

Генерация фотографий по профессиям в веб-приложении на основе ComfyUI

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В работе описана реализация генерации фотореалистичных изображений человека в выбранной профессии на основе загруженного изображения лица и параметров профессии. Источник данных представлен справочником professions.json, где для каждой профессии заданы сцена, действие, рабочая одежда и инструменты, используемые при формировании детализированного текстового описания. На серверной стороне реализован API-эндпоинт генерации, который валидирует выбранную профессию, объединяет базовый стиль, требования к композиции и негативные ограничения с профессиональным описанием, после чего подставляет параметры в JSON-шаблон workflow и отправляет запрос в локальный ComfyUI через HTTP API. На клиенте выполняется отправка запроса генерации и ожидание результата методом периодического опроса статуса до получения URL готового изображения.

Ключевые слова: генерация изображений; ComfyUI; workflow; промпт; негативный промпт; фотореализм; веб-приложение; Node.js; Express; JavaScript; JSON; HTTP API; polling.

Generation of photos by profession in a ComfyUI-based web application

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University

Student

Abstract

The paper describes the implementation of the generation of photorealistic images of a person in a chosen profession based on the uploaded face image and profession parameters. The data source is provided by the professions directory.json, where the scene, action, work clothes, and tools used in forming a detailed text description are set for each profession. A generation API endpoint is implemented on the server side, which validates the chosen profession, combines the basic style, composition requirements, and negative limitations with a professional description, after which it inserts the parameters into the workflow JSON template and sends a request to the local ComfyUI via the HTTP API. The client sends a generation request and waits for the result by periodically polling the status until the URL of the finished image is received.

Keywords: image generation; ComfyUI; workflow; industrial design; negative industrial design; photorealism; web application; Node.js; Express; JavaScript; JSON; HTTP API; polling.

1 Введение

1.1 Актуальность

Генерация визуальных материалов по профессиональным сценариям востребована в профориентационных сервисах, обучающих платформах и презентационных материалах, где требуется быстро получать реалистичные изображения в заданном контексте. Практическую значимость имеют решения, обеспечивающие стабильную композицию (полный рост, видимые руки и инструмент), контроль нежелательных артефактов (ошибки анатомии, кроп, мультяшность) и воспроизводимость результатов за счёт структурированных данных профессий и типового workflow генератора.

1.2 Обзор исследований

Перед анализом собственной реализации важно рассмотреть научные работы, посвящённые использованию веб-приложений. В работе Ванг З. рассматривается применение ComfyUI для генерации архитектурных визуализаций с акцентом на управляемость стилистики результата [1]. Исследование В. В. Бронской и А. Р. Ибатуллиной демонстрирует практическое применение React при создании пользовательских интерфейсов, где используются динамические элементы, включая визуальные карусели и интерактивные блоки [2]. В статье Д.А. Хайруллина и М.Р. Богданова рассматриваются особенности применения статической типизации в React-приложениях, включая методики типизации компонентов, хуков и состояния, а также влияние строгой типизации на надёжность пользовательских интерфейсов [3]. К.Р. Удальцов исследует вопросы обеспечения качества кода в командной разработке, предлагая формализацию шаблонов безопасных pull request-ов для проектов на React и TypeScript в рамках Git Flow [4]. Особое внимание созданию библиотек компонентов на основе TypeScript уделяется в работе Т.М. Воронко, где описываются принципы модульности, повторного использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5].

1.3 Цель исследования

Цель исследования – разработать и описать механизм генерации фотореалистичных изображений человека в выбранной профессии в веб-приложении на основе ComfyUI, включающий формирование промпта из параметров профессии, применение негативных ограничений, подстановку данных в workflow и получение результата через API.

2 Материалы и методы

В качестве основы использовано веб-приложение на базе Node.js и JavaScript, где серверная часть реализована на Express и принимает запрос

генерации с `imageUrl`, `professionId` и `clientId`. Данные профессий хранятся в `professions.json` и включают поля `scene`, `action`, `wardrobe`, `tools`, на основании которых на сервере формируется профессиональный фрагмент промпта, объединяемый с базовым стилем фотореализма, требованиями к композиции (полный рост, видимые руки/ноги, инструмент в кадре) и негативным промптом для подавления артефактов. Для генерации используется локальный ComfyUI через HTTP API

