Grado en Ingeniería Electrónica Industrial y Automática.

Curso 2016-2017

Septiembre 2017

Trabajo Fin de Grado

“Modelado y simulación de la zona mar-patio de un puerto de contenedores.”

Alejandro Alonso Méndez

Tutor/es

Iván Castilla Rodríguez

Rosa María Aguilar Chinea

Agradecimientos.

A mi madre y mi hermana que siempre ha estado apoyándome y nunca se han rendido en esa ardua tarea.

A mi pareja que siempre ha aportado su granito de arena en esta tan larga experiencia haciendo que nunca me haya quedado rezagado.

A Iván y Rosi, tutores de este proyecto que siempre han sido amables y siempre han estado a toda disposición para las dudas e inconvenientes que se me han ido presentando, así como por guiarme tan bien por el camino a seguir.

Y a todas las personas que han formado parte de esta experiencia y que han aportado algo a ella.

Resumen.

En el puerto de Santa Cruz de Tenerife y, en general, en la mayoría de los puertos, la gestión de las operaciones de carga y descarga de contenedores desde y hacia los barcos es un proceso complejo en el que es necesaria una coordinación y planificación adecuada de los recursos disponibles. Estos recursos materiales y humanos se coordinan de manera planificada y estructurada siguiendo unos pasos marcados por una planificación diaria. Dichos recursos se encuentran en constante movimiento, debido a que, en el caso de parada, repercutiría en unas pérdidas económicos tanto para los proveedores como para los clientes.

Debido a la imposibilidad de comprobar en el momento y en el mismo lugar las mejoras de las estrategias para el mejor funcionamiento y solución de los problemas que pudieran surgir, se realiza este proyecto de modelado y simulación como alternativa a la parada del funcionamiento normal de la parte mar-patio de un puerto.

Para poder simular un modelo realista de dichas acciones se ha programado en lenguaje Java con una herramienta que utiliza este lenguaje, creada por el personal docente de la universidad de La Laguna, cuyo objetivo es facilitar la obtención de resultados a la hora de simular y modelar el modelo real. Esta herramienta se denomina PSIGHOS y dispone de una serie de clases predeterminadas las cuales se han usado para simplificar la simulación del modelo.

Al finalizar este proyecto se han obtenido resultados positivos de una simulación real, la cual resulta ser una base de futuras investigaciones y mejoras para la obtención de un modelo común de todas las zonas del puerto.

Abstract.

In the Santa Cruz de Tenerife's port and, in general, in most of the ports around the world, the management of the loading and unloading operation of containers from and to the ships is a very difficult process which requires coordination and a good planning between the available resources. Material and human resources are coordinated in a planned and structured way following a few steps marked by daily planning. These resources are in constant movement, because a stop of the production would imply economic losses for both the suppliers and the customers.

Due to the impossibility of checking, in the same time and in the same place, the performance of the strategies for the best functioning and solution of the problems that could appear, this modelling and simulation project is carried out as an alternative to stop the normal operation of the sea-container terminal part of the port.

We have developed a realistic model that comprises these operations using the programming language Java. More specifically, we have used a tool that uses this language for programming. This tool has been made by the faculty members of the university of La Laguna and is aimed to facilitate the creation and obtaining of results from a simulation. This tool is called PSIGHOS and has some predetermined class which have been used to simplify the simulation of the model.

At the end of this project, we have obtained results from a number of examples that simulate the port. Thus, this will be a basis for future research and improvements to obtain a common model from all the port areas.

Índice general.

1. Introducción.

1.1. Los puertos y su problemática.

Un puerto es un espacio dedicado, cerca de una costa, para la carga y descarga, embarque y desembarque de los distintos barcos. Se diferencian tres zonas: la zona de mar, la zona de patio y la zona de tierra [9].

La zona de mar es la zona del puerto por la cual llegan los barcos a atracar. En ella se incluyen obras, como diques o esclusas con el fin de proteger a los barcos del oleaje, zonas de señalización y una dársena que es donde atracan los barcos para la permanencia y realizar las operaciones de carga y descarga.

En la zona de patio se definen los diferentes bloques de contenedores donde se diferencian las zonas donde va a ir cada contenedor. Cuando se habla de contenedores en el contexto de operaciones portuarias en todo el mundo, en general, se emplea como referencia la medida estándar de un contenedor, expresada en términos de *twenty-foot equivalent units* (TEU). Existen zonas especiales para los contenedores especiales, ya sea de refrigeración de mercancías peligrosas, etcétera.

La última zona, la tierra es el lugar en el que llegan los camiones, es decir, el exterior del puerto. Existe una zona de espera en la que los camiones esperan a que se les cargue el contenedor destinado.

Dado la ubicación, geográficamente hablando, de nuestras islas, el puerto de Santa Cruz de Tenerife se ha convertido en uno de los puertos referencia en el océano Atlántico, siendo una conexión importante entre América y Europa y entre Europa y África. No obstante, se ve superado por el puerto de Las Palmas de Gran Canaria, que se encuentra situado, en el top 5, como uno de los puertos que más movimientos realiza en España.

En los puertos, el principal problema es que hay que realizar un volumen muy elevado de tareas durante la jornada, sujetas a limitaciones estrictas de tiempo y empleando unos recursos limitados. Se disponen de determinado número de grúas de barco, camiones con grúa incorporada que operan en el interior del muelle y de grúas en los diferentes bloques que abarcan cada una diferentes bloques de patio. Además, el rango de operación de estos recursos depende mucho de las condiciones climatológicas, del turno, si es nocturno o diurno, etc. También se ven afectados por los posibles accidentes que puedan ocurrir, o por la situación en la que se encuentran los elementos, en este caso, los contenedores, ya sea en el bloque de patio

o su situación en el barco. En el barco, un contenedor que tiene como destino el puerto en el que se encuentra puede ser que se encuentre apilado en el fondo de una pila de contenedores, lo que supone el problema de que tendremos que realizar más movimientos que si se encontrara en la cima de una pila de contenedores. Esto mismo sucede si se encuentra en la misma situación en uno de los bloques del patio, tendremos que realizar más movimientos que si se encuentra en la cima de la pila. Por lo que se podría producir un retraso en el plan establecido, retrasando tanto al barco como a los camiones que se encuentran en la zona de espera.

Otro problema que nos podemos encontrar es la urgencia de las mercancías. Por ejemplo, en el caso de materias para el hospital, si éste se queda sin material se necesitaría con urgencia realizar los movimientos necesarios para que ese contenedor llegue lo antes posible al hospital, lo que generaría un paro parcial de las acciones que se estaban realizando y dedicar gran parte de los recursos a esos movimientos.

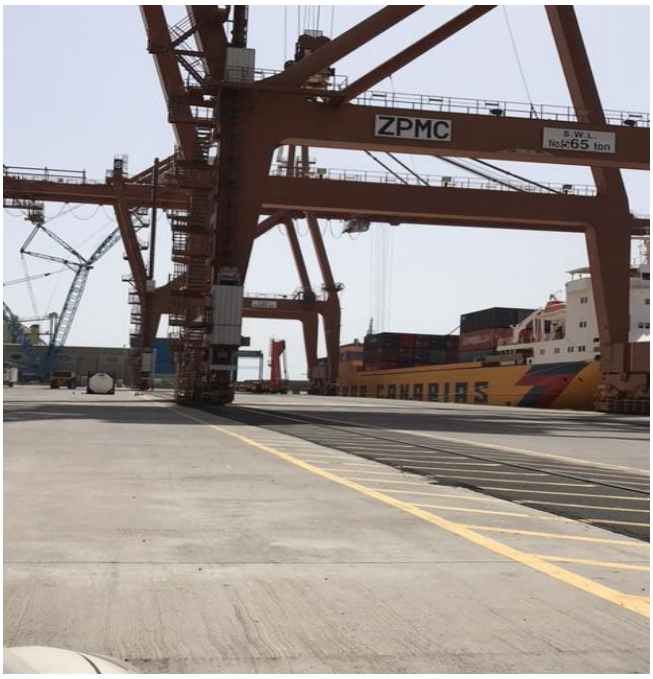

Ilustración 1. Imagen en la que se pueden observar los raíles de las grúas de barco.

Se tiene en cuenta, en el caso de las grúas de barco, cuya misión es descargar los contenedores desde el barco hacia los camiones, no se pueden cruzar entre ellas, es decir, estas grúas se mueven a lo largo de raíles los cuales son comunes para todas. Esto habría que tenerlo en cuenta dado que puede surgir un conflicto si no se declara previamente qué movimientos ha de hacer cada una de estas grúas.

Para paliar la posible problemática que pueda surgir, los encargados realizan un plan cada día de los movimientos que se tienen que realizar en cada turno de estibadores.

1.2. La simulación como solución.

Para conocer las mejores opciones en las diferentes situaciones en el escenario de los puertos, no podremos interrumpir la producción y las acciones planificadas dado que supondría una pérdida de horas de trabajo y, por lo tanto, un perjuicio económico para los actores implicados en las operaciones portuarias, tanto los proveedores de servicios como los clientes. Como

solución a este problema el uso de la simulación se muestra como una alternativa más económica, flexible, rápida y adaptable que permite analizar el problema y las posibles soluciones sin interrumpir el normal funcionamiento del puerto.

Esta herramienta se podrá utilizar a la hora de planificar los movimientos a realizar en los diferentes turnos de estibadores conociendo los posibles factores de retraso y de conflicto entre los recursos. Y, así, realizar una mejor planificación diaria provocando un mejor flujo de contenedores y procurando tener los recursos el menor tiempo ocioso posible, dado que podremos conocer este dato.

Nos podrá ayudar también a la hora de colocar los diferentes contenedores en los distintos bloques para mejorar el flujo de salida terrestre de los contenedores, de manera que no se provoquen atascos entre las grúas de patio y los camiones que llegan externamente. Ayudará a que los camiones exteriores no queden en cola esperando a que su lugar, en la zona de espera, se encuentre disponible, debido a que todo se encontrará planificado.

1.3. Objetivos.

El principal objetivo de este proyecto es la realización de un modelo dinámico que simule el funcionamiento de la zona mar-patio de un puerto como el de Santa Cruz de Tenerife.

Para ello hemos de conocer el funcionamiento, tanto general como específico, de un puerto de manera que esto nos pueda servir a la hora de elaborar nuestro sistema dinámico.

Uno de los objetivos que nos hemos marcado ha sido aprender a simular y a modelar diferentes sistemas dinámicos para la correcta implementación de estos.

Con el fin de aprender, de manera correcta, a simular los diferentes modelos dinámicos y discretos, se ha marcado como objetivo conocer diferentes softwares diseñados específicamente para modelar y simular modelos dinámicos.

También se ha fijado como objetivo la creación de una pequeña interfaz que sirva como prototipo de herramienta de ayuda a la toma de decisiones de los gestores del puerto.

Este proyecto se ha realizado con el fin de que sirva como base para futuras investigaciones. Y poder dar un conocimiento inicial y una inspiración en las futuras investigaciones o proyectos relacionados con nuestro proyecto.

1.4. Estructura de la memoria.

A continuación, se describe brevemente la estructura por la que está formada la memoria.

CAPITULO 1: ***Introducción.*** En este capítulo se explica brevemente el tema del que trata la memoria, el cual se explicará a lo largo de ella. Se tratarán los problemas y los objetivos de esta memoria.

CAPITULO 2: ***El puerto como sistema.*** Se explicará con todo detalle el funcionamiento de un puerto al completo. Conocimientos necesarios para la correcta comprensión del trabajo.

CAPITULO 3: ***Modelado y simulación.*** Explicación de la elección del modelo para llegar a los resultados esperados.

CAPITULO 4: ***Resultados.*** Presentación de algunos ejemplos y sus resultados.

CAPITULO 5: ***Conclusiones.***

CAPITULO 6: ***Bibliografía.***

CAPITULO 7: ***Anexos.***

2. El puerto como sistema.

En este capítulo se expondrá con todo detalle el funcionamiento de un puerto desde que el barco llega hasta que el camión exterior se lleva el contenedor del puerto o hasta que se lo lleva el barco.

El puerto puede plantearse como un sistema dinámico que tiene como entradas y salidas un flujo de contenedores. Como se puede observar en la Ilustración 2, existe un flujo bidireccional de contenedores en todas las zonas. Todas se alimentan de entrada y salida de contenedores, tanto en el mar con la descarga y carga de contenedores, como en el patio con el estacionamiento y salida de los mismo, como en la zona de tierra en la cual llegan contenedores de exportación y contenedores de importación que anteriormente llegaron en el barco para la salida de éstos.

Ilustración 2. Flujo de contenedores de un puerto.

