Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

КУРСОВАЯ РАБОТА (ПРОЕКТ)

по дисциплине «Проектирование и архитектура программных систем»
наименование дисциплины

на тему Алгоритмическая библиотека

Направление подготовки (специальность) 09.03.04
(код, наименование)
Программная инженерия

Автор работы (проекта) Рефуз К. Б. _____
(инициалы, фамилия) (подпись, дата)

Группа ПО-12б

Руководитель работы (проекта) А. А. Чаплыгин _____
(инициалы, фамилия) (подпись, дата)

Работа (проект) защищена _____
(дата)

Оценка _____

Члены комиссии _____ _____
подпись, дата фамилия и. о.

_____ _____
подпись, дата фамилия и. о.

_____ _____
подпись, дата фамилия и. о.

Курск 2024 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ (ПРОЕКТ)

Студента Рефуз К. Б., шифр 21-06-0377, группа ПО-12б

1. Тема «Алгоритмическая библиотека » .

2. Срок предоставления работы к защите «15» Март 2024 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

1) разработка концептуальной модели алгоритмической библиотеки;
2) разработать алгоритмическая библиотека;
3) Составление и тестирование алгоритмических библиотек.

3.2. Входные данные и требуемые результаты для программы:

1) Входными данными для программной системы являются: команды растрового редактора.

2) Выходными данными для программной системы являются: фигуры, выводящиеся на окно приложения, .

4. Содержание работы (по разделам):

4.1. Введение

4.1. Анализ предметной области

4.2. Техническое задание: основание для разработки, назначение разработки, требования к программной системе, требования к оформлению документации.

4.3. Технический проект: общие сведения о программной системе, проект данных программной системы, проектирование архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.4. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.5. Заключение

4.6. Список использованных источников

5. Перечень графического материала:

Руководитель работы (проекта)		А. А. Чаплыгин
	(подпись, дата)	(инициалы, фамилия)
Задание принял к исполнению		Рефуз К. Б.
	(подпись, дата)	(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 33 страницам. Работа содержит 6 иллюстраций, 5 таблиц, 10 библиографических источников и 0 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: алгоритм, администратор, пользовател, устаревшие методы, добавление, удаление, модификация.

Объектом разработки является редактор алгоритмической библиотеки, который позволяет пользователю искать различные предлагаемые алгоритмы в программе и позволяет администратору изменять добавлять и удалять алгоритм .

Целью проекта является создание удобной и функциональной алгоритмической библиотеки, облегчающей поиск алгоритмов

В процессе создания библиотека алгоритмов была создана с использованием современного языка программирования Python.

ABSTRACT

The volume of work is 33 pages. The work contains 6 illustrations, 5 tables, 10 bibliographic sources and 0 sheets of graphic material. The number of applications is 2. The graphic material is presented in annex A. The layout of the site, including the connection of components, is presented in annex B.

The list of keywords: algorithm, administrator, user, outdated methods, addition, delete, modification.

The object of the development is the editor of the algorithmic library, which allows the user to search for various proposed algorithms in the program and allows the administrator to change, add and delete the algorithm.

The aim of the project is to create a convenient and functional algorithmic library that facilitates the search for algorithms

During the creation process, the algorithm library was created using the modern Python programming language.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИС – информационная система.

ИТ – информационные технологии.

КТС – комплекс технических средств.

ОМТС – отдел материально-технического снабжения.

ПО – программное обеспечение.

РП – рабочий проект.

СУБД – система управления базами данных.

ТЗ – техническое задание.

ТП – технический проект.

ВВЕДЕНИЕ

Библиотека алгоритмов - это набор предварительно написанных функций, классов или модулей, которые обеспечивают эффективные и готовые к использованию реализации обычно используемых алгоритмов.

Алгоритмические библиотеки сыграли решающую роль в развитии вычислительной техники и решении проблем. Первоначально появившиеся в 1950-х и 1960-х годах, эти библиотеки были разработаны для предоставления эффективных решений алгоритмических задач, возникавших при обработке данных на ранних компьютерах. Со временем эти библиотеки эволюционировали и теперь включают в себя широкий спектр алгоритмов, от классических алгоритмов сортировки до алгоритмов расширенного поиска, методов оптимизации и алгоритмов построения графиков.

В настоящее время существует множество графических редакторов. Они имеют различную стоимость и набор функций. Некоторые из них предназначены для профессиональной работы с изображениями, а другие - для обычных пользователей.

Цель данной работы - разработка библиотеки алгоритмов. Для достижения этой цели необходимо решить *следующие задачи:*

- Провести анализ предметной области;
- Разработать концептуальную модель библиотеки алгоритмов;
- Реализовать библиотеку.

Структура и объем работы. Отчет состоит из введения, 4 основных разделов, заключения, списка использованных источников и 2 приложений. Текст отчета о квалификации составляет 8 страницам.

Во введении сформулирована цель работы, определены задачи разработки, описана структура работы и представлен краткий обзор каждого из разделов.

В первом разделе на этапе описания технических характеристик предметной области собирается информация о существующих библиотеках алгоритмов.

Во втором разделе на этапе составления технического задания устанавливаются требования к разрабатываемой библиотеке алгоритмов.

В третьем разделе на этапе технического проектирования представлены проектные решения для библиотеки алгоритмов.

В четвертом разделе представлен список классов и их методов, использованных при разработке библиотеки алгоритмов, а также проведено тестирование разработанной библиотеки.

В заключении изложены основные результаты работы, полученные в процессе разработки.

В приложении А представлен графический материал. В приложении Б представлены фрагменты исходного кода.

