РУКОВОДСТВО ПО УСТАНОВКЕ СИСТЕМЫ «СЕРВИС ГЕНЕРАЦИИ ТЕКСТА НА ЯЗЫКАХ ПРОГРАММИРОВАНИЯ»

Версия 1.2 • 16/12/2024

ООО «МВС ИИ»
Г. МОСКВА

ОГЛАВЛЕНИЕ

> В каждую модель, предоставляемую нашим клиентам и партнерам, мы встраиваем уникальный водяной знак. Это необходимо для установления факта утечки модели от клиента или партнера. Просим вас ответственно относиться к обеспечению безопасности хранения модели.

1 — О ПРОДУКТЕ

«Сервис генерации текста на языках программирования» или «Kodify» (далее по тексту будет использовано второе наименование ПО) от МВС ИИ представляет собой AI-ассистента разработчика, инструмент, который использует искусственный интеллект для автоматизации рутинных процессов и помощи разработчикам в выполнении различных задач при написании программного кода. ПО предназначено для облегчения процесса разработки, предоставляя инструменты для генерации кода, улучшения его качества и автоматизации рутинных задач. Kodify поддерживает различные языки программирования и интеграции с популярными инструментами разработки.

Функционал продукта:

Kodify от МВС ИИ предоставляет функционал для автоматической генерации кода на основе данных из редактируемого пользователем файла. Продукт поддерживает различные языки программирования (например, Python, C#, Java, Go, JavaScript). ПО обеспечивает автопродление кода с помощью LLM, которая на основе уже написанного пользователем кода генерирует завершение строки. Интеграция с инструментами разработки включает встраивание специального плагина в популярные интегрированные среды разработки (IDE), таких как Visual Studio Code.

Преимущества продукта:

- **Повышение производительности:**
Kodify от МВС ИИ обеспечивает повышение производительности разработчиков за счет автоматической генерации кода и упрощения процесса разработки, что ускоряет процесс написания кода и снижает время на исправление ошибок и оптимизацию.
- **Улучшение качества программного обеспечения:**
Продукт способствует улучшению качества программного обеспечения посредством генерации кода высокого качества с меньшим количеством багов и уязвимостей.

2 — АРХИТЕКТУРА

Пользователь взаимодействует с системой через интерфейс Visual Studio Code (VSCode). В данном контексте IDE используется как интерфейс для отправки запросов к системе Kodify через специальный плагин, устанавливаемый отдельно. Установка плагина в IDE даёт пользователю возможность обращаться к Kodify для для автоматической генерации кода на основе данных из редактируемого пользователем файла, а также другим функциям Kodify.

Kodify — это основная система, которая выполняет обработку запросов. MTS Kodify предоставляет REST API, к которому обращается плагин, установленный в IDE.

