ANÁLISIS Y DISEÑO DE UN SOFTWARE PARA OPTIMIZAR EL PICKING APLICANDO LA TECNOLOGÍA DE LA REALIDAD AUMENTADA

Renzo Gamero-Rodríguez

Piura, diciembre de 2012

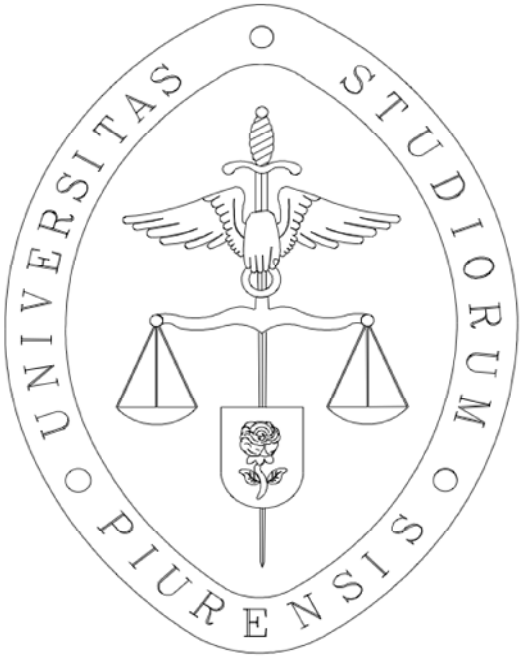
FACULTAD DE INGENIERÍA

Área Departamental de Ingeniería Industrial y Sistemas

Gamero, R. (2012). *Análisis y diseño de un software para optimizar el Picking aplicando la tecnología de la realidad aumentada*. Tesis de pregrado no publicado en Ingeniería Industrial y de Sistemas. Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Industrial y de Sistemas. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA



ANÁLISIS Y DISEÑO DE UN SOFTWARE PARA OPTIMIZAR EL PICKING APLICANDO LA TECNOLOGÍA DE LA REALIDAD AUMENTADA

Tesis para optar el Título de Ingeniero Industrial y de Sistemas

RENZO DOMINGO GAMERO RODRÍGUEZ

Asesor: Dr. Omar Hurtado Jara

Piura, Diciembre 2012

A mi madre y a mi hermano por creer en mí, brindarme su apoyo incondicional y estar conmigo siempre. Todo esto se los debo a ustedes.

Prólogo

La logística involucra una serie de procesos que tiene como objetivo satisfacer las necesidades del cliente al proporcionarle los productos de manera oportuna al mínimo coste. Uno de estos procesos logísticos es el *Picking*, el cual se define como la recogida de productos en un almacén[1].

Por su naturaleza, el *Picking* generalmente es un proceso poco eficiente[2]; además, de ser el más costoso en el almacén pues suele ser intensivo en mano de obra. La investigación presenta una propuesta que permita optimizar el proceso de *Picking* y por consiguiente, hacer más eficientes las operaciones en el almacén.

Por lo expuesto anteriormente, esta investigación propone específicamente el análisis, diseño y prototipo de un *software* para optimizar el *Picking*. Dicho software calcula la ruta óptima en la recogida de productos y la expone al operario mediante la tecnología de la Realidad Aumentada.

Esta investigación cobra sentido por el vertiginoso avance de las aplicaciones de Realidad Aumentada que actualmente tienen un desarrollo incipiente en el campo industrial.

[1] Cfr. Mauleón, M. (2006). Logística y Costos. Madrid. Díaz de Santos. 119 /513
[2] Involucra un elevado numero de desplazamientos del operario en el almacén

Además, la presente investigación basada en esta tecnología permitirá el desarrollo de nuevas aplicaciones que optimicen otros procesos industriales.

Resumen

El procedimiento del *Picking* al que llamaremos “tradicional” es poco eficiente y las opciones para optimizarlo dependen en gran medida del diseño del almacén. Las opciones para optimizarlo implican grandes cambios en su estructura y en la capacitación del personal que realizara las labores de *Picking*, dichos cambios son costosos e involucran un elevado tiempo de realización.

Por ese motivo, el procedimiento del *Picking* cobra relevancia e inspira el trabajo de esta tesis que comprende el análisis, diseño y prototipo de un *software* que optimice la ruta del operario, mediante el algoritmo *Simulated Annealing*; para posteriormente guiar al operario a través de la ruta óptima y asistir en las diferentes etapas de *Picking* brindando información relevante al usuario mediante la tecnología de la realidad aumentada

Índice General

Introducción

Las aplicaciones de la tecnología de la realidad aumentada se encuentran en pleno desarrollo y la versatilidad de la misma la hace susceptible de ser aplicada a casi cualquier campo de estudio. Por este motivo es importante explorar dicha tecnología.

Vemos que en el campo de la logística el proceso tradicional del *Picking* es poco eficiente por los numerosos desplazamientos que realiza el trabajador al recoger los productos., es por ello que resulta indispensable el desarrollo de nuevos procedimientos y herramientas que permitan optimizarlo. Esta investigación propone el diseño de un *software* que guie al operario por el camino de recorrido más corto utilizando la tecnología de la Realidad Aumentada.

La investigación vincula a la ingeniería de *software* y la logística. Con el objetivo de optimizar el proceso de *Picking*. Es por eso que la investigación se estructura de la siguiente manera:

En el primer capítulo, se expone la importancia de seguir un procedimiento estructurado de análisis y diseño para el desarrollo de un *software*.

En el segundo capítulo, se valora la trascendencia de la realidad aumentada en la actualidad.

En el tercer capítulo, se expone la importancia del *Picking* para las empresas.

En el cuarto capítulo, se expone como se resolverá este problema de optimización proponiendo el uso del algoritmo *Simulated Annealing* y las ventajas frente al *Picking* tradicional.

El quinto, expone el análisis y diseño propiamente, en el se incluye: un diagrama de componentes, el diagrama de clases, pantallas de usuario, el diseño de la base de datos, diagrama de casos de uso y el diagrama de despliegue.

El sexto y ultimo capitulo, se describe al prototipo y a sus componentes.

Capítulo 1
Ingeniería de Software

1. Ingeniería de software

El desarrollo de *hardware* de computadoras con circuitos integrados hizo posible la creación de *software* que antes hubiera sido imposible de realizar por los requerimientos de memoria y velocidad de procesamiento. La falta de planificación en los inicios de esta práctica se tradujeron en una crisis de *software* pues en la mayoría de los casos se requería de una mayor inversión y prolongar los cronogramas de trabajo para obtener un *software* poco fiable que necesitaba de un mantenimiento constante y además, no tenía la posibilidad de modificar y mejorar el *software*.

Es en 1968 en la primera conferencia organizada por la OTAN sobre el desarrollo de *software* es donde nace la noción de Ingeniería de *Software* pues era indispensable formalizar el campo de estudio y crear técnicas y métodos que faciliten el proceso de desarrollo de *software* cuando se incrementaba la complejidad.

En un inicio la Ingeniería de *software* tenía un desarrollo incipiente impulsado por la práctica dispersa y no coordinada de ingenieros electrónicos motivados por el desarrollo de una nueva tecnología, pero la repercusión de la disciplina en ámbitos académicos, sociales y económicos hicieron evidente la necesidad de una correcta estructuración y sistematización. Esto llevo a la conformación de lo que hoy conocemos como ingeniería de *software*.

El recorrido de esta disciplina ha sido vertiginoso, sin embargo; los fundamentos sobre los que se basa han permitido la elaboración de una definición, como la siguiente: “La

Ingeniería del *software* es una disciplina de la Ingeniería que comprende todos los aspectos de la producción de *software* desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza". [1]

La Ingeniería de *software* involucra la ejecución de un procedimiento para el desarrollo de *software*, que corresponde a un determinado enfoque, en ese sentido, lo que aquí nos compete es seguir el procedimiento que resuelva de la mejor manera el problema al que nos enfrentamos.

Independientemente de la magnitud del problema a resolver, la ingeniería de *software* orienta el proceso de desarrollo de *software* mediante sus la ejecución de sus principios fundamentales como; la organización del sistema y las nociones fundamentales de procesos.

2. Importancia del proceso de desarrollo de software

El proceso de desarrollo de un *software* es complejo, porque comprende un conjunto de actividades que en sí mismas se configuran por un determinado procedimiento. El conjunto de las actividades que componen el desarrollo de *software* tienen la finalidad de orientar a los ingenieros para obtener un producto de calidad.

En un plano general, podemos decir que son cuatro las actividades fundamentales que se siguen en el desarrollo de un *software* son las siguientes:

1. La especificación del *software*, donde los ingenieros recogen los requerimientos del cliente, para definir el *software* y establecer las restricciones sobre su operación.
2. El desarrollo del *software*, en esta etapa se diferencian dos grandes procesos, el diseño del *software* y su posterior programación. Es en el diseño del *software* donde se especifica toda la información necesaria que facilite la programación del *software*.
3. La validación del *software*, donde el cliente aprueba que el *software* desarrollado satisface sus necesidades.
4. La evolución del *software* que surge como respuesta a los cambios requeridos por el cliente. [2]

---

[1] Sommerville, I. (2005). *Ingeniería de Software*. Madrid. Pearson.6/691

[2] Sommerville, I. (2005). *Ingeniería de Software*. Madrid. Pearson.7/691

Podemos ver que cada una de las actividades fundamentales es compleja en si misma porque el desarrollo de un *software* es una actividad netamente creativa; que exige un procedimiento, de lo contrario corremos el riesgo de hacer un *software* ineficaz al no encausar correctamente dicha creatividad.

En la presente investigación, desarrollamos el análisis y diseño de un *software* que optimice el *Picking*, y la manera más efectiva de realizar un diseño de alta calidad es seguir un enfoque sistemático y organizado que solamente puede proveer la ingeniería de *software*.

Capítulo 2
Realidad Aumentada

1. Evolución histórica, valoración y propuesta de la definición de Realidad Aumentada

La Realidad Aumentada se ha desarrollado estos últimos años de manera más técnica que teórica, por lo que la literatura referida a este tema es insuficiente y en gran medida cambiante debido al pleno auge de su desarrollo. Este desarrollo conllevo a una constante evolución en el concepto de Realidad Aumentada, el cual aun no ha quedado plenamente establecido.

A continuación veremos diferentes definiciones de sobre la Realidad Aumentada que hasta el momento se han propuesto a lo largo de su corta y vertiginosa existencia.

En el año 1990 Tom Caudell la describe como: "*la aumentación de la realidad física mediante el uso de técnicas que la mezclan con contenido digital (virtual). La mezcla producida por esta combinación de contenido produce nuevos mundos mixtos en los que se mezclan objetos reales y virtuales permitiendo usar esta tecnología como puente para la interacción hombre-máquina. Algunos investigadores creen que la RA representa, fundamentalmente, un nuevo método para organizar e interactuar con la información*."[3]

En 1994 Paul Milgram profesor del Departamento de Ingeniería Mecánica e Industrial de la Universidad de Toronto y Fumio Kishino profesor del Departamento de Electrónica, Sistemas de Información y Energía de la Universidad de Osaka; definen la Realidad

[3] Juan Bonnin. 2008. *Realidad Aumentada*. 14 de marzo de 2012, de http://outernet.tk/

Aumentada como: "*un entorno continuo que abarca desde el entorno real a un entorno virtual puro. En medio [esta] la Realidad Aumentada (más cerca del entorno real) y Virtualidad Aumentada (está más cerca del entorno virtual)*". [4]

Esta definición contempla la relación entre dos entornos uno real y uno virtual, sin embargo; descuida muchos elementos como el soporte tecnológico que se debe utilizar, esto se debe a lo temprana de esta definición y que básicamente en ese año todavía no existían aplicaciones directas solo eran experimentos de las capacidades de la tecnología sin relación a ningún objetivo concreto.

En 1997 Ronald Azuma propone que: "*la realidad aumentada combina elementos reales y virtuales, es interactiva en tiempo real y esta registrada en 3D*".[5] Dicha definición, expone un progreso en la comprensión de la tecnología porque alude a la participación activa del sujeto que la experimenta en un momento y lugar determinados, además, enfatiza la característica –hasta el momento- visual de la tecnología, incorporando el diseño en 3D de los objetos que se superponen a la realidad.

En la obra "Realidad aumentada: una nueva lente para ver el mundo" de la Fundación Telefónica (Telefónica, 2010) propone una completa definición de realidad aumentada:

"*Para explicar de manera sencilla en qué consiste la realidad aumentada hay que hacer referencia a los sentidos humanos a través de los cuales percibimos el mundo que nos rodea. Nuestra realidad física es entendida a través de la vista, el oído, el olfato, el tacto y el gusto. La realidad aumentada viene a potenciar esos cinco sentidos con una nueva lente gracias a la cual la información del mundo real se complementa con la del digital. Bajo el paraguas de realidad aumentada se agrupan así aquellas tecnologías que permiten la superposición, en tiempo real, de imágenes, marcadores o información generados virtualmente, sobre imágenes del mundo real. Se crea de esta manera un entorno en el que la información y los objetos virtuales se fusionan con los objetos reales ofreciendo una experiencia tal para el usuario que puede llegar a pensar que forma parte de su realidad cotidiana olvidando incluso la tecnología que le da soporte*".

De acuerdo con la definición de la Fundación Telefónica, podemos ver que los investigadores de esta nueva tecnología ya son conscientes de las capacidades de ella en los usuarios. Sin embargo, supone un grave error creer que los elementos añadidos de la realidad aumentada pueden no diferenciarse de los de la realidad. A pesar, de que la presente investigación no tiene como propósito una reflexión filosófica sobre el impacto de la Realidad Aumentada en los sentidos y en las facultades de la persona, no podemos rehuir al problema aquí planteado. Es indispensable aclarar que la realidad no puede ser

[4] José Carlos Cortizo y Luis Ignacio Díaz. *Madrid Sistemas inteligentes*. 15 de marzo de 2012, de http://www.madrimasd.org/blogs/sistemas_inteligentes/2008/03/21/87063

[5] 2010. *Realidad Aumentada*. 14 de marzo de 2012, de http://www.realidadvirtual.com/realidad-aumentada/

reducida a un constructo de percepciones yuxtapuestas, guiadas por la lógica de un *software*, sino que la realidad es lo que ocurre verdaderamente y no lo fantástico o ilusorio que podamos percibir.

Una de las muchas definiciones de Realidad Aumentada que pulula la web con gran aceptación es: "*Supone una inmersión, por parte del usuario, en un mundo que resulta de la "unión" entre el mundo real y el mundo virtual, ya que el usuario de este tipo de aplicaciones, podrá ver a través de su Smartphone u otro dispositivo con una cámara y una aplicación que genere esa información*".[6]

Otros autores describen la realidad aumentada como: "*la tecnología capaz de reconstruir y ampliar nuestra experiencia del mundo real a través de la superposición en la imagen de distintas capas de datos virtuales. El objetivo central es construir una realidad híbrida, que mezcle la visión en tiempo real de un lugar u objeto con información propia y contextual desde distintas fuentes y orígenes en internet*"[7]

Acerca de esta definición podemos decir que el propósito de la Realidad Aumentada no es propiamente crear una realidad híbrida, sino enriquecer el conocimiento del usuario con información oportuna.

Después de apreciar estas definiciones podemos distinguir diversos elementos en común. El primero; la yuxtaposición del contenido en 3D sobre la realidad que apreciamos, de esta manera se añade información a la realidad y de ahí el origen del nombre.

El segundo; es la necesidad de un dispositivo que nos permita apreciar el contenido adicional, hoy en día puede ser cualquier celular de última generación con conexión a internet.

El último elemento es la simultaneidad en la ocurrencia de eventos. La información que se añade a la realidad se hace en tiempo real.

Analizando prudentemente las definiciones más aceptadas y tras evaluar la bibliografía consultada sobre el tema, podemos proponer la siguiente definición de Realidad Aumentada:

---

[6] *Realidad Aumentada*. 15 de marzo de 2012 de http://arealidadaumentada.wordpress.com/2012/02/20/que-es-la-realidad-aumentada/

[7] @yamilsalinas. 15 de marzo 2012, http://www.yamilsalinas.net/2009/11/17/%C2%BFque-es-la-realidad-aumentada-el-puente-entre-el-mundo-real-y-el-mundo-virtual/

La Realidad Aumentada es un sistema de interacción que toma como entrada la información que proviene del mundo real y genera información de salida (tal como objetos, imágenes, texto, etc.) que se superpone en tiempo real sobre la realidad, desplegándose en la percepción que el usuario tiene de su entorno físico, consiguiendo así un aumento en el conocimiento que el usuario tiene sobre los objetos de su entorno. Todo ello mediante un dispositivo.

2. Elementos para el funcionamiento de la Realidad Aumentada

Para el funcionamiento de la realidad aumentada es necesario un marcador y un dispositivo que permita la visualización de los objetos insertados. El marcador puede ser un dibujo, fotografía, una posición geográfica o cualquier otro elemento que pueda ser identificado por un procesador.

El marcador que se utilizará en este estudio es un dibujo compuesto por un marco de color negro con un diseño en el centro. Las librerías más comunes de Realidad Aumentada calculan la posición del objeto a insertar mediante el análisis de las coordenadas de las esquinas del marcador como se observa en la Figura 1.1.

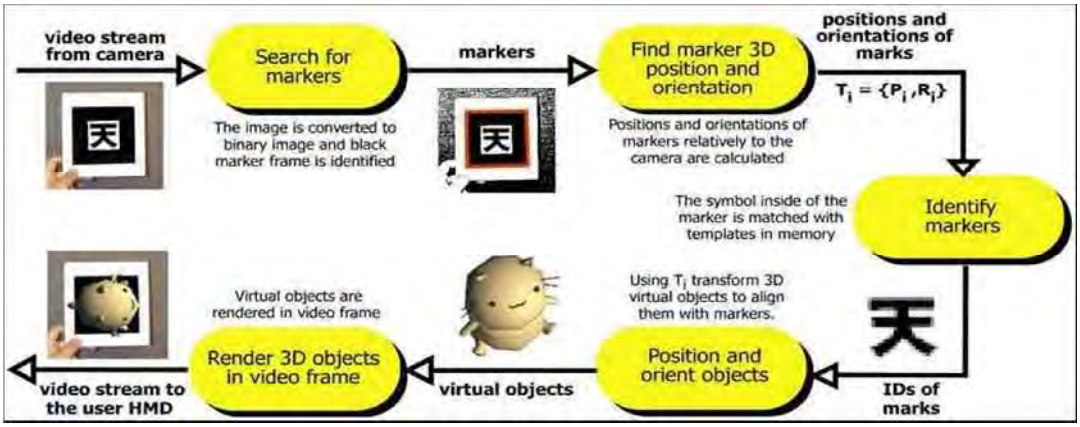


Figura 2. 1 Proceso de Reconocimiento de Marcadores. Fuente: http://outernet.tk/

3. Importancia de la Realidad Aumentada

La tecnología de la Realidad Aumentada ha incrementado vertiginosamente su campo de aplicación, entre las más recientes se encuentran aplicaciones militares en el desarrollo de pantallas que muestren mayor información a los pilotos de combate sin dejar de apreciar la realidad, las aplicaciones médicas van desde el tratamiento de fobias hasta aplicaciones

futuras en entornos altamente riesgosos como añadir información durante una cirugía que requiera de mucha precisión. Las aplicaciones industriales tienen pocos años de desarrollo.

En nuestro país la aplicación de esta tecnología se da incipientemente en el campo de la publicidad con respecto a otros países.

Un factor importante en el uso de esta nueva tecnología es la expansión del uso de teléfonos "inteligentes".

Según el autor Philip Kotler, la tecnología de la Realidad Aumentada es una innovación perturbadora debido a que puede producir un cambio dramático en el mercado.

La define de la siguiente manera: "*Tecnología perturbadora, o innovación perturbadora, es un termino que describe una innovación, producto o servicio tecnológico que utiliza una estrategia "perturbadora" en vez de una "evolutiva" o "vigorizante" para invalidar en un mercado tecnologías dominantes o productos de statu quo. Se le ha mostrado sistemáticamente a la comunidad investigadora que la mayoría de las innovaciones perturbadoras son una minoría en comparación con las evolutivas, que introducen al mercado una innovación de más alto desempeño. Los ejemplos de innovaciones perturbadoras son raros*".[8]

Podemos decir que la base de la innovación perturbadora es producir un cambio dramático en el mercado, haciendo que las tecnologías de *statu quo* queden rápidamente obsoletas; creando turbulencia para todos los participantes involucrados, tanto en las tecnologías existentes como en las cambiadas. Philip Kotler afirma que en cinco años las siguientes tecnologías reconfigurarán el mercado mundial: computación ubicua y en nube, computación contextual, de virtualización y estructural, realidad incrementada, y redes sociales y *software* social. La tecnología perturbadora tiene el potencial de ser el definitivo "cambiador de juego" que puede producir caos en toda una industria, es por ese motivo que supone un gran riesgo para los actuales lideres del mercado; pero, supone una gran oportunidad para quien este atento al cambio y sepa aprovecharlo.[9]

Finalmente; podemos decir que la realidad aumentada, al ser una tecnología novedosa cuyas posibilidades aun no han sido totalmente exploradas en la práctica y mucho menos a

---

[8] Kotler, P. (2010). *Caótica: Administración y marketing en tiempos de caos*. Bogotá. Norma. 29/240

[9] Cfr. Kotler, P. (2010). *Caótica: Administración y marketing en tiempos de caos*. Bogotá. Norma.29-31/240

nivel académico, supone una oportunidad de investigación en el campo de la Ingeniería Industrial como en esta tesis en la que se aplica esta tecnología al proceso de *Picking*.

Capítulo 3
Logística

1. Definición de Logística

El camino desde que un producto sale de la fábrica o el almacén hasta que llega al consumidor final es en la mayoría de los casos complejo por que involucran gran número de factores dinámicos, todo ello en el marco competitivo del mundo globalizado.

Diferentes autores conciben la logística como "*la empresa encargada de satisfacer las necesidades del cliente, proporcionándole los productos en el momento, lugar y cantidad en la que demande el cliente, todo ello al mínimo coste"* [10], aquí se destaca la satisfacción de las necesidades del cliente.

Aunque algunos autores consideren que la logística, en sentido estricto, es sólo la gestión del flujo de materiales; una concepción más completa es la que lleva a la logística a toda actividad en que existan unos recursos que gestionar con criterios de efectividad y de manera más concreta es un conjunto de actividades y procedimientos prácticos, que tienen como finalidad que todos los recursos necesarios para conseguir un fin estén disponibles en el momento, lugar, modo y cantidad precisa al mínimo coste.[11]

---

[10] López, R. (2006).*Operaciones de Almacenaje*. España. Thomson. 2/178

[11] Lozano, J. (2002). *Cómo y dónde optimizar los costes logísticos*. España. Fundación Confemetal. 187/581

2. Necesidad de almacenaje

Todas las empresas necesitan algún tipo de almacenamiento de productos. Esto supone asumir un costo pues los productos requerirán de espacio, personal y equipos para su manipulación y conservación. Esta claro que esta inversión deberá ser mínima.

Según las necesidades de la empresa puede configurarse un tipo de almacén. En las empresas industriales suele haber un almacén de productos terminados, de materias primas y uno de repuestos de maquinarias. En cambio, en las empresas comerciales sólo es necesario un almacén de productos y del manejo de este dependerá la capacidad de la empresa para responder a la demanda.[12]

3. Tipos de almacén

Los diseños de almacenes son muy variados y dependen en gran medida de las necesidades de la empresa.

Principalmente existen los siguientes tipos:

- Almacenes en bloque.- Es el almacenamiento de productos formando bloques, es decir, uno encima de otro. Esto representa un bajo costo, pero puede originar un deterioro en los productos, un pobre empleo de volumen y un manejo deficiente del stock.

- Estanterías fijas.- Los estantes son los elementos más ampliamente utilizados. La importancia recae en el diseño de este almacén pues es determinante para su rendimiento. Sus ventajas más importantes son la buena localización de productos así como la flexibilidad para organizar los productos y ampliar la estantería. La clave esta en maximizar la utilización del espacio disponible.

- Estanterías dinámica.- Este tipo de estanterías permite situar de manera automática los productos a la línea por gravedad con un orden FIFO (*First in First out*) debido a la utilización de rodillos. Al igual que en la estantería fija, el diseño determina su rendimiento. Facilita el proceso de *Picking*, reduciendo movimientos; sin embargo, se reducen posiciones para realizarlo. Una desventaja es que requiere de paletas especiales.

[12] Cfr. López, R. (2006).*Operaciones de Almacenaje*. España. Thomson. 2-4/178

- Sistemas compactos.- Son estanterías móviles que funcionan con railes en el suelo. El costo de instalación es alto y el funcionamiento es lento, sólo es recomendable para productos de poco movimiento.

- Sistemas robotizados.- En los casos anteriores las operaciones pueden ser apoyadas por diferentes equipos (carretillas, montacargas, entre otros) pero, los almacenes robotizados tienen la ventaja de combinar estos equipos y estanterías fijas formando instalaciones que buscan la máxima utilización del espacio. Los movimientos se realizan sin intervención humana con equipos transelevadores[13]. El objetivo es la gestión automática del almacén. Sin embargo, hay dos desventajas: la primera es la alta estandarización de pallets y la segunda es el estudio complejo de su funcionamiento.

4. El *Picking* como principal herramienta de la logística en almacenes

El *Picking* se define como la preparación de pedidos de un almacén. El procedimiento involucra, a grandes rasgos, recolectar una lista de productos de las estanterías de un almacén y conformar un pedido que luego podrá ser enviado al cliente.

Esta actividad busca satisfacer las necesidades de los clientes y es aquí donde recae la mayor importancia de la logística, pues debemos tomar en cuenta que el momento oportuno de llegada es percibido por el cliente como un beneficio, de esta forma el *Picking* aporta un valor añadido al producto que conforma uno de los beneficios externos del producto.

Es la actividad más costosa del almacén debido a la duración de los desplazamientos del operario, pues el procedimiento involucra: Primero, buscar los productos. Segundo, retornar a la zona de preparación de pedidos. Tercero, extraer de las estanterías. Cuarto, devolución de las unidades sobrantes. Quinto, acondicionamiento del pedido y control.

Esto representa entre el 45% y el 75% del coste total de las operaciones de un almacén. [14]

[13] Equipo que puede realizar movimientos de traslación y elevación simultáneamente. Realiza la introducción/extracción.

[14] Cfr. Mauleón, M. (2006). *Logística y Costos*. Madrid. Díaz de Santos. 119 /513

Capítulo 4

Importancia de la optimización del *Picking* con Realidad Aumentada

1. Optimización del *Picking*: minimizando el recorrido total aplicando la tecnología de la Realidad Aumentada

La propuesta de mejora involucra el análisis y diseño de un *software* que gestione los pedidos y guie al operario por el recorrido óptimo en la secuencia de *Picking*. La tecnología de la realidad aumentada permitirá superponer objetos 3D que indicarán al operario que dirección seguir y que producto recoger, reduciendo considerablemente el tiempo total de este proceso.

El *Picking* esta compuesto por diferentes fases que corresponden a las tareas que se deben realizar; uno de los objetivos de esta investigación es reducir el tiempo de cada una de ellas para optimizar el *Picking*.

Las fases son: preparativos, recorridos, extracción, verificación y acondicionamiento y elaboración del packing list del transportista o el documento que exija el almacén. La primera fase involucra el lanzamiento de las ordenes de pedido; tarea que sólo puede ser realizada por el jefe de almacén y se transmite automáticamente al dispositivo que utiliza el trabajador que realiza el *Picking*. Esto reduce considerablemente los errores que se puedan producir por si el procedimiento fuese más largo o burocrático.

La fase de recorridos es sin duda la que conlleva el mayor tiempo de todo el proceso. Esto se debe a que en gran medida depende del conocimiento sobre la distribución de los productos en el almacén y la disposición que tenga el trabajador. El *software* propuesto muestra indicaciones visuales del producto y la ruta optima a seguir, que se yuxtaponen a

la realidad apreciada visualmente mediante el dispositivo utilizado (Tablet, teléfono inteligente, etc.), con este nuevo procedimiento la capacitación necesaria es mínima debido a lo intuitivo de las instrucciones a seguir.

La fase de extracción se ve facilitada por las indicaciones visuales que se le dan al operario sobre la cantidad de producto a extraer.

La última fase de verificación y acondicionamiento el tiempo es reducido mediante el cambio de estado del pedido de manera automática, el cual puede observarse desde las ventanas del *software*.

2. Algoritmo de optimización

El objetivo es optimizar el recorrido del trabajador en el almacén durante la realización del *Picking*. Las características del problema permiten definirlo como el problema del agente viajero, que consiste en determinar la ruta de mínima distancia que debe seguir un agente viajero de tal manera que, partiendo desde su domicilio pase por cada ciudad una y sólo una vez y retorne a su domicilio. El objetivo puede ser también minimizar el costo de transporte o el tiempo de viaje.[15]

Para resolver este problema de optimización debemos tener en cuenta que la dinámica de un almacén exige que el *software* brinde soluciones en el menor tiempo posible, es por ello que para resolver el problema del agente viajero utilizaremos un algoritmo heurístico. El algoritmo seleccionado es el *Simulated Annealing*[16] de la clase de procedimientos de mejora, el cual es mucho más efectivo en problemas de baja extensión.[17]

[15] Angulo, C. (2011). *Investigación de Operaciones*. Universidad de Piura. 118/223

[16] Algoritmo de aproximación o heurístico para la resolución del problema del agente viajero. Fue diseñado basándose en el proceso físico de enfriamiento de un líquido hasta su punto de congelación, cuyo propósito es conseguir una estructura cristalina ordenada. El procedimiento busca "escapar" sistemáticamente de los óptimos locales (resultados que dan la apariencia de haber encontrado el optimo global) para encontrar una solución optima.

[17] Angulo, C. (1996). *Implementación del Simulated Annealing para la resolución de problemas de diseño de rutas de reparto con restricciones de capacidad y de tiempo en redes dispersas no euclideas*. Disertación doctoral de Ingeniería Industrial. Escuela Superior de Ingenieros Industriales de San Sebastián.

3. Ventajas de la utilización de realidad aumentada en el *Picking* frente al *Picking* tradicional [18]

- No se requiere modificar estructuralmente al almacén, pues sólo es necesario agregar los marcadores a la estantería ya existente.
- Las instrucciones visuales son intuitivas, por lo tanto, no se requiere entrenamiento específico del trabajador.
- Los marcadores permiten un registro automático de números de serie.
- La preparación de los pedidos y la posterior comunicación a los trabajadores se realiza sin errores.
- La disposición de los trabajadores se incrementa debido a que se les ofrece herramientas de última tecnología.

[18]Cfr. Kisoft Vision: *Un revolucionario sistema óptico de preparación de pedidos*. 14 de marzo de 2012, de http://www.knapp.com/cms/cms.php?pageName=866&sid=5fadf7dca31cac9497f6eebbc574b6e8

Capítulo 5
Análisis y Diseño de Software

Este capítulo esta dedicado a presentar el diseño del *software* de optimización de *Picking*; para ello exploramos cinco temas fundamentales expresados en cada uno de los apartados que componen el capítulo.

El primero de los apartados es el de diagrama de casos de uso, en el cual se recogen los requerimientos del usuario. El segundo, es el diagrama de componentes que establece la estructura lógica o de programación propuesta para el *software*. El tercero, es el diagrama de clases que describe los la relación de las clases, además se incluye una descripción de cada clase y la interacción con las pantallas del *software*. El cuarto, es el diagrama de base de datos, donde se describe el modelo relacional que representa la estructura de almacenamiento de información que manejara el *software*. Por último, el quinto apartado se dedica al diagrama de despliegue que muestra como se distribuye los componentes del sistema en los nodos físicos o hardware.

1. Diagrama de casos de uso.

Antes de definir los diagramas de caso de uso, debemos entender los elementos que lo conforman. Son dos y a continuación los explicamos. El primero, el propio caso de uso es una función del sistema, define lo que este debe hacer, puede ser efectuar un cálculo, mostrar cierta información, etc. El segundo, es un actor que representa a alguien o algo externo al sistema y que interactúa con él, es decir define un rol que un usuario desempeña con respecto al sistema. Conjugando estos elementos podemos definir a un diagrama de

caso de uso como una representación de las funciones que el sistema ejecuta (funciones del sistema) y la relación del sistema con los actores [19]

En ese sentido, el diagrama de casos de uso nos muestra un conjunto de funciones cuya ejecución (sucesión de acciones) produce un resultado observable y valioso para un actor en particular, en nuestro caso los actores involucrados son el jefe de almacén y el operario que realiza el *Picking*. En la Figura 5.1 se puede apreciar el diagrama de caso de uso para el actor "Jefe de almacén" y la descripción de cada caso de uso se distribuye en las Tablas 5.1 a 5.10. En la Figura 5.2 se aprecia el diagrama de caso de uso para el actor "Operario" y su descripción se detalla en la Tabla 5.11.

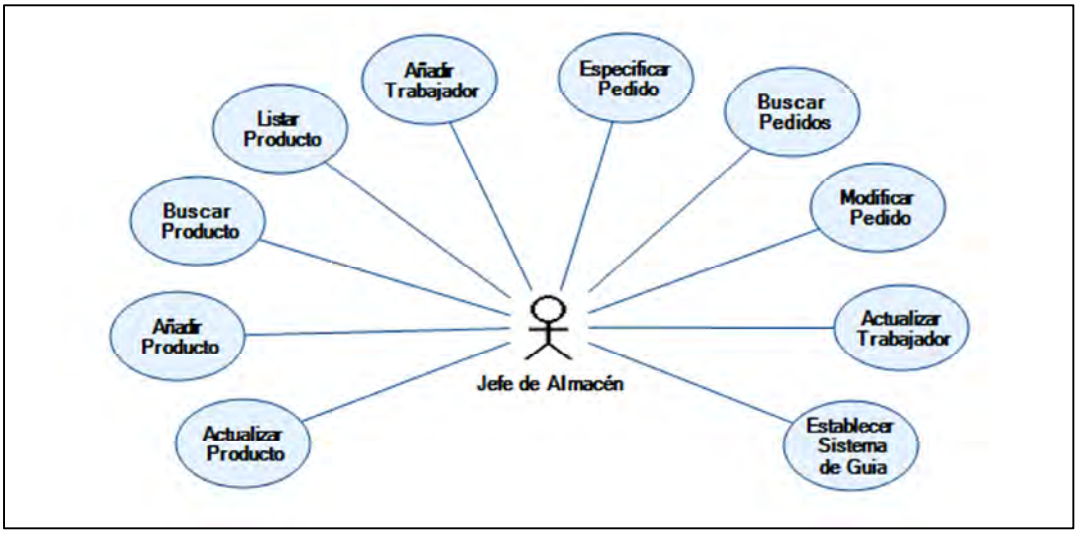


Figura 5. 1 Diagrama de Casos de Uso-Jefe de Almacén.

Fuente: Elaboración Propia.

Tabla 5. 1 Caso de uso: Actualizar Producto

Actualizar Producto	
Actores	Jefe de Almacén
Descripción	El jefe de almacén selecciona un producto determinado y actualiza su información.
Precondiciones	Registro de producto ingresado en la base de datos.
Postcondiciones	Registro modificado del producto.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Administración del

[19] Cfr. Matsukawa, S. (2004). *Análisis y diseño orientado a objetos con UML y Rational Rose*. Macro. 92/520

	Almacén". 3. Ingresar a la ventana "Actualizar Producto". 4. Ingresar los datos a modificar. 5. Presionar botón "Actualizar Producto".

Tabla 5. 2 Caso de uso: Añadir Producto

Añadir Producto	
Actores	Jefe de Almacén
Descripción	El jefe de almacén puede ingresar un producto nuevo al sistema.
Precondiciones	Tener toda la información del producto.
Postcondiciones	Registro del producto almacenado en la base de datos.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Administración del Almacén". 3. Ingresar a la ventana "Añadir Producto". 4. Ingresar los datos. 5. Presionar botón "Añadir Producto".

Tabla 5. 3 Caso de uso: Buscar Producto

Buscar Producto	
Actores	Jefe de Almacén
Descripción	El jefe de almacén puede buscar productos en la base de datos.
Precondiciones	Registro del producto buscado en la base de datos.
Postcondiciones	Producto encontrado si esta almacenado en la base de datos.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Administración del Almacén". 3. Ingresar a la ventana "Buscar Producto". 4. Ingresar los criterios de búsqueda. 5. Presionar botón "Buscar".

