Grado en Ingeniería Electrónica Industrial y Automática

Curso 2017-2018

Marzo 2018

Trabajo Fin de Grado

“Modelado y simulación de la zona patio-tierra de una terminal de contenedores”.

Daniel Manuel Pérez Melián

Tutor/es

Iván Castilla Rodríguez

Rosa María Aguilar Chinea

Resumen.

La toma de decisiones y la planificación de las actividades dentro de cualquier puerto son de gran importancia debido a que un simple fallo puede perjudicar de manera negativa a la actividad portuaria. Es por este motivo que se realiza una exhaustiva planificación, día a día, para que todas las actividades y decisiones estén coordinadas y se lleven a cabo de una manera eficaz. Para realizar las actividades de una forma óptima, todos los recursos humanos y materiales tienen que estar coordinados para evitar que se produzcan conflictos durante el proceso.

Podemos estudiar el conjunto de las actividades que se realizan como uno o varios sistemas interconectados, en los cuales se analiza el impacto de modificar su funcionamiento o la asignación de sus recursos. Debido al coste y las dificultades de este análisis, el uso de modelos nos permite entender el funcionamiento del sistema y disponer de una representación de la realidad, mediante simulación, donde se puede cuantificar el resultado de hacernos preguntas del tipo "¿qué pasaría si…?"

Para llegar a un modelo aproximado de la zona de patio-tierra de la terminal de contenedores del puerto de Santa Cruz de Tenerife, se ha utilizado la librería de simulación de eventos discretos PSIGHOS. PSIGHOS es una librería en Java creada por docentes de la Universidad de La laguna, que define un conjunto de clases que nos ha ayudado a simplificar la implementación del modelo, su simulación y la obtención de resultados de las simulaciones.

Para facilitar la interacción del usuario con el modelo se ha creado una interfaz gráfica que contiene una serie de ventanas las cuales nos permiten introducir los parámetros del modelo y tener resultados inmediatos mediante cuadros de texto o también de manera gráfica con diagrama de barras.

Se ha probado con la herramienta desarrollada varios ejemplos de casos de estudio de forma altamente satisfactoria. Con estos casos de estudio hemos querido reproducir la interacción que tendría un decisor con la herramienta.

Este proyecto abre la puerta al menos a dos vías de trabajos futuros: la mejora progresiva del nivel de detalle del modelo realizado y el desarrollo de una herramienta más potente de ayuda a la toma de decisiones.

Abstract.

The decision making and the planning of the activities within any port are of immense importance because a simple failure can negatively affect the port activity. For this reason, an exhaustive planning is carried out, day by day, so that all the activities and decisions are coordinated and carried out in an effective way. To carry out activities in an optimal way, all human and material resources must be coordinated to avoid conflicts during the process.

We can study the set of activities that are carried out as one or several interconnected systems, in which the impact of modifying their operation or the allocation of their resources is analyzed. Due to the cost and difficulties of this analysis, the use of models allows us to understand the functioning of the system and to have a representation of reality, through simulation, where we can quantify the result of asking questions of the type “what if ...?”

We have used the PSIGHOS discrete events simulation library to obtain an approximate model of the yard-land area of the container terminal of the port of Santa Cruz de Tenerife. PSIGHOS is a Java library created by researchers from the University of La Laguna, which defines a set of classes that help us simplify the implementation of the model, simulate it and obtain simulation results.

To facilitate the user's interaction with the model, a graphic interface has been created that contains a series of windows which allow us to enter the parameters of the model and have immediate results through text boxes or graphically with a bar diagram.

We have successfully tested several examples of case studies with the developed tool. With these case studies we wanted to reproduce the interaction that a decision maker would have with the tool.

This project opens the door to at least two lines of future work: the progressive improvement of the level of detail of the model carried out, and the development of a more powerful tool to help decision-making.

Índice

1. Introducción

1.1. Definición de Puerto

El término puerto puede ser utilizado con diferentes significados. Se trata, en su acepción más amplia, de la infraestructura que incluye diversos servicios para la realización de una cierta operación.

"El significado más habitual de puerto se asocia al espacio que, situado en una orilla o en la costa, permite que las embarcaciones desarrollen operaciones de descarga y carga o de desembarco y embarque. Cuando dicha infraestructura se halla junto al océano, se habla de puerto marítimo" [1].

Por esta definición, podemos decir que el puerto ofrece a las embarcaciones una zona segura donde poder resguardarse y realizar sus diferentes tareas. Esta seguridad la consigue mediante la construcción de diferentes infraestructuras como son los diques o esclusas, que protegen a los barcos del oleaje y favorecen a realizar las tareas de carga y descarga de una manera segura, y también el descenso y ascenso de pasajeros a los barcos con mayor seguridad.

1.2. Zonas de un puerto y tipos de puertos

Los puertos marítimos incluyen zonas para estancias de barcos denominadas dársenas, zonas de amarres de las embarcaciones llamadas muelles, almacenes donde poder almacenar mercaderías y distintas señalizaciones que sirven a los barcos como guía para el ingreso y regreso al puerto.

En todos los puertos podemos diferenciar tres áreas que se distinguen claramente, en función de actividades que en ellas se realicen:

- Zona marítima. En esta zona es donde se produce el atraque de los barcos siguiendo una serie de señalizaciones para que este atraque se produzca de una forma segura. Señalar que en esta zona la profundidad de las aguas es mínima.
- Zona terrestre o zona de patio. En nuestro caso, al ser un puerto de comercio lo definiremos como zona de patio. En este espacio se distinguen diferentes bloques de contenedores. Estos contenedores son descargados de los barcos y ubicados en su zona según del tipo que sean (mercancías peligrosas, de refrigeración, etcétera). En los puertos de comercio de todo el mundo, la medida estándar utilizada para los contenedores es TEU (Twenty-foot equivalent units).
- Zona de enlace o tierra. Esta zona está definida para la entrega de los contenedores a los camiones externos que vienen a recogerlos. Es decir, es la zona donde se produce la descarga del contenedor en el camión correspondiente. En esta zona también podemos encontrarnos las oficinas de las autoridades portuarias, almacenes, bodegas, etcétera.

Dentro de los puertos marítimos podemos diferenciar dos tipos:

- Los de altura, que realizan movimientos entre puertos internacionales.
- Los de cabotaje, que realizan movimientos entre puertos nacionales.

Los puertos marítimos de cualquier país generan una actividad económica muy importante, convirtiéndose en una base fundamental para su economía. Según varios estudios realizados por la Organización Mundial del Comercio (OMC), el

80% de las mercancías que se comercializan en todo el mundo se trasladan mediante los puertos marítimos.

1.3. Terminal de contenedores de Tenerife

La terminal de contenedores de Tenerife, con una ubicación estratégica, opera como un puerto de exportación/importación para el tráfico de mercancías a Tenerife e islas adyacentes. Dadas las características geográficas favorables de esta terminal, permite ser un punto de trasbordo de contenedores que tienen distintas rutas como, por ejemplo, conexión de Lejano Oriente, Medio Oriente y Mediterráneo con el Oeste de África y Sudamérica entre otras muchas rutas de conexión. Por este motivo cuenta con gran cantidad de recursos tanto físicos como humanos para ofrecer un servicio de calidad. Dentro del océano Atlántico, el puerto de Santa Cruz de Tenerife se ha convertido en un referente para el tráfico de mercancías, viéndose superado por el puerto de Las Palmas de Gran Canaria como uno de los puertos que más movimientos realiza en España [8].

1.4. Organización, desarrollo y problemática de las actividades

La organización y desarrollo de las actividades dentro de un puerto son de una gran complejidad, ya que llevan una exhaustiva organización y siempre puede verse afectada por las condiciones climatológicas, como pueden ser el oleaje, viento, lluvia, etcétera. Además, las tareas a realizar siempre estarán sujetas a las limitaciones horarias y también al número de recursos disponibles para la realización de estas. El número de movimientos que realizan al día (por turno de trabajo) los estibadores son planificados por los encargados con el fin de que no haya ningún problema de ejecución de las tareas.

La actividad zona patio-zona tierra puede verse afectada por el número de grúas *transtainer* disponibles en el bloque de contenedores, por el número de *reach stackers* (grúas de patio) disponibles, por el número de aparcamientos disponibles para los camiones que vienen a recoger los contenedores o por la ubicación de los contenedores, ya que un contenedor que viene a ser recogido puede estar en una mala ubicación y esto dificulta la tarea, ya que se tendría que llevar un proceso de recolocación de contenedores, por las grúas *transtainer*, para finalmente poder sacar ese contenedor de la pila de contenedores. Cabe destacar que hay contenedores especiales, como los que traen materiales hospitalarios, los cuales tienen prioridad sobre cualquier otro contenedor a la hora de ser sustraído de la pila de contenedores y a los cuales se destinan todos los recursos disponibles.

La pila de contenedores siempre suele ser de 5 contenedores, uno encima del otro, con unas dimensiones de 20 TAU o pies.

1.5. Objetivo del proyecto

El objetivo de este proyecto es poder simular la zona patio-tierra de la terminal de contenedores del puerto de Santa Cruz de Tenerife mediante la realización de un modelo dinámico. Dicho modelo parte desde que los *reach stackers* son llamadas para la descarga del contenedor en el bloque de contenedores y realizan la transferencia del contenedor con el camión externo que viene a recogerlo.

Se pretende que este proyecto sirva para ver el correcto funcionamiento de los recursos y para analizar futuras mejoras, por ejemplo, en cuanto a los recursos que se puedan incorporar o quitar para llegar a dar un mejor servicio y sacar el mayor rendimiento de la jornada laboral.
Para la elaboración de este modelo, se ha necesitado la ayuda de PSIGHOS que es un conjunto de clases en lenguaje de programación Java, ya creadas por personal docente de la Universidad de La Laguna, que facilita la implementación del modelo y su simulación. Cabe decir que aprender a programar en Java fue otros de los objetivos a cumplir ya que era algo imprescindible para el uso de PSIGHOS.
Por último, se ha propuesto la idea de realizar una interfaz gráfica simple para la interacción con el usuario donde se introducen los datos de la simulación y ver de forma gráfica distintos resultados que analizar.
Para conseguir la realización de este modelo, se ha tenido que investigar y entender el funcionamiento de un puerto marítimo en general, obtener una idea de funcionamiento especifica de la terminal de contenedores e intentar elaborar un modelo lo más próximo a la realidad.
Cabe destacar que este proyecto forma parte de una línea de investigación sobre la simulación de diferentes partes del puerto, del que ya se ha defendido el proyecto titulado *"Modelado y simulación de la zona mar-patio de un puerto de contenedores"* realizado por el compañero Alejandro Alonso Méndez. Ambos proyectos sientan la base de futuros desarrollos en esta línea.

1.6. Estructura de la memoria

En el capítulo 2 se definen los conceptos fundamentales sobre simulación, haciendo énfasis en los aspectos más relevantes a seguir para conceptualizar un modelo de simulación, así como las ventajas que este aporta. Se concluye introduciendo brevemente la simulación de eventos discretos.

Dentro del capítulo 3, se describe una terminal de contenedores y sus componentes principales.

El funcionamiento de la zona escogida en este proyecto, la presentación del modelo conceptual, la implementación/parametrización del modelo, y las interfaces gráficas se detallarán en el capítulo 4: *Simulación zona patio-tierra.*

Los resultados de nuestras simulaciones serán expuestos en el capítulo 5 denominado *Resultados.*

El apartado de conclusiones podemos encontrarlo dentro del capítulo 6, donde se valora el resultado de los objetivos propuestos y se sugieren futuras investigaciones en esta materia.

Para terminar, se incluye el listado de bibliografía citada en el texto y anexos con el código utilizado.

2. La simulación

2.1. Definición de Simulación

Según R. E. Shannon [2]: "*La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los límites impuestos por un cierto criterio o un conjunto de ellos- para el funcionamiento del sistema*".

2.2. Realización de un modelo de simulación y sus ventajas

Para comenzar a realizar una simulación hay que definir unos criterios para estudiar el contexto del problema. Primero se identifican los objetivos del proyecto, seguidamente se miden que parámetros son efectivos y nos pueden dar información relevante del modelo. A continuación, se establecen los objetivos específicos del modelado. Luego, se hace una formulación del modelo donde es necesario definir todas las variables que forman parte de él, sus relaciones lógicas y los diagramas de flujo que describan de forma completa el modelo. Por último, se haría una implementación del modelo en computadora, con un conjunto de datos representativos para obtener los resultados deseados, y se verificarían para ver si el modelo simulado cumple con los requisitos de diseño para los que se elaboró [3].

En nuestro caso, la simulación, nos ayudaría para realizar distintos escenarios, obtener resultados inmediatos que nos pueden ayudar a la hora de realizar una planificación más eficiente y mejorar la calidad del servicio que se le presta al cliente. Si no tuviéramos la posibilidad de realizar una simulación, implantar mejoras sería más costoso, ya que habría que interrumpir la actividad productiva de trabajo para ejecutar pruebas de mejoras sin la certeza de que sean eficaces.

Con la simulación, en nuestro caso, podemos obtener unos parámetros de tiempo que nos pueden decir cuánto tardará un contenedor en salir del puerto o simplemente si el número de recursos que estamos utilizando es del todo eficiente.

Se puede decir que la simulación es una herramienta adaptable a cualquier sistema que quiera simular, simple, económica y te permite obtener resultados de inmediato.

2.3. Simulación de eventos discretos

"*La simulación de eventos discretos se refiere al modelado computacional de sistemas que evolucionan en el tiempo mediante cambios instantáneos en las variables de estado. Una simulación de eventos discretos modela la operación de un sistema como una secuencia discreta de eventos en el tiempo. Cada evento ocurre en un instante de particular en el tiempo y marca un cambio de estado en el sistema. Entre sucesos consecutivos, no se supone que ocurra ningún cambio en el sistema; por tanto, la simulación puede saltar directamente en el tiempo de un evento al siguiente. Esto contrasta con la simulación continua en la que la simulación rastrea continuamente la dinámica del sistema a lo largo del tiempo. Debido a que las simulaciones de eventos discretos no tienen que simular cada segmento de tiempo, generalmente pueden ejecutarse mucho más rápido que la simulación continua*" [4].

3. Terminal de contenedores

3.1. Concepto de terminal de contenedores

"*Una terminal de contenedores es un intercambiador intermodal dotado de una capacidad determinada de almacenamiento en tierra capaz de regular los diferentes ritmos de llegada de los medios de transporte terrestre y marítimos"* [5].

El objetivo principal es garantizar una organización eficaz en el intercambio de contenedores y proporcionar los medios para que la carga y descarga de contenedores se produzca en las mejores condiciones.

Una terminal de contenedores puede ser definida como un sistema compuesto por varios subsistemas que tienen conexión entre sí. Los subsistemas son:

- El de carga y descarga de contenedores.
- El de almacenamiento de contenedores (que abarca la mayor superficie de la terminal).
- El de la recepción y entrega terrestre.
- El de la conexión interna (este asegura el transporte horizontal de los contenedores entre los subsistemas anteriores).

3.1.1. Subsistema de carga y descarga de contenedores

Este se encarga de resolver la interfaz marítima. Este subsistema se caracteriza por el predominio del buque y las consecuencias que ello conlleva.

El principal objetivo de este subsistema es atender la demanda de carga y descarga de contenedores del buque con la mayor rapidez y seguridad posible, tanto en la atención directa al barco como en lo que respecta a la relación con el medio de transporte usado para mover las cargas de la terminal.

La eficiencia de este subsistema depende los recursos disponibles para la realización de las distintas actividades.

3.1.2. Subsistema de almacenamiento

El principal objetivo de este subsistema es proporcionar una forma eficaz de atender los diferentes ritmos que existen entre la carga y descarga de buques, y la recepción y entrega de las mercancías a los modos de transporte terrestre. Es atendida por los medios de manipulación de contenedores (grúas de patio o reach stackers).

El subsistema de almacenamiento viene determinado a gran medida por el tipo de medios de manipulación que van a utilizar. Estos, a su vez, permiten grados de apilamiento y posibilidades de automatización muy dispares de manera que la elección de estos medios condiciona de una manera esencial a la propia terminal.

Dentro de esta zona de almacenamiento, también podemos encontrarnos unas zonas adicionales, que son para el uso específico de ciertos contenedores como son:

- Contenedores refrigerados.
- Contenedores de mercancías peligrosas.

- Contenedores que tienen que pasar por inspecciones.

Dentro de esta zona de almacenamiento también podremos encontrarnos las oficinas de la terminal, los talleres mecánicos o los almacenes.

3.1.3. Subsistema de la recepción y entrega terrestre

Este subsistema se encarga de la interfaz terrestre, donde se tiene que atender generalmente al transporte por carretera. El principal objetivo de este subsistema es facilitar la recepción o entrega de mercancías de una manera rápida en condiciones de seguridad en la obtención de la información con el elevado número de intercambio documental y de información.

3.1.4. Subsistema de la conexión interna

Este subsistema permite el intercambio de mercancías entre los diferentes subsistemas.

La principal labor del subsistema de interconexión es servir eficazmente como medio de distribución interior de los contenedores, atendiendo a los requerimientos específicos que le exijan los demás subsistemas. Son exigibles a este subsistema, la rapidez adecuada, la seguridad, fiabilidad mecánica, así como la correspondiente al funcionamiento lógico, es decir, la reducción o eliminación de errores de entrega.

3.2. Contenedores

"*Un contenedor es un recipiente de carga para el transporte marítimo o fluvial, transporte terrestre y transporte multimodal. Se trata de unidades estancas que protegen las mercancías de la climatología y que están fabricadas de acuerdo con la normativa ISO (International Organization for Standardization), en concreto, ISO-668; por ese motivo, también se conocen con el nombre de contenedores ISO. Fabricados principalmente de aluminio, acero y de materiales sintéticos. Pueden soportar cargas de hasta seis contenedores apilados encima sin que se deformen*" [6].

3.2.1. Dimensiones del contenedor

Los contenedores más utilizados a nivel mundial son los de 20 pies y 40 pies de largo, con un volumen aproximado de 32.6 m^3 y 66,7 m^3 respectivamente. En Europa son las medidas más utilizadas.

Hay varios tipos de medidas, variando en largo y alto:

- El ancho se fija en 7,9 pies (2,35 m).
- El alto varía entre 8 pies y 6 pulgadas (2,62 m) y 9 pies y 6 pulgadas (2,92 m).
- El largo varía entre 8 pies (2,44 m), 10 pies (3,05 m), 20 pies (6,10 m), 40 pies (12,19 m), 45 pies (13,72 m), 48 pies (14,63 m) y 53 pies (16,15 m).

