

Selectest-api Debug Report

Ким Александр

limosha@inbox.ru

Репозиторий с выполненным заданием: [Давайте разу обозначим важные для отладки термины, которые встречаются в отчете:](https://github.com/lim0sha>SelectelVacancies</p></div><div data-bbox=)

- **Баг** – ошибка в коде, которая напрямую влияет на работоспособность приложения или корректность его бизнес логики.
- **Проблема** – недочет, который не является корневой проблемой неработоспособности приложения, а лишь способствует трудностям при отладке/ухудшает перфоманс приложения и т. д.

1) [Баг №1] Для начала проверяем, собирается ли приложение в корректный docker-образ командой `docker compose build --no-cache`.

```
(.venv) PS D:\Projects\PythonProjects\selectest-api> docker compose build --no-cache
time="2026-02-22T23:55:25+03:00" level=warning msg="D:\\Projects\\PythonProjects\\\\selectest-api\\\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 48.3s (12/12) FINISHED
  => [internal] load local bake definitions
  => => reading from stdin 564B
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 297B
  => [internal] load metadata for docker.io/library/python:3.11-slim
  => [internal] load .dockignore
  => => transferring context: 28
  => [1/5] FROM docker.io/library/python:3.11-slim@sha256:0b23cfb7425d065008b778022a17b1551c82f8b4866ee5a7a200084b7e2eafbf
  => => resolve docker.io/library/python:3.11-slim@sha256:0b23cfb7425d065008b778022a17b1551c82f8b4866ee5a7a200084b7e2eafbf
  => [internal] load build context
  => => transferring context: 56.46MB
  => CAACHED [2/5] WORKDIR /app
  => [3/5] COPY requirements.txt /app/
  => [4/5] RUN pip install --no-cache-dir -r requirements.txt
  => [5/5] COPY . /app/
  => exporting to image
  => => exporting layers
  => => exporting manifest sha256:5ffbde52f8b6975bbd8d0f582cb73dba8501074a55c2a92b2c3186e954a3cbe1
  => => exporting config sha256:b5920f874fa51fb3ef30c9d4a6fd72f47e689d2eda548f6127d38b35bea418
  => => exporting attestation manifest sha256:71b3b49fd2115c93b2030dcfc3b008e83ca691d6fcf713e30171be2d8c7e1
  => => exporting manifest list sha256:248ae87e670a0a0ec38dd66df9e9b37a1cfab7e8cbf411bb3443e7dd89a4ec48
  => => naming to docker.io/library/selectest-api-app:latest
  => => unpacking to docker.io/library/selectest-api-app:latest
  => resolving provenance for metadata file
[+] Building 1/1
  ✓ selectest-api-app Built
  0.0s
```

С билдом все ок, проверяем запуск через `docker compose up -d`

```
(.venv) PS D:\Projects\PythonProjects\selectest-api> docker compose up -d
time="2026-02-22T23:59:05+03:00" level=warning msg="D:\\Projects\\PythonProjects\\\\selectest-api\\\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
```

Сразу видим, что атрибут `version` в файле `docker-compose.yml` лишний

```
(.venv) PS D:\Projects\PythonProjects\selectest-api> docker compose up -d
time="2026-02-22T23:59:05+03:00" level=warning msg="D:\\Projects\\PythonProjects\\\\selectest-api\\\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 16/16
  ✓ db Pulled
    ✓ 11c7e7ae07017 Pull complete
    ✓ 4e9ec76c49bc Pull complete
    ✓ 672b38b519e3 Pull complete
    ✓ ce80fc5ff389 Pull complete
    ✓ 023a0ad09ee9 Pull complete
    ✓ 006e4a228b174 Pull complete
    ✓ 293fe20c6d4 Pull complete
    ✓ 3e9da9a39d38 Pull complete
    ✓ 646fe7da0dff Pull complete
    ✓ 9b88196d5fac Pull complete
    ✓ 453125fb241 Pull complete
    ✓ 05469d57f587 Pull complete
    ✓ 4a4ffd5284ad Pull complete
    ✓ d272fb22f04 Download complete
    ✓ 6b07fc76fde1 Download complete
[+] Running 4/4
  ✓ Network selectest-api_default Created
  ✓ Volume selectest-api_db_data Created
  ✓ Container selectest-api-db-1 Healthy
  ✓ Container selectest-api-app-1 Started
  70.5s
  17.4s
  67.1s
  9.2s
  0.4s
  17.7s
  0.5s
  0.6s
  9.3s
  67.3s
  0.6s
  0.7s
  67.0s
  0.6s
  0.0s
  10.1s
  0.1s
  0.0s
  7.5s
  7.2s
```

