Tutorial 2

Servidor WEB com Persistência

Objetivos:

- Criar um Servidor WEB com Persistência para gerir imagens

2.1 Introdução

Pretendemos criar uma aplicação *Web* para gerir imagens como se apresenta na Figura 2.1. Vamos considerar uma aplicação de utilização livre, ou seja sem identificação nem registo de utilizadores. Assim sendo a página de entrada apenas identifica a aplicação e tem uma barra de navegação para aceder às funcionalidades pretendidas. Posteriormente pode ser adicionado um sistema de registo e de identificação de utilizadores (*login/password*).

Figura 2.1: Página principal da aplicação *Web*

A aplicação deve permitir ao utilizador: carregar imagens à sua escolha (*Upload*) com identificação do autor, de um nome que a identifique e a data e hora de carregamento; vizualizar as images existentes, todas ou filtradas por um autor em particular; vizualizar uma imagem em particular com todos os comentários feitos pelos utilizadores; e acrescentar comentários a imagens. Também deve ter um sistema de pontuação de popularidade de uma imagem através de um sistema de votação de gostos/desgostos (*likes/deslikes*).

A segunda página ***Gallery*** irá listar todas as imagens presentes no sistema, eventualmente ordenadas por ordem ascendente do nome, permitindo a sua visualização. As imagens em sí estarão armazenadas no sistema de ficheiros da aplicação Web. Mas em vez de ser usado o nome original do ficheiro ela deve ser identificada por um nome criado a partir da síntese do conteúdo da própria imagem. Esta página apresentará uma lista de imagens e deverá permitir que elas sejam filtradas por um critério, por exemplo, pelo autor.

Quando uma imagem é selecionada deverá ser aberta uma nova página autónoma (***Image***) que apresenta todas as suas propriedades (autor, nome, hora e data de *upload* no sistema e comentários efetuados pelos utilizadores). Deverá ainda permitir que o utilizador acrescente um novo comentário à imagem selecionada. Deverá também ter um sistema de pontuação de popularidade da imagem (*likes/deslikes*).

A terceira página ***Upload*** permitirá o carregamento de uma nova imagem no sistema. Cada imagem terá associada um nome e o utilizador que a carregou no sistema, bem como a data e hora de carregamento.

A página ***About*** deverá conter informação sobre o(s) autor(es) da aplicação. Poderá eventualmente ter uma quarta página ***Image Manipulation*** que permita simular algoritmos de processamento de imagens.

2.2 Organização

Como já foi referido anteriormente, uma aplicação *Web* dinâmica com persistência é normalmente composta pelos elementos que se apresentam na Figura 2.2. Comparando com a aplicação *Web* dinâmica desenvolvida anteriormente temos mais o diretório `uploads` para armazenar os ficheiros que vão ser carregados pela aplicação e a base de dados `database.db` para registar toda a informação importante e persistente sobre as imagens.

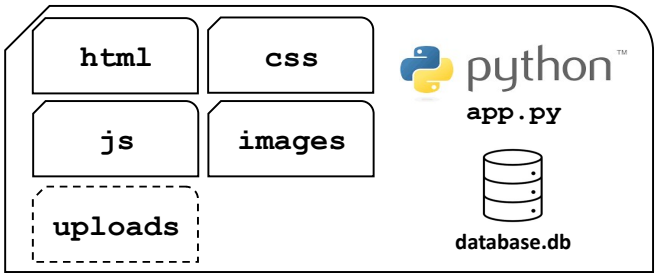

Figura 2.2: Aplicação Web com persistência

Exercício 2.1

No sítio da disciplina existe um arquivo ZIP chamado `TutorialWEB2.zip`. Obtenha este ficheiro, coloque-o dentro do diretório da disciplina e extraia o seu conteúdo.

Entre no diretório e compare o seu conteúdo (diretórios e ficheiros) com a Figura 2.2. Veja os ficheiros existentes em cada diretório.

