

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ТЕОРЕТИЧЕСКИЙ ОБЗОР	4
1.1 Актуальность и практическая значимость	4
1.2 Обязательные требования к проекту	4
1.2.1 Оформление по PEP 8	4
1.2.2 Принципа DRY.....	5
1.2.3 Читаемые сообщения коммита, отображающих суть изменений.....	5
1.2.4 Система документирования исходного кода.....	5
1.2.5 Файла README.md wiki-страницы.....	5
1.2.6 Использование функций, классов, наследования.....	5
1.3 Обзор используемых инструментов.....	6
1.3.1 Библиотека Python Flask	6
1.3.2 Библиотека Python Matplotlib	6
1.3.3 Библиотека JavaScript CodeMirror.....	6
2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ	7
2.1 Применение политики PEP 8 в коде	7
2.2 Применение DRY-политики	7
2.3 Работать вместе и успешно модифицировать код.....	8
2.4 Использование системы документирования исходного кода.....	8
2.5 Внешний вид файла README.md	9
2.6 Wiki-страница с документацией и описанием программы	9
2.7 Использование функций	10
2.8 Использование классов	11
2.9 Использование наследования	11
2.10 Оригинальность темы.....	12
2.11 Полнота реализации	12
ЗАКЛЮЧЕНИЕ	14

ВВЕДЕНИЕ

Этот проект направлен на разработку и настройку сложной системы для *создания геометрических фигур и управления ими в системе координат*. Система позволяет пользователям создавать и изменять точки, линии, многоугольники и окружности, а также выполнять различные операции с этими геометрическими элементами. Кроме того, приложение позволяет пользователям загружать сгенерированные изображения для личного использования.

Вдохновением для этого проекта послужили требования *Лаборатории 7* (Введение в использование **Matplotlib** для построения графиков) и *Лаборатории 9* (Введение в использование **Flask** для разработки веб-приложений). В этом проекте мы интегрируем **Matplotlib** и **Flask** для создания веб-приложения, способного генерировать определяемую пользователем систему координат.

Для выполнения этого проекта наша команда была разделена на три отдельные роли:

1. В рамках первой роли, порученной Дау Минь Шы, он выполняет задачу разработки веб-приложения с использованием популярного фреймворка Flask, что требует от него профессиональных знаний и опыта работы с данной технологией. Он также отвечает за выполнение внутренних задач, которые могут быть связаны с поддержкой и оптимизацией работы веб-приложения, а также с поддержанием функциональности в соответствии с требованиями проекта.
2. В рамках второй роли, отведенной Нгуен Тхань Чунг, ему доверено выполнение задач, связанных с созданием качественных шаблонов для веб-приложения. Кроме того, он ответственен за обработку интерфейсных аспектов приложения, включая создание удобной навигации и реализацию функциональности, которая позволяет пользователям легко и быстро находить нужную информацию.
3. В рамках третьей роли в проекте Нгуен Кхань Нгок выполняет задачу подготовки сложных документов, которые являются необходимыми для успешной реализации проекта. В рамках данной задачи она берет на себя ответственность за тщательный анализ и подробное изучение требований проекта, выявление всех необходимых документов и формализацию их содержания в соответствии с установленными стандартами и требованиями.

1 ТЕОРЕТИЧЕСКИЙ ОБЗОР

1.1 Актуальность и практическая значимость

В течение нашего первого года обучения мы часто сталкивались с необходимостью построения системы координат по различным предметам, включая математику, физику и линейную алгебру, среди прочих. Традиционный подход создания системы координат вручную с помощью ручки и бумаги требует много времени и неэффективен. Следовательно, мы решили разработать веб-приложение, чтобы облегчить нам этот процесс.

1.2 Обязательные требования к проекту

1.2.1 Оформление по PEP 8

PEP 8 - это руководство по написанию кода на языке программирования Python, которое определяет стандарты и рекомендации для оформления кода, структуры программы, именования переменных и т.д. Соблюдение этих стандартов может улучшить читаемость кода, сделать его более понятным и снизить вероятность ошибок. PEP8 является частью серии рекомендаций **Python Enhancement Proposals** (PEP), созданных для развития языка Python.

