ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А.Н. Тихонова

Руководство разработчика к информационно-аналитическому приложению

студентов группы БИВ182
Брагина Никиты Евгеньевича
Черновой Анастасии Александровны
Самойлова Андрея Владимировича

научный руководитель
Поляков Константин Львович

МОСКВА 2019

Содержание

Содержание 2
Назначение и условия применения программы 3
Характеристика программы Coffee Shop Chain.....3
Требования к характеристикам 3
Архитектура приложения 4
Листинг скрипта.....4

Назначение и условия применения программы

Информационная система Корпоративного Хранения Данных (ИС-КХД) предназначена для оптимизации технологии принятия тактических и стратегических управленческих решений конечными бизнес-пользователями на основе информации о всех аспектах закупочной деятельности Компании.

ИС-КХД предоставляет возможность работы с отчетностью.

При работе с отчетностью используется инструмент Coffee Shop Chain, который предоставляет следующие возможности:

- формирование табличных и кросс-табличных отчетов;
- экспорт и импорт результатов анализа;
- составление отчётности

Состав технических и программных средств требования, а также условия организационного, технического и технологического характера содержится в документе руководства пользователя

Характеристика программы Coffee Shop Chain

Coffee Shop Chain в составе ИС-КХД предназначен для автоматизации подготовки, настройки отчетных форм по показателям деятельности, а также для углубленного исследования данных на основе корпоративной информации хранилища данных.

Требования к характеристикам

- Microsoft Windows
- Стандартные решения Windows (EXCEL, 7zip)
- Python (3.7 и выше) с библиотеками pandas, tkinter, matplotlib