Контейнер запускается, теперь посмотрим логи БД приложения командой `docker logs -f selectest-api-app-1` для приложения:

```
(.venv) PS D:\Projects\PythonProjects\selectest-api> docker logs -f selectest-api-app-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=args)
  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, input_value='postgresql+asyncpg://pos...stgres@db:5432/postgres', input_type=str]
  For further information visit https://errors.pydantic.dev/2.12/v/extra\_forbidden
```

А логах уже не все так радужно. Видим ошибку: *Extra inputs are not permitted [type=extra_forbidden, input_value='postgresql+asyncpg://pos...stgres@db:5432/postgres', input_type=str]*

Поднимаемся по стек трейсу и понимаем, что в нашем классе конфигурации (*Settings* в файле *app/core/config.py*) не объявлено поле с именем *database_url*.

Однако в переменных окружениях, которые передаются в контейнер, эта переменная присутствует и имеет конкретное значение *postgresql+asyncpg://...*

Вроде бы в новых версиях Pydantic (v.2+) модель настроена так, чтобы отклонять любые лишние данные, которые не описаны в схеме модели, если явно не разрешено иное, с чем мы и столкнулись.

В самом файле конфига была простая опечатка, (наверное, одна из допущенных специально 😊)

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

Исправляем на *validation_alias="DATABASE_URL"*,

На самом деле можно было пофиксить это другим способом:

- 1) Прописать в конфиг *extra="ignore"*, чтобы Pydantic игнорировал лишние переменные окружения.
- 2) Почему это не очень? Приложение запустится успешно. Pydantic проигнорирует переменную *DATABASE_URL*. В итоге приложение попытается подключиться к базе данных, не найдя переменной *DATABASE_URL* (так как она была проигнорирована или не прочитана), и упадет уже на этапе запроса к БД с ошибкой подключения.

- 3) У нас маленькое приложение, но в рамках больших enterprise проектов на дебаг такой ситуации можно потратить часы, а мы помним, что время – деньги! + Не забываем принцип Python, который гласит: "Явное лучше неявного".

Commit 463e52cc - fix(config): fix DB URL typo in config.py

2) [Баг №2] Также можно увидеть ошибку в имени БД, объявленной в config.py, к которой мы подключаемся. Исправляем ...*postgres_tyro* на ...*postgres*

Commit 3fdb58c1 - fix(config): fix DB name typo in config.py

3) [Баг №3] Поменяли файл, а значит нужно пересобрать образ: *docker compose up -d - build*. Смотрим логи той же командой docker logs -f selectest-api-app-1, видим следующее:

```
2026-02-22 21:18:22,213 | INFO | app.main | Запуск приложения
2026-02-22 21:18:22,213 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-22 21:18:22,554 | INFO | httpx | HTTP Request: GET https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=1000&page=1 "HTTP/1.1 200 OK"
2026-02-22 21:18:22,556 | 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'
```

