**Федеральное государственное автономное образовательное учреждение высшего образования**

**“Национальный исследовательский университет “Высшая школа экономики”**

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

Департамент компьютерной инженерии (или департамент электронной инженерии)

**ОП «Информатика и вычислительная техника»**

**Курс: Проектный семинар “Python в науке о данных”**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**для приложения «Рынок IT-вакансий»**

**Студенты:**

**Ратке Владислав Андреевич**

**(8 (910) 151-51-55, varatke@edu.hse.ru),**

Отливан Дмитрий Алексеевич,

Полевщиков Алексей Геннадьевич

**Группа:** БИВ223 (Бригада 6)

**Руководитель:**

Поляков Константин Львович

**МОСКВА 2023**

Оглавление

[Описание приложения 3](#_Toc137936045)

[Требования к компьютеру 4](#_Toc137936046)

[Описание архитектуры приложения 5](#_Toc137936047)

[Описание структуры каталогов 6](#_Toc137936048)

[Листинг основного кода 7](#_Toc137936049)

[Листинг дополнительного модуля 19](#_Toc137936050)

# Описание приложения

Разработанное приложение представляет собой сервис для поиска актуальных IT-вакансий в разных городах России. Оно предназначено для управления информацией о должностях, компаниях, городах, в которых находятся перечисленные компании, о требуемых навыках и о формате работы.

Функционал разработанного приложения включает в себя:

1. загрузка данных: пользователь может выбрать одну из трех таблиц (города, должности, вакансии) для загрузки в память программы;
2. гистограмма: при выборе этой опции пользователю предлагается выбрать столбец из загруженных данных. Затем приложение строит гистограмму, отображающую распределение значений в выбранном столбце;
3. диаграмма рассеивания: при выборе этой опции пользователю предлагается выбрать два столбца из загруженных данных. Приложение строит диаграмму рассеивания, которая показывает взаимосвязь между значениями этих двух столбцов;
4. круговая диаграмма: при выборе этой опции приложение создает круговую диаграмму, отображающую процентное распределение значений в определенном столбце таблицы;
5. показать таблицу: при выборе этой опции пользователь может выбрать столбцы из загруженных данных и просмотреть значения в виде таблицы. Данные отображаются в окне приложения, где пользователь может редактировать значения и сохранить изменения в формате Excel или CSV;
6. помощь: при выборе этой опции открывается окно с текстом справки, где пользователь может ознакомиться с инструкциями по использованию приложения;
7. цвет фона: приложение предлагает пользователю выбрать цвет фона окна между красным, белым и черным.

Таким образом, разработанное приложение обеспечивает удобное управление информацией.

# Требования к компьютеру

Требования к операционной системе: разработанное приложение подходит для операционных систем Windows, MacOS, Linux.

Ресурсы компьютера: разработанное приложение не требует особенных ресурсов, поэтому для его работы необходимы системные требования, подходящие для интерпретатора Python:

1. 2 и более ядерный процессор
2. 3 и более ГБ оперативной памяти

Требуемое место на диске: 1.5 Мб.

# Описание архитектуры приложения

Архитектура кода представляет собой простое графическое приложение для анализа данных. Вот общая структура кода:

1. Импорт необходимых модулей: numpy и pandas для работы с данными;   
   tkinter для создания графического интерфейса; matplotlib для построения графиков; os для работы с файловой системой.
2. Создание глобального объекта DB, который будет использоваться для хранения данных.
3. Определение функции rgb\_hack, которая используется для преобразования RGB-значений в цветовой код.
4. Определение класса Loads, содержащего методы для загрузки данных из разных источников.
5. Определение функции hst, которая создает гистограмму на основе выбранного столбца данных.
6. Определение функции sct, которая создает диаграмму рассеивания на основе выбранных столбцов данных.
7. Определение функции cercle, которая создает круговую диаграмму на основе данных о ключевых навыках.
8. Определение функции vak\_excel, которая сохраняет данные в формате Excel.
9. Определение функции vak\_csv, которая сохраняет данные в формате CSV.
10. Определение функции add\_data, которая позволяет добавить новую запись в базу данных.
11. Определение функции show, которая отображает выбранные столбцы данных в виде таблицы с возможностью редактирования значений.
12. Определение функции hlp, которая отображает справку о приложении.
13. Определение класса Colors, содержащего методы для изменения цвета фона приложения.
14. Создание основного окна приложения с заданными размерами и заголовком.
15. Создание основного меню приложения с несколькими категориями и подкатегориями.
16. Привязка функций к соответствующим пунктам меню.
17. Запуск основного цикла обработки событий приложения.