1 Анализ предметной области

1.1 История развития алгоритмических библиотек

История разработки алгоритмических библиотек восходит к ранним дням информатики и программирования. В первые десятилетия развития информатики программы часто писались для конкретных задач, а алгоритмы часто встраивались непосредственно в код этих программ. Однако с ростом сложности программного обеспечения и компьютерных систем становилось все более очевидным, что для эффективного управления алгоритмами необходим модульный и многоразовый подход.

В 1950-х и 1960-х годах, когда информатика начала выделяться в отдельную академическую область, начали разрабатываться первые алгоритмические библиотеки. Эти ранние библиотеки часто были специфичны для конкретной платформы или языка программирования и в основном использовались в рамках исследований и разработок в университетах и исследовательских лабораториях.

В течение следующих десятилетий, с расширением компьютерной индустрии и появлением языков программирования, алгоритмические библиотеки росли в геометрической прогрессии. Стандартизированные и обобщенные библиотеки были разработаны для широкого спектра областей, от математики и информатики до практических приложений, таких как манипулирование строками символов, сжатие данных и криптография.

В 1980-х и 1990-х годах, с появлением интернета и появлением программного обеспечения с открытым исходным кодом, многие алгоритмические библиотеки были разработаны и распространены как бесплатное программное обеспечение.

Сегодня алгоритмические библиотеки являются неотъемлемой частью инструментария любого программиста или разработчика программного обеспечения. Они используются во множестве приложений, от простых алгоритмов сортировки до передовых методов машинного обучения и искусственного интеллекта.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является потребность в создании растрового редактора в рамках проекта по предмету ”Проектирование и разработка программных систем”.

2.2 Цель и назначение разработки

Основная цель этого проекта-разработать алгоритмические библиотеки enn используя технологии Python.

Цель разработки алгоритмической библиотеки - объединить простой и сложный лагорифмический поиск.

Задачи данной разработки включают:

для пользователя:

- используется для поиска алгоритма.
- позволяет отображать содержимое поляризатора.

для администратора:

- позволяет изменить режим пропуска администратора.
- позволяет изменять алгоритм.
- позволяет добавить алгоритм.
- используется для подавления алгоритма.

2.3 Описание алгоритмической библиотеки

Библиотека алгоритмов представляет собой программное обеспечение, разработанное для организации и управления набором алгоритмов. Она обладает следующими особенностями:

1. **Навигация:** Доступен боковой панельный список всех алгоритмов с возможностью прокрутки, обеспечивая удобную навигацию между доступными вариантами.

2. **Отображение содержимого:** Для выбранного алгоритма предоставляется область текста, где отображается его содержание с подробным описа-

нием и примерами использования, что помогает пользователям понять его работу.

3. **Режимы доступа:** Реализованы два режима доступа - пользовательский и администраторский, с обеспечением безопасности через аутентификацию по паролю для режима администратора.

4. **Редактирование и управление:** Администраторы имеют возможность добавлять, редактировать и удалять алгоритмы, а также изменять пароль администратора для обеспечения полного контроля над библиотекой алгоритмов.

2.3.1 Функциональные элементы

Интерфейс должен предоставлять следующие элементы и функции:

- **Выбор инструментов:** Пользователи должны иметь возможность выбирать необходимые инструменты из специальной панели, обеспечивая таким образом простой и интуитивный выбор функций библиотеки.

- **Создание алгоритмов:** Должна быть возможность создавать новые алгоритмы и определять их характеристики с помощью специальных инструментов, позволяя пользователям вносить свой вклад в содержание библиотеки.

- **Редактирование алгоритмов:** Пользователи должны иметь возможность изменять существующие алгоритмы, настраивая их параметры или добавляя дополнительные функции, чтобы соответствовать их конкретным потребностям.

- **Удаление алгоритмов:** Должна быть возможность удаления нежелательных алгоритмов из библиотеки, предоставляя пользователям полный контроль над ее содержимым и качеством.

Композиция интерфейса редактора представлена на рисунке 2.1 и 2.2.

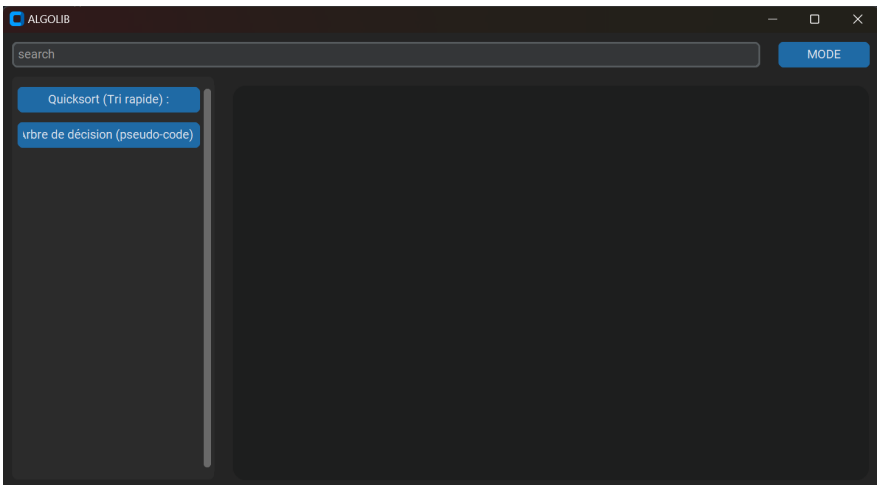

Рисунок 2.1 – Компоновка графического интерфейса алгоритмической библиотеки.