Tabla 5. 4 Caso de uso: Listar Producto

Listar Producto	
Actores	Jefe de Almacén
Descripción	Muestra una lista de los productos y sus características principales.
Precondiciones	Registro de productos almacenados en la base de datos.
Postcondiciones	Listado de productos mostrado en pantalla.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Administración del Almacén". 3. Ingresar a la ventana "Productos".

Tabla 5. 5 Caso de uso: Añadir Trabajador

Añadir Trabajador	
Actores	Jefe de Almacén
Descripción	El jefe de almacén puede ingresar la información de un trabajador al sistema.
Precondiciones	Tener información completa del trabajador.
Postcondiciones	Nuevo registro de un trabajador en la base de datos.
Escenario básico	6. Iniciar sesión. 7. Ir al menú ventana y presionar "Administración del Almacén". 8. Ingresar a la ventana "Añadir Trabajador". 9. Ingresar los datos. 10. Presionar botón "Añadir Trabajador".

Tabla 5. 6 Caso de uso: Especificar Pedido

Especificar Pedido	
Actores	Jefe de Almacén
Descripción	El jefe de almacén ingresa un nuevo pedido al sistema y determina el detalle del mismo.
Precondiciones	Ingreso al sistema con usuario y contraseña
Postcondiciones	Pedido validado
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Pedidos". 3. Ingresar a la ventana "Realizar Pedido" 4. Ingresar el listado de productos. 5. Presionar botón "Enviar Pedido"

Tabla 5. 7 Caso de uso: Buscar Pedido

Buscar Pedido	
Actores	Jefe de Almacén
Descripción	El jefe de almacén ingresa un nuevo pedido al sistema y determina el detalle del mismo.
Precondiciones	Ingreso al sistema con usuario y contraseña.
Postcondiciones	Pedido validado.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar "Pedidos". 3. Ingresar a la ventana "Realizar Pedido" 4. Ingresar el listado de productos. 5. Presionar botón "Enviar Pedido"

Tabla 5. 8 Caso de uso: Modificar Pedido

Modificar Pedido	
Actores	Jefe de Almacén
Descripción	El jefe de almacén puede modificar pedidos recientes.
Precondiciones	Pedido a modificar ingresado en la base de datos.
Postcondiciones	Registro de pedido modificado.
Escenario básico	1. Iniciar sesión. 2. Ir al menú ventana y presionar “Pedidos”. 3. Ingresar a la ventana “Modificar Pedido”. 4. Seleccionar pedido. 5. Añadir/eliminar productos. 6. Presionar botón “Enviar Pedido”.

Tabla 5. 9 Caso de uso: Actualizar Trabajador

Actualizar Trabajador	
Actores	Jefe de Almacén
Descripción	El jefe de almacén selecciona el perfil de un trabajador y actualiza su información.
Precondiciones	Registro de trabajador ingresado al sistema.
Postcondiciones	Registro modificado del trabajador.
Escenario básico	1. Iniciar sesión 2. Ir al menú ventana y presionar “Administración del Almacén” 3. Ingresar a la ventana “Actualizar Trabajador” 4. Buscar el registro del trabajador que se desea modificar. 5. Ingresar los datos a modificar. 6. Presionar botón “Actualizar Trabajador”

Tabla 5. 10 Caso de uso: Establecer Sistema de Guía

Establecer Sistema de Guía	
Actores	Jefe de Almacén
Descripción	El jefe de almacén dibuja un diagrama del almacén y se establece los marcadores de tipo ubicación que tendrán que ser colocados a lo largo de la estantería.
Precondiciones	Ingreso al sistema.
Postcondiciones	Creación de todos los marcadores de ubicación.
Escenario básico	1. Iniciar sesión 2. Ir al menú ventana y presionar “Establecer Sistema de Guía” 3. Añadir estantes siguiendo el modelo del almacén. 4. Presionar el botón “Terminado”. 5. Presionar el botón generar marcadores.

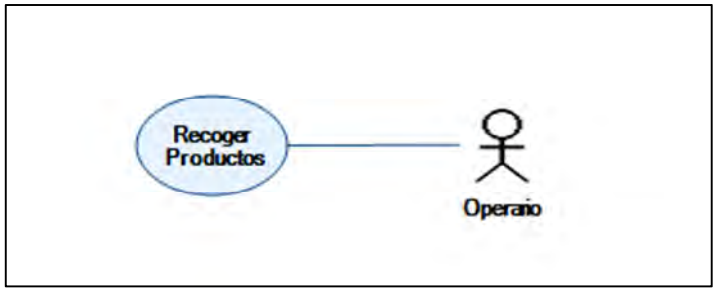


Figura 5. 2 Diagrama de Casos de Uso-Operario. Fuente: Elaboración Propia.

Tabla 5. 11 Caso de uso: Recoger Productos

Recoger Productos	
Actores	Operario
Descripción	El operario recoge los pedidos que el sistema le muestra, es decir, el *software* asiste al operario en esta etapa.
Precondiciones	Pedido asignado a operario activo.
Postcondiciones	Pedido completado.
Escenario básico	1. Iniciar sesión. 2. Recibir guía del sistema y recoger la lista de productos. 3. Pedido completado.

2. Diagrama de componentes

Un componente representa la implementación física de un paquete que contiene elementos lógicos como las clases o interfaces. Ahora bien, un diagrama de componentes muestra como esta diseñado el sistema en base a un conjunto de componentes, que expone detalladamente como colaboran entre si en base a dos aspectos fundamentales: el estructural y el de comportamiento.

El diseño propuesto emplea como enfoque el modelo de vista controlador para realizar el diagrama de componentes, que se define como un esquema de arquitectura de *software* que separa la vista de usuario, la lógica del control y los datos de la aplicación. La finalidad de este concepto es facilitar el mantenimiento y la reusabilidad de los componentes. Son muchas las ventajas que se obtienen y por señalar algunas tenemos que al ser independiente la interfaz de usuario se puede añadir nuevas vistas para un mismo modelo, se pueden añadir vistas sin necesidad de modificar la lógica de control o la estructura de datos. Este último aspecto es muy importante para el diseño de *software* de esta tesis pues se prevé la utilización de distintos dispositivos los cuales exigen requieren configuraciones diferentes.

La importancia del diagrama de componentes radica en que permite visualizar la estructura final del sistema y facilita la modificación y reusabilidad del mismo.

El diagrama de componentes del diseño propuesto puede verse en la Figura 5.3 y una ampliación de los objetos de dominio en la Figura 5.4.

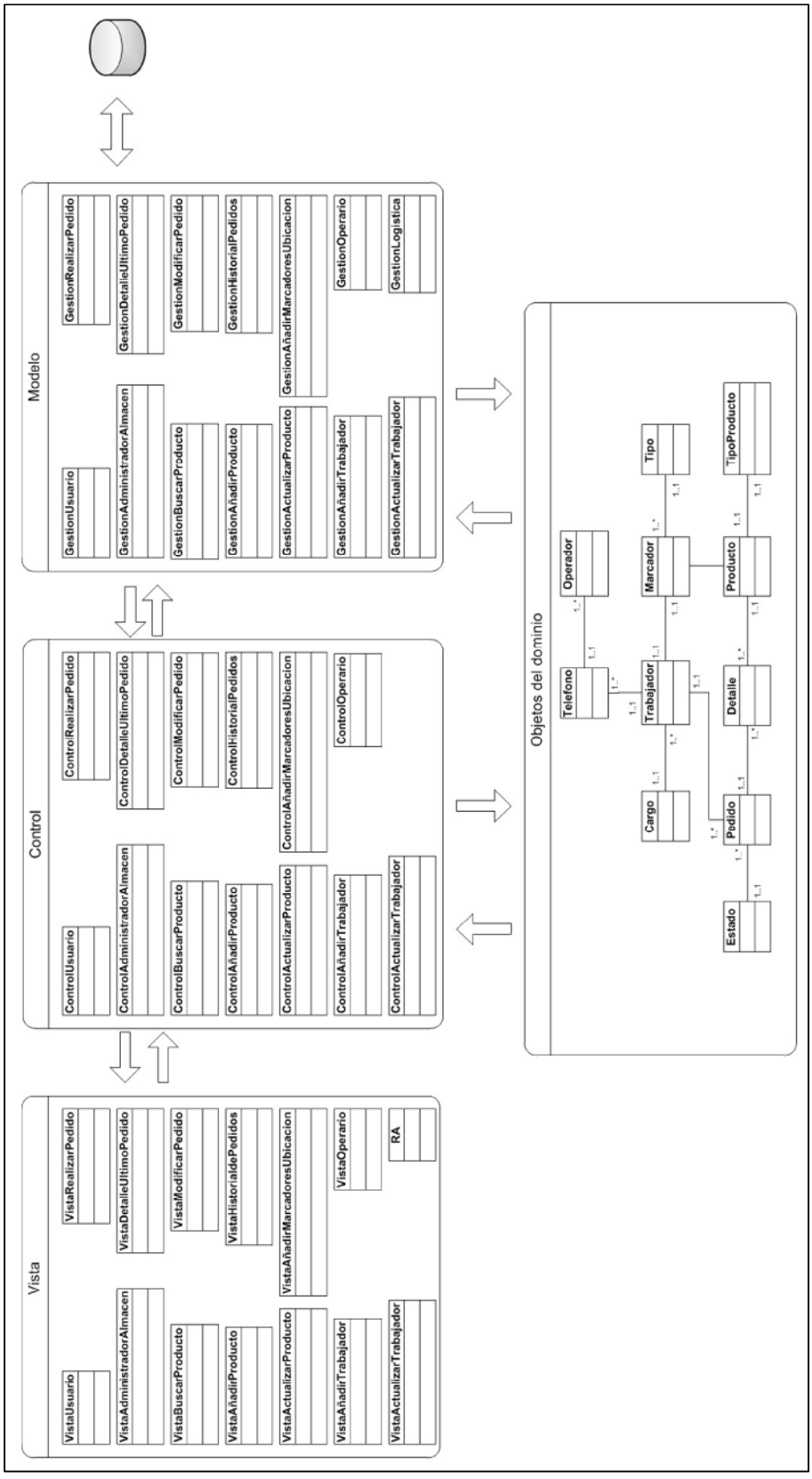


Figura 5. 3 Diagrama de Componentes. Fuente: Elaboración Propia.

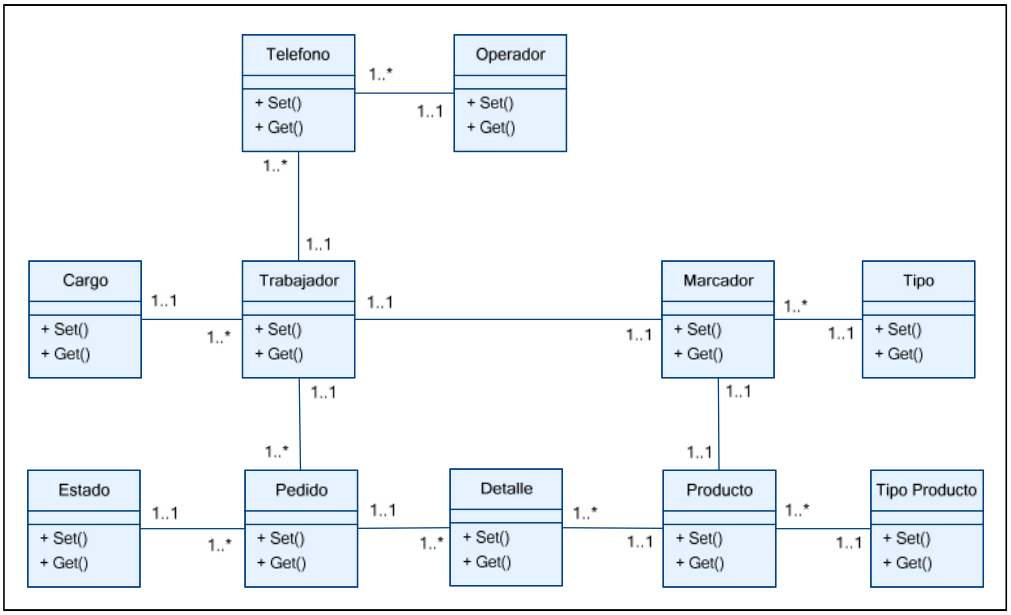


Figura 5. 4 Clases de Dominio. Fuente: Elaboración Propia.

3. Diagrama de clases y pantallas

Las figuras que veremos a continuación representan a las pantallas del *software*, estas se distribuyen en cinco grupos con funciones específicas a los que se puede acceder desde el menú "Ventana". A cada pantalla le corresponden tres tablas referidas al esquema de arquitectura (Vista, Modelo y Control). El primero, contiene a la ventana de inicio de sesión (Figura 5.6 / Tabla 5.12, Tabla-5.13 y Tabla 5.14) y la ventana de cambio de contraseña (Figura 5.7 / Tabla 5.15, Tabla-5.16 y Tabla 5.17), aquí se agrupa las pantallas referidas al ingreso al sistema. El segundo, contiene a las ventanas Administración del Almacén (Figura 5.8 / Tabla 5.18, Tabla-5.19 y Tabla 5.20), Buscar Producto (Figura 5.9 / Tabla 5.21, Tabla-5.22 y Tabla 5.23), Añadir Producto (Figura 5.10 / Tabla 5.24, Tabla-5.25 y Tabla 5.26), Actualizar Producto (Figura 5.11 / Tabla 5.27, Tabla-5.28 y Tabla 5.29), Añadir Trabajador (Figura 5.12 / Tabla 5.30, Tabla-5.31 y Tabla 5.32) y Actualizar Trabajador (Figura 5.13 / Tabla 5.33, Tabla-5.34 y Tabla 5.35), este grupo contiene pantallas que brindan información sobre los productos del almacén. El tercero, contiene a las ventanas Realizar un Pedido (Figura 5.14 / Tabla 5.36, Tabla-5.37 y Tabla 5.38), Detalle Último Pedido (Figura 5.15 / Tabla 5.39, Tabla-5.40 y Tabla 5.41), Modificar Pedido (5.16 / Tabla 5.42, Tabla-5.43 y Tabla 5.44) e Historial de Pedidos (Figura 5.17 / Tabla 5.45, Tabla-5.46 y Tabla 5.47), aquí se encuentran las pantallas que permiten gestionar los pedidos. El cuarto, comprende a la ventana Añadir Marcadores de ubicación (Figura 5.18 / Tabla 5.48, Tabla-5.49 y Tabla 5.50), es en esta pantalla donde se establecen los marcadores que se añadirán al almacén. El quinto, comprende las pantallas que observara el operario (Figura 5.19 y Figura 5.20 / Tabla 5.51, Tabla-5.52 y Tabla 5.53). Se incluyen tres tablas adicionales: la primera, es la tabla de Realidad Aumentada (Tabla 5.54). La segunda, es la tabla Control Logística (Tabla 5.55). La tercera, es la tabla Gestión Logística (Tabla 5.56). El diagrama de clase puede verse en la Figura 5.5

Figura 5. 5 Diagrama de Clases. Fuente: Elaboración Propia.

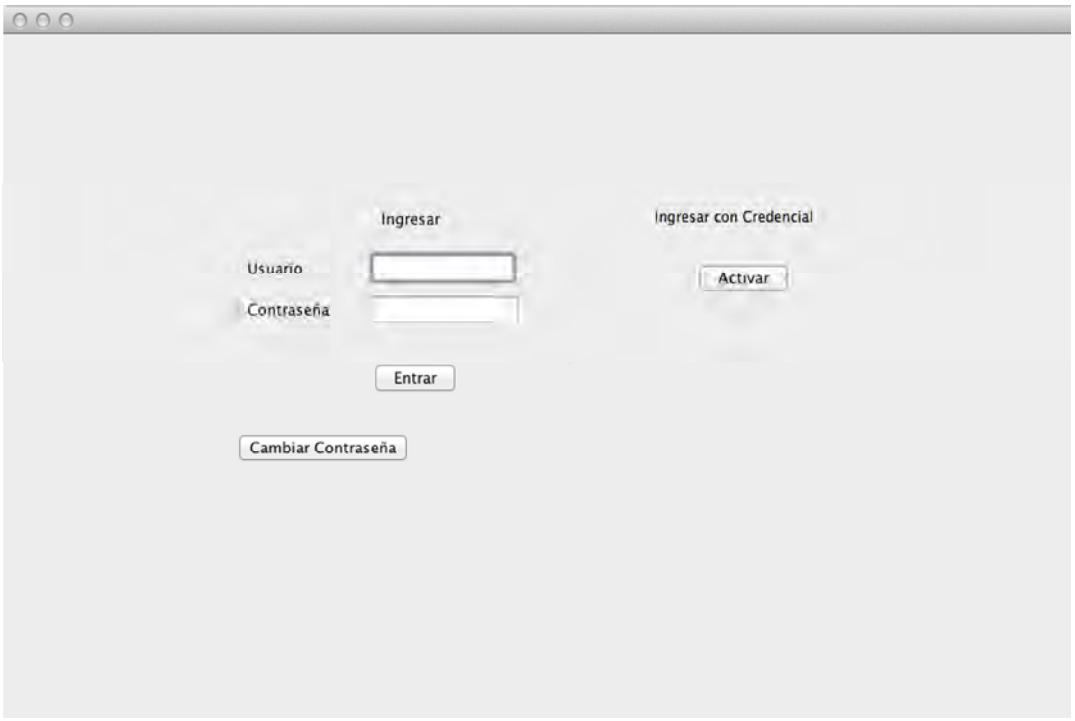


Figura 5. 6 Inicio de Sesión. Fuente: Elaboración Propia.

Tabla 5. 12 Clase: Vista Usuario

Clase	VistaUsuario		
Atributos	Nombre	Tipo	Descripción
	VarINombre	String	Almacena el usuario ingresado.
	VarIPass	String	Almacena la contraseña ingresada.
Métodos	Nombre	Validar	
	Descripción	Método que valida el ingreso de datos. Devuelve los atributos “VarINombre” y “VarIPass“.	
La pantalla tiene dos campos; en el primero de ellos se ingresa el usuario. En el segundo, se ingresa la contraseña correspondiente. El proceso de ingreso es completado presionando el botón “Entrar” que ejecuta la validación. El evento del botón “Activar” activa los métodos de validación.			

Tabla 5. 13 Clase: Control Usuario

Clase	ControlUsuario		
Atributos	Nombre	Tipo	Descripción
	VarArchivo	String	Almacena el nombre del archivo del marcador que lee el método “IdentificarMarcador”.

Métodos	Nombre	IdentificarMarcador
	Descripción	Método que realiza la lectura del marcador y devuelve el nombre del archivo en el atributo “VarArchivo”.
	Nombre	ValidarIngresoMarcador
	Descripción	Método que realiza la validación del ingreso mediante la lectura de un marcador (credencial). Invoca al método “Cqr” de la clase “GestionUsuario” para obtener el nombre del usuario.
	Nombre	ValidarIngreso
	Descripción	Método que realiza la validación del ingreso al sistema mediante la comparación del valor de retorno del método “Ccontraseña” de la clase “GestionUsuario” y la variable “VarIPass” de ingreso que se obtiene mediante la invocación del método “Validar” de la clase” VistaUsuario”.

Tabla 5. 14 Clase: Gestión Usuario

Clase	GestionUsuario		
Atributos	Nombre	Tipo	Descripción
	VarNombre	String	Almacena el usuario ingresado en la clase vista de usuario.
	VarContraseña	String	Almacena la contraseña correspondiente al Usuario ingresado.
	VarValidacion	Boolean	El valor verdadero expresa que el código QR es valido y permite el ingreso al sistema.
Métodos	Nombre	Ccontraseña	
	Descripción	Método que ejecuta una consulta a la base de datos para obtener la contraseña asociada a un usuario. Requiere del atributo “VarNombre” para realizar la consulta y devuelve la variable “VarContraseña”.	
	Nombre	Cqr	
	Descripción	Método que ejecuta una consulta a la base de datos para devolver el nombre del usuario correspondiente al marcador que se utilizo para ingresar al sistema. Requiere del atributo “VarArchivo”. Devuelve el nombre del usuario.	

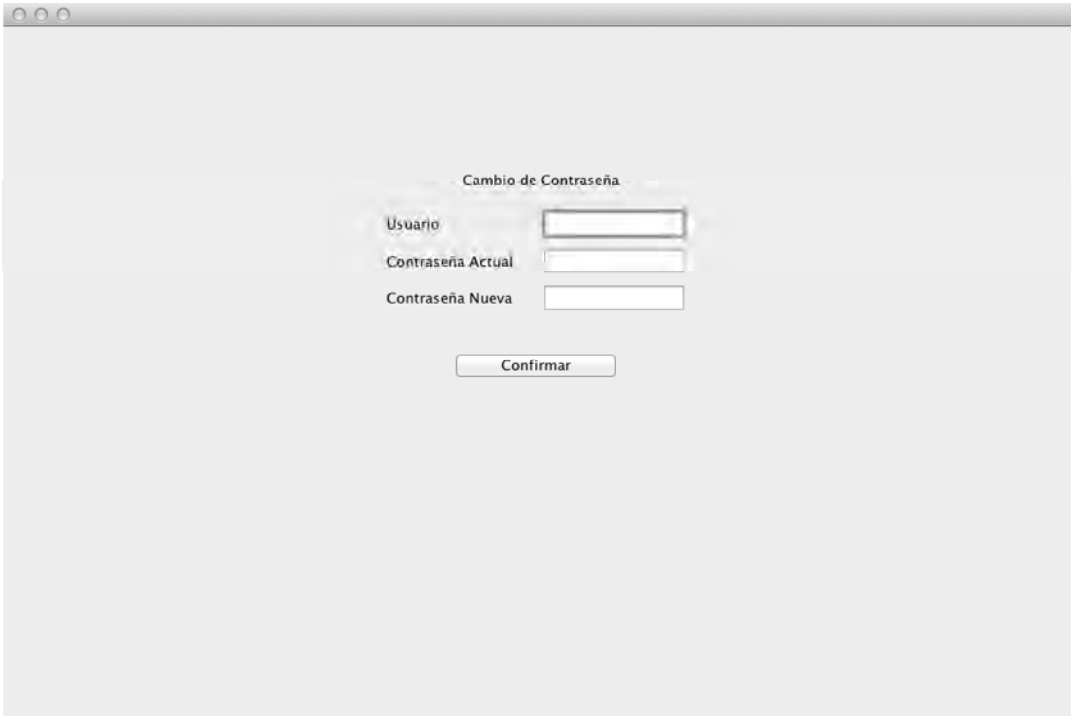


Figura 5. 7 Cambio de Contraseña. Fuente: Elaboración Propia.

Tabla 5. 15 Clase: Vista Cambio de Contraseña

Clase	VistaCambioC		
Atributos	Nombre	Tipo	Descripción
	VarUsuario	String	Almacena el usuario ingresado.
	VarPass	String	Almacena la contraseña actual del usuario.
	VarPassN	String	Almacena la contraseña nueva del usuario.
Métodos	Nombre	Validar	
	Descripción	Método que valida el ingreso de datos. Devuelve los atributos “VarUsuario”, “VarPass” y “VarPassN”.	
Esta pantalla tiene tres campos en los que se ingresa el usuario, la contraseña actual y la contraseña a la que se quiere cambiar. El cambio se completa presionando el botón “Confirmar”.			

Tabla 5. 16 Clase: Control Cambio de Contraseña

Clase	ControlCambioC		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	CambioC	
	Descripción	Método que recibe los valores de retorno del método “Validar” de la clase “VistaCambioC” y los inserta en el método “ConsultaCambioC” de la clase “GestionCambioC”.	

Tabla 5. 17 Clase: Gestión Cambio de Contraseña

Clase	GestionCambioC		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaCambioC	
	Descripción	Método que ejecuta una consulta a la base de datos. Requiere de tres atributos para realizar el cambio de contraseña de usuario.	

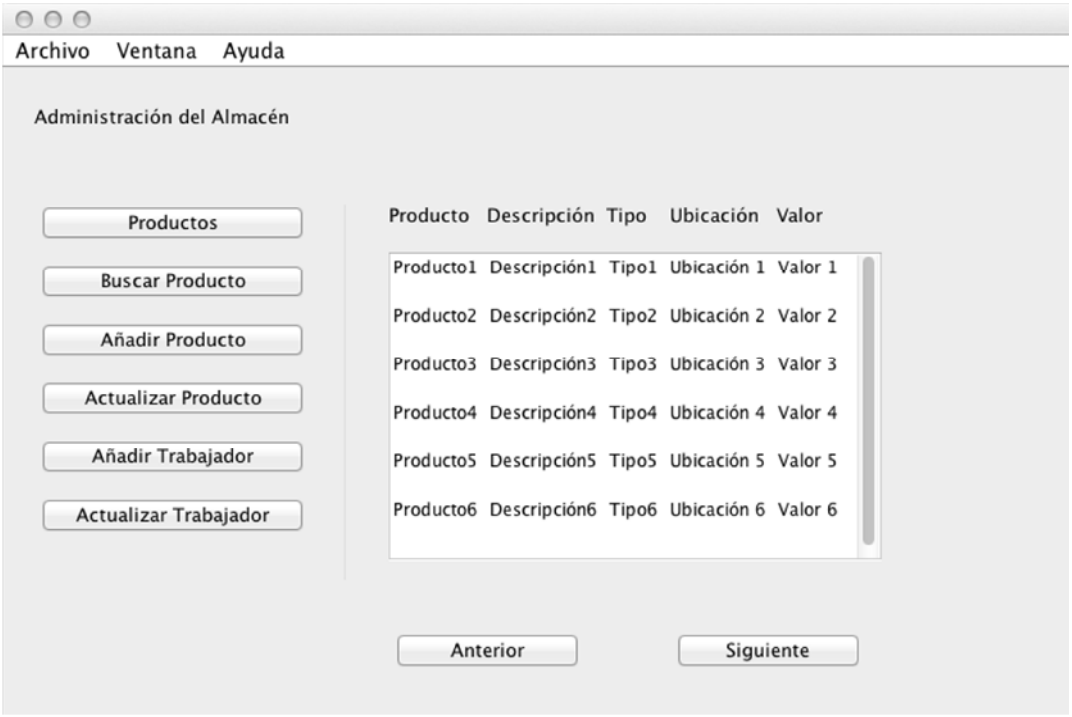


Figura 5. 8 Administración del Almacén. Fuente: Elaboración Propia.

Tabla 5. 18 Clase: Vista Administración del Almacén

Clase	VistaAdministradorAlmacen		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	MostrarProductosVista	
	Descripción	Método que muestra un listado de productos con sus características más relevantes. Invoca al método "MostrarProductosControl" de la clase "ControlAdministradorAlmacen".	
Esta pantalla es la primera en aparecer luego del inicio de sesión. Se recomienda una paginación de seis registros al mostrar el listado de productos. Los eventos de los botones "Atrás" y "Siguiente" determinan el retroceso o avance en la lectura de los registros.			

Tabla 5. 19 Clase: Control Administración del Almacén

Clase	ControlAdministradorAlmacen		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	MostrarProductosControl	
	Descripción	Método que muestra un listado de productos con sus características más relevantes. Invoca al método "MostrarProductos" de la clase "GestionAdministradorAlmacen".	

Tabla 5. 20 Clase: Gestión Administración del Almacén

Clase	GestionAdministradorAlmacen		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	MostrarProductos	
	Descripción	Método que ejecuta una consulta a la base de datos y muestra un listado de productos con sus características (Nombre del producto, Descripción, Tipo, DistanciaX, DistanciaY, DistanciaZ, y Valor).	

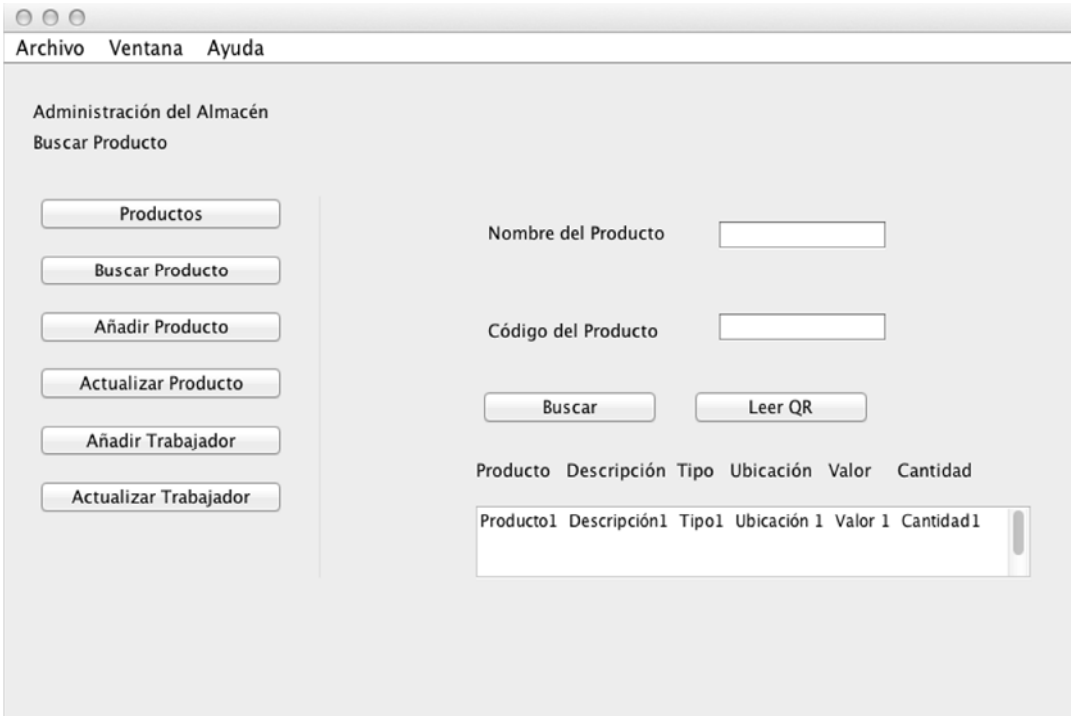


Figura 5. 9 Buscar Producto. Fuente: Elaboración Propia.

Tabla 5. 21 Clase: Vista Buscar Producto

Clase	VistaBuscarProducto		
Atributos	Nombre	Tipo	Descripción
	VarNombreP	String	Almacena el nombre del producto buscado.
	VarCodigoP	String	Almacena el código del producto buscado.
Métodos	Nombre	ValidarINombreProducto	
	Descripción	Método que valida el ingreso del nombre del producto. Devuelve un atributo con el nombre ingresado.	
	Nombre	ValidarICodigoProducto	
	Descripción	Método que valida el ingreso del Código del producto. Devuelve el atributo “VarCodigoP”.	
	Nombre	MostrarProducto	
	Descripción	Método que muestra los el registro del producto buscado. Requiere de los valores de retorno de los métodos “BusquedaconNombre”, “BusquedaconCodigo” y “BusquedaconMarcador” de la clase “ControlBuscarProducto” según sea el caso.	
Esta pantalla tiene dos campos en los que se introduce el nombre de producto o el código de producto como criterio de búsqueda, el evento del botón “buscar” inicia la búsqueda. El botón “Leer QR” activa el lector de marcador que identifica el producto. El botón “Productos” permite regresar a esta pantalla.			

Tabla 5. 22 Clase: Control Buscar Producto

Clase	ControlBuscarProducto		
Atributos	Nombre	Tipo	Descripción
	VarMarcador	String	Almacena el nombre del archivo de un marcador.
Métodos	Nombre	BusquedamedianteIngreso	
	Descripción	Método que ejecuta la búsqueda de un producto mediante el ingreso del nombre del producto o el código del mismo. Invoca a los métodos “BusquedaconNombre” y “BusquedaconCodigo”.	
	Nombre	IdentificarMarcadorProducto	
	Descripción	Método que identifica el marcador leído y devuelve el nombre del archivo en la variable “VarMarcador”.	
	Nombre	BusquedaconMarcador	
	Descripción	Método que inserta en el método “ConsultaProductoconMarcador” de la clase “GestionBuscarProducto” el valor de retorno del método “IdentificarMarcadorProducto” de esta clase. Devuelve el nombre del producto y sus características principales.	

	Nombre	BusquedaconCodigo
	Descripción	Método que inserta en el método “ConsultaProductoconCodigo” de la clase “GestionBuscarProducto” el valor de retorno del método “ValidarICodigoProducto” de la clase “VistaBuscarProducto”. Devuelve el nombre del producto y sus características principales.
	Nombre	BusquedaconNombre
	Descripción	Método que inserta en el método “ConsultaProductoconNombre” de la clase “GestionBuscarProducto” el valor de retorno del método “ValidarINombreProducto” de la clase “VistaBuscarProducto”. Devuelve el nombre del producto y sus características principales.

Tabla 5. 23 Clase: Gestión Buscar Producto

Clase	GestionBuscarProducto		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaProductoconNombre	
	Descripción	Método que realiza una consulta a la base de datos para buscar un producto mostrar sus características principales (Nombre del producto, descripción, tipo, ubicación, valor y cantidad). Requiere del nombre del producto para realizar la consulta.	
	Nombre	ConsultaProductoconCodigo	
	Descripción	Método que realiza una consulta a la base de datos para buscar un producto mostrar sus características principales (Nombre del producto, descripción, tipo, ubicación, valor y cantidad). Requiere del código del producto para realizar la consulta.	
	Nombre	ConsultaProductoconMarcador	
	Descripción	Método que realiza una consulta a la base de datos para buscar un producto mostrar sus características principales (Nombre del producto, descripción, tipo, ubicación, valor y cantidad). Requiere del nombre del archivo del marcador del producto para realizar la consulta.	

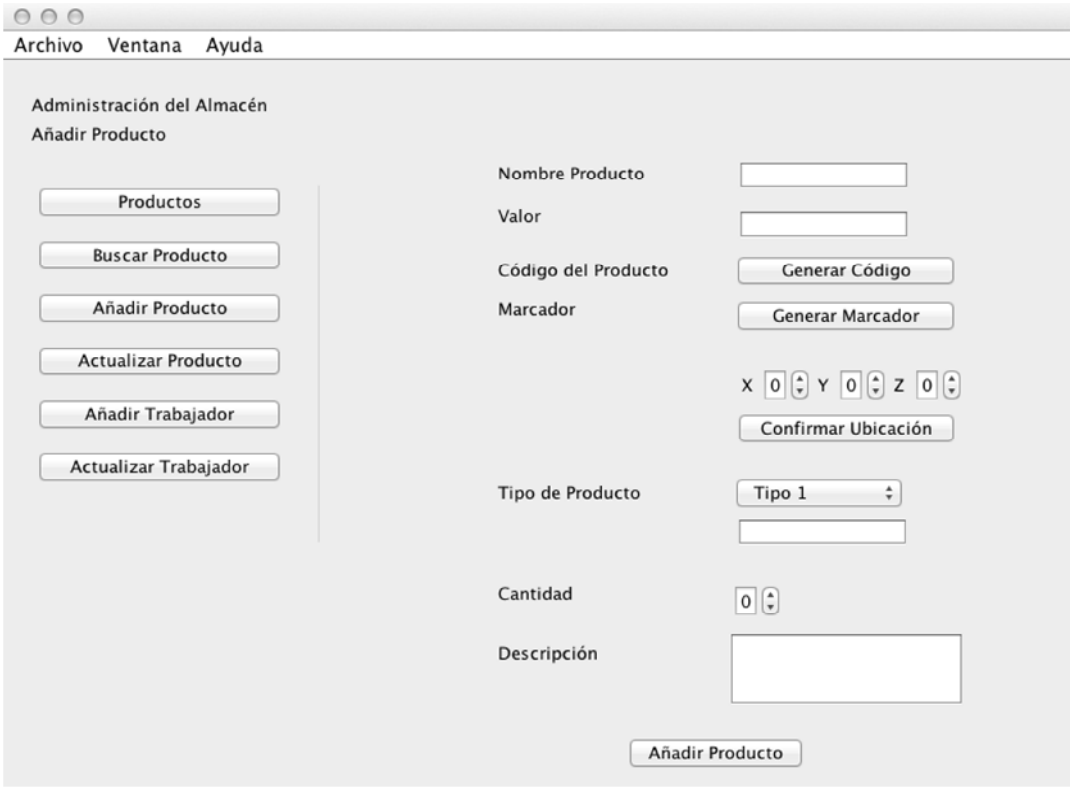


Figura 5. 10 Añadir Producto. Fuente: Elaboración Propia.

