**Computação Paralela e Distribuída**
**2023 / 2024**

Licenciatura em Engenharia Informática

Trabalho Prático #3 – API RESTful

## Introdução

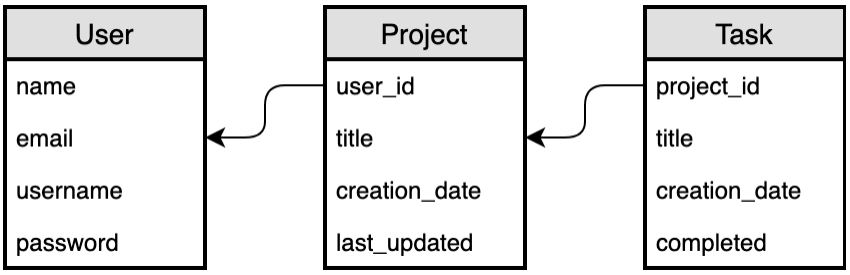
Pretende-se construir a aplicação *Tasklists*, um *software-as-a-service* para a gestão de projectos e tarefas. O *backend* será implementado usando a web framework *Flask* (http://flask.pocoo.org/). Como esta aplicação poderá ter vários *frontends* (web, mobile, etc), o objectivo deste trabalho será implementar a API *Restful*.

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

- **app.py** → código inicial da aplicação e as definições dos *endpoints REST*
- **models.py** → implementação do acesso a uma base de dados sqlite3
- **schema.sql** → código *sql* para criação da base de dados
- **tests.py** → código com unittests (incompleto)
- **static/** → pasta com *html* e código *javascript* que implementa pedidos à API

## Desenvolvimento

A informação na aplicação persiste numa base de dados SQLite3 com a seguinte estrutura:




O código fornecido contém uma implementação inicial que trata do acesso à base de dados e contém as definições dos *endpoints* a implementar. Estes são:

1. **Utilizador**:
   - `/api/user/register/` - Registar um utilizador.
   - `/api/user/` - Obter e modificar informação de um utilizador.

2. **Projectos:**
   - `/api/projects/` - Obter lista de projectos ou adicionar um projecto a um utilizador.
   - `/api/projects/<id>/` - Obter, editar ou remover um projecto a um utilizador.

3. **Tarefas:**
   - `/api/projects/<id>/tasks/` - Obter lista de tarefas ou adicionar nova tarefa.
   - `/api/projects/<id>/tasks/<id>/` - Obter, editar ou remover uma tarefa.

De notar que todos os *endpoints* da API apenas devem receber e/ou retornar JSON.

**Autenticação e autorização**

Todos os endpoints (excepto o *register*) requerem autenticação para se poder identificar o utilizador e verificar se este tem permissões para aceder ao projecto e às respectivas tarefas. O esquema de autenticação a usar será o *basic auth* (https://en.wikipedia.org/wiki/Basic_access_authentication).

O código fornecido contém um exemplo de autenticação no *endpoint* dos *users*. Para usar *Basic Auth* no Postman: https://learning.postman.com/docs/sending-requests/authorization/authorization-types/#basic-auth

## Testes

O ficheiro *tests.py* contém *unittests* cujo objectivo é testar o funcionamento correcto da API. Dado que os testes estão bastante incompletos, a implementação correcta destes será cotado como um extra. Caso utilizem ORMs (SQLAlchemy, etc.) deverão adaptar o ficheiro de testes de modo a utilizar correctamente o ORM.

## 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 30/Junho/2024.** Deverá também ser colocado no *zip* um ficheiro de texto com a identificação dos alunos, alguma descrição da solução que queiram fazer, e os 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) extensão da API para suportar mais funcionalidades – ex: mensagens privadas, novos endpoints para mudar apenas o completed das tarefas, etc; (2) *deployment* da aplicação na *cloud* (ex: *pythonanywhere.com*); (3) implementação de *unittests* relevantes; (4) utilização de ORMs (*SQLAlchemy*, *peewee*, etc.); (5) utilização de *plugins* que resolvam problemas concretos (*flask-admin*, *flask-mail*, *flask-restful*, http://flask.pocoo.org/extensions/, etc.); (6) implementação da aplicação web; (7) outros.. sejam criativos!

Bom trabalho!