Соответствие стандарту PEP 8 сделает исходный код Python более читабельным и удобным для сопровождения. Некоторые правила форматирования исходного кода Python в соответствии со стандартами PEP 8:

- Используйте 4 пробела для каждого уровня отступа.
- Используйте пустые строки для разделения разных частей исходного кода, например, между функциями, между методами класса, между импортами и т. д.
- Используйте символы подчеркивания для разделения слов в именах переменных и функций, например *my_variable*.
- Используйте осмысленные и понятные имена переменных и функций и избегайте коротких или запутанных имен.
- Используйте правила верхнего и нижнего регистра, например, используйте строчные буквы для имен переменных и прописные буквы для имен констант.
- Используйте одинарные кавычки для строк, если только строка не содержит круглых скобок.
- И еще много правил.

1.2.2 Принципа DRY

Принцип DRY (*Don't Repeat Yourself*) - это принцип разработки программного обеспечения, который заключается в том, чтобы избегать повторения кода в одном и том же проекте.

- Использовать функции и классы для повторяющихся участков кода.
- Использовать переменные для повторяющихся значений.
- Использовать модули и библиотеки для общих функций и классов.
- Использовать шаблоны проектирования.
- Проверять код на повторения. Иногда повторения кода могут быть.

Соблюдение принципа DRY сложным, но может привести к более эффективному и понятному коду, который будет легче поддерживать и тестировать в будущем.

1.2.3 Читаемые сообщения коммита, отображающих суть изменений

Читаемые сообщения коммита - это важный аспект разработки программного обеспечения, который позволяет упростить процесс совместной работы и улучшить понимание изменений, внесенных в код.

1.2.4 Система документирования исходного кода

Система документирования исходного кода - это набор инструментов и методов, которые используются для создания документации, описывающей функции, классы и модули в исходном коде.

1.2.5 Файла README.md wiki-страницы

Файл README.md - это файл, который содержит информацию о проекте и инструкции по его использованию.

Wiki-страницы - это веб-страницы, которые содержат документацию, инструкции и другую информацию о проекте. Они часто используются для документирования проектов в системах управления проектами, таких как GitHub.

1.2.6 Использование функций, классов, наследования

В разработке программ не обойтись без использования функций, классов и наследования. Это основные инструменты для организации и структурирования кода.

Все эти инструменты вместе позволяют структурировать код программы, разбив его на логические блоки - функции и классы, и организовывать иерархию классов с наследованием. Это делает код:

- Модульным - разные части можно разрабатывать и исправлять независимо;
- Легко расширяемым - новые классы могут наследоваться от существующих;
- Легко тестируемым - отдельные функции и классы проще тестировать;
- Легче понимаемым - структурированный код проще читать и понимать.

1.3 Обзор используемых инструментов

1.3.1 Библиотека Python Flask

Flask - это легковесный фреймворк для создания веб-приложений на языке программирования Python. Он предоставляет минимальный набор инструментов для создания веб-приложений, включая маршрутизацию URL, шаблонизацию, обработку форм и поддержку HTTP-запросов. Flask также имеет обширную документацию и сообщество, что делает его очень доступным и популярным для начинающих и опытных разработчиков.

В этом проекте мы используем Flask в качестве веб-сервера. Он получит запрос от пользователя и вернет файл изображения png, который представляет собой построенный график. Кроме того, он также берет на себя роль создания html-страниц (рендеринг сервера).

1.3.2 Библиотека Python Matplotlib

Matplotlib - это библиотека для построения графиков на языке программирования Python. Она предоставляет широкий спектр инструментов для создания различных типов графиков, включая линейные, столбчатые, круговые, точечные и т.д.

В этом проекте Matplotlib используется для создания системы координат и сохранения ее в виде файла png. Затем эти изображения будут возвращены пользователю сервером Flask по запросу.

1.3.3 Библиотека JavaScript CodeMirror

CodeMirror - это библиотека JavaScript для создания интерактивных редакторов кода в веб-браузере. CodeMirror используется во многих web-приложениях, IDE и онлайн-редакторах кода, таких как GitHub, Bitbucket, CodePen и других.

В этом проекте CodeMirror используется как пользовательская библиотека JavaScript. Это редактор, так что пользователи могут легко добавлять точки, сегменты, многоугольники, круги, а также поддерживает некоторые другие функции редактора, такие как отмена, повтор, ...

2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

2.1 Применение политики PEP 8 в коде

Применение политики **PEP 8** в коде может быть достигнуто несколькими способами. В этом проекте мы использовали библиотеку **autopep8**. Библиотека **autopep8** предоставляет опции для автоматического изменения кода в соответствии с интервалами **PEP 8**, именами переменных, пробелами и другими правилами.