Los contenedores, que llegan en el barco desde el puerto de origen, se encuentran ubicados en un emplazamiento determinado del barco que es conocido por el puerto de origen. Esta información es enviada, mediante internet, a los responsables de realizar los planes diarios del puerto de recepción de la mercancía, en nuestro caso el puerto de Santa Cruz de Tenerife, para la correcta colocación y ahorro de tiempo en cuanto a los movimientos se refiere. Una vez recibida toda la información, los trabajadores de la sección de mando, realizan el plan diario en el cual primero se prioriza la descarga de contenedores que llegan desde el puerto de origen y una vez terminada dicha descarga se comienza con la carga de contenedores que están a la espera de ser enviados desde Tenerife hacia otro puerto.

Estos planes diarios, se realizan teniendo en cuenta el número de contenedores a descargar, la situación donde se encuentran para no tener que realizar más movimientos de los necesarios, si el barco que arriba se trata de uno rodante (como es el caso de los coches) o de carga de contenedores, hasta la situación de los contenedores en el patio. El plan puede sufrir modificaciones una vez que se está realizando, por petición del capataz al mando y por aceptación del capitán del barco, que una vez que éste zarpa del puerto de origen, es él el responsable de toda la carga del barco. También se podrán tener en cuenta que se pueden sufrir modificaciones por acciones no previstas como son los accidentes laborales, las averías de la maquinaria, el movimiento del barco debido a un fuerte oleaje, etcétera.

2.1. Contenedores.

Según la Wikipedia [6]: “un contenedor es un recipiente de carga para el transporte, que protege a la mercancía de las condiciones climatológicas, fabricados según la ISO-668 en la cual se exponen las clasificaciones, dimensiones y rangos que han de tener los contenedores de forma estándar en pulgadas. Están principalmente fabricados de acero, aluminio y de algunos materiales sintéticos y pueden soportar cargas de hasta seis contenedores apilados encima sin que se deformen.”

Existen diferentes tipos de contenedores según la norma ISO 668 [1]:

- TEU (Twenty feet Equivalent Unit):
  - Longitud: 20 pies.
  - Ancho: 8 pies.
  - Altura: 8 pies.
  - Volumen: 32 metros cúbicos.
  - Peso propio: 24 kilo newton.
  - Carga máxima: 220 kilo newton.
  - Carga media: 120 kilo newton.
- 2TEU ó FEU (Forty feet Equivalent Unit):
  - Longitud: 20 pies.
  - Ancho: 8 pies.
  - Altura: 8 pies.
  - Volumen: 65 metros cúbicos.
  - Peso propio: 45 kilo newton.
  - Carga máxima: 270 kilo newton.
  - Carga media: 175 kilo newton.

También existen diferentes tipos de contenedores que no cumplen la norma ISO:

- Oversize. Longitud superior 40 pies.
- High Cube. Altura superior a 8 pies y 6 pulgadas.
- Overwidth. Ancho superior a 8 pies.
- Reefers. Refrigerados, contiene un equipo propio de generación y mantención de frío.
- Conair. Aislantes, mantiene el contenedor a una temperatura constante.

- Heated. Térmicos, contiene equipo propio de generación de y mantención de calor.
- Open Top. Techo de lona removible que permite la carga y descarga superior.
- Ventilated. Permite el intercambio de aire con el exterior.
- Platform. Presenta una única base.
- Flatrack. Presenta una configuración con una base y dos paredes.
- Dry bulk. Graneleros, contiene tomas superiores y la descarga se realiza por precipitación.
- Tank. Tanque, se suele utilizar para el transporte de productos químicos.

2.2. Zona mar-patio.

Esta zona es donde se realiza la mayor parte del trabajo del muelle. Una vez se tienen todos los movimientos a realizar mediante el plan y atraca el barco por la zona de mar, se comienza a trabajar en descargar los contenedores cuyo destino es el puerto. Para ello se utilizan las grúas de barco, denominadas grúas Post-Panamax [5]. Estas grúas están dirigidas por un operario encargado de descargar correctamente los contenedores, siempre siguiendo el plan, depositándolos en unos camiones, conducidos por otro operario, propiedad de la misma empresa que trabaja en el puerto y cuyos movimientos se limitan al puerto. Todos los movimientos de las grúas de barco están supervisados mediante cámaras transmitiendo en tiempo real por la sala de control del puerto, en la cual, si el operario tiene alguna duda contacta con dicha sala y ésta le soluciona el cómo proceder a solucionar ese problema. Puede ocurrir que el contenedor a descargar se encuentre desplazado o que no se encuentre en el lugar correspondiente, etcétera.

Ilustración 3. Grúa de barco descargando contenedor en maffi.

Existen algunos barcos que llegan con carga rodante, es decir, la carga a descargar son coches, para los cuales no es necesario las grúas de barco ni los maffis. Los coches se descargan

mediante una rampa de la que viene provisto el barco, y se colocan en la zona del patio asignada para este evento.

Los camiones internos se utilizan para llevar el contenedor desde la zona de mar hacia la zona de patio. Su finalidad, se basa en, una vez que le han cargado el contenedor llevarlo al bloque del patio asignado para ese contenedor. Estos camiones que se encuentran dentro del puerto se denominan Maffis.

Una vez el maffi llega con el contenedor al bloque del patio asignado, allí le espera una grúa de patio, en la cual se encuentra el operario correspondiente, que se encarga de descargar el contenedor del maffi y colocarlo de determinada forma en el bloque correspondiente. El contenedor, depositado en dicho bloque a la espera de que llegue su destinatario, ya sea otro barco u otra empresa externa para llevárselo hacia el exterior del puerto.

Ilustración 4. Maffi con contenedor hacia el patio.

Existen algunos contenedores que, por norma, no pueden tocar tierra. En este caso, los contenedores, con esta categoría especial se procede de diferentes formas, dejando pasar el camión exterior hasta la zona de las grúas Post-Panamax para que éstas lo descarguen

directamente en él y que ya pueda encaminarse hacia el exterior o colocándolo en un maffi y que cuando acabe el turno de estibadores volverlo a cagar al barco.

El proceso de descarga siempre es el primero que se realiza a la hora del atraque de un barco, mientras que el proceso de carga se realiza una vez haya terminado la descarga. El proceso de carga se basa en realizar los mismos pasos que la descarga en orden inverso para los contenedores cuyo destino sea un puerto de los que el barco tiene previsto atracar. Se colocan de tal forma que no interfieran en los contenedores que se encuentren destinados a otro puerto que no sea el de origen, debido a que los movimientos extras son una pérdida de tiempo y por lo tanto una pérdida de dinero.

Ilustración 5. Grúa de patio colocando contenedor en un camión.

2.3. Zona patio-tierra.

En la zona de pato se pueden encontrar contenedores de importación, los que llegan desde el puerto de origen, los de exportación, son los que están listos y preparados cargarlos al barco

que tienen como destino otro puerto, los “vacíos”, son lo que se encuentran vacíos con la finalidad de devolverlos al puerto de origen, frigoríficos, materiales peligrosos, etcétera. Todos estos contenedores se distribuyen de forma que no se produzca pérdidas de tiempo o retrasos en a la hora de realizar los trabajos de carga y descarga y de salida terrestre de los contenedores.

Existen diferentes técnicas para la correcta colocación de los contenedores en el patio, pero solo con la experiencia se logran adquirir los conocimientos para hacerlo correcta y ordenadamente. Se suelen apreciar que los contenedores de corta estancia en el puerto ya sea porque su salida será marítima, se colocan lo más cerca posible de la zona de mar, o porque su salida será terrestre que se colocarán en la zona más cercana a la salida exterior del puerto.

En cuanto a los de categorías especiales, se colocarán en las zonas especiales habilitadas para ello, ya sean los frigoríficos o los químicos o los de materiales peligrosos.

En el caso de que la carga sea rodante, como por ejemplo coches o camiones, se colocan directamente en una zona habilitada para este evento en concreto. Esta zona ha de ser bastante grande debido a que las cargas rodantes abarcan más espacio y no se pueden apilar.

En la zona de tierra es donde se encuentra la zona de espera de los camiones. El tiempo de espera se basa en el tiempo que se tarda desde que se expide la orden hasta que el camión se marcha de la zona de espera. Una vez que el camión llega se manda la orden para activar el proceso de llevado del contenedor específico hacia el camión que ha llegado. Para ello, los operarios de las grúas de patio se ponen a trabajar en sacar dicho contenedor del bloque en el que se encontraba para cargarlo en unos camiones con una grúa incorporada, denominados Reach Stackers. Estos camiones transportan el contenedor hacia la zona de espera dónde cargan el camión correspondiente. Una vez cargado los camiones ya emprenden su salida terrestre al exterior del puerto y los reach stackers continúan con otra operación.

3. Modelado y simulación.

3.1. Simulación de eventos discretos.

La simulación de eventos discretos es una técnica informática para modelar dinámicamente sistemas. Con esta técnica se modela el comportamiento de un sistema atendiendo a los instantes de tiempo en que ocurren cambios en el estado del sistema. A estos instantes se les denomina eventos y se asume que no ocurre ningún cambio en el estado del sistema entre dos eventos consecutivos. Cada evento ocurre en un instante de tiempo determinado y marca un cambio de estado en el sistema, por lo que la simulación puede saltar directamente en el tiempo de un evento al siguiente. La gran ventaja de este tipo de técnicas frente a la simulación continua es que no es necesario simular cada instante de tiempo, sino únicamente aquellos en los que ocurren hechos destacados, permitiendo una ejecución mucho más eficiente en un computador [4].

3.2. Modelo conceptual.

En este trabajo de fin de grado, nos hemos centrado en el modelado y simulación de las operaciones mar-patio. Para definir nuestro modelo, nos hemos basado en los resultados que se esperarían de este experimento. Definiendo todas las variables que se creían que podrían entrar en el análisis y los flujos de trabajo del experimento.

La Ilustración 6 representa de forma esquemática como diagrama de flujo, el funcionamiento de las operaciones mar-patio de un puerto como el de Santa Cruz de Tenerife.

Este modelo parte de la base de que los elementos a analizar su movimiento son los contenedores. De ahí, que cuando llegue el barco la primera actividad sería cuando el contenedor, con ayuda de una grúa de barco, se descarga desde el barco y se coloca en el maffi correspondiente. A esta actividad le sucede la siguiente actividad que sería lo que tarda el maffi en llegar desde la zona donde se encuentran las grúas de barco hasta los bloques de patio donde se encuentran los diferentes contenedores apilados. Con la ayuda de una grúa de patio ésta deposita el contenedor en el correspondiente bloque de patio que le haya tocado, a partir de dicha actividad, el maffi lo que hace es volver para liberar ese recurso y que vuelva a comenzar con otro contenedor de nuevo, eso corresponde a la última actividad de este diagrama de flujo inicialmente escogido por las necesidades de las actividades por las que atraviesa un contenedor.

En el caso de la actividad barco-camión, necesita un recurso tipo grúa de barco aparte de un recurso tipo camión. En la siguiente actividad, camión ida, se necesita un camión asignado para que esa actividad comience. Así, en la actividad camión-patio, utiliza un recurso del tipo grúa de patio al que se le añade un camión que ha de estar disponible. Y, por último, en la actividad camión vuelta e necesario asignarle un recurso del tipo camión.

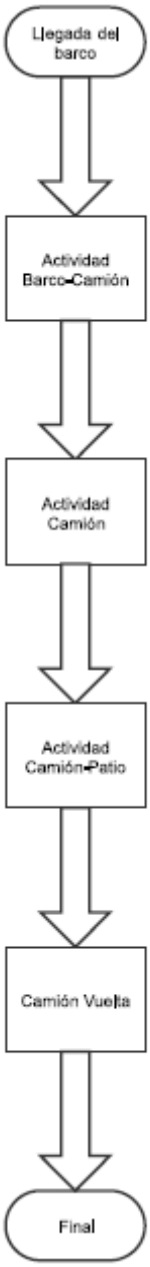

Ilustración 6. Flujo de trabajo del proyecto.

3.3. Implementación del modelo.

La implementación del modelo conceptual fue relativamente directa, con la excepción del uso del camión. Hay que tener en cuenta que el camión es necesario durante todas las tareas definidas en el modelo conceptual, por lo que se decidió añadir una actividad denominada "todo" que englobara a las cuatro actividades ya definidas. Cada actividad tiene unos recursos asignados, cuando estos recursos se encuentran disponibles, es cuando comienza la actividad solicitada que se llevará a cabo, en caso contrario, comenzará una cola de elementos esperando a que los recursos que se necesitan para realiza dicha actividad queden disponibles. Por este motivo se decide crear la actividad denominada "todo", la cual, no actúa activamente en el modelo. Cuyo objetivo es que cada inicio de ciclo se reserve un recurso del tipo camión y no lo devuelva hasta que la actividad acabe al finalizar cada ciclo.

