**КУРСОВОЙ ПРОЕКТ**

**Тема: Разработка программного модуля «Научные труды сотрудников»**

**Специальность «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студентка группы 32ИС-21** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Арутюнян С.К.** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **В.Ю. Назаров** |

**Москва 2023**

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**по дисциплине: МДК.01.01 Разработка программных модулей**

**Специальность «Информационные системы и программирование»**

**Студентка группы 32ИС-21 Арутюнян Софья**

**ТЕМА: Разработка программного модуля «Научные труды сотрудников»**

**Дата выдачи задания «**\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

**Срок сдачи работы «**\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Москва 2023**

Содержание

[Введение 4](#_Toc153724211)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc153724212)

[1.1 Информационное обеспечение задачи 5](#_Toc153724213)

[1.2 Обзор и анализ существующих программных решений 5](#_Toc153724214)

[1.3 Постановка задачи 6](#_Toc153724215)

[2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММЫ 7](#_Toc153724216)

[2.1 Схема работы 7](#_Toc153724217)

[2.2 Диаграмма прецедентов 8](#_Toc153724218)

[2.3 План тестирования 9](#_Toc153724219)

[2.4 Выбор средств разработки приложения 11](#_Toc153724220)

[2.5 Описание базы данных 11](#_Toc153724221)

[2.5.1 Структура базы данных 12](#_Toc153724222)

[2.6 Реализация интерфейса приложения 13](#_Toc153724223)

[2.7 Реализация программного кода 17](#_Toc153724224)

[2.7.1 Используемые библиотеки 17](#_Toc153724225)

[2.8 Отладка и тестирование 22](#_Toc153724226)

[ЗАКЛЮЧЕНИЕ 29](#_Toc153724227)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc153724228)

[ПРИЛОЖЕНИЕ 1 31](#_Toc153724229)

[ПРИЛОЖЕНИЕ 2 32](#_Toc153724230)

[ПРИЛОЖЕНИЕ 3 33](#_Toc153724231)

[ПРИЛОЖЕНИЕ 4 34](#_Toc153724232)

[ПРИЛОЖЕНИЕ 5 38](#_Toc153724233)

[ПРИЛОЖЕНИЕ 6 39](#_Toc153724234)

[ПРИЛОЖЕНИЕ 7 43](#_Toc153724235)

[ПРИЛОЖЕНИЕ 8 44](#_Toc153724236)

[ПРИЛОЖЕНИЕ 9 46](#_Toc153724237)

[ПРИЛОЖЕНИЕ 10 48](#_Toc153724238)

**Введение**

Тема "Научные труды сотрудников" остается актуальной в современном обществе. Несмотря на ценность журналов и книг, они уступают интернету по востребованности. Современная научная деятельность тесно связана с использованием информационных технологий. В связи с объемом и сложностью научной информации существует необходимость в системе, способной эффективно управлять данными и обеспечивать их удобный доступ, анализ и распространение.

Целью разработки программного модуля "Научные труды сотрудников" становится повышение эффективности научной работы. Этот модуль предоставляет ученым возможность представлять результаты исследований, а пользователям - удобный доступ к материалам и просмотр трудов интересующих ученых.

Первый раздел "Анализ предметной области" представляет результаты мониторинга существующих программных решений и постановку задач курсового проекта. Второй раздел "Проектирование и разработка программы" описывает этапы проектирования, кодирования и тестирование программы.

Приложение "Russscience" направлено на создание удобного и функционального инструмента для публикации научных исследовательских трудов сотрудников и обеспечения легкого доступа к ним для всех заинтересованных пользователей.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

* 1. **Информационное обеспечение задачи**

Рассмотрены существующие методы и инструменты, которые используются для публикации научных трудов и просмотра статей ученых.

Платформы для размещения научных публикаций:

* Журналы и издания
* Онлайн-платформы

Возможности загрузки и публикации статей, а также существующие базы данных научных работ рассмотрим ниже.

В современном информационном веке, насыщенном инновациями и интернетом, большинство людей используют онлайн-ресурсы для поиска информации и чтения статей. В связи с этим важно предоставить больше возможностей для чтения электронных статей. Сайты или приложения, регулярно обновляющиеся, будут более успешными в этой среде.

* 1. **Обзор и анализ существующих программных решений**

Инструменты, использующиеся в реализации публикаций научных трудов сотрудников:

* Система управления научными журналами, например, Open Journal Systems. Предоставляет функциональность для организации процесса рецензирования, публикации и архивирования научных статей. OJS является открытым и бесплатным инструментом, который

широко используется в научных журналах по всему миру.

* Онлайн-платформы для публикации научных работ, например, Sci.Guide. Предоставляют возможность ученым загружать и делиться своими научными работами, а также взаимодействовать с другими учеными. Некоторые из этих платформ также предоставляют функциональность для организации процесса рецензирования.
* Базы данных научных статей, такие как Scopus, и Web of Science. Они предоставляют доступ к большому количеству научных статей и позволяют искать и фильтровать статьи по различным критериям.

Обзор и анализ современных программных решений показывает, что существует широкий спектр инструментов и платформ, обеспечивающие доступ к научным материалам, а также упрощают процесс публикации и взаимодействия. Это открывает новые возможности для исследователей, студентов и других заинтересованных лиц в получении и обмене научной информацией.

* 1. **Постановка задачи**

Программа будет использоваться двумя группами лиц: администратор и пользователь.

Запущенная с правами администратора, она имеет такие возможности:

* Просмотр существующих данных, а точнее список научных сотрудников, список наименований статей каждого сотрудника и сами статьи.
* Добавление новых научных сотрудников.
* Добавление новых наименований статей, а также их содержание.
* Удаление научных сотрудников.
* Удаление наименований статей, а также их содержание.

Удалить ранее добавленные данные о научных сотрудниках и их трудах можно только в случаи, если учёный был лишен заслуженного признания за свои научные открытия.

При добавлении новой информации, автоматически обновляется база данных.

Запущенная с правами пользователя, она имеет такие возможности:

* Просмотр существующих научных сотрудников в виде список.
* Просмотр наименований статей сотрудников в виде списка.
* Просмотр содержания статей каждого сотрудника.

**2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММЫ**

* 1. **Схема работы**

В блок-схеме описан принцип работы программы «Russscience» с точки зрения двух ролей – администратора и пользователя. Процесс просмотра и чтения статей определённого учёного состоит из трех этапов:

1) этап просмотра всех научных сотрудников;

2) этап просмотра списка публикаций определённого учёного;

3)этап прочтения выбранной статьи

Пользователь – наиболее ограниченный по функционалу класс программы. Он может переходить по приложению, просматривать список учёных, выбирать определённых людей, определенные статьи и смотреть их содержание.

Администратор приложения обладает правами добавление и удаления в ручном режиме новых научных сотрудников и их публикаций (Рис.1).

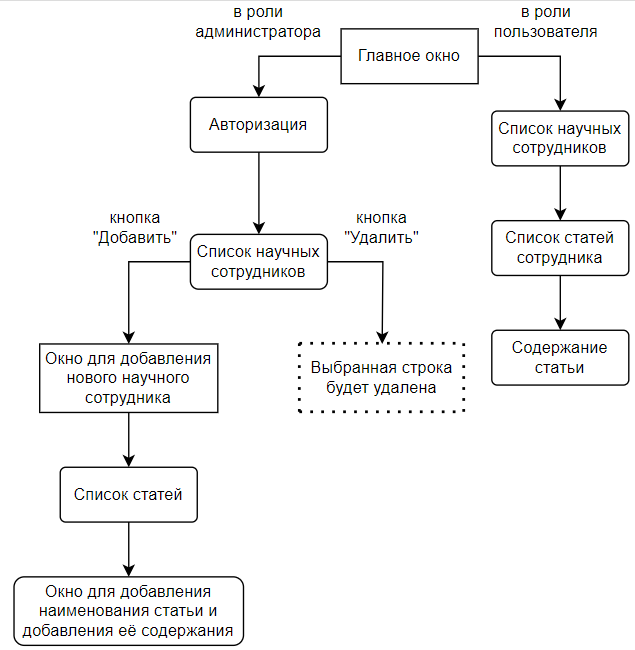


Рис. 1 Схема работы

**2.2 Диаграмма прецедентов**

Для разработки приложения необходимо определить функциональный ряд, который должен выполняться на протяжении активной деятельности. В приложении «Russscience» существует две роли, рассмотрим каждую из них (Рис.2).