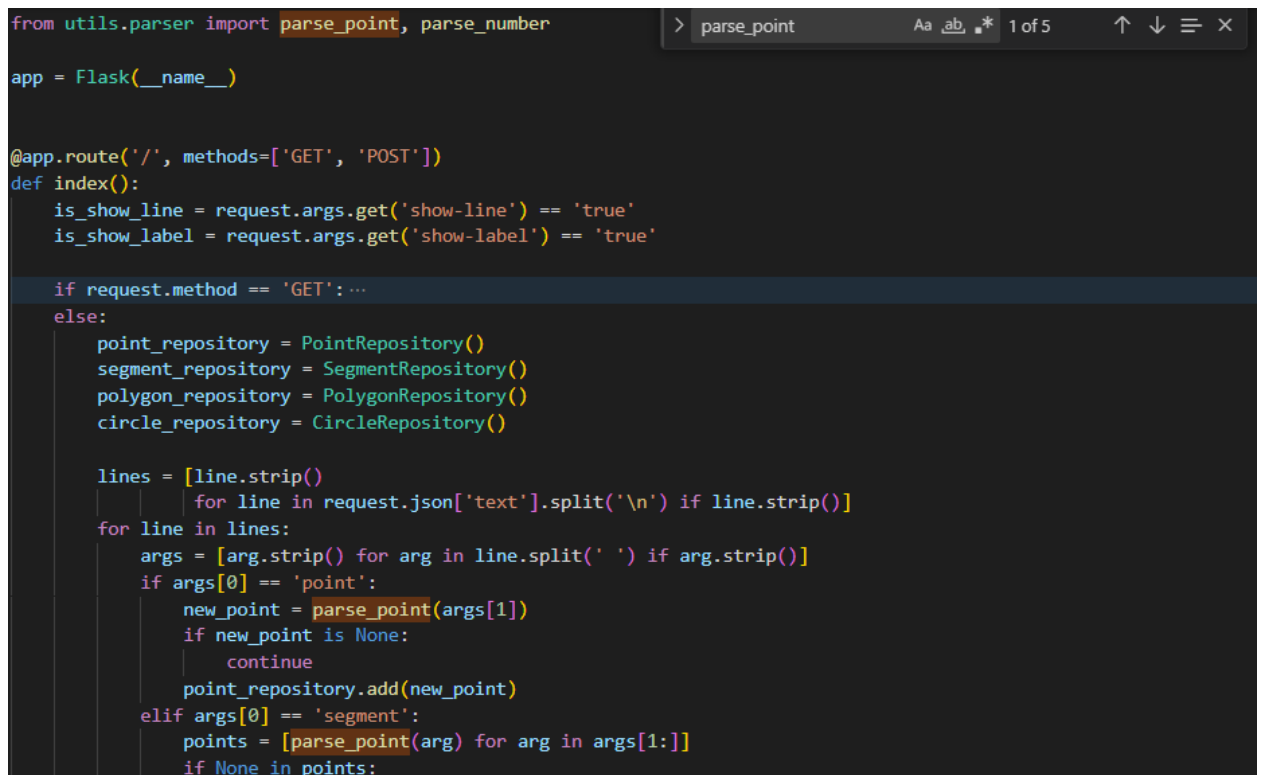
Например, следующий пример сценария:

```
autopep8 --in-place --aggressive --aggressive <filename>
```

2.2 Применение DRY-политики

Применение **DRY-политики** (*Don't Repeat Yourself*) заключается в том, чтобы избегать дублирования кода в программном проекте. Это означает, что каждая часть функциональности должна быть реализована только один раз, а затем использоваться повторно в других местах, где это необходимо. Мы использовали функции и классы для реализации политики **DRY**.