De seguida execute a aplicação Web (`python3 app.py`). No navegador *Web* invoque a aplicação através do endereço `http://localhost:8080/`.

Aceda à página ***About*** e verifique que não tem identificação de autor e data. Complete a página (`about.html`) a seu gosto. Aceda também à página ***UpLoad*** e verifique que ela está diferente da página da aplicação WEB que foi criada no primeiro tutorial. Aceda ainda à página ***Gallery*** e verifique que ela está incompleta e não faz nada.

Neste tipo de aplicação *Web* com persistência todas as páginas HTML (exceção feita à página ***About***) têm que ter Javascript para implementar a dinâmica das páginas, seja para atualizar, consultar ou alterar a base de dados relativamente à imagem ou imagens que estão a ser processadas. Pelo que, associada a cada página haverá um ficheiro Javascript com o mesmo nome.

Para acrescentar a funcionalidade pretendida às páginas `upload.html`, `gallery.html` e `image.html` temos que começar por abordar a organização da base de dados que precisamos para armazenar a informação relativa às imagens.

2.2.1 Base de dados

Para programar a funcionalidade pretendida vamos precisar de uma base de dados com três tabelas (ver Figura 2.3). A primeira e mais importante tabela da base de dados é a que vai armazenar os dados mais importantes sobre a imagem e o seu armazenamento na aplicação, que vamos designar por `images`.

Os campos desta tabela são: o autor da imagem (`author`), ou seja o utilizador que carregou a imagem no sistema; o nome que o utilizador lhe atribuiu (`name`); a data e hora do carregamento da imagem (`datetime`); e o caminho para a imagem (`path`), ou seja o diretório onde se encontra o ficheiro. Poderíamos também armazenar a pontuação de popularidade da imagem nesta tabela, mas como vamos implementar essa funcionalidade mais tarde vamos optar por ter uma tabela à parte para guardar essa informação.

Vamos também precisar de uma tabela para armazenar os comentários da imagem, que vamos designar por `comments`, constituída pelos seguintes campos: o autor do comentário; o comentário propriamente dito e a data e hora a que ele foi feito.

Por sua vez a tabela de pontuação da popularidade de uma imagem gostos/desgostos (*likes/deslikes*), que vamos designar por **`votes`**, é constituída pelos seguintes campos: os votos positivos *likes*; e os votos negativos *deslikes*.

Todas as tabelas de uma base de dados relacional devem ter chaves primárias. Isto é ainda mais importante na tabela **`images`** porque vamos precisar de aceder às tabelas **`comments`** e **`votes`** a partir de uma imagem em concreto, por exemplo usando o seu número de ordem de carregamento no sistema. A chave primária da imagem será depois usada como chave estrangeira nas restantes tabelas para obter a restante informação acerca da imagem, como é o caso dos comentários e da votação de popularidade.

```
CREATE DATABASE database;

/* Images table */
CREATE TABLE images (
    id INTEGER PRIMARY KEY AUTOINCREMENT,  /* primary key */
    name TEXT,
    author TEXT,
    path TEXT,
    datetime TEXT
);

/* Comments table */
CREATE TABLE comments (
    id INTEGER PRIMARY KEY AUTOINCREMENT,  /* primary key */
    idimg INTEGER,  /* foreign key */
    user TEXT,
    comment TEXT,
    datetime TEXT
);

/* Votes table */
CREATE TABLE votes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,  /* primary key */
    idimg INTEGER,  /* foreign key */
    ups INTEGER,
    downs INTEGER
);
```

Figura 2.3: Descrição da base de dados

Exercício 2.2

Edite a descrição da base de dados da Figura 2.3 num ficheiro de código **`sqlite3`** designado, por exemplo, por **`database.sql`**. Depois invoque o **`sqlite3`** em modo interativo e crie a base de dados e as tabelas.

Claro que também vamos precisar de *queries* de atualização (`INSERT`), de consulta (`SELECT`) e de alteração (`UPDATE`) da informação destas tabelas, que aparecerão à medida que formos desenvolvendo a aplicação.

