Sistema de información para el control de despacho de productos y selección de rutas óptimas para la Distribuidora Santander en Tuluá, Valle.

Carlos Andrés López Ramírez
Código: 200557891

Universidad del Valle Sede Tuluá
Ingeniería de Sistemas
16 de febrero de 2016

Sistema de información para el control de despacho de productos y selección de rutas óptimas para la Distribuidora Santander en Tuluá, Valle.

Carlos Andrés López Ramírez

Código: 200557891

Trabajo de grado.

Director

Julián Andrés Rodas Laverde, Ingeniero

Universidad del Valle Sede Tuluá

Ingeniería de Sistemas

16 de febrero de 2016

Nota de Aceptación

_____
_____
_____
_____
_____
_____
_____
_____
_____

Presidente del Jurado

_____

Jurado

_____

Jurado

_____

Tuluá, 16 de febrero de 2016

Agradecimientos

A mi familia que a pesar de los inconvenientes siguió apoyarme, a los profesores que me colaboraron en las dificultades de este trabajo.

ÍNDICE DE FIGURAS

LISTA DE TABLAS

RESUMEN

El presente trabajo tiene como propósito identificar los elementos que influyen en los proceso de control del despacho de pedidos y la selección de ruta de la Distribuidora Santander. Se implementó una aplicación que ayude a mejorar la calidad y la eficiencia en la prestación de los servicios dados y de soporte al control de las operaciones de los domicilios. Mediante el análisis de los procesos se planteó una solución al problema del despacho de pedidos y selección de rutas y se diseñó el sistema de información y los algoritmos, aplicando técnicas de ingeniería para definir un modelo que se adapte a las necesidades del cliente. Finalmente se validó el sistema aplicando pruebas de software.

Palabras Clave: Algoritmo, Software, Algoritmo Genético

ESTRUCTURA DEL DOCUMENTO

Capítulo 1. Se presenta una introducción al tema tratado exponiendo de manera abreviada cada uno de los puntos del proyecto.

Capítulo 2. Se hace una breve exposición del estado del arte del problema, dando información de cada uno de los puntos que se tuvieron en cuenta en el desarrollo de la solución, exponiendo la teoría necesaria que facilite la comprensión del contenido del documento y las técnicas aplicadas en la solución del problema.

Capítulo 3.Se describe el desarrollo del proyecto exponiendo tres partes fundamentales, el análisis, diseño e implementación. En el análisis se exponen los métodos utilizados para la extracción de la información necesaria para el desarrollo de la solución. En el diseño se da una vista de las características básicas de la solución, estructuras, algoritmos y otros aspectos que forman parte del sistema desarrollado y en la implementación se expone el resultado de las dos fases anteriores, mostrando características importantes de la solución.

Capítulo 4. Se exponen diferentes resultados obtenidos en el desarrollo que permiten conocer el alcance de la solución dada al problema de rutas y otros aspectos del análisis.

Capítulo 5. Se dan las conclusiones del proyecto a través del análisis de los resultados.

1. INTRODUCCIÓN

El transporte es un problema aplicable a múltiples campos y que diariamente se puede evidenciar su importancia. La necesidad de transporte, no solo de personas sino también de mercancías e información, es un tema de gran interés debido a la creciente demanda de la población.

Estas necesidades no se enfocan solo en la cantidad de vehículos o redes que presten el servicio, sino también en que las rutas por las que se desplazan los vehículos o información sean las mejores. En el caso del transporte de mercancía, el problema es más complejo, ya que los clientes requieren tener sus productos en tiempos y horarios aceptables que no alteren sus actividades, razón por la cual las empresas buscan formas de mejorar esta labor vital para su economía [1]. Además tratan de reducir los costos adjuntos que en muchas ocasiones pueden generar pérdidas debido a una mala planeación.

En el caso del transporte de mercancía, variables como los horarios de los clientes, la cantidad de productos, los costos del transporte, el número de vehículos, y la satisfacción de los clientes, son factores vitales que influyen en la economía e imagen de la empresa[1].

El presente trabajo se enfoca en el transporte de productos de la Distribuidora Santander, una pequeña empresa de la ciudad de Tuluá que además de vender sus productos presta el servicio de domicilio a sus clientes y a medida que crece, aumentan las quejas de los domicilios por su demora haciendo evidente la necesidad de una solución que se adapte a sus necesidades y permita mantener el control de este proceso.

1.1. PLANTEAMIENTO DEL PROBLEMA

La Distribuidora Santander es una microempresa que vende bebidas al mayor y al detal en su local ubicado en la ciudad de Tuluá, pero la mayoría de sus ventas son realizadas a domicilio. Ésta dispone de cuatro empleados que reparten los productos, tres de ellos lo hacen en moto con remolque, y el último en un carro.

Esta empresa tiene varios problemas en su forma de operar, para el despacho de los pedidos se realiza un proceso manual donde se anotan los productos despachados y los elementos que deben ser devueltos por los clientes, lo que da lugar a errores humanos y genera descontrol, de esta misma forma son tratados los cambios de productos. En el caso de productos pendientes no se tiene una forma de establecer si el producto fue o no entregado al cliente, causando que se envié el producto en más de una ocasión, y en el caso de las devoluciones, depende de la memoria de la persona encargada de recibirlas ya que, en el momento, no se tiene un control establecido para esto. Además de lo anterior, la ruta para entregar los pedidos es selecciona por cada empleado a su criterio teniendo en cuenta la capacidad del remolque, que en diversas ocasiones es sobrepasada, esto ocasiona, en muchos casos, demoras en la entrega de los productos y sobrecostos ya que la ruta seleccionada puede no ser la más óptima.

A pesar de tener software de facturación y contabilidad, al momento en que los empleados entregan el dinero de los pedidos lo hacen de memoria, ya que no se controla el número de factura que se despacha ni se usan facturas de triple copia, lo que en muchos casos causa demoras en sus salidas y descontrol en caja. Teniendo en cuenta los problemas anteriormente nombrados, ¿En qué forma se puede mejorar el control en despacho, devolución y cambio de productos y la elección de las rutas?

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollar un sistema de información para el control en el despacho de productos y selección de ruta óptima para la Distribuidora Santander en Tuluá Valle del Cauca

1.2.2. OBJETIVOS ESPECIFICOS

- Implementar un módulo para el control del despacho, cambio y devolución de productos.
- Implementar un módulo para la selección de ruta optima teniendo en cuenta la capacidad de cada remolque y el orden de los pedidos.
- Implementar un módulo para generar reportes de costos de transportes y trayecto de cada domicilio.
- Permitir la visualización de rutas en un mapa de la ciudad a través de dispositivos móviles.
- Implementar el sistema de información en la Distribuidora Santander

1.3. RESULTADOS ESPERADOS

De cada fase del desarrollo del proyecto se obtienen diferentes entregables como se muestra en el siguiente cuadro:

Análisis:	• Requerimientos. • Historias de usuario.
Diseño:	• Diagrama de Clases. • Modelo de base de datos. • Modelo de la Vista. • Modelo de navegación.
Codificación:	• Código fuente de la aplicación.
Pruebas:	• Pruebas unitarias. • Pruebas de aceptación.
Implementación:	• Manual del usuario. • Manual de instalación. • CD con software y archivos necesarios para la instalación.

