Prática CSLP - 1

por Diogo Silva, 89348 e Pedro Oliveira, 89156

Introdução

Serve o seguinte relatório como reportório, explicação e prova de execução dos exercícios propstos no primeiro guião prático da disciplina de Complementos de Linguagens de Programação (CSLP).

Projetos Escolhidos

Para esta prática foi nos pedido que escolhessemos 2-3 projetos criados em prévias cadeiras, de forma a que pudessemos, sobre estes, aprender os conceitos de Documentação, GIT e Docker-Containers. Escolhemos então os seguintes projetos:

SpamBot Finder

- **Cadeira**: Métodos Probabilísticos para Engenhaira Informática
- **Autores**: Diogo Silva, Vasco Rámos
- **Linguagem**: Java (8)
- **Utilização**:
    - A) Executando os comandos (através do diretório MPEI):
        - $javac src/*.java
        - $java src.actualFinalTest
    - B) Abrindo o diretório 'Old Eclipse Files' no Eclipse IDE e executando através daí ;
- **Descrição**: Este projeto baseia-se na utilização dos vários conceitos aprendidos nas aulas de MPEI e aplicação destes a um problema real. Foram utilizados neste projeto conceitos como procura de similaridades com uso de MinHash, distáncias de Jacqard, contadores estocásticos, counting bloom filters, entre outros. A ideia base deste projeto consiste na procura de e identificação de possíveis "Spambots" através da analise de um conjunto de críticas de jogos postadas por utilizadores no site GoodOlGames.com através da analise e procura de reviews similares.

Drive-thru Simulation with a Token Ring

- **Cadeira**: Computação Distribuida
- **Autores**: Pedro Oliveira
- **Linguagem**: Python3
- **Utilização**:
    - A) Executando no terminal:
        - python3 simulation.py
        - python3 client.py (tantas vezes quantos clientes quiser testar);
    - B) Executando o programa bash ./run.sh, que irá iniciar o simulation.py e 20 client.py;
- **Descrição**: Este projeto baseia-se na implementação de um Token Ring, uma implementação de uma network P2P, para simular um restaurante. Foram implementadas 4 entidades, um restaurante, um

chefe, um empregado e um rececionista; cada um com uma função diferente para a simulação. Para obter a comunicação entre estes atores, foi implementada uma num Node genérico que cada um vai usar. Estes nós são os que se vão ligar e formar o Token Ring e, a partir daqui, vai ser possivel transmitir mensagens entre cada um.

Flappy Bird Machine Learning

- **Cadeira**: -
- **Autores**: Diogo Silva, Pedro Escaleira
- **Linguagem**: Python
- **Utilização**:
    - A) Executando os comandos (através do diretório flappy_bird_ml):
        - $python3 main.py
- **Descrição**: Este projeto foi realizado para diversão, não estando portanto associado a nenhuma UC. Como introdução, tanto a desenvolvimento de jogos utilizando o PyGame, como à criação de um algorítmo genético de machine learning, este projeto baseiou-se na execução, tanto de um "Flappy Bird clone", como num algorítmo que gerasse vários agentes inteligentes, que fossem evoluindo de geração em geração (sendo que cada geração termina assim que todos os membros desta morrerem).

Repositório GitHub

Todos os ficheiros, tanto desta como de todas as próximas práticas, poderá ser encontrado no repositório:

https://github.com/HerouFenix/CSLP

Para esta prática em especifico, basta navegar para o diretório Prática1.

Posteriormente, todos os ficheiros relativos a um projeto (código, Doxyfile, Dockerfile, docs, etc.) encontra-se no subdiretório com o nome do projeto.

Docs

Para encontrar a documentação, gerada pelo Doxygen, relativa a cada um dos projetos basta navegar para o diretório: **Pratica1 > *Nome do projeto* > docs**

Os Doxyfiles utilizados para a geração das documentações pode ser encontrada na pasta raiz de cada projeto (Pratica1 > *Nome do projeto*)

Dockerhub

SpamBot Finder

- **Dockerhub**: https://cloud.docker.com/repository/registry-1.docker.io/fenixds/spambot-finder
- **Utilização**:
    - $sudo docker build --tag=spambotfinder .

    - $sudo docker run -i spambotfinder
- **Dockerfile**:

```
 FROM openjdk:8
COPY . /app
WORKDIR /app
RUN javac src/*.java
CMD ["java", "src/actualFinalTest"]
```

- **Notas**: Na altura, este projeto foi criado em Eclipse-IDE. O problema com isto é a que a gestão das packages feita automáticamente pelo Eclipse não se aplica aquando da compilação do source code por terminal (i.e, quando executavamos o comando $javac algumas classes de Teste que dependiam de classes Modules não estavam a conseguir ser compiladas por não conseguirem encontrar estas ultimas). Como forma de evitar dores de cabeças desnecessárias, alteramos a estrutura do projeto, metendo todos os ficheiros .java num subdiretório da raiz do projeto - src, e todos os ficheiros de suporte (como os ficheiros de texto com a informação analisada), num outro subdiretório da raiz do projeto - Files.

