Отчет по отладке приложения Федоров Дмитрий Александрович

1. Шаг 1: Запуск и исправление бага №1.

- Что сделал: запустил docker compose up и получил ошибку с рисунка 1.

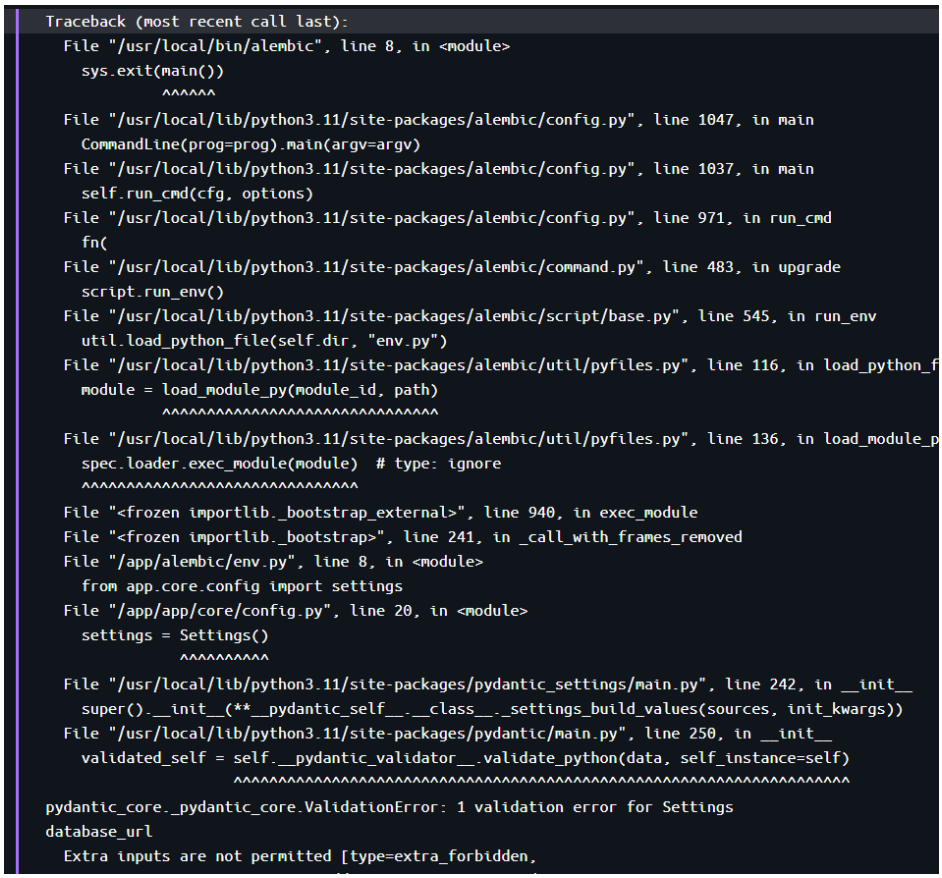

Рисунок 1 – Ошибка при запуске docker compose

- Проблема: Контейнер с приложением падает сразу после запуска из-за ошибки в конфиге.

- Решение: Проблема заключалась в validation_alias, в файле core/config.py в строчке 14 вместо псевдонима DATABASE_URL было написано DATABSE_URL.

2. Шаг 2: Исправление бага №2.

- Проблема: Несуществующая версия fastapi.

- Файл и строка: requirements.txt на 10 строке (рисунок 2).

Рисунок 2 – Несуществующая версия fastapi

- Решение: Удаление этой строчки, чтобы не возникало багов из-за версий.

3. Шаг 3: Исправление бага №3

- Проблема: Парсинг данных каждые 5 секунд, вместо 5 минут.

- Файл и строка: services/scheduler.py 13 строка (рисунок 3 и 4).