Los contenedores más utilizados son los *Dry Van* que son los contenedores estándar, cerrados herméticamente y sin refrigeración o ventilación. Sus medidas interiores son:

- ***Para 20 pies (20´ x 8´ x 8,6´´)***
  - Tara: 2300 Kg / 5070 lb
  - Carga máxima: 28180 Kg / 62130 lb
  - Peso bruto: 30480 Kg / 67200 lb
  - Largo: 5898 mm / 19´4´´
  - Ancho: 2352 mm / 7´9´´
  - Altura: 2393 mm / 7´10´´
  - Capacidad: 33,2 m^3 / 1172 ft^3

- ***Para 40 pies (40´ x 8´ x 8,6´´)***
  - Tara: 3750 Kg / 8265 lb
  - Carga máxima: 28750 Kg / 63385 lb
  - Peso bruto: 32500 Kg / 71650 lb
  - Largo: 12025 mm / 39´6´´
  - Ancho: 2352 mm / 7´9´´
  - Altura: 2393 mm / 7´10´´
  - Capacidad: 67,7 m^3 / 2390 ft^3

- ***Para 40 pies High Cube (20´ x 8´ x 9,6´´)*** (Contenedores estándar de 40 pies que se caracterizan por su sobrealtura (9´6 pies))
  - Tara: 3940 Kg / 8685 lb
  - Carga máxima: 28560 Kg / 62936 lb
  - Peso bruto: 32500 Kg / 71650 lb
  - Largo: 12032 mm / 39´6´´
  - Ancho: 2352 mm / 7´9´´
  - Altura: 2698 mm / 8´10´´
  - Capacidad: 76 m^3 / 2700 ft^3

3.2.2. Tipos de contenedores

A parte de los ya mencionados contenedores estándar *Dry Van* existen otros tipos de contenedores como, por ejemplo:

- *Reefer*: contenedores refrigerados, de 20 o 40 pies. Cuentan con un sistema de conservación de frio o calor y termostato.
- *Open Top*: mismas medidas que el anterior. Están abiertos por la parte de arriba.
- *Flat Rack*: carecen de paredes laterales en algunos casos tampoco cuentan con paredes delanteras y posteriores.
- *Open Side*: de 20 o 40 pies de largo y está abierto por uno de sus lados.
- *Tank* o contenedor cisterna: sirven para el transporte de líquidos a granel.
- *Conair*: son aislante y mantiene el contenedor a una temperatura constante.
- *Flexi – Tank*: consiste en un contenedor estándar de 20 pies normalmente que en su interior se instala un depósito flexible de polietileno de un solo uso.

Ilustración 1. Tipos de contenedores

3.3. Grúas

Dentro de la terminal de contenedores nos encontraremos con distintas grúas encargadas de realizar las actividades de carga y descarga de contenedores. Las de más relevancia son:

- Las grúas *transtainers* que pueden desplazarse tanto por railes como por neumáticos. Si es por railes su rango de operaciones es limitado mientras que si es sobre neumáticos el rango de operaciones es mucho mayor. Sobre neumáticos las ruedas frontales de estas grúas pueden girar 90 grados facilitando muchas maniobras de trabajo.

  Estas grúas *transtainers* son usadas en todos los puertos debido a que puede usarse en diversidad de trabajos de manipulación de cargas. Están equipadas con spreaders, cucharas, ganchos y otros aparejos de elevación y sujeción de cargas.

Ilustración 2. Grúa transtainer cargando contendor

- Las grúas de patio o *reach stackers* son unos vehículos para el manejo de contenedores en pequeñas terminales o puertos de tamaño medio. Pueden transportar un contenedor en pequeñas distancias de una forma rápida y apilarlos de una manera fácil. Estas grúas han ganado protagonismo en el mercado debido a su flexibilidad y amplia capacidad para apilar y transportar contenedores dentro de las terminales de contenedores.

Ilustración 3. Reach stackers apilando contenedor

4. Simulación zona patio-tierra

4.1. Funcionamiento zona patio-tierra

En la zona de patio nos encontramos con los distintos tipos de contenedores, pudiendo ser de exportación, importación, refrigerados, materiales peligrosos, vacíos, de carga rodante (vehículos, camiones), de trasbordo, etcétera.

Todos ellos tienen una ubicación dentro de la terminal de contenedores y unas especificaciones concretas para así facilitar las actividades:

- De descarga/carga del contenedor en el patio mediante grúa de patio o *reach stackers*
- De descarga del contenedor en los camiones externos.

Cabe decir, que los contenedores que van a estar pocos días en el puerto, si su salida es mediante barco suelen ubicarse lo más próximo a la zona de mar y si su salida es mediante tierra se ubicarán más cerca de la zona de espera de los camiones externos.

También nos encontramos con las *reach stackers* o grúas de patio que son unas grúas-camión encargadas de recibir las ubicaciones de los contenedores, cargarlos del bloque de contenedores y llevarlos hacia la zona de espera donde estarán los camiones de recogida.

La zona de tierra está definida como la zona de espera de los camiones externos que vienen a recoger su carga. Este tiempo de espera del camión se define como el tiempo que se tarda en la zona de espera desde que el camión entra en dicha zona hasta que sale de ella. Antes de que el camión externo entre en la zona de espera, éste tiene que identificarse en una cabina de control. Debe identificarse el conductor del camión y poner el código del contenedor que viene a recoger, ya sea con contenido o vacío. Una vez identificado el camión y el contenedor que viene a recoger, se genera un ticket que le indica al camión qué aparcamiento de la zona de espera tiene que ocupar. En un proceso paralelo, se le manda a la grúa *transtainers* la identificación del contenedor que tiene que extraer. Una vez localizado el contenedor, se genera una petición de recogida a la grúa de patio o *reach stackers* para que cargue el contenedor y se lo lleve al camión que estará esperando en la zona de espera indicada. Una vez se descargue el contenedor al camión la *reach stacker* puede volver a la zona de espera de las grúas de patio o puede realizar alguna otra actividad que se le indique.

4.2. Modelo conceptual

En este apartado vamos a explicar de una forma sencilla y esquemática el funcionamiento de nuestro modelo. Nuestro proyecto se va a centrar en un sólo bloque de contenedores, ya que viendo el funcionamiento de este modelo podemos ampliarlo con relativa facilidad a varios bloques de contenedores.

Para la realización de nuestro proyecto nos hemos elaborado un pequeño esquema que representa un bloque de contenedores (Ilustración 4). En este esquema se puede visualizar el bloque de contenedores, los tramos por donde circularán las grúas y la zona de espera donde se ubicarán los camiones externos.

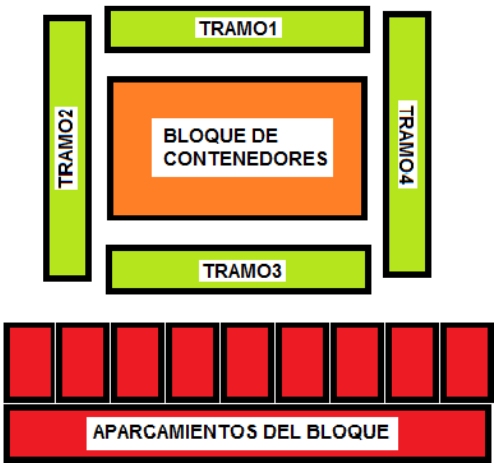

Ilustración 4. Esquema del modelo

Nuestro proyecto se compone de una serie de actividades las cuales van a requerir una serie de recursos y unos ciertos tiempos de duración. En el diagrama de bloques de la Ilustración 5 podemos visualizar en qué sentido se desarrollarían las actividades y que recursos son necesarios para ejecutar las actividades.

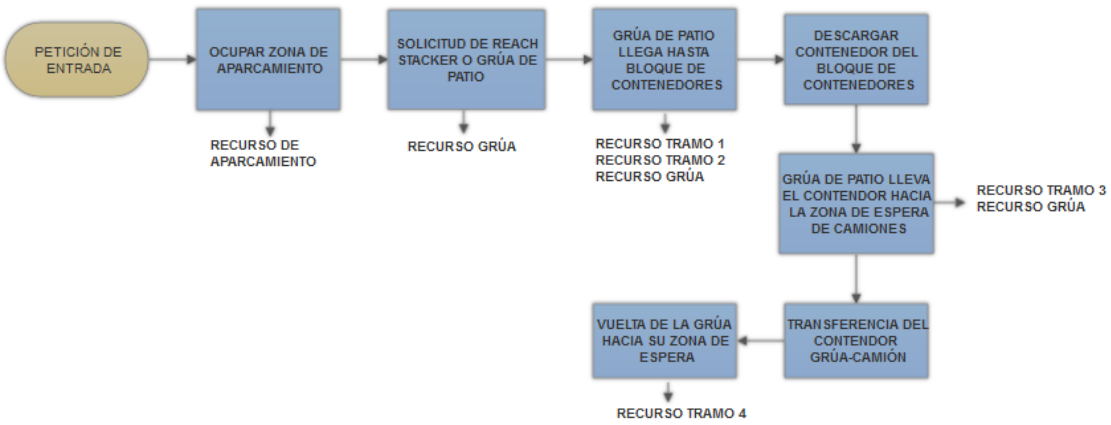

Ilustración 5. Diagrama de las actividades.

Para la elaboración de este modelo se ha tenido que declarar un elemento principal llamado *petición,* sobre este se crearán las actividades y se asignarán los recursos necesarios para el desarrollo de las actividades. La primera actividad es ocupar zona de aparcamiento, en la cual necesitamos un recurso aparcamiento que permita el desarrollo de esta. Luego, una vez tengamos el aparcamiento, comenzará la siguiente actividad donde se requiere la disponibilidad de una *reach stacker* que es necesaria para

transportar el contenedor desde el bloque de contenedores hasta la zona de aparcamiento del camión. Una vez tengamos el recurso de la *reach stacker,* empezará la actividad de llegar hasta el bloque de contenedores, para su recorrido necesita los recursos tramo 1 y tramo 2. En el tramo 2, es donde se realizará la actividad de descarga del contenedor, aquí abría una grúa transtainer que se ha simulado con un tiempo de descarga del contenedor en patio. Cargado el contendor, la *reach stacker* pasará a ocupar el tramo 3, en el que realiza la transferencia del contenedor con el camión. Finalizada la transferencia, se ocupará el tramo 4 y seguidamente quedará finalizado todo el flujo de actividades de nuestro modelo.

4.3. Implementación del modelo

Para la implementación del modelo, lo primero que hicimos fue crearnos un diagrama de flujo de las acciones que se llevarían a cabo durante todas las actividades de nuestro modelo. Para ello, utilizamos tres módulos que nos ofrece PSIGHOS para la interacción con los recursos y la definición de esperas en tiempo:

- Request: con el cual indicamos que una acción requiere o necesita un tipo de recurso.
- Released: indicamos que la acción ha finalizado y que el recurso que requería debe liberarse
- Delay: nos permite asignar un tiempo durante el cual una entidad (la petición, en nuestro caso) debe esperar antes de seguir avanzando.

Resaltar que dentro de este modelo es muy importante el cambio que se produce entre tramos, ya que si el tramo siguiente no está libre no puedo dejar el tramo en el que estoy. Por ejemplo, si estoy en el tramo uno y quiero pasar al tramo dos, no puedo hacerlo directamente sino tengo que requerir primero el tramo dos a ver si está disponible y si lo está entonces puedo dejar el tramo uno y ocupar el tramo dos. Entre esta secuencia hay un tiempo que marcará el tiempo que se permanece en el tramo anterior al que vas a ocupar. Con esta lógica nos aseguramos de que la grúa no se quede perdida entre los tramos, sino que siempre este siguiendo el flujo que se le indica.

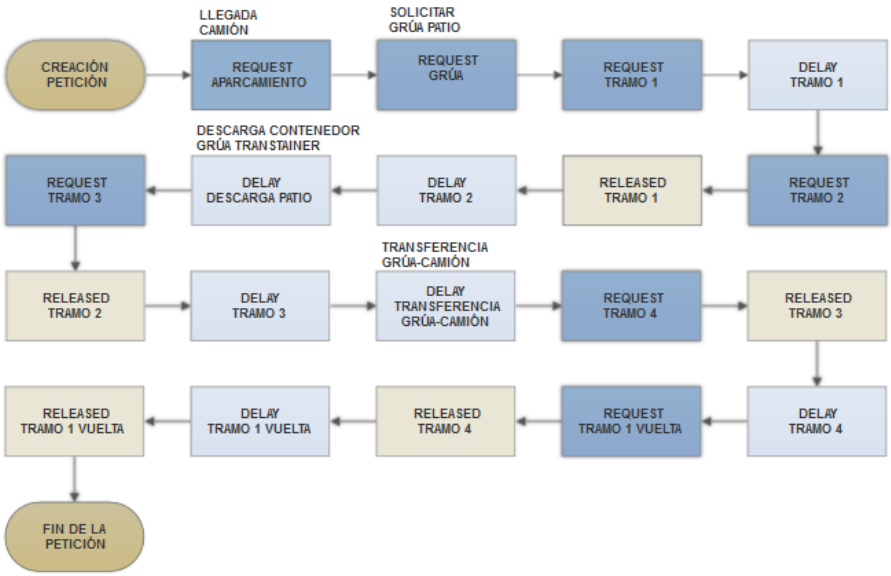

Ilustración 6. Diagrama de flujo de las acciones. Se puede apreciar lo explicado anteriormente del funcionamiento de los tramos.

En la Ilustración 6 se puede ver de forma gráfica como se estructura nuestro modelo a simular. La secuencia de movimientos de la grúa siempre será la siguiente:

- En un principio suponemos que la grúa de patio o *reach stacker* estará en una zona de espera, en la cual espera a que se le asigne una petición de recogida de un contenedor.
- Una vez asignada una petición de recogida, pasará a ocupar el tramo uno donde permanecerá un cierto tiempo antes de pasar a ocupar el tramo dos.
- Antes de pasar del tramo dos, se asegurará de que este tramo esta liberado y puede ser ocupado, y entonces soltará el tramo uno.
- Una vez en el tramo dos se dispondrá a realizar la descarga del contendor, el cual ya ha sido localizado por la grúa *transtainer*. La descarga de contendores siempre se realizará en el tramo dos, siendo los otros tramos para la circulación de las grúas. La grúa *transtainer* ha sido simulada mediante un tiempo de descarga en el patio.
- Antes de pasar del tramo tres, se asegurará de que este tramo esta liberado y puede ser ocupado, y entonces soltará el tramo dos.
- Una vez en el tramo tres, se producirá la transferencia del contenedor con el camión externo, que estará esperando en los aparcamientos del bloque. La transferencia fue definida mediante un tiempo aproximado de cuanto puede tardar en descargar la grúa un contenedor al camión.
- Antes de pasar del tramo cuatro, se asegurará de que este tramo esta liberado y puede ser ocupado, y entonces soltará el tramo tres.
- El tramo cuatro es utilizado como regreso de la grúa hacia la zona de espera, teniendo que pasar, por último, por el tramo 1 para confirmar que ya ha finalizado la petición actual y que la grúa ha sido liberada. En este momento se da por concluida la petición y ya la grúa está disponible para recibir otra petición de descarga de contenedor.

La asignación de los aparcamientos para los camiones, en la zona de espera, se realizará de una forma aleatoria. Según vayan llegando los camiones se creará la petición de recogida del contenedor y se les irá asignando un aparcamiento del bloque de contenedores hasta que este llegue a su ocupación total. En el caso de que un camión llegue y no tenga aparcamiento tendrá que esperar hasta que se libere un aparcamiento y entonces pueda acceder al mismo. Mientras no tenga aparcamiento no se puede generar la petición de recogida.

Con las grúas ocurrirá un proceso similar al de los camiones. El número de grúas no será tanto como el número de aparcamientos. Por lo tanto, un camión puede llegar a ocupar una plaza de aparcamiento y la grúa puede estar realizando otra petición anterior a la suya. Entonces, el camión tendrá que esperar un cierto tiempo hasta que la grúa pueda gestionar su petición de entrada.

La funcionalidad de los tramos ya se explicó anteriormente en los movimientos de las grúas dentro del patio de contenedores. Pero también puede pasar que se forme algún tipo de “embudo” en alguno de los tramos, debido a que en algunos tramos las grúas estarán más tiempo y esto puede generar una cola de espera por ocupar ese tramo.

Una vez tenemos el modelo compilado y sin errores, el siguiente paso es ver el correcto funcionamiento del flujo de actividades y la extracción de datos a analizar. Para ello, SIGHOS proporciona unas clases denominadas *listeners*, que son los encargados de escuchar los eventos que pasan en la simulación y realizan unas acciones para poder recibir la información de esos eventos. El primer *listener* utilizado fue para ver si el flujo

de acciones seguía con el orden establecido y se cumplían todos los Request, Released y Delay definidos en el modelo. Viendo el correcto funcionamiento de este listener, pasamos a realizar otro dónde esta vez sacaríamos el tiempo de estancia del camión, es decir, el tiempo que tarda el camión en entrar al puerto y salir del mismo o también se podría ver como el tiempo que se tarda por petición de entrada. Y, por último, realizamos un *listener* donde pudimos ver todos los conflictos que se producían en los recursos de nuestra simulación.

A continuación, se expondrán partes del código con una breve explicación de las clases utilizadas para ver de una forma dinámica como ha sido la creación de este proyecto.

En nuestra clase “Element Type” definimos el único elemento de nuestra simulación que son las peticiones de recogida de contenedor.

```
89 //TIPOS DE ELEMENTOS DE NUESTRO PROGRAMA
90     ElementType Peticion = new ElementType(this, PETICION_A);
91
```

Ilustración 7. Parte del código donde se definen los elementos.

Los recursos son aquellas partes del modelo necesarias para realizar las actividades. Por ejemplo, para la actividad ocupar zona de aparcamiento tiene que haber un recurso aparcamiento que permita que la actividad se lleve a cabo. Estos recursos pueden ser de tipo humano o material, dependiendo del modelo a realizar, y también pueden que estén ocupados o disponibles dependiendo si hay otra actividad utilizando los recursos. En nuestro modelo, están definidos como “ResourceType” y han sido creados 6 tipos de recursos, que son:

- Recurso de tipo grúa
- Recurso de tipo aparcamiento del bloque
- Recurso de tipo tramo 1
- Recurso de tipo tramo 2
- Recurso de tipo tramo 3
- Recurso de tipo tramo 4

Cabe decir que los tramos fueron definidos como recursos para poder definir un recorrido por el que la grúa hiciese su recorrido. El funcionamiento de estos tramos se explicará de forma detallada en la implementación del modelo.

```
94 // TIPOS DE RECURSOS DE NUESTRO PROGRAMA
95     ResourceType rtGrua = new ResourceType(this, "Grúa");
96     ResourceType rtParkingA = new ResourceType(this, "Aparcamiento Bloque A");
97     ResourceType rtTramoA1 = new ResourceType(this, "Tramo A1");
98     ResourceType rtTramoA2 = new ResourceType(this, "Tramo A2");
99     ResourceType rtTramoA3 = new ResourceType(this, "Tramo A3");
100    ResourceType rtTramoA4 = new ResourceType(this, "Tramo A4");
```

Ilustración 8. Fragmento del código donde están definidos los recursos.

