Ciências ULisboa | Informática

Aplicações Distribuídas

2017/18

Projeto 3

1. Descrição geral

A avaliação da cadeira de aplicações distribuídas está divida em quatro projetos. O projeto 3 não tem ligação com os anteriores.

O objetivo geral do projeto será concretizar um serviço WEB para gerir um sistema simplificado de classificação de series de TV. A implementação vai utilizar o estilo arquitetural REST [1] e uma base de dados relacional acessível pela linguagem SQL. Para este efeito no servidor será utilizada a *framework* de desenvolvimento WEB *Flask* [2] e o motor de base de dados SQL *sqlite* [3]. O programa cliente utilizará o módulo *requests* [4] para implementar a interação cliente/servidor baseada no HTTP.

2. Esquema da base de dados

A definição da base de dados assenta nos conceitos envolvidos: utilizador, lista de series, classificação, serie, categoria, episódio. Cada lista de series criada por um utilizador contem uma ou mais series e cada serie está associada a uma categoria. Cada serie contem um ou mais episódios. Cada conceito corresponde a uma tabela de acordo com a figura seguinte, onde também se ilustram as várias relações.

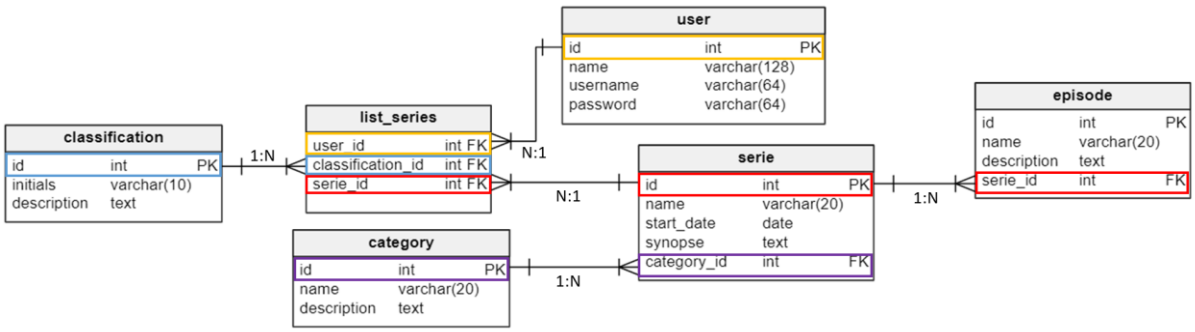

Figura 1 - Esquema da base de dados.

Para criar a base de dados poderá ser utilizado o seguinte código em SQL.

```
PRAGMA foreign_keys = ON;

CREATE TABLE users (
      id                INTEGER PRIMARY KEY,
      name              VARCHAR(128),
      username          VARCHAR(64),
      password          VARCHAR(64)
);

CREATE TABLE classification (
      id                INTEGER PRIMARY KEY,
      initials          VARCHAR(10),
      description       TEXT
);

CREATE TABLE category (
      id                INTEGER PRIMARY KEY,
      name              VARCHAR(20),
      description       TEXT
);

CREATE TABLE list_series (
      user_id           INTEGER,
      classification_id INTEGER,
      serie_id          INTEGER,
      FOREIGN KEY(user_id) REFERENCES users(id),
      FOREIGN KEY(classification_id) REFERENCES classification(id),
      FOREIGN KEY(serie_id) REFERENCES serie(id)
);

CREATE TABLE serie (
      id                INTEGER PRIMARY KEY,
      name              VARCHAR(20),
      start_date        DATE,
      synopse           TEXT,
      category_id       INTEGER,
      FOREIGN KEY(category_id) REFERENCES category(id)
);

CREATE TABLE episode (
      id                INTEGER PRIMARY KEY,
      name              TEXT,
      decription        TEXT,
      serie_id          INTEGER,
      FOREIGN KEY(serie_id) REFERENCES serie(id)
);
```

A primera linha desta listagem serve para que o *sqlite* possa suportar chaves estrangeiras. Por omissão, na instalação de alguns sistemas operativos, essa opção está desabilitada.

