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 и тестирование автоматизированной 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. обеспечить ведение базы данных;
  4. организация различных уровней доступа;
  5. использование авторизации;
  6. модульное тестирование продукта;

1. Содержание пояснительной записки:

* титульный лист;
* задание к курсовому проекту;
* содержание;
* введение;
* обследование предметной области;
* формулирование требований и технического задания;
* проектирование информационной системы с помощью UML-диаграмм;
* реализация проекта информационной системы средствами языка Python и СУБД MySQL;
* обоснование выбора программных средств;
* разработка базы данных «Студенты ВУЗа»;
* описание состава (модулей) и логики работы программы;
* разработка модульных тестов и проведение тестирования 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](#_Toc199032716)

[**1** **ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ «СТУДЕНТЫ ВУЗА».** 6](#_Toc199032717)

[**Обследование предметной области** 6](#_Toc199032718)

[**Техническое задание и требования** 7](#_Toc199032719)

[**Проектирование информационной системы с помощью UML-диаграмм.** 13](#_Toc199032720)

[**2** **РЕАЛИЗАЦИЯ ПРОЕКТА ИНФОРМАЦИОННОЙ СИСТЕМЫ СРЕДСТВАМИ ЯЗЫКА PYTHON И СУБД MYSQL** 16](#_Toc199032721)

[**Обоснование выбора программных средств** 16](#_Toc199032722)

[**Разработка базы данных «Студенты вуза»** 17](#_Toc199032723)

[**Описание состава (модулей) и логики работы программы** 19](#_Toc199032724)

[**Разработка модульных тестов и проведение тестирования приложения;** 31](#_Toc199032725)

[**Заключение** 32](#_Toc199032726)

[**Список использованных источников** 33](#_Toc199032727)

[**Приложение А Исходный код приложения** 34](#_Toc199032728)

[**Приложение Б Руководство пользователя** 35](#_Toc199032729)

# **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 умений, таких как навык планирования и анализа разработки коммерческого или некоммерческого продукта, работы с данными в формате SQL-таблиц, автоматизации и оптимизации процессов, обеспечения 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.

**Цель проекта**: поэтапная реализация автоматизированного программного обеспечения работы с базой данных информационной системы «Студенты ВУЗа» 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;

# **1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ «СТУДЕНТЫ ВУЗА».**

В данном разделе будут рассмотрены теоретические аспекты разработки нашей информационной системы, для начала нам нужно определиться с предметной областью, проанализировать ее и описать. Требуется понять, кем она будет использоваться, какой функционал она должна нести и в общем – зачем она нужна.

# **Обследование предметной области**

Выбранная предметная область представляет собой высшее учебное заведение, заведомо заинтересованное в учете обучающихся в данных момент студентов. Потребности кроме учета могут разниться и на деле их огромное множество, далее мы опишем их подробно. Лица, которые пользуются и, соответственно, допущены к пользованию информационной системой, делятся на три категории: преподаватели, администраторы и сторонний персонал.

Администраторы, это те, кто обычно занимается учетом студентов, они следят за тем, чтобы отчисленных убирали из всех списков, а новых студентов добавляли, они также отвечают за корректность данных, указанных студентом. Эти работники особенно нуждаются в функциях редактирования, а, так же как и для остальных групп в легкой навигации. Преподавателям же может потребоваться добавить новоиспеченного студента в систему, если пока этого не случилось, например, после перевода из одного учебного заведения в другое. Да и использование информации для заполнения документов – тоже нужно и важно. Для прочих работников, таких как бухгалтерия, тоже важно использовать информацию об обучающихся.

Создание информационной системы выступает как автоматизация внутренних процессов учебного заведения. Цифровизация добавляет комфорт в работу сотрудников, увеличивает скорость их работы и экономит место, размещать всю информацию на бумаге – дорого и неудобно, еще и не безопасно.

Однако важно отметить, что наша предметная область очень обширна и многослойна, поэтому реализовать досконально точную информационную систему, отвечающую всем потребностям и, важно отметить, реальности – тяжело. Как и говоря о том, что такой цели поставлено не было. Поэтому формирование и обследование предметной области ограничено и упрощено. Для курсового проекта нам нужна лишь модель, пример того, для чего мы создаем информационную систему.

# **Техническое задание и требования**

1. Предметная область: ВУЗ

2. Этап разработки раздела «Общие сведения»:

Полное наименование ИС: Студенты ВУЗа

Предприятие-разработчик системы: Герасименко София Евгеньевна

Предприятие-заказчик системы: ВУЗ N/СИБАДИ

Система создается на основании технического задания (ТЗ). ТЗ на АС является основным документом, определяющим требования и порядок создания автоматизированной системы, в соответствии с которым проводится разработка АС и ее приемка при вводе в действие. Кроме того, при создании системы используются ГОСТ 34.602-89 “Техническое задание на создание автоматизированной системы”.

Плановый срок начала работ: 3 мая 2025

Плановый срок окончания работ: 27 мая 2025

Автоматизируемая система создается на некоммерческой основе.

