Общество с ограниченной ответственностью «УМАРТА»

Сервис для автоматического перевода документов с использованием LLM-моделей «УМАРТА.Переводчик»

Руководство администратора

Листов 23

Москва, 2025 г.

АННОТАЦИЯ

Настоящий документ представляет собой руководство администратора программного обеспечения «Сервис для автоматического перевода документов с использованием LLM-моделей "УМАРТА.Переводчик"».

Документ охватывает следующие аспекты:

- архитектуру Системы (FastAPI backend, React frontend с Ant Design, PostgreSQL, Nginx, Docker Compose);
- структуру базы данных (таблицы users, files, subscription_plans, user_subscriptions, translation_jobs);
- описание API и команд взаимодействия (на основе фактических curl-запросов и маршрутов OpenAPI);
- конфигурацию системы (переменные окружения, структура файлов, параметры docker-сервисов);
- практические команды и операции: запуск системы, создание администратора, мониторинг, резервное копирование и восстановление;
- интеграцию с LLM-провайдерами (OpenAI, DeepSeek, LMStudio и др.).

Инструкция предназначена для системных администраторов и DevOps-специалистов, обеспечивающих установку, конфигурацию, эксплуатацию и техническую поддержку Системы.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

Полное наименование: Сервис для автоматического перевода документов с использованием LLM-моделей «УМАРТА.Переводчик».

Краткое наименование: Система, УМАРТА.Переводчик.

1.1 Краткое описание возможностей

Система предназначена для автоматизированного перевода текстовых файлов с использованием современных языковых моделей (LLM).

Ключевые возможности включают:

- Регистрация и авторизация пользователей, а также разграничение прав доступа (администратор, пользователь).
- Загрузка файлов для последующего перевода с указанием языка и предметной области.
- Интеграция с LLM-провайдерами, такими как OpenAI, DeepSeek, LMStudio, с возможностью настройки глобального или пользовательского провайдера.
- Гибкая система подписок и тарифных планов, включая лимиты по символам и размеру файлов.
- Обработка переводов в очереди, с отображением прогресса, ошибок и результатов.
- Интеграция с платёжной системой ЮKassa для оформления подписок.

1.2 Требования к квалификации пользователей

Для эффективной работы с Системой системный администратор должен обладать следующими навыками и компетенциями:

- знание принципов работы контейнеризации и управления окружением (Docker, Docker Compose);
- опыт администрирования веб-приложений на базе стеков FastAPI, React, PostgreSQL;
- базовые навыки работы с Linux-системами и командной строкой;
- понимание принципов работы API и умение выполнять HTTP-запросы (в том числе с использованием curl);
- навыки настройки и мониторинга веб-сервера Nginx;
- опыт работы с системами контроля версий (Git) и CI/CD-пайплайнами;
- знание основ информационной безопасности и защиты данных при работе с LLM-провайдерами (OpenAI, DeepSeek и др.).

2 СТРАКТУРА СИСТЕМЫ

2.1 АРХИТЕКТУРА СИСТЕМЫ

Система состоит из следующих компонентов:

- **Backend**: FastAPI приложение на Python;
- **Frontend**: React приложение с TypeScript и Ant Design;
- **База данных**: PostgreSQL;
- **Веб-сервер**: Nginx;
- **Контейнеризация**: Docker Compose.

2.2 ОСНОВНЫЕ МОДУЛИ СИСТЕМЫ

Ниже приведён перечень основных модулей, реализованных в Системе, и их назначения:

- **Аутентификация** (`app/auth/`) - регистрация, авторизация, управление пользователями;
- **Файлы** (`app/files/`) - загрузка и обработка файлов для перевода;
- **Подписки** (`app/subscriptions/`) - управление тарифными планами и подписками;
- **Платежи** (`app/payments/`) - интеграция с ЮKassa;
- **Переводы** (`app/translation`/) - система очередей и обработки переводов;
- **LLM** (`app/llm/`) - интеграция с провайдерами ИИ (DeepSeek, OpenAI, LMStudio);
- **Система** (`app/system/`) - системные настройки и языки.

