Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Факультет Информационные системы в управлении

Кафедра Информационная безопасность

Направление Информационная безопасность

Профиль Безопасность автоматизированных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине: «Языки программирования»

Тема проекта «Разработка приложения по анализу уязвимостей ОС Centos по данным банка ФСТЭК на ЯП python»

Выполнил:

студент группы БИб-21И1Z2 Лопаткин Алексей Николаевич

Курсовой проект сдан на проверку:

«\_\_»\_\_\_\_\_\_\_\_2022 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Лопаткин

(подпись студента)

Курсовой проект допущен к защите:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Руководитель проекта

доцент кафедры, к.п.н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

(подпись преподавателя)

Курсовой проект защищен с оценкой:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Руководитель проекта

доцент кафедры, к.п.н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

(подпись преподавателя)

Омск 2022

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Кафедра «Информационная безопасность»

УТВЕРЖДАЮ

Зав. кафедрой ИБ, д.п.н., проф.

\_\_\_\_\_\_\_\_\_\_\_\_\_ З.В. Семенова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**ЗАДАНИЕ**

**к курсовому проекту**

студенту группы БИб-21ИZ2 Лопаткин Алексей Николаевич

1. Тема курсового проекта: «Разработка приложения по анализу уязвимостей по данным банка ФСТЭК на языке программирования python»
2. Исходные данные к курсовому проекту:

* документация по языку программирования Python;
* требования к функционалу приложения:
  1. загрузка данных банка угроз ФСТЭК из документа MS Excel;
  2. анализ данных о распределении количества уязвимостей по годам   
     и уровням опасности за указанный пользователем период;
  3. выгрузка отчета в документ MS Word.

1. Содержание пояснительной записки:

* титульный лист;
* задание к курсовому проекту;
* содержание;
* введение;
* теоретические основы разработки приложений на языке программирования Python;
* проектирование интерфейса и реализация функционала приложения на Python;
* заключение;
* список использованных источников;
* приложения.

1. Перечень демонстрационного материала для сопровождения доклада при защите курсового проекта:

* презентация Microsoft PowerPoint.

1. Срок сдачи проекта: « \_\_ » \_\_\_\_ 2022 г.

Задание выдано « 01 » февраля 2022 г.

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

(подпись преподавателя)

Задание к исполнению принял « 10 » февраля 2020 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Лопаткин

(подпись студента)

Оглавление

[**Введение** 4](#_Toc107251379)

[**1** **ОСНОВЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ НА Python** 5](#_Toc107251380)

[1.1 Инструментарий для разработки приложений на Python 5](#_Toc107251381)

[1.2 Инструментарий для создания пользовательского интерфейса 5](#_Toc107251382)

[1.3 Настройка среды разработки 5](#_Toc107251383)

[Рисунок 7 – Завершение установки 11](#_Toc107251384)

[10. Если вы только учите язык Python и установили PyCharm впервые, тогда выберите пункт «I do not have a previous version of PyCharm or I do not want to import my settings» и «ОК». 11](#_Toc107251385)

[11. PyCharm запустится. Первый раз этой займет больше времени. 11](#_Toc107251386)

[12. После приветственного окна нажмите «Create New Project» 11](#_Toc107251387)

[13. Выберите директорию для сохранения проекта. Во втором поле нужно указать местоположение интерпретатора Python (который уже должен быть установлен). Обычно PyCharm находит его самостоятельно. Если не удалось, нужно указать путь к нему и нажать «Create» (Рисунок 8). 11](#_Toc107251388)

[Рисунок 8 – Выбор пути установки 11](#_Toc107251389)

[14. Нажмите «File > New > Python File». Введите название нового файла. Он откроется с чистой консолью. Можно начинать писать программу. 11](#_Toc107251390)

[2.1 Инструменты для создания интерфейса приложения 12](#_Toc107251391)

[2.2 Инструменты для реализации функционала приложения 14](#_Toc107251392)

[2.3 Результаты проверки работы приложения 20](#_Toc107251393)

[**Заключение** 23](#_Toc107251394)

[**Список использованных источников** 24](#_Toc107251395)

[**Приложение А Исходный код приложения** 26](#_Toc107251396)

# **Введение**

PYTHON - это объектно-ориентированный язык программирования с широкими возможностями и простым синтаксисом. Возможность легко реализовать нужные функции без компиляции, т.к Python — это интерпретируемый язык то есть до запуска представляет из себя обычный текстовый файл. Так же он логичен и хорошо спроектирован, так же он мультиплатформенный что позволяет писать на разных устройствах. Синтаксис сравнительно короткий и код куда меньше в сравнении с другими языками.

С появлением цифровых технологий, так же появляются уязвимости. Минимизировать ущерб от различных угроз и уязвимостей информационной безопасности возможно благодаря анализу банка угроз и уязвимостей ФСТЭК, который может быть реализована в приложение по автоматизации данного процесса для ОС Centos.

**Цель проекта**: автоматизация задачи анализа уязвимостей OS Centos по данным банка уязвимостей ФСТЭК посредством языка программирования Python.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. выполнить настройку среды разработки приложения на языке Python;
2. спроектировать интерфейс приложения и реализовать функционал   
   в соответствии с требованиями, изложенными в задании;
3. протестировать приложение и разработать руководство пользователя.

# **1 ОСНОВЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ НА Python**

Основной упор в Python делается на скорости написания кода (а не на скорости выполнения кода, как например в языках С и C++). Поэтому в первую очередь Python удобно использовать там, где нужно быстро написать что-то работающее.

Все чаще Python используется для анализа данных, как в науке, так и коммерческой сфере. Этому способствует простота языка и большое разнообразие открытых библиотек.

## 1.1 Инструментарий для разработки приложений на Python

Краткие сведения об интерпретаторе Python, редакторах программного кода и IDE с поддержкой Python.

## 1.2 Инструментарий для создания пользовательского интерфейса

Краткие сведения о возможностях Python по созданию текстового и графического пользовательского интерфейса.

## 1.3 Настройка среды разработки

Для того что бы начать работу с проектом, нам нужно установить среду разработки. Было решено использовать PyCharm. Далее будет рассмотрена установка PyCharm и создание проекта.

1. Установите интерпретатор PyCharm с официального сайта JetBrains (Рисунок 1).



Рисунок 1 - Скачивание PyCharm

2. При нажатии «Download» будет произведено перенаправление на страницу загрузки PyCharm (Рисунок 2).

