|  |  |
| --- | --- |
| К Г ЭУ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  Федеральное государственное бюджетное образовательное учреждение высшего образования  «КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ» |

Отчет по лабораторной работе №8

по дисциплине «Проектный практикум по управлению разработкой и разработке программного обеспечения»

**«Архитектурный планировщик»**

**Выполнили:**

Шафеев Вадим ТРП-3-21

Смоленский Илья ТРП-3-21

**Проверил:**

доцент кафедры ИТИС, к.ф.-м.н. Хуснутдинов Р. М.

Казань, 2024г.

**Цель работы:** Разработать алгоритм и написать программу «Файловый менеджер» для работы с файлами и каталогами. При разработке программного кода возможно объединение в группы не более чем из двух человек. Составить техническое задание и справочную документацию для данного программного обеспечения. Выгрузить программный код на платформу GitHub.

**1. НАЗНАЧЕНИЕ РАЗРАБОТКИ**

Разработка файлового менеджера имеет своей целью создание удобного и многофункционального инструмента для управления файлами и папками на различных устройствах. Основное назначение данной разработки заключается в упрощении управления данными, поскольку файловый менеджер предоставляет интуитивно понятный интерфейс, позволяющий пользователям легко выполнять основные операции с файлами (создание, копирование, перемещение, удаление и переименование) без необходимости глубокого понимания файловой системы. Наличие простого и быстрого доступа к функциям управления файлами повышает эффективность рабочей среды, позволяя пользователям сосредоточиться на выполнении своих задач и предотвращая чрезмерные временные затраты на поиск и организацию данных.

Файловый менеджер также обеспечивает возможность управления файлами различных типов, что позволяет пользователям обрабатывать текстовые документы, изображения, видео и другие форматы без необходимости изменения программного обеспечения. В современных условиях актуальность облачных технологий неуклонно растёт, и файловый менеджер должен поддерживать интеграцию с облачными сервисами, позволяя пользователям управлять своими данными как локально, так и удалённо. Кроме того, разработка файлового менеджера включает в себя механизмы для изменения атрибутов и прав доступа к файлам, что важно для защиты конфиденциальной информации и предотвращения несанкционированного доступа.

Для оптимизации работы с большими объемами данных файл менеджер должен предоставлять функции поиска и фильтрации, что упростит нахождение необходимых файлов. В результате, разработка файлового менеджера направлена на создание надежного, эффективного и многофункционального инструмента, который отвечает потребностям пользователей и обеспечивает удобство при управлении данными в современном цифровом мире.

2. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

**Функциональные возможности программы:**

1. Создание, открытие/просмотр, перемещение, переименование, копирование, удаление, изменение атрибутов и свойств, поиск каталогов и назначение прав.
2. Создание, открытие/просмотр, редактирование, перемещение, переименование, копирование, удаление, изменение атрибутов и свойств, поиск файлов и назначение прав.

**3. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

3.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

3.2 Разрабатываемое программное обеспечение должно включать справочную систему.

3.3 В состав сопровождающей документации должны входить:

3.3.1 Расчетно-пояснительная записка на 25-30 листах формата А4 (без приложений 5.3.2, 5.3.3 и 5.3.4).

3.3.2 Техническое задание (Приложение 1).

3.3.3 Руководство пользователя (Приложение 2).

3.3.4 Руководство системного программиста (Приложение 3).

3.4 Графическая часть должна быть включена в расчетно-пояснительную записку в качестве иллюстраций (выбрать 7-8 обязательных чертежей):

3.4.1 Диаграмма вариантов использования (объектный подход).

3.4.2 Концептуальная модель предметной области (объектный подход).

3.4.3 Схемы взаимодействия объектов, объектная декомпозиция (объектный подход).

3.4.4 Функциональная диаграмма программного обеспечения (или его части) (структурный подход).

3.4.5 Диаграмма потоков данных программного обеспечения или его части (структурный подход).

3.4.6 Диаграммы (схемы) компонентов структур данных.

3.4.7 Схема структурная программного обеспечения.

