Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Институт Информационные системы, экономика и управление

Кафедра Информационная безопасность

Специальность Информационная безопасность

Специализация Информационная безопасность

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине: «Технологии и методы программирования»

Тема проекта: Разработка информационной системы «Студенты ВУЗа»

Выполнил:

студент группы БИб-23Э1 Герасименко София Евгеньевна

Курсовой проект сдан на проверку:

«13» мая 2025 г.

Студент 

(подпись студента)

Курсовой проект допущен к защите:

«15» мая 2025 г.

Руководитель проекта

доцент кафедры, к.п.н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Абрамченко)

Курсовой проект защищен с оценкой: отлично

«27» мая 2025 г.

Руководитель проекта

доцент кафедры, к.п.н \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Абрамченко

Омск 2025

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Кафедра «Информационная безопасность»

УТВЕРЖДАЮ

И.О. Зав. кафедрой, к.т.н., доцент кафедры

\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Толкачева

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**ЗАДАНИЕ**

**к курсовому проекту**

студенту группы БИб-23Э1 Герасименко Софии Евгеньевне

1. Тема курсового проекта: Разработка информационной системы «Студенты ВУЗа»
2. Исходные данные к курсовому проекту:

* документация по языку программирования Python;
* требования к функционалу приложения:
  1. разработка базы данных с использованием Python и SQL;
  2. реализация графического интерфейса информационной системы на PyQt5;
  3. внедрение функций редактирования и поиска по базе данных;
  4. организация различных уровней доступа;
  5. использование авторизации;
  6. тестирование продукта модулем unnitest;

ВСТАВИТЬ ПРАВКА ПО ТРЕБОВАНИЯМ

1. Содержание пояснительной записки:

* титульный лист;
* задание к курсовому проекту;
* содержание;
* введение;
* формирование информационной системы
* разработка базы данных с использованием SQL
* проектирование интерфейса приложения на библиотеке Python – PyQt5;
* реализация функционала приложения на Python;
* внедрение тестирования работы приложения
* заключение;
* список использованных источников;
* приложения.

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

* презентация Microsoft PowerPoint.

1. Срок сдачи проекта: « 20» мая 2025 г.

Задание выдано « 1 » марта 2025 г.

Руководитель проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Абрамченко Н.В.

Задание к исполнению принял « 3 » марта 2025 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание**

[**Введение** 5](#_Toc168532979)

[**1** **ОПИСАНИЕ ЯП И СРЕДЫ РАЗРАБОТКИ** 7](#_Toc168532980)

[**2** **РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ** 12](#_Toc168532981)

[**3**  **РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ПРИЛОЖЕНИЯ** 29](#_Toc168532982)

[**Заключение** 35](#_Toc168532983)

[**Список использованных источников** 36](#_Toc168532984)

[**Приложение А Исходный код приложения** 37](#_Toc168532985)

[**Приложение Б Руководство пользователя** 37](#_Toc168532986)

# **Введение**

Работа с базами данных является если не главным, то ключевым умением множества специалистов информационных технологий. В данной поставленной задаче база данных представляет собой модель списка личных данных всех лиц, обучающихся в некотором заведении, что напрямую перекликается с профилем обучения – информационной безопасностью.

Курсовой проект является совокупностью многих базовых умений для программистов и представителей безопасности, таких как навык планирования разработки коммерческого или некоммерческого продукта, работы с данными в формате SQL-таблиц, автоматизация и оптимизация процессов, обеспечение целостности данных, путем внедрения уровней доступа, дополнительно ориентирование в объемном проекте и создание собственного приложения, с последующим тестированием работы – полезные навыки для работника IT-сферы.

Изучение нескольких языков программирования помогает развить личную универсальность как специалиста. Конкретно реализация проекта на языке Python, развивает не только умение использовать систематические знания о программировании, но и творческую часть, что участвует в создании дизайна интерфейса.

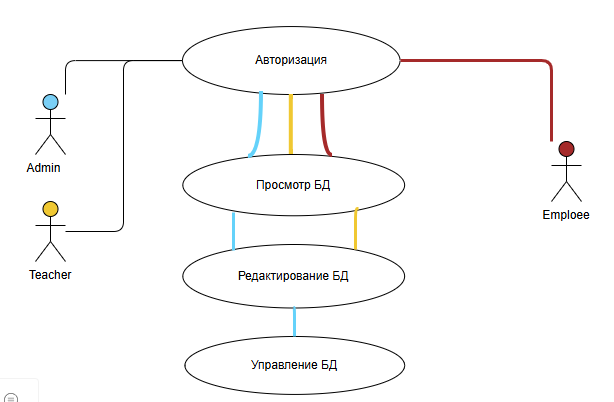
Рассматривая положительные стороны языка можно выделить разнообразное количество библиотек и модулей, упрощающих выполнение многообразия задач. В дополнение Python – это сравнительно молодой язык, который можно назвать более упрощенным для новых пользователей, ведь определенные базовые действия выполняет самостоятельно, позволяя сократить время и сложность выполнения задания.

Именно по вышеперечисленным причинам и факторам выбор языка – наиболее оптимален и оправдан, а курсовой проект со своими задачами считается актуальным и полезным для развития множества профессиональных навыков.

**Цель проекта**: поэтапная реализация автоматизированного программного обеспечения работы с базой данных информационной системы «Единая база данных обучающихся» посредством языка программирования Python.