Пользователь:

* Просмотр списка научных сотрудников
* Просмотр списка статей определённого сотрудника
* Просмотр содержимого определённой статьи

Администратор:

* Авторизация
* Просмотр списка научных сотрудников
* Добавление нового научного сотрудника
* Удаление научного сотрудника из списка
* Просмотр списка статей определённого сотрудника
* Добавление статьи
* Определение наименования статьи
* Добавление текста статьи
* Просмотр содержимого определённой статьи

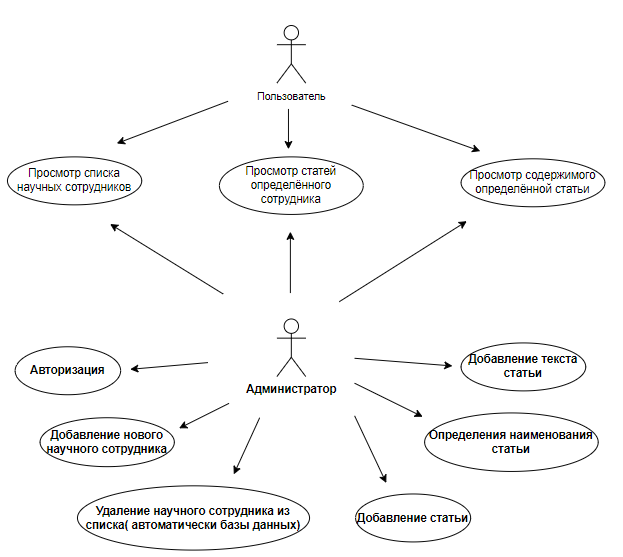


Рис. 2 Диаграмма прецедентов

**2.3 План тестирования**

После окончания создания проекта идет этап тестирования. Готовое программное приложение проверяется вручную, либо с помощью автоматизированных программ.

Проверка игрового приложения «Russscience» будет осуществляться методом тестирования «Чёрный ящик» т.е. ручным способом при помощи многократного использования всех функций. Чтобы протестировать приложение будут использоваться основные функции: авторизация, добавление нового научного сотрудника, новой статьи и текста.

Версия тест плана: 0,01

Дата составления: 12.12.2023

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Наименование функциональности** | **Наименования поля** | **Тестовый набор** | **Ожидаемый результат** | **Результат тестирования** |
| **1** | Авторизация | Логин и пароль | Пустые поля, только логин или пароль, неверный логин и пароль. | Сообщение об ошибке. | Сообщение об ошибке: «Неверный логин или пароль» и «Введите корректный логин и пароль». |
| Логин и пароль | Введены верные логин и пароль | Вход в систему. | Вход в систему. |
| **2** | Добавление научного сотрудника | Сотрудник | Пустые поля или что-то недостающее из наименования поля. | Сообщение – предупреждение. | Сообщение -предупреждение: ««Введите корректные данные». |
| Сотрудник | Все данные введены корректно. | Добавление новой записи о научном сотруднике. | Добавление новой записи о научном сотруднике. |
| **3** | Добавление фотографии научного сотрудника (дополнительный пункт для добавления научного сотрудника) | Название файла с фотографией | Пустые поля или неверное наименование файла (он должен быть в одной папке с проектом). | Отсутствие фотографии на экране. | Отсутствие фотографии на экране. |
| Название файла с фотографией | Наименование файла введено верно. | Видимость фотографии на экране. | Видимость фотографии на экране. |
| **4** | Добавление названия статьи | Название статьи | Пустые поля, некорректные данные. | Сообщение – предупреждение. | Добавление новой статьи, видимость её на экране. Если поле пустое, добавляется «пустота», но под соответственным номером. |
| Название статьи | Все данные введены корректно. | Добавление новой статьи, видимость её на экране. | Добавление новой статьи, видимость её на экране. |
| **5** | Заполнение статьи содержанием | Текст статьи | Пустые поля, некорректные данные. | Сообщение – предупреждение. | Добавление содержания, видимость его на экране. Если поле пустое, добавляется «пустота», но под соответственным номером. |
| Текст статьи | Введенный корректно текст. | Добавление содержания статьи, видимость его при открытии определенной статьи. | Добавление содержания статьи, видимость его при открытии определенной статьи. |
| **6** | Удаление научного сотрудника | Удалить | Нажатие на кнопку «Удалить». | Выбранный учёный будет удалён. | Выбранный учёный будет удалён. |
| **7** | Удаление статьи научного сотрудника | Удалить | Нажатие на кнопку «Удалить». | Выбранная статья будет удалена. | Выбранная статья будет удалена. |

## 2.4 Выбор средств разработки приложения

Средства разработки приложения:

* Язык программирования (Python)
* Платформа написания интерфейса (Qt Designer)
* Наличие необходимых библиотек (PyQt6)
* Поддержка различных операционных систем (Windows10…)

## 2.5 Описание базы данных

Для хранения и управления информацией о научных сотрудниках, статьями и данными для входа была выбрана СУБД SQlite Studio. База данных «date» включает в себя следующие таблицы:

* Таблица Admin

Хранит логин и пароль для авторизации администратора (Рис.3).



Рис.3

* Таблица Employees

Хранит имена научных сотрудников и наименования (Рис.4).



Рис.4

* Таблица Articles

Содержит названия статьей и их содержание (Рис.5).

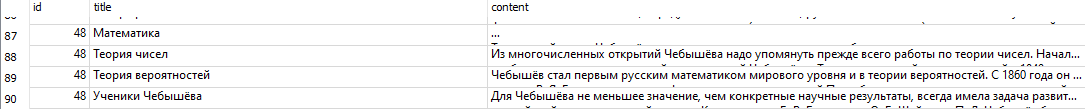


Рис.5

### 2.5.1 Структура базы данных

**admins:**

login (TEXT): Логин администратора.

password (TEXT): Пароль администратора.

**employees:**

id (INTEGER):

name (TEXT):

img (TEXT):

**articles:**

id (INTEGER): id научного сотрудника.

title (TEXT): Название статьи.

content (TEXT): Содержание статьи.

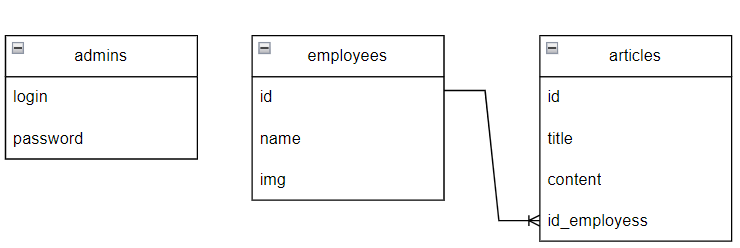


Рис. 6 Диаграмма сущностей

В системе имеется только один администратор, у которого есть свои учетные данные для входа.

Каждый научный сотрудник может иметь несколько статей. Между научным сотрудником и его статьями существует связь "один ко многим".

**2.6 Реализация интерфейса приложения**

Для реализации интерфейса программы «Russscience» использована среда Qt Designer. Это кроссплатформенная свободная среда для разработки графических интерфейсов для программ, использующих библиотеку Qt. Входит в состав Qt framework.

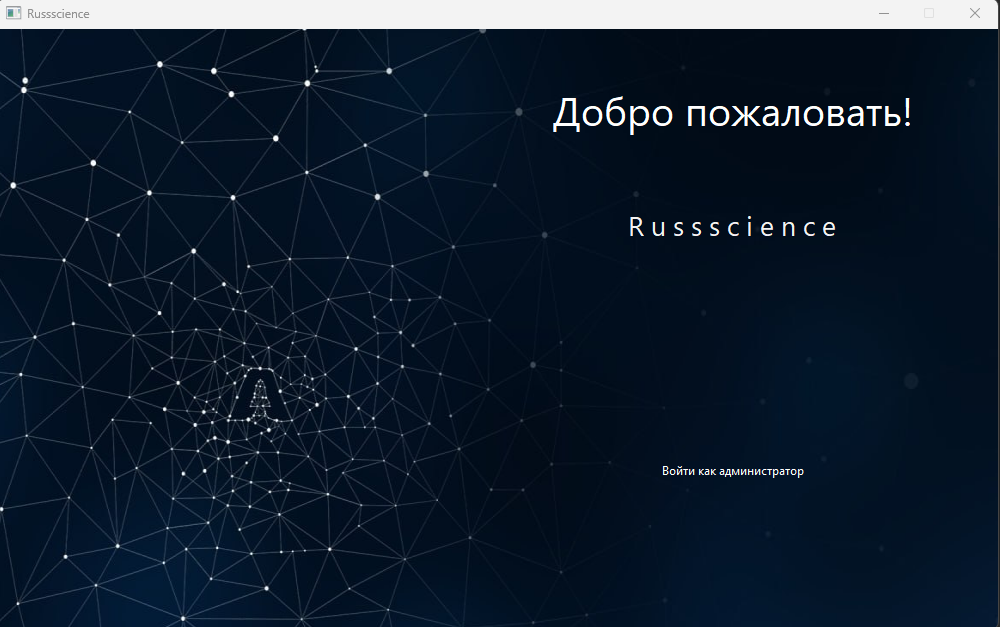


Рис. 3 - главное окно «Russscience»

В главном окне программы размещены две кнопки с различными функциями. Для пользователей с правами администратора доступна кнопка "Войти как администратор", при нажатии на которую открывается новое окно для авторизации. Для обычных пользователей авторизация и соответственно регистрации не предусмотрена, существует кнопка "Добро пожаловать!", которая позволяет просматривать публикации научных сотрудников без использования специального шифра (Рис.3).