Архитектура приложения

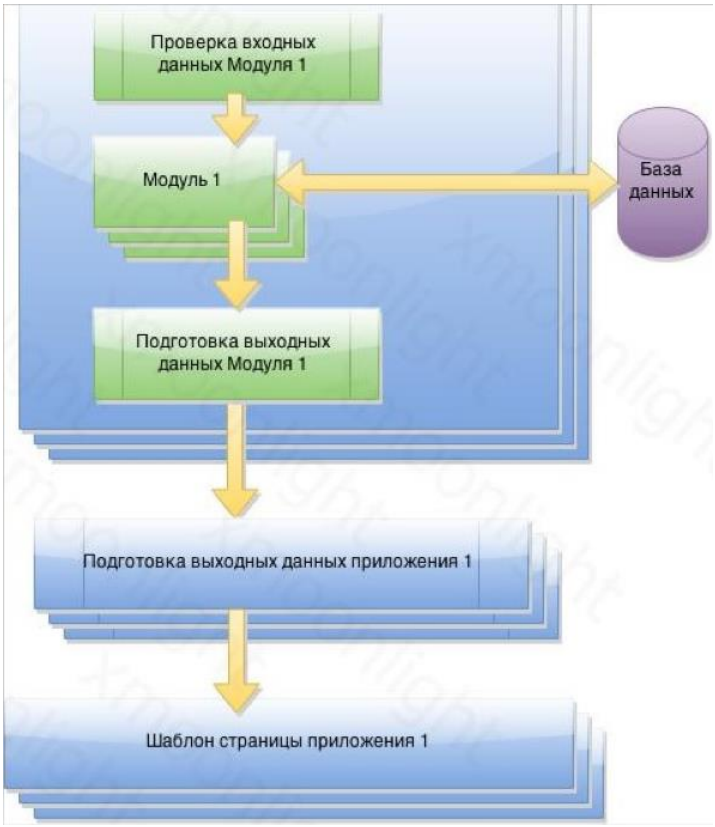

Рис. 1

Листинг скрипта

```
import pandas as pd
import tkinter as tki
from tkinter import filedialog
from tkinter import messagebox as mb
from pandas import DataFrame
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# ! /usr/bin/env python
# -*- coding: windows-1251 -*-

# списки базы данных
surfs_db = pd.DataFrame()
```

```
post_db = None
coffee_db = None
# пути к спискам баз данных
surfs_path = None
post_path = None
coffee_path = None
# основные параметры приложения, импортируемые из config.txt
style = None
size = None
btn_color = None
btn_go_back_color = None
options = None


def main():
    """
    Создает главное окно с основным итерфесом и импортирует config
    Автор: Брагин Никита БИВ182
    """

    root = tki.Tk()

    # импорт параметров приложения из config.txt
    curr_style = tki.StringVar()
    curr_style.set('empty')
    curr_size = tki.StringVar()
    curr_size.set('empty')
    curr_btn_color = tki.StringVar()
    curr_btn_color.set('empty')
    curr_btn_go_back_color = tki.StringVar()
    curr_btn_go_back_color.set('empty')
    curr_options = tki.StringVar()
    curr_options.set('empty')

    config = open('config.txt')
    for line in config:
```

```
            line = line[:-1]
            if line[0] == '!':
                pass
            elif curr_style.get() == 'empty':
                curr_style.set(line)
            elif curr_size.get() == 'empty':
                curr_size.set(line)
            elif curr_btn_color.get() == 'empty':
                curr_btn_color.set(line)
            elif curr_btn_go_back_color.get() == 'empty':
                curr_btn_go_back_color.set(line)
            elif curr_options.get() == 'empty':
                curr_options.set(line)

        # присваивание конфига глобальным переменным
        global style, size, btn_color, btn_go_back_color, options
        style = curr_style.get()
        size = curr_size.get()
        btn_color = curr_btn_color.get()
        btn_go_back_color = curr_btn_go_back_color.get()
        options = curr_options.get()

        # создание главного окна приложения
        root.title("База данных кофеен")
        root.geometry(options)
        main_interface(root)
        root.mainloop()


    def main_interface(root):
        """
        Создает интерфейс в основном окне приложения
        :param root: главное окно приложения
        Автор: Брагин Никита БИВ182
        """
```

```
    # создание кнопок для навигации в основном окне приложения
    btn_enter_bd = tki.Button(root, text="Внести базу данных", font=(style, size, 'bold'),
bg=btn_color,
                              fg='white', command=lambda: enter_bd(root))
    btn_show_bd = tki.Button(root, text="Показать базу данных", font=(style, size, 'bold'),
bg=btn_color,
                             fg='white', command=lambda: show_bd(root))
    btn_report_bd = tki.Button(root, text="Построить очет", font=(style, size, 'bold'),
bg=btn_color,
                               fg='white', command=lambda: report_bd(root))

    # размещение кнопок
    btn_enter_bd.grid(row=0, column=0, pady=10, padx=475)
    btn_show_bd.grid(row=1, column=0, pady=10, padx=475)
    btn_report_bd.grid(row=2, column=0, pady=10, padx=475)


def enter_bd(root):
    """
    Окно для прописывания пути к спискам базы данных
    :param root: главное окно приложения
    """

    # очистка окна приложения
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    # создание кнопок для прописывания пути к спискаам базы данных
    btn_path_1 = tki.Button(root, text="Путь к кофейням", font=(style, size, 'bold'),
bg=btn_color,
                            fg='white', command=lambda: path_surfs(root))
    btn_path_2 = tki.Button(root, text="Путь к поставщикам", font=(style, size, 'bold'),
bg=btn_color,
                            fg='white', command=lambda: path_post(root))
    btn_path_3 = tki.Button(root, text="Путь к сортам кофе", font=(style, size, 'bold'),
```

```
bg=btn_color,
                    fg='white', command=lambda: path_coffee(root))
    btn_go_back = tki.Button(root, text="Назад",  font=(style, size, 'bold'),
bg=btn_go_back_color,
                     fg='white', command=lambda: go_back(root))

    # размещение кнопок
    btn_path_1.grid(row=0, column=0, pady=10, padx=475)
    btn_path_2.grid(row=1, column=0, pady=10, padx=475)
    btn_path_3.grid(row=2, column=0, pady=10, padx=475)
    btn_go_back.grid(row=3, column=0, pady=10, padx=475)


def path_surfs(root):
    """
    Задает путь к списку кофеен
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # запрос пути к списку кофеен
    file = filedialog.askopenfile(parent=root, mode='rb', title='Выберите файл')
    global surfs_db, surfs_path
    surfs_path = file.name
    surfs_db = pd.read_csv(file, delimiter=";",  sep='\s*, \s', encoding="windows-1251")
    file.close()


def path_post(root):
    """
    Задает путь к списку поставщиков
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # запрос пути к списку поставщиков