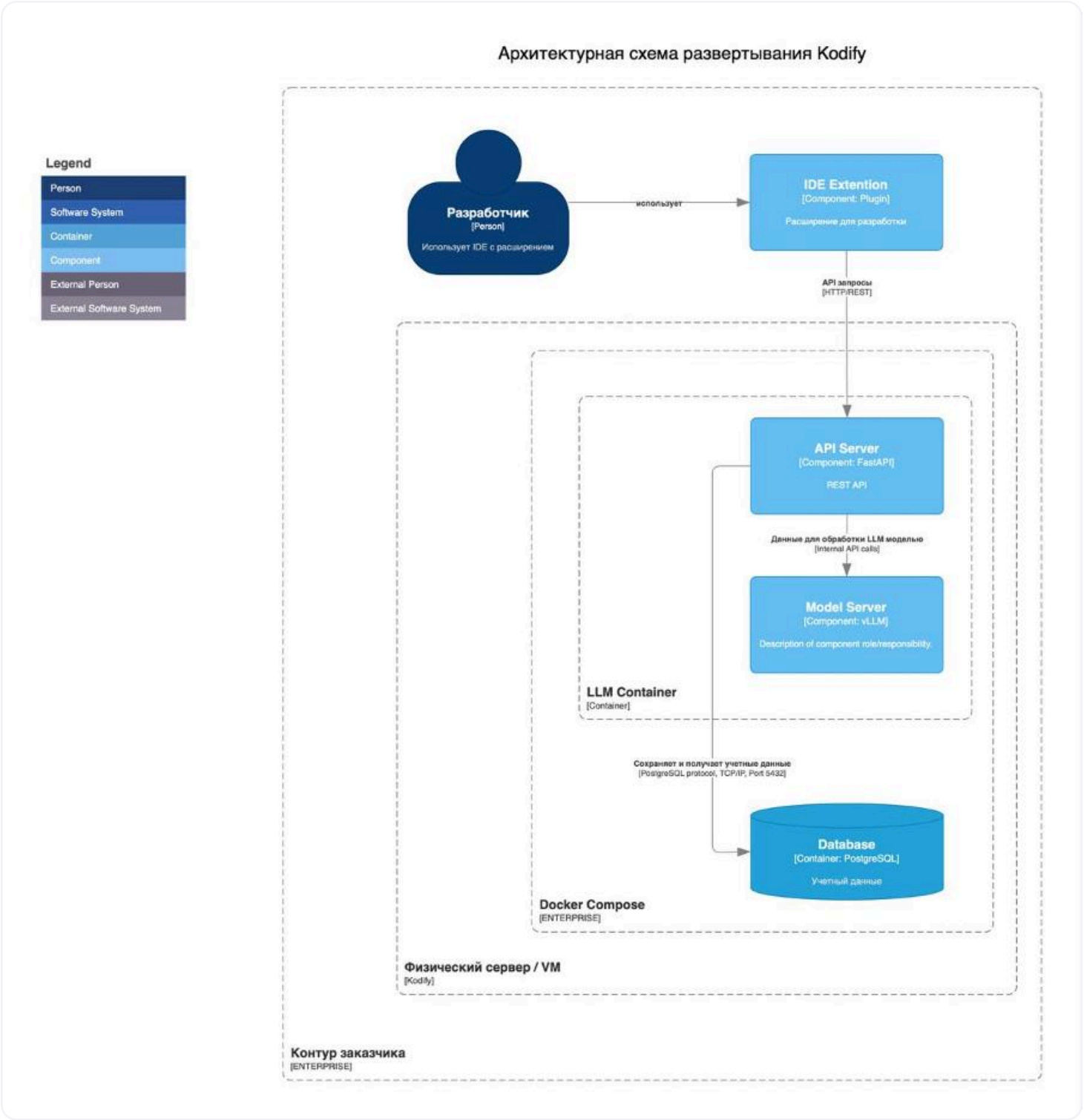

Схема 1. Взаимодействие компонентов

3 — ОСНОВНЫЕ ТЕРМИНЫ В ДОКУМЕНТЕ

Термин	Определение
Искусственный интеллект (ИИ)	Это область компьютерных наук, занимающаяся созданием машин, способных выполнять задачи, требующие человеческого интеллекта, например, восприятие, рассуждение, обучение и решение проблем.
Машинное обучение	Отрасль искусственного интеллекта, в которой машины обучаются выполнять задачи, анализируя и обобщая данные, а не благодаря прямому программированию.
Генеративные модели	Это типы искусственного интеллекта, которые могут создавать новый контент, например тексты, изображения и музыку. Они учатся на больших объемах данных и затем генерируют новый контент, имитируя то, что они видели.
Дообучение (Fine-tuning)	Процесс настройки предобученной модели под конкретную задачу путем дополнительного обучения на более мелком или специфическом наборе данных.
Нейронная сеть	Компьютерная модель, вдохновленная биологическими нейронами, используемая для распознавания паттернов и обработки информации в задачах машинного обучения.
Токен	Минимальная единица текста (например, слово или символ), используемая в обработке естественного языка для анализа и генерации текста.
Натуральный язык (NLP)	Область ИИ, занимающаяся взаимодействием между компьютерами и людьми на естественных языках, таких как русский или английский.
API	Набор правил и инструментов для взаимодействия программного обеспечения. API позволяет различным приложениям обмениваться данными и функциональностью. (Application Programming Interface)
Docker	Платформа для автоматизации развёртывания приложений в контейнерах, что обеспечивает их изоляцию и независимость от среды выполнения.
JSON	Лёгкий формат обмена данными, легко читаемый человеком и парсируемый компьютером. (JavaScript Object Notation)

4 — ПРОГРАММНЫЕ И АППАРАТНЫЕ ТРЕБОВАНИЯ

Требования к программному обеспечению рабочей станции

Для работы системы необходимо, чтобы выполнялись следующие требования к программному обеспечению:

Таблица 1. Требования к программному обеспечению

Ресурс	Требования
Операционная система	Rockylinux8 или astralinux.ru/library/astra/ubi18:1.8.1
Рекомендованная ОС	Rockylinux8 или astralinux.ru/library/astra/ubi18:1.8.1
Docker	Docker version 24.0.4
Nvidia-Docker	Docker version 24.0.4, build 3713ee1 https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/1.8.1/install-guide.html https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html
Интернет	Наличие доступа к Интернет для контейнеров и дополнительных загрузок ПО

Требования к аппаратному обеспечению рабочей станции

Для работы системы необходимо, чтобы выполнялись следующие требования к аппаратным ресурсам:

Таблица 2. Требования к аппаратному обеспечению

Ресурс	Требования
CPU	от 8
RAM	от 32 GB
GPU	1x NVIDIA A100 с 40 GB видеопамяти
SSD	500 GB
Видеодрайвер	Driver Version: 525.105.17, CUDA Version: 12.0

5 — РАЗВЕРТЫВАНИЕ СИСТЕМЫ

> Для развертывания системы в собственной инфраструктуре необходимо выполнить пункты из данного раздела.

1. Скачивание файлов

Скачайте файл docker-compose и env-файл. Вам будут переданы логин и пароль от УЗ Artifactory. Перейдите по ссылке https://artifactory.mts.ai/ui/login/ и пройдите аутентификацию, используя полученные учётные данные. Ссылку на папку с вашим образом вам предоставят отдельно. Поместите скачанные файлы в желаемую директорию.

2. Проверка и конфигурация env-файла

Ознакомьтесь с содержимым env-файла, который передан вместе с docker-compose файлом. Этот файл содержит значения переменных окружения. При необходимости, сконфигурируйте переменные. Подробнее о переменных рассказано в разделе «6 — ПРОВЕРКА И КОНФИГУРАЦИЯ ENV-ФАЙЛА».

Также проверьте заполнение docker-compose файла. Все переменные должны быть заполнены как на скриншоте ниже. Редактировать можно только переменную ports, если вы хотите указать иной адрес для приложения.