После успешной входа, пользователь попадает на страницу со списком выдающихся умов России. Здесь он может выбрать интересующую его личность, просто нажав на фамилию (Рис. 4). В результате открывается новое окно с фотографией выбранного человека и списком его значимых достижений (Рис. 5). Также доступна возможность прочитать статью целиком (Рис. 6). При желании вернуться на предыдущий этап и выбрать что-то другое, пользователь может использовать кнопку "Обратно".

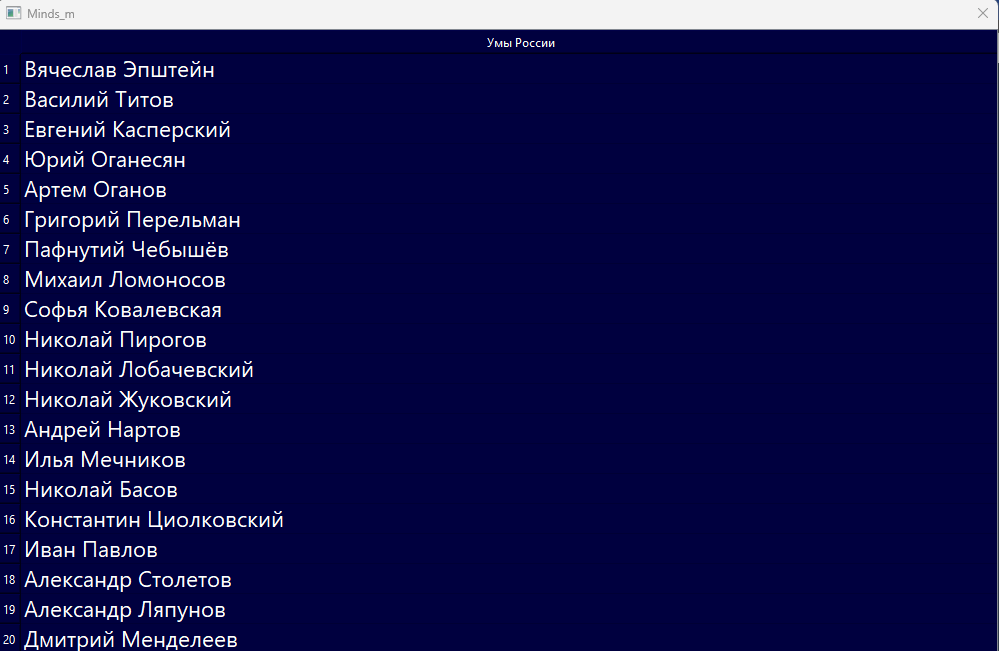


Рис. 4 – окно научных сотрудников «Russscience»

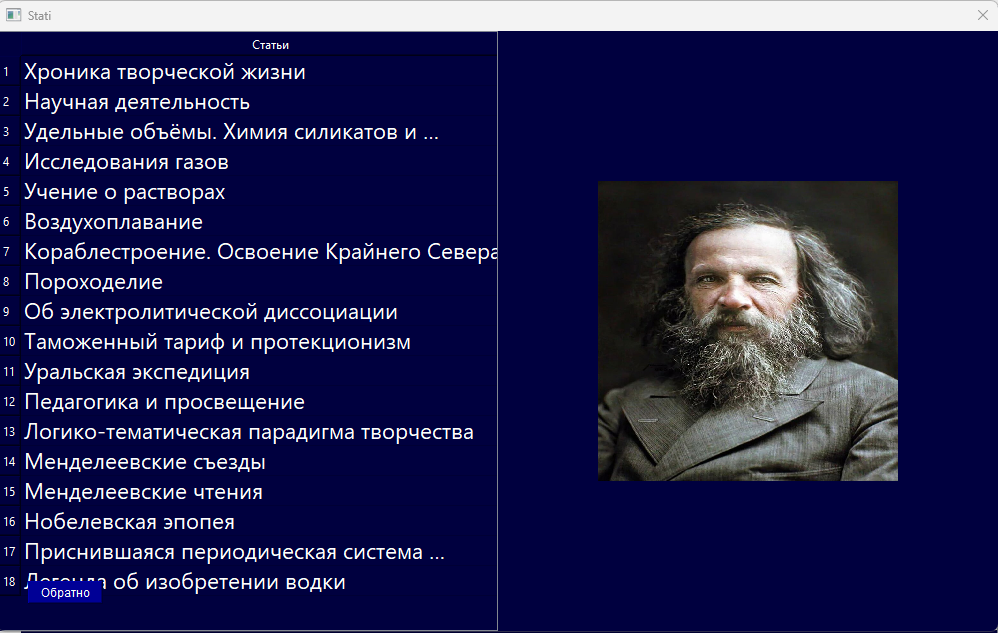


Рис. 5 – окно статей учёного «Russscience»

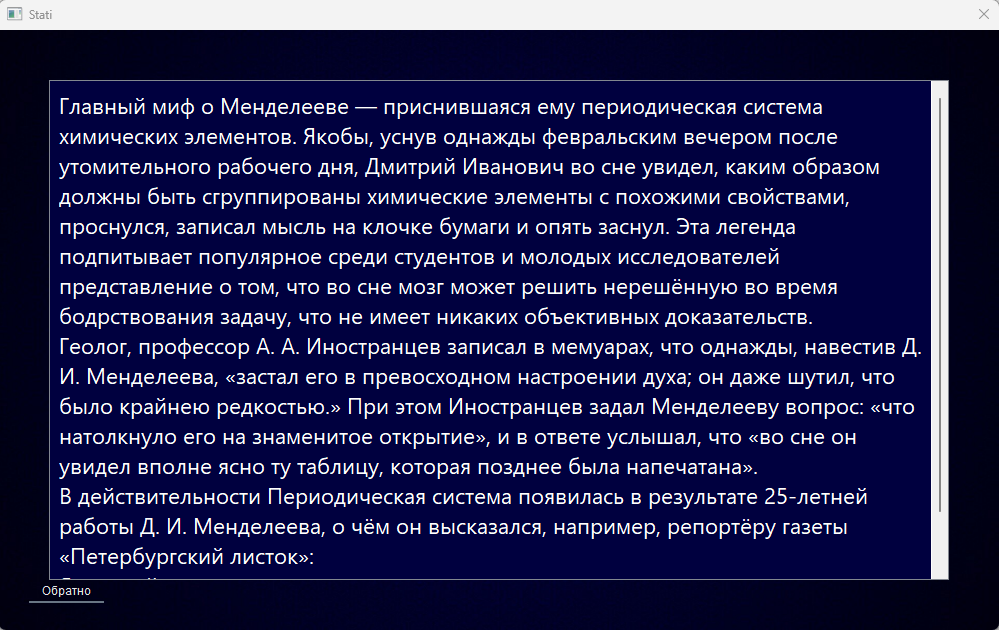


Рис. 6 – окно содержания статьи «Russscience»