3 Результаты и обсуждения

Описание профессий вынесено в файл `professions.json` и используется как единый источник параметров для построения текстового описания сцены. Для каждой профессии задаются поля `label`, `scene`, `action`, `wardrobe`, `tools`, которые формируют контекст будущего изображения: окружение, действие персонажа, рабочая одежда и видимые инструменты. Такой формат позволяет поддерживать единый уровень детализации и управлять «сценарием» генерации на уровне данных без изменения серверной логики (рис. 1).

```
public > {} professions.json > ...
112 {
113   "traits": { "social": 2, "precision": 2, "manual": 1 },
114 },
115 {
116   "id": "waiter",
117   "label": "Официант",
118   "promptLabel": "waiter",
119   "scene": "restaurant dining area, tables set",
120   "action": "standing full-body in the restaurant, one hand holding a tray with dishes, other hand holdi",
121   "wardrobe": "service uniform, apron",
122   "tools": "tray, order pad",
123   "traits": { "social": 3, "leadership": 1, "manual": 1 }
124 },
125 {
126   "id": "sales-consultant",
127   "label": "Продавец-консультант",
128   "promptLabel": "sales consultant",
129   "scene": "modern retail store, shelves visible",
130   "action": "standing full-body in a retail store, one hand pointing to a product shelf, other hand hold",
131   "wardrobe": "store uniform, name badge",
132   "tools": "tablet, product shelf",
133   "traits": { "social": 3, "leadership": 1 }
134 },
135 {
136   "id": "hairstylist",
137   "label": "Парикмахер",
138   "promptLabel": "hairstylist",
139   "scene": "hair salon, mirrors, tool stations",
140   "action": "standing full-body at a salon chair, both hands holding scissors and comb near a mannequin",
141   "wardrobe": "stylist apron, dark outfit",
142   "tools": "scissors, comb, mannequin head",
143   "traits": { "creative": 2, "social": 2, "manual": 2, "precision": 1 }
144 },
145 {
146   "id": "makeup-artist",
147   "label": "Визажист",
148   "promptLabel": "makeup artist",
149   "scene": "makeup studio, lit mirror",
150   "action": "standing full-body at a makeup station, one hand holding a brush, other hand holding a pale
```

Рис. 1. Параметры профессии для сборки описания

Генерация запускается через серверный обработчик `POST /api/generate`. Сервер принимает изображение лица в формате `imageUrl`, идентификатор профессии и `clientId`, после чего выполняет проверку корректности выбранной профессии по справочнику. При отсутствии профессии сервер возвращает

ошибку, исключая дальнейшую обработку некорректного запроса. Наличие отдельного этапа валидации обеспечивает устойчивость механизма генерации и предотвращает формирование промпта для несуществующих сценариев

Для стабилизации результатов применяются три компонента текстовых ограничений. Во-первых, задаётся базовый стиль фотореализма, фиксирующий требования к освещению, цвету и детализации, а также необходимость сохранения черт лица, соответствующих загруженному изображению. Во-вторых, добавляется композиционное описание, обеспечивающее полный рост, видимость рук и ног, корректное положение человека и присутствие рабочей среды. В-третьих, используется негативный промпт, исключаяющий типовые дефекты генерации: мультяшность, низкое качество, деформации лица и рук, лишние конечности, кроп кадра, перекрытие лица предметами, а также нежелательный контент. Данные ограничения формируют «рамки» результата, внутри которых далее уточняется профессиональный сценарий (рис. 2).

```

21 const config = {
22   backend: (process.env.GENERATION_BACKEND || "comfyui").toLowerCase(),
23   mockGeneration: process.env.MOCK_GENERATION === "1",
24   replicate: {
25     apiToken: process.env.REPLICATE_API_TOKEN || "",
26     modelVersion: normalizeModelVersion(
27       process.env.REPLICATE_MODEL_VERSION || ""
28     ),
29     imageField: process.env.REPLICATE_IMAGE_FIELD || "image",
30     promptField: process.env.REPLICATE_PROMPT_FIELD || "prompt",
31     negativePromptField: process.env.REPLICATE_NEGATIVE_PROMPT_FIELD || "",
32     extraInput: process.env.REPLICATE_EXTRA_INPUT || "{}",
33   },
34   comfy: {
35     url: process.env.COMFYUI_URL || "http://127.0.0.1:8188",
36     publicUrl: process.env.COMFYUI_PUBLIC_URL || "",
37     promptPath: resolvePromptPath(
38       process.env.COMFYUI_PROMPT_PATH ||
39       path.join(__dirname, "comfyui", "prompt_faceid_api.json")
40     ),
41     checkpoint: process.env.COMFYUI_CHECKPOINT || "",
42     preset: process.env.COMFYUI_PRESET || "FACEID PLUS V2",
43     provider: process.env.COMFYUI_PROVIDER || "CUDA",
44     loraStrength: parseFloat(process.env.COMFYUI_LORA_STRENGTH || "0.6"),
45   },
46 };
47
48 const comfyBaseUrl = config.comfy.url.replace(/\/$/, "");
49 let comfyPromptTemplate = null;
50
51 const baseStylePrompt =
52   "photorealistic full-body documentary photo, natural lighting, true-to-life colors, sharp focus, rea
53 const compositionPrompt =
54   "full-body, head-to-toe visible, feet visible, standing upright, wide shot, subject centered, single
55 const negativePrompt =
56   "cartoon, anime, illustration, painting, 3d, cgi, render, plastic skin, oversmoothed, blurry, lowres
57
58 app.use(express.json({ limit: "15mb" }));
59 app.use(express.static(publicDir));
60
61 app.get("/api/health", (req, res) => {
62   res.json({ ok: true });
63 });