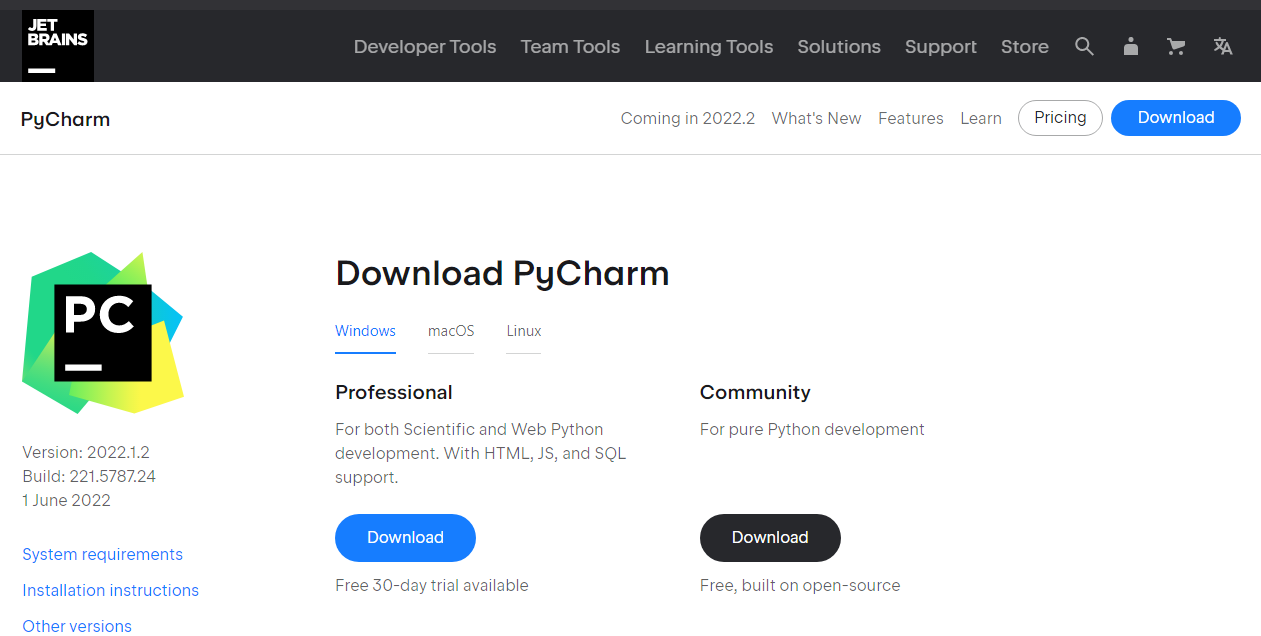


Рисунок 2 – Страница загрузки PyCharm

3. Нужно выбрать версию PyCharm: Community или Professional.

Professional — это платная версия с полным набором функций. Она идеально подходит для профессиональной разработки. Community бесплатная. Ею можно пользоваться благодаря набору базовых возможностей. После нажатия на кнопку «Загрузить» скачивание начнется автоматически (Рисунок 3).

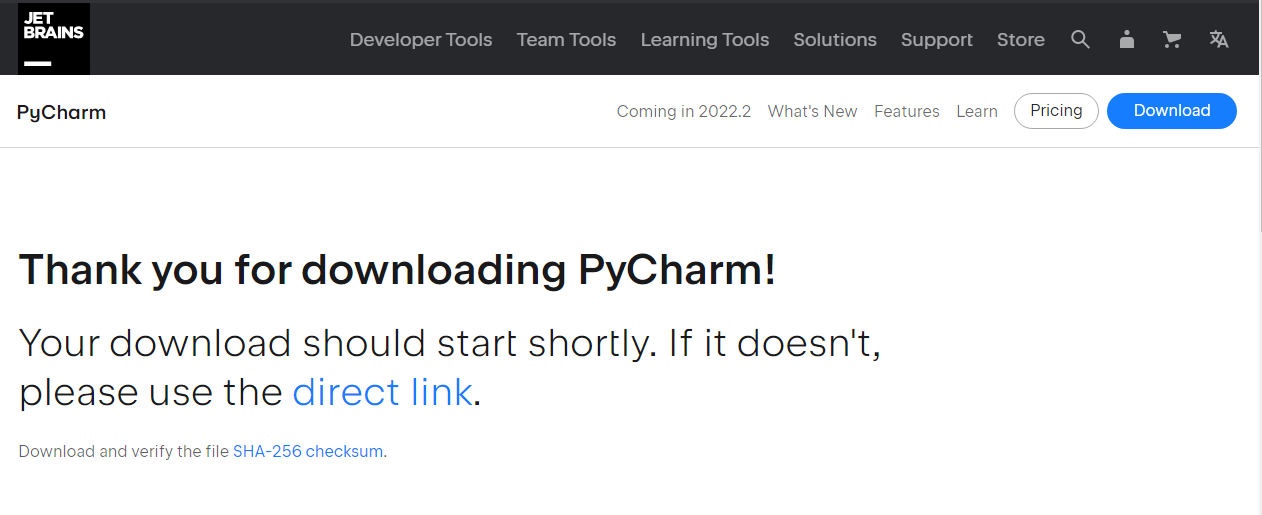


Рисунок 3 – Загрузка PyCharm

4. После загрузки необходимо запустить установку (pycharm-community-2022.1.2.exe)

Нас приветствует установщик pycharm, нажимаем «Next» (Рисунок 4).

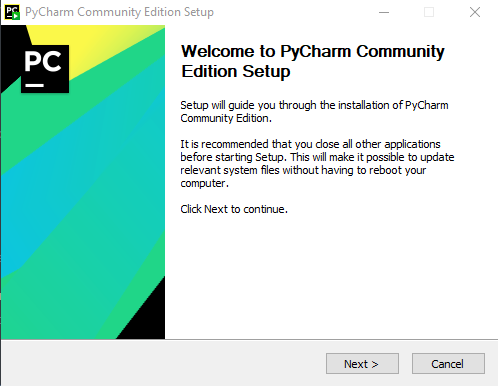


Рисунок 4 – Запуск установки

5. Поле выбираем расположение для установки на диске (Рисунок 5).

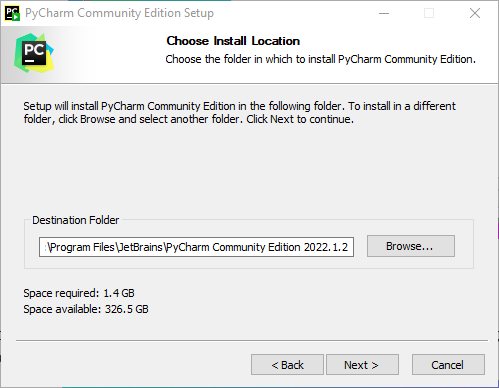


Рисунок 5 – Выбор места инсталляции

1. на следующем меню установщик просит нас выбрать пункты, которые нам нужны, создать ярлык на рабочем столе, обновить контекстное меню для файлов добавив «Открыть директорию как проект», создать ассоциацию с файлами, у которых расширение «.py», добавить папку «bin» в путь установки (Рисунок 6).

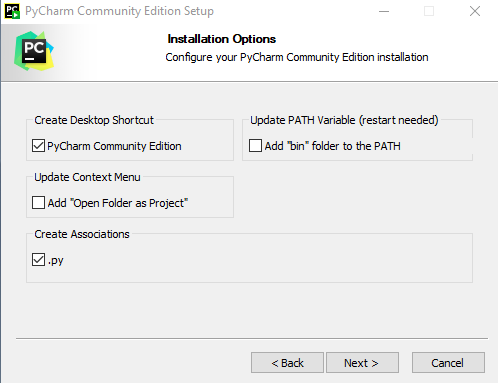


Рисунок 6 – Расставление галочек

7. Далее установщик попросит указать имя для отображения в стартовом меню. По умолчанию это JetBrains. Нажмите «Install».

8. Установка начнется автоматически. Дождитесь завершения.

9. После завершения установки поставьте галочку напротив «Run PyCharm Community Edition» для запуска программы, а затем «Finish» (Рисунок 7).