```

```
    file = filedialog.askopenfile(parent=root, mode='rb', title='Выберите файл')
    global post_db, post_path
    post_db = pd.read_csv(file, delimiter=";", sep='\s*, \s', encoding="windows-1251")
    post_path = file.name
    file.close()


def path_coffee(root):
    """
    Задает путь к списку сортов кофе
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # запрос пути к списку сортов кофе
    file = filedialog.askopenfile(parent=root, mode='rb', title='Выберите файл')
    global coffee_db, coffee_path
    coffee_db = pd.read_csv(file, delimiter=";", sep='\s*, \s', encoding="windows-1251")
    coffee_path = file.name
    file.close()


def show_bd(root):
    """
    Окно для вывода списков базы данных
    :param root: главное окно приложения
    """

    # провеерка на пустоту списков базы данных
    if surfs_db is None or post_db is None or coffee_db is None:
        mb.showerror("Ошибка", "Сначала укажите пути к файлам")
    else:
        list1 = root.grid_slaves()
        for l in list1:
            l.destroy()
```

```
    # кнопски для выаода списков
    btn_list_1 = tki.Button(root, text="Кофейни",  font=(style, size, 'bold'), bg=btn_color,
fg='white',
                            command=lambda: button_surfs(root))
    btn_list_2 = tki.Button(root, text="Поставщики",  font=(style, size, 'bold'),
bg=btn_color, fg='white',
                            command=lambda: button_post(root))
    btn_list_3 = tki.Button(root, text="Сорта кофе", font=(style, size, 'bold'), bg=btn_color,
fg='white',
                            command=lambda: button_coffee(root))
    # кнопка для возвращения в основное окно приложения
    btn_go_back = tki.Button(root, text="Назад",  font=(style, size, 'bold'),
bg=btn_go_back_color,
                             fg='white', command=lambda: go_back(root))
    # кнопка для записи кортежа данных в базу данных
    btn_write = tki.Button(root, text="Добавить данные",  font=(style, size, 'bold'),
bg=btn_color,
                           fg='white', command=lambda: write(root))
    # кнопка для сохранения базы данных
    btn_save = tki.Button(root, text="Сохранить",  font=(style, size, 'bold'),
bg=btn_go_back_color,
                          fg='white', command=lambda: save())
    # кнопка для удаления кортежа по ключу
    btn_delete = tki.Button(root, text="Удалить данные", font=(style, size, 'bold'),
bg=btn_go_back_color,
                            fg='white', command=lambda: delete(root))

    btn_list_1.grid(row=0, column=1, pady=0, padx=10, sticky="w")
    btn_list_2.grid(row=0, column=2, pady=0, padx=10, sticky="w")
    btn_list_3.grid(row=0, column=3, pady=0, padx=10, sticky="w")
    btn_write.grid(row=0, column=4, pady=0, padx=10, sticky="w")
    btn_delete.grid(row=0, column=5, pady=0, padx=10, sticky="w")
    btn_save.grid(row=0, column=6, pady=0, padx=10, sticky="w")
    btn_go_back.grid(row=0, column=7, pady=0, padx=10, sticky="w")
```

```
def go_back(root):
    """
    Функция кнопки "назад" для возвращения в основное окно приложения
    :param root: главное окно приложения
    """

    # очистка окна и последующее построение интерфейса основного окна
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    main_interface(root)


def go_back_to_show_bd(root):
    """
    Функция кнопки "назад" для возвращения в окно вывода списков базы данных
    :param root: главное окно приложения
    """

    # очистка окна и последующее построение интерфейса окна вывода списков базы
данных
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    show_bd(root)


def button_surfs(root):
    """
    Создает кнопку для вывода словря базы данных в окне для вывода
    :param root: главное окно пприложения
    Автор: Брагин Никита БИВ182
    """

    global surfs_db
```

```
    # очистка окна и последующее построение интерфейса окна вывода списков базы
данных
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    show_bd(root)

    # вывод названия колонок в списке кофеен
    columns_list = list(surfs_db.columns.values)
    length = len(columns_list)
    for i in range(0, length):
        current_column = columns_list[i]
        current_column_label = tki.Label(root, text=current_column, font=(style, size, 'bold'))
        current_column_label.grid(row=1, column=1+i, pady=0, padx=10, sticky="w")

    # вывод данных из списка кофеен
    df = pd.DataFrame(data=surfs_db)
    number_rows = len(df.index)
    number_columns = len(df.columns)
    for i in range(0, number_rows):
        for j in range(0, number_columns):
            current = df.iat[i, j]
            current_label = tki.Label(root, text=current, font=(style, size))
            current_label.grid(row=2+i, column=1+j, pady=0, padx=10, sticky="w")


def button_post(root):
    """
    Создает кнопку для вывода словря базы данных окне для вывода
    :param root: главное окно пприложения
    Автор: Брагин Никита БИВ182
    """

    global post_db

    # очистка окна и последующее построение интерфейса окна вывода списков базы
```