Para implementar el modelo anteriormente descrito, se planteó el software Arena, pero existía el problema de la visión que se tenía para realizar la simulación. Se descartó debido a que dicho software acotaba a la hora de simular dicho sistema. Debido a esto, se opta por utilizar una herramienta de simulación de eventos discretos denominada PSIGHOS. Esta herramienta se basa en el lenguaje de programación Java que se ha utilizado con una interfaz de Eclipse para Java. Ha sido creada y diseñada por personal docente de la universidad de La Laguna, cuyo fin es crear una herramienta para la facilitación de la obtención de resultados a la hora de simular e implementar en Java un modelo para su posterior modelado.

Esta herramienta cuenta de forma predefinida con una serie de clases que simplifican las tareas de modelado. La facilidad de este simulador se basa en la definición de los diferentes elementos necesarios para el modelo y no la creación desde el inicio de todos los elementos necesarios.

En el modelo descrito en este documento, en el punto 3.2., se han utilizado las diferentes clases de este simulador, las cuales han simplificado el trabajo. La primera clase utilizada es la clase principal, la clase "Simulation", donde se expresa todo lo necesario para el modelaje de nuestro modelo. La clase "Experiment" se ha utilizado para expresar el número de experimentos que se van a realizar una vez se arranque la simulación, los listener que deben ser atendidos en las simulaciones y el main de la simulación, en nuestro caso.

En cuanto a la definición de los elementos del modelo, los elementos son la parte más importante de éste, ya que son las partes que interaccionan con el sistema. Los elementos son aquellos para los cuales se crean las actividades y los recursos para ayudar a realizar esas

actividades. En la clase “Element Type” se definen los tipos de elementos que existen en la simulación, en nuestro caso los elementos serán los contenedores.

```
44          //El unico tipo de elemento son los contenedores.
45          ElementType etcontenedores = new ElementType(this, CONTENEDOR);
```

Ilustración 7. Muestra de código donde se definen los elementos.

Los recursos son los mecanismos necesarios para que los elementos puedan realizar las actividades definidas en los modelos. Estos medios pueden ser materiales o humanos y pueden estar disponibles u ocupados si otra actividad los está utilizando en el mismo momento que otra actividad lo solicita para su utilización. Cada recurso individual se representa con un objeto de la clase “Resource”. Todos los recursos del mismo tipo se agrupan mediante la clase “ResourceType”. En nuestro caso, los tipos de recurso son las grúas de barco, las grúas de patio y los camiones interiores del puerto, los maffis. Para definir estos tipos de recursos con PSIGHOS, solo es necesario asignarles un nombre.

```
47          //Hay tres tipos de recursos.
48          ResourceType rtGruaBarco = new ResourceType(this, GRUA_BARCO);
49          ResourceType rtCamion = new ResourceType(this, CAMION);
50          ResourceType rtGruaPatio = new ResourceType(this, GRUA_PATIO);
```

Ilustración 8. Muestra de código en la que aparecen definidos los tipos de recursos.

Se define en el simulador una clase para crear grupos de trabajo el cual define los grupos de recursos que se necesitan para llevar a cabo las diferentes actividades. Cada actividad tiene un grupo de trabajo cada uno con sus recursos correspondientes asignados. En nuestro caso las actividades como camión ida y camión vuelta solo necesitan un grupo de trabajo de un recurso del tipo camión, mientras que en la actividad barco-camión se requiere un grupo de trabajo de un camión y una grúa de barco al igual que en la actividad camión-patio que se requiere un camión y una grúa de patio. Por lo tanto, definimos tres grupos de trabajo con la clase específica de dicha herramienta “Workgroup”. Se añade por último un grupo de trabajo vacío debido a lo explicado anteriormente, la actividad que engloba las demás actividades, al comenzar un ciclo de trabajo, reserva un recurso del tipo camión, por lo tanto, cuando llegue la actividad camión ida, el camión ya se encuentra reservado, por lo que, solo debe transcurrir el tiempo que tarde en realizarse esa actividad. Entonces, a la actividad camión ida, se le relaciona con el grupo de trabajo vacío ya que, si no, se le asignarían dos recursos del tipo camión.

```
79          //Creamos los grupos de trabajo.
80          WorkGroup wgBarcoCamion = new WorkGroup(this, new ResourceType[] {rtGruaBarco}, new int[] {1});
81          WorkGroup wgCamion = new WorkGroup(this, rtCamion, 1);
82          WorkGroup wgCamionPatio = new WorkGroup(this, new ResourceType[] {rtGruaPatio}, new int[] {1});
83          WorkGroup wgVacio = new WorkGroup(this);
```

Ilustración 9. Muestra de código en el que aparecen la definición de los grupos de trabajo.

Las actividades son las acciones que se realizan con los elementos con los recursos como ayuda material o humana. A estas se le asignan alguno o varios de los diferentes grupos de trabajos creados. A las actividades se les define mediante estos factores nombrados, pero se le debe añadir el tiempo que tarda en realizarse la tarea asignada para la cual se creó dicha actividad. Para crear las actividades utilizando la herramienta PSIGHOS se utiliza la clase "Activity". En nuestro proyecto las actividades son las mencionadas en el punto anterior (3.2. Modelo conceptual) barco-camión, camión ida, camión-patio, camión vuelta.

Esta herramienta, a la hora de conectar los pasos a seguir en nuestro grupo de trabajo se utiliza el método "link", con este método se consiguen crear el flujo de trabajo definiendo el inicio de dicho flujo.

La clase "ElementGenerator" se utiliza para crear los elementos que van a interactuar en la simulación. En nuestro caso, utilizamos esta clase como metáfora del atraque del barco, con los contenedores que el usuario tenga que descargar, al puerto para comenzar con el flujo de trabajo de los diferentes contenedores.

En el caso del puerto, los estibadores ("Trabajador que se ocupa en la carga y descarga de un buque u otro medio de transporte y distribuye convenientemente los pesos en él." Según la RAE [8]), tienen turnos por ley de 6 horas diarias, por ello, se ha creado para este caso un ciclo diario de 6 horas que comienza a las 6:00 am y finaliza a las 12:00. Para poder llevar a cabo este ciclo se utiliza una herramienta propia de PSIGHOS que se denomina: "SimulationPeriodicCycle" y que se puede configurar para que el ciclo de trabajo sea el deseado, la unidad de tiempo en la que se va a medir éste y las veces que se va a repetir este ciclo.

Una vez plasmado el modelo en la simulación, se necesitarán conocer los diferentes resultados, para ello se hacen los denominados listeners, que son utilizados con dicho fin. Estos listeners han ido cubriendo las necesidades a medida que el proyecto ha ido evolucionando. En un primer momento el listener necesario era para conocer los tiempos que tarda cada contenedor en realizar cada actividad, así como cuánto tarda cada actividad en realizar su tarea. Esto se ha utilizado para continuar avanzando en el proyecto en la medida de cambiar tiempos más reales,

los cuales siguen siendo fijos, aumentar la complejidad, etc. El listener final nos muestra los tiempos por contenedor de inicio y fin de cada actividad.

Para finalizar, se han realizado diferentes ventanas y gráficos para la mejora de la visualización y la mejora de la interfaz con el usuario de manera que sea más fácil para la comprensión de los resultados, de manera que no haga necesaria tener experiencia en programación ni en Java. Estas ventanas se describen en el siguiente apartado.

3.4. Herramientas para la toma de decisiones.

Una vez conocidos los tiempos de cada elemento en cada actividad, es posible usar esta información para ayudar a la toma de decisiones. Para ello se ha dispuesto de una interfaz de usuario, la cual, se basa en una serie de ventanas programadas en Java. La primera de ellas ayuda a la programación debido a que son los datos que requiere el problema para conocer los tiempos finales.

En esta primera ventana, el programa le pide al usuario que introduzca el número de datos que requiere el problema, por lo que el usuario debe introducir los contenedores que llegan en el barco y se desean descargar en el puerto, el número de maffis disponibles en ese momento en el puerto, el número de grúas de barco que se le decide colocar a ese barco, normalmente suelen ser una o dos teniendo en cuenta las dificultades que conlleva tener disponibles más de una grúa en un barco y por último introducir el número de grúas de patio disponible en os diferentes bloques del patio para descargar los camiones en ellos.

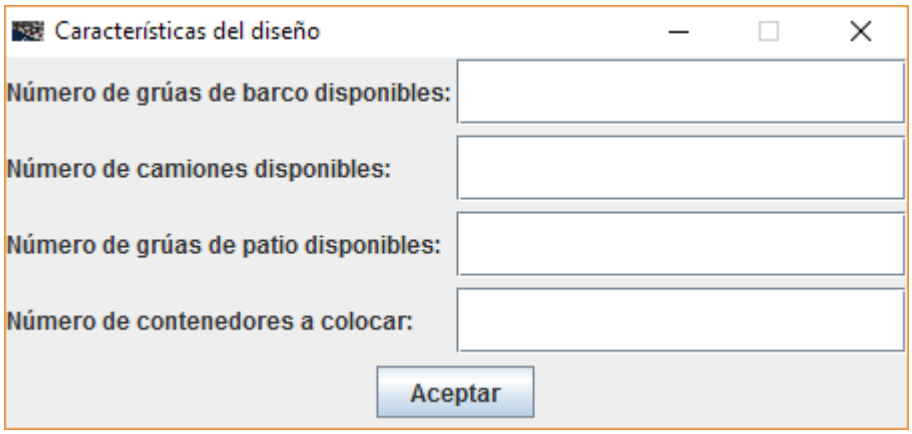

Ilustración 10. Interfaz de usuario 1.

A partir de la introducción de esos datos, aparece otra ventana para la confirmación de los datos del diseño, en la cual, se pueden confirmar los datos del diseño o, en caso de error, cancelar y volver a introducir los datos de nuevo.

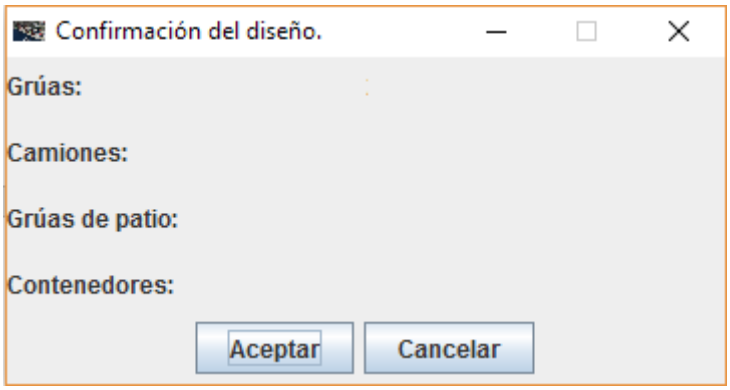

Ilustración 11. Interfaz de usuario 2.

Una vez se conozcan estos datos, el programa comienza su funcionamiento. Simula el modelo que se ha descrito anteriormente para conocer los diferentes tiempos que tardan los elementos en realizar las actividades si no ocurriera ningún imprevisto. Para ello, se muestra por pantalla de nuevo otra ventana que contiene los datos del programa, añadiendo el tiempo que se tardó en realizar todo el trabajo para esos datos, es decir, el tiempo que la simulación ha tardado en realizar el flujo de trabajo completo para todos los elementos y con todos los recursos disponibles. Se añade también dos opciones, las cuales se pueden marcar si se quiere conocer el gráfico de elementos que han entrado en conflicto en alguno de los momentos, ya que no han

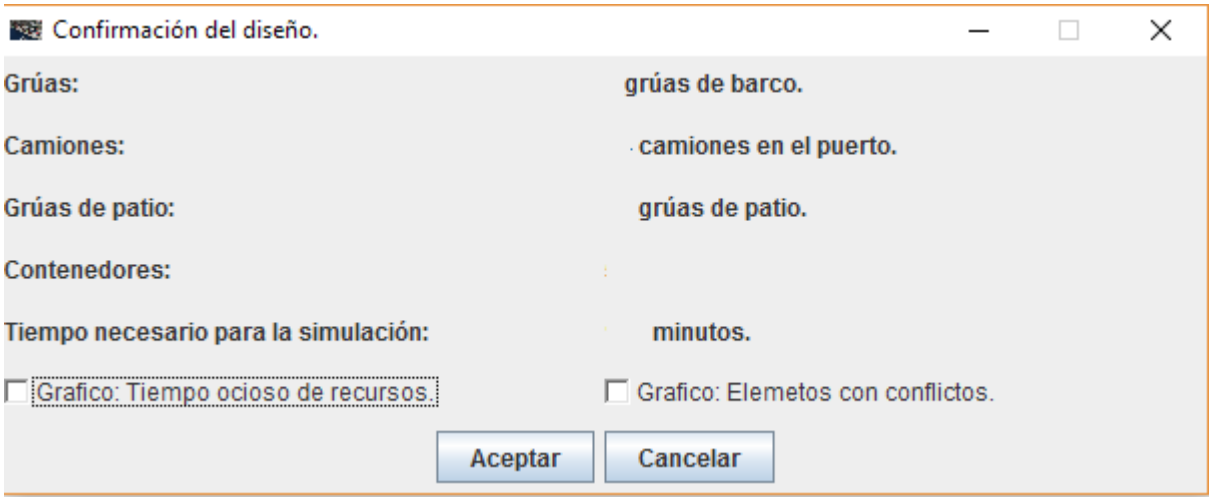

Ilustración 12. Interfaz de usuario 3.

tenido recursos para realizar las actividades y, por otro lado, se puede conocer el gráfico del tiempo ocioso de los diferentes recursos en el caso de que lo hubiera.