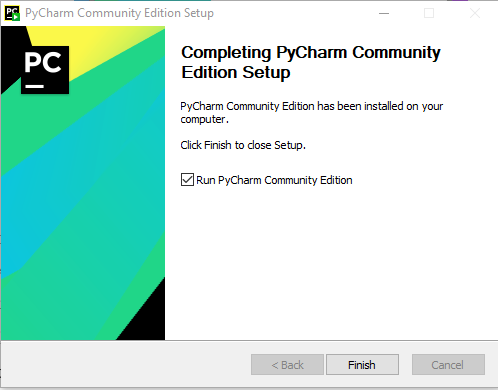


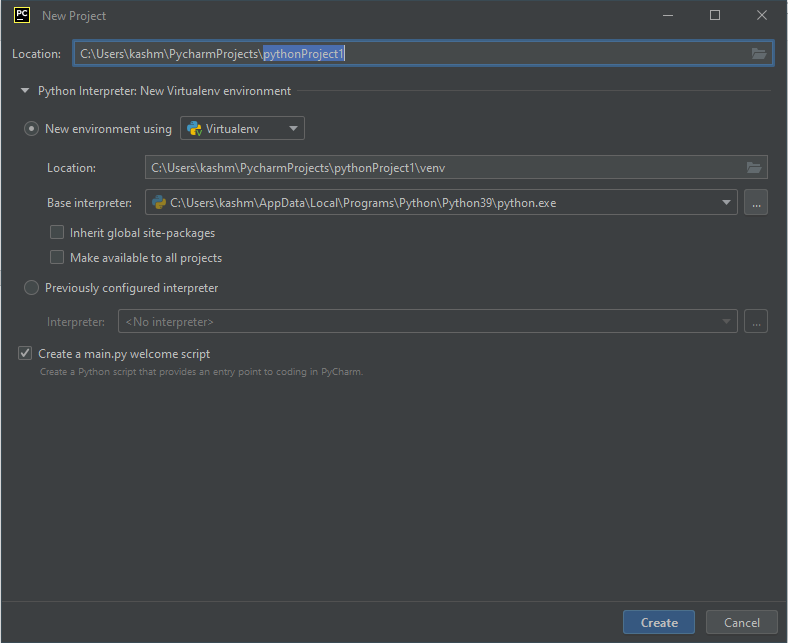
Рисунок 7 – Завершение установки

# 10. Если вы только учите язык Python и установили PyCharm впервые, тогда выберите пункт «I do not have a previous version of PyCharm or I do not want to import my settings» и «ОК».

# PyCharm запустится. Первый раз этой займет больше времени.

# После приветственного окна нажмите «Create New Project»

# Выберите директорию для сохранения проекта. Во втором поле нужно указать местоположение интерпретатора Python (который уже должен быть установлен). Обычно PyCharm находит его самостоятельно. Если не удалось, нужно указать путь к нему и нажать «Create» (Рисунок 8).



# Рисунок 8 – Выбор пути установки

# Нажмите «File > New > Python File». Введите название нового файла. Он откроется с чистой консолью. Можно начинать писать программу.

Таким образом, к основам разработки на Python, относится установка и настройка среды PyCharm, а также такой инструментарии пользовательского интерфейса как графическая библиотека Tkinter.**2 ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА И РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ПРИЛОЖЕНИЯ**

Банк данных угроз безопасности информации. Федеральная служба по техническому и экспортному контролю. ФСТЭК России. Государственный научно-исследовательский испытательный институт проблем технической защиты информации. ФАУ «ГНИИИ ПТЗИ ФСТЭК России».

Сведения о структуре документа MS Excel, содержащего данные банка уязвимостей ФСТЭК.

## 2.1 Инструменты для создания интерфейса приложения

Для создания приложения с графическим интерфейсом необходимо воспользоваться инструментарием Tkinter. Для создания графического интерфейса необходимо выполнить следующие шаги:

1. Установить библиотеку Tkinter
2. Написать окно программы и кнопки
3. Разместить все необходимые элементы внутри окна программы
4. Настроить свойства отдельных элементов для работы приложения

Результатом стал графический интерфейс, представленный на рисунке 9.

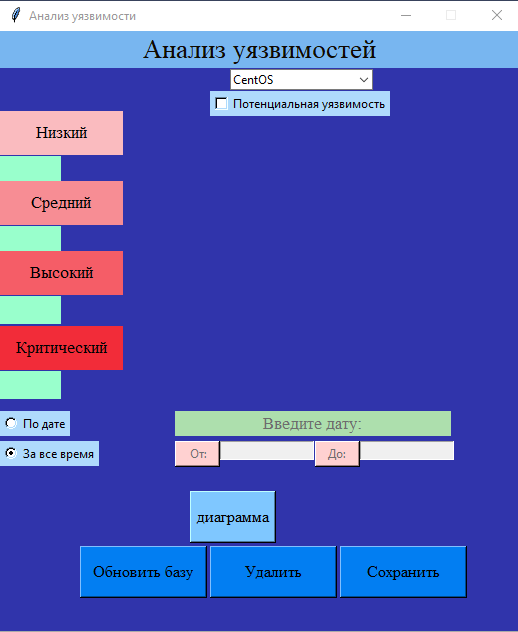


Рисунок 9 – Графический интерфейс приложения

Код для создания элементов кнопок представлен на рисунке 10.

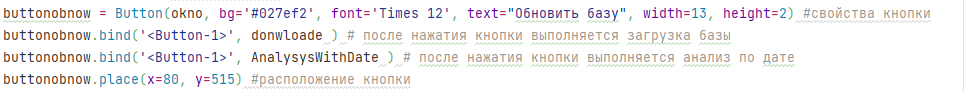


Рисунок 10 – код элемента “Button”

Графический интерфейс продуманным пользователь может как производить отбор по датам, так и за все время. При необходимости пользователь может просмотреть отчёт по анализу в Word. Туда входит не только количество всех видов уязвимостей, но и график уязвимостей по уровням опасности и датам.

## 2.2 Инструменты для реализации функционала приложения

Справочные сведения о возможностях библиотек Python для работы   
с документами MS Office.

Для того, чтобы начать реализовывать функционал приложения необходимо установить дополнительные библиотеки, которые будут в дальнейшем применены. Добавить библиотеки можно следующим образом: внизу, в панели с левой стороны переходим во вкладку Python Packages, там будут находиться уже установленные библиотеки (Рисунок 11). В поисковой строке вводим имя необходимой библиотеки, выбираем её в списке и устанавливаем (кнопка «Install» справа) (Рисунок 12). Во время установки выполняется копирование библиотек в проект и его обновление.

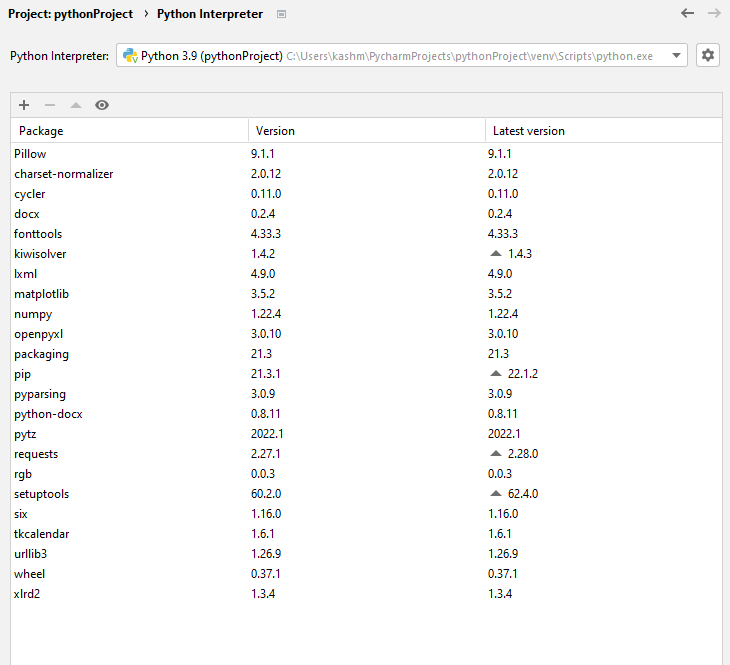


Рисунок 11 – Библиотеки

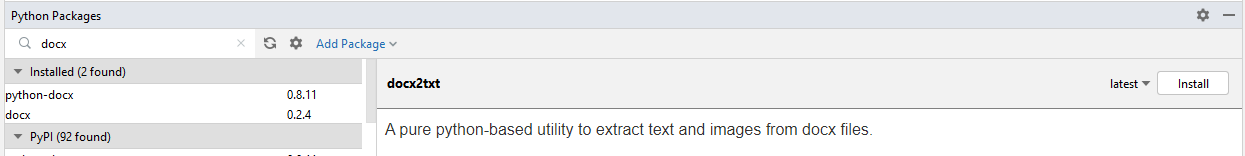


Рисунок 12 – Добавление библиотек

Для дальнейшей работы необходимо прописать используемые директивы в заголовке кода приложения (Рисунок 13).