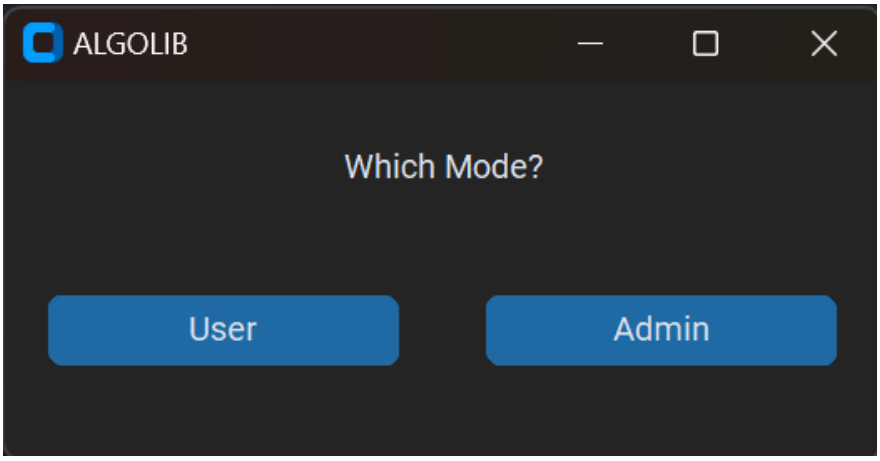

Рисунок 2.2 – интерфейс с выбором из двух доступных режимов

2.4 Требования пользователя к интерфейсу алгоритмической библиотеки

Ралгоритмическая библиотека должна создать простой и удобный интерфейс и предоставить следующие функции:

- бар поиска для фильтрации результатов по названию алгоритма.
- левая боковая панель с раскрывающимся списком всех алгоритмов.
- текстовая область с правой стороны для отображения содержимого выбранного алгоритма.

2.5 Моделирование вариантов использования

2.5.1 Диаграмма прецедентов

Для библиотеки алгоритмов было разработано моделирование, позволяющее четко визуализировать различные варианты использования этой библиотеки. Это моделирование облегчает физическую разработку и подробный анализ взаимодействий между различными функциями и компонентами библиотеки. При создании этого моделирования предпочтение было отдано использованию языка визуального моделирования UML.

Диаграмма вариантов использования описывает основную функциональность библиотеки алгоритмов. Это включает в себя действия, которые библиотека будет выполнять при ее использовании. Диаграмма представляет систему в виде серии вариантов использования, которые предлагаются пользователям или объектам, взаимодействующим с библиотекой. Вариант использования описывает набор действий, которые пользователь может выполнять с помощью библиотеки алгоритмов.

На основе анализа функциональной области библиотеки алгоритмов необходимо реализовать следующие варианты использования :

(a) Поиск алгоритмов по названию.

(b) Просмотр подробной информации о конкретном алгоритме, включая его описание.

(c) Добавление новых алгоритмов в библиотеку или изменение существующих.

(d) Удаление алгоритмов из библиотеки.

(e) Обновление информации о библиотеке и ее содержимом.

(f) Изменение пароля администратора.

Рисунок 2.3 – Диаграмма прецедентов

2.5.2 Сценарии прецедентов программы

(a) Сценарий для случая использования ”Поиск алгоритма”:

- Основной исполнитель: Пользователь;

- Заинтересованные стороны и их требования: Пользователь вводит название алгоритма в строку поиска;

- Предварительное условие: Пользователь авторизован;

- Основной успешный сценарий: Алгоритм автоматически сортируется при вводе пользователем.

(b) Сценарий для случая использования ”Показать описание выбранного алгоритма”:

- Основной исполнитель: Пользователь;

- Заинтересованные стороны и их требования: Пользователь должен выбрать алгоритм и отобразить его описание;

- Предварительное условие: Пользователь открыл библиотеку алгоритмов;

- Основной успешный сценарий: Пользователь нажимает на соответствующую кнопку для выбора желаемого алгоритма и отображения его описания.

(c) Сценарий для случая использования ”Добавить новый алгоритм”:

- Основной исполнитель: Администратор;

- Заинтересованные стороны и их требования: Администратор должен добавить новый алгоритм в библиотеку;

- Предварительное условие: Администратор авторизован как администратор;

- Основной успешный сценарий: Администратор нажимает на кнопку ”Добавить алгоритм”и следует инструкциям для успешного добавления алгоритма.

(d) Сценарий для случая использования ”Удалить алгоритм”:

- Основной исполнитель: Администратор;

- Заинтересованные стороны и их требования: Администратор должен удалить существующий алгоритм из библиотеки;

- Предварительное условие: Администратор авторизован как администратор;

- Основной успешный сценарий: Администратор нажимает на кнопку ”Удалить алгоритм”и следует инструкциям для успешного удаления алгоритма.

(e) Сценарий для случая использования ”Изменить алгоритм”:

- Основной исполнитель: Администратор;

- Заинтересованные стороны и их требования: Администратор должен изменить существующий алгоритм в библиотеке;

- Предварительное условие: Администратор авторизован как администратор;

- Основной успешный сценарий: Администратор нажимает на кнопку ”Изменить алгоритм”и следует инструкциям для успешного изменения алгоритма.

(f) Сценарий для случая использования ”Изменить пароль”:

- Основной исполнитель: Администратор;

- Заинтересованные стороны и их требования: Администратор должен изменить свой пароль;

- Предварительное условие: Администратор авторизован как администратор;

- Основной успешный сценарий: Администратор нажимает на кнопку ”Изменить пароль”и следует инструкциям для успешного изменения пароля.

2.6 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Вы должны спроектировать и разработать алгоритмическую библиотеку. библиотека реализована на Python.

3.2 Описание используемых библеотек и языков программирования

Проект реализован с использованием языка Python.

3.3 Технические детали проекта

3.3.1 Интерфейс редактора

Интерфейс библиотеки алгоритмов состоит из панели поиска для фильтрации результатов по названию алгоритма, левой боковой панели с раскрывающимся списком всех алгоритмов, текстового поля с правой стороны для отображения содержимого выбранного алгоритма и кнопки для переключения режимов..