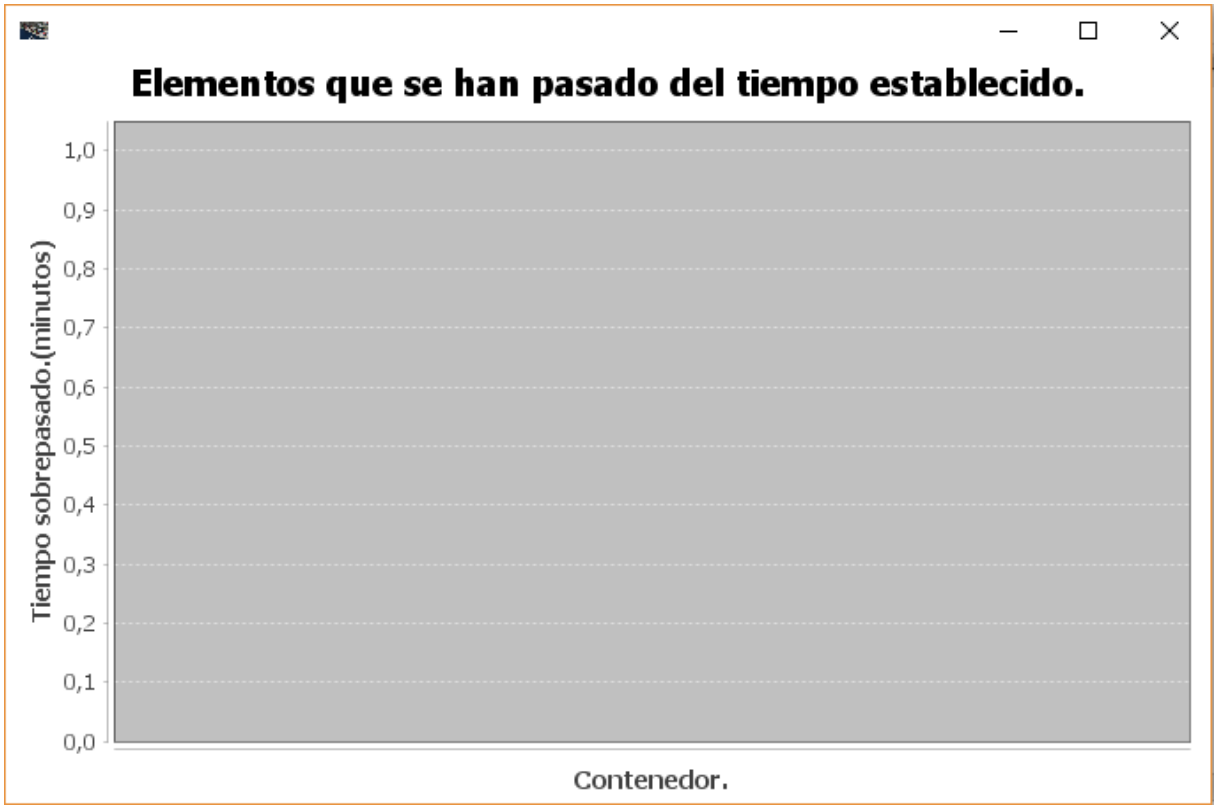

Ilustración 13. Ejemplo de gráfico de tiempo de conflicto de elementos.

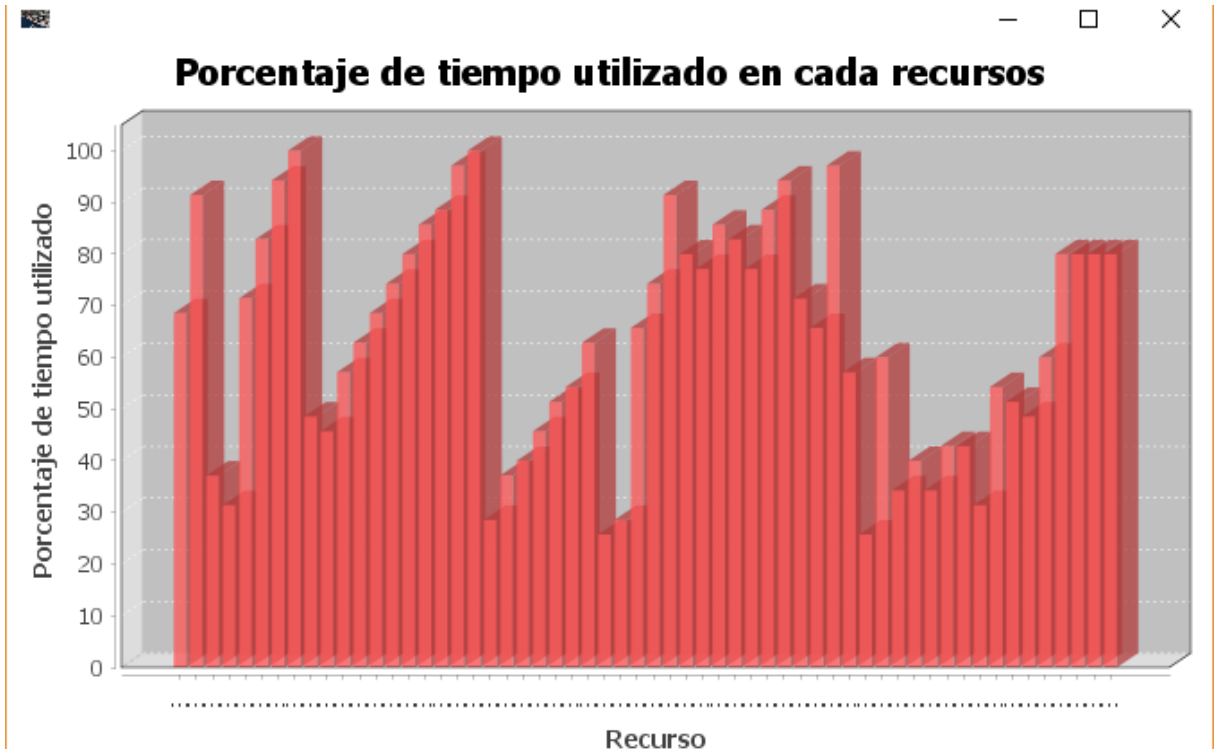

Ilustración 14. Ejemplo de gráfico de tiempo ocioso de los recursos.

Estos son unos ejemplos de los diferentes gráficos en los cuales se muestra la información deseada por el usuario, aparte de eso, en la consola de Eclipse se muestran los tiempos que tarda cada elemento en realizar cada actividad y en el momento que empieza a realizarla y el minuto en que la finaliza.

4. Resultados.

Para comprobar el correcto funcionamiento de la simulación y la interfaz desarrolladas, se han creado algunos ejemplos de casos de estudio en el puerto.

El primer ejemplo hemos optado por diseñar un pequeño ejemplo de la forma más real que se conoce para intentar adecuarnos a la realidad. Para ello se ha escogido que el barco llega con 60 contenedores para descargar, y tenemos disponibles una grúa de barco, 6 maffis y dos grúas de patio en el bloque correspondiente. Con estos datos, se ha obtenido con la simulación que se tardan en descargar este número de contenedores 375 minutos.

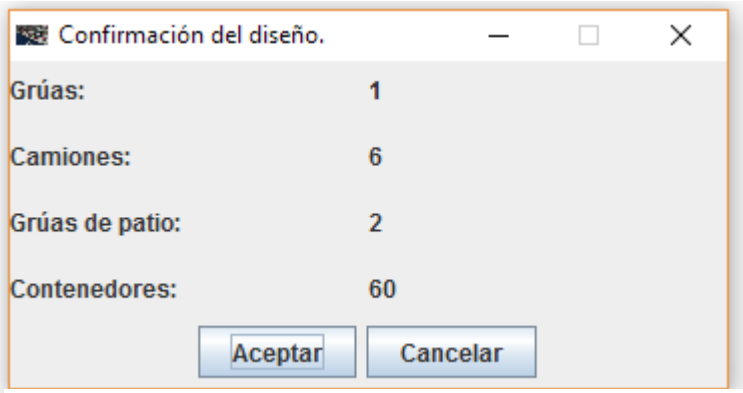

Ilustración 15. Ejemplo 1.

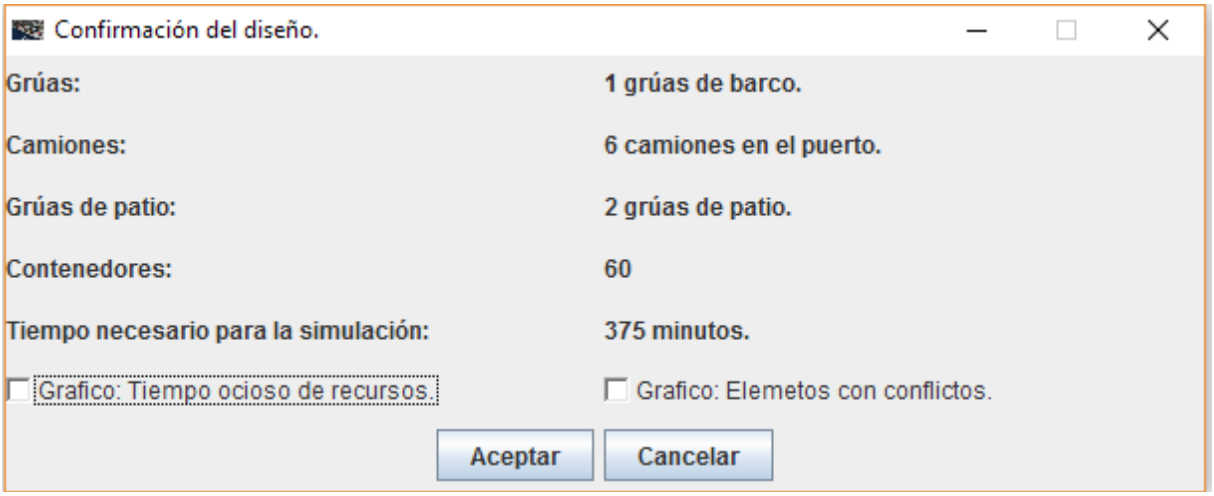

Ilustración 16. Ejemplo 1.

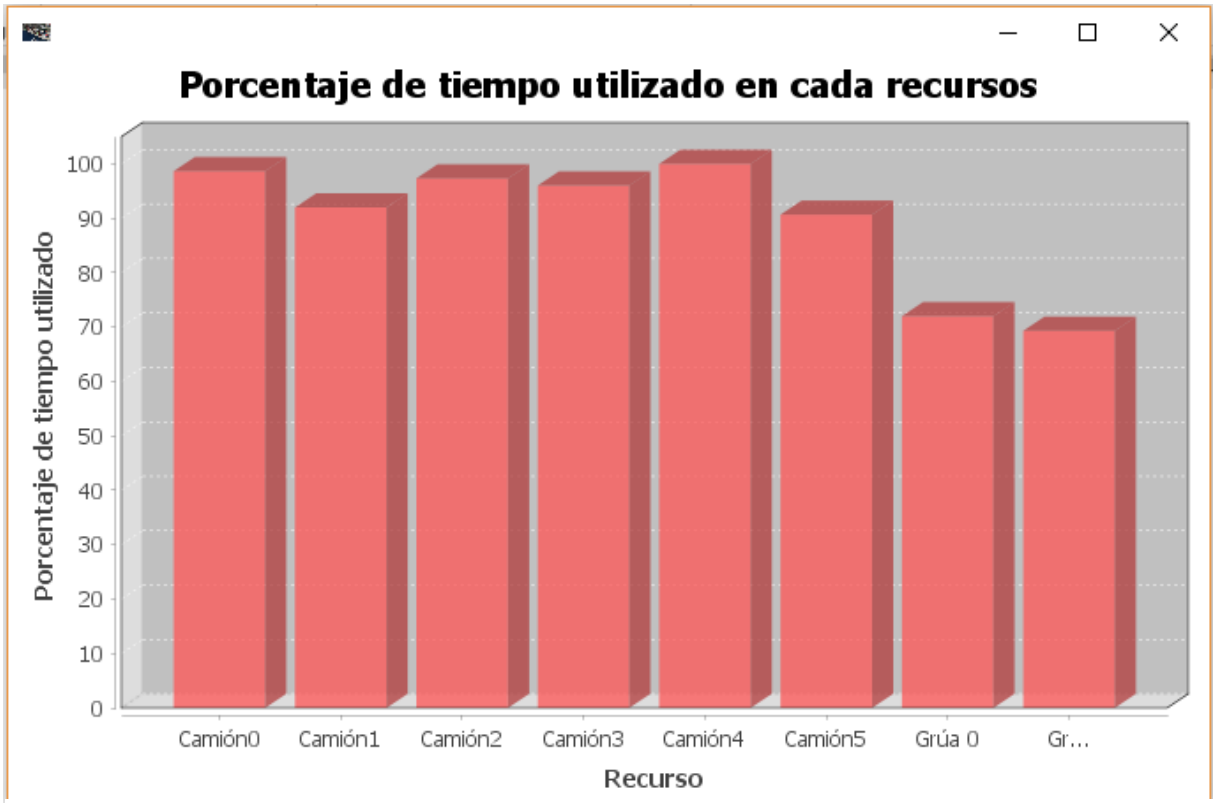

Ilustración 177. Ejemplo 1.