2.2.2 Página *Upload*

Uma aplicação *Web* para gerir imagens necessita obviamente de permitir o carregamento de imagens para posterior visualização e processamento. Por isso vamos começar por desenvolver a página de carregamento das imagens.

Na aplicação WEB que foi desenvolvida e apresentada no primeiro tutorial o carregamento das imagens era automaticamente executada assim que ela era selecionada. Nesta nova aplicação mais sofisticada vamos separar a visualização da imagem, do carregamento no servidor, até porque é necessário adquirir os dados relativos ao nome do utilizador que carrega a imagem e também do respetivo nome da imagem (ver Figura 2.4).

Para esse efeito a página `upload.html` tem dois blocos (marca `<div>`). O primeiro serve para fazer apenas a seleção da imagem e invocar a função `updatePhoto()` para visualizar a imagem selecionada. Esta atuação permite ao utilizador fazer múltiplas escolhas de imagens sem necessariamente as carregar no sistema. O segundo serve para fazer a aquisição dos dados da image e invocar a função `uploadImage()`. Esta nova função é que, após validar a existência dos dados, envia efetivamente a imagem para armazenamento. Para partilhar o ficheiro de imagem entre as várias funções existe a variável global `file`.

Figura 2.4: Página ***Upload***

Exercício 2.3

Complete o código da função `sendFile()` para obter os dados extras sobre a imagem e acabar de preencher o formulário do método `POST`.

Agora do lado do servidor temos que acrescentar ao método **upload** os novos parâmetros de entrada **nameImg** e **authorImg**. A informação horária **datetime** é obtida a partir do módulo **time**, enquanto que o campo **path** é dado pelo diretório **uploads** concatenado com o nome do ficheiro, na forma **uploads/*nome do ficheiro***. Depois é preciso definir o *query* apropriado para inserir a informação da nova imagem na tabela **images**.

Exercício 2.4

Complete e altere o código do método do servidor **upload** para armazenar a imagem no diretório **uploads** e armazenar os dados da imagem na tabela **images**.

Invoque a aplicação **app.py** aceda à página ***UpLoad*** e carregue uma imagem.

Verifique se a imagem foi efetivamente armazenada no diretório **uploads** e repare que o nome é dado por uma síntese **SHA256** do conteúdo da imagem. Usando o **sqlite3** em modo interativo verifique também a informação armazenada na tabela **images**.

2.2.3 Página *Gallery*

Esta página é constituída por um botão que permite selecionar as imagens que serão apresentadas. Quando a página é inicialmente carregada, ou quando não é selecionado qualquer autor, ou quando são selecionados todos os autores (**all**), serão apresentadas todas as imagens armazenadas. Se for indicado um autor específico serão apresentadas apenas as suas imagens.

As imagens são apresentadas no bloco (marca ** **) com identificação **id="showimages"**. Para esse efeito o código Javascript do ficheiro **gallery.js** invoca o método **list**. Este método quando invocado, devolve um objeto **json**, que se apresenta parcialmente na Figura 2.5, com a lista de todas as imagens existentes na aplicação *Web* carregadas pelo autor indicado, ou neste caso para todos os autores.

Cada imagem deverá conter a sua identificação **id** (número de ordem de registo na tabela), a data e a hora de criação, o seu nome, o utilizador que a carregou e o *path* para o ficheiro da imagem. Depois a função **showimages()** mostra as imagens precedidas pela respetiva informação tal como se mostra na Figura 2.6 onde se pode ver a primeira imagem e parte da segunda.

Exercício 2.5

Complete o código da função **showimages()** para mostrar as imagens com dimensão **width="550px"** e **height="450px"**. Complete também o método do servidor **list**.