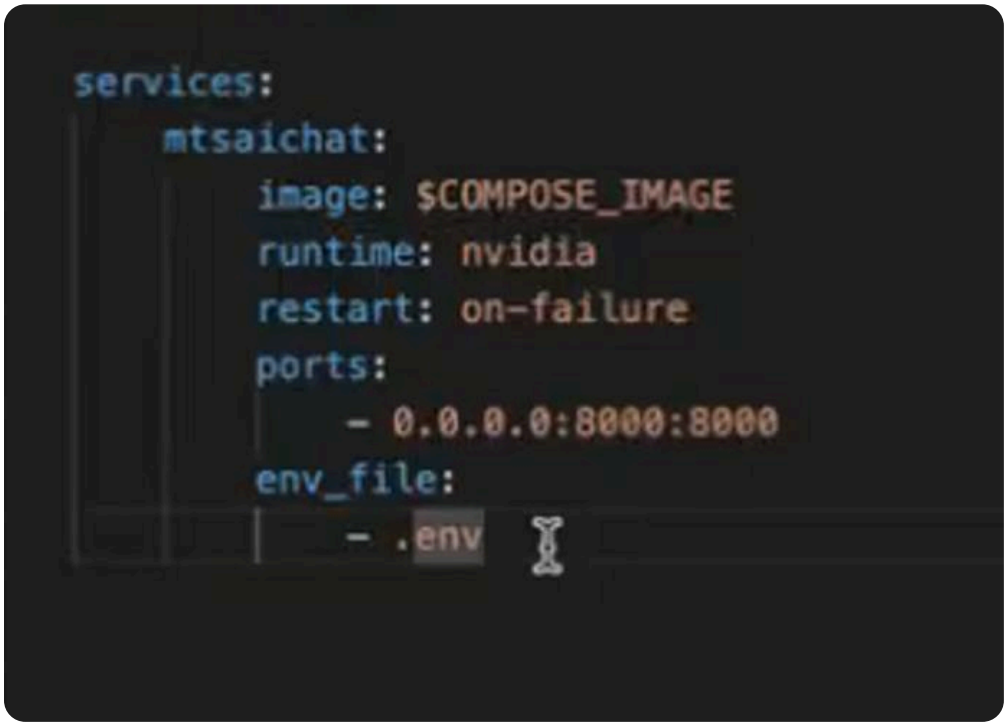

Рисунок 1. Заполнение docker-compose

При использовании пользовательской авторизации (MTSAI_AUTH=true) вы можете развернуть базу данных PostgreSQL (работает на версии 16.4, но нет жестких требований к версии) одновременно с сервисом, используя Docker Compose. Для этого добавьте в docker-compose.yml сервис базы данных:

```
services:
 db:
  image: postgres:latest
  restart: on-failure
  env_file: .env
  ports:
   - ${DB_LOCAL_PORT}:${DB_PORT}
  volumes:
   - mtsaichat_pgdata:/var/lib/postgresql/data

volumes:
 mtsaichat_pgdata:
```

> Volume (mtsaichat_pgdata) обеспечивает сохранность данных базы между перезапусками контейнера. Данные будут храниться в директории /var/lib/postgresql/data внутри контейнера и будут доступны после остановки и удаления контейнера.

При развертывании системы на нескольких видеокартах необходимо добавить в docker-compose.yml дополнительный параметр:

```
shm_size: 64g
```

3. Далее необходимо залогиниться в artifactory с использованием вашей учетной записи через использование команды:

```
docker login artifactory.mts.ai
```

4. Миграция БД

Актуально в том случае, если планируется использовать журналирование (значение переменной LOGS в ENV-файле “True”). Если журналирование отключено, то данный этап можно пропустить и перейти к шагу №5.

- Если приложение эксплуатируется в среде docker-compose, то необходимо дополнить docker-compose файл. Требуется добавить строчку:

```
entrypoint: alembic -c /vllm-workspace/middleware/gpt_logger/alembic.ini upgrade head && sh app.sh
```

- Если приложение эксплуатируется в среде Kubernetes, необходимо создать init-контейнер с тем же образом, что и в основном контейнере сервиса. В нём используйте команду:

```
alembic -c /vllm-workspace/middleware/gpt_logger/alembic.ini upgrade head
```

5. Запуск контейнера

Запустите контейнер с помощью команды:

```
docker compose up -d
```

> Убедитесь, что у вас установлен Docker Compose.

6. Проверка работы системы

Проверьте, что система работает корректно с помощью API-запроса GET/health. Запрос подробно описан в пункте 8.1. Проверка работоспособности модели.

Либо откройте логи контейнера с помощью команды:

```
docker logs CONTAINER_ID
```

При успешной установке приложения там отобразится следующее:

```
INFO: Application startup complete/
INFO: Unicorn running on http://.....
```

6 — ПРОВЕРКА И КОНФИГУРАЦИЯ ENV-ФАЙЛА

> Вместе с docker-compose файлом вам будет передан env-файл, который содержит значения переменных окружения. Вам необходимо ознакомиться с этими значениями и при необходимости сконфигурировать их. В файле будут переменные, указанные ниже.

6.1 — ПЕРЕМЕННЫЕ ПОТРЕБЛЕНИЯ ВИДЕОПАМЯТИ

Таблица 5.1 Переменные потребления видеопамяти

Переменная	Определение
GPUUTIL	Параметр определяет значение передаваемое в --gpu_memory_utilization. Параметр определяет долю видеопамяти GPU, которая будет зарезервирована для использования модели, включая память для весов модели, активации и KV кэша (ключ-значение). Значение этого параметра может варьироваться от 0 до 1. Значение 0.9 означает, что 90% видеопамяти GPU будет зарезервировано. Это позволяет увеличить размер кэша KV, что может улучшить пропускную способность модели.
DTYPE	Определяет то, в каком типе данных используются веса модели (auto,half,float16,bfloat16,float,float32). По умолчанию установлен на тип данных, совместимый с видеокартой A100 80 GB. В этом случае нет необходимости добавлять параметр в ENV-файл. Если вы собираетесь использовать иную видеокарту, то обратитесь за консультацией к специалисту со стороны МТС ИИ, вам подскажут какое значение переменной необходимо установить.

