РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ ГЕНЕРАЦИИ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ STABLE DIFFUSION

СТУДЕНТ: ВАСИЛЬЕВ ВЯЧЕСЛАВ

ГАБЫШЕВ ВЛАДИМИР

ДМИТРИЕВА ВАСИЛИНА

ЗЛОБИНА СОФЬЯ

РАЧЕЕВ НИКИТА

ОГОРОДОВ ДАНИЛ

ГРУППА: КИСП 9-22(1, 2) ГР

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc199538326)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc199538327)

[1.1 ТЕРМИНОЛОГИЯ ПО ПРОЕКТУ И ГЛОССАРИЙ 5](#_Toc199538328)

[1.2 РАСПРЕДЕЛЕНИЕ РОЛЕЙ 7](#_Toc199538329)

[1.3 СТЕК ТЕХНОЛОГИЙ 9](#_Toc199538330)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 12](#_Toc199538331)

[2.1 АРХИТЕКТУРА ПО 12](#_Toc199538332)

[2.2 РАЗРАБОТКА ПРОЕКТА ПО РОЛЯМ 12](#_Toc199538333)

[2.3 КОНТРОЛЬ ВЫПОЛНЕНИЯ ПЛАНА 19](#_Toc199538334)

[ЗАКЛЮЧЕНИЕ 20](#_Toc199538335)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ. 21](#_Toc199538336)

# ВВЕДЕНИЕ

В последние годы искусственный интеллект (ИИ) достиг значительных успехов, в особенности в области генерации изображений. Модели, такие как Stable Diffusion, позволяют создавать высококачественные изображения на основе текстовых описаний.

Эти достижения открывают новые возможности для художников, дизайнеров и всех, кто интересуется созданием визуального контента.

Целью данного отчета является описание процесса разработки десктопного приложения, которое позволяет пользователям генерировать изображения с помощью модели Stable Deffusion. В отчете рассматривается актуальность проекта, предмет и объект исследования, проводится теоретический анализ, описывается архитектура приложения и распределение ролей при разработке.

Актуальность проекта:

1. Доступность и Удобство:

Локальное использование это обеспечивает:

Более высокую скорость, конфиденциальность, независимость от интернета, простой интерфейс, гибкость настройки.

1. Повышение производительности и функциональности: Использование ресурсов ПК, интеграция с локальными файлами, возможность расширения, индивидуальная настройка
2. Развитие и Популяризация Stable Diffusion: Снижение барьера входа, развитие сообщества, образовательный потенциал
3. 4. Коммерческий Потенциал: Создание контента, разработка игр, индивидуальное творчество

Предмет и объект исследования:

Предмет исследования: Процесс разработки десктопного приложения для генерации изображений с использованием модели Stable Diffusion. Объект исследования: Структура, функциональности и производительности десктопного приложения для генерации изображений на основе Stable Diffusion.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# 1.1 ТЕРМИНОЛОГИЯ ПО ПРОЕКТУ И ГЛОССАРИЙ

Stable Diffusion — это продвинутая модель генерации изображений, основанная на принципах диффузионных моделей. Она позволяет создавать высококачественные изображения из случайного шума, проходя через ряд промежуточных состояний. Данный процесс включает как этап кодификации, так и этап декодирования, каждый из которых играет критически важную роль в отношении к качеству и достоверности генерируемых изображений.

Десктопные приложение – это программное обеспечение, которое работает из-под операционной системы компьютера пользователя. К ним относится большинство приложений, устанавливаемых на персональный компьютер. Такие приложения могут работать без подключение интернету, они более производительные.

Генерация изображений – процесс создания изображений с помощью алгоритмов и моделей машинного обучений, основанных на обработке и генерации визуальной информации.

Интерфейс пользователя(UI) – это все, что помогает людям управлять устройствами и программами с помощью голоса, нажатий, жестов, через командную строку.

Пользовательский опыт(UX) – общие впечатления пользователя от взаимодействия с приложением, включающие в себя удобство и эффективность использования.

Машинное обучение(ML) – это область искусственного интеллекта(AI), занимающаяся разработкой алгоритмов и моделей, которые способны обучаться, используя данные, составлять прогнозы, а также принимать решения без программирования.