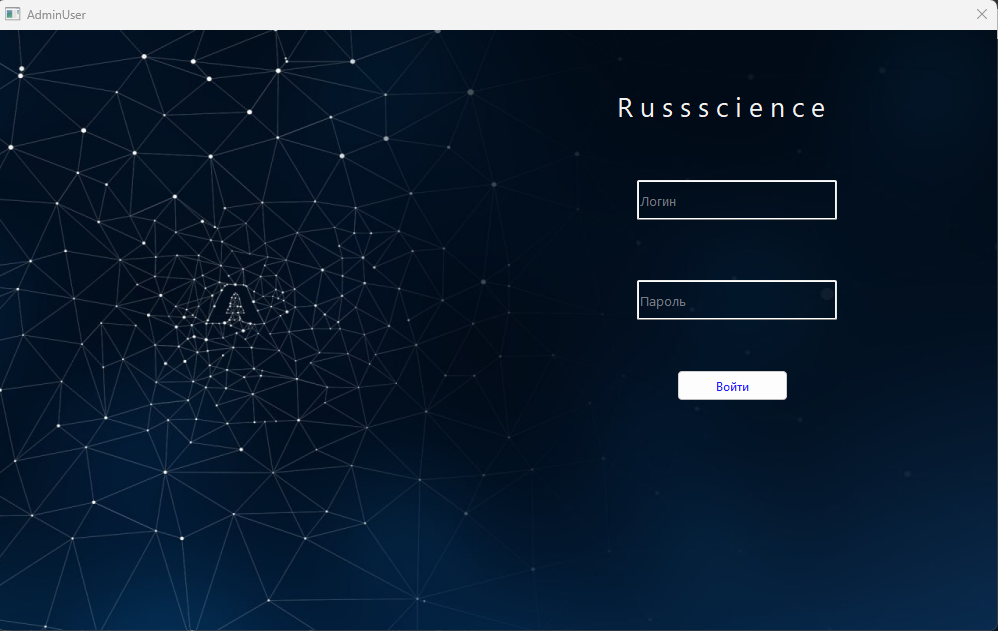


Рис. 7 – окно авторизации «Russscience»

При активации функции “Войти как администратор” открывается окно авторизации с уникальным логином и паролем, хранящимися в базе данных. Авторизация администратора предоставляет дополнительные возможности, недоступные обычному пользователю.

Администратор имеет возможность добавлять новых научных сотрудников, при нажатии кнопки "Добавить" появляется окно, где необходимо указать Имя и Фамилию ученого, а также название файла с фотографии. Чтобы фотография появилась на экране, она должна находиться в той же папке, что и проект. После завершения процесса, список учёных обновится.

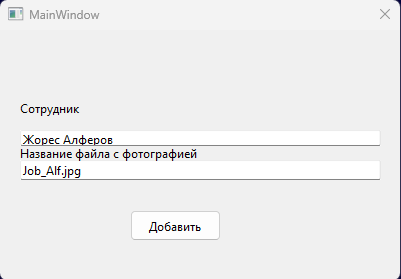


Рис. 8 Окно добавления учёных

Для добавления статьи в портфолио учёного необходимо выбрать одного из научных сотрудников и нажать кнопку "Добавить". После этого откроется окно, где можно указать название публикации и ввести ее содержание (Рис. 9).

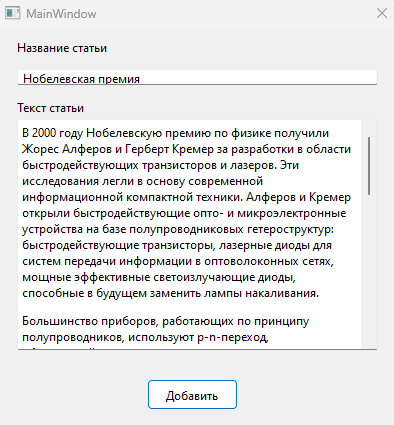


Рис. 9 Окно добавления публикации

**2.7 Реализация программного кода**

В данном разделе будет рассматриваться написание кода.

### 2.7.1 Используемые библиотеки

import sys

from PyQt6.QtWidgets import QTableWidgetItem, QAbstractItemView, QLabel, \

QVBoxLayout, QScrollArea, QWidget

from PyQt6 import uic

from PyQt6.QtGui import QPixmap

import sqlite3

from PyQt6.QtWidgets import \*

from PyQt6.QtGui import \*

from PyQt6.QtCore import \*

sys: Этот модуль предоставляет доступ к переменным, поддерживаемым интерпретатором Python, а также к функциям, взаимодействующим напрямую с интерпретатором. Он используется для управления окружением выполнения программ на этом языке.

PyQt6.QtWidgets: Этот модуль содержит классы для создания настольных приложений. В данном случае он для создания элементов графического интерфейса, таких как QTableWidgetItem, QAbstractItemView, QLabel, QVBoxLayout, QScrollArea и QWidget. Эти классы являются частью библиотеки PyQt .

PyQt6.uic: Этот модуль предоставляет инструменты для загрузки файлов пользовательского интерфейса, созданных с помощью Qt Designer, в программы на Python.

PyQt6.QtGui: Этот модуль содержит классы для работы с графическими элементами, такими как QPixmap. В данном контексте, он используется для отображения изображений в GUI.

sqlite3: Этот встроенный модуль Python, он используется для взаимодействия с базой данных SQLite, вероятно, для хранения и извлечения данных в приложении.

PyQt6.QtCore: Этот модуль содержит основные функции, не связанные с GUI. В данном контексте он используется для обработки основной функциональности, такой как сигналы и другие основные аспекты логики приложения.

В общем, эти библиотеки используются вместе для создания настольного приложения с графическим интерфейсом пользователя, подключения к базе данных (SQLite) и различных элементов GUI для отображения и взаимодействия с данными.

Для входа в систему в качестве обычного пользователя авторизация не требуется. Данный класс описывает главное окно приложения. Здесь загружается пользовательский интерфейс и устанавливаются обработчики событий для кнопок.(См. Приложение 1).

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка пользовательского интерфейса из файла 'interface.ui'

uic.loadUi('interface.ui', self)

self.setFixedSize(1000, 600)

#Соединение сигнала "clicked" кнопки pushButton\_2 с методом show\_admin\_window

self.pushButton\_2.clicked.connect(self.show\_admin\_window)

#Соединение сигнала "clicked" кнопки pushButton с методом show\_minds\_window

self.pushButton.clicked.connect(self.show\_minds\_window)

def show\_admin\_window(self):

#Определение метода для отображения окна администратора.

admin\_window.show()

Чтобы войти в аккаунт администратора, потребуется уникальный код, который установит связь с базой данных и извлечет из нее логин и пароль. Также важно уделить внимание интерфейсу и сопутствующим ему кнопкам. (См. Приложение 2).

# Получение введенного логина из виджета login и password

login = self.login.text()

password = self.password.text()

# Проверка на ввод цифр, символов или пустых значений

if not login or not password or not login.isalpha() or not password.isalnum():

QMessageBox.critical(self, 'Ошибка', 'Введите корректный логин и пароль')

return

# Подключение к базе данных

connection = sqlite3.connect('data.db')

# Создание объекта курсора для выполнения SQL-запросов

cursor = connection.cursor()

# Выполнение SQL-запроса для выбора всех данных из таблицы 'admins'

cursor.execute('SELECT \* FROM admins')

admins = cursor.fetchall()

В приложении существуют виджеты с большим размером текста, для удобства просмотра информации используется прокручиваемая область- ScrollArea (См. Приложение 3).

class ScrollLabel(QScrollArea):

#Конструктор для инициализации виджета прокручиваемой области

def \_\_init\_\_(self, \*args, \*\*kwargs):

QScrollArea.\_\_init\_\_(self, \*args, \*\*kwargs)

# возможность изменения размера виджета

self.setWidgetResizable(True)

# Устанавливается content в качестве виджета прокручиваемой области

content = QWidget(self)

self.setWidget(content)

# Устанавливается выравнивание текста

self.label.setAlignment(Qt.AlignmentFlag.AlignLeft | Qt.AlignmentFlag.AlignTop)

Администратор имеет возможность управлять списком научных сотрудников (дополнять и исключать), списком публикаций конкретного сотрудника (дополнять и исключать), обновленная информация сразу же становится видна на виджете и одновременно отражает изменения в базе данных. (См. Приложение 4)

# Устанавливаем соединение с базой данных 'data.db'

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

## Выполняем SQL-запрос для удаления записи с указанным id из таблицы 'employees'

cursor.execute(f'DELETE FROM employees WHERE id = {self.id\_del}')

## Фиксируем изменения в базе данных

connection.commit()

cursor.close()

connection.close()

self.table\_view()

Администратор при желании может добавить нового учёного нажав кнопку «Добавить» (см. Приложение 4). Появится окно, где нужно указать имя учёного и наименование файла, который хранит его фотографию (См. Приложение 5).

def addemp(self):

# Получаются значения из полей ввода и сохраняются в переменные

emp\_name = self.lineEdit.text()

emp\_img = self.lineEdit\_1.text()

# Соединение с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

# Выполняется SQL-запрос для получения всех id из таблицы

cursor.execute(f'SELECT id FROM employees')

employees = cursor.fetchall()

# Вычисляется новый id для нового сотрудника путем увеличения последнего id в таблице на 1

emp\_id = employees[-1][0] + 1

# Выполняется SQL-запрос для вставки новой записи в таблицу с полученными значениями

cursor.execute(f'INSERT INTO employees VALUES ({emp\_id}, "{emp\_name}", "{emp\_img}")')

connection.commit()

cursor.close()

connection.close()

empl\_window.close()

minds\_window.table\_view()

adminminds\_window.table\_view()

Код для отображения окна с публикациями определенного учёного с привилегиями администратора т.е. возможностью добавления и удаления статей (См. Приложение 6).

