

Sistemas Distribuidos: Fun with queues

Universidad de Alicante — November 7, 2021

- Ander Dorado Bole 48827596E
- Daniel Asensi Roch 48776120C

Introducción

El objetivo de esta práctica es que los alumnos extiendan y afiancen sus conocimientos sobre el uso de la tecnología de comunicación básica sockets, así como streaming de eventos y colas. El objetivo de la práctica a desarrollar es construir un sistema distribuido que monitorice y gestione los tiempos de espera de las colas en las atracciones de un parque temático. La solución será capaz de controlar la localización de los visitantes del parque y los tiempos de espera en las atracciones, informando de los mismos a los visitantes los cuales, ante tiempos excesivamente largos, tomarán la decisión de dirigirse a otra atracción.

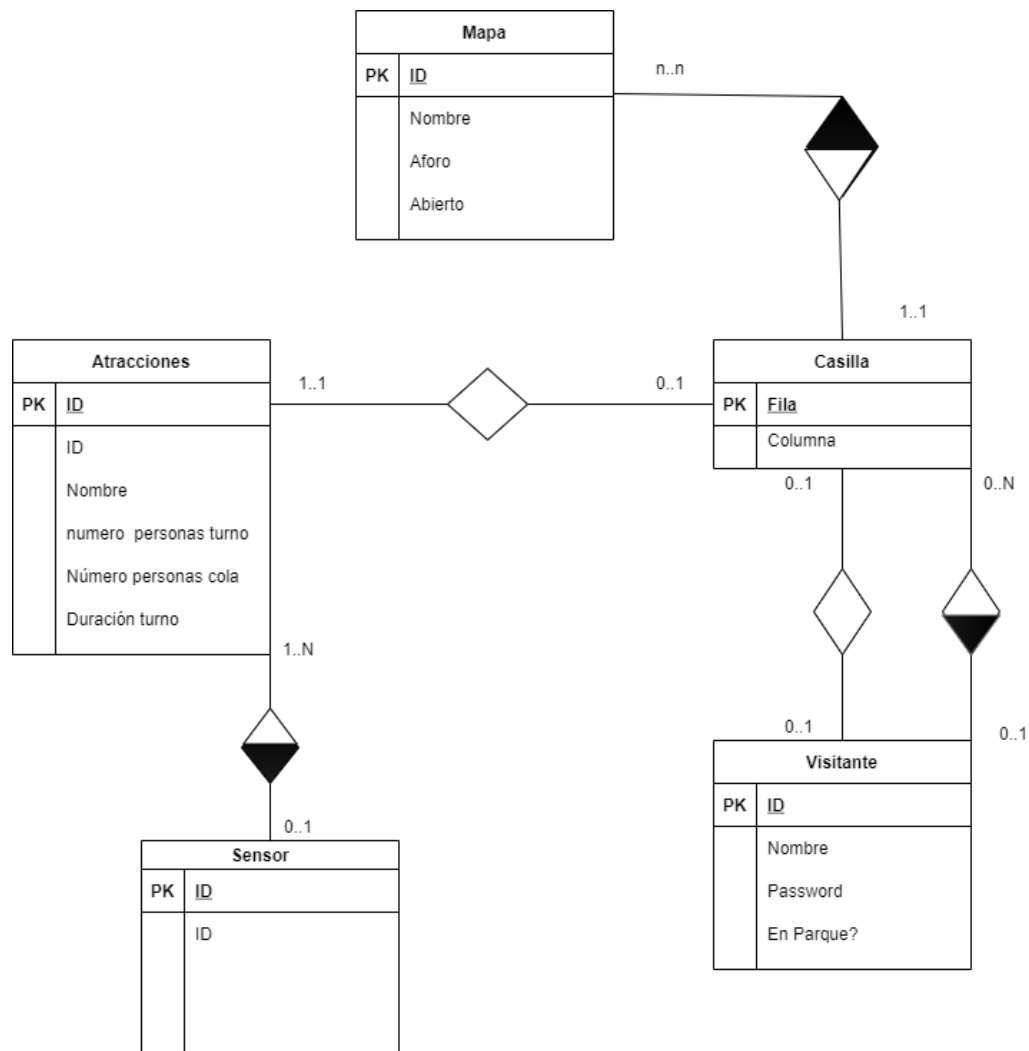
1 Tecnologías Utilizadas

1.1 Bases de Datos

En primer lugar, una de las tecnologías requeridas para la práctica es el uso de una base de datos, para ellos hemos utilizado un servidor remoto gratuito:

<https://remotemysql.com/phpmyadmin/index.php?db=UFD6yb2Mz3>

En nuestra base de datos hemos seguido el siguiente esquema:



Relación

Entidad

Esta base de datos es la que actualizaremos y accederemos a posterior desde los módulos de Registry y Engine para poder obtener y actualizar los datos de las atracciones y los visitantes.

