Fun With Queues
Guía de Despliegue

Sergio Baeza Carrasco,53979913V
Oliver Vincent Rice, Y1421890K

Universitat d'Alacant
Universidad de Alicante

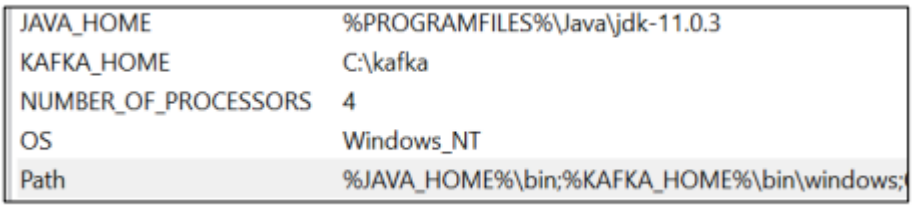
Sergio Baeza Carrasco, 53979913V
Oliver Vincent Rice, Y1421890K

ÍNDICE

1. Requerimientos

Todos los programas **se han desarrollado** en Python 3.9.0, por lo que el primer paso será instalar esta versión.

Una vez instalado **python**, el siguiente paso es instalar el Kafka. Podemos utilizar el que viene en la carpeta instalación simplemente copiando y pegando esa carpeta en la **raíz**. Después de copiarlo habría que cerciorarse que existen las **variables de entorno creadas**.



Si optamos por descargarnos **Kafka** desde su propia página web, descargamos la versión **2.8.1** los archivos binarios lo pegamos en la raíz y una vez hecho esto añadimos las **variables de entorno** mostradas arriba**.**
Una vez hecho esto configuramos tanto el **zookeeper** como el **broker:**
\config\zookeeper.properties
\config\server.properties

Y por último añadiremos los topics ejecutando el archivo de la carpeta de instalación create_topics.bat

Por defecto el puerto del Kafka usando **nuestra carpeta** será 9092.

Para arrancar **Kafka** tendremos que ir a la carpeta **run** e iniciar primero **zookeeper_start**, y posteriormente **broker_start**.

Una vez arrancado el Kafka podemos empezar con el despliegue de aplicaciones.

2. FWQ_Sensor

Esta aplicación se encarga de devolver el **número de visitantes** que se encuentran a la cola de una atracción. Para ello recibirá como parámetro lo siguiente:

1. ip:puerto del **servidor Kafka**
2. **Número (id)** de la atracción

Así también dispondremos de un **ejecutable** que nos solicitará los parámetros y realizará la ejecución del programa de la siguiente manera:
python ./FWQ_Sensor.py ip:puerto id

La aplicación empezará a enviar la información y si se le escribe algo por consola cambiará el número de usuarios al especificado.

```
Introduce la IP:Puerto del Kafka: 127.0.0.1:9092
Introduce el id de la atraccion: 1
[INFO] Launching sensor on 1 attraction, connection to 127.0.0.1:9092
[USER MANAGER] Setting random users on the attraction
[USER MANAGER] By default there is 184 users waiting right now
[INFO] Sending the info: 1:184 to the topic: sensorinfo
[INFO] Sending the info: 1:184 to the topic: sensorinfo
[INFO] Sending the info: 1:184 to the topic: sensorinfo
```

3. FWQ_WaitingTimeServer

Esta aplicación se encarga de recoger todas las informaciones de los sensores y calcular los tiempos a partir de la base de datos que se encuentra en el archivo **config/attractions.**

El formato del archivo **attractions.json**.

```
"attractions": [
    {
        "id": 1,
        "cycleTime": 20,
        "cycleUsers": 5
    },
```

cycleTime -> Incluirá el tiempo que tarda un turno de la atracción
cycleUsers -> Incluirá cuantos usuarios participan en un turno de la atracción

Además a la aplicación habrá que pasarle como parámetros:

1. puerto donde quieres que se abra el servidor
2. ip:puerto del **servidor Kafka**

Así también dispondremos de un **ejecutable** que nos solicitará los parámetros y realizará la ejecución del programa de la siguiente manera:
python FWQ_WaitingTimeServer.py puerto ip:puerto(Kafka)

La aplicación empezará a recibir información de los sensores que estén activos y devolverá esta información cuando un usuario se conecte al servidor por sockets y se lo solicite con **info**.

```
Introduce la IP:Puerto del Kafka: localhost:9092
Introduce el puerto donde se abrira el servidor: 5011
[INFO] Launching server on 192.168.56.1 connecting with Kafka on: localhost:9092
[INFO] Reading information of attractions.json...
[INFO] Getting info from attractions.json...
[INFO] Loaded 4 attraction/s
[INFO] Estimated times: [-1, -1, -1, -1]
[SERVER] Starting Socket Server on 192.168.56.1
[SERVER] Active connections: 0
[TIMES] The attraction 1 has a estimated time of 720 mins
[TIMES] The attraction 1 has a estimated time of 720 mins
```

La propia aplicación mostrará la **ip** donde nos debemos de conectar, en la pantalla se aprecia 192.168.56.1 y el puerto **5011**

4. FWQ_Engine

Esta aplicación gestiona el parque gracias a sus siguientes **funcionalidades**:

- **Recibir información** de los usuarios
- **Enviar** el **mapa**
- Permitir la **entrada** al parque
- Recibir **información** del servidor de tiempos de espera
- Conexión a la **base de datos**