Os dados inseridos na tabela `classification e category` são os que se seguem e poderão ser inseridos pelo seguinte código SQL.

```
INSERT INTO classification (id, initials, description) VALUES
      (1, "M", "Mau"),
      (2, "MM", "Mais ou menos"),
      (3, "S", "Suficiente"),
      (4, "B", "Bom"),
      (5, "MB", "Muito bom")
;

INSERT INTO category (id, name, description) VALUES
      (1, "Ação", "Ação"),
      (2, "Animação", "Animação"),
      (3, "Artes Marciais", "Artes Marciais"),
      (4, "Aventura", "Aventura"),
      (5, "Biografia", "Biografia"),
      (6, "Clássico", "Clássico"),
      (7, "Comédia", "Comédia"),
      (8, "Drama", "Drama"),
      (9, "Ficção científica", "Ficção científica"),
      (10, "Musical", "Musical"),
      (11, "Policial", "Policial"),
      (12, "Romance", "Romance"),
      (13, "Suspense", "Suspense"),
      (14, "Terror", "Terror")
;
```

A aplicação pretendida deverá contemplar uma rotina de inicialização que verifica se a base de dados já existe. Caso não exista ela deverá ser criada e inicializada com o código apresentado (criação das tabelas e inserção dos registos nas tabelas `classification e category`).

3. O programa cliente

O programa cliente aceita interactivamente três operações e seus parâmetros (introduzidos via teclado numa consola), e comunica com o servidor para que este processe as operações e armazene a informação numa base de dados. A tabela 1 mostra detalhadamente as operações que o cliente deverá suportar.

Tabela 1 - Lista de operações que o cliente aceita e parâmetros correspondentes.

Operação	Parâmetros	Observações
ADD	- USER <nome> <username> <password> - SERIE <nome da serie> <data de inicio> <synopse> <categoria> - EPISODIO <nome do episódio> <descrição> <id da serie> - <id_user> <id da serie> <initials da classificação>	**categoria** = id, deve estar previamente na tabela category. A definição de um **episódio** implica a sua associação a uma serie. Será necessário indicar o <id da serie>. A última variante serve para adicionar, a uma serie, a respetiva classificação. A classificação é feita por iniciais (initials).

REMOVE ou SHOW	- USER <id do user> - SERIE <id da serie> - EPISODIO <id do episódio> - ALL <USERS | SERIE | EPISODIO> - ALL SERIE_U <id do user> - ALL SERIE_C <id da categoria> - ALL EPISODIO <id da serie>	As quatro últimas variantes permitem obter: • todos os utilizadores, ou series ou episódios definidos; • todas as series de um utilizador; • todas as series de uma categoria; • todos os episódios de uma serie.
UPDATE	- SERIE <id do user> <id da serie> <classificação> - USER <id do user> <password>	

Convém relembrar que os id's (chaves primárias) dos elementos nas diferentes tabelas são números inteiros.

O cliente comunicará com o servidor através de mensagens em HTTP e usará uma representação utilizando JSON [5]. Para este efeito os alunos utilizarão o módulo *requests* [4]. As mensagens terão de respeitar a API REST definida pelo serviço WEB.

4. O serviço WEB

O serviço WEB será implementado com recurso à *framework* Flask [2] e a API REST será disponibilizada através de três URLs de base:

1. /utilizadores
   Para operações relativas a utilizadores.
2. /series
   Para operações relativas a series.
3. /episodios
   Para operações relativas a episódios.

É muito importante que os alunos planeiem a API REST antes de iniciarem a implementação. Sugere-se que façam uma tabela onde definam a correspondência entre as operações suportadas, o método do HTTP, as URLs dos recursos, os parâmetros das operações, e as possíveis respostas HTTP com que o serviço responderá ao cliente.

Quando o serviço recebe uma mensagem de um cliente, a operação deverá ser implementada sobre a base de dados e a resposta será preparada segundo os padrões REST utilizando JSON para transmitir a representação dos recursos ou o conteúdo de outras mensagens.

Para o acesso à base de dados, os alunos devem procurar na documentação sobre Flask a forma de fazer com que a ligação à base de dados exista de forma automática sempre que a aplicação recebe um pedido.

5. Tratamento de erros

Sempre que a operação não possa ser executada ou que algum erro inesperado ocorra, o serviço deverá responder ao cliente com uma resposta HTTP incluindo uma descrição detalhada do

problema segundo o formato apresentado na aula TP07 sobre JSON. O cliente apresentará a informação da descrição detalhada na consola.

6. Check Point

O check point do projeto consiste em apresentar ao docente o funcionamento do projeto, demonstrando todas as funcionalidades deste, incluindo tratamento de erro. Assim, sendo o grupo de trabalho deve preparar uma bateria de testes para apresentar e responder às questões colocadas pelo docente. A classificação do projeto depende de ambas as partes, i.e., apresentação e respostas dadas.

O check point é realizado na semana de **30 de abril a 04 de maio, na aula da PL** onde os elementos de grupo pertencem, ou a maioria dos elementos de grupo. Todos os elementos do grupo têm de estar presentes, caso contrário os alunos em falta obtêm classificação zero.

7. Referências

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[2] http://flask.pocoo.org/

[3] https://www.sqlite.org/

[4] http://docs.python-requests.org/en/master/

[5] http://www.json.org/