Tabla 1: *Fases del desarrollo y entregables*

Como resultado del cumplimiento de los objetivos se tiene como resultado en el software los siguientes:

- Un módulo en el cual se verá el listado de facturas pendientes por despachar, los cambios y productos pendientes de facturas ya despachadas, además se debe tener en cuenta los pedidos que deben ser despachados a horas específicas.
- El sistema deberá retornar la ruta óptima para el transporte de pedidos teniendo en cuenta la capacidad de los remolques y variables como calles bloqueadas, entregas prioritarias, además de esto las rutas se deben calcular en el menor tiempo posible, es decir, no más de 5 minutos.
- El sistema deberá permitir asignar una ruta ya calculada a un domicilio y relacionar los productos que deben retornar, además de registrar las devoluciones y faltantes de productos retornables.
- El sistema generará reportes sobre los costos de transportes medidos a través del recorrido de las rutas.
- Cada domicilio podrá ver su respectiva ruta en un mapa a través de dispositivos móviles, como Smartphone o Tablet PC.
- El sistema será puesto en marcha utilizando PostgreSQL, como sistema gestor de base de datos.
- El sistema será compatible con los navegadores Mozilla Firefox y Google Chrome y se utilizará GlassFish Server Open Source Edition como servidor web, la Distribuidora Santander debe disponer de un servidor local en el cual será alojado el sistema.

2. MARCO DE REFERENCIA

2.1. MARCO TEORICO

2.1.1. TRANSPORTE

El transporte es un tema ampliamente estudiado del que a diario se puede evidenciar la necesidad de llevar a cabo estudios para un desarrollo ágil de las actividades asociadas a este, que satisfagan las expectativas de la creciente población y empresas que se benefician de esta actividad.

Se puede definir el transporte como todos los requisitos relacionados con la necesidad de situar productos o personas en los puntos de destino correspondientes de acuerdo con unas condiciones de seguridad, rapidez y costo que permitan satisfacer las necesidades de los clientes. La función del transporte está relacionada con diferentes aspectos, tanto de un punto de vista jurídico, como técnico-económico. [1]

Para poder realizar esta acción es necesario tener los elementos necesarios, tales como:

- Infraestructura: es la parte física necesaria para dar aplicación al transporte, es decir, vías que permitan movilizar personas y/o productos.
- Vehículo: Es el instrumento que permite el traslado de personas u objetos.
- El operador: es la persona encargada de conducir el vehículo.
- Normas y leyes: Es la parte principal del sistema de transportes, es la que dictamina la manera de trasladarse de un lugar a otro, asimismo es la que regula y norma la operación de todos los demandantes y ofertantes del servicio de transporte. [2]

Hay aspectos a tener en cuenta en lo referido al transporte, el tiempo no se enfoca solo al físico de personas o mercancías, si no al periodo comprendido desde que la persona o mercancía está dispuesta para su abordo o carga hasta que llega físicamente a su lugar de destino, en lo cual se debe incluir tiempos de espera, carga y descarga - abordo y desabordo de vehículos, paros en la ruta, transbordos, etc. [1]

Una parte importante del transporte es la planificación, tarea que implica estudiar las demandas de movilidad de personas y productos. Este tema es de gran importancia debido a la constante necesidad de movilización, en este caso las empresas que prestan el servicio de transporte de personas y/o mercancía encuentran la necesidad de desarrollar formas de cumplir con su

labor oportunamente y esto se logra mediante la planificación y selección de las rutas. [1]

2.1.2. PROBLEMA DE RUTAS

La selección de rutas es un problema que se aplica a muchos contextos, desde la vida diaria de las personas, hasta la industria que busca formas de reducir la cantidad de insumos y mejorar el funcionamiento de sus procesos.

Este problema sigue siendo estudiado debido a la amplia gama de aplicaciones que tienen en el mundo real sobre el transporte y otros campos en que la selección de rutas puede ser aplicado como lo son la impresión de circuitos, el análisis de estructura de cristales, redes de comunicación, etc.

Debido a lo anteriormente nombrado, se han desarrollado algoritmos y técnicas que permiten encontrar o acercarse a una solución óptima. Entre estas se utilizan algoritmos de programación dinámica, programación línea y computación evolutiva.

Algunos los algoritmos de programación dinámica y voraz son los algoritmos de Dijkstra, Bellman-Ford, búsqueda A*, Floyd-Warshall y de Johnson [3] entre otros, los cuales aplican a diferentes contextos del problema para encontrar rutas óptimas. Además de estos existen también otros métodos matemáticos de hallar rutas óptimas, como lo son la Teoría de perturbaciones [4] [5], el algoritmo de colonia de hormigas [6] [7], el método de Vogel [8][9][10] y las técnicas de computación evolutiva, redes neuronales, algoritmos genéticos, etc.

2.1.3. PROBLEMA DEL AGENTE VIAJERO

El problema del agente viajero (TSP, por sus siglas en inglés) consiste en, dado una lista de vértices y los pesos de las aristas, hallar el camino más corto que pase por cada nodo una sola vez, y regrese al de origen. Este es un problema NP-Completo de optimización combinatoria, que tiene una amplia gama de aplicaciones en la investigación de operaciones y ciencias de la computación y del que se han sacado diferentes variaciones, este puede ser formulado de la siguiente forma.

$$sea\ x_{ij} = \begin{cases} 1, & si\ existe\ un\ camino\ entre\ i\ y\ j \\ 0, & en\ tro\ caso \end{cases}$$

$$Para\ el\ conjunto\ de\ ciudades\ 0, 1, \dots, n$$

$$Sean\ u_i, para\ i = 0, 1, \dots, n\ variables\ artificiales$$

$$y\ sea\ c_{ij}\ la\ distancia\ desde\ \ i\ a\ \ j$$

Entonces el modelo de programación lineal de enteros se puede escribir como:

$$min \sum_{i=0}^{n} \sum_{i \neq j}^{n} {}_{j=0} c_{ij} x_{ij}\,, \qquad 0 \leq x_{ij} \leq 1$$

$$Siendo\ x_{ij}\ un\ entero, \quad i, j = 0, 1, \dots, n$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1, \quad j = 0, 1, \dots, n$$

$$\sum_{j=0, i \neq j}^{n} x_{ij} = 1, \quad i = 0, 1, \dots, n$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad 1 \leq i \neq j \leq n$$

Esta última restricción previene que se generen caminos internos evitando así que se omitan nodos.

2.1.3.1. VARIACIONES DEL TSP

El TSP tiene diversas variaciones que han sido originadas en casos de la vida real o aplicaciones potenciales, algunas de estas son [19]:

- Max-TSP: a diferencia del TSP, el objetivo de esta variación es encontrar el camino en el cual la suma total de pesos de las aristas sea máxima.

- TSP con cuello de Botella: en este caso, el objetivo es encontrar el camino tal que el costo del eje más grande sea el menor posible.

- TSP con múltiples visitas: En este se busca la ruta para un vendedor, que inicia en un nodo de un grafo *G* y visita cada nodo al menos una vez y regresa al nodo de inicio de tal forma que la distancia del recorrido sea mínima.

- TSP con múltiples viajeros: Asuma que *n* vendedores están localizados en X_1 donde $X_1 \in G$. Iniciando en X_1 cada vendedor visitara un subconjunto X_i de nodos de *G* exactamente una vez y regresara X_1. El interés en este problema es encontrar una partición $X_1, X_2, \dots, X_m$ de $V - \{1\}$ y una ruta para cada uno de los *m* vendedores tal que:

  I. $|X_i| \geq 1 \qquad para\ cada\ i$

  II. $\cup_{i=1}^{m} X_i = V - \{1\}$

  III. $X_i \cap X_j = 0 \quad para\ i \neq j$

IV. *La distancia viajada por todos los vendedores esta minimizada.*

- Problema de ruteo de vehículos: Esta variación tiene un enfoque de aplicación a la realidad y tiene en cuenta no solo la ruta, sino también, parámetros económicos y de tiempo.

