

# 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.

```
Traceback (most recent call last):
  File "/usr/local/bin/alembic", line 8, in <module>
    sys.exit(main())
          ^^^^^^
  File "/usr/local/lib/python3.11/site-packages/alembic/config.py", line 1047, in main
    Commandline(prog=prog).main(argv=argv)
  File "/usr/local/lib/python3.11/site-packages/alembic/config.py", line 1037, in main
    self.run_cmd(cfg, options)
  File "/usr/local/lib/python3.11/site-packages/alembic/config.py", line 971, in run_cmd
    fn(
  File "/usr/local/lib/python3.11/site-packages/alembic/command.py", line 483, in upgrade
    script.run_env()
  File "/usr/local/lib/python3.11/site-packages/alembic/script/base.py", line 545, in run_env
    util.load_python_file(self.dir, "env.py")
  File "/usr/local/lib/python3.11/site-packages/alembic/util/pyfiles.py", line 116, in load_python_file
    module = load_module_py(module_id, path)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/alembic/util/pyfiles.py", line 136, in load_module_py
    spec.loader.exec_module(module) # type: ignore
          ^^^^^^^^^^^^^^^^^^^^^^^^^^
File "<frozen importlib._bootstrap_external>", line 940, in exec_module
File "<frozen importlib._bootstrap>", line 241, in _call_with_frames_removed
  File "/app/alembic/env.py", line 8, in <module>
    from app.core.config import settings
  File "/app/app/core/config.py", line 20, in <module>
    settings = Settings()
          ^^^^^^^
  File "/usr/local/lib/python3.11/site-packages/pydantic_settings/main.py", line 242, in __init__
    super().__init__(**_pydantic_self__._class_._settings_build_values(sources, init_kwargs))
  File "/usr/local/lib/python3.11/site-packages/pydantic/main.py", line 250, in __init__
    validated_self = self._pydantic_validator_.validate_python(data, self_instance=self)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
pydantic_core._pydantic_core.ValidationError: 1 validation error for Settings
database_url
  Extra inputs are not permitted [type=extra_forbidden]
```

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 строке 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).

```
1 fastapi
2 uvicorn[standard]
3 sqlalchemy>=2.0
4 asyncpg
5 alembic
6 pydantic
7 pydantic-settings
8 htpx
9 apscheduler
10 fastapi==999.0.0; python_version < "3.8"
11
```

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).

```
1 from typing import Awaitable, Callable
2
3 from apscheduler.schedulers.asyncio import AsyncIOScheduler
4
5 from app.core.config import settings
6
7
8 def create_scheduler(job: Callable[], Awaitable[None]) -> AsyncIOScheduler:
9     scheduler = AsyncIOScheduler()
10    scheduler.add_job(
11        job,
12        trigger="interval",
13        seconds=settings.parse_schedule_minutes,
14        coalesce=True,
15        max_instances=1,
16    )
17    return scheduler
18
```

Рисунок 3 – Неправильный параметр при создании задачи на парсинг

```

2026-02-18 10:22:32,198 | INFO | apscheduler.executors.default | Running job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:37 UTC)" (scheduled at 2026-02-18 10:22:32.198764+00:00)
2026-02-18 10:22:32,199 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-18 10:22:35,042 | INFO | httpx | HTTP Request: GET
https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=100&page=1 "HTTP/1.1 200 OK"
2026-02-18 10:22:35,107 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 0
2026-02-18 10:22:35,112 | INFO | apscheduler.executors.default | Job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:37 UTC)" executed successfully
2026-02-18 10:22:37,199 | INFO | apscheduler.executors.default | Running job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:42 UTC)" (scheduled at 2026-02-18 10:22:37.198764+00:00)
2026-02-18 10:22:37,199 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-18 10:22:39,765 | INFO | httpx | HTTP Request: GET
https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=100&page=1 "HTTP/1.1 200 OK"
2026-02-18 10:22:39,803 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 0
2026-02-18 10:22:39,805 | INFO | apscheduler.executors.default | Job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:42 UTC)" executed successfully
2026-02-18 10:22:42,199 | INFO | apscheduler.executors.default | Running job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:47 UTC)" (scheduled at 2026-02-18 10:22:42.198764+00:00)
2026-02-18 10:22:42,200 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-18 10:22:44,737 | INFO | httpx | HTTP Request: GET
https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=100&page=1 "HTTP/1.1 200 OK"
2026-02-18 10:22:44,782 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 0
2026-02-18 10:22:44,783 | INFO | apscheduler.executors.default | Job "_run_parse_job (trigger: interval[0:00:05], next run at: 2026-02-18 10:22:47 UTC)" executed successfully

