



VERSIÓN 0.0

07/11/2021

# PRÁCTICA 2: GESTOR DE COLAS EN PARQUES TEMÁTICOS

“FUN WITH QUEUES”

ALUMNOS: JUAN GARCÍA MARTÍNEZ – 20085694R

RAÚL SIGÜENZA SANCHEZ -

SISTEMAS DISTRIBUIDOS  
UNIVERSIDAD ALICANTE – 2021/2022

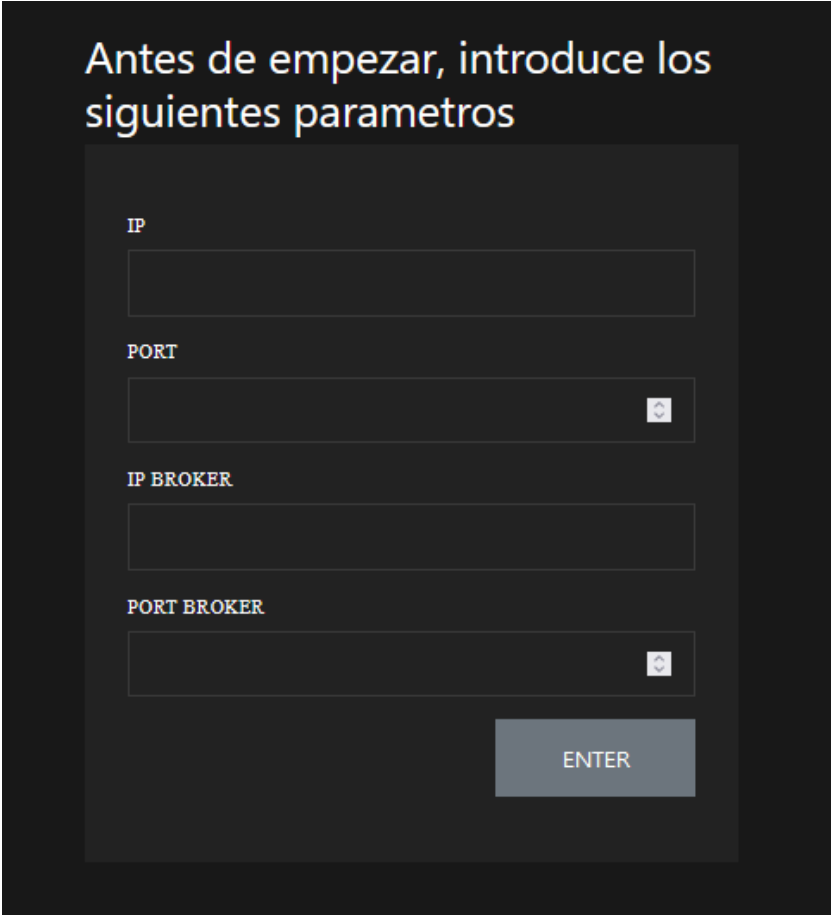


## COMPONENTES DEL SOFTWARE DESARROLLADOS

### FWQ\_VISITOR

Este componente será la interfaz del programa. El usuario se conectará a esta para realizar cualquier operación que permita el sistema central.

El visitante inicialmente tendrá que introducir los parámetros necesarios para conectarse al sistema central.



Antes de empezar, introduce los siguientes parametros

IP

PORT

IP BROKER

PORT BROKER

ENTER

Una vez introducido los parámetros, se mostrará la autenticación del usuario.

The image shows a dark-themed user interface for authentication. It is divided into two main sections: 'Login if you have an account' on the left and 'Create a new account' on the right. The login section has two input fields labeled 'Username' and 'Password', followed by a 'Login' button. The registration section has five input fields labeled 'Username', 'Name', 'email', 'Password', and 'Password verification', followed by a 'Singup' button. All text and labels are in a light color, likely white or light gray, against the dark background.

Si el usuario tiene una cuenta, solo debe de iniciar sesión para acceder por completo a todas las funcionalidades. En cambio, si el usuario no tiene cuenta puede crearse una cuenta, siempre y cuando respete los requisitos para registrarse.

- Intento de inicio de sesión o registro sin que la aplicación FWQ\_Registry no esté en escucha.

FWQ\_Registry no esta activo en este momento. Espera a que se inicie.