Набор данных – коллекция данных, используемая для обучения и тестирования модели, в данном случае набор изображений и их текстовых описаний.

Текстовое описание – краткое описание, на основе которой модель генерирует изображение.

API – набор определенных правил стандартов, позволяющий различным программам взаимодействовать друг с другом. В контексте генерации изображений может использоваться для доступа к внешним библиотекам или сервисам.

Библиотеки – наборы инструментов и функций, которые упрощают разработку и позволяют использовать готовые решения, например, библиотеки для работы с нейронными сетями.

Фреймворк (Framework) в программировании - это готовая структура, набор инструментов, библиотек и правил, которые упрощают и ускоряют разработку программного обеспечения.

Python - это универсальный, высокоуровневый язык программирования, известный своей читаемостью и простотой использования. Он стал одним из самых популярных языков в мире благодаря своей гибкости и широкому спектру применения.

Бэкенд разработка – это серверная часть – это невидимая для пользователя часть, отвечает за логику приложения, безопасность, взаимодействие с базой данных.

Фротэнд разработка – это клиенсткая часть, все что видит пользователь и с чем взаимодействует. Отвечает за визуальное представление и пользовательский опыт.

Стек технологий – это набор инструментов, технологий, языков программирования, фреймворков, библиотек, баз данных и другого программного обеспечения, которые используется для создания и запуска конкретного программного решения. Он определяет всю инфраструктуру, необходимую для разработки, развертывания и поддержки приложения.

Архитектура ПО – это фундаментальная организация программной системы, включающая в себя ее компоненты, отношения между ними, окружение и принципы, которыми руководствуется ее проектирование и эволюция. Это как план здания, определяющий, где будут расположены комнаты, как они будут связаны между собой, и какие материалы будут использованы. Архитектура определяет высокоуровневую структуру системы, в то время как детали реализации остаются за рамками архитектурного описания.

USE CASE – это текстовое описание последовательности действий, выполняемых пользователем для достижения определенной цели при взаимодействии с программой системой.

# 1.2 РАСПРЕДЕЛЕНИЕ РОЛЕЙ

Этот раздел посвящен описанию распределения ролей и ответственности между участниками проекта.

Габышев Владимир – отвечает за бэкенд-разработку.

Выполнил:

* Загрузка инициализация модели Stable Diffusion (). StableDiffusionPipeline;
* Разработка функции: generate (prompt, resolution, steps);
* Обработка ввода пользователя (проверка на пустой запрос);
* Извлечение параметров разрешения;
* Вызов модели Stable Diffusion для генерации изображения;
* Изменение размера изображения;
* Генерация уникального имени файла и сохранение изображения в каталоге истории;
* Создание списка изображений для отображения в галерее;
* Разработка механизма обработки ошибок и отображения сообщений об ошибках;
* Оптимизация процесса генерации;
* Управление хранением истории изображений.

Васильев Вячеслав – Руководитель проекта и основной разработчик интерфейса ( GUI).

Выполнил:

* Разработка структуры пользовательского интерфейса (gradio blocks, rows, columns);
* Создание и стилизация всех элементов интерфейса:
* Текстовое поле для ввода описания (). prompt;
* Выпадающий список для выбора разрешения (). resolution
* Слайдер для выбора количества шагов (). steps;
* Кнопка «Сгенерировать» (). generate\_btn;
* Текстовое поле для отображения статуса (). status;
* Элемент для отображения статуса (). output
* Галерея для отображения истории изображений (). gallery

Дмитриева Василина – Технический писатель. Отвечает за документацию.

Выполнила:

* Подготовка полного отчета по проекту;
* Описание архитектуры системы;
* Составление руководства пользователя.

Злобина Софья – тестировщик.

Выполнила:

* Тестирование приложения:
* Проверка корректности работы всех функций (генерации изображений, выбор разрешения, изменения количество шагов, отображение истории);
* Тестирование обработки ошибок;
* Проверка производительности.

Рачеев Никита – ответственный за вспомогательные функции, управление ресурсами и обработку исключений.