Debemos recalcar que para el correcto funcionamiento de la práctica deberemos tener una base de datos extra que será a la que se conecte nuestro modulo de servidor de tiempos de espera, esta no ha sido representada en el entidad relación ya que se trata de un archivo de texto sencillo.

1.2 Sockets

La primera tecnología utilizada para el desarrollo de esta práctica ha sido el uso de *sockets*. Esta tecnología es utilizada más concretamente por FWQ_WaitingServer para comunicar los tiempos de espera de las atracciones a FWQ_Engine y entre FWQ_Registry y los visitantes para atender a las peticiones de registrarse y/o modificar la información de su perfil. Ambos *sockets* son concurrentes.

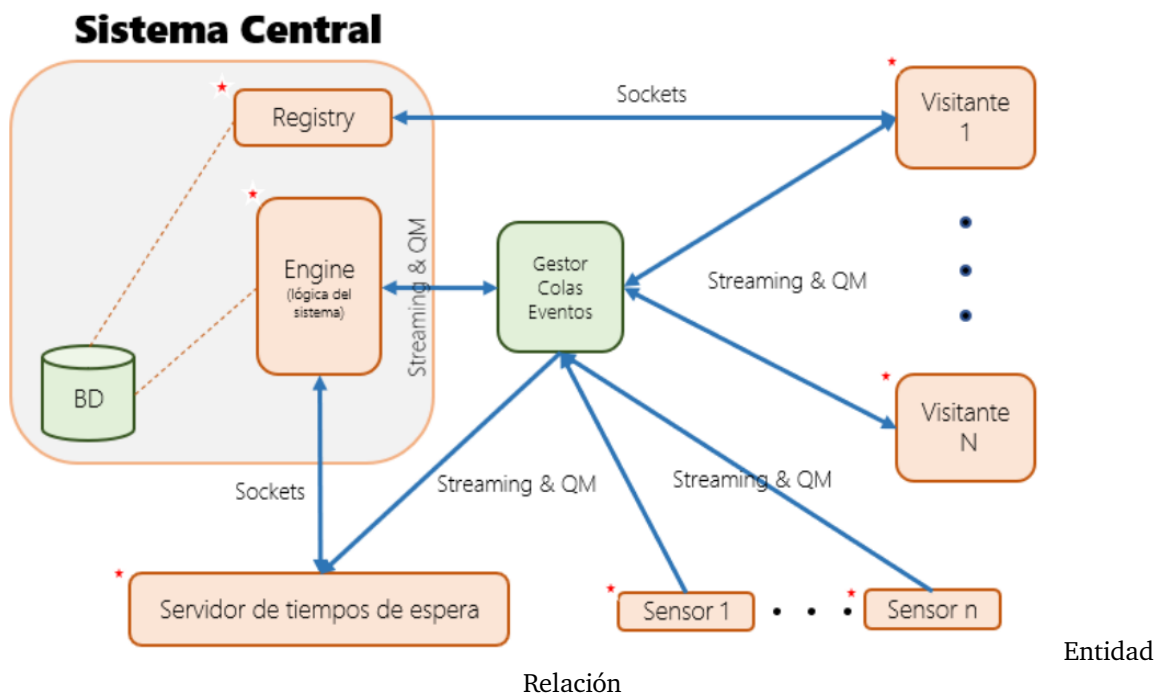
1.3 Gestor de colas: Kafka

Para la gestión de colas hemos utilizado la tecnología de *Kafka Apache* que nos permite de manera sencilla administrar todas las peticiones de cada servicio.

1.4 Interfaz gráfica: JFrame

Todas las interfaces gráficas implementadas han utilizado JFrame.

1.5 Esquema Utilizado



1.6 Registry

Este será el servidor que conecte a la base de datos, se encargará de registrar y actualizar los datos de los usuarios mediante el uso de sockets, lo que nos permitirá, como medida de seguridad, que aun no estando el Engine operativo se puedan registrar usuarios y modificar sus credenciales, así como que si el Registry no se encuentra el operativo, los usuarios podrán seguir accediendo al parque de manera correcta, cabe recalcar que este servidor es concurrente lo que permitirá mediante hilos que varios usuarios se registren a la vez.