2.1.4. EL PROBLEMA DE LA MOCHILA

El problema de la mochila al igual que el TSP es un problema de optimización combinatoria, NP-Duro.

Este problema consiste en dados n objetos cada uno con un peso w_j y un valor v_j, se debe escoger el conjunto de objetos cuyo valor sea máximo, sin exceder el peso máximo W [24].

En este problema la solución se suele representar mediante un vector $[x_1, x_2, \dots, x_n]$ donde cada componente i y su valor indican si el objeto ha sido seleccionado o no. Considerando esta representación se puede formular del siguiente modo:

$$KP = \begin{cases} max: \sum_{i=1}^{n} v_i x_i \\ s.a, \\ \sum_{i=1}^{a} w_i x_i \leq W \end{cases}$$

2.1.5. ALGORITMOS EVOLUTIVOS

Los algoritmos evolutivos son métodos de resolución de problemas de optimización o búsqueda que tienen como elemento clave de su diseño los postulados de la evolución biológica. Estos trabajan con un conjunto de potenciales soluciones llamada población de individuos, las cuales se mezclan para obtener individuos más aptos, mejores soluciones. Cada individuo es medido a través de una función de aptitud, para determinar qué tan bueno es [14][16].

2.1.5.1. ALGORITMOS GENETICOS (GA's)

Los algoritmos genéticos son los más conocidos y utilizados de los algoritmos evolutivos. Estos son simulaciones de la selección natural que pueden resolver problemas de optimización, aunque se pueden aplicar a una gama de problemas más amplia.

Estos son algoritmos iterativos que funcionan en base a un conjunto de posibles soluciones que mejoran en cada generación, al conjunto de soluciones se le llama población y cada posible solución es conocida como individuo. Los individuos de un GA son representados en una estructura de acuerdo al problema a resolver, esta estructura se le llama cromosoma.

La representación más comunes de un cromosoma son la binaria, vectorial, matricial, etc. esta puede cambiar de acuerdo al problema a solucionar. En este caso se utiliza una representación vectorial.

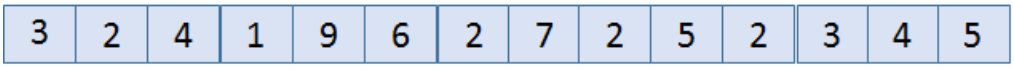


***Figura 1**: Representación de un cromosoma*

En cada iteración se crea nueva población con el fin de hallar nuevas soluciones potenciales, manteniendo siempre los individuos más aptos de la generación anterior para garantizar la convergencia del algoritmo. Esta nueva población se consigue utilizando el operador de cruce, en el cual dos individuos son utilizados como padres y nuevos individuos son formados por el intercambio de sub-secuencias entre los padres y evaluándolos con por medio de una función de aptitud.

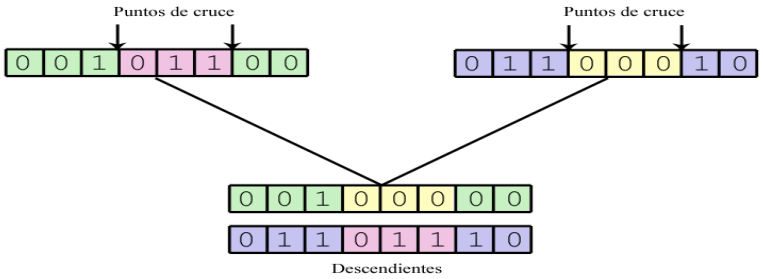


***Figura 2**: Cruce [20]*

La mutación es otra forma de explorar nuevas soluciones potenciales al problema. En la biología la mutación es relativamente rara, al menos en la medida en que afecta a la descendencia. En la mayoría de las implementaciones de GA's, la mutación es también rara, (en un orden de un 2%), pero, en general, no se puede decir que sea el ajuste correcto para la tasa de mutación de los GA's. La mejor tasa de mutación depende del problema, la representación, el tamaño de la población, codificación, entre otros factores.

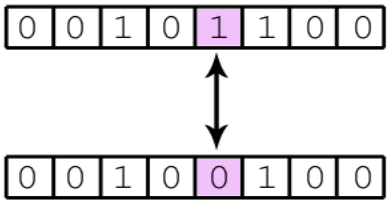


***Figura 3**: Mutación [20]*

Después del cruce y la mutación se realiza la selección, esta operación consiste en escoger los individuos de la nueva generación y los de la generación anterior que se conservaran.

Existe también otra operación que se realiza en el caso de algoritmos genéticos en que se aplica el modelo de islas. El modelo de islas no trabaja sobre una sola población, si no sobre un conjunto de poblaciones que se tratan de forma individual y se mezclan a través de la migración. Esta operación consiste en sacar individuos de una población e incorporarlos en otra.

El modelo de islas es un modelo usado en algoritmos genéticos distribuidos, pero no necesariamente debe ser un algoritmo distribuido para su aplicación. Este modelo se puede aplicar para computación en paralelo en una misma máquina.

2.2. ESTADO DEL ARTE

2.2.1. ANTECEDENTES

El problema de la selección de ruta óptima ha sido tratado en muchas ocasiones dando como resultado diversos algoritmos y métodos para conseguir una solución, debido a esto se han desarrollado aplicaciones que usan dichas técnicas para ofrecer productos de software que faciliten a las empresas realizar la distribución de sus productos en forma más eficiente, esto es también aplicado al transporte público.

Se pueden encontrar diversas herramientas de software para la selección de rutas óptimas para empresas de transporte de mercancías, en el siguiente cuadro se nombran algunas aplicaciones con sus funciones.

Software	Características
Sistema de gestión logística (SGL) [21]	• Herramienta de Simulación de Ruteos • Herramientas de Diseño de la Redo de Transporte • Herramienta de Estimación de Costos de Transporte • Ruteo Dinámico de los Pedidos • Armado de Repartos • Emisión de las Hojas de Ruta • Interfaz de Pedidos a Ratear • Geo-Codificación de los Puntos de Entrega

Software	Características
	• Indicadores de Gestión • Calculo del costo de Transporte
Opti-Time Tour Solver	• Capacidades de los vehículos (hasta 8 productos o dimensiones) • Capacidades de los vehículos (hasta 24 productos o dimensiones) • Tiempo de cargamento per unidad de cuantidad • Gestión de varios almacenes (para distintas ubicaciones de agencias o almacenes) • Restricción de circulación de los pesos pesados: altura, peso, vehículos largos o anchos, hauteur, poids, véhicule large ou long, transporte de materias peligrosas, etc. • Posibilidad de evitar las secciones donde hay que pagar, y de privilegiar los viales principales.
Logis Plan [22]	• Automatización de la planificación de rutas. • Optimización del uso de la flota • Cumplimiento de normativas y restricciones • Respuesta eficiente a cambios de última hora • Reducción de los costes asociados a la distribución • Mejora de la calidad de servicio
Servinformacion Ruteador logístico [23]	• Evalúa y optimiza el orden de visita a los clientes. • Reduce el tiempo de recorrido de los equipos logrando más tiempo para intervenciones o entregas. • Ahorro de tiempo en la planificación de rutas. • Mejora el reparto de unidades sobre el terreno garantizando una disminución en la distancia recorrida. • Configura todas las variables teniendo en cuenta la parametrización de cargas de vehículos como peso, kilos, toneladas, unidades, volumen y cajas. • El sistema genera la secuencia óptima de visita con base en la menor distancia o la ruta más rápida teniendo en cuenta el direccionamiento vial, restricciones y giros prohibidos. • Disminución de costos de: distribución, tiempo recorrido en visitas, número de camiones, tiempo de entrega, número de vendedores y pago de fletes, gracias a la elección de la mejor ruta. • Guía de distancia total y tiempo estimado de ruta.