Después de haber creado los tipos de recursos, necesitamos crear una clase que nos genere n-tipos de recursos necesarios para nuestro modelo. Para ello acudimos a los “Resource” y a una de las clases del PSIGHOS, llamada “addGenericResources()”, que nos permitirá crear esos n-tipos de recursos.

```
105     Resource[] resGruas = rtGrua.addGenericResources(nGruas);
106     Resource[] resParkingA = rtParkingA.addGenericResources(nCallesA);
107     Resource[] resTramoA1 = rtTramoA1.addGenericResources(nCallesInt1);
108     Resource[] resTramoA2 = rtTramoA2.addGenericResources(nCallesInt2);
109     Resource[] resTramoA3 = rtTramoA3.addGenericResources(nCallesInt3);
110     Resource[] resTramoA4 = rtTramoA4.addGenericResources(nCallesInt4);
111
```

Ilustración 9. Muestra de código donde se definen los Resource.

Seguidamente creamos unos “WorkGroup” en los que se agrupan los recursos necesarios para realizar las distintas actividades. Cada actividad de nuestro modelo tendrá definida su “WorkGroup” con sus recursos necesario para llevar a cabo las actividades. En nuestro modelo se han creado siete grupos de trabajo, que son:

- Grupo de trabajo grúa el cual necesita un recurso de tipo grúa.
- Grupo de trabajo aparcamiento el cual necesita un recurso de tipo aparcamiento del bloque.
- Grupo de trabajo tramo 1 el cual necesita un recurso de tipo tramo 1.
- Grupo de trabajo tramo 2 el cual necesita un recurso de tipo tramo 2.
- Grupo de trabajo tramo 3 el cual necesita un recurso de tipo tramo 3.
- Grupo de trabajo tramo 4 el cual necesita un recurso de tipo tramo 4.
- Grupo de trabajo vacío. Este grupo de trabajo se crea para que cuando el proceso llegue al tramo 4, que es el último tramo que ocupa, se indique que el proceso ha finalizado, que la grúa ha quedado libre y que ya se le puede asignar otra actividad.

```
118     WorkGroup wgGrua = new WorkGroup(this, rtGrua, 1);
119     WorkGroup wgParkingA = new WorkGroup(this, rtParkingA, 1);
120     WorkGroup wgTramoA1 = new WorkGroup(this, rtTramoA1, 1);
121     WorkGroup wgTramoA2 = new WorkGroup(this, rtTramoA2, 1);
122     WorkGroup wgTramoA3 = new WorkGroup(this, rtTramoA3, 1);
123     WorkGroup wgTramoA4 = new WorkGroup(this, rtTramoA4, 1);
124     WorkGroup wgVacio = new WorkGroup(this);
```

Ilustración 10. Parte de código donde están definidos los “WorkGroup”.

Las acciones que componen nuestras actividades vendrán definidas mediante “RequestResourcesFlow” y “ReleasedResourceFlow”. Estas dos clases vienen determinadas por la herramienta PSIGHOS. Con esto podemos determinar cuándo comienza y acaba una acción de alguna de las actividades, y también que grupo de trabajo se le asigna para realizar la actividad.

```
126 // DEFINIMOS TODAS LAS ACCIONES QUE LLEVARA A CABO NUESTRO PROCESO
127         RequestResourcesFlow reqCamion = new RequestResourcesFlow(this, LLEGADA);
128         reqCamion.addWorkGroup(wgParkingA);
129         ReleaseResourcesFlow relCamion = new ReleaseResourcesFlow(this, FINAL, wgParkingA);
130
131         RequestResourcesFlow reqGrua = new RequestResourcesFlow(this, GRUAIN);
132         reqGrua.addWorkGroup(wgGrua);
133         ReleaseResourcesFlow relGrua = new ReleaseResourcesFlow(this,GRUAOFF,wgGrua);
134
135         RequestResourcesFlow reqTramoPatio1 = new RequestResourcesFlow(this, ON_TRAMO1);
136         reqTramoPatio1.addWorkGroup(wgTramoA1);
137         ReleaseResourcesFlow relTramoPatio1 = new ReleaseResourcesFlow(this,OFF_TRAMO1,wgTramoA1);
138
139         RequestResourcesFlow reqTramoCentro1 = new RequestResourcesFlow(this, ON_TRAMO2);
140         reqTramoCentro1.addWorkGroup(wgTramoA2);
141         ReleaseResourcesFlow relTramoCentro1 = new ReleaseResourcesFlow(this,OFF_TRAMO2,wgTramoA2);
142
143         RequestResourcesFlow reqtramotierra1 = new RequestResourcesFlow(this, ON_TRAMO3);
144         reqtramotierra1.addWorkGroup(wgTramoA3);
145         ReleaseResourcesFlow reltramotierra1 = new ReleaseResourcesFlow(this,OFF_TRAMO3,wgTramoA3);
146
147         RequestResourcesFlow reqtramocentro2 = new RequestResourcesFlow(this, ON_TRAMO4);
148         reqtramocentro2.addWorkGroup(wgTramoA4);
149         ReleaseResourcesFlow reltramocentro2 = new ReleaseResourcesFlow(this,OFF_TRAMO4,wgTramoA4);
150
151         RequestResourcesFlow reqtramopatio1vuelta = new RequestResourcesFlow(this, ON_TRAMO1VUELTA);
152         reqtramopatio1vuelta.addWorkGroup(wgVacio);
153         ReleaseResourcesFlow reltramopatio1vuelta = new ReleaseResourcesFlow(this,OFF_TRAMO1VUELTA,wgVacio);
```

Ilustración 11. Parte del código donde se definen las acciones de nuestro modelo.

Por ejemplo, la acción “RequestResourceFlow reqcamion” nos indica la llegada de un camión y la necesidad de coger un aparcamiento para realizar su actividad. Se le asigna un aparcamiento del bloque para la espera de su contenedor y una vez se haya terminado la actividad con el ““ReleasedResourceFlow reqcamion” indicamos que ha finalizado y que el aparcamiento ocupado ha quedado libre.

Para cada una de las acciones el procedimiento es exactamente igual, se indica primero que va a requerir la acción, luego se le asigna un grupo de trabajo y cuando acabe se indica que ha finalizado.

Para generar nuestro modelo tenemos que unir todas las acciones creadas anteriormente con los “RequestResourcesFlow” y “ReleasedResourceFlow” mediante el método “link”. Este método nos permite crear un flujo de trabajo definiendo el inicio, las acciones con sus tiempos y el final de dicho flujo.

```
166 // Primero enlazo las acciones desde que se inicia el pedido hasta que la grúa llega al tramo central
167 reqCamion.link(reqGrua).link(reqTramoPatio1).link(treqPatio1).link(reqTramoCentro1).link(relTramoPatio1).link(treqCentro1);
168 // Después enlazo con la descarga
169 treqCentro1.link(tDescargar);
170 // Ahora enlazo las acciones desde que se descarga hasta que la grúa llega a donde está el camión
171 tDescargar.link(reqtramotierra1).link(relTramoCentro1).link(treqTierra1);
172 // Ahora enlazo desde que se realiza la tranferencia hasta que empieza la vuelta de la grua hasta el tramo central de vuelta
173 treqTierra1.link(tTransferencia).link(reqtramocentro2).link(reltramotierra1).link(treqCentro2);
174 // Se enlazan las actividades que van desde el tramo central de vuelta hasta la zona de parking de las gruas
175 treqCentro2.link(reqtramopatio1vuelta).link(reltramocentro2).link(tTramopatio1vuelta).link(reltramopatio1vuelta).link(relGrua).link(relCamion);
```

Ilustración 12. Parte de código donde está el flujo de trabajo del modelo

La clase “ElementGenerator” la utilizamos para crear los elementos que van a interactuar en la simulación. En el caso de nuestra simulación, generará las peticiones de entrada que son nuestro elemento principal e inicializará el flujo de trabajo que está definido anteriormente. En nuestro caso la generación de peticiones coincidirá con el número de camiones que llegarán por hora a recoger su contenedor.

```
177 //GENERADOR DE ELEMENTOS
178         TimeDrivenElementGenerator gen = new TimeDrivenElementGenerator(this, nCamiones, Peticion, reqCamion, CamionCycle);
179
```

Ilustración 13. Línea del código donde se define el “ElementGenerator”.

También se inicializa el “CamionCycle” que es el período en el que los camiones irán llegando al puerto, en nuestro caso serán los camiones que llegarán por hora a recoger

su contenedor. En el puerto, los trabajadores, por ley tienen turnos de trabajo de seis horas diarias. Por lo tanto, en nuestras simulaciones simularemos un turno de trabajo de seis horas solamente.

```
111
112 //CON ESTO DEFINIMOS LOS CAMIONES QUE LLEGARAN POR HORA
113     SimulationPeriodicCycle CamionCycle = SimulationPeriodicCycle.newHourlyCycle(unit);
114
```

Ilustración 14. Línea del código donde se definen los camiones por hora que llegarán.

Una vez realizados estos pasos, como se explicó en el inicio de esta sección lo que queda es comprobar que no hay errores de compilación, que el flujo de actividades se cumple correctamente y empezar a extraer información mediante los *listeners*.

4.3.1. Parametrización del modelo

Para disponer de tiempos de actividad que incorporaran un comportamiento estocástico, se definieron mediante una distribución Gamma. "*Esta distribución Gamma es adecuada para modelizar el comportamiento de variables aleatorias continuas con asimetría positiva. En su expresión se encuentran dos parámetros, siempre positivos como son alfa y beta, de los que depende su forma y alcance por la derecha. El parámetro alfa sitúa la máxima intensidad de probabilidad y por este motivo es denominada la forma de la distribución. El parámetro beta es el que determina la forma o alcance de la asimetría positiva desplazando la densidad de probabilidad en la cola de la derecha* "[7].

```
155 // SE DEFINEN LOS TIEMPOS QUE TARDARA EL PROCESO EN PASAR POR LOS TRAMOS DEL PATIO Y EN REALIZAR
156 // LAS ACTIVIDADES DE LOCALIZACION,DESCARGA, TRANSFERENCIA, ETC
157         DelayFlow treqPatio1 = new DelayFlow(this, "Tiempo Tramo 1", TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_PATIO, BETA_TRAMO_PATIO));
158         DelayFlow treqCentro1 = new DelayFlow(this, "Tiempo Tramo 2", TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_CENTRO, BETA_TRAMO_CENTRO));
159         DelayFlow tDescargar = new DelayFlow(this, DESCARGA, TimeFunctionFactory.getInstance("GammaVariate", ALFA_DESCARGA_CONTENEDOR, BETA_DESCARGA_CONTENEDOR));
160         DelayFlow treqTierra1 = new DelayFlow(this, "Tiempo Tramo 3", TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_TIERRA, BETA_TRAMO_TIERRA));
161         DelayFlow tTransferencia = new DelayFlow(this,TRANSFERENCIA,TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRANSFERENCIA, BETA_TRANSFERENCIA ));
162         DelayFlow treqCentro2 = new DelayFlow(this,"Tiempo Tramo 4", TimeFunctionFactory.getInstance("GammaVariate", ALFA_CENTRO_VUELTA, BETA_CENTRO_VUELTA));
163         DelayFlow tTramopatio1vuelta = new DelayFlow(this, "Tiempo Tramo 1 vuelta", TimeFunctionFactory.getInstance("GammaVariate", ALFA_VUELTA_PATIO1, BETA_VUELTA_PATIO1));
```

Ilustración 15. Fragmento de código donde están definidos los tiempos.

Los parámetros alfa y beta se calcularon a partir de la media y un error estándar que puede ser modificado por el usuario.

```
29     private static final double ERROR = 0.25;
30     private static final long MEDIA_DESCARGA_CONTENEDOR = 4*60;
31     private static final double ALFA_DESCARGA_CONTENEDOR = (MEDIA_DESCARGA_CONTENEDOR * MEDIA_DESCARGA_CONTENEDOR * 1.0) / VARIANZA_DESCARGA_CONTENEDOR;
32     private static final double BETA_DESCARGA_CONTENEDOR = VARIANZA_DESCARGA_CONTENEDOR / (MEDIA_DESCARGA_CONTENEDOR * 1.0);
```

Ilustración 16. Parte del código donde se definen los parámetros alfa y beta, el error y también la media que corresponde con el tiempo en segundo que dura la actividad

El cálculo de Alfa y Beta viene determinado por:

$$\text{(ALFA) } \alpha = \frac{(Media)^2}{varianza} \qquad \text{(BETA) } \beta = \frac{varianza}{Media}$$

Tanto el error como los tiempos promedio de cada actividad se dejaron como parámetros configurables por el usuario mediante la interfaz gráfica.

5. Interfaces gráficas

Para finalizar con nuestro modelo, llevamos a cabo la creación de diferentes ventanas y gráficos para una mejor visualización y una mejor interacción con el usuario. En la primera ventana podemos introducir diferentes parámetros que definirán nuestra simulación, luego se pasa a una ventana de resultados de tiempos, en la cual también podemos obtener dos graficas con distintos resultados de la simulación. Las interfaces gráficas se explicarán en detalle en el siguiente apartado.

Una vez determinado el modelo a simular, podemos plantearnos diferentes situaciones de simulación. Para ello hemos creado una primera ventana llamada “Datos de entrada” que sirve al usuario para introducir diferentes datos necesarios en el modelo y obtener diferentes resultados. Esta primera ventana, tiene un aspecto simple y fácil de usar para cualquier usuario.

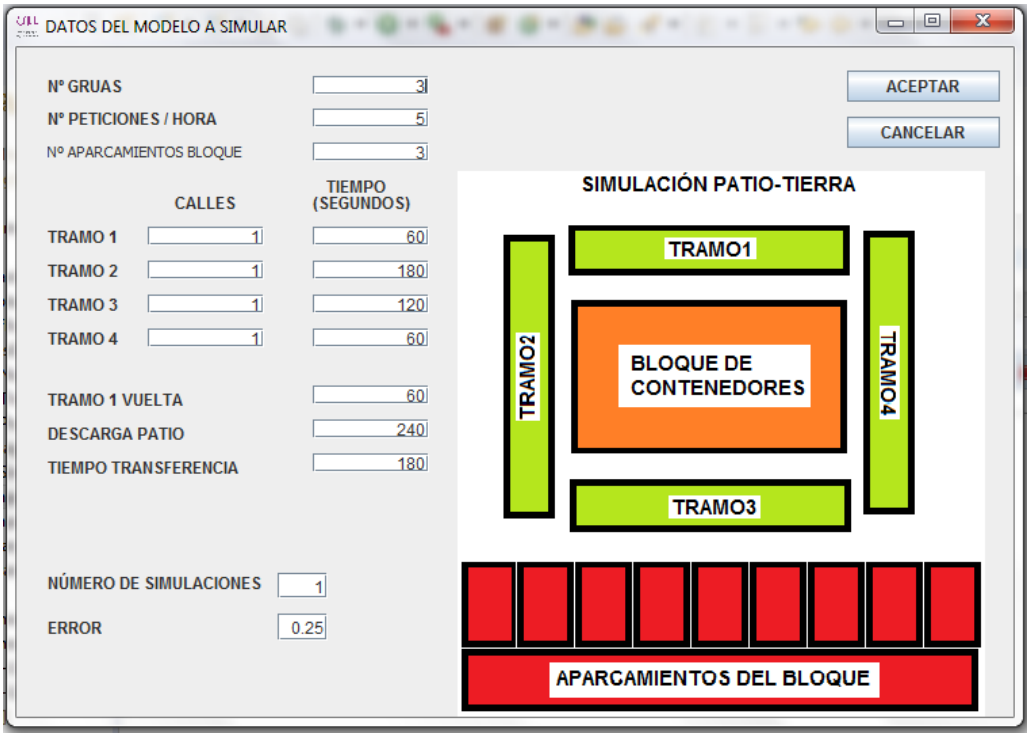

Ilustración 17. Ventada de datos de entrada.

En ella se introducen datos como: número de grúas, número de peticiones/hora, número de aparcamientos del bloque, número de calles de los tramos, tiempo (en segundos) que tardará en recorrer los tramos, realizar la descarga en el patio y realizar la transferencia, número de simulaciones a realizar y el error que se le define.

Después de introducir todos estos datos pulsamos el botón aceptar y comienza la simulación. Una vez acabada la simulación se abrirá otra ventana llamada “Tiempo de extracción de contenedores”.

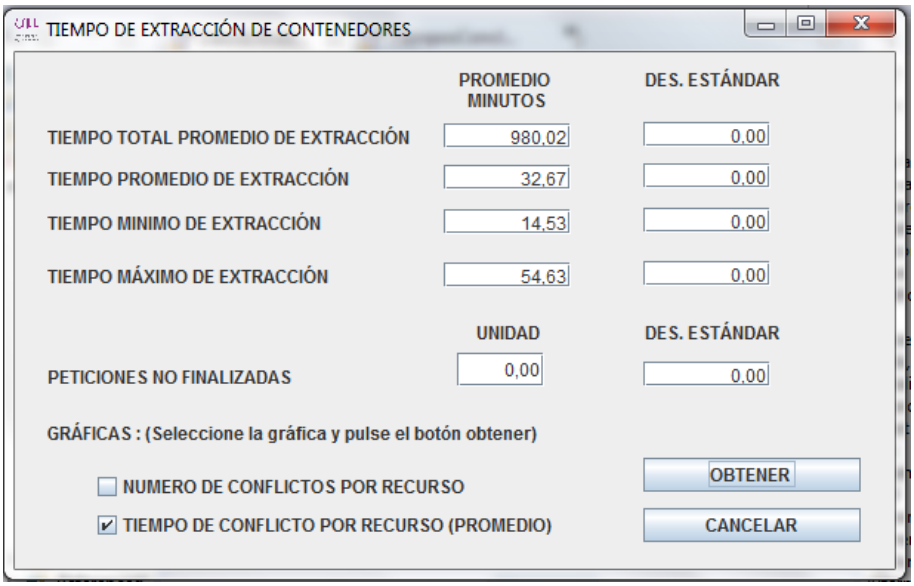

Ilustración 18. Ventana de tiempo de extracción de contenedores.

En esta ventana podemos observar distintos tiempos extraídos de la simulación, la desviación estándar para dar más fiabilidad a los tiempos, el número de peticiones no acabadas y también podemos obtener la gráfica con el número de conflictos por recurso y la gráfica de tiempo promedio de los conflictos. Para obtenerlas solo tenemos que seleccionar la gráfica requerida y pulsar el botón obtener.