Tabla 5. 24 Clase: Vista Añadir Producto

Clase	VistaAñadirProducto		
Atributos	Nombre	Tipo	Descripción
	VarNombre	String	Almacena el nombre del producto ingresado.
	VarValor	Float	Almacena el valor monetario del producto ingresado.
	VarCantidad	Int	Almacena la cantidad de producto ingresado.
	VarDescripcion	String	Almacena la descripción del producto ingresado.
	VarUx	Float	Almacena el valor de la coordenada x de la ubicación del producto ingresado.
	VarUy	Float	Almacena el valor de la coordenada y de la ubicación del producto ingresado.
	VarUz	Float	Almacena el valor de la coordenada z de la ubicación del producto ingresado.
	VarTipo	String	Almacena el valor de tipo del producto ingresado.

Métodos	Nombre	ValidarAñadirProducto
	Descripción	Método que valida el tipo de dato ingresado en cada casillero y que haya sido escogido algún valor de tipo para el producto. Este método devuelve todas las variables de la clase una vez validadas.
Esta pantalla presenta campos requeridos para el ingreso de un nuevo producto. El botón “Generar Código” al ser activado genera un nuevo código para el producto. El botón “Generar Marcador” genera un nuevo archivo de imagen de un marcador y activa el método “GenerarMarcador” de la clase “ControlAñadirProducto”. En el caso de tipo de producto, puede añadirse de dos formas; la primera seleccionando tipos ya ingresados a la base de datos o añadiendo un nuevo tipo al escribirlo en el campo.		

Tabla 5. 25 Clase: Control Añadir Producto

Clase	ControlAñadirProducto		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	GenerarMarcador	
	Descripción	Método que genera el archivo de un nuevo marcador e inserta un nuevo registró a la tabla marcador mediante la invocación del método “” de la clase “”. Devuelve el id del registro insertado.	
	Nombre	GenerarCodigo	
	Descripción	Método que genera un nuevo código del producto.	
	Nombre	AñadirProducto	
	Descripción	Método que ingresa el registro de un nuevo producto a la base de datos; para proseguir con el registro invoca el método “Control” de esta misma clase. Requiere de los valores de retorno del método “ValidarAñadirProducto” de la clase “VistaAñadirProducto” y de los métodos “GenerarMarcador”, “GenerarCodigo” de esta clase. El valor del tipo puede ser el ingresado al campo o el valor seleccionado para ello se invocara los métodos” ValidarAñadirProducto” de la clase “VistaAñadirProducto” y “AñadirTipo” de l a clase “GestionAñadirProducto” respectivamente. Cuando se hayan completado los datos necesarios se invoca al método “AñadirMarcador” de la clase “GestionAñadirProducto” y se insertar los valores a la base de datos.	
	Nombre	Control	
	Descripción	Método que realiza una búsqueda en la base de datos para evitar el ingreso de dos productos iguales.	

Tabla 5. 26 Clase: Gestión Añadir Producto

Clase	GestionAñadirProducto		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	AñadirDatos	
	Descripción	Método que ejecuta la consulta para añadir los datos de un nuevo producto. Recibe los valores de retorno del método "ValidarAñadirProducto" de la clase "VistaAñadirProducto".	
	Nombre	AñadirMarcador	
	Descripción	Método que añade un nuevo registro en la tabla marcador. El tipo de marcador es de código de producto de manera predeterminada.	
	Nombre	AñadirTipo	
	Descripción	Método que inserta a la base de datos un nuevo registró a la tabla "TipoProducto". Requiere del atributo "VarTipo" que se obtiene del método "ValidarAñadirProducto" de la clase "VistaAñadirProducto". Devuelve el id del registro ingresado.	

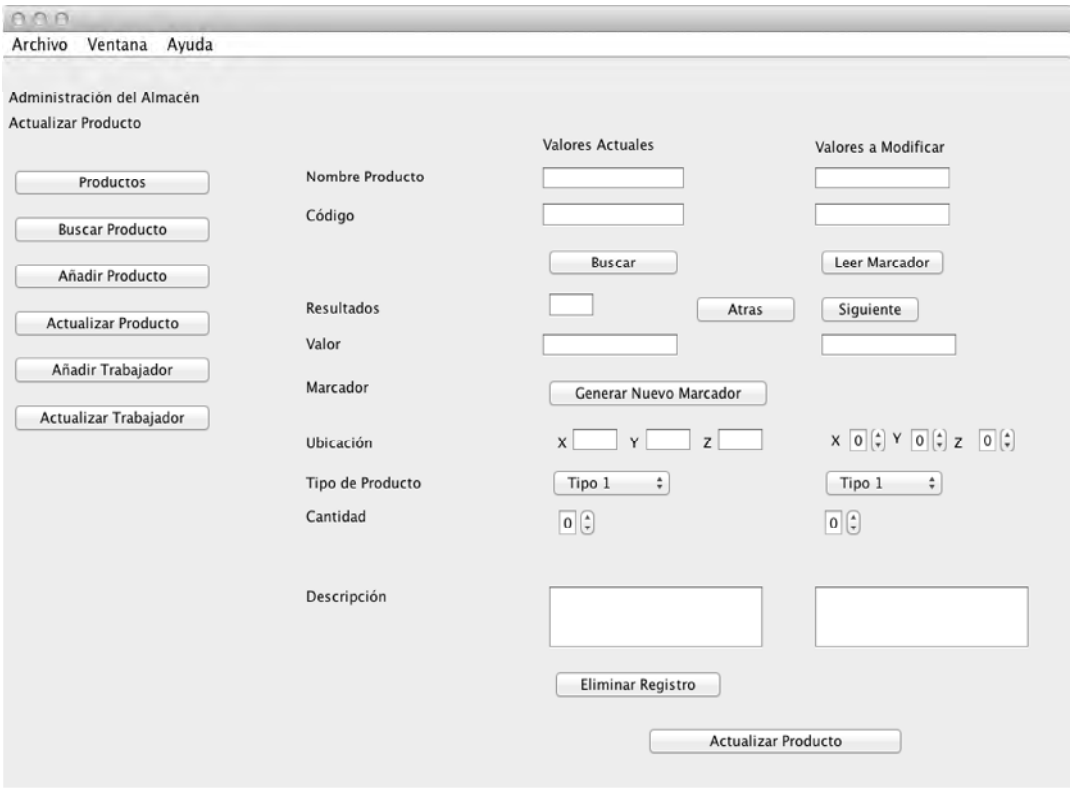


Figura 5. 11Actualizar Producto. Fuente: Elaboración Propia.

Tabla 5. 27 Clase: Vista Actualizar Producto

Clase	VistaActualizarProducto		
Atributos	Nombre	Tipo	Descripción
	VarNombreB	String	Almacena el nombre del producto ingresado para búsqueda.
	VarNombreM	String	Almacena el nuevo nombre del producto para modificación.
	VarValor	Float	Almacena el nuevo valor monetario del producto.
	VarCodigo	Int	Almacena el nuevo código del producto.
	VarUx	Float	Almacena el nuevo valor de la coordenada x de la ubicación del producto.
	VarUy	Float	Almacena el nuevo valor de la coordenada y de la ubicación del producto.
	VarUz	Float	Almacena el nuevo valor de la coordenada z de la ubicación del producto.
	VarTipo	String	Almacena el nuevo valor de tipo del producto.
	VarCantidad	Int	Almacena la nueva cantidad de producto.
	VarDescripcion	String	Almacena la nueva descripción del producto.
Métodos	Nombre	ValidarIngreso	
	Descripción	Método que valida el ingreso de los valores. Devuelve los valores ingresados validados.	
	Nombre	Mostrar	
	Descripción	Método que muestra en pantalla los valores de retorno del método “IdentificarProducto” de la clase “ControlActualizarProducto”.	
Esta pantalla presenta dos columnas: la primera (izquierda) muestra las características principales del producto buscado. En caso haya mas de un resultado estos podrán ser observados mediante los eventos de los botones “Siguiente” y “Atrás”. Los valores a modificar se ingresan en la columna de la derecha y el proceso es completado presionando el botón “Actualizar Producto”.			

Tabla 5. 28 Clase: Control Actualizar Producto

Clase	ControlActualizarProducto		
Atributos	Nombre	Tipo	Descripción
	VarNMarcador	String	Almacena el nombre del archivo de un marcador.
Métodos	Nombre	LeerMarcador	
	Descripción	Método que reconoce el marcador mostrado y devuelve el nombre del archivo.	
	Nombre	IdentificarProducto	
	Descripción	Método que identifica el producto buscado. Puede realizar este proceso de dos maneras distintas. La primera es insertando el atributo “VarNombreB” en el método “ConsultaProductoBuscado” de la clase “GestionActualizarProducto”. La segunda manera es insertando el valor de retorno del método “LeerMarcador” de esta clase en el método “ConsultaProductoBuscadoLector” de la clase “GestionActualizarProducto”. Devuelve el nombre del producto y sus características descritas en los métodos de consulta.	
	Nombre	Actualizar	
	Descripción	Método que realiza la actualización del registro de producto. Para realizar este procedimiento se inserta en el método “ActualizarProducto” de la clase “GestionActualizarProducto” los valores de retorno de los métodos “IdentificarProducto” de esta clase, “ValidarIngreso” de la clase “VistaActualizarProducto” y el valor de retorno del método “ConsultaObteneridProducto” de la clase ” GestionActualizarProducto”. Si se recibe el evento del botón “Eliminar Registro” se invoca el método “EliminarRegistro” de la clase “GestionActualizarProducto”.	

Tabla 5. 29 Clase: Gestión Actualizar Producto

Clase	GestionActualizarProducto		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaProductoBuscado	
	Descripción	Método que ejecuta una consulta para mostrar los atributos de un producto buscado. Requiere del atributo “VarNombreB”.El método debe mostrar los siguientes atributos: Nombre del producto, valor, código del producto, distancia en eje X, distancia en eje Y, distancia en eje Z, tipo, cantidad y la descripción.	

	Nombre	ConsultaProductoBuscadoLector
	Descripción	Método que ejecuta una consulta para mostrar los atributos de un producto buscado. Requiere del valor de retorno “VarNMarcador”.
	Nombre	ConsultaObtenеridProducto
	Descripción	Método que realiza una consulta a la base de datos para obtener el atributo “idProducto”. Requiere el nombre del producto ingresado.
	Nombre	ActualizarProducto
	Descripción	Método que actualiza el registro de un producto. Requiere de los valores actuales, los valores de los atributos que se van a modificar y el valor del registro “idProducto”.
	Nombre	EliminarRegistro
	Descripción	Método que elimina el registro que se muestra en la ventana actualizar producto. Requiere

Figura 5. 12Añadir Trabajador. Fuente: Elaboración Propia.

Tabla 5. 30 Clase: Vista Añadir Trabajador

Clase	VistaAñadirTrabajador		
Atributos	Nombre	Tipo	Descripción
	VarNombres	String	Almacena los nombres del trabajador.
	VarApellidos	String	Almacena los apellidos del trabajador.
	VarEdad	Int	Almacena la edad del trabajador.
	VarDireccion	String	Almacena la dirección del trabajador.
	VarCorreo	String	Almacena la dirección de correo del trabajador.
	VarTelefono	Int	Almacena el teléfono del trabajador.
	VarOperador	String	Almacena el operador telefónico del trabajador
	VarCargo	Int	Almacena el cargo del trabajador. Se refiere directamente al id del mismo.
Métodos	Nombre	ValidarIngreso	
	Descripción	Método que valida el ingreso de los atributos referidos a los datos de un trabajador. Devuelve todas las variables una vez validadas.	
Esta pantalla presenta campos para añadir los atributos necesarios y tres botones de opción múltiple (Cargo, Marcador y Operador). El evento del botón “Añadir Trabajador” ejecuta el ingreso.			

Tabla 5. 31 Clase: Control Añadir Trabajador

Clase	ControlAñadirTrabajador		
Atributos	Nombre	Tipo	Descripción
	VarMarcador	String	Almacena el nombre del archivo del marcador generado.
Métodos	Nombre	AsignarMarcador	
	Descripción	Método que genera un nuevo marcador. Devuelve el nombre del archivo.	
	Nombre	ControlAñadir	
	Descripción	Método que se encarga del ingreso de los datos de un trabajador a la base de datos. Invoca a los métodos “ValdidarIngreso” de la clase “VistaAñadirTrabajador” e “IntroducirMarcador” de la clase “GestionAñadirTrabajador” para obtener sus valores de retorno, los cuales serán necesarios para invocar al método “AñadirTrabajador” de la clase “GestionAñadirTrabajador” y completar el ingreso.	

Tabla 5. 32 Clase: Gestión Añadir Trabajador

Clase	GestionAñadirTrabajador		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	AñadirTrabajador	
	Descripción	Método que ingresa los datos del trabajador a la base de datos. Requiere de variables para obtener los datos que va a añadir y de un método que proporcione el "id" del marcador asignado.	
	Nombre	IntroducirMarcador	
	Descripción	Método que introduce el nombre del marcador asignado y devuelve el id del registro ingresado. Requiere del valor de retorno del método "AsignarMarcador" de la clase "ControlAñadirTrabajador".	

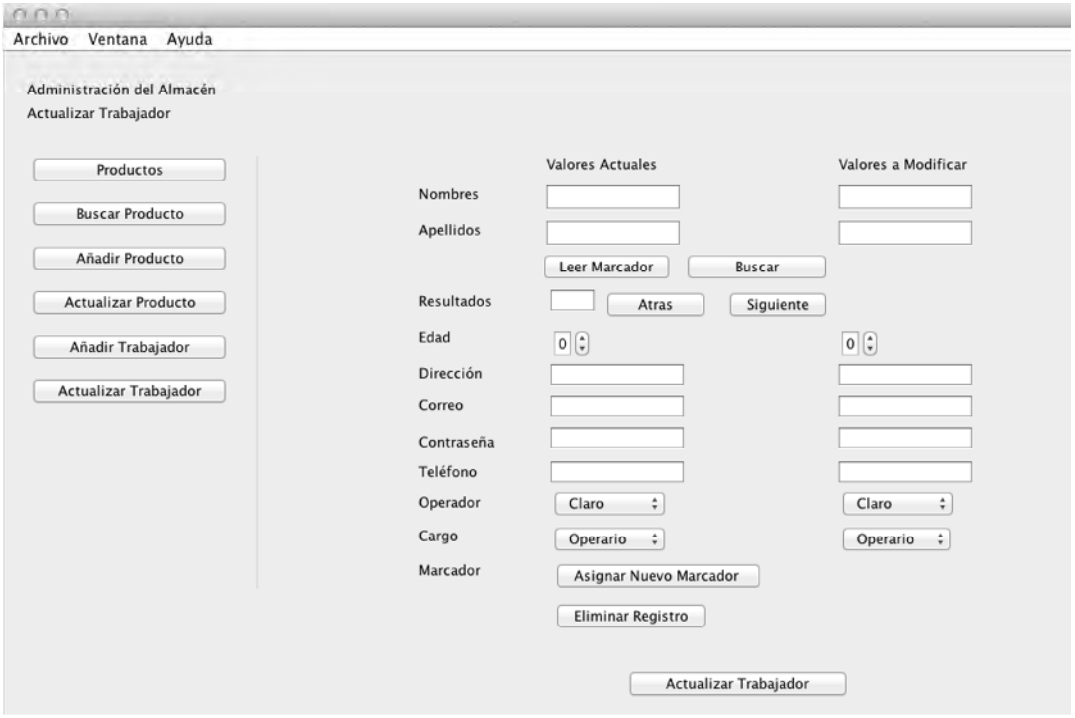


Figura 5. 13Actualizar Trabajador. Fuente: Elaboración Propia.

Tabla 5. 33 Clase: Vista Actualizar Trabajador

Clase	VistaActualizarTrabajador		
Atributos	Nombre	Tipo	Descripción
	VarNombre	String	Almacena el valor ingresado del nombre del trabajador.
	VarNombreM	String	Almacena el valor modificado del nombre del trabajador.
	VarApellido	String	Almacena el valor ingresado del apellido del trabajador.
	VarApellidoM	String	Almacena el valor modificado del apellido del trabajador.
	VarEdadM	Int	Almacena el valor modificado de edad del trabajador.
	VarDireccionM	String	Almacena el valor modificado de la dirección del trabajador.
	VarCorreoM	String	Almacena el valor modificado del correo del trabajador.
	VarTelefonoM	Int	Almacena el valor modificado del teléfono del trabajador.
	VarPassM	Char	Almacena el valor modificado de la contraseña del trabajador.
	VarOpM	String	Almacena el valor modificado del operador telefónico.
	VarCargoM	String	Almacena el valor modificado del cargo del trabajador.
	VarQRM	String	Almacena el valor modificado del nombre del producto. xxx
Métodos	Nombre	ValidarIngresoBusqueda	
	Descripción	Método que se encarga de validar los datos ingresados. Devuelve los atributos una vez validados. Los atributos ingresados son “VarNombre” y “VarApellido”.	
	Nombre	ValidarIngresoModificar	
	Descripción	Método que valida el tipo de dato de los valores ingresados en la columna “Valores a Modificar”.	
	Nombre	MostrarBusqueda	
	Descripción	Método que muestra los datos de un trabajador como resultado de una búsqueda. Si la búsqueda da como resultado varios registros, la navegación se realizara con los botones “Atrás” y “Siguiente”.	
Esta pantalla muestra dos columnas de campos; La primera, tiene los campos nombres y apellidos que son introducidos como criterio de búsqueda, la navegación de los datos mostrados se realiza con la ejecución de los eventos de los botones “Atrás” y “Siguiente”. La segunda, son los campos que se rellenaran en caso se busque la actualización.			

Tabla 5. 34 Clase: Control Actualizar Trabajador

Clase	ControlActualizarTrabajador		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	LeerMarcador	
	Descripción	Método que se encarga de leer el marcador mostrado y devuelve el nombre del archivo del marcador.	
	Nombre	ActualizarDatosControl	
	Descripción	Método que actualiza un registro de la tabla trabajador. Realiza los siguientes dos procesos: El primero; identificar y mostrar los atributos del producto buscado. El segundo; actualizar los atributos del producto. El primer proceso puede llevarse a cabo de dos formas: La primera, mediante el ingreso del nombre del producto o su código, esto se logra invocando al método "ValidarIngresoBusqueda" de la clase "VistaActualizarTrabajador" e insertando sus valores de retorno en el método "ConsultaTrabajador" de la clase "GestionActualizarTrabajador" y para finalizar estos datos son ingresados al método "MostrarBusqueda" de la clase "VistaActualizarTrabajador" completando el proceso. La segunda, mediante la lectura del marcador invocando al método "LeerMarcador" de esta clase e insertando su valor de retorno en el método "ConsultaTrabajadorMarcador" de la clase "GestionActualizarTrabajador", para finalizar se inserta los valores de retorno de este ultimo método en el método "MostrarBusqueda" de la clase "VistaActualizarTrabajador". El segundo proceso requiere del atributo "idTrabajador" que puede ser obtenido de los métodos "ConsultaId" (Si se ingreso el nombre o el código del producto) o "MarcadorId" (Si se utilizo el lector del marcador) de la clase "GestionActualizarTrabajador". Además, se requiere de lo valores de retorno del método "ValidarIngresoModificar" de la clase "VistaActualizarTrabajador". Estos valores son insertados en el método "ActualizarDatos" de la clase "GestionActualizarTrabajador". En caso un atributo no sea ingresado la actualización no lo afectara. ". Si se recibe el evento del botón "Eliminar Registro" se invoca el método "EliminarRegistro" de la clase "GestionActualizarProducto".	

Tabla 5. 35 Clase: Gestión Actualizar Trabajador

Clase	GestionActualizarTrabajador		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaTrabajador	
	Descripción	Método que realiza una consulta a la base de datos. Requiere de, al menos, un valor de retorno del método sea el nombre del producto o el código. Devuelve los atributos nombre del trabajador, apellidos del trabajador, edad, dirección, correo, contraseña, número telefónico, operador y cargo.	
	Nombre	ConsultaTrabajadorMarcador	
	Descripción	Método que realiza una consulta a la base de datos. Requiere del valor de retorno del método "LeerMarcador" de la clase "ControlActualizarTrabajador". Devuelve los atributos nombre del trabajador, apellidos del trabajador, edad, dirección, correo, contraseña, número telefónico, operador y cargo.	
	Nombre	ConsultaId	
	Descripción	Método que realiza una consulta a la base de datos. Requiere de al menos un valor de retorno del método "ValidarIngresoBusqueda" de la clase "VistaActualizarTrabajador". Devuelve el atributo "idTrabajador".	
	Nombre	ActualizarDatos	
	Descripción	Método que ejecuta una actualización de datos. Requiere del atributo "idTrabajador" y de los atributos a modificar.	
	Nombre	MarcadorId	
	Descripción	Método que devuelve el atributo "idTrabajador" y requiere el nombre del archivo del marcador.	
	Nombre	EliminarRegistro	
	Descripción	Método que elimina el registro que se muestra en la ventana actualizar producto. Requiere	

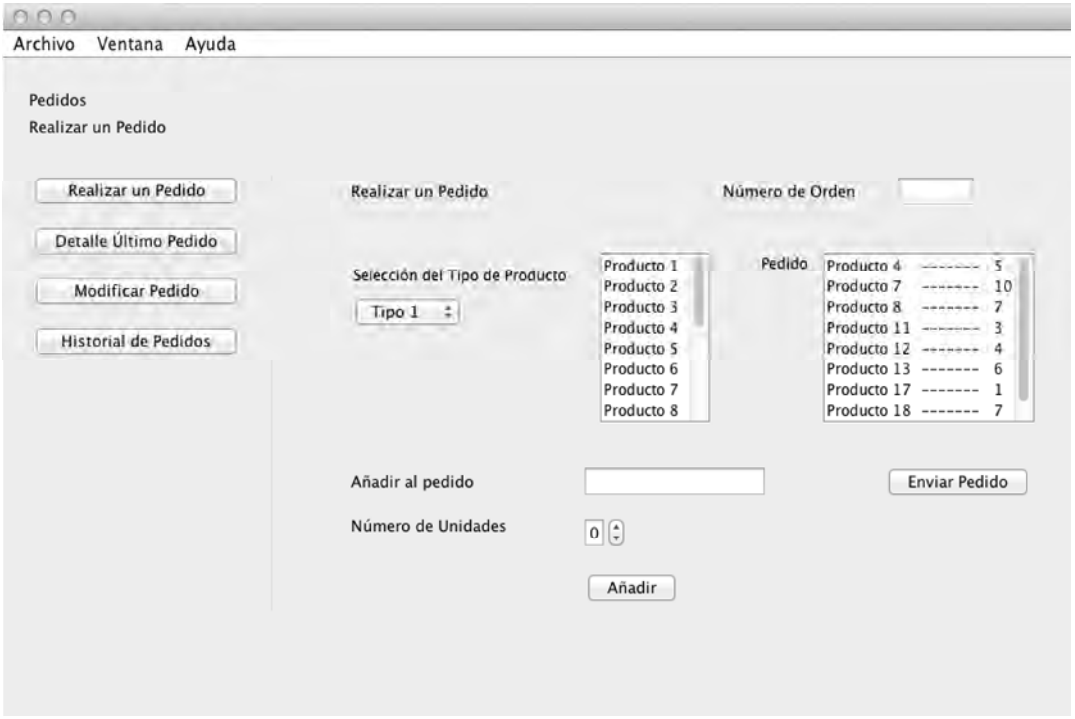


Figura 5. 14Realizar Pedido. Fuente: Elaboración Propia.

Tabla 5. 36 Clase: Vista Realizar Pedido

Clase	VistaRealizarPedido		
Atributos	Nombre	Tipo	Descripción
	VarCantidad	Int	Almacena la cantidad de producto que se ingresa.
Métodos	Nombre	MostrarProductos	
	Descripción	Método que mediante la acción de botón “Tipo” invoca al método “ProductosporTipo” de la clase “ControlRealizarPedido” para poder mostrar los productos de un mismo tipo.	
	Nombre	MostrarPedidoParcial	
	Descripción	Método que muestra y almacena, en un array, una lista con los productos del pedido aun sin enviar. Requiere dos atributos los cuales son el “idProducto” y la cantidad de dicho producto en “CantidadP” que se recogen mediante los eventos de dos botones. El método devuelve un array con los valores de “idProducto” y “CantidadP”.	
Esta pantalla posee dos listas de producto: La primera se activa por el evento del botón “Tipo” el cual determina la lista que se mostrara. La segunda, es el listado de los productos que ya conforman el pedido. Para añadir un producto se selecciona de la lista y aparecerá en el campo “Añadir al pedido” seguidamente se introduce la cantidad y el evento del botón “Añadir” lo añade a la lista de “Pedido”. El pedido es enviado debido al evento del botón “Enviar Pedido”.			

Tabla 5. 37 Clase: Control Realizar Pedido

Clase	ControlRealizarPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	SesionActiva	
	Descripción	Método que de las sesiones activas de los operarios selecciona una y devuelve el atributo “idTrabajador”.	
	Nombre	GenerarNumerodeOrden	
	Descripción	Método que genera un nuevo número de orden y lo devuelve.	
	Nombre	FechayHora	
	Descripción	Método que toma la fecha y la hora al momento de la invocación y los devuelve como valores de retorno.	
	Nombre	AñadirPedidoControl	
	Descripción	Método que añade un nuevo registro de un pedido a la base de datos. Para ello invoca a los métodos “GenerarNumerodeOrden” y “FechayHora” de esta clase y a los métodos “DatosdeSesion” y “AñadirPedido” de la clase “GestionRealizarPedido”. Para añadir un nuevo registro se agrupa los valores de retorno de los métodos “DatosdeSesion”, “FechayHora”, “GenerarNumerodeOrden” y se establece el estado como “en proceso” con todos estos valores de retorno se invoca al método “AñadirPedido”. El valor de retorno de este método es el idPedido del registro que se ha creado.	
	Nombre	AñadirDetalleControl	
	Descripción	Método que añade el detalle de un pedido. Primero se debe recolectar los valores de retorno necesarios para completar el registro. Se Invoca a los métodos “Detalle” de la clase “GestionRealizarPedido” y al método “AñadirPedidoControl” de esta clase y con todos estos datos se invoca al método “AñadirDetalle” de la clase “GestionRealizarPedido” y se completan los datos necesarios con los valores de retorno del método “MostrarPedidoParcial” de la clase “VistaRealizarPedido”.	
	Nombre	ProductosporTipo	
	Descripción	Método que requiere el valor del tipo de producto en el método “ConsultaProductosporTipo” de la clase” GestionRealizarPedido”.	

Tabla 5. 38 Clase: Gestión Realizar Pedido

Clase	GestionRealizarPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaProductosporTipo	
	Descripción	Método que realiza una consulta para mostrar todos los productos de un mismo tipo. Requiere de la variable "VaridTipoProducto" para ejecutar la consulta.	
	Nombre	Detalle	
	Descripción	Método que realiza varias consultas a la base de datos y muestra los siguientes atributos en un array: Trabajador_Marcador_idMarcador, Trabajador_Cargo_idCargo, Producto_Marcador_idMarcador, Producto_idProducto y CantidadP. Requiere los valores de retorno del método "MostraPedidoParcial" de la clase "VistaRealizarPedido" y del valor de retorno del método "SesionActiva" de la clase "ControlRealizarPedido".	
	Nombre	DatosdeSesion	
	Descripción	Método que recibe el atributo idTrabajador y devuelve los siguientes atributos: Cargo_idCargo, Marcador_idMarcador. Invoca al método "SesionActiva" de la clase "ControlRealizarPedido" para obtener su valor de retorno.	
	Nombre	AñadirPedido	
	Descripción	Método que realiza la inserción de un registro a la tabla pedido de la base de datos.	
	Nombre	AñadirDetalle	
	Descripción	Método que realiza la inserción de un registro a la tabla detalle de la base de datos.	

Figura 5. 15Detalle Último Pedido. Fuente: Elaboración Propia.

Tabla 5. 39 Clase: Vista Detalle Último Pedido

Clase	VistaDetalleUltimoPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	MostrarDetalleUltimoPedido	
	Descripción	Método que muestra el detalle del ultimo pedido mediante la invocación del método “DetalleUltimoPedido” de la clase “ControlDetalleUltimoPedido”.	

Tabla 5. 40 Clase: Control Detalle Último Pedido

Clase	ControlDetalleUltimoPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	DetalleUltimoPedido	
	Descripción	Método que invoca al método “ConsultaDetalleUltimoPedido” de la clase “GestionDetalleUltimoPedido”.	

Tabla 5. 41 Clase: Gestión Detalle Último Pedido

Clase	GestionDetalleUltimoPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaDetalleUltimoPedido	
	Descripción	Método que ejecuta una consulta a la base de datos. Consulta el último pedido añadido a la base de datos. Los valores que mostrara la consulta son: numero de orden, fecha, hora y el listado de producto y cantidad.	

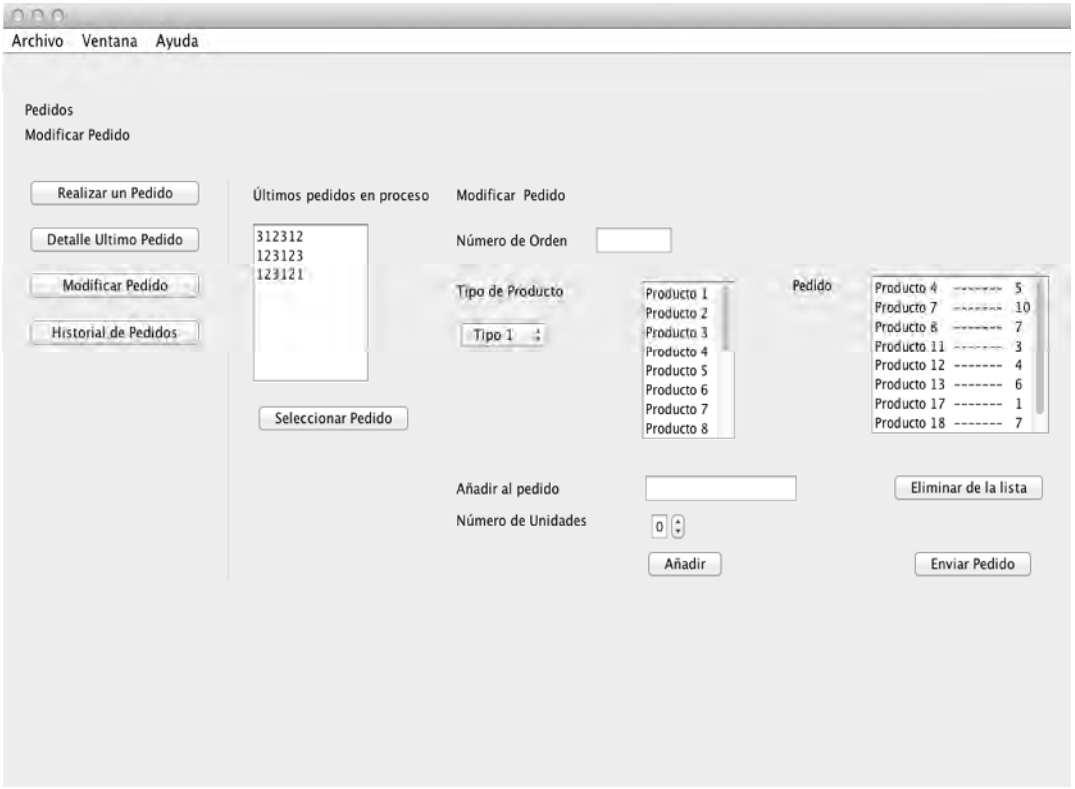


Figura 5. 16 Modificar Pedido. Fuente: Elaboración Propia.

Tabla 5. 42 Clase: Vista Modificar Pedido

Clase	VistaModificarPedido		
Atributos	Nombre	Tipo	Descripción
	VarNumeroO	Int	Almacena el número de orden de un pedido.
Métodos	Nombre	ValidarIngreso	
	Descripción	Método que valida la selección del Numero de orden. El valor de retorno es el atributo ingresado.	
	Nombre	MostrarUltimosPedidosenProceso	
	Descripción	Método que muestra una consulta a la base de datos. Invoca al método “UltimosPedidosenProceso” de la clase “ControlModificarPedido”.	
	Nombre	MostrarProductos	
	Descripción	Método que toma la acción de un botón y toma el valor del “idTipoProducto” el cual es ingresado al método “ProductosporTipo” de la clase “ControlRealizarPedido” para poder mostrar los productos de un mismo tipo.	
	Nombre	MostrarSeleccion	
	Descripción	Método que muestra un listado con los productos y su cantidad referidos a un pedido mediante la invocación del método “Seleccion” de la clase “ControlModificarPedido”.	
	Nombre	MostraProductosdePedido	
	Descripción	Método que muestra los productos de un pedido luego de eliminar uno de ellos del detalle.	
	Nombre	PedidoParcialN	
	Descripción	Método que muestra y almacena en un array una lista con los productos del pedido aun sin enviar. Requiere dos atributos los cuales son el “idProducto” y la cantidad de dicho producto en “CantidadP” que se recogen mediante los eventos de dos botones. El método devuelve un array con los valores de “idProducto” y “CantidadP”	
Para modificar un pedido, se selecciona el número de orden y se presiona el botón “Seleccionar Pedido”, la información del pedido aparecerá y podrá seguirse la misma dinámica de la pantalla Realizar Pedido con la diferencia que aquí pueden eliminarse productos de la lista pedido. Para completar la modificación se presiona el botón “Enviar Pedido”.			

Tabla 5. 43 Clase: Control Modificar Pedido

Clase	ControlModificarPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	UltimosPedidosenProceso	
	Descripción	Método que invoca al método "ConsultaUltimosPedidosenProceso" de la clase "GestionModificarPedido".	
	Nombre	AñadirPedidoModificado	
	Descripción	Método que añade un registro de un pedido a la base de datos. Para ello invoca a los métodos "ValidarIngreso" dela clase "VistaModificarPedido", el método "FechayHora" de la clase "ControlRealizarPedido" y a los métodos "DatosdeSesion" y "AñadirPedido" de la clase "GestionRealizarPedido. Para añadir un nuevo registro se agrupa los valores de retorno de los métodos "DatosdeSesion", "FechayHora", "ValidarIngreso" y se establece el estado como "en proceso" con todos estos valores de retorno se invoca al método "AñadirPedido". El valor de retorno de este método es el idPedido del registro que se ha creado.	
	Nombre	AñadirDetalleModificado	
	Descripción	Método que añade el detalle de un pedido. Primero se debe recolectar los valores de retorno necesarios para completar el registro. Se Invoca a los métodos "Detalle" de la clase "GestionRealizarPedido" y al método "AñadirPedidoControl" de esta clase y con todos estos datos se invoca al método "AñadirDetalle" de la clase "GestionRealizarPedido" y se completan los datos necesarios con los valores de retorno del método "MostrarPedidoParcialN" de la clase "VistaModificarPedido".	
	Nombre	Seleccion	
	Descripción	Método que inserta el valor de retorno del método "ValidarIngreso" de la clase "VistaModificarPedido" en el método "ConsultaProductoyCantidad" de la clase" GestionModificarPedido". Devuelve la consulta de la base de datos.	

	Nombre	ProductosdePedido
	Descripción	Método que invoca a los métodos “EliminarProductodelDetalle” y “ConsultaProductoyCantidad” de la clase “GestionModificarPedido” los cuales requieren de los atributos idProducto, idPedido y numero de orden, que son obtenidos: mediante el evento de la selección de producto y el evento del botón “Eliminar de la lista”, mediante la invocación del método “ConsultaidPedido” de la clase “GestionModificarPedido” y la invocación del “ValidarIngreso” de la clase “VistaModificarPedido” respectivamente.

Tabla 5. 44 Clase: Gestión Modificar Pedido

Clase	GestionModificarPedido		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaUltimosPedidosenProceso	
	Descripción	Método que realiza una consulta a la base de datos. Devuelve los números de orden de los pedidos en proceso.	
	Nombre	ConsultaProductoyCantidad	
	Descripción	Método que realiza una consulta a la base de datos. Devuelve el nombre del producto y su cantidad. Requiere del atributo número de orden.	
	Nombre	EliminarProductodelDetalle	
	Descripción	Método que realiza una consulta a la base de datos. Su función es eliminar un producto de la tabla detalle. Requiere de dos atributos: El idProducto y el idPedido.	
	Nombre	ConsultaidPedido	
	Descripción	Método que devuelve el atributo idPedido al ser ingresado el numero de orden.	