Invoque a aplicação **app.py** aceda à página ***Gallery*** e verifique se consegue ver as imagens. Experimente alterar as imagens apresentadas usando como filtro o autor.

```
[
    {
        "id": "1",
        "name": "flor plastica",
        "author": "adrego",
        "path": "uploads/b51a48dc0062e60a6b5b...585475232711dec331b214e03f6823528c.jpg",
        "datetime": "date:27-08-2022 time:10:20:38",
    },
    {
        "id": "2",
        "name": "ribeira",
        "author": "adrego",
        "path": "uploads/4578c23ef23c398060ec...f2e74183347f88ed95cc0e93630ee08770.jpg",
        "datetime": "date:27-08-2022 time:10:21:07",
    },
    . . .
]
```


Figura 2.5: Objeto **`json`** com a lista de dicionários da informação das imagens

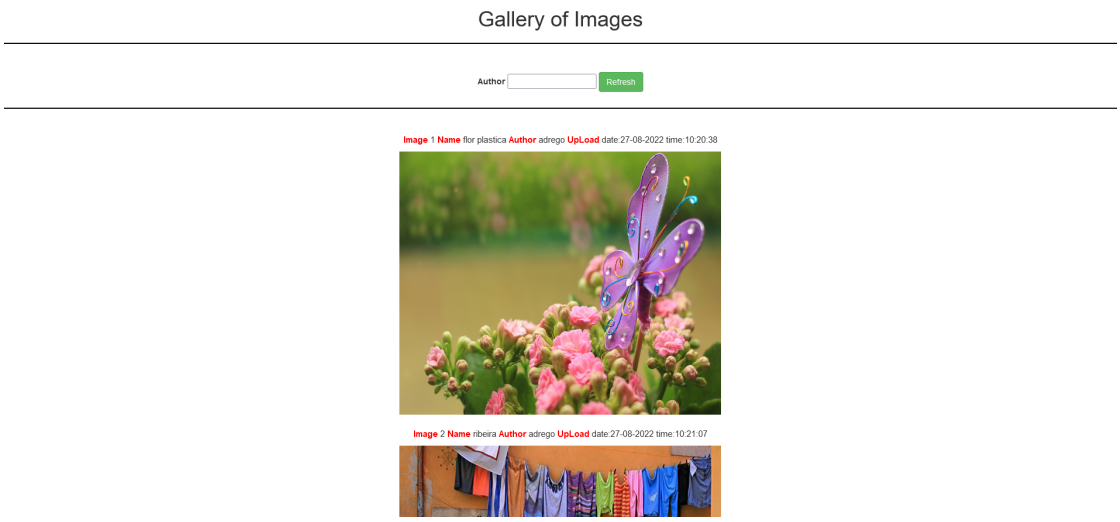

Figura 2.6: Página ***Gallery***

2.2.4 Página *Image*

Quando na página ***Gallery*** se clica sobre uma imagem a página ***Image*** da imagem selecionada é carregada, tal como se pode ver na Figura 2.7 para a primeira imagem da galeria de imagens. Esta página apresenta no lado esquerdo a imagem precedida pela respetiva informação e no lado direito todos os comentários efetuados pelos utilizadores. No rodapé temos a possibilidade de introduzir um novo comentário e de fazer a votação de gostos/desgostos.

