Architectures for Big Data - First assignement

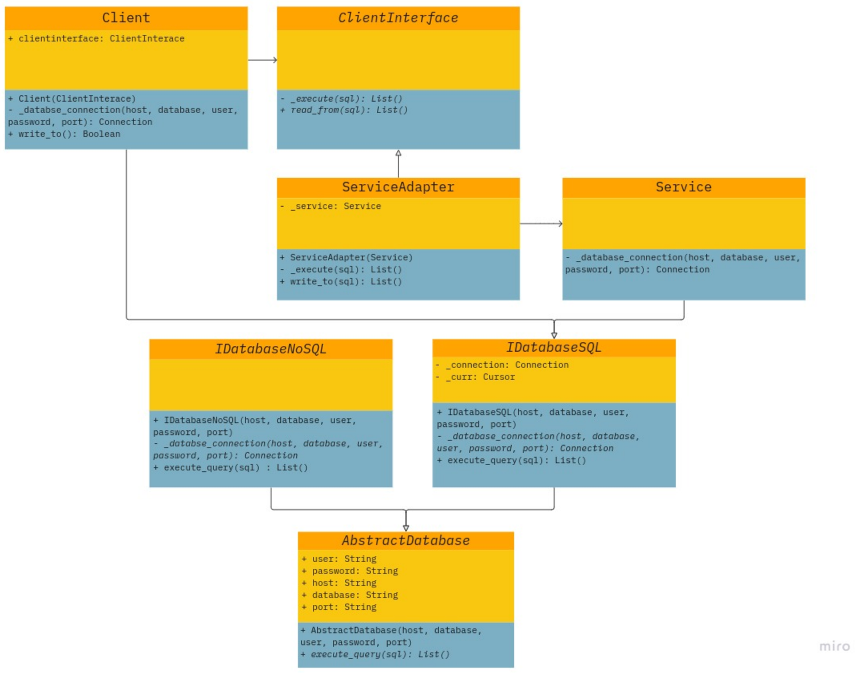
Federico Cristiano Bruzzone & Andrea Longoni

1 Adapter (Wrapper) Pattern

We thought to use **Adapter (Wrapper)** pattern beacuse this allow us to keep the same *client* and make it to comunicate with different types of *services*. The client is the historical database, where we want to write on it, and the services are the external database, from which we want to read on it.

1.1 Structure

This implementation uses the object composition principle: the adapter implements the interface of one object and wraps the other one.



Client
+ clientinterface: ClientInterace
+ Client(ClientInterace)
- _databse_connection(host, database, user, password, port): Connection
+ write_to(): Boolean
ClientInterface
- _execute(sql): List()
+ read_from(sql): List()
ServiceAdapter
- _service: Service
+ ServiceAdapter(Service)
- _execute(sql): List()
+ write_to(sql): List()
Service
- _database_connection(host, database, user, password, port): Connection
IDatabaseNoSQL
+ IDatabaseNoSQL(host, database, user, password, port)
- _databse_connection(host, database, user, password, port): Connection
+ execute_query(sql) : List()
IDatabaseSQL
- _connection: Connection
- _curr: Cursor
+ IDatabaseSQL(host, database, user, password, port)
- _database_connection(host, database, user, password, port): Connection
+ execute_query(sql): List()
AbstractDatabase
+ user: String
+ password: String
+ host: String
+ database: String
+ port: String
+ AbstractDatabase(host, database, user, password, port)
+ execute_query(sql): List()
miro

1. *AbstractDatabase* is an abstract class that contains data to connect with any database

2. *IDatabaseSQL & IDatabaseNoSQL* are an abstract classes that define behavior of the specified database

3. *service* is a concrete class that contains the methods to read from the external database

4. *ServiceAdapter* is a concrete class that inherits from *ClientInterface* and contains a *Service* object. Its purpose is to get data from service and make it readable for the client through the implemented *ClientInterface* methods.

5. *ClientInterface* is an interface that contains the delcarations of methods defined for each adapters
6. *Client* is a concrete class that contains the methods to write in the historical database, to make it, it use the *ClientInterface* methods to read adapted data from the external database.

2 Software Architecture Pillars

2.1 Being the framework for satisfying requirements

Functional

Our code is able to read from the external database and write to the historical database without any problems.

Technical

We are able to read from any database and any table of them, if the adapters of the databases are setted. And we can write on the historical database if the same table exists and the right fields are setted.

Security

Query precompiled and prepare statement

2.2 Being the technical basis for design

In the our code you can find an interface called *ClientInterface* and its implementation that allows modularization because the *Client* stay unchenged and you could change, add, delete and modify the *Services* as you prefer.

2.3 Being the managerial basis for cost estimation and process management

2.4 Enabling component reuse

Our code is reusable beacuse if you want to change the databse where you read you should only write a new *ServiceAdapter* and *Service* to connect them to the *Client*.

So, if you to use a NoSQL type database, you will implement a new *ServiceAdapter* and *Service* that will inherit from *IDatabaseNoSQL*. Acutally, the most importat thing is that the other part of the code will not change.

2.5 Allowing a tidy scalability

Our code allow you to do more INSERT at a time. With an only one Query, thanks to the *ServiceAdapter*, you could read a set of tuples and write them on the historical database with a for loops.

TODO:

In our code is not implemented, but a possibile solution for more scalability is to implement a multi-thread read/write structure.

2.6 Avoiding handover and people lock-in

To avoiding handover and people lock-in we could write more comments and documentation about our code.