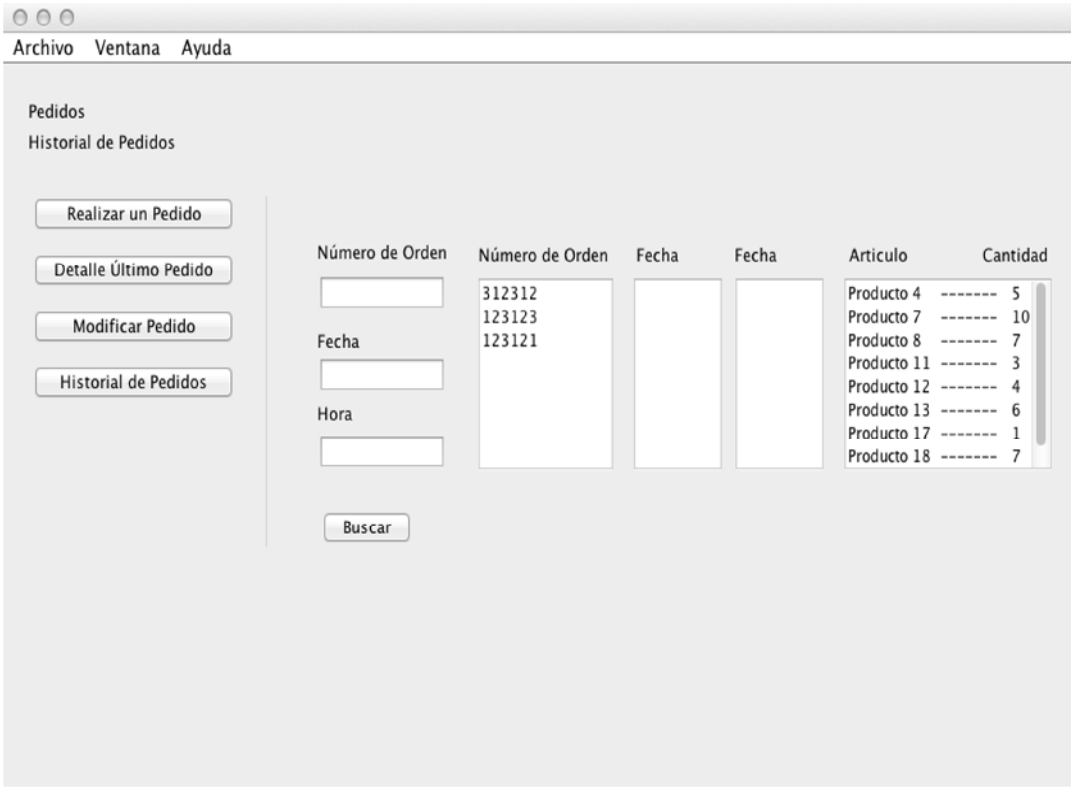


Figura 5. 17Historial de Pedidos. Fuente: Elaboración Propia.

Tabla 5. 45 Clase: Vista Historial de Pedidos

Clase	VistaHistorialdePedidos		
Atributos	Nombre	Tipo	Descripción
	VarNumeroO	Int	Almacena el Número de Orden Buscado.
	VarFecha	String	Almacena la fecha que discrimina la búsqueda.
	VarHora	String	Almacena la fecha que discrimina la búsqueda.
Métodos	Nombre	ValidarIngreso	
	Descripción	Método que recibe los atributos: “VarNumeroO”, “VarFecha” y “VarHora” los cuales almacenan los valores ingresados; el método valida el tipo de dato y devuelve los valores los valores ingresados validados.	
	Nombre	MostrarDetalleUltimoPedido	
	Descripción	Método que invoca al método “DetalleUltimoPedido” de la clase “ControlHistorialdePedidos”.	
En esta pantalla se puede revisar el historial de los pedidos. Los criterios de búsqueda pueden ser el número de orden, la fecha o la hora.			

Tabla 5. 46 Clase: Control Historial de Pedidos

Clase	ControlHistorialdePedidos		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	DetalleUltimoPedido	
	Descripción	Método que invoca al método “ValidarIngreso” de la clase “VistaHistorialdePedidos” y obtiene valores de retorno que serán insertados en el método “ConsultaDetalleUltimoPedido” de la clase “GestionHistorialdePedidos” para obtener una consulta a la base de datos.	

Tabla 5. 47 Clase: Gestión Historial de Pedidos

Clase	GestionHistorialdePedidos		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaDetalleUltimoPedido	
	Descripción	Método que ejecuta una consulta a la base de datos. Requiere de al memos un valor de retorno para realizar la consulta correctamente. Devuelve el número de orden, fecha, hora, nombre del producto y la cantidad.	

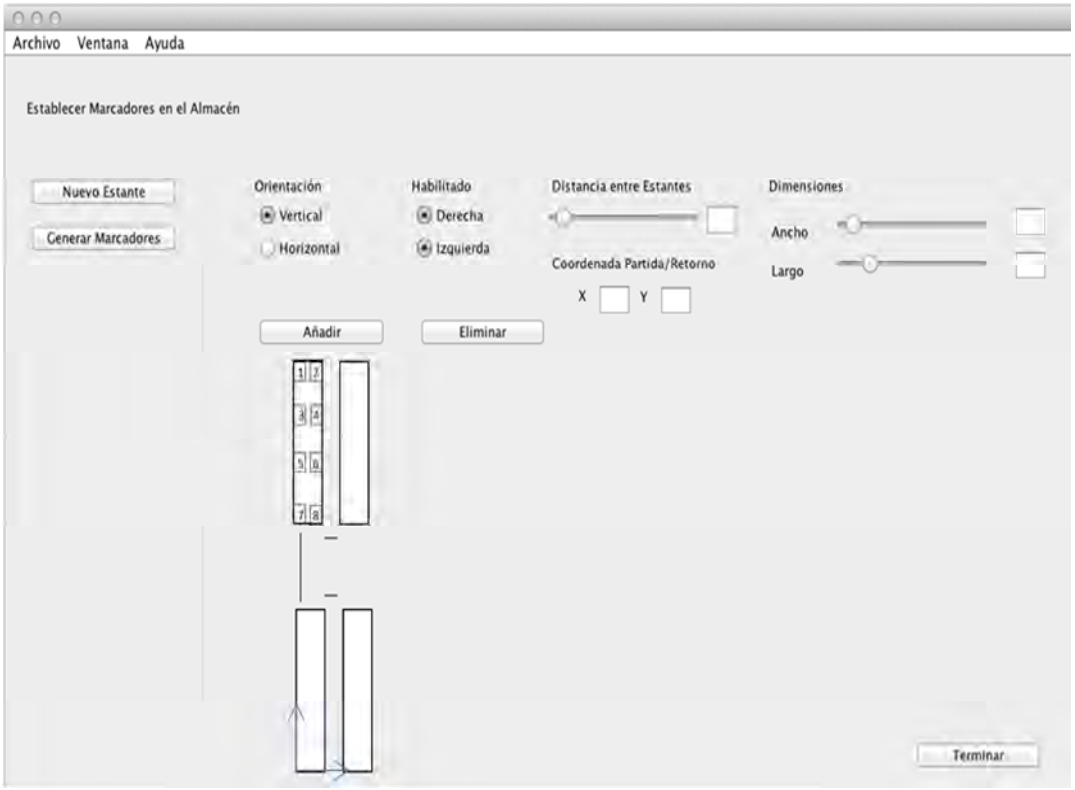


Figura 5. 18Añadir Marcadores de Ubicación. Fuente: Elaboración Propia.

Tabla 5. 48 Clase: Vista Añadir Marcadores de Ubicación

Clase	VistaAñadirMarcadoresUbicacion		
Atributos	Nombre	Tipo	Descripción
	VarVertical	Bool	Es afirmativo si la orientación es vertical.
	VarHo	Bool	Es afirmativo si la orientación es horizontal.
	VarDer	Bool	Es afirmativo si el estante esta habilitado o almacena productos en el lado derecho. Puede darse que el estante se encuentre pegado a la pared.
	VarIz	Bool	Es afirmativo si el estante esta habilitado o almacena productos en el lado izquierdo.
	VarAncho	Float	Almacena el valor numérico del ancho del estante.
	VarAlto	Float	Almacena el valor numérico del largo del estante.
	VarMarcador	Int	Almacena el número del marcador mas cercado al punto de partida.
	VarDistancia	ArrayFloat	Almacena un array con las distancias de separación de cada estante y sus vecinos.
	VarX	Float	Coordenada en x de la ubicación del punto de partida y retorno.
	VarY	Float	Coordenada en x de la ubicación del punto de partida y retorno.

Métodos	Nombre	NuevoEstante
	Descripción	Método que recibe los eventos de los botones sobre de las preferencias ingresadas de un estante nuevo. El método también invoca al método “EliminarEstante” de esta clase
	Nombre	PuntodePartida
	Descripción	Método que valida el ingreso de las coordenadas del punto de partida. Devuelve las variables “VarX” y “VarY”.
	Nombre	EliminarEstante
	Descripción	Método que elimina las preferencias de un estante seleccionado. Actúa cuando recibe el evento de el botón “Eliminar”
Pantalla en la que se genera un modelo del almacén y se crean y distribuyen los marcadores de tipo ubicación.		

Tabla 5. 49 Clase: Control Añadir Marcadores de Ubicación

Clase	ControlAñadirMarcadoresUbicacion		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	GenerarMarcador	
	Descripción	Método que genera marcadores requiere el nombre del archivo.	
	Nombre	Control	
	Descripción	Método que almacena los atributos de todos los estantes añadidos y calcula la cantidad y las coordenadas de los marcadores. El procedimiento se realiza en tres etapas: La primera, se encarga de obtener las preferencias ingresadas invocando al método “NuevoEstante” de la clase “VistaAñadirMarcadoresUbicacion”, este proceso termina cuando se recibe el evento del botón “Terminado”. La segunda, realiza el calculo de marcadores necesarios teniendo en cuenta que el rango entre ellos es de 2 a 2.4 metros. Además, asigna un número a cada marcador generado. El punto de referencia viene predeterminado y sirve para establecer las coordenadas de los marcadores generados. Para generar los marcadores se invoca al método “GenerarMarcador” de esta clase, se inserta en este método el número de marcador y el archivo queda generado. La tercera etapa consiste en ingresar estos marcadores a la base de datos, para ello es necesario invocar al método “AñadirMarcador” de la clase “GestionAñadirMarcadoresUbicacion” en el cual se inserta el nombre del archivo y las coordenadas generadas.	

	Nombre	AñadirPuntodePartida
	Descripción	Método que añade un marcador a la base de datos que se encuentra en la sala de recepción. Para ello recibe las coordenadas invocando al método “PuntodePartida” de la clase “VistaAñadirMarcadoresUbicacion” e insertando estos datos en el método “AñadirMarcador” de la clase “GestionAñadirMarcadoresUbicacion”, el nombre queda predeterminado como “partida”.

Tabla 4. 50 Clase: Gestión Añadir Marcadores de Ubicación

Clase	GestionAñadirMarcadoresUbicacion		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	AñadirMarcador	
	Descripción	Método que ingresa el registro de un marcador a la base de datos. Requiere del nombre del archivo y las coordenadas, el tipo se establece como “ubicación”.	

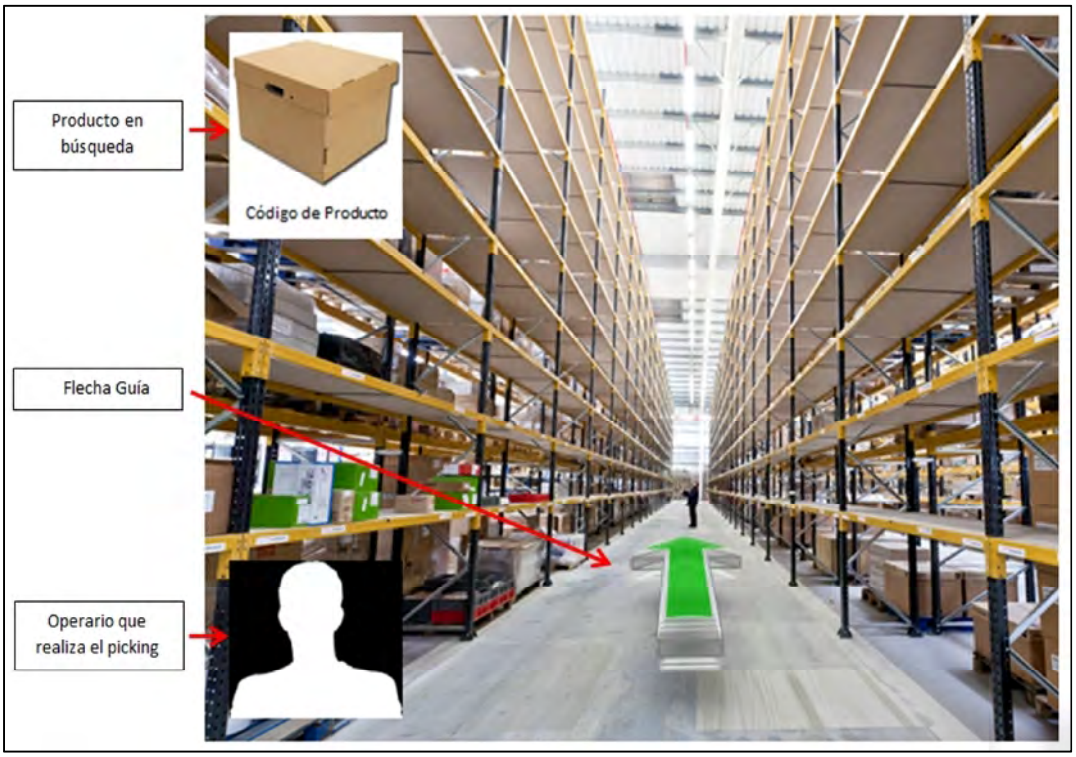


Figura 5. 19 Pantalla de visualización de usuario que realiza el Picking- Guía. Fuente: Elaboración Propia.

Figura 5. 20 Pantalla de Visualización de usuario que realiza el Picking-Indicador de Ubicación. Fuente: Elaboración Propia.

Tabla 5. 51 Clase: Vista Operario

Clase	VistaOperario		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	IngresoSistema	
	Descripción	Método que valida el ingreso al sistema mediante la invocación del método “ValidarIngreso” de la clase “ControlOperario”.	
	Nombre	MostrarImagenOp	
	Descripción	Método que muestra la imagen del operario del *Picking*, mediante la invocación del método “ImagenOpControl” de la clase “ControlOperario”.	
	Nombre	MostrarImagenProducto	
	Descripción	Método que muestra la imagen del operario del *Picking*, mediante la invocación del método “ImagenProductoControl” de la clase “ControlOperario”.	
Esta pantalla es en realidad lo que muestra la cámara que utiliza el operario. Sobre esta pantalla se insertar objetos 3D que servirán para guiar al operario en la ruta.			

Tabla 5. 52 Clase: Control Operario

Clase	ControlOperario		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ValidarIngreso	
	Descripción	Método que realiza la validación del ingreso al sistema de los operarios que realizan el *Picking* mediante la invocación de los métodos “IdentificarMarcador” y “ValidarIngresoMarcador” de la clase “ControlUsuario”.	
	Nombre	ImagenOpControl	
	Descripción	Método que invoca al método “SesionActiva” de la clase “ControlRealizarPedido” e inserta la variable de retorno en el método “ImagenOp” de la clase “GestionOperario”. Este método devuelve el valor de retorno del segundo método.	
	Nombre	ImagenProductoControl	
	Descripción	Método que inserta el atributo “idProducto” en el método “ImagenProducto” de la clase “GestionOperario”.	
	Nombre	LectorUbicacion	
	Descripción	Método que reconoce los marcadores de tipo ubicación. Y determina la ubicación del operario en el almacén.	
	Nombre	PreparacionPicking	
	Descripción	Método que permite realizar el recojo de productos. El proceso consta de 5 pasos: El primero, invoca al método “PedidoenProceso” de la clase “GestionOperario”, para obtener el atributo “idPedido”. El segundo es insertar el atributo “idPedido” en el método “ListadoProductoUbicacion” de la clase “GestionOperario” y obtener el listado de productos con respectivas ubicaciones. El tercer paso es insertar el array obtenido del paso anterior en el método “OptimizarRuta” de la clase “ControlLogistica”. Este método devuelve un array con la ruta optimizada y el atributo “idPedido”	

	Nombre	*Picking*
	Descripción	Método que realiza las operaciones que facilitan el *Picking*. Es un proceso iterativo que continúa hasta que son recogidos todos los productos o se finaliza el proceso. El primer paso es obtener la ruta optimizada del método "Preparacion*Picking*" de esta clase. Luego obtener el "idProducto" del primer producto e insertarlo en el método "MostrarImagenProducto" de la clase "VistaOperario", con esto se puede observar la imagen del producto en pantalla. El segundo paso es guiar al operario para que llegue al producto, para ello se debe invocar al método "Guiar" de la clase "RA" e insertar el valor de retorno del método "LectorUbicacion" de esta clase y la ubicación del producto buscado. El tercer paso es invocar al método "LectorConfirmacionProducto" de la clase "RA" e insertar los atributos "idProducto" e "idTrabajador", con este método se esperan tres posibles resultados y el proceso completo vuelve a ser iniciado hasta que se completo la lista de productos. El último paso es invocar al método "Cambiodeestadopedido" de la clase "GestionOperario" y cambiar de estado.

Tabla 5. 53 Clase: Gestión Operario

Clase	GestionOperario		
Atributos	Nombre	Tipo	Descripción
	VarFoto	String	Almacena la dirección del archivo de la foto del operario.
	VarFotoP	String	Almacena la dirección del archivo de la foto del producto.
Métodos	Nombre	ImagenOp	
	Descripción	Método que realiza una consulta a la base de datos. Requiere del atributo "idTrabajador" y devuelve el atributo "VarFoto".	
	Nombre	ImagenProducto	
	Descripción	Método que realiza una consulta a la base de datos. Requiere del atributo "idProducto" y devuelve el atributo "VarFotoP".	
	Nombre	PedidoenProceso	
	Descripción	Método que realiza una consulta a la base de datos para obtener un array con los "idPedido" de los pedidos en proceso.	

	Nombre	ListadoProductoUbicacion
	Descripción	Método que realiza una consulta a la base de datos y devuelve el idProducto, el nombre del producto y su ubicación (distancia en los tres ejes). Requiere el atributo "idPedido".
	Nombre	Cambiodeestadopedido
	Descripción	Método que realiza el cambio de estado de un pedido a "completado".
	Nombre	Cambiodeestadopedidoproceso
	Descripción	Método que realiza el cambio de estado de un pedido, de "en proceso" a "asignado".
	Nombre	ConsultaidTrabajador
	Descripción	Método que devuelve el atributo "idTrabajador" relacionado con un pedido específico, para ello requiere el atributo "idPedido".

Tabla 5. 54 Clase: Realidad Aumentada

Clase	RA		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	Guiar	
	Descripción	Método que incrusta una flecha en la pantalla que sirve como guía para el operario. Para realizar esta operación requiere de la posición actual y de la posición del producto buscado.	
	Nombre	LectorConfirmacionProducto	
	Descripción	Método capaz de reconocer el marcador de un producto determinado, la credencial del operario que realiza el *Picking* y un marcador predefinido como error. Para ello requiere de los atributos "idProducto" y "idTrabajador" devuelve tres valores booleanos dependiendo si son reconocidos los marcadores. Las opciones son las siguientes: La primera, si reconoce el marcador del producto y la credencial, expresa que se recogió la cantidad correcta de producto. La segunda, si reconoce el marcador del producto y el marcador de error, expresa que hay un error en la cantidad de producto. La tercera, si reconoce solo el marcador de error, significa que no se encontró el marcador del producto.	
En esta clase se encuentran almacenadas las librerías de realidad aumentada.			

Tabla 5. 55 Clase: Control Logística

Clase	ControlLogistica		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	OptimizarRuta	
	Descripción	Método que recibe el listado de productos y sus respectivas ubicaciones y devuelve la ruta optima que debe seguirse para que el recorrido total sea el mínimo. El algoritmo utiliza el punto de partida del operario el cual se obtiene mediante la invocación del método "ConsultaPuntoPartida" de la clase "GestionLogistica".	

Tabla 5. 56 Clase: Gestión Logística

Clase	GestionLogistica		
Atributos	Nombre	Tipo	Descripción
Métodos	Nombre	ConsultaPuntoPartida	
	Descripción	Método que realiza una consulta a la base de datos. Devuelve las coordenadas del marcador con el nombre "partida"	

4. Base de datos

Una base de datos es una colección de datos que contiene información relevante para una empresa. Se diseñan para gestionar grandes cantidades de información, en este caso, la información de un almacén.

El diseño propuesto sigue el modelo relacional de base de datos, el cual es una colección de tablas que representan tanto los datos como sus relaciones. [20]

A continuación veremos el modelo relacional de la base de datos del diseño de este *software* (Figura 5.21). La descripción de cada tabla de la base de datos .se encuentra en las Tablas de la 5.57 a 5.67.

[20] Cfr. Silberschatz, A; Korth, Henry y Sudarshan, S. (2006). *Fundamentos de diseño de bases de datos*. Mac Graw Hill

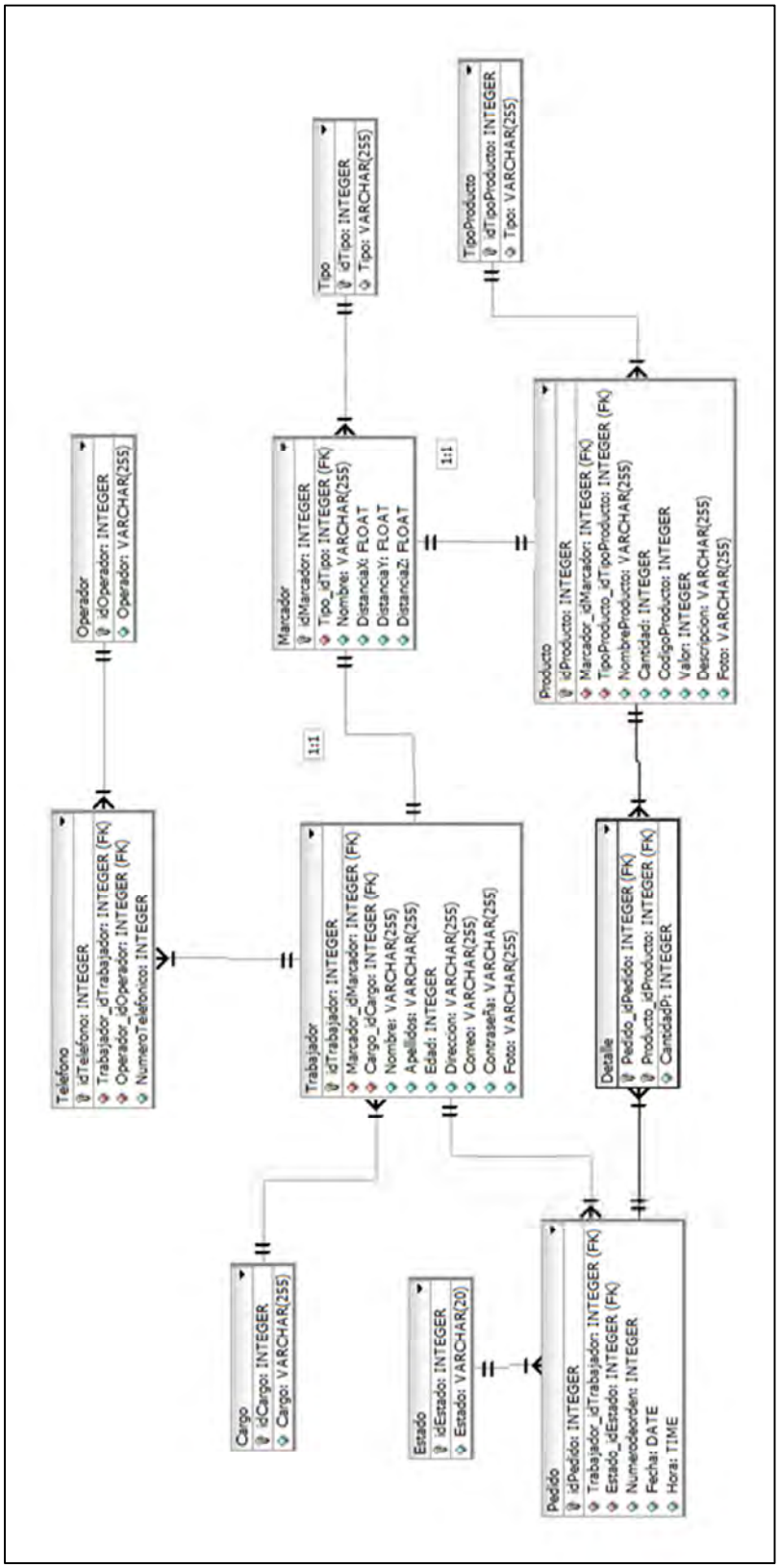


Figura 5. 21Modelo Relacional. Fuente: Elaboración Propia.

Tabla 5. 57 Cargo

Cargo			
Contiene los cargos de los trabajadores.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idCargo	INTEGER	PK	Llave primaria de la tabla.
Cargo	VARCHAR(255)		Puesto que ocupa el trabajador en la empresa.

Tabla 5. 58 Detalle

Detalle			
Contiene la cantidad de cada producto de un pedido.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
Pedido_idPedido	INTEGER	PK, FK	Llave primaria de la tabla, referencia a la tabla Pedido.
Producto_Marcador_id Marcador	INTEGER	PK, FK	Llave primaria de la tabla, referencia a la tabla Marcador.
Producto_idProducto	INTEGER	PK, FK	Llave primaria de la tabla, referencia a la tabla Producto.
CantidadP	INTEGER		Cantidad de producto solicitado.

Tabla 5. 59 Estado

Estado			
Contiene el estado del pedido.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idEstado	INTEGER	PK	Llave primaria de la tabla.
Estado	VARCHAR(20)		Estado en el que se encuentra el pedido, puede ser: en proceso o completado.

Tabla 5. 60 Marcador

Marcador			
Contiene a los marcadores que utiliza el sistema.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idMarcador	INTEGER	PK	Llave primaria de la tabla.
Tipo_idTipo	INTEGER	FK	Llave foránea que hace referencia al Tipo de marcador.
Nombre	VARCHAR(255)		Ruta del archivo donde se encuentra el marcador.
DistanciaX	FLOAT		Distancia en el eje x.
DistanciaY	FLOAT		Distancia en el eje y.
DistanciaZ	FLOAT		Distancia en el eje z.

Tabla 5. 61 Operador

Operador			
Contiene a las empresas operadoras de teléfonos.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idOperador	INTEGER	PK	Llave primaria de la tabla.
Operador	VARCHAR(255)		Nombre de la empresa operadora. Puede ser claro, telefónica o nextel.

Tabla 5. 62 Pedido

Pedido			
Contiene a los pedidos que son solicitados.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idPedido	INTEGER	PK	Llave primaria de la tabla.
Trabajador_Marcador_idMarcador	INTEGER	FK	Llave foránea que hace referencia al marcador del Trabajador que realiza el pedido.
Trabajador_idTrabajador	INTEGER	FK	Llave foránea que hace referencia al Trabajador que realiza el pedido.
Estado_idEstado	INTEGER	FK	Llave foránea que hace referencia al Estado en el que se encuentra el pedido.

Numerodeorden	INTEGER		Numero de orden del pedido.
Fecha	DATE		Fecha en la que se hizo el pedido.
Hora	TIME		Hora en la que se emitió el pedido.

Tabla 5. 63 Producto

Producto			
Contiene a los productos del almacén y sus características.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idProducto	INTEGER	PK	Llave primaria de la tabla.
Marcador_idMarcador	INTEGER	FK	Llave foránea que hace referencia al Marcador de cada producto.
TipoProducto_idTipo Producto	INTEGER	FK	Llave foránea que hace referencia al Tipo de Producto.
NombreProducto	VARCHAR(255)		Nombre del producto.
Cantidad	INTEGER		Cantidad de producto en el almacén.
CodigoProducto	INTEGER		Código numérico del producto.
Valor	FLOAT		Valor monetario de producto.
Descripcion	VARCHAR(255)		Descripción del producto.
Foto	VARCHAR(255)		Foto del producto.

Tabla 5. 64 Teléfono

Telefono			
Contiene los teléfonos de los trabajadores.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idTelefono	INTEGER	PK	Llave primaria de la tabla.
Trabajador_Marcador _idMarcador	INTEGER	FK	Llave foránea que hace referencia al Marcador del trabajador que posee el número telefónico.
Trabajador_idTrabaja dor	INTEGER	FK	Llave foránea que hace referencia al Trabajador que posee el número telefónico.
Operador_idOperador	INTEGER	FK	Llave foránea que hace referencia al Operador del número telefónico.

NumeroTelefonico	INTEGER		Numero telefónico del trabajador.

Tabla 5. 65 Tipo

Tipo			
Contiene los tipos de marcadores.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idTipo	INTEGER	PK	Llave primaria de la tabla.
Tipo	VARCHAR(255)		Tipo de marcador. Los marcadores pueden ser tres tipos. El primero, es de tipo credencial. El segundo, es de tipo ubicación pues mediante ellos se determina la ubicación del operador. El tercer tipo esta relacionado con el producto.

Tabla 5. 66 Tipo de Producto

TipoProducto			
Contiene los tipos de productos del almacén.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idTipoProducto	INTEGER	PK	Llave primaria de la tabla.
Tipo	VARCHAR(255)		Tipo o familia de producto.

Tabla 5. 67 Trabajador

Trabajador			
Contiene información de los trabajadores del almacén.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idTrabajador	INTEGER	PK	Llave primaria de la tabla.
Marcador_idMarcador	INTEGER	FK	Llave foránea que hace referencia al Marcador del trabajador.
Cargo_idCargo	INTEGER	FK	Llave foránea que hace referencia al Cargo del trabajador.
Nombre	VARCHAR(255)		Nombre del trabajador.

Apellidos	VARCHAR(255)		Apellidos del trabajador
Edad	INTEGER		Edad del trabajador.
Direccion	VARCHAR(255)		Dirección del trabajador.
Correo	VARCHAR(255)		Correo electrónico del trabajador.
Contraseña	CHAR(32)		Contraseña de ingreso al sistema del trabajador. Encriptada con MD5.
Foto	VARCHAR(255)		Foto del trabajador.

5. Diagrama de Despliegue

Un diagrama de despliegue muestra los componentes del sistema desplegados en cada nodo, del mismo. Los nodos pueden ser procesadores (aquellos que tienen capacidad de procesamiento como una computadora, *tablet*, etc.) o dispositivos (una impresora).

Este diagrama facilitaría la implementación del *software* en el almacén al apreciar gráficamente la distribución de los componentes en los nodos.

El *software* se extiende a tres ambientes de la empresa:

El primero, es el área de sistema, donde se encuentra alojado el *software*. El segundo, el área de recepción de pedidos, donde se encuentra una terminal en la que el jefe de almacén tiene acceso al *software*. El tercer ambiente es el área de almacén donde se disponen de dispositivos como teléfonos inteligentes, *tablets* o lentes de realidad aumentada, los cuales servirán para que el operario pueda entrar al sistema y realice el *Picking*.

El jefe de almacén ingresa un nuevo pedido desde la terminal, el cual es asignado al operario que se encuentren con una sesión activa en el sistema. El operario realiza el *Picking* con la asistencia del *software* y el pedido es completado.

Los dispositivos que se requieren para el funcionamiento del sistema son: una computadora estándar conectada al servidor central de la empresa y dispositivos móviles con conexión a internet. Los dispositivos móviles pueden ser celulares con cámara, *tablets* o lentes de realidad aumentada los cuales son idóneos para este procedimiento debido a que dejan las manos libres.

El diagrama de despliegue propuesto se puede apreciar en la Figura 5.22

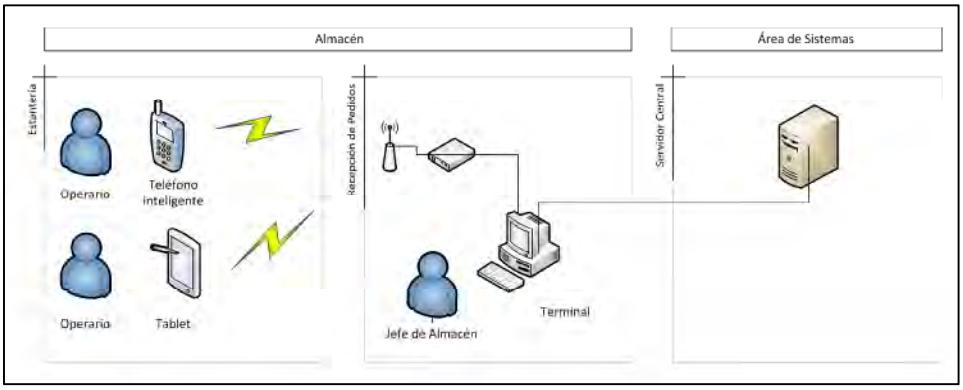


Figura 5. 22 Diagrama de Despliegue. Fuente: Elaboración Propia.

Capítulo 6
Prototipo

En el presente capítulo se expone el desarrollo del prototipo para concretar la propuesta del software de optimización del proceso de *Picking* mediante la tecnología de la realidad aumentada. El prototipo desarrollado consta de dos módulos, de optimización y de realidad aumentada. La finalidad de este prototipo es reducir el tiempo del proceso de *Picking*, guiando al operario por la ruta óptima mediante un dispositivo móvil.

Existe una estrecha relación entre ambos módulos; el módulo de optimización ejecuta los cálculos necesarios, luego proporciona los datos al módulo de realidad aumentada para presentarlos de manera apropiada mediante la superposición de gráficos con video en directo.

1. Módulo de optimización

El módulo de optimización calcula la ruta óptima que un operario de *Picking* debe seguir para recoger los productos de un almacén. El *software* utiliza un código base [21] que ejecutando el algoritmo *Simulated Annealing* calcula la ruta óptima.

El software permite personalizar el diseño del almacén, representando uno ya existente, mediante el ingreso de datos específicos como el número y el tamaño de los estantes, el ancho de los pasillos y el ancho de corredores.

---

[21] Cfr. Jeff Heaton. Heaton Research. 17 de junio de 201, de http://www.heatonresearch.com/articles/64/page1.html

Al iniciar el software se desarrollan tres etapas secuenciales, que son las siguientes:

1. Cálculo preliminar de la ruta óptima. En esta etapa el algoritmo calcula la ruta óptima sin tener en cuenta la distribución de los estantes, creando un grafo hamiltoniano que solo une a los productos.

2. Cálculo de la ruta óptima: En esta etapa el método llamado “mejora de ruta” adapta cada tramo del grafo inicial a la distribución del almacén, respetando estantes, corredores y pasillos.

3. Cálculo de dirección de marcadores: A partir de la ruta óptima obtenida en la fase anterior el método llamado “dirección marcador” calcula en cada tramo la dirección que deben tener los marcadores, lo que posibilita el desplazamiento del operador por la ruta óptima.

Además, el *software* permite la modificación de parámetros propios del algoritmo utilizado como; la temperatura y la variable delta; Un tercer parámetro que es el criterio de parada del algoritmo es calculado por el *software* mediante una fórmula determinada por simulación que tiene como único parámetro la cantidad de productos. El objetivo de la fórmula para determinar el criterio de parada es reducir el tiempo de cálculo sin comprometer el nivel de optimización.

Como se expone en la tabla 6.1, se promediaron los resultados de 50 simulaciones por cada combinación de producto y ciclos entre soluciones (número de ciclos que trascurren sin que la solución obtenida haya cambiado); de todas estas opciones se toma en cuenta el número de ciclos entre soluciones que sean inmediatamente menores al parámetro de la simulación. Por ejemplo, en la fila de productos con valor igual a 50 escogemos el número 54 porque es menor al valor del parámetro de la simulación igual a 60 de la columna de ciclos entre soluciones. Debemos notar que el parámetro de ciclos entre soluciones en la simulación se utilizó como menor y no menor igual y es el motivo por el cual no se escogió el valor de 39 de la fila antes mencionada. Cuando el valor de ciclo entre soluciones experimental es menor que el valor de la simulación significa que para esa combinación de valor de producto y ciclos entre soluciones ya no es posible encontrar mejores soluciones incrementando el valor del parámetro de criterio de parada. Con este criterio se determinan cuatro puntos sobre los cuales se realizó un ajuste polinómico, como se puede apreciar en la figura 6.1.