Выполнил:

* Разработка вспомогательных функций для преобразования данных, работы с файлами и т.д.
* Разработка функций для управления памятью;
* Обработка исключений и ошибок в функции. generate;
* Обеспечение корректного освобождения ресурсов (например, закрытие соединений, удаление временных файлов).
* Валидация вводимых параметров (например, проверка формата разрешения).

Огородов Данил – отвечает за фронтенд разработку.

Выполнил:

* Разработка кастомного CSS для стилизации интерфейса (темная тема, стили кнопок, шрифты);
* Настройка размещения элементов (использование и для макета). Column Row;
* Организация взаимодействия между элементами интерфейса и логикой (обработка событий нажатия кнопок с помощью). generate\_btn. click().
* Работа с отображением: отображение сгенерированного изображения, отображение истории изображений.

# 1.3 СТЕК ТЕХНОЛОГИЙ

В этом разделе будет подробно описан выбранный стек технологий для разработки десктопного приложения. Будет обоснован выбор каждой технологии, ее преимущества и недостатки в контексте данного проекта. Языком программирования был выбран Python. Это один из самых популярных языков для работы с моделями машинного обучения и генеративными алгоритмами. Также имеется множество библиотек для работы с графическими интерфейсами и API.

Язык программирование:

Python – это язык программирования. Это высокоуровневый, интерпретируемый язык, который отличается простым синтаксисом.

Фреймворки и библиотеки:

Gradio - это опенсорсная Python библиотека, предназначенная для быстрой и легкой демонстрации и отладки моделей машинного обучения (ML) через интерактивные веб-интерфейсы. Она позволяет разработчикам создавать удобные и доступные приложения для демонстрации возможностей их моделей, сбора обратной связи от пользователей и отладки проблем. Gradio позволяет взаимодействовать с вашими моделями машинного обучения прямо из браузера, без необходимости написания сложного веб-фронтенда.

PyTorch – это мощный и гибкий фреймворк машинного обучения с открытым исходным кодом, разработанный Facebook’s AI Research lab (FAIR). Он завоевал огромную популярность в академической и промышленной среде благодаря своей простоте использования, динамической вычислительной графе и обширной экосистеме инструментов и библиотек. В контексте diffusers, PyTorch играет ключевую роль, обеспечивая инфраструктуру для обучения, реализации и развертывания сложных нейронных сетей, лежащих в основе диффузионных моделей.

Diffusers – это библиотека с открытым исходным кодом, разработанная Hugging Face, предоставляющая готовые к использованию диффузионные модели (Diffusion Models) для задач генерации изображений и звука. Она позволяет исследователям и разработчикам быстро интегрировать и экспериментировать с различными диффузионными моделями, не требуя глубоких знаний внутренней работы этих моделей.

Pillow - это мощная и удобная в использовании библиотека Python, предназначенная для работы с изображениями. Она является форком оригинальной библиотеки Python Imaging Library (PIL), которая долгое время не поддерживалась. Pillow предоставляет широкий набор инструментов для открытия, манипулирования и сохранения изображений различных форматов.

Модуль OS в Python предоставляет широкий набор функций для взаимодействия с операционной системой, на которой выполняется скрипт. Он позволяет выполнять операции, которые обычно выполняются из командной строки или через графический интерфейс операционной системы. Этот модуль является частью стандартной библиотеки Python, поэтому не требует дополнительной установки.

Модуль uuid (Universally Unique Identifier) в Python предоставляет функциональность для генерации уникальных идентификаторов, известных также как UUIDs или GUIDs (Globally Unique Identifiers). Эти идентификаторы используются для однозначной идентификации объектов в распределенных системах, базах данных и других приложениях, где требуется глобальная уникальность. Модуль uuid является частью стандартной библиотеки Python и не требует дополнительной установки.