Порядок оформления и предъявления заказчику результатов работы по созданию системы определяется после получения начальной версии продукта, в которой должны быть реализованы все основные функции, определенные в ТЗ и утвержденные заказчиком.

3. Этап разработки раздела «Назначение и цели создания системы»:

Вид автоматизируемой деятельности: Ведение учета учащихся

Перечень автоматизируемых процессов: Учет, редактирование, доступ, навигация

Наименование и значение показателей, которые будут достигнуты в результате внедрения БД:

* увеличение скорости работы сотрудников
* сохранение целостности информации
* уменьшение физических носителей информации, конкретно – бумажных

4. Этап разработки раздела «Характеристики объекта автоматизации»

Стандартный региональный ВУЗ, состоящий из N-количества сотрудников и N-количества обучающихся (далее, от 20 человек).

Организационная структура изображена на рисунке 2.

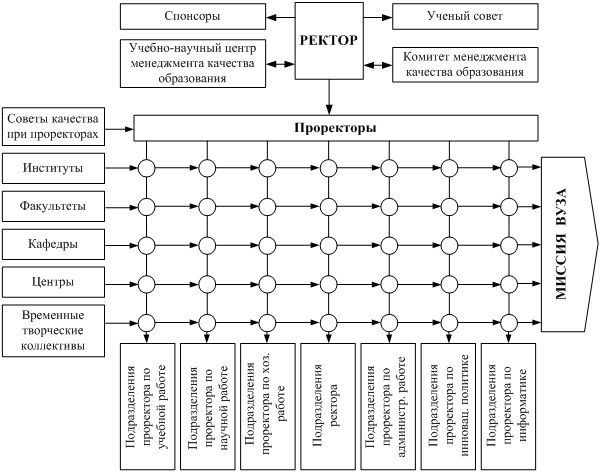


Рисунок 1– Организационная структура ВУЗа

Описание автоматизируемых процессов, информационные потоки автоматизируемых процессов: ведение списка обучающихся, навигация по спискам, редактирование данных, обеспечение уровней доступа или сохранение целостности данных.

5. Этап разработки раздела «Требования к ИС»

Информационная система должна состоять из базы данных на основе языка SQL, с интуитивно-понятным графическим интерфейсом. Должны быть осуществлены следующие требования:

1. Разграничение пользователей по группам/уровням доступа, только администраторы имеют право на удаление студента из базы данных, инструменты навигации доступны всем, элемент добавления студента доступен администраторам и преподавателям;
2. Наличие функций удаления, добавления студента в таблицу графического интерфейса и соответственно в базу данных;
3. Обеспечение навигации по таблице с помощью фильтров и поиска по совпадению;

ИС должна соответствовать требованиям технического задания на ее создание и развитие, а также требованиям нормативно-технических документов, действующих в ведомстве заказчика ИС.

Ввод в действие ИС должен приводить к полезным технико-экономическим, социальным результатам:

1. Повышение комфорта в рабочей среде для сотрудников
2. Экономия средств на физические носители информации

Технические средства ИС должны быть установлены так, чтобы обеспечивались их безопасная эксплуатация и техническое обслуживание. Требования безопасности устанавливаются в инструкциях по эксплуатации технических средств.

Требования к функциям, выполняемым системой:

1. Иметь высокую скорость выполнения
2. Исправно выполнять свою работу

Данная информационная система разрабатывается с расчетом на нескольких пользователей – администраторов, преподавателей и персонала. Таким образом, разрабатываемая система должна обеспечивать решение вышеперечисленных задач.

В готовом виде она должна быть максимально простой и удобной: все операции должны выполняться с помощью элементарных действий пользователя.

Требования к программному обеспечению ИС:

1. Поддержка Python 3.6 и выше
2. Требования к техническому обеспечению АС:

Минимальные требования:

|  |  |
| --- | --- |
| Компонент | Требования |
| Процессор | 1 ГГц (x86/x64) |
| ОЗУ | 512 МБ (1 ГБ для комфортной работы) |
| Место на диске | 50 МБ (для Python и зависимостей) |
| Экран | Разрешение 1024×768 |

Таблица 1 - Минимальные требования

Рекомендуемые характеристики:

|  |  |
| --- | --- |
| Компонент | Рекомендации |
| Процессор | 2+ ядерный, 2 ГГц |
| ОЗУ | 2 ГБ+ |
| ОС | Windows 10+, macOS 10.15+, Linux с современным DE |

Таблица 2- Рекомендуемые характеристики

6. Поддерживаемые операционные системы

Windows (7/10/11, x86/x64)

macOS (10.15 Catalina и новее)

Linux (Ubuntu 20.04+, Fedora 32+, Debian 11+ с графической средой)

7. Этап разработки раздела «Стадии и этапы разработки»

Разработка должна быть проведена в три стадии:

* разработка технического задания;
* рабочее проектирование;
* внедрение.

8. Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка модели автоматизируемых процессов и функциональной модели ИС;
2. разработки логической и физической моделей данных;
3. разработка программы;
4. разработка программной документации;
5. испытания программы.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах заказчика.