Tabla 6. 1

		Ciclos entre soluciones						Puntos obtenidos	
		10	**20**	**40**	**60**	**80**			
Productos	**10**	4	4	22	6	27		10	4
	20	7	14	29	19	52		20	14
	50	9	19	39	54	54		50	54
	100	9	19	39	59	74		100	74

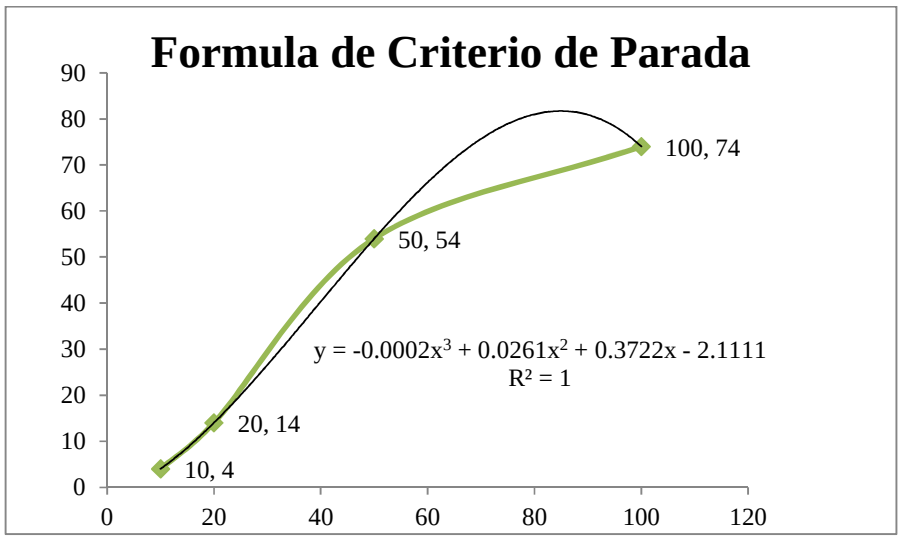


Figura 6. 1:

2. Módulo de Realidad Aumentada

La función del módulo de realidad aumentada consiste en guiar el recorrido de un operario en el proceso de *Picking*, combinando información generada por ordenador con video en directo[22]. En este caso, la información generada es proporcionada por el módulo de optimización.

[22] Cfr. Mullen, T.(2011). Realidad Aumentada: Crea tus propias aplicaciones. España. Anaya. 21/319

El *software* fue desarrollado en el IDE[23] k 4.6 con la estructura *Flarmanager* que comprende un conjunto de herramientas para el desarrollo de aplicaciones de internet y un código base desarrollado por Rob Hawkes [24] que permite el reconocimiento de múltiples marcadores.

El *software* requiere de una distribución especifica de marcadores en el almacén por medio de los cuales determinará la ubicación del dispositivo móvil (celular o tablet) utilizado y en consecuencia la del operario. Además, cada producto deberá tener un marcador para que el programa pueda reconocerlos.

El programa reconoce cinco tipos de marcadores, son los siguientes:

1. Credencial: Identifica los datos de cada operario de *Picking*.

2. Productos: Identifica los datos del producto como las coordenadas de su ubicación, su imagen y su cantidad.

3. Flechas guía: Identifica a los marcadores que permiten el desplazamiento del operario en los pasillos y corredores del almacén.

4. Flechas de estante: Permiten la búsqueda del producto en su respectivo estante.

5. Alerta: Es un marcador utilizado por el operario en caso haya alguna inconformidad en el proceso de *Picking* de un producto como; por ejemplo, la ausencia del producto.

[23] Por sus siglas en ingles, entorno de programación integrado. Es un *software* con herramientas de programación.

[24] Cfr. Rob Hawkes. Tracking multiple augmented reality markers with flarmanager and papervision. 15 de julio de 2012, de http://rawkes.com/articles/tracking-multiple-augmented-reality-markers-flarmanager-and-papervision#download

Conclusiones

El análisis y diseño del *software* que optimiza el *Picking* ha sido realizado satisfactoriamente por dos motivos: el primero; es la comprensión del proceso de *Picking* y su repercusión en la logística. El segundo; es la comprensión de la versátil tecnología de la Realidad Aumentada para aplicarla con éxito a la optimización de los procesos industriales.

Los principios fundamentales de la Ingeniería Industrial y de Sistemas permiten que el cambio del entorno se vea como una oportunidad para optimizar procesos industriales utilizando tecnologías innovadoras.

El profundo conocimiento de los fundamentos de la Ingeniería Industrial y de Sistemas permite reconocer el potencial de una nueva tecnología para aprovecharla en el campo industrial, así mimo, los procedimientos, y metodologías aprendidas a lo largo de la carrera permitieron realizar un correcto análisis y diseño de *software*.

La presente investigación es una propuesta que expone claramente como las nuevas tecnologías contribuyen a la optimización de procesos industriales, logrando una importante ventaja competitiva; en este caso la tecnología de la Realidad Aumentada aplicada a la optimización del *Picking.*

El algoritmo propuesto, *Simulated Annealing*, es el adecuado para resolver el problema de optimización típico de un almacén. En referencia a los pedidos, estos pueden ser de diversa extensión, algunos pueden incluir pocos productos, mientras que otros involucran un elevado número de ellos.

La utilización de la tecnología de la Realidad Aumentada supone una reducción considerable en los costos de capacitación, pues el diseño propuesto permite un manejo

intuitivo, desarrollando una correcta aplicación de la tecnología a las necesidades del trabajador que realiza el *Picking*, compensando las graves deficiencias del método tradicional.

Bibliografía

1. 2010. Realidad Aumentada. 14 de marzo de 2012, de http://www.realidadvirtual.com/realidad-aumentada/

2. Angulo, C. (1996). *Implementación del Simulated Annealing para la resolución de problemas de diseño de rutas de reparto con restricciones de capacidad y de tiempo en redes dispersas no euclideas*. Disertación doctoral de Ingeniería Industrial. Escuela Superior de Ingenieros Industriales de San Sebastián.

3. Angulo, C. (2011). *Investigación de Operaciones*. Universidad de Piura.

4. Jeff Heaton. Heaton Research. 17 de junio de 201, de http://www.heatonresearch.com/articles/64/page1.html

5. José Carlos Cortizo y Luis Ignacio Díaz. Madri+d Sistemas inteligentes. 15 de marzo de 2012, de http://www.madrimasd.org/blogs/sistemas_inteligentes/2008/03/21/87063

6. Juan Bonnin. 2008. Realidad Aumentada. 14 de marzo de 2012, de http://outernet.tk/

7. Kisoft Vision: un revolucionario sistema óptico de preparación de pedidos. 14 de marzo de 2012, de http://www.knapp.com/cms/cms.php?pageName=866&sid=5fadf7dca31cac9497f6eebbc574b6e8

8. Kotler, P. (2010). Caótica: *Administración y marketing en tiempos de caos*. Bogotá. Norma.

9. Mullen, T.(2011). Realidad Aumentada: Crea tus propias aplicaciones. España. Anaya.

10. López, R. (2006).*Operaciones de Almacenaje*. España. Thomson.

11. Lozano, J. (2002). *Cómo y dónde optimizar los costes logísticos*. España. Fundación Confemetal.

12. Mauleón, M. (2006). *Logística y Costos*. Madrid. Díaz de Santos.

13. Realidad Aumentada. 15 de marzo de 2012 de http://arealidadaumentada.wordpress.com/2012/02/20/que-es-la-realidad-aumentada/

14. Rob Hawkes. Tracking multiple augmented reality markers with flarmanager and papervision. 15 de julio de 2012, de http://rawkes.com/articles/tracking-multiple-augmented-reality-markers-flarmanager-and-papervision#download

15. Sommerville, I. (2005). *Ingeniería de Software*. Madrid. Pearson.

16. Yamilsalinas. 15 de marzo 2012, http://www.yamilsalinas.net/2009/11/17/%C2%BFque-es-la-realidad-aumentada-el-puente-entre-el-mundo-real-y-el-mundo-virtual/

Anexos

Anexo A: Módulo de Optimización

Anexo A-1: SimulateAnnealing.java

SimulateAnnealing

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package recocidosimulado;

/**
* Simulated Annealing and the Traveling Salesman
* Copyright 2005 by Heaton Research, Inc.
* by Jeff Heaton (http://www.heatonresearch.com) 12-2005
* ----------------------------------------------------
* This source code is copyrighted.
* You may reuse this code in your own compiled projects.
* However, if you would like to redistribute this source code
* in any form, you must obtain permission from Heaton Research.
* (support@heatonresearch.com).* This class implements SimulatedAnnealing.
*
```

```
* Want to learn more about Neural Network Programming in Java?
* Have a look at our e-book:
* http://www.heatonresearch.com/articles/series/1/
* @author Jeff Heaton (http://www.jeffheaton.com)
* @version 1.0
*/
public class SimulateAnnealing extends Thread {
/**
* The TravelingSalesman object that owns this object.
*/
protected SimulatedAnnealingClient owner;
/**
* The current temperature.
*/
protected double temperature;
/**
* The length of the current path.
*/
protected double pathlength;
/**
* The length of the best path.
*/
protected double minimallength;
static int selec=0;
/**
* The current order of cities.
*/
protected int order[];
/**
* The best order of cities.
*/
protected int minimalorder[];
        /**
        * Constructor
        * @param owner The TravelingSalesman class that owns this object.
        */
        SimulateAnnealing(SimulatedAnnealingClient owner){
                this.owner = owner;
                order = new int[owner.getCount()];
                minimalorder = new int[owner.getCount()];
        }
```

```
/**
* Called to determine if annealing should take place.
* @param d The distance.
* @return True if annealing should take place.
*/

public boolean anneal(double d){
	if (temperature < 1.0E-4) {
		if (d > 0.0)
		return true;
	else
		return false;
	}
	if (Math.random() < Math.exp(d / temperature))
		return true;
	else
		return false;
	}

/**
* Used to ensure that the passed in integer is within thr city range.
* @param i A city index.
* @return A city index that will be less than CITY_COUNT
*/

public int mod(int i){
	return i % owner.getCount();
}

/**
* Run as a background thread. This method is called to
* perform the simulated annealing.
*/

@Override
public void run(){
	long tiempoInicio= System.currentTimeMillis();int cycle=0;
	int sameCount = 0;
	temperature = owner.getStartingTemperature();initorder(order);
	initorder(minimalorder);
	pathlength = length1();
```

```
minimallength = pathlength;
int x=0; int y=0; int z=0; int zz=0;
int ciclos[] = new int[30];
int count=0;
owner.setStatus("Sistema en linea");
int productos;

double IgualResultado;
productos=(TravelingSalesman.ciudades);
IgualResultado=-0.0002*productos*productos*productos+
0.0261*productos*productos+ 0.3722*productos-2.1111;
while (sameCount<IgualResultado) {
        x++;
        // update the screen
        owner.update();
        owner.setStatus("Ciclo=" + cycle +
        "  Longitud=" + minimallength + "  Temperatura=" +
        temperature);

        // make adjustments to city order(annealing)

        for (int j2 = 0; j2 < owner.getCount() * owner.getCount(); j2++) {
                int i1=(int)Math.floor((double)owner.getCount()* Math.random());
                int j1 = (int)Math.floor((double)owner.getCount()* Math.random());
                double d = owner.getError(i1, i1 + 1) +
                owner.getError(j1, j1 + 1) - owner.getError(i1, j1) –
                owner.getError(i1 + 1, j1 + 1);
                if (anneal(d)) {
                        zz++;
                        if (j1 < i1) {
                                int k1 = i1;
                                i1 = j1;
                                j1 = k1;
                        }
                        for (; j1 > i1; j1--) {
                                int i2 = order[i1 + 1];
                                order[i1 + 1] = order[j1];
                                order[j1] = i2;
                                i1++;
                        }
                }
```

```
            }

            // See if this improved anything
            pathlength = length1();
            if (pathlength < minimallength) {
                        y++;

                        ciclos[count]=cycle;
                        count++;
                        minimallength = pathlength;
                        for(int k2 = 0; k2 < owner.getCount(); k2++)
                                    minimalorder[k2] = order[k2];
                                    sameCount=0;
            }else
                        sameCount++;
                        temperature = owner.getDelta() * temperature;
                        cycle++;
                        z++;
}

selec++;

TravelingSalesman.SolucionFinal=
recolectorordenador(TravelingSalesman.SolucionFinal,
TravelingSalesman.vx,TravelingSalesman.vy,0,1, TravelingSalesman.var);
TravelingSalesman.generarparametros( TravelingSalesman.SolucionFinal);

TravelingSalesman.mejoraderuta();

TravelingSalesman.solnueva=1;
long hora; long restohora; long minuto;
long restominuto; long segundo; long restosegundo;
long Ttotal=System.currentTimeMillis()-tiempoInicio;
hora=Ttotal/3600000;
restohora=Ttotal%3600000;
minuto=restohora/60000;
restominuto=restohora%60000;
segundo=restominuto/1000;
restosegundo=restominuto%1000;
owner.setStatus("Solución encontrada después de  "
+ cycle + " ciclos."
+ "                Longitud= "+ minimallength
```

```
            + "                Temperatura= "+temperature
            + "                Tiempo= "+minuto+" min "+segundo+" s");

            owner.update();
    }

    public static int [][] recolectorordenador(int [][] solucionfinal,int x,
    int y, int ox, int oy, int numCities){
            int array[][] = new int[numCities+1][2];
            int array2[][] = new int[numCities+1][2];
            array[0][0]=solucionfinal[0][2];
            array[0][1]=solucionfinal[0][3];
            array[1][0]=solucionfinal[0][0];
            array[1][1]=solucionfinal[0][1];

            for(int t=2;t<(array2.length);t++){
                    array[t][0]=solucionfinal[t-1][0];
                    array[t][1]=solucionfinal[t-1][1];
            }

            array2=ordenador(array, x, y, ox, oy);
            solucionfinal[0][2]=array2[0][0];
            solucionfinal[0][3]=array2[0][1];
            solucionfinal[0][0]=array2[1][0];
            solucionfinal[0][1]=array2[1][1];

            for(int t=2;t<(array2.length);t++){
                    solucionfinal[t-1][0]=array2[t][0];
                    solucionfinal[t-1][1]=array2[t][1];
            }
            return solucionfinal;
    }

    public static int [][] ordenador(int [][] array,int x, int y, int ox, int oy){
            int array2[][] = new int[array.length][array[0].length];
            int f=0; int or=0;
            for(int u=0; u<array.length; u++){
                    if(array[u][ox]==x && array[u][oy]==y){
                            f=u;
                    }
            }
```

```
            for(int u=f; u<array.length; u++){
                        array2[or][ox]=array[u][ox];
                        array2[or][oy]=array[u][oy];
                        or++;

            }
            for(int u=0; u<f; u++){
                        array2[or][ox]=array[u][ox];
                        array2[or][oy]=array[u][oy];
                        or++;
            }
            return array2;
}
/**
* Return the length of the current path through the cities.
* @return The length of the current path through the cities.
*/

public double length1(){
            double d = 0.0;
            for (int i = 1; i <= owner.getCount(); i++)
                        d += owner.getError(i, i - 1);
                        return d;
            }
/**
* Set the specified array to have a list of the cities in order.
* @param an An array to hold the cities.
*/