Инструменты для разработки:

Visual Studio Code - текстовый редактор для разработчиков, разработанный компанией Microsoft. Работает на Windows, macOS и Linux.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 АРХИТЕКТУРА ПО

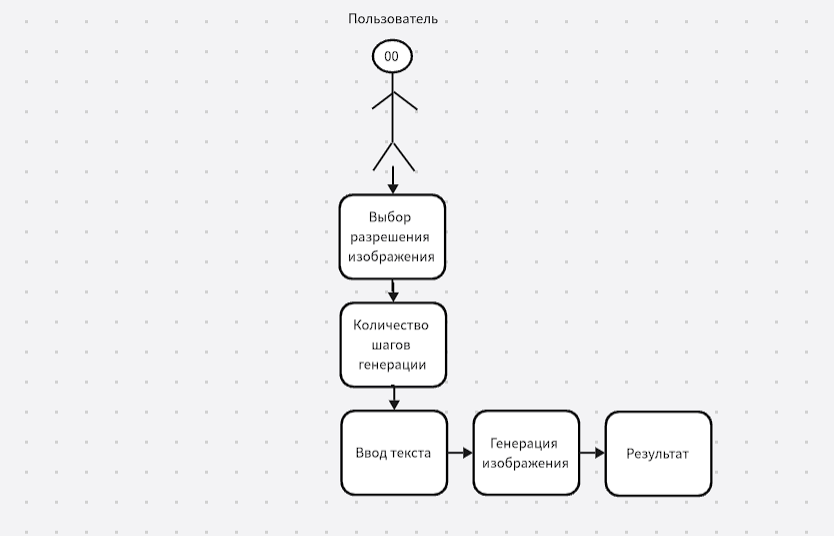


Рисунок 1. Архитектура ПО

# 2.2 РАЗРАБОТКА ПРОЕКТА ПО РОЛЯМ

1. Васильев Вячеслав и Огородов Данил:

gradio as gr: Фреймворк для создания пользовательских интерфейсов машинного обучения. Предоставляет компоненты для ввода данных (текст, изображения, числа) и отображения результатов.

gr.Blocks: Создание структуры приложения Gradio.

gr.Row: Организация макета интерфейса.gr.Column.

gr.Markdown: Добавление текстовой информации.

gr.Textbox: Создание текстового поля для ввода текстового запроса ().prompt.

gr.Dropdown: Создание выпадающего списка для выбора разрешения изображения ().resolution.

gr.Slider: Создание слайдера для выбора количества шагов генерации ().steps.

gr.Button: Создание кнопки «Сгенерировать» ().generate\_btn.

gr.Image: Создание элемента для отображения сгенерированного изображения ().output.

gr.Gallery: Создание галереи для отображения истории сгенерированных изображений ().gallery.

custom\_css: Стилизация элементов интерфейса с использованием CSS.

Выполненные задачи (GUI):

Создание основной структуры интерфейса в:with gr.Blocks(...)

Конфигурация блоков Gradio (тема, стили).

Создание и стилизация всех элементов (текстовое поле, выпадающий список, слайдер, кнопка, изображение, галерея).

Организация макетов (использование и ).gr.Rowgr.Column.

Реализация визуальных аспектов:

Применение кастомного CSS для стилизации интерфейса.

Определение отображения изображений в галерее.

Управление элементами интерфейса:

Определение входных и выходных параметров для функции генерации ().generate\_btn.click().

Определение функции, которая будет вызываться при нажатии на кнопку «Сгенерировать» ().generate.

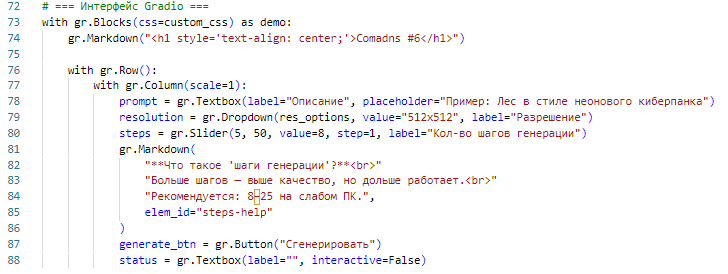


Рисунок 2 интерфейс gradio

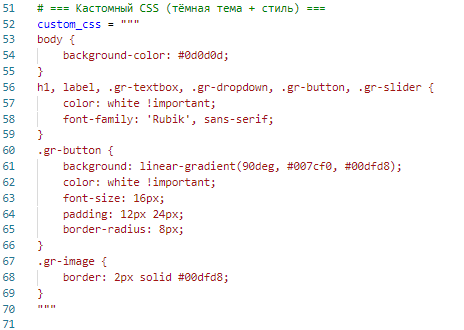


Рисунок 3 оформление

Габышев Владимир:

Импорт библиотек: Блок кода, импортирующий необходимые библиотеки (gradio, torch, diffusers, PIL, os, uuid).

