Ciências ULisboa | Informática

Aplicações Distribuídas

2019/20

Projeto 4

1. Descrição geral

A avaliação da cadeira de aplicações distribuídas está divida em quatro projetos. O quarto projeto incide sobre medidas de desempenho e de segurança para o projeto anterior.

No terceiro projeto foi concretizado um serviço WEB para gerir um sistema simplificado de classificação de albuns de música de utilizadores. Para esse efeito no servidor foi utilizada a *framework* de desenvolvimento WEB *Flask* [2] e o motor de base de dados SQL *sqlite* [3]. O programa cliente utiliza o módulo *requests* [4] para implementar a interação cliente/servidor baseada no HTTP.

2. Desempenho

Assumindo que o serviço é executado através do servidor WSGI embutido no *Flask* e que este executa num computador com múltiplos núcleos de processamento, os alunos deverão alterar o servidor do projeto 3 para que este responda eficientemente a múltiplos clientes. Além das alterações ao código deverão justificar o método escolhido no ficheiro README que será entregue.

Ainda no sentido de aumentar o desempenho, independentemente das capacidades ou configuração do servidor, o cliente desenvolvido através do módulo *requests* deverá fazer com que os vários pedidos feitos ao servidor sejam enviados na mesma ligação TCP (ligações persistentes no HTTP).

3. Autenticação, confidencialidade da comunicação e autorização

O protocolo SSL/TLS deverá ser utilizado com o *Flask* e com o módulo *requests* para que o cliente e o servidor verifiquem mutuamente a autenticidade do interlocutor e para que a comunicação seja confidencial (cifrada). Para este efeito, cliente e servidor deverão ter certificados de chave pública assinados por uma *Certificate Authority (CA)*. A implementação no *Flask* será feita através da classe `SSLContext` do módulo `ssl` [1] da biblioteca padrão do *Python*. O protocolo *OAuth2* [2][3] deverá ser utilizado com o *Flask* e com o módulo *requests-oauthlib* [4][5] para que o cliente possa pedir autorização para usar os recursos disponiblizados pelo servidor. Para este efeito, o URI do servidor deverá estar registado numa API de OAuth (google, facebook, twitter, spotify, ...), de forma que o cliente possa ser autenticado e autorizado por esta API.

3.1. Criação de certificados

Para que os certificados do cliente e do servidor possam ser assinados por uma CA é necessário criar um certificado para a CA fictícia:

`openssl genrsa -out root.key 2048`
(cria a chave da CA)

`openssl req -x509 -new -nodes -key root.key -sha256 -days 365 -out root.pem`
(cria um certificado autoassinado para a CA)

De seguida serão geradas as chaves do servidor e do cliente de acordo com o seguinte padrão de comando:

`openssl genrsa -out <nome do ficheiro>.key 2048`

Para que a CA assine os certificados do cliente e do servidor ter-se-á que emitir um pedido de assinatura de certificado para cada um. Isso pode ser feito de acordo com o padrão de comando:

```
openssl req -new -nodes -key <nome do ficheiro>.key -sha256 -days 365 \
-out <nome do ficheiro>.csr
```

Na posse dos pedidos de assinatura a CA procede à geração dos certificados assinados por si. Isso pode ser feito para o cliente e para o servidor pelo seguinte padrão de comando:

```
openssl x509 -req -in <nome do ficheiro>.csr -CA root.pem -CAkey root.key \
-CAcreateserial -out <nome do ficheiro>.crt -days 365 -sha256
```

Através deste método cliente e servidor deverão, cada um, ter um par de ficheiros `<nome>.crt` e `<nome>.key`. Estes serão utilizados nos programas para que a autenticação e comunicação cifrada com o interlocutor sejam possíveis. Na implementação da autenticação os programas cliente e servidor terão que utilizar o certificado da CA (root.pem) para validar a assinatura do certificado apresentado pelo interlocutor.

4. Entrega

A entrega do projeto 4 consiste em colocar todos os ficheiros *.py* e *.db* dos projetos 3 e 4, o ficheiro README e o ficheiro com o esquema da base de dados em SQL numa diretoria cujo nome deve seguir exatamente o padrão **grupoXX** (por exemplo grupo01 ou grupo23). Nesta diretoria serão criadas duas subdiretorias com os nomes projeto3 e projeto4. Em cada uma destas diretorias será colocado o ficheiro README correspondente ao projeto, bem como três subdiretorias com os nomes client, server e certs (este último apenas num caso). Nestas serão colocados os ficheiros referentes ao cliente (grupoXXX/projetoX/client), ao servidor (grupoXXX/projetoX/server/) e aos certificados (grupoXXX/projeto4/certs/).

Juntamente com os ficheiros *.py* e *.db* deverá ser enviado um ficheiro de texto README.txt (não é .pdf nem .rtf nem .doc nem .docx) onde os alunos descrevem o modo de utilização e funcionamento do projeto (software desenvolvido). Deverão relatar também toda a informação que acharem pertinente sobre a sua implementação do projeto (por exemplo, limitações). Neste ficheiro os alunos, também, colocarão um exemplo para cada operação que a aplicação cliente suporta.

A diretoria será incluída num ficheiro ZIP cujo nome deve seguir exatamente o padrão **grupoXX.zip**. Esse ficheiro será submetido num recurso a disponibilizar para o efeito na página de AD no moodle da FCUL.

Note que a **entrega deve conter apenas os ficheiros .py, .sql, certificados e o ficheiro README.txt, qualquer outro ficheiro vai ser ignorado**.

O prazo de entrega é terça-feira, dia 02/06/2020, até às 23:55hs.

5. Bibliografia

[1] https://docs.python.org/3/library/ssl.html

[2] https://aaronparecki.com/oauth-2-simplified/

[3] https://oauth.net

[4] https://pypi.python.org/pypi/requests-oauthlib

[5] http://requests-oauthlib.readthedocs.io/en/latest/index.html