2.3 ТРЕБОВАНИЯ К ТЕХНИЧЕСКОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Требования к минимальным характеристикам технических средств:

- Не менее 4 ГБ свободной оперативной памяти;
- Не менее 10 ГБ свободного пространства на диске;
- Свободные порты:
  - 80 – для веб-доступа (nginx);
  - 5432 – для подключения к базе данных PostgreSQL.

Требование к программному обеспечению на пользовательском устройстве:

- Наличие установленных компонентов:
  - Docker версии 20.10 и выше;
  - Docker Compose версии 2.0 и выше.

3 СТРУКТУРА БАЗЫ ДАННЫХ

Структура включает несколько основных таблиц, обеспечивающих хранение информации о пользователях, загруженных файлах, подписках и заданиях на перевод. Ниже приведено описание ключевых таблиц и их назначение.

3.1 Основные таблицы

3.1.1 `users` – Пользователи

Таблица содержит данные зарегистрированных пользователей, включая учётные данные, статус верификации, роль, используемого LLM-провайдера и информацию о блокировках.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    hashed_password VARCHAR(255),
    is_verified BOOLEAN DEFAULT FALSE,
    role VARCHAR(50) DEFAULT 'user',
    llm_provider VARCHAR(50),
    login_attempts INTEGER DEFAULT 0,
    is_blocked BOOLEAN DEFAULT FALSE,
    blocked_until TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3.1.2 `files` - Файлы для перевода

Таблица предназначена для хранения информации о загруженных пользователями документах. Включает данные о путях к оригинальному и переведённому файлу, статус обработки, языковые параметры, рейтинг пользователя и параметры LLM-провайдера.

```
CREATE TABLE files (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    original_name VARCHAR(255) NOT NULL,
    file_path VARCHAR(512) UNIQUE NOT NULL,
    translated_path VARCHAR(512),
    language VARCHAR(10) NOT NULL,
    domain VARCHAR(255),
    status VARCHAR(50) DEFAULT 'pending',
    char_count_original INTEGER,
    is_temporary BOOLEAN DEFAULT FALSE,
    temp_id VARCHAR(36) UNIQUE,
    llm_provider VARCHAR(50),
    user_rating INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    translated_at TIMESTAMP
);
```

3.1.3 `subscription_plans` - Тарифные планы

Содержит данные о доступных тарифах системы, включая стоимость, ограничения по символам и размеру файлов, а также описание включённых функций. Используется как публичной частью системы, так и для административного управления.

```
CREATE TABLE subscription_plans (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) UNIQUE NOT NULL,
    description TEXT,
    price_monthly DECIMAL(10,2),
    price_yearly DECIMAL(10,2),
    currency_code VARCHAR(3) DEFAULT 'RUB',
    char_limit_monthly INTEGER,
    char_limit_daily_free INTEGER,
    max_file_size_mb INTEGER DEFAULT 10,
    max_total_upload_size_mb INTEGER DEFAULT 250,
    is_active BOOLEAN DEFAULT TRUE,
    is_public BOOLEAN DEFAULT TRUE,
    is_featured BOOLEAN DEFAULT FALSE,
    duration_days INTEGER DEFAULT 30,
    allows_auto_renew BOOLEAN DEFAULT TRUE,
    features JSONB,
    display_order INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3.1.4 `user_subscriptions` - Подписки пользователей

Связывает конкретного пользователя с тарифным планом. Хранит данные о дате начала, статусе подписки, возможности автопродления и информации о способе оплаты через ЮKassa.