CD

- **Dockerhub**: https://cloud.docker.com/repository/registry-1.docker.io/drpunpun/drivethrup2p
- **Utilização**:
    - $sudo docker build --tag=drivethrup2p .
    - $sudo docker run -i drivethrup2p
- **Dockerfile**:

```
 FROM ubuntu
#Get python3
RUN apt-get update && apt-get install -y python3 procps
WORKDIR /project
COPY . /project
CMD [ "./run.sh" ]
```

- **Notas**: Para poder correr este programa, sempre foi preciso correr mais que um processo python. Na altura, isto foi resolvido usando um programa bash que lança um terminal por processo. Como na altura o docker não reconhecia o comando gnome-terminal (para abrir um terminal), o projeto foi ligeiramente alterado para, em vez de imprimir para o terminal, escrever um ficheiro à parte. Para terminar o processo da simulation, também foi preciso usar alguma logica adicional: na altura apenas foi preciso usar ctrl-c para terminar o projeto; mas agora, como estou a escrever num ficheiro, tenho de terminar o processo usando comandos bash. Para isso, foi preciso identificar o processo (pelos comandos ps a | grep simulation.py) e termina-lo. Então, para correr estes processos, precisei de usar uma imagem Ubuntu e instalar os programas necessários (python3 e procps)

Flappy Bird Machine Learning

- **Dockerhub**: https://cloud.docker.com/repository/registry-1.docker.io/drpunpun/flappy_bird_ml

- **Utilização**:
  - $docker build --tag=flappy_bird_ml .
  - $docker run flappy_bird_ml
- **Dockerfile**:

```
 # Use an official Debian runtime as a parent image
FROM debian:buster
# Set the working directory to /game
WORKDIR /game
# Copy the current directory contents into the container at /game
COPY . /game
# Install pip
RUN apt-get update && apt install -y python3-pip
# Install numPy
RUN pip3 install numpy
# Install pyGame
RUN pip3 install pygame
# Run the game
CMD ["python3", "main.py"]
```

- **Notas**: Este projeto tem o problema de precisar de usar uma video output para poder funcionar (pela natureza do pygames). Isto resultou em problemas ao fazer o Dockerfile, pois não soubemos como configurá-lo para aceitar um video device.

Git Activity

- b8e71b9 Merge pull request #3 from HerouFenix/feature/pratica1/CD_docker |\
  | * d730281 done the CD project's dockerfile |/
- 9ff25a8 Merge pull request #2 from HerouFenix/feature/pratica1/CD_doc |\
  | * f2ab969 merged conflicts | * 963db46 Merge remote-tracking branch 'origin' into feature/pratica1/CD_doc | |\
  | * | 7b1d350 Finished most of the documentation | * | b712847 Merge remote-tracking branch 'origin' into feature/pratica1/CD_doc | |\ \
  | * | | bf3725e finished commenting 3 classes, 2 remaining | * | | 873cf7a Started CD documentation
- | | | bb4c173 Merge pull request #1 from HerouFenix/feature/pratica1/flappy_docker |\ \ \ \
  | |/
  |/| | |
  | * | | 00c05b4 fixed conflicts | |\ \ \
  | |/ / /
  |/| | |
- | | | da02715 Added relatorio v0.7
- | | | ea24878 added exercise sheets
- | | | df60d29 some more minor changes
- | | | 4d08587 some changes to folder
- | | | c09854a Created container for MPEI | * | | 2e9ea54 tried docker on flappy bird | * | | a1bc175 first |/ /

/

- | | 6338104 finished documenting flappy
- | | d1935fa Merge remote-tracking branch 'origin' into feature/pratica1/flappy_bird_doc |\ \ \
  | | |/
  | |/|
  | * | 56acfd4 Update README.md | * | 61aea0c Update README.md | * | a45f0c3 Create LICENSE | * |
  8e42492 Create README.md
- | | 35f6ff9 Added doxyfile
- | | 610a48d Merge remote-tracking branch 'origin' into feature/pratica1/flappy_bird_doc |\ \ \
  | |/ /
  | * | 9ba5454 Added Doxyfile | |/
- | a090572 Started flappy bird documentation |/
- bf3634c Added Doxygen documentation + Refactored Project comments
- 645044d First push containing projects required for first practical class
- 5531eff Updated gitignore
- e5a8de5 Initial commit

> Written with StackEdit.