Приемо-сдаточные испытания должны проводиться на объекте заказчика в оговоренные сроки. Приемо-сдаточные испытания программы должны проводиться согласно разработанной исполнителем и согласованной заказчиком программы и методик испытаний. Ход проведения приемо-сдаточных испытаний заказчик и исполнитель документируют в протоколе проведения испытаний. На основании протокола проведения испытаний исполнитель совместно с заказчиком подписывает акт приемки-сдачи программы в эксплуатацию.

# **Проектирование информационной системы с помощью UML-диаграмм.**

Проектирование информационной системы нужно начать с составления сценария работы, а именно, нужно представить, как пользователи будут действовать в рамках системы, какие действия им нужно и придется выполнять. Поскольку у каждой категории есть определенные задачи и значимость, функции, которые они могут выполнять среди базы данных должны и будут ограничены. Систематизировав и обозначив эти зависимости в Use-Case диаграмме, мы можем наблюдать за взаимодействиями пользователей с этой информационной системой (Рис.2).

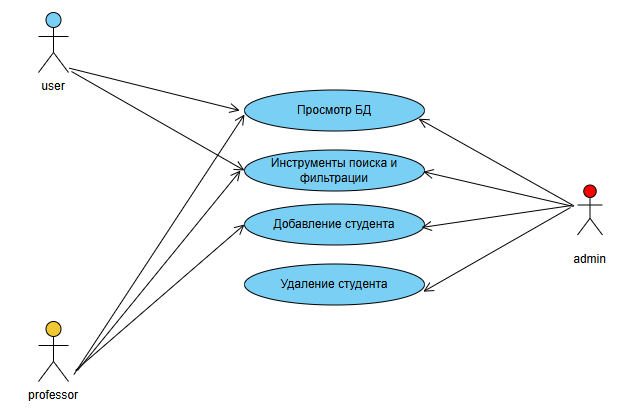


Рисунок 2 - Use-Case диаграмма информационной системы "Студенты ВУЗа"

На диаграмме обозначены «user», «professor» и «admin». Соответственно: пользователь (персонал), преподаватель и администратор.

Когда разработка приблизилась к созданию кода, нужно разобраться, какие взаимосвязи будут между элементами проекта и в какой последовательности будет действовать пользователь уже в отношении программного обеспечения. Для этого созданы UML-диаграммы последовательности и UML-диаграммы классов. В диаграмме последовательности мы проследим действия пользователя от входа в систему, до всех функций, которые можно использовать. С помощью этой диаграммы будет наглядно описаны зависимости между запросами пользователя и последовательностью работы кода. Непосредственно эту схему можно изучить на рисунке 3.