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

1. описать план разработки продукта;
2. провести анализ информационной системы и структуры кода;
3. реализовать интерфейс приложения;
4. реализовать функционал в соответствии с требованиями, изложенными в задании;
5. организовать тестирование продукта, посредством внешнего воздействия;



# **1 ОПИСАНИЕ ЯП И ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**Описание языка разработки**

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Мы используем Python, так как, кроме прочих преимуществ, данный язык относится к группе языков, обеспечивающих быстрое освоение и свободу в действиях за счет собственной оптимизации, в сравнении с языком С++, обучение которого было осуществлено на протяжении более одного учебного года, Python позволяет ускорить процесс разработки, прививает привычку к структурированию кода, параллельно открывая множество возможностей, таких как:

* Работа с xml/html файлами
* Работа с http запросами
* GUI (графический интерфейс)
* Создание веб-сценариев
* Работа с FTP.
* Работа с изображениями, аудио и видео файлами
* Робототехника
* Программирование математических и научных вычислений

PyQt5 – объемная библиотека GUI для работ с интерфейсами и, соответственно, для их создания на языке Python, обеспечивающая достаточный объем функций и параметров для комфортного использования.

PyQt5 представляет собой мощный инструмент, позволяющий обширно и модульно произвести создание интерфейса программы, параллельно закрепляя знания объектно-ориентированного программирования, что является полезным дополнением в разработке, актуальным для выполнения курсового проекта.

# **2 РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ**

Банк уязвимостей Федеральной службы по техническому и экспортному контролю (ФСТЭК) представляет собой базу данных, которая содержит информацию о потенциальных уязвимостях в программном обеспечении, используемом государственными учреждениями. Банк уязвимостей создан для защиты государственных информационных систем. Он позволяет быстро обнаруживать уязвимости и предлагать решения для их устранения. Информация о банке уязвимостей может быть передана не только сотрудниками ФСТЭК, но и другими организациями, имеющими право на это. Банк уязвимостей доступен только авторизованным пользователям.

**Структура документа MSExcel, содержащего данные банка уязвимостей ФСТЭК.**

Документ MS Excel, содержащий данные банка уязвимостей ФСТЭК, имеет следующую структуру:  
 1. Листы с данными - каждый лист соответствует определенной категории уязвимостей (например, "Уязвимости ОС", "Уязвимости приложений" и т.д.).  
 2. Колонки с данными - каждая колонка содержит определенный тип информации о уязвимостях, такой как название уязвимости, описание, уровень опасности, рекомендации по устранению и т.д.

Переходим к обзору готового интерфейса.

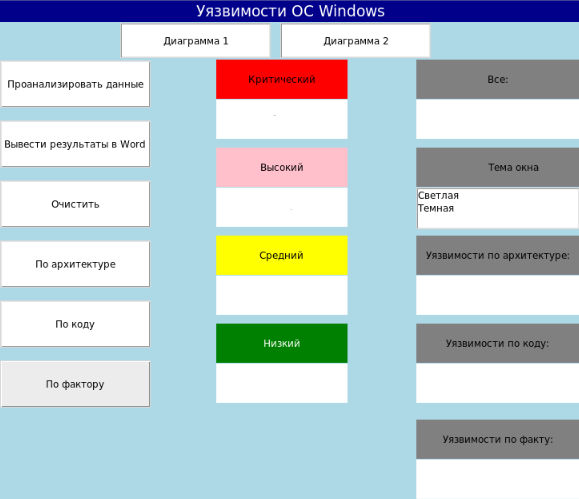
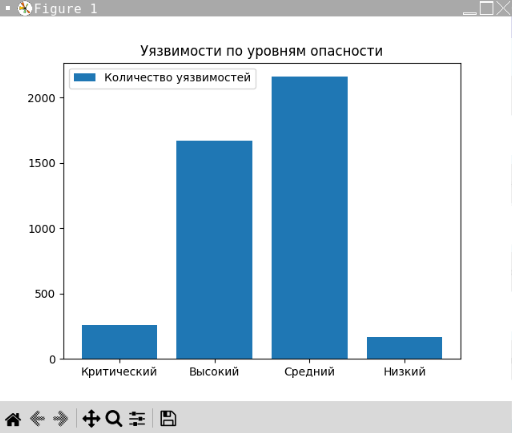


Рисунок 2.1 - Интерфейс приложения

На рисунке 2.1 показан готовый вид графического интерфейса.



**Рисунок 2. 2 - Диаграмма №1**

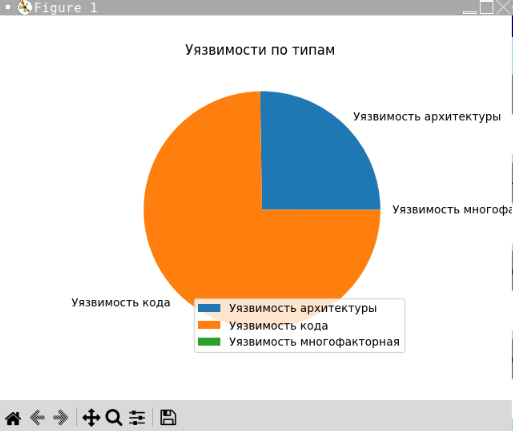


Рисунок 2. 3-Диаграмма №2

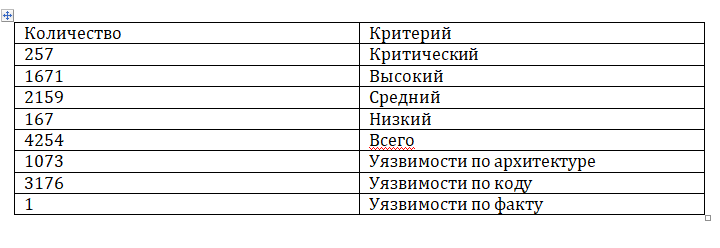
На рисунке **2.2** и **2.3** показаны диаграммы, которые были созданы по данным из таблицы Excel. 

