Инструкция по установке и обновлению "**Протокол**Эксперт"

Backend-сервис, база данных PostgreSQL и файловое хранилище протоколов/аудиозаписей

1. Введение

Документ содержит порядок установки, первичной настройки, запуска, проверки работоспособности и обновления экземпляра программного обеспечения ProtocolExpert.

ProtocolExpert является backend-сервисом для распознавания аудиозаписей, генерации протоколов с использованием LLM и отправки результатов по электронной почте.

2. Служба технической поддержки

Связаться со службой технической поддержки можно по электронной почте support@rootcode.ru.

3. Область применения

Программное обеспечение ProtocolExpert предназначено для автоматизации протоколирования совещаний, встреч и иных устных коммуникаций на основе аудиозаписей.

Система выполняет загрузку аудио, распознавание речи, формирование структурированного протокола, сохранение результата и отправку протокола по email.

4. Краткое описание возможностей

- регистрация и авторизация пользователей;
- подтверждение email и восстановление пароля;
- загрузка аудиозаписей через API;
- транскрибация аудио через faster-whisper large-v3;
- поддержка CPU и CUDA-режимов ASR;
- нарезка длинных аудиофайлов на CUDA-чанки с перекрытием;
- генерация протоколов через RouterAI/OpenAI-compatible API;
- сохранение HTML/TXT/PDF-протоколов;
- отправка протоколов и аудиозаписей через Microsoft Graph API;
- встроенная очередь задач обработки протоколов;
- хранение пользователей, ролей и задач обработки в PostgreSQL.

5. Условия установки и эксплуатации

5.1. Требования к программному обеспечению

- Операционная система Linux: Ubuntu 20.04/22.04/24.04 LTS, Debian 11/12 или другой современный Linux-дистрибутив.
- Docker Engine актуальной версии.
- Docker Compose v2.
- Доступ к сети Интернет для загрузки Docker-образов, Python-зависимостей, модели faster-whisper и обращения к внешним API.
- Для GPU-режима: установленный драйвер NVIDIA, CUDA runtime/toolkit и NVIDIA Container Toolkit.
- Для пользователей: современный браузер на основе Chromium, Firefox или Safari.

5.2. Аппаратные требования

- CPU: 64-битный процессор x86_64, рекомендуется 4 ядра и выше.
- RAM: минимум 8 ГБ, рекомендуется 16 ГБ и выше при обработке длинных записей.
- Диск: минимум 30 ГБ свободного пространства, объем зависит от количества аудиозаписей и протоколов.
- SSD рекомендуется для ускорения работы с аудиофайлами, базой данных и кэшем моделей.
- GPU: NVIDIA GPU рекомендуется для ускорения ASR; для модели large-v3 желательно 8 ГБ VRAM и выше.

5.3. Сетевые требования

- Открытый внешний порт приложения, соответствующий WEB_HOST_PORT.
- Внутренний порт backend по умолчанию: 9777.
- PostgreSQL публикуется через POSTGRES_HOST_PORT, если требуется доступ снаружи хоста.
- Исходящий HTTPS-доступ к RouterAI, Microsoft Graph API, Infisical и Hugging Face.
- При production-развертывании рекомендуется использовать reverse proxy с HTTPS, например Nginx или Traefik.

6. Подготовка сервера

6.1. Установка Docker и Docker Compose

Установите Docker Engine и Docker Compose согласно официальной документации Docker для выбранного Linux-дистрибутива.

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
```

```
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo $VERSION_CODENAME)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
docker --version
```

6.2. Поддержка GPU

Если планируется запуск ASR на CUDA, установите драйвер NVIDIA и NVIDIA Container Toolkit.

```
sudo apt install -y nvidia-driver-550
sudo reboot
nvidia-smi
```

Для контейнеров установите NVIDIA Container Toolkit:

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o
/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-
toolkit.list | sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-
toolkit-keyring.gpg] https://#g' | sudo tee /etc/apt/sources.list.d/nvidia-container-
toolkit.list
sudo apt update
sudo apt install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
docker run --rm --gpus all nvidia/cuda:12.2.0-base-ubuntu22.04 nvidia-smi
```

Если GPU не используется, установите ASR_DEVICE=cpu в .env.

