Projeto Final

Objetivos:

- Planeamento e preparação do projeto final.

Este guião apresenta as regras do projeto final da disciplina de Laboratórios de Informática.

1.1 Regras

O trabalho deve ser realizado por um grupo de 4 alunos e entregue via a plataforma `http://elearning.ua.pt`, dentro do prazo lá indicado. A entrega deverá ser feita por **apenas um dos membros** do grupo e deve consistir de **um único ficheiro** ZIP com o código todo da aplicação WEB (tal como o colocaram no xcoa) e o relatório que deve incluir o pdf final e todos os ficheiros com código fonte (.tex, .bib, etc.) e todas as imagens ou outros recursos necessários à compilação do pdf.

Na elaboração do relatório recomenda-se a adoção do estilo e estrutura de relatório descrito nas aulas teórico-práticas e a utilização de recursos de escrita como: referências a fontes externas, referências a figuras e tabelas, tabela de conteúdo, resumo, conclusões, etc.

O relatório tem como objetivos descrever o enquadramento das tecnologias usadas, descrever a implementação, apresentar provas que demonstrem o seu funcionamento correto (p.ex, capturas de ecrã) e analisar os resultados obtidos (p.ex, analisar as capturas).

É obrigatório incluir uma secção "Contribuições dos autores" onde se descrevem resumidamente as contribuições de cada elemento do grupo e se avalia a percentagem de trabalho de cada um. Esta auto-avaliação poderá afetar a ponderação da nota a atribuir a cada elemento.

É obrigatório identificar claramente os autores, indicando também o seu número mecanográfico e o seu *email*. Também é obrigatório identificar claramente a área utilizada no servidor `XCOA` (*login* do grupo) e o nome do projeto criado na plataforma `Code.UA`.

1.2 Avaliação

A avaliação irá incidir sobre:

1. **cumprimento dos objetivos através das funções implementadas**,
2. **qualidade do código produzido e dos comentários**,
3. **testes práticos, funcionais e unitários realizados**,
4. **estrutura e conteúdo do relatório**,
5. **utilização das funcionalidades de tarefas do code.ua e git**,
6. **apresentação e discussão do trabalho**

Relatórios meramente descritivos sem qualquer descrição da aplicação, apresentação dos resultados obtidos, testes efetuados, ou discussão serão fracamente avaliados.

Só serão avaliados trabalhos enviados via a plataforma `http://elearning.ua.pt`. Ficheiros corrompidos ou inválidos não serão posteriormente avaliados e não será permitido o reenvio.

Muito Importante: verifique o número (N) de identificação do seu grupo na página de inscrição dos grupos de projeto.

Deve ser criado um projeto no `Code.UA` com o nome **labi2022gN**.

O projeto deve ficar a funcionar no servidor XCOA utilizando para esse efeito a conta **labiprojN**. Se ainda não alterou a senha desta conta, lembre-se que a senha original é igual ao nome de utilizador (labiprojN). Assim que seja possível um dos elementos do grupo deve alterar a senha (usando o comando `passwd`) e partilhá-la com os restantes elementos do grupo.

1.3 Tema Proposto

Este projeto visa criar uma plataforma digital de colecionismo de imagens digitais, ou de forma coloquial: uma caderneta digital de cromos. É proposto o desenvolvimento de uma plataforma Web Online que armazena as imagens originais de forma segura e que faz o acompanhamento das trocas digitais entre os colecionadores a que vamos chamar de **Editora**.

A editora terá utilizadores com dois perfis distintos:

- **Curadores** responsáveis por fazer *upload* das imagens;
- **Colecionistas** que podem requisitar imagens livres (não pertencentes a outro colecionista) e trocar imagens com outro colecionista.

A editora protege as imagens de roubo cifrando as mesmas com uma chave que apenas o administrador da Editora conhece e que a fornece no lançamento da plataforma web. Sempre que um curador faz *upload* de uma imagem, a mesma tem que ser armazenada de forma segura pelo servidor. Colecionistas podem consultar todas as imagens disponíveis, mas estas imagens contêm uma marca de água visível (gerada pelo servidor) com a identificação do utilizador que detém a imagem. Através da plataforma web, um utilizador pode ceder a propriedade de uma imagem a outro utilizador, ficando no entanto registado todo o histórico de transações.

O colecionista pode assim consultar não só as suas imagens mas também as restantes imagens e quem é o detentor das mesmas. O colecionista nunca terá acesso a nenhuma imagem sem marca de água.

Existirá valor na integração dos componentes e na disponibilização de um serviço completo, mas assume-se que os componentes poderão funcionar de forma isolada. Este aspeto também é vital para que seja possível testar o funcionamento isolado de partes do sistema.