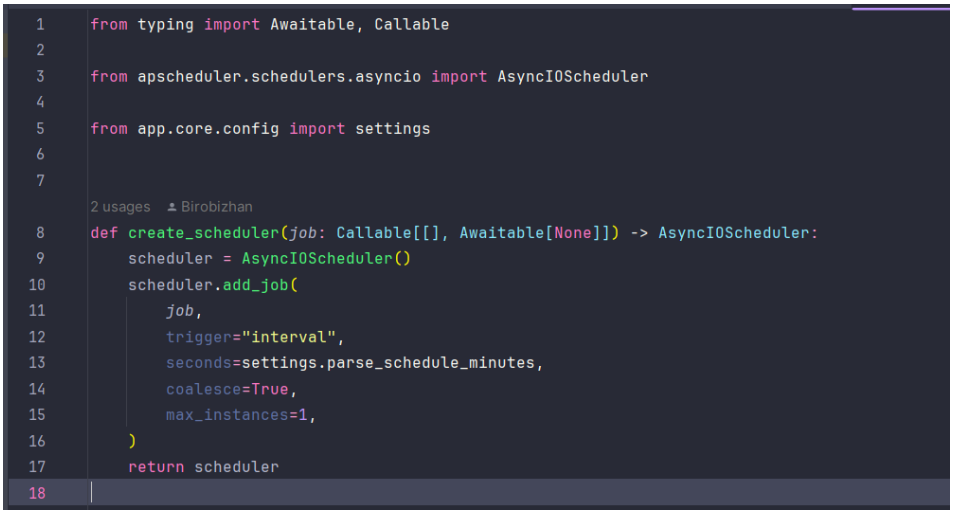

Рисунок 3 – Неправильный параметр при создании задачи на парсинг

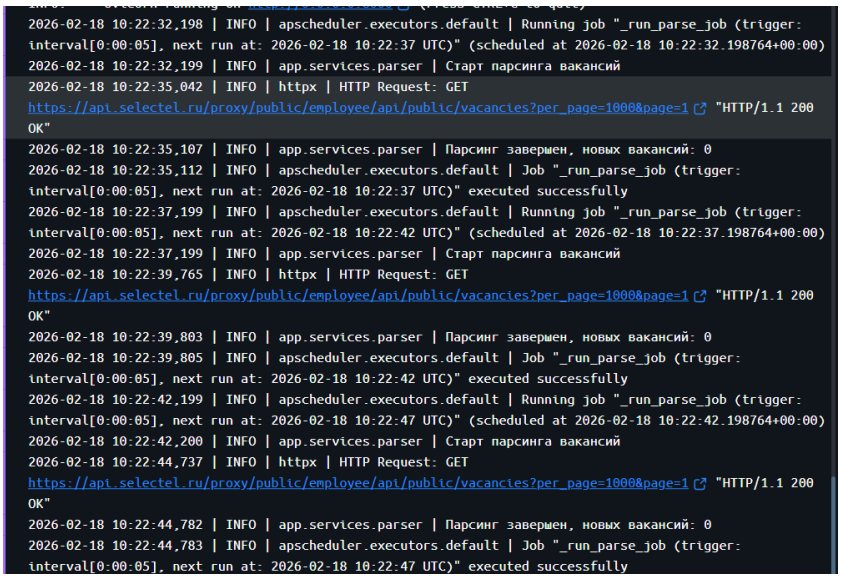

Рисунок 4 – Парсинг вакансий каждые 5 секунд

- Решение: замена параметра seconds на minutes.

4. Шаг 4: Исправление бага №4.

- Проблема: Ошибка при парсинге: Ошибка фонового парсинга 'NoneType' object has no attribute 'name' (рисунок 5).

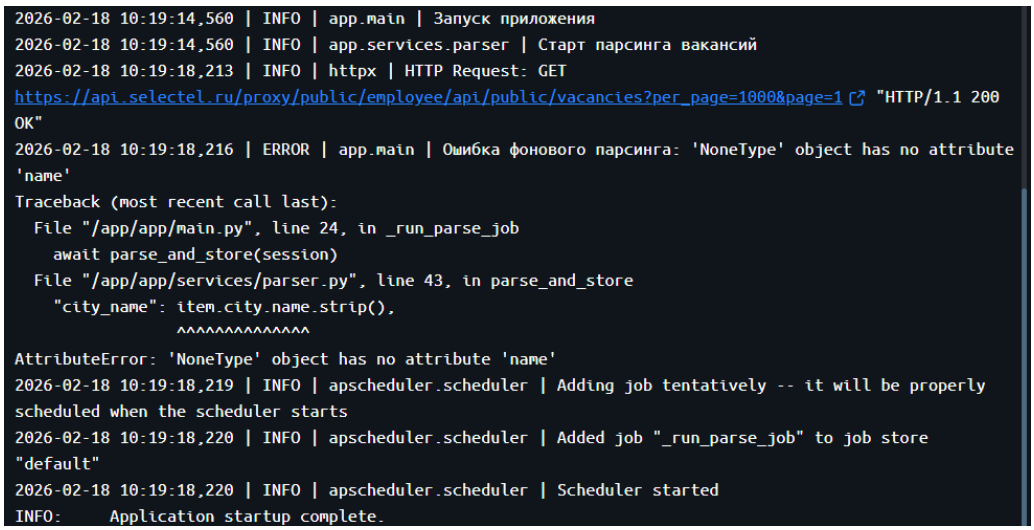

Рисунок 5 – Ошибка 'NoneType' object has no attribute 'name'

- Файл и строка: services/parser.py 43 строка (рисунок 6).

