Computação Paralela e Distribuída
2023 / 2024

Licenciatura em Engenharia Informática

Trabalho Prático #2 – Serviço de JSON-RPC

Introdução

Neste trabalho prático pretende-se construir um serviço de JSON-RPC. Este serviço é constituído por um servidor que disponibiliza o acesso remoto a um conjunto de funções e um cliente que acede às funções registadas no servidor.

Para as mensagens será usado o formato JSON-RPC 2.0 (http://www.jsonrpc.org/specification). O seguinte excerto exemplifica a utilização de JSON-RPC para a invocação da função *add(2, 3)*.

```
Cliente → Servidor
{"id": 1, "jsonrpc": "2.0", "method": "add", "params": [2, 3]}

Servidor → Cliente
{"id": 1, "jsonrpc": "2.0", "result": 5}
```

Este trabalho tem por base o código fornecido e é constituído pelos seguintes ficheiros:

- *server.py* → Contém o servidor na classe *JSONRPCServer* e exemplo de execução no *main*.
- *client.py* → Contém o cliente na classe *JSONRPCClient* e exemplo de execução no *main*.
- *functions.py* → Funções que devem ser disponibilizadas pelo servidor.
- *tests_server.py* → Alguns *unittests* que validam o funcionamento correcto do servidor.
- *tests_client.py* → Alguns *unittests* que validam o funcionamento correcto do cliente.
- server.py → Contém o servidor na classe JSONRPCServer e exemplo de execução no main.

Desenvolvimento

O código fornecido contém uma implementação inicial que deverá ser expandida pelos alunos.

Servidor: Após receber uma ligação de um cliente (método *JSONRPCServer.start()*), é executado o método ***handle_client(conn)***. Este método deverá ser alterado de forma a que o servidor lide correctamente com os pedidos e as respostas em formato JSON-RPC.

Cliente: Após criar o objecto *JSONRPCClient*, a invocação de um método não existente desta classe executará eventualmente o método ***invoke(method, params)***. Este método deverá ser alterado de forma a enviar o pedido e tratar correctamente as respostas.

Testes

Os ficheiros *tests_server.py* e *tests_client.py* contêm um conjunto de *unittests* que servem para testar o funcionamento correcto do servidor e do cliente, e de certos aspectos do protocolo.

Em relação ao servidor, alguns destes testes são relativamente simples de se entender: por exemplo o teste *TestProtocolFields.testVersion* verifica se o servidor coloca correctamente o campo *jsonrpc* na resposta enquanto o teste *TestResults.testAdd* valida o resultado quando se invoca a função *add*.

Existem também outros testes para verificar se o servidor valida correctamente os erros nos pedidos dos clientes (*TestErrors*) e testes como os *TestEdgeCases testNoParams* e *testNoParamsNeeded* que verificam se o servidor valida correctamente a utilização (ou não) dos parâmetros das funções.

Existem também testes para o cliente, como por exemplo o teste *TestProtocol.testID que* verifica se o cliente envia correctamente o campo *id* e os testes em *TestErrors* que verificam se o cliente lança as respectivas excepções quando o método não existe ou quando os parâmetros não são enviados correctamente.

Os alunos não deverão modificar nada nas classes *JSONRPCServer* e *JSONRPCClient* que previnam a execução dos testes. Recomenda-se executar os testes regularmente de forma a verificar se estão a funcionar e se os programas estão a cumprir com os requisitos.

Entrega e avaliação

Os trabalhos deverão ser realizados em grupos de 2 alunos da mesma turma de laboratório, e deverão ser originais. Aos trabalhos plagiados ou cujo código tenha sido partilhado com outros serão atribuídos nota **zero**.

Todos os ficheiros deverão ser colocados num **ficheiro zip** (com os números dos elementos do grupo) e submetidos via moodle **até às 23:55 do dia 02/Junho/2024.** Deverá também ser colocado no *zip* um ficheiro em PDF com a identificação dos alunos, uma descrição da solução, e a descrição dos extras implementados.

Irá considerar-se a seguinte grelha de avaliação:

Correcção da solução (testes)	12 val.
Qualidade e modularização do código (pylint, etc.)	04 val.
Extras (~1 valor por extra)	04 val.

Alguns extras a considerar: (1) parâmetros com nome (*named parameters*), (2) várias funções num único pedido (*batch*), (3) reutilização de ligações *tcp/ip*, (4) clientes em paralelo (*multithreading*), (5) respostas assíncronas, (6) outros.

Bom trabalho!