1.7 WaitingServer

Este servidor es el servidor que mediante sockets le enviará la información de los tiempos de espera a su cliente, en nuestro caso en el engine. La lectura de estos tiempos de espera se realizará mediante una base de datos ficticia, es decir un fichero de texto.

1.8 FWQ_VISITOR

Este será cliente del servidor registry, se presentará de manera amigable al usuario con una ventana donde aparecerán 3 opciones: Registrar Nuevo Usuario, Iniciar Sesión, Editar Usuario, de estas opciones las funcionarán mediante sockets son Registrar Nuevo usuario y Editar Usuario. La funcionalidad de iniciar sesión se llevará a cabo mediante los gestores de colas que nos proporciona Kafka server.

1.8.1 Registrar Nuevo

Esta funcionalidad abrirá una nueva ventana, donde se pedirá que el usuario inserte su ID por el que será conocido dentro del parque, su contraseña de acceso y su nombre de pila, esos datos serán enviados tras apretar el botón aceptar al servidor y este mismo mediante hilos registrará al usuario en la base de datos.

1.8.2 Editar Usuario

Esta funcionalidad abrirá una nueva ventana, donde se pedirá que el usuario inserte su ID y su contraseña de acceso, además de su nuevo id y su nueva contraseña. Después de presionar el botón aceptar se enviará la información al servidor y el hilo realizará las modificaciones en la base de datos.

1.8.3 Iniciar Sesión

Esta funcionalidad se lleva a cabo mediante el gestor de colas de Kafka, primero el FWQ_VISITOR abrirá una ventana para que el usuario pueda logguese, en ella deberá escribir su ID y su contraseña, estos datos serán enviados al Engine mediante un mensaje desde el productor, el Engine lo consumirá y mirará en la base de datos si existe el usuario y si cabe en el parque, una vez hecho esto generará un token que asignará al usuario y lo devolverá para que este pueda entrar al parque y visualizar el mapa.

1.8.4 Salir del parque

Este será otro de los productores que posee el Visitante, este enviará su id y su token de inicio de sesión así como cerrará su hilo para dejar de leer movimientos y cerrar las pestañas del mapa.

1.9 Engine

Se encarga de la lógica del parque, desde este se accederá a la base de datos y se realizarán las funciones de inicio de sesión, carga de valores en el mapa y realización de actualizaciones en el mismo. Este contará con una parte de cliente, siendo servido por tiempos de espera y otra Streaming.

1.9.1 Engine Cliente

El Engine como cliente recibirá el parámetro de a que puerto conectarse, para comenzar a consumir la información de los tiempos de espera que previamente dicha clase ha leído de su base de datos.

1.9.2 Engine Streaming

El engine como Streaming contará con varios productores y consumidores que funcionarán en hilos diferentes de su ejecución, cabe recalcar que estos consumidores no han sido realizados con el creador de consumidores que ofrece Kafka, ya que la preparación de los mismos es engorrosa y repetitiva, ello hemos creado un clase Consumer que realizará todo esto mediante parámetros.

El primer consumidor que deberá de tener nuestro Engine será el de inicio de sesión este mandará unas credenciales que deberán ser buscadas en la base de datos y devueltas mediante un productor a la clase Visitor adjuntado a su vez un Token que servirá para que el visitante pueda iniciar sesión y enviar sus movimientos.