```
CREATE TABLE user_subscriptions (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    plan_id INTEGER REFERENCES subscription_plans(id),
    payment_id INTEGER REFERENCES payments(id),
    start_date TIMESTAMP NOT NULL,
    end_date TIMESTAMP,
    status VARCHAR(50) DEFAULT 'active',
    auto_renew BOOLEAN DEFAULT FALSE,
    yookassa_payment_method_id VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3.1.5 `translation_jobs` - Задания на перевод

Таблица фиксирует процессы перевода, включая языковые параметры, статус, прогресс, ошибки и метаинформацию. Является центральной для системы обработки переводов.

```
CREATE TABLE translation_jobs (
```

```
    id SERIAL PRIMARY KEY,
    file_id INTEGER REFERENCES files(id),
    status VARCHAR(20) DEFAULT 'pending',
    source_language VARCHAR(10) NOT NULL,
    target_language VARCHAR(10) NOT NULL,
    domain VARCHAR(255),
    total_blocks INTEGER DEFAULT 0,
    processed_blocks INTEGER DEFAULT 0,
    failed_blocks INTEGER DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    started_at TIMESTAMP,
    completed_at TIMESTAMP,
    error_message TEXT,
    retry_count INTEGER DEFAULT 0,
    job_metadata TEXT
);
```

4 ЗАПУСК СИСТЕМЫ

4.1 ПОДГОТОВКА ОКРУЖЕНИЯ

1. Создайте файл `.env` на основе `env.example`:

```
cp env.example .env
```

2. Настройте переменные окружения в `.env`:

```
# База данных
DATABASE_URL=postgresql://user:password@db:5432/umarta_translator
POSTGRES_USER=user
POSTGRES_PASSWORD=password
POSTGRES_DB=umarta_translator

# JWT
SECRET_KEY=<your-secret-key-here>
ACCESS_TOKEN_EXPIRE_MINUTES=30
REFRESH_TOKEN_EXPIRE_DAYS=7

# SMTP
SMTP_HOST=smtp.mail.ru
SMTP_PORT=465
SMTP_USER=<your-email-username >
SMTP_PASSWORD=<your-app-password>
EMAILS_FROM_EMAIL=<your-email >
EMAILS_FROM_NAME=Translator

# LLM провайдеры
DEEPSEEK_ENABLED=true
DEEPSEEK_API_KEY=<your-deepseek-key>
DEEPSEEK_MODEL=deepseek-chat
OPENAI_ENABLED=false
OPENAI_API_KEY=<your-openai-key>
OPENAI_MODEL=gpt-4-turbo
LMSTUDIO_ENABLED=false
LMSTUDIO_MODEL=<your-local-model>
GIGACHAT_ENABLED=true
GIGACHAT_API_KEY=<your-api-key>
GIGACHAT_API_URL =https://gigachat.devices.sberbank.ru/api/v1
GIGACHAT_AUTH_URL =https://ngw.devices.sberbank.ru:9443/api/v2/oauth
GIGACHAT_SCOPE =GIGACHAT_API_CORP
GIGACHAT_MODEL =GigaChat-2
GIGACHAT_MAX_CONCURRENT=20
GIGACHAT_TIMEOUT=30

# Файлы
FILES_DIR=/app/files
ALLOWED_EXTENSIONS=.txt,.docx,.xlsx,.pptx
MAX_FILES=10
```

```
FRONTEND_MAX_FILE_SIZE_MB=10
BACKEND_MAX_FILE_SIZE_MB=50
NGINX_MAX_TOTAL_SIZE_MB=300

# ЮKassa
YOOKASSA_SHOP_ID=<your-shop-id>
YOOKASSA_SECRET_KEY=<your-secret-key>
YOOKASSA_RETURN_URL=http://localhost/payment-success

# Домен
SERVER_DOMAIN=localhost
SERVER_HOST=http://localhost
CLIENT_HOST=http://localhost
```

4.2 Запуск через Docker Compose

```
# Запуск всех сервисов
docker-compose up -d

# Просмотр логов
docker-compose logs -f

# Остановка
docker-compose down
```

4.3 Структура сервисов

Система развёртывается с использованием Docker Compose и включает несколько изолированных сервисов, взаимодействующих друг с другом в рамках одного контура. Ниже приведено краткое описание каждого из них:

- **backend**: FastAPI приложение на порту 8000.
- **frontend**: React приложение (собирается в статику).
- **nginx**: Веб-сервер на порту 80.
- **db**: PostgreSQL на порту 5432.