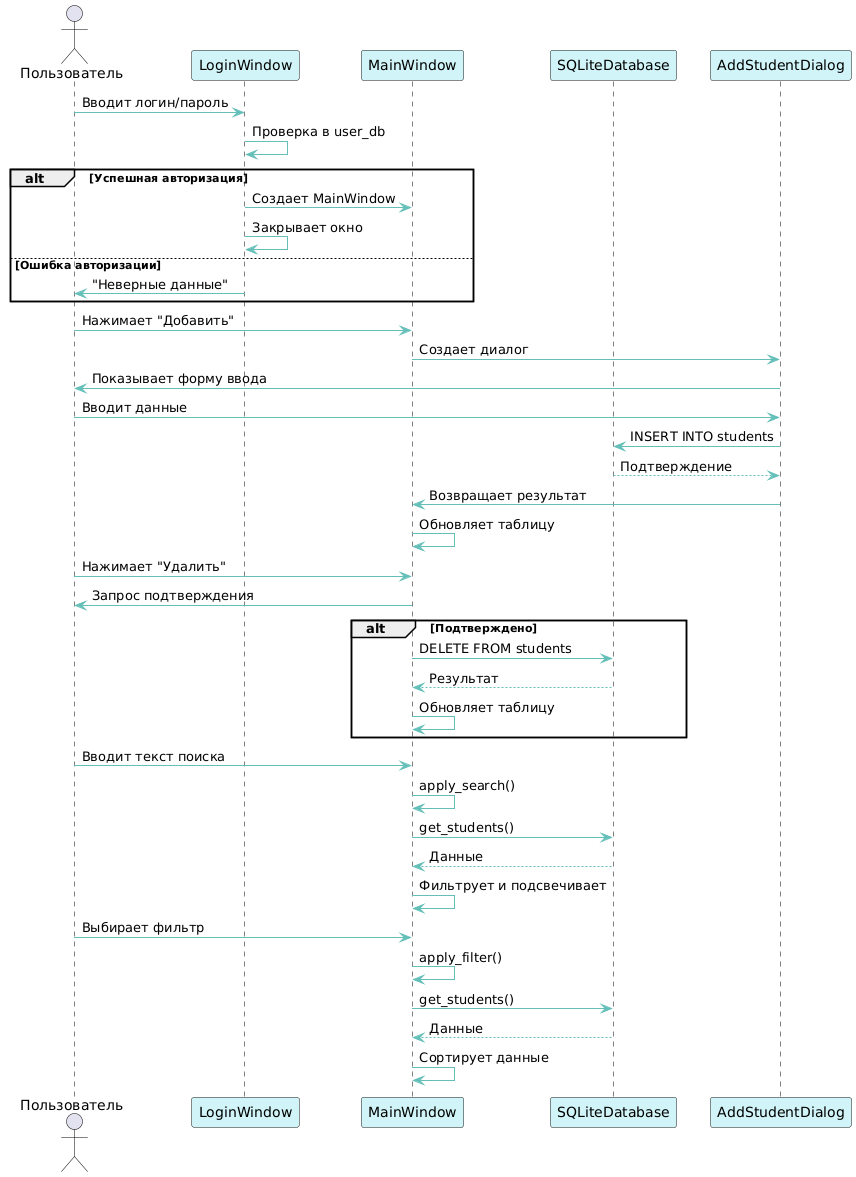
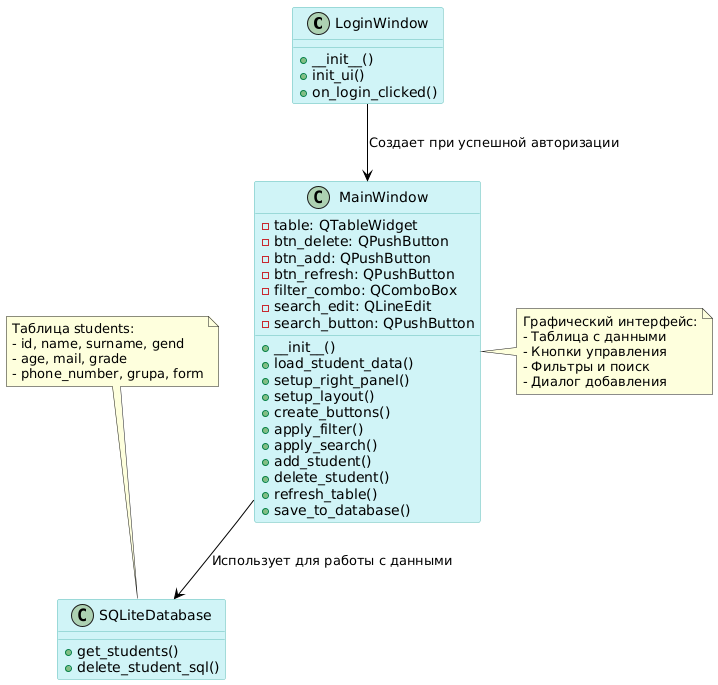


Рисунок 3 – Диаграмма последовательности



# **2 РЕАЛИЗАЦИЯ ПРОЕКТА ИНФОРМАЦИОННОЙ СИСТЕМЫ СРЕДСТВАМИ ЯЗЫКА PYTHON И СУБД MYSQL**

Как только мы спланировали и спроектировали нашу информационную систему, пришло время ее создания как полноценного приложения. В данной главе будет описан выбор средств, процесс кодирования и общие принципы работы, также мы проведем тестирование системы.

# **Обоснование выбора программных средств**

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
* 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.

SQLite3 является встроенной библиотекой языка Python для работы с одноименной реляционной системой управления базами данных. В отличие от клиент-серверных СУБД, SQLite реализует бессерверную архитектуру, что делает её идеальным решением для локального хранения структурированных данных в desktop-приложениях. Дополнительно, оказался наиболее удобным в работе.

# **Разработка базы данных «Студенты вуза»**

В отношении нашей информационной системы стоит начать c формирования источника информации - базы данных, содержащей требуемые категории данных, таких как:

* личные данные (ФИО, пол, возраст)
* контактные данные (электронная почта, номер мобильного телефона)
* специальные данные (курс обучения, название группы, форма обучения)

Реализовать данную базу будем с помощью SQL-таблиц, на основе СУБД – SQlite Studio и библиотеки sqlite3. Создаем файл в проекте с названием «sql.py» и импортируем библиотеку. Кратко изучив синтаксис SQL, сформируем базу данных с названием «students.db», для этого мы должны установить соединение с sqlite3 и записать нашу «шапку» таблицы – виды данных студентов, дополнительно уточняя тип данных и допустимое количество символов записи. Стоит уточнить, что мы прописываем создание пустой таблицы только в том случае, если она еще не существует, в коде – « CREATE TABLE IF NOT EXISTS». Полный код создания пустой таблицы находится на рисунке 5.