# Метод для удаления статьи из базы данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute(f'DELETE FROM articles WHERE title = "{self.id\_del\_st}"')

connection.commit()

cursor.close()

connection.close()

# Перезагрузка отображения таблицы после удаления

self.table\_view()

Код для добавления новой статьи учёного. Окно с полями для ввода наименования статьи и её содержания (См. Приложение 7).

def addemp(self):

# Получаем данные из виджетов

title = self.lineEdit.text()

content = self.textEdit.toPlainText()

# Устанавливаем соединение с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute(f'INSERT INTO articles VALUES ({id\_admin}, "{title}", "{content}")')

# Фиксируем изменения и закрываем соединение

connection.commit()

cursor.close()

connection.close()

article\_window.close()

# Обновляем таблицы в окнах adminstati\_window и stati\_window

adminstati\_window.table\_view()

stati\_window.table\_view()

Пользовательское окно, отображающее список научных сотрудников. При клике на одного из них произойдет соединение со следующем окном публикаций учёного (См. Приложение 8)

# Метод для отображения таблицы сотрудников

def table\_view(self):

# Установка соединения с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute('SELECT \* FROM employees')

employees = cursor.fetchall()

# Установка количества строк и столбцов в таблице

self.tableView.setRowCount(len(employees))

self.tableView.setColumnCount(1)

# Запрет редактирования ячеек

self.tableView.setEditTriggers(QAbstractItemView.EditTrigger(0))

self.tableView.setHorizontalHeaderItem(0, QTableWidgetItem('Умы России'))

Пользовательское окно, отображающее список публикаций выбранного учёного, при клике на одну из них, на экране отобразится её содержание в прокручиваемой области (См. Приложение 9).

* 1. **Отладка и тестирование**

Требования к приложению:

1. Процесс авторизации, который доступен только администратору с использованием специального логина и пароля, хранящихся в базе данных.
2. Администратор должен иметь возможность добавлять новых научных сотрудников.
3. Администратор должен иметь возможность добавлять новые статьи.
4. Администратор должен иметь возможность создавать текста.
5. Администратор должен иметь возможность удалять научных сотрудников.
6. Администратор должен иметь возможность удалять статьи вместе с их содержанием.
7. Видоизменённые списки при вмешательстве администратора.

Ограничения:

1. При заполнении полей логина и пароля для ввода некорректными данными (цифры, символы, неправильные данные) должно обрабатываться исключение.
2. Поля и кнопки для добавления и удаления новой информации должны быть активны.
3. Изменённая информация должна влечь за собой обновлённую базу данных.

Тестирование авторизации администратора.

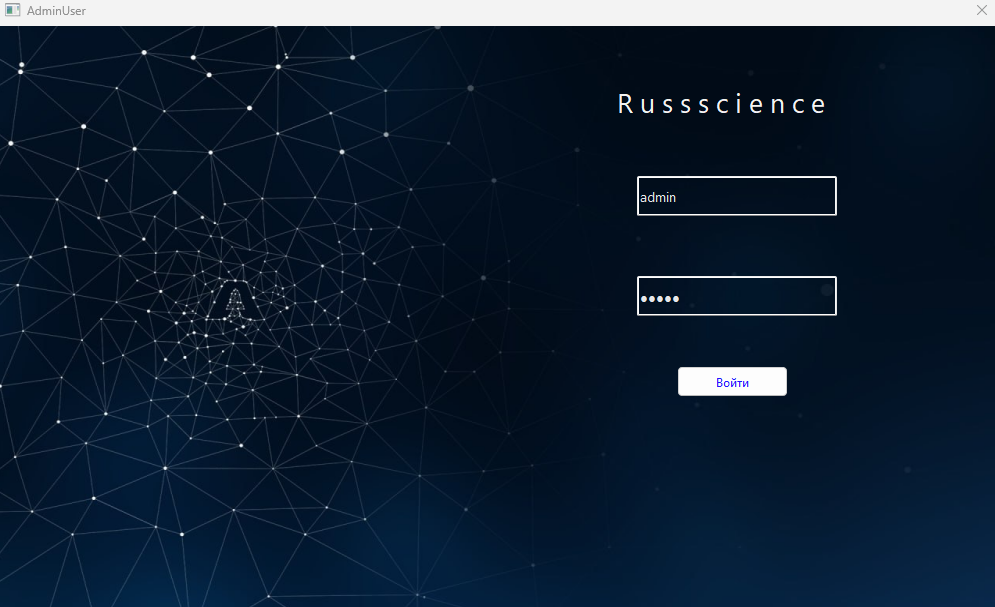
****

Рис. 10 – окно авторизации

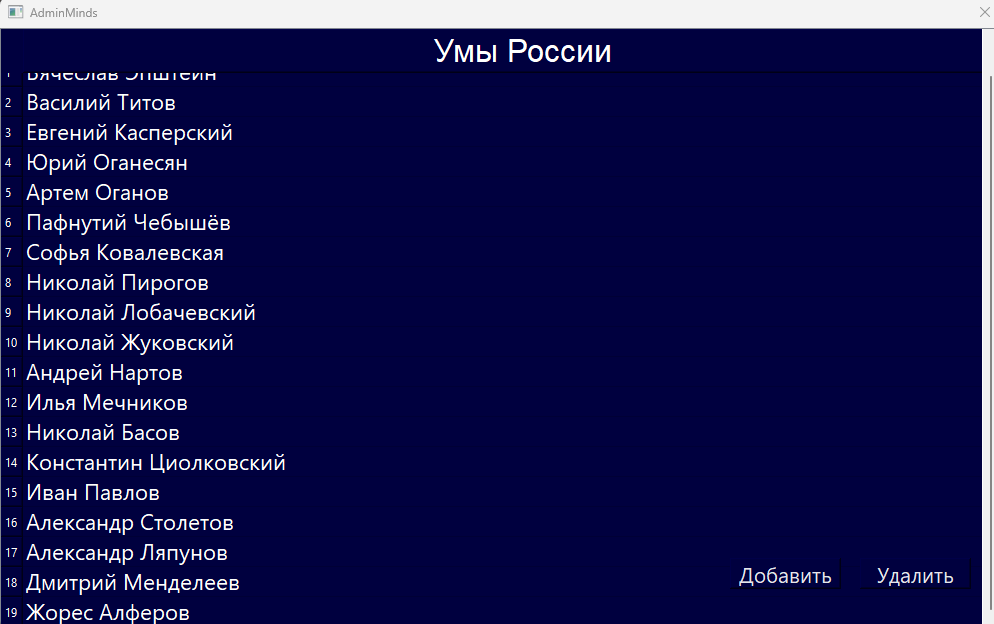


Рис. 11 – окно администратора список научных сотрудников

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя переменной** | **Назначение переменной** | **Тип переменной** | **Диапазон типа** |
| Login | Логин | Text | Максимальная длина строки ограничена объёмом память |
| Password | Пароль | Text | Максимальная длина строки ограничена объёмом память |

Таб.1 Спецификация переменных

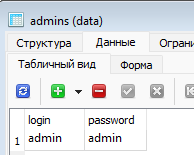


Рис. 12 – Таблица с логином и паролем в базе данных

|  |  |  |
| --- | --- | --- |
| **№ Теста** | **Login** | **Password** |
| **1.** | admin | admin |
| **2.** | логин | Пароль |
| **3.** | !\_»% | !\_»% |
| **4.** | 001 | 001 |
| **5.** | Пустое значение | Пустое значение |