Такая архитектура позволяет пользователю загружать данные из разных источников, анализировать и визуализировать их с помощью графиков, а также сохранять и редактировать данные. Код разделен на отдельные функции и классы для повышения читаемости, модульности и повторного использования кода.

# Описание структуры каталогов

Размещается в стандартной структуре каталогов work:

1. Data — содержит базу данных.
2. Graphics — содержит копии графических отчетов.
3. Library —содержит 2 модуля: function.py и graph.py, использующихся при работе приложения.
4. Notes — содержит документацию, в нем размещается Руководства пользователя и разработчика.
5. Scripts — содержит главный файл приложения main.py и файл с определением параметров настройки приложения

# Листинг основного кода

main.py

# -\*- coding: utf-8 -\*-

"""

Графический анализ данных

pandas version 2.0.0

numpy version 1.24.2

tkinter version 0.3.1

"""

import tkinter as tki

from tkinter import ttk

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

os.chdir("C:/work/scripts")

DB = pd.DataFrame()

def rgb\_hack(rgb):

"""

Использование RGB формата для цветов

https://docs.python.org/3/library/stdtypes.html#old-string-formatting

"""

return "#%02x%02x%02x" % rgb

class Loads:

"""

Class for loading tables

"""

def load\_city():

"""

Загрузка данных

Returns

-------

DB - фрейм с данными

"""

global DB

DB = pd.read\_csv("C:/work/data/города.csv", delimiter=";")

DB = DB.head(25)

return 0

def load\_dolzh():

"""

Загрузка данных

Returns

-------

DB - фрейм с данными

"""

global DB

DB = pd.read\_csv("C:/work/data/должности.csv", delimiter=";")

DB = DB.head(25)

return 0

def load\_vac():

"""

Загрузка данных

Returns

-------

DB - фрейм с данными

"""

global DB

DB = pd.read\_csv("C:/work/data/овакансии1.csv", delimiter=";")

DB = DB.head(25)

return 0

def hst():

"""

Создание гистограммы

Returns

-------

None.

"""

def clck():

"""

Обработка нажатия кнопки

"""

t\_1 = DB[[cmb.get()]]

x\_1 = t\_1[t\_1.columns[0]].unique()

y\_1 = [len(DB[DB[t\_1.columns[0]] == i]) for i in x\_1]

fig, ax\_1 = plt.subplots()

ax\_1.bar(x\_1, y\_1)

plt.title("График зависимостей")

plt.xlabel(t\_1.columns[0])

plt.ylabel("Количество")

ax\_1.set\_facecolor('seashell')

fig.set\_facecolor('floralwhite')

fig.set\_figwidth(8)

fig.set\_figheight(4)

plt.show()

lbl.destroy()

cmb.destroy()

btn.destroy()

btn2.destroy()

lbl = tki.Label(text='Выбор показателя >', font=('Arial', 12, 'italic'),

bg='blue',fg='yellow')

lbl.grid(column=0, row=0)

cmb = ttk.Combobox(root)

global DB

cmb['values'] = tuple(DB.columns)

cmb.current(0) #Значение по умолчанию

cmb.focus()

cmb.grid(column=1, row=0)

btn = tki.Button(root, text='Построить', font=('Arial', 12, 'italic'),

bg='blue',fg='red', command=clck)

btn.grid(column=2, row=0)

def destroy1():

lbl.destroy()

cmb.destroy()

btn.destroy()

btn2.destroy()

btn2 = tki.Button(root, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn2.grid(column=3, row=0)

return 0

def sct():

"""

Создание диаграммы рассеивания

Returns

-------

None.

"""

def clck():

"""

Обработка нажатия кнопки

"""

from matplotlib.pyplot import scatter, show

w\_1 = lb\_1.curselection()

args = [lst[k] for k in w\_1]

scatter(DB[[args[0]]], DB[[args[1]]])

show()

lbl.destroy()

lb\_1.destroy()

btn.destroy()

btn2.destroy()

lbl = tki.Label(text='Выберите два показателя (x,y) >', font=('Arial', 12, 'italic'),

bg='blue',fg='yellow')

lbl.grid(column=0, row=0)

global DB

lst = list(DB.columns)

nln = len(lst)

lb\_1 = tki.Listbox(root, selectmode=tki.EXTENDED, height=nln, font="Times 14")

for i in lst:

lb\_1.insert(tki.END, i)

lb\_1.grid(column=1, row=0)

btn = tki.Button(root, text='Построить', font=('Arial', 12, 'italic'),

bg='blue',fg='red', command=clck)

btn.grid(column=2, row=0)

def destroy1():

lbl.destroy()

lb\_1.destroy()

btn.destroy()

btn2.destroy()

btn2 = tki.Button(root, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn2.grid(column=3, row=0)

return 0

def cercle():

"""

Returns

-------

TYPE

DESCRIPTION.