### Login if you have an account

Username

Password

Login

### Create a new account

Username

Name

email

Password

Password verification

Singup

- Una vez introducido los datos correctamente en el registro se mostrará el siguiente menú:

Registro realizado

jose, What do you want to do?

Update your profile

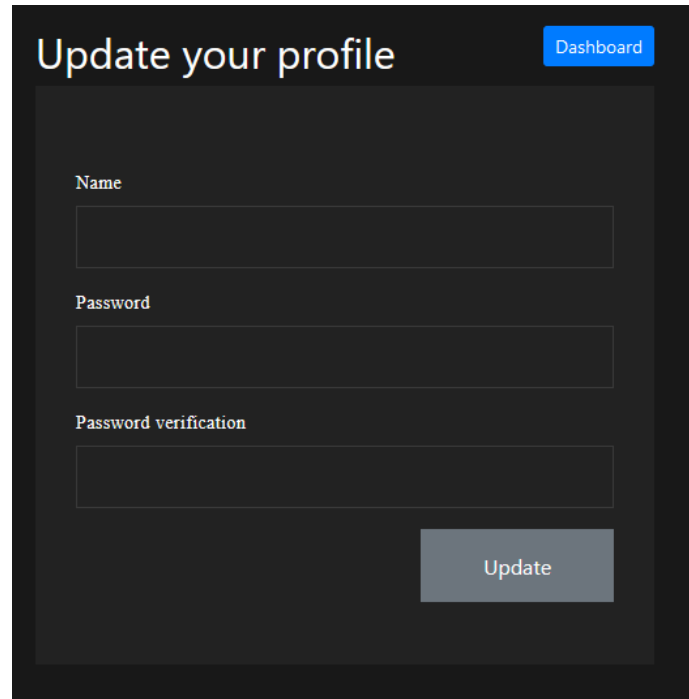
Enter the park

Logout

Una vez llegado aquí ya sea mediante registro o inicio de sesión solo podremos acceder a las opciones de “*Update your profile*” y “*Logout*” ya que hasta aquí solo necesitaríamos FWQ\_Registry.

## Update you profile.

- Esta opción del menú permite la modificación del usuario, solo si el usuario está en sesión

A screenshot of a web form titled "Update your profile" in a dark-themed interface. The form contains three input fields: "Name", "Password", and "Password verification". Each field is represented by a dark rectangular box with a light border. Below the "Password verification" field is a light gray button labeled "Update". In the top right corner of the form area, there is a blue button labeled "Dashboard".

- Podremos modificar solo el nombre y la contraseña, pero en el futuro se podrá actualizar todo el perfil incluyendo el alias y más elementos como foto de perfil, email, etc.
- En caso de que hayamos accedido a esta vista de forma errónea, hemos añadido la posibilidad de volver al menú con el botón "*Dashboard*"

Ejemplo de una modificación correcta del perfil de jose:

The screenshot shows a dark-themed web interface for updating a profile. At the top right is a blue button labeled "Dashboard". The main heading is "Update your profile". Below it, there are three input fields: "Name" (containing "Jose Manuel"), "Password" (with four dots), and "Password verification" (with four dots and a cursor). A grey "Update" button is at the bottom right.

This screenshot shows the same "Update your profile" form after a successful update. A green banner at the top displays the message: "Has modificado tu perfil correctamente genio, idolo, crack". The form fields are now empty, and the "Update" button remains visible at the bottom right.

Ejemplo de una modificación errónea del perfil de jose:

The screenshot shows a dark-themed web interface for updating a profile. At the top right is a blue button labeled "Dashboard". The main heading is "Update your profile". Below it are three input fields: "Name" containing "Jose Manuel", "Password" with four dots, and "Password verification" with four dots. A blue border highlights the "Password verification" field. At the bottom right is a grey "Update" button.

This screenshot shows the same "Update your profile" form, but with an error. A yellow banner at the top contains the message "Hay que escribir las contraseñas igual, inútil". The input fields for "Name", "Password", and "Password verification" are now empty. The "Update" button remains at the bottom right.

## Logout.

- Esta opción del menú permitirá cerrar sesión, redirigiéndote a home donde podrás volver a iniciar sesión o registrarte.