3.4.8 Схема функциональная программного обеспечения.

3.4.9 Схемы (модели) процессов (методов формирования результатов, механизмы выводов и т.п.).

3.4.10 Диаграммы классов предметной области и/или интерфейсной части программного обеспечения.

3.4.11 Схема взаимодействия модулей Константайна (структурный подход)

3.4.12 Граф состояний интерфейса.

3.4.13 Структурная схема меню.

3.4.14 Графы диалогов.

3.4.15 Формы интерфейса.

3.4.16 Схемы алгоритмов модулей (подпрограмм).

3.4.17 Диаграммы компоновки программных компонентов.

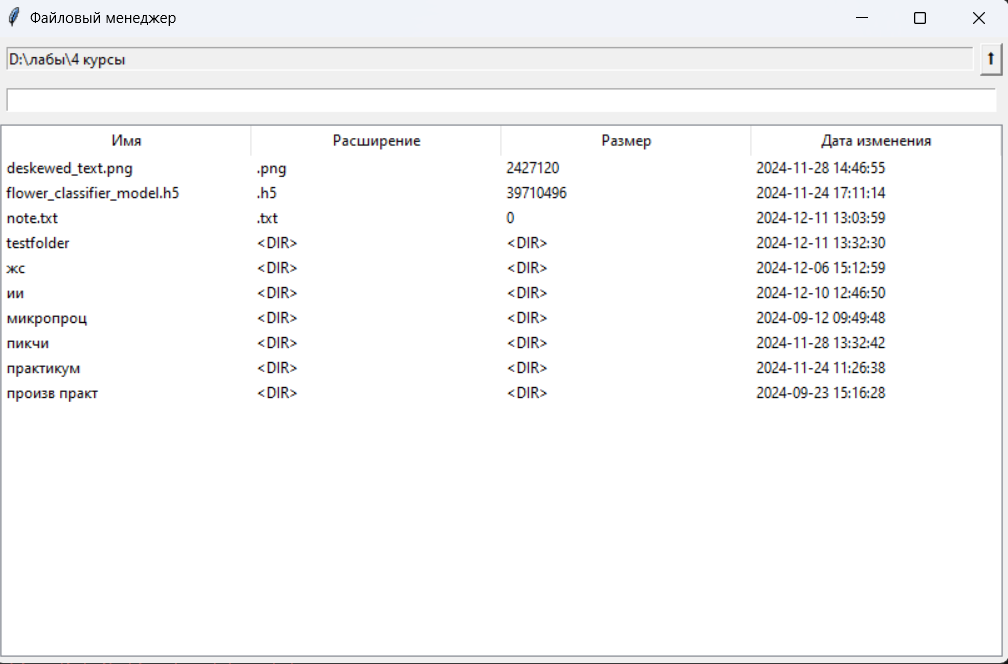
3.4.18 Таблица характеристик (инструментальных средств разработки, языка, среды программирования, средств автоматизации разработки, методов тестирования, подхода к разработке).

3.4.19 Таблицы тестов.

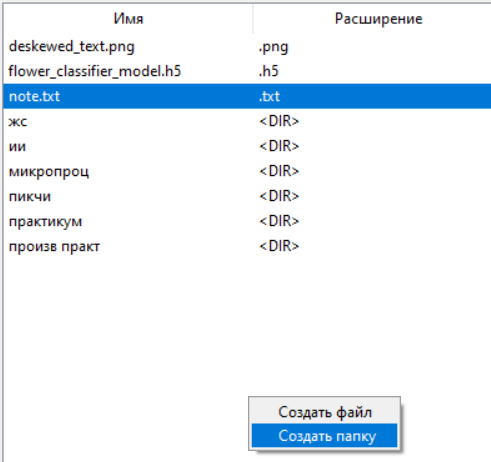
**4. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок, дни | Отчетность |
| 1. | Этап разработки | 6 дней | Техническое задание |
| 2. | Этап тестирования | 2 дня | Спецификации программного обеспечения |