"""

global DB

if DB.columns[1] != "ЗАР\_ПЛАТ":

return 0

# данные

skills = DB["ТРЕБ\_НАВ"].unique()

data = [len(DB[DB["ТРЕБ\_НАВ"] == i]) for i in skills]

print(data)

# Расстояние м/у дольками

explode = (0.1, 0.0, 0.2, 0.3, 0.0, 0.1, 0.0, 0.2)

# Цвета долек

colors = ( "orange", "cyan", "brown",

"grey", "indigo", "blue", "black", "red", "yellow")

# Свойства "долек"

wp\_1 = { 'linewidth' : 1, 'edgecolor' : "green" }

# Создание аргументов autocpt

def func(pct, allvalues):

absolute = int(pct / 100.\*np.sum(allvalues))

return "{:.1f}%\n({:d})".format(pct, absolute)

# Создание графика

fig, ax\_1 = plt.subplots(figsize =(6, 5))

wedges, texts, autotexts = ax\_1.pie(data,

autopct = lambda pct: func(pct, data),

explode = explode,

labels = skills,

shadow = True,

colors = colors,

startangle = 90,

wedgeprops = wp\_1,

textprops = dict(color ="magenta"))

# Легенда

ax\_1.legend(wedges, skills,

title ="Навыки",

loc ="center left",

bbox\_to\_anchor =(1, 0, 0.5, 1))

plt.setp(autotexts, size = 7, weight ="bold")

ax\_1.set\_title("Ключевые навыки")

plt.show()

def box\_plt():

"""

Создание гистограммы

Returns

-------

None.

"""

def clck():

"""

Обработка нажатия кнопки

"""

import matplotlib.pyplot as plt

#from sklearn import preprocessing

t\_1 = DB[[cmb.get()]]

x\_1 = t\_1[t\_1.columns[0]].unique()

# Предобработка данных

data\_1 = []

data\_2 = []

for i in x\_1:

data\_1.append(i)

data\_2.append(len(t\_1[t\_1[t\_1.columns[0]] == i]))

#строим boxplot по количеству

plt.boxplot(data\_2, labels=['Количество'])

plt.title("box plot")

plt.show()

lbl = tki.Label(text='Выбор показателя >', font=('Arial', 12, 'italic'),

bg='blue',fg='yellow')

lbl.grid(column=0, row=0)

cmb = ttk.Combobox(root)

global DB

cmb['values'] = tuple(DB.columns)

cmb.current(0) #Значение по умолчанию

cmb.focus()

cmb.grid(column=1, row=0)

btn = tki.Button(root, text='Построить', font=('Arial', 12, 'italic'),

bg='blue',fg='red', command=clck)

btn.grid(column=2, row=0)

def destroy1():

lbl.destroy()

cmb.destroy()

btn.destroy()

btn2.destroy()

btn2 = tki.Button(root, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn2.grid(column=3, row=0)

return 0

F\_DESTROYING = False

def vak\_excel():

"""

Returns

-------

None.

"""

TMP.to\_excel("C:/work/output/new\_data.csv")

TOP.destroy()

BOTTOM.destroy()

def vak\_csv():

"""

Function for csv-file

"""

TMP.to\_csv("C:/work/output/new\_data.csv")

TOP.destroy()

BOTTOM.destroy()

def add\_data():

"""

Function for message

"""

def message():

values = entry.get().split(";")

DB.loc[len(DB.index)] = values

DB.to\_csv("C:/work/output/after\_changes.csv")

title.destroy()

cols.destroy()

f\_btn.destroy()

form.destroy()

TOP.destroy()