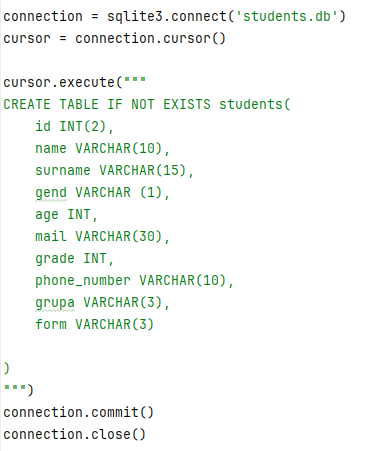


Рисунок 5 – Создание пустой базы данных

Далее, заполняем или дословно вставляем в таблицу записи о 20-ти тестовых студентах конструкцией «INSERT INTO». В коде это выглядит как монолитный поток текста, также стоит отметить, что после каждой манипуляции с базой, мы подтверждаем изменения конструкцией «.commit», отключаемся от базы и подключаемся заново. Такие действия можно охарактеризовать строго предосторожностью. На этом работа с созданием базы данных закончена и по значениям она примет следующий вид (см. рис. 6). Подтверждаем изменения и закрываем соединение.



Рисунок 6 – Заполнение данными

Таким образом, мы осуществили формирование базы данных и готовы приступать к формированию графического интерфейса.

# **Описание состава (модулей) и логики работы программы**

Создание графического интерфейса оказалось самым сложным и объемным процессом, по истечению которого, мы получили следующие результаты (Рис.7-8).

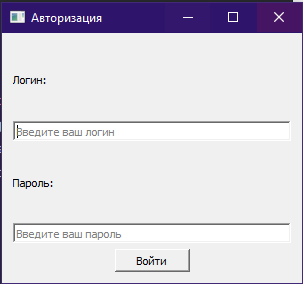


Рисунок 7 – Окно авторизации

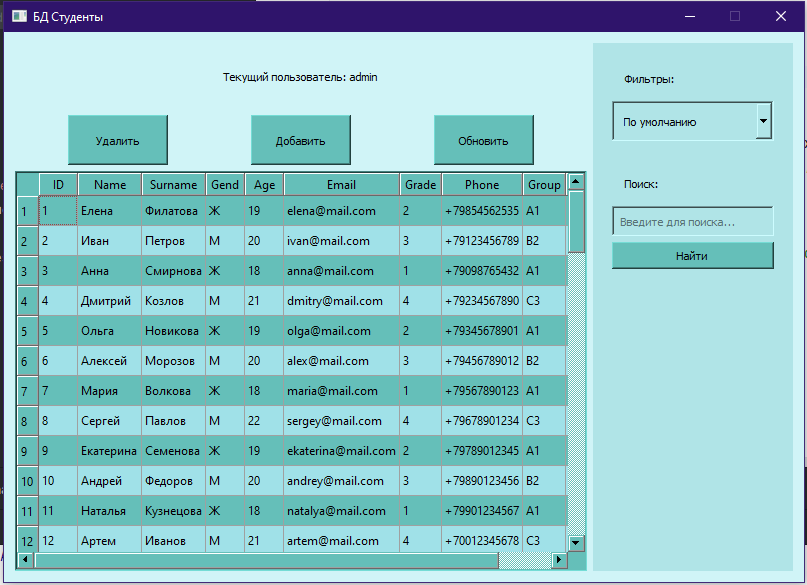


Рисунок 8 – Основное окно приложения

Для осуществления и запуска приложения в файле «sqlGUI.py» потребуются следующие библиотеки и функции:

1. import sys - используется для работы с системными функциями;
2. import re - стандартная библиотека для работы с регулярными выражениями;
3. from PyQt5 import Qt - основной модуль фреймворка pyqt5, содержит базовые классы для создания gui-приложений;
4. from PyQt5.QtCore import Qt as QtCoreQt, QSize  
   from PyQt5.QtGui import QFont, QIcon  
   from PyQt5.QtWidgets import (QApplication, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QMessageBox, QHBoxLayout, QFrame, QTableWidget, QTableWidgetItem, QInputDialog, QComboBox)- модули PyQt5 для графических элементов;
5. import sqlite3 – библиотека базы данных;
6. from sql import get\_students, delete\_student\_sql – функции;

**Разбор окна авторизации**

Для создания окна авторизации, как и для последующих окон, мы создаем собственный класс. Мы выберем имя «LoginWindow», типа QWidget – виджет.

class LoginWindow(QWidget):  
 def \_\_init\_\_(self): - опишем метод инициализации класса  
 super().\_\_init\_\_() – позволим вызывать метод из родительских классов в определенном порядке  
 self.setWindowTitle('Авторизация')- укажем имя окна  
 self.setGeometry(200, 200, 300, 250)- установим размер окна  
 self.init\_ui()- метод создания остальных составляющих окна

В методе init\_ui()мы задействуем следующие модули:

1. QLabel – текстовая метка для обозначения окон ввода (см. рис. 9)
2. QLineEdit – окно ввода для логинов и паролей (см. рис. 10)
3. QPushButton – кнопка (см. рис. 11)
4. QVBoxLayout – модуль раскладки виджетов

