Implement a server application in C++, which is a simple key-value storage. The server should use TCP protocol for network communications. The server stores a number of string keys and associated string values and provides an access to read and write them. Upon receiving connection from client, server reads provided request data then processes the request, writes output to the client and disconnects the client. Request and response data is a null-terminated JSON string.

The server supports two types of requests: read and write.

Example of read request:
{"request": "read", "key": "some_key"}
Examples of response:
{"status": "ok", "value": "some_value"}
{"status": "error", "description": "error description"}

Example of write request:
{"request": "write", "key": "some_key", "value": "some_value"}
Examples of response:
{"status": "ok"}
{"status": "error", "description": "error description"}

Server should check that request data is a valid JSON, has all required fields, and key/value are JSON strings. If the request is not correct, the server should send an error response. It should also send an error response in case the read request is sent for a non-existing key.

You must implement the server using Boost.Asio library. You can use any open source library to work with JSON, for example GitHub - nlohmann/json: JSON for Modern C++.
*Use asynchronous Boost.Asio to handle TCP connections.
The data in the storage should not be deleted when the server is restarted. You can use any database and the corresponding libraries.
*Use PostgreSQL for the storage implementation.

Note that the server should support multiple concurrent client connections, which could read/write the storage simultaneously. The storage doesn't need to be very performant, but it should be robust when dealing with concurrent operations.

Please provide server-side tests to make sure everything works as expected. You can use any test framework (but catch or gtest are preferred). For example a test case for storage:

```
TEST_CASE("Write request stores key-value pair") {
    Storage storage;
    storage.write("username", "user");
    REQUIRE(storage.read("username") == "user");
}
```

Please provide source code for your solution, files required to build it and instructions how to build and use it.

Everything marked by * are bonus subtasks. This is not required, but will provide an additional benefit.