```

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).

```

2026-02-18 10:19:14,560 | INFO | app.main | Запуск приложения
2026-02-18 10:19:14,560 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-18 10:19:18,213 | INFO | httpx | HTTP Request: GET
https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=100&page=1 "HTTP/1.1 200 OK"
2026-02-18 10:19:18,216 | ERROR | app.main | Ошибка фонового парсинга: 'NoneType' object has no attribute 'name'
Traceback (most recent call last):
  File "/app/app/main.py", line 24, in _run_parse_job
    await parse_and_store(session)
  File "/app/app/services/parser.py", line 43, in parse_and_store
    "city_name": item.city.name.strip(),
                ^^^^^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'name'
2026-02-18 10:19:18,219 | INFO | apscheduler.scheduler | Adding job tentatively -- it will be properly scheduled when the scheduler starts
2026-02-18 10:19:18,220 | INFO | apscheduler.scheduler | Added job "_run_parse_job" to job store "default"
2026-02-18 10:19:18,220 | INFO | apscheduler.scheduler | Scheduler started
INFO: Application startup complete.

```

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

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

```

34         payload = await fetch_page(client, page)
35         parsed_payloads = []
36         for item in payload.items:
37             parsed_payloads.append(
38                 {
39                     "external_id": item.id,
40                     "title": item.title,
41                     "timetable_mode_name": item.timetable_mode.name,
42                     "tag_name": item.tag.name,
43                     "city_name": item.city.name.strip(),
44                     "published_at": item.published_at,
45                     "is_remote_available": item.is_remote_available,
46                     "is_hot": item.is_hot,
47                 }
48             )
49
50         created_count = await upsert_external_vacancies(session, parsed_payloads)
51         created_total += created_count
52
53         if page >= payload.page_count:
54             break

```

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).

```

45 @router.post(path="/", response_model=VacancyRead, status_code=status.HTTP_201_CREATED)
46 async def create_vacancy_endpoint(
47     payload: VacancyCreate, session: AsyncSession = Depends(get_session)
48 ) -> VacancyRead:
49     if payload.external_id is not None:
50         existing = await get_vacancy_by_external_id(session, payload.external_id)
51         if existing:
52             return JSONResponse(
53                 status_code=status.HTTP_200_OK,
54                 content={"detail": "Vacancy with external_id already exists"},
55             )
56     return await create_vacancy(session, payload)
57
58

```

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

```
29     timeout = httpx.Timeout(timeout=10.0, read=20.0)
30     try:
31         client = httpx.AsyncClient(timeout=timeout)
32         page = 1
33         while True:
34             payload = await fetch_page(client, page)
35             parsed_payloads = []
36             for item in payload.items:
37                 parsed_payloads.append(
38                     {
39                         "external_id": item.id,
40                         "title": item.title,
41                         "timetable_mode_name": item.timetable_mode.name,
42                         "tag_name": item.tag.name,
43                         "city_name": item.city.name.strip() if item.city else 'Не указано',
44                         "published_at": item.published_at,
45                         "is_remote_available": item.is_remote_available,
46                         "is_hot": item.is_hot,
47                     }
48                 )
49             created_count = await upsert_external_vacancies(session, parsed_payloads)
50             created_total += created_count
51
52             if page >= payload.page_count:
53                 break
54             page += 1
55     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).