3.4 Структура проекта

3.4.1 Диаграмма классов

На рисунке 3.1 показана диаграмма классов для моего проекта Библиотека алгоритмики. Эта диаграмма иллюстрирует взаимодействие между различными классами.

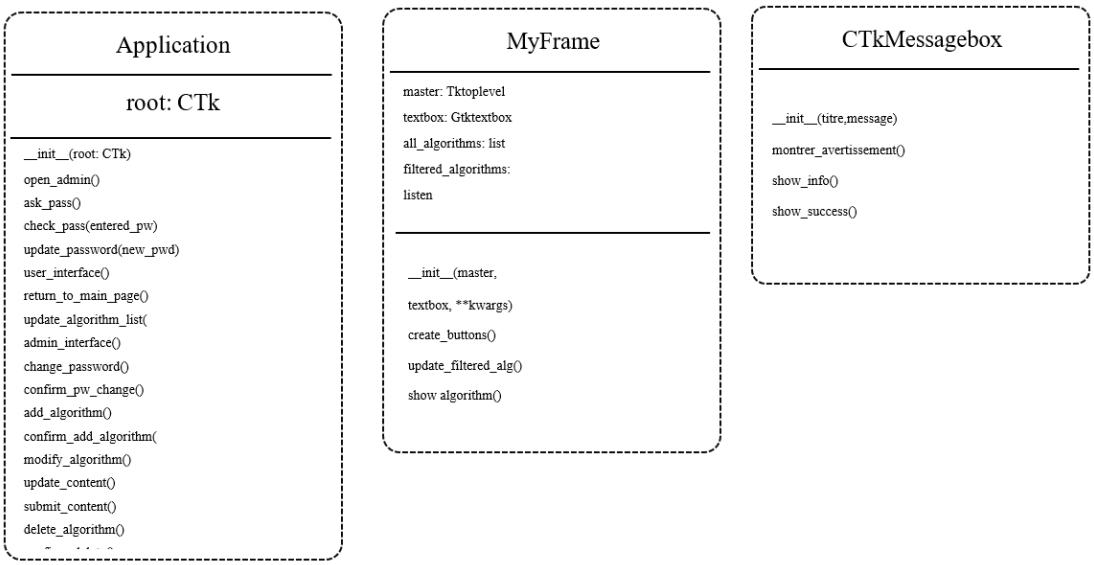

Рисунок 3.1 – Диаграмма классов

4 Рабочий проект

4.1 Классы, используемые для разработки редактора

Можно выделить следующий список классов и их методов, использованных при разработке web-приложения (таблица 4.1). Пример таблицы с уменьшенным межстрочным интервалом.

Таблица 4.1 – Описание классов, используемых в приложении

Название класса	Модуль, к которому относится класс	Описание класса	Методы
1	2	3	4
Application	app.py	Этот класс представляет главное приложение ALGOLIB	_init_(root: CTk), open_admin(),ask_pass(), check_pass(entered_pw), update_password(), user_interface(), return_to_main_page() ,update_algorithm_list(), admin_interface(), change_password() ,confirm_pw_change(), add_algorithm(), confirm_add_algorithm() ,modify_algorithm(), update_content(), submit_content(), delete_algorithm().

Продолжение таблицы 4.1

1	2	3	4
MyFrame	my_frame.py	Этот класс представляет рамку, содержащую алгоритмы.	_init_(master,textbox, **kwargs), create_buttons(), update_filtered_alg(), show algorithm(), master: Tktoplevel, textbox: Gtktextbox, all_algorithms: list, filtered_algorithms:, listen
CTk Messagebox	CTkMessageboxь	Этот класс представляет окно сообщения. Он отображает сообщения для пользователя.	Имеет дополнительные две точки для cross и init.

4.2 Системное тестирование алгоритмической библиотеки .

(a) Случай использования «алгоритм поиска»:

- основной исполнитель: пользователь;

- заинтересованные лица и их требования: пользователю необходимо добавить определенную фигуры на окно редактора;

- предусловие: пользователь открыл растровый редактор;

- ожидаемый результат: пользователь нажимает нужную горячую клавишу, ставит две координатные точки фигуры с помощью мышки, после этого фигура появляется на окно редактора. В итоге на окно редактора добавлятся выбранный пользователем геомерический обьект.

- результат представлен на рисунке 4.1

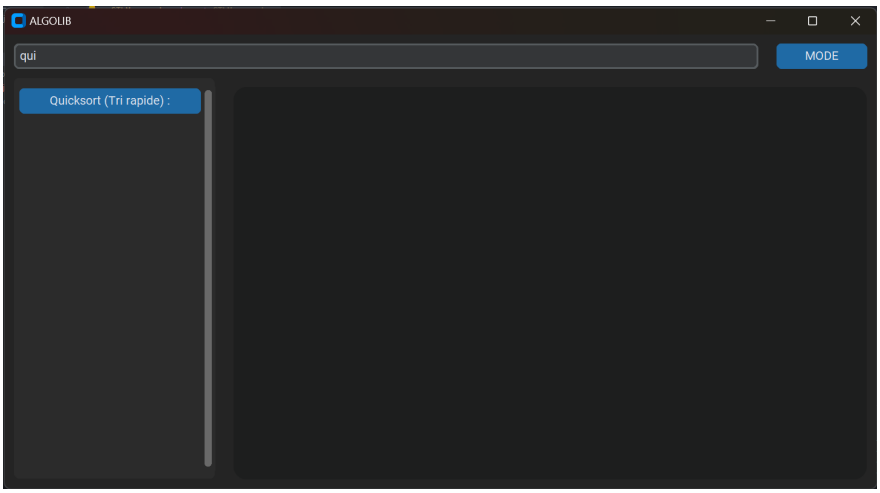

Рисунок 4.1 – алгоритм поиска

(b) Случай использования «Удаление геометрического обьекта»:

- вариант использования ”показать содержимое алгоритма”
- главный исполнитель: пользователь;
- заинтересованные стороны и их требования: пользователь нажимает кнопку в алгоритме, который он хочет;
- предварительное условие: пользователь открыл алгоритмическую библиотеку.
- ожидаемый результат: отображается описание географии.
- результат показан на рисунке 4.2.