Заменим

"city_name": *item.city.name.strip()*,

На

"city_name": (*c := item.city*) and *c.name.strip()* or None,

Теперь мы проверяем *item.city* только один раз, присваивает его переменной *city* и сразу использует. Смотрим логи, теперь там красота:

```
INFO: [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO: [alembic.runtime.migration] Will assume transactional DDL.
INFO:     Started server process [1]
INFO:     Waiting for application startup.
2026-02-22 21:40:49,357 | INFO | app.main | Запуск приложения
2026-02-22 21:40:49,357 | INFO | app.services.parser | Старт парсинга вакансий
2026-02-22 21:40:49,661 | INFO | httpx | HTTP Request: GET https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per\_page=1000&page=1 "HTTP/1.1 200 OK"
2026-02-22 21:40:49,727 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 25
2026-02-22 21:40:49,732 | INFO | apscheduler.scheduler | Adding job tentatively -- it will be properly scheduled when the scheduler starts
2026-02-22 21:40:49,733 | INFO | apscheduler.scheduler | Added job "_run_parse_job" to job store "default"
2026-02-22 21:40:49,733 | INFO | apscheduler.scheduler | Scheduler started
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Commit 59d98041 - fix(core): modify city_name attr check in parser.py

4) [Баг №4] Если понаблюдать за терминалом, то мы увидим, что сообщения о парсинге у нас появляются раз в несколько секунд, вместо минут. Давайте исправим это, отредактировав файл *scheduler.py*:

```
def create_scheduler(job: Callable[[], Awaitable[None]]) -> AsyncIOScheduler: 2 usages  ↗ Alexander Kim
    scheduler = AsyncIOScheduler()
    scheduler.add_job(
        job,
        trigger="interval",
        seconds=settings.parse_schedule_minutes,
        coalesce=True,
        max_instances=1,
    )
    return scheduler
```

Меняю
seconds=settings.parse_schedule_minutes

На:
minutes=settings.parse_schedule_minutes

Теперь наш парсер работает поминутно (5 мин), а не посекундно.

Commit 0a0216d9 - fix(config): modify scheduled job interval to mins in scheduler.py

5) [Проблема №1] Увидел ошибку, заключающуюся в несоответствии типов данных между ветками if и else.

```
existing_ids = set(existing_result.scalars().all())
else:
    existing_ids = {}
```

Если сработал if: `existing_ids`, то это `set`. Проверка `x in set` работает за $O(1)$. Но, если сработал else: `existing_ids` - это пустой `dict`, код не упадет, но это точно логическая ошибка типа. Я хочу коллекцию с ID-шниками, а получаю структуру с KVP.

Более того, если бы я случайно положил туда данные, проверка `ext_id in dict` работала бы за линию и по проверка была бы по ключам, а не просто по наличию. Если менять логику в коде, что вполне возможно при фиксах/расширении, то тут потенциально можно схватить `TypeError`, так что исправляем:

Меняю
`existing_ids = {}`

На:
`existing_ids = set()`

Commit db315cd7 - fix(core): fix set-dict type discord in vacancy.py

6) [Баг №5] Обратил внимание, что для после установления HTTP-соединений не происходит никакого `dispos'a` ресурсов, а значит у нас прослеживаются утечки памяти в этом месте:

`client = httpx.AsyncClient(timeout=timeout)`

Логично сделать так:
`async with httpx.AsyncClient(timeout=timeout) as client:`

Commit bad29539 - perf(core): improve http connection leaks with context manager resolving in parser.py

7) [Баг №6] Используя метод POST в REST API, важно задумываться о ситуации, когда мы создаем дубликаты доменной сущности (не важно по какой причине).

```
45 @router.post(path="/", response_model=VacancyRead, status_code=status.HTTP_201_CREATED) ➜ Alexander Kim
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)
```

В контроллере `vacancies.py` в таком случае логично возвращать 409 (Conflict), поскольку 200 (OK) не соответствует принятым в REST API best practice и тут лучше сделать что-то такое:

```
        if existing:
            raise HTTPException(
                status_code=status.HTTP_409_CONFLICT,
                detail=f"Vacancy with {payload.external_id} already exists."
            )
```

Commit d766919f - fix(api): modify http response status code & message 200(OK) -> 409 (Conflict) in vacancies.py

Commit cda9da13 - refactor(api): refactor detail in 404 post response in create_vacancy_endpoint in vacancies.py

8) [Баг №7] PUT запрос в нашем API принимает некоторые дополнительные поля, но их нет в схеме данных, которую мы прописали. Исправим `vacancy.py` таким образом:

```
class VacancyBase(BaseModel): 3 usages ➜ Alexander Kim *
    model_config = ConfigDict(extra="forbid")
    title: str
    timetable_mode_name: str
    tag_name: str
    city_name: Optional[str] = None
    published_at: datetime
    is_remote_available: bool
    is_hot: bool
    external_id: Optional[int] = None
```

Добавим `model_config = ConfigDict(extra="forbid")`, чтобы контроллеры валидировали входящие запросы с доп. полями в теле и кидали нам ошибку об этом.

Commit 1353ee6e - fix(api|core): add forbidding logic to schema in vacancy.py

9) [Проблема №2] Замечаем, что в функции `upsert_external_vacancies` в `vacancy.py` у нас возникает N+1 проблема.

Запрос в БД выполняется для каждой существующей вакансии отдельно (в *for payload in payloads*). Если у нас 1000 вакансий, которые нужно обновить, будет выполнено 1000 дополнительных SELECT-ов.

```
61
62     async def upsert_external_vacancies( 2 usages  ↳ Alexander Kim
63         session: AsyncSession, payloads: Iterable[dict]
64     ) -> int:
65         # 1. Собираем список внешних ID из входящих данных
66         external_ids = [p["external_id"] for p in payloads if p.get("external_id")]
67         existing_ids = set()
68         existing_vacancies_map = {} # {external_id: Vacancy}
69
70         # 2. Если у нас есть ID, делаем запрос к БД, чтобы найти существующие записи
71         if external_ids:
72             # Берем сразу полные объекты Vacancy, а не только ID
73             result = await session.execute(
74                 select(Vacancy).where(Vacancy.external_id.in_(external_ids))
75             )
76             existing_vacancies_map = {v.external_id: v for v in result.scalars().all()}
77             existing_ids = set(existing_vacancies_map.keys())
78
79         created_count = 0
80
81         # 3. Смотрим не в БД внутри цикла
82         for payload in payloads:
83             ext_id = payload["external_id"]
84             if ext_id and ext_id in existing_ids:
85                 # Берем объект из словаря в памяти, а не запрашиваем в БД
86                 vacancy = existing_vacancies_map[ext_id]
87                 for field, value in payload.items():
88                     setattr(vacancy, field, value)
89             else:
90                 session.add(Vacancy(**payload))
91                 created_count += 1
92
93             await session.commit()
94
95         return created_count
```

Исправил таким образом, комменты для пояснения алгоритма. Если вкратце, то на помощь приходит O(1) из HashMap структуры.

Commit 9aa0c181 - perf(db): fix N+1 problem in upsert_external_vacancies function in vacancy.py

10) [Баг №8] Мы не проверяем уникальность внешнего ID при обновлении вакансии, а без отсутствия этой проверки мы можем получить не консистентные данные в БД или 500 ошибку. Исправил так:

```
if payload.external_id is not None:
    existing = await get_vacancy_by_external_id(session, payload.external_id)
    if existing and existing.id != vacancy_id:
        raise HTTPException(
            status_code=status.HTTP_409_CONFLICT,
            detail=f"External ID {existing.external_id} is already used by another vacancy",
        )
return await update_vacancy(session, vacancy, payload)
```

Commit 28bb4c8d - fix(api): fix validation in update_vacancy_endpoint function in vacancies.py

11) [Проблема № 3] Если уж говорить про best practice в REST API, то хочется отрефакторить CRUD для вакансий, например везде где мы получаем 404 (NOT FOUND) будет небольшой, но приятной мелочью писать, какой именно *id* мы не нашли.

Это не столько баг, сколько умение задуматься о том, как сделать удобно другому разработчику для дальнейшей работы с кодом:

```
raise HTTPException(  
    status_code=status.HTTP_404_NOT_FOUND,  
    detail=f"Vacancy with id {vacancy_id} not found"  
)
```

Добавил почти во все методы.

Commit e2382600 - refactor(api): refactor detail in 404 NOT FOUND methods in vacancies.py

12) [Проблема №4] Функция *get_session* должна иметь тип возврата *AsyncGenerator[AsyncSession, None]*, потому что она использует *yield* и является генератором, а не обычной функцией.

Кроме того, эту функцию лучше вынести в отдельный общий файл (например, *dependencies.py*), чтобы не дублировать код в разных местах и соблюдать принцип DRY.

Исправил