Tabla 2: Algunas aplicaciones existentes

A pesar de existir diversas aplicaciones para seleccionar y planificar rutas la mayoría que se encuentran son dedicadas a transporte de personas. Además de esto, en las aplicaciones anteriormente nombradas se escogen las rutas con anticipación, es decir, no permiten crear un recorrido en tiempo real, aunque algunas de estas como Tour Solver trabajan con GPS para encontrar desvíos en casos de bloqueos en la ruta escogida.

2.2.2. METODOLOGIA

Para el desarrollo de este proyecto se escogió como metodología de desarrollo Extreme Programming (XP) debido a su tolerancia a cambios en los requerimientos del sistema y la flexibilidad en la documentación.

Teniendo en cuenta sus características, se procedió a realizar las entrevistas con el cliente para llenar las historias de usuario y hacer el levantamiento de los requerimientos del sistema. Estos procesos se dividieron en varias fases

- Análisis: En esta fase se reconocen los problemas a solucionar y profundiza en el funcionamiento de los procesos para definir los principales aspectos a mejorar definiéndolos en las historia s de usuario y los requerimientos del sistema.
- Diseño: En esta fase se realiza el modelado de la aplicación en base a los documentos conseguidos en la fase de análisis.
- Codificación: se codifica la aplicación con base al diseño.
- Pruebas: Se realizan pruebas para determinar los errores en la implementación y las diferencias con el diseño.
- Implementación: Se pone en marcha la aplicación y se valida con el cliente posibles cambios.

3. DESARROLLO DEL PROYECTO

3.1. ANALISIS

En la fase de análisis se hicieron entrevistas con el cliente con el fin de obtener información de las necesidades a suplir con el desarrollo de la aplicación, la información recolectada, se filtró y separó con el fin de sacar las partes importantes para el desarrollo del proyecto. En primer lugar se indagó las funciones que cumple cada empleado dentro de la empresa para determinar quién hace qué, y así definir los actores del sistema.

Actores	Rol
Administrador del sistema	Es la persona encargada de controlar y supervisar el funcionamiento del entorno y quien define los roles y permisos de los usuarios.
Supervisor	Persona encargada de verificar que las funciones se cumplan adecuadamente, puede realizar las tareas de los demás usuarios.
Despacho	Persona encargada de revisar facturas pendiente y pedidos y autorizar la salida de estos, además de crear los inventarios de los domicilios y registrar devoluciones y cambios
Domicilio	Persona encargada de llevar los pedidos al cliente, estos pueden revisar la información de las facturas, inventarios y rutas.

Tabla 3: *Actores y Roles*

Ya definidos los roles se procedió a realizar la recolección de requerimientos e historias de usuario para aclarar las funciones que deberá tener el sistema.

3.1.1. REQUERIMIENTOS

A través de entrevistas con el cliente se procedió a realizar el levantamiento de requerimientos del sistema, preguntando acerca de los distintos procesos que se realizan en la empresa y presenciando la forma como los empleados realizan cada actividad. Mediante estas reuniones se pudieron apreciar las necesidades y falencias actuales de la empresa y se logo definir las principales

necesidades. Estas se plantean en la especificación de requerimientos del sistema.

Información general			
Requerimiento No.	*REQ021*	*Fecha de creación*	*02/09/2014*
Autor	Carlos Andrés López	*Prioridad*	Alta
Descripción			
Usuarios	Despachador, Administrador, Supervisor.		
El sistema en el módulo de despacho deberá permitir al usuario registrar la información de los clientes.			
Entradas	Número cedula, Nombre, Teléfono, Celular, Dirección, Edificio, apartamento, Barrio.		
Precondición			
Referencias	Historias de Usuario – US021		

Tabla 4: *Requerimiento 21 registro de clientes*

Información general			
Requerimiento No.	*REQ022*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario buscar clientes.			
Entradas	Número Cédula.		
Precondición			
Referencias	Historias de Usuario – US022		

Tabla 5: *Requerimiento 22 búsqueda de clientes*

Información general			
Requerimiento No.	*REQ024*	*Fecha de creación*	*02/09/2014*
Autor	Carlos Andrés López	*Prioridad*	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario crear domicilios de los clientes.			
Entradas	Teléfono, Dirección, Edificio, Apartamento, Barrio.		
Precondición	*REQ022*		
Referencias	Historias de Usuario – US024		

Tabla 6: *Requerimiento 24 registro de domicilios del cliente*

Información general		***Pendiente***	
Requerimiento No.	*REQ026*	*Fecha de creación*	*02/09/2014*
Autor	Carlos Andrés López	*Prioridad*	Alta
Descripción			
Usuarios	Despachador, Administrador.		

El sistema en el módulo de despacho deberá permitir al usuario registrar las facturas de los clientes	
Entradas	Número, Fecha, Hora, Cliente, Dirección, Lista de productos y cantidades.
Precondición	
Referencias	Historias de Usuario – US026

Tabla 7: *Requerimiento 26, registro de facturas*

Información general			
Requerimiento No.	*REQ030*	*Fecha de creación*	*02/09/2014*
Autor	Carlos Andrés López	*Prioridad*	Alta
Descripción			
Usuarios	Administrador, Despachador.		
El sistema en el módulo de administración de rutas deberá permitir al usuario configurar las rutas a través de las mediciones de distancias obtenidas mediante el recorrido entre dos puntos geográficos.			
Entradas	Dirección de Partida, Dirección Llegada, Distancia de Ida, Distancia de Vuelta.		
Precondición	Las direcciones deben estar registradas en el sistema.		
Referencias	Historias de Usuario – US030		

Tabla 8: *Requerimiento 30 configuración de rutas.*

Información general			
Requerimiento No.	*REQ041*	*Fecha de creación*	*02/09/2014*
Autor	Carlos Andrés López	*Prioridad*	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario crear inventario de la ruta.			
Entradas	Lista de Envases y Cantidades		
Precondición			
Referencias	Historias de Usuario – US041		

Tabla 9: *Requerimiento 41 creación de inventarios*

3.1.2. HISTORIAS DE USUARIO

Estas contienen el comportamiento de diferentes eventos del programa en forma de la forma narrada por el cliente.

US021	*Prioridad: 5*	***Estimado (Horas): 2***
Nombre	Registrar Cliente	
Como	Administrador, Despachador, Supervisor.	
Quiero	Poder Registrar Clientes	
De forma que	Se tiene su información como referencia para los procesos asociados a estos.	
Referencia	*REQ021*	
Criterio de aceptación		
Escenario	Registro de cliente	
Dado	Que se desea registrar un cliente.	
Cuando	Hay un cliente nuevo.	

