Ciências ULisboa | Informática

Aplicações Distribuídas

2022/23

Projeto 4

1. Descrição geral

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

No terceiro projeto foi concretizado um serviço WEB para gerir um sistema simplificado de exploração de voos pela Europa. Para esse efeito, foram utilizados no servidor a *framework* de desenvolvimento WEB *Flask* e o motor de base de dados SQL *sqlite*. O programa cliente utiliza o módulo *requests* para implementar a interação cliente/servidor baseada no HTTP.

Este quarto projeto terá o seu foco no desenvolvimento de três novos componentes: **SSL/TLS, Zookeeper e OAuth2.**

2. Autenticação e Confidencialidade da comunicação

O protocolo SSL/TLS v1.3 deverá ser utilizado com o *Flask* e com o módulo *requests* (ver os parâmetros `cert` e `verify`) para que o cliente e o servidor verifiquem mutuamente a autenticidade do interlocutor e para que a comunicação seja confidencial (cifrada). Para este efeito, o cliente e o servidor deverão ter certificados de chave pública assinados por uma *Certificate Authority (CA)*. Ou seja, os certificados não devem ser auto-assinados (ver os exercícios do Guião da **PL08**). A implementação no *Flask* será feita através da classe `SSLContext` do módulo `ssl` [1] da biblioteca padrão do *Python*. Recomenda-se passar um objeto `SSLContext` à variável `ssl_context` no arranque do *Flask* (ver o Guião da PL08).

3. Alterações Relevantes e Zookeeper

O Projecto 3 sofre agora uma alteração no seu funcionamento para que vários clientes possam usar o sistema de forma eficiente. Quando um cliente inicia a sua interação com o servidor e efectua uma pesquisa `/search`, deverá desencadear um preenchimento da DB tal e qual proposto no Projecto 3. Contudo, agora a pesquisa do cliente deve ficar registada com um ID único. Esta deverá ficar inequivocamente associada ao Cliente. Exemplo: o Cliente A faz duas pesquisas seguidas. Cliente B faz uma pesquisa. O servidor regista as pesquisas sob os IDs s123 e s124 para o Cliente A e sob o ID s125 para o Cliente B.

Associado a cada pesquisa, deverá haver uma tabela com a lista de viagens ID. Deste modo, o recurso `/search` deverá retornar além dos dados já existentes do Projecto 3, um ID de pesquisa único (exemplo: s123).

Assim sendo, o recurso `/filter` será alterado face ao Projecto 3, tendo agora um comportamento mais simples. O utilizador poderá agora indicar a search ID e os parâmetros a filtrar (`dst, airline, e n`) para fazer uma filtragem de dados, pois o servidor sabe exactamente a que viagens o cliente se refere. Desta feita, não será necessário enviar dados

JSON com viagens. O servidor deverá ir removendo entradas/viagens dessa tabela a cada chamada do `/filter`.

Exemplo de duas filtragens relacionadas e que podem executar consecutivamente:

```
[GET] /filter/s123?dst=lisboa # Tabela s123 fica agora com 120 viagens
[GET] /filter/s123?n=4  # Tabela s123 fica agora com 26 viagens
```

O Zookeeper será utilizado para monitorizar quando um cliente se desliga. Nesse momento, todas as pesquisas desse cliente deverão ser removidas, ie, devem ser apagadas todas as tabelas de Pesquisas associadas àquele cliente.

Nota: Deverá garantir com recurso à Autenticação que apenas o cliente que gerou uma dada pesquisa pode utilizar `/filter` da pesquisa em questão. Ex.: apenas o Cliente A pode filtrar a pesquisa s123. Apenas Cliente B pode filtrar a pesquisa s125.

4. Autorização e Nova Funcionalidade

O protocolo *OAuth2* [2][3] deverá ser utilizado com o *Flask* e com o módulo *requests_oauthlib* [4][5] para que a aplicação possa pedir autorização para usar os recursos disponiblizados pelo servidor de recursos do **Google Calendar.** Para este efeito, o URI de callback do servidor deverá estar registado na API de OAuth do Google de forma que o mesmo possa ser receber o código de autorização para pedir tokens de acesso (ver o Guião da PL09).

A aplicação Flask deverá incluir três novos recursos (`/login`, `/callback` e `/add_event`, semelhantes aos do **Guião da PL09**). Os dois primeiros recursos são necessários para que o utilizador possa dar autorização através do navegador à aplicação para aceder a recursos protegidos antes de iniciar a interação com os outros recursos implementados no Projeto 3. O recurso protegido que será acedido pelo recurso `/add_event` será o Calendário do Google (developers.google.com/calendar/api). Este recurso deve ser acompanhado do ID de uma viagem escolhida pelo Utilizador. O propósito deste recurso será permitir ao Cliente pedir ao Servidor que adicione dois eventos no seu calendário Google em seu nome. É espectável que no fim deste processo, sejam criados dois eventos no calendário, representando o tempo dos dois voos.

5. Entrega

A entrega do Projeto 4 consiste em colocar todos os ficheiros *.py* e *.db*, 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** (onde XX é o número do grupo). Deve-se criar três subdiretorias com os nomes **client**, **server** e **certs**. Nestas serão colocados os ficheiros referentes ao cliente (grupoXX/client), ao servidor (grupoXX/server/) e aos certificados (grupoXX/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).

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 as diretorias, os ficheiros .py, .sql, certificados e o ficheiro README.txt, qualquer outro ficheiro vai ser ignorado**.

O prazo de entrega é domingo, dia 21/05/2023, até às 23h59m (Lisboa).

6. 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

[6] googleapis.github.io/google-api-python-client/docs/dyn/calendar_v3.events.html#insert