Например, мы используем функцию **parse_point** (хранится в файле *utils/parser.py*) или создаем классы **Repository**, **Shape**, и т. д. и применяем наследование, чтобы помочь другому классу повторно использовать метод своего предка.



```
from utils.parser import parse_point, parse_number

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    is_show_line = request.args.get('show-line') == 'true'
    is_show_label = request.args.get('show-label') == 'true'

    if request.method == 'GET': ...
    else:
        point_repository = PointRepository()
        segment_repository = SegmentRepository()
        polygon_repository = PolygonRepository()
        circle_repository = CircleRepository()

        lines = [line.strip()
                  for line in request.json['text'].split('\n') if line.strip()]
        for line in lines:
            args = [arg.strip() for arg in line.split(' ') if arg.strip()]
            if args[0] == 'point':
                new_point = parse_point(args[1])
                if new_point is None:
                    continue
                point_repository.add(new_point)
            elif args[0] == 'segment':
                points = [parse_point(arg) for arg in args[1:]]
                if None in points:
```

Рисунок 1: Повторное использование функции *parse_point* в файле *app.py*.

2.3 Работать вместе и успешно модифицировать код

Мы вместе делали проект, помогали друг другу в трудную минуту и закончили его. Каждое вносимое нами изменение имеет определенное значение, никакое изменение не считается излишним.

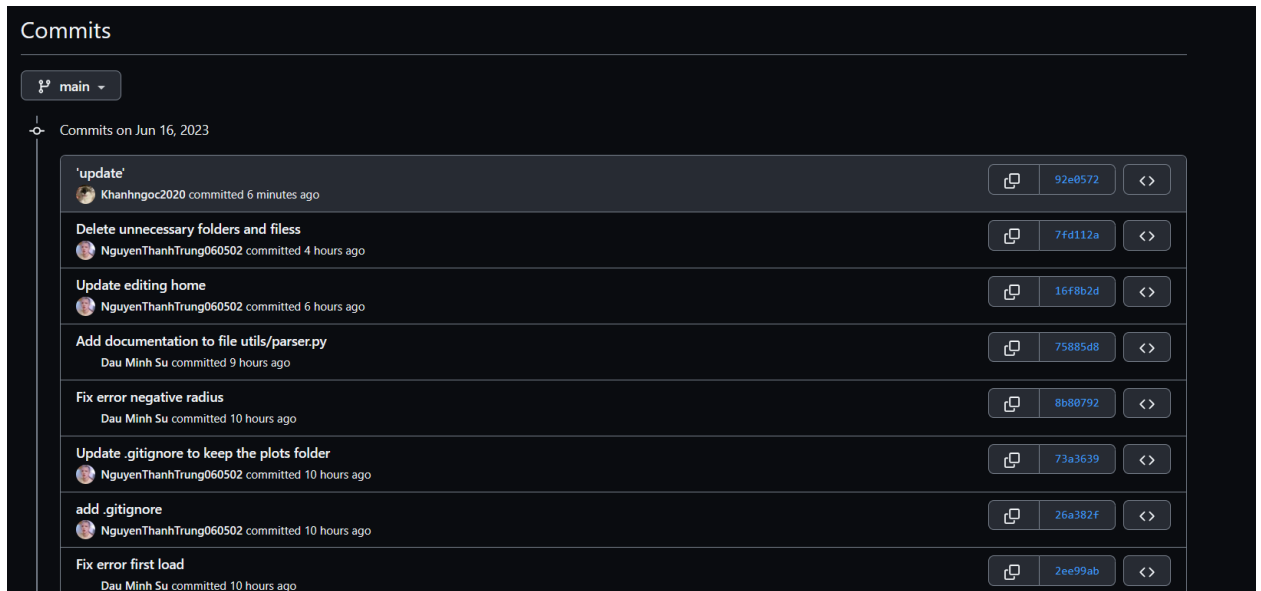


Рисунок 2: Список коммитов, созданных членами команды.

2.4 Использование системы документирования исходного кода

В ходе выполнения проекта мы применили библиотеку reStructuredText для форматирования кода. Представляем два изображения, которые иллюстрируют использование документации исходного кода.

Первое изображение демонстрирует пример документации, встраиваемой в функцию `__init__` класса `Segment`, а второе изображение иллюстрирует документацию в функции абстракции `get_x` абстрактного класса `Shape`.

```
class Segment(Shape):
    def __init__(self, start, end):
        """
        Создание сегмента с 2 конечными точками
        :param start: первая конечная точка - Point
        :param end: вторая конечная точка - Point
        """
        self.start = start
        self.end = end
```

Рисунок 3: Документация функции `__init__` в классе `Segment`.

```
class Shape(ABC):
    @abstractmethod
    def get_x(self):
        """
        Получить массив координат x для всех вершин
        """
        pass
```

Рисунок 4: Документация абстрактной функции `__get_x__` в классе `Shape`.