Таб. 2 Тестовый сценарий

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ввод данных** | **Ожидаемый результат** | **Реальный результат** |
| **1.** |  | Ошибка, сообщение об ошибке. |  |
| **2.** |  | Ошибка, сообщение об ошибке. |  |
| **3.** |  | Ошибка, сообщение об ошибке. |  |
| **4.** |  | Ошибка, сообщение об ошибке. |  |
| **5.** |  | Ошибка, сообщение об ошибке. |  |

Таб. 3 Тест авторизации

|  |  |  |
| --- | --- | --- |
| **№ Теста** | **Описание ошибки** | **Рекомендации** |
| **2.** | В ходе проведения второго теста было обнаружено, что при введении неверного логина и пароля появляется сообщение об ошибке. Для дальнейшего перехода по окнам, процесс остановлен. | Рекомендаций нет, программа сработала так, как и было задумано, и успешно выявила ошибку. |
| **3.** | В ходе проведения третьего теста было обнаружено, что при введении символьных значений появляется сообщение об ошибке. Для дальнейшего перехода по окнам, процесс остановлен. | Рекомендаций нет, программа сработала так, как и было задумано, и успешно выявила ошибку. |
| **4.** | В ходе проведения четвёртого теста было обнаружено, что при введении числовых значений появляется сообщение об ошибке. Для дальнейшего перехода по окнам, процесс остановлен. | Рекомендаций нет, программа сработала так, как и было задумано, и успешно выявила ошибку. |
| **5.** | В ходе проведения пятого теста было обнаружено, если оставить поля с пустыми значениями, появляется сообщение об ошибке. Для дальнейшего перехода по окнам, процесс остановлен. | Рекомендаций нет, программа сработала так, как и было задумано, и успешно выявила ошибку. |

Таб. 4 Отчет об ошибках

|  |  |  |
| --- | --- | --- |
| № | Тест | Отметка о прохождении |
| 1. | admin, admin - корректная функция и результат соответствующий. | + |

Таб. 5 Позитивные тесты

|  |  |  |
| --- | --- | --- |
| № | Тест | Отметка о прохождении |
| 2. | Логин, Пароль - некорректный ввод, результат соответствующий требованиям. | - |
| 3. | !\_»% , !\_»%- некорректный ввод, результат соответствующий требованиям. | - |
| 4. | 001, 001 - некорректный ввод, результат соответствующий требованиям. | - |
| 5. | Пустые значения- некорректный ввод, результат соответствующий требованиям. | - |

Таб. 6 Негативные тесты

Тестирование добавление нового научного сотрудника.

|  |  |  |
| --- | --- | --- |
| **№ Теста** | **Сотрудник** | **Название файла с фотографией** |
| **1.** | Жорес Алферов | Job\_Alf.jpg |
| **2.** | 1111 | Job\_Alf. |
| **3.** | Пустое значение | Пустой значение |

Таб. 7 Тестовый сценарий

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ввод данных** | **Ожидаемый результат** | **Реальный результат** |
| **1.** |  | Добавление новой записи о научном сотруднике. |  |
| **2.** |  | Сообщение-предупреждение о некорректно введенных данных. Добавление новой записи о научном сотруднике. |  |
| **3.** |  | Сообщение об ошибке. |  |

Таб.8 Тест на добавление нового научного сотрудника

|  |  |  |
| --- | --- | --- |
| **№ Теста** | **Описание ошибки** | **Рекомендации** |
| **2.** | В ходе проведения второго теста было обнаружено, что при введении некорректных данных сообщение об ошибке не выводится. Добавление осуществляется. | Рекомендации: установить ограничения при введении данных и создать вывод сообщения о некорректных данных. |
| **3.** | В ходе проведения второго теста было обнаружено, что при введении некорректных данных сообщение об ошибке не выводится. Добавление осуществляется. | Рекомендации: установить ограничения при введении данных и создать вывод сообщения о некорректных данных. |

Таб. 9 Отчет об ошибках

|  |  |  |
| --- | --- | --- |
| № | Тест | Отметка о прохождении |
| 1. | Жорес Алферов, Job\_Alf.jpg - корректная функция и результат соответствующий. | + |

Таб. 10 Позитивные тесты

|  |  |  |
| --- | --- | --- |
| № | Тест | Отметка о прохождении |
| 2. | 1111, Job\_Alf. - некорректный ввод, результат несоответствующий требованиям. | - |
| 3. | Пустые данные- некорректный ввод, результат несоответствующий требованиям. | - |

Таб. 11 Негативные тесты

*Вывод:*

Код работает корректно и проходит базовые тесты, на некорректно вводимые данные реагирует в соответствии с требованиями. Рекомендуется установить ограничения при введении данных и создать вывод сообщения о некорректных данных.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта на тему: «Разработка программного модуля «Научные труды сотрудников» было сделано следующее:

* использованы методические указания для курсового проектирования по дисциплине «Разработка программных модулей»;
* проведено предварительное исследование существующих платформ для публикации научных статей и сред для разработки;
* проанализированы и определены задачи к программному продукту в виде приложения для просмотра научных трудов сотрудников, выбрана подходящая платформа, язык программирования, среда разработки;
* разработаны алгоритмы для написания кода программного продукта;
* спроектирован и реализован удобный пользовательский интерфейс;
* протестированы основные функции игрового приложения.

Цель курсового проекта была достигнута. Создано графическое приложение «Russscience» для публикации научных и исследовательских трудов и обеспечения широкого доступа к ним, построенное на библиотеке PyQt6.

Проект успешно реализует базовый функционал управления данными о научных сотрудниках, их статьях. Он включает в себя функции управления изменениями и просмотра сотрудников и их научных трудов.

Код проекта учитывает возможность возникновения ошибок, что повышает его устойчивость

Проект представляет собой хорошую основу, в будущем может быть модернизирован в более масштабное приложение.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Python 3 и PyQt 6. Разработка приложений/ Н. А. Прохоренок, В. А. Дронов. -СПб.: Б:ХВ-Петербурr, 2023. - 832 с.: ил. - (Профессиональное программирование)
2. Руководство в картинках к Open Journal Systems Version 3.0‘Изучаем OJS’/ Дмитрий Данилов – перевод на русский язык. // <https://language-pnu.ru/static/components/website_footer/ojs-3-manual.pdf>
3. Приложение с GUI на Python и PyQt6. Построение GUI в Qt Designer. // <https://www.youtube.com/watch?v=au01t_WOe_Q>
4. Наукометрические базы данных: полный и актуальный перечень. // <https://xn--80aegcaa6cbngm5a6c1ci.xn--p1ai/naukometricheskie-bazy-dannyh/>
5. Web of Science // <https://www.webofscience.com/wos/author/search>
6. Scopus. // <https://www.scopus.com/home.uri>
7. Sci.Guide // <http://www.prometeus.nsc.ru/sciguide/page03.ssi>
8. ГОСТ Р 1.4-2004 Стандартизация в Российской Федерации. Стандарты организаций. Общие положения
9. ГОСТ Р 1.5-2004 Стандартизация в Российской Федерации. Стандарты национальные Российской Федерации. Правила построения, изложения, оформления и обозначения.
10. ГОСТ 7.32-2001. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. (введен Постановлением Госстандарта России от 04.09.2001 N 367-ст) (ред. от 07.09.2005)

# ПРИЛОЖЕНИЕ 1

Листинг 1. Главное окно

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка пользовательского интерфейса из файла 'interface.ui'

uic.loadUi('interface.ui', self)

self.setFixedSize(1000, 600)

#Соединение сигнала "clicked" кнопки pushButton\_2 с методом show\_admin\_window

self.pushButton\_2.clicked.connect(self.show\_admin\_window)

#Соединение сигнала "clicked" кнопки pushButton с методом show\_minds\_window

self.pushButton.clicked.connect(self.show\_minds\_window)

def show\_admin\_window(self):

#Определение метода для отображения окна администратора.

admin\_window.show()

def show\_minds\_window(self):

#Вызов метода show() для отображения окна с информацией о разумах.

minds\_window.show()

#Вызов метода table\_view() для отображения таблицы в окне с информацией о разумах.

minds\_window.table\_view()

# ПРИЛОЖЕНИЕ 2

Листинг 2. Авторизация

class AdminWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка пользовательского интерфейса из файла 'Adminn.ui'

uic.loadUi('Adminn.ui', self)

self.setFixedSize(1000, 600)

# Соединение сигнала "clicked" кнопки vxod с методом auth

self.vxod.clicked.connect(self.auth)

# Определение метода для аутентификации администратора.

def auth(self):