данных
```
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    show_bd(root)

    # вывод названия колонок в списке поставщиков
    columns_list = list(post_db.columns.values)
    length = len(columns_list)
    for i in range(0, length):
        current_column = columns_list[i]
        current_column_label = tki.Label(root, text=current_column, font=(style, size, 'bold'))
        current_column_label.grid(row=1, column=1 + i, pady=0, padx=10, sticky="w")

    # вывод данных из списка поставщиков
    df = pd.DataFrame(data=post_db)
    number_rows = len(df.index)
    number_columns = len(df.columns)
    for i in range(0, number_rows):
        for j in range(0, number_columns):
            current = df.iat[i, j]
            current_label = tki.Label(root, text=current, font=(style, size))
            current_label.grid(row=2 + i, column=1 + j, pady=0, padx=10, sticky="w")


def button_coffee(root):
    """
    Создает кнопку для вывода словря базы данных в окне для вывода
    :param root: главное окно пприложения
    Автор: Брагин Никита БИВ182
    """

    global coffee_db

    # очистка окна и последующее построение интерфейса окна вывода списков базы
данных
```

```
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    show_bd(root)

    # вывод названия колонок в списке сортов кофе
    columns_list = list(coffee_db.columns.values)
    length = len(columns_list)
    for i in range(0, length):
        current_column = columns_list[i]
        current_column_label = tki.Label(root, text=current_column, font=(style, size, 'bold'))
        current_column_label.grid(row=1, column=1 + i, pady=0, padx=10, sticky="w")

    # вывод данных из списка сортов кофе
    df = pd.DataFrame(data=coffee_db)
    number_rows = len(df.index)
    number_columns = len(df.columns)
    for i in range(0, number_rows):
        for j in range(0, number_columns):
            current = df.iat[i, j]
            current_label = tki.Label(root, text=current, font=(style, size))
            current_label.grid(row=2 + i, column=1 + j, pady=0, padx=10, sticky="w")


def write(root):
    """
    Окно для записи картежа в базу данных
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # очистка окна и последующее построение интерфейса окна ввода кортежв в базу
данных
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
```

```
    label = tki.Label(root, text="Для добавления данных заполните поля:", font=(style, size))
    label.grid(row=0, column=0, columnspan=5, pady=0, padx=0)

    # поля ввода данных кортежа
    v = tki.StringVar()
    label = tki.Label(root, text="Кофейня:", font=(style, size))
    label.grid(row=1, column=0, columnspan=3, pady=0, padx=0, sticky="e")
    e = tki.Entry(master=None, textvariable=v)
    e.grid(row=1, column=3, pady=0, padx=20, sticky="w")

    v1 = tki.StringVar()
    label = tki.Label(root, text="Управляющий:", font=(style, size))
    label.grid(row=2, column=0, columnspan=3, pady=0, padx=0, sticky="e")
    e = tki.Entry(master=None, textvariable=v1)
    e.grid(row=2, column=3, pady=0, padx=20, sticky="w")

    v2 = tki.StringVar()
    label = tki.Label(root, text="Телефон кофейни:", font=(style, size))
    label.grid(row=3, column=0, columnspan=3, pady=0, padx=0, sticky="e")
    e = tki.Entry(master=None, textvariable=v2)
    e.grid(row=3, column=3, pady=0, padx=20, sticky="w")

    v3 = tki.StringVar()
    label = tki.Label(root, text="Кг в месяц:", font=(style, size))
    label.grid(row=4, column=0, columnspan=3, pady=0, padx=0, sticky="e")
    e = tki.Entry(master=None, textvariable=v3)
    e.grid(row=4, column=3, pady=0, padx=20, sticky="w")

    v4 = tki.StringVar()
    label = tki.Label(root, text="Поставщик:", font=(style, size))
    label.grid(row=5, column=0, columnspan=3, pady=0, padx=0, sticky="e")
    e = tki.Entry(master=None, textvariable=v4)
    e.grid(row=5, column=3, pady=0, padx=20, sticky="w")
```

```
v5 = tki.StringVar()
label = tki.Label(root, text="Телефон постащика:", font=(style, size))
label.grid(row=6, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v5)
e.grid(row=6, column=3, pady=0, padx=20, sticky="w")

v6 = tki.StringVar()
label = tki.Label(root, text="Сорт кофе:", font=(style, size))
