Projeto 2

Objetivos:

- Planeamento e preparação do projeto 2.

Este guião apresenta as regras do segundo projeto 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 e o relatório final (em PDF).
Depois da entrega, irá existir uma defesa do projeto, onde se pretende que os alunos apresentem o trabalho entregue, potencialmente explicando as decisões que tomaram durante a implementação.

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 comprovem 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.

É obrigatório identificar claramente a área utilizada no servidor `XCOA` e o projeto 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 avaliados à posteriori e não será permitido o reenvio.

Deve ser utilizado um projeto na plataforma Code.UA com um identificador no formato `labi2019-p2-gX`. Substitua o caratere `X` pelo valor 1. Se não for possível criar este projeto, incremente o número até que seja aceite. Não salte números.

1.3 Tema Proposto

O objetivo do projeto é a criação de um sistema que permita visualizar uma pequena biblioteca de imagens, com suporte para a procura e identificação dos objetos contidos nas imagens armazenadas (gatos, automóveis, pessoas, ...) . A área científica relacionada com o reconhecimento e classificação em imagens, assim como outras técnicas de aprendizagem, é muito importante para o mundo atual. Este trabalho irá permitir fazer uso de soluções que apliquem estas técnicas, numa vertente de caixa negra (irão usar a tecnologia, e não desenvolver notas técnicas de reconhecimento e classificação). Alguns dos conceitos específicos de reconhecimento de objetos serão abordados noutras disciplinas de caráter mais avançado.

Um pequeno tutorial e uma demonstração de alguns destes conceitos podem ser encontrados nos vídeos `https://www.youtube.com/watch?v=4eIBisqx9_g` e `https://www.youtube.com/watch?v=MPU2HistivI`.

1.4 Objetivos

O projeto deve consistir numa aplicação web (página mais aplicação), com base de dados e código para processamento das imagens, que deverá ficar a funcionar no XCOA.

O sistema proposto tem de ser composto pelos seguintes componentes:

- **Interface Web**
- **Aplicação Web**
- **Persistência**
- **Processador de Imagens**

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. Deverá fazer uso de soluções como o Twitter bootstrap ou semelhante.

A primeira página irá apresentar uma listagem com todos os nomes de objetos (classes) já detetados, de imagens inseridas no sistema. Os alunos poderão escolher como apresentar esta lista, podendo-se considerar a existência de uma pesquisa, iconografia específica, ou simplemente os termos.

A segunda página irá apresentar pequenas imagens com todos objetos de um determinado tipo. Estes objetos serão extraídos das imagens como descrido posteriormente neste guião. Deverá igualmente permitir a especificação de um limiar de confiança da deteção, numa escala de 0 a 100. Por defeito deverá ser utilizada uma confiança de 50.

A terceira página irá permitir o envio de uma nova imagem para o sistema.

A quarta página irá permitir listar objetos através de uma pesquisa, combinando a cor e o tipo de objeto. Por exemplo, deverá permitir pesquisar por "red car", resultando na apresentação de potenciais carros com cor predominantemente vermelha. Para a determinação do que é vermelho (ou outra qualquer cor), poderão assumir-se valores limites nas componentes de imagem definidos pelos alunos. No caso particular da cor vermelha, no modo de cores RGB, pode-se considerar que o canal **R** (Vermelho) é muito intenso, enquanto os restantes não o são.

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

1.4.2 Aplicação Web

A aplicação web consiste num programa **python** que serve conteúdos estáticos (**html**, **css**, **js**). Apresenta métodos que permitem o fluxo de informação entre os diversos componentes.

Em particular, esta aplicação deverá expor as seguintes funções (podendo expor mais):