```
31     async with httpx.AsyncClient(timeout=timeout) as client:
32         page = 1
33         while True:
34             payload = await fetch_page(client, page)
35             parsed_payloads = []
36             for item in payload.items:
37                 parsed_payloads.append(
38                     {
39                         "external_id": item.id,
40                         "title": item.title,
41                         "timetable_mode_name": item.timetable_mode.name,
42                         "tag_name": item.tag.name,
43                         "city_name": item.city.name.strip() if item.city else 'Не указано',
44                         "published_at": item.published_at,
45                         "is_remote_available": item.is_remote_available,
46                         "is_hot": item.is_hot,
47                     }
48                 )
49
50             created_count = await upsert_external_vacancies(session, parsed_payloads)
51             created_total += created_count
52
53             if page >= payload.page_count:
54                 break
55             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).

```
62     async def upsert_external_vacancies(
63         session: AsyncSession, payloads: Iterable[dict]
64     ) -> int:
65         external_ids = [payload["external_id"] for payload in payloads if payload["external_id"]]
66         if external_ids:
67             existing_result = await session.execute(
68                 select(Vacancy.external_id).where(Vacancy.external_id.in_(external_ids))
69             )
70             existing_ids = set(existing_result.scalars().all())
71         else:
72             existing_ids = {}
73
74         created_count = 0
75         for payload in payloads:
76             ext_id = payload["external_id"]
77             if ext_id and ext_id in existing_ids:
78                 result = await session.execute(
79                     select(Vacancy).where(Vacancy.external_id == ext_id)
80                 )
81                 vacancy = result.scalar_one()
82                 for field, value in payload.items():
83                     setattr(vacancy, field, value)
84                 else:
85                     session.add(Vacancy(**payload))
86                     created_count += 1
87
88         await session.commit()
89     return created_count
```

Рисунок 11 – Проблема N+1 запросов к БД

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

```
62     async def upsert_external_vacancies(
63         session: AsyncSession, payloads: Iterable[dict]
64     ) -> int:
65         payload_map = {p["external_id"]: p for p in payloads if p.get("external_id")}
66         if not payload_map:
67             return 0
68
69         stmt = select(Vacancy).where(Vacancy.external_id.in_(payload_map.keys()))
70         result = await session.execute(stmt)
71         existing_vacancies = {v.external_id: v for v in result.scalars().all()}
72
73         created_count = 0
74         for ext_id, payload in payload_map.items():
75             if ext_id in existing_vacancies:
76                 vacancy = existing_vacancies[ext_id]
77                 for field, value in payload.items():
78                     setattr(vacancy, field, value)
79             else:
80                 session.add(Vacancy(**payload))
81                 created_count += 1
82
83         await session.commit()
84         return created_count
85
```

Рисунок 12 – Исправление ошибки №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).

```

20 usages  ▲ Birobizhan
9   class Vacancy(Base):
10    __tablename__ = "vacancies"
11    __table_args__ = (UniqueConstraint(*columns: "external_id", name="uq_vacancies_external_id"),)
12
13    id: Mapped[int] = mapped_column(Integer, primary_key=True, autoincrement=True)
14    title: Mapped[str] = mapped_column(String, nullable=False)
15    timetable_mode_name: Mapped[str] = mapped_column(String, nullable=False)
16    tag_name: Mapped[str] = mapped_column(String, nullable=False)
17    city_name: Mapped[str | None] = mapped_column(String, nullable=True)
18    published_at: Mapped[datetime] = mapped_column(DateTime(timezone=True), nullable=False)
19    is_remote_available: Mapped[bool] = mapped_column(Boolean, nullable=False, default=False)
20    is_hot: Mapped[bool] = mapped_column(Boolean, nullable=False, default=False)
21    created_at: Mapped[datetime] = mapped_column(
22        DateTime(timezone=True),
23        nullable=False,
24        server_default=func.now(),
25    )
26    external_id: Mapped[int | None] = mapped_column(Integer, nullable=True)
27