label.grid(row=7, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v6)
e.grid(row=7, column=3, pady=0, padx=20, sticky="w")

v7 = tki.StringVar()
label = tki.Label(root, text="Наценка:", font=(style, size))
label.grid(row=8, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v7)
e.grid(row=8, column=3, pady=0, padx=20, sticky="w")

v8 = tki.StringVar()
label = tki.Label(root, text="Цена за кг:", font=(style, size))
label.grid(row=9, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v8)
e.grid(row=9, column=3, pady=0, padx=20, sticky="w")

v9 = tki.StringVar()
label = tki.Label(root, text="Высота сбора:", font=(style, size))
label.grid(row=10, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v9)
e.grid(row=10, column=3, pady=0, padx=20, sticky="w")

v10 = tki.StringVar()
label = tki.Label(root, text="Страна производитель:", font=(style, size))
label.grid(row=11, column=0, columnspan=3, pady=0, padx=0, sticky="e")
e = tki.Entry(master=None, textvariable=v10)
e.grid(row=11, column=3, pady=0, padx=20, sticky="w")
```

```
    # кнопка для записи данных кортежа в дазу данных
    btn_write = tki.Button(root, text="Записать",  font=(style, size, 'bold'), bg=btn_color,
fg='white',
                           command=lambda: write_check(v, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10))
    btn_write.grid(row=12, column=0, columnspan=3, pady=5, padx=20)

    # кнопка для возвращения назад в основное окно приложения
    go_back_to_show = tki.Button(root, text="Назад",  font=(style, size, 'bold'),
bg=btn_go_back_color, fg='white',
                                 command=lambda: go_back_to_show_bd(root))
    go_back_to_show.grid(row=13, column=0, columnspan=3, pady=5, padx=20)

    root.mainloop()


def write_check(v, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10):
    """
    Функция записи картежа в базу данных
    :param v: поле "кофейня"
    :param v1: поле "управляющий"
    :param v2: поле "телефон кофейни"
    :param v3: поле "кг в месяц"
    :param v4: поле "поставщик"
    :param v5: поле "телефон поставщика"
    :param v6: поле "сорт кофе"
    :param v7: поле "наценка"
    :param v8: поле "цена за кг"
    :param v9: поле "высота сбора"
    :param v10: поле "страна производитель"
    Автор: Брагин Никита БИВ182
    """
    global surfs_db, post_db, coffee_db

    # чтение данных кортежа, введенного пользователем
    coffee_place = v.get()
    manager = v1.get()
```

```
    coffee_tel = v2.get()
    coffee_amount = v3.get()
    post = v4.get()
    post_tel = v5.get()
    coffee_sort = v6.get()
    markup = v7.get()
    coffee_price = v8.get()
    coffee_height = v9.get()
    coffee_country = v10.get()

    # проверка ввода кортежа
    flag = True
    if (len(coffee_place) == 0 or len(manager) == 0 or len(coffee_tel) == 0 or
len(coffee_amount) == 0 or len(post) == 0
            or len(post_tel) == 0 or len(coffee_sort) == 0 or len(markup) == 0 or
len(coffee_price) == 0 or
            len(coffee_height) == 0 or len(coffee_country) == 0):
        flag = False
        mb.showerror("Ошибка", "Заполните все поля")
    else:
        try:
            x = int(coffee_tel)
            del x
        except ValueError:
            flag = False
            mb.showerror("Ошибка", "Укажите номера телефонов цифрами")
        if flag:
            try:
                x = int(coffee_amount)
                del x
            except ValueError:
                flag = False
                mb.showerror("Ошибка", "Укажите колличество кн в месяц цифрами")
            if flag:
                try:
                    x = int(markup)
```

```
            del x
        except ValueError:
            flag = False
            mb.showerror("Ошибка", "Укажите наценку цифрами")
        if flag:
            try:
                x = int(coffee_price)
                del x
            except ValueError:
                flag = False
                mb.showerror("Ошибка", "Укажите цену за кг цифрами")
            if flag:
                try:
                    x = int(coffee_height)
                    del x
                except ValueError:
                    flag = False
                    mb.showerror("Ошибка", "Укажите высоту сбора цифрами")