2.5 Внешний вид файла README.md

Предоставляется возможность получить доступ к файлу **README.md** посредством перехода по данной [ссылке](#).

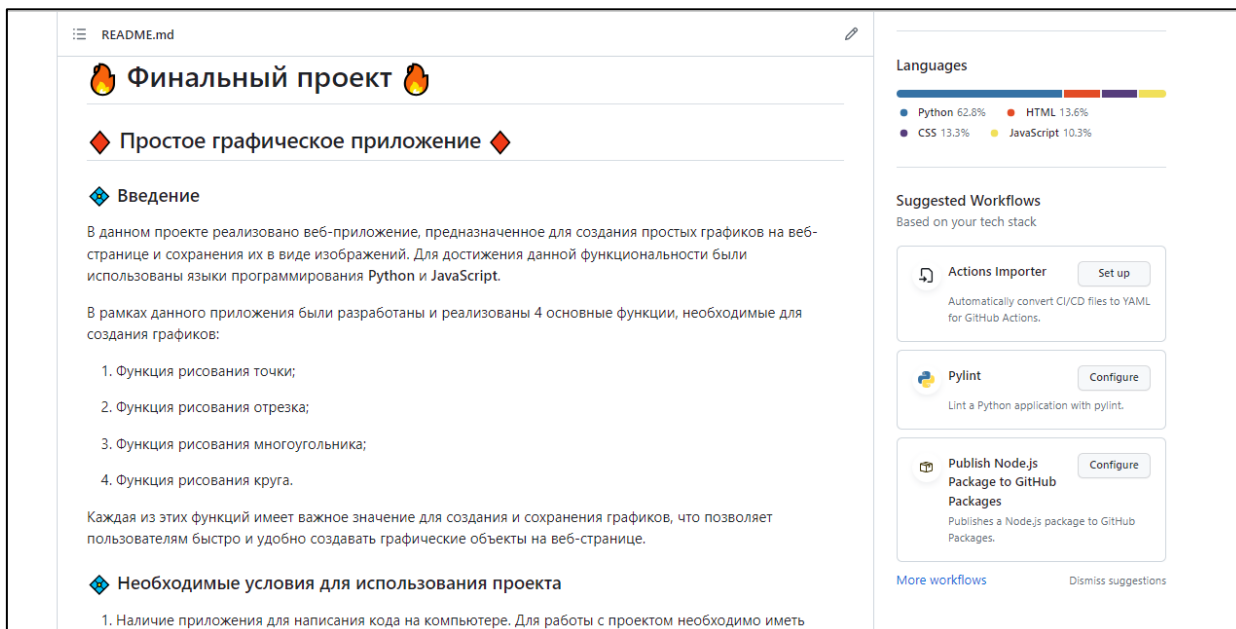


Рисунок 5: Предварительный просмотр **README.md** на *GitHub*.

2.6 Wiki-страница с документацией и описанием программы

Предоставляется возможность получить доступ к файлу **wiki-страница** посредством перехода по данной [ссылке](#).