5.1.1 Переменные для запуска сервиса на нескольких видеокартах

Переменные, необходимые для корректной работы сервиса при использовании нескольких видеокарт.

tensor-parallel-size	Переменная, необходимая для запуска сервиса на нескольких видеокартах. Перед использованием проконсультируйтесь с сотрудником МТС ИИ, так как в зависимости от количества видеокарт и используемой модели Cotype, может потребоваться или переменная tensor-parallel-size, или pipeline-parallel-size. Пример заполнения: ADDITIONAL_ARGS =--tensor-parallel-size=<кол-во карт>
pipeline-parallel-size	Переменная, необходимая для запуска сервиса на нескольких видеокартах. Перед использованием проконсультируйтесь с сотрудником МТС ИИ, так как в зависимости от количества видеокарт и используемой модели Cotype, может потребоваться или переменная tensor-parallel-size, или pipeline-parallel-size. Пример заполнения: ADDITIONAL_ARGS =--pipeline-parallel-size=<кол-во карт>
PYTORCH_CUDA_ALLOC_CONF	В случае если сервер рестартится при инференсе системы из-за нехватки памяти, то следует использовать данную переменную с значением PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True. Это уменьшает кол-во резервируемой памяти pytorch, что решает проблему падения. В ином случае использовать данную переменную не следует.

6.2 — ПЕРЕМЕННЫЕ АВТОРИЗАЦИИ

Переменные авторизации используются для хранения и управления данными, необходимыми для аутентификации и авторизации пользователей в системе.

Таблица 6. Переменные авторизации

Переменная	Определение
MTSAI_AUTH	Этот параметр определяет метод авторизации. При установке значения true, активируется пользовательская авторизация, которая требует специфических данных аутентификации. Значение false активирует однотокеновую или dummy авторизацию. Выбор между однотокеновой и dummy авторизацией зависит от значения переменной VLLM_API_KEY.
COMPOSE_IMAGE	Ссылка на образ в Artifactory, которая будет передана клиенту отдельно. Необходима для скачивания экземпляра ПО.
AUTH_TYPE	Признак использования пользовательской авторизации. Если вы используете пользовательскую авторизацию, следует указать значение «DB» Дефолтное значение «NO», можно не указывать если используйте иной тип авторизации.
VLLM_API_KEY	Токен авторизации. Строка, указывающая токен для авторизации, одинаковый для всех пользователей системы. Необходимо заполнить только в том случае, если AUTH_TYPE: «NO». Любое значение, которое вы укажете, станет токеном и будет активирована однотокеновая авторизация.

> Однотокеновая авторизация (one-token auth) — это авторизация, при которой существует только 1 токен, способный пройти авторизацию (его значение необходимо задать в рамках конфигурации env).
>
> Пользовательская авторизация - это авторизация, при которой каждому пользователю создается уникальный токен. Информация о пользователях хранится в БД.

6.3 — ПЕРЕМЕННЫЕ БАЗЫ ДАННЫХ

Переменные базы данных используются для хранения конфигурационных данных, необходимых для подключения и работы с базой данных при включённой пользовательской авторизации (MTSAI_AUTH = true). Не используются при ином типе авторизации и отсутствуют по умолчанию в env-файле. Добавьте их в файл вручную только в том случае, если вам необходима пользовательская авторизация.

Таблица 7. Переменные базы данных

Переменная	Определение
USERS_TABLE	Идентификатор таблицы в базе данных PostgreSQL. Требует указания полного имени таблицы. Этот параметр необходим, когда используется пользовательская авторизация для хранения имени таблицы с данными о пользователях.
DB_URL	Заполняется следующим образом: DB_URL=postgresql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/${DB_NAME} Содержит информацию о БД с пользователями и данные для подключения к этой БД. Не используется при ином типе авторизации, кроме пользовательской.

В случае, если вы разворачиваете PostgreSQL одновременно с сервисом, используя Docker Compose, добавьте в .env файл следующие переменные:

- желаемые параметры доступа к базе данных:

```
DB_USER=your_username
DB_PASSWORD=your_password
DB_NAME=your_database
```

- алиасы для PostgreSQL:

```
POSTGRES_USER=${DB_USER}
POSTGRES_PASSWORD=${DB_PASSWORD}
POSTGRES_DB=${DB_NAME}
```

- параметры подключения:

```
DB_HOST=db
DB_PORT=5432
DB_LOCAL_PORT=5432
```

> Значение DB_HOST должно быть равно 'db' - это имя сервиса в docker-compose.
> DB_LOCAL_PORT определяет, на каком порту база будет доступна на вашей машине.
> Измените, если порт 5432 занят.

При первом запуске система автоматически создаст необходимую базу данных и таблицу пользователей. Дополнительных действий по инициализации базы данных не требуется.