Рисунок 2. 4 - Документ Word

На рисунке 2.4 показан документ Word, который был создан по данным из таблицы Excel.

Во время создания интерфейса были использованы такие виджеты как:



Рисунок 2. 5- Кнопка

На рисунке 2.5 показан виджет Button - в Tkinter представляет собой кнопку, которую можно нажимать для выполнения определенного действия.

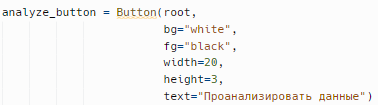


Рисунок 2. 6- Фрагмент кода для кнопки

На рисунке 2.6 представлен код кнопки, которая создается на интерфейсе приложения и позволяет выполнить функцию - analyz.

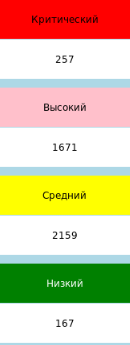


Рисунок 2. 7-Текстовые метки Label

На рисунке 2.7 показаны несколько label-ов - это виджеты, которые представляют собой текстовые метки. Он может отображать текст, изображения и другие объекты на экране.

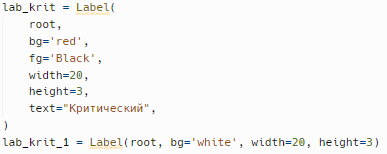


Рисунок 2. 8-Фрагмент кода для создания lable

На рисунке 2.8 показан код для создания текстовой метки Label, но и в общих чертах, данные параметры применяются ко всем объектам библиотеки Tkinter.

Некоторые команды для изменения вида текстовой метки:

1. «width» - изменение ширины Label
2. «height» - изменение ширины Label
3. «fg» - изменение цвета текста, который будет написан в Label
4. «bg» - изменение цвета самого Label
5. «text» - изменение текста в Label

В интерфейсе присутствует функция изменения цвета заднего фона, реализованная через объект Listbox (Рисунок 2.9).



Рисунок 2. 9-Изменение темы окна через Listbox

Объект формируется следующим образом, указанным на рисунке 2.10.

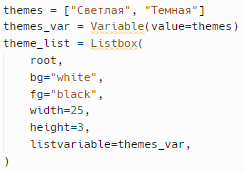


Рисунок 2. 10- Метод реализации Listbox

За расположение или позиционирование виджетов на интерфейсе отвечают несколько методов:

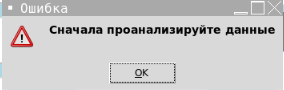
1. .pack
2. .place
3. .grid

В данной работе были задействованы первые два варианта, с перевесом в сторону метода .place . Разница между ними описывается следующим образом (Таблица 1).

Таблица 1- Методы расположения

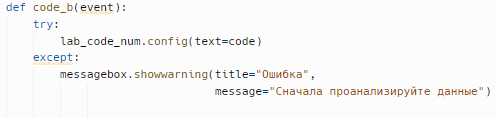
|  |  |
| --- | --- |
| Название | Принцип работы |
| Pack | Прикрепляет виджет посредством «сторон света», отталкиваясь от соседних объектов, можно указать отступы. |
| Place | Является методом, основанным на координатах x и y, оказался наиболее удобным, пусть и является затратным по времени. |
| Grid | Содержит в себе невидимую таблицу, по которой и идет прикрепление виджетов. |

1. Всплывающее окно дополняет список использованных методов **(Дополнение 1):**



Дополнение 1 – Всплывающее окно

Реализация описывается в каждой функции, где окно необходимо, образец кода выглядит следующее **(Дополнение 2):**



Дополнение 2-Образец вызова всплывающего окна

**Инструкция по запуску приложения и изображения с результатами работы.**

Для корректного запуска приложения - необходимо установить определенное количество нужных библиотек для работы приложения:

* xlrd2
* Tkinter
* Python-docx
* matplotlib

После того, как программа запустилась - пользователю будет показан интерфейс приложения. Чтобы вывести количество угроз по уровню - нужно нажать на кнопку «Проанализировать данные». **(Рисунок 2.11)**

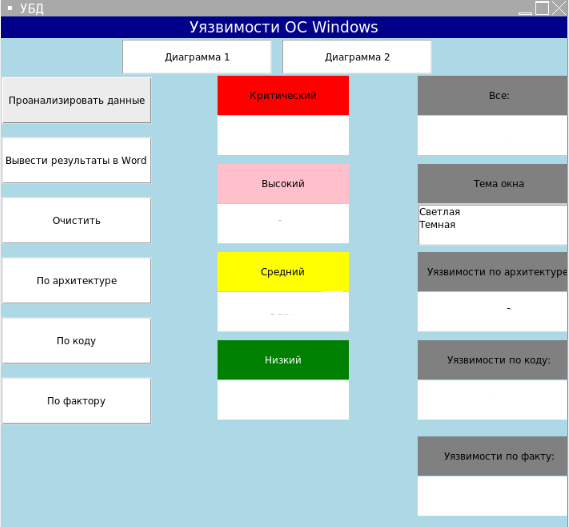


Рисунок 2. 11- Кнопка «Проанализировать данные»

Для того чтобы данные сохранились в таблицу в новый документ Word - пользователю необходимо нажать на кнопку «Вывести результаты в Word» **(Рисунок 2.12)**