En la Tabla 1 se puede observar el tiempo que ha tardado cada contenedor en cada actividad, en la primera columna se observa el identificador de cada contenedor. El tiempo de la Tabla 1 se mide en minutos. Mientras que en el resto de columnas se muestra el tiempo inicial y final de cada una de las diferentes actividades. En la segunda columna se muestr el tiempo inicial de cada contenedor en la actividad barco-camión, así como la siguiente columna se muestra la finalización de la misma actividad. La siguiente columna muestra el tiempo en el que cada contenedor inicia la actividad camión ida para que la siguiente columna muestre la finalización de la mismay, por último se muestra el tiempo de iniciación y finalización de los contenedores en la actividad camión-patio. Los tiempos en los cuales la casilla aparecce un “No terminado”, significa que, ese contenedor pasado el tiempo establecido por turno de estibadores (6 horas) no ha logrado completar esas actividades por lo que esos contenedores quedarían dispuestos a restarle el tiempo correspondiente al siguiente turno de estibadores.

Contenedor	TiempoIni_BC	TiempoFinBC	TiempoIni_CI	TiempoFinCI	TiempoIni_CP	TiempoFinCP
0	345	350	350	365	No terminado	No terminado
5	15	20	20	35	35	40
6	25	30	30	45	45	50
7	250	255	255	270	270	275
8	340	345	345	360	No terminado	No terminado
10	330	335	335	350	350	355
12	320	325	325	340	340	345
13	305	310	310	325	325	330
14	300	305	305	320	320	325
15	295	300	300	315	315	320
16	290	295	295	310	310	315
17	285	290	290	305	305	310
18	280	285	285	300	300	305
19	265	270	270	285	285	290
20	260	265	265	280	280	285
21	5	10	10	25	25	30
22	175	180	180	195	195	200
23	225	230	230	245	245	250
24	245	250	250	265	265	270
25	255	260	260	275	275	280
26	220	225	225	240	240	245
27	215	220	220	235	235	240
28	210	215	215	230	230	235
29	335	340	340	355	355	360
30	200	205	205	220	220	225
31	185	190	190	205	205	210
32	180	185	185	200	200	205
33	325	330	330	345	345	350
34	170	175	175	190	190	195
35	165	170	170	185	185	190
36	160	165	165	180	180	185
37	145	150	150	165	165	170
38	140	145	145	160	160	165
39	135	140	140	155	155	160
40	130	135	135	150	150	155
41	125	130	130	145	145	150
42	120	125	125	140	140	145
43	105	110	110	125	125	130
44	100	105	105	120	120	125
45	95	100	100	115	115	120
46	90	95	95	110	110	115
47	85	90	90	105	105	110
48	80	85	85	100	100	105
49	65	70	70	85	85	90
50	60	65	65	80	80	85
51	55	60	60	75	75	80
52	50	55	55	70	70	75
53	40	45	45	60	60	65
54	45	50	50	65	65	70
55	205	210	210	225	225	230
56	10	15	15	30	30	35
57	240	245	245	260	260	265
58	20	25	25	40	40	45
59	0	5	5	20	20	25

Tabla 1. Ejemplo 1.

Para el ejemplo 2, se ha diseñado aumentando el número de recursos y disminuyendo el número de contenedores para conseguir que todos los contenedores acaben todas las actividades en el tiempo establecido. Se han aumentado el número de grúas de barco hasta disponer de 3, el número de camiones hasta 6 y el número de grúas de patio que se tienen disponibles 2; todo esto para 48 contenedores.

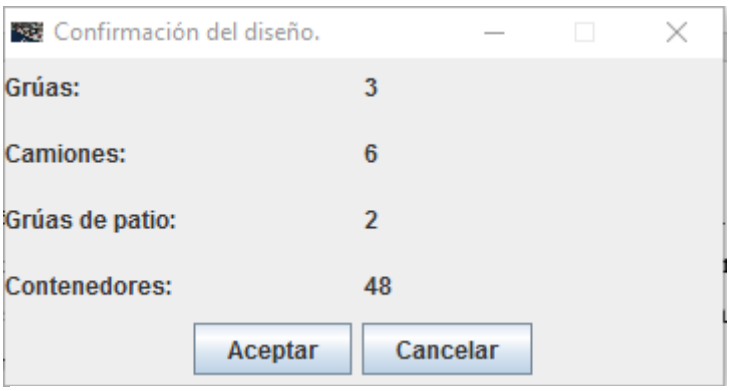

Ilustración 1818. Ejemplo 2.

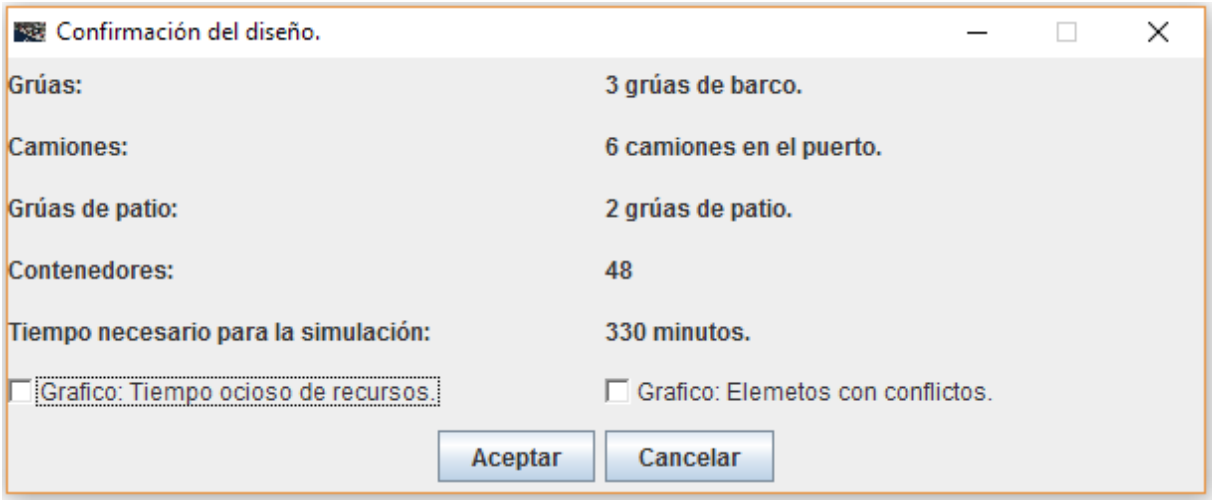

Ilustración 19. Ejemplo 2.

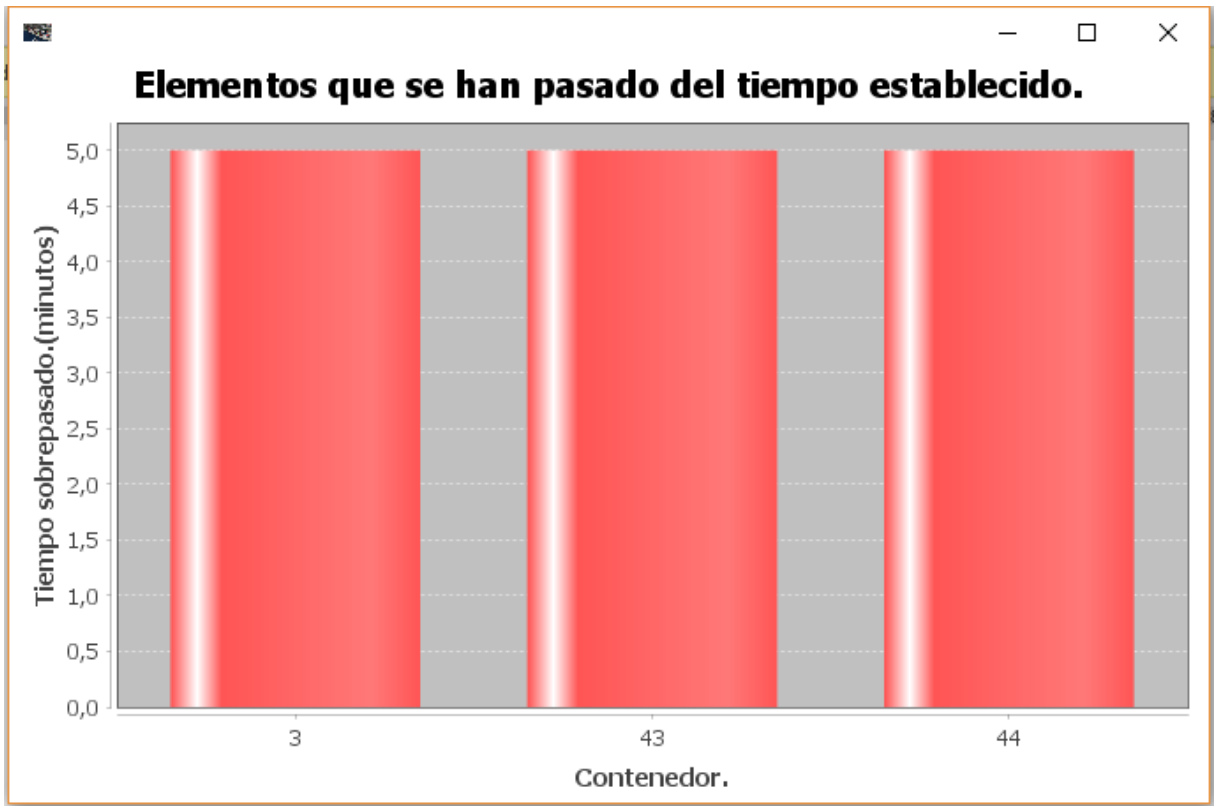

Ilustración 20. Ejemplo 2.

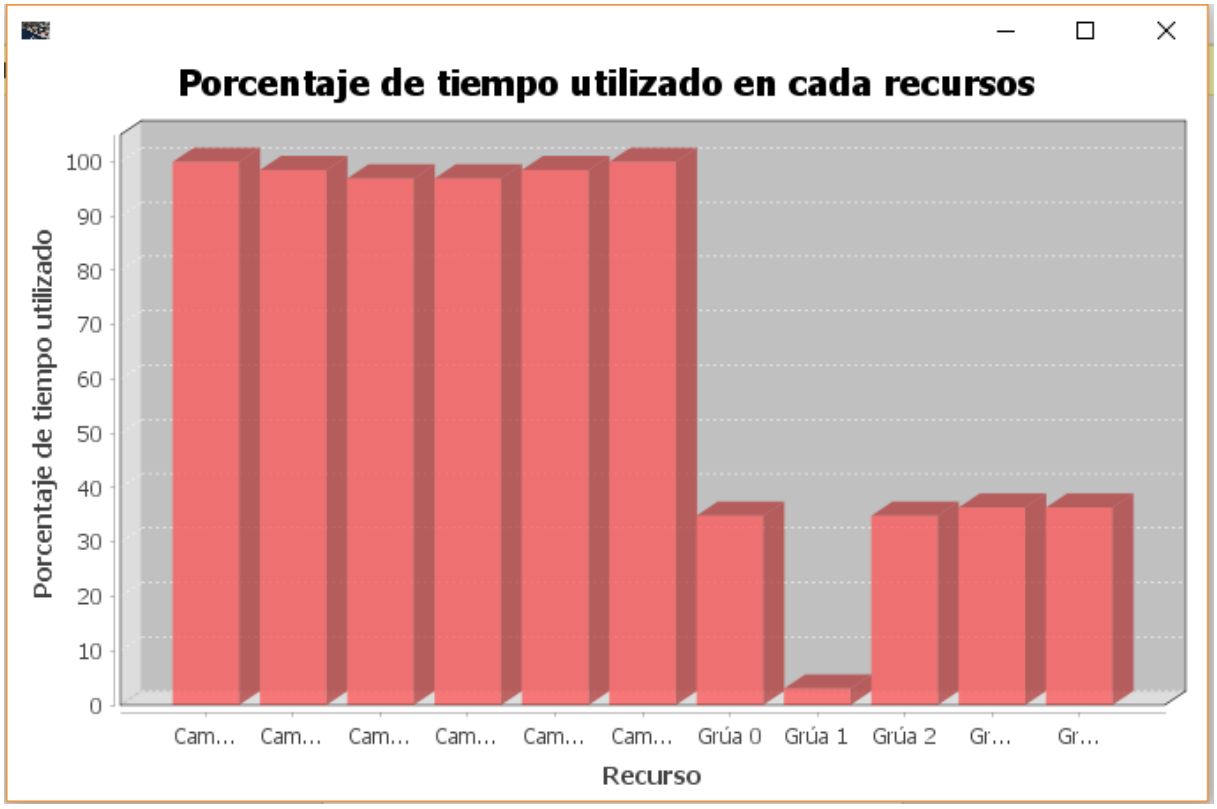

Ilustración 21. Ejemplo 2.