public void initorder(int an[]){
            for (int i = 0; i < owner.getCount(); i++)
            an[i] = i;
            }
}
```

Anexo A-2: TravelingSalesman.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package recocidosimulado;

import java.applet.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;

/**
* Simulated Annealing and the Traveling Salesman
* Copyright 2005 by Heaton Research, Inc.
* by Jeff Heaton (http://www.heatonresearch.com) 12-2005
* ----------------------------------------------------
* This source code is copyrighted.
* You may reuse this code in your own compiled projects.
* However, if you would like to redistribute this source code
* in any form, you must obtain permission from Heaton Research.
* (support@heatonresearch.com).
* This class implements the view of the path of cities.
* Want to learn more about Neural Network Programming in Java?
* Have a look at our e-book:
*
* http://www.heatonresearch.com/articles/series/1/
*
* @author Jeff Heaton (http://www.jeffheaton.com)
* @version 1.0
*/
public class TravelingSalesman extends Applet implements SimulatedAnnealingClient{
/**
* The cities to be visited.
*/
static private City cities[];
/**
* The simulated annealing worker class.
*/
private SimulateAnnealing worker;
```

```
/**
* The Start button.
*/
private Button ctrlStart;
/**
* The TextField that holds the number of cities.
*/
static private TextField ctrlCities;
static private TextField ctrlestantes;
static private TextField ctrlfilas;
static private TextField ctrlaltoestante;
static private TextField ctrlanchoestante;
static private TextField ctrlespacioverticalestante;
static private TextField ctrlespaciohorizontalestante;
static private TextField ctrlespacioinicioy;
static private TextField ctrlespacioiniciox;
/**
* The TextField for the starting temperature.
*/
private TextField ctrlTemp;
int abc=0;
int ab=0;
int prueba[]=new int [500];
int aleatoriosx[]=new int[800];
int aleatoriosy[]=new int[800];
static int ciudades;
public  static int SolucionFinal[][]=new int[800][4];
static int Parametros[][]=new int[800][8];
static int SolucionNueva[][]=new int[800][2];
static int aleatoriosT[][]=new int[800][3];
static int Marcadores[][]=new int[1000][6];
static int Prd[][]=new int[800][4];
static int var=0;
static int vx=105;
static int vy=40;
static int selector=0;
static int solnueva=0;
static int total=0;
static int tl=0;
static int CountMX=0;
static int l=0;
```

```
/**
* The TextField for the delta value.
*/
private TextField ctrlDelta;
/**
* Holds the buttons and other controls, forms a strip across
* the bottom of the applet.
*/
private Panel ctrlButtons;
/**
* Has the thread been started?
*/
private boolean started;
/**
* The current status, which is displayed just above the controls.
*/
private String status = "";
	/**
	* The init method, sets up the applet.
	*/

	@Override
	public void init(){
		setLayout(new BorderLayout());

		// setup the controls
		ctrlButtons = new Panel();
		ctrlStart = new Button("Iniciar");
		ctrlButtons.add(ctrlStart);
		ctrlButtons.add(new Label("# Productos"));
		ctrlButtons.add(ctrlCities = new TextField(3));
		ctrlButtons.add(new Label("Temperatura"));
		ctrlButtons.add(ctrlTemp = new TextField(2));
		ctrlButtons.add(new Label("Delta"));
		ctrlButtons.add(ctrlDelta = new TextField(3));
		ctrlButtons.add(new Label("Estantes"));
		ctrlButtons.add(ctrlestantes = new TextField(2));
		ctrlButtons.add(new Label("Filas"));
		ctrlButtons.add(ctrlfilas = new TextField(2));
		ctrlButtons.add(new Label("Alto E."));
		ctrlButtons.add(ctrlaltoestante = new TextField(3));
		ctrlButtons.add(new Label("Ancho E."));
```

```
		ctrlButtons.add(ctrlanchoestante = new TextField(3));
		ctrlButtons.add(new Label("E. V."));
		ctrlButtons.add(ctrlespacioverticalestante = new TextField(3));
		ctrlButtons.add(new Label("E. H."));
		ctrlButtons.add(ctrlespaciohorizontalestante = new TextField(3));
		ctrlButtons.add(new Label("E. I. X."));
		ctrlButtons.add(ctrlespacioiniciox = new TextField(3));
		ctrlButtons.add(new Label("E. I. Y."));

		ctrlButtons.add(ctrlespacioinicioy = new TextField(3));
		this.add(ctrlButtons, BorderLayout.SOUTH);

		ctrlTemp.setText("10");
		ctrlDelta.setText("0.99");
		ctrlCities.setText("10");
		ctrlestantes.setText("20");
		ctrlfilas.setText("6");
		ctrlaltoestante.setText("50");
		ctrlanchoestante.setText("25");
		ctrlespacioverticalestante.setText("30");
		ctrlespaciohorizontalestante.setText("30");
		ctrlespacioinicioy.setText("50");
		ctrlespacioiniciox.setText("100");

		// add an action listener for the button
			ctrlStart.addActionListener(new ActionListener(){
				public void actionPerformed(ActionEvent arg0){
				// System.out.println(arg0.getActionCommand());
				selector=1;
				solnueva=0;
				update();
				startThread();
				update();
			}
		});
}
/**
* Called when the applet should be repainted.
* @param g The Graphics object used to paint.
*/
```

```
@Override
public void paint(Graphics g){
	update();
}

/**
* Called to get the number of cities.
* @return The number of cities.
*/

@Override
public int getCount(){
	return cities.length;
}

/**
* Returns the distance between two cities.
* @param i The first city.
* @param j The second city.
* @return The distance between the two cities.
*/

@Override
public double getError(int i, int j){
	int c1 = worker.order[i % cities.length];
	int c2 = worker.order[j % cities.length];
	return cities[c1].proximity(cities[c2]);
}

/**
* Returns the starting temperature for the
* annealing process.
* @return The starting temperature for the annealing process.
*/

@Override
public double getStartingTemperature(){
	double result0;
	try{
		result0 = new Double(ctrlTemp.getText()).doubleValue();
	} catch (NumberFormatException e){
		result0 = 10;
```

```
		}
		return result0;
	}

	public double getEstantes(){
		double result1;
		try{
			result1 = new Double(ctrlestantes.getText()).doubleValue();
		} catch (NumberFormatException e){
			result1 = 10;
		}
		return result1;
	}

	public double getfilas(){
		double result1;
		try{
			result1 = new Double(ctrlestantes.getText()).doubleValue();
		} catch (NumberFormatException e){
			result1 = 10;
		}
		return result1;
	}

	public double getespacioiniciox(){
		double result1;
		try{
			result1 =
			new Double (ctrlespacioiniciox.getText()).doubleValue();
		} catch (NumberFormatException e){
			result1 = 10;
		}
			return result1;
		}

	public double getespacioinicioy(){
		double result1;
		try{
			result1 =
			new Double(ctrlespacioinicioy.getText()).doubleValue();
		} catch (NumberFormatException e){
```

```
                result1 = 10;
        }
        return result1;
}

public double getespaciohorizontalestante(){
        double result1;
        try{
                result1 =
                new Double(ctrlespaciohorizontalestante.getText()).
                doubleValue();
        } catch (NumberFormatException e){
                result1 = 10;
        }
        return result1;
}

public double getespacioverticalestante(){
        double result1;
        try{
                result1 =
                new Double(ctrlespacioverticalestante.getText()).
                doubleValue();
        } catch (NumberFormatException e){
                result1 = 10;
        }
        return result1;
}

public double getanchoestante(){
        double result1;
        try{
                result1 =
                new Double(ctrlanchoestante.getText()).
                doubleValue();
        } catch (NumberFormatException e){
                result1 = 10;
        }
        return result1;
}

/**
```

```
* Called to determine if annealing should take place.
* @param d The distance.
* @return True if annealing should take place.
*/

@Override
public double getDelta(){
        double result2;
        try{
                result2 =
                new Double(ctrlDelta.getText()).doubleValue();
        } catch (NumberFormatException e){
                result2 = 10;
        }
        return result2;
}
Public void calcularmarcador(){
        //calcular marcadores del trayecto
        int Dist=0;
        int NMayor=0;
        int NMenor=0;

        for(int j=0;j<(TravelingSalesman.total+2);j++){
                blue.fillOval(SolucionNueva[j][0]-3,SolucionNueva[j][1]-3,6,6);
        }

        int puntox=0;  int puntoy=0;  int btx=0; int bty=0;
        int ti=0; int ou=0; float gt=0;
        for(int j=0;j<TravelingSalesman.total-1;j++){
                for(int i=0;i<CountMX;i++){
                        try{
                                numCities = Integer.parseInt(ctrlCities.getText());
                        } catch (NumberFormatException e){}
                        gt=0;
                        for(int jr=1;jr<numCities+1;jr++){
                                if(SolucionNueva[j][0]==aleatoriosT[jr][0] &&
                                SolucionNueva[j][1]==aleatoriosT[jr][1] ){
                                        gt=1;
                                }
                        }
                        btx=0; bty=0; ou=0;  ti=0;
```

```
do{
        if(SolucionNueva[j][0]==SolucionNueva[j+1][0]
        && gt==0){
                if(SolucionNueva[j][1]>SolucionNueva[j+1][1]){
                        NMayor=SolucionNueva[j][1];
                        NMenor=SolucionNueva[j+1][1];
                }
                if(SolucionNueva[j][1]<SolucionNueva[j+1][1]){
                        NMayor=SolucionNueva[j+1][1];
                        NMenor=SolucionNueva[j][1];
                }
                puntox=(SolucionNueva[j][0]);
                puntoy=(bty*5)+(NMenor);
                bty++;
                if(puntoy>=NMenor && puntoy<NMayor){
                        ou=0;
                }else{
                        ti=1; ou=1;
                }
        }
        if(SolucionNueva[j][1]==SolucionNueva[j+1][1]
        && gt==0){
                 if(SolucionNueva[j][0]>SolucionNueva[j+1][0]){
                        NMayor=SolucionNueva[j][0];
                        NMenor=SolucionNueva[j+1][0];
                }
                if(SolucionNueva[j][0]<SolucionNueva[j+1][0]){
                        NMayor=SolucionNueva[j+1][0];
                        NMenor=SolucionNueva[j][0];
                }
                puntox=(btx*5)+(NMenor);
                puntoy=(SolucionNueva[j][1]);
                btx++;
                if(puntox>=NMenor && puntox<NMayor){
                        ou=0;
                }else{
                        ti=1; ou=1;
                }
        }


        if(distancia(Marcadores[i][0],Marcadores[i][1],puntox,
```

```
                                puntoy)<20 && ou==0){
                                        Marcadores[i][4]=j;
                                        rojo.drawLine(Marcadores[i][0],
                                        Marcadores[i][1], puntox, puntoy);
                                        yellow.fillOval(Marcadores[i][0]-2,
                                        Marcadores[i][1]-2,4,4);
                                        Marcadores[i][2]=2; //en la ruta
                                }
                        }while(ti==0);
                }
        }
        for(int i=0;i<CountMX;i++){
                btx=0; bty=0; ou=0; ti=0;
                do{
                        if(SolucionNueva[TravelingSalesman.total-1][0]==
                        SolucionNueva[0][0]){
                                if(SolucionNueva[TravelingSalesman.total-1][1]>
                                SolucionNueva[0][1]){
                                        NMayor=
                                        SolucionNueva[TravelingSalesman.total-1][1];
                                        NMenor=SolucionNueva[0][1];
                                }
                                if(SolucionNueva[TravelingSalesman.total-1][1]<
                                SolucionNueva[0][1]){
                                        NMayor=SolucionNueva[0][1];
                                        NMenor=
                                        SolucionNueva[TravelingSalesman.total-1][1];
                                }
                                puntox=(SolucionNueva[TravelingSalesman.total-1][0]);
                                puntoy=(bty*5)+(NMenor);
                                bty++;
                                if(puntoy>=NMenor && puntoy<NMayor){
                                        ou=0;
                                }else{
                                        ti=1; ou=1;
                                }
                        }
                        if(SolucionNueva[TravelingSalesman.total-1][1]==
                        SolucionNueva[0][1]){
```

```
                                    if(SolucionNueva[TravelingSalesman.total-1][0]>
                                    SolucionNueva[0][0]){
                                            NMayor=
                                            SolucionNueva[TravelingSalesman.total-1][0];
                                            NMenor=SolucionNueva[0][0];
                                    }
                                    if(SolucionNueva[TravelingSalesman.total-1][0]<
                                    SolucionNueva[0][0]){
                                            NMayor=SolucionNueva[0][0];
                                            NMenor=
                                            SolucionNueva[TravelingSalesman.total-1][0];
                                    }
                                    puntox=(btx*5)+(NMenor);
                                    puntoy=(SolucionNueva[TravelingSalesman.total-1][1]);
                                    btx++;
                                    if(puntox>=NMenor && puntox<NMayor){
                                            ou=0;
                                    }else{
                                            ti=1;  ou=1;
                                    }
                            }

                            if(distancia(Marcadores[i][0],Marcadores[i][1],puntox,puntoy)<
                            21 && ou==0){
                                    Marcadores[i][4]=TravelingSalesman.total-1;
                                    rojo.drawLine(Marcadores[i][0], Marcadores[i][1],
                                    puntox, puntoy);
                                    yellow.fillOval(Marcadores[i][0],Marcadores[i][1],4,4);
                                    Marcadores[i][2]=1; //en la ruta

                            }
                    }while(ti==0);
            }
    }
```

```
@Override
```

```
public  void update(){
        Image img = createImage(getBounds().width,
        getBounds().height);
        Graphics g       = img.getGraphics();
        Graphics blue    = img.getGraphics();
        Graphics w       = img.getGraphics();
        Graphics magenta = img.getGraphics();
        Graphics c1      = img.getGraphics();
        Graphics negro   = img.getGraphics();
        Graphics orange  = img.getGraphics();
        Graphics yellow  = img.getGraphics();
        Graphics rojo  = img.getGraphics();
        int width = getBounds().width;
        int height = getBounds().height;
        g.setColor(Color.black);
        g.fillRect(0, 0, width, height);
        rojo.setColor(Color.red);
        magenta.setColor(Color.MAGENTA);
        blue.setColor(Color.BLUE);
        w.setColor(Color.WHITE);
        negro.setColor(Color.BLACK);
        orange.setColor(Color.ORANGE);
        yellow.setColor(Color.YELLOW);
        c1.setColor(Color.LIGHT_GRAY);
        int altoestante=100;
        int anchoestante=50;

        int espacioverticalestante=50;
        int espaciohorizontalestante=50;
        int espacioinicioy=50;
        int espacioiniciox=100;
        int filas=2;
        int estantesporfilas=2;

        try{
                estantesporfilas =
                Integer.parseInt(ctrlestantes.getText());
        } catch (NumberFormatException e){}

        try{
                filas = Integer.parseInt(ctrlfilas.getText());
```

```
} catch (NumberFormatException e) {}

try{
        altoestante = Integer.parseInt(ctrlaltoestante.getText());
} catch (NumberFormatException e) {}

try{
        anchoestante =
        Integer.parseInt(ctrlanchoestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioverticalestante =
        Integer.parseInt(ctrlespacioverticalestante.getText());
} catch (NumberFormatException e) {}

try{
        espaciohorizontalestante =
        Integer.parseInt(ctrlespaciohorizontalestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioinicioy =
        Integer.parseInt(ctrlespacioinicioy.getText());
} catch (NumberFormatException e) {}


try{
        espacioiniciox =
        Integer.parseInt(ctrlespacioiniciox.getText());
} catch (NumberFormatException e) {}

int markX=0;
int markY=0;
for(int p=0; p<filas; p++){
        for(int n=0; n<estantesporfilas; n++){
                for (int xx = (n*(anchoestante +
                espaciohorizontalestante)) + espacioiniciox;
                xx < (espacioiniciox+anchoestante)+
                ((anchoestante + espaciohorizontalestante)*
                n); xx++){
                        for(int yy =(p*(altoestante +
```

```
                                        espacioverticalestante))+ espacioinicioy;
                                        yy<(espacioinicioy+altoestante)+
                                        ((altoestante+espacioverticalestante)*p); yy++){
                                                blue.fillOval(xx,yy,5,5);
                                        }
                                }
                        }
                }

        blue.fillOval(vx-15,vy-15,30,30);

        int x=0; int h; int y=0;
        int n=0; int z=0; int a=0;
        int b=0; int c=0; int fi=17;

        int Q[] = new int[estantesporfilas*2+1];
        int W[] = new int[estantesporfilas*2+1];
        int ejex[] = new int[estantesporfilas*2+1];

        for(int l=0; l<(estantesporfilas*2); l++){
                Q[l]=n; h=l; h=h%2;
                if(h==0){
                        x++; n=x;
                }
                if(h==1){
                        y++; n=y;
                }
        }

        for(int v=1; v<(estantesporfilas*2+1); v++){
                W[0]=0; W[v]=Q[v-1];
        }
        int G=0;

        for(int v=0; v<(estantesporfilas*2); v++){
                h=v; h=h%2;
                if(h==0){
                        G=5;
                }
                if(h==1){
                        G=-5;
```

```
            }
            ejex[v] = espacioinicio x  +  anchoestante *
            Q[v] +  espaciohorizontalestante*W[v] + G;
}

int QQ[] = new int[(filas*2+1)];
int WW[] = new int[(filas*2+1)];
int ejey[] = new int[(filas*2+1)];

for(int l=0; l<(filas*2); l++){
            QQ[l]=n; h=l; h=h%2;
            if(h==0){
                        x++; n=x;
            }
            if(h==1){
                        y++; n=y;
            }
}

for(int v=1; v<(filas*2+1); v++){
            WW[0]=0; WW[v]=QQ[v-1];
}

for(int v=0; v<(filas*2); v++){
            h=v; h=h%2;
            if(h==0){


                        G=5;
            }
            if(h==1){
                        G=-5;
            }
            ejey[v]=espacioinicioy +
            altoestante*Q[v] + espacioverticalestante*W[v]+G;
}

int D; int hh=0;
int DD=((ejey[1]-ejey[0])+1)*filas;
int ejeycompleto[] = new int[DD];
for(int k=0; k<filas; k++){
D=k*2;
```

```
            for(int t=ejey[D]; t<ejey[D+1]+1; t++){
                        c1.fillOval(ejex[1],t,5,5);
                        ejeycompleto[hh]=t;
                        hh++;
            }
}

for(int GG=0; GG<estantesporfilas*2; GG++ ){
            for(int ff=0; ff<DD; ff++ ){
                        w.fillOval(ejex[GG],ejeycompleto[ff],5,5);
            }
}


int contMx=0; int contMy=0;
int pk=0; int p=0;

for( pk=0; pk<ejex.length-1; pk++){
            for( p=0; p<ejey.length-1; p++){
                        if(pk%2==0){
                                    Marcadores[contMx][0]=ejex[pk]-4;
                                    Marcadores[contMx][1]=(ejey[p]);
                                    contMx++;
                        }else{
                                    Marcadores[contMx][0]=(ejex[pk]+5);
                                    Marcadores[contMx][1]=(ejey[p]);
                                    contMx++;
                        }
                        if(p%2==0){
                                    if(pk%2==0){
                                                Marcadores[contMx][0]=(ejex[pk]-4);
                                                Marcadores[contMx][1]=((ejey[p]+ejey[p+1])/2);
                                                contMx++;
                                    }else{
                                                Marcadores[contMx][0]=(ejex[pk]+5);
                                                Marcadores[contMx][1]=((ejey[p]+ejey[p+1])/2);
                                                contMx++;
                                    }
                        }

                        if(pk%2==0){
```

```
                        if(p%2==0){
                                Marcadores[contMx][0]= ((ejex[pk]+ejex[pk+1])/2);
                                Marcadores[contMx][1]=(ejey[p]-5);
                                contMx++;
                                Marcadores[contMx][3]=2;
                        }else{
                                Marcadores[contMx][0]= ((ejex[pk]+ejex[pk+1])/2);
                                Marcadores[contMx][1]=(ejey[p]+5);
                                contMx++;
                                Marcadores[contMx][3]=2;
                        }
                }
        }
}
CountMX=contMx;tl++;
for(int yll=0;yll<CountMX;yll++){
        if(Marcadores[yll][3]!=2){
                Marcadores[yll][3]=1;
        }
}

int lk=0;int aa=0;
int count=0; int numCities=0;

try{
        numCities = Integer.parseInt(ctrlCities.getText());
} catch (NumberFormatException e){}
var=numCities;
ciudades=numCities;
Random rnd=new Random();
int nn;
int mm;

for (int xp=0;xp<numCities+aa;xp++){
        int nnx= rnd.nextInt(ejex.length-1);
        int mmy= rnd.nextInt(ejeycompleto.length-1);
        for(int cc=0;cc<xp; cc++){
                if (xp==0){
                        lk=0;
                }else{
                        if(aleatoriosx[cc] ==ejex[nnx]&&
                                aleatoriosy[cc]==ejeycompleto[mmy]){
```

```
                                        lk++;
                                        aa++;
                                }
                        }
                }
                if (lk==0){  // guardar en arrays
                        aleatoriosx[count]=ejex[nnx];
                        aleatoriosy[count]=ejeycompleto[mmy];
                        count++;
                }
                lk=0;
        }
        abc++;

        if (cities != null){
                g.setColor(Color.green);
                for (int i = 0; i < cities.length; i++){
                        int xpos = cities[i].getx();
                        int ypos = cities[i].gety();
                        g.fillOval(xpos - 5, ypos - 5, 10, 10);
                }
                g.setColor(Color.white);
                if(solnueva==0){
                        for (int i = 0; i < cities.length; i++){
                                int icity = worker.minimalorder[i];
                                if (i != 0){

                                        int last =worker.minimalorder[i - 1];
                                        g.drawLine(cities[icity].getx(), cities[icity].gety(),
                                        cities[last].getx(),
                                        cities[last].gety());
                                        SolucionFinal[i-1][0]=cities[icity].getx();
                                        SolucionFinal[i-1][1]=cities[icity].gety();
                                        SolucionFinal[i-1][2]=cities[last].getx();
                                        SolucionFinal[i-1][3]=cities[last].gety();
                                }
                        }
                }
                if(solnueva==1){
                        g.setColor(Color.yellow );
                        for (int i = 0; i < TravelingSalesman.total-1; i++){
```

```
                                g.drawLine(SolucionNueva[i][0], SolucionNueva[i][1],
                                SolucionNueva[(i+1)][0], SolucionNueva[(i+1)][1]);
                                if(i==(TravelingSalesman.total-2)){
                                        g.setColor(Color.red );
                                        g.drawLine(SolucionNueva[0][0],
                                        SolucionNueva[0][1], SolucionNueva
                                        [(TravelingSalesman.total-1)][0],
                                        SolucionNueva[(TravelingSalesman.total-1)][1]);
                                }
                        }
                        g.setColor(Color.white);
                                calcularmarcador();
                                ubicacionmarcador();
                                }
                        g.setColor(Color.white);

                        }
                FontMetrics fm = g.getFontMetrics();
                g.drawString(status, 0, ctrlButtons.getBounds().y - fm.getHeight());
                getGraphics().drawImage(img, 0, 0, this);

        }
/**
* Called to determine if annealing should take place.
* @param d The distance.
* @return True if annealing should take place.
**/

        @Override
        public void setStatus(String status){
                this.status = status;
        }

        public static int [][] generarnumaleatorios(int altoestante,
                int anchoestante, int espacioverticalestante,int espaciohorizontalestante,
                int espacioinicioy,int espacioiniciox,int filas,int estantesporfilas){
                int x=0; int y=0; int n=0;
                int h; int z=0;int a=0;
                int b=0; int c=0; int fi=17;
                int Q[] = new int[estantesporfilas*2+1];
                int W[] = new int[estantesporfilas*2+1];
```

```
int ejex[] = new int[estantesporfilas*2+1];

for(int l=0; l<(estantesporfilas*2); l++){
        Q[l]=n; h=l; h=h%2;
        if(h==0){
                x++; n=x;
        }
        if(h==1){
                y++; n=y;
        }
}

for(int v=1; v<(estantesporfilas*2+1); v++){
        W[0]=0; W[v]=Q[v-1];
}
int G=0;
for(int v=0; v<(estantesporfilas*2); v++){
        h=v; h=h%2;
        if(h==0){
                G=5;
        }
        if(h==1){
                G=-5;
        }
        ejex[v]=espacioiniciox+anchoestante*Q[v]
        +espaciohorizontalestante*W[v]+G;
}

int QQ[] = new int[(filas*2+1)];
int WW[] = new int[(filas*2+1)];
int ejey[] = new int[(filas*2+1)];
for(int l=0; l<(filas*2); l++){
        QQ[l]=n; h=l; h=h%2;
        if(h==0){
                x++; n=x;
        }
        if(h==1){
                y++; n=y;
        }
}
```

```
for(int v=1; v<(filas*2+1); v++){
        WW[0]=0; WW[v]=QQ[v-1];
}

for(int v=0; v<(filas*2); v++){
        h=v; h=h%2;
        if(h==0){
                G=5;
        }
        if(h==1){
                G=-5;
        }
        ejey[v]=espacioinicioy+altoestante*Q[v]+
        espacioverticalestante*W[v]+G;
}

int D;
int DD;
DD=((ejey[1]-ejey[0])+1)*filas;
int ejeycompleto[] = new int[DD];
int hh=0;

for(int k=0; k<filas; k++){
D=k*2;
        for(int t=ejey[D]; t<ejey[D+1]+1; t++){
                ejeycompleto[hh]=t; hh++;
        }
}

int lk=0; int aa=0; int count=0; int numCities=0;
try{
        numCities = Integer.parseInt(ctrlCities.getText());
} catch (NumberFormatException e){}

ciudades=numCities;
Random rnd=new Random();
int nn;
int mm;
int aleatoriosx[] = new int[numCities+1];

int aleatoriosy[] = new int[numCities+1];
```

```
		int aleatorios[][] = new int[numCities+1][2];

		for (int xp=0;xp<numCities+aa;xp++){
			int nnx= rnd.nextInt(ejex.length-1);
			int mmy= rnd.nextInt(ejeycompleto.length-1);
			for(int cc=0;cc<xp; cc++){
				if (xp==0){
					lk=0;
				}else{
					if(aleatoriosx[cc]==ejex[nnx] &&
					aleatoriosy[cc]==ejeycompleto[mmy]){
						lk++;
						aa++;
					}
				}
			}
			if (lk==0){  // guardar en arrays
				aleatoriosx[count]=ejex[nnx];
				aleatoriosy[count]=ejeycompleto[mmy];
				count++;
			}
			lk=0;
		}
		for(int v=0;v<numCities;v++){
			aleatorios[v][0]=aleatoriosx[v];
			aleatorios[v][1]=aleatoriosy[v];
		}
		return aleatorios;
	}

	/**
	* Start the background thread.
	*/

	public static void ubicacionmarcador(){
		System.out.println("----------añadir direccion a marcadores---------");
		//--------------------añadir direccion a marcadores--------------------
		int ord=1;
		int Prdsolicitado=1;
		int numCities=0;
		int orden[][] = new int[100][3];
```

```
try{
numCities = Integer.parseInt(ctrlCities.getText());
} catch (NumberFormatException e){}

for(int ll=0;ll<=TravelingSalesman.total-1;ll++){
        for(int yll=0;yll<numCities-1;yll++){
                if(SolucionNueva[ll][0]==Prd[yll][0] &&
                SolucionNueva[ll][1]==Prd[yll][1]){
                        orden[ord-1][0]=SolucionNueva[ll][0];
                        orden[ord-1][1]=SolucionNueva[ll][1];
                        orden[ord-1][2]=ord;
                        //Prd[yll][2]=ord;
                        ord++;
                }
        }
}
int inicio=0;
int fin=0;
if(Prdsolicitado==1){
        inicio=0;
        for(int ll=0;ll<=TravelingSalesman.total-1;ll++){
                for(int yll=0;yll<numCities;yll++){
                        if(SolucionNueva[ll][0]==orden[Prdsolicitado][0] &&
                        SolucionNueva[ll][1]==orden[Prdsolicitado][1]){
                                fin=ll;
                        }
                }
        }
}
if(Prdsolicitado>1 && Prdsolicitado!=numCities ){
        for(int ll=0;ll<=TravelingSalesman.total-1;ll++){
                for(int yll=0;yll<numCities;yll++){
                        if(SolucionNueva[ll][0]==orden[Prdsolicitado-1][0] &&
                        SolucionNueva[ll][1]==orden[Prdsolicitado-1][1]){
                                inicio=ll;
                        }
                }
        }
        for(int ll=0;ll<=TravelingSalesman.total-1;ll++){
                for(int yll=0;yll<numCities;yll++){
                        if(SolucionNueva[ll][0]==orden[Prdsolicitado][0] &&
```

```
                                        SolucionNueva[ll][1]==orden[Prdsolicitado][1]){
                                                fin=ll;
                                        }
                                }
                        }
                }
                int Mayor;
                int Menor;

                for(int yll=inicio;yll<=fin;yll++){
                        for(int i=0;i<CountMX;i++){
                                int a=Marcadores[i][1];
                                int b=SolucionNueva[yll][1];
                                int c=SolucionNueva[yll+1][1];
                                int d=Marcadores[i][3];
                                int e=Marcadores[i][5];
                                int f=SolucionNueva[yll+1][0];
                                int g=SolucionNueva[yll+2][0];
                                int h=SolucionNueva[yll][0];
                                int ii=Marcadores[i][0];
                                int jj=SolucionNueva[yll+2][1];
                                if((yll+1)!=fin){
                                        if(h==f){    //igual x
                                                if(b>c){   //ay mayor
                                                        if(a<=b && a>=c){
                                                                if(d==1){ //tipo 1 =0
                                                                        e=0;
                                                                }
                                                                if(d==2){
                                                                        e=1;
                                                                }
                                                                if(d==2){
                                                                        e=2;
                                                                }
                                                        }else{
                                                                if(f>g){
                                                                        if(a<c){
                                                                                if(d==1 ){
                                                                                e=0;
                                                                                }
                                                                        }
```

```
                    }else{
                        if(a<c){
                            if(d==1 ){
                            e=2;
                            }
                        }
                    }
                }
            }else{   //by mayor
                if(a>=b && a<=c){
                    if(d==1){ //tipo 1 =0
                        e=0;
                    }
                    if(d==2 && ii<h){
                        e=2;
                    }
                    if(d==2 && ii>h){
                        e=1;
                    }
                }else{
                    if(f>g){
                        if(a>c){
                            if(d==1 ){
                            e=2;
                            }
                        }
                    }else{
                        if(a>c){
                            if(d==1 ){
                            e=1;
                            }
                        }
                    }
                }
            }
        }
        if(b==c){    // igual y
            if(h>f){  //ax mayor
                if(ii<=h && ii>=f){
                    if(d==2){ //tipo 1 =0
```

```
                    e=0;
                }
                if(d==1 && a<b){
                    e=2;
                }
                if(d==1 && a>b){
                    e=1;
                }
            }else{
                if(c>jj){
                    if(ii<f){
                        if(d==1 ){
                        e=1;
                        }
                    }
                }else{
                    if(ii<f){
                        if(d==1 ){
                        e=2;
                        }
                    }
                }
            }
        }else{   //bx mayor
            if(ii>=h &&ii<=f){
                if(d==2){  //tipo 1 =0
                    e=0;
                }
                if(d==1 && a<b){
                    e=1;
                }
                if(d==1 && a>b){
                    e=2;
                }
            }else{
                if(c>jj){
                    if(ii>f){
                        if(d==1 ){
                        e=1;
                        }
```

```
                                                    }

                                                }else{
                                                    if(ii>f){
                                                        if(d==1 ){
                                                        e=2;
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
}
//--------------------añadir direccion a marcadores--------------------

if( Prdsolicitado==numCities ){
        inicio=numCities;
        fin=0;
}

for(yll=inicio;yll<=fin;yll++){
        for(int i=0;i<CountMX;i++){
                int a=Marcadores[i][1];
                int b=SolucionNueva[yll][1];
                int c=SolucionNueva[yll+1][1];
                int d=Marcadores[i][3];
                int e=Marcadores[i][5];
                int f=SolucionNueva[yll+1][0];
                int g=SolucionNueva[yll+2][0];
                int h=SolucionNueva[yll][0];
                int ii=Marcadores[i][0];

                int jj=SolucionNueva[yll+2][1];
                if((yll+1)!=fin){
                        if(h==f){    //igual x
                                if(b>c){   //ay mayor
                                        if(a<=b && a>=c){
                                                if(d==1 ){ //tipo 1 =0
                                                        e=0;
                                                }
```

```
                    if(d==2  ){
                        e=1;
                    }
                    if(d==2 ){
                        e=2;
                    }
                }else{
                    if(f>g){
                        if(a<c){
                        if(d==1){
                        e=0;
                            }
                        }
                    }else{
                        if(a<c){
                        if(d==1 ){
                        e=2;
                            }
                        }
                    }
                }
            }else{//by mayor
                if(a>=b && a<=c){
                    if(d==1){ //tipo 1 =0
                        e=0;
                    }
                    if(d==2 && ii<h){
                        e=2;
                    }
                    if(d==2 && ii>h){
                        e=1;
                    }
                }else{

                    if(f>g){
                        if(a>c){
                        if(d==1 ){
                        e=2;
                            }
                        }
                    }else{
```

```
                                        if(a>c){
                                        if(d==1 ){
                                        e=1;
                                                }
                                        }
                                }
                        }
                }
}
if( b==c){    // igual y
                if(h>f){  //ax mayor
                        if(ii<=h && ii>=f){
                                if(d==2){
                                        e=0;
                                }
                                if(d==1 && a<b){
                                        e=2;
                                }
                                if(d==1 && a>b){
                                        e=1;
                                }
                        }else{
                                if(c>jj){
                                        if(ii<f){
                                        if(d==1 ){
                                        e=1;
                                                }
                                        }
                                }else{
                                        if(ii<f){
                                        if(d==1 ){
                                        e=2;
                                                }

                                        }
                                }
                        }
                }else{//bx mayor
                        if(ii>=h && ii<=f){
                                if(d==2){ //tipo 1 =0
                                        e=0;
                                }
```

```
                                                    if(d==1 && a<b){
                                                        e=1;
                                                    }
                                                    if(d==1 && a>b){
                                                        e=2;
                                                    }
                                                }else{
                                                    if(c>jj){
                                                        if(ii>f){
                                                        if(d==1 ){
                                                        e=1;
                                                            }
                                                        }
                                                    }else{
                                                        if(ii>f){
                                                        if(d==1 ){
                                                        e=2;
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }

            public static void  generarparametros(int [][] SolucionFinal2){

                int altoestante=100;
                int anchoestante=50;
                int espacioverticalestante=50;
                int espaciohorizontalestante=50;
                int espacioinicioy=50;
                int espacioiniciox=100;
                int filas=2;
                int estantesporfilas=2;
                int numCities=0;
```

```
try{
        numCities = Integer.parseInt(ctrlCities.getText());
} catch (NumberFormatException e){}

try{
        estantesporfilas = Integer.parseInt(ctrlestantes.getText());
} catch (NumberFormatException e) {}

try{
        filas = Integer.parseInt(ctrlfilas.getText());
} catch (NumberFormatException e) {}

try{
        altoestante = Integer.parseInt(ctrlaltoestante.getText());
} catch (NumberFormatException e) {}

try{
        anchoestante = Integer.parseInt(ctrlanchoestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioverticalestante =
        Integer.parseInt(ctrlespacioverticalestante.getText());
} catch (NumberFormatException e) {}

try{
        espaciohorizontalestante =
        Integer.parseInt(ctrlespaciohorizontalestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioinicioy = Integer.parseInt(ctrlespacioinicioy.getText());
} catch (NumberFormatException e) {}


try{
        espacioiniciox = Integer.parseInt(ctrlespacioiniciox.getText());
} catch (NumberFormatException e) {}

int x=0; int y=0; int n=0; int h; int z=0;
int a=0; int b=0; int c=0; int fi=17;
```

```
int Q[] = new int[estantesporfilas*2+1];
int W[] = new int[estantesporfilas*2+1];
int ejex[] = new int[estantesporfilas*2+1];
for(int l=0; l<(estantesporfilas*2); l++){
        Q[l]=n;  h=l; h=h%2;
        if(h==0){
                x++; n=x;
        }
        if(h==1){
                y++; n=y;
        }
}
for(int v=1; v<(estantesporfilas*2+1); v++){
        W[0]=0; W[v]=Q[v-1];
}
int G=0;
for(int v=0; v<(estantesporfilas*2); v++){
        h=v; h=h%2;
        if(h==0){
                G=5;
        }
        if(h==1){
                G=-5;
        }
        ejex[v]=espacioiniciox+anchoestante*
        Q[v]+espaciohorizontalestante*W[v]+G;
}
int QQ[] = new int[(filas*2+1)];
int WW[] = new int[(filas*2+1)];
int ejey[] = new int[(filas*2+1)];

for(int l=0; l<(filas*2); l++){
        QQ[l]=n; h=l; h=h%2;

        if(h==0){
                x++; n=x;
        }
        if(h==1){
                y++; n=y;
        }
}
```

```
for(int v=1; v<(filas*2+1); v++){
        WW[0]=0; WW[v]=QQ[v-1];
}
for(int v=0; v<(filas*2); v++){
        h=v; h=h%2;
        if(h==0){
                G=5;
}
        if(h==1){
                G=-5;
        }
        ejey[v]=espacioinicioy+altoestante*
        Q[v]+espacioverticalestante*W[v]+G;
}

int D;
int DD;
DD=((ejey[1]-ejey[0])+1)*filas;
int ejeycompleto[] = new int[DD];
int hh=0;

for(int k=0; k<filas; k++){
        D=k*2;
        for(int t=ejey[D]; t<ejey[D+1]+1; t++){
                ejeycompleto[hh]=t;
                hh++;
        }
}

for(int df=0; df<(ejey.length-1);df++){
        int xh;
        xh=(ejey[df]+ejey[df+1])/2;

}

for(int i=1; i<(cities.length-1);i++){
        for(int t=0;t<ejex.length;t++){
                //se analiza cada fila del arreglo solucion final
                //Parametros la primera columna corresponde al estante al que
                pertenece
                // la segunda columna se refiere al lado en el que se encuentra
                en el estante
```

```
                    if(SolucionFinal2[0][2]==ejex[t]){
                            Parametros[0][6]=SolucionFinal2[0][2];
                            int l; int u; l=t%2;
                            if(l==0){
                                    Parametros[0][0]=((t/2)+1);
                                    Parametros[0][2]=0;
                            }
                            if(l==1){
                                    u=t-1;
                                    Parametros[0][0]=((u/2)+1);
                                    Parametros[0][2]=1;
                            }
                    }
                    if(SolucionFinal2[0][0]==ejex[t]){
                            Parametros[1][6]=SolucionFinal2[0][0];
                            int l; int u; l=t%2;
                            if(l==0){
                                    Parametros[1][0]=((t/2)+1);
                                    Parametros[1][2]=0;
                            }
                            if(l==1){
                                    u=t-1;
                                    Parametros[1][0]=((u/2)+1);
                                    Parametros[1][2]=1;
                            }
                    }
                    if(SolucionFinal2[i][0]==ejex[t]){
                            Parametros[i+1][6]=SolucionFinal2[i][0];
                            int l; int u; l=t%2;
                            if(l==0){
                                    Parametros[i+1][0]=((t/2)+1);
                                    Parametros[i+1][2]=0;
                            }

                            if(l==1){
                                    u=t-1;
                                    Parametros[i+1][0]=((u/2)+1);
                                    Parametros[i+1][2]=1;
                            }
                    }
            }
```

```
}

for(int op=1;op<filas+1;op++){
        int k;
        if(SolucionFinal2[0][3]>=ejey[((op*2)-1)-1] &&
        SolucionFinal2[0][3]<=ejey[((op*2)-1)] ){

                Parametros[0][7]=SolucionFinal2[0][3];
                Parametros[0][1]=op;
                k=((ejey[((op*2)-1)])+(ejey[((op*2)-1)-1]))/2;

                if(SolucionFinal2[0][3]>=ejey[((op*2)-1)-1] &&
                SolucionFinal2[0][3]<=k){
                        Parametros[0][3]=1;
                }else{
                        Parametros[0][3]=0;
                }
        }
        if(SolucionFinal2[0][1]>=ejey[((op*2)-1)-1] &&
        SolucionFinal2[0][1]<=ejey[((op*2)-1)] ){
                Parametros[1][7]=SolucionFinal2[0][1];
                Parametros[1][1]=op;
                k=((ejey[((op*2)-1)])+(ejey[((op*2)-1)-1]))/2;

                if(SolucionFinal2[0][1]>=ejey[((op*2)-1)-1] &&
                SolucionFinal2[0][1]<=k){
                        Parametros[1][3]=1;
                }else{
                        Parametros[1][3]=0;
                }
        }

        for(int i=1; i<(cities.length-1);i++){
                if(SolucionFinal2[i][1]>=ejey[((op*2)-1)-1] &&
                SolucionFinal2[i][1]<=ejey[((op*2)-1)] ){
                        Parametros[i+1][7]=SolucionFinal2[i][1];
                        Parametros[i+1][1]=op;
                        k=((ejey[((op*2)-1)])+(ejey[((op*2)-1)-1]))/2;
                        if(SolucionFinal2[i][1]>=ejey[(((op*2)-1)-1)] &&
                        SolucionFinal2[i][1]<=k){
                                Parametros[i+1][3]=1;
                        }else{
```

```
                                        Parametros[1+1][3]=0;
                                    }
                                }
                            }
                        }

                        for(int hht=0; hht<cities.length; hht++){
                            if(Parametros[hht][0]==Parametros[hht+1][0]){
                                //se almacena en la columna 4 de parametros el lado del
                                producto ---"Contiguo X"
                                Parametros[hht][4]=1;
                            }else{
                                Parametros[hht][4]=0;
                            }
                        }

                        for(int yh=0; yh<(cities.length-1); yh++){
                            if(Parametros[yh][1]==Parametros[yh+1][1]){
                                //se almacena en la columna 5 de parametros el lado del
                                producto ---"Contiguo y"
                                Parametros[yh][5]=1;
                            }else{
                                Parametros[yh][5]=0;
                            }
                        }

                        Parametros[0][0]=1; Parametros[0][1]=1;
                        Parametros[0][2]=0; Parametros[0][3]=1;
                        Parametros[0][4]=0; Parametros[0][5]=0;
                        Parametros[0][6]=vx; Parametros[0][7]=vy;

    }

    public static void mejoraderuta(){
        //Comparacion de caso
        //Estante fila lado posicion ContiguoX ContiguoY----Parametros[][]
        int altoestante=100;
        int anchoestante=50;
        int espacioverticalestante=50;
        int espaciohorizontalestante=50;
        int espacioinicioy=50;
        int espacioiniciox=100;
```

```
int filas=2;
int estantesporfilas=2;
int numCities=0;

try{
        numCities = Integer.parseInt(ctrlCities.getText());
} catch (NumberFormatException e){}

try{
        estantesporfilas = Integer.parseInt(ctrlestantes.getText());
} catch (NumberFormatException e) {}

try{
        filas = Integer.parseInt(ctrlfilas.getText());
} catch (NumberFormatException e) {}

try{
        altoestante = Integer.parseInt(ctrlaltoestante.getText());
} catch (NumberFormatException e) {}

try{
        anchoestante = Integer.parseInt(ctrlanchoestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioverticalestante =
        Integer.parseInt(ctrlespacioverticalestante.getText());
} catch (NumberFormatException e) {}

try{
        espaciohorizontalestante =
        Integer.parseInt(ctrlespaciohorizontalestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioinicioy =
        Integer.parseInt(ctrlespacioinicioy.getText());
} catch (NumberFormatException e) {}



try{
```

```
            espacioiniciox =
            Integer.parseInt(ctrlespacioiniciox.getText());
} catch (NumberFormatException e) {}

int ll=0;
int ejex[] = new int[estantesporfilas*2+1];
int ejey[] = new int[(filas*2+1)];

do{
            ll=1; int x=0; int y=0; int n=0; int h;
            int z=0; int a=0; int b=0; int c=0; int fi=17;
            int Q[] = new int[estantesporfilas*2+1];
            int W[] = new int[estantesporfilas*2+1];

            for(int l=0; l<(estantesporfilas*2); l++){
                        Q[l]=n; h=l; h=h%2;
                        if(h==0){
                                    x++; n=x;
                        }
                        if(h==1){
                                    y++; n=y;
                        }
            }
            for(int v=1; v<(estantesporfilas*2+1); v++){
                        W[0]=0; W[v]=Q[v-1];
            }
            int G=0;
            for(int v=0; v<(estantesporfilas*2); v++){
                        h=v; h=h%2;
                        if(h==0){
                                    G=5;
                        }
                        if(h==1){
                                    G=-5;
                        }
                        ejex[v]=espacioiniciox+anchoestante*
                        Q[v]+espaciohorizontalestante*W[v]+G;
            }
```

```
int QQ[] = new int[(filas*2+1)];
int WW[] = new int[(filas*2+1)];

for(int l=0; l<(filas*2); l++){
        QQ[l]=n; h=l; h=h%2;
        if(h==0){
                x++; n=x;
        }
        if(h==1){
                y++; n=y;
        }
}
for(int v=1; v<(filas*2+1); v++){
        WW[0]=0; WW[v]=QQ[v-1];
}
for(int v=0; v<(filas*2); v++){
        h=v; h=h%2;
        if(h==0){
                G=5;
        }
        if(h==1){
                G=-5;
        }
        ejey[v]=espacioinicioy+altoestante*
        Q[v]+espacioverticalestante*W[v]+G;
}

int D;
int DD;
DD=((ejey[1]-ejey[0])+1)*filas;
int ejeycompleto[] = new int[DD];
int hh=0;

for(int k=0; k<filas; k++){
        D=k*2;
        for(int t=ejey[D]; t<ejey[D+1]+1; t++){
                ejeycompleto[hh]=t;
                hh++;
        }

}
}while(ll==0);
```

```
int PuntoX=0; int PuntoY=0; int PuntoXX=0;
int PuntoYY=0; int PuntoXXX=0; int PuntoYYY=0;
int PuntoXXXX=0; int PuntoYYYY=0; int NumNodos;
int ContadorNodos[]= new int[numCities+2];
int hu; int sumaNodos=0; float d1; float d2; float d3;
float d4; float d5; float dt1=0; float dt2=0;
int ft=(ejex[2]-ejex[1])/2; int caso;

for(int g=0; g<cities.length; g++){
        // caso inicio
        caso=1;
        if(g==0){
                caso=0;
                if(Parametros[(g+1)][2]==0){    // lado izquierdo
                        PuntoX=(Parametros[g+1][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(Parametros[g+1][6]-ft);
                        PuntoYY=(Parametros[g+1][7]);
                }else{    //lado derecho
                        PuntoX=(Parametros[g+1][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(Parametros[g+1][6]+ft);
                        PuntoYY=(Parametros[g+1][7]);
                }
                NumNodos=2;
                ContadorNodos[g]=NumNodos;
                sumaNodos=0;
                for(int yt=0; yt<(g);yt++){
                        sumaNodos=sumaNodos+ContadorNodos[yt];
                }
                hu=g+sumaNodos;
                SolucionNueva[hu][0]=Parametros[g][6];   ///en x
                SolucionNueva[hu][1]=Parametros[g][7];   ///en y
                SolucionNueva[(hu+1)][0]=PuntoX;
                SolucionNueva[(hu+1)][1]=PuntoY;
                SolucionNueva[(hu+2)][0]=PuntoXX;
                SolucionNueva[(hu+2)][1]=PuntoYY;
                sumaNodos=0;
        }
```

```
// caso 1
if(Parametros[g][0]!=Parametros[g+1][0] &&
Parametros[g][1]!=Parametros[g+1][1] &&
Parametros[g][2]==Parametros[g+1][2] &&
g<numCities){
        caso=0;
        //calcular si el punto esta arriba o abajo
        if(Parametros[g][1]>Parametros[g+1][1]){
                if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=((ejex[(((Parametros[g][0]*
                        2)-1)-1)])-ft);
                        PuntoYY=((ejey[((((Parametros[(g+1)][1])*
                        2)-1))])+ft);
                        PuntoXXX=((ejex[((((Parametros[(g+1)][0])*
                        2)-1)-1)])-ft);  // I
                        PuntoYYY=((ejey[((((Parametros[(g+1)][1])*
                        2)-1))])+ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=((ejex[(((Parametros[g][0])*2)-1)])+ft);
                        PuntoYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                        PuntoXXX=((ejex[((((Parametros[g+1][0])*
                        2)-1))])+ft);     // II
                        PuntoYYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                         PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }
        }else{
                if(g!=0){
                        if(Parametros[g][2]==0){

                                PuntoX=(Parametros[g][6]-ft);
                                PuntoY=(Parametros[g][7]);
                                PuntoXX=((ejex[(((Parametros[g][0]*
                                2)-1)-1)])-ft);
```

```
                PuntoYY=(ejey[(((Parametros[g+1][1]*
                2)-1)-1)])-ft;
                PuntoXXX=ejex[((Parametros[g+1][0]*
                2)-1)-1]-ft;  // I
                PuntoYYY=ejey[((Parametros[g+1][1]*
                2)-1)-1]-ft;
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }else{
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=ejex[((Parametros[g][0])*
                2)-1]+ft;
                PuntoYY=ejey[(((Parametros[g+1][1])*
                2)-1)-1]-ft;
                PuntoXXX=ejex[((Parametros[g+1][0]*
                2)-1)]+ft;
                PuntoYYY=ejey[((Parametros[g+1][1]*
                2)-1)-1]-ft;
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }
        }
        if(g==0){
            if(Parametros[g][2]==0){
                PuntoX=(Parametros[g][6]-ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=((ejex[(((Parametros[g][0]*
                2)-1)-1)])-ft);
                PuntoYY=
                ((ejey[(((((Parametros[(g+1)][1])*
                2)-1)-1)])-ft);
                PuntoXXX=
                ((ejex[(((((Parametros[(g+1)][0])*
                2)-1)-1)])-ft);  // I
                PuntoYYY=
                ((ejey[(((((Parametros[(g+1)][1])*
                2)-1)-1)])-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
```

```
                    }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=((ejex[(((Parametros[g][0])*
                        2)-1)])+ft);
                        PuntoYY=
                        ((ejey[(((((Parametros[g+1][1])*
                        2)-1)-1)])-ft);
                        PuntoXXX=
                        ((ejex[(((((Parametros[g+1][0])*
                        2)-1))])+ft);     // II
                        PuntoYYY=
                        ((ejey[(((((Parametros[g+1][1])*
                        2)-1)-1)])-ft);
                        PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                    }
                }
            }

            NumNodos=4;
            ContadorNodos[g]=NumNodos;

            sumaNodos=0;
            for(int yt=0; yt<(g);yt++){
                sumaNodos=sumaNodos+ContadorNodos[yt];
            }
            hu=g+sumaNodos;
            SolucionNueva[hu][0]=Parametros[g][6];   ///en x
            SolucionNueva[hu][1]=Parametros[g][7];   ///en y
            SolucionNueva[(hu+1)][0]=PuntoX;
            SolucionNueva[(hu+1)][1]=PuntoY;
            SolucionNueva[(hu+2)][0]=PuntoXX;
            SolucionNueva[(hu+2)][1]=PuntoYY;
            SolucionNueva[(hu+3)][0]=PuntoXXX;
            SolucionNueva[(hu+3)][1]=PuntoYYY;
            SolucionNueva[(hu+4)][0]=PuntoXXXX;
            SolucionNueva[(hu+4)][1]=PuntoYYYY;
            sumaNodos=0;
        }
        //caso 1
```

```
//caso 2
if(Parametros[g][0]!=Parametros[g+1][0] &&
Parametros[g][1]!=Parametros[g+1][1] &&
Parametros[g][2]!=Parametros[g+1][2] &&
g<numCities){

        caso=0;
        //calcular si el punto esta arriba o abajo
        if(Parametros[g][1]>Parametros[g+1][1]){
                //generar primer punto
                 if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=((ejex[((((Parametros[(g)][0])*
                        2)-1)-1)])-ft);
                        PuntoYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                        PuntoXXX=((ejex[((((Parametros[(g+1)][0])*
                        2)-1))])+ft);
                        PuntoYYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                        PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=((ejex[((((Parametros[(g)][0])*
                        2)-1))])+ft);
                        PuntoYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                        PuntoXXX=((ejex[((((Parametros[(g+1)][0])*
                        2)-1)-1)])-ft);
                        PuntoYYY=((ejey[((((Parametros[g+1][1])*
                        2)-1))])+ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }
        }else{
                if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
```

```
                PuntoXX=((ejex[(((((Parametros[(g)][0])*
                2)-1)-1)])-ft);
                PuntoYY=((ejey[(((((Parametros[g+1][1])*
                2)-1)-1)])-ft);
                PuntoXXX=((ejex[(((((Parametros[(g+1)][0])*
                2)-1))])+ft);
                PuntoYYY=((ejey[(((((Parametros[g+1][1])*
                2)-1)-1)])-ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
        }else{
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=((ejex[(((((Parametros[(g)][0])*
                2)-1))])+ft);
                PuntoYY=((ejey[(((((Parametros[g+1][1])*
                2)-1)-1)])-ft);
                PuntoXXX=((ejex[(((((Parametros[(g+1)][0])*
                2)-1)-1)])-ft);
                PuntoYYY=((ejey[(((((Parametros[g+1][1])*
                2)-1)-1)])-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
        }
}

NumNodos=4;
ContadorNodos[g]=NumNodos;
for(int yt=0; yt<g;yt++){
        sumaNodos=sumaNodos+ContadorNodos[yt];
}
hu=g+sumaNodos;
SolucionNueva[hu][0]=Parametros[g][6];   ///en x
SolucionNueva[hu][1]=Parametros[g][7];   ///en y

SolucionNueva[hu+1][0]=PuntoX;
SolucionNueva[hu+1][1]=PuntoY;
SolucionNueva[hu+2][0]=PuntoXX;
SolucionNueva[hu+2][1]=PuntoYY;
SolucionNueva[hu+3][0]=PuntoXXX;
SolucionNueva[hu+3][1]=PuntoYYY;
SolucionNueva[hu+4][0]=PuntoXXXX;
```

```
                SolucionNueva[hu+4][1]=PuntoYYYY;
                sumaNodos=0;

        }
        //caso 2

        //caso 3
        if(Parametros[g][0]==Parametros[g+1][0] &&
        Parametros[g][1]!=Parametros[g+1][1] &&
        Parametros[g][2]==Parametros[g+1][2] &&
        g<numCities  ){
                caso=0;

                //calcular si el punto esta arriba o abajo
                if(Parametros[g][1]>Parametros[g+1][1]){
                        if(Parametros[g][2]==0){
                                PuntoX=(Parametros[g][6]-ft);
                                PuntoY=(Parametros[g][7]);
                                PuntoXX=(ejex[(((Parametros[g][0]*
                                2)-1)-1)]-ft);
                                PuntoYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=(ejex[(((Parametros[g+1][0]*
                                2)-1)-1)]-ft);
                                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                                2)-1))]+ft);
                                PuntoXXXX=(Parametros[g+1][6]-ft);
                                PuntoYYYY=(Parametros[g+1][7]);
                        }else{
                                PuntoX=(Parametros[g][6]+ft);
                                PuntoY=(Parametros[g][7]);
                                PuntoXX=(ejex[((Parametros[g][0])*
                                2)-1]+ft);
                                PuntoYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=(ejex[(((Parametros[g+1][0]*
                                2)-1))]+ft);
                                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                                2)-1))]+ft);
                                PuntoXXXX=(Parametros[g+1][6]+ft);
                                PuntoYYYY=(Parametros[g+1][7]);
```

```
            }
}else{
            if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[(((Parametros[g][0]*
                        2)-1)-1)]-ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0]*
                        2)-1)-1)]-ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                        2)-1)-1)]-ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);
            }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[(((Parametros[g][0]*
                        2)-1))]+ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*
                        2)-1))]+ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0])*
                        2)-1)]+ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                        2)-1)-1)]-ft);
                        PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);
            }
}
NumNodos=4;

ContadorNodos[g]=NumNodos;
sumaNodos=0;
for(int yt=0; yt<g;yt++){
            sumaNodos=sumaNodos+ContadorNodos[yt];
}
hu=g+sumaNodos;
SolucionNueva[hu][0]=Parametros[g][6];   ///en x
SolucionNueva[hu][1]=Parametros[g][7];   ///en y

SolucionNueva[hu+1][0]=PuntoX;
SolucionNueva[hu+1][1]=PuntoY;
```

```
            SolucionNueva[hu+2][0]=PuntoXX;
            SolucionNueva[hu+2][1]=PuntoYY;
            SolucionNueva[hu+3][0]=PuntoXXX;
            SolucionNueva[hu+3][1]=PuntoYYY;
            SolucionNueva[hu+4][0]=PuntoXXXX;
            SolucionNueva[hu+4][1]=PuntoYYYY;
            sumaNodos=0;
}
//caso 3

//caso 4
if(Parametros[g][0]==Parametros[g+1][0] &&
Parametros[g][1]==Parametros[g+1][1] &&
Parametros[g][2]!=Parametros[g+1][2] &&
g<numCities){
            caso=0;
            //opcion 1   arriba
            if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[((((Parametros[g][0]*2)-1)-1)]-ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0])*2)-1]+ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);
                        PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);


                        d1=distancia(Parametros[g][6], Parametros[g][7],
                        PuntoX, PuntoY);          // 0  x
                        d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                        d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                        PuntoYYY);   // x  XX
                        d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                        PuntoYYYY);   //  x  XX
                        d5=distancia(PuntoXXXX, PuntoYYYY,
                        Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1

                        dt1=d1+d2+d3+d4+d5;

                        PuntoX=(Parametros[g][6]-ft);
```

```
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);   // II
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
                PuntoXXX=(ejex[((Parametros[g+1][0])*2)-1]+ft);     //
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);

                d1=distancia(Parametros[g][6], Parametros[g][7],
                PuntoX, PuntoY);    // 0  x
                d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                PuntoYYY);   //  x  XX
                d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                PuntoYYYY);   //  x  XX
                d5=distancia(PuntoXXXX, PuntoYYYY,
                Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1

                dt2=d1+d2+d3+d4+d5;
        }
        if(Parametros[g][2]==1){

                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);   //  III
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                PuntoXXX=(ejex[(((((Parametros[g+1][0])*2)-1)-1)]-ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);

                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);

                d1=distancia(Parametros[g][6], Parametros[g][7],
                PuntoX, PuntoY);    // 0  x
                d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                PuntoYYY);   //  x  XX
                d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                PuntoYYYY);   //  x  XX
                d5=distancia(PuntoXXXX, PuntoYYYY,
                Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1
```

```
                dt1=d1+d2+d3+d4+d5;

                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);   //  IV
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((((Parametros[g+1][0])*2)-1)-1)]-ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);

                d1=distancia(Parametros[g][6], Parametros[g][7],
                PuntoX,
                PuntoY); //0  x
                d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                PuntoYYY);  //x  XX
                d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                PuntoYYYY);//  x  XX
                d5=distancia(PuntoXXXX, PuntoYYYY,
                Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1

                dt2=d1+d2+d3+d4+d5;
        }

        if(dt1<dt2){
                if(Parametros[g][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);

                        PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                        PuntoXXX=(ejex[((Parametros[g+1][0])*
                        2)-1]+ft);     //
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                        2)-1))]+ft);
                        PuntoXXXX=(Parametros[g+1][6]+ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
```

```
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                PuntoXXX=(ejex[(((((Parametros[g+1][0])*
                2)-1)-1)]-ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1))]+ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
        }
}else{
        if(Parametros[g][2]==0){
                PuntoX=(Parametros[g][6]-ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[((Parametros[g+1][0])*
                2)-1]+ft);     //
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
        }else{
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((((Parametros[g+1][0])*
                2)-1)-1)]-ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
        }
}

NumNodos=4;

ContadorNodos[g]=NumNodos;
sumaNodos=0;
for(int yt=0; yt<g;yt++){
```

```
                    sumaNodos=sumaNodos+ContadorNodos[yt];
                }
                hu=g+sumaNodos;
                SolucionNueva[hu][0]=Parametros[g][6];   ///en x
                SolucionNueva[hu][1]=Parametros[g][7];   ///en y

                SolucionNueva[hu+1][0]=PuntoX;
                SolucionNueva[hu+1][1]=PuntoY;
                SolucionNueva[hu+2][0]=PuntoXX;
                SolucionNueva[hu+2][1]=PuntoYY;
                SolucionNueva[hu+3][0]=PuntoXXX;
                SolucionNueva[hu+3][1]=PuntoYYY;
                SolucionNueva[hu+4][0]=PuntoXXXX;
                SolucionNueva[hu+4][1]=PuntoYYYY;

                sumaNodos=0;
        }
        //caso 4

        //caso 5
        if(Parametros[g][0]!=Parametros[g+1][0] &&
        Parametros[g][1]==Parametros[g+1][1] &&
        g<numCities){
                caso=0;
                //opcion  lado 0-  0
                if(Parametros[g][2]==0 && Parametros[g+1][2]==0){
                        PuntoX=(Parametros[g][6]-ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);   //  I
                        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1)-1)]-ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);

                        d1=distancia(Parametros[g][6], Parametros[g][7],
                        PuntoX,PuntoY);// 0 x
                        d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                        d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                        PuntoYYY);
                        d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
```

```
PuntoYYYY);
d5=distancia(PuntoXXXX, PuntoYYYY,
Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1

dt1=d1+d2+d3+d4+d5;

PuntoX=(Parametros[g][6]-ft);
PuntoY=(Parametros[g][7]);
PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);   //  II
PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1)-1)]-ft);     //
PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1)-1)]-ft);
PuntoXXXX=(Parametros[g+1][6]-ft);
PuntoYYYY=(Parametros[g+1][7]);

d1=distancia(Parametros[g][6], Parametros[g][7],
PuntoX,PuntoY);          // 0  x
d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
PuntoYYY);
d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
PuntoYYYY);
d5=distancia(PuntoXXXX, PuntoYYYY,
Parametros[g+1][6], Parametros[g+1][7]);   //  xx  1

dt2=d1+d2+d3+d4+d5;

if(dt1<dt2){
        PuntoX=(Parametros[g][6]-ft);
        PuntoY=(Parametros[g][7]);
        PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);
        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
        PuntoXXX=(ejex[(((Parametros[g+1][0]*
        2)-1)-1)]-ft);
        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
        2)-1))]+ft);

        PuntoXXXX=(Parametros[g+1][6]-ft);
        PuntoYYYY=(Parametros[g+1][7]);
}else{
        PuntoX=(Parametros[g][6]-ft);
        PuntoY=(Parametros[g][7]);
```

```
                PuntoXX=(ejex[((((Parametros[g][0]*2)-1)-1)]-ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[((((Parametros[g+1][0]*
                2)-1)-1)]-ft);     //
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }

        }

        //opcion  lado 0-  1
        if(Parametros[g][2]==0 && Parametros[g+1][2]==1){
            PuntoX=(Parametros[g][6]-ft);
            PuntoY=(Parametros[g][7]);
            PuntoXX=(ejex[((((Parametros[g][0]*2)-1)-1)]-ft);   //  I
            PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
            PuntoXXX=(ejex[((((Parametros[g+1][0]*2)-1))]+ft);
            PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);
            PuntoXXXX=(Parametros[g+1][6]+ft);
            PuntoYYYY=(Parametros[g+1][7]);

            d1=distancia(Parametros[g][6], Parametros[g][7],
            PuntoX, PuntoY);          // 0  x
            d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
            d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
            PuntoYYY);
            d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
            PuntoYYYY);
            d5=distancia(PuntoXXXX, PuntoYYYY,
            Parametros[g+1][6],
            Parametros[g+1][7]);   //  xx  1

            dt1=d1+d2+d3+d4+d5;

            PuntoX=(Parametros[g][6]-ft);
            PuntoY=(Parametros[g][7]);
            PuntoXX=(ejex[((((Parametros[g][0]*2)-1)-1)]-ft);   //  II
            PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
```

```
PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1))]+ft);     //
PuntoYYY=(ejey[(((Parametros[g+1][1])*2)-1)-1]-ft);
PuntoXXXX=(Parametros[g+1][6]+ft);
PuntoYYYY=(Parametros[g+1][7]);

d1=distancia(Parametros[g][6], Parametros[g][7],
PuntoX,PuntoY);          // 0  x
d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
PuntoYYY);
d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
PuntoYYYY);
d5=distancia(PuntoXXXX, PuntoYYYY,
Parametros[g+1][6],
Parametros[g+1][7]);   //  xx  1

dt2=d1+d2+d3+d4+d5;

if(dt1<dt2){
        PuntoX=(Parametros[g][6]-ft);
        PuntoY=(Parametros[g][7]);
        PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);
        PuntoYY=(ejey[(((((Parametros[g][1])*
        2)-1))]+ft);
        PuntoXXX=(ejex[(((Parametros[g+1][0]*
        2)-1))]+ft);
        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
        2)-1))]+ft);
        PuntoXXXX=(Parametros[g+1][6]+ft);
        PuntoYYYY=(Parametros[g+1][7]);

}else{

        PuntoY=(Parametros[g][7]);
        PuntoXX=(ejex[(((Parametros[g][0]*2)-1)-1)]-ft);
        PuntoYY=(ejey[(((((Parametros[g][1])*
        2)-1)-1)]-ft);
        PuntoXXX=(ejex[(((Parametros[g+1][0]*
        2)-1))]+ft);     //
        PuntoYYY=(ejey[(((Parametros[g+1][1])*
        2)-1)-1]-ft);
        PuntoXXXX=(Parametros[g+1][6]+ft);
```

```
                    PuntoYYYY=(Parametros[g+1][7]);
                }
            }

            //opcion  lado 1-  0
            if(Parametros[g][2]==1 && Parametros[g+1][2]==0){
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);   //  I
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1)-1)]-ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);

                d1=distancia(Parametros[g][6], Parametros[g][7],
                PuntoX, PuntoY); // 0  x
                d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
                PuntoYYY);
                d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                PuntoYYYY);
                d5=distancia(PuntoXXXX, PuntoYYYY,
                Parametros[g+1][6],
                Parametros[g+1][7]);   //  xx  1

                dt1=d1+d2+d3+d4+d5;

                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);   //  II
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1)-1)]-ft);     //
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);

                d1=distancia(Parametros[g][6], Parametros[g][7],
                PuntoX, PuntoY);          // 0  x
                d2=distancia(PuntoX, PuntoY, PuntoXX, PuntoYY);
                d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
```

```
                PuntoYYY);
                d4=distancia(PuntoXXX, PuntoYYY, PuntoXXXX,
                PuntoYYYY);
                d5=distancia(PuntoXXXX, PuntoYYYY,
                Parametros[g+1][6],Parametros[g+1][7]);   //  xx  1

                dt2=d1+d2+d3+d4+d5;

                if(dt1<dt2){
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[(((Parametros[g][0]*
                        2)-1))]+ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0]*
                        2)-1)-1)]-ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                        2)-1))]+ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);

                }else{
                        PuntoX=(Parametros[g][6]+ft);
                        PuntoY=(Parametros[g][7]);
                        PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
                        PuntoYY=(ejey[(((((Parametros[g][1])*
                        2)-1)-1)]-ft);
                        PuntoXXX=(ejex[(((Parametros[g+1][0]*
                        2)-1)-1)]-ft);
                        PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                        2)-1)-1)]-ft);
                        PuntoXXXX=(Parametros[g+1][6]-ft);
                        PuntoYYYY=(Parametros[g+1][7]);
                }
        }


        //opcion  lado 1-  1
        if(Parametros[g][2]==1 && Parametros[g+1][2]==1){
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);   //  I
```

```
PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1))]+ft);
PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1))]+ft);
PuntoXXXX=(Parametros[g+1][6]+ft);
PuntoYYYY=(Parametros[g+1][7]);

d1=distancia(Parametros[g][6],
Parametros[g][7], PuntoX,PuntoY);
d2=distancia(PuntoX, PuntoY, PuntoXX,
PuntoYY);
d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
PuntoYYY);
d4=distancia(PuntoXXX, PuntoYYY,
PuntoXXXX, PuntoYYYY);
d5=distancia(PuntoXXXX, PuntoYYYY,
Parametros[g+1][6],Parametros[g+1][7]);

dt1=d1+d2+d3+d4+d5;

PuntoX=(Parametros[g][6]+ft);
PuntoY=(Parametros[g][7]);
PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
PuntoYY=(ejey[(((((Parametros[g][1])*2)-1)-1)]-ft);
PuntoXXX=(ejex[(((Parametros[g+1][0]*2)-1))]+ft);
PuntoYYY=(ejey[(((((Parametros[g+1][1])*2)-1)-1)]-ft);
PuntoXXXX=(Parametros[g+1][6]+ft);
PuntoYYYY=(Parametros[g+1][7]);
d1=distancia(Parametros[g][6],
Parametros[g][7], PuntoX,PuntoY);// 0  x
d2=distancia(PuntoX, PuntoY, PuntoXX,PuntoYY);
d3=distancia(PuntoXX, PuntoYY, PuntoXXX,
PuntoYYY);
d4=distancia(PuntoXXX, PuntoYYY,
PuntoXXXX, PuntoYYYY);

d5=distancia(PuntoXXXX, PuntoYYYY,
Parametros[g+1][6],Parametros[g+1][7]);
dt2=d1+d2+d3+d4+d5;
```

```
            if(dt1<dt2){
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*2)-1))]+ft);
                PuntoXXX=(ejex[(((Parametros[g+1][0]*
                2)-1))]+ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1))]+ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }else{

                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*2)-1))]+ft);
                PuntoYY=(ejey[(((((Parametros[g][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((Parametros[g+1][0]*
                2)-1))]+ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }
        }
        NumNodos=4;
        ContadorNodos[g]=NumNodos;
        sumaNodos=0;
        for(int yt=0; yt<g;yt++){
            sumaNodos=sumaNodos+ContadorNodos[yt];
        }
        hu=g+sumaNodos;

        SolucionNueva[hu][0]=Parametros[g][6];   ///en x
        SolucionNueva[hu][1]=Parametros[g][7];   ///en y

        SolucionNueva[hu+1][0]=PuntoX;
        SolucionNueva[hu+1][1]=PuntoY;
        SolucionNueva[hu+2][0]=PuntoXX;
        SolucionNueva[hu+2][1]=PuntoYY;
        SolucionNueva[hu+3][0]=PuntoXXX;
```

```
                SolucionNueva[hu+3][1]=PuntoYYY;
                SolucionNueva[hu+4][0]=PuntoXXXX;
                SolucionNueva[hu+4][1]=PuntoYYYY;

                sumaNodos=0;
        }
        //caso 5

        // caso 6
        if(Parametros[g][0]==Parametros[g+1][0] &&
        Parametros[g][1]!=Parametros[g+1][1] &&
        Parametros[g][2]!=Parametros[g+1][2] && g<numCities){
                caso=0;
                //calcular si el punto esta arriba o abajo
                if(Parametros[g][1]>Parametros[g+1][1]){
                        if(Parametros[g][2]==0){
                                PuntoX=(Parametros[g][6]-ft);
                                PuntoY=(Parametros[g][7]);
                                PuntoXX=(ejex[(((Parametros[g][0]*
                                2)-1)-1)]-ft);   // I
                                PuntoYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=(Parametros[g+1][6]+ft);   // I
                                PuntoYYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXXX=(Parametros[g+1][6]+ft);
                                PuntoYYYY=(Parametros[g+1][7]);
                        }else{
                                PuntoX=(Parametros[g][6]+ft);
                                PuntoY=(Parametros[g][7]);
                                PuntoXX=(ejex[((Parametros[g][0])*
                                2)-1]+ft);     // II
                                PuntoYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=(Parametros[g+1][6]-ft);     // II
                                PuntoYYY=(ejey[(((((Parametros[g][1])*
                                2)-1)-1)]-ft);
                                PuntoXXXX=(Parametros[g+1][6]-ft);
                                PuntoYYYY=(Parametros[g+1][7]);
                        }
                }else{
```

```
            if(Parametros[g][2]==1){
                PuntoX=(Parametros[g][6]+ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*
                2)-1))]+ft);
                PuntoYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((Parametros[g+1][0]*
                2)-1)-1)]-ft);   //
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]-ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }else{
                PuntoX=(Parametros[g][6]-ft);
                PuntoY=(Parametros[g][7]);
                PuntoXX=(ejex[(((Parametros[g][0]*
                2)-1)-1)]-ft);
                PuntoYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXX=(ejex[(((((Parametros[g+1][0])*
                2)-1))]+ft);
                PuntoYYY=(ejey[(((((Parametros[g+1][1])*
                2)-1)-1)]-ft);
                PuntoXXXX=(Parametros[g+1][6]+ft);
                PuntoYYYY=(Parametros[g+1][7]);
            }
        }

        NumNodos=4;
        ContadorNodos[g]=NumNodos;

        sumaNodos=0;
        for(int yt=0; yt<g;yt++){
            sumaNodos=sumaNodos+ContadorNodos[yt];
        }
        hu=g+sumaNodos;
        SolucionNueva[hu][0]=Parametros[g][6];   ///en x
        SolucionNueva[hu][1]=Parametros[g][7];   ///en y

        SolucionNueva[hu+1][0]=PuntoX;
        SolucionNueva[hu+1][1]=PuntoY;
```

```
                SolucionNueva[hu+2][0]=PuntoXX;
                SolucionNueva[hu+2][1]=PuntoYY;

                SolucionNueva[hu+3][0]=PuntoXXX;
                SolucionNueva[hu+3][1]=PuntoYYY;
                SolucionNueva[hu+4][0]=PuntoXXXX;
                SolucionNueva[hu+4][1]=PuntoYYYY;

                sumaNodos=0;
        }
        // caso 6

        // caso fin
        if(g==(cities.length-1)){ //generar punto entre ultimo punto y llegada.
                caso=0;
                int u=cities.length-1;
                        if(Parametros[u][2]==0){    // lado izquierdo
                                PuntoX=((ejex[(((Parametros[u][0]*2)-1)-1)])-ft);
                                PuntoY=Parametros[u][7];
                                PuntoXX=((ejex[(((Parametros[u][0]*
                                2)-1)-1)])-ft);
                                PuntoYY=(ejey[(((((Parametros[0][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=((ejex[(((Parametros[u][0]*
                                2)-1)-1)])-ft);
                                PuntoYYY=Parametros[0][7];
                        }else{                  //lado derecho
                                PuntoX=((ejex[((Parametros[u][0]*2)-1)])+ft);
                                PuntoY=Parametros[u][7];
                                PuntoXX=((ejex[((Parametros[u][0]*2)-1)])+ft);
                                PuntoYY=(ejey[(((((Parametros[0][1])*
                                2)-1)-1)]-ft);
                                PuntoXXX=((ejex[((Parametros[u][0]*2)-1)])+ft);
                                PuntoYYY=Parametros[0][7];
                }
                NumNodos=3;
                ContadorNodos[g]=NumNodos;
                for(int yt=0; yt<g;yt++){
                        sumaNodos=sumaNodos+ContadorNodos[yt];
                }
```

```
                    hu=g+sumaNodos;
                    SolucionNueva[hu][0]=Parametros[g][6];   ///en x
                    SolucionNueva[hu][1]=Parametros[g][7];   ///en y

                    SolucionNueva[hu+1][0]=PuntoX;
                    SolucionNueva[hu+1][1]=PuntoY;
                    SolucionNueva[hu+2][0]=PuntoXX;
                    SolucionNueva[hu+2][1]=PuntoYY;
                    SolucionNueva[hu+3][0]=PuntoXXX;
                    SolucionNueva[hu+3][1]=PuntoYYY;
                    sumaNodos=0;
                }
                // caso fin

                //caso agregar solo numeros
                if(caso==1){
                    NumNodos=0;
                    ContadorNodos[(g)]=NumNodos;
                    sumaNodos=0;
                    for(int yt=0; yt<(g);yt++){
                        sumaNodos=sumaNodos+ContadorNodos[yt];
                    }
                    hu=g+sumaNodos;
                    SolucionNueva[hu][0]=Parametros[g][6];   ///en x
                    SolucionNueva[hu][1]=Parametros[g][7];   ///en y
                    sumaNodos=0;
                }

                //caso agregar solo numeros
            }///final de for
            ///////  metodo de mejora de ruta

            TravelingSalesman.solnueva=1;
            int suma=0;
            for(int re=0; re<ContadorNodos.length; re++){
                suma=suma+ContadorNodos[re];
            }
            TravelingSalesman.total=(suma+numCities+1);
            ContadorNodos=null;
        }

        public void startThread(){
```

```
int filas=2;
int estantesporfilas=2;
int altoestante=100;
int anchoestante=50;
int espacioverticalestante=50;
int espaciohorizontalestante=50;
int espacioinicioy=50;
int espacioiniciox=100;
int numCities = 50;

try{
        estantesporfilas = Integer.parseInt(ctrlestantes.getText());
} catch (NumberFormatException e){}

try{
        filas = Integer.parseInt(ctrlfilas.getText());
} catch (NumberFormatException e) {}

try{
        altoestante = Integer.parseInt(ctrlaltoestante.getText());
} catch (NumberFormatException e) {}

try{
        anchoestante = Integer.parseInt(ctrlanchoestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioverticalestante =
        Integer.parseInt(ctrlespacioverticalestante.getText());
} catch (NumberFormatException e) {}

try{
        espaciohorizontalestante =
        Integer.parseInt(ctrlespaciohorizontalestante.getText());
} catch (NumberFormatException e) {}

try{
        espacioinicioy = Integer.parseInt(ctrlespacioinicioy.getText());
} catch (NumberFormatException e) {}
```

```
                try{
                        espacioiniciox = Integer.parseInt(ctrlespacioiniciox.getText());
                } catch (NumberFormatException e) {}

                try{
                        numCities = Integer.parseInt(ctrlCities.getText());
                } catch (NumberFormatException e) {}

                int aleatorios[][] = new int[numCities+1][2];
                aleatorios=generarnumaleatorios(altoestante,anchoestante,
                espacioverticalestante,espaciohorizontalestante,espacioinicioy,
                espacioiniciox, filas, estantesporfilas);

                aleatorios[numCities][0]=vx;
                aleatorios[numCities][1]=vy;
                Prd=aleatorios;

                cities = new City[numCities+1];
                for (int i = 0; i < aleatorios.length; i++){
                        cities[i] = new City((int) aleatorios[i][0],(int) aleatorios[i][1]);
                        if((Math.random() * (getBounds().height - 80))>500)
                        System.out.println(Math.random() * (getBounds().height - 80)
                        +"    y  mayor que 500   " +i );
                }
                started = true;
                // start up the background thread
                if (worker != null)
                        worker = null;

                        worker = new SimulateAnnealing(this);
                        worker.setPriority(Thread.MIN_PRIORITY);
                        worker.start();
                        update();
                        started = true;
        }
}
```