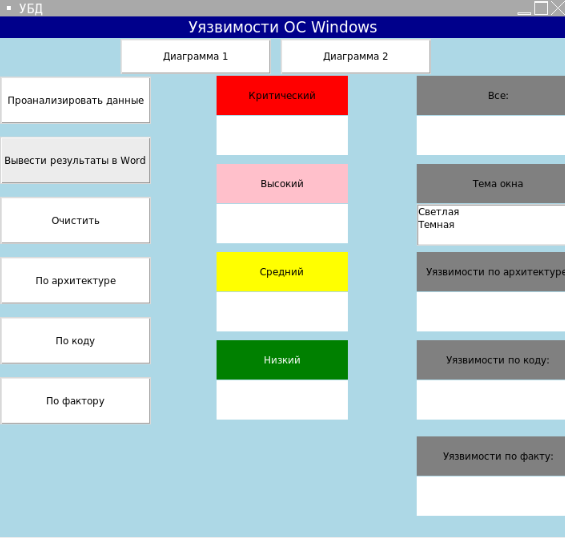


Рисунок 2.12- Кнопка «Вывести результаты в Word»

Для вывода данных в диаграмму - нужно нажать на кнопку «Диаграмма 1» или «Диаграмма 2» **(Рисунок 2.13)**

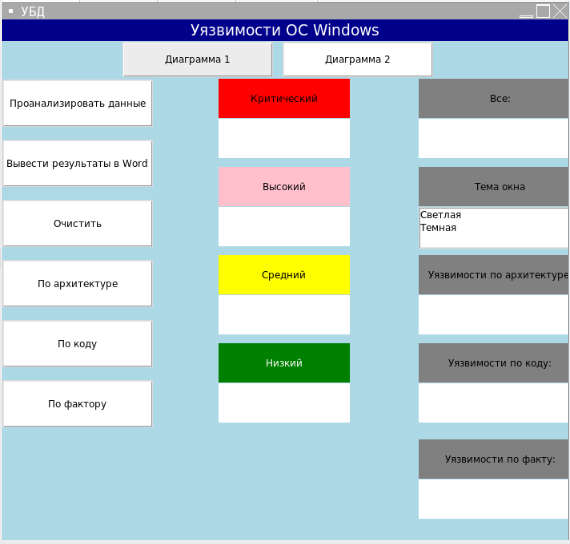


Рисунок 2. 13- Кнопки «Диаграмма»

Для очистки значений - нужно нажать на кнопку «Очистить». **(Рисунок 2.14)**

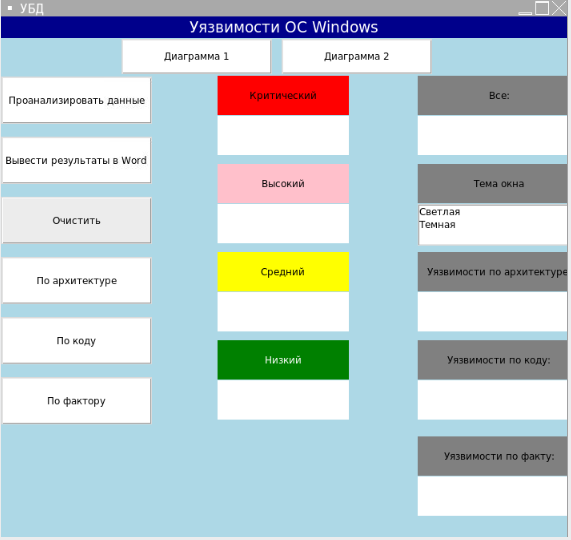
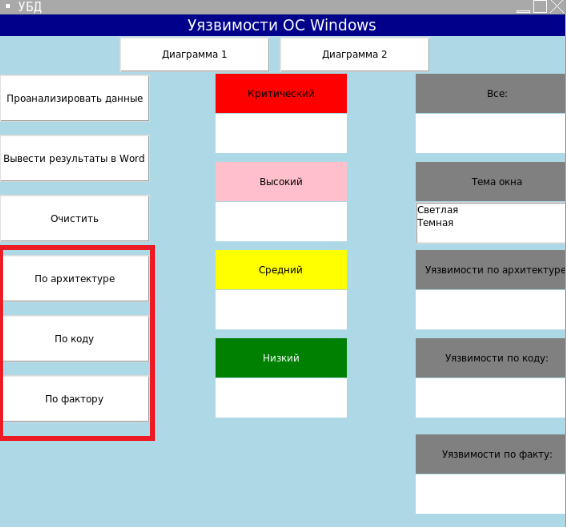


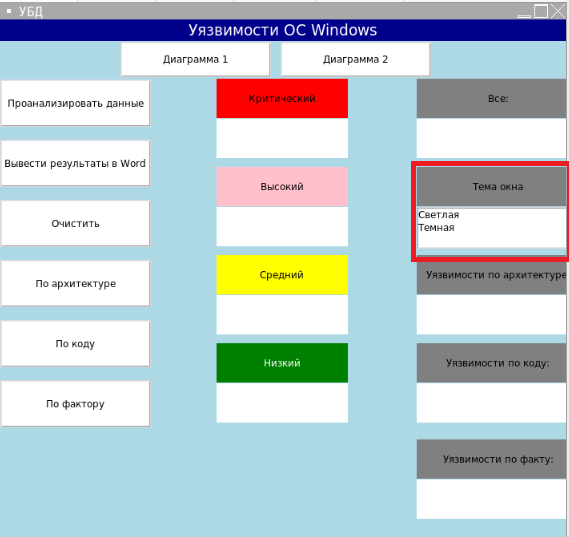
Рисунок 2. 14- Кнопка «Очистить»