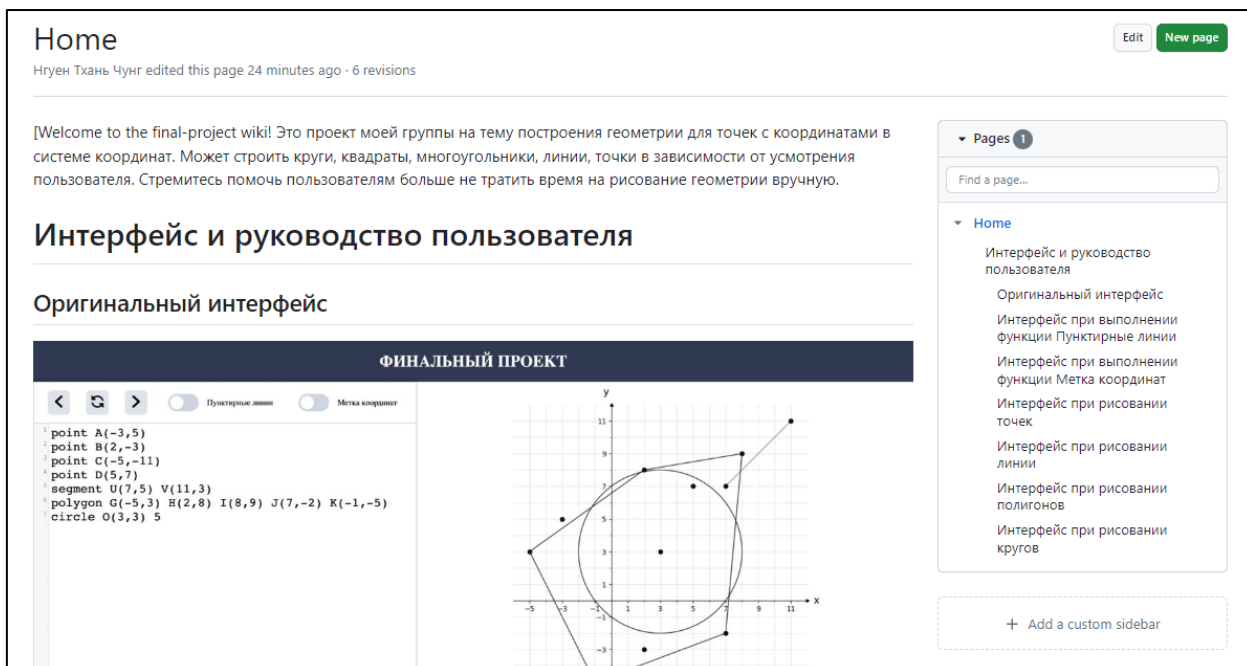


Рисунок 6: Предварительный просмотр **wiki-страница** на *GitHub*.

2.7 Использование функций

В рамках выполнения проекта мы применили множество функций с целью предотвращения дублирования кода и соблюдения принципа **DRY**. В предыдущей части мы продемонстрировали использование функции **parse_point** для сокращения кода и увеличения скорости написания кода.

Приводим несколько дополнительных примеров использования функции в нашем проекте. Первый пример относится к функции **get_min_max**, которая используется для получения диапазона значений в списке (хранится в файле *utils/graph-creator.py*). Благодаря использованию данной функции мы теперь можем легко построить график. Другой пример представляет функцию **parse_number** (хранящуюся в файле *utils/parser.py*), которая преобразует строковое значение в числовое с плавающей точкой и является вспомогательной утилитой для функции **parse_point**.

```
def get_min_max(repository):
    """
    Получить максимальное значение, минимальное значение и их разницу в списке
    :param repository: список для продолжения
    :return: максимальное значение, минимальное значение и их разница
    """
    val_min, val_max = -10, 10
    if len(repository) == 1:
        val_min = min(repository) - 5
        val_max = max(repository) + 5
    elif len(repository) > 0:
        val_min = min(repository)
        val_max = max(repository)
    delta = val_max - val_min
    return val_min, val_max, delta
```

Рисунок 7: Функции **get_min_max** в файле *utils/graph-creator.py*.

```
def parse_point(raw):
    """
    Разобрать точку (Point) из строки
    :param raw: необработанная строка должна быть проанализирована
    :return: вернуть точку (Point), если его можно проанализировать, иначе вернуть None
    """
    args = [arg.strip() for arg in re.split(r'[\s,()]+', raw) if arg.strip()]

    if len(args) != 3:
        return None
    if parse_number(args[1]) is None:
        return None
    if parse_number(args[2]) is None:
        return None

    return Point(args[1], args[2], args[0])
```

Рисунок 8: Функции **parse_point** в файле *utils/parser.py*.

2.8 Использование классов

В нашем проекте находятся две ключевые директории: «**model**» и «**repository**». В этих директориях содержатся файлы, которые используют классы. Размещение классов в соответствующих директориях, где они могут наиболее эффективно выполнять свои функции и имеют наибольший смысл, может значительно улучшить организацию кода и снизить его сложность.

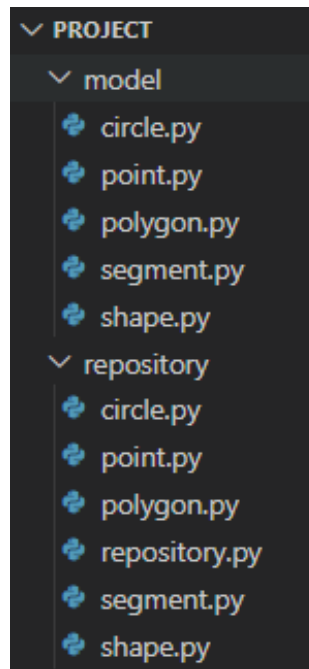


Рисунок 9: Список используемых классов в проекте.