5 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

5.1 СОЗДАНИЕ АДМИНИСТРАТОРА

Для создания первого администратора используйте скрипт:

```
# Войдите в контейнер backend
docker exec -it backend bash

# Запустите скрипт создания администратора
python -m app.init_admin
```

5.2 API ДЛЯ УПРАВЛЕНИЯ ПОЛЬЗОВАТЕЛЯМИ

5.2.1 Получение списка пользователей

Для получения списка всех зарегистрированных пользователей выполните следующий запрос:

```
curl -X GET "http://localhost/api/auth/users" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

5.2.2 Удаление пользователя

Чтобы удалить пользователя по его идентификатору (`user_id`), используйте следующую команду:

```
curl -X DELETE "http://localhost/api/auth/users/{user_id}" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

5.2.3 Разблокировка пользователя

Для разблокировки ранее заблокированного пользователя выполните следующий запрос:

```
curl -X POST "http://localhost/api/auth/admin/unblock/{user_id}" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

6 УПРАВЛЕНИЕ ТАРИФНЫМИ ПЛАНАМИ

6.1 Создание тарифного плана

Для создания нового тарифного плана выполните следующий запрос. Укажите основные параметры, такие как название, цены, ограничения и доступность:

```
curl -X POST "http://localhost/api/subscriptions/admin/plans/" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Базовый",
    "description": "Базовый тарифный план",
    "price_monthly": 500.00,
    "price_yearly": 5000.00,
    "char_limit_monthly": 100000,
    "char_limit_daily_free": 1000,
    "max_file_size_mb": 10,
    "max_total_upload_size_mb": 100,
    "is_active": true,
    "is_public": true,
    "currency_code": "RUB"
  }'
```

6.2 Назначение подписки пользователю

Для назначения конкретному пользователю ранее созданного тарифного плана используйте следующий запрос:

```
curl -X POST "http://localhost/api/subscriptions/admin/users/{user_id}/subscriptions/" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": 1,
    "plan_id": 1,
    "start_date": "2024-01-01T00:00:00Z",
    "end_date": "2024-02-01T00:00:00Z",
    "status": "active",
    "auto_renew": false
  }'
```

7 УПРАВЛЕНИЕ ФАЙЛАМИ

7.1 ПРОСМОТР ВСЕХ ФАЙЛОВ (ДЛЯ АДМИНИСТРАТОРА)

Чтобы получить список всех файлов, загруженных в систему, используйте следующий запрос:

```
curl -X GET "http://localhost/api/files" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

7.2 ПРОСМОТР ФАЙЛОВ КОНКРЕТНОГО ПОЛЬЗОВАТЕЛЯ

Для фильтрации файлов по пользователю укажите `user_id` в параметрах запроса:

```
curl -X GET "http://localhost/api/files?user_id={user_id}" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

7.3 УДАЛЕНИЕ ФАЙЛОВ

Для удаления одного или нескольких файлов выполните запрос, указав список `file_ids`:

```
curl -X POST "http://localhost/api/files/delete" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "file_ids": [1, 2, 3]
  }'
```

8 НАСТРОЙКА LLM-ПРОВАЙДЕРОВ

8.1 Глобальные настройки

8.1.1 Получение текущего провайдера

Чтобы узнать, какой провайдер ИИ установлен по умолчанию для всей системы, используйте следующий запрос:

```
curl -X GET "http://localhost/api/admin/settings/llm" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>"
```

8.1.2 Установка глобального провайдера

Для установки провайдера по умолчанию, который будет использоваться всеми пользователями без индивидуальных настроек используйте следующий запрос:

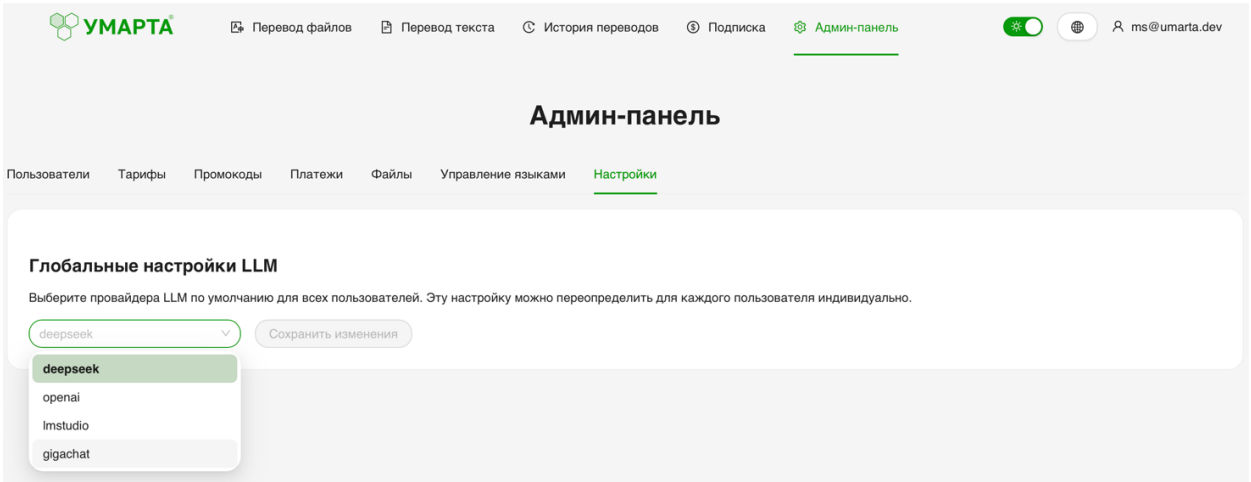
```
curl -X PUT "http://localhost/api/admin/settings/llm" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "provider": "deepseek"
  }'
```

8.2 Персональные настройки пользователя

8.2.1 Установка провайдера для пользователя

Если нужно задать индивидуального LLM-провайдера для пользователя, используйте следующий запрос, либо воспользуйтесь интерфейсом администратора:

```
curl -X PUT "http://localhost/api/auth/admin/users/{user_id}/llm" \
  -H "Authorization: Bearer <YOUR_ADMIN_TOKEN>" \
  -H "Content-Type: application/json" \
  -d '{
    "provider": "openai"
  }'
```



9 МОНИТОРИНГ СИСТЕМЫ

9.1 Проверка здоровья API

Для оперативной проверки, доступен ли backend и работает ли API, выполните следующий запрос:

```
curl -X GET "http://localhost/api/health"
```

9.2 Логи системы

```
# Логи всех сервисов
docker-compose logs -f

# Логи конкретного сервиса
docker-compose logs -f backend
docker-compose logs -f nginx
docker-compose logs -f db
```

9.3 Мониторинг базы данных

```
# Подключение к PostgreSQL
docker exec -it postgres_db psql -U user -d umarta_translator

# Полезные запросы
SELECT COUNT(*) FROM users;
SELECT COUNT(*) FROM files;
SELECT status, COUNT(*) FROM files GROUP BY status;
SELECT COUNT(*) FROM translation_jobs;
```

10 РЕЗЕРВНОЕ КОПИРОВАНИЕ

10.1 Создание бэкапа базы данных

Для создания резервной копии базы данных используется следующая команда:

```
# Создание дампа
docker exec postgres_db pg_dump -U user umarta_translator > backup.sql
```

Для восстановления базы данных из ранее созданного дампа выполните:

```
# Восстановление из дампа
docker exec -i postgres_db psql -U user umarta_translator < backup.sql
```

10.2 Резервное копирование файлов

Для резервного копирования файлов, загруженных пользователями, можно создать архив следующей командой:

```
# Создание архива файлов
tar -czf files_backup.tar.gz backend/files/

# Восстановление файлов
tar -xzf files_backup.tar.gz
```

11 ОБНОВЛЕНИЕ СИСТЕМЫ

11.1 Обновление кода

Для обновления серверной части системы выполните следующие шаги:

```
# Остановка сервисов
docker-compose down

# Обновление кода из репозитория
git pull origin main

# Пересборка контейнеров
docker-compose build

# Запуск обновленной системы
docker-compose up -d
```

11.2 Миграции базы данных

После обновления кода необходимо применить актуальные миграции базы данных. Для этого используйте следующую команду:

```
# Выполнение миграций
docker exec backend alembic upgrade head
```

Если были внесены изменения в модели и требуется создать новую миграцию, выполните:

```
# Создание новой миграции (при изменении моделей)
docker exec backend alembic revision --autogenerate -m "Описание изменений"
```

12 КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

12.1 Основные конфигурационные файлы

- `.env` - переменные окружения;
- `docker-compose.yml` - конфигурация Docker;
- `backend/app/config.py` - настройки приложения;
- `nginx/default.conf.template` - конфигурация Nginx.

12.2 Важные директории

- `backend/files/` - загруженные файлы;
- `backend/migrations/versions/` - миграции базы данных;
- `frontend/src/` - исходный код фронтенда;
- `marketing_pages/` - маркетинговые страницы.

13 УСТРАНЕНИЕ НЕПОЛАДОК

13.1 ТИПИЧНЫЕ ПРОБЛЕМЫ

1. **Контейнер не запускается**
   - Проверьте логи: `docker-compose logs service_name`;
   - Проверьте переменные окружения в `.env`.
2. **База данных недоступна**
   - Убедитесь, что PostgreSQL запущен;
   - Проверьте настройки подключения в `DATABASE_URL`.
3. **Ошибки при переводе**
   - Проверьте настройки LLM провайдеров;
   - Убедитесь, что API ключи корректны.
4. **Проблемы с загрузкой файлов**
   - Проверьте права доступа к директории `backend/files/`;
   - Убедитесь, что размер файлов не превышает лимиты.

13.2 ПОЛЕЗНЫЕ КОМАНДЫ ДЛЯ ДИАГНОСТИКИ

```
# Статус контейнеров
docker-compose ps

# Использование ресурсов
docker stats

# Проверка сетевого подключения
docker exec backend ping db

# Проверка портов
netstat -tlnp | grep :80
netstat -tlnp | grep :5432
```

14 БЕЗОПАСНОСТЬ

14.1 Рекомендации по безопасности

- Регулярно обновляйте пароли и ключи.
- Используйте HTTPS в продакшене.
- Настройте файрвол для ограничения доступа.
- Регулярно создавайте резервные копии.
- Мониторьте логи на предмет подозрительной активности.

14.2 Настройка HTTPS (для продакшена)

Для настройки HTTPS необходимо:

1. Получить SSL сертификат.
2. Обновить конфигурацию Nginx.
3. Изменить переменные окружения для использования HTTPS.

15 ПОДДЕРЖКА

При возникновении проблем:

1. Проверьте логи системы.
2. Убедитесь в корректности конфигурации.
3. Проверьте статус всех сервисов.
4. При необходимости обратитесь к разработчикам.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Перечень сокращений, которые используются в настоящем документе, представлен в таблице:

Таблица 1

Сокращение	Расшифровка
API	Application Programming Interface — интерфейс прикладного программирования
DB	Database — база данных
HTTP	HyperText Transfer Protocol — протокол передачи гипертекста
HTTPS	HyperText Transfer Protocol Secure — защищённый протокол HTTP
ID	Identifier — уникальный идентификатор
JSON	JavaScript Object Notation — формат обмена данными
JWT	JSON Web Token — веб-токен в формате JSON
LLM	Large Language Model — большая языковая модель
SQL	Structured Query Language — язык структурированных запросов
UI	User Interface — пользовательский интерфейс
UI/UX	User Interface / User Experience — интерфейс и пользовательский опыт

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (стр.) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
	измененных	замененных	новых	аннулированных					