7. Получение дистрибутива

Скопируйте архив или директорию проекта на сервер, например в /opt/protocolexpert.

```
sudo mkdir -p /opt/protocolexpert
sudo chown -R $USER:$USER /opt/protocolexpert
cp protocolexpert.tar.gz /opt/protocolexpert/
cd /opt/protocolexpert
tar -zxf protocolexpert.tar.gz --strip-components=1
```

Если проект поставляется как git-репозиторий, используйте git clone или git pull согласно регламенту поставки.

8. Настройка переменных окружения

Создайте .env на основе .env.template и заполните обязательные значения.

```
cd /opt/protocolexpert
cp .env.template .env
vim .env
```

Переменная	Описание
APP_IMAGE	Docker-образ backend-приложения, например registry.example/protocolexpert:latest или локально собранный образ.
WEB_PORT	Порт uvicorn внутри контейнера, по умолчанию 9777.
WEB_HOST_PORT	Порт хоста, опубликованный наружу, например 9777.
POSTGRES_DB	Имя базы данных PostgreSQL.
POSTGRES_USER	Пользователь PostgreSQL.
POSTGRES_PASSWORD	Пароль пользователя PostgreSQL.
POSTGRES_HOST_PORT	Порт PostgreSQL на хосте, если требуется публикация.
DATABASE_URL	Строка подключения backend к PostgreSQL.
JWT_SECRET_KEY	Секретный ключ для подписи JWT.
ROUTERAI_API_KEY	API-ключ RouterAI для генерации протоколов.
AZURE_CLIENT_ID	Client ID приложения Azure AD для Microsoft Graph.
AZURE_CLIENT_SECRET	Client Secret приложения Azure AD.
AZURE_TENANT_ID	Tenant ID Azure AD.
SENDER_EMAIL	Email-адрес отправителя.
APP_URL	Публичный URL сервиса для ссылок в письмах.
ADMIN_LOGIN / ADMIN_PASSWORD	Логин и пароль встроенного администратора без истечения доступа.
ASR_DEVICE	cpu, cuda или auto.
ASR_COMPUTE_TYPE	Тип вычислений ASR, например int8, float16, int8_float16.

8.1. Пример .env для Docker-развертывания

```
APP_IMAGE=protocolexpert:latest
WEB_PORT=9777
WEB_HOST_PORT=9777

POSTGRES_DB=protocolexpert
POSTGRES_USER=protocolexpert
POSTGRES_PASSWORD=change-me
POSTGRES_HOST_PORT=5432
DATABASE_URL=postgresql+psycopg://protocolexpert:change-me@protocolexpert-
postgres:5432/protocolexpert

JWT_SECRET_KEY=change-me-long-random-secret
APP_URL=https://app.protocolexpert.ru
```

```
ASR_DEVICE=cpu
ASR_COMPUTE_TYPE=int8
ASR_CPU_COMPUTE_TYPE=int8
ASR_BEAM_SIZE=5
ASR_NUM_WORKERS=1
ASR_CHUNK_SECONDS=300
ASR_CHUNK_OVERLAP_SECONDS=10
ASR_CHUNK_DEDUP_MAX_WORDS=80
ASR_RETRY_WITHOUT_VAD_ON_EMPTY=true
ASR_DEBUG_LOGS=false

ROUTERAI_API_KEY=change-me
AZURE_CLIENT_ID=change-me
AZURE_CLIENT_SECRET=change-me
AZURE_TENANT_ID=change-me
SENDER_EMAIL=robot@example.com

ADMIN_LOGIN=AdMin
ADMIN_PASSWORD=change-me
```

9. Сборка Docker-образа приложения

Если готовый образ не поставляется, соберите backend-образ локально на сервере.

```
cd /opt/protocolexpert
docker build -t protocolexpert:latest .
```

Значение APP_IMAGE в .env должно соответствовать имени собранного или полученного образа.

10. Запуск сервиса и базы данных

docker-compose.template содержит два сервиса: protocolexpert-postgres и protocolexpert-web.

```
cd /opt/protocolexpert
docker compose -f docker-compose.template up -d protocolexpert-postgres
docker compose -f docker-compose.template up -d protocolexpert-web
```

Также можно запустить оба сервиса одной командой:

```
docker compose -f docker-compose.template up -d
```

При первом запуске backend создаст таблицы в PostgreSQL через SQLAlchemy Base.metadata.create_all и выполнит встроенную инициализацию ролей.

11. Проверка работоспособности

1. Проверьте статус контейнеров.