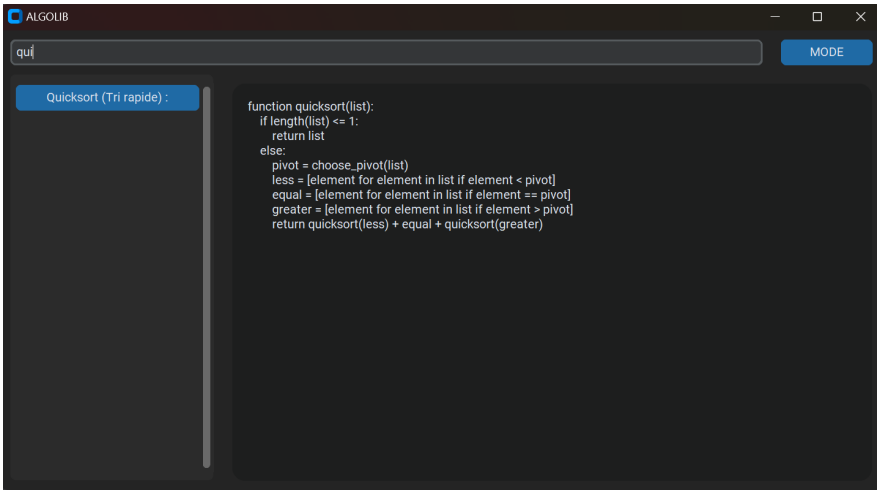

Рисунок 4.2 – показан на рисунке

ЗАКЛЮЧЕНИЕ

Преимуществом алгоритмических библиотек является их гибкость, возможность быстрого поиска сложных алгоритмов.

Основные результаты работы:

(a) анализ предметной области. .

(b) была разработана концептуальная библиотечно-алгоритмическая модель. Разработайте модель системных данных. Системные требования определены.

(c) была выполнена разработка алгоритмической библиотеки.

(d) алгоритмическая библиотека была реализована и протестирована. Проведение калибровки.

Все требования, объявленные в техническом задании, были полностью реализованы, все задачи, поставленные в начале разработки проекта, были также решены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фримен, А. Практикум по программированию на Python / А. Фримен. – Москва : Вильямс, 2015. – 720 с. – ISBN 978-5-8459-1799-7.

2. Страуструп, Б. Язык программирования C++. Лекции и упражнения / Б. Страуструп. – Санкт-Петербург : БХВ-Петербург, 2018. – 880 с. – ISBN 978-5-9579-2180-5.

3. Гудман, Д. Полное руководство по разработке веб-приложений с использованием Django / Д. Гудман. – Москва : ДМК Пресс, 2017. – 624 с. – ISBN 978-5-94074-625-8.

4. Вандер Плас, Дж. Python для сложных задач. Наука о данных и машинное обучение / Дж. Вандер Плас. – Санкт-Петербург : Питер, 2016. – 560 с. – ISBN 978-5-496-01049-8.

5. Лутц, М. Изучаем Python. Программирование игр, визуализация данных, веб-приложения / М. Лутц. – Москва : ДМК Пресс, 2019. – 864 с. – ISBN 978-5-97060-729-5.

6. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – Санкт-Петербург : Питер, 2014. – 896 с. – ISBN 978-5-496-01049-8.

7. Сталлингс, У. Компьютерные сети. Принципы, технологии, протоколы / У. Сталлингс. – Москва : Издательский дом Вильямс, 2017. – 864 с. – ISBN 978-5-8459-2125-3.

8. Мартин, Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин. – Санкт-Петербург : Питер, 2018. – 464 с. – ISBN 978-5-4461-1089-3.

9. Календер, Д. Принципы объектно-ориентированного программирования на C++ с примерами на C и Java / Д. Календер. – Москва : ДМК Пресс, 2016. – 512 с. – ISBN 978-5-97060-858-2.

10. Хантер, Э. Программирование на Python 3. Подробное руководство / Э. Хантер. – Москва : ДМК Пресс, 2019. – 560 с. – ISBN 978-5-97060-827-8.