2.9 Использование наследования

В отчете рассматривается древовидная диаграмма, отображающая файлы и папки в каталоге «**repository**». Эти лишние элементы могут усложнить структуру проекта и увеличить время разработки. Предлагается упростить каталог, удалив лишние файлы и папки, не влияющие на функциональность и качество проекта. Это повысит эффективность команды и ускорит разработку, а также сократит объем кода и улучшит читаемость.

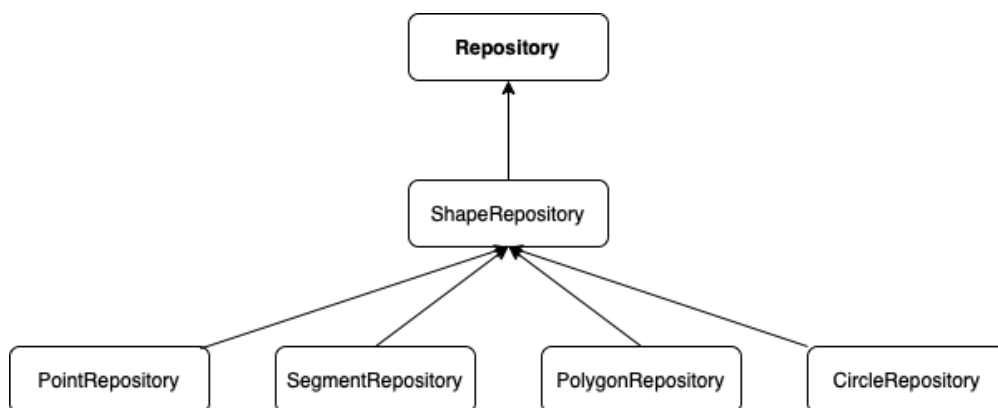


Рисунок 10: Дерево наследования в директории «**repository**».

Нижеприведенная древовидная диаграмма является иллюстрацией избыточности в каталоге «**model**»: (Та же функция, что и на схеме выше - «**repository**».)

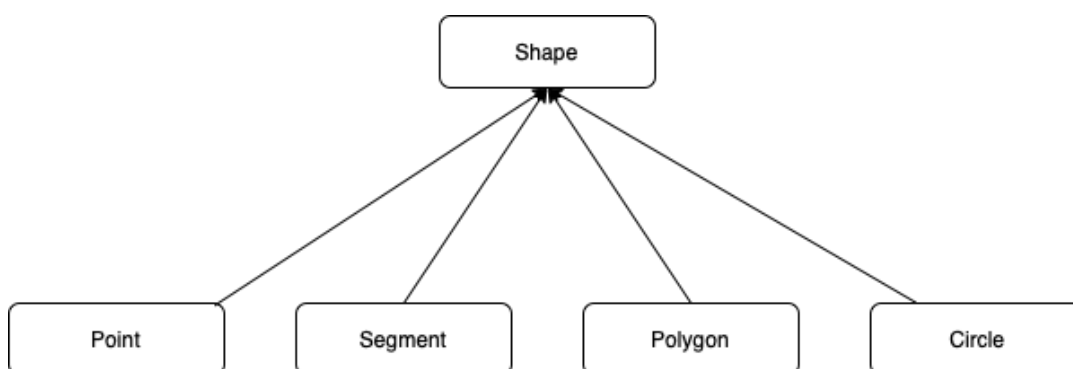


Рисунок 11: Дерево наследования в директории «**model**».

2.10 Оригинальность темы

Концепция данного проекта основана на двух предыдущих лабораторных работах, которые были выполнены ранее в контексте изучения программирования:

- *Лабораторная работа 7:* Рисование графиков при помощи Matplotlib – библиотеки для создания качественных и информативных графиков;
- *Лабораторная работа 9:* Создание веб-приложения с использованием Flask – мощного и гибкого инструмента для создания веб-приложений.

Мы решили выбрать тему, исходя из нашего опыта необходимости частого построения систем координат. С преимуществами, которые предоставляет пользователям графическое приложение:

- Во-первых, это помогает пользователям лучше понять данные, представляя их в виде графика, что повышает способность понимать и анализировать данные.
- Кроме того, графическое приложение также помогает пользователям легко сравнивать и визуализировать статистические данные, упрощая анализ и принятие решений.
- Они помогают представить результаты анализа в более понятной и убедительной форме.
- Короче говоря, приложение для построения графиков — это полезный и необходимый инструмент для разных областей, помогающий пользователям легко понимать и анализировать данные, визуализировать результаты исследований и принимать респектабельные решения.