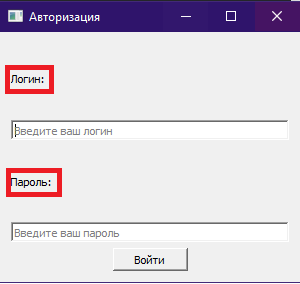


Рисунок 9 – QLabel

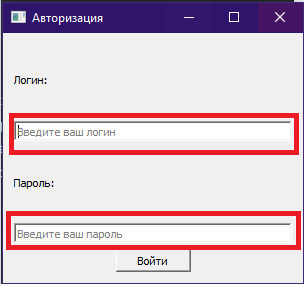


Рисунок 10 – QlineEdit

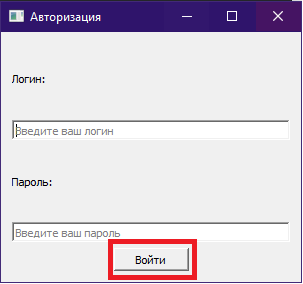


Рисунок 11- QPushButton

Полный код метода init\_ui (Рис.12):



Рисунок 12 - Полный код

Использованные функции:

1. setPlaceholderText – устанавливает текстовую подсказку в окне ввода
2. connect – соединяет интерактивный элемент с функцией
3. addWidget – добавляет виджет в раскладку
4. addStretch – растягивает элемент по раскладке
5. addLayout – добавляет раскладку в раскладку

В методе задействовано две раскладки, горизонтальная QHBoxLayout и вертикальная QVBoxLayout, посредством размещения горизонтальной раскладки в вертикальную мы получаем вертикальное расположение текстовых меток и полей ввода, а внизу одну кнопку посередине. По этому принципу работают все раскладки в коде.

Переходя к основной части – авторизации, мы подключаем функцию on\_login\_clicked к кнопке «Вход», полный код изображен ниже (рис.13).



Рисунок 13- Полный код функции on\_login\_clicked

Поясним принцип работы авторизации:

|  |
| --- |
| def on\_login\_clicked(self):  login = self.edit\_login.text()- собираем информацию из поля ввода для логина;  password = self.edit\_password.text() – для пароля;  auth = False  for user in user\_db:- ищем в массиве с логинами и паролями;  if user[0] == login and user[1] == password: - если логин соответствует правильному паролю ;  auth = True – авторизация истинна;  global current\_user  current\_user = login – нынешний пользователь это логин;  break   if auth:  self.main\_window = MainWindow()- если авторизация истина;  self.main\_window.current\_user\_label.setText(f"Текущий пользователь: {current\_user}") – установка текстовой метки нынешнего пользователя ;  self.main\_window.show()- открытие основного окна;  self.close()- окно авторизации закрывается;   else:  QMessageBox.warning(self, 'Ошибка', 'Неверные данные или доступ отклонен')- обработка неверного ввода; |

Таким образом, мы создали окно авторизации и теперь переходим к основному окну.

**Разбор основного окна**

Аналогично окну авторизации создаем класс и прописываем элементы (рис.14):



Рисунок 14- Основное окно и его элементы

Использованные модули:

1. QLabel – текстовая метка для нынешнего пользователя (см. рис. 15)
2. QPushButton – кнопка (см. рис. 16)
3. QV/HBoxLayout – модули раскладки виджетов
4. QTableWidget- модуль таблицы (см. рис.17)

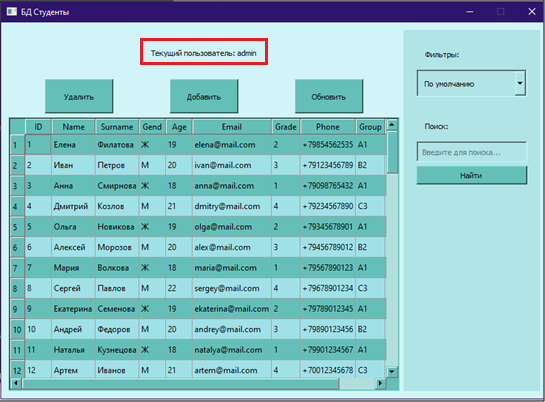


Рисунок 15 – QLabel

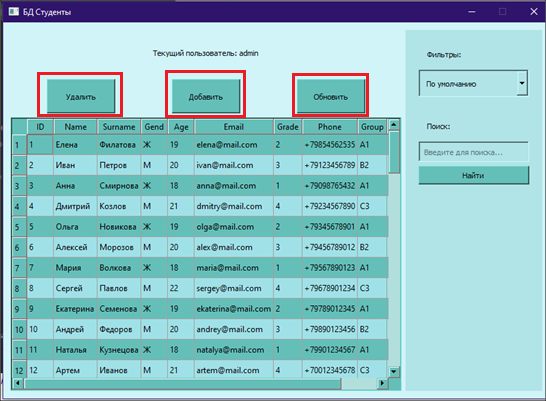


Рисунок 16 - QPushButton

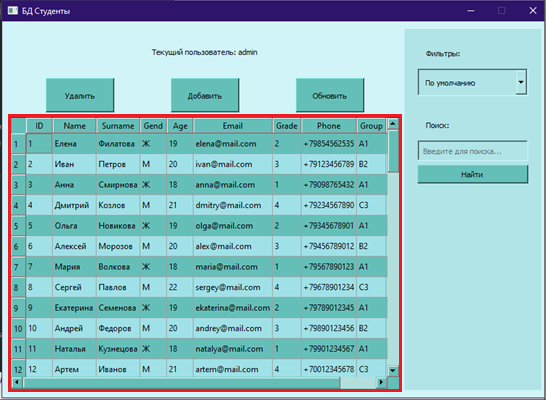


Рисунок 17- QTableWidget

Использованные функции:

1. setFixedSize – установить фиксированный размер;
2. setStyleSheet – установить стиль, синтаксис CSS;
3. setAlignment(QtCoreQt.AlignCenter)- установить расположение по раскладке, конкретно – по центру;
4. setFixedHeight – установить фиксированную высоту;
5. setEditTriggers(QTableWidget.NoEditTriggers) – отключение интерактивности таблицы;
6. load\_student\_data()  
   setup\_right\_panel()  
   create\_buttons()  
   setup\_layout() – функции/методы, далее подробнее;

Метод setup\_layout – основная раскладка окна, код можно наблюдать на рисунке 18.



Рисунок 18 - Метод setup\_layout

Список раскладок:

1. buttons\_layout – включает в себя кнопки;
2. user\_controls – включает одну метку нынешнего пользователя;
3. left\_layout - включает в себя раскладки выше;
4. right\_panel – элемент правой панели (рамка)далее подробнее;
5. main\_layout – включает в себя раскладки выше;

Использованные функции:

1. setSpacing – установка расстояния между элементами в раскладке;
2. setContentsMargins – установка полей раскладки ;

Метод load\_student\_data – код нужен для выгрузки базы данных в таблицу, код ниже на рисунке 19.



Рисунок 19 - Метод load\_student\_data

Пояснение работы и функций:

|  |
| --- |
| def load\_student\_data(self):   students\_data = get\_students()- взять информацию студентов из функции get\_students – далее подробнее;  if students\_data and students\_data != "no data": - если данные поступили или нет;  if isinstance(students\_data, str): - если данные относятся к строкам;  try: - то;  students\_data = eval(students\_data)- выполняет строку, переданную ему в качестве аргумента;  except:- иначе;  students\_data = [] – данные пусты  if students\_data and isinstance(students\_data, (list, tuple)): - если данные принадлежат контейнеру лист  self.table.setRowCount(len(students\_data))- длина строк равноценна длине количества студентов, setRowCount- установить счетчик строк;  self.table.setColumnCount(10)- установка количества колонок таблицы;  headers = ["ID", "Name", "Surname", "Gend", "Age", "Email", "Grade", "Phone", "Group", "Form"] – заголовки колонок;  self.table.setHorizontalHeaderLabels(headers) – уточнение,что заголовки горизонтальны;   for row\_idx, student in enumerate(students\_data): - для строки информации ;  for col\_idx, value in enumerate(student):- для столбца;  item = QTableWidgetItem(str(value))- наполнение ячейки информацией;  self.table.setItem(row\_idx, col\_idx, item) – наполнение;  self.table.resizeColumnsToContents()- установить размеры ячеек исходя из размера информации;  self.table.setAlternatingRowColors(True)- разрешить альтернативный цвет для строк;  self.table.setStyleSheet("alternate-background-color: #a0e1e8; background-color: #65bfb9;")- цвета таблицы; |

Подробнее о функции get\_students, находящейся в файле sql.py:

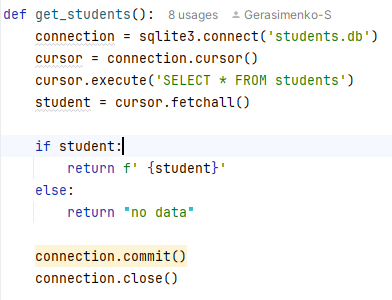
****

Рисунок 20- функция get\_students

|  |
| --- |
| def get\_students():  connection = sqlite3.connect('students.db')- подключение к БД;  cursor = connection.cursor() – использовать переменную как управление;  cursor.execute('SELECT \* FROM students') – выделить все из таблицы;  student = cursor.fetchall()- в буфере вся информация;   if student: - если не пусто;  return f' {student}'- возвращай информацию в переменной;  else:  return "no data"- иначе вывод сообщения;   connection.commit()- подтвердить;  connection.close()- закрыть соеденение; |

Таким образом, мы описали принцип появления информации в таблице, перейдем к кнопкам.