BOTTOM.destroy()

global F\_DESTROYING

F\_DESTROYING = True

form = tki.LabelFrame(root, bg='black', fg="yellow", font=('Arial', 14, 'bold'))

form.grid(column=2, row=0)

title = tki.Label(form, text='Введи данные через ;', font=('Arial', 14, 'bold'),

bg='black',fg='yellow')

title.grid(column=0,row=0)

atrib = " | ".join(DB.columns)

cols = tki.Label(form, text=atrib, font=('Arial', 12),

bg='black',fg='yellow')

cols.grid(column=0,row=1)

entry = tki.Entry(form, bg="black", fg="yellow", font=('Arial', 10))

entry.grid(column=0, row=2)

f\_btn = tki.Button(form, text='OK',

font=('Arial', 12, 'bold'), bg='yellow', fg='black',

command=message)

f\_btn.grid(column=0, row=3)

def destroy1():

BOTTOM.destroy()

TOP.destroy()

btn\_d.destroy()

form.destroy()

btn\_d = tki.Button(form, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn\_d.grid(column=0, row=4)

def show():

"""

Funstion for click

"""

def clck():

"""

Обработка нажатия кнопки

"""

w\_1 = lb\_1.curselection()

args = [lst[k] for k in w\_1]

lbl.destroy()

lb\_1.destroy()

btn.destroy()

btn2.destroy()

global TMP

TMP = DB[args]

height = TMP.shape[0]

width = TMP.shape[1]

# Формируем массив указателей на виджеты Entry

pnt = np.empty(shape=(height, width), dtype="O")

# Формируем массив указателей на текстовые буферы для передачи данных Tcl/Tk

vrs = np.empty(shape=(height, width), dtype="O")

# Построение изображения

# До любых обращений к tkinter

# Создаем фрейм (контейнер) в котором будет размещена таблица

# https://www.tutorialspoint.com/python/tk\_labelframe.htm

global TOP

TOP = tki.LabelFrame(root, text="Справочник",

bg=rgb\_hack((0, 255, 255)))

TOP.grid(column=0, row=0)

global BOTTOM

BOTTOM = tki.LabelFrame(root, text="Управление",

bg=rgb\_hack((0, 255, 125)))

BOTTOM.grid(column=1, row=0, sticky="w")

# Инициализация указателей на буферы

for i in range(height):

for j in range(width):

vrs[i, j] = tki.StringVar()

# Построение таблицы

for i in range(height):

for j in range(width):

pnt[i, j] = tki.Entry(TOP, textvariable = vrs[i, j])

pnt[i, j].grid(row=i, column=j)

# Заполнение таблицы значениями

for i in range(height):

for j in range(width):

cnt = TMP.iloc[i, j]

vrs[i, j].set(str(cnt))

# ----------------------------------------------------------------

btn\_1 = tki.Button(BOTTOM, text='Сохранить в Excel',

font=('Arial', 12, 'italic'), bg='purple', fg='white',

command=vak\_excel)

btn\_1.grid(column=0, row=0, sticky="w")

btn\_2 = tki.Button(BOTTOM, text='Сохранить в CSV',

font=('Arial', 12, 'italic'), bg='blue', fg='black',

command=vak\_csv)

btn\_2.grid(column=0, row=1, sticky="w")

btn\_3 = tki.Button(BOTTOM, text='Добавить запись',

font=('Arial', 12, 'italic'), bg='yellow', fg='blue',

command=add\_data)

btn\_3.grid(column=0, row=2, sticky="w")

def destroy1():

btn\_1.destroy()

btn\_2.destroy()

btn\_3.destroy()

BOTTOM.destroy()

TOP.destroy()

lbl.destroy()

lb\_1.destroy()

btn.destroy()

btn\_d.destroy()

btn\_d = tki.Button(BOTTOM, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn\_d.grid(column=0, row=3)

lbl = tki.Label(text='Выберите столбцы >', font=('Arial', 14, 'italic'),

bg='black',fg='yellow')

lbl.grid(column=0, row=0)

lst = list(DB.columns)

nln = len(lst)

lb\_1 = tki.Listbox(root, selectmode=tki.EXTENDED, height=nln, font="Times 14")

for i in lst:

lb\_1.insert(tki.END, i)

lb\_1.grid(column=1, row=0)

btn = tki.Button(root, text='Показать', font=('Arial', 12, 'italic'),

bg='blue',fg='red', command=clck)

btn.grid(column=2, row=0)

def destroy1():

lbl.destroy()

lb\_1.destroy()

btn.destroy()

btn2.destroy()

btn2 = tki.Button(root, text='Очистить поле', font=('Arial', 12, 'italic'),

bg='black',fg='yellow', command=destroy1)

btn2.grid(column=3, row=0)

def hlp():

"""

Помощь

Returns

-------

None.