ПРИЛОЖЕНИЕ А

Фрагменты исходного кода программы

app.py

```
# app.py
import json
import hashlib  # Import the hashlib library
from CTkMessagebox import CTkMessagebox
import customtkinter as ctk
from my_frame import MyFrame

class App(ctk.CTk):
    """Classe principale représentant l'application ALGOLIB."""

    def __init__(self, root):
        """Constructeur de la classe App."""
        self.root = root
        self.root.title("")
        self.root.geometry("350x150")

        # Set window properties
        self.root.title("ALGOLIB")

        # Set appearance mode to "dark"
        ctk.set_appearance_mode("dark")

        self.root.grid_columnconfigure((0, 1), weight=1)

        # Label for mode selection
        self.label = ctk.CTkLabel(self.root, text="Which Mode?")
        self.label.grid(row=0, column=0, columnspan=2, pady=(20, 1), sticky="
            n")

        # Button for User mode
        self.button_user = ctk.CTkButton(self.root, text="User", command=self
            .user_interface)
        self.button_user.grid(row=1, column=0, padx=(0, 0), pady=(0, 0))

        # Button for Admin mode
        self.button_admin = ctk.CTkButton(self.root, text="Admin", command=
            self.open_admin)
        self.button_admin.grid(row=1, column=1, padx=(0, 0), pady=(0, 0))

        # Configure grid weights for responsiveness
        self.root.grid_rowconfigure(1, weight=1)
        self.root.grid_columnconfigure(1, weight=1)

    def open_admin(self):
        """Ouvre l'interface administrateur."""
        password = self.ask_pass()
        if self.check_pass(password):
            self.admin_interface()
        else:
```

```
47              CTkMessagebox(title="Information", message="Invalid Password")
48
49      def ask_pass(self):
50          """Demande le mot de passe à l'utilisateur."""
51          password_window = ctk.CTkToplevel(self.root)
52          password_window.title("Enter Password")
53          password_window.geometry("300x200")
54
55          label = ctk.CTkLabel(password_window, text="Enter Password:")
56          label.pack(pady=10)
57
58          entry = ctk.CTkEntry(password_window, show="*")
59          entry.pack(pady=10)
60
61          entered_password = ctk.StringVar()
62
63          def submit_password():
64              entered_password.set(entry.get())
65              password_window.destroy()
66
67          button = ctk.CTkButton(password_window, text="Submit", command=
                submit_password)
68          button.pack()
69
70          password_window.wait_window()
71          return entered_password.get()
72
73      def check_pass(self, entered_password):
74          """Vérifie si le mot de passe est valide."""
75          try:
76              with open("passwords.json", "r") as file:
77                  passwords = json.load(file)
78          except FileNotFoundError:
79              passwords = {}
80
81          # Hash the entered password using SHA-256
82          entered_password_hash = hashlib.sha256(entered_password.encode()).
                hexdigest()
83
84          # Compare the hashed password with the stored hash value
85          return passwords.get("admin_password") == entered_password_hash
86
87      def update_password(self, new_password):
88          """Met à jour le mot de passe dans le fichier JSON avec le hachage.
                """
89          # Hash the new password using SHA-256
90          new_password_hash = hashlib.sha256(new_password.encode()).hexdigest()
91
92          # Load the existing JSON data
93          with open('passwords.json', 'r') as json_file:
94              data = json.load(json_file)
95
96          # Update the hashed password
97          data['admin_password'] = new_password_hash
```

```
98
99          # Write the updated data back to the JSON file
100         with open('passwords.json', 'w') as json_file:
101             json.dump(data, json_file)
102
103     def user_interface(self):
104         """Interface utilisateur."""
105         self.root.withdraw()
106         user_interface = ctk.CTkToplevel(self.root)
107         user_interface.title("ALGOLIB")
108         user_interface.geometry("960x500")
109
110          # Barre de recherche en haut
111         search_bar = ctk.CTkEntry(user_interface, placeholder_text="search")
112         search_bar.grid(row=0, column=0, columnspan=2, pady=10, padx=10,
                sticky="we")
113
114         # Bouton à côté de la barre de recherche
115         self.mode = ctk.CTkButton(user_interface, text="MODE", width=100,
                command=self.return_to_main_page)
116         self.mode.grid(row=0, column=2, pady=10, padx=10, sticky="w")
117
118         # Champ de texte à droite, occupant le reste de la page
119         self.textbox = ctk.CTkTextbox(user_interface, corner_radius=15)
120         self.textbox.grid(row=1, column=1,columnspan=2, padx=10, pady=10,
                sticky="nsew")
121
122         # Passer le CTkTextbox à la classe MyFrame
123         self.scrollable_frame = MyFrame(master=user_interface, textbox=self.
                textbox, width=200, height=450)
124         self.scrollable_frame.grid(row=1, column=0, padx=10, pady=(0, 10),
                sticky="ns")
125
126         user_interface.grid_rowconfigure(1, weight=1)  # Définir le poids à 1
                pour la ligne contenant le champ de texte
127         user_interface.grid_columnconfigure(1, weight=1)  # Définir le poids
                à 1 pour la colonne contenant le champ de texte
128         user_interface.grid_columnconfigure(0, weight=0)  # Définir le poids
                à 0 pour la colonne contenant le cadre déroulant
129
130         ctk.set_appearance_mode("dark")
131
132         # Associez la fonction de mise à jour à l'événement de modification
                de la barre de recherche
133         search_bar.bind("<KeyRelease>", lambda event, s=search_bar: self.
                update_algorithm_list(s.get(), user_interface))
134
135     def return_to_main_page(self):
136         """Retourne à la page principale."""
137         for widget in self.root.winfo_children():
138             widget.destroy()
139         self.root.deiconify()
140         new_app = App(self.root)
141
```

```
142     def update_algorithm_list(self, search_text, user_interface):
143         """Met à jour la liste des algorithmes filtrés dans MyFrame."""
144         self.scrollable_frame.update_filtered_algorithms(search_text)
145 
146         # Réactivez la mise à jour après un court délai (10 ms)
147         user_interface.after(10, lambda: setattr(self.scrollable_frame, '
                update_needed', True))
148 
149     def admin_interface(self):
150         """Interface administrateur."""
151         self.root.withdraw()
152         admin_interface = ctk.CTkToplevel(self.root)
153         admin_interface.title("Admin Mode")
154         admin_interface.geometry("960x500")
155 
156         # Center the buttons vertically and horizontally
157         admin_interface.grid_rowconfigure(1, weight=1)
158         admin_interface.grid_columnconfigure(0, weight=2)
159         admin_interface.grid_columnconfigure(6, weight=1)
160 
161         # Change mode button
162         self.mode = ctk.CTkButton(admin_interface, text="MODE", width=100,
                command=self.return_to_main_page)
163         self.mode.grid(row=0, column=6, pady=10, padx=10, sticky="ne")  #
                Place the button in the top-right corner
164 
165         # Change password admin
166         self.password_change = ctk.CTkButton(admin_interface, text="Change
                Password", width=100,command=self.change_password )
167         self.password_change.grid(row=1, column=2, pady=10, padx=10)
168 
169         # Add an algo
170         self.add_algo = ctk.CTkButton(admin_interface, text="Add An Algorithm
                ", width=100, command=self.add_algorithm)
171         self.add_algo.grid(row=1, column=3, pady=10, padx=10)
172 
173         # Modify an algo
174         self.modify_algo = ctk.CTkButton(admin_interface, text="Modify an
                Algorithm", width=100, command=self.modify_algorithm)
175         self.modify_algo.grid(row=1, column=4, pady=10, padx=10)
176 
177         # Delete an algo
178         self.delete_algo = ctk.CTkButton(admin_interface, text="Delete an
                Algo", width=100, command=self.delete_algorithm)
179         self.delete_algo.grid(row=1, column=5, pady=10, padx=10)
180         ctk.set_appearance_mode("system")
181 
182     def change_password(self):
183         """Change le mot de passe."""
184         # Create a Toplevel window for changing password
185         password_window = ctk.CTkToplevel(self.root)
186         password_window.title("Change Password")