Entonces	Se ingresa el menú de clientes y selecciona la opción “Nuevo”, se despliega el formulario para registro de clientes, se llenan los datos requeridos, se guarda la información y se confirma la acción.

Tabla 10: *Historia de usuario registro de clientes*

US022	*Prioridad: 5*	***Estimado (Horas): 1***
Nombre	Buscar Cliente	
Como	Administrador	
Quiero	Poder Buscar Clientes	
De forma que	Se pueda ver la información referente a este.	
Referencia	*REQ022*	
Criterio de aceptación		
Escenario	Búsqueda de cliente	
Dado	Que se desea buscar un cliente.	
Cuando	Se ingresa al menú de clientes	
Entonces	Se ingresa el número de identificación del cliente en el campo de búsqueda y se da clic en la opción “Buscar”, el sistema muestra la información de este, si existe, de lo contrario muestra un mensaje informando que no se encontraron resultados.	

Tabla 11: *Historia de usuario búsqueda de clientes*

US024	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Agregar Domicilio de Cliente	
Como	Administrador	
Quiero	Poder agregar un domicilio a un cliente.	
De forma que	Se pueda tener los diferentes domicilios de estos.	
Referencia	*REQ024*	
Criterio de aceptación		
Escenario	Agregar domicilio de cliente	
Dado	Que se desea necesita crear un nuevo domicilio del cliente	
Cuando	El cliente realiza un pedido para un domicilio no registrado.	
Entonces	Después de buscar el cliente se da clic en la opción “Actualizar” y después en “Agregar Domicilio”, se llenan los campos requeridos, se guarda la información y se confirma la acción.	

Tabla 12: *Historia de usuario registro de domicilios del cliente*

US026	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registrar Factura	
Como	Administrador, Despachador, Supervisor.	
Quiero	Poder Registrar la información de las facturas de los clientes cliente.	
De forma que	Se pueda tener la información de estas para usarlas en los procesos de control de salida y envío de pedidos.	
Referencia	*REQ019*	
Criterio de aceptación		
Escenario	Registro de factura de cliente	
Dado	Que se desea registrar la factura de un cliente	
Cuando	Hace un pedido	
Entonces	Se ingresa en la opción facturas y se selecciona la opción “Nuevo” se ingresa el número de la factura, la fecha y la hora, en las opciones de cliente se abre	

	una ventana y se realiza la búsqueda, se selecciona el cliente y se agregan los productos de la factura, después se selecciona la opción guardar y se confirma la acción.

Tabla 13: *Historia de usuario registro de facturas*

US030	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Configurar Rutas	
Como	Administrador	
Quiero	Poder configurar las distancias de los recorridos entre dos estaciones.	
De forma que	Se pueda usar esta información en el cálculo de rutas.	
Referencia	*REQ030*	
Criterio de aceptación		
Escenario	Configuración de rutas	
Dado	Que se desea Configurar las distancias de las rutas.	
Cuando	Se crea un nuevo cliente o domicilio.	
Entonces	Se ingresa en la opción rutas, se selecciona la opción configura, se despliega una lista de direcciones. Se selecciona una dirección de la lista y se da clic en la opción continuar, después se abre una ventana donde se muestra una lista de las direcciones relacionadas, se selecciona la dirección de destino y se da clic en la opción actualizar, se llena el formulario y seda clic en la opción aceptar.	

Tabla 14: *Historia de usuario configuración de rutas*

US041	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Creación de Inventario de retorno	
Como	Administrador, Despachador	
Quiero	Poder crear inventarios de retorno	
De forma que	Se pueda tener control de los productos que deben ser devueltos por los domicilios.	
Referencia	*REQ041*	
Criterio de aceptación		
Escenario	Crear inventario de retorno.	
Dado	Que se requiere saber los que productos deben regresar	
Cuando	Se envían los pedidos de una ruta.	
Entonces	En la opción “Envíos” se selecciona una ruta de la lista, y se da clic en la opción inventario, se abre una venta con el formulario de creación de inventario, se seleccionan agregan los envases al inventario y se da clic en la opción “Guardar”.	

Tabla 15: *Historia de usuario creación de inventario*

3.2. DISEÑO

3.2.1. MODELO DEL SISTEMA

Para la creación del sistema se estudió la estructura de los procesos y datos implicados en el despacho y selección de rutas. Para esto, basado en los requerimientos y las historias de usuarios recolectadas, se definió la estructura del sistema y el modelo de datos a utilizar.

3.2.1.1. ARQUITECTURA

Parte importante del desarrollo de una aplicación es la selección del modelo arquitectónico a implementar acorde al tipo de aplicación. En este caso la aplicación se desarrolló en un entorno web, por lo tanto se utilizó el patrón arquitectónico de capas MVC (Modelo-Vista-Control).

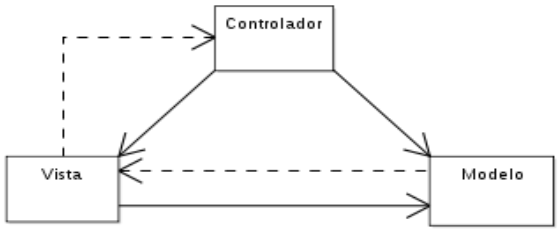


***Figura 4**: Diagrama MVC*

3.2.1.2. DISEÑO DE LA BASE DE DATOS

Para diseñar la base de datos, se tomó los requerimientos y se procedió a clasificar los datos para definir las entidades, atributos y relaciones entre estas.

En la Figura 5 se muestra la relación de los productos, la factura y los envases.

Se puede observar que la tabla *producto* está relacionada con la tabla *presentación*, esto se debe a la forma como se trabaja en la Distribuidora Santander, donde un producto tiene varias presentaciones estando todas estas bajo un mismo código, por esto el código del producto está incluido en la llave primaria de la tabla presentación. Para relacionar el producto y la presentación correspondiente con la factura se utiliza la tabla *presentacion_factura* donde se relacionan el producto y su presentación con la factura.