# ввод кортежа с проверкой на дублироание
if flag:
    flag_coffee = False
    flag_post = False
    flag_surf = False

    s1 = coffee_db.loc[coffee_db['сорт кофе'] == v6]
    if s1.empty:
        flag_coffee = True
    s1 = post_db.loc[post_db['поставщик'] == v4]
    s2 = post_db.loc[post_db['телефон_поставщика'] == v5]
    if s1.empty or s2.empty:
        flag_post = True
    s1 = surfs_db.loc[surfs_db['кофейня'] == v]
    s2 = surfs_db.loc[surfs_db['управляющий'] == v1]
    s3 = surfs_db.loc[surfs_db['телефон'] == v2]
    if s1.empty or s2.empty or s3.empty:
```

```
            flag_surf = True

        if flag_coffee and flag_post and flag_surf:
            mb.showerror("Ошибка", "Такая кофейня, такой поставщик и такой сорт кофе
уже сущесвуют")
        elif not flag_surf and flag_post and flag_coffee:
            cur_len = len(surfs_db.index)
            surfs_db.loc[cur_len + 1] = [coffee_place] + [manager] + [coffee_tel] +
[coffee_amount] + [post]
            mb.showinfo("Инфо", "Данные новой кофени успешно введены")
        elif not flag_surf and not flag_post and flag_coffee:
            cur_len = len(surfs_db.index)
            surfs_db.loc[cur_len + 1] = [coffee_place] + [manager] + [coffee_tel] +
[coffee_amount] + [post]
            cur_len = len(post_db.index)
            post_db.loc[cur_len + 1] = [post] + [post_tel] + [coffee_sort] + [markup]
            mb.showinfo("Инфо", "Данные новой кофейни и нового поставшика успешно
введены")
        elif not flag_surf and not flag_post and not flag_coffee:
            cur_len = len(surfs_db.index)
            surfs_db.loc[cur_len + 1] = [coffee_place] + [manager] + [coffee_tel] +
[coffee_amount] + [post]
            cur_len = len(post_db.index)
            post_db.loc[cur_len + 1] = [post] + [post_tel] + [coffee_sort] + [markup]
            cur_len = len(coffee_db.index)
            coffee_db.loc[cur_len + 1] = [coffee_sort] + [coffee_price] + [coffee_height] +
[coffee_country]
            mb.showinfo("Инфо", "Данные новой кофейни, нового поставщика и нового
сорта кофе успешно введены")
        elif flag_surf and not flag_post and flag_coffee:
            mb.showerror("Ошибка", "Если вы хотите изменить поставшика в кофейне, то
сначла удалите данные о кофейне")
        elif flag_surf and not flag_post and not flag_coffee:
            mb.showerror("Ошибка", "Если вы хотите изменить поставшика в кофейне и его
данные, то сначла удалите данные"
                         " о кофейне и о поставщике")
```

```
        elif flag_surf and flag_post and not flag_coffee:
            mb.showerror("Ошибка", "Если вы хотите изменить данные поставщика, то
сначала удалите их")
        elif not flag_surf and flag_post and not flag_coffee:
            mb.showerror("Ошибка", "Если вы хотите добавить новую кофейню и изменить
данные поставщика, то сначала"
                                   " удалите данные об этом поставщике")


def save():
    """
    Функция сохранения датафрейма в файл
    Автор: Брагин Никита БИВ182
    """

    surfs_db.to_csv(surfs_path, index=None, header=True, encoding="windows-1251",
sep=";")
    post_db.to_csv(post_path, index=None, header=True, encoding="windows-1251", sep=";")
    coffee_db.to_csv(coffee_path, index=None, header=True, encoding="windows-1251",
sep=";")
    mb.showinfo("Инфо", "Данные успешно сохранены")


def delete(root):
    """
    Функция удаления кортежа данных по ключу
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # очистка окна приложения
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    label = tki.Label(root, text="Введите ключ для удаления данных: телефон кофейни,
```

```
телефон поставщика или название"
                        " сорта кофе", font=(style, size))
    label.grid(row=0, column=0, columnspan=6, pady=0, padx=0, sticky="w")

    key = tki.StringVar()
    label = tki.Label(root, text="Ключ:", font=(style, size))
    label.grid(row=1, column=0, pady=0, padx=0, sticky="w")

    # создание поля для ввода ключа, по которому будет осуществляться удаление
    e = tki.Entry(master=None, textvariable=key)
    e.grid(row=1, column=1, pady=0, padx=20, sticky="w")

    # кнопка для удаления данных по ключу