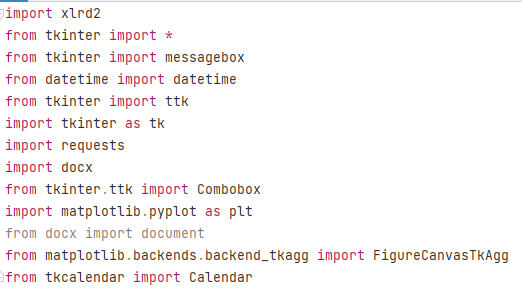


Рисунок 13 – Пространство имён

Перед написанием кода кнопок отбора уязвимостей из документа, необходимо создать событие загрузки самой базы (Рисунок 14)

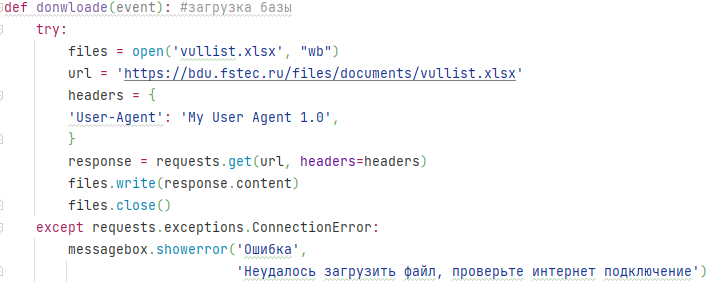


Рисунок 14 – Событие для загрузки базы

Теперь можно создать функции для отбора по уязвимостям по дете или без неё рисунок 15 и 16

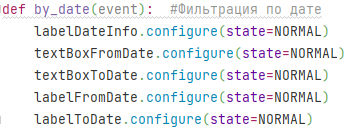


Рисунок 15 – События для отбора по дате

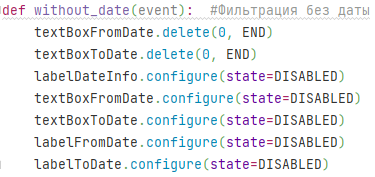


Рисунок 16 – Событие без отбора по дате

Также функцияя анализа базы по уровням уязвимостей

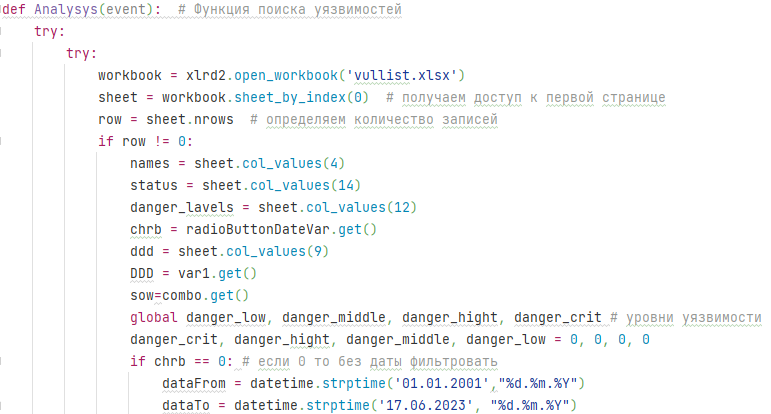


Рисунок 1



Рисунок 1



Рисунок 1

Так же для того что бы проинформировать пользователя об ошибке

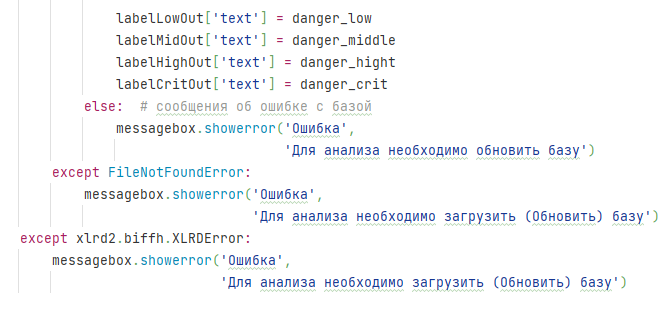


Рисунок 1

Для более наглядного обзора, создал функцию диаграммы



Рисунок 1

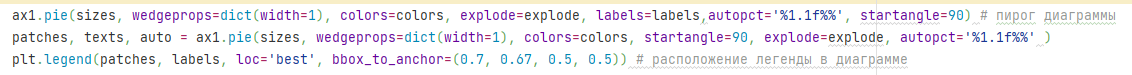


Рисунок 1

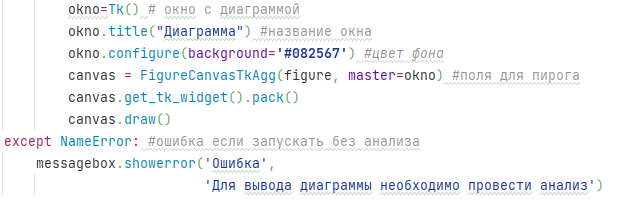


Рисунок 1

Для удобства выбора дат , функция для отчистки полей календаре.

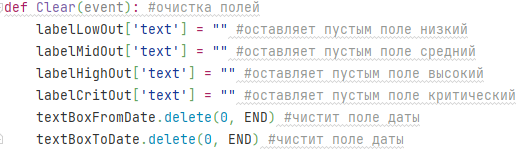


Рисунок 1

Для сохранения результатов в документ docx,

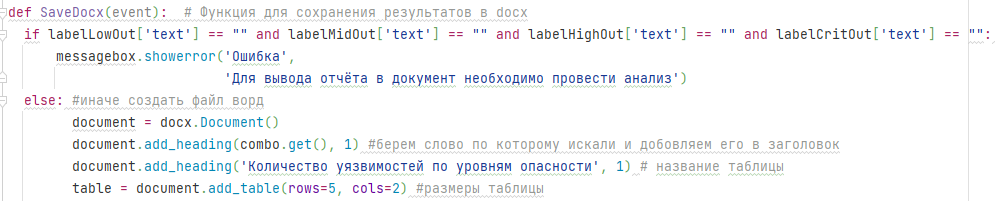


Рисунок 1

Создание оглавления ячеек в таблице



Рисунок 1

Метод для переноса данных об уязвимостях в яцейки таблицы, остальные яцейки заполняются такимже образом