Следующая группа кнопок вывода количества уязвимостей по классам содержит три кнопки: «По архитектуре», «По коду», «По фактору». При их использовании вывод осуществится в одноименных виджетах справа: «Уязвимости по коду», «Уязвимости по архитектуре», «Уязвимости по факту». Под текстом «по факту» или «по фактору», подразумевается количество многофакторных уязвимостей. Такое сокращение произошло из эстетических и размерных причин. **(Рисунок 2.15)**



**Рисунок 2.15 – Группа кнопок вывода по классу уязвимостей**

Для выбора определенной темы интерфейса - необходимо выбрать её в ListBox «Тема окна». **(Рисунок 2.16)**



**Рисунок 2.16 - Выбор темы**

В интерфейсе приложения присутствуют всплывающие окна, сообщающие об отсутствии анализа данных. Вывод окна осуществится, если пользователь попытается воспользоваться функциями, не выводя перед этим нужных для этого данных. К примеру, при попытке создать диаграмму №2, в случае отсутствия данных об уязвимостях по классам, пользователя ожидает следующее сообщение (Рисунок 2.17):

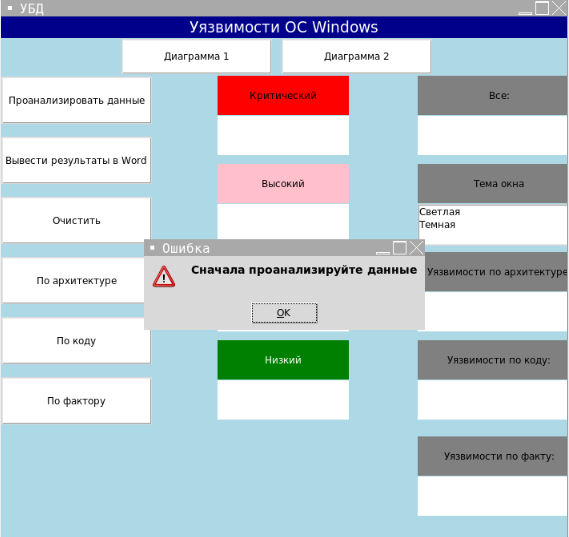
****

Рисунок 2. 17-Всплывающее окно

**Обоснование удобства использования приложения и корректности результатов его работы**

Приложение выполняет функцию автоматизации анализа уязвимостей, существенно ускоряя процесс. Сравнительно простой и понятный интерфейс интуитивен для любого пользователя. Лаконичный дизайн приятен глазу и не доставляет неудобств.

В ходе реализации поставленных задач были предусмотрены все возможные неточности. Работа приложения протестирована досконально и не содержит ошибок. Приложением можно пользоваться без каких-либо системных ограничений.

**Таким образом,** нами были изучены и использованы разные методы библиотеки Tkinter такие как: Button, Label, ListBox, MessageBox, что в следствие позволило создать интерфейс приложения, включая использование азов дизайна (подбор цветов, расположение объектов на полотне, шрифты).

# **3 РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ПРИЛОЖЕНИЯ**

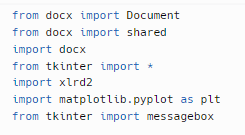
Python имеет множество библиотек, которые позволяют работать с документами MS Office, но в курсовом проекте участие примут только две.

1*. xlrd2* - отвечает за чтение данных из файлов формата Microsoft Excel (XLS и XLSX)

2*. python-docx* - библиотека для работы с документами Word.

**Реализация приложения**

Для работы всего приложения, сначала нужно подключить необходимые библиотеки. **(Рисунок 3.1)**



**Рисунок 3.1 - Необходимые библиотеки**

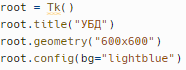
1) *from tkinter import \** - импортирует библиотеку «Tkinter», которая используется для создания графических интерфейсов.

2) *import xlrd2* - импортирует модуль «xlrd2», который используется для чтения файлов Excel.

3) *from docx import Document* - импортирует класс «Document» из модуля «docx», который используется для создания и редактирования документов Word

4) *import matpolib.pyplot as plt* – импортирует модуль plt, который нужен для создания диаграмм.

2. Для создания главного экрана приложения был использован данный код: **(Рисунок 3.2)**



**Рисунок 3.2 - Код программы для главного окна**

*root.title* - позволяет изменить название нашего главного окна

*root.geometry -* позволяет изменить размер нашего главного окна

*root.config(bg=”lightblue”)* – позволяет указать цвет фона главного окна

Открываем документ MS Excel со всеми уязвимостями. **(Рисунок 3.3)**



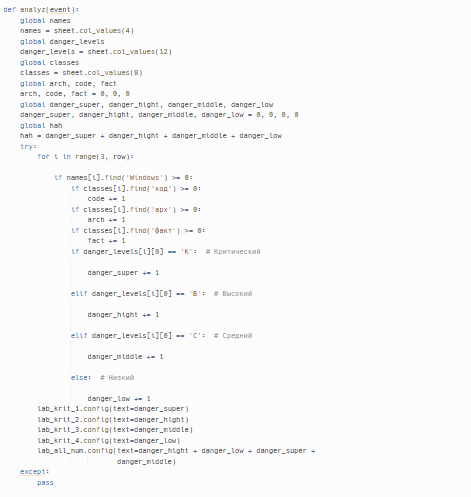
**Рисунок 3.3 - Код программы для открытия MS Excel**

Данный код (рисунок 3.3) открывает файл MS Excel «vullist.xlsx» с помощью библиотеки «xlrd2».

1. 1)*workbook = xlrd2.open\_workbook(*'vullist.xlsx') - открывает файл MS Excel `vullist.xlsx` и сохраняет его в переменную `workbook`.