Anexo A-3: City.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package recocidosimulado;

/**
* Simulated Annealing and the Traveling Salesman
* Copyright 2005 by Heaton Research, Inc.
* by Jeff Heaton (http://www.heatonresearch.com) 12-2005
* ------------------------------------------------
* This source code is copyrighted.
* You may reuse this code in your own compiled projects.
* However, if you would like to redistribute this source code
* in any form, you must obtain permission from Heaton Research.
* (support@heatonresearch.com).
*
* This class implements a city that must be visited.
* ------------------------------------------------
* Want to learn more about Neural Network Programming in Java?
* Have a look at our e-book:
* http://www.heatonresearch.com/articles/series/1/
* @author Jeff Heaton (http://www.jeffheaton.com)
* @version 1.0
*/
class City {

/**
* The city's x position.
*/
private int xpos;
/**
* The city's y position.
*/
private int ypos;
```

```
/**
* Constructor.
* @param x The city's x position
* @param y The city's y position.
*/
public City(int x, int y) {
        xpos = x;
        ypos = y;
}

/**
* Return's the city's x position.
* @return The city's x position.
*/
public int getx() {
        return xpos;
}

/**
* Returns the city's y position.
* @return The city's y position.
*/

public int gety() {
        return ypos;
}

/**
* Returns how close the city is to another city.
* @param cother The other city.
* @return A distance.
*/
public int proximity(City cother) {
        return proximity(cother.getx(),cother.gety());
}

/**
* Returns how far this city is from a a specific point.
* This method uses the pythagorean theorum to calculate
* the distance.
* @param x The x coordinate
```

```
        * @param y The y coordinate
        * @return The distance.
        */
        public int proximity(int x, int y) {
                int xdiff = xpos - x;
                int ydiff = ypos - y;
                return(int)Math.sqrt( xdiff*xdiff + ydiff*ydiff );
        }
}
```

Anexo A-4: SimulatedAnnealingClient.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package recocidosimulado;

/**
* Simulated Annealing and the Traveling Salesman
* Copyright 2005 by Heaton Research, Inc.
* by Jeff Heaton (http://www.heatonresearch.com) 12-2005
* ------------------------------------------------
* This source code is copyrighted.
* You may reuse this code in your own compiled projects.
* However, if you would like to redistribute this source code
* in any form, you must obtain permission from Heaton Research.
* (support@heatonresearch.com).* This class implements a city that must be visited.
* http://www.heatonresearch.com/articles/series/1/
*
* @author Jeff Heaton (http://www.jeffheaton.com)
* @version 1.0
*/
public interface SimulatedAnnealingClient {
        public int getCount();
        public double getError(int i,int j);
        public double getStartingTemperature();
        public double getDelta();
        public void update();
        public void setStatus(String status);
}
```