Рисунок 6 – Проблема в файле parser.py

- Решение: баг связан с тем, что не у всех вакансий есть поле city, решением проблемы стало изменение строки "city_name": item.city.name.strip(), на "city_name": item.city.name.strip() if item.city else 'Не указано', или можно заменить 'Не указано' на None.

5. Шаг 5: Исправление бага №5.

- Проблема: Возврат JSONResponse со статус кодом 200 вместо вызова HTTPException со статус кодом 400 или 409.

- Файл и строка: api/v1/vacancies.py строки 52–54 (рисунок 7).

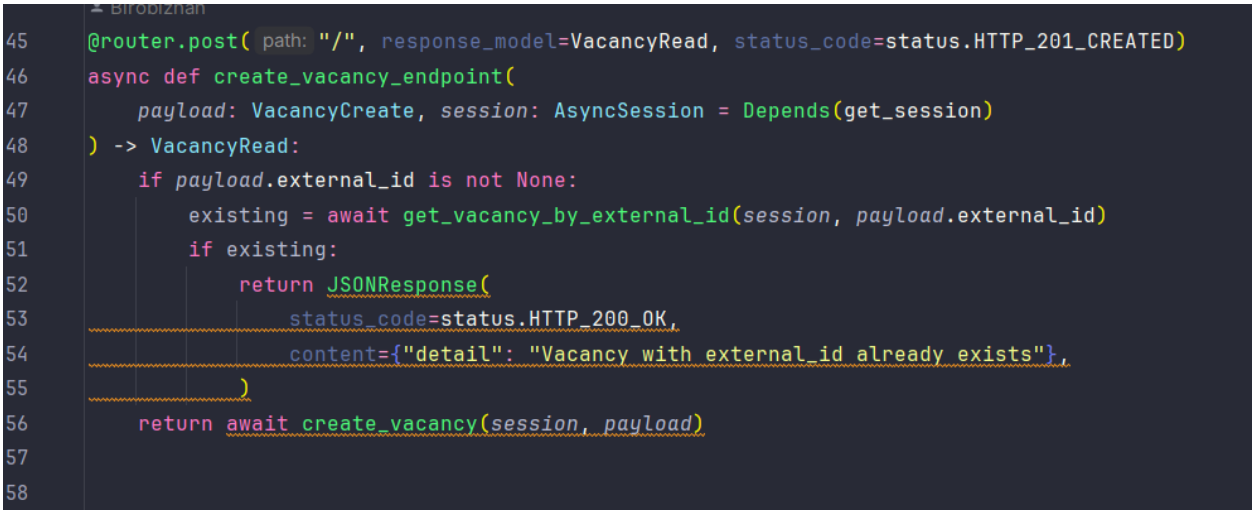

Рисунок 7 – Ошибка возврата json вместо вызова ошибки

- Решение: Изменение строки

return JSONResponse(status_code=status.HTTP_200_OK, content={"detail": "Vacancy with external_id already exists"},) на raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST, detail="Vacancy with this external_id already exists")

6. Шаг 6: Исправление бага №6.

- Проблема: название базы данных в database_url.

- Файл и строка: core/config.py строка 13 (рисунок 8).

```
1   from pydantic import Field
2   from pydantic_settings import BaseSettings, SettingsConfigDict
3
4
    1 usage   Birobizhan
5   class Settings(BaseSettings):
6       model_config = SettingsConfigDict(
7           env_file=".env",
8           env_file_encoding="utf-8",
9           case_sensitive=False,
10      )
11
12      database_url: str = Field(
13          "postgresql+asyncpg://postgres:postgres@db:5432/postgres_typo",
14          validation_alias="DATABASE_URL",
15      )
16      log_level: str = "INFO"
17      parse_schedule_minutes: int = 5
18
19
20  settings = Settings()
```

Рисунок 8 – Ошибка в database_url

- Решение: хоть это ошибка и не мешает прямо сейчас, но при отсутствии файла .env будет подставляться совсем другая база данных с названием postgres_typo вместо postgres.

Шаг 7: Исправление бага №7.

- Проблема: Незакрытое соединение в httpx клиенте.