2)*sheet = workbook.sheet\_by\_index(0)* - выбирает первый лист в книге и сохраняет его в переменную «sheet».

Фрагмент кода для поиска количество уязвимостей. **(Рисунок 3.4)**



**Рисунок 3.4 - Код программы для поиска кол-во уязвимостей**

*names = sheet.col\_values(4)* – присваивание переменной столбца, отвечающего за название ПО;

*danger\_levels = sheet.col\_values(12)* – присваивание переменной столбца, отвечающего за уровни опасности;

*classes = sheet.col\_values(8)* – присваивание переменной столбца, отвечающего за классы угроз;

*hah = danger\_super + danger\_hight + danger\_middle + danger\_low* – сумма всех угроз;

*if names[i].find('Windows') >= 0* – при наличии совпадения со словом «Windows»:

*if classes[i].find('код') >= 0* – при наличии совпадения со строкой «арх» :

*code += 1* – счетчик + 1;

*if danger\_levels[i][0] == 'К':* # если первая буква «К»

*danger\_super += 1* – соответсвующий счетчик +1;

*lab\_krit\_1.config(text=danger\_super) –* передача соответствующего значения в нужный виджет;

5. Фрагмент кода Word **(Рисунок 3.5)**



**Рисунок 3.5 - Фрагмент кода Word**

*document = Document()* = создаем документ

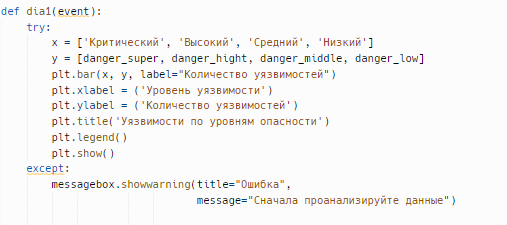
*table = document.add\_table(rows=8, cols=2)* = добавляем определенное количество строк и колонок

*table.cells(0,1).text* *= “Критический”* - добавляем название для первой строчки

*table.cells(1, 1).text = str(danger\_super)* = добавляем количество уязвимостей

*doc.save('test.docx')* = сохраняем наш документ

6. Фрагмент кода для диаграммы №1. **(Рисунок 3.6)**



**Рисунок 3.6 - Фрагмент кода для диаграммы №1**

*x=[‘Критический’, ‘Высокий’, ‘Средний’, ‘Низкий’ ] –* ось по уровню уязвимостей;

*y=[danger\_super, danger\_hight, danger\_middle, danger\_low] –* числовая ось по попеременным;

*plt.xlabel = ('Уровень уязвимости') –* подпись оси х;

*plt.ylabel = ('Количество уязвимостей')* – подпись оси у;

*plt.title('Уязвимости по уровням опасности')* – подпись окна диаграммы;

*plt.show()* – утверждение диаграммы;

7. Фрагмент кода для смены темы окна **(Рисунок 3. 7):**

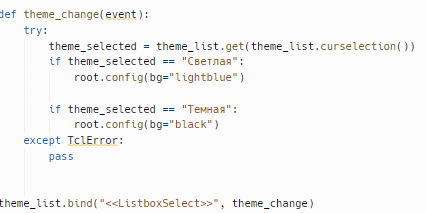


Рисунок 3.7 – Фрагмент кода для смены темы окна

*theme\_selected = theme\_list.get(theme\_list.curselection())* – подключение к листу тем Listbox;

*if theme\_selected == "Светлая":* - если тема выбрана светлой;

*root.config(bg="lightblue")* – изменить цвет заднего фона всего окна;

8. Фрагмент кода расположений виджетов и привязки к объекту **(Рисунок 3.8 ):**



Рисунок 3.8 – Фрагмент кода для расположения виджета и привязки кода

*word\_button.bind("<Button-1>", word) – bind*- метод привязки;

*<Button-1>* - способ активации кнопки;

*word* – функция, к которой идет привязка;

*word\_button.place(x=0, y=150)* – х – координата по оси х;

- у – координата по оси у;

**Таким образом**, с использованием множества функций, в основном имеющих в своем составе числовые вычисления и методов привязки виджетов, функционал приложения реализован.

# **Заключение**

В ходе выполнения курсового проекта были рассмотрены основы разработки приложения, уточняя, пользовательского интерфейса на платформе Replit, за счет своего функционала, позволяющего пропустить этап установки стороннего ПО. Разработка осуществлялась с помощью языка программирования Python.

Затрагивая самую важную библиотеку, использованную в проекте – Tkinter, выделим самые главные объекты, на которых и строится интерфейс: Label, Button, Listbox, Messagebox (Всплывающее окно). Основными библиотеками стали: xlrd2, python-docx, Tkinter, matpolib, которые мы изучили и использовали. Именно эти библиотеки, по личному опыту, оказались наиболее комфортными в работе.

Для реализации функционала была произведена обьемная работа с файлами Excel и Word. Базой данных в проекте, выступил vullist.xlsx - файл Excel, содержащий данные об уязвимостях различного ПО, предоставленного ФСТЭК-ом.

В приложение были добавлены удобные функции, такие, как: поиск угроз, просмотр количества уязвимостей по классу, просмотр диаграмм, общего количества уязвимостей и смена темы окна. Немало важно то, что приложение позволяет сохранить данные в MS Word - это добавляет приложению функциональности и упрощает работу с данными.

Было написано руководство пользователя, содержащее инструкции по использованию приложения, а также описание доступных функций и возможностей их работы. Приложение было протестировано на соответствие требованиям и корректность работы, были выявлены и исправлены ошибки и недочеты в функциональности и интерфейсе приложения.