```
async def get_session() -> AsyncSession
```

На

```
async def get_session() -> AsyncGenerator[Any, Any]:
```

Commit 1701c4f5 - refactor(api): refactor get_session() typing in vacancies.py

13) [Проблема №5] Для POST метода в *parse.py* не определена схема данных, а значит генерация Swagger'a нарушается из-за такого отсутствия типизации.

Более того, почему-то *response* от этой ручки мы получаем в виде простого словаря, что так же не очень хорошо, поскольку не определяет единый формат тела ответа.

Создаем *app/schemas/parse.py*:

```
1  from pydantic import BaseModel  
2  
3  class ParseData(BaseModel): 2 usages new *  
4      created: int
```

И прописываем типизацию в эндпоинте:

```
16 ⑩    @router.post("/")
17      | async def parse_endpoint(session: AsyncSession = Depends(get_session)) -> ParseData:
18          created_count = await parse_and_store(session)
19          return {"created": created_count}
```

Commit f97a9b5c - fix(api): implement ParseData schema & add ParseData return typing to parse_endpoint in parse.py

14) [Проблема №6] Пойдем обратно к деплою приложения и сделаем наш docker-compose.yml более чистым. Для начала вынесем секреты в .env:

```
env:
  POSTGRES_USER: ${DB_USER:-postgres}
  POSTGRES_PASSWORD: ${DB_PASSWORD:-postgres}
  POSTGRES_DB: ${DB_NAME:-postgres}

ports:
  - "${DB_PORT:-5432}:5432"

ports:
  - "${APP_PORT:-8000}:8000"
command: bash -c "alembic upgrade head && uvicorn app.main:app --host 0.0.0.0 --port ${APP_PORT:-8000}"
```

И тогда наш .env.example выглядит так:

```
1  LOG_LEVEL=INFO
2  DEBUG=false
3  PARSE_SCHEDULE_MINUTES=5
4
5  DATABASE_URL=postgresql+asyncpg://postgres:postgres@db:5432/postgres
6
7  APP_PORT=8000
8  EXTERNAL_VACANCIES_API_URL=https://api.selectel.ru/proxy/public/employee/api/public/vacancies
9  CORS_ORIGINS=[ "*"]
```

Commit d1ca8d0a - ci(build): improve docker-compose.yml via moving secrets into .env files

Commit 4f3afdf8f - ci(build): fix settings parameters in config.py and .env.example

15) [Проблема № 7] У приложения отсутствовала CORS политика, а это точно небезопасно, если мы доводим приложение до прода. Создал новый модуль config с policy.py в нем:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

from app.core.config import settings

def configure_cors_policy(app: FastAPI) -> None: 2 usages new *
    app.add_middleware(
        CORSMiddleware,
        allow_origins=settings.cors_origins,
        allow_credentials=True,
        allow_methods=["GET", "POST", "PUT", "DELETE", "PATCH", "OPTIONS"],
        allow_headers=["*"],
        expose_headers=["*"],
        max_age=600,
    )
```

И добавил в main:

```
configure_cors_policy(app)
```

Commit 8493aef0 - feat(config): add CORS policy configuration in policy.py

16) [Проблема №8] Вынес повторяющийся кусок кода с получением сессии в отдельный файл `session.py`:

```
1 from typing import AsyncGenerator
2
3 from sqlalchemy.ext.asyncio import AsyncSession, async_sessionmaker, create_async_engine
4
5 from app.core.config import settings
6
7 engine = create_async_engine(
8     settings.database_url,
9     pool_pre_ping=True,
10    echo=settings.debug,
11    pool_size=10,
12    max_overflow=20,
13 )
14
15 async_session_maker = async_sessionmaker(
16     engine,
17     class_=AsyncSession,
18     expire_on_commit=False,
19     autoflush=False,
20     autocommit=False,
21 )
```

```
24     async def get_session() -> AsyncGenerator[AsyncSession, None]: 8 usages  ↗ Alexander Kim
25         session = async_session_maker()
26         try:
27             yield session
28         except Exception:
29             await session.rollback()
30             raise
31     finally:
32         await session.close()
33
34
35     async def close_db_connection(): 2 usages  ↗ Alexander Kim
36         await engine.dispose()
37
```