Para desplegarla tendremos que ejecutar los siguientes parámetros:

1. ip:puerto del **servidor Kafka**
2. Número **máximo** de visitantes
3. ip:puerto del **servidor de Tiempos de Espera**

Así también dispondremos de un **ejecutable** que nos solicitará los parámetros y realizará la ejecución del programa de la siguiente manera:

python ./FWQ_Engine.py ip:puerto(KAFKA) maxVisitantes ip:puerto(WTS)

La aplicación se pondrá en funcionamiento **independientemente** de si los otros módulos están desplegados o no.

```
Introduce la IP:Puerto del Kafka: 127.0.0.1:9092
Introduce el numero de visitiantes maximos: 10
Introduce la ip:puerto del servidor de tiempos de espera:  127.0.0.1:9092
[DATABASE] Connecting to data base
[DATABASE] Getting information of attractions
[DATABASE] There is/are 4 attractions
[MODULE] Starting Waiting Time Server module...
[WAITING TIME SERVER] Starting task
[MODULE] Starting Login System
[MODULE] Starting Kafka Consumer for User Info
[WAITING TIME SERVER] Connecting on ('127.0.0.1', 9092)
[MODULE] Starting Map Sender
[MODULE] Starting Keep Alive User
[WAITING TIME SERVER] Requesting info...
[WAITING TIME SERVER] Got a response:
[LOGIN] Awaiting for info on Kafka Server topic = logindetails
[MAP SENDER] Sending map to the topic: mapinfo
```

La base de datos irá en una carpeta llamada **bd**, con nombre **basededatos:**
./bd/basededatos.db
./FWQ_Engine

5. FWQ_Registry

Esta aplicación permite a los visitantes **registrarse o editar su perfil** mediante sockets.

1. **Puerto** donde se abre el servidor

Así también dispondremos de un **ejecutable** que nos solicitará los parámetros y realizará la ejecución del programa de la siguiente manera:

python ./FWQ_Registry.py puerto

La aplicación se pondrá en funcionamiento independientemente de si los otros módulos están desplegados o no.

```
Introduce el puerto donde quieres que se te abra el servidor Registry: 5001
[STARTING] Servidor inicializándose...
[LISTENING] Servidor a la escucha en 192.168.56.1
```

La propia aplicación mostrará la **ip** donde nos debemos de conectar, en la pantalla se aprecia **192.168.56.1** y el puerto que hemos introducido **5001**

La base de datos irá en una carpeta llamada **bd**, con nombre **basededatos:**
./bd/basededatos.db
./FWQ_Registry

6. FWQ_Visitor

Esta aplicación **representa a un visitante** con las siguientes funciones:

- Registrarse
- Editar perfil
- Entrar al parque

Para desplegarla tendremos que ejecutar los siguientes parámetros:

1. ip:puerto del **servidor Kafka**
2. ip:puerto del **servidor Registry**
3. ip:puerto del **servidor API**

Así también dispondremos de un **ejecutable** que nos solicitará los parámetros y realizará la ejecución del programa de la siguiente manera:

python ./FWQ_Visitor.py ip:puerto(Registry) ip:puerto(KAFKA) ip:puerto(API)

La aplicación se pondrá en funcionamiento independientemente de si los otros módulos están desplegados o no.

```
Introduce la IP:Puerto del Kafka: 127.0.0.1:9092
Introduce la IP:Puerto del Registry: 127.0.0.1:9092
##############################################################################################################
## BIENVENIDO AL RESORT                                                                                     ##
##                                                                                                          ##
## Menu:                                                                                                    ##
## 1. Entrar al parque                                                                                      ##
## 2. Editar perfil                                                                                         ##
## 3. Registrarse                                                                                           ##
## 4. Salir                                                                                                 ##
##############################################################################################################
Introduce la opcion que quieras hacer:
```

7. Front

Esta aplicación **ASP.NET** representa el mapa en un sitio web. Para ello utiliza las llamadas a la API. Como tenemos la solución, será necesario configurar el endpoint de la API_Engine y desplegarla.

Para configurar el endpoint lo haremos en el archivo Default.aspx.cs, la función loadMap().

```
string url = @"https://localhost:8080/map";
```

8. API_Register

Esta aplicación permite Registrar o Editar a un usuario con la conexión a la base de datos.

python ./API_Registry.py ip:puerto

La aplicación se pondrá en funcionamiento independientemente de si los otros módulos están desplegados o no.

La base de datos irá en una carpeta llamada **bd**, con nombre **basededatos: ./bd/basededatos.db**

Los endpoints de la api serán

- **POST /users** permite crear un usuario - 201
  {
  “username”:
  “password”:
  }
- **PUT /users** permite editar el usuario - 201
  {
  “username”:
  “password”:
  “usernew”:
  “option”:
  }

9. API_Engine

Permite visualizar información del mapa

python ./API_Engine.py ip:puerto

La aplicación se pondrá en funcionamiento independientemente de si los otros módulos están desplegados o no.

La base de datos irá en una carpeta llamada **bd**, con nombre **basededatos:**
./bd/basededatos.db

Los endpoints de la api serán todos **GET** y serán los siguientes:

- /map/attractions - 200
- /map/users
- /map/cities - 200
- /map/info - 200
- /map - 200