UNIVERSITÀ DI PISA

Computer Engineering

Performance Evaluation of Computer Systems and Networks

Slotted random-access wireless network

Group Project Report

TEAM MEMBERS:
Tommaso Burlon
Francesco Iemma
Olgerti Xhanej

Academic Year: 2020/2021

Contents

Chapter 1

Introduction

PokèMongo is a gaming application in which users compete each other to build up the best Team choosing from the set of Pokemon available in the environment. Every user can make just one single Team.

1.1 Description

Every Team is composed by up to 6 distinct Pokemons and is assigned to a numerical value based on features and properties of the chosen Pokemons, for ranking purposes.

Users can also navigate through the ranking in order to visualize the best teams (according to the values cited before), most used/caught Pokemons.

The user can also search a specific Pokemon using the Pokedex tool, in which he/she can browse Pokemons according to specific search filters (e.g. Pokemon name, Type, Points...).

Moreover, as a "real" Pokemon Trainer, the user is invited to "Catch 'em 'all", i.e. to catch Pokemon in order to create/update his own team. Thus, it is provided to the user a prefix number of daily Pokeball to be used to try to catch them.

At each Pokemon is associated a probability to catch it, the higher the Pokemon's value, the lower the probability.

Under discussion are the following ideas:

- Creating a "social" structure in which users can follow each other in order to share his/her own team
- Creating a chat system to pair with the social structure
- Reduce catchable Pokemons to a daily subset of the entire Pokemon Database

1.2 Code Snippets

Other things: let's show some code snippets!

```
    import requests
    import json


    #exampleW
    new_json = []
    description = ""

    for i in range(500, 894):
        response = requests.get(f"https://pokeapi.co/api/v2/pokemon/{i}/")
        work_string_json = response.json()
        response = requests.get(f"https://pokeapi.co/api/v2/pokemon-species/{i}/")
        work_string_json2 = response.json()

        for desc in work_string_json2['flavor_text_entries']:
            if(desc['language']['name'] == "en"):
                description = desc['flavor_text']
                break

        curr_json = {
            "id": work_string_json['id'],
            "name": work_string_json['name'],
            "weight": work_string_json['weight'],
            "height": work_string_json['height'],
            "capture_rate": work_string_json2['capture_rate'],
            "biology": description,
            "types": [],
            "portrait": work_string_json['sprites']['other']['official-artwork']['front_default'],
            "sprite": work_string_json['sprites']['front_default']
        }

        print(i)
        for i in work_string_json['types']:
            curr_json["types"].append(i['type']['name'])

        new_json.append(curr_json)

    with open('pokemon2.json', 'a', encoding='utf-8') as f:
        json.dump(new_json, f, ensure_ascii=False, indent=4)
```

Listing 1.1: Python example

```
    package it.unipi.dii.lsmsd.pokeMongo.utils;

import java.time.LocalDate;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javafx.scene.control.*;

public class FormValidatorPokeMongo {

```

```
10      /**
11       * In this section are present the event handler for the '
        setOnKeyReleased' event in the form.
12       */
13      public static void handleName(TextField nameTF, Label
        invalidNameLabel){
14          if(FormValidatorPokeMongo.isPersonNoun(nameTF.getText()))
15              invalidNameLabel.setVisible(false);
16          else
17              invalidNameLabel.setVisible(true);
18      }
19
20      /**
21       * Check if the string contains only letters, spaces, dots and
        apostrophes.
22       */
23      public static boolean isPersonNoun(String possibleNoun){
24          Pattern pattern = Pattern.compile("^[a-zA-Z '.]*$");
25          Matcher matcher = pattern.matcher(possibleNoun);
26          return matcher.find();
27      }
28
29      public static void handleEmail(TextField emailTF, Label
        invalidEmailLabel){
30          if(FormValidatorPokeMongo.isValidEmail(emailTF.getText()))
31              invalidEmailLabel.setVisible(false);
32          else
33              invalidEmailLabel.setVisible(true);
34      }
35
36      /**
37       * Check if the email follows the format example@domain.tld
38       */
39      public static boolean isValidEmail(String possibleEmail){
40          Pattern pattern = Pattern.compile("^[\\w-\\.]+@([\\w-]+\\.)
        +[\\w-]{2,4}$");
41          Matcher matcher = pattern.matcher(possibleEmail);
42          return matcher.find();
43      }
44
45      public static void handlePassword(TextField passwordTF, Label
        invalidPasswordLabel){
46          if(FormValidatorPokeMongo.isValidPassword(passwordTF.
        getText()))
47              invalidPasswordLabel.setVisible(false);
48          else
49              invalidPasswordLabel.setVisible(true);
50      }
51
52      /**
53       * Checks if the password contains minimum eight characters, at
         least one letter and one number.
54       */
55      public static boolean isValidPassword(String possiblePassword){
56          Pattern pattern = Pattern.compile("^(?=.*[A-Za-z])(?=.*\\d)
        [A-Za-z\\d]{8,}$");
57          Matcher matcher = pattern.matcher(possiblePassword);
```

```
58          return matcher.find();
59      }
60
61      public static void handleConfirmField(TextField fieldTF,
        TextField confirmFieldTF, Label invalidConfirmFieldLabel){
62          String password = fieldTF.getText(), confirmPassword =
        confirmFieldTF.getText();
63
64          if(password.equals(confirmPassword))
65              invalidConfirmFieldLabel.setVisible(false);
66          else
67              invalidConfirmFieldLabel.setVisible(true);
68      }
69
70      /**
71       * Checks if the birthday date selected is valid: future dates
        cannot be picked
72       */
73      public static void handleBirthday(DatePicker birthdayDP, Label
        invalidBirthdayLabel){
74          LocalDate localDate = birthdayDP.getValue();
75          LocalDate today = LocalDate.now();
76          System.out.println(today);
77
78          if(localDate.isAfter(today)){
79              invalidBirthdayLabel.setVisible(true);
80          } else {
81              invalidBirthdayLabel.setVisible(false);
82          }
83      }
84  }
```

Listing 1.2: Java example