Tabla 2. Ejemplo2.

Contenedor	TiempoIni_BC	TiempoFinBC	TiempoIni_CI	TiempoFinCI	TiempoIni_CP	TiempoFinCP
0	290	295	295	310	310	315
1	290	295	295	310	310	315
2	285	290	290	305	305	310
3	5	10	10	25	30	35
4	90	95	95	110	110	115
5	170	175	175	190	190	195
6	280	285	285	300	300	305
7	280	285	285	300	300	305
8	250	255	255	270	270	275
9	285	290	290	305	305	310
10	245	250	250	265	265	270
11	240	245	245	260	260	265
12	240	245	245	260	260	265
13	210	215	215	230	230	235
14	210	215	215	230	230	235
15	205	210	210	225	225	230
16	205	210	210	225	225	230
17	200	205	205	220	220	225
18	200	205	205	220	220	225
19	125	130	130	145	145	150
20	160	165	165	180	180	185
21	165	170	170	185	185	190
22	165	170	170	185	185	190
23	170	175	175	190	190	195
24	160	165	165	180	180	185
25	130	135	135	150	150	155
26	130	135	135	150	150	155
27	250	255	255	270	270	275
28	125	130	130	145	145	150
29	120	125	125	140	140	145
30	120	125	125	140	140	145
31	245	250	250	265	265	270
32	90	95	95	110	110	115
33	85	90	90	105	105	110
34	85	90	90	105	105	110
35	80	85	85	100	100	105
36	80	85	85	100	100	105
37	50	55	55	70	70	75
38	50	55	55	70	70	75
39	45	50	50	65	65	70
40	45	50	50	65	65	70
41	40	45	45	60	60	65
42	40	45	45	60	60	65
43	5	10	10	25	30	35
44	0	5	5	20	25	30
45	0	5	5	20	20	25
46	5	10	10	25	25	30
47	0	5	5	20	20	25

En el ejemplo 2, se puede observar que los contenedores cuya identificación son el número 3, el 43 y el 44, se sobrepasan en algún momento 5 minutos de lo que deberían. Observando la Tabla 2, igualmente expresada que la Tabla 1, se nota que el contenedor número 3, al finalizar la actividad camión ida, espera 5 minutos para realizar la siguiente actividad, nótese el conflicto que se provoca en ese momento. En los demás contenedores con conflictos ocurre exactamente lo mismo, se retrasan 5 minutos en el inicio de la actividad camión-patio. Esto se muestra en el gráfico de conflictos (Ilustración 20), ya que dichos contenedores se observan 5 minutos de retraso a la hora de acabar las actividades. En la Ilustración 21 se puede observar que la grúa 1 se utiliza un porcentaje muy bajo y que las otras dos grúas se reparten todo el trabajo.

5. Conclusiones.

- Se ha logrado comprender el funcionamiento de carga y descarga de un puerto, así como sus inconvenientes a la hora de realizar dichos trabajos a lo largo de la jornada laboral. Se ha estudiado la forma de resolver estos inconvenientes de la manera más adecuada e intentando que la simulación sirva en el caso de ruptura de la planificación.
- Se ha aprendido a utilizar un nuevo lenguaje de programación, Java.
- Se ha aprendido a utilizar la herramienta de simulación PSIGHOS, basada en Java, tratando de emplearla de la forma más productiva posible para conseguir modelar y simular el problema planteado.
- Se ha podido aprender a modelar, siguiendo todos los pasos que éste conlleva, desde realizar el modelo hasta poder simularlo.
- Se ha creado un primer modelo de la zona mar-patio del puerto, que puede utilizarse para representar las operaciones de descarga de uno o más barcos.
- Se ha creado un prototipo de una herramienta de ayuda a la toma de decisiones, que podría ayudar a los gestores del puerto.
- Se ha realizado una base de un proyecto que se puede empezar a utilizar, pero que sin duda se pudiera ampliar para que reprodujera fielmente la realidad que surge tanto en el puerto de Santa Cruz de Tenerife, como en el resto de puertos del mundo.

5.1. Conclusions.

- I have understood the loading and unloading operations of a port, as well as its troubles when carrying out such work throughout the working day. I have studied how to solve these problems in the most appropriate way and I have tried to make the simulation works in the event of a breakdown of planning.
- I have learnt a new programming language, Java.
- I have learnt how to use PSIGHOS, a java-based simulation tool; and I have tried to use it productively to model and simulate the posed problem.
- I have learnt to model, following all the steps that it entails, from making the model to being able to simulate it.
- I have created a first model of the sea-yard area of the port, which could be employed to represent the unloading operations of one or more vessels.
- I have created a decision-aid tool prototype, which could help the port managers.
- I have made a base for a project that can be used already, but which could be expanded to reproduce accurately the reality that appear in the Santa Cruz de Tenerife's port and in the rest of ports around the world.

6. Bibliografía.

[1].https://upcommons.upc.edu/handle/2099.1/5906

[2].http://dinamica-de-sistemas.com/revista/0608o.htm

[3].https://es.wikipedia.org/wiki/Puerto

[4].https://en.wikipedia.org/wiki/Discrete_event_simulation

[5].https://es.wikipedia.org/wiki/Gr%C3%BAa_p%C3%B3rtico_para_contenedores

[6].https://es.wikipedia.org/wiki/TEU_(unidad_de_medida)

[7].http://www.rae.es/

[8].https://stackoverflow.com/

[9].https://definicion.de/puerto/

7. Anexos.

Anexo I. Código de la simulación.

```
package puerto2;
import javax.swing.JFrame;

import es.ull.iis.simulation.model.Experiment;
import es.ull.iis.simulation.model.Simulation;
import es.ull.iis.simulation.model.TimeUnit;
import infoReceiver.InfoActividad2;
import infoReceiver.InfoTiempoFinal;

public class PrototipoBasicoMainPrueba extends Experiment {
   private int nGruas;
   private int nCamiones;
   private int nGruasPatio;
   private int nContenedores;

   public PrototipoBasicoMainPrueba(int  nExperiments, int nGruas, int
nCamiones, int nGruasPatio, int nContenedores) {
      super("Experimentos prototipo Basico", nExperiments);
      this.nGruas = nGruas;
      this.nCamiones = nCamiones;
      this.nGruasPatio = nGruasPatio;
      this.nContenedores = nContenedores;
   }

   public Simulation getSimulation(int ind) {

      Simulation sim = new PrototipoBasicoPrueba(ind, "Puerto" + ind,
TimeUnit.MINUTE, 0, 365 * 24 * 60, nGruas, nCamiones, nGruasPatio,
nContenedores);
      sim.addInfoReceiver(new InfoTiempoFinal(nGruas, nCamiones,
nGruasPatio, nContenedores));
      sim.addInfoReceiver(new InfoActividad2());
      return sim;
   }

   public static void main(String[] args){
      JFrame v = new VentanaPrueba();
      v.setVisible(true);
   }

}
```

A1.1. Archivo “PrototipoBasicoMainPrueba.java”.

```
package puerto2;
import es.ull.iis.simulation.model.ElementType;
import es.ull.iis.simulation.model.Resource;
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.SimulationTimeFunction;
import es.ull.iis.simulation.model.TimeDrivenElementGenerator;
import es.ull.iis.simulation.model.TimeUnit;
import es.ull.iis.simulation.model.WorkGroup;
import es.ull.iis.simulation.model.flow.ActivityFlow;
import es.ull.iis.simulation.model.flow.ReleaseResourcesFlow;
import es.ull.iis.simulation.model.flow.RequestResourcesFlow;


public class PrototipoBasicoPrueba extends Simulation {
   static final long T_DESCARGA = 5L;
   static final long T_CAMION = 15L;
   static final long T_D_PATIO = 5L;
   static final String CONTENEDOR = "Contenedor";
   public static final String GRUA_BARCO = "Grua Barco";
   public static final String GRUA_PATIO = "Grua Patio";
   static final String REQ_CAMION = "Reserva Camión";
   static final String REL_CAMION = "Libera Camión";
   public static final String CAMION = "Camión";
   static final String GRUA = "Grúa ";
   public static final String DESCARGA = "Descarga el contenedor del barco
hasta el camión";
   public static final String COMPLETO = "Completo";
   public static final String CAMION_PATIO = "Descarga del contenedor en el
bloque de Patio";
   public static final String CAMION_IDA = "Camión ida";
   public static final String CAMION_VUELTA = "Camión vuelta";
   private int nGruas;
   private int nCamiones;
   private int nGruasPatio;
   private int nContenedores;

   public PrototipoBasicoPrueba(int id, String description, TimeUnit unit,
long startTs,
         long endTs, int nGruas, int nCamiones, int nGruasPatio, int
nContenedores) {
      super(id, description, unit, startTs, endTs);
      this.nGruas = nGruas;
      this.nCamiones = nCamiones;
      this.nGruasPatio = nGruasPatio;

      //El unico tipo de elemento son los contenedores.
      ElementType etcontenedores = new ElementType(this, CONTENEDOR);

      //Hay tres tipos de recursos.
      ResourceType rtGruaBarco = new ResourceType(this, GRUA_BARCO);
      ResourceType rtCamion = new ResourceType(this, CAMION);
      ResourceType rtGruaPatio = new ResourceType(this, GRUA_PATIO);

      //Ciclo diario de todos los recursos.
      SimulationPeriodicCycle DailyCycle = new
SimulationPeriodicCycle(TimeUnit.MINUTE, 360, new
SimulationTimeFunction(TimeUnit.MINUTE, "ConstantVariate", 24*60), 1);

      Resource[] resGruasBarco = new Resource[nGruas];
```

```
     for (int i = 0; i < nGruas; i++) {
       resGruasBarco[i] = new Resource(this, GRUA + i);
       resGruasBarco[i].addTimeTableEntry(DailyCycle, 6 * 60,
rtGruaBarco);//Asignamos los roles a cada recurso
     }

     Resource[] resCamion = new Resource[nCamiones];
     for (int i = 0; i < nCamiones; i++) {
       resCamion[i] = new Resource(this, CAMION + i);
       resCamion[i].addTimeTableEntry(DailyCycle, 6 * 60, rtCamion);
     }

     Resource[] resGruaPatio = new Resource[nGruasPatio];
     for (int i = 0; i < nGruasPatio; i++) {
       resGruaPatio[i] = new Resource(this, GRUA_PATIO + i);
       resGruaPatio[i].addTimeTableEntry(DailyCycle, 6 * 60, rtGruaPatio);
     }

     //Actividades a realizar.
     ActivityFlow actBarcoCamion = new ActivityFlow(this, DESCARGA);
     ActivityFlow actCamion = new ActivityFlow(this, CAMION_IDA);
     ActivityFlow actCamionPatio = new ActivityFlow(this, CAMION_PATIO);
     ActivityFlow actCamionVuelta = new ActivityFlow(this, CAMION_VUELTA);

     //Creamos los grupos de trabajo.
     WorkGroup wgBarcoCamion = new WorkGroup(this, new ResourceType[]
{rtGruaBarco}, new int[] {1});
     WorkGroup wgCamion = new WorkGroup(this, rtCamion, 1);
     WorkGroup wgCamionPatio = new WorkGroup(this, new ResourceType[]
{rtGruaPatio}, new int[] {1});
     WorkGroup wgVacio = new WorkGroup(this);

     //Asignamos tiempo a las actividades.
     //Y linkamos las actividades creando nuestro flujo.
     actBarcoCamion.addWorkGroup(0, wgBarcoCamion, T_DESCARGA);
     actCamion.addWorkGroup(0, wgVacio, T_CAMION);
     actCamionVuelta.addWorkGroup(0, wgVacio, T_CAMION);
     actCamionPatio.addWorkGroup(0, wgCamionPatio, T_D_PATIO);

     actBarcoCamion.link(actCamion);
     actCamion.link(actCamionPatio);
     actCamionPatio.link(actCamionVuelta);

     //Hacemos un request que reserva un camion hasta el final del ciclo
que con un release lo liberamos
     RequestResourcesFlow reqCamion = new RequestResourcesFlow(this,
REQ_CAMION, 1);
     reqCamion.addWorkGroup(wgCamion);
     ReleaseResourcesFlow relCamion = new ReleaseResourcesFlow(this,
REL_CAMION, 1);

     reqCamion.link(actBarcoCamion);
     actCamionVuelta.link(relCamion);


     //Generador de elemetos.
     TimeDrivenElementGenerator gen = new TimeDrivenElementGenerator(this,
nContenedores, etcontenedores, reqCamion, DailyCycle);

   }
}
```

