

# Отчёт по отладке приложения Selectel-Vacancies-API

[github.com/imronaxl/selectest-api/](https://github.com/imronaxl/selectest-api/)

12 февраля 2026 г.

## Содержание

<b>1</b>	<b>Анализ и первичный запуск</b>	<b>2</b>
<b>2</b>	<b>Исправление багов</b>	<b>3</b>
2.1	Баг №1 – Опечатка в алиасе переменной окружения . . . . .	3
2.2	Баг №2 – Неверное имя БД по умолчанию . . . . .	3
2.3	Баг №3 – Падение при отсутствии города (AttributeError) . . . . .	3
2.4	Баг №4 – Утечка HTTP-соединений (незакрытый клиент) . . . . .	3
2.5	Баг №5 – Неверный интервал запуска (секунды вместо минут) . . . . .	4
2.6	Баг №6 – Нарушение контракта при создании дубликата (POST) . . . . .	4
2.7	Баг №7 – N+1 запросов в <code>upsert_external_vacancies</code> . . . . .	4
2.8	Баг №8 – Отсутствие проверки уникальности <code>external_id</code> при обновлении (PUT) . . . . .	5
<b>3</b>	<b>Результаты тестирования</b>	<b>6</b>
3.1	Протокол выполнения . . . . .	6
<b>4</b>	<b>Итог</b>	<b>8</b>

# 1 Анализ и первичный запуск

## Что сделано

Запущен Docker Compose:

```
1 docker compose up --build
```

Листинг 1: Первичный запуск

## Проблема

Контейнер app не может подключиться к базе данных:

```
1 sqlalchemy.exc.OperationalError: (psycopg2.OperationalError)
2 FATAL: database "postgres_tpyo" does not exist
```

Листинг 2: Ошибка подключения

## Причина

В файле конфигурации допущены две ошибки:

- Опечатка в алиасе переменной окружения (DATABSE\_URL вместо DATABASE\_URL);
- Имя базы данных по умолчанию (postgres\_tpyo) не соответствует тому, что указано в docker-compose.yml (postgres).

## Решение

Исправлен файл app/core/config.py.

### Было:

```
1 database_url: str = Field(
2     "postgresql+asyncpg://postgres:postgres@db:5432/postgres_tpyo",
3     validation_alias="DATABSE_URL",
4 )
```

### Стало:

```
1 database_url: str = Field(
2     "postgresql+asyncpg://postgres:postgres@db:5432/postgres",
3     validation_alias="DATABASE_URL",
4 )
```

После исправления приложение успешно подключилось к БД, миграции Alembic применились автоматически.

## 2 Исправление багов

### 2.1 Баг №1 – Опечатка в алиасе переменной окружения

- **Файл:** `app/core/config.py`
- **Строка:** 13
- **Код до:** `validation_alias="DATABSE_URL"`
- **Код после:** `validation_alias="DATABASE_URL"`
- **Причина:** Пропущена буква «А» в имени переменной, из-за чего значение из `.env` не считывалось.

### 2.2 Баг №2 – Неверное имя БД по умолчанию

- **Файл:** `app/core/config.py`
- **Строка:** 12
- **Код до:** `...@db:5432/postgres_typo`
- **Код после:** `...@db:5432/postgres`
- **Причина:** Несоответствие дефолтного значения и реальной конфигурации Docker.

### 2.3 Баг №3 – Падение при отсутствии города (AttributeError)

- **Файл:** `app/services/parser.py`
- **Строка:** 39
- **Код до:** `"city_name": item.city.name.strip()`
- **Код после:** `"city_name": item.city.name.strip() if item.city else None`
- **Причина:** Поле `city` в JSON-ответе Selectel может быть `null`. Обращение к атрибуту `name` у `None` вызывает исключение и останавливает парсинг.

### 2.4 Баг №4 – Утечка HTTP-соединений (незакрытый клиент)

- **Файл:** `app/services/parser.py`
- **Строка:** 26-28
- **Код до:**

```
1 client = httpx.AsyncClient(timeout=timeout)
2 # ... client
3 # client.close()
```

- **Код после:**

```
1 async with httpx.AsyncClient(timeout=timeout) as client:
2     # ... client
```

- **Причина:** HTTP-клиент создавался, но никогда не закрывался, что приводило к исчерпанию файловых дескрипторов и утечке памяти при длительной работе.

## 2.5 Баг №5 – Неверный интервал запуска (секунды вместо минут)

- Файл: app/services/scheduler.py
- Строка: 11
- Код до: seconds=settings.parse\_schedule\_minutes
- Код после: minutes=settings.parse\_schedule\_minutes
- Причина: В настройках указаны минуты, но параметр ошибочно передан в seconds. Задача запускалась каждые 5 секунд вместо 5 минут.

## 2.6 Баг №6 – Нарушение контракта при создании дубликата (POST)

- Файл: app/api/v1/vacancies.py
- Строка: 43–46
- Код до:

```
1 return JsonResponse(  
2     status_code=status.HTTP_200_OK,  
3     content={"detail": "Vacancy with external_id already  
4         exists"},  
5 )
```

- Код после:

```
1 raise HTTPException(  
2     status_code=status.HTTP_409_CONFLICT,  
3     detail="Vacancy with external_id already exists",  
4 )
```

- Причина: Возврат статуса 200 вместо 409 нарушает REST-практики и не соответствует объявленной Pydantic-схеме, что приводило к ошибкам валидации в Swagger.