2. Проверьте логи backend-приложения.
3. Проверьте health endpoint.
4. Проверьте доступность базы данных и создание таблиц.

```
docker compose -f docker-compose.template ps
docker compose -f docker-compose.template logs -f protocolexpert-web
curl http://localhost:9777/health
docker compose -f docker-compose.template exec protocolexpert-postgres psql -U
$POSTGRES_USER -d $POSTGRES_DB -c '\dt'
```

Успешный ответ health endpoint:

```
{"status":"ok"}
```

12. Проверка пользовательского сценария

5. Откройте frontend или API-клиент.
6. Выполните регистрацию пользователя.
7. Подтвердите email по ссылке из письма.
8. Выполните вход через /auth/login.
9. Отправьте аудиофайл на /generate-protocol.
10. Проверьте появление файла протокола в директории protocols.
11. Отправьте протокол через /send-protocol и проверьте получение письма.

13. Файловые директории и volumes

Путь	**Назначение**
./postgres-data	Постоянное хранилище PostgreSQL.
./recordings	Сохраненные аудиозаписи пользователей.
./protocols	Сохраненные протоколы HTML/TXT/PDF.
./hf-cache	Кэш моделей Hugging Face внутри контейнера.

Эти директории необходимо включать в резервное копирование при эксплуатации сервиса.

14. Резервное копирование базы данных

Для создания резервной копии PostgreSQL используйте pg_dump из контейнера базы данных.

```
mkdir -p backups
docker compose -f docker-compose.template exec -T protocolexpert-postgres pg_dump -U
$POSTGRES_USER $POSTGRES_DB > backups/protocolexpert_$(date +%Y%m%d_%H%M%S).sql
```

Для восстановления используйте psql:

```
cat backups/protocolexpert_YYYYMMDD_HHMMSS.sql | docker compose -f docker-
compose.template exec -T protocolexpert-postgres psql -U $POSTGRES_USER -d $POSTGRES_DB
```

15. Остановка и перезапуск

```
docker compose -f docker-compose.template restart protocolexpert-web
docker compose -f docker-compose.template stop
docker compose -f docker-compose.template up -d
```

Команда down удаляет контейнеры, но не удаляет volumes/директории проекта. Перед использованием down убедитесь, что данные сохранены.

```
docker compose -f docker-compose.template down
```

16. Порядок обновления

12. Сделайте резервную копию базы данных и файловых директорий.
13. Получите новую версию исходного кода или Docker-образа.
14. При необходимости пересоберите Docker-образ.
15. Обновите APP_IMAGE в .env, если используется новый тег образа.
16. Перезапустите сервисы через docker compose up -d.
17. Проверьте логи и health endpoint.

```
docker compose -f docker-compose.template pull
docker compose -f docker-compose.template up -d
docker compose -f docker-compose.template logs -f protocolexpert-web
curl http://localhost:9777/health
```

17. Диагностика типовых проблем

Проблема	**Решение**
Backend не стартует из-за DATABASE_URL	Проверьте DATABASE_URL в .env и доступность контейнера PostgreSQL.
Не отправляются письма	Проверьте Azure AD параметры, SENDER_EMAIL и права Microsoft Graph Mail.Send/Mail.ReadWrite.
Модель долго загружается	Проверьте доступ к Hugging Face и наличие hf-cache volume.
CUDA не используется	Проверьте ASR_DEVICE=cuda, nvidia-smi, NVIDIA Container Toolkit и наличие CUDA/cuBLAS/cuDNN библиотек.
Протокол не формируется	Проверьте ROUTERAI_API_KEY, доступ к RouterAI и логи protocolexpert-web.
Нет доступа после пробного периода	Проверьте users.access_expires_at или используйте env-admin из ADMIN_LOGIN/ADMIN_PASSWORD.

18. Минимальный checklist развертывания

18. Установлены Docker и Docker Compose.
19. При необходимости настроена NVIDIA GPU-поддержка.
20. Проект скопирован в /opt/protocolexpert.
21. .env создан и заполнен.
22. Docker-образ приложения доступен локально или в registry.
23. Запущены protocolexpert-postgres и protocolexpert-web.
24. Команда docker compose ps показывает running/healthy containers.
25. GET /health возвращает {"status":"ok"}.
26. Проверены регистрация, вход, генерация протокола и отправка email.