1.4 Objetivos

O projeto deve consistir numa aplicação Web, com base de dados e código para processamento das imagens e cifras, que deverá ficar a funcionar no servidor XCOA.

O sistema proposto tem de ser composto pelos seguintes componentes:

- **Interface Web**
- **Aplicação Web**
- **Persistência**
- **Processador de Imagens - cifra & marca de água**

1.4.1 Interface Web

Este componente deve ser composto por um mínimo de 5 páginas (funcionalidades), desenvolvidas através de HyperText Markup Language (HTML)[1], JavaScript (JS)[2] e Cascading Style Sheets (CSS)[3], fornecendo o único interface para interação com o sistema. Desde que as funcionalidades aqui pedidas sejam mantidas, os alunos poderão adicionar outras páginas e outras funcionalidades.

A primeira página deverá apresentar uma caixa de *login* com um *link* adicional para uma página dedicada ao registo de novos utilizadores.

Após um *login* com sucesso o utilizador deverá ter acesso, na segunda página, a uma lista de coleções de imagens e à opção de criar uma nova coleção (assumindo desta forma o papel de curador).

Optando por criar uma nova coleção, na terceira página, deverá proceder ao envio de uma ou mais imagens para o servidor. Para este efeito deverá enviar uma imagem de cada vez através do seu *browser*.

Acedendo a uma coleção, na quarta página, deverá ter acesso a duas opções:

- Poder ver todas as imagens da mesma coleção que tenham um proprietário. **Atenção** todas a imagens deverão conter uma marca de água. Cada imagem deverá ainda apontar para uma página com as propriedades dessa imagem (detentor da imagem, histórico de transações). Nessa mesma página deverá ser possível iniciar a transferência de propriedade de uma imagem, caso seja seu proprietário, para outro utilizador apenas com base no seu *username*;

- Poder adquirir uma imagem que ainda não tenha propriedade (*draft*).

A quinta página deverá conter informação sobre os alunos e o repositório utilizado.

Todas as páginas deverão obter informação através de pedidos à aplicação Web, que deverá devolver um objeto JavaScript Object Notation (JSON)[4]. Excetuando a página inicial e de registo, todas as páginas deverão estar protegidas por um *access_token* que não é mais do que uma *string random* que representa um utilizador autenticado.

1.4.2 Aplicação Web

A aplicação Web consiste num programa `python` que serve conteúdos estáticos (`html`, `css`, `js`), apresentando métodos que permitem a navegação entre os diversos componentes.

Irá também fornecer uma interface programática que permita obter informação, ou inserir informação relativa às imagens, autores e comentários efetuados. Esta interface irá responder sempre com objetos JSON.

Assume-se que as imagens são identificadas por um identificador criado de uma síntese do conteúdo da imagem.

Deverão ser considerados pelo menos os seguintes métodos:

- **`/users/auth`**: Devolve um *access_token* em troca de um *username* e *password* válidos;

```
{
    "authentication": "OK",
    "token": "slhquJSO"
}
```

- **`/users/create`**: Cria um novo utilizador com o *username* solicitado e uma *password* gerada de forma aleatória com apenas caracteres *ascii* válidos.

```
{
    "username": "dgomes",
    "password": "BrTTMJpH"
}
```

- **`/users/valid`**: Permite testar se um dado token ainda é válido.

- **`/cromos/id`**: Lista todas as imagens/cromos. Se fornecido um *id*, apenas os cromos dessa coleção.

```
[
    [1, "path/para/imagem.jpg", "carla", [
                                        ["ts": 101230132, "ana"],
                                        ["ts": 101250132, "bernardo"]
                                    ],
    [2, "path/para/imagem.jpeg", "ana", ""]
]
```

- **`/cromos/name/create`**: Faz *upload* de uma imagem para a coleção com o nome *name*.

```
{
    "id": "id da colecção",
    "image_id": "id da foto"
}
```

1.4.3 Persistência

Este componente deverá ser composto por métodos que permitem o registo de informação numa base de dados relacional e a obtenção de informação da mesma. Os métodos serão utilizados pela Aplicação Web para registar uma referência para as imagens enviadas e os respetivos curadores (utilizadores que detêm as imagens).

Deverá ser utilizada uma base de dados `SQLite3`, localizada no mesmo diretório da aplicação.

Os ficheiros contendo as imagens deverão ser armazenados no sistema de ficheiros, sendo que a base de dados relacional apenas terá informação sobre as imagens. Isto é, a base de dados terá informação sobre a data de inserção, comentários e outras propriedades, mas o ficheiro em si não deverá ser armazenado na base de dados. Recomenda-se que as imagens sejam armazenadas com base no seu identificador.