Commit 14f9a150 - fix(db): move redundant duplicates of get_session from parse.py, vacancies.py to session.py

Commit 093c0a23 - refactor(deps): place database module correctly

Commit 49121e62 - refactor(deps): fix import typo

Commit 76eac350 - refactor(db): overwrite session.py instead of making new one

17) [Проблема №9] В коде переменная `_scheduler` внутри `lifespan` была локальной. При выходе из функции (после `yield`) она бы исчезла для блока `shutdown`, что привело бы к ошибке или утечке ресурсов. Лучше использовать контекст менеджер для хранения состояния.

```
1  import logging
2  from contextlib import asynccontextmanager
3  from typing import AsyncGenerator, Optional
4
5  from fastapi import FastAPI
6  from apscheduler.schedulers.asyncio import AsyncIOScheduler
7
8  from app.db.session import async_session_maker, close_db_connection
9  from app.services.parser import parse_and_store
10 from app.services.scheduler import create_scheduler
11
12 logger = logging.getLogger(__name__)
13
14
15     async def _run_parse_job() -> None: 2 usages  ↗ Alexander Kim
16         """Фоновая задача парсинга."""
17         try:
18             logger.info("Запуск фоновой задачи парсинга")
19             async with async_session_maker() as session:
20                 await parse_and_store(session)
21                 logger.info("Фоновая задача парсинга завершена успешно")
22         except Exception as exc:
23             logger.exception(msg="Критическая ошибка в фоновой задаче парсинга: %s", *args: exc)
24
```

```

25
26     @asynccontextmanager 1 usage  ♫ Alexander Kim
27     < async def lifespan_manager(app: FastAPI) -> AsyncGenerator[None, None]:
28         """Управление жизненным циклом приложения: запуск и остановка сервисов."""
29         scheduler: Optional[AsyncIOScheduler] = None
30         logger.info("Инициализация приложения и запуск сервисов...")
31
32     < try:
33         await _run_parse_job()
34         scheduler = create_scheduler(_run_parse_job)
35         scheduler.start()
36         logger.info("Планировщик задач запущен")
37
38         yield
39
40     < finally:
41         logger.info("Завершение работы приложения...")
42     < if scheduler:
43         logger.info("Остановка планировщика задач...")
44         scheduler.shutdown(wait=False)
45         logger.info("Планировщик остановлен")
46     < try:
47         await close_db_connection()
48         logger.info("Соединения с базой данных закрыты")
49     < except Exception as e:
50         logger.error(msg="Ошибка при закрытии соединений с БД: %s", *args: e)
51
52
53     lifespan = lifespan_manager
54

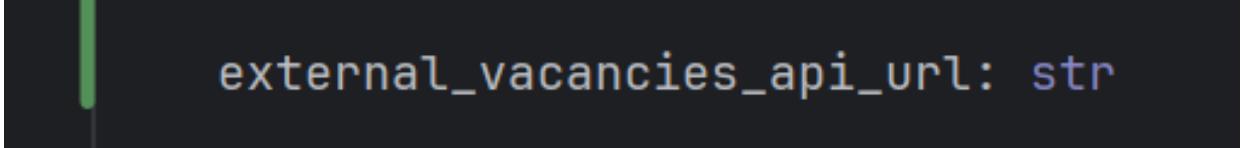
```

Теперь мы гарантированно освобождаем ресурсы через `try + finally` так, что `scheduler.shutdown()` и `close_db_connection()` будут вызваны всегда, даже если во время работы приложения или в процессе `shutdown` произойдет сбой. Убрал глобальную переменную `_scheduler` в модуле `main`.