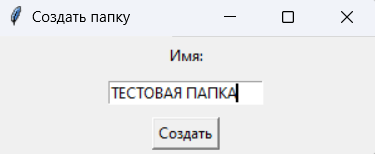
**СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ**

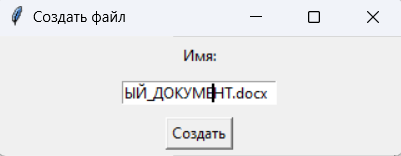


Основное окно программы

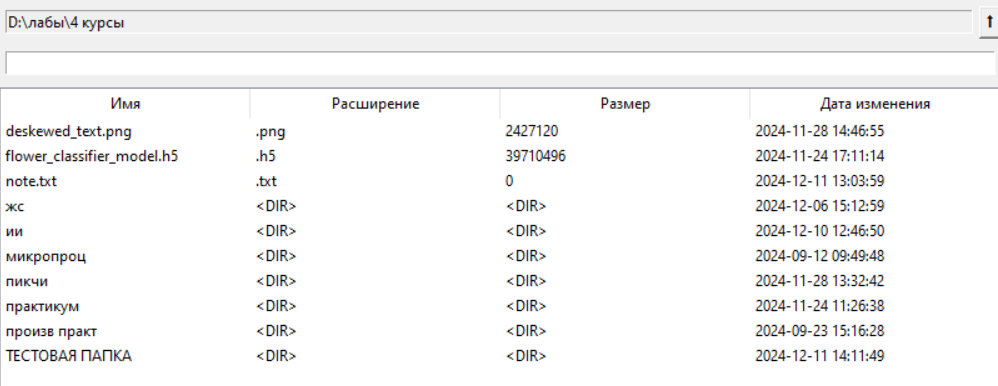


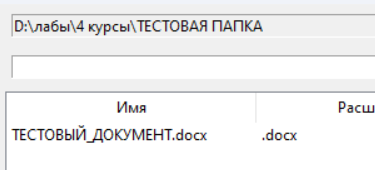
Контекстное меню для создания файлов и директорий



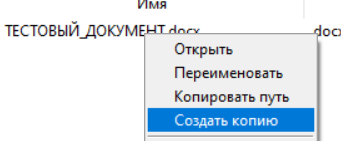
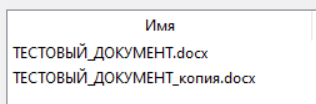