- Файл и строка: services/parser.py строка 31 (рисунок 9).

```
    timeout = httpx.Timeout( timeout: 10.0, read=20.0)
    try:
        client = httpx.AsyncClient(timeout=timeout)
        page = 1
        while True:
            payload = await fetch_page(client, page)
            parsed_payloads = []
            for item in payload.items:
                parsed_payloads.append(
                    {
                        "external_id": item.id,
                        "title": item.title,
                        "timetable_mode_name": item.timetable_mode.name,
                        "tag_name": item.tag.name,
                        "city_name": item.city.name.strip() if item.city else 'Не указано',
                        "published_at": item.published_at,
                        "is_remote_available": item.is_remote_available,
                        "is_hot": item.is_hot,
                    }
                )

            created_count = await upsert_external_vacancies(session, parsed_payloads)
            created_total += created_count

            if page >= payload.page_count:
                break
            page += 1
    except (httpx.RequestError, httpx.HTTPStatusError) as exc:
```

Рисунок 9 – Незакрытое соединение httpx клиента

- Решение: изменил строку client = httpx.AsyncClient(timeout=timeout)

На async with httpx.AsyncClient(timeout=timeout) as client: потому что в первом варианте не закрывается соединение и это приводит к утечке соединений и ресурсов (рисунок 10).

```
    async with httpx.AsyncClient(timeout=timeout) as client:
        page = 1
        while True:
            payload = await fetch_page(client, page)
            parsed_payloads = []
            for item in payload.items:
                parsed_payloads.append(
                    {
                        "external_id": item.id,
                        "title": item.title,
                        "timetable_mode_name": item.timetable_mode.name,
                        "tag_name": item.tag.name,
                        "city_name": item.city.name.strip() if item.city else 'Не указано',
                        "published_at": item.published_at,
                        "is_remote_available": item.is_remote_available,
                        "is_hot": item.is_hot,
                    }
                )

            created_count = await upsert_external_vacancies(session, parsed_payloads)
            created_total += created_count

            if page >= payload.page_count:
                break
            page += 1
```

Рисунок 10 – Исправленное подключение через async with

8. Шаг 8: Улучшение программы.

- Проблема: N+1 запросов к базе данных.

- Файл и строка: crud/vacancy.py строки после 62 (рисунок 11).

```python
async def upsert_external_vacancies(
    session: AsyncSession, payloads: Iterable[dict]
) -> int:
    external_ids = [payload["external_id"] for payload in payloads if payload["external_id"]]
    if external_ids:
        existing_result = await session.execute(
            select(Vacancy.external_id).where(Vacancy.external_id.in_(external_ids))
        )
        existing_ids = set(existing_result.scalars().all())
    else:
        existing_ids = {}

    created_count = 0
    for payload in payloads:
        ext_id = payload["external_id"]
        if ext_id and ext_id in existing_ids:
            result = await session.execute(
                select(Vacancy).where(Vacancy.external_id == ext_id)
            )
            vacancy = result.scalar_one()
            for field, value in payload.items():
                setattr(vacancy, field, value)
        else:
            session.add(Vacancy(**payload))
            created_count += 1

    await session.commit()
    return created_count
```

Рисунок 11 – Проблема N+1 запросов к БД

- Решение: Переработка функции upsert_external_vacancies таким образом, чтобы выбирались сразу все записи с помощью in и потом уже происходит вставка или обновление данных (рисунок 12).

```python
async def upsert_external_vacancies(
    session: AsyncSession, payloads: Iterable[dict]
) -> int:
    payload_map = {p["external_id"]: p for p in payloads if p.get("external_id")}
    if not payload_map:
        return 0

    stmt = select(Vacancy).where(Vacancy.external_id.in_(payload_map.keys()))
    result = await session.execute(stmt)
    existing_vacancies = {v.external_id: v for v in result.scalars().all()}

    created_count = 0
    for ext_id, payload in payload_map.items():
        if ext_id in existing_vacancies:
            vacancy = existing_vacancies[ext_id]
            for field, value in payload.items():
                setattr(vacancy, field, value)
        else:
            session.add(Vacancy(**payload))
            created_count += 1

    await session.commit()
    return created_count

```

Рисунок 12 – Исправление ошибки N+1

9. Шаг 9: Потенциальная проблема/баг №8.

- Проблема: unique ограничение на nullable=true поле external_id в модели базы данных.

- Файл и строка: models/vacancy строки 11 и 26 (рисунок 13).

Рисунок 13 – Проблема с unique и nullable=True

- Решение: убрать __table_args__ и изменить поле external_id так, как показано на рисунке 14 и применить миграции через alembic.