Рисунок 1

Метод сохранения документа с именем



Рисунок 1

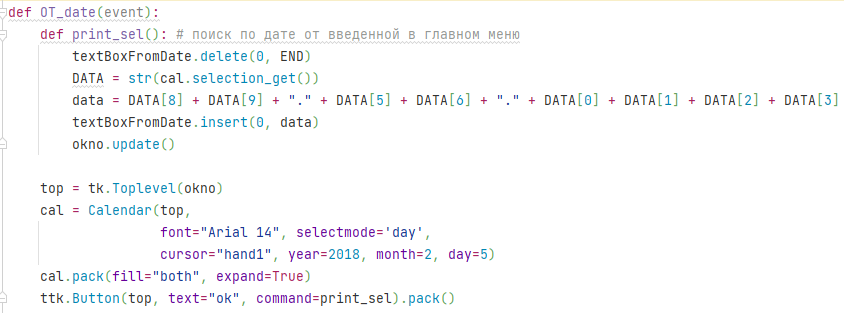


Рисунок 1

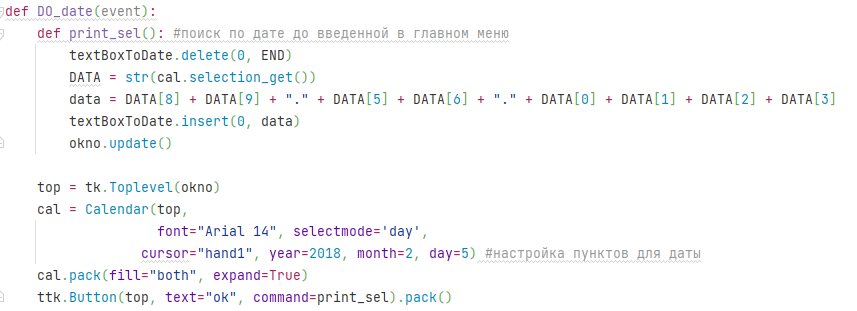


Рисунок 1

В результате был спроектирован интерфейс приложения и реализован функционал в соответствии с требованиями, изложенными в задании. Так в готовом программном осуществлена возможность вывода отобранных данных в таблицу, вывод их в Word и показ диаграммы с уязвимостями, анализ уязвимостей определенный период времени и показания количества уязвимостей по уровням опасности.

## 2.3 Результаты проверки работы приложения

Краткая инструкция по запуску приложения и скриншоты с результатами работы.

Приложение запускается файлом «main.py», который находится в папке «PyProjects» Работа приложения представлена на рисунке 22.

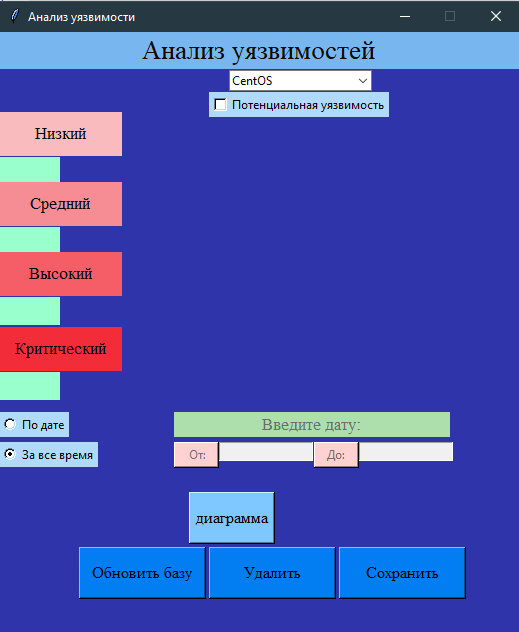


Рисунок – Работа приложения

Таким образом, в основу проектирования интерфейса было положено: сбор и хранение информации об угрозах и уязвимостях ПО, используемого в государственных организациях Российской Федерации, включая информационные системы и системы управления критичными производственными процессами. (библиотеки)

Реализация функционала приложения основана на загрузке банка данных уязвимостей с сайта ФСТЭК, возможностей вывода отобранных данных в таблицу Word, показа диаграммы с уязвимостями, а также отбора за определенный период времени и показания количества уязвимостей по уровням опасности.

# **Заключение**

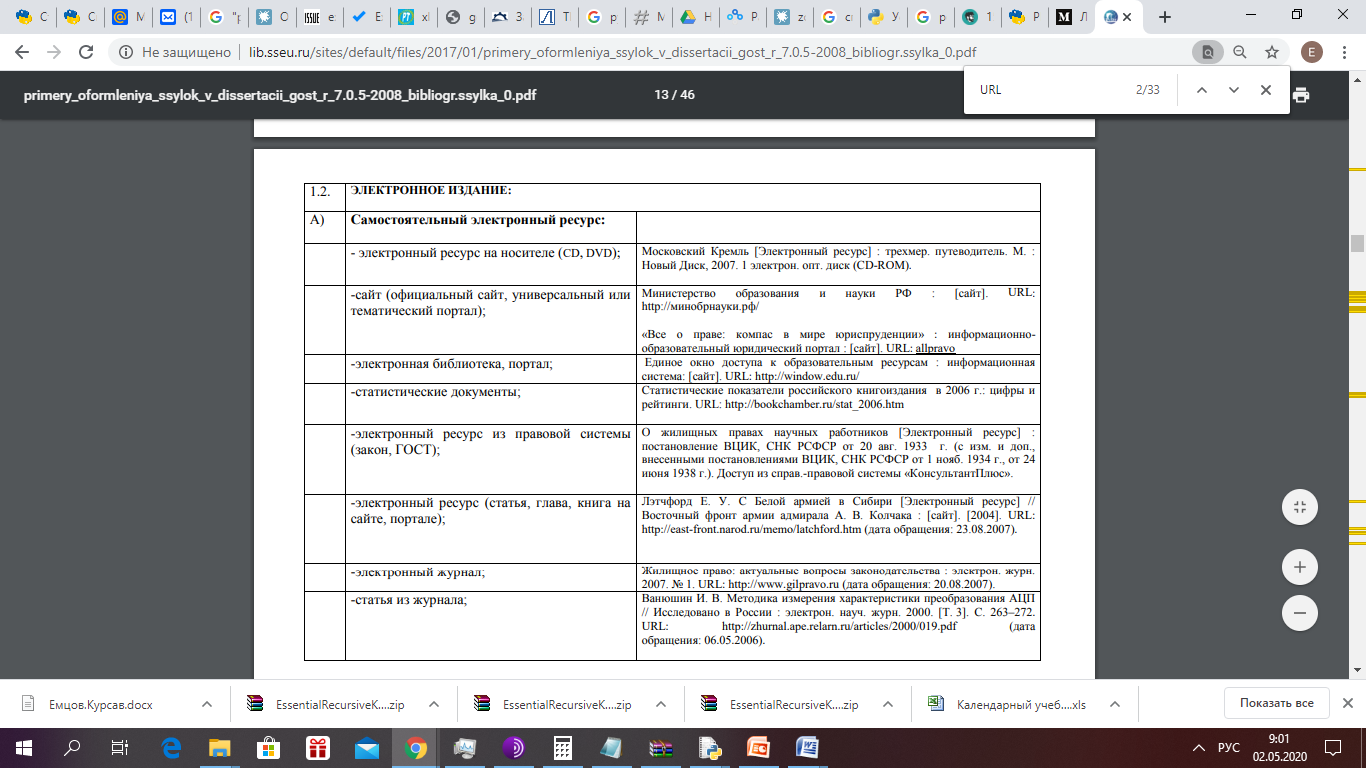
выполнил настройку среды разработки приложения на языке Python;