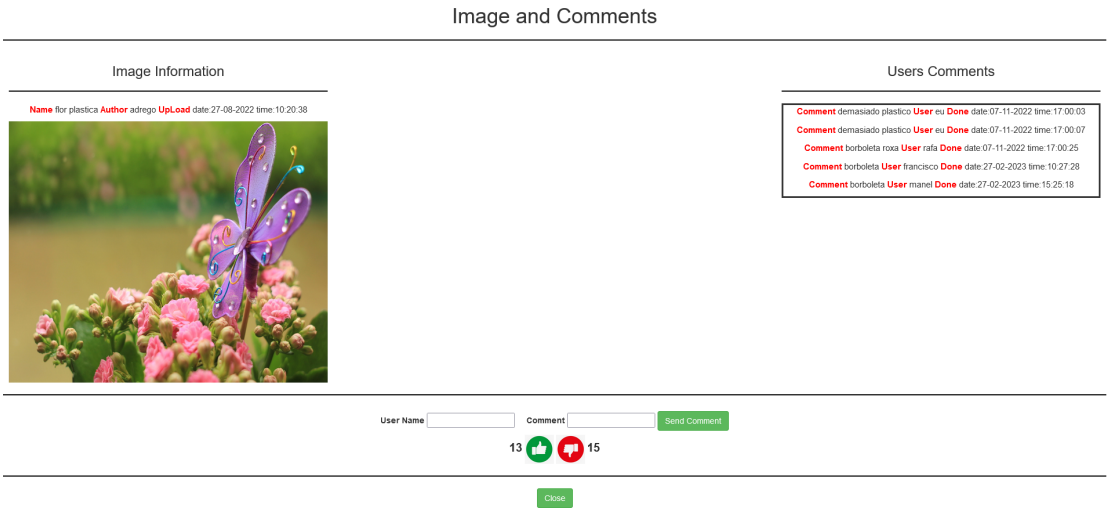

Figura 2.7: Página ***Image*** da primeira imagem

A Figura 2.8 apresenta uma imagem que não tem qualquer comentário nem votos.

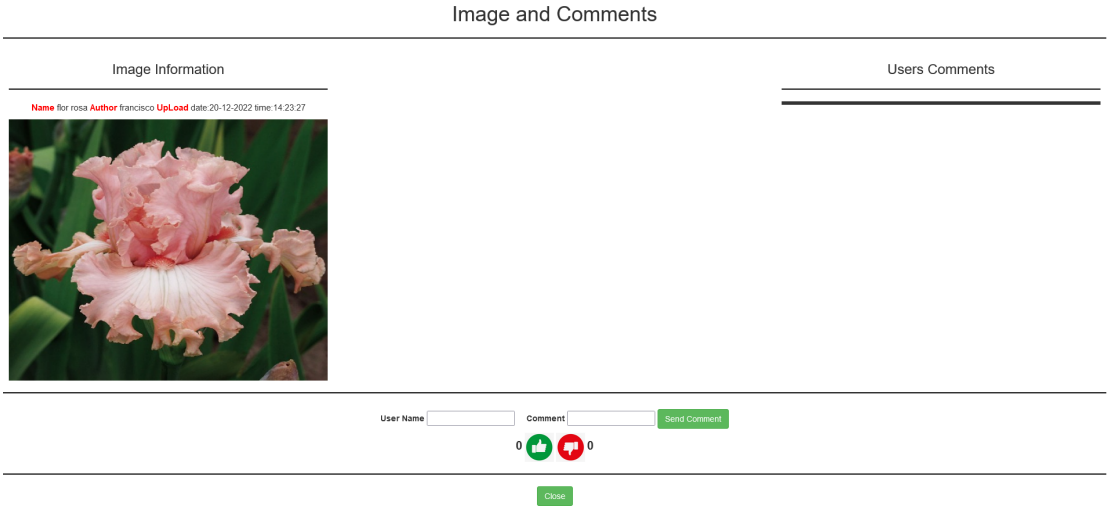

Figura 2.8: Página ***Image*** de outra imagem

Esta página **`image.html`** tem dois blocos (marca **`<div>`**) lado a lado. No lado esquerdo (**`id="imageinfo"`**) é desenhada a imagem precedida pela respetiva informação. E no lado logo direito (**`id="comments"`**) são escritos os comentários. Em rodapé temos um bloco que permite inserir um novo comentário e proceder à votação.