Загрузка модели Stable Diffusion: Блок кода, загружающий предварительно обученную модель Stable Diffusion.

Определение функции generate: Функция, которая принимает входные параметры (текстовый запрос, разрешение, количество шагов), генерирует изображение с помощью модели Stable Diffusion, сохраняет его в историю, создает галерею последних изображений и возвращает изображение, галерею и статус.

Список разрешений: Блок кода, определяющий список доступных разрешений изображения.

Инициализация: Создание папки history для хранения изображений.

Запуск сервера: Строка demo.launch(share=True), которая запускает веб-сервер Gradio и предоставляет публичную ссылку для доступа к интерфейсу.

Импорт библиотек:

gradio as gr: Импортирует библиотеку Gradio для создания веб-интерфейса. as gr - это псевдоним, позволяющий обращаться к библиотеке, используя gr вместо gradio.

torch: Импортирует библиотеку PyTorch, используемую для работы с тензорами и нейронными сетями.

diffusers: Импортирует библиотеку Diffusers, предоставляющую предварительно обученные диффузионные модели, в данном случае Stable Diffusion.

PIL (Pillow): Импортирует библиотеку Pillow (Python Imaging Library) для работы с изображениями.

os: Импортирует модуль os для взаимодействия с операционной системой, например, для создания папок и работы с файлами.

uuid: Импортирует модуль uuid для генерации уникальных идентификаторов.

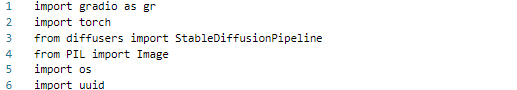


Рисунок 4 библиотеки

StableDiffusionPipeline.from\_pretrained(...): Загружает предварительно обученную модель Stable Diffusion с репозитория Hugging Face. «CompVis/stable-diffusion-v1-4» указывает на конкретную версию модели.

torch\_dtype=torch.float32: Указывает тип данных для тензоров, используемый моделью.

safety\_checker=None: Отключает проверку безопасности контента, генерируемого моделью. В реальных приложениях рекомендуется использовать проверку безопасности.

pipe.to(«cpu»): Перемещает модель на CPU. Для ускорения генерации можно использовать GPU, заменив «cpu» на «cuda». Однако, для работы на GPU требуется установленная и настроенная библиотека CUDA.

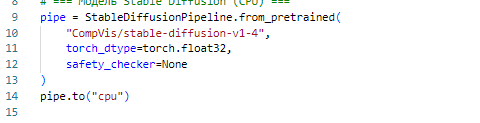


Рисунок 5 Модель

os.makedirs(«history», exist\_ok=True): Создает папку с именем «history» для хранения сгенерированных изображений. exist\_ok=True предотвращает возникновение ошибки, если папка уже существует.



Рисунок 6 История

Функция generate принимает три аргумента:

prompt: Текстовый запрос для генерации изображения.

resolution: Разрешение изображения в формате «ширинаxвысота» (например, «512x512»).

steps: Количество шагов, используемых моделью Stable Diffusion для генерации изображения. Чем больше шагов, тем выше качество изображения, но тем дольше время генерации.

if not prompt:: Проверяет, был ли введен текстовый запрос. Если запрос пустой, возвращает None для изображения, пустой список для галереи и сообщение «Введите запрос…».

try...except: Обрабатывает возможные исключения, которые могут возникнуть во время генерации изображения.

width, height = map(int, resolution.split(«x»)): Разделяет строку разрешения на ширину и высоту и преобразует их в целые числа.

image = pipe(prompt, num\_inference\_steps=steps).images[0]: Генерирует изображение с использованием модели Stable Diffusion.

pipe(prompt, num\_inference\_steps=steps): Вызывает модель Stable Diffusion для генерации изображения на основе текстового запроса prompt и количества шагов steps.