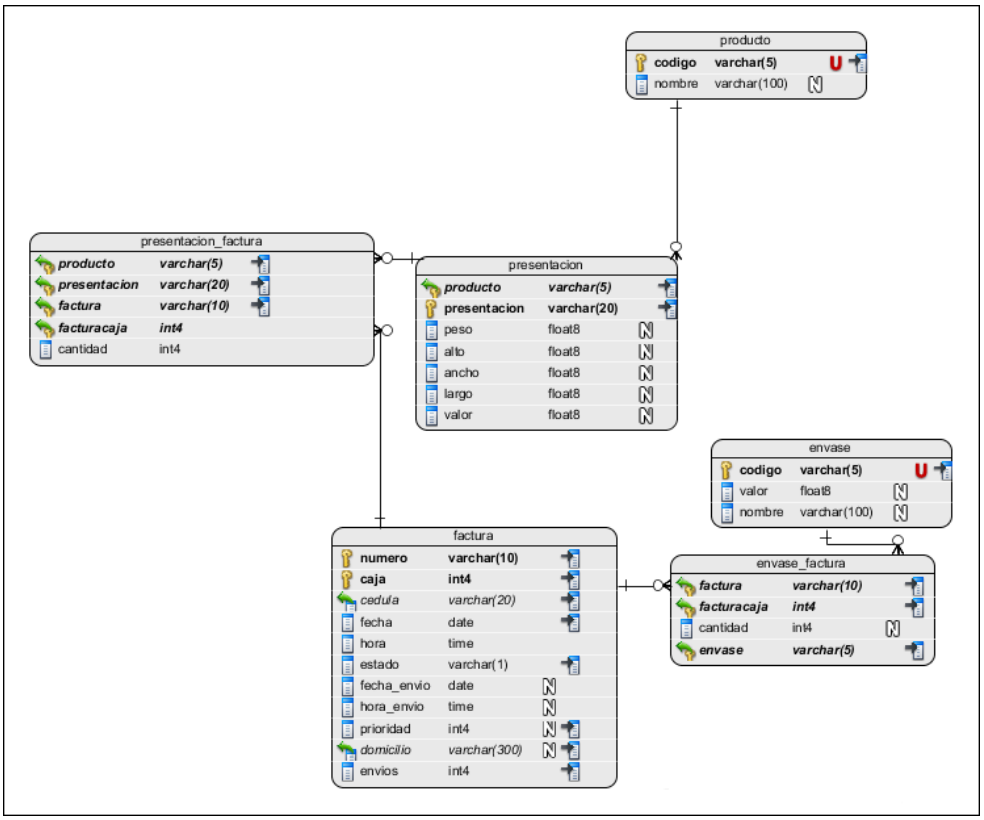


Figura 5: *Modelo de Base de datos, factura*

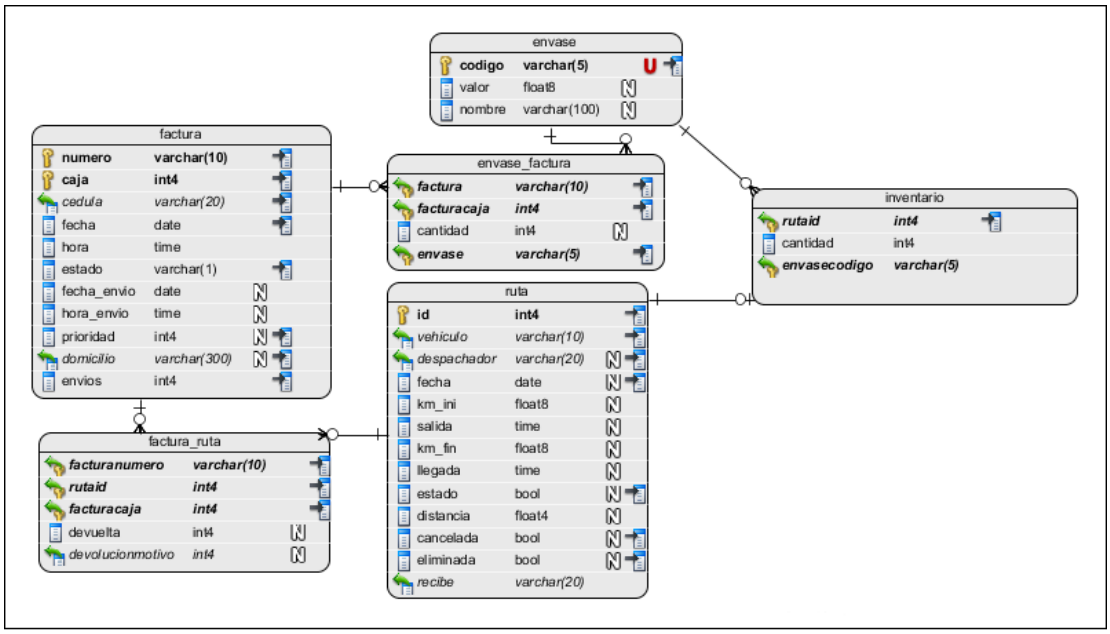


Figura 6: *Modelo de base de datos, rutas e inventario.*

En la figura 6 se muestra la sección de la relación entre las rutas las facturas y el inventario. En este caso el inventario se hace de forma general para cada ruta y no para cada factura, esto se debe que, dado el volumen de los envases en muchas ocasiones para maximizar la cantidad de producto en los remolques se completa las canastas con dos productos diferentes.

La tabla *factura_rutas* relaciona las facturas que se llevan en una ruta, con esto se da la opción que una factura pertenezca a más de una ruta. Una factura puede estar presente en varios recorridos cuando esta es devuelta y reenviada.

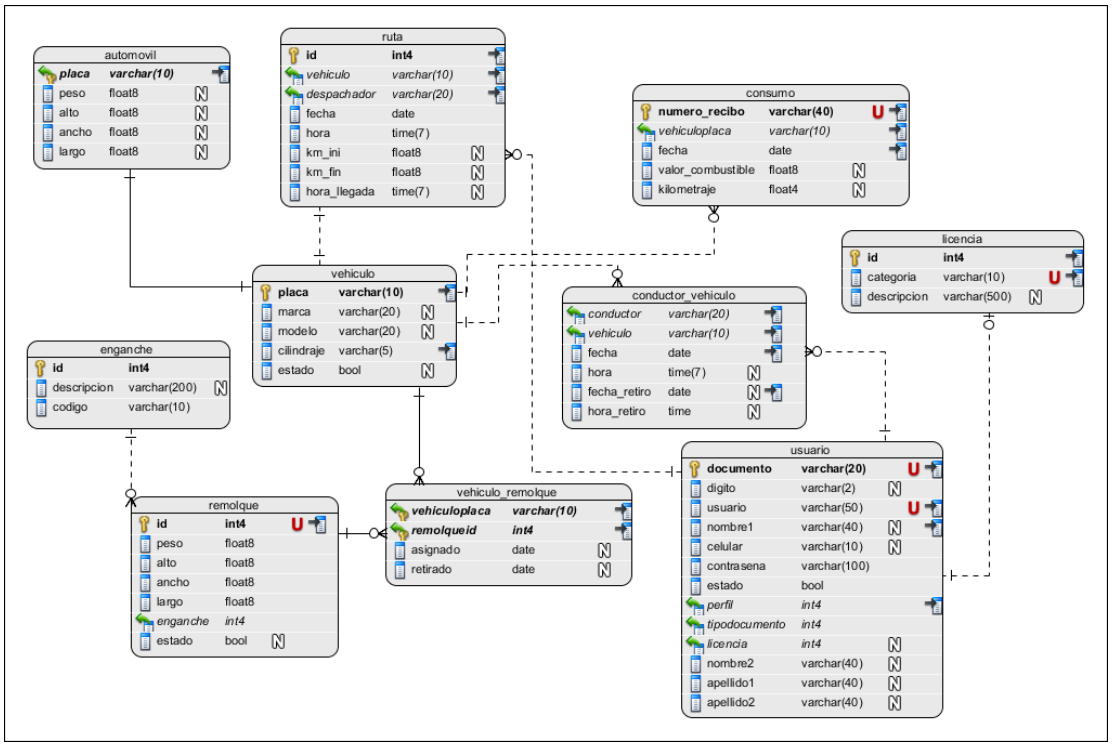


Figura 7: *Modelo de base de datos, rutas, vehículos, conductores*

En la Figura 7 se muestra la sección de la base de datos que guarda la información de las rutas, su relación con los vehículos, los remolques y los usuarios. La ruta tiene dos campos para control de la distancia recorrida por los vehículos, y el tiempo gastado en el recorrido. La tabla *usuarios* se relaciona con los vehículos y la ruta, la relación con los vehículos muestra los usuarios que reparten los pedidos y la relación con la ruta es para tener control de que usuario realiza el despacho.