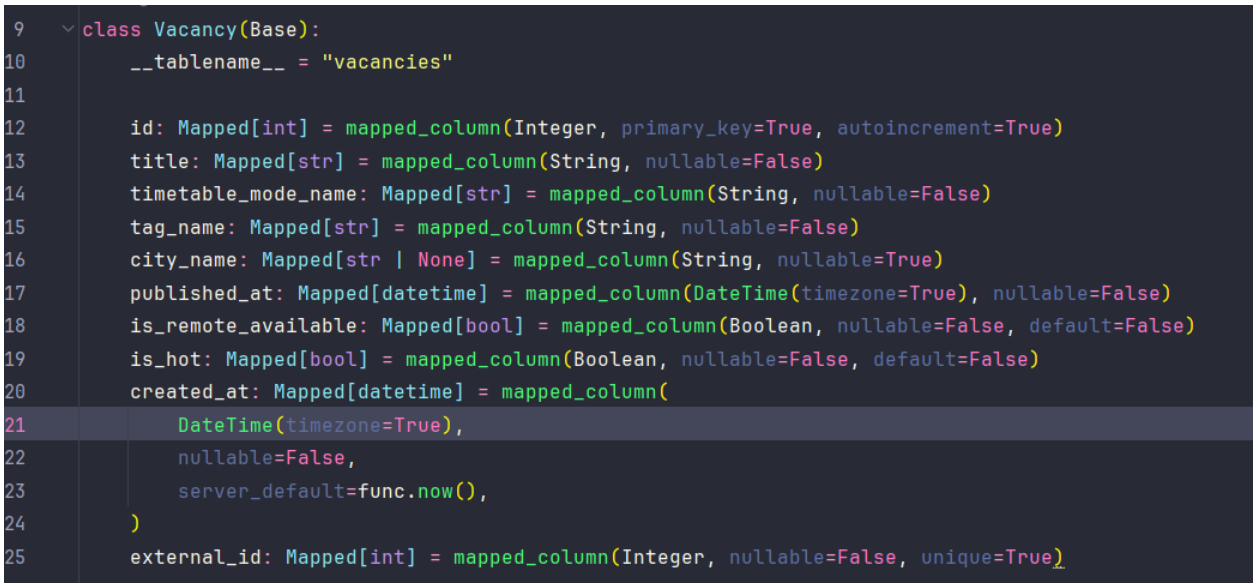

Рисунок 14 – Изменение модели вакансии

Итоги:

- Все эндпоинты работают корректно (рисунки 15–22).

- Приложение возвращает корректные HTTP-статусы и данные.

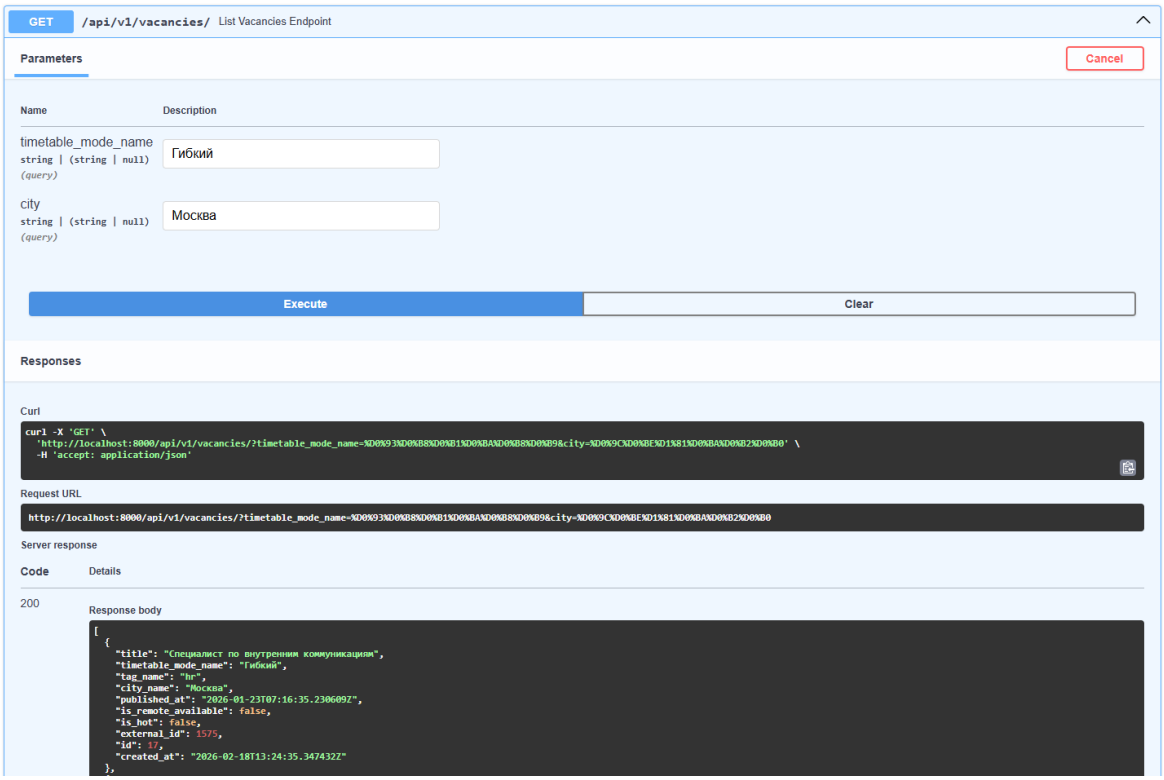

Рисунок 15 – Работающий эндпоинт с получением всех вакансий по фильтрам (город и график)