# Получение введенного логина из виджета login и password

login = self.login.text()

password = self.password.text()

# Проверка на ввод цифр, символов или пустых значений

if not login or not password or not login.isalpha() or not password.isalnum():

QMessageBox.critical(self, 'Ошибка', 'Введите корректный логин и пароль')

return

# Подключение к базе данных

connection = sqlite3.connect('data.db')

# Создание объекта курсора для выполнения SQL-запросов

cursor = connection.cursor()

# Выполнение SQL-запроса для выбора всех данных из таблицы 'admins'

cursor.execute('SELECT \* FROM admins')

admins = cursor.fetchall()

# Проверка соответствия введенного логина и пароля администратора в базе данных

for logpas in admins:

if logpas[0] == login and logpas[1] == password:

return adminminds\_window.show()

QMessageBox.critical(self, 'Ошибка', 'Неверный логин или пароль')

# ПРИЛОЖЕНИЕ 3

Листинг 3. Прокручиваемая область

class ScrollLabel(QScrollArea):

#Конструктор для инициализации виджета прокручиваемой области

def \_\_init\_\_(self, \*args, \*\*kwargs):

QScrollArea.\_\_init\_\_(self, \*args, \*\*kwargs)

# возможность изменения размера виджета

self.setWidgetResizable(True)

# Устанавливается content в качестве виджета прокручиваемой области

content = QWidget(self)

self.setWidget(content)

# Создается вертикальное расположение виджетов

lay = QVBoxLayout(content)

self.label = QLabel(content)

self.label.setStyleSheet('color: white; font-size: 22px; background-color: rgb(0, 0, 63);border:none;')

content.setStyleSheet('color: white; background-color: rgb(0, 0, 63);border:none;')

# Устанавливается выравнивание текста

self.label.setAlignment(Qt.AlignmentFlag.AlignLeft | Qt.AlignmentFlag.AlignTop)

# Включается перенос слов в метке

self.label.setWordWrap(True)

# Метка добавляется в вертикальное расположение

lay.addWidget(self.label)

# Установки текста в label

def setText(self, text):

self.label.setText(text)

# ПРИЛОЖЕНИЕ 4

Листинг 4. Окно учёных администратора.

class AdminMindsWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

#Загрузка пользовательского интерфейса из файла 'AdminMinds.ui'

uic.loadUi('AdminMinds.ui', self)

self.setFixedSize(1000, 600)

#Соединение сигнала "doubleClicked" таблицы tableView

self.tableView.doubleClicked.connect(self.on\_click)

#Соединение сигнала "clicked" таблицы tableView

self.tableView.clicked.connect(self.on\_click\_del\_id)

#Соединение сигнала "clicked" с кнопками

self.pushButton\_2.clicked.connect(self.on\_click\_del)

self.pushButton.clicked.connect(self.on\_click\_add)

self.id\_del = -1

#Вызов метода table\_view для отображения данных в таблице при инициализации окна

self.table\_view()

def on\_click\_add(self):

empl\_window.show()

def on\_click\_del(self):

# Устанавливаем соединение с базой данных 'data.db'

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

## Выполняем SQL-запрос для удаления записи с указанным id из таблицы 'employees'

cursor.execute(f'DELETE FROM employees WHERE id = {self.id\_del}')

## Фиксируем изменения в базе данных

connection.commit()

cursor.close()

connection.close()

self.table\_view()

#Определение метода для отображения данных в таблице.

def table\_view(self):

# Устанавливаем соединение с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute('SELECT \* FROM employees')

#Получаем все записи из результата запроса.

employees = cursor.fetchall()

#Устанавливаем количество строк в таблице равным количеству записей в базе данных.

self.tableView.setRowCount(len(employees))

#Устанавливаем количество столбцов в таблице равным 1.

self.tableView.setColumnCount(1)

self.tableView.setEditTriggers(QAbstractItemView.EditTrigger(0))

header\_item = QTableWidgetItem('Умы России')

header\_item.setFont(QFont('Arial', 24)) # Замените 'Arial' на шрифт, который вы предпочитаете

self.tableView.setHorizontalHeaderItem(0, header\_item)

#Устанавливаем вид и размеры таблицы

self.tableView.horizontalHeader().setDefaultSectionSize(1000)

self.tableView.setStyleSheet(

"QHeaderView::section { background-color:rgb(0, 0, 63); color:white} QTableWidget {"

"background-color:rgb(0, 0, 63); color:white; font-size: 22px}QTableWidget "

"QTableCornerButton::section {background-color: rgb(0, 0, 63);}")

#Заполняем таблицу данными из базы данных.

x = 0

for name in employees:

self.tableView.setItem(x, 0, QTableWidgetItem(name[1]))

x += 1

# Определение метода для получения id\_del сотрудника по выбранному имени

def on\_click\_del\_id(self):

# Получаем имя сотрудника из выбранной строки таблицы

name = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

# Устанавливаем соединение с базой данных 'data.db'

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

# Выполняем SQL-запрос для получения id сотрудника по его имени

cursor.execute(f'SELECT id FROM employees WHERE name = "{name}"')

# Получаем результат запроса

employees = cursor.fetchall()

# Устанавливаем id\_del для последующего удаления

self.id\_del = employees[0][0]

# Определение метода для обработки события выбора сотрудника в таблице

def on\_click(self):

# Глобальная переменная для хранения id администратора

global id\_admin

# Получаем имя сотрудника из выбранной строки таблицы

name\_q = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

# Устанавливаем соединение с базой данных 'data.db'

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

# Выполняем SQL-запрос для получения id сотрудника по его имени

cursor.execute(f'SELECT id FROM employees WHERE name = "{name\_q}"')

# Получаем результат запроса

employees = cursor.fetchall()

# Устанавливаем id администратора для последующего использования

id\_admin = employees[0][0]

# Выполняем SQL-запрос для получения изображения сотрудника

cursor.execute(f'SELECT img FROM employees WHERE name = "{name\_q}"')

# Получаем результат запроса и загружаем изображение в окно

adminstati\_window.load\_image(cursor.fetchall()[0][0])

# Отображаем окно с изображением и обновляем отображение таблицы

adminstati\_window.show()

adminstati\_window.table\_view()

# ПРИЛОЖЕНИЕ 5

Листинг 5. Добавление сотрудника

class Employees(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка интерфейса из файла employees.ui

uic.loadUi('employees.ui', self)

self.setFixedSize(400, 250)

# Привязывается сигнал нажатия кнопки

self.pushButton.clicked.connect(self.addemp)

def addemp(self):

# Получаются значения из полей ввода и сохраняются в переменные

emp\_name = self.lineEdit.text()

emp\_img = self.lineEdit\_1.text()

# Соединение с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

# Выполняется SQL-запрос для получения всех id из таблицы

cursor.execute(f'SELECT id FROM employees')

employees = cursor.fetchall()

# Вычисляется новый id для нового сотрудника путем увеличения последнего id в таблице на 1

emp\_id = employees[-1][0] + 1

# Выполняется SQL-запрос для вставки новой записи в таблицу с полученными значениями

cursor.execute(f'INSERT INTO employees VALUES ({emp\_id}, "{emp\_name}", "{emp\_img}")')

connection.commit()

cursor.close()

connection.close()

empl\_window.close()

minds\_window.table\_view()

adminminds\_window.table\_view()

# ПРИЛОЖЕНИЕ 6

Листинг 6. Администраторское окно публикаций учёного

class AdminStatiWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка интерфейса из файла AdminStati.ui

uic.loadUi('AdminStati.ui', self)

self.setFixedSize(1000, 600)

# Инициализация отображения таблицы

self.table\_view()

# Инициализация переменной для хранения выбранного заголовка

self.id\_del\_st = ''

# Привязка кнопки к методу удаления

self.pushButton\_2.clicked.connect(self.on\_click\_del)

# Привязка кнопки к методу добавления

self.pushButton.clicked.connect(self.on\_click\_add)

# Привязка кнопки к методу закрытия окна

self.pushButton\_leave.clicked.connect(self.on\_click\_leave)

# Метод для загрузки изображения и отображения его на виджете

def load\_image(self, file\_name):

# Создание объекта QPixmap из файла

pixmap = QPixmap(file\_name)

pixmap = pixmap.scaled(300, 300)

# Создание QLabel для отображения изображения

self.label1 = QLabel(self)

# Установка QPixmap в QLabel

self.label1.setPixmap(pixmap)

# Установка размеров QLabel

self.label1.resize(300, 300)