A1.2. Archivo “PrototipoBasicoPrueba.java”.

```
package puerto2;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import java.awt.Toolkit;



public class VentanaPedirDatos extends JFrame {

   public VentanaPedirDatos(){
      super("Características del diseño");

   setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\el_ma\\Desk
top\\tct_2_v2.jpg"));
      setLocationRelativeTo(null);
      setResizable(false);
      setType(Type.POPUP);
      setSize(456, 214);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

      //Panel de las etiquetas y de los cuadros de texto con forma de
rejilla
      JPanel panelGruas = new JPanel();
      GridLayout gl = new GridLayout(4, 2, 0, 5);
      JTextField texto2 = new JTextField(10);
      JTextField texto3 = new JTextField(10);
      JTextField texto4 = new JTextField(10);
      panelGruas.setLayout(gl);
      panelGruas.add(new JLabel("Número de grúas de barco disponibles:"));
      JTextField texto1 = new JTextField(10);
      panelGruas.add(texto1);
      panelGruas.add(new JLabel("Número de camiones disponibles:"));
      panelGruas.add(texto2);
      panelGruas.add(new JLabel("Número de grúas de patio disponibles:"));
      panelGruas.add(texto3);
      panelGruas.add(new JLabel("Número de contenedores a colocar:"));
      panelGruas.add(texto4);

      //Panel del boton aceptar
      JPanel panelBoton = new JPanel();
      panelBoton.setLayout(new FlowLayout());
      JButton botonAceptar = new JButton("Aceptar");
      panelBoton.add(botonAceptar);

      botonAceptar.addActionListener(new ActionListener(){
         public void actionPerformed(ActionEvent e){
            VentanaConfirmacionDatos Ventana2 = new
VentanaConfirmacionDatos(texto1, texto2, texto3, texto4);
            Ventana2.setVisible(true);
            CloseFrame();
```

```
      }
   });

   Container cp = getContentPane();
   cp.add(panelGruas, BorderLayout.CENTER);
   cp.add(panelBoton, BorderLayout.SOUTH);

}

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

A1.3. Archivo “VentanaPedirDatos.java”.

```
package puerto2;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import java.awt.Toolkit;

public class VentanaConfirmacionDatos extends JFrame {

   public JTextField cuadrotexto1;
   public JTextField cuadrotexto2;
   public JTextField cuadrotexto3;
   public JTextField cuadrotexto4;


   public VentanaConfirmacionDatos(JTextField cuadroTexto1, JTextField
cuadroTexto2, JTextField cuadroTexto3, JTextField cuadroTexto4) {
      super("Confirmación del diseño.");

   setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\el_ma\\Desk
top\\tct_2_v2.jpg"));
      setSize(364, 192);
      setLocationRelativeTo(null);
      setResizable(false);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

      this.cuadrotexto1 = cuadroTexto1;
      this.cuadrotexto2 = cuadroTexto2;
      this.cuadrotexto3 = cuadroTexto3;
      this.cuadrotexto4 = cuadroTexto4;

      String tx1 = cuadroTexto1.getText();
      String tx2 = cuadroTexto2.getText();
      String tx3 = cuadroTexto3.getText();
      String tx4 = cuadroTexto4.getText();

      //Panel de pedidos.
      JPanel panelPedidos= new JPanel();
      GridLayout gl = new GridLayout(4, 2, 0, 5);
      panelPedidos.setLayout(gl);
      panelPedidos.add(new JLabel("Grúas:"));
      panelPedidos.add(new JLabel(tx1));
      panelPedidos.add(new JLabel("Camiones:"));
      panelPedidos.add(new JLabel(tx2));
      panelPedidos.add(new JLabel("Grúas de patio:"));
      panelPedidos.add(new JLabel(tx3));
      panelPedidos.add(new JLabel("Contenedores:"));
      panelPedidos.add(new JLabel(tx4));

      //Botones.
      JPanel panelBotones = new JPanel();
```

```
        panelBotones.setLayout(new FlowLayout());
        JButton botonAceptar = new JButton("Aceptar");
        JButton botonCancelar = new JButton("Cancelar");
        panelBotones.add(botonAceptar);
        panelBotones.add(botonCancelar);


        //Al pulsar el botón aceptar se inicia la simulación con los datos de
gruas, camiones, guras de patio y contenedores.
        botonAceptar.addActionListener(new EventoAceptar(cuadroTexto1,
cuadroTexto2, cuadroTexto3, cuadroTexto4));
        botonAceptar.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
            CloseFrame();
          }
        });

        //Cuando toco el boton de cancelar se cierra esa ventana para volver a
poner los datos
        botonCancelar.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
            CloseFrame();
          }
        });

        Container cp = getContentPane();
        cp.add(panelPedidos, BorderLayout.CENTER);
        cp.add(panelBotones, BorderLayout.SOUTH);
        }

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

A1.4. Archivo “VentanaConfirmacionDatos.java”.

```
package puerto2;

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

import javax.swing.JTextField;

public class EventoAceptar implements ActionListener{

   public JTextField cuadroTexto1;
   public JTextField cuadroTexto2;
   public JTextField cuadroTexto3;
   public JTextField cuadroTexto4;

   public EventoAceptar(JTextField cuadroTexto1, JTextField cuadroTexto2,
JTextField cuadroTexto3, JTextField cuadroTexto4){
      this.cuadroTexto1 = cuadroTexto1;
      this.cuadroTexto2 = cuadroTexto2;
      this.cuadroTexto3 = cuadroTexto3;
      this.cuadroTexto4 = cuadroTexto4;
   }

   public void actionPerformed(ActionEvent e){

      if (cuadroTexto1 == cuadroTexto1 && cuadroTexto2 == cuadroTexto2 &&
cuadroTexto3 == cuadroTexto3 && cuadroTexto4 == cuadroTexto4){
         String ct1 = cuadroTexto1.getText();
         int x1 = Integer.parseInt(ct1);
         String ct2 = cuadroTexto2.getText();
         int x2 = Integer.parseInt(ct2);
         String ct3 = cuadroTexto3.getText();
         int x3 = Integer.parseInt(ct3);
         String ct4 = cuadroTexto4.getText();
         int x4 = Integer.parseInt(ct4);

      PrototipoBasicoMainPrueba sim = new PrototipoBasicoMainPrueba(1, x1,
x2, x3, x4);
         sim.start();

      }
   }

}
```

A1.5. Archivo “EventoAceptar.java”.

```
package puerto2;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.TreeMap;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import es.ull.iis.simulation.model.Resource;

import java.awt.Toolkit;
import java.awt.Checkbox;

public class VentanaResultados extends JFrame {

   public static JTextField cuadroTexto1;
   public static JTextField cuadroTexto2;
   public static JTextField cuadroTexto3;
   public static JTextField cuadroTexto4;
   public static long tiempo;


   public VentanaResultados(int nGruas, int nCamiones, int nGruasPatio, int
nContenedores, long tiempo, TreeMap<Integer, Long> tiemposPorContenedor,
TreeMap<Resource, Long> tiempoOciosoCamiones,
        TreeMap<Resource, Long> tiempoOciosoGruasBarco, TreeMap<Resource,
Long> tiempoOciosoGruasPatio) {
     super("Confirmación del diseño.");

   setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\el_ma\\Desk
top\\tct_2_v2.jpg"));
     setSize(607, 247);
     setLocationRelativeTo(null);
     setResizable(false);
     setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

     this.cuadroTexto1 = new JTextField(10);
     cuadroTexto1.setText("" + nGruas);
     this.cuadroTexto2 = new JTextField(10);
     cuadroTexto2.setText("" +  nCamiones);
     this.cuadroTexto3 = new JTextField(10);
     cuadroTexto3.setText("" + nGruasPatio);
     this.cuadroTexto4 = new JTextField(10);
     cuadroTexto4.setText("" + nContenedores);
     this.tiempo = tiempo;

     String tx1 = cuadroTexto1.getText();
     String tx2 = cuadroTexto2.getText();
     String tx3 = cuadroTexto3.getText();
     String tx4 = cuadroTexto4.getText();
     String tTotal = String.valueOf(tiempo);

     //Panel de pedidos.
```

```
        JPanel panelPedidos= new JPanel();
        GridLayout gl = new GridLayout(6, 2, 0, 5);
        panelPedidos.setLayout(gl);
        panelPedidos.add(new JLabel("Grúas:"));
        panelPedidos.add(new JLabel(tx1 + " grúas de barco."));
        panelPedidos.add(new JLabel("Camiones:"));
        panelPedidos.add(new JLabel(tx2 + " camiones en el puerto."));
        panelPedidos.add(new JLabel("Grúas de patio:"));
        panelPedidos.add(new JLabel(tx3 + " grúas de patio."));
        panelPedidos.add(new JLabel("Contenedores:"));
        panelPedidos.add(new JLabel(tx4));
        panelPedidos.add(new JLabel("Tiempo necesario para la simulación:"));
        panelPedidos.add(new JLabel(tTotal + " minutos."));

        //Botones.
        JPanel panelBotones = new JPanel();
        panelBotones.setLayout(new FlowLayout());
        JButton botonAceptar = new JButton("Aceptar");
        JButton botonCancelar = new JButton("Cancelar");
        panelBotones.add(botonAceptar);
        panelBotones.add(botonCancelar);


        Container cp = getContentPane();
        cp.add(panelPedidos, BorderLayout.CENTER);

        Checkbox cb1 = new Checkbox("Grafico: Tiempo ocioso de recursos.");
        panelPedidos.add(cb1);


        Checkbox cb2 = new Checkbox("Grafico: Elemetos con conflictos.");
        panelPedidos.add(cb2);
        cp.add(panelBotones, BorderLayout.SOUTH);

        //Al pulsar el botón aceptar se inicia la simulación con los datos de
gruas, camiones, guras de patio y contenedores.
        botonAceptar.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
            if (cb1.getState() == true){
              new
GraficoPorcentajeTiempoUtiliacionRecursos(tiempoOciosoCamiones,
tiempoOciosoGruasBarco, tiempoOciosoGruasPatio, tiempo);
            }
            else{
            }
            if (cb2.getState() == true){
              new GraficoTiempoSobrepasado(tiemposPorContenedor);
            }
            else{
            }
            CloseFrame();
          }
        });


        //Cuando toco el boton de cancelar se cierra esa ventana para volver a
poner los datos
        botonCancelar.addActionListener(new ActionListener(){
          public void actionPerformed(ActionEvent e){
            CloseFrame();
          }
```

```
        });

    }
    public void CloseFrame(){
        super.dispose();
    }

}
```

A1.6. Archivo “VentanaResultdos.java”.

```
package puerto2;

import java.awt.BorderLayout;
import java.io.File;
import java.util.TreeMap;

import javax.swing.JFrame;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import java.awt.Toolkit;

public class GraficoTiempoSobrepasado {

   public GraficoTiempoSobrepasado (TreeMap<Integer, Long> tiempos){

      DefaultCategoryDataset datos = new DefaultCategoryDataset();

      for ( Integer containerId : tiempos.keySet()){
        if(tiempos.get(containerId) > 25){
          datos.setValue(tiempos.get(containerId) - 25, "Tiempo
sobrepasado.", containerId);
        }
      }


      JFreeChart chart = ChartFactory.createBarChart(
           "Elementos que se han pasado del tiempo establecido.",
           "Contenedor.",
           "Tiempo sobrepasado.(minutos)",
           datos,
           PlotOrientation.VERTICAL,
           false,
           true,
           false);

      ChartPanel chartpanel = new ChartPanel(chart);

      JPanel jPanel = new JPanel();
      jPanel.setLayout(new BorderLayout());
      jPanel.add(chartpanel, BorderLayout.CENTER);

      JFrame jFrame = new JFrame();

   jFrame.setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\el_m
a\\Desktop\\tct_2_v2.jpg"));
      jFrame.getContentPane().add(jPanel);
      jFrame.pack();
      jFrame.setVisible(true);
      jFrame.setLocationRelativeTo(null);
      jFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);


      try{
```

```
        ChartUtilities.saveChartAsJPEG(new
File("C:\\Users\\el_ma\\Desktop\\Grafico_Conflictos.jpg"), chart, 500,
300);
     }catch (Exception e){
        System.out.println("Problema al crear el grafico");
     }

   }
}
```

