Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação Concorrente, Inverno de 2018/2019

Terceira Série de Exercícios

1. Realize um servidor com interface TCP para acesso a filas de mensagens usado o modelo de interacção pedido-resposta.
   - Os pedidos e as respostas são objectos JSON.
   - Os pedidos tem a seguinte estrutura genérica, inspirada na estrutura dos pedidos HTTP.

```
{
    “method”: “a string with the command operation”,
    “path”: “a string with a resource path”,
    “headers”: { … a JSON object where all fields are strings … },
    “payload”: { … a JSON object …}
}
```

   - As respostas tem a seguinte estrutura genérica, inspirada na estrutura das respostas HTTP.

```
{
    “status”: … an integer … ,
    “headers”: { … a JSON object where all fields are strings… },
    “payload”: { … a JSON object …}
}
```

   - O servidor aceita as seguintes operações:
     - CREATE - garante a existência da fila com o nome definido no campo **`path`**.
     - SEND - envia a mensagem presente em **`payload`** para a fila com nome definido em **`path`**.
     - RECEIVE - retira uma mensagem da fila com nome definido em **`path`** e retorna-a no campo **`payload`**. O tempo máximo de espera, em milisegundos, é definido no *header* **`timeout`**.
     - SHUTDOWN - inicia o processo de encerramento do servidor, ficando à espera que este esteja concluído. O tempo máximo de espera, em milisegundos, é definido no *header* **`timeout`**.
   - O servidor usa os seguintes códigos de **`status`**:
     - 200 - operação realizada com sucesso.
     - 204 - *timeout*.
     - 400 - pedido incorrecto (e.g. formato inválido).
     - 404 - fila não existente.
     - 500 - erro de servidor.
     - 503 - serviço indisponível (e.g. em processo de *shutdown*).
   - A implementação do servidor deve também ter em conta os seguintes aspectos:
     - Utilização do modelo TAP (*Task based Asynchronous Pattern*), evitando o bloqueio de *threads*, nomeadamente na recepção de mensagens e na espera pelo encerramento do servidor. Tire partido dos métodos assíncronos disponíveis bo C# a partir da versão 5.0.
     - Limitação, por concepção, do número máximo de pedidos que o servidor pode processar simultaneamente.

- Funcionalidade de registo suportada por uma *thread* de baixa prioridade (*logger thread*) criada para o efeito. As mensagens com os relatórios devem ser passadas das *threads* que servem pedidos (produtoras) para a *logger thread* usando um mecanismo de comunicação que minimize o tempo de bloqueio das *threads* produtoras. A funcionalidade de registo deve ter o mínimo de influência no tempo de serviço, admitindo-se inclusivamente a possibilidade de ignorar relatórios.

Deve também apresentar uma aplicação cliente do servidor implementado que permita testar toda a sua funcionalidade.

Nota: Em anexo é fornecido um servidor de eco (`JsonEchoServer.cs`) que exemplifica o código usado para seriação e deseriação de objectos JSON. Tenha em consideração que terá de incluir no *build* o pacote *Nuget* `Newtonsoft.Json`, versão 12.0.1, como aqui se descreve.

2. Usando a TPL (*Task Parallel Library*) e, se achar conveniente, os métodos assíncronos do C#, implemente uma aplicação que:
   - Dado o caminho para uma pasta e uma *string*.
   - Apresenta, para todos os ficheiros contidos nessa pasta, as linhas onde ocorre essa *string*.

A aplicação contém interface gráfica para recolha dos parâmetros de pesquisa e para apresentação dos resultados. Para melhorar a experiência de utilização, a apresentação dos resultados é realizada de forma incremental enquanto a pesquisa decorre. A interface gráfica inclui botão para cancelamento da pesquisa em curso.

Data limite de entrega: 3 de Janeiro de 2018

ISEL, 7 de Dezembro de 2017