Ilustración 19. Gráfica de numero de conflictos por recurso.

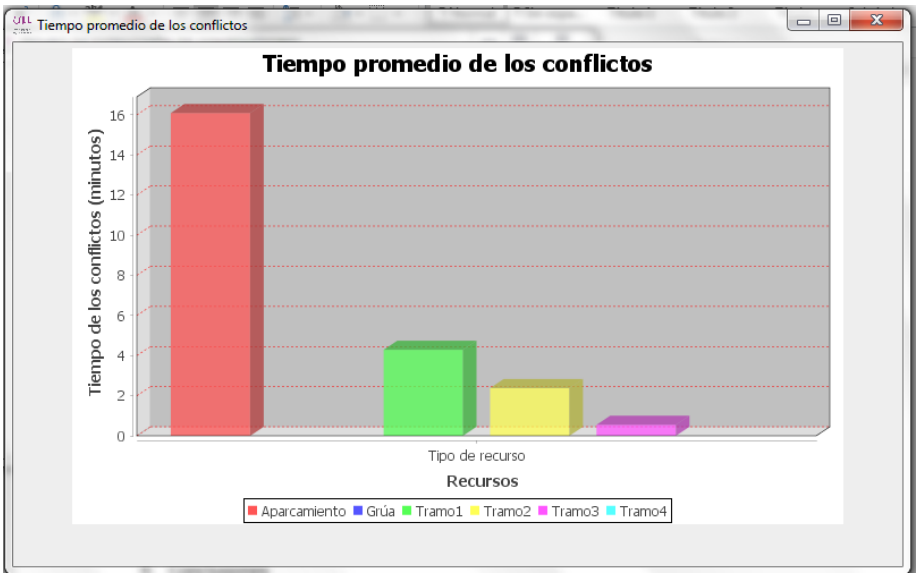

Ilustración 20. Gráfica de tiempo promedio de los conflictos.

6. Resultados

Para comenzar este apartado hemos pedido información mediante correo electrónico a la terminal de contenedores de Tenerife. Se solicitó el volumen de contenedores que se movían al año y cuantas grúas de patio tenían disponible.

El primer experimento va a partir de los datos dados para que los resultados sean lo más próximo a la realidad. Después haremos una segunda tanda de experimentos, con 4 escenarios modificando distintos parámetros, para ver como varían los tiempos, los conflictos por recurso y el tiempo promedio de los conflictos. También se calculará la desviación estándar para más fiabilidad de los datos.

Los datos proporcionados por la terminal de contenedores fueron:

- El número de movimientos de puerta anual es de unos 130.000 esto es entrega de contenedor a transporte terrestre. Haciendo las cuentas pertinentes nos sale que al día se realizan unos 90 movimientos de contenedores, por lo tanto, por hora serán unas 15 peticiones de recogida
- Para la recepción del contenedor en el bloque suelen utilizar 1 ó 2 grúas de patio o *reach stackers*.
- Del número de aparcamientos, en la zona de espera, no tenemos ningún dato disponible. Suponemos que sus dimensiones serán grandes y hemos definido que serán de 15 aparcamientos como mínimo.
- El tiempo que esperan los camiones por su contenedor como máximo será de 20 minutos, pudiendo variar ligeramente. Nosotros en nuestra simulación hemos predeterminado un tiempo de estancia de 18 minutos, pudiendo variar si el usuario lo cree oportuno antes de realizar alguna simulación. Este tiempo se ha impuesto al azar, dentro del tiempo dado por el puerto, ya que no sabemos ciertamente cuánto tarda la grúa en cada tramo, en la descarga o en la transferencia.

Con estos experimentos que se realizan a continuación se pretende ver la funcionalidad de la aplicación realizada para este proyecto. Y así, verificar de una forma visual y mediante datos de que esta herramienta puede ayudar a simular diferentes escenarios reales que se presentan en un puerto. Pudiendo ser una herramienta que ayude a

encontrar un modelo óptimo para que se realicen todas las peticiones de recogida de contendores al día.

- **Primer experimento** *(Datos proporcionados por el puerto)*

Parámetros	**Escenario 1**	**Escenario 2**
N.º de grúas	1	2
N.º de peticiones/hora	15	15
N.º aparcamientos bloque	15	15
N.º de simulaciones	100	100
Calles tramos	1	1
Error	0.25	0.25
***Resultados**	**Escenario 1**	**Escenario 2**
Tiempo total promedio de extracción (minutos)	15266.95 (113.54)	12109.90 (198.65)
Tiempo promedio de extracción (minutos)	169.63 (1.26)	134.55 (2.21)
Tiempo mínimo de extracción (minutos)	16.93 (1.80)	17.01 (1.94)
Tiempo máximo de extracción (minutos)	300 (0.00)	240 (0.00)
Peticiones no acabadas (unidad)	69.31 (0.61)	51 (0.92)

Tabla 1. Parámetros y resultados de la simulación (* Todos los resultados están expresados como "promedio (desviación típica)" de las simulaciones realizadas)

Como podemos observar, en la tabla 1, los resultados obtenidos con una y dos grúas muestran ciertas diferencias en cuanto a tiempos y peticiones no finalizadas. En tiempo se puede ver que con dos grúas se termina antes la actividad, pero en ambos casos se dejan muchas peticiones sin finalizar y esto no es un buen dato, ya que el objetivo es atender al día todas las peticiones. Otro dato que observar es el tiempo máximo de extracción de ambos experimentos, siendo de 5 horas para el escenario uno y 4 horas para el escenario dos. Esto es un dato negativo porque un camión no puede entrar a puerto y estar 4 o 5 horas esperando hasta tener su contenedor y salir del puerto. Provoca una cola de camiones al ocupar una zona de aparcamiento.

Las siguientes gráficas nos muestras el número de conflictos por recurso que se produjeron para cada escenario, uno y dos, respectivamente.

Ilustración 21. Escenario 1. Número de conflicto por recurso.

En el escenario uno, podemos ver que no hay conflictos en cuanto a los tramos por los que recorre la grúa, dato que ya esperábamos debido a que sólo hay una grúa para gestionar todas las peticiones. Si hay conflicto en cuanto a disponibilidad de aparcamiento y disponibilidad de grúa.

Los conflictos de aparcamiento están directamente relacionados con el número de grúas, provocando un efecto en cadena, ya que el número de *reach stackers* siempre será mucho menor que el número de aparcamiento y, por lo tanto, es imposible atender a todos los camiones formándose así largas colas para disponer de una grúa.

Los conflictos de grúa también se imaginaban porque haciendo cálculos experimentales se llegaba a la conclusión de que se necesitan 4 grúas para atender a todas las peticiones sin que hubiera conflictos a la hora de requerirlas.

Ilustración 22. Escenario 2. Número de conflictos por recurso.

En el escenario dos, al haber dos grúas ya se producen conflictos a la hora de requerir los tramos por los que circulan. También se ve que el tramo dos es donde más conflicto va a ver debido a que esa zona es donde se produce la descarga del contenedor y donde más tiempo va a permanecer la grúa.

Los conflictos de aparcamiento decrecen respecto al escenario uno, pero están en números muy similares y puede ser debido a lo mencionado anteriormente en el escenario uno.

Los conflictos de grúa en este escenario crecen. Lo más normal comparando los escenarios es que en este caso disminuyeran los conflictos de grúa debido a que hay dos grúas.

Ahora se expondrán dos gráficas, escenario uno y escenario dos, en las cuales se muestran el tiempo promedio de los conflictos.

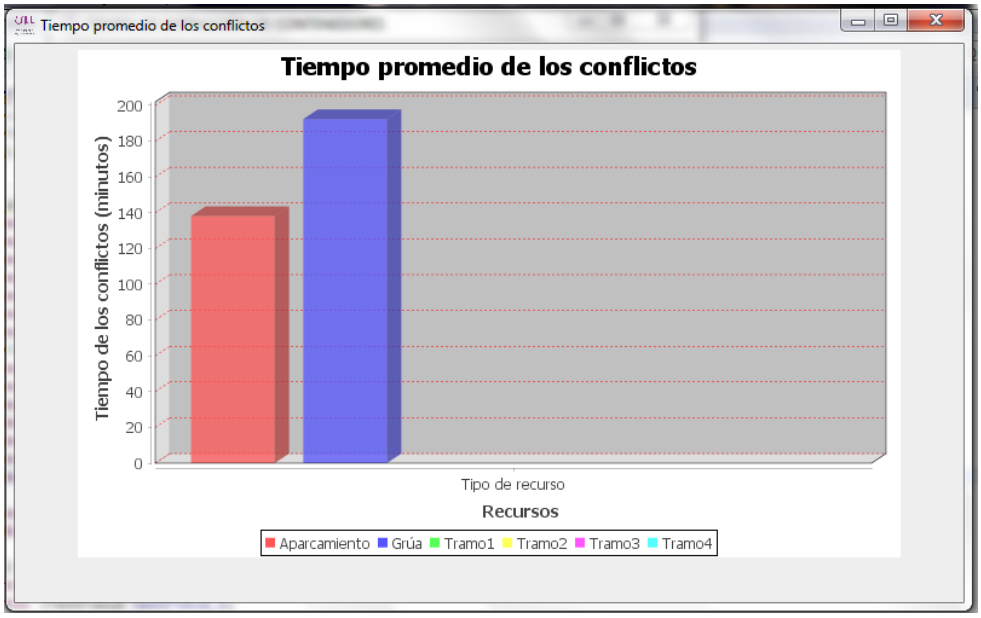

Ilustración 23. Escenario 1. Tiempo promedio de los conflictos.

Podemos ver que para el escenario uno, el tiempo promedio que se esperó para coger un aparcamiento se aproxima a 140 minutos, lo que equivale a más de 2 horas de espera. Mientras que el tiempo promedio para coger una grúa está entorno a los 190 minutos, lo que equivale a más de tres horas de espera.

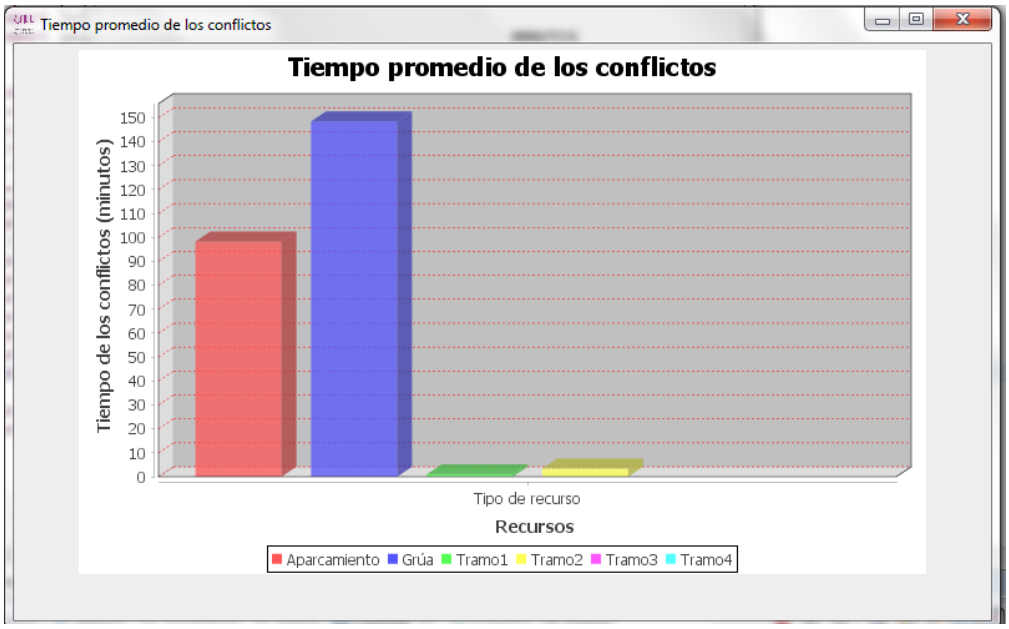

Ilustración 24. Escenario 2. Tiempo promedio de los conflictos.

En el escenario dos vemos que la espera por ocupar los tramos no supera los 5 minutos, lo cual es un tiempo bueno para que el proceso vaya de manera fluida. En este caso el tiempo promedio para ocupar aparcamiento se aproxima a 100 minutos y el tiempo promedio para coger grúa se aproxima a 150 minutos, ambos tiempos respecto al escenario uno baja su valor y esto es un dato para poder decir que el escenario dos ha ido más fluido que el escenario uno.

Se puede concluir que, con dos grúas, el proceso va de una manera fluida ya que para ocupar los recursos no se espera tanto tiempo y el número de peticiones atendidas es mayor que con una grúa.

- **Segundo experimento** *(Datos experimentales)*

Para este segundo experimento vamos a variar algunos datos para ver si el comportamiento mejora o simplemente se mantiene en la misma línea del primer experimento. El turno de trabajo y las peticiones que llegan por hora se mantendrán, al igual que los tiempos en los tramos, la descarga y la transferencia. El número de simulaciones también será de 100 para tener una muestra amplia.

En la tabla 2, que mostramos a continuación, expondremos 4 escenarios. Con estos escenarios buscamos ver cómo afecta la variación de los recursos en el modelo. En estos escenarios variaremos:

- N.º de grúas
- N.º de peticiones/hora
- N.º aparcamientos bloque
- N.º de calles de los tramos

Parámetro	Escenario 1	Escenario 2	Escenario 3	Escenario 4
N.º de grúas	2	3	3	4
N.º de peticiones/hora	15	15	15	15
N.º aparcamientos bloque	20	20	20	20
Calles tramo 1	1	1	2	2
Calles tramo 2	2	3	3	2
Calles tramo 3	2	2	3	2
Calles tramo 4	1	1	2	2
N.º de simulaciones	100	100	100	100
Error	0.25	0.25	0.25	0.25
Resultado*	**Escenario 1**	**Escenario 2**	**Escenario 3**	**Escenario 4**
Tiempo total para extracción (minutos)	11668.66 (151.24)	8232.67 (191.11)	8015.80 (191.40)	5261.63 (214.83)
Tiempo promedio de extracción (minutos)	129.65 (1.68)	91.47 (2.12)	89.06 (2.13)	58.46 (2.39)
Tiempo mínimo de extracción (minutos)	16.65 (1.57)	16.16 (1.54)	15.77 (1.36)	16.19 (1.32)
Tiempo máximo de extracción (minutos)	240.00 (0.00)	171.53 (5.51)	165.94 (5.83)	102.66 (5.13)
Peticiones no acabadas (unidad)	48.80 (0.88)	28.98 (1.04)	28.08 (1.05)	11.54 (1.13)

Tabla 2. Parámetros y resultados del segundo experimento (* Todos los resultados están expresados como "promedio (desviación típica)" de las simulaciones realizadas)

Como vemos en la tabla 2, según vayamos aumentado el número de recursos los tiempos de nuestra simulación irán disminuyendo. Esto tiene sentido ya que a más recursos menos tiempo se tarda en realizar las actividades. En comparación con el primer experimento hemos aumentado el número de aparcamientos en 5 unidades, pasando de 15 a 20 aparcamientos, por lo tanto, si llegara algún camión de más, de los 15 que se esperan por hora, tiene sitio donde ubicarse siempre y cuando no esté lleno. En un principio el aumento de capacidad de los aparcamientos no parece significativo, pero a medida que vamos aumentando el número de grúas los conflictos por aparcamiento van disminuyendo hasta ser menos de la mitad de las peticiones totales.

En el tramo dos y tres es donde más tiempo tardará la grúa, ya que es donde se hace la descarga del contenedor y la transferencia con el camión respectivamente. Si aumentamos el número de calles de estos tramos conseguimos evitar el efecto embudo y, por lo tanto, la actividad irá de una manera más rápida y se reduce también el número de peticiones no acabadas.

En cuanto el error siempre hemos hecho las simulaciones con el mismo error, simulando situaciones favorables para ver situaciones en las mejores condiciones disponibles. Este error podría irse modificando y ver qué es lo que pasa con situaciones donde hay mayor incertidumbre.

Ilustración 25. Escenario 1. Tabla 2. Número de conflictos por recurso.

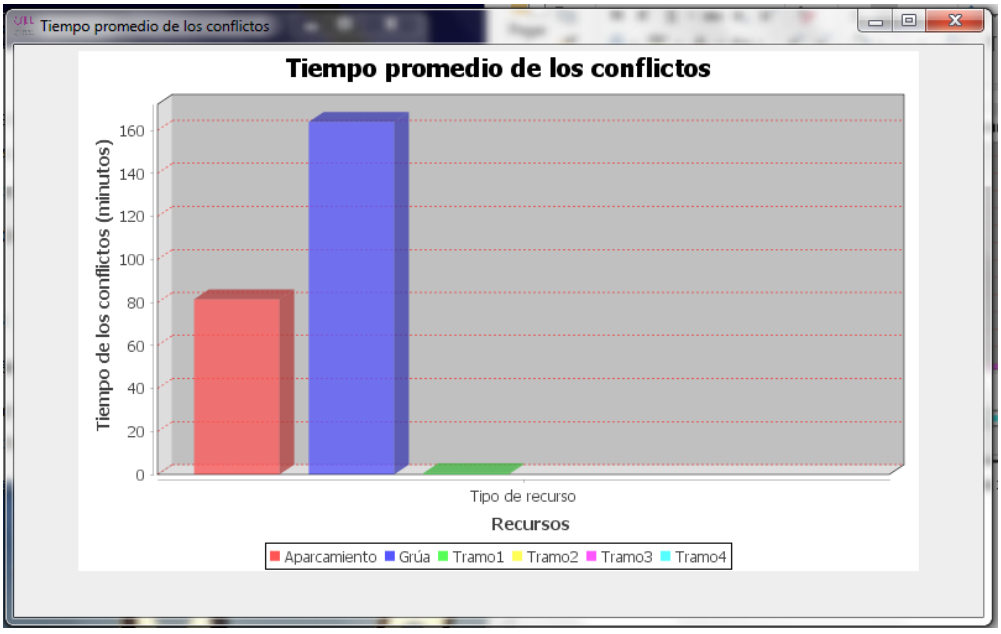

Ilustración 26. Escenario 1. Tabla 2. Tiempo promedio de los conflictos.