"""

def clck():

"""

Обработка нажатия кнопки

"""

text.destroy()

scrolly.destroy()

scrollx.destroy()

btn.destroy()

text = tki.Text(width=20, height=7, bg="blue",

fg='white', wrap = tki.WORD)

text.grid(row = 0, column = 0)

scrolly = tki.Scrollbar(command=text.yview)

scrolly.grid(row = 0, column = 1)

scrollx = tki.Scrollbar(orient=tki.HORIZONTAL, command=text.xview)

scrollx.grid(row = 1, column = 0)

text.config(yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

with open("C:/work/data/text\_h.txt", "r", encoding='utf-8') as fln:

txt = fln.read()

text.insert(tki.END, txt)

btn = tki.Button(root, text='Ок', font=('Arial', 12, 'italic'),

bg='blue',fg='red', command=clck)

btn.grid(row = 2, column = 0)

class Colors():

"""

Class with functions for colors

"""

def red():

"""

Makes window red

"""

root["bg"] = "red"

def white():

"""

Makes window white

"""

root["bg"] = "white"

def black():

"""

Makes window black

"""

root["bg"] = "black"

root = tki.Tk()

root.geometry('900x650+30+30')

root.title("Анализ рынка вакансий IT")

# Создание меню

mainmenu = tki.Menu(root, tearoff=0)

dan = tki.Menu(mainmenu, tearoff=0)

dan.add\_command(label="Гистограмма", command = hst)

dan.add\_command(label="Диаграмма рассеивания", command = sct)

dan.add\_command(label="Круговая диаграмма", command = cercle)

dan.add\_command(label="Ящик с усамми", command = box\_plt)

dan.add\_command(label="Показать таблицу", command = show)

tables = tki.Menu(mainmenu, tearoff=0)

tables.add\_command(label="Города", command = Loads.load\_city)

tables.add\_command(label="Должности", command = Loads.load\_dolzh)

tables.add\_command(label="О вакансии", command = Loads.load\_vac)

working = tki.Menu(mainmenu, tearoff=0)

working.add\_cascade(label="Загрузка таблицы", menu=tables)

working.add\_command(label="Завершить", command = root.destroy)

colors = tki.Menu(mainmenu, tearoff=0)

colors.add\_command(label="Красный", command = Colors.red)

colors.add\_command(label="Белый", command = Colors.white)

colors.add\_command(label="Черный", command = Colors.black)

# Размещение пунктов в меню

mainmenu.add\_cascade(label="Файл", menu=working)

mainmenu.add\_cascade(label="Анализ данных", menu = dan)

mainmenu.add\_command(label="Помощь", command = hlp)

mainmenu.add\_cascade(label="Фон", menu=colors)

mainmenu.add\_command(label="Выход", command = root.destroy)

root.config(menu=mainmenu)

root.mainloop()

# Листинг дополнительного модуля

merging.py

"""

Денормализация таблиц и создание сводной таблицы

pandas version 2.0.0

Автор: Отливан Дмитрий

"""

import pandas as pd

df\_AboutVacancy = pd.read\_pickle("/work/data/VACS.pkl")

df\_Cities = pd.read\_pickle("/work/data/City.pkl")

df\_Dolzh = pd.read\_pickle("/work/data/Dolzh.pkl")

def denormalization(df\_AboutVacancy, df\_Cities, df\_Dolzh):

'''

Денормализует таблицы Города, О вакансии, Должности в одну общую

Вохвращает общую таблицу SUMMARY

'''

SUMMARY = pd.merge(df\_Dolzh, df\_Cities, on = "КОМП")

SUMMARY = pd.concat([SUMMARY, df\_AboutVacancy], axis=1, join="inner")

return SUMMARY

def report(df\_AboutVacancy, df\_Cities, df\_Dolzh):

'''

Составляет сводную таблицу по средним зарплатам в различных городах по должностям

Возвращает эту таблицу NUMREP

'''

SUMMARY = pd.merge(df\_Dolzh, df\_Cities, on = "КОМП")

SUMMARY = pd.concat([SUMMARY, df\_AboutVacancy], axis=1, join="inner")

NUMREP = pd.pivot\_table(SUMMARY, index="ДОЛЖН", columns="ГОР", values="ЗАР\_ПЛАТ", aggfunc=["mean"])

return NUMREP