    button_delete = tki.Button(root, text="Удалить", font=(style, size, 'bold'), bg=btn_color,
                        fg='white', command=lambda: delete_check(key))
    button_delete.grid(row=2, column=1, columnspan=2, pady=10, padx=350)

    go_back_to_show = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color, fg='white',
                        command=lambda: go_back_to_show_bd(root))
    go_back_to_show.grid(row=3, column=1, columnspan=3, pady=5, padx=20)


def delete_check(key):
    """
    Проверка удаления по ключу
    Автор: Брагин Никита БИВ182
    """

    flag_surf = False
    flag_post = False
    flag_coffee = False

    global surfs_db, post_db, coffee_db

    # чтнеие ключа, введенного пользователем
```

```
key = key.get()

# проверка ключа на существование в базе данных
try:
    key = int(key)
    s1 = surfs_db.loc[surfs_db['телефон'] == key]
    if s1.empty:
        s1 = post_db.loc[post_db['телефон_поставщика'] == key]
        if s1.empty:
            pass
        else:
            flag_post = True
    else:
        flag_surf = True
except ValueError:
    key_list = [key, ]
    s1 = coffee_db.loc[coffee_db['сорт_кофе'].isin(key_list)]
    if s1.empty:
        pass
    else:
        flag_coffee = True

main_db = surfs_db.merge(post_db, how='left', on='поставщик')
main_db = main_db.merge(coffee_db, how='left', on='сорт_кофе')

if not flag_surf and not flag_post and not flag_coffee:
    mb.showerror("Ошибка", "Ключ не найден")
elif flag_surf:
    main_db = main_db.loc[main_db['телефон'] != key]
elif flag_post:
    main_db = main_db.loc[main_db['телефон_поставщика'] != key]
elif flag_coffee:
    key_list_1 = [key, ]
    main_db = main_db.loc[~main_db['сорт_кофе'].isin(key_list_1)]

surfs_db = DataFrame(main_db, columns=['кофейня', 'управляющий', 'телефон',
```

```
'кг_в_месяц', 'поставщик'])
    main_db = main_db.drop(['кофейня', 'управляющий', 'телефон', 'кг_в_месяц'], axis=1)
    main_db = main_db.drop_duplicates(subset=['поставщик'], keep='first')

    post_db = DataFrame(main_db, columns=['поставщик', 'телефон_поставщика',
'сорт_кофе', 'наценка'])
    main_db = main_db.drop(['поставщик', 'телефон_поставщика', 'наценка'], axis=1)
    main_db = main_db.drop_duplicates(subset=['сорт_кофе'], keep='first')

    coffee_db = DataFrame(main_db, columns=['сорт_кофе', 'цена_за_кг', 'высота_сбора',
'страна_производитель'])


def report_bd(root):
    """
    Окно для создания отчетов
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # проверка на ввод списков базы данных
    if surfs_db is None or post_db is None or coffee_db is None:
        mb.showerror("Ошибка", "Сначала укажите пути к файлам")
    else:
        # очистка окна приложения
        list1 = root.grid_slaves()
        for l in list1:
            l.destroy()

        label = tki.Label(root, text="Отчеты:", font=('Times New Roman', 12))
        label.grid(row=0, column=0, pady=10, padx=350)
        btn_report_1 = tki.Button(root, text="Столбчатая диаграмма сорт кофе - цена",
font=(style, size, 'bold'),
                                  bg=btn_color, fg='white', command=lambda: report_1(root))
        btn_report_2 = tki.Button(root, text="Гистограмма наценки по сортам", font=(style,
size, 'bold'), bg=btn_color,
```

```
                        fg='white', command=lambda: report_2(root))
    btn_report_3 = tki.Button(root, text="Бокс-Вискер  по кг в месяц среди кофеен",
font=(style, size, 'bold'),
                        bg=btn_color, fg='white', command=lambda: report_3(root))
    btn_report_4 = tki.Button(root, text="Диаграмма рассеивания цен на кофе от высоты
его сбора",
                        font=(style, size, 'bold'), bg=btn_color, fg='white', command=lambda:
report_4(root))
    btn_go_back = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color,
                       fg='white', command=lambda: go_back(root))

    btn_report_1.grid(row=1, column=0, pady=10, padx=350)
    btn_report_2.grid(row=2, column=0, pady=10, padx=350)
    btn_report_3.grid(row=3, column=0, pady=10, padx=350)
    btn_report_4.grid(row=4, column=0, pady=10, padx=350)
    btn_go_back.grid(row=5, column=0, pady=10, padx=350)


def report_1(root):
    """
    Создает отчет "столбчатая диаграмма"