```

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.

```

9   class Vacancy(Base):
10    __tablename__ = "vacancies"
11
12    id: Mapped[int] = mapped_column(Integer, primary_key=True, autoincrement=True)
13    title: Mapped[str] = mapped_column(String, nullable=False)
14    timetable_mode_name: Mapped[str] = mapped_column(String, nullable=False)
15    tag_name: Mapped[str] = mapped_column(String, nullable=False)
16    city_name: Mapped[str | None] = mapped_column(String, nullable=True)
17    published_at: Mapped[datetime] = mapped_column(DateTime(timezone=True), nullable=False)
18    is_remote_available: Mapped[bool] = mapped_column(Boolean, nullable=False, default=False)
19    is_hot: Mapped[bool] = mapped_column(Boolean, nullable=False, default=False)
20    created_at: Mapped[datetime] = mapped_column(
21        DateTime(timezone=True),
22        nullable=False,
23        server_default=func.now(),
24    )
25    external_id: Mapped[int] = mapped_column(Integer, nullable=False, unique=True)

```

Рисунок 14 – Изменение модели вакансии

## Итоги:

- Все эндпоинты работают корректно (рисунки 15–22).
- Приложение возвращает корректные HTTP-статусы и данные.

GET /api/v1/vacancies/ List Vacancies Endpoint

**Parameters**

Name	Description
timetable_mode_name	string   (string   null) (query) Гибкий
city	string   (string   null) (query) Москва

**Responses**

Curl

```
curl -X 'GET' \
  'http://localhost:8000/api/v1/vacancies/?timetable_mode_name=%D0%93%D0%88%D0%95%D0%8B%D0%88%D0%98&city=%D0%93%D0%8E%D0%98%D0%9A%D0%82%D0%90' \
  -H 'accept: application/json'
```

Request URL

[http://localhost:8000/api/v1/vacancies/?timetable\\_mode\\_name=%D0%93%D0%88%D0%95%D0%8B%D0%88%D0%98&city=%D0%93%D0%8E%D0%98%D0%9A%D0%82%D0%90](http://localhost:8000/api/v1/vacancies/?timetable_mode_name=%D0%93%D0%88%D0%95%D0%8B%D0%88%D0%98&city=%D0%93%D0%8E%D0%98%D0%9A%D0%82%D0%90)

Server response

Code	Details
200	Response body

```
[
  {
    "title": "Специалист по внутренним коммуникациям",
    "timetable_mode_name": "Гибкий",
    "tag_name": "hr",
    "city_name": "Москва",
    "published_at": "2026-01-23T07:16:35.230609Z",
    "is_remote_available": false,
    "is_hot": false,
    "external_id": 1575,
    "id": 17,
    "created_at": "2026-02-18T13:24:35.347432Z"
  }
]
```

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

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/api/v1/vacancies/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "string",
    "timetable_mode_name": "string",
    "tag_name": "string",
    "city_name": "string",
    "published_at": "2026-02-18T11:28:24.451Z",
    "is_remote_available": true,
    "is_hot": true,
    "external_id": 0,
    "id": 77,
    "created_at": "2026-02-18T13:28:59.443110Z"
  }'
```

Request URL

<http://localhost:8000/api/v1/vacancies/>

Server response

Code	Details
201	Response body

```
{
  "title": "string",
  "timetable_mode_name": "string",
  "tag_name": "string",
  "city_name": "string",
  "published_at": "2026-02-18T11:28:24.451000Z",
  "is_remote_available": true,
  "is_hot": true,
  "external_id": 0,
  "id": 77,
  "created_at": "2026-02-18T13:28:59.443110Z"
}
```

Response headers

Рисунок 16 – Создание новой вакансии

Curl