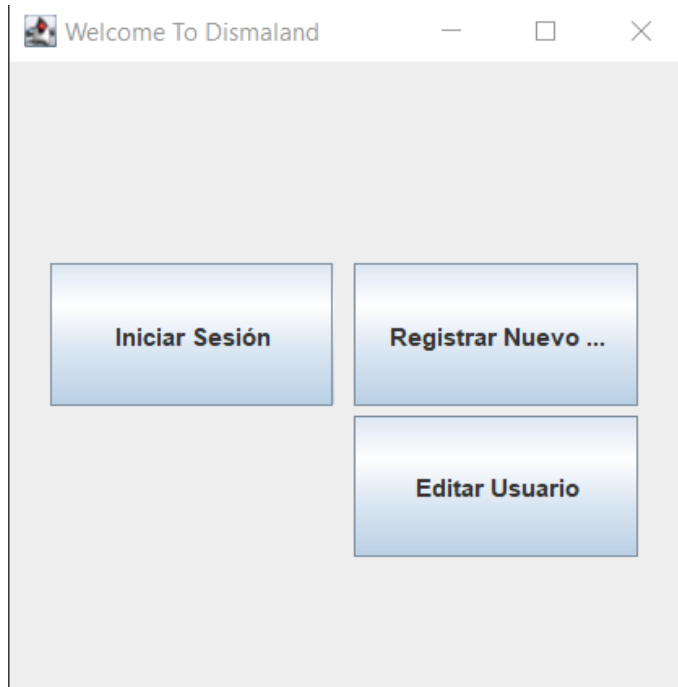
El consumidor de movimientos recibirá los movimientos generados por la lógica del visitante y los implementará al mapa para de nuevo volverlos a enviar a todos los Visitantes.

El consumidor para salir del parque permanecerá a la escucha de los visitantes que deseen abandonar las instalaciones, de ellos recibirá su id y actualizará la ultima casilla donde se quedaron el mapa para que puedan abandonar el mapa por completo.

2 Pruebas de Ejecución

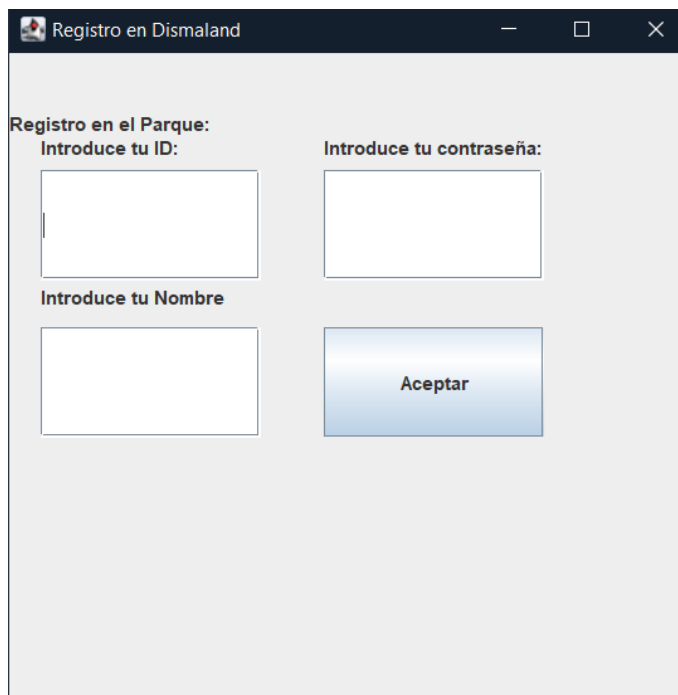
En este apartado mostraremos las pruebas de ejecución de nuestro sistema, para ello adjuntaremos capturas de las ventanas creadas, así como de las consolas de comandos recibiendo datos desde los productores.

2.0.1 Ventana de selección de acción



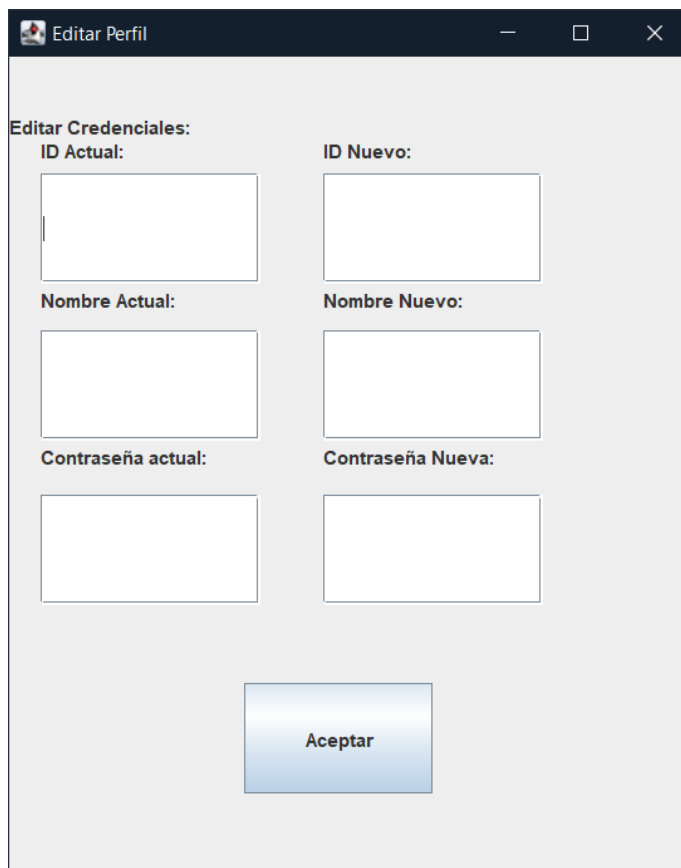
Ventana de Selección

2.0.2 Ventana de Registro de usuario



Registro de Usuario

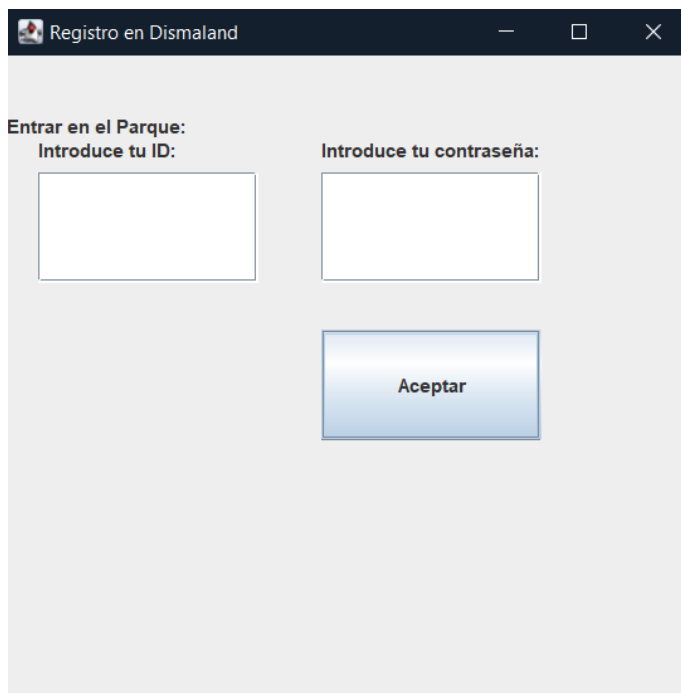
2.0.3 Ventana de Edición de Usuario



The screenshot shows a window titled "Editar Perfil" with a standard Windows title bar. The main content area is titled "Editar Credenciales:". Below this title, there are two columns of input fields. The left column contains three fields labeled "ID Actual:", "Nombre Actual:", and "Contraseña actual:". The right column contains three fields labeled "ID Nuevo:", "Nombre Nuevo:", and "Contraseña Nueva:". Each label is positioned above its corresponding empty text input box. At the bottom center of the window, there is a blue button with the text "Aceptar".

Ventana de Edición de Usuario

2.0.4 Ventana de Inicio de Sesión



The screenshot shows a window titled "Registro en Dismaland" with a standard Windows title bar. The main content area is titled "Entrar en el Parque:". Below this title, there are two input fields. The left field is labeled "Introduce tu ID:" and the right field is labeled "Introduce tu contraseña:". Each label is positioned above its corresponding empty text input box. At the bottom center of the window, there is a blue button with the text "Aceptar".

Ventana de Inicio de sesión

2.0.5 Ventana de Mapa

	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	70	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	anderdb	0	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-
13	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Ventana de Mapa

2.0.6 Ventana de Sensores

Sensor Manager

Id de atracción: 5

Enviar dato

Start

Stop

Ventana de Sensor

3 DEPLOY

1. Configurar archivo server.properties de kafka
2. Arrancar zookeeper-server-start con su configuración
3. Arrancar kafka-server-start con su configuración
4. Crear los TOPICS a utilizar
5. Arrancar FWQ_Engine <IP-TimeServer> <Puerto-TimeServer> <Numero máximo de visitantes> <IP-broker> <Puerto-Broker>
6. Arrancar FWQ_Registry <puerto registry>
7. Arrancar FWQ_WaitingServer <puertoEscucha> <IPBrooker> <PuertoBroker>
8. Arrancar FWQ_Sensor <IP Broker/Bootstrap-servers> <Puerto Broker/Bootstrap-servers> <ID Atraccion>
9. Arrancar FWQ_Visitor <IP Registry> <Puerto Registry> <IP Bootstrap> <Puerto Bootstrap>