Рисунок 16 – Создание новой вакансии

Рисунок 17 – Попытка создать вакансию с существующим external_id

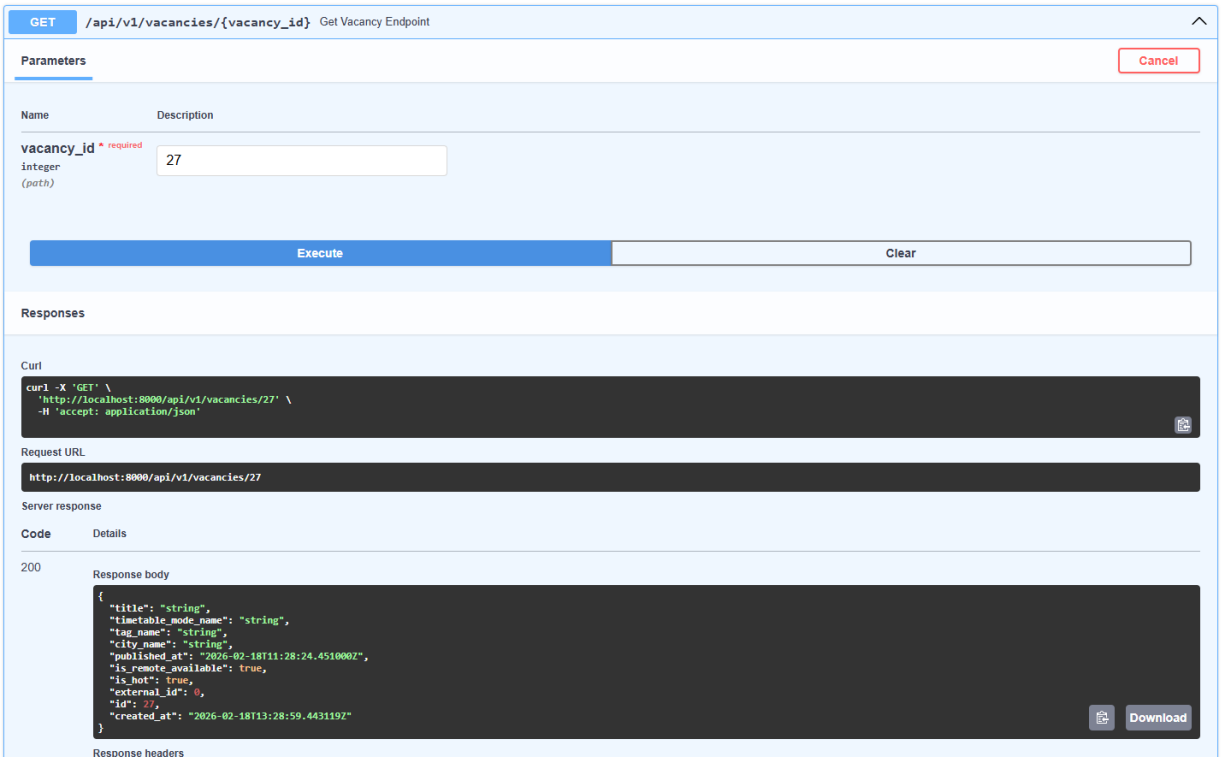

Рисунок 18 – Получение вакансии по id