Has cerrado la sesión

### Login if you have an account

Username

Password

Login

### Create a new account

Username

Name

email

Password

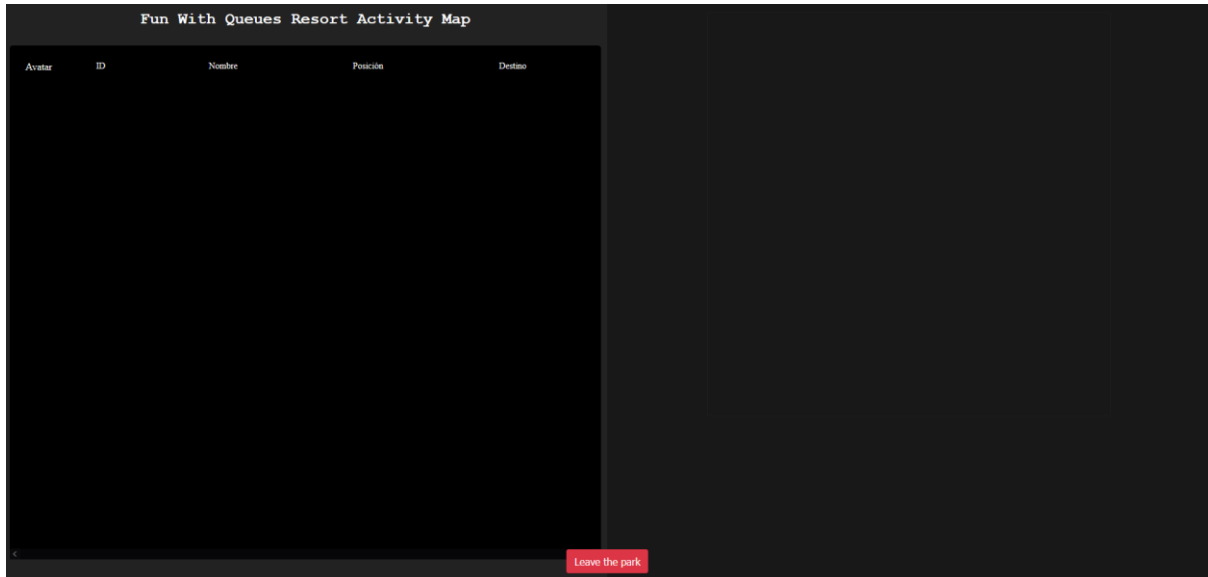
Password verification

Singup

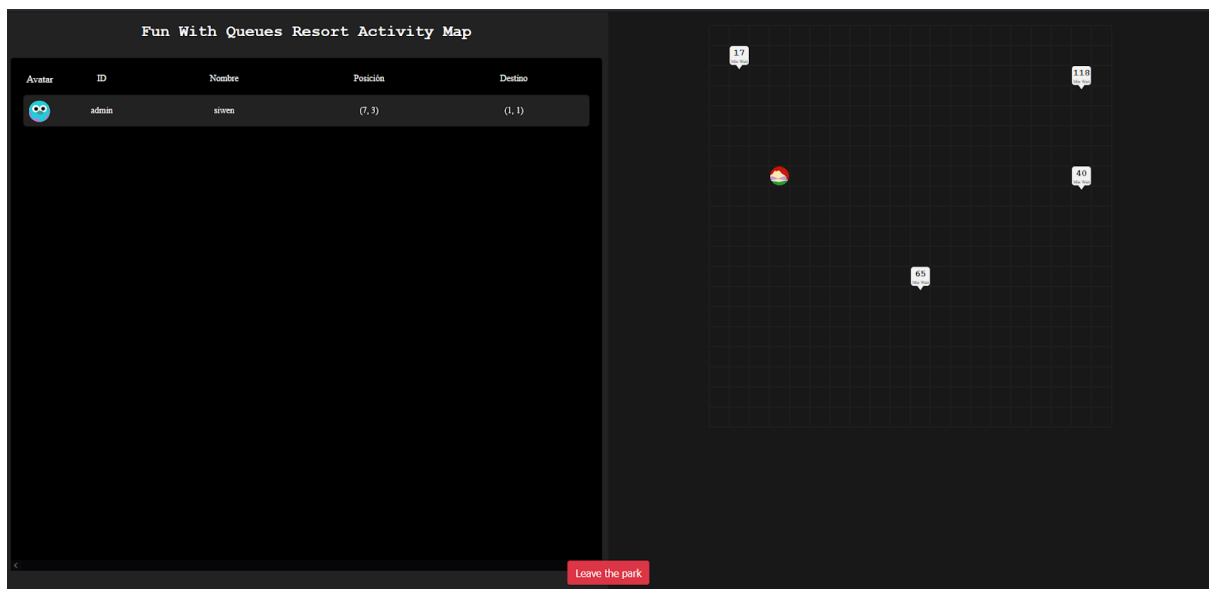


## Enter the park.

- Para poder acceder a esta opción, es necesario, que la aplicación FWQ\_Engine este activa, en caso contrario al usuario se le mostrara un mapa vacío.