# Установка позиции QLabel в окне

self.label1.move(600, 150)

def on\_click\_leave(self):

# Метод для закрытия окна

adminstati\_window.close()

def on\_click\_add(self):

# Метод для отображения article\_window

article\_window.show()

def on\_click\_del(self):

# Метод для удаления статьи из базы данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute(f'DELETE FROM articles WHERE title = "{self.id\_del\_st}"')

connection.commit()

cursor.close()

connection.close()

# Перезагрузка отображения таблицы после удаления

self.table\_view()

def show\_minds\_window(self):

# Метод для закрытия окна

adminstati\_window.close()

# Метод для отображения таблицы статей

def table\_view(self):

# Установка соединения с базой данных

connection = sqlite3.connect('data.db')

# Создание курсора для выполнения SQL-запросов

cursor = connection.cursor()

cursor.execute(f'SELECT \* FROM articles WHERE id = {id\_admin}')

articles = cursor.fetchall()

# Установка количества строк и столбцов в таблице

self.tableView.setRowCount(len(articles))

self.tableView.setColumnCount(1)

self.tableView.setHorizontalHeaderItem(0, QTableWidgetItem('Статьи'))

self.tableView.horizontalHeader().setDefaultSectionSize(500)

# Запрет редактирования ячеек

self.tableView.setEditTriggers(QAbstractItemView.EditTrigger(0))

self.tableView.setStyleSheet(

"QHeaderView::section { background-color:rgb(0, 0, 63); color:white} QTableWidget {"

"background-color:rgb(0, 0, 63); color:white; font-size: 22px}QTableWidget "

"QTableCornerButton::section {background-color: rgb(0, 0, 63);}")

x = 0

for title in articles:

# Заполнение ячеек таблицы данными из базы данных

self.tableView.setItem(x, 0, QTableWidgetItem(title[1]))

x += 1

# Привязка событий к таблице

self.tableView.clicked.connect(self.on\_click)

self.tableView.doubleClicked.connect(self.on\_double\_click)

def on\_click(self):

# Метод при клике на ячейку таблицы для получения выбранного заголовка

self.id\_del\_st = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

# Метод при двойном клике на ячейку таблицы для отображения stati\_content\_window

def on\_double\_click(self):

# Использование глобальной переменной для передачи заголовка статьи

global title

# Получение выбранного заголовка

title = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

# Отображение окна

stati\_content\_window.show()

# Вызов метода для отображения содержимого окна

stati\_content\_window.show\_label()

# ПРИЛОЖЕНИЕ 7

Листинг 7. Добавление статьи

class Article(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загружаем пользовательский интерфейс из файла 'article.ui'

uic.loadUi('article.ui', self)

self.setFixedSize(400, 400)

# Связываем событие нажатия кнопки с методом addemp

self.pushButton.clicked.connect(self.addemp)

def addemp(self):

# Получаем данные из виджетов

title = self.lineEdit.text()

content = self.textEdit.toPlainText()

# Устанавливаем соединение с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute(f'INSERT INTO articles VALUES ({id\_admin}, "{title}", "{content}")')

# Фиксируем изменения и закрываем соединение

connection.commit()

cursor.close()

connection.close()

article\_window.close()

# Обновляем таблицы в окнах adminstati\_window и stati\_window

adminstati\_window.table\_view()

stati\_window.table\_view()

# ПРИЛОЖЕНИЕ 8

Листинг 8. Отображение списка учёных

class MindsWindow(QDialog):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Загрузка интерфейса из файла Minds.ui

uic.loadUi('Minds.ui', self)

self.setFixedSize(1000, 650)

# Инициализация отображения таблицы

self.table\_view()

# Метод для отображения таблицы сотрудников

def table\_view(self):

# Установка соединения с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute('SELECT \* FROM employees')

employees = cursor.fetchall()

# Установка количества строк и столбцов в таблице

self.tableView.setRowCount(len(employees))

self.tableView.setColumnCount(1)

# Запрет редактирования ячеек

self.tableView.setEditTriggers(QAbstractItemView.EditTrigger(0))

self.tableView.setHorizontalHeaderItem(0, QTableWidgetItem('Умы России'))

self.tableView.horizontalHeader().setDefaultSectionSize(1000)

self.tableView.setStyleSheet(

"QHeaderView::section { background-color:rgb(0, 0, 63); color:white} QTableWidget {"

"background-color:rgb(0, 0, 63); color:white; font-size: 22px}QTableWidget "

"QTableCornerButton::section {background-color: rgb(0, 0, 63);}")

x = 0

# Заполнение ячеек таблицы данными из базы данных

for name in employees:

self.tableView.setItem(x, 0, QTableWidgetItem(name[1]))

x += 1

# Привязка события клика по таблице к методу on\_click

self.tableView.clicked.connect(self.on\_click)

# Метод при клике на ячейку таблицы

def on\_click(self):

# Использование глобальной переменной для передачи идентификатора сотрудника

global id

# Получение выбранного имени

name = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

# Установка соединения с базой данных

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

cursor.execute(f'SELECT id FROM employees WHERE name = "{name}"')

employees = cursor.fetchall()

# Получение идентификатора сотрудника

id = employees[0][0]

# Выполнение SQL-запроса для получения изображения сотрудника и его отображение

cursor.execute(f'SELECT img FROM employees WHERE name = "{name}"')

stati\_window.load\_image(cursor.fetchall()[0][0])

stati\_window.show()

# Обновление отображения таблицы в stati\_window

stati\_window.table\_view()

# ПРИЛОЖЕНИЕ 9

Листинг 9. Просмотр публикаций

class StatiWindow(QDialog):

def \_\_init\_\_(self, ):

super().\_\_init\_\_()

# Загрузка интерфейса из файла Stati.ui

uic.loadUi('Stati.ui', self)

self.setFixedSize(1000, 600)

self.pushButton.clicked.connect(self.show\_minds\_window)

def show\_minds\_window(self):

stati\_window.close()

def load\_image(self, file\_name):

#Загружает изображение из файла file\_name

pixmap = QPixmap(file\_name)

pixmap = pixmap.scaled(300, 300)

#Отображение изображения на виджете

self.label1 = QLabel(self)

self.label1.setPixmap(pixmap)

self.label1.resize(300, 300)

#Размер и положение виджета

self.label1.move(600, 150)

def table\_view(self):

# Установка соединения с базой данных SQLite

connection = sqlite3.connect('data.db')

cursor = connection.cursor()

# Выполнение SQL-запроса для получения статей

cursor.execute(f'SELECT \* FROM articles WHERE id = {id}')

articles = cursor.fetchall()

# Установка количества строк и столбцов в таблице

self.tableView.setRowCount(len(articles))

self.tableView.setColumnCount(1)

self.tableView.setHorizontalHeaderItem(0, QTableWidgetItem('Статьи'))

self.tableView.horizontalHeader().setDefaultSectionSize(500)

# Запрет редактирования ячеек

self.tableView.setEditTriggers(QAbstractItemView.EditTrigger(0))

self.tableView.setStyleSheet(

"QHeaderView::section { background-color:rgb(0, 0, 63); color:white} QTableWidget {"

"background-color:rgb(0, 0, 63); color:white; font-size: 22px}QTableWidget "

"QTableCornerButton::section {background-color: rgb(0, 0, 63);}")

x = 0

# Заполнение таблицы

for title in articles:

self.tableView.setItem(x, 0, QTableWidgetItem(title[1]))

x += 1

self.tableView.clicked.connect(self.on\_click)

def on\_click(self):

global title

# Получение текста выбранной ячейки таблицы

title = self.tableView.item(self.tableView.selectedItems()[0].row(), 0).text()

stati\_content\_window.show()

# Вызов метода для отображения контента в окне

stati\_content\_window.show\_label()

# ПРИЛОЖЕНИЕ 10

Листинг 10. Завершение программы# Проверка, запущен ли скрипт напрямую (а не импортирован как модуль)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

# Инициализируем некоторые переменные, используемые в приложении

id = -1

id\_admin = -1

title = ''

# Создаем экземпляры различных окон нашего приложения

window = MainWindow()

admin\_window = AdminWindow()

minds\_window = MindsWindow()

stati\_window = StatiWindow()

article\_window = Article()

stati\_content\_window = StatiContentWindow()

adminminds\_window = AdminMindsWindow()

adminstati\_window = AdminStatiWindow()

empl\_window = Employees()

# Показываем главное окно

# Завершаем выполнение приложения при закрытии главного окна

window.show()

sys.exit(app.exec())