Файлы

- **main.py**: Основной файл приложения (FastAPI).
- **models.py**: Описание модели (ORM-класс) для таблицы в SQLite.
- **database.py**: Настройка подключения к базе данных и создание базового класса моделей.
- **requirements.txt**: Список зависимостей.
- **Dockerfile**: Описание образа для сборки Docker-контейнера.

database.py

- Используем create_engine для подключения к SQLite.
- SessionLocal будет использоваться для получения сессии БД в эндпоинтах через зависимость.
- Base — базовый класс для всех моделей SQLAlchemy.

models.py

- URLItem — модель для таблицы short_urls.
- id — первичный ключ.
- short_id — уникальный короткий идентификатор.
- full_url — исходная длинная ссылка.

Base берется из database.py. При первом запуске main.py вызовет Base.metadata.create_all(engine), создав таблицу short_urls, если она не существует.

main.py

- Импортируем необходимые модули.
- Создаем приложение FastAPI.
- URLCreate — Pydantic-модель для входных данных при создании короткой ссылки. HttpUrl обеспечивает валидацию, что url - корректный URL.
- get_db() — зависимость для получения сессии БД. Используем yield для автоматического закрытия сессии после обработки запроса.
- generate_short_id() - Функция генерирует случайный короткий ID длиной 6 символов (буквы и цифры).
- shorten_url():
    - При POST /shorten мы принимаем URLCreate с длинным URL.
    - Генерируем до 10 попыток короткий ID, проверяем в БД, нет ли коллизий.
    - Если нет, создаем запись в БД, коммитим изменения и возвращаем короткую ссылку.
    - Если не удалось найти уникальный ID, возвращаем 500 ошибку.
- redirect_to_full():
    - При GET /{short_id} проверяем, есть ли такая запись в БД.
    - Если есть, возвращаем RedirectResponse на full_url.

    - Если нет, 404 ошибка.
- get_stats() - GET /stats/{short_id}: Возвращает информацию о сокращенной ссылке без редиректа. Можно расширить логику для статистики (количество переходов и т.д.).

Dockerfile

- FROM python:3.9-slim: Используем легкий образ Python.
- WORKDIR /app: Переключаемся в рабочую директорию.
- COPY requirements.txt . и RUN pip install: Устанавливаем зависимости.
- COPY . .: Копируем все файлы приложения в образ.
- VOLUME ["/app/data"]: Объявляем том для каталога /app/data. БД url.db будет храниться там, чтобы данные сохранялись между перезапусками контейнера.
- CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]: Запускаем приложение через uvicorn.

Команды для тестирования

Создание новой сокращенной ссылки:

```
curl -X POST http://localhost:8001/shorten \
 -H "Content-Type: application/json" \
 -d '{"url":"https://www.example.com"}'
```

- -X POST — запрос на создание.
- -H "Content-Type: application/json" — данные отправляются в формате JSON.
- В -d передаем тело запроса с полем url.

Переход по сокращенной ссылке:

```
curl -i http://localhost:8000/shorturl
```

- -i показывает заголовки ответа.
- Если ссылка найдена, вы получите статус-код 307 или 302 с заголовком Location: https://www.example.com, что означает перенаправление.