```
curl -X 'POST' \
  http://localhost:8000/api/v1/vacancies/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
-d '{
  "title": "string",
  "timetable_mode_name": "string",
  "tag_name": "string",
  "city_name": "string",
  "published_at": "2026-02-18T11:28:24.451Z",
  "is_remote_available": true,
  "is_hot": true,
  "external_id": 0
}'
```

Request URL

```
http://localhost:8000/api/v1/vacancies/
```

Server response

Code	Details
400 Undocumented	Error: Bad Request Response body

```
{
  "detail": "Vacancy with this external_id already exists"
}
```

[Response headers](#) [Copy](#) [Download](#)

Рисунок 17 – Попытка создать вакансию с существующим external\_id

GET /api/v1/vacancies/{vacancy\_id} Get Vacancy Endpoint

Parameters

Name	Description
<b>vacancy_id</b> * required integer (path)	27

Responses

Code	Details
200	Response body

```
{
  "title": "string",
  "timetable_mode_name": "string",
  "tag_name": "string",
  "city_name": "string",
  "published_at": "2026-02-18T11:28:24.451000Z",
  "is_remote_available": true,
  "is_hot": true,
  "external_id": 0,
  "id": 27,
  "created_at": "2026-02-18T13:28:59.443119Z"
}
```

[Response headers](#) [Copy](#) [Download](#)

Рисунок 18 – Получение вакансии по id

Curl

```
curl -X 'PUT' \
  'http://localhost:8000/api/v1/vacancies/27' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
-d '{
    "title": "python",
    "timetable_mode_name": "string",
    "tag_name": "string",
    "city_name": "string",
    "published_at": "2026-02-18T11:28:35.084Z",
    "is_remote_available": true,
    "is_hot": true,
    "external_id": 0
}'
```

Request URL

```
http://localhost:8000/api/v1/vacancies/27
```

Server response

Code	Details
200	Response body <pre>{     "title": "python",     "timetable_mode_name": "string",     "tag_name": "string",     "city_name": "string",     "published_at": "2026-02-18T11:28:35.084000Z",     "is_remote_available": true,     "is_hot": true,     "external_id": 0,     "id": 27,     "created_at": "2026-02-18T13:28:59.443119Z" }</pre>

Response headers

Рисунок 19 – Изменение названия вакансии

**DELETE** /api/v1/vacancies/{vacancy\_id} Delete Vacancy Endpoint

Parameters

Name	Description
<b>vacancy_id</b> <small>* required</small>	integer (path) 27

Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8000/api/v1/vacancies/27' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8000/api/v1/vacancies/27
```

Server response

Code	Details
204	Response headers

Рисунок 20 – Удаление вакансии

The screenshot shows a REST API endpoint for deleting a vacancy. The URL is `/api/v1/vacancies/{vacancy_id}`. The method is `GET`. The endpoint description is "Get vacancy endpoint".  
Parameters:  
Name: `vacancy_id` (required, integer, path) Value: 27  
Buttons: Execute, Clear  
Responses:  
Curl command:  

```
curl -X 'GET' \
'http://localhost:8000/api/v1/vacancies/27' \
-H 'accept: application/json'
```

  
Request URL: `http://localhost:8000/api/v1/vacancies/27`  
Server response:  
Code: 404 Undocumented Details: Error: Not Found  
Response body:  

```
{ "detail": "Not found" }
```

  
Buttons: Объяснить, Download

Рисунок 21 – Подтверждение удаления вакансии

The screenshot shows a REST API endpoint for parsing data. The URL is `/api/v1/parse/`. The method is `POST`. The endpoint description is "Parse Endpoint".  
Parameters:  
No parameters  
Buttons: Execute, Clear  
Responses:  
Curl command:  

```
curl -X 'POST' \
'http://localhost:8000/api/v1/parse/' \
-H 'accept: application/json' \
-d ''
```

  
Request URL: `http://localhost:8000/api/v1/parse/`  
Server response:  
Code: 200 Details: Response body  

```
{ "created": 0 }
```

  
Buttons: Объяснить, Download

Рисунок 22 – Работа эндпоинта с парсером