Метод create\_buttons подключает кнопкам свои функции, таким образом на рисунке 20:

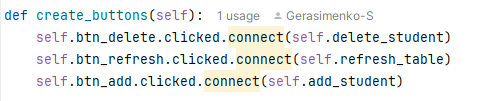


Рисунок 21- Метод create\_buttons

Соответственно опишем каждую функцию кнопок, первой опишем обновление таблиц - функцию refresh\_table.

|  |
| --- |
| def refresh\_table(self):  students\_data = get\_students()  self.table.setRowCount(0)   if students\_data and students\_data != "no data":  if isinstance(students\_data, str):  try:  students\_data = eval(students\_data)  except:  students\_data = []   self.table.setRowCount(len(students\_data))  for row\_idx, student in enumerate(students\_data):  for col\_idx, value in enumerate(student):  item = QTableWidgetItem(str(value))  self.table.setItem(row\_idx, col\_idx, item) |

Эта функция идентична функции load\_student\_data, работает одинаково, за исключением, что не создает таблицу заново, а лишь по базе данных перезаполняет содержимое ячеек.

Следующая функция – delete\_student, пояснение функций библиотек ведем параллельно:

|  |
| --- |
| def delete\_student(self):  selected = self.table.selectedItems()- выбранная ячейка;  if selected: - если ячейка чему-то равна;  row = selected[0].row() – выбранный ряд;  student\_id = self.table.item(row, 0).text()-ряд равен ID   reply = QMessageBox.question(  self, 'Подтверждение',  f"Удалить студента ID {student\_id}?",  QMessageBox.Yes | QMessageBox.No  ) – сообщение предупреждение (рисунок 21,1) , правда ли пользователь хочет удалить студента с этим ID;    Рисунок 21.1 – Обьект типа QMessageBox  if reply == QMessageBox.Yes: - если ответ да;  self.table.removeRow(row)- убрать ряд из таблицы;  delete\_student\_sql(student\_id)- далее подробно;  Из файла «sql.py» мы используем функцию delete\_student\_sql, удаление из базы данных. Опишем ее тоже:  def delete\_student\_sql(student\_id):  connection = sqlite3.connect('students.db')  cursor = connection.cursor()   cursor.execute("DELETE FROM students WHERE id = ?", (student\_id,)) – удаляем по ID;   connection.commit()  return True |

Последняя кнопка – кнопка добавления, работает от двух функций, разберем их ниже:

|  |
| --- |
| def add\_student(self):  try:  dialog = QInputDialog(self)- создаем диалоговое окно (рис.22)    Рисунок 22 – Диалоговое окна типа QInputDialog  dialog.setWindowTitle("Добавить студента")  dialog.setLabelText(  "Введите данные через запятую:\nИмя,Фамилия,Пол(M/Ж),Возраст,Email,Курс,Телефон,Группа,Форма(Б/К/Ц/ВБ)") – правила ввода;   if dialog.exec\_():- если диалоговое окно продолжает существовать;  input\_data = dialog.textValue()- собираем данные из ввода;  data = [x.strip() for x in input\_data.split(',')] – делим данные по запятой;   if len(data) != 9:- если длина оказывается меньше 9;  raise ValueError("Нужно ввести ровно 9 параметров")- сообщаем об ошибке;    if data[2] not in ['М', 'Ж']:- если пол не соответствует правилам;   raise ValueError("Пол должен быть M или Ж")- сообщаем об этом;   if not data[3].isdigit() or not data[5].isdigit(): - если числовые данными такими не являются;  raise ValueError("Возраст и курс должны быть числами")- сообщаем;   if self.save\_to\_database(data):- сохраняем в базу;  self.refresh\_table()-обновляем таблицу;  QMessageBox.information(self, "Успех", "Студент добавлен")- информируем об успехе;   except ValueError as ve:  QMessageBox.warning(self, "Ошибка ввода", str(ve))- выводят сообщения на экран об ошибке ввода;  except Exception as e:  QMessageBox.critical(self, "Ошибка", f"Ошибка: {str(e)}")- о сторонней ошибке; |

Теперь нужно сохранить нового студента в базу данных, следующим образом:

|  |
| --- |
| def save\_to\_database(self, student\_data):  try:  connection = sqlite3.connect('students.db')  cursor = connection.cursor()   cursor.execute("""  INSERT INTO students (name, surname, gend, age, mail, grade, phone\_number, grupa, form)  VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)  """, student\_data)-вставляем студента в базу;   connection.commit()  connection.close()  return True  except Exception as e:  QMessageBox.critical(self, "Ошибка БД", f"Ошибка при сохранении: {str(e)}")-обработка ошибок;  return False |

Таким образом мы описали принцип работы всей основной части интерфейса, кроме определения нынешнего пользователя и уровней доступа, работает это так:

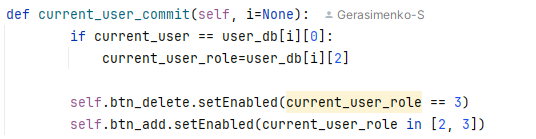


Рисунок 23 - Функция определения нынешнего пользователя

# **Разработка модульных тестов и проведение тестирования приложения;**

# **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 Excelи Word. Базой данных в проекте, выступил vullist.xlsx- файлExcel, содержащий данные об уязвимостях различного ПО, предоставленного ФСТЭК-ом.

В приложение были добавлены удобные функции, такие, как: поиск угроз, просмотрколичества 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. ОсновыTkinter: [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**

# **Приложение Б Руководство пользователя**