A1.7. Archivo “GraficoTiempoSobrepasado.java”.

```
package puerto2;

import java.awt.BorderLayout;
import java.awt.Toolkit;
import java.io.File;
import java.util.TreeMap;

import javax.swing.JFrame;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

import es.ull.iis.simulation.model.Resource;

public class GraficoPorcentajeTiempoUtiliacionRecursos {

  public GraficoPorcentajeTiempoUtiliacionRecursos(TreeMap<Resource, Long>
tiempoOciosoCamiones, TreeMap<Resource, Long> tiempoOciosoGruasBarco,
TreeMap<Resource, Long> tiempoOciosoGruasPatio,
       long tMaximo) {
     DefaultCategoryDataset datos = new DefaultCategoryDataset();

     for (Resource recurso : tiempoOciosoCamiones.keySet()){
       double d = (double)tiempoOciosoCamiones.get(recurso);
       datos.setValue((d / tMaximo) * 100, "Porcentaje de tiempo
utilizado", recurso.getDescription());
     }
     for (Resource recurso : tiempoOciosoGruasBarco.keySet()){
       double d = (double)tiempoOciosoGruasBarco.get(recurso);
       datos.setValue((d / tMaximo) * 100, "Porcentaje de tiempo
utilizado", recurso.getDescription());
     }
     for (Resource recurso : tiempoOciosoGruasPatio.keySet()){
       double d = (double)tiempoOciosoGruasPatio.get(recurso);
       datos.setValue((d / tMaximo) * 100, "Porcentaje de tiempo
utilizado", recurso.getDescription());
     }

     JFreeChart chart = ChartFactory.createBarChart3D(
          "Porcentaje de tiempo utilizado en cada recursos",
          "Recurso",
          "Porcentaje de tiempo utilizado",
          datos,
          PlotOrientation.VERTICAL,
          false,
          true,
          false);

     ChartPanel chartpanel = new ChartPanel(chart);

     JPanel jPanel = new JPanel();
     jPanel.setLayout(new BorderLayout());
     jPanel.add(chartpanel, BorderLayout.CENTER);

     JFrame jFrame = new JFrame();
```

```
   jFrame.setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\el_m
a\\Desktop\\tct_2_v2.jpg"));
      jFrame.getContentPane().add(jPanel);
      jFrame.pack();
      jFrame.setVisible(true);
      jFrame.setLocationRelativeTo(null);
      jFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

      try{
        ChartUtilities.saveChartAsJPEG(new
File("C:\\Users\\el_ma\\Desktop\\Grafico_Ocioso.jpg"), chart, 800, 300);
      }catch (Exception e){
        System.out.println("Problema al crear el gráfico");
      }
   }
}
```

A1.8. Archivo “GraficoPorcentajeTiempoUtiliacionRecursos.java”

```
package infoReceiver;


import java.util.TreeMap;

import es.ull.iis.simulation.info.ElementActionInfo;
import es.ull.iis.simulation.info.ElementInfo;
import es.ull.iis.simulation.info.ResourceUsageInfo;
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;
import es.ull.iis.simulation.model.Element;
import es.ull.iis.simulation.model.Resource;
import puerto2.PrototipoBasicoPrueba;
import puerto2.VentanaResultados;

public class InfoTiempoFinal extends View {
   TreeMap<Element, Long> tiemposIni = new TreeMap<Element, Long>();
   TreeMap<Element, Long> tiemposFin = new TreeMap<Element, Long>();
   TreeMap<Integer, Long> ContainerIni_BC = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_CI = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_CP = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> tiempos = new TreeMap<Integer, Long>();
   TreeMap<Resource, Long> camionUltimo = new TreeMap<Resource, Long>();
   TreeMap<Resource, Long> camionAcumulado = new TreeMap<Resource, Long>();
   TreeMap<Resource, Long> gruaBarcoUltimo = new TreeMap<Resource, Long>();
   TreeMap<Resource, Long> gruaBarcoAcumulado = new TreeMap<Resource,
Long>();
   TreeMap<Resource, Long> gruaPatioUltimo = new TreeMap<Resource, Long>();
   TreeMap<Resource, Long> gruaPatioAcumulado = new TreeMap<Resource,
Long>();
   Long resta = 0L;
   long tMaximo = 0L;
   private int nGruas;
   private int nCamiones;
   private int nGruasPatio;
   private int nContenedores;

   public InfoTiempoFinal(int nGruas, int nCamiones, int nGruasPatio, int
nContenedores) {
      //Listener que dice cuando se acabó de colocar todos los containers y
los camiones están en su sitio//
      super("Tiempo que se tarda en realizar el ciclo completo y tengo el
tiemmpo que tarda cada contenedor");
      addEntrance(ElementInfo.class);
      addEntrance(ElementActionInfo.class);
      addEntrance(SimulationStartInfo.class);
      addEntrance(SimulationEndInfo.class);
      addEntrance(ResourceUsageInfo.class);
      this.nGruas = nGruas;
      this.nCamiones = nCamiones;
      this.nGruasPatio = nGruasPatio;
      this.nContenedores = nContenedores;

   }

   @Override
   public void infoEmited(SimulationInfo info) {
```

```
     //Listener para conocer el tiempo que cada contenedor se pasa del
tiempo establecido//
     if (info instanceof ElementActionInfo){
        final ElementActionInfo eInfo = (ElementActionInfo)info;
        final int containerId = eInfo.getElement().getIdentifier();
        switch (eInfo.getType()){
        case REQ:
          break;
        case END:
          if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_IDA)){
            ContainerFin_CI.put(containerId, eInfo.getTs());
            }
          if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_PATIO)){
            ContainerFin_CP.put(containerId, eInfo.getTs());

            resta = ContainerFin_CP.get(containerId) -
ContainerIni_BC.get(containerId);
            tiempos.put(containerId, resta);
            }
          break;
        case START:
          if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.DESCAR
GA)){
            ContainerIni_BC.put(containerId, eInfo.getTs());
          }
          break;
        case INTACT:
          break;
        case RESACT:
          break;
        default:
          break;
        }
     }
     //Aqui está el listener para conocer el tiempo ocioso de los
recursos//
     if(info instanceof ResourceUsageInfo){
        final ResourceUsageInfo eInfo = (ResourceUsageInfo)info;
        final Resource recurso = eInfo.getResource();
        switch (eInfo.getType()){
        case CAUGHT:

   if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.CAMION)){
            camionUltimo.put(recurso, eInfo.getTs());
          }

   if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.GRUA_BARCO)){
            gruaBarcoUltimo.put(recurso, eInfo.getTs());
          }

   if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.GRUA_PATIO)){
            gruaPatioUltimo.put(recurso, eInfo.getTs());
          }
```

```
          break;
        case RELEASED:

  if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.CAMION)){
            long acumulado = 0;
            if (camionAcumulado.containsKey(recurso)) {
              acumulado = camionAcumulado.get(recurso);
            }
            camionAcumulado.put(recurso, acumulado + eInfo.getTs() -
camionUltimo.get(recurso));
          }

  if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.GRUA_BARCO)){
            long acumulado = 0;
            if (gruaBarcoAcumulado.containsKey(recurso)) {
              acumulado = gruaBarcoAcumulado.get(recurso);
            }
            gruaBarcoAcumulado.put(recurso, acumulado + eInfo.getTs() -
gruaBarcoUltimo.get(recurso));
          }

  if(eInfo.getResourceType().getDescription().contains(PrototipoBasicoPrueb
a.GRUA_PATIO)){
            long acumulado = 0;
            if (gruaPatioAcumulado.containsKey(recurso)) {
              acumulado = gruaPatioAcumulado.get(recurso);
            }
            gruaPatioAcumulado.put(recurso, acumulado + eInfo.getTs() -
gruaPatioUltimo.get(recurso));
          }
          break;
      }
    }
    //Listener que haya el tiempo que ha durado la simulacion en
terminar//
    else if (info instanceof ElementInfo) {
      ElementInfo eInfo = (ElementInfo)info;
      switch (eInfo.getType()) {
      case FINISH:
        tiemposFin.put(eInfo.getElement(), eInfo.getTs() - 360);
        if(tMaximo < eInfo.getTs()){
          tMaximo = eInfo.getTs() - 360;
        }
        break;
      case START:
        tiemposIni.put(eInfo.getElement(), eInfo.getTs() - 360);
        break;
      default:
        break;
      }

    }
    else if (info instanceof SimulationEndInfo){
      VentanaResultados tiempo = new VentanaResultados(nGruas, nCamiones,
nGruasPatio, nContenedores, tMaximo, tiempos, camionAcumulado,
gruaBarcoAcumulado, gruaPatioAcumulado);
      tiempo.setVisible(true);
    }
```

```
    }
}
```

A1.9. Archivo “InfoTiempoFinal.java”.

```
package infoReceiver;

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;
import puerto2.PrototipoBasicoPrueba;

//Listener que muestra los tiempos que tarda cada elemento en cada
actividad.

public class InfoActividad2 extends View {

   TreeMap<Integer, Long> ContainerIni_BC = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_BC = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerIni_CI = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_CI = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerIni_CP = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_CP = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerIni_CV = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_CV = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerIni_COM = new TreeMap<Integer, Long>();
   TreeMap<Integer, Long> ContainerFin_COM = new TreeMap<Integer, Long>();
   long tMaximo = 0L;

   public InfoActividad2() {
      super("Tiempo que tarda en realizar la actividad Barco Camión");
      addEntrance(ElementActionInfo.class);
      addEntrance(SimulationStartInfo.class);
      addEntrance(SimulationEndInfo.class);
   }

   @Override
   public void infoEmited(SimulationInfo info) {
      if (info instanceof ElementActionInfo) {
         final ElementActionInfo eInfo = (ElementActionInfo)info;
         final int containerId = eInfo.getElement().getIdentifier();
         switch (eInfo.getType()) {
         case REQ:
           break;
         case END:
           if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.DESCAR
GA)){
              ContainerFin_BC.put(containerId, eInfo.getTs()-360);
           }
           if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_IDA)){
              ContainerFin_CI.put(containerId, eInfo.getTs()-360);
              }
           if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_PATIO)){
              ContainerFin_CP.put(containerId, eInfo.getTs()-360);
              }
```

```
            if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_VUELTA)){
               ContainerFin_CV.put(containerId, eInfo.getTs()-360);
               }
            break;
          case START:
            if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.DESCAR
GA)){
               ContainerIni_BC.put(containerId, eInfo.getTs()-360);
            }
            if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_IDA)){
               ContainerIni_CI.put(containerId, eInfo.getTs()-360);
               }
            if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_PATIO)){
               ContainerIni_CP.put(containerId, eInfo.getTs()-360);
            }
            if
(eInfo.getActivity().getDescription().contains(PrototipoBasicoPrueba.CAMION
_VUELTA)){
               ContainerIni_CV.put(containerId, eInfo.getTs()-360);
            }
            break;
          case INTACT:
            break;
          case RESACT:
            break;
          default:
            break;

          }

      }
      else if (info instanceof SimulationEndInfo){
        for ( Integer containerId : ContainerIni_BC.keySet()){
          if(ContainerFin_BC.containsKey(containerId)){
            System.out.println(containerId + "\t\t" +
ContainerIni_BC.get(containerId) + "\t\t" +
ContainerFin_BC.get(containerId) +
                  "\t\t" + ContainerIni_CI.get(containerId) + "\t\t" +
ContainerFin_CI.get(containerId) +
                  "\t\t" + ContainerIni_CP.get(containerId) +"\t\t" +
ContainerFin_CP.get(containerId));
          }
        }
      }
      else if (info instanceof SimulationStartInfo){

    System.out.println("Container\tTiempoIni_BC\tTiempoFinBC\tTiempoIni_CI\tT
iempoFinCI\tTiempoIni_CP\tTiempoFinCP\t");
      }
    }
}
```

A1.10. Archivo “InfoActividad2.java”.