É obrigatório que o acesso à base de dados seja realizada através de métodos desenvolvidos pelos alunos, correspondendo a todas as ações previstas pela aplicação Web (obter listagem de imagens, obter autenticação dos utilizadores, obter informação de uma imagem, obter nome do ficheiro de uma imagem).

1.4.4 Processador de imagens

Este componente, que executa como parte da aplicação Web, terá métodos para lidar com as imagens enviadas para o sistema ou a obtenção das mesmas. Em particular ele deverá suportar o armazenamento e obtenção de uma imagem.

1.4.5 Testes

Todos os métodos com caráter público dos módulos de persistência, processamento de imagens devem ser cobertos com testes unitários. Estes testes devem no mínimo verificar o funcionamento correto de cada método. Deverão igualmente ser consideradas situações de erro, tais como o envio de argumentos em formatos incorretos ou com valores incorretos.

Os métodos expostos pela aplicação Web deverão ser igualmente testados, através da realização de pedidos HyperText Transfer Protocol (HTTP)[5]. Recomenda-se a utilização da biblioteca **`python requests`**. Esta biblioteca permite o envio de pedidos específicos e a obtenção do objeto JSON com o resultado, que pode depois ser validado.

1.5 Bónus

Cada projeto poderá beneficiar de até 2 valores de bónus, caso implemente funcionalidades adicionais de relevo, sendo a sua atribuição definida pelos docentes. Recomenda-se que os alunos discutam potenciais bonificações antes da entrega final.

1.6 Notas Relevantes

1.6.1 Portas TCP

Como estudado, o sistema Linux apenas permite criar um socket em cada porta da mesma família. Não será possível executar múltiplas aplicações, num mesmo servidor, sem que se tomem precauções adequadas. Desta forma, é obrigatório que cada grupo execute a sua aplicação numa porta distinta. Neste caso estarão atribuídas as portas $10000 + N$ em que N é o número do grupo.

Depois será possivel aceder ao `XCOA` de forma a que o servidor redirecione os pedidos HTTP para a aplicação `cherrypy` correta.

Como exemplo, no caso do grupo 8, será possível aceder a `http://xcoa.av.it.pt/labiproj8`, sendo que estes pedidos serão autenticados e reenviados para uma aplicação à escuta na porta 10008.

Um acesso à listagem de imagens deste grupo deverá ser feita acedendo ao URL `https://xcoa.av.it.pt/labiproj8/cromos/`, sendo que o servidor `XCOA` irá reenviar a informação para uma aplicação no endereço `http://127.0.0.1:10008/cromos`.

De forma a iniciar uma aplicação numa porta alternativa, o seguinte excerto pode ser utilizado como base:

```
import cherrypy
cherrypy.config.update({'server.socket_port': 10008,})

class HelloWorld(object):
    def index(self):
            return "Hello World!"

    index.exposed = True

cherrypy.quickstart(HeloWorld())
```

1.6.2 Javascript

Uma componente relevante do projeto assenta na utilização de Javascript e no acesso aos métodos expostos pela Aplicação Web. Recomenda-se que os alunos consultem a documentação de módulos como o **`JQuery`**, em particular as funcionalidades relacionadas como o acesso a recursos externos (ex. ajax get e post).

De acordo com a documentação presente em `https://api.jquery.com/jQuery.get/`, o exemplo seguinte é relevante para a obtenção de objetos fornecidos pela Aplicação Web.

```
$.get( "api/list?start=0&filter=cats,dogs&order=date")
  .done(function(data) {
    // Got a JSON object from the server.
    console.log(data)
  })
  .fail(function() {
    console.log( "error getting the image list" );
  });
```

Da mesma forma existe um método que permite enviar informação através de um pedido HTTP `POST`: `https://api.jquery.com/jQuery.post/`.

```
$.post( "/api/comment/add", { author: "John", comment: "great pic!" })
  .done(function( data ) {
    // Got answer from server
    console.log(data);
  });
```

Referências

[1] W3C. «HTML 4.01 Specification». (1999), URL: `http://www.w3.org/TR/1999/REC-html401-19991224/`.

[2] ECMA International, *Standard ECMA-262 – ECMAScript Language Specification*, Padrão, dez. de 1999. URL: `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.

[3] W3C. «Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification». (2001), URL: `http://www.w3.org/TR/2011/REC-CSS2-20110607/`.

[4] E. T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 7159, Internet Engineering Task Force, mar. de 2014.

[5] R. Fielding, J. Gettys, J. Mogul et al., *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), Updated by RFCs 2817, 5785, 6266, Internet Engineering Task Force, jun. de 1999.

Glossário

CSS Cascading Style Sheets

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JS JavaScript

JSON JavaScript Object Notation

URL Uniform Resource Locator

ZIP