Anexo B: Módulo de Realidad Aumentada

Anexo B-1: Multiple markers.as

```
/*
 * Code inspired by the examples provided by Transmote
[http://words.transmote.com/wp/flarmanager/inside-flarmanager/2d-marker-tracking/]

 */
package {
	/* Tweener Class [http://code.google.com/p/tweener/] */
	//import Evento.*;
	import caurina.transitions.Tweener;
	import com.transmote.flar.FLARManager;
	import com.transmote.flar.marker.FLARMarker;
	import com.transmote.flar.marker.FLARMarkerEvent;
	import com.transmote.flar.utils.geom.FLARPVGeomUtils;
	import flash.display.Loader;
	import flash.display.Shape;
	import flash.display.Sprite;
	import flash.events.*;
	import flash.events.Event;
	import flash.events.MouseEvent;
	import flash.filters.GlowFilter;
	import flash.geom.Point;
	import flash.media.Sound;
	import flash.media.SoundChannel;
	import flash.media.SoundLoaderContext;
	import flash.media.SoundTransform;
	import flash.net.URLRequest;
	import flash.text.TextField;
	import flash.utils.Dictionary;
	import org.libspark.flartoolkit.support.pv3d.FLARCamera3D;
	import org.papervision3d.lights.PointLight3D;
	import org.papervision3d.materials.shadematerials.FlatShadeMaterial;
	import org.papervision3d.materials.utils.MaterialsList;
	import org.papervision3d.objects.DisplayObject3D;
	import org.papervision3d.objects.primitives.Cube;
	import org.papervision3d.render.LazyRenderEngine;
	import org.papervision3d.scenes.Scene3D;
	import org.papervision3d.view.Viewport3D;
```

```
/* Change output settings */
[SWF(width="640", height="480", frameRate="25", backgroundColor="#000000")]
public class Multiple_Markers extends Sprite {
        /* FLARManager pointer */
        private var fm:FLARManager;
        /* Papervision Scene3D pointer */
        private var scene3D:Scene3D;
        /* Papervision Viewport3D pointer */
        private var viewport3D:Viewport3D;
        /* FLARToolkit FLARCamera3D pointer */
        private var camera3D:FLARCamera3D;
        /* Papervision render engine pointer */
        private var lre:LazyRenderEngine;
        /* Papervision PointLight3D pointer */
        private var pointLight3D:PointLight3D;
        /* Initialise glow filter to add a white border around selected objects in our scene */
        private var glow:GlowFilter = new GlowFilter(0xFFFFFF, 1, 7, 7, 30, 1, false, false);
        /* Vector storing references to all markers on screen, grouped by pattern id */
        private var markersByPatternId:Vector.<Vector.<FLARMarker>>;
        /* Dictionary storing references to marker containers,
        indexed by relevant marker object */
        private var containersByMarker:Dictionary;
        /* Dictionary storing references to the corner nodes for each marker,
        indexed by relevant marker object */
        private var nodesByMarker:Dictionary;

        /* Constructor method */
        public static const CLICK:String = "onClick";
        public static const CLICKK:String = "onClick2";
        public static const CLICKKK:String = "onClick3";
        private var dibu:Shape=null;  private var borrado:int=0;
        private var contador0:int=0;
        private var contador1:int=0;  private var contador2:int=0;  private var
         Alp:Number=0.4;  private var pivote:Number=0;  private
        var punto:Shape=null; private var Valpha:Number=0.9;
        private var contador3:int=0;  private var contador4:int=0;
        private var contador5:int=0;

        private var Ab:int=0;  private var Ar:int=0;  private var Iz:int=0;  private var De:int=0;
        private var iIz:int=0; private var sIz:int=0; private var iDe:int=0;
```

```
private var sDe:int=0; private var O:int=0;

private var D:int=2;                private var C:int=2;          private var I:int=2;
private var ID:int;                 private var Prd:int=10;       private var Er:int=0;
private var fc=new Shape();         private var fi=new Shape();
private var fd=new Shape();         private var circulo:Shape = new Shape();
private var abajo:Shape = new Shape();
private var arriba:Shape = new Shape();
private var derecha:Shape = new Shape();
private var izquierda:Shape = new Shape();
private var infizquierdo:Shape = new Shape();
private var supizquierdo:Shape = new Shape();
private var infderecho:Shape = new Shape();
private var supderecho:Shape = new Shape();
private var nFlechasEstante:int=2; private var nFlechasGuia:int=2;
private var imagenID:Loader = new Loader();
private var imagenProducto:Loader = new Loader();
private var IDcontador:int=0; private var Prdcontador:int=0;

/*
private var Marcador:Array =[[0,0,10,3,5,"ID_1.png"],[1,0,10,3,5,"ID_2.png"],
[2,1,10,3,5,"producto1.png"], [3,1,10,3,5,"pieza3.png"], [4,3,10,3,5],
[5,2,10,3,5]];
private var Trayectoria:Array=      [[0,""],[1,""],[2,""],[3,""],[4,""],[5,""]];
*/

/*
private var Marcador:Array =[[0,0,10,3,5,"ID_1.png"],[1,1,10,3,5,"producto1.png"],
[2,2,10,3,5,],[3,2,10,3,5], [4,2,10,3,5],[5,3,10,3,5]];
private var Trayectoria:Array=[[2,"C"], [3,"D"]];
*/

/*
private var Marcador:Array =[[0,0,0,0,0,"ID_1.png"],[1,1,10,13,1.5,"producto1.png"],
[2,2,0,0,0,],[3,3,10,5, 1.8],[4,3,10,10,1.8],[5,3,10,15,1.8]];
private var Trayectoria:Array=[[2,"C"],[2,"C"]];
*/

private var Marcador:Array =[[0,0,0,0,0,"ID_1.png","Usuario 1"],
[1,1,10,13,1,"producto1.png",0,5],[2,1,10,4 ,1.8,"pieza3.png",0,10],[3,2,10,5, 1.8],
[4,3,10,10,1.8],[5,4,0,0,0]];
```

```
private var Trayectoria:Array;
private var Productos:Array=[[1,"a"],[2,"a"]];

public function Selector(Pd:int):void{
	if(Pd==0){
		Trayectoria=[[3,"I"],[3,"I"]];
	}
	if(Pd==1){
		Trayectoria=[[3,"D"],[3,"D"]];
	}
	if(Pd==Productos.length){
		Trayectoria=[[3,"C"],[3,"C"]];
	}
}
private var t:int=0;
private var rec:int=0;
private var Producto:int=0;
private var visto:int=0;
private var validador:int=0;
public function Multiple_Markers(){
	/* Run augmented reality initialisation */
	this.initFLAR();
	addEventListener(Multiple_Markers.CLICK,FlechasGuia);
	addEventListener(Multiple_Markers.CLICKKK,FlechaEstante);
}

private function FlechaEstante(e:Event):void{

	 var Colores:Array=[["0x27F71C"],["0x27F71C"], ["0xBAB7B7"]];
	//  0 verde 1 plomo F40707 rojo

	circulo.graphics.lineStyle(1, 0xBAB7B7);
	if(Ab==0){ abajo.graphics.lineStyle(1, 0x27F71C); }
	if(Ab==1){ abajo.graphics.lineStyle(1, 0xBAB7B7); }

	if(Ar==0){ arriba.graphics.lineStyle(1, 0x27F71C); }
	if(Ar==1){ arriba.graphics.lineStyle(1, 0xBAB7B7); }

	if(Iz==0){ izquierda.graphics.lineStyle(1, 0x27F71C); }
	if(Iz==1){ izquierda.graphics.lineStyle(1, 0xBAB7B7); }
```

```
if(De==0){ derecha.graphics.lineStyle(1, 0x27F71C); }
if(De==1){ derecha.graphics.lineStyle(1, 0xBAB7B7); }

if(iIz==0){ infizquierdo.graphics.lineStyle(1, 0x27F71C); }
if(iIz==1){ infizquierdo.graphics.lineStyle(1, 0xBAB7B7); }

if(sIz==0){ supizquierdo.graphics.lineStyle(1, 0x27F71C); }
if(sIz==1){ supizquierdo.graphics.lineStyle(1, 0xBAB7B7); }

if(iDe==0){ infderecho.graphics.lineStyle(1, 0x27F71C); }
if(iDe==1){ infderecho.graphics.lineStyle(1, 0xBAB7B7); }

if(iDe==0){ infderecho.graphics.lineStyle(1, 0x27F71C); }
if(iDe==1){ infderecho.graphics.lineStyle(1, 0xBAB7B7); }

if(sDe==0){ supderecho.graphics.lineStyle(1, 0x27F71C); }
if(sDe==1){ supderecho.graphics.lineStyle(1, 0xBAB7B7); }

var NAb:int=Ab; var NAr:int=Ar; var NIz:int=Iz;
var NDe:int=De; var NiIz:int=iIz; var NsIz:int=sIz;
var NiDe:int=iDe; var NsDe:int=sDe;

if(Ab==2 && Ar==2 && Iz==2 && De==2 && iIz==2 && sIz==2 &&
iDe==2 && sDe==2){
        abajo.graphics.lineStyle(1, 0xF90606);
        arriba.graphics.lineStyle(1, 0xF90606);
        izquierda.graphics.lineStyle(1, 0xF90606);
        derecha.graphics.lineStyle(1, 0xF90606);
        infizquierdo.graphics.lineStyle(1, 0xF90606);
        supizquierdo.graphics.lineStyle(1, 0xF90606);
        infderecho.graphics.lineStyle(1, 0xF90606);
        supderecho.graphics.lineStyle(1, 0xF90606);
        circulo.graphics.lineStyle(1, 0xBAB7B7);
}
var x:int=320;
var y:int=240
circulo.graphics.drawCircle(x, y, 10);
//----------------------mod invertida----------------------------
arriba.graphics.drawRect(x-10,y+20,1,1);    //abajo
arriba.graphics.lineTo(x+10,y+20);
arriba.graphics.lineTo(x+10,y+100);
arriba.graphics.lineTo(x+25,y+100);
```

```
arriba.graphics.lineTo(x,y+130);
arriba.graphics.lineTo(x-25,y+100);
arriba.graphics.lineTo(x-10,y+100);
arriba.graphics.lineTo(x-10,y+20);
arriba.graphics.drawRect(x-10,y+20,1,1);

abajo.graphics.drawRect(x-10,y-20,1,1);    //arriba
abajo.graphics.lineTo(x+10,y-20);
abajo.graphics.lineTo(x+10,y-100);
abajo.graphics.lineTo(x+25,y-100);
abajo.graphics.lineTo(x,y-130);
abajo.graphics.lineTo(x-25,y-100);
abajo.graphics.lineTo(x-10,y-100);
abajo.graphics.lineTo(x-10,y-20);
abajo.graphics.drawRect(x-10,y-20,1,1);

derecha.graphics.drawRect(x+20,y+10,1,1);   //derecha
derecha.graphics.lineTo(x+100,y+10);//
derecha.graphics.lineTo(x+100,y+25);//
derecha.graphics.lineTo(x+130,y);//
derecha.graphics.lineTo(x+100,y-25);//
derecha.graphics.lineTo(x+100,y-10);//
derecha.graphics.lineTo(x+20,y-10);
derecha.graphics.lineTo(x+20,y+10);
derecha.graphics.drawRect(x+20,y+10,1,1);

izquierda.graphics.drawRect(x-20,y+10,1,1);//  izquierda
izquierda.graphics.lineTo(x-100,y+10);//
izquierda.graphics.lineTo(x-100,y+25);//
izquierda.graphics.lineTo(x-130,y);//
izquierda.graphics.lineTo(x-100,y-25);//
izquierda.graphics.lineTo(x-100,y-10);//
izquierda.graphics.lineTo(x-20,y-10);
izquierda.graphics.lineTo(x-20,y+10);
izquierda.graphics.drawRect(x-20,y+10,1,1);

supderecho.graphics.drawRect(x+20,y+10,1,1);//  //Inf derecho
supderecho.graphics.lineTo(x+10,y+20);//
supderecho.graphics.lineTo(x+10+56.57,y+20+56.57);//
supderecho.graphics.lineTo(x+10+56.57-10.61,y+20+56.57+10.61);//
supderecho.graphics.lineTo(x+14.14+91.92,y+14.14+91.92);//
```

```
            supderecho.graphics.lineTo(x+20+56.57+10.61,y+10+56.57-10.61);//
            supderecho.graphics.lineTo(x+20+56.57,y+10+56.57);
            supderecho.graphics.lineTo(x+20,y+10);
            supderecho.graphics.drawRect(x+20,y+10,1,1);

            infderecho.graphics.drawRect(x+20,y-10,1,1);//  //sup derecho
            infderecho.graphics.lineTo(x+10,y-20);//
            infderecho.graphics.lineTo(x+10+56.57,y-20-56.57);//
            infderecho.graphics.lineTo(x+10+56.57-10.61,y-20-56.57-10.61);//
            infderecho.graphics.lineTo(x+14.14+91.92,y-14.14-91.92);//
            infderecho.graphics.lineTo(x+20+56.57+10.61,y-10-56.57+10.61);//
            infderecho.graphics.lineTo(x+20+56.57,y-10-56.57);
            infderecho.graphics.lineTo(x+20,y-10);
            infderecho.graphics.drawRect(x+20,y-10,1,1);

            supizquierdo.graphics.drawRect(x-20,y+10,1,1);//  //inf izquierdo
            supizquierdo.graphics.lineTo(x-10,y+20);//
            supizquierdo.graphics.lineTo(x-10-56.57,y+20+56.57);//
            supizquierdo.graphics.lineTo(x-10-56.57+10.61,y+20+56.57+10.61);//
            supizquierdo.graphics.lineTo(x-14.14-91.92,y+14.14+91.92);//
            supizquierdo.graphics.lineTo(x-20-56.57-10.61,y+10+56.57-10.61);//
            supizquierdo.graphics.lineTo(x-20-56.57,y+10+56.57);
            supizquierdo.graphics.lineTo(x-20,y+10);
            supizquierdo.graphics.drawRect(x-20,y+10,1,1);

            infizquierdo.graphics.drawRect(x-20,y-10,1,1);//  //sup izquierdo
            infizquierdo.graphics.lineTo(x-10,y-20);//
            infizquierdo.graphics.lineTo(x-10-56.57,y-20-56.57);//
            infizquierdo.graphics.lineTo(x-10-56.57+10.61,y-20-56.57-10.61);//
            infizquierdo.graphics.lineTo(x-14.14-91.92,y-14.14-91.92);//
            infizquierdo.graphics.lineTo(x-20-56.57-10.61,y-10-56.57+10.61);//
            infizquierdo.graphics.lineTo(x-20-56.57,y-10-56.57);
            infizquierdo.graphics.lineTo(x-20,y-10);
            infizquierdo.graphics.drawRect(x-20,y-10,1,1);
            //----------------------mod invertida-----------------------------

            addChild(circulo); addChild(abajo); addChild(arriba);
            addChild(derecha); addChild(izquierda); addChild(infizquierdo);
            addChild(supizquierdo); addChild(infderecho); addChild(supderecho);
            nFlechasEstante=1;
    }
    public function FlechasGuia(e:Event):void{
```

```
if(C==0){
        fc.graphics.lineStyle(1, 0x27F71C); // verde
}
if(C==1){
        fc.graphics.lineStyle(1, 0xBAB7B7); // plomo
}
if(I==0){
        fi.graphics.lineStyle(1, 0x27F71C); //verde
}
if(I==1){
        fi.graphics.lineStyle(1, 0xBAB7B7); //plomo
}
if(D==0){
        fd.graphics.lineStyle(1, 0x27F71C); // verde
}
if(D==1){
        fd.graphics.lineStyle(1, 0xBAB7B7); // plomo
}
if(C==2 && I==2 && D==2){
        fc.graphics.lineStyle(1, 0xF90606); // rojo
        fi.graphics.lineStyle(1, 0xF90606); // rojo
        fd.graphics.lineStyle(1, 0xF90606); // rojo
}

//.....................................centro
var xx:int=320;         var yy:int=450;
var cos:int=0;          var sen:int=100;
var alto:int=10;        var espacio:int=60;
var xc:int; var yc:int; var xcc:int; var ycc:int;
var xi:int; var yi:int; var xii:int; var yii:int;
var xd:int; var yd:int; var xdd:int; var ydd:int;
var base:Number=30;   var aletas:Number=11;
var altocuerpo:Number=40;
var basemedia:Number=base/2; var altocabeza:Number=25;

xc=xx;
yc=yy;

fc.graphics.drawRect(xc-0,yc-0,1,1);//1
fc.graphics.lineTo(xc-0,yc-0);//1
fc.graphics.lineTo(xc-(cos/100)*altocuerpo,yc-(sen/100)*altocuerpo);//2
```

```
fc.graphics.lineTo(xc-(cos/100)*altocuerpo-(sen/100)*
aletas,yc-(sen/100)*altocuerpo+aletas*(cos/100));//3
fc.graphics.lineTo(xc-(altocuerpo+altocabeza+(basemedia*
( (cos/100)/(sen/100) ))  )*(cos/100)+basemedia/(sen/100),
yc-(altocuerpo+altocabeza+(basemedia*( (cos/100)/(sen/100) )))*
(sen/100));//4
fc.graphics.lineTo(xc+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,yc-base*
(cos/100)-altocuerpo*(sen/100)-aletas*(cos/100));//5
fc.graphics.lineTo(xc+base*(sen/100)-altocuerpo*(cos/100) ,
yc-base*(cos/100)-altocuerpo*(sen/100));//6
fc.graphics.lineTo(xc+base*(sen/100) ,yc-base*(cos/100));//7
fc.graphics.lineTo(xc-0,yc-0);//1
fc.graphics.drawRect(xc-0,yc-0,1,1);//

xcc=xx;
ycc=yy+alto;

fc.graphics.drawRect(xcc-0,ycc-0,1,1);//1
fc.graphics.lineTo(xcc-0,ycc-0);//1
fc.graphics.lineTo(xcc-(cos/100)*altocuerpo,ycc-(sen/100)*altocuerpo);//2
fc.graphics.lineTo(xcc-(cos/100)*altocuerpo-(sen/100)*aletas,
ycc-(sen/100)*altocuerpo+aletas*(cos/100));//3
fc.graphics.lineTo(xcc-(altocuerpo+altocabeza+(basemedia*
( (cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100) ,
ycc-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100));//4
fc.graphics.lineTo(xcc+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,ycc-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100));//5
fc.graphics.lineTo(xcc+base*(sen/100)-altocuerpo*
(cos/100) ,ycc-base*(cos/100)-altocuerpo*(sen/100));//6
fc.graphics.lineTo(xcc+base*(sen/100) ,ycc-base*(cos/100));//7
fc.graphics.lineTo(xcc-0,ycc-0);//1
fc.graphics.drawRect(xcc-0,ycc-0,1,1);//
fc.graphics.drawRect(xcc-0,ycc-0,1,-alto);//1
fc.graphics.drawRect(xcc-(cos/100)*altocuerpo,
ycc-(sen/100)*altocuerpo,1,-alto);//2
fc.graphics.drawRect(xcc-(cos/100)*altocuerpo-(sen/100)*
aletas,ycc-(sen/100)*altocuerpo+aletas*(cos/100),1,-alto);//3
fc.graphics.drawRect(xcc-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100) ,
```

```
ycc-(altocuerpo+altocabeza+(basemedia*( (cos/100)/(sen/100) )))*
(sen/100),1,-alto);//4
fc.graphics.drawRect(xcc+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,ycc-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100),1,-alto);//5
fc.graphics.drawRect(xcc+base*(sen/100)-altocuerpo*
(cos/100) ,ycc-base*(cos/100)-altocuerpo*(sen/100),1,-alto);//6
fc.graphics.drawRect(xcc+base*(sen/100) ,ycc-base*(cos/100),1,-alto);//7
//.....centro

//.....izquierda
cos=94; sen=34;
xi=xx-espacio-base*(cos/100)+base/2;
yi=yy;
fi.graphics.drawRect(xi-0,yi-0,1,1);//1
fi.graphics.lineTo(xi-0,yi-0);//1
fi.graphics.lineTo(xi-(cos/100)*altocuerpo,yi-(sen/100)*altocuerpo);//2
fi.graphics.lineTo(xi-(cos/100)*altocuerpo-(sen/100)*
aletas,yi-(sen/100)*altocuerpo+aletas*(cos/100));//3
fi.graphics.lineTo(xi-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) ))  )*(cos/100)+basemedia/(sen/100) ,
yi-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100));//4
fi.graphics.lineTo(xi+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,yi-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100));//5
fi.graphics.lineTo(xi+base*(sen/100)-altocuerpo*
(cos/100) ,yi-base*(cos/100)-altocuerpo*(sen/100));//6
fi.graphics.lineTo(xi+base*(sen/100) ,yi-base*(cos/100));//7
fi.graphics.lineTo(xi-0,yi-0);//1
fi.graphics.drawRect(xi-0,yi-0,1,1);//

yii=yy+alto;
xii=xi;

fi.graphics.drawRect(xii-0,yii-0,1,1);//1
fi.graphics.lineTo(xii-0,yii-0);//1
fi.graphics.lineTo(xii-(cos/100)*altocuerpo,yii-(sen/100)*altocuerpo);//2
fi.graphics.lineTo(xii-(cos/100)*altocuerpo-(sen/100)*
aletas,yii-(sen/100)*altocuerpo+aletas*(cos/100));//3
fi.graphics.lineTo(xii-(altocuerpo+altocabeza+(basemedia*
```

```
((cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100) ,
yii-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100));//4
fi.graphics.lineTo(xii+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,yii-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100));//5
fi.graphics.lineTo(xii+base*(sen/100)-altocuerpo*
(cos/100) ,yii-base*(cos/100)-altocuerpo*(sen/100));//6
fi.graphics.lineTo(xii+base*(sen/100) ,yii-base*(cos/100));//7
fi.graphics.lineTo(xii-0,yii-0);//1
fi.graphics.drawRect(xii-0,yii-0,1,1);//
fi.graphics.drawRect(xii-0,yii-0,1,-alto);//1
fi.graphics.drawRect(xii-(cos/100)*altocuerpo,yii-(sen/100)*
altocuerpo,1,-alto);//2
fi.graphics.drawRect(xii-(cos/100)*altocuerpo-(sen/100)*aletas,
yii-(sen/100)*altocuerpo+aletas*(cos/100),1,-alto);//3
fi.graphics.drawRect(xii-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100),
yii-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100),1,-alto);//4
fi.graphics.drawRect(xii+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,yii-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100),1,-alto);//5
fi.graphics.drawRect(xii+base*(sen/100)-altocuerpo*
(cos/100) ,yii-base*(cos/100)-altocuerpo*(sen/100),1,-alto);//6
fi.graphics.drawRect(xii+base*(sen/100) ,yii-base*(cos/100),1,-alto);//7

//.....izquierda

//.....derecha
cos=-94;
sen=34;

xd=xx+espacio+base;
yd=yy+base*(cos/100);
fd.graphics.drawRect(xd-0,yd-0,1,1);//1
fd.graphics.lineTo(xd-0,yd-0);//1
fd.graphics.lineTo(xd-(cos/100)*altocuerpo,yd-(sen/100)*altocuerpo);//2
fd.graphics.lineTo(xd-(cos/100)*altocuerpo-(sen/100)*
aletas,yd-(sen/100)*altocuerpo+aletas*(cos/100));//3
fd.graphics.lineTo(xd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) ))  )*(cos/100)+basemedia/(sen/100) ,
```

```
yd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100));//4
fd.graphics.lineTo(xd+base*(sen/100)-altocuerpo*(cos/100)+
aletas*(sen/100) ,yd-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100));//5
fd.graphics.lineTo(xd+base*(sen/100)-altocuerpo*(cos/100),
yd-base*(cos/100)-altocuerpo*(sen/100));//6
fd.graphics.lineTo(xd+base*(sen/100) ,yd-base*(cos/100));//7
fd.graphics.lineTo(xd-0,yd-0);//1
fd.graphics.drawRect(xd-0,yd-0,1,1);//

ydd=yy+base*(cos/100)+alto;
xdd=xd;

fd.graphics.drawRect(xdd-0,ydd-0,1,1);//1
fd.graphics.lineTo(xdd-0,ydd-0);//1
fd.graphics.lineTo(xdd-(cos/100)*altocuerpo,ydd-(sen/100)*altocuerpo);//2
fd.graphics.lineTo(xdd-(cos/100)*altocuerpo-(sen/100)*
aletas,ydd-(sen/100)*altocuerpo+aletas*(cos/100));//3
fd.graphics.lineTo(xdd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100),
ydd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100));//4
fd.graphics.lineTo(xdd+base*(sen/100)-altocuerpo*
(cos/100)+aletas*(sen/100) ,ydd-base*(cos/100)-altocuerpo*
(sen/100)-aletas*(cos/100));//5
fd.graphics.lineTo(xdd+base*(sen/100)-altocuerpo*
(cos/100) ,ydd-base*(cos/100)-altocuerpo*(sen/100));//6
fd.graphics.lineTo(xdd+base*(sen/100) ,ydd-base*(cos/100));//7
fd.graphics.lineTo(xdd-0,ydd-0);//1
fd.graphics.drawRect(xdd-0,ydd-0,1,1);//
fd.graphics.drawRect(xdd-0,ydd-0,1,-alto);//1
fd.graphics.drawRect(xdd-(cos/100)*
altocuerpo,ydd-(sen/100)*altocuerpo,1,-alto);//2
fd.graphics.drawRect(xdd-(cos/100)*altocuerpo-(sen/100)*
aletas,ydd-(sen/100)*altocuerpo+aletas*(cos/100),1,-alto);//3
fd.graphics.drawRect(xdd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(cos/100)+basemedia/(sen/100),
ydd-(altocuerpo+altocabeza+(basemedia*
((cos/100)/(sen/100) )))*(sen/100),1,-alto);//4
fd.graphics.drawRect(xdd+base*(sen/100)-altocuerpo*(cos/100)+
```

```
            aletas*(sen/100) ,ydd-base*(cos/100)-altocuerpo*
            (sen/100)-aletas*(cos/100),1,-alto);//5
            fd.graphics.drawRect(xdd+base*(sen/100)-altocuerpo*(cos/100) ,
            ydd-base*(cos/100)-altocuerpo*(sen/100),1,-alto);//6
            fd.graphics.drawRect(xdd+base*(sen/100) ,ydd-base*(cos/100),1,-alto);//7

            //.....derecha

            addChild(fc);
            addChild(fi);
            addChild(fd);
            nFlechasGuia=1;
}
public function Control(I:int,C:int,D:int, Ab:int, Ar:int, Iz:int, De:int, iIz:int,
sIz:int, iDe:int, sDe:int,O:int,ID:int, Prd:int,Er:int ):void{

            D=this.D; C=this.C; I=this.I;Ab=this.Ab; Ar=this.Ar; Iz=this.Iz;
            De=this.De; iIz=this.iIz; sIz=this.sIz; iDe=this.iDe;
            sDe=this.sDe; O=this.O; ID=this.ID; Prd=this.Prd; Er=this.Er;

             var Cuadrado:Shape = new Shape();
            Cuadrado.graphics.lineStyle(120, 0x000000); // verde
            Cuadrado.graphics.drawRect(80,80,200,100);//1

            //----------------Procedimiento 2.0 ----------------------

            //-------Añade primer producto -----------
            if(t==0){
                        if(contador1==0){
                                    CargarImagen(Marcador[(Productos[Producto][0])][5],1);
                                    CargarImagen(Marcador[ID][5],2);
                                    ID=Marcador.length;
                                    Selector(Producto);
                                    contador1++;
                                    t=2;
                                    validador=0;
                        }
            }
            if(visto!=1 && t!=3){
                        ID=Marcador.length;
                        validador=0;
            }
```

```
//-------Añade primer producto ----------


if(t==2 && Producto<Productos.length){
        var CantidadProducto:TextField = new TextField();

        contador5=1;
        CantidadProducto.text ="C= "
        + Marcador[(Productos[Producto][0])][7];
        CantidadProducto.x=100;
        CantidadProducto.y=50;
        CantidadProducto.textColor = 0x19F61D;
        CantidadProducto.scaleX=2;
        CantidadProducto.scaleY=2;
        addChild(CantidadProducto);
}

if(t==2 && ID<Marcador.length  && rec==1 &&
Prd==Productos[Producto][0] && visto==1 && validador==1){
 //recogida de producto con exito
        Productos[Producto][1]="Valido";
        validador=0;
        var p:int=0;
        visto=0;
        contador3++;
        Producto++;
        if(nFlechasGuia==1){
                removeChild(fi);
                removeChild(fc);
                removeChild(fd);
                nFlechasGuia=0;
        }
        if(nFlechasEstante==1){
                removeChild(abajo);
                removeChild(arriba);
                removeChild(derecha);
                removeChild(izquierda);
                removeChild(infizquierdo);
                removeChild(supizquierdo);
                removeChild(infderecho);
```

```
                removeChild(supderecho);
                removeChild(circulo);
                nFlechasEstante=0;
        }
        Selector(Producto);
        rec=0;
        ID=Marcador.length;

        if(Producto< Productos.length){
                p=Productos[Producto][0];
                removeChild(imagenProducto);
                CargarImagen(Marcador[p][5],1);

        }else{
                t=3;
        }
}

if( t==2 && ID<Marcador.length  && rec==1 &&
Prd==Productos[Producto][0] && visto==1 && Er==1){
//recogida de producto con algun error
        Productos[Producto][1]="Alerta";
        Er=0;
        var p:int=0;
        visto=0;
        Producto++;
        if(nFlechasGuia==1){
                removeChild(fi);
                removeChild(fc);
                removeChild(fd);
                nFlechasGuia=0;
        }

        if(nFlechasEstante==1){
                removeChild(abajo);
                removeChild(arriba);
                removeChild(derecha);
                removeChild(izquierda);
                removeChild(infizquierdo);
                removeChild(supizquierdo);
                removeChild(infderecho);
                removeChild(supderecho);
```

```
                removeChild(circulo);
                nFlechasEstante=0;
            }
            Selector(Producto);
            rec=0;
            ID=Marcador.length;


            if(Producto< Productos.length){
                p=Productos[Producto][0];
                removeChild(imagenProducto);
                CargarImagen(Marcador[p][5],1);

            }else{
                t=3;
            }
        }
        if(t==3){
            if(contador4==0){
                removeChild(imagenID);
                removeChild(imagenProducto);
                addChild(Cuadrado);
                Selector(Productos.legnth);
                contador4=1;

            }
            if(validador==1){
                if(nFlechasGuia==1){
                    removeChild(fi);
                    removeChild(fc);
                    removeChild(fd);
                    nFlechasGuia=0;
                }
                var miTexto1:TextField = new TextField();
                miTexto1.text ="Picking Terminado ";
                miTexto1.x=50;
                miTexto1.y=50;
                miTexto1.textColor = 0x19F61D;
                miTexto1.scaleX=2;
                miTexto1.scaleY=2;
                addChild(miTexto1);
```

```
                    var miTexto2:TextField = new TextField();
                    miTexto2.text ="Usuario= "+Marcador[0][6];
                    miTexto2.x=50;
                    miTexto2.y=80;
                    miTexto2.textColor = 0x19F61D;
                    miTexto2.scaleX=2;
                    miTexto2.scaleY=2;
                    addChild(miTexto2);

                    for (var u:int=0; u<Productos.length; u++) {
                            var miTexto3:TextField = new TextField();
                            miTexto3.text =
                            "Producto"+u+"="+Productos[u][1];
                            miTexto3.x=50;
                            miTexto3.y=110+30*u;
                            miTexto3.textColor = 0x19F61D;
                            miTexto3.scaleX=2;
                            miTexto3.scaleY=2;
                            addChild(miTexto3);
                    }
            }
}
//--------------Añade flechas para ubicarse en el estante.--------

if(contador0==0){
            Sonido();
}
contador0++;
if(O==2){
            if(nFlechasGuia==1){
                    removeChild(fi);
                    removeChild(fc);
                    removeChild(fd);
                    nFlechasGuia=0;
            }
            dispatchEvent(new Event(Multiple_Markers.CLICKKK));
            // Flechas estante
}
//--------------Añade flechas para ubicarse en el estante.--------

//--------------Añade flechas para ubicarse en el almacen.--------
if(O==1){
```

```
                    if(nFlechasEstante==1){
                        removeChild(abajo);
                        removeChild(arriba);
                        removeChild(derecha);
                        removeChild(izquierda);
                        removeChild(infizquierdo);

                        removeChild(supizquierdo);
                        removeChild(infderecho);
                        removeChild(supderecho);
                        removeChild(circulo);
                        nFlechasEstante=0;
                    }
                    dispatchEvent(new Event(Multiple_Markers.CLICK));
                    //flechasGuia
                }
                //--------------Añade flechas para ubicarse en el almacen.--------

                //-----------------Procedimiento 2.0 -----------------------
            }
            /* Augmented reality initialisation */
            private function initFLAR():void {
                /* Initialise FLARManager */
                this.fm = new FLARManager("flarConfig.xml");
                /* Temporary declaration of how many patterns are being used */
                var numPatterns:int = 6;
                /* Initialise markerByPatternId vector object */
                this.markersByPatternId =
                new Vector.<Vector.<FLARMarker>>(numPatterns, true);
                /* Loop through each pattern */
                while (numPatterns--) {
                    /* Add empty Vector to each pattern */
                    this.markersByPatternId[numPatterns] =
                    new Vector.<FLARMarker>();
                }
                /* Initialise empty containersByMarker dictionary object */
                this.containersByMarker = new Dictionary(true);
                /* Initialise empty nodesByMarker dictionary object */
                this.nodesByMarker = new Dictionary(true);
                /* Event listener for when a new marker is recognised */
                fm.addEventListener
```

```
            (FLARMarkerEvent.MARKER_ADDED, this.onAdded);
            /* Event listener for when a marker is removed */
            fm.addEventListener
            (FLARMarkerEvent.MARKER_REMOVED, this.onRemoved);
            /* Event listener for when the FLARManager object has loaded */
            fm.addEventListener(Event.INIT, this.onFlarManagerLoad);
            /* Display webcam */
            this.addChild(Sprite(fm.flarSource));
    }
    private var posicion:int = 0;
    public function CargarImagen(url:String,valor:int):void{
            posicion=valor;
            if(valor==1){
                    imagenID.contentLoaderInfo.
                    addEventListener(Event.COMPLETE, imagenCargada);
                    imagenID.contentLoaderInfo.
                    addEventListener(IOErrorEvent.IO_ERROR, imagenError);
                    imagenID.contentLoaderInfo.addEventListener(ProgressEvent.
                    PROGRESS, imagenProgreso);
                    imagenID.load( new URLRequest(url));
            }
            if(valor==2){
                    imagenProducto.contentLoaderInfo.
                    addEventListener(Event.COMPLETE, imagenCargada);
                    imagenProducto.contentLoaderInfo.
                    addEventListener(IOErrorEvent.IO_ERROR, imagenError);
                    imagenProducto.contentLoaderInfo.
                    addEventListener(ProgressEvent.PROGRESS,
                    imagenProgreso);
                    imagenProducto.load( new URLRequest(url));
            }
    }
    private function imagenCargada(e:Event):void {
            trace("Cargada: " + e);
            if(posicion==1){
                    imagenID.x=20;
                    imagenID.y=20;
            }
            if(posicion==2){
                    imagenProducto.x=1;
                    imagenProducto.y=300;
            }
```

```
            posicion=0;
            addChild(imagenID);
            addChild(imagenProducto);
}
private function imagenError(e:IOErrorEvent):void {
            trace("Error: " + e);
}
private function imagenProgreso(e:ProgressEvent):void {
            trace("Cargados=" + e.bytesLoaded + " Total=" + e.bytesTotal);
}
public function Sonido():void {
            var mySound:Sound = new Sound();
            var myChannel:SoundChannel = new SoundChannel();
            var myTransform = new SoundTransform();
            var lastPosition:Number = 0;
            mySound.load(new URLRequest("5.mp3"));
            myChannel = mySound.play();
            myTransform.volume = 0.5;
            myChannel.soundTransform = myTransform;
}

/* Run if FLARManager object has loaded */
private function onFlarManagerLoad(e:Event):void {
            /* Remove event listener so this method doesn't run again */
            this.fm.removeEventListener(Event.INIT, this.onFlarManagerLoad);
            /* Run Papervision initialisation method */
            this.initPaperVision();
}

/* Run when a new marker is recognised */
private function onAdded(e:FLARMarkerEvent):void {
            /* Run method to add a new marker */
            this.addMarker(e.marker);
}

/* Run when a marker is removed */
private function onRemoved(e:FLARMarkerEvent):void {
            /* Run method to remove a marker */
            this.removeMarker(e.marker);

}
```

```
/* Add a new marker to the system */
private function addMarker(marker:FLARMarker):void {
        /* Store reference to list of existing markers with same pattern id */
        var markerList:Vector.<FLARMarker> =
        this.markersByPatternId[marker.patternId];
        /* Add new marker to the list */
        markerList.push(marker);
        /* Initialise the marker container object */
        var container:DisplayObject3D = new DisplayObject3D();
        /* Prepare material to be used by the Papervision cube based
        on pattern id */
        var flatShaderMat:FlatShadeMaterial = new
        FlatShadeMaterial(pointLight3D, getColorByPatternId(marker.patternId),
        getColorByPatternId(marker.patternId, true));
        /* Add material to all sides of the cube */
        var cubeMaterials:MaterialsList = new
        MaterialsList({all: flatShaderMat});
        /* Initialise the cube with material and set dimensions of all sides to 40 */
        var cube:Cube = new Cube(cubeMaterials, 40, 40, 40);

        /* Shift cube upwards so it sits on top of paper instead of
        being cut half-way */
        cube.z = 0.5 * 40;

        /* Scale cube to 0 so it's invisible */
        cube.scale = 0;
        /* Add animation which scales cube to full size */
        Tweener.addTween(cube, {scale: 1, time:0.5,
        transition:"easeInOutExpo"});

        /* Set cube to be individually affected by filters */
        cube.useOwnContainer = true;
        /* Add cellshaded border using glow filter */
        cube.filters = [this.glow];
        /* Add finished cube object to marker container */
        container.addChild(cube);
        /* Add marker container to the Papervision scene */
        this.scene3D.addChild(container);
        /* Add marker container to containersByMarker Dictionary object */
        this.containersByMarker[marker] = container;
}
```

```
/* Remove a marker from the system */
private function removeMarker(marker:FLARMarker):void {
        /* Store reference to list of existing markers with same pattern id */
        var markerList:Vector.<FLARMarker> =
        this.markersByPatternId[marker.patternId];
        /* Find index value of marker to be removed */
        var markerIndex:uint = markerList.indexOf(marker);
        /* If marker exists in markerList */
        if (markerIndex != -1) {
                /* Remove marker from markersByPatternId */
                markerList.splice(markerIndex, 1);
        }
        /* Store reference to marker container from containersByMarker
        Dictionary object */
        var container:DisplayObject3D = this.containersByMarker[marker];
        /* If a container exists */
        if (container) {
                /* Remove container from the Papervision scene */
                this.scene3D.removeChild(container);
        }
        /* Remove container reference from containersByMarker
        Dictionary object */
        delete this.containersByMarker[marker]
        /* Clear any corner nodes for this marker from the display */
        this.nodesByMarker[marker].graphics.clear();
        /* Remove reference to corner nodes for this marker from
        nodesByMarker Dictionary object */
        delete this.nodesByMarker[marker];
}

/* Papervision initialisation method */
private function initPaperVision():void {
        /* Initialise a new Papervision scene */
        this.scene3D = new Scene3D();
        /* Initialise a new FLARCamera3D object to enable full AR goodness */
        this.camera3D = new FLARCamera3D(this.fm.cameraParams);
        /* Define a new Papervision viewport object */
        this.viewport3D = new Viewport3D(640, 480, true);
        /* Add viewport to the main scene */
        this.addChild(this.viewport3D);
```

```
            /* Define a new Papervision point light */
            this.pointLight3D = new PointLight3D(true, false);
            /* Set light position */
            this.pointLight3D.x = 1000;
            this.pointLight3D.y = 1000;
            this.pointLight3D.z = -1000;
            /* Add light to the Papervision scene */
            this.scene3D.addChild(pointLight3D);
            /* Initialise the Papervision render engine */
            this.lre = new LazyRenderEngine(this.scene3D,
            this.camera3D, this.viewport3D);
            /* Create event listner to run a method on each frame */
            this.addEventListener(Event.ENTER_FRAME, this.onEnterFrame);
}

/* Method to run on each frame */
private function onEnterFrame(e:Event):void {
            /* Loop through corner nodes */
            for (var marker:Object in nodesByMarker) {
                        /* Clear any corner nodes for this marker from the display */
                        nodesByMarker[marker].graphics.clear();
            }
            /* Run method to update markers */
            this.updateMarkers();
            /* Render the Papervision scene */
            this.lre.render();
}

private var opnum:int=0;
/* Update markers method */

private function updateMarkers():void {
            /* Store reference to amount of patterns being tracked */
            var i:int = this.markersByPatternId.length;
            /* Store reference to list of existing markers */
            var markerList:Vector.<FLARMarker>;
            /* Empty marker variable */
            var marker:FLARMarker;
            /* Empty container variable */
            var container:DisplayObject3D;
            /* Empty integer */
            var j:int;
```

```
/* Loop through all tracked patterns */
while (i--) {
        /* Store reference to all markers with this pattern id */
        markerList = this.markersByPatternId[i];
        /* Amount of markers with this pattern */
        j = markerList.length;
        /* Loop through markers with this pattern */
        while (j--) {
                /* Store reference to current marker */
                marker = markerList[j];
                /* Initialise a new Shape object */
                var nodes:Shape = new Shape();
                /* Define line style for corner nodes */
                /* Store reference to coordinates for each corner
                of the marker */
                var corners:Vector.<Point> = marker.corners;
                var TipoMarcador:int=0;
                var PdUEz:Number=0;
                var PdUEy:Number=0;
                var McdUEz:Number=0;
                var McdUEy:Number=0;
                var Ry:Number=0;
                var Rz:Number=0;

                for (var u:int=0; u<Marcador.length; u++) {
                        var Yu:int= Marcador[u][1];
                        if(Yu==0){
                        // ID(solo se asignan al iniciar el programa)
                                if(marker.patternId == u){
                                        ID=u; O=0;
                                        validador=1;
                                        Control(I,C,D,Ab,Ar,Iz,De,
                                        iIz,sIz,iDe,sDe,O,ID,Prd,Er);
                                }
                        }
                        if(Yu==1){          // Producto
                                if(marker.patternId == u){
                                        Prd=u; O=0; visto=1;
                                        Control(I,C,D,Ab,Ar,Iz,De,
                                        iIz,sIz,iDe,sDe,O,ID,Prd,Er);
                                }
```

```
}
if(Yu==2){
// Flechas Guia(no cambian en todo el
trayecto hacia 1 producto)
        var tu:int= Trayectoria.length;
        if(marker.patternId == u){
                C=2; D=2; I=2;
                rec=1; O=1;

                for (var t:uint=0; t<tu; t++) {
                        var A:int=
                        Trayectoria[t][0];
                        var B:int=
                        Trayectoria[t][1];
                        if(A==u){
                                if(B=="C"){
                                C=0;
                                D=1;
                                I=1;
                                }
                                if(B=="D"){
                                C=1;
                                D=0;
                                I=1;
                                }
                                if(B=="I"){
                                C=1;
                                D=1;
                                I=0;
                                }
                        }
                        Control(I,C,D,Ab,Ar,
                        Iz,De,iIz,sIz,iDe,sDe,
                        O,ID,Prd,Er);
                }
        }
}
if(Yu==3){
// Flechas Estante(cambian en cada producto)
        if(marker.patternId == u){
                var EE:int= Marcador.length;
                for (var tt:uint=0; tt<EE; tt++) {
```

```
var FF:int= Marcador[tt][1];
var GG:int= Marcador[tt][4];
var HH:int= Marcador[tt][3];
        if(FF==1){
                PdUEz=G;
                PdUEy=HH;
        }

}
McdUEz=Marcador[u][4];
McdUEy=Marcador[u][3];
Ry=-(PdUEy-McdUEy);
Rz=-(PdUEz-McdUEz);
Ab=2; Ar=2; Iz=2; De=2;
iIz=2; sIz=2; iDe=2; sDe=2;

if(Ry>0 && Rz>0){
        sDe=0; Ab=1;
        Ar=1; Iz=1;
        De=1; iIz=1;
        sIz=1; iDe=1;
}  // I
if(Ry<0 && Rz>0){
        sIz=0; Ab=1;
        Ar=1; Iz=1;
        De=1; iIz=1;
        iDe=1; sDe=1;
}// II
if(Ry<0 && Rz<0){
        iIz=0; Ab=1;
        Ar=1; Iz=1;
        De=1; sIz=1;
        iDe=1; sDe=1;
}       // III
if(Ry>0 && Rz<0){
        iDe=0; Ab=1;
        Ar=1; Iz=1;
        De=1; iIz=1;
        sIz=1; sDe=1;
}       // IV
```

```
                                        if(Ry>0 && Rz==0){
                                                De=0;Ab=1;
                                                Ar=1; Iz=1;
                                                iIz=1; sIz=1;
                                                iDe=1; sDe=1;
                                        }  // I



                                        if(Ry<0 && Rz==0){
                                                Iz=0; Ab=1;
                                                Ar=1; De=1;
                                                iIz=1; sIz=1;
                                                iDe=1; sDe=1;
                                        }       // II
                                        if(Ry==0 && Rz<0){
                                                Ab=0; Ar=1;
                                                Iz=1; De=1;
                                                iIz=1; sIz=1;
                                                iDe=1;sDe=1;
                                        }       // III
                                        if(Ry==0 && Rz>0){
                                                Ar=0; Ab=1;
                                                Iz=1; De=1;
                                                iIz=1; sIz=1;
                                                iDe=1; sDe=1;
                                        }       // IV
                                        O=2;
                                        Control(I,C,D,Ab,
                                        Ar,Iz,De,iIz,sIz,iDe,
                                        sDe,O,ID,Prd,Er);
                                }
                        }
                        if(Yu==4){   // error
                                if(marker.patternId == u){
                                        Er=1;
                                        Control(I,C,D,Ab,
                                        Ar,Iz,De,iIz,sIz,iDe,
                                        sDe,O,ID,Prd,Er);
                                }
                        }
                }
```

```
                                        /* Empty coordinate variable */
                                        var vertex:Point;
                                        /* Loop through rest of corner coordinates */
                                        for (var c:uint=0; c<corners.length; c++) {
                                        /* Store reference to current corner coordinates */
                                                vertex = corners[c];


                                                /* Draw a 2D circle at these coordinates */
                                                nodes.graphics.
                                                drawCircle(vertex.x, vertex.y, 5);
                                        }
                                        /* Add reference to corner nodes to
                                        nodesByMarker Dictionary object */
                                        this.nodesByMarker[marker] = nodes;
                                        /* Add corner nodes to the main scene */
                                        /* Find reference to marker container in
                                        containersByMarker Dictionary object */
                                        container = this.containersByMarker[marker];
                                        /* Transform container to new position in 3d space */
                                        container.transform =FLARPVGeomUtils.
                                        convertFLARMatrixToPVMatrix
                                        (marker.transformMatrix);
                                }
                        }
                }

                /* Get colour values dependent on pattern id */
                private function getColorByPatternId(patternId:int, shaded:Boolean=false):Number{
                        switch (patternId) {
                                case 0:
                                        if (!shaded)
                                                return 0xFF1919;
                                        return 0x730000;
                                case 1:
                                        if (!shaded)
                                                return 0xFF19E8;
                                        return 0x730067;
                                case 2:
                                        if (!shaded)
                                                return 0x9E19FF;
```

```
                    return 0x420073;
                case 3:
                    if (!shaded)
                        return 0x192EFF;
                    return 0x000A73;


                case 4:
                    if (!shaded)
                        return 0x1996FF;
                    return 0x003E73;
                case 5:
                    if (!shaded)
                        return 0x19FDFF;
                    return 0x007273;
                default:
                    if (!shaded)
                        return 0xCCCCCC;
                    return 0x666666;
            }
        }
    }
}
```