Test Task Back-end Developer (Python)

Primary Objective - comprehensively examine an individual's methodology for selecting appropriate problem-solving strategies, algorithms and design principles. This includes evaluating their capacity to produce maintainable, extensible, and testable code, as well as their overall approach to understanding and addressing complex problems.

One of the important conditions:

DO NOT use anything external except for basic standard frameworks and utilities - no boilerplate code (your own previous work is undesirable). However, naturally, you can use an existing data/json parser and/or similar fundamental tools.

Stack:

- Python 3.8+
- Flask/FastAPI
- PostgreSQL (or other RDBMS)
- Asyncpg (or other async dbapi/adapter)
- SQLAlchemy 2.0
- Asyncio
- Pytest
- Docker [optional]

Asynchronous Data Retrieval

You can use any available method to achieve "asynchrony" (concurrent computing), including paradigms such as Reactive and so on.

1) There are three data sources (for the test task, these can be three simple tables with an id and some text field).

Example:

```
-- Create table
CREATE TABLE data_1 (
    id INT PRIMARY KEY,
    name VARCHAR(255)
);
-- Insert data into table
INSERT INTO data_1 (id, name)
VALUES (1, 'Test 1'), (2, 'Test 2');
```

The IDs are distributed as follows:

- 1st source: IDs 1-10, 31-40;
- 2nd source: IDs 11-20, 41-50;
- 3rd source: IDs 21-30, 51-60;

2) There is a single common access point to these data sources (a Flask/FastAPI application) that provides a correlated result. The access point is available via HTTP.

Example:

```
[
    {“id”:1,”name”:”Test 1”},
    {“id”:2,”name”:”Test 2”}
]
```

3) This access point should make requests to all data sources "asynchronously" and wait for the results from all of them.
4) Upon receiving the results from all sources, return the data sorted by ID (data from all sources).
5) An error from any of the sources is ignored and interpreted as missing data.
6) A timeout is also considered an error (2 seconds).
7) Write unit-tests to cover all added logic, preferably use mocks for db connection.
8) It is essential to demonstrate the use of both ORM and raw SQL.
9) Provide code comments, docstrings, documentation and typing annotations [bonus].