- Si el FWQ\_Engine está activo se mostrará lo siguiente:



Aquí tenemos un pequeño fallo que nos percatamos al final y es que el FWQ\_Engine actualiza el mapa del usuario cuando recibe las atracciones del FWQ\_WaitingTimeServer y debería de ser cuando el usuario introdujese su nueva posición en el mapa.

No da error, pero si está solo el FQW\_Engine con el FQW\_Visitor se quedará cómo si el Engine no estuviese conectado, hasta que se conecte el FWQ\_WaitingTimeServer.

[Leave the park.](#)

Esta opción solo se da si has entrado al parque y si le das, te redirige al menú, donde podrás actualizar tu perfil, entrar al parque o cerrar sesión.

## FWQ\_REGISTRY

- Esta aplicación se comunica mediante socket e hilos con el visitante, recibiendo así la información necesaria para crear y modificar los usuarios.

```
def handle_client(conn, addr):
    print(f"[NUEVA CONEXION] {addr} connected.")

    connected = True
    while connected:
        try:
            msg_length = conn.recv(HEADER).decode(FORMAT)
            if msg_length:
                msg_length = int(msg_length)
                msg = conn.recv(msg_length).decode(FORMAT)
                if msg == FIN:
                    connected = False
                elif "crearPerfil" in msg:
                    result = crearPerfil(msg)
                    conn.send(result.encode(FORMAT))

                elif "modificarPerfil" in msg:
                    result = modificarPerfil(msg)
                    conn.send(result.encode(FORMAT))
                elif "iniciarSesion" in msg:
                    result = iniciarSesion(msg)
                    conn.send(result.encode(FORMAT))
                    USUARIOS.append({'conexion': addr,
                                    'alias': result})
                    print(USUARIOS)
                print(f" He recibido del cliente [{addr}] el mensaje: {msg}")
        except ConnectionResetError:
            connected = False
            for user in USUARIOS:
                if user['conexion'] == addr:
                    connection = sqlite3.connect('user.db')
                    c = connection.cursor()
                    c.execute(f"update user set inPark = '0' where alias = '{user['alias']}'")
                    connection.commit()
                    connection.close()
                    USUARIOS.remove({'conexion': addr,
                                      'alias': result})
            print(bcolors.WARNING + f"Se ha forzado la interrupcion de la aplicacion en {addr}. Vuelva a conectarse." + bcolors.RESET)
```

Para ello, esta aplicación requiere una base de datos donde guardar y actualizar esos usuarios.

Esta también se encargará de enviar un mensaje de confirmación al cliente, sobre la acción que ha realizado, ya sea correcta o errónea.

## FWQ\_ENGINE

- Esta aplicación implementa toda la lógica del sistema, la cual recibirá eventos del usuario mediante el gestor de colas (Kafka) y se conectará a FWQ\_WaitingTimeServer para recibir los posibles cambios que se hayan realizado en las atracciones.
- Estas son las funciones que realiza:

```
> def send(msg): ...  
  
> def vacio(list): ...  
  
> def calcularTiempo(ciclo, capacidad, cola): ...  
  
> def cambiarMapa(serverResponse): ...  
  
> def producirKafka(): ...  
  
> def start(): ...  
  
> def usuarioAlMapa(userSql): ...  
  
> def gestionarEntrada(data): ...  
  
> def gestionarSalida(data): ...  
  
> def handle_visitor(): ...
```

- Send: Envía un mensaje mediante socket.
- vacio: Comprueba que una lista no esté vacía.
- calcularTiempo: calcula el tiempo que falta para que la atracción quede libre.
- producirKafka: conecta con el cliente para pasarle el mapa.
- start: Conecta con WaitingTimeServer para recibir las atracciones.
- usuarioAlMapa: introduce un usuario al mapa
- gestionarEntrada: comprueba que el usuario es correcto y que el número de visitantes no sobrepasa el límite y llama a la función usuarioAl-Mapa
- gestionarSalida: el usuario mediante kafka manda un mensaje y este método recibe que quiere salir del parque y lo sacará
- handle\_visitor: llamará a las funciones necesarias respecto el mensaje que haya enviado el usuario mediante Kafka
-