```

Рис. 2. Базовые промпты и негативные ограничения

В качестве генератора используется локальный ComfyUI, с которым сервер взаимодействует через HTTP API. Для запуска генерации сервер

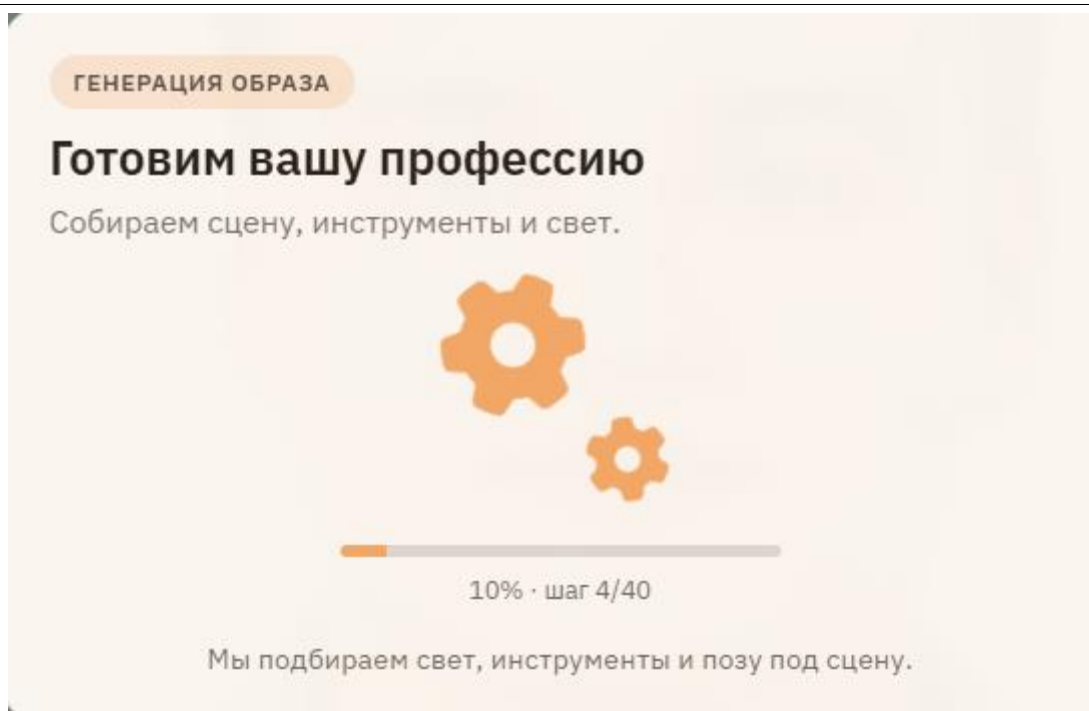


Рис. 4. Ожидание результата (polling)

После получения URL результат отображается в пользовательском интерфейсе: ссылка извлекается из ответа и передаётся в функцию показа результата, которая подставляет источник в элемент изображения. Это завершает цепочку генерации: от выбора профессии и загрузки лица до появления готового изображения по профессиональному сценарию (рис. 5).