## 2.7 Баг №7 – N+1 запросов в upsert\_external\_vacancies

- Файл: app/crud/vacancy.py
- Строка: 72–89 (в исходной версии)
- Код до (упрощённо):

```
1 existing_ids = set(...) # external_id  
2 for payload in payloads:  
3     if ext_id in existing_ids:  
4         result = await session.execute(select(Vacancy).where  
5             (...))  
6         vacancy = result.scalar_one() # N+1!  
7     ...
```

- Код после:

```

1 existing_map = {}
2 if external_ids:
3     res = await session.execute(
4         select(Vacancy).where(Vacancy.external_id.in_(
5             external_ids))
6     )
7     existing_map = {v.external_id: v for v in res.scalars().
8         all()}
9
10 for payload in payload_list:
11     ext_id = payload["external_id"]
12     if ext_id in existing_map:
13         vacancy = existing_map[ext_id]
14         for field, value in payload.items():
15             setattr(vacancy, field, value)
16     else:
17         session.add(Vacancy(**payload))
18     created_count += 1

```

- **Причина:** Отдельный SELECT для каждой существующей вакансии. При 1000 записей — 1000 лишних запросов к БД.

## 2.8 Баг №8 – Отсутствие проверки уникальности external\_id при обновлении (PUT)

- Файл: app/api/v1/vacancies.py
- Строка: 56–71
- Код до:

```

1 #
2 return await update_vacancy(session, vacancy, payload)

```

- Код после:

```

1 if payload.external_id is not None:
2     existing = await get_vacancy_by_external_id(session,
3         payload.external_id)
4     if existing and existing.id != vacancy_id:
5         raise HTTPException(
6             status_code=status.HTTP_409_CONFLICT,
7             detail="External ID already in use by another
8                 vacancy",
9         )
10 return await update_vacancy(session, vacancy, payload)

```

- **Причина:** Отсутствие бизнес-валидации позволяло присвоить вакансии external\_id, уже закреплённый за другой записью, что вызывало IntegrityError (500).

## 3 Результаты тестирования

Все исправления были проверены с помощью автоматизированного тестового скрипта `test_project.py`. Скрипт выполняет:

- запуск Docker-контейнеров;
- проверку доступности API;
- 10 интеграционных тестов (включая граничные случаи);
- анализ логов планировщика;
- остановку и очистку контейнеров.

### 3.1 Протокол выполнения

```
1 =====
2
3      : 2026-02-12 11:41:07
4 =====
5
6      : 10
7      : 0
8
9      #1:
10     : POST http://localhost:8000/api/v1/parse/
11           : 200,
12           :
13           : {"created":0}
14
15     #2:
16     : POST http://localhost:8000/api/v1/vacancies/
17           : 201,
18           :
19           : {"title":"Python Developer", ..., "external_id"
20           :1001}
21
22     #3:                                     (409 Conflict)
23     : POST http://localhost:8000/api/v1/vacancies/
24           : 409,
25           :
26           : {"detail":"Vacancy with external_id already exists"}
27
28     #4:
29     : GET http://localhost:8000/api/v1/vacancies/
           : 200,
           :
```

```

30         : 200,
31         :
32         #6:
33         : PUT http://localhost:8000/api/v1/vacancies/26
34         : 200,
35         :
36         #7:
37         : POST http://localhost:8000/api/v1/vacancies/
38         : 201,
39         :
40         #8:
41         external_id (409)
42         : PUT http://localhost:8000/api/v1/vacancies/26
43         : 409,
44         : {"detail": "External ID already in use by another
45         vacancy"}
46         #9:
47         : DELETE http://localhost:8000/api/v1/vacancies/27
48         : 204,
49         :
50         #10:
51         : LOG
52         : >=2
53         : 3
54         : 70 ,

```

Листинг 3: Результаты тестирования

## 4 Итог

После внесения всех исправлений:

- **Приложение стабильно запускается** в Docker без ошибок подключения к БД.
- **Парсер** корректно обрабатывает любые ответы внешнего API (включая `null`-поля), HTTP-клиент закрывается, утечек нет.
- **Фоновая задача** запускается строго с заданным интервалом (5 минут по умолчанию).
- **CRUD-операции** полностью работоспособны:
  - Создание валидирует уникальность `external_id` и возвращает `201 Created` / `409 Conflict`;
  - Обновление проверяет конфликты по `external_id` и отдаёт `409 Conflict` при попытке занять чужой идентификатор;
  - Чтение, фильтрация, удаление работают согласно спецификации.
- **Производительность** операций массового обновления (`upsert`) оптимизирована: `N+1` запросов устранён.
- **Все 8 багов исправлены**, ни одного нового дефекта не внесено.

Код исправленного приложения доступен на GitHub:

<https://github.com/Imronaxl/selectest-api/tree/main/selectest-api>