторить отмененное действие')

        redo\_action.triggered.connect(self.text\_edit.redo)

        edit\_menu.addAction(redo\_action)

        edit\_menu.addSeparator()

        cut\_action = QAction('&Вырезать', self)

        cut\_action.setShortcut(QKeySequence.StandardKey.Cut)

        cut\_action.setStatusTip('Вырезать выделенный текст')

        cut\_action.triggered.connect(self.text\_edit.cut)

        edit\_menu.addAction(cut\_action)

        copy\_action = QAction('&Копировать', self)

        copy\_action.setShortcut(QKeySequence.StandardKey.Copy)

        copy\_action.setStatusTip('Копировать выделенный текст')

        copy\_action.triggered.connect(self.text\_edit.copy)

        edit\_menu.addAction(copy\_action)

        paste\_action = QAction('&Вставить', self)

        paste\_action.setShortcut(QKeySequence.StandardKey.Paste)

        paste\_action.setStatusTip('Вставить текст из буфера обмена')

        paste\_action.triggered.connect(self.text\_edit.paste)

        edit\_menu.addAction(paste\_action)

        help\_menu = menubar.addMenu('&Справка')

        about\_action = QAction('&О программе', self)

        about\_action.setStatusTip('Информация о программе')

        about\_action.triggered.connect(self.show\_about)

        help\_menu.addAction(about\_action)

    def init\_statusbar(self):

        self.status\_bar = QStatusBar()

        self.setStatusBar(self.status\_bar)

        self.status\_bar.showMessage('Готов')

    def connect\_signals(self):

        self.text\_edit.textChanged.connect(self.on\_text\_changed)

    def on\_text\_changed(self):

        if not self.is\_modified:

            self.is\_modified = True

            self.update\_title()

    def update\_title(self):

        if self.current\_file:

            filename = os.path.basename(self.current\_file)

        else:

            filename = "Новый документ"

        modified\_indicator = " \*" if self.is\_modified else ""

        self.setWindowTitle(f"{filename}{modified\_indicator} - Текстовый редактор")

    def new\_file(self):

        if self.check\_unsaved\_changes():

            self.text\_edit.clear()

            self.current\_file = None

            self.is\_modified = False

            self.update\_title()

            self.status\_bar.showMessage('Создан новый документ')

    def open\_file(self):

        if not self.check\_unsaved\_changes():

            return

        file\_path, \_ = QFileDialog.getOpenFileName(

            self,

            'Открыть файл',

            '',

            'Текстовые файлы (\*.txt);;Все файлы (\*.\*)'

        )

        if file\_path:

            try:

                with open(file\_path, 'r', encoding='utf-8') as file:

                    content = file.read()

                    self.text\_edit.setPlainText(content)

                self.current\_file = file\_path

                self.is\_modified = False

                self.update\_title()

                self.status\_bar.showMessage(f'Файл открыт: {os.path.basename(file\_path)}')

            except Exception as e:

                QMessageBox.critical(

                    self,

                    'Ошибка',

                    f'Не удалось открыть файл:\n{str(e)}'

                )

    def save\_file(self):

        if self.current\_file:

            return self.save\_to\_file(self.current\_file)

        else:

            return self.save\_file\_as()

    def save\_file\_as(self):

        file\_path, \_ = QFileDialog.getSaveFileName(

            self,

            'Сохранить файл как',

            '',

            'Текстовые файлы (\*.txt);;Все файлы (\*.\*)'

        )

        if file\_path:

            return self.save\_to\_file(file\_path)

        return False

    def save\_to\_file(self, file\_path):

        try:

            with open(file\_path, 'w', encoding='utf-8') as file:

                file.write(self.text\_edit.toPlainText())

            self.current\_file = file\_path

            self.is\_modified = False

            self.update\_title()

            self.status\_bar.showMessage(f'Файл сохранен: {os.path.basename(file\_path)}')

            return True

        except Exception as e:

            QMessageBox.critical(

                self,

                'Ошибка',

                f'Не удалось сохранить файл:\n{str(e)}'

            )

            return False

    def check\_unsaved\_changes(self):

        if not self.is\_modified:

            return True

        reply = QMessageBox.question(

            self,

            'Несохраненные изменения',

            'Документ был изменен.\nСохранить изменения?',

            QMessageBox.StandardButton.Save |

            QMessageBox.StandardButton.Discard |

            QMessageBox.StandardButton.Cancel,

            QMessageBox.StandardButton.Save

        )

        if reply == QMessageBox.StandardButton.Save:

            return self.save\_file()

        elif reply == QMessageBox.StandardButton.Discard:

            return True

        else:

            return False

    def close\_application(self):

        if self.check\_unsaved\_changes():

            self.close()

    def closeEvent(self, event):

        if self.check\_unsaved\_changes():

            event.accept()

        else:

            event.ignore()

    def show\_about(self):

        QMessageBox.about(

            self,

            'О программе',

            '''<h3>Текстовый редактор</h3>

            <p>Версия: 1.0</p>

            <p>Простой текстовый редактор на PyQt6</p>

            <p><b>Возможности:</b></p>

            <ul>

            <li>Создание, открытие и сохранение файлов</li>

            <li>Базовые операции редактирования</li>

            <li>Проверка несохраненных изменений</li>

            </ul>

            <p>© 2025 Текстовый редактор</p>'''

        )

def main():

    app = QApplication(sys.argv)

    app.setStyle('Fusion')

    editor = TextEditor()

    editor.show()

    sys.exit(app.exec())

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

    main()

На рисунке 1 представлено рабочее приложение.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Рабочее приложение.

Мы импортируем fbs's ApplicationContext:

from fbs\_runtime.application\_context.PyQt6 import ApplicationContext

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Код для импорт в проекте.

Через терминал импорт пакетов fbs, но из-за новой версии пайтена нельзя получить бесплатную версию (см. рис. 3 и 4).

A screen shot of a computer program

AI-generated content may be incorrect.

Рисунок 3 – Ошибка импорта.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 – Цены сайта.

Вместо него было использовано pip install pyinstaller (см. ри. 5).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 - pip install pyinstaller.

Результаты на рисунках 6-8.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 – Успешный импорт.Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Файл создан.



Рисунок 8 – Расположение файла.

# Вывод:

В ходе работы создано приложение на PyQt6, которое было скомпилировано в автономный исполняемый файл с помощью pyinstaller (с fbs возникли проблемы), а также подготовлен установщик для удобной установки пользователями.