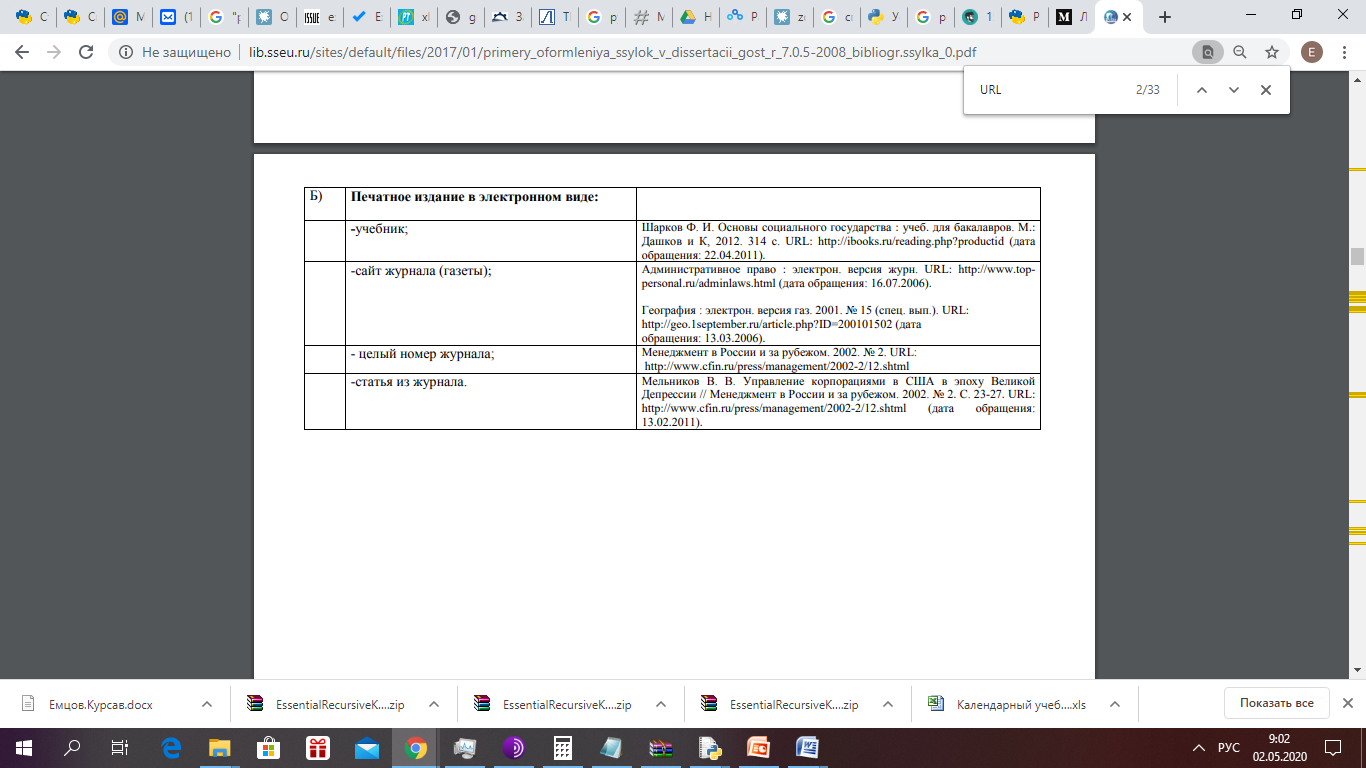
спроектировал интерфейс приложения и реализовать функционал   
в соответствии с требованиями, изложенными в задании;

протестировал приложение на его функционал и разработал руководство пользователя.

# **Список использованных источников**

1. Источник данных об уязвимостях (страница сайта ФСТЭК).
2. https://www.geeksforgeeks.org/working-with-tables-python-docx-module/
3. Источники с информацией об интерпретаторах Python (отличия 2-го и 3-го, актуальная версия, установка под разные ОС и т.д.).
4. Источники с информацией о текстовых редакторах кода и IDE для Python (онлайн-сервисы, настольное ПО и мобильные приложения).
5. Источники с информацией о возможностях создания пользовательских интерфейсов на Python (текстовые интерфейсы, графические интерфейсы настольных, мобильных и веб-приложений).
6. Источники со справочной информацией по функционалу используемых модулей (tkinter, numpy, docx, xlsreader).
7. Источники с рекомендациями по написанию и документированию программного кода (оформление кода, выбор имен объектов, комментирование и пр.).





# **Приложение А Исходный код приложения**

import xlrd2  
from tkinter import \*  
from tkinter import messagebox  
from datetime import datetime  
from tkinter import ttk  
import tkinter as tk  
import requests  
import docx  
from tkinter.ttk import Combobox  
import matplotlib.pyplot as plt  
from docx import document  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from tkcalendar import Calendar  
  
def donwloade(event): #загрузка базы  
 try:  
 files = open('vullist.xlsx', "wb")  
 url = 'https://bdu.fstec.ru/files/documents/vullist.xlsx'  
 headers = {  
 'User-Agent': 'My User Agent 1.0',  
 }  
 response = requests.get(url, headers=headers)  
 files.write(response.content)  
 files.close()  
 except requests.exceptions.ConnectionError:  
 messagebox.showerror('Ошибка',  
 'Неудалось загрузить файл, проверьте интернет подключение')  
def by\_date(event): #Фильтрация по дате  
 labelDateInfo.configure(state=NORMAL)  
 textBoxFromDate.configure(state=NORMAL)  
 textBoxToDate.configure(state=NORMAL)  
 labelFromDate.configure(state=NORMAL)  
 labelToDate.configure(state=NORMAL)  
def without\_date(event): #Фильтрация без даты  
 textBoxFromDate.delete(0, END)  
 textBoxToDate.delete(0, END)  
 labelDateInfo.configure(state=DISABLED)  
 textBoxFromDate.configure(state=DISABLED)  
 textBoxToDate.configure(state=DISABLED)  
 labelFromDate.configure(state=DISABLED)  
 labelToDate.configure(state=DISABLED)  
def AnalysysWithDate(event): # Функция для проверки правильности ввода даты  
 chrb = radioButtonDateVar.get() # состояние кнопок, 0 - за все время, 1 - по дате  
 if chrb == 0:  
 Analysys(event)  
 else:  
 dataFrom = textBoxFromDate.get()  
 dataTo = textBoxToDate.get()  
 if len(dataFrom and dataTo) == 10 and (dataFrom[2] and dataTo[2]) == '.' and (dataFrom[5] and dataTo[5]) == '.' and dataFrom[6:].isnumeric() and dataTo[6:].isnumeric() and dataFrom[:2].isnumeric() and dataTo[:2].isnumeric() and dataFrom[3:5].isnumeric() and dataTo[3:5].isnumeric():  
 tsFrom = datetime(year=int(dataFrom[6:]), month=int(dataFrom[3:5]), day=int(dataFrom[:2]))  
 tsTo = datetime(year=int(dataFrom[6:]), month=int(dataFrom[3:5]), day=int(dataFrom[:2]))  
 if (tsFrom.date and tsTo.day) > 0 and (tsFrom.day and tsTo.day) < 32 and (tsFrom.month and tsTo.month) > 0 and (tsFrom.month and tsTo.month) < 13 and (tsFrom.year and tsTo.year) > 1900:  
 dataFrom = datetime.strptime(textBoxFromDate.get(), "%d.%m.%Y")  
 dataTo = datetime.strptime(textBoxToDate.get(), "%d.%m.%Y")  
 Analysys(event)  
 else:  
 messagebox.showerror('Ошибка',  
 'Некорректно введена дата')  
 else:  
 messagebox.showerror('Ошибка',  
 'Некорректно введена дата')  
  