Окна создания файлов и директорий

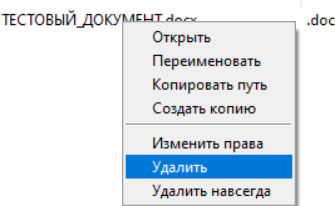


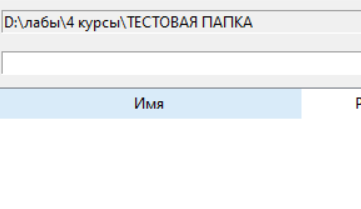


Результаты создания

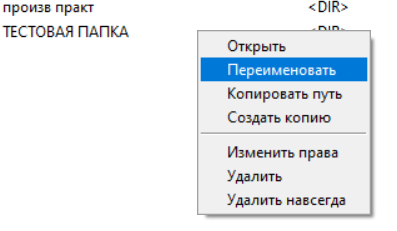
 

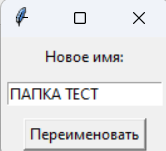
Создание копии файла

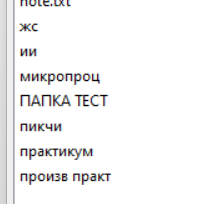




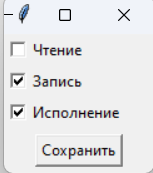
Удаление файлов



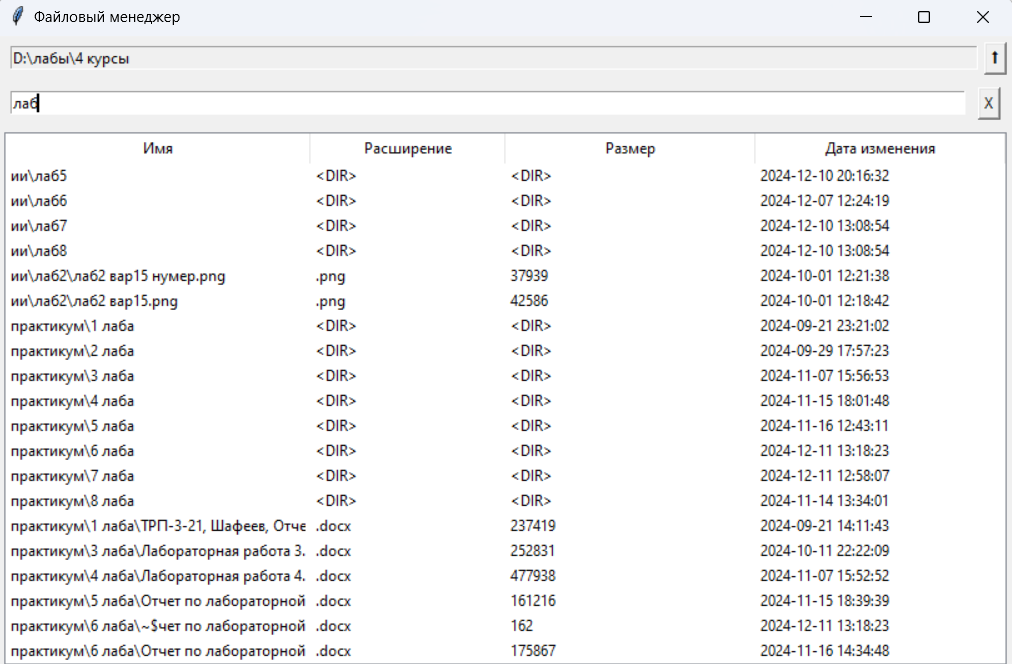




Переименование файлов и директорий



Управление правами доступа



Поиск файлов и директорий

**Код программы:**

import os

import shutil

import tkinter as tk

from tkinter import ttk, filedialog, messagebox

from datetime import datetime

import send2trash  # Для перемещения файлов в корзину

import stat  # Для изменения прав доступа

def get\_file\_info(path):

    info = []

    for item in os.listdir(path):

        full\_path = os.path.join(path, item)

        is\_dir = os.path.isdir(full\_path)

        size = os.path.getsize(full\_path) if not is\_dir else "<DIR>"

        mtime = datetime.fromtimestamp(os.path.getmtime(full\_path)).strftime('%Y-%m-%d %H:%M:%S')

        ext = os.path.splitext(item)[1] if not is\_dir else "<DIR>"

        info.append((item, ext, size, mtime))

    return info

def search\_files\_recursive(query, path):

    results = []

    query\_lower = query.lower()

    for root, dirs, files in os.walk(path):

        for item in files + dirs:

            if query\_lower in item.lower():

                full\_path = os.path.join(root, item)

                is\_dir = os.path.isdir(full\_path)

                size = os.path.getsize(full\_path) if not is\_dir else "<DIR>"

                mtime = datetime.fromtimestamp(os.path.getmtime(full\_path)).strftime('%Y-%m-%d %H:%M:%S')

                ext = os.path.splitext(item)[1] if not is\_dir else "<DIR>"

                relative\_path = os.path.relpath(full\_path, path)

                results.append((relative\_path, ext, size, mtime))

    return results

def update\_file\_list(tree, path\_var, search\_var=None):

    path = path\_var.get()

    tree.delete(\*tree.get\_children())

    path\_var.set(os.path.abspath(path))

    if search\_var and search\_var.get():

        query = search\_var.get()

        file\_info = search\_files\_recursive(query, path)

    else:

        file\_info = get\_file\_info(path)

    for item, ext, size, mtime in file\_info:

        tree.insert('', 'end', values=(item, ext, size, mtime))

def go\_up(tree, path\_var, search\_var):

    path = os.path.abspath(os.path.join(path\_var.get(), '..'))