## FWQ\_SENSOR

- Esta aplicación se conectará al gestor de colas como un productor, enviando los nuevos tiempos de espera de X atracción cada 3 segundos, y éstos los recibirá el FWQ\_WaitingTimeServer para actualizarlos en la base de datos y que los usuarios puedan mantenerse al corriente de todas las atracciones disponibles.

```
try:
    IP_BROKER = sys.argv[2]
    PUERTO_BROKER = int(sys.argv[3])

    ID = int(sys.argv[1])
    FORMAT = 'utf-8'

    fallo = True

    while(fallo):
        try:
            producer = KafkaProducer(bootstrap_servers = IP_BROKER + ':' + str(PUERTO_BROKER),
                                     value_serializer=lambda x: dumps(x).encode(FORMAT))

            while True:
                numVisitantes = random.randint(10,120)
                data = [ID, numVisitantes]
                producer.send('SD', value=data)
                print(bcolors.OK + f'Se envió {data} correctamente a la dirección {IP_BROKER}:{PUERTO_BROKER}' + bcolors.RESET)
                sleep(3)

            except kafka.errors.NoBrokersAvailable:
                print(bcolors.FAIL + f'Actualmente no hay un broker disponible en la dirección {IP_BROKER}:{PUERTO_BROKER}. Espere a que s
                sleep(30)

    except IndexError:
        print(bcolors.FAIL + 'FWQ_Sensor requiere <ID> <IP_BROKER> <PUERTO_BROKER>' + bcolors.RESET)
    except ValueError:
        print(bcolors.FAIL + 'No se puede convertir una palabra a un int. Por favor, introduce los datos correctamente.' + bcolors.RESET)
```

## FWQ\_WAITINGTIMESERVER

- Esta aplicación es un servidor que está conectado mediante sockets al engine, enviando los tiempos de espera de las atracciones, cuando le llegue el mensaje “Tiempos”, siendo estos los últimos que ha recibido de los sensores activados mediante kafka.
- En la siguiente imagen le mostramos cómo manejamos el engine:

```
def handle_engine(conn, addr):
    print(f"[NUEVA CONEXION] {addr} connected.")

    connected = True

    while connected:
        try:
            #Lado del cliente
            msg_length = conn.recv(HEADER).decode(FORMAT)
            if msg_length:
                msg_length = int(msg_length)
                msg = conn.recv(msg_length).decode(FORMAT)
                if msg == FIN:
                    connected = False
                    print(bcolors.OK + f" He recibido del cliente [{addr}] el mensaje: {msg}" + bcolors.RESET)
                elif msg == "Tiempos":
                    connection = sqlite3.connect('atraccion.db')
                    c = connection.cursor()
                    c.execute('select * from atraccion')
                    msg = c.fetchall()
                    conn.send(str(msg).encode(FORMAT))
                    print(bcolors.OK + f" He recibido del cliente [{addr}] el mensaje: {msg}" + bcolors.RESET)
                else:
                    print(bcolors.OK + f" He recibido del cliente [{addr}] el mensaje: {msg}" + bcolors.RESET)
        except sqlite3.Error:
            print(bcolors.FAIL + "No se ha podido conctar a la base de datos. Intentalo de nuevo mas tarde." + bcolors.RESET)
            break
        except ConnectionResetError:
            print(bcolors.FAIL + "Se ha perdido la conexion con FWQ_Engine. Esperando a que se conecte nuevamente." + bcolors.RESET)
            break
        except ValueError:
            print(bcolors.FAIL + 'No se puede convertir una palabra a un int. Por favor, introduce los datos correctamente.' + bcolors.RESET)
```