# **Список использованных источников**

1. БДУ – уязвимости: [bdu.fstec.ru]. URL: https://bdu.fstec.ru/vul
2. Библиотека Matplotlib для построения графиков. — Текст : электронный // : [сайт]. — URL: https://skillbox.ru/media/code/biblioteka-matplotlib-dlya-postroeniya-grafikov/ (дата обращения: 05.06.2025).
3. Руководство по Tkinter [metanit]. <URL:https://metanit.com/python/tkinter/>
4. Глоссарий SkillFactory. Python — URL: https://blog.skillfactory.ru/glossary/python/ (дата обращения: 01.05.2025).
5. История Python // ВКонтакте. — Режим доступа: https://vk.com/@makehtmleducation-python-history (дата обращения: 24.04.2025).
6. Лучшие IDE и редакторы кода для Python: [tproger.ru]. URL:https://tproger.ru/translations/python-ide/
7. Tkinter. Программирование GUI на Python: [younglinux.info]. URL:  <https://younglinux.info/tkinter.php>
8. Основы PyCharm // Хабр URL: https://habr.com/ru/articles/720480/ (дата обращения: 15.04.2025).
9. Основы Tkinter: [metanit]. <URL:https://metanit.com/python/tkinter/1.1.php>
10. Пишем десктоп-приложение на Python с помощью Tkinter: [skillbox.ru]. URL:https://skillbox.ru/media/code/pishem-desktopprilozhenie-na-python-s-pomoshchyu-tkinter/

# **Приложение А Исходный код приложения**

from docx import Document

from docx import shared

import docx

from tkinter import \*

import xlrd2

import matplotlib.pyplot as plt

from tkinter import messagebox

workbook = xlrd2.open\_workbook('vullist.xlsx')

sheet = workbook.sheet\_by\_index(0)

row = sheet.nrows

print('Всего записей:', row)

def analyz(event):

global names

names = sheet.col\_values(4)

global danger\_levels

danger\_levels = sheet.col\_values(12)

global classes

classes = sheet.col\_values(8)

global arch, code, fact

arch, code, fact = 0, 0, 0

global danger\_super, danger\_hight, danger\_middle, danger\_low

danger\_super, danger\_hight, danger\_middle, danger\_low = 0, 0, 0, 0

global hah

hah = danger\_super + danger\_hight + danger\_middle + danger\_low

try:

for i in range(3, row):

if names[i].find('Windows') >= 0:

if classes[i].find('код') >= 0:

code += 1

if classes[i].find('арх') >= 0:

arch += 1

if classes[i].find('факт') >= 0:

fact += 1

if danger\_levels[i][0] == 'К': # Критический

danger\_super += 1

elif danger\_levels[i][0] == 'В': # Высокий

danger\_hight += 1

elif danger\_levels[i][0] == 'С': # Средний

danger\_middle += 1

else: # Низкий

danger\_low += 1

lab\_krit\_1.config(text=danger\_super)

lab\_krit\_2.config(text=danger\_hight)

lab\_krit\_3.config(text=danger\_middle)

lab\_krit\_4.config(text=danger\_low)

lab\_all\_num.config(text=danger\_hight + danger\_low + danger\_super +

danger\_middle)

except:

pass

root = Tk()

root.title("УБД")

root.geometry("600x600")

root.config(bg="lightblue")

lab\_name = Label(root,

bg="dark blue",

fg="white",

text="Уязвимости ОС Windows",

font="Arial 14",

width=100)

lab\_krit = Label(

root,

bg='red',

fg='Black',

width=20,

height=3,

text="Критический",

)

lab\_krit\_1 = Label(root, bg='white', width=20, height=3)

lab\_high = Label(root,

bg='pink',

fg='black',

width=20,

height=3,

text="Высокий")

lab\_krit\_2 = Label(root, bg='white', width=20, height=3)

lab\_mid = Label(root,

bg='yellow',

fg='black',

width=20,

height=3,

text="Средний")

lab\_krit\_3 = Label(root, bg='white', width=20, height=3)

lab\_low = Label(root,

bg='green',

fg='white',

width=20,

height=3,

text="Низкий")

lab\_krit\_4 = Label(root, bg='white', width=20, height=3)

lab\_all = Label(root, bg='grey', fg='black', width=25, height=3, text="Все:")

lab\_all\_num = Label(root, bg='white', fg='black', width=25, height=3)

dia1\_but = Button(

root,

bg="white",

width=20,

height=2,

text="Диаграмма 1",

fg="black",

)

dia2\_but = Button(

root,

bg="white",

width=20,

height=2,

text="Диаграмма 2",

fg="black",

)

def dia1(event):

try:

x = ['Критический', 'Высокий', 'Средний', 'Низкий']

y = [danger\_super, danger\_hight, danger\_middle, danger\_low]

plt.bar(x, y, label="Количество уязвимостей")

plt.xlabel = ('Уровень уязвимости')

plt.ylabel = ('Количество уязвимостей')

plt.title('Уязвимости по уровням опасности')

plt.legend()

plt.show()

except:

messagebox.showwarning(title="Ошибка",

message="Сначала проанализируйте данные")

dia1\_but.bind("<Button-1>", dia1)

def dia2(event):

try:

labels = [

"Уязвимость архитектуры", "Уязвимость кода",

"Уязвимость многофакторная"

]

vals = [arch, code, fact]

plt.pie(vals, labels=labels)

plt.title('Уязвимости по типам ')

plt.legend()

plt.show()

except:

messagebox.showwarning(title="Ошибка",

message="Сначала проанализируйте данные")

dia2\_but.bind("<Button-1>", dia2)