    поле "сорт кофе"
    поле "цена за килограмм"
    Автор: Анастасия Чернова БИВ182
    """
    global coffee_db

    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    df1 = DataFrame(coffee_db, columns=['сорт_кофе', 'цена_за_кг'])
    df1 = df1[['сорт_кофе', 'цена_за_кг']].groupby('сорт_кофе').sum()
    figure1 = plt.Figure(figsize=(8, 10), dpi=60)
    ax1 = figure1.add_subplot(111)
```

```
    bar1 = FigureCanvasTkAgg(figure1, root)
    bar1.get_tk_widget().grid(row=0, column=0)
    df1.plot(kind='bar', legend=True, ax=ax1)
    ax1.set_title('Соотношение цены за кг к сорту кофе')
    ax1.legend()
    ax1.set_xlabel('сорт кофе')

    btn_go_back_to_report = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color,
                                       fg='white', command=lambda: go_back_to_report(root))
    btn_go_back_to_report.grid(row=2, column=0, columnspan=3)


def report_2(root):
    """
    Создает отчет "Гистограмма наценки по сортам"
    поле "сорт кофе"
    поле "наценка"
    Автор: Анастасия Чернова БИВ182
    """
    global post_db

    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    curr_db = pd.DataFrame(post_db)
    df1 = curr_db.groupby(
        ['сорт_кофе']
    ).agg(
        {
            'наценка': "mean"})
    df1.rename(columns={'сорт_кофе': 'сорт_кофе', 'наценка': 'средняя_наценка'},
inplace=True)

    figure1 = plt.Figure(figsize=(8, 10), dpi=60)
```

```
    ax1 = figure1.add_subplot(111)
    bar1 = FigureCanvasTkAgg(figure1, root)
    bar1.get_tk_widget().grid(row=0, column=0)
    df1.plot(kind='hist', legend=True, ax=ax1)
    ax1.set_title('Гистограмма наценки по сортам')
    ax1.legend()
    ax1.set_xlabel('наценка')

    btn_go_back_to_report = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color,
                                       fg='white', command=lambda: go_back_to_report(root))
    btn_go_back_to_report.grid(row=2, column=0, columnspan=3)


def report_3(root):
    """
    Создает отчет "КГ в месяц среди кофеен"
    поле "кг в месц"
    поле "кофейня"
    Автор: Самойлов Андрей БИВ182
    """
    global surfs_db

    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    curr_db = pd.DataFrame(surfs_db)
    df1 = curr_db[['кофейня', 'кг_в_месяц']]
    figure1 = plt.Figure(figsize=(8, 10), dpi=60)
    ax1 = figure1.add_subplot(111)
    bar1 = FigureCanvasTkAgg(figure1, root)
    bar1.get_tk_widget().grid(row=0, column=0)
    df1.boxplot(ax=ax1)
    ax1.set_title('КГ в месяц среди кофеен')
```

```
    btn_go_back_to_report = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color,
                                       fg='white', command=lambda: go_back_to_report(root))
    btn_go_back_to_report.grid(row=2, column=0, columnspan=3)


def report_4(root):
    """
    Создает отчет "график зависимости цены кофе от высоты сбора"
    поле "кг в месц"
    поле "кофейня"
    Автор: Самойлов Андрей БИВ182
    """
    global coffee_db

    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    curr_db = pd.DataFrame(coffee_db)
    df1 = curr_db[['цена_за_кг', 'высота_сбора']]

    figure1 = plt.Figure(figsize=(8, 10), dpi=60)
    ax1 = figure1.add_subplot(111)
    bar1 = FigureCanvasTkAgg(figure1, root)
    bar1.get_tk_widget().grid(row=0, column=0)
    df1.plot(kind='scatter', x='цена_за_кг', y='высота_сбора', ax=ax1)
    ax1.set_title('Зависимость цены кофе от высоты сбора')

    btn_go_back_to_report = tki.Button(root, text="Назад", font=(style, size, 'bold'),
bg=btn_go_back_color,
                                       fg='white', command=lambda: go_back_to_report(root))
    btn_go_back_to_report.grid(row=2, column=0, columnspan=3)


def go_back_to_report(root):
```

```
    """
    Функция кнопки для возвращения в окно создания отчетов
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()

    report_bd(root)


def go_back_to_report_bd(root):
    """
    Функция кнопки для возвращения в окно вывода списков базы данных
    :param root: главное окно приложения
    Автор: Брагин Никита БИВ182
    """

    # очистка окна приложения и последующее построение интерфейса окна отчетов
    list1 = root.grid_slaves()
    for l in list1:
        l.destroy()
    report_bd(root)


main()
```