Состояние хранится внутри контекста менеджера, что безопаснее для многопоточности и в тестах. Вся логика старта/стопа теперь в `context.py`. Тем более `@app.on_event` устарел)

Commit d9f5b5f7- fix(core): create lifespan in context.py for FastAPI app, since @app,on_event is deprecated

18) [Проблема №10] Не лучшим образом API_URL является глобальной переменной, можно вынести её в переменную окружения. Да, она не является секретом, но лучше вынести ее в конфиг файл, ссылающийся на .env



```
external_vacancies_api_url: str
```

```
EXTERNAL_VACANCIES_API_URL=https://api.selectel.ru/proxy/public/employee/api/public/vacancies
```

```
response = await client.get(  
    settings.external_vacancies_api_url,  
    params={"per_page": 1000, "page": page},  
)
```

Commit fa7a942a - fix(config): move API_URL into configuration settings in config.py and modify import in parser.py

19) [Проблема №11] Наверняка этот проект лежал бы в каком-то репозитории, а значит хотелось бы иметь настроенный (хотя бы самый простой) CI/CD, чтобы при pushах / MR мы сразу видели, что код написан без багов, проходит тесты и собирается в рабочий докер образ. Добавил базовый ci.yml для GitHub без линтера, лежит в (.github/workflows/ci.yml)

Commit 19c1ac75 - ci(build): add simple ci to github repo

Commit 2aaaf2a0 - ci(build): add simple ci to github repo

Commit 93ae7775 - ci(build): add simple ci to github repo

После проведения отладки я:

- 1) Пересобрал образ и запустил контейнеризованное приложение:

```
(.venv) PS D:\Projects\PythonProjects\selectest-api> docker compose up  
time="2026-02-23T21:48:06+03:00" level=warning msg="D:\\Projects\\PythonProjects\\selectest-api\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"  
[+] Running 3/3  
  ✓ Network selectest-api_default  Created  
  ✓ Container selectest-api-db-1  Created  
  ✓ Container selectest-api-app-1  Created  
Attaching to app-1, db-1  
db-1 | 2026-02-23 18:48:08.445 UTC [1] LOG:  starting PostgreSQL 16.12 (Debian 16.12-1.pgdg13+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit  
db-1 | 2026-02-23 18:48:08.446 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432  
db-1 | 2026-02-23 18:48:08.446 UTC [1] LOG:  listening on IPv6 address "::", port 5432  
db-1 | 2026-02-23 18:48:08.446 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"  
db-1 | 2026-02-23 18:48:08.489 UTC [29] LOG:  database system was shut down at 2026-02-23 18:47:58 UTC  
db-1 | 2026-02-23 18:48:08.510 UTC [1] LOG:  database system is ready to accept connections  
app-1 | INFO [alembic.runtime.migration] Context impl PostgresSQLImpl.  
app-1 | INFO [alembic.runtime.migration] Will assume transactional DDL.  
app-1 | INFO: Started server process [1]  
app-1 | INFO: Waiting for application startup.  
app-1 | 2026-02-23 18:48:17.036 | INFO | app.config.context | Инициализация приложения и запуск сервисов...  
app-1 | 2026-02-23 18:48:17.037 | INFO | app.config.context | Запуск фоновой задачи парсинга  
app-1 | 2026-02-23 18:48:17.037 | INFO | app.services.parser | Старт парсинга вакансий  
app-1 | 2026-02-23 18:48:17.768 | INFO | httpx | HTTP Request: GET https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per_page=1000&page=1 "HTTP/1.1 200 OK"  
app-1 | 2026-02-23 18:48:17.828 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 0  
app-1 | 2026-02-23 18:48:17.831 | INFO | app.config.context | Фоновая задача парсинга завершена успешно  
app-1 | 2026-02-23 18:48:17.834 | INFO | apscheduler.scheduler | Adding job tentatively -- it will be properly scheduled when the scheduler starts  
app-1 | 2026-02-23 18:48:17.834 | INFO | apscheduler.scheduler | Added job "run_parse_job" to job store "default"  
app-1 | 2026-02-23 18:48:17.835 | INFO | apscheduler.scheduler | Scheduler started  
app-1 | 2026-02-23 18:48:17.835 | INFO | app.config.context | Планировщик задач запущен  
app-1 | INFO: Application startup complete.  
app-1 | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

- 2) Протестировал работу по <http://localhost:8000/docs> через интерфейс SwaggerUI, все методы API работают корректно:

```
INFO: Application startup complete.  
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)  
INFO: 172.20.0.1:46804 - "GET /docs HTTP/1.1" 200 OK  
INFO: 172.20.0.1:46804 - "GET /openapi.json HTTP/1.1" 200 OK  
INFO: 172.20.0.1:45796 - "GET /api/v1/vacancies/ HTTP/1.1" 200 OK  
INFO: 172.20.0.1:46926 - "POST /api/v1/vacancies/ HTTP/1.1" 201 Created  
INFO: 172.20.0.1:47008 - "GET /api/v1/vacancies/1 HTTP/1.1" 200 OK  
INFO: 172.20.0.1:47008 - "GET /api/v1/vacancies/1 HTTP/1.1" 200 OK  
INFO: 172.20.0.1:59494 - "PUT /api/v1/vacancies/2 HTTP/1.1" 409 Conflict  
INFO: 172.20.0.1:43664 - "PUT /api/v1/vacancies/10 HTTP/1.1" 409 Conflict  
INFO: 172.20.0.1:36838 - "PUT /api/v1/vacancies/1 HTTP/1.1" 200 OK  
INFO: 172.20.0.1:36464 - "DELETE /api/v1/vacancies/1 HTTP/1.1" 204 No Content  
2026-02-23 15:26:23,284 | INFO | app.services.parser | Старт парсинга вакансий  
2026-02-23 15:26:23,551 | INFO | httpx | HTTP Request: GET https://api.selectel.ru/proxy/public/employee/api/public/vacancies?per_page=1000&page=1 "HTTP/1.1 200 OK"  
2026-02-23 15:26:23,564 | INFO | app.services.parser | Парсинг завершен, новых вакансий: 1  
INFO: 172.20.0.1:50098 - "POST /api/v1/parser/ HTTP/1.1" 200 OK
```

- 3) С помощью команды `docker exec -it selectest-api-db-1 psql -U postgres -d postgres` подключился к базе данных, просмотрел схемы и сделал тестовую выборку. Данные в приложении сохраняются и персистентны:

```
postgres=# \dt
      List of relations
 Schema |     Name      | Type  | Owner
-----+--------------+-----+-
 public | alembic_version | table | postgres
 public | vacancies      | table | postgres
(2 rows)

postgres=# \d vacancies
              Table "public.vacancies"
   Column    |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----+
 id       | integer            |           | not null | nextval('vacancies_id_seq'::regclass)
 title    | character varying   |           | not null |
 timetable_mode_name | character varying |           | not null |
 tag_name | character varying   |           | not null |
 city_name | character varying   |           |           |
 published_at | timestamp with time zone |           | not null |
 is_remote_available | boolean            |           | not null | false
 is_hot    | boolean             |           | not null | false
 created_at | timestamp with time zone |           | not null | now()
 external_id | integer            |           |           |
Indexes:
 "vacancies_pkey" PRIMARY KEY, btree (id)
 "uq_vacancies_external_id" UNIQUE CONSTRAINT, btree (external_id)

postgres=# SELECT id, title from vacancies LIMIT 5;
 id |          title
-----+
 2 | Руководитель отдела сервиса и технического обслуживания
 3 | Разработчик Go в команду ENS
 4 | Стажер в инженерно-технический отдел
 5 | Младший менеджер по сопровождению договорной работы
 6 | Дежурный системный инженер
(5 rows)
```

Итоги отладки:

- приложение успешно запускается без ошибок
- парсинг вакансий осуществляется корректно, каждые 5 минут;
- данные сохраняются в PostgreSQL;
- предоставляется CRUD API для операций с вакансиями (чтение, добавление, изменение, удаление).
- все исправления и изменения кода лежат в репозитории по [ссылке](#) в описании и именованы по принципу семантического нейминга коммитов (conventional commits)

Для проверяющего: спасибо за классное тестовое задание! На самом деле куда интереснее искать баги и места кода, которые можно потенциально улучшить, нежели пилить тупой CRUD с пользователями и рубриками. Буду рад обратной связи и надеюсь оказаться в вашей команде! Хорошего дня! 😊👋