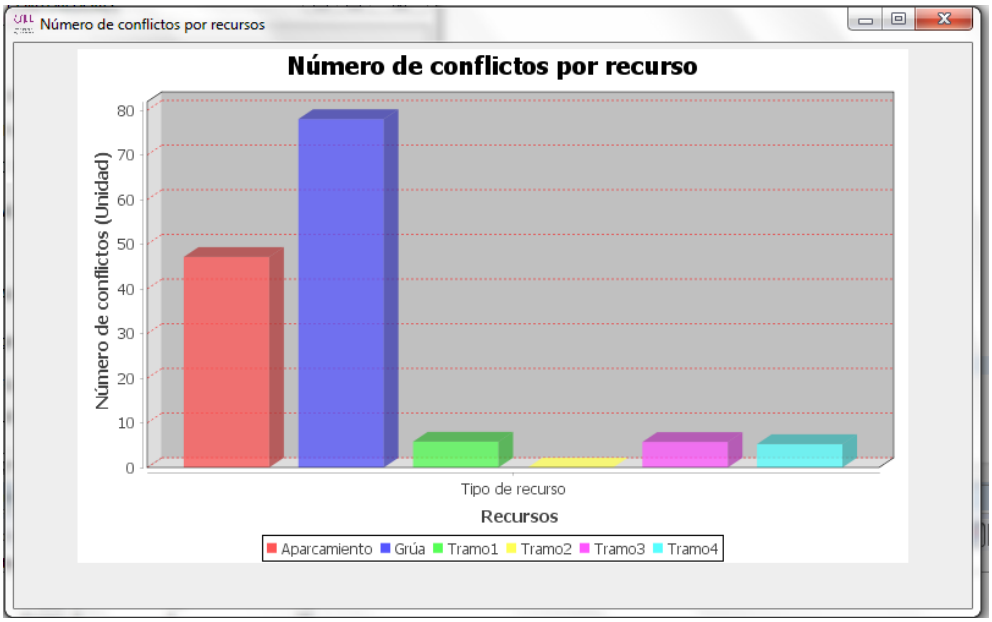

Ilustración 27. Escenario 2. Tabla 2. Número de conflictos por recurso.

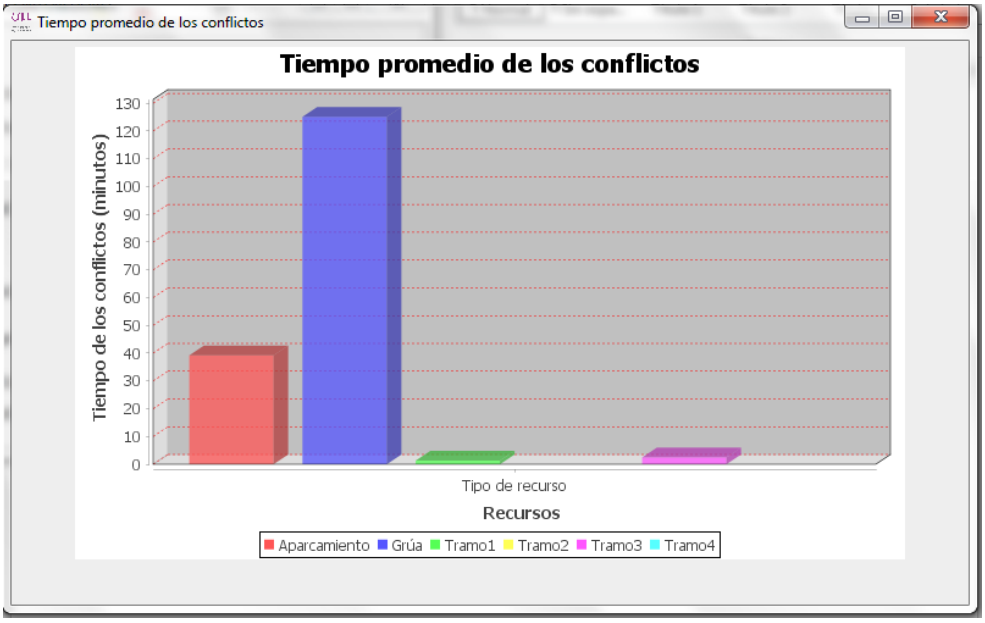

Ilustración 28. Escenario 2. Tabla 2. Tiempo promedio de los conflictos.

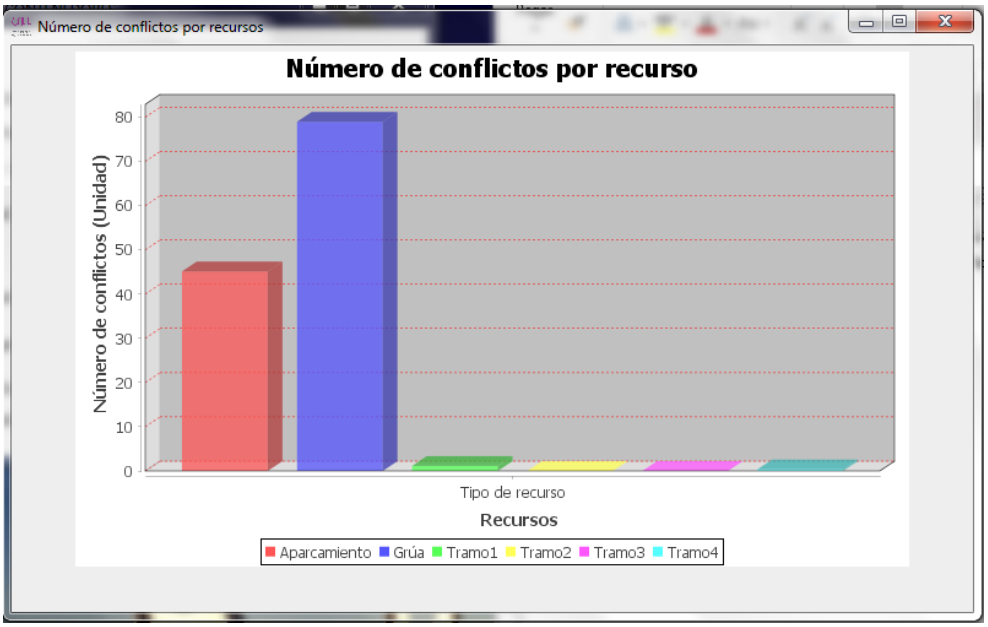

Ilustración 29. Escenario 3. Tabla 2. Número de conflictos por recurso.

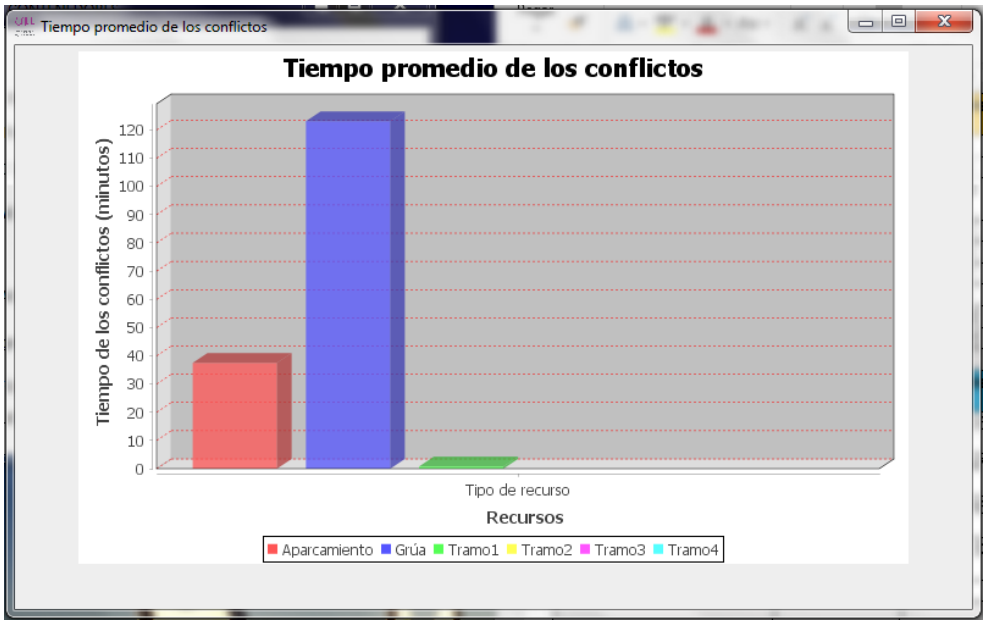

Ilustración 30. Escenario 3. Tabla 2. Tiempo promedio de los conflictos.

Ilustración 31. Escenario 4. Tabla 2. Numero de conflictos por recurso.

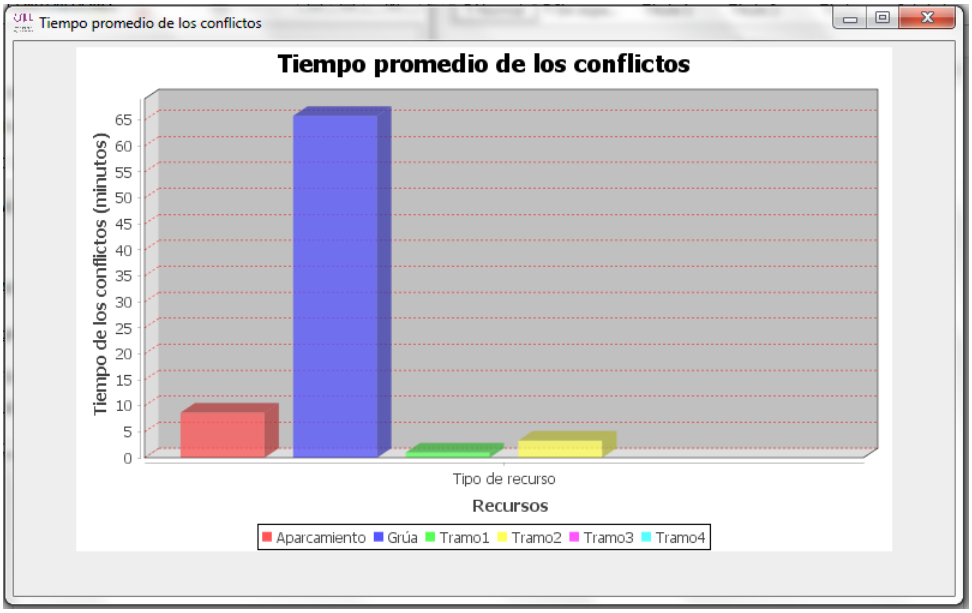

Ilustración 32. Escenario 4. Tabla 2. Tiempo promedio de los conflictos.

Como conclusión a estos escenarios expuestos podemos comentar que siempre en mayor o menor cantidad tendremos conflictos en cuanto a aparcamientos y grúas, ya que estos dos recursos son los fundamentales en la simulación. La consecuencia negativa directa de estos dos conflictos se ve reflejada en el número de peticiones que no se acaban al día. Los conflictos en los tramos aumentarán en cuanto haya más grúas que calles de los tramos. Una solución para esto sería poner el mismo número de grúa que de calles en los tramos y así se evitarían muchos conflictos por tramos. Esta solución no es del todo válida porque dependemos de la capacidad que tenga la terminal de contenedores y del volumen que ocupen los bloques de contenedores.

También podemos ver que en cuanto se van aumentando los valores de los recursos los tiempos van disminuyendo y el número de peticiones no acabadas se va reduciendo. Esto quiere decir que se llegará a una combinación de recursos que harán que salgan

todas nuestras peticiones en el turno de trabajo. Destacar que si se aumenta el número de calles se podría aumentar en número de grúas *transtainer*, que en nuestro caso esta simulada mediante un tiempo de descarga, y esto ayudaría a sacar más contenedores hacia sus camiones, con lo cual es proceso iría de una forma más fluida.

Para ver la importancia de la grúa de patio o *reach stacker* debemos fijarnos en el tiempo promedio que se tarda en esperar por este recurso. Vemos que siempre son tiempos muy elevados y los cuales no se pueden permitir, ya que interesa que la actividad se realice lo más rápido posible y sin atascos, que sería lo ideal.

Los conflictos de grúa no van a disminuir ya que son 15 peticiones/hora por un número muy reducido de grúas que atienden las peticiones. Por este motivo, siempre vamos a tener un número muy elevado de conflictos en cuanto a grúas de patio.

Por último, podemos decir que se podría seguir realizando simulaciones para ver cómo influye el número y tipo de recursos en el funcionamiento del sistema. no obstante, siempre hay que ver la capacidad que tenemos para instalar los recursos y hacer una planificación muy estricta que deben cumplir todos los recursos para que el modelo simulado pueda cumplirse en la realidad.

7. Conclusiones

En este trabajo se ha entendido el funcionamiento de un puerto, en especial, la zona patio-tierra de una terminal de contenedores, conociendo las actividades que se llevan a cabo para la sustracción de un contendor del puerto.

Con la elaboración de este Trabajo Fin de Grado se ha conseguido programar en lenguaje de programación Java, el cual era un reto ya que nunca se había visto este tipo de lenguaje de programación. Además, se ha comprendido que la programación puede solucionar de forma rápida muchos problemas con solo un par de líneas de código.

También se ha logrado realizar un modelo de simulación básico, con el cual llevar a cabo la idea principal de este proyecto que era el modelado y simulación de la zona patio-tierra de un puerto. Esta herramienta de simulación realizada es capaz de simular las actividades de un bloque de contenedores, pudiéndose ampliar a más bloques de contenedores. Esta idea podría valer para ayudar a planificar la actividad de los recursos en la zona patio-tierra. Para la obtención del modelo se ha aprendido a utilizar la herramienta PSIGHOS, la cual me ha ayudado y facilitado el trabajo a la hora de modelar y simular el proyecto propuesto. Cabe decir, que este proyecto queda abierto a posibles investigaciones y modificaciones, con la idea de que algún día pueda plantearse como un prototipo real para facilitar las actividades de un puerto.

Otro objetivo realizado y concluyente de este proyecto, fue la realización de varias interfaces gráficas, que constituyen el prototipo de una herramienta de ayuda a la toma de decisiones en las cuales el usuario puede introducir diferentes datos y obtener resultados de forma inmediata mediante gráficas o cuadro de texto.

Por último, se ha comprendido que tanto el modelado y la simulación como la programación son unas herramientas muy útiles y eficaces dentro de la ingeniería, ya que pueden ayudar a solucionar muchos problemas reales de una forma casi instantánea.

7.1. Conclusions

In this work I have understood the operation of a port, especially the yard-land area of a container terminal, by knowing the activities that are carried out for the removal of a container from the port.

With the elaboration of this Final Degree Project it has been possible to program in Java programming language, which was a challenge since I had never seen this type of programming language before. Moreover, I have understood that programming can quickly solve many problems with just a couple of lines of code.

It has also been possible to create a basic simulation model, with which to carry out the main idea of this project, which was the modeling and simulation of the yard-land area of a port. This simulation tool can simulate the activities of a block of containers, being able to expand to more container blocks. This idea could be used to help plan the activity of the resources in the yard-land area. I have learned to use the PSIGHOS tool to obtain the model, which helped me and facilitated the work when modeling and simulating the proposed project. It should be said that this project is open to possible research and modifications, with the idea that one day it could be considered as a real prototype to facilitate the activities of a port.

Another objective realized and conclusive of this project, was the accomplishment of several graphical interfaces, intended as a prototype of a decision-aid tool, in which the user can introduce different data and obtain results of immediate form by means of graphs or picture of text.

Finally, I have understood that both modeling and simulation, and programming are very useful and effective tools within engineering, since they can help solve many real problems almost instantaneously.

8. Bibliografía

[1]. https://definicion.de/puerto-maritimo/

[2]. https://es.wikipedia.org/wiki/Simulaci%C3%B3n

[3]. https://es.wikipedia.org/wiki/Simulaci%C3%B3n

[4]. http://cursos.aiu.edu/Simulacion%20de%20Eventos/PDF/Tema%201.pdf

[5]. https://upcommons.upc.edu/bitstream/handle/2099.1/6271/05.pdf

[6]. https://es.wikipedia.org/wiki/Contenedor

[7]. http://www.scielo.org.co/pdf/prosp/v12n1/v12n1a12.pdf

[8].http://www.tctenerife.es/Plantillas/template2.aspx?IdA=1&IdF=178&idM=275&nvl=1&lan=es

9. Anexos

9.1. Anexo I. Código del proyecto

9.1.1. IdeaPort1Main.java

```
package es.ull.iis.simulation.portYardEarth;

import es.ull.iis.simulation.inforeceiver.StdInfoView;
import es.ull.iis.simulation.model.Experiment;
import es.ull.iis.simulation.model.TimeUnit;
import es.ull.iis.simulation.model.Simulation;

public class IdeaPort1Main extends Experiment {
	final private static String DESCRIPTION = "Simulación zona patio-
tierra del puerto de Santa Cruz de Tenerife";
	private static final TimeUnit PORT_TIME_UNIT = TimeUnit.SECOND;
	private static final long START_TS = 0;
	private static final long END_TS = 6*60*60;//segundos
	private int nGruas;
	private int nCallesA;
	private int nCamiones;
	private int nCallesInt1;
	private int nCallesInt2;
	private int nCallesInt3;
	private int nCallesInt4;
	private long tramo1;
	private long tramo2;
	private long tramo3;
	private long tramo4;
	private long descarga;
	private long transferencia;
	private long tramo1Vuelta;

	public IdeaPort1Main(int nGruas, int nCallesA,int nCamiones,int
nCallesInt1,int nCallesInt2, int nCallesInt3, int nCallesInt4, long tramo1,
 long tramo2,long tramo3, long tramo4, long tramo1Vuelta, long descarga,
 long transferencia, int nSim, double error){
		super("Prueba PATIO-TIERRA", nSim);
		this.nGruas = nGruas;
		this.nCallesA = nCallesA;
		this.nCamiones = nCamiones;
		this.nCallesInt1 = nCallesInt1;
		this.nCallesInt2 = nCallesInt2;
		this.nCallesInt3 = nCallesInt3;
		this.nCallesInt4 = nCallesInt4;
		this.tramo1 = tramo1;
		this.tramo2 = tramo2;
		this.tramo3 = tramo3;
		this.tramo4 = tramo4;
		this.tramo1Vuelta = tramo1Vuelta;
		this.descarga = descarga;
		this.transferencia = transferencia;
		PortSimulation.setERROR(error);
		TiempoEstanciaListener.setnSim(nSim);
	}

	public Simulation getSimulation(int ind){

//Creamos la simulacion pasandole todos los parametros
```

```
		Simulation simul = new PortSimulation(ind, DESCRIPTION + ind,
PORT_TIME_UNIT, START_TS, END_TS, nGruas,nCamiones, nCallesA,nCallesInt1,
nCallesInt2,nCallesInt3,nCallesInt4,tramo1,tramo2,tramo3, tramo4,
tramo1Vuelta, descarga,transferencia);
		//Le pasamos los listener con los que obtenemos los resultados
			simul.addInfoReceiver(new StdInfoView());
			simul.addInfoReceiver(new TiempoEstanciaListener());
			simul.addInfoReceiver(new ConflictoListener(nExperiments));

		return simul;
		}
}
```