.images[0]: Извлекает первое (и единственное) изображение из результата.

image = image.resize((width, height)): Изменяет размер изображения до заданного разрешения.

filename = f«{uuid.uuid4().hex}.png»: Генерирует уникальное имя файла для изображения, используя UUID.

uuid.uuid4(): Создает новый UUIDv4 (случайный UUID).

.hex: Преобразует UUID в шестнадцатеричную строку.

path = os.path.join(«history», filename): Создает полный путь к файлу, объединяя имя папки «history» и имя файла.

image.save(path): Сохраняет изображение в файл.

files = ...: Получает список всех файлов в папке «history», сортирует их по времени изменения (от новых к старым) и выбирает последние 6 файлов.

gallery = [Image.open(f) for f in files[:6]]: Открывает выбранные файлы изображений и создает список объектов Image для галереи.

return image, gallery, f«Готово! Кол-во шагов: {steps}»: Возвращает сгенерированное изображение, галерею последних изображений и сообщение о статусе.

except Exception as e:: Обрабатывает любые исключения, произошедшие во время генерации изображения. Возвращает None для изображения, пустой список для галереи и сообщение об ошибке.

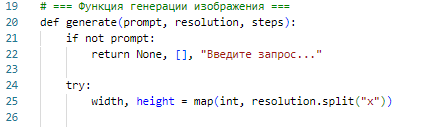


Рисунок 7 функция

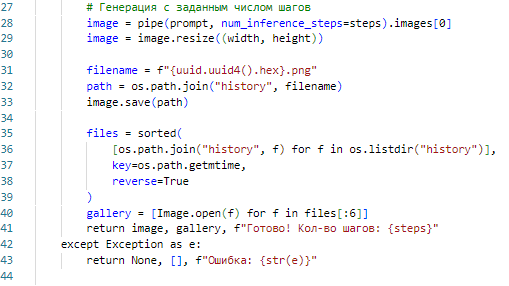


Рисунок 8 Генерация

res\_options: Определяет список доступных разрешений изображения для выбора в веб-интерфейсе.

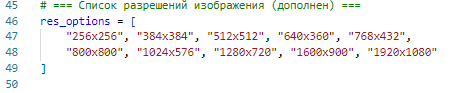


Рисунок 9 Список с разрешениями

Вот что получилось:

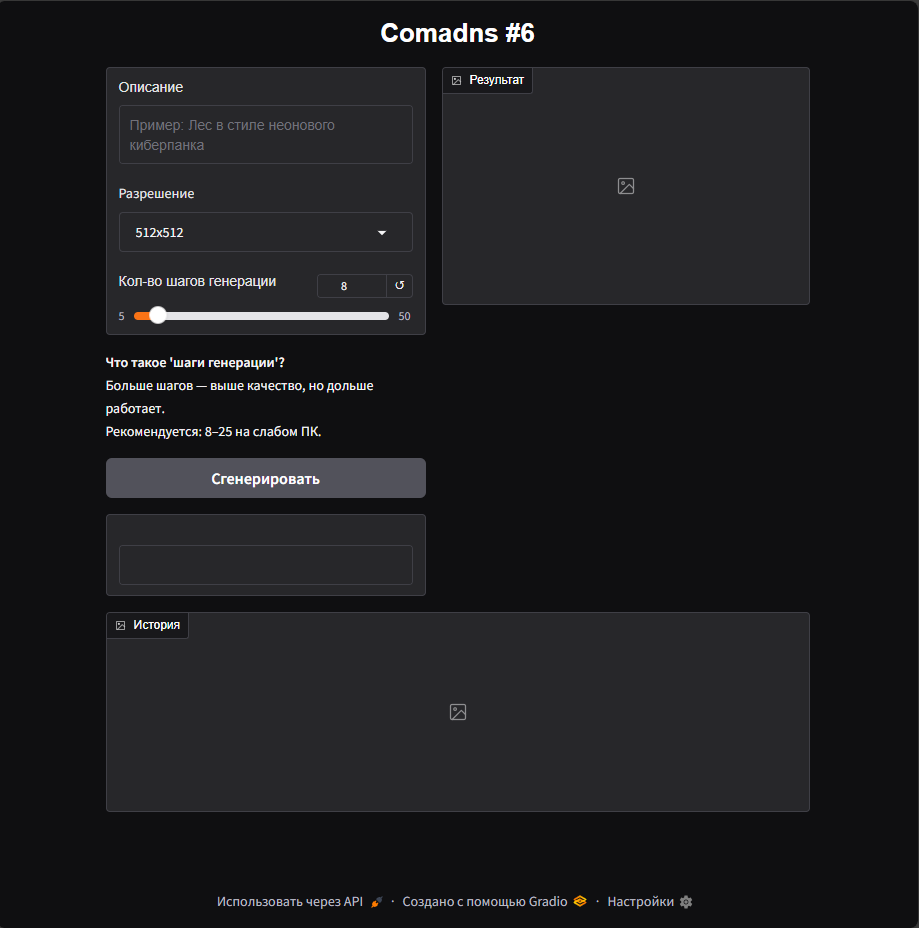


Рисунок 9 результат

# 2.3 КОНТРОЛЬ ВЫПОЛНЕНИЯ ПЛАНА

Выполненные задачи при разработке

Реализация ядра системы (Python):

Интеграция модели Stable Diffusion для генерации изображений на основе текстовых запросов.

Реализация загрузки и изменения размеров изображений (PIL - Pillow).

Графический интерфейс (Gradio):

Создание интерфейса приложения с помощью gr.Blocks.

Реализация интерактивных элементов:

Текстовое поле для ввода запроса (описание изображения).

Выпадающий список для выбора разрешения.

Слайдер для выбора количества шагов генерации.

Кнопка “Сгенерировать”.

Область для отображения результата (сгенерированного изображения).

Область для отображения истории изображений (галерея).

Настройка связей между элементами интерфейса и функцией генерации.

Функция generate принимает параметры из UI (запрос, разрешение, шаги) и возвращает результат.

Использование generate\_btn.click(...) для вызова generate при нажатии кнопки.

Дополнительный функционал:

Отображение статуса генерации в текстовом поле.

Отображение истории сгенерированных изображений в виде галереи.

Настройка кастомной CSS для изменения внешнего вида интерфейса (тёмная тема, стилизация).

Генерация уникальных имен файлов для сохранения изображений в историю.

Отображение сообщения об ошибке при возникновении проблем.

# ЗАКЛЮЧЕНИЕ

Проект «Разработка десктопного приложения для генерации изображений с использованием модели Stable Diffusion» стал значимым вкладом в область применения искусственного интеллекта в творчестве и дизайне. Мы успешно интегрировали мощную модель Stable Diffusion, что позволило пользователям создавать высококачественные изображения на основе текстовых описаний.

Разработанный интуитивно понятный интерфейс обеспечивает легкость в использовании, позволяя пользователям быстро вводить текстовые подсказки и настраивать параметры генерации. Кроссплатформенность приложения делает его доступным для широкой аудитории.

В будущем мы планируем расширить функционал приложения, добавив возможности редактирования изображений и интеграцию с другими графическими инструментами. Также рассматривается возможность дообучения модели на пользовательских данных для повышения качества генерации и создания более персонализированных результатов. Создание сообщества пользователей для обмена изображениями и идеями станет важным шагом к улучшению качества генерируемых изображений и развитию проекта в целом.

Таким образом, данный проект не только продемонстрировал потенциал искусственного интеллекта в области компьютерной графики, но и открыл новые горизонты для дальнейших исследований и разработок в этой захватывающей области.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.

<https://docs.python.org/3/>

<https://www.gradio.app/docs/>

<https://pytorch.org/docs/stable/>

<https://huggingface.co/docs/diffusers/index>

<https://pillow.readthedocs.io/en/stable/>

<https://docs.python.org/3/library/os.html>

<https://docs.python.org/3/library/uuid.html>

<https://huggingface.co/CompVis/stable-diffusion-v1-4>