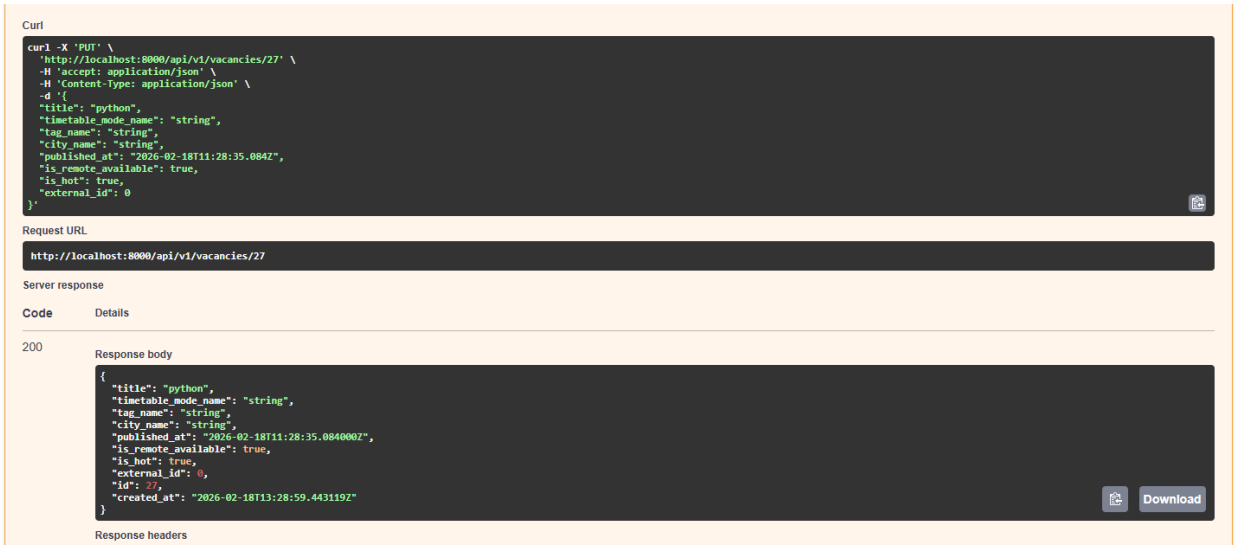

Рисунок 19 – Изменение названия вакансии

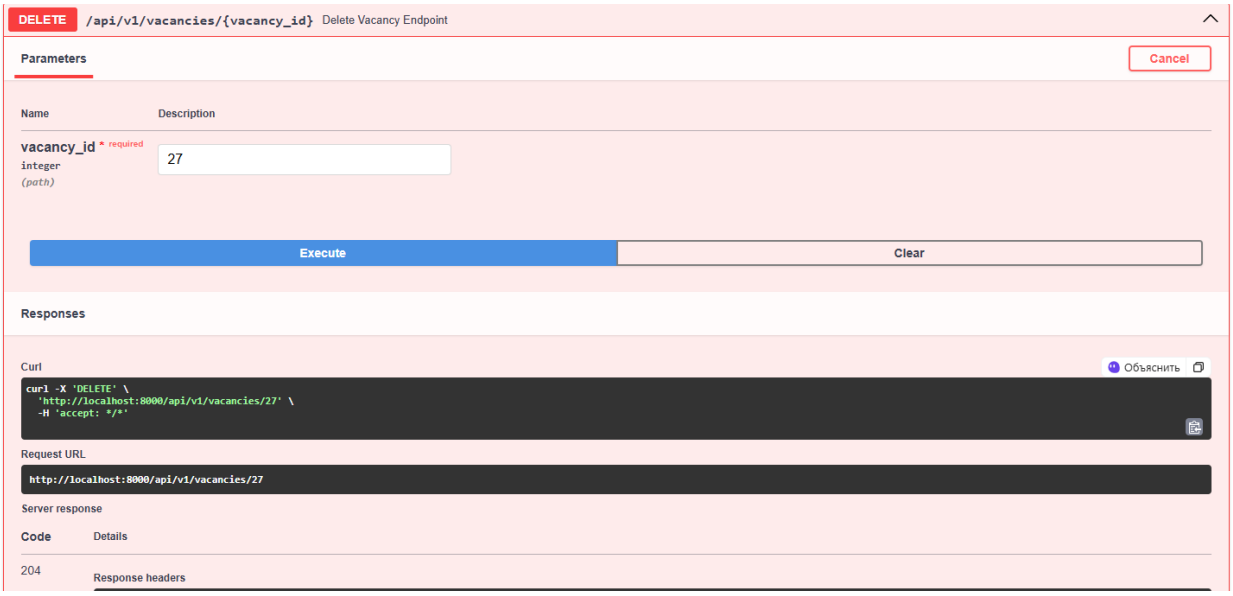

Рисунок 20 – Удаление вакансии