9.1.2. PortSimulation.java

```
package es.ull.iis.simulation.portYardEarth;

import es.ull.iis.simulation.model.TimeUnit;
import es.ull.iis.simulation.model.flow.DelayFlow;
import es.ull.iis.simulation.model.flow.ReleaseResourcesFlow;
import es.ull.iis.simulation.model.flow.RequestResourcesFlow;
import es.ull.iis.simulation.model.Resource;
import es.ull.iis.simulation.model.WorkGroup;
import es.ull.iis.function.TimeFunctionFactory;
import es.ull.iis.simulation.model.ElementType;
import es.ull.iis.simulation.model.ResourceType;
import es.ull.iis.simulation.model.Simulation;
import es.ull.iis.simulation.model.SimulationPeriodicCycle;
import es.ull.iis.simulation.model.TimeDrivenElementGenerator;


public class PortSimulation extends Simulation {

// Se definen los tiempos de todas las actividades
private static double ERROR = 0.25;
private static final long MEDIA_DESCARGA_CONTENEDOR = 5*60;
private static final double VARIANZA_DESCARGA_CONTENEDOR = ERROR *
MEDIA_DESCARGA_CONTENEDOR * ERROR * MEDIA_DESCARGA_CONTENEDOR;
private static final double ALFA_DESCARGA_CONTENEDOR =
(MEDIA_DESCARGA_CONTENEDOR * MEDIA_DESCARGA_CONTENEDOR * 1.0) /
VARIANZA_DESCARGA_CONTENEDOR;
private static final double BETA_DESCARGA_CONTENEDOR =
VARIANZA_DESCARGA_CONTENEDOR / (MEDIA_DESCARGA_CONTENEDOR * 1.0);

private static final long MEDIA_TRANSFERENCIA = 3*60;
private static final double VARIANZA_TRANSFERENCIA = ERROR *
MEDIA_TRANSFERENCIA * ERROR * MEDIA_TRANSFERENCIA;
private static final double ALFA_TRANSFERENCIA = (MEDIA_TRANSFERENCIA *
MEDIA_TRANSFERENCIA * 1.0) / VARIANZA_TRANSFERENCIA;
private static final double BETA_TRANSFERENCIA = VARIANZA_TRANSFERENCIA /
(MEDIA_TRANSFERENCIA * 1.0);

private static final long MEDIA_TRAMO_PATIO = 1*60; //Tramo1
private static final double VARIANZA_TRAMO_PATIO = ERROR * MEDIA_TRAMO_PATIO
* ERROR * MEDIA_TRAMO_PATIO;
private static final double ALFA_TRAMO_PATIO = (MEDIA_TRAMO_PATIO *
MEDIA_TRAMO_PATIO * 1.0) / VARIANZA_TRAMO_PATIO;
private static final double BETA_TRAMO_PATIO = VARIANZA_TRAMO_PATIO /
(MEDIA_TRAMO_PATIO * 1.0);
```

```
   private static final long MEDIA_TRAMO_CENTRO = 3*60; //Tramo2
   private static final double VARIANZA_TRAMO_CENTRO = ERROR *
   MEDIA_TRAMO_CENTRO * ERROR * MEDIA_TRAMO_CENTRO;
   private static final double ALFA_TRAMO_CENTRO = (MEDIA_TRAMO_CENTRO *
   MEDIA_TRAMO_CENTRO * 1.0) / VARIANZA_TRAMO_CENTRO;
   private static final double BETA_TRAMO_CENTRO = VARIANZA_TRAMO_CENTRO /
   (MEDIA_TRAMO_CENTRO * 1.0);

   private static final long MEDIA_TRAMO_TIERRA = 3*60; //Tramo3
   private static final double VARIANZA_TRAMO_TIERRA = ERROR *
   MEDIA_TRAMO_TIERRA * ERROR * MEDIA_TRAMO_TIERRA;
   private static final double ALFA_TRAMO_TIERRA = (MEDIA_TRAMO_TIERRA *
   MEDIA_TRAMO_TIERRA * 1.0) / VARIANZA_TRAMO_TIERRA;
   private static final double BETA_TRAMO_TIERRA = VARIANZA_TRAMO_TIERRA /
   (MEDIA_TRAMO_TIERRA * 1.0);

   private static final long MEDIA_CENTRO_VUELTA = 1*60; //Tramo4
   private static final double VARIANZA_CENTRO_VUELTA = ERROR *
   MEDIA_CENTRO_VUELTA * ERROR * MEDIA_CENTRO_VUELTA;
   private static final double ALFA_CENTRO_VUELTA = (MEDIA_CENTRO_VUELTA *
   MEDIA_CENTRO_VUELTA * 1.0) / VARIANZA_CENTRO_VUELTA;
   private static final double BETA_CENTRO_VUELTA = VARIANZA_CENTRO_VUELTA /
   (MEDIA_CENTRO_VUELTA * 1.0);

   private static final long MEDIA_VUELTA_PATIO1 = 1*60; //Tramo1Vuelta
   private static final double VARIANZA_VUELTA_PATIO1 = ERROR *
   MEDIA_VUELTA_PATIO1 * ERROR * MEDIA_VUELTA_PATIO1;
   private static final double ALFA_VUELTA_PATIO1 = (MEDIA_VUELTA_PATIO1*
   MEDIA_VUELTA_PATIO1 * 1.0) / VARIANZA_VUELTA_PATIO1;
   private static final double BETA_VUELTA_PATIO1 = VARIANZA_VUELTA_PATIO1 /
   (MEDIA_VUELTA_PATIO1* 1.0);


   protected static final String PETICION_A = "Peticion A";
   protected static final String LLEGADA = "LLegada de camion, coger
aparcamiento";
   protected static final String FINAL = "Final. Dejar libre aparcamiento";
   protected static final String GRUAIN = "Coger grúa";
   protected static final String GRUAOFF = "Fin de grúa";
   protected static final String ON_TRAMO1 = "Coger tramo 1 del patio";
   protected static final String OFF_TRAMO1 = "Soltar tramo 1 del patio";
   protected static final String ON_TRAMO2 = "Coger tramo 2 del patio";
   protected static final String OFF_TRAMO2 = "Soltar tramo 2 del patio";
   protected static final String ON_TRAMO3 = "Coger tramo 3 del patio";
   protected static final String OFF_TRAMO3 = "Soltar tramo 3 del patio";
   protected static final String ON_TRAMO4 = "Coger tramo 4 del patio";
   protected static final String OFF_TRAMO4 = "Soltar tramo 4 del patio";
   protected static final String ON_TRAMO1VUELTA = "Retorno por tramo 1 del
patio";
   protected static final String OFF_TRAMO1VUELTA = "Retorno para soltar tramo 1
del patio";
   protected static final String DESCARGA = "Localizacion y descarga del
contenedor en el patio";
   private static final String TRANSFERENCIA = "Transferencia del contenedor
grúa-camión";

         public PortSimulation(int id, String description, TimeUnit unit, long
   startTs, long endTs, int nGruas, int nCallesA,int nCamiones,int nCallesInt1,
   int nCallesInt2, int nCallesInt3, int nCallesInt4, long tramo1, long tramo2,
```

```
long tramo3, long tramo4, long tramo1Vuelta, long descarga, long
transferencia) {
      super(id, "Simulación Puerto Patio-Tierra", unit, startTs, endTs);

//TIPOS DE ELEMENTOS DE NUESTRO PROGRAMA
//SIRVE PARA AGRUPAR A TODOS LOS ELEMENTOS DE UN TIPO CONCRETO

      ElementType Peticion = new ElementType(this, PETICION_A);

// TIPOS DE RECURSOS DE NUESTRO PROGRAMA
ResourceType rtGrua = new ResourceType(this, "Grúa");
ResourceType rtParkingA = new ResourceType(this, "Aparcamiento Bloque A");
ResourceType rtTramoA1 = new ResourceType(this, "Tramo A1");
ResourceType rtTramoA2 = new ResourceType(this, "Tramo A2");
ResourceType rtTramoA3 = new ResourceType(this, "Tramo A3");
ResourceType rtTramoA4 = new ResourceType(this, "Tramo A4");

// RECURSO DE NUESTRO PROGRAMA
// SE DEFINEN A TRAVES DE LOS TIPOS DE RECURSOS
// CREAMOS LOS RECURSOS ESPECIFICOS
    Resource[] resGruas = rtGrua.addGenericResources(nGruas);
    Resource[] resParkingA = rtParkingA.addGenericResources(nCallesA);
    Resource[] resTramoA1 = rtTramoA1.addGenericResources(nCallesInt1);
    Resource[] resTramoA2 = rtTramoA2.addGenericResources(nCallesInt2);
    Resource[] resTramoA3 = rtTramoA3.addGenericResources(nCallesInt3);
    Resource[] resTramoA4 = rtTramoA4.addGenericResources(nCallesInt4);

//CON ESTO DEFINIMOS LOS CAMIONES QUE LLEGARAN POR HORA
SimulationPeriodicCycle CamionCycle =
SimulationPeriodicCycle.newHourlyCycle(unit);

// WORKGROUP DE NUESTRO PROGRAMA
// REPRESENTAN EL NUMERO Y TIPO DE RECURSOS QUE NECESITO
// PARA REALIZAR UNA TAREA
        WorkGroup wgGrua = new WorkGroup(this, rtGrua, 1);
        WorkGroup wgParkingA = new WorkGroup(this, rtParkingA, 1);
        WorkGroup wgTramoA1 = new WorkGroup(this, rtTramoA1, 1);
        WorkGroup wgTramoA2 = new WorkGroup(this, rtTramoA2, 1);
        WorkGroup wgTramoA3 = new WorkGroup(this, rtTramoA3, 1);
        WorkGroup wgTramoA4 = new WorkGroup(this, rtTramoA4, 1);
        WorkGroup wgVacio = new WorkGroup(this);

// DEFINIMOS TODAS LAS ACCIONES QUE LLEVARA A CABO NUESTRO PROCESO
// TODO ELLO ESTA DEFINIDO EN PAPEL MEDIANTE UN ESQUEMA
RequestResourcesFlow reqCamion = new RequestResourcesFlow(this, LLEGADA);
            reqCamion.addWorkGroup(wgParkingA);
ReleaseResourcesFlow relCamion = new ReleaseResourcesFlow(this, FINAL,
wgParkingA);

RequestResourcesFlow reqGrua = new RequestResourcesFlow(this, GRUAIN);
            reqGrua.addWorkGroup(wgGrua);
ReleaseResourcesFlow relGrua = new ReleaseResourcesFlow(this,GRUAOFF,wgGrua);

RequestResourcesFlow reqTramoPatio1 = new RequestResourcesFlow(this,
ON_TRAMO1);
            reqTramoPatio1.addWorkGroup(wgTramoA1);
ReleaseResourcesFlow relTramoPatio1 = new
ReleaseResourcesFlow(this,OFF_TRAMO1,wgTramoA1);
```

```
RequestResourcesFlow reqTramoCentro1 = new RequestResourcesFlow(this,
ON_TRAMO2);
            reqTramoCentro1.addWorkGroup(wgTramoA2);
ReleaseResourcesFlow relTramoCentro1 = new
ReleaseResourcesFlow(this,OFF_TRAMO2,wgTramoA2);

RequestResourcesFlow reqtramotierra1 = new RequestResourcesFlow(this,
ON_TRAMO3);
            reqtramotierra1.addWorkGroup(wgTramoA3);
ReleaseResourcesFlow reltramotierra1 = new
ReleaseResourcesFlow(this,OFF_TRAMO3,wgTramoA3);

RequestResourcesFlow reqtramocentro2 = new RequestResourcesFlow(this,
ON_TRAMO4);
            reqtramocentro2.addWorkGroup(wgTramoA4);
ReleaseResourcesFlow reltramocentro2 = new
ReleaseResourcesFlow(this,OFF_TRAMO4,wgTramoA4);

RequestResourcesFlow reqtramopatio1vuelta = new RequestResourcesFlow(this,
ON_TRAMO1VUELTA);
            reqtramopatio1vuelta.addWorkGroup(wgVacio);
ReleaseResourcesFlow reltramopatio1vuelta = new
ReleaseResourcesFlow(this,OFF_TRAMO1VUELTA,wgVacio);

// SE DEFINEN LOS TIEMPOS QUE TARDARA EL PROCESO EN PASAR POR LOS TRAMOS DEL
PATIO Y EN REALIZAR
// LAS ACTIVIDADES DE LOCALIZACION,DESCARGA, TRANSFERENCIA, ETC
DelayFlow treqPatio1 = new DelayFlow(this, "Tiempo Tramo 1",
TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_PATIO,
BETA_TRAMO_PATIO));
DelayFlow treqCentro1 = new DelayFlow(this, "Tiempo Tramo 2",
TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_CENTRO,
BETA_TRAMO_CENTRO));
DelayFlow tDescargar = new DelayFlow(this, DESCARGA,
TimeFunctionFactory.getInstance("GammaVariate", ALFA_DESCARGA_CONTENEDOR,
BETA_DESCARGA_CONTENEDOR));
DelayFlow treqTierra1 = new DelayFlow(this, "Tiempo Tramo 3",
TimeFunctionFactory.getInstance("GammaVariate", ALFA_TRAMO_TIERRA,
BETA_TRAMO_TIERRA));
DelayFlow tTransferencia = new
DelayFlow(this,TRANSFERENCIA,TimeFunctionFactory.getInstance("GammaVariate",
ALFA_TRANSFERENCIA, BETA_TRANSFERENCIA ));
DelayFlow treqCentro2 = new DelayFlow(this,"Tiempo Tramo 4",
TimeFunctionFactory.getInstance("GammaVariate", ALFA_CENTRO_VUELTA,
BETA_CENTRO_VUELTA));
DelayFlow tTramopatio1vuelta = new DelayFlow(this, "Tiempo Tramo 1 vuelta",
TimeFunctionFactory.getInstance("GammaVariate", ALFA_VUELTA_PATIO1,
BETA_VUELTA_PATIO1));


//ACTIVIDADES LINKEADAS DE NUESTRO PROCESO. AQUI PODEMOS VER COMO FLUIRA
NUESTRO PROCESO
// Primero enlazo las acciones desde que se inicia el pedido hasta que la
grúa llega al tramo central
      reqCamion.link(reqGrua).link(reqTramoPatio1).link(treqPatio1).link(req
      TramoCentro1).link(relTramoPatio1).link(treqCentro1);
// Después enlazo con la descarga
            treqCentro1.link(tDescargar);
```

```
// Ahora enlazo las acciones desde que se descarga hasta que la grúa llega a
donde está el camión
		tDescargar.link(reqtramotierra1).link(relTramoCentro1).link(treqTierra
1);
// Ahora enlazo desde que se realiza la tranferencia hasta que empieza la
vuelta de la grua hasta el tramo central de vuelta
		treqTierra1.link(tTransferencia).link(reqtramocentro2).link(reltramoti
		erra1).link(treqCentro2);
// Se enlazan las actividades que van desde el tramo central de vuelta hasta
la zona de parking de las gruas
		treqCentro2.link(reqtramopatio1vuelta).link(reltramocentro2).link(tTra
		mopatio1vuelta).link(reltramopatio1vuelta).link(relGrua).link(relCamio
		n);
//GENERADOR DE ELEMENTOS
TimeDrivenElementGenerator gen = new TimeDrivenElementGenerator(this,
nCamiones, Peticion, reqCamion, CamionCycle);

		}
		public static double getERROR() {
			return ERROR;
		}
		public static void setERROR(double eRROR) {
			ERROR = eRROR;
		}

}
```

9.1.3. TiempoEstanciaListener.java

```
/**
 *
 */
package es.ull.iis.simulation.portYardEarth;

import java.util.TreeMap;
import es.ull.iis.simulation.info.ElementInfo;
import es.ull.iis.simulation.info.SimulationEndInfo;
import es.ull.iis.simulation.info.SimulationInfo;
import es.ull.iis.simulation.inforeceiver.View;
import es.ull.iis.simulation.model.Element;
import es.ull.iis.util.Statistics;
/**
 * @author Daniel
 *
 */
public class TiempoEstanciaListener extends View {
		protected static int nSim = 1;
		protected static double[] maxGlobal;
		protected static double[] minGlobal;
		protected static double[] promedioGlobal;
		protected static double[] sumaGlobal;
		protected static double contadorGlobal[];
		protected static TreeMap<Element, Long> tEstancia;
		//Para recoger los datos internos de la clase, son propios de la clase
		private double max = 0.0;
		private Long min = Long.MAX_VALUE;
		private double promedio = 0.0;
		private double suma = 0.0;
		private Long contador = (long) 0;
```

```
	public TiempoEstanciaListener() {
		super("");
		tEstancia = new TreeMap<Element, Long>();
		addEntrance(ElementInfo.class);
		addEntrance(SimulationEndInfo.class);
	}
	public void infoEmited(SimulationInfo info) {
		if (info instanceof ElementInfo) {
			ElementInfo eInfo = (ElementInfo)info;
			switch(eInfo.getType()) {
			case FINISH:
				long tStart = tEstancia.get(eInfo.getElement());
				tEstancia.put(eInfo.getElement(), eInfo.getTs() +
tStart);
				eInfo.getTs();
				break;
			case START:
				tEstancia.put(eInfo.getElement(), -eInfo.getTs());
				eInfo.getTs();
				break;
			default:
				break;
			}
		}
		else if (info instanceof SimulationEndInfo) {

			System.out.println("\tPetición\tNo acabadas\t");
		long endTs = ((SimulationEndInfo) info).getTs();
			for(Element elem : tEstancia.keySet()){
				if(tEstancia.get(elem) < 0){
					if(-tEstancia.get(elem) < endTs){
						tEstancia.put(elem, endTs +
tEstancia.get(elem));
						System.out.println("\t"+ elem +
"\t\tNo acabó");
						contador = contador + 1;
					}
					else{
						tEstancia.remove(elem);
				}
			}
		}
			System.out.println("\tPetición\tTiempo extracción");
			for (Element elem : tEstancia.keySet()) {
				suma = (double)suma + tEstancia.get(elem);
				if(tEstancia.get(elem) > max){
					max = tEstancia.get(elem);

				}
				if(tEstancia.get(elem) < min){
					min = tEstancia.get(elem);
				}

				else {
					System.out.println( "\t"+ elem + "\t\t " +
tEstancia.get(elem));
				}
			}
```

```
                    promedio = (double)suma / (double)tEstancia.size();

                    System.out.println("El tiempo total de extracción será: "
+ suma/60);
                    System.out.println("El tiempo promedio de extracción será:
" + promedio/60);
                    System.out.println("El tiempo mínimo de extracción será: "
+ min/60);
                    System.out.println("El tiempo máximo de extracción será: "
+ max/60);
                    System.out.println("No acabaron: " + contador);

                    sumaGlobal [info.getSimul().getIdentifier()] = suma;
                    promedioGlobal[info.getSimul().getIdentifier()] =
promedio;
                    minGlobal [info.getSimul().getIdentifier()] = min;
                    maxGlobal [info.getSimul().getIdentifier()] = max;
             contadorGlobal [info.getSimul().getIdentifier()] = contador;

             }
      }
      public static int getnSim() {
             return nSim;
      }
      public static void setnSim(int nSim) {
             TiempoEstanciaListener.nSim = nSim;
             promedioGlobal = new double[nSim];
             minGlobal = new double[nSim];
             maxGlobal = new double[nSim];
             sumaGlobal = new double[nSim];
             contadorGlobal = new double[nSim];
      }

      public static double getPromedioGlobal() {
             return Statistics.average(promedioGlobal);
      }

      public static double getSDPromedioGlobal() {
             return Statistics.stdDev(promedioGlobal);
      }
      public static double getMinGlobal() {
             return Statistics.average(minGlobal);
      }

      public static double getSDMinGlobal() {
             return Statistics.stdDev(minGlobal);
      }
      public static double getMaxGlobal() {
             return Statistics.average(maxGlobal);
      }

      public static double getSDMaxGlobal() {
             return Statistics.stdDev(maxGlobal);
      }
      public static double getSumaGlobal() {
             return Statistics.average(sumaGlobal);
      }

      public static double getSDSumaGlobal() {
```

```
		return Statistics.stdDev(sumaGlobal);
	}
	public static double getContadorGlobal() {
		return Statistics.average(contadorGlobal);
	}

	public static double getSDContadorGlobal() {
		return Statistics.stdDev(contadorGlobal);
	}
}
```