Para poder preencher estes dois blocos o código Javascript do ficheiro **image.js** invoca o método **comments**. Este método quando invocado, devolve um objeto **json**, tal como se apresenta na Figura 2.9, com a informação da imagem e todos os seus comentários.


```
{
    "id": "1",
    "name": "flor plastica",
    "author": "adrego",
    "path": "uploads/b51a48dc0062e60a6b5b...585475232711dec331b214e03f6823528c.jpg",
    "datetime": "date:27-08-2022 time:10:20:38",
    "comments": [
                  {
                    "id": 1,
                    "user": "eu",
                    "comment": "demasiado plastico",
                    "datetime": "date:07-11-2022 time:17:00:03",
                  },
                  { ... },
                  {
                    "id": 3,
                    "user": "rafa",
                    "comment": "borboleta roxa",
                    "datetime": "date:07-11-2022 time:17:00:25",
                  },
                  { ... }
    ]
}
```


Figura 2.9: Objeto **json** com a informação da imagem e comentários

Exercício 2.6

Invoque o **sqlite3** em modo interativo e introduza alguns comentários numa imagem.

Complete a função **showimageandinfo()**, para mostrar a imagem com dimensão **width="550px"** e **height="450px"**, e o método do servidor **coments**.

Invoque de novo a aplicação **app.py** aceda à página ***Gallery*** e depois verique se a página ***Image*** mostra a imagem e os seus comentários.

Verifique também se a página é bem desenhada mesmo para imagens sem comentários.

Complete a função **newcomment()** e o método do servidor **newcoment**.

Verifique se a página ***Image*** é redesenhada para mostrar o novo comentário.

Exercício 2.7

Invoque o **`sqlite3`** em modo interativo e introduza na tabela **`votes`** entradas para todas as imagens. Coloque algumas imagens com votos nulos e outros com valores positivos.

Complete a função **`showimageandinfo()`** para mostrar a votação.

Invoque de novo a aplicação **`app.py`** aceda à página ***Gallery*** e depois verique se a página ***Image*** mostra a imagem, os seus comentários e os votos.

Complete a função **`upvote()`** e o método do servidor **`upvote`**.

Invoque outra vez a aplicação **`app.py`** e verique se a página ***Image*** mostra a imagem, os seus comentários e os votos positivos devidamente atualizados.

Complete a função **`downvote()`** e o método do servidor **`downvote`**.

Invoque mais uma vez a aplicação **`app.py`** e verique se a página ***Image*** mostra a imagem, os seus comentários e os votos negativos devidamente atualizados.

2.3 Para Aprofundar

Esta aplicação *Web* pode ser muito útil, por exemplo, para servir de plataforma para simular algoritmos de manipulação e de efeitos sobre imagens. Assim podemos ter uma quarta página ***Image Manipulation*** para esse efeito. A Figura 2.10 apresenta esta página quando ela é acionada na página inicial da aplicação.

Figura 2.10: Página ***Image Manipulation*** inicial

Após ser selecionada uma imagem através do seu número de ordem de carregamento no sistema, ela é exibida no lado direito, tal como se apresenta na Figura 2.11. Utilizando o seletor de algoritmos e escolhendo, por exemplo, o algoritmo de trocas de canais de cor Verde/Vermelho a imagem alterada é exibida no lado esquerdo.

Figura 2.11: Página ***Image Manipulation*** final

Exercício 2.8

Acrecente à página **`index.html`** o código HTML necessário para ter mais uma entrada na barra de navegação designada por **`Image Manipulation`** ou outra designação à sua escolha.

Desenvolva uma página HTML designada por **`process.html`** e o respetivo código Javascript (**`process.js`**) para realizar a funcionalidade pretendida. Apresente as imagens na página com dimensão **`width="550px"`** e **`height="450px"`**.

Desenvolva também um método do servidor, designado por **`imageproc`**, que recebe como argumento o *path* do ficheiro da imagem que se pretende processar e que devolve o *path* do novo ficheiro da imagem criada pelo algoritmo de processamento.

Sugestão 1: Crie o ficheiro de saída dos algoritmos de processamento num diretório temporário, por exemplo, designado por **`tmp`** e implemente um mecanismo de remoção do ficheiro assim que ele não é necessário (método do servidor designado por **`deletefile`**).

Sugestão 2: Implemente os algoritmos de processamto de imagem num ficheiro python autónomo que deve ser importado na aplicação **`app.py`**.