#конец пновки по дате/за все время

lab\_arc = Label(root,

bg='grey',

fg='black',

width=25,

height=3,

text="Уязвимости по архитектуре:")

lab\_arc\_num = Label(root, bg='white', fg='black', width=25, height=3)

def arc\_b(event):

try:

lab\_arc\_num.config(text=arch)

except:

messagebox.showwarning(title="Ошибка",

message="Сначала проанализируйте данные")

arc\_button = Button(root,

bg='white',

fg='black',

width=20,

height=3,

text="По архитектуре")

arc\_button.bind('<Button-1>', arc\_b)

lab\_code = Label(root,

bg='grey',

fg='black',

width=25,

height=3,

text="Уязвимости по коду:")

lab\_code\_num = Label(root, bg='white', fg='black', width=25, height=3)

def code\_b(event):

try:

lab\_code\_num.config(text=code)

except:

messagebox.showwarning(title="Ошибка",

message="Сначала проанализируйте данные")

code\_button = Button(root,

bg='white',

fg='black',

width=20,

height=3,

text="По коду")

code\_button.bind('<Button-1>', code\_b)

def fact\_b(event):

try:

lab\_fact\_num.config(text=fact)

except:

messagebox.showwarning(title="Ошибка",

message="Сначала проанализируйте данные")

fact\_button = Button(root,

bg='white',

fg='black',

width=20,

height=3,

text="По фактору")

fact\_button.bind('<Button-1>', fact\_b)

lab\_fact = Label(root,

bg='grey',

fg='black',

width=25,

height=3,

text=("Уязвимости по факту:"))

lab\_fact\_num = Label(root, bg='white', fg='black', width=25, height=3)

ot\_label = Label(root, bg="light blue", fg="black", width=5, text="От:")

do\_label = Label(root, bg="light blue", fg="black", width=5, text="До:")

#кнопка очистки

clean\_button = Button(root, bg="white", width=20, height=3, text="Очистить")

def clean(event):

try:

lab\_krit\_1.config(text="")

lab\_krit\_2.config(text="")

lab\_krit\_3.config(text="")

lab\_krit\_4.config(text="")

lab\_all\_num.config(text="")

lab\_arc\_num.config(text="")

lab\_code\_num.config(text="")

lab\_fact\_num.config(text="")

except:

pass

clean\_button.bind("<Button-1>", clean)

# конец кнопки очистки

#analiz

analyze\_button = Button(root,

bg="white",

fg="black",

width=20,

height=3,

text="Проанализировать данные")

analyze\_button.bind("<Button-1>", analyz)

#analiz end

lab\_name.pack(side=TOP)

dia1\_but.place(x=150, y=28)

dia2\_but.place(x=350, y=28)

analyze\_button.place(x=0, y=75)

clean\_button.place(x=0, y=225)

arc\_button.place(x=0, y=300)

code\_button.place(x=0, y=375)

fact\_button.place(x=0, y=450)

lab\_krit.place(x=270, y=74)

lab\_krit\_1.place(x=270, y=124)

lab\_high.place(x=270, y=184)

lab\_krit\_2.place(x=270, y=234)

lab\_mid.place(x=270, y=294)

lab\_krit\_3.place(x=270, y=344)

lab\_low.place(x=270, y=404)

lab\_krit\_4.place(x=270, y=454)

lab\_all.place(x=520, y=74)

lab\_all\_num.place(x=520, y=124)

#theme

theme\_label = Label(root,

bg="grey",

fg="black",

width=25,

height=3,

text="Тема окна")

themes = ["Светлая", "Темная"]

themes\_var = Variable(value=themes)

theme\_list = Listbox(

root,

bg="white",

fg="black",

width=25,

height=3,

listvariable=themes\_var,

)

def theme\_change(event):

try:

theme\_selected = theme\_list.get(theme\_list.curselection())

if theme\_selected == "Светлая":

root.config(bg="lightblue")

if theme\_selected == "Темная":

root.config(bg="black")

except TclError:

pass

theme\_list.bind("<<ListboxSelect>>", theme\_change)

theme\_label.place(x=520, y=184)

theme\_list.place(x=520, y=234)

#theme end

#word

word\_button = Button(root,

bg="white",

width=20,

height=3,

text="Вывести результаты в Word")

def word(event):

try:

doc = docx.Document()

table = doc.add\_table(rows=8, cols=2)

table.style = 'Table Grid'

table.cell(0, 1).text = "Критический"

table.cell(1, 1).text = "Высокий"

table.cell(2, 1).text = "Средний"

table.cell(3, 1).text = "Низкий"

table.cell(4, 1).text = "Всего"

table.cell(5, 1).text = "Уязвимости по архитектуре"

table.cell(6, 1).text = "Уязвимости по коду"

table.cell(7, 1).text = "Уязвимости по факту"

table.cell(0, 0).text = str(danger\_super)

table.cell(1, 0).text = str(danger\_hight)

table.cell(2, 0).text = str(danger\_middle)

table.cell(3, 0).text = str(danger\_low)

table.cell(4, 0).text = str(hah)

table.cell(5, 0).text = str(arch)

table.cell(6, 0).text = str(code)

table.cell(7, 0).text = str(fact)

doc.save('table.docx')

except:

pass

word\_button.bind("<Button-1>", word)

word\_button.place(x=0, y=150)

#word end

lab\_arc.place(x=520, y=294)

lab\_arc\_num.place(x=520, y=344)

lab\_code.place(x=520, y=404)

lab\_code\_num.place(x=520, y=454)

lab\_fact.place(x=520, y=524)

lab\_fact\_num.place(x=520, y=574)

root.mainloop()

# **Приложение Б Руководство пользователя**