9.1.4. ConflictosListener.java

```
package es.ull.iis.simulation.portYardEarth;

import java.util.TreeMap;

import es.ull.iis.simulation.info.ElementActionInfo;

import es.ull.iis.simulation.info.SimulationEndInfo;
import es.ull.iis.simulation.info.SimulationInfo;
import es.ull.iis.simulation.info.SimulationStartInfo;
import es.ull.iis.simulation.inforeceiver.View;


public class ConflictoListener extends View {

// Esta variable se procesa igual que promedioGlobal en
TiempoEstanciaListener
	protected static double contadorEsperaInicioGlobal = 0.0;
	protected static double contadorReservaGruaGlobal = 0.0;
	protected static double contadorTramo1Global = 0.0;
	protected static double contadorTramo2Global = 0.0;
	protected static double contadorTramo3Global = 0.0;
	protected static double contadorTramo4Global = 0.0;

	protected static double promedioEsperaInicioGlobal = 0.0;
	protected static double promedioReservaGruaGlobal = 0.0;
	protected static double promedioTramo1Global = 0.0;
	protected static double promedioTramo2Global = 0.0;
	protected static double promedioTramo3Global = 0.0;
	protected static double promedioTramo4Global = 0.0;

	TreeMap<Integer, Long> esperaInicio = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> reservaGrua = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> esperaFin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> reservaGruaFin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo1 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo2 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo3 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo4 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo1Fin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo2Fin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo3Fin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo4Fin = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> InicioParking = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> InicioGrua = new TreeMap<Integer, Long>();

	private int nSim;
	public ConflictoListener(int nSim){
```

```
            super("");
            addEntrance(ElementActionInfo.class);
            addEntrance(SimulationStartInfo.class);
            addEntrance(SimulationEndInfo.class);
            this.nSim = nSim;
      }

      public void infoEmited(SimulationInfo info){
            if (info instanceof ElementActionInfo){
                  final ElementActionInfo eInfo = (ElementActionInfo) info;
                  final int conflictoId =
eInfo.getElement().getIdentifier();
                  switch (eInfo.getType()){
                  case REQ:
// Lo ponemos en negativo para identificarlo en caso que llegue el final de
simulación y no se haya cogido el recurso correspondiente

      if(eInfo.getActivity().getDescription().contains(PortSimulation.LLEGAD
A)){
                              esperaInicio.put(conflictoId, -
eInfo.getTs());
                        }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.GRUAIN
)){
                              reservaGrua.put(conflictoId, eInfo.getTs());
                        }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO1)){
                              tramo1.put(conflictoId, eInfo.getTs());
                        }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO2)){
                              tramo2.put(conflictoId, eInfo.getTs());
                        }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO3)){
                              tramo3.put(conflictoId, eInfo.getTs());
                        }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO4)){
                              tramo4.put(conflictoId, eInfo.getTs());
                        }
                        break;
                  case ACQ:

      if(eInfo.getActivity().getDescription().contains(PortSimulation.LLEGAD
A)){
            if (eInfo.getTs() > -esperaInicio.get(conflictoId)) {
                  esperaInicio.put(conflictoId, eInfo.getTs() +
            esperaInicio.get(conflictoId));//tiempo que esta en cola o
            esperando por el aparcamiento
                              }
                              else {
                                    esperaInicio.remove(conflictoId);
```

```
                        }
                    }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.GRUAIN
)){
            if (eInfo.getTs() > reservaGrua.get(conflictoId)){
                  reservaGrua.put(conflictoId, eInfo.getTs() -
            reservaGrua.get(conflictoId));//tiempo que espera por el recurso
            grua
                        }
                  else{
                        reservaGrua.remove(conflictoId);
                        }
                  }


      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO1)){
            if (eInfo.getTs() > tramo1.get(conflictoId)){
                  tramo1.put(conflictoId, eInfo.getTs() -
      tramo1.get(conflictoId));//tiempo que espera por el recurso tramo 1
                  }
                  else{
                        tramo1.remove(conflictoId);
                        }
                  }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO2)){
            if (eInfo.getTs() > tramo2.get(conflictoId)){
            tramo2.put(conflictoId, eInfo.getTs() -
tramo2.get(conflictoId));//tiempo que espera por el recurso tramo 2
                  }
                  else{
                        tramo2.remove(conflictoId);
                        }
                  }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO3)){
            if (eInfo.getTs() > tramo3.get(conflictoId)){
                              tramo3.put(conflictoId, eInfo.getTs()
- tramo3.get(conflictoId));//tiempo que espera por el recurso tramo 3
                  }
                  else{
                        tramo3.remove(conflictoId);
                        }
                  }

      if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO4)){
                        if (eInfo.getTs() > tramo4.get(conflictoId)){
                              tramo4.put(conflictoId, eInfo.getTs()
- tramo4.get(conflictoId));//tiempo que espera por el recurso tramo 4
                  }
                  else{
                        tramo4.remove(conflictoId);
                        }
                  }
```

```
                        break;
                    case END:

        if(eInfo.getActivity().getDescription().contains(PortSimulation.LLEGAD
A)){
                                esperaFin.put(conflictoId, eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.GRUAIN
)){
                                reservaGruaFin.put(conflictoId,
eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO1)){
                                tramo1Fin.put(conflictoId, eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO2)){
                                tramo2Fin.put(conflictoId, eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO3)){
                                tramo3Fin.put(conflictoId, eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.ON_TRA
MO4)){
                                tramo4Fin.put(conflictoId, eInfo.getTs());
                            }
                            break;
                    case START:

        if(eInfo.getActivity().getDescription().contains(PortSimulation.LLEGAD
A)){
                            InicioParking.put(conflictoId, eInfo.getTs());
                            }

        if(eInfo.getActivity().getDescription().contains(PortSimulation.GRUAIN
)){
                                InicioGrua.put(conflictoId, eInfo.getTs());
                            }

                            break;
                    }
            }
            else if (info instanceof SimulationEndInfo) {
                    long endTs = ((SimulationEndInfo) info).getTs();

contadorEsperaInicioGlobal = contadorEsperaInicioGlobal + esperaInicio.size()
/ (double) nSim;
contadorReservaGruaGlobal = contadorReservaGruaGlobal + reservaGrua.size() /
(double) nSim;
contadorTramo1Global = contadorTramo1Global + tramo1.size() / (double) nSim;
contadorTramo2Global = contadorTramo2Global + tramo2.size() / (double) nSim;
contadorTramo3Global = contadorTramo3Global + tramo3.size() / (double) nSim;
```

```
contadorTramo4Global = contadorTramo4Global + tramo4.size() / (double) nSim;

System.out.println("\tTipo conflicto\tNúmero de conflictos");
System.out.println("\tAparcamiento "+"\t\t" +
String.format("%.2f",contadorEsperaInicioGlobal));
System.out.println("\tGrúa " + "\t\t\t" +
String.format("%.2f",contadorReservaGruaGlobal));
System.out.println("\tTramo 1 " + "\t\t" +
String.format("%.2f",contadorTramo1Global));
System.out.println("\tTramo 2 " + "\t\t" +
String.format("%.2f",contadorTramo2Global));
System.out.println("\tTramo 3 " + "\t\t" +
String.format("%.2f",contadorTramo3Global));
System.out.println("\tTramo 4 " + "\t\t" +
String.format("%.2f",contadorTramo4Global));


System.out.println("\tConflicto\tPetición\tTiempo de espera");
promedioEsperaInicioGlobal = promedioEsperaInicioGlobal +
(limpiaYCalculaConflictos(esperaInicio, "Aparcamiento", endTs) /
(double)esperaInicio.size()) / (double)nSim;
promedioReservaGruaGlobal = promedioReservaGruaGlobal +
(limpiaYCalculaConflictos(reservaGrua, "Grúa", endTs) /
(double)reservaGrua.size()) / (double)nSim;
promedioTramo1Global = promedioTramo1Global +
(limpiaYCalculaConflictos(tramo1, "Tramo 1", endTs) / (double) tramo1.size())
/ (double)nSim;
promedioTramo2Global = promedioTramo2Global +
(limpiaYCalculaConflictos(tramo2, "Tramo 2", endTs) / (double) tramo2.size())
/ (double)nSim;
promedioTramo3Global = promedioTramo3Global +
(limpiaYCalculaConflictos(tramo3, "Tramo 3", endTs) / (double) tramo3.size())
/ (double)nSim;
promedioTramo4Global = promedioTramo4Global +
(limpiaYCalculaConflictos(tramo4, "Tramo 4", endTs) / (double) tramo4.size())
/ (double)nSim;
            }
      }

//Este es un metodo que utilizamos para comprobar que no hayan peticiones de
//recursos que no se hayan atendido
//antes de terminar la simulación

      long limpiaYCalculaConflictos(TreeMap<Integer, Long> conflictos,
String texto, long endTs) {
// Comprobamos que no haya peticiones de recursos que no se hayan atendido
antes //de terminar la simulación
            for (Integer conflictoId : conflictos.keySet()) {
                  if (conflictos.get(conflictoId) < 0) {
                        if (-conflictos.get(conflictoId) < endTs) {
                              conflictos.put(conflictoId, endTs +
conflictos.get(conflictoId));//tiempo que esta en cola o esperando por el
aparcamiento
                        }
                        else {
                              conflictos.remove(conflictoId);
                        }
                  }
            }
```

```
            long sumaconflictos = 0;
            for (Integer conflictoId : conflictos.keySet()) {
                  System.out.println( "\t" + texto + "  "+ "\t" +" "+
conflictoId + "\t\t" +   conflictos.get(conflictoId));
                  sumaconflictos = sumaconflictos +
conflictos.get(conflictoId);
            }
            return sumaconflictos;
      }
}
```

9.1.5. DatosEntrada.java

```
package es.ull.iis.simulation.portYardEarth;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;
import java.awt.Toolkit;
import javax.swing.ImageIcon;


public class DatosEntrada extends JFrame {
private static final int NGRUAS = 2;
private static final int NCALLESA = 10;
private static final int NCAMIONES = 15;
private static final int NCALLESINT1 = 1;
private static final int NCALLESINT2 = 1;
private static final int NCALLESINT3 = 1;
private static final int NCALLESINT4 = 1;
//TIEMPO EN MINUTOS
private static final long MEDIA_TRAMO_PATIO = 1*60; //Tramo1
private static final long MEDIA_TRAMO_CENTRO = 3*60; //Tramo2
private static final long MEDIA_TRAMO_TIERRA = 3*60; //Tramo3
private static final long MEDIA_CENTRO_VUELTA = 1*60; //Tramo4
private static final long MEDIA_DESCARGA_CONTENEDOR = 6*60;//Descarga
private static final long MEDIA_TRANSFERENCIA = 3*60;//Transferencia
private static final long MEDIA_VUELTA_PATIO1 = 1*60; //Tramo1Vuelta

private static final int NSIM = 1;
private final double ERROR = 0.25;

      private JPanel contentPane;
      protected static JTextField textField;
      protected static JTextField textField_1;
      protected static JTextField textField_2;
      protected static JTextField textField_3;
      protected static JTextField textField_4;
      protected static JTextField textField_5;
      protected static JTextField textField_6;
      protected static JTextField textField_7;
```

```
	protected static JTextField textField_8;
	protected static JTextField textField_9;
	protected static JTextField textField_10;
	protected static JTextField textField_11;
	protected static JTextField textField_12;
	protected static JTextField textField_13;
	protected static JTextField textField_14;
	private JButton btnCancelar;
	private JLabel lblNewLabel;
	private JLabel lblTiempo;
	protected static JTextField textField_15;


	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					DatosEntrada frame = new DatosEntrada();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public DatosEntrada(){

		super("IdeaPort1Main");


	setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Daniel\\
Desktop\\TFG General\\logotipo-secundario-ULL.png"));
		setBackground(Color.WHITE);
		setForeground(Color.WHITE);
		setTitle("DATOS DEL MODELO A SIMULAR");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 753, 544);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JButton btnAceptar = new JButton("ACEPTAR");
		btnAceptar.addActionListener(new Aceptar());
		btnAceptar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				CloseFrame();
			}
		});
		btnAceptar.setBounds(614, 17, 113, 23);
		contentPane.add(btnAceptar);

		JLabel lblGruas = new JLabel("N\u00BA GRUAS");
		lblGruas.setBounds(23, 21, 59, 14);
		contentPane.add(lblGruas);
```

```
                JLabel lblCallesBloque = new JLabel("N\u00BA APARCAMIENTOS
BLOQUE");
                lblCallesBloque.setFont(new Font("Tahoma", Font.PLAIN, 11));
                lblCallesBloque.setBounds(23, 70, 153, 14);
                contentPane.add(lblCallesBloque);

                JLabel lblCamiones = new JLabel("N\u00BA PETICIONES / HORA");
                lblCamiones.setBounds(23, 45, 125, 14);
                contentPane.add(lblCamiones);

                JLabel lblCALLESTRAMO1 = new JLabel("TRAMO 1");
                lblCALLESTRAMO1.setBounds(23, 132, 94, 14);
                contentPane.add(lblCALLESTRAMO1);

                JLabel lblNewLabel_1 = new JLabel("TRAMO 2");
                lblNewLabel_1.setBounds(23, 157, 94, 14);
                contentPane.add(lblNewLabel_1);

                JLabel lblNewLabel_2 = new JLabel("TRAMO 3");
                lblNewLabel_2.setBounds(23, 182, 94, 14);
                contentPane.add(lblNewLabel_2);

                JLabel lblNewLabel_3 = new JLabel("TRAMO 4");
                lblNewLabel_3.setBounds(23, 207, 94, 14);
                contentPane.add(lblNewLabel_3);

                textField = new JTextField();
                textField.setBounds(219, 21, 86, 14);
                contentPane.add(textField);
                textField.setColumns(10);
                textField.setText(""+ NGRUAS);
                textField.setHorizontalAlignment(JTextField.RIGHT);

                textField_1 = new JTextField();
                textField_1.setBounds(219, 45, 86, 14);
                contentPane.add(textField_1);
                textField_1.setColumns(10);
                textField_1.setText(""+ NCAMIONES);
                textField_1.setHorizontalAlignment(JTextField.RIGHT);

                textField_2 = new JTextField();
                textField_2.setColumns(10);
                textField_2.setBounds(219, 70, 86, 14);
                contentPane.add(textField_2);
                textField_2.setText(""+ NCALLESA);
                textField_2.setHorizontalAlignment(JTextField.RIGHT);


                textField_3 = new JTextField();
                textField_3.setColumns(10);
                textField_3.setBounds(97, 132, 86, 14);
                contentPane.add(textField_3);
                textField_3.setText(""+ NCALLESINT1);
                textField_3.setHorizontalAlignment(JTextField.RIGHT);


                textField_4 = new JTextField();
                textField_4.setColumns(10);
```

```
textField_4.setBounds(97, 157, 86, 14);
contentPane.add(textField_4);
textField_4.setText(""+ NCALLESINT2);
textField_4.setHorizontalAlignment(JTextField.RIGHT);


textField_5 = new JTextField();
textField_5.setColumns(10);
textField_5.setBounds(97, 182, 86, 14);
contentPane.add(textField_5);
textField_5.setText(""+ NCALLESINT3);
textField_5.setHorizontalAlignment(JTextField.RIGHT);


textField_6 = new JTextField();
textField_6.setColumns(10);
textField_6.setBounds(97, 207, 86, 14);
contentPane.add(textField_6);
textField_6.setText(""+ NCALLESINT4);
textField_6.setHorizontalAlignment(JTextField.RIGHT);


btnCancelar = new JButton("CANCELAR");
btnCancelar.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		try{
			Thread.sleep(100);
			System.exit(0);
		}catch(Exception excep){
			System.exit(0);
		}
		CloseFrame();
	}


});
btnCancelar.setBounds(614, 51, 113, 23);
contentPane.add(btnCancelar);

lblNewLabel = new JLabel("CALLES");
lblNewLabel.setBounds(117, 107, 46, 14);
contentPane.add(lblNewLabel);

lblTiempo = new JLabel("TIEMPO");
lblTiempo.setBounds(229, 95, 46, 14);
contentPane.add(lblTiempo);

textField_7 = new JTextField();
textField_7.setColumns(10);
textField_7.setBounds(219, 132, 86, 14);
contentPane.add(textField_7);
textField_7.setText(""+ MEDIA_TRAMO_PATIO);
textField_7.setHorizontalAlignment(JTextField.RIGHT);


textField_8 = new JTextField();
textField_8.setColumns(10);
textField_8.setBounds(219, 157, 86, 14);
contentPane.add(textField_8);
```

```
                textField_8.setText(""+ MEDIA_TRAMO_CENTRO);
                textField_8.setHorizontalAlignment(JTextField.RIGHT);

                textField_9 = new JTextField();
                textField_9.setColumns(10);
                textField_9.setBounds(219, 182, 86, 14);
                contentPane.add(textField_9);
                textField_9.setText(""+ MEDIA_TRAMO_TIERRA);
                textField_9.setHorizontalAlignment(JTextField.RIGHT);


                textField_10 = new JTextField();
                textField_10.setColumns(10);
                textField_10.setBounds(219, 207, 86, 14);
                contentPane.add(textField_10);
                textField_10.setText(""+ MEDIA_CENTRO_VUELTA);
                textField_10.setHorizontalAlignment(JTextField.RIGHT);

                JLabel lblTiempoTramoVuelta = new JLabel("TRAMO 1 VUELTA");
                lblTiempoTramoVuelta.setBounds(23, 252, 133, 14);
                contentPane.add(lblTiempoTramoVuelta);

                JLabel lblTiempoDescargaPatio = new JLabel("DESCARGA PATIO");
                lblTiempoDescargaPatio.setBounds(23, 277, 140, 14);
                contentPane.add(lblTiempoDescargaPatio);

                JLabel lblTiempoTransferencia = new JLabel("TIEMPO
TRANSFERENCIA");
                lblTiempoTransferencia.setBounds(23, 302, 160, 14);
                contentPane.add(lblTiempoTransferencia);

                textField_11 = new JTextField();
                textField_11.setColumns(10);
                textField_11.setBounds(219, 249, 86, 14);
                contentPane.add(textField_11);
                textField_11.setText(""+ MEDIA_VUELTA_PATIO1);
                textField_11.setHorizontalAlignment(JTextField.RIGHT);


                textField_12 = new JTextField();
                textField_12.setColumns(10);
                textField_12.setBounds(219, 274, 86, 14);
                contentPane.add(textField_12);
                textField_12.setText(""+ MEDIA_DESCARGA_CONTENEDOR);
                textField_12.setHorizontalAlignment(JTextField.RIGHT);


                textField_13 = new JTextField();
                textField_13.setColumns(10);
                textField_13.setBounds(219, 299, 86, 14);
                contentPane.add(textField_13);
                textField_13.setText(""+ MEDIA_TRANSFERENCIA);
                textField_13.setHorizontalAlignment(JTextField.RIGHT);

                JLabel label = new JLabel("");
                label.setBounds(412, 182, 46, 14);
                contentPane.add(label);

                JLabel lblNewLabel_4 = new JLabel("");
```

```
            lblNewLabel_4.setIcon(new
ImageIcon("C:\\Users\\Daniel\\Desktop\\TFG General\\ESQUEMATFG.png"));
            lblNewLabel_4.setBounds(326, 90, 401, 405);
            contentPane.add(lblNewLabel_4);

            JLabel lblNmeroDeSimulaciones = new JLabel("N\u00DAMERO DE
SIMULACIONES");
            lblNmeroDeSimulaciones.setBounds(23, 387, 160, 14);
            contentPane.add(lblNmeroDeSimulaciones);

            textField_14 = new JTextField();
            textField_14.setBounds(193, 387, 37, 19);
            contentPane.add(textField_14);
            textField_14.setColumns(10);
            textField_14.setText(""+ NSIM);
            textField_14.setHorizontalAlignment(JTextField.RIGHT);

            JLabel lblError = new JLabel("ERROR");
            lblError.setBounds(23, 420, 46, 14);
            contentPane.add(lblError);

            textField_15 = new JTextField();
            textField_15.setBounds(193, 417, 37, 20);
            contentPane.add(textField_15);
            textField_15.setColumns(10);
            textField_15.setText(""+ ERROR);
            textField_15.setHorizontalAlignment(JTextField.RIGHT);

            JLabel lblsegundos = new JLabel("(SEGUNDOS)");
            lblsegundos.setBounds(219, 107, 86, 14);
            contentPane.add(lblsegundos);

      }

      protected void CloseFrame() {
            super.dispose();

      }
}
```