6.4 — ПЕРЕМЕННЫЕ НАСТРОЙКИ ГЕНЕРАЦИИ МОДЕЛИ

Переменные настройки генерации модели устанавливают ограничения при обработке запросов POST/chat/completions.

Таблица 8. Переменные настройки генерации модели

Переменная	Определение
MAX_N	Устанавливает ограничение для опционального параметра «n», использующегося в запросе POST/chat/completions. Параметр отвечает за то, сколько ответов модель сгенерирует по запросу пользователя. Необходимо установить максимально допустимое значение для параметра “n”, чтобы избежать ситуации, когда пользователь задал слишком большой “n” и на один запрос модель генерирует сотни или тысячи ответов. Значение MAX_N по умолчанию: 10.

6.5 — ПЕРЕМЕННЫЕ ЖУРНАЛИРОВАНИЯ

Переменные журналирования используются для настройки журналирования api-запросов и определения таблицы БД, куда будет сохранён журнал обработки пользовательских запросов. Все переменные из данного раздела обязательны к заполнению, кроме переменной LOGS.

Переменная	Определение
LOGS	Журналирование отключено по умолчанию. Для того чтобы его активировать нужно указать значение переменной True. Если не установить данное значение и не указывать переменную в ENV-файле, то журналирование будет отключено. Для работы текущей версии ПО журналирование должно быть отключено.
LOGS_DB_HOST	Адрес сервера PostgreSQL
LOGS_DB_PORT	Порт PostgreSQL, куда будут сохраняться журнал запросов
LOGS_DB_NAME	Имя базы данных, в которую будут сохраняться журнал запросов
LOGS_DB_USERNAME	Имя пользователя для подключения к базе данных
LOGS_DB_PASSWORD	Пароль пользователя для подключения к базе данных
LOGS_DB_ENGINE	Протокол для синхронного доступа к PostgreSQL. Значение по умолчанию: postgresql Рекомендуем оставить данное значение и не редактировать переменную.
LOGS_DB_ASYNC_ENGINE	Протокол для асинхронного доступа к PostgreSQL. Значение по умолчанию: asyncpg Рекомендуем оставить данное значение и не редактировать переменную.

6.6 — ПРИМЕР ЗАПОЛНЕНИЯ ФАЙЛА

6.6.1 Однотокеновая авторизация:

- GPUUTIL: 0.9
- MTSAI_AUTH: false
- VLLM_API_KEY: <токен авторизации>
- COMPOSE_IMAGE: <ссылка на скачивание образа>
- USERS_TABLE: <переменная отсутствует>
- AUTH_TYPE: <переменная отсутствует>
- DB_URL: <переменная отсутствует>
- MAX_N: 10

6.6.2 Пользовательская авторизация:

- GPUUTIL: 0.9
- MTSAI_AUTH: true
- VLLM_API_KEY: <переменная отсутствует>
- COMPOSE_IMAGE: <ссылка на скачивание образа>
- USERS_TABLE: <идентификатор таблицы>
- AUTH_TYPE: DB
- DB_URL:postgresql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/${DB_NAME}
- MAX_N: 10

6.6.3 Отключение журналированиия:

- LOGS – переменная отсутствует в ENV-файле
- LOGS_DB_HOST: <HOST>
- LOGS_DB_POST: <PORT>
- LOGS_DB_NAME: <DB_NAME>
- LOGS_DB_USERNAME: <USERNAME>
- LOGS_DB_PASSWORD: <PASSWORD>
- LOGS_DB_ENGINE: postgresql
- LOGS_DB_ASYNC_ENGINE: asyncpg

Переменные журналирования, кроме LOGS, обязательны к заполнению, даже если вы хотите чтобы журналирование было отключено. Они необходимы для корректного запуска приложения.

6.6.4 Активное журналирование:

- LOGS: True
- LOGS_DB_HOST: <HOST>
- LOGS_DB_POST: <PORT>
- LOGS_DB_NAME: <DB_NAME>
- LOGS_DB_USERNAME: <USERNAME>
- LOGS_DB_PASSWORD: <PASSWORD>
- LOGS_DB_ENGINE: postgresql
- LOGS_DB_ASYNC_ENGINE: asyncpg

> При необходимости скорректируйте значения переменных в env-файле.
> После внесения изменений, поместите env-файл в ту же директорию, где находится docker-compose.yml.