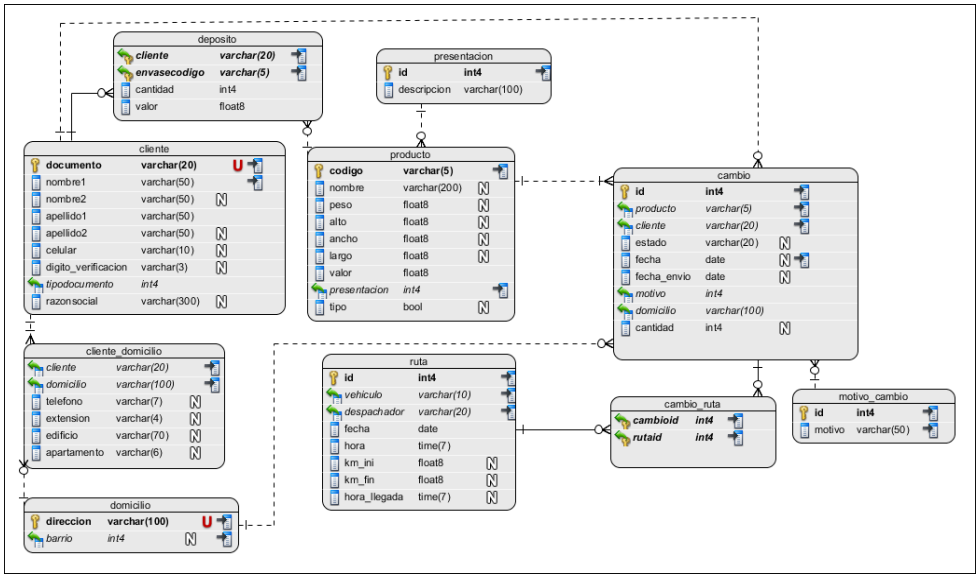


Figura 8*: Modelo de base de datos cambios y depósitos.*

En la Figura 8 se puede observar las relaciones entra los cambio y depósitos con los clientes, las facturas y las rutas. Los cambios tienen un motivo que permite reconocer porque se realiza estos están relacionados con la dirección del cliente para saber dónde se deben entregar. En el caso de los depósitos, están relacionados con el cliente y no con la dirección, esto se debe a que un cliente que tiene varios locales puede devolver el producto en depósito de cualquiera de sus negocios.

3.2.1.3. ESTRUCTURA DEL SISTEMA

En el diagrama de clases se plasma la estructura de la aplicación y la relación entre algunos objetos de la capa del modelo. Estos en este segmento del diagrama se puede ver la relación entre la factura, el cliente el domicilio y los productos.

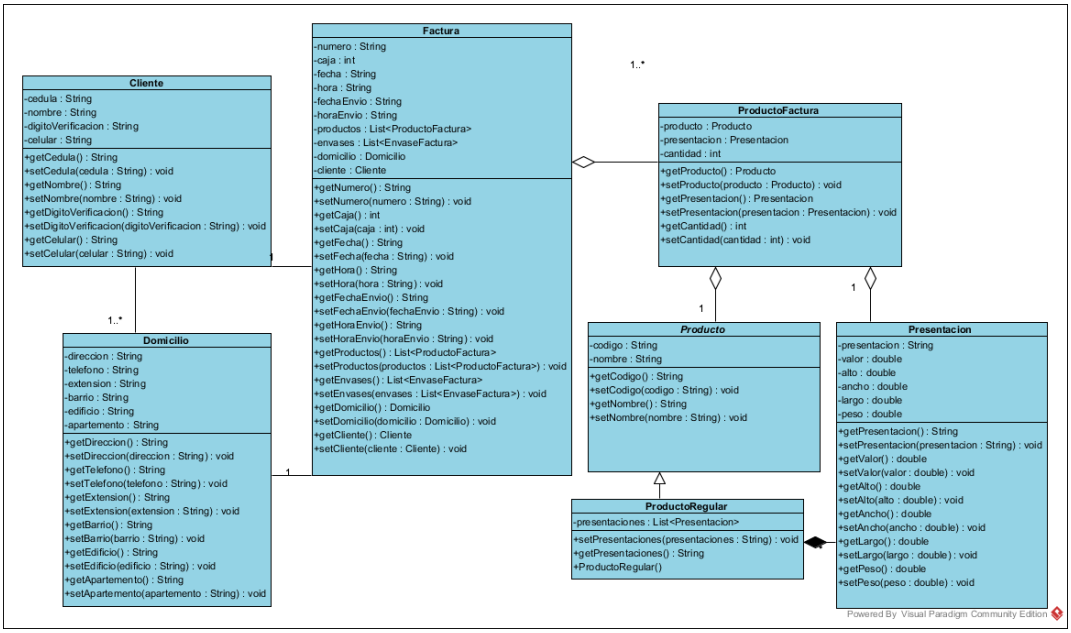


Figura 9 Diagrama de clases, relación facturas, cliente y productos

A diferencia de los demás objetos los objetos, para los nodos se serializa el objeto y se guarda en formato binario en disco en el archivo nodos.obj, esto con el fin de mantener el grafo y reducir el tiempo que cuesta leer de base de datos y crear el grafo completo, ya que al guardar un solo nodo se guarda toda la información de este. Lo correspondiente a las rutas se guarda en base de datos ya que es necesario tener una referencia de quien la recorrió y del inventario.

En la figura 10 se muestra parte de la estructura de la aplicación y como está relacionada con el MVC, las clases Producto, Presentación, ProductoRegular y ProductoFactura hacen parte del modelo, mientras que la clase ColeccionProductos es parte de la capa de control y las clases, CrearProducto, BuscarProducto, EditarProducto y EliminarProducto son parte de la vista.

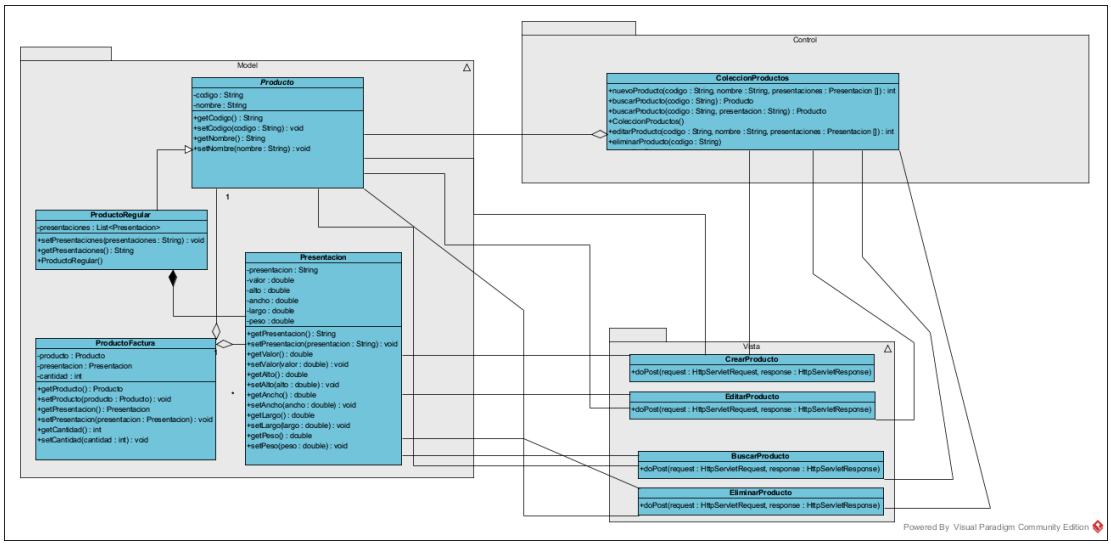


Figura 10: *Estructura de la aplicacio0n, módulo de productos*

3.2.2. DISEÑO DE LA SOLUCION AL PROBLEMA DE RUTAS

Anteriormente se nombraron algunos algoritmos y técnicas usadas para resolver el problema de rutas, estos toman una parte sencilla del problema y, con excepción del problema de ruteo de vehículos, no tienen en cuenta factores importantes como la capacidad de carga del medio0 de transporte, la cual es una importante restricción en el problema a resolver.