9.1.6. Aceptar.java

```
package es.ull.iis.simulation.portYardEarth;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Aceptar implements ActionListener {

      public Aceptar() {
      }

      public void actionPerformed(ActionEvent arg0) {

IdeaPort1Main sim = new
IdeaPort1Main(Integer.parseInt(DatosEntrada.textField.getText()),Integer.pars
eInt(DatosEntrada.textField_1.getText()),Integer.parseInt(DatosEntrada.textFi
eld_2.getText()),Integer.parseInt(DatosEntrada.textField_3.getText()),Integer
.parseInt(DatosEntrada.textField_4.getText()),Integer.parseInt(DatosEntrada.t
extField_5.getText()),Integer.parseInt(DatosEntrada.textField_6.getText()),In
```

```
teger.parseInt(DatosEntrada.textField_7.getText()),Integer.parseInt(DatosEntr
ada.textField_8.getText()),Integer.parseInt(DatosEntrada.textField_9.getText(
)),Integer.parseInt(DatosEntrada.textField_10.getText()),Integer.parseInt(Dat
osEntrada.textField_11.getText()),Integer.parseInt(DatosEntrada.textField_12.
getText()),Integer.parseInt(DatosEntrada.textField_13.getText()),Integer.pars
eInt(DatosEntrada.textField_14.getText()),Double.parseDouble(DatosEntrada.tex
tField_15.getText()));
			sim.start();
			TiemposConcluyentes Ventana2 = new TiemposConcluyentes();
			Ventana2.setVisible(true);
		}

}
```

9.1.7. TiemposConcluyentes.java

```
package es.ull.iis.simulation.portYardEarth;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.WindowListener;
import java.util.TreeMap;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import java.awt.Toolkit;
import es.ull.iis.simulation.portYardEarth.TiempoEstanciaListener;
import javax.swing.SwingConstants;

public class TiemposConcluyentes extends JFrame {

	private JPanel contentPane;
	public JTextField textField;
	public JTextField textField_1;
	public JTextField textField_2;
	public JTextField textField_3;
	public JTextField textField_6;
	TreeMap<Integer, Long> esperaInicio = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> reservaGrua = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo1 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo2 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo3 = new TreeMap<Integer, Long>();
	TreeMap<Integer, Long> tramo4 = new TreeMap<Integer, Long>();
	private JTextField SDTIMETOTALtextField_4;
	private JTextField SDTIMEPROMEDIOtextField_5;
	private JTextField SDTIMEMINtextField_7;
	private JTextField SDTIMEMAXtextField_8;
	private JTextField SDNOFINALIZADAStextField_9;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
```

```
                                TiemposConcluyentes frame = new
TiemposConcluyentes();
                                frame.setVisible(true);
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                        }
                });
        }

        /**
         * Create the frame.
         * @return
         */

        public TiemposConcluyentes() {

                super("DatosEntrada");


        setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Daniel\\
Desktop\\TFG General\\logotipo-secundario-ULL.png"));
                setTitle("TIEMPO DE EXTRACCI\u00D3N DE CONTENEDORES");
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setBounds(100, 100, 611, 387);
                contentPane = new JPanel();
                contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
                setContentPane(contentPane);
                contentPane.setLayout(null);

                JLabel lblGruasDisponibles = new JLabel("TIEMPO TOTAL PROMEDIO
DE EXTRACCI\u00D3N ");
                lblGruasDisponibles.setBounds(22, 50, 258, 14);
                contentPane.add(lblGruasDisponibles);

                JLabel lblPeticionesCamiones = new JLabel("TIEMPO PROMEDIO DE
EXTRACCI\u00D3N  ");
                lblPeticionesCamiones.setBounds(22, 78, 235, 14);
                contentPane.add(lblPeticionesCamiones);

                JLabel lblAparcamientosDelBloque = new JLabel("TIEMPO MINIMO DE
EXTRACCI\u00D3N");
                lblAparcamientosDelBloque.setBounds(22, 108, 235, 14);
                contentPane.add(lblAparcamientosDelBloque);

                JLabel lblCallesTramo = new JLabel("TIEMPO M\u00C1XIMO DE
EXTRACCI\u00D3N");
                lblCallesTramo.setBounds(22, 144, 235, 14);
                contentPane.add(lblCallesTramo);

                JCheckBox chckbxNewCheckBox = new JCheckBox("NUMERO DE
CONFLICTOS POR RECURSO");
                chckbxNewCheckBox.setBounds(52, 282, 306, 23);
                contentPane.add(chckbxNewCheckBox);

                JCheckBox chckbxTiempoDeConflicto = new JCheckBox("TIEMPO DE
CONFLICTO POR RECURSO (PROMEDIO)");
                chckbxTiempoDeConflicto.setBounds(52, 308, 322, 23);
                contentPane.add(chckbxTiempoDeConflicto);
```

```
		JButton btnVerificar = new JButton("OBTENER ");
		btnVerificar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(chckbxNewCheckBox.isSelected() == true){
					GrafNumConflictos Conflictos = new
GrafNumConflictos();
					Conflictos.setVisible(true);
				}
				CloseFrame();
				if (chckbxTiempoDeConflicto.isSelected() == true){
					GrafTimeConflictos Promedio = new
GrafTimeConflictos();
					Promedio.setVisible(true);
					}
				CloseFrame();
			}
			private void CloseFrame() {


			}
		});
		btnVerificar.setBounds(425, 274, 147, 23);
		contentPane.add(btnVerificar);

		textField = new JTextField();
		textField.setBounds(290, 48, 86, 17);
		contentPane.add(textField);
		textField.setColumns(10);
		textField.setText("" +
String.format("%.2f",TiempoEstanciaListener.getSumaGlobal()/60));//tiempo
total
		textField.setHorizontalAlignment(JTextField.RIGHT);

		textField_1 = new JTextField();
		textField_1.setColumns(10);
		textField_1.setBounds(290, 76, 86, 17);
		contentPane.add(textField_1);
		textField_1.setText(""+ String.format("%.2f",
TiempoEstanciaListener.getPromedioGlobal()/60));//tiempo promedio
//
	System.out.println(TiempoEstanciaListener.getSDPromedioGlobal());
		textField_1.setHorizontalAlignment(JTextField.RIGHT);

		textField_2 = new JTextField();
		textField_2.setColumns(10);
		textField_2.setBounds(290, 106, 86, 17);
		contentPane.add(textField_2);
		textField_2.setText(""+
String.format("%.2f",TiempoEstanciaListener.getMinGlobal()/60)); //minimo
		textField_2.setHorizontalAlignment(JTextField.RIGHT);

		textField_3 = new JTextField();
		textField_3.setColumns(10);
		textField_3.setBounds(290, 142, 86, 17);
		contentPane.add(textField_3);
		textField_3.setText(""+
String.format("%.2f",TiempoEstanciaListener.getMaxGlobal()/60));//maximo
		textField_3.setHorizontalAlignment(JTextField.RIGHT);
```

```
            textField_6 = new JTextField();
            textField_6.setColumns(10);
            textField_6.setBounds(299, 203, 59, 23);
            contentPane.add(textField_6); // no acabaron
            textField_6.setText(""+
String.format("%.2f",TiempoEstanciaListener.getContadorGlobal()));
            textField_6.setHorizontalAlignment(JTextField.RIGHT);

            JButton btnCancelar = new JButton("CANCELAR");
            btnCancelar.setBounds(425, 308, 147, 23);
            contentPane.add(btnCancelar);

            JLabel lblMinutos = new JLabel("MINUTOS");
            lblMinutos.setBounds(306, 25, 59, 14);
            contentPane.add(lblMinutos);

            JLabel lblGrficasseleccioneLa = new JLabel("GR\u00C1FICAS :
(Seleccione la gr\u00E1fica y pulse el bot\u00F3n obtener)");
            lblGrficasseleccioneLa.setBounds(22, 250, 427, 14);
            contentPane.add(lblGrficasseleccioneLa);

            JLabel lblPeticionesNoFinalizadas = new JLabel("PETICIONES NO
FINALIZADAS");
            lblPeticionesNoFinalizadas.setBounds(22, 212, 167, 14);
            contentPane.add(lblPeticionesNoFinalizadas);

            JLabel lblPromedio = new JLabel("PROMEDIO");
            lblPromedio.setBounds(300, 11, 74, 14);
            contentPane.add(lblPromedio);

            SDTIMETOTALtextField_4 = new JTextField();
            SDTIMETOTALtextField_4.setText(""+
String.format("%.2f",TiempoEstanciaListener.getSDSumaGlobal()/60));

      SDTIMETOTALtextField_4.setHorizontalAlignment(SwingConstants.RIGHT);
            SDTIMETOTALtextField_4.setColumns(10);
            SDTIMETOTALtextField_4.setBounds(425, 47, 86, 17);
            contentPane.add(SDTIMETOTALtextField_4);

            SDTIMEPROMEDIOtextField_5 = new JTextField();
            SDTIMEPROMEDIOtextField_5.setText(""+
String.format("%.2f",TiempoEstanciaListener.getSDPromedioGlobal()/60));

      SDTIMEPROMEDIOtextField_5.setHorizontalAlignment(SwingConstants.RIGHT)
;
            SDTIMEPROMEDIOtextField_5.setColumns(10);
            SDTIMEPROMEDIOtextField_5.setBounds(425, 75, 86, 17);
            contentPane.add(SDTIMEPROMEDIOtextField_5);

            SDTIMEMINtextField_7 = new JTextField();
            SDTIMEMINtextField_7.setText(""+
String.format("%.2f",TiempoEstanciaListener.getSDMinGlobal()/60));

      SDTIMEMINtextField_7.setHorizontalAlignment(SwingConstants.RIGHT);
            SDTIMEMINtextField_7.setColumns(10);
            SDTIMEMINtextField_7.setBounds(425, 105, 86, 17);
            contentPane.add(SDTIMEMINtextField_7);
```

```
            SDTIMEMAXtextField_8 = new JTextField();
            SDTIMEMAXtextField_8.setText(""+
String.format("%.2f",TiempoEstanciaListener.getSDMaxGlobal()/60));

      SDTIMEMAXtextField_8.setHorizontalAlignment(SwingConstants.RIGHT);
            SDTIMEMAXtextField_8.setColumns(10);
            SDTIMEMAXtextField_8.setBounds(425, 141, 86, 17);
            contentPane.add(SDTIMEMAXtextField_8);

            JLabel lblDesEstndar = new JLabel("DES. EST\u00C1NDAR");
            lblDesEstndar.setBounds(426, 11, 97, 14);
            contentPane.add(lblDesEstndar);

            JLabel lblUnidad = new JLabel("UNIDAD");
            lblUnidad.setBounds(312, 182, 46, 14);
            contentPane.add(lblUnidad);

            JLabel label = new JLabel("DES. EST\u00C1NDAR");
            label.setBounds(426, 182, 97, 14);
            contentPane.add(label);

            SDNOFINALIZADAStextField_9 = new JTextField();
            SDNOFINALIZADAStextField_9.setText(""+
String.format("%.2f",TiempoEstanciaListener.getSDContadorGlobal()));

      SDNOFINALIZADAStextField_9.setHorizontalAlignment(SwingConstants.RIGHT
);
            SDNOFINALIZADAStextField_9.setColumns(10);
            SDNOFINALIZADAStextField_9.setBounds(425, 209, 86, 17);
            contentPane.add(SDNOFINALIZADAStextField_9);

            btnCancelar.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {

                        CloseFrame();
                  }
            });
      }
      private void CloseFrame() {
            super.dispose();
      }
}
```

9.1.8. GrafNumConflictos.java

```
package es.ull.iis.simulation.portYardEarth;

import java.awt.Color;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

public class GrafNumConflictos extends JFrame{
```

```
      JPanel panel;
    public GrafNumConflictos(){

      setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Daniel\\
Desktop\\TFG General\\logotipo-secundario-ULL.png"));
        setTitle("Número de conflictos por recursos");
        setSize(800,500);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setVisible(true);
        init();
    }
    private void init() {
        panel = new JPanel();
        getContentPane().add(panel);

        // Fuente de Datos
        DefaultCategoryDataset conflictos = new DefaultCategoryDataset();
        conflictos.addValue(ConflictoListener.contadorEsperaInicioGlobal,
"Aparcamiento", "Tipo de recurso");
        conflictos.addValue(ConflictoListener.contadorReservaGruaGlobal,
"Grúa", "Tipo de recurso");
        conflictos.addValue(ConflictoListener.contadorTramo1Global, "Tramo1",
"Tipo de recurso");
        conflictos.addValue(ConflictoListener.contadorTramo2Global, "Tramo2",
"Tipo de recurso");
        conflictos.addValue(ConflictoListener.contadorTramo3Global, "Tramo3",
"Tipo de recurso");
        conflictos.addValue(ConflictoListener.contadorTramo4Global, "Tramo4",
"Tipo de recurso");

        // Creando el Grafico
        JFreeChart chart = ChartFactory.createBarChart3D
        ("Número de conflictos por recurso","Recursos", "Número de conflictos
(Unidad)",
        conflictos, PlotOrientation.VERTICAL, true,true, false);
        chart.setBackgroundPaint(Color.white);
        chart.getTitle().setPaint(Color.black);
        CategoryPlot p = chart.getCategoryPlot();
        p.setRangeGridlinePaint(Color.red);
        // Mostrar Grafico
        ChartPanel chartPanel = new ChartPanel(chart);
        panel.add(chartPanel);
    }
}
```

9.1.9. GrafTimeConflictos.java

```
package es.ull.iis.simulation.portYardEarth;

import java.awt.Color;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
```

```
public class GrafTimeConflictos extends JFrame{
    JPanel panel;

    public GrafTimeConflictos(){

      setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Daniel\\
Desktop\\TFG General\\logotipo-secundario-ULL.png"));
        setTitle("Tiempo promedio de los conflictos");
        setSize(800,500);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setVisible(true);
        init();
    }
    private void init() {
        panel = new JPanel();
        getContentPane().add(panel);

        // Fuente de Datos
        DefaultCategoryDataset Promedio = new DefaultCategoryDataset();
        Promedio.addValue(ConflictoListener.promedioEsperaInicioGlobal/60,
"Aparcamiento", "Tipo de recurso");
        Promedio.addValue(ConflictoListener.promedioReservaGruaGlobal/60,
"Grúa", "Tipo de recurso");
        Promedio.addValue(ConflictoListener.promedioTramo1Global/60,
"Tramo1", "Tipo de recurso");
        Promedio.addValue(ConflictoListener.promedioTramo2Global/60,
"Tramo2", "Tipo de recurso");
        Promedio.addValue(ConflictoListener.promedioTramo3Global/60,
"Tramo3", "Tipo de recurso");
        Promedio.addValue(ConflictoListener.promedioTramo4Global/60,
"Tramo4", "Tipo de recurso");

        // Creando el Grafico
        JFreeChart chart = ChartFactory.createBarChart3D
        ("Tiempo promedio de los conflictos","Recursos", "Tiempo de los
conflictos (minutos)",
        Promedio, PlotOrientation.VERTICAL, true,true, false);
        chart.setBackgroundPaint(Color.white);
        chart.getTitle().setPaint(Color.black);
        CategoryPlot p = chart.getCategoryPlot();
        p.setRangeGridlinePaint(Color.red);
        // Mostrar Grafico
        ChartPanel chartPanel = new ChartPanel(chart);
        panel.add(chartPanel);
    }
}
```