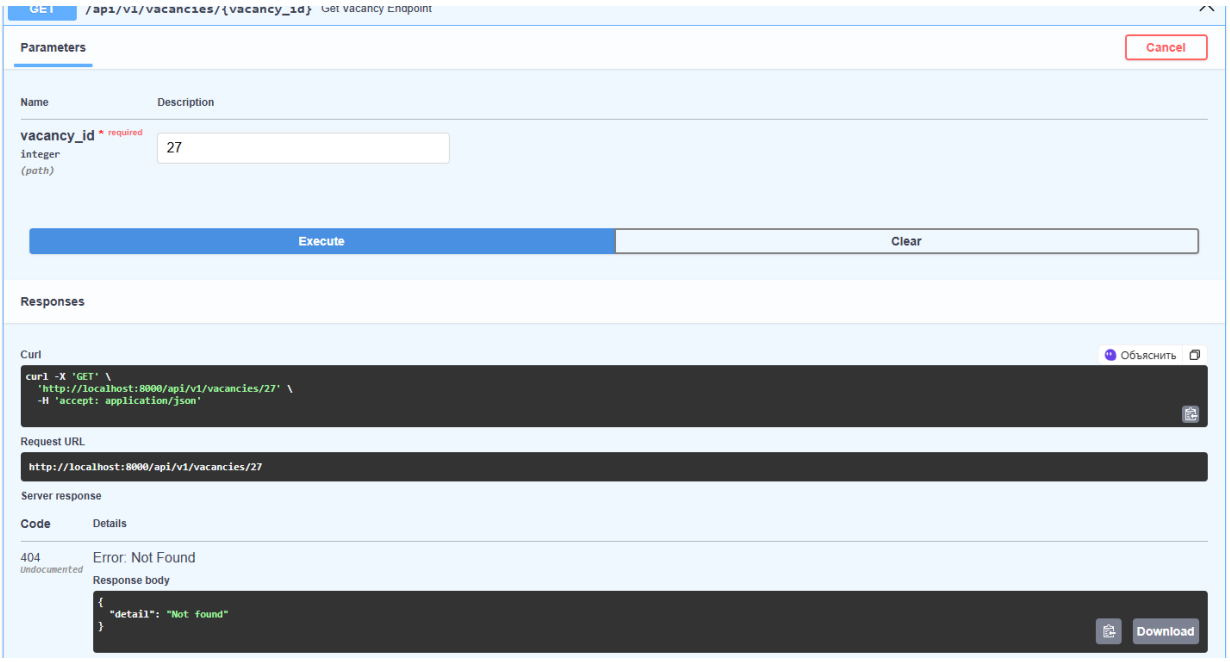

Рисунок 21 – Подтверждение удаления вакансии

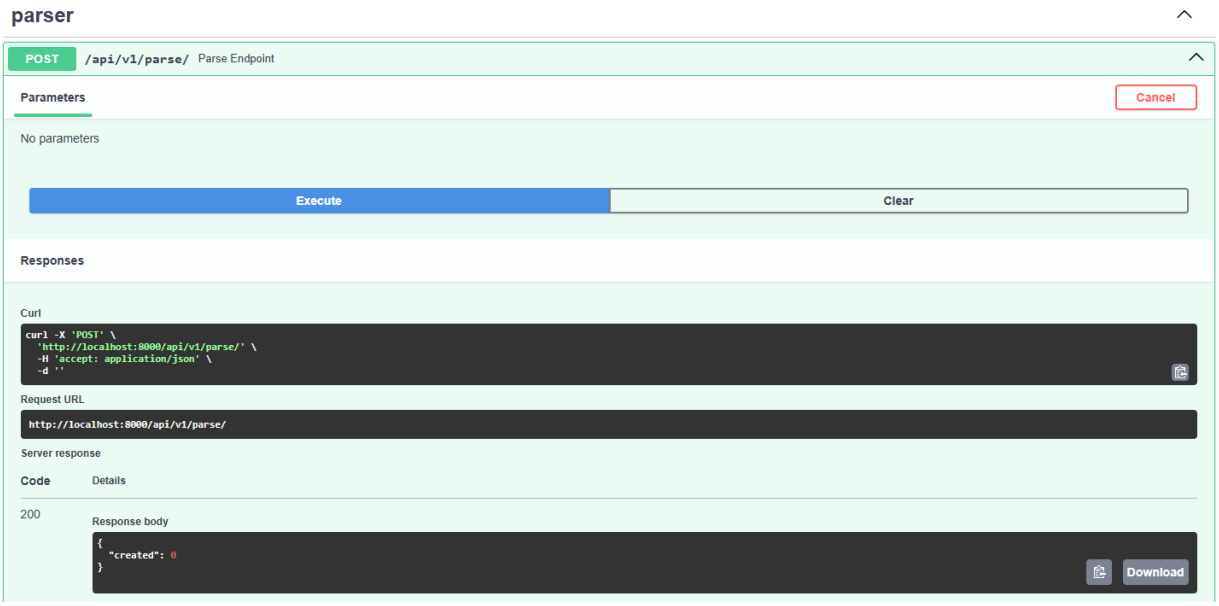

Рисунок 22 – Работа эндпоинта с парсером