187         password_window.geometry("200x200")
188 
```

```
189         # Label and Entry for new password
190         label_password = ctk.CTkLabel(password_window, text="Enter new
                password:")
191         label_password.pack(pady=10)
192 
193         new_password_entry = ctk.CTkEntry(password_window, show="*")
194         new_password_entry.pack(pady=10)
195 
196         # Button to confirm password change
197         confirm_button = ctk.CTkButton(password_window, text="Change Password
                ", command=lambda: self.confirm_password_change(password_window,
                new_password_entry.get()))
198         confirm_button.pack(pady=10)
199         ctk.set_appearance_mode("dark")
200 
201     def confirm_password_change(self, password_window, new_password):
202         """Confirme le changement de mot de passe."""
203         if new_password:
204             # Update the password in the JSON file
205             self.update_password(new_password)
206             CTkMessagebox(title="Success", message="Password changed
                    successfully!")
207             password_window.destroy()
208         else:
209             CTkMessagebox(title="Warning", message="Please enter a new
                    password.")
210 
211     def add_algorithm(self):
212         """Ajoute un algorithme."""
213         # Create a new Toplevel window for adding an algorithm
214         add_algo_window = ctk.CTkToplevel(self.root)
215         add_algo_window.title("Add Algorithm")
216         add_algo_window.geometry("200x400")
217 
218         # Entry fields for algorithm name and content
219         ctk.CTkLabel(add_algo_window, text="Algorithm Name:").pack(pady=5)
220 
221         algo_name_entry = ctk.CTkEntry(add_algo_window)
222         algo_name_entry.pack(pady=10)
223 
224         ctk.CTkLabel(add_algo_window, text="Algorithm Content:").pack(pady=5,
                padx=10)
225         algo_content_entry = ctk.CTkTextbox(add_algo_window, height=200,
                width=180, wrap="word")  # Adjusted height
226         algo_content_entry.pack(pady=10)
227 
228         # Button to confirm the addition
229         confirm_button = ctk.CTkButton(add_algo_window, text="Add Algorithm",
                command=lambda: self.confirm_add_algorithm(add_algo_window,
                algo_name_entry.get(), algo_content_entry.get("1.0", "end-1c")))
230         confirm_button.pack(pady=10)
231         ctk.set_appearance_mode("dark")
232 
233     def confirm_add_algorithm(self, add_algo_window, name, content):
```

```
        """Confirme l'ajout de l'algorithme."""
        if name and content:
            # Load existing JSON data
            with open('algorithms.json', 'r') as json_file:
                data = json.load(json_file)

            # Add the new algorithm
            new_algo = {"name": name, "content": content}
            data["algorithms"].append(new_algo)

            # Write the updated data back to the JSON file
            with open('algorithms.json', 'w') as json_file:
                json.dump(data, json_file)

            CTkMessagebox(title="Success", message="Algorithm added
                successfully!")
            add_algo_window.destroy()
        else:
            CTkMessagebox.showwarning(title="Warning", message="Please enter
                both algorithm name and content.")

    def modify_algorithm(self):
        """Modifie un algorithme."""
        # Create a new window for algorithm modification
        modify = ctk.CTkToplevel(self.root)
        modify.title("Modify Algorithm")
        modify.geometry("500x500")

        # Load JSON from a file
        with open('algorithms.json', 'r') as file:
            self.all_algorithms = json.load(file)['algorithms']

        self.label2 = ctk.CTkLabel(modify, text="select your algorithm:",
            fg_color="transparent")
        self.label2.pack(pady=5)
        # Create a combobox to select the algorithm
        self.selected_algo_var = ctk.StringVar(value="")  # Use a StringVar
        self.selected_algo = ctk.CTkComboBox(modify, values=[algo['name'] for
            algo in self.all_algorithms], variable=self.selected_algo_var)
        self.selected_algo.pack(pady=10)

        # Modifier le contenu label
        self.label1 = ctk.CTkLabel(modify, text="Change the content of your
            algorithm:", fg_color="transparent")
        self.label1.pack(pady=5)

        # Create a Textbox to display the content of the selected algorithm
        self.content_textbox = ctk.CTkTextbox(modify, height=300, width=400,
            wrap="word")
        self.content_textbox.pack(pady=10)

        # Create a Submit button
        self.confirm = ctk.CTkButton(modify, text="Submit", command=self.
            submit_content)
```

```
281         self.confirm.pack(pady=10)
282 
283         # Link content update to the StringVar
284         self.selected_algo_var.trace_add("write", self.update_content)
285         self.root.withdraw()
286 
287     def update_content(self, *args):
288         """Update content based on selected algorithm."""
289         selected_algo_name = self.selected_algo_var.get()
290         selected_algo_content = next((algo["content"] for algo in self.
                all_algorithms if algo["name"] == selected_algo_name), "")
291         self.content_textbox.configure(state="normal")  # Enable read/write
                mode
292         self.content_textbox.delete("1.0", "end")  # Clear existing content
293         self.content_textbox.insert("1.0", selected_algo_content)
294 
295     def submit_content(self):
296         """Submit modified content."""
297         selected_algo_name = self.selected_algo_var.get()
298         new_content = self.content_textbox.get("1.0", "end-1c")  # Get
                content from TextBox
299 
300         # Update content of the selected algorithm
301         for algo in self.all_algorithms:
302             if algo["name"] == selected_algo_name:
303                 algo["content"] = new_content
304 
305         # Save changes to the JSON file
306         with open('algorithms.json', 'w') as file:
307             json.dump({"algorithms": self.all_algorithms}, file)
308 
309         # Display a confirmation message or perform other necessary actions
310         print("Modification enregistrée avec succès!")
311         CTkMessagebox(title="Success", message="Change saved successfully!")
312 
313     def delete_algorithm(self):
314         """Delete an algorithm."""
315         # Create a new window for algorithm deletion
316         delete_window = ctk.CTkToplevel(self.root)
317         delete_window.title("Delete Algorithm")
318         delete_window.geometry("300x150")
319 
320         # Load JSON from a file
321         with open('algorithms.json', 'r') as file:
322             self.all_algorithms = json.load(file)['algorithms']
323 
324         self.label1 = ctk.CTkLabel(delete_window, text="Choose the Algorith
                to delete:", fg_color="transparent")
325         self.label1.pack(pady=5)
326         # Create a combobox to select the algorithm to delete
327         selected_algo_var = ctk.StringVar(value="")
328         delete_algo_combobox = ctk.CTkComboBox(delete_window, values=[algo['
                name'] for algo in self.all_algorithms], variable=
                selected_algo_var)
```

```
329         delete_algo_combobox.pack(pady=10)
330
331         # Create a confirmation button to delete the algorithm
332         confirm_delete_button = ctk.CTkButton(delete_window, text="Delete",
                command=lambda: self.confirm_delete(selected_algo_var.get(),
                delete_window))
333         confirm_delete_button.pack(pady=10)
334
335     def confirm_delete(self, selected_algo_name, delete_window):
336         """Confirm deletion of the algorithm."""
337         if selected_algo_name:
338             # Remove the algorithm from the list
339             self.all_algorithms = [algo for algo in self.all_algorithms if
                    algo['name'] != selected_algo_name]
340
341             # Save changes to the JSON file
342             with open('algorithms.json', 'w') as file:
343                 json.dump({"algorithms": self.all_algorithms}, file)
344
345             # Display a confirmation message or perform other necessary
                    actions
346             print(f"Algorithm '{selected_algo_name}' supprimé avec succès!")
347
348             CTkMessagebox(title="Success", message="Algorithm deleted with
                    success!")
349             delete_window.destroy()
350         else:
351             CTkMessagebox.showwarning(title="Warning", message="Please select
                    an algorithm to delete.")
```