2.11 Полнота реализации

В рамках данного проекта было создано веб-приложение, которое позволяет пользователям создавать графики, как описано в разделе «Введение». Это приложение обладает интуитивно понятным и удобным интерфейсом, который позволяет пользователям быстро овладеть его функциональностью.

Создание данного веб-приложения предоставляет пользователям возможность создавать графику на веб-странице быстро и легко. Благодаря этому приложению, пользователи могут создавать различные простые рисунки, такие как точки, линии, многоугольники и круги, которые в дальнейшем могут быть сохранены в виде изображений.

Веб-приложение, помимо своих основных функций, также предоставляет пользователям набор вспомогательных функций, которые облегчают процесс работы с приложением и повышают его функциональность. Ниже приведена таблица с перечислением указанных вспомогательных функций:

Функция	Описание
	Отменить изменение в редакторе CodeMirror.
	Сбросить редактор CodeMirror до значения по умолчанию.
	Повторить отмену в редакторе CodeMirror.
 Пунктирные линии	Провести пунктирной линии к каждой точке.
 Метка координат	Показать метки точек.
	Обновлять построенное изображение по значению в CodeMirror.
	Очистить все в CodeMirror.
	Скачать построенное изображение.

Удовлетворить всем требованиям, которые необходимы проекту

Требование	Реализация
Оформление по PEP 8 (для других языков аккуратность)	Использовали библиотеку autoper8
Соблюдение принципа DRY	Использовали классы, наследование и функции
Наличие коммитов от всех участников проекта	Все члены имеют коммиты
Наличие читаемых сообщений коммита, отображающих суть изменений	Сообщение коммиты четко написаны
Использование системы документирования исходного кода	Использовали reStructuredText
Наличие файла README.md с коротким описанием программы	Есть файл README.md
Наличие wiki-страницы с документацией и описанием программы	Есть wiki-page
Использование функций	Использовали
Использование классов	Использовали
Использование наследования	Использовали
Оригинальность темы	Новые темы и идеи
Полнота реализации	Готовое приложение

ЗАКЛЮЧЕНИЕ

Мы рады, что завершили этот проект и достигли наших первоначальных целей. На нашем простом графическом веб-сайте есть все необходимые функции, в том числе графические функции, графическая геометрия и хранение графиков. Это позволяет пользователям легко создавать и хранить диаграммы и графики для различных целей.

За время работы мы столкнулись со многими проблемами и трудностями. Однако нам удалось их решить и успешно завершить проект. Одной из самых больших проблем является изучение таких библиотек, как: Flask, Matplotlib, CodeMirror, а затем их объединение для создания полноценного и качественного продукта.

Мы также узнали много других навыков по пути:

- Знать, как применять и использовать библиотеки Python наиболее оптимальным образом.
- Кроме того, мы также научились писать документацию, wiki-страницу и README.md, чтобы иметь возможность представить пользователям обзор и доступный обзор возможностей и функциональности нашего продукта.
- Важно работать в команде, разделять работу на каждого члена команды, координировать друг с другом, чтобы иметь возможность завершить проект максимально полно и качественно.

В будущем мы планируем предложить добавить новые функции в наше приложение, чтобы оно стало еще более полнофункциональным, например, некоторые из функций могут быть следующими:

- Для отрезка может быть задана его длина.
- Для полигонов:
 - Может вычислять периметр и площадь многоугольников.
 - Определить тип многоугольника: прямоугольный треугольник, равнобедренный треугольник, прямоугольник, квадрат, пятиугольник, шестиугольник,
- Для кругов:
 - Можно нарисовать больше треугольников, вписанных в круги, треугольников, описанных в кругах.
 - Создать касательную к окружности в точке.
- И так далее...
- Кроме того, мы также хотим оптимизировать веб-сайт, чтобы увеличить скорость загрузки страниц и сократить время отклика веб-сайта.

Таким образом, мы очень довольны результатами этого проекта и считаем, что он принесет большую пользу пользователям. Мы приобрели много новых навыков и будем продолжать разрабатывать продукты для удовлетворения растущих потребностей наших пользователей.