def Analysys(event): # Функция поиска уязвимостей  
 try:  
 try:  
 workbook = xlrd2.open\_workbook('vullist.xlsx')  
 sheet = workbook.sheet\_by\_index(0) # получаем доступ к первой странице  
 row = sheet.nrows # определяем количество записей  
 if row != 0:  
 names = sheet.col\_values(4)  
 status = sheet.col\_values(14)  
 danger\_lavels = sheet.col\_values(12)  
 chrb = radioButtonDateVar.get()  
 ddd = sheet.col\_values(9)  
 DDD = var1.get()  
 sow=combo.get()  
 global danger\_low, danger\_middle, danger\_hight, danger\_crit # уровни уязвимости  
 danger\_crit, danger\_hight, danger\_middle, danger\_low = 0, 0, 0, 0  
 if chrb == 0: # если 0 то без даты фильтровать  
 dataFrom = datetime.strptime('01.01.2001',"%d.%m.%Y")  
 dataTo = datetime.strptime('17.06.2023', "%d.%m.%Y")  
 else: # тогда  
 dataFrom = datetime.strptime(textBoxFromDate.get(), "%d.%m.%Y")  
 dataTo = datetime.strptime(textBoxToDate.get(), "%d.%m.%Y")  
  
 for i in range(9, row):  
 if ddd[i] != '':  
 ddd[i] = datetime.strptime(ddd[i], "%d.%m.%Y")  
 else:  
 ddd[i] = datetime.strptime('01.01.2001', "%d.%m.%Y")  
 for i in range(4, row):  
 if (str(ddd[i]) >= str(dataFrom)) and (str(ddd[i]) <= str(dataTo)):  
 if DDD == 1:  
 if status[i].find("Потенциальная уязвимость") >= 0 and names[i].find(sow) >= 0:  
 if danger\_lavels[i][0] == 'К': # Критический  
 danger\_crit += 1  
  
 elif danger\_lavels[i][0] == 'В': # Высокий  
 danger\_hight += 1  
  
 elif danger\_lavels[i][0] == 'С': # Средний  
 danger\_middle += 1  
  
 else: # Низкий  
 danger\_low += 1  
 else:  
 if names[i].find(sow) >= 0:  
  
 if danger\_lavels[i][0] == 'К': # Критический  
 danger\_crit += 1  
  
 elif danger\_lavels[i][0] == 'В': # Высокий  
 danger\_hight += 1  
  
 elif danger\_lavels[i][0] == 'С': # Средний  
 danger\_middle += 1  
  
 else: # Низкий  
 danger\_low += 1  
  
 labelLowOut['text'] = danger\_low  
 labelMidOut['text'] = danger\_middle  
 labelHighOut['text'] = danger\_hight  
 labelCritOut['text'] = danger\_crit  
 else: # сообщения об ошибке с базой  
 messagebox.showerror('Ошибка',  
 'Для анализа необходимо обновить базу')  
 except FileNotFoundError:  
 messagebox.showerror('Ошибка',  
 'Для анализа необходимо загрузить (Обновить) базу')  
 except xlrd2.biffh.XLRDError:  
 messagebox.showerror('Ошибка',  
 'Для анализа необходимо загрузить (Обновить) базу')  
  
def diagramma(event):  
 try:  
 if danger\_low == 0 and danger\_middle == 0 and danger\_hight == 0 and danger\_crit ==0:  
 messagebox.showerror('Ошибка',  
 'Для вывода диаграммы необходимо хоть одно не равное 0')  
 else:  
 labels = 'Низкий', 'Средний', 'Высокий', 'Критический' #название паев  
 sizes = [danger\_low, danger\_middle, danger\_hight, danger\_crit] #паи пирога  
 colors = ("#c0c0c0", "#fff44f", "#FFAF18",  
 "#E52E2A") # цвет паев  
 figure, ax1 = plt.subplots()  
 explode = (0, 0.1, 0, 0)  
  
 ax1.pie(sizes, wedgeprops=dict(width=1), colors=colors, explode=explode, labels=labels,autopct='%1.1f%%', startangle=90) # пирог диаграммы  
 patches, texts, auto = ax1.pie(sizes, wedgeprops=dict(width=1), colors=colors, startangle=90, explode=explode, autopct='%1.1f%%' )  
 plt.legend(patches, labels, loc='best', bbox\_to\_anchor=(0.7, 0.67, 0.5, 0.5)) # расположение легенды в диаграмме  
  
 okno=Tk() # окно с диаграммой  
 okno.title("Диаграмма") #название окна  
 okno.configure(background='#082567') #цвет фона  
 canvas = FigureCanvasTkAgg(figure, master=okno) #поля для пирога  
 canvas.get\_tk\_widget().pack()  
 canvas.draw()  
 except NameError: #ошибка если запускать без анализа  
 messagebox.showerror('Ошибка',  
 'Для вывода диаграммы необходимо провести анализ')  
  
def Clear(event): #очистка полей  
 labelLowOut['text'] = "" #оставляет пустым поле низкий  
 labelMidOut['text'] = "" #оставляет пустым поле средний  
 labelHighOut['text'] = "" #оставляет пустым поле высокий  
 labelCritOut['text'] = "" #оставляет пустым поле критический  
 textBoxFromDate.delete(0, END) #чистит поле даты  
 textBoxToDate.delete(0, END) #чистит поле даты  
  
def SaveDocx(event): # Функция для сохранения результатов в docx  
 if labelLowOut['text'] == "" and labelMidOut['text'] == "" and labelHighOut['text'] == "" and labelCritOut['text'] == "": # если все поля уязвимостей пустые выдть ошибку  
 messagebox.showerror('Ошибка',  
 'Для вывода отчёта в документ необходимо провести анализ')  
 else: #иначе создать файл ворд  
 document = docx.Document()  
 document.add\_heading(combo.get(), 1) #берем слово по которому искали и добовляем его в заголовок  
 document.add\_heading('Количество уязвимостей по уровням опасности', 1) # название таблицы  
 table = document.add\_table(rows=5, cols=2) #размеры таблицы  
  
 nazvcell = table.rows[0].cells # ячейка названия  
 nazvcell[0].text = 'Уровень' # 1 столбец название  
 nazvcell[1].text = 'Количество' # 2 столбец название  
  
 cellforlow = table.rows[1].cells # ячейка для данных Низкий  
 cellforlow[0].text = 'Низкий' # 1 столбец название  
 cellforlow[1].text = str(labelLowOut['text']) # 2 столбец количество  
  
 cellformid = table.rows[2].cells #ячейка для данных Средний  
 cellformid[0].text = 'Средний' # 1 столбец название  
 cellformid[1].text = str(labelMidOut['text']) # 2 столбец количество  
  
 cellsforhigh = table.rows[3].cells #ячейка для данных Высокий  
 cellsforhigh[0].text = 'Высокий' # 1 столбец название  
 cellsforhigh[1].text = str(labelHighOut['text']) # 2 столбец количество  
  
 cellsforcrit = table.rows[4].cells #ячейка для данных Критический  
 cellsforcrit[0].text = 'Критический' # 1 столбец название  
 cellsforcrit[1].text = str(labelCritOut['text']) # 2 столбец количество  
  
 document.save('Анализ уязвимостей ' + combo.get() + '.docx') #название файла  
  
def OT\_date(event):  
 def print\_sel(): # поиск по дате от введенной в главном меню  
 textBoxFromDate.delete(0, END)  
 DATA = str(cal.selection\_get())  
 data = DATA[8] + DATA[9] + "." + DATA[5] + DATA[6] + "." + DATA[0] + DATA[1] + DATA[2] + DATA[3]  
 textBoxFromDate.insert(0, data)  
 okno.update()  
  
 top = tk.Toplevel(okno)  
 cal = Calendar(top,  
 font="Arial 14", selectmode='day',  
 cursor="hand1", year=2018, month=2, day=5)  
 cal.pack(fill="both", expand=True)  
 ttk.Button(top, text="ok", command=print\_sel).pack()  
  