Partiendo de la capacidad de carga de los vehículos se deben tener en cuenta tres problemas:

- No se conoce la cantidad de pedidos.
- La capacidad de los vehículos no puede ser superada
- La antigüedad de los pedidos debe ser tenida en cuenta.

Al superarse la capacidad de los vehículos sigue siendo necesario tener en cuenta los pedidos que no caben, ya que puede ser posible obtener una mejor solución con estos, por otro lado, la antigüedad de los pedidos es algo de gran importancia, no se trata solo de hallar una ruta optima, sino también de mejorar el servicio de la empresa.

3.2.2.1. SELECCIÓN DEL ALGORITMO

Hay muchas técnicas que se pueden emplear para solucionar, en este caso se decidió dar una solución con algoritmos evolutivos, más precisamente algoritmos genéticos.

Otros factores que influyeron en la selección de los algoritmos genético son:

- Trabajan con una codificación de los parámetros y no con los parámetros mismos.

- Su técnica de búsqueda es global. Utiliza un subconjunto del espacio total, para obtener información sobre el universo de búsqueda, a través de las evaluaciones de la función a optimizar.

- No necesitan conocimiento previo sobre el problema a resolver ya que su función de aptitud trabaja con las restricciones.

- Utilizan operadores probabilísticos, en vez de los típicos operadores determinísticos de las técnicas tradicionales, haciendo que sean menos afectados por los máximos y mínimos locales en problemas de optimización, maximizar o minimizar.

3.2.2.2. ALGORITMO

Después de escoger los algoritmos genéticos, se procedió a analizar la forma de implementación y a reconocer las restricciones que limitan la solución del problema, la representación de la solución, los operadores y las condiciones que indican que se encontró una solución.

Antes de empezar a nombrar las estructuras y operaciones a utilizar se debe aclarar las condiciones que influyeron en la elección de la solución y el modelo de programación genética implementado.

3.2.2.2.1. CARACTERISTICAS DEL PROBLEMA

Este problema tiene dos restricciones, el peso y volumen máximo de los vehículos. Estas son importantes ya que la capacidad de carga no puede ser superada, teniendo así dos problemas, maximizar la carga y reducir la distancia recorrida. Para la maximización de la carga tenemos las siguientes restricciones:

$$Sea\ N = \{0, 1, \dots, n\}\ el\ conjunto\ de\ nodos$$

$$R = \{0, 1, 2 \dots, k\}\ el\ conjuntode\ vehiculo$$

$$\forall n_i \in N \; P(n_i) \;\; Peso\ del\ nodo\ i$$

$$\forall n_i \in N \;\; V(n_i) \;\; Volumen\ del\ nodo\ i$$

$$\forall r_k \in R \;\; P_{max}(r_t)\ Peso\ soportado\ del\ vehiculo\ k$$

$$\forall r_k \in R \;\; V_{max}(r_t)\ Volumen\ soportado\ del\ vehiculo\ k$$

$$\sum_{i=j}^{m} P(n_i) \leq \; P_{max}(r_t)\,, \qquad \sum_{i=j}^{m} V(n_i) \leq V_{max}(r_t)$$

$$Donde\ r_t\ es\ el\ vehiculo\ t$$

Las anteriores restricciones hacen variar la cantidad de nodos que puede visitar un vehículo dependiendo del tamaño y peso de los pedidos, para evitar problemas con las operaciones de cruce se implementó el modelo de islas.

3.2.2.2.2. ADAPTACION MODELO DE ISLAS

En el modelo de islas originalmente cada población en evaluada de forma independiente y al final se toman los mejores individuos de cada una para seleccionar el que más se acerque a la solución óptima. En este caso las poblaciones son demasiado pequeñas y realizar la unión de resultados no genera una diferencia suficientemente grande a realizar una evaluación global generando el riesgo de que en las soluciones tomadas salgan algunas muy alejadas de la solución ideal. Esto es riesgoso ya que podría ser seleccionada una mala solución.

El cruce y la mutación pueden tener como resultado individuos que no cumplen con la característica de la población de origen, estos individuos son migrados a otra población con característica diferente donde cumpla con las condiciones de la población.

También es posible que se generen individuos para los cuales no haya una población, en este caso es creada, lo que da lugar a que algunas tengan como máximo un único individuo. Esto causa problemas a la rora de realizar las operaciones, por lo tanto, para aumentar la población se realiza una clonación y se muta en una de sus partes para generar un nuevo miembro apto.

3.2.2.3. FUNCIÓN DE APTITUD

Para evaluar la aptitud de los individuos hay que tener en cuenta dos condiciones, la carga y la distancia a recorrer, el problema principal a resolver es conseguir la ruta más corta pero este por sí mismo no es suficiente, ya que esta ruta dependerá de la cantidad pedidos que lleve el vehículo, la cual debe ser máxima. De aquí surge el conjunto de restricciones.

3.2.2.3.1 MAXIMIZAR LA CARGA

La carga está dividida en dos restricciones, el peso y el volumen de los pedidos, que deben ser máximos sin sobrepasar la capacidad del vehículo. En las características del problema se definió las restricciones de los vehículos. Con estas dos condiciones se restringe la carga máxima que puede llevar cada vehículo y se pueden expresar de la siguiente forma:

$$\max\left(\frac{\sum_{i=j}^{m} P(n_i)}{P_{max}(r_t)} + \frac{\sum_{i=j}^{m} V(n_i)}{V_{max}(r_t)}\right) \leq 2$$

Cuando el resultado de esta operación, para el vehículo r_t, es igual a 2 la caga es máxima.

3.2.2.3.1. MINIMIZAR LA RUTA

La minimización de la ruta es un problema combinatoria ya que dependiendo del orden en que sean visitados los nodos variara la distancia a recorrer. Esto se puede definir de la siguiente forma utilizando la definición del algoritmo de Bellman-Ford:

Sea $D_{i,j}$ el camino más corto del nodo i al nodo j si este existe, o ∞ en otro caso.

$$\min\left(long(i,m) + D_{m,i-1}\right)$$

$$Para\ i,j = 1,2,3,\dots n$$

$$D_{i,j} = 0, si\ i = j$$

Habiendo definido las restricciones se puede expresar la función que evaluara el algoritmo genético. Para esto es necesario unir los dos problemas anteriormente nombrados, maximización de la carga y minimización de la ruta.

Ya que la minimización de la ruta depende del orden en que se visitan los nodos no se puede transformar a otra función, por lo tanto se hace necesario transformar la función de maximización de carga para que genere impacto en la minimización de la ruta. En este caso se tomó la siguiente función:

$$f(x) = \begin{cases} (x-1)^2 + 1 \; si \; x \; \in [0,1] \\ 2 \; en \; otro \; caso. \end{cases}$$

$$Donde \; x = \sum_{t=1}^{k} \frac{1}{2k} \left(\frac{\sum_