- **/list?type=names**: Devolve um objeto JavaScript Object Notation (JSON)[4] contendo a lista de objetos que já foram detetadas pelo sistema.
- **/list?type=detected**: Devolve um objeto JSON contendo a lista das pequenas imagens contendo os objetos extraídos. Um exemplo da possível resposta é apresentado de seguida. Este exemplo considera que foram enviadas duas imagens (7612328513f8ce4b2c2e70c31f6dfb43 e 8a76af9d0f1a56f16e44a7a3ec4c4cd6), resultando em 3 objetos extraídos (2 para a classe **car** e um para a classe **boat**):

```
[
    "car": [
        {
         "original": "7612328513f8ce4b2c2e70c31f6dfb43",
         "image": "fe5a11e3882f74dd72dab1885f9edcb7",
         "confidence": 70
        },
        {
         "original": "7612328513f8ce4b2c2e70c31f6dfb43",
         "image": "c5f79160f12f729696ace449c7ede98c",
         "confidence": 34
        }
    ],
    "boat": [
        {
         "original": "8a76af9d0f1a56f16e44a7a3ec4c4cd6",
         "image": "9c2210d8d18533f7d8a771585b79ec88",
         "confidence": 99
        }
    ]
]
```

- **/list?type=detected&name=NAME**: Devolve um objeto JSON contendo a lista das pequenas imagens contendo os objetos extraídos, que sejam relativos a um objeto. O resultado será semelhante ao anterior, mas com o filtro considerado.
- **/list?type=detected&name=NAME&color=COLOR**: Devolve um objeto JSON contendo a lista das pequenas imagens contendo os objetos extraídos, que sejam relativos a um objeto. O resultado será semelhante ao anterior, mas com o filtro considerado.

- `/put`: Permite enviar uma nova imagem para ser processada e armazenada.
- `/get?id=IDENTIFIER`: Permite obter uma imagem completa, ou imagem extraída com um objeto, através do identificador fornecido. O identificador deve corresponder a uma síntese do conteúdo do próprio ficheiro a obter.

Opcionalmente, os métodos que listam objetos poderão aceitar um argumento `threshold` que filtra imediatamente a confiança das deteções.

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 as imagens enviadas, os objetos detetados em cada imagem, os objetos extraídos, e as cores.

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 a mesma.

É obrigatório que o acesso à base de dados seja realizada através de métodos desenvolvidos pelos alunos. Estes métodos deverão representar também a ação a realizar (p.ex, `search_object_by_name(name)`)

1.4.4 Processador de imagens

Este componente deverá processar uma imagem recém-enviada para o sistema, realizando as diversas funções necessárias.

Em primeiro lugar a imagem deverá ser enviada para um sistema de classificação. Este encontra-se disponível no endereço `http://image-dnn-sgh-jpbarraca.ws.atnog.av.it.pt/` e devolve um objeto JSON com as deteções obtidas de uma imagem enviada. As deteções incluem o nome, a confiança e as coordenadas.

O serviço pode ser testado manualmente, ou acedido programaticamente. O exemplo seguinte envia uma imagem `image.jpg` para deteção e imprime o resultado:

```
import requests

session = requests.Session()
URL="http://image-dnn-sgh-jpbarraca.ws.atnog.av.it.pt/process"
with open('image.jpg', 'rb') as f:
    file = {'img': f.read()}
```

```
r = session.post(url=URL, files=file, data=dict(thr=0.5))
if r.status_code == 200:
    print(r.json())
```

Imagens que não contenham qualquer objeto detetado devem ser ignoradas.

De seguida será necessário recortar os objetos detetados, criando imagens individuais para cada objeto. Tanto estas imagens, como a imagem originalmente enviada, devem ser referenciadas na base de dados.

Finalmente será necessário analisar os objetos e determinar que cores possuem. Deverá considerar-se apenas a cor dominante do objeto (a que possui mais pixeis dessa cor).

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 possível aceder ao `XCOA` de forma a que o servidor redirecione os pedidos HyperText Transfer Protocol (HTTP)[5] para a aplicação `cherrypy` correta.
Como exemplo, no caso do grupo 8, será possível aceder a
`http://xcoa.av.it.pt/labi2019-p2-g8`, 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/labi2019-p2-g8/list?type=names`, sendo que o servidor `XCOA` irá reenviar a informação para uma aplicação no endereço
`http://127.0.0.1:10008/list?type=names`.

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())
```

Glossário

CSS Cascading Style Sheets

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JS JavaScript

JSON JavaScript Object Notation

URL Uniform Resource Locator

Referências

[1] W3C. (1999). HTML 4.01 Specification, 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. (2001). Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, 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, H. Frystyk, L. Masinter, P. Leach e T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), Updated by RFCs 2817, 5785, 6266, Internet Engineering Task Force, jun. de 1999.