    path\_var.set(path)

    search\_var.set("")

    update\_file\_list(tree, path\_var)

def on\_item\_double\_click(event, tree, path\_var, search\_var):

    selected\_item = tree.focus()

    if not selected\_item:

        return

    item\_name = tree.item(selected\_item)['values'][0]

    new\_path = os.path.join(path\_var.get(), item\_name)

    if os.path.isdir(new\_path):

        path\_var.set(new\_path)

        search\_var.set("")

        update\_file\_list(tree, path\_var)

    else:

        os.startfile(new\_path)

def clear\_search(tree, path\_var, search\_var, clear\_button):

    search\_var.set("")

    clear\_button.pack\_forget()

    update\_file\_list(tree, path\_var)

def on\_search(event, tree, path\_var, search\_var, clear\_button):

    query = search\_var.get()

    if query:

        update\_file\_list(tree, path\_var, search\_var)

        clear\_button.pack(side='right', padx=5)

def create\_new\_item(path, is\_folder):

    def submit():

        name = entry.get()

        if not name:

            messagebox.showerror("Ошибка", "Имя не может быть пустым!")

            return

        new\_path = os.path.join(path, name)

        try:

            if is\_folder:

                os.mkdir(new\_path)

            else:

                open(new\_path, 'w').close()

            dialog.destroy()

            update\_file\_list(tree, path\_var)

        except Exception as e:

            messagebox.showerror("Ошибка", f"Не удалось создать: {e}")

    dialog = tk.Toplevel()

    dialog.title("Создать" + (" папку" if is\_folder else " файл"))

    tk.Label(dialog, text="Имя:").pack(padx=5, pady=5)

    entry = tk.Entry(dialog)

    entry.pack(padx=5, pady=5)

    tk.Button(dialog, text="Создать", command=submit).pack(padx=5, pady=5)

def change\_permissions(item\_path):

    def submit():

        try:

            mode = 0

            if read\_var.get():

                mode |= stat.S\_IRUSR

            if write\_var.get():

                mode |= stat.S\_IWUSR

            if exec\_var.get():

                mode |= stat.S\_IXUSR

            os.chmod(item\_path, mode)

            perm\_dialog.destroy()

            messagebox.showinfo("Успешно", "Права изменены успешно!")

        except Exception as e:

            messagebox.showerror("Ошибка", f"Не удалось изменить права: {e}")

    perm\_dialog = tk.Toplevel()

    perm\_dialog.title("Изменить права")

    read\_var = tk.BooleanVar(value=os.access(item\_path, os.R\_OK))

    write\_var = tk.BooleanVar(value=os.access(item\_path, os.W\_OK))

    exec\_var = tk.BooleanVar(value=os.access(item\_path, os.X\_OK))

    tk.Checkbutton(perm\_dialog, text="Чтение", variable=read\_var).pack(anchor='w')

    tk.Checkbutton(perm\_dialog, text="Запись", variable=write\_var).pack(anchor='w')

    tk.Checkbutton(perm\_dialog, text="Исполнение", variable=exec\_var).pack(anchor='w')

    tk.Button(perm\_dialog, text="Сохранить", command=submit).pack(padx=5, pady=5)

def on\_right\_click(event, tree, path\_var):

    def open\_item():

        os.startfile(item\_path)

    def rename\_item():

        def submit():

            new\_name = entry.get()

            if not new\_name:

                messagebox.showerror("Ошибка", "Имя не может быть пустым!")

                return

            new\_path = os.path.join(path\_var.get(), new\_name)

            try:

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

                rename\_dialog.destroy()

                update\_file\_list(tree, path\_var)

            except Exception as e:

                messagebox.showerror("Ошибка", f"Не удалось переименовать: {e}")

        rename\_dialog = tk.Toplevel()

        rename\_dialog.title("Переименовать")

        tk.Label(rename\_dialog, text="Новое имя:").pack(padx=5, pady=5)

        entry = tk.Entry(rename\_dialog)

        entry.pack(padx=5, pady=5)

        tk.Button(rename\_dialog, text="Переименовать", command=submit).pack(padx=5, pady=5)

    def copy\_path():

        root.clipboard\_clear()

        root.clipboard\_append(item\_path)

        root.update()

    def delete\_item(permanent):

        try:

            if permanent:

                if os.path.isdir(item\_path):

                    shutil.rmtree(item\_path)

                else:

                    os.remove(item\_path)

            else:

                send2trash.send2trash(item\_path)

            update\_file\_list(tree, path\_var)

        except Exception as e:

            messagebox.showerror("Ошибка", f"Не удалось удалить: {e}")

    selected\_item = tree.identify\_row(event.y)

    if not selected\_item:

        menu = tk.Menu(root, tearoff=0)

        menu.add\_command(label="Создать файл", command=lambda: create\_new\_item(path\_var.get(), is\_folder=False))

        menu.add\_command(label="Создать папку", command=lambda: create\_new\_item(path\_var.get(), is\_folder=True))

        menu.post(event.x\_root, event.y\_root)

    else:

        item\_name = tree.item(selected\_item)['values'][0]

        item\_path = os.path.join(path\_var.get(), item\_name)

        menu = tk.Menu(root, tearoff=0)

        menu.add\_command(label="Открыть", command=open\_item)

        menu.add\_command(label="Переименовать", command=rename\_item)

        menu.add\_command(label="Копировать путь", command=copy\_path)

        menu.add\_command(label="Создать копию", command=lambda: shutil.copy(item\_path, item\_path + "\_копия"))

        menu.add\_separator()

        menu.add\_command(label="Изменить права", command=lambda: change\_permissions(item\_path))

        menu.add\_command(label="Удалить", command=lambda: delete\_item(permanent=False))

        menu.add\_command(label="Удалить навсегда", command=lambda: delete\_item(permanent=True))

        menu.post(event.x\_root, event.y\_root)

def create\_main\_window():

    global root, tree, path\_var

    root = tk.Tk()

    root.title("Файловый менеджер")

    path\_var = tk.StringVar(value=os.getcwd())

    search\_var = tk.StringVar()

    # Верхняя панель

    top\_frame = tk.Frame(root)

    top\_frame.pack(fill='x', padx=5, pady=5)

    path\_label = tk.Entry(top\_frame, textvariable=path\_var, state='readonly', width=80)

    path\_label.pack(side='left', fill='x', expand=True, padx=5)

    up\_button = tk.Button(top\_frame, text="⬆", command=lambda: go\_up(tree, path\_var, search\_var))

    up\_button.pack(side='right')

    # Поисковая панель

    search\_frame = tk.Frame(root)

    search\_frame.pack(fill='x', padx=5, pady=5)

    search\_entry = tk.Entry(search\_frame, textvariable=search\_var, width=80)

    search\_entry.pack(side='left', fill='x', expand=True, padx=5)

    search\_entry.bind("<Return>", lambda e: on\_search(e, tree, path\_var, search\_var, clear\_button))

    clear\_button = tk.Button(search\_frame, text="X", command=lambda: clear\_search(tree, path\_var, search\_var, clear\_button))

    # Таблица с файлами

    columns = ('Имя', 'Расширение', 'Размер', 'Дата изменения')

    tree = ttk.Treeview(root, columns=columns, show='headings', height=20)

    tree.pack(fill='both', expand=True, padx=5, pady=5)

    for col in columns:

        tree.heading(col, text=col)

        tree.column(col, anchor='w')

    tree.bind("<Double-1>", lambda e: on\_item\_double\_click(e, tree, path\_var, search\_var))

    tree.bind("<Button-3>", lambda e: on\_right\_click(e, tree, path\_var))

    # Начальное обновление списка файлов

    update\_file\_list(tree, path\_var)

    root.mainloop()

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

    create\_main\_window()

Ссылка на проект на GitHub: <https://github.com/S-punk/lab_7.git>

**Вывод:** в ходе работы был программно реализован «Файловый менеджер». Было создано техническое задание, а также справочная документация.