def DO\_date(event):  
 def print\_sel(): #поиск по дате до введенной в главном меню  
 textBoxToDate.delete(0, END)  
 DATA = str(cal.selection\_get())  
 data = DATA[8] + DATA[9] + "." + DATA[5] + DATA[6] + "." + DATA[0] + DATA[1] + DATA[2] + DATA[3]  
 textBoxToDate.insert(0, data)  
 okno.update()  
  
 top = tk.Toplevel(okno)  
 cal = Calendar(top,  
 font="Arial 14", selectmode='day',  
 cursor="hand1", year=2018, month=2, day=5) #настройка пунктов для даты  
 cal.pack(fill="both", expand=True)  
 ttk.Button(top, text="ok", command=print\_sel).pack()  
  
okno = Tk() # Главное меню  
okno.resizable(0, 0) #запрещает изменять размер главного меню  
okno.title("Анализ уязвимости") # Название окна  
okno.geometry("520x600") # Размер окна  
okno.configure(background='#3034ab') #цвет фона  
labelLow = Label(okno, width=13, height=2, bg='#fabbbf', font='Times 13', text="Низкий") #надпись уровня низкий  
labelLow.place(x=0, y=80) #расположение надписи  
labelLowOut = Label(okno, bg='#99ffcc', font='Times 15', fg='black', width=5) #поля уровня низкий  
labelLowOut.place(x=0, y=125) #расположение поля  
labelMid = Label(okno, width=13, height=2, bg='#f78d94', font='Times 13', text="Средний") #надпись уровня средний  
labelMid.place(x=0, y=150) #расположение надписи  
labelMidOut = Label(okno, bg='#99ffcc', font='Times 15', fg='black', width=5) #поля уровня средний  
labelMidOut.place(x=0, y=195) #расположение поля  
labelHigh = Label(okno, width=13, height=2, bg='#f55d67', font='Times 13', text="Высокий") #надпись уровня высокий  
labelHigh.place(x=0, y=220) #расположение надписи  
labelHighOut = Label(okno, bg='#99ffcc', font='Times 15', fg='black', width=5) #поля уровня высокий  
labelHighOut.place(x=0, y=265) #расположение поля  
labelCrit = Label(okno, width=13, height=2, bg='#f22c39', font='Times 13', text="Критический") #надпись уровня критический  
labelCrit.place(x=0, y=295) #расположение надписи  
labelCritOut = Label(okno, bg='#99ffcc', font='Times 15', fg='black', width=5) #поля уровня критический  
labelCritOut.place(x=0, y=340) #расположение поля  
radioButtonDateVar = BooleanVar() #кнопоки  
radioButtonDateVar.set(0) # по умолчанию не выбрано  
radioButtonDateOn = Radiobutton(okno, text="По дате", bg='#afdafc', variable=radioButtonDateVar, value=1) #свойства кнопки  
radioButtonDateOn.bind('<Button-1>', by\_date) #бинд функциии после нажатия, фильтрация по дате  
radioButtonDateOn.pack(anchor=W) #растяжения элемента на west(левая сторона)  
radioButtonDateOn.place(x=0, y=380) #расположение  
radioButtonwithoutDate = Radiobutton(okno, text="За все время", bg='#afdafc', variable=radioButtonDateVar, value=0) #свойства кнопки  
radioButtonwithoutDate.bind('<Button-1>', without\_date ) #бинд функциии после нажатия, фильтрация без конкретной даты  
radioButtonwithoutDate.pack(anchor=W) #растяжения элемента на west(левая сторона)  
radioButtonwithoutDate.place(x=0, y=410) #расположение кнопки  
combo = Combobox(okno) #список  
combo['values'] = ('CentOS', 'Red Hat Enterprise', 'Red Hat Enterprise Linux', 'Red Hat Inc.') #содержание списка  
combo.current(0) # установите вариант по умолчанию CentOS  
combo.grid(column=0, row=0) # количество колонок и строк в списке  
combo.place(x=230, y=38) #расположение списка  
var1 = BooleanVar() # переменная булевская для состояния кнопки 0/1 выбрано/не выбрано  
var1.set(0) #не выбрано по дефолту  
c1 = Checkbutton(okno, bg = '#afdafc', text="Потенциальная уязвимость", variable=var1, onvalue=1, offvalue=0) #свойства  
c1.pack(anchor=W, padx=10) # позиционирование копки. растяжения элемента на west(левая сторона). отступы по горизонтали  
c1.grid(column=0, row=0) # количество колонок и строк  
c1.place(x=210, y=60) #расположение  
buttonobnow = Button(okno, bg='#027ef2', font='Times 12', text="Обновить базу", width=13, height=2) #свойства кнопки  
buttonobnow.bind('<Button-1>', donwloade ) # после нажатия кнопки выполняется загрузка базы  
buttonobnow.bind('<Button-1>', AnalysysWithDate ) # после нажатия кнопки выполняется анализ по дате  
buttonobnow.place(x=80, y=515) #расположение кнопки  
buttonClear = Button(okno, bg='#027ef2', font='Times 12', text="Удалить", width=13, height=2) #свойства кнопки  
buttonClear.place(x=210, y=515) #расположение кнопки  
buttonClear.bind('<Button-1>', Clear) #бинд функциии после нажатия, чистануть все поля  
buttonSave = Button(okno, bg='#027ef2', font='Times 12', text="Сохранить", width=13, height=2) #свойства кнопки  
buttonSave.place(x=340, y=515) #расположение кнопки  
buttonSave.bind('<Button-1>', SaveDocx)  
buttonDiagram = Button(okno, bg='#7fc7ff', font='Times 12', text="диаграмма", height=2) #свойства кнопки  
buttonDiagram.place(x=190, y=460) #расположение кнопки  
buttonDiagram.bind('<Button-1>', diagramma) #бинд функциии после нажатия, вызов диаграммы  
labelDate = Label(okno, text="Введите дату:", state=DISABLED, bg='#addfad', font='Times 13', fg='#000', width=30) #свойства надписи  
labelDate.place(x=175, y=380) #расположение кнопки  
labelFromDate = Button(okno, text=" От:", state=DISABLED, bg='#ffd1d1', fg='black', width=5) #свойства кнопки  
labelFromDate.place(x=175, y=410) #расположение кнопки  
labelFromDate.bind('<Button-1>', OT\_date) #бинд функциии после нажатия, определяет от какой даты фильтровать записи  
textBoxFromDate=Entry(okno, state=DISABLED, width=15) #свойства поля  
textBoxFromDate.place(x=220, y=410) #расположение кнопки  
labelToDate = Button(okno, text="До:", state=DISABLED, bg='#ffd1d1', fg='black', width=5) #свойства кнопки  
labelToDate.place(x=315, y=410) #расположение кнопки  
labelToDate.bind('<Button-1>', DO\_date) #бинд функциии после нажатия, определяет до какой даты фильтровать записи  
textBoxToDate = Entry(okno, state=DISABLED, width=15) #свойства поля  
textBoxToDate.place(x=360, y=410) #расположение кнопки  
labelDateInfo = Label(okno, text="Анализ уязвимостей", bg='#78b6f0', font='Times 20', fg='#0f0901', width=50) #свойства надписи  
labelDateInfo.pack()  
labelToInfo = Label(okno, bg='#78b6f0', fg='black', width=20) #свойства надписи

okno.mainloop()**Приложение Б  
Руководство пользователя**

Системные требования.

Краткая инструкция по запуску и использованию приложения.