my_frame.py

```
1  import os
2  import json
3  import customtkinter as ctk
4
5  class MyFrame(ctk.CTkScrollableFrame):
6      """Classe représentant le cadre contenant les algorithmes."""
7
8      def __init__(self, master, textbox, **kwargs):
9          """Constructeur de la classe MyFrame."""
10         super().__init__(master, **kwargs)
11         self.textbox = textbox
12
13         # Get the directory of the current script
14         current_directory = os.path.dirname(os.path.abspath(__file__))
15
16         # Construct the full path to 'algorithms.json'
17         algorithms_file_path = os.path.join(current_directory, "algorithms.
               json")
18
19         # Ajouter des widgets dans le cadre...
20         with open(algorithms_file_path, "r") as file:
21             self.all_algorithms = json.load(file).get("algorithms", [])
22
```

```
23          self.filtered_algorithms = self.all_algorithms  # Initialisez avec
                tous les algorithmes
24
25          self.create_buttons()
26
27          self.grid_columnconfigure(0, weight=1)
28
29      def create_buttons(self):
30          """Crée les boutons en fonction des algorithmes filtrés."""
31          for row, algorithm in enumerate(self.filtered_algorithms, start=1):
32              algorithm_name = algorithm.get("name", "")
33              algorithm_button = ctk.CTkButton(self, text=algorithm_name)
34              algorithm_button.grid(row=row, column=0, pady=5, sticky="ew")
35
36              algorithm_button.configure(command=lambda algo=algorithm: self.
                    show_algorithm(algo))
37
38      def update_filtered_algorithms(self, search_text):
39          """Met à jour la liste des algorithmes filtrés en fonction de la
                chaîne de recherche."""
40          if search_text:
41              self.filtered_algorithms = [algo for algo in self.all_algorithms
                    if search_text.lower() in algo.get("name", "").lower()]
42          else:
43              self.filtered_algorithms = self.all_algorithms
44
45          # Supprimez tous les boutons actuels
46          for widget in self.winfo_children():
47              widget.destroy()
48
49          # Recréez les boutons avec les algorithmes filtrés
50          self.create_buttons()
51
52      def show_algorithm(self, algorithm):
53          """Mettez à jour le CTkTextbox avec le contenu de l'algorithme
                sélectionné."""
54          algorithm_content = algorithm.get("content", "")
55          self.textbox.configure(state="normal")  # Activez le mode lecture/
                écriture
56          self.textbox.delete("1.0", "end")  # Effacer le contenu existant
57          self.textbox.insert("1.0", algorithm_content)
58          self.textbox.configure(state="disabled")  # Désactivez le mode
                écriture après l'affichage
```