- En la siguiente imagen mostramos cómo manejamos los mensajes recibidos por el sensor mediante Kafka:

```
def actualizarTiempos(atraccion):
    atraccion = atraccion[1:len(atraccion)-1]
    print(bcolors.KAFKA + atraccion + bcolors.RESET)
    atraccion = re.split(",", atraccion)

    try:
        connection = sqlite3.connect('atraccion.db')
        c = connection.cursor()
        c.execute(f'update atraccion set numVisitantes = {atraccion[1]} where id = {atraccion[0]}')
        connection.commit()
        c.execute(f'select * from atraccion where id = {atraccion[0]}')
        result = c.fetchall()
        c.close()
        if result == []:
            print(bcolors.FAIL + "No se ha podido actualizar la atraccion correctamente. Revisa si los datos d")
    except sqlite3.Error:
        print(bcolors.FAIL + "No se ha podido conctar a la base de datos. Intentalo de nuevo mas tarde." + bcc

def handle_kafka():

    fallo = True

    while(fallo):
        try:
            consumer = KafkaConsumer('SD', bootstrap_servers=IP_BROKER+':'+ str(PUERTO_BROKER))
            fallo = False
            print(bcolors.KAFKA + f'[LISTENING] Escuchando mensajes de Kafka en el servidor {IP_BROKER}:{PUERTO_BROKER}')
            for msg in consumer:
                msg = msg.value.decode(FORMAT)
                actualizarTiempos(msg)

        except kafka.errors.NoBrokersAvailable:
            print(bcolors.FAIL + 'Actualmente no hay un broker disponible en la dirección ' + IP_BROKER + ':' + PUERTO_BROKER)
            sleep(30)
```

## DESPLIEGUE

### FWQ\_Registry.

```
PS C:\Users\johns\Desktop\SDPractica\FWQPython> python .\FWQ_Registry.py 5050
[STARTING] Servidor inicializándose...
[LISTENING] Servidor a la escucha en ('192.168.0.173', 5050)
0
```

### FWQ\_Visitor.

```
(django) C:\Users\johns\Desktop\SDPractica\FWQPython\FWQ_Visitor>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 07, 2021 - 21:14:15
Django version 3.2.9, using settings 'FWQ_Visitor.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

### FWQ\_Engine

```
PS C:\Users\johns\Desktop\SDPractica\FWQPython> python .\FWQ_Engine.py 192.168.0.173 9092 3 192.168.0.173 1500
[LISTENING] Escuchando mensajes de Kafka en el servidor 192.168.0.173:9092
No se ha podido establecer la conexion con [('192.168.0.173', 1500)], intentando reconectar...
```

### FWQ\_WaitingTimeServer.

```
PS C:\Users\johns\Desktop\SDPractica\FWQPython> python .\FWQ_WaitingTimeServer.py 1500 192.168.0.173 9092
[STARTING] FWQ_WaitingTimeServer inicializándose...
[LISTENING] FWQ_WaitingTimeServer a la escucha en ('192.168.0.173', 1500)
[LISTENING] Escuchando mensajes de Kafka en el servidor 192.168.0.173:9092
[NUEVA CONEXION] ('192.168.0.173', 55437) connected.
He recibido del cliente [('192.168.0.173', 55437)] el mensaje: [(0, 21, 10, 20, 14), (1, 58, 20, 20, 98), (2, 158, 10, 20, 60), (3, 250, 15, 18, 60)]
```

### FWQ\_Sensor.

```
PS C:\Users\johns\Desktop\SDPractica\FWQPython> python .\FWQ_Sensor.py 0 192.168.0.173 9092
Se envió [0, 105] correctamente a la dirección 192.168.0.173:9092
Se envió [0, 112] correctamente a la dirección 192.168.0.173:9092
Se envió [0, 118] correctamente a la dirección 192.168.0.173:9092
```

## Kafka Zookeeper.

```
Windows PowerShell
server.ZooKeeperServer)
[2021-11-07 09:54:01,293] INFO minSessionTimeout set to 6000 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-07 09:54:01,293] INFO maxSessionTimeout set to 60000 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-07 09:54:01,294] INFO Created server with tickTime 3000 minSessionTimeout 6000 maxSessionTimeout 60000 datadir C:\kafka\data\zookeeper\version-2 snapdir C:\kafka\data\zookeeper\version-2 (org.apache.zookeeper.server.ZooKeeperServer)
[2021-11-07 09:54:01,316] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactory)
[2021-11-07 09:54:01,320] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector thread(s), 24 worker threads, and 64 kB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2021-11-07 09:54:01,323] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2021-11-07 09:54:01,336] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2021-11-07 09:54:01,376] INFO Reading snapshot C:\kafka\data\zookeeper\version-2\snapshot.152 (org.apache.zookeeper.server.persistence.FileSnap)
[2021-11-07 09:54:01,405] INFO Snapshotting: 0x17c to C:\kafka\data\zookeeper\version-2\snapshot.17c (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2021-11-07 09:54:01,416] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PreRequestProcessor)
[2021-11-07 09:54:01,426] INFO Using checkIntervalMs=60000 maxPerMinute=10000 (org.apache.zookeeper.server.ContainerManager)
[2021-11-07 09:54:18,816] INFO Creating new log file: log.17d (org.apache.zookeeper.server.persistence.FileTxnLog)
[2021-11-07 09:54:19,589] INFO Expiring session 0x1000f5f76ad0002, timeout of 18000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
```

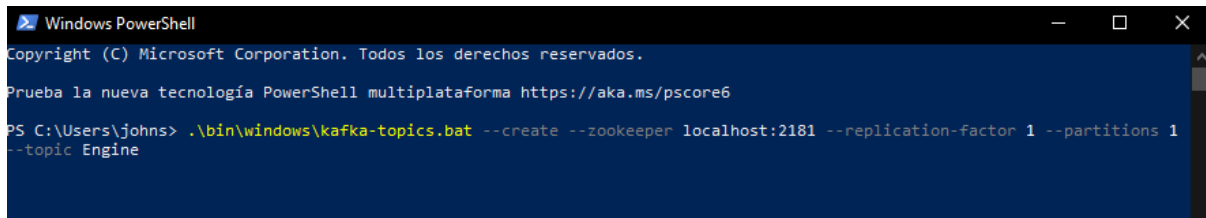
## Kafka Server.

```
Windows PowerShell
rollerRequestThread)
[2021-11-07 12:22:23,170] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(Visitor-0, SD-0, Engine-0, foobar-0) (kafka.server.ReplicaFetcherManager)
[2021-11-07 12:22:23,180] INFO [Partition Visitor-0 broker=0] Log loaded for partition Visitor-0 with initial high watermark 516 (kafka.cluster.Partition)
[2021-11-07 12:22:23,184] INFO [Partition SD-0 broker=0] Log loaded for partition SD-0 with initial high watermark 294 (kafka.cluster.Partition)
[2021-11-07 12:22:23,185] INFO [Partition Engine-0 broker=0] Log loaded for partition Engine-0 with initial high watermark 4462 (kafka.cluster.Partition)
[2021-11-07 12:22:23,185] INFO [Partition foobar-0 broker=0] Log loaded for partition foobar-0 with initial high watermark 0 (kafka.cluster.Partition)
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-12].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-13].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-14].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-15].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-16].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-17].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-18].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-19].
log4j:ERROR Failed to rename [C:\kafka/logs/controller.log] to [C:\kafka/logs/controller.log.2021-11-07-20].
```



## Creación de topics en Kafka

Engine:



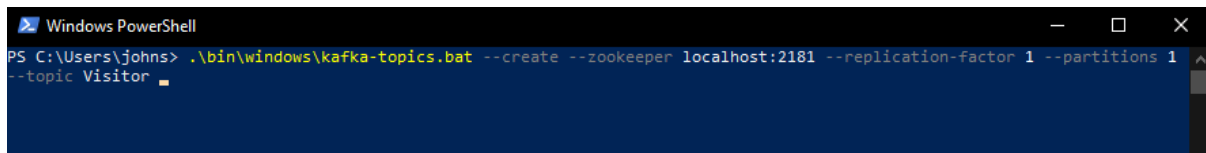
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\johns> .\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1
--topic Engine
```

Se emplea para la conexión Engine-Visitor

Visitor:

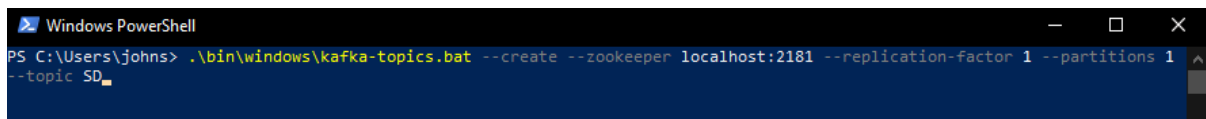


```
Windows PowerShell

PS C:\Users\johns> .\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1
--topic Visitor
```

Se emplea para la conexión Visitor-Engine

SD:



```
Windows PowerShell

PS C:\Users\johns> .\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1
--topic SD
```

Se emplea para la conexión Sensor-WaitingTimeServer