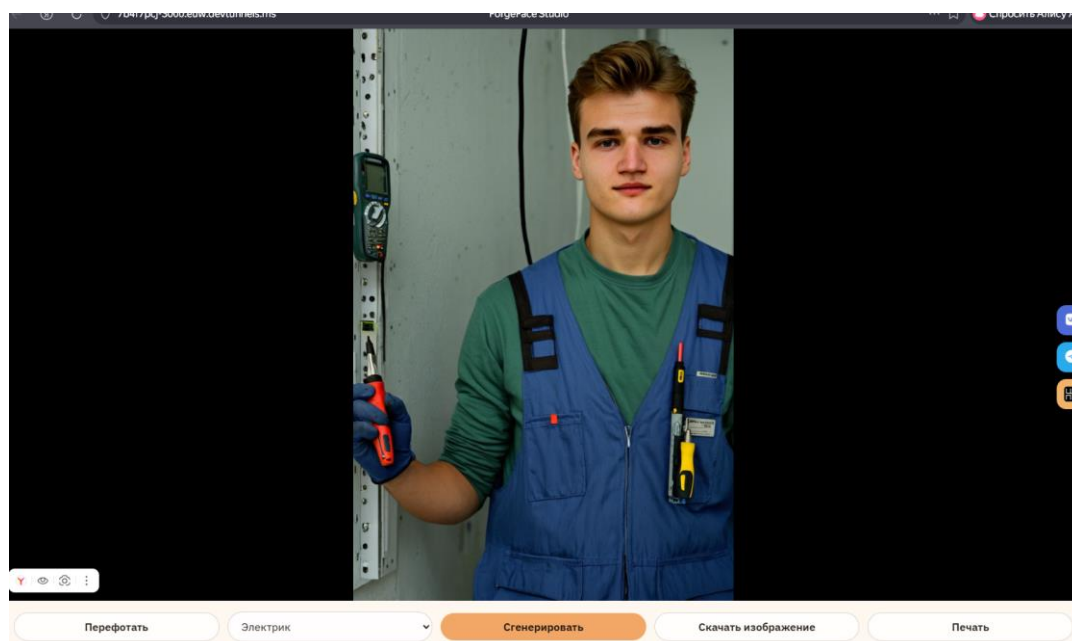


Рис. 5. Вывод результата на экран

4 Выводы

В ходе работы реализован механизм генерации фотореалистичных изображений человека в выбранной профессии на основе структурированных

данных профессий и локального генератора ComfyUI. Параметры профессии (сцена, действие, одежда и инструменты) хранятся в professions.json и используются для формирования профессионального сценария, который объединяется с базовыми требованиями к стилю, композиции и негативными ограничениями, повышающими стабильность результата. Запуск генерации выполняется через серверный эндпоинт POST /api/generate, где производится валидация профессии, сборка итогового промпта и подстановка параметров в JSON-workflow перед отправкой в ComfyUI с получением идентификатора задачи. На клиенте реализованы отправка запроса и ожидание результата методом периодического опроса статуса до получения URL итогового изображения и его отображения в интерфейсе.

Библиографический список

1. Wang Z. research on ComfyUI-based architectural rendering generation: constructing architect-oriented style transfer workflows through a functionalist architecture case study // Engineering Journal of Don. 2025. № 6 (126). С. 215–234.
2. Бронская В. В., Ибатуллина А. Р. Применение JavaScript-библиотеки React для создания приложения интернет-магазина одежды // Известия высших учебных заведений. Технология текстильной промышленности. 2023. № 4 (406). С. 169–175.
3. Хайруллин Д.А., Богданов М.Р. Изучение использования TypeScript в React-приложениях, преимущества статической типизации, подходы к типизации компонентов, хуков и состояния // Международный журнал гуманитарных и естественных наук. 2024. № 12-3 (99). С. 233–238.
4. Удальцов К.Р. Формализация шаблонов безопасных pull request-ов для React-приложений с TypeScript в командах с Git Flow // Актуальные проблемы социально-гуманитарного и научно-технического знания. 2025. № 4 (44). С. 35–36.
5. Воронко Т.М. Создание React-библиотеки с использованием TypeScript // Электронные системы и технологии. Материалы 58-й научной конференции аспирантов, магистрантов и студентов БГУИР. Минск, 2022. С. 200–202.