**Aplicación java**

**gestión barreras y básculas**

Tabla de contenido

[Resumen 3](#_Toc384310795)

[Código fuente 3](#_Toc384310796)

[Archivo de configuración 3](#_Toc384310797)

[librerías de aplicación 3](#_Toc384310798)

[Configuración acceso a básculas y barreras 3](#_Toc384310799)

[Formas de ejecución 4](#_Toc384310800)

[Flujo de llamadas en ejecución 4](#_Toc384310801)

[Función consultaMatriculas() 4](#_Toc384310802)

[Función trataMatricula 4](#_Toc384310803)

[Clase ControlCarril 5](#_Toc384310804)

[Clase Bascula 5](#_Toc384310805)

[Clase Datos 5](#_Toc384310806)

# Resumen

Este documento explicará brevemente parte de desarrollo de la aplicación usada para el control de accesos y registros de entradas de una infraestructura formada por básculas de peso y barreras.

La aplicación pretende ejecutarse en forma de *demonio*, siempre en ejecución.

# Código fuente

## Archivo de configuración

Existe un archivo llamado ***config.xml*** donde se definen determinados valores necesarios para la ejecución de la aplicación.

Parte de la configuración también se lee de BBDD (función Datos.cargaConfigCamaras() ) para ver la disposición de las cámaras en carriles y los dispositivos de barrera y básculas que tienen asociadas.

Las tablas son: camaras y carriles.

## librerías de aplicación

Se usan las librerías (*modbus*) creadas para el acceso a la báscula y a la barrera. Se llaman residuos.jar y seroUtils.jar

## Configuración acceso a básculas y barreras

Existen dos archivos para configurar la ip, puerto y tiempos de pesada dentro de *residuos.jar*.

Indicador.properties para las básculas

ORIONPLUS01B1\_IP=192.168.0.60

ORIONPLUS01B1\_PORT=502

ORIONPLUS01B1\_DIR\_STATUS=0

ORIONPLUS01B1\_DIR\_PUAT=3

ORIONPLUS01B1\_DIR\_PESO\_NETO=5

ORIONPLUS01B1\_DIR\_PESO\_TARA=7

# Segundos necesarios para enterder que la pesada es estable y definitiva

ORIONPLUS01B1\_SEG\_PESO\_MAX=3

# Tiempo de desbloqueo de la pesada

ORIONPLUS01B1\_TIEMPO\_TIMEOUT=21

# Margen que indica pesada cero en kg

ORIONPLUS01B1\_MARGEN\_PESADA\_CERO=200

Autómata.properties para las barreras

OMRON\_01\_IP=192.168.250.1

OMRON\_01\_PORT=502

OMRON\_01\_DIR\_TELM\_SUBIR=1000

OMRON\_01\_DIR\_TELM\_BAJAR=1001

OMRON\_01\_DIR\_EST\_SUBIDA=1002

OMRON\_01\_DIR\_EST\_DETECCION=1003

## Formas de ejecución

La aplicación se puede ejecutar de tres formas:

1. "java -jar basculas.jar"

//sin depuración.

1. “java -jar basculas.jar debug matricula C74432CRW"

//solo muestra mensajes de esa matricula

1. "java -jar basculas.jar debug"

//muestra todos los mensajes en un fichero log incremental según //estándar log4j

## funciones y clases

Primeramente la aplicación se ejecuta desde main(), en donde se carga la configuración inicial del programa del archivo “*config.xml*” y continua con la función de ejecutar(). Ésta última crea la clase de acceso a Datos y llama a una función que de forma infinita irá consultando los registros de bbdd (*consultaMatriculas()*).

### Función consultaMatriculas()

Mira los carriles que están definidos por bbdd, por cada uno de ellos crea una clase de control de carril *ControlCarril* y pone en blanco una variable de control de ocupación de carril llamada *operacionCarril* (Hashtable<idCarril,matriculaQueOcupaCarril>)

Después por cada carril, mira si no está ocupado y después si tiene matriculas.

Si llega el caso que la variable marca que está el carril ocupado, pero no hay matriculas pendientes para ese carril, se interpreta que el carril ha quedado libre. Esto entra dentro de la gestión de errores de la aplicación. Puede darse el caso de quedarse un registro pendiente por no haber llegado al final del hilo, al dar un error de báscula (peso) o de acceso a base de datos.

### Función trataMatricula

Se encarga de crear un hilo *hiloCamara*. Por cada matricula se crea un proceso run() del hilo, encargado de registrar en bbdd el movimiento, incidencias y peso, dependiendo del tipo de carril.

Hay un **switch** (*accionesCamara*(camara)) {

**case**

Encargado de ver el tipo de acción a realizar según la cámara definida en bbdd.

### Clase ControlCarril

Se creará una clase por cada carril y realiza las operaciones de peso y accionar las barreras.

### Clase BÁSCULA

Esta clase abre acceso a la báscula, pesa y cierra el acceso a la báscula.

Inicialmente se crea una clase Bascula por cada báscula o dispositivo existente. Esto lo hace en lee de BBDD la configuración de cámaras y carriles.

### Clase Datos

Contiene las operaciones de acceso a base de datos.

# gestion de memoria

La aplicación demonio ControlPlanta.jar tiene un archivo ejecutar.bat en donde se especifica la memoria reservada para su ejecución.

Java –Xms64m –Xmx128m –XX:PermSize=64m –XX:MaxPermSize=128m –jar ControlPlanta.jar debug

La aplicación Tomcat se cambiará para gestionar su memoria en el archivo catalina.bat.

Set JAVA\_OPTS=%JAVA\_OPTS% -Xms128m –Xmx512m –XX:PermSize=128m –XX:MaxPermSize=256m

Por si la ejecución del archivo startup.bat no funciona desde la línea de comandos, hay que añadir las siguientes variables al archivo startup.bat.

Set JAVA\_HOME=C:\Program Files\Java\jdk1.7.0\_51\bin

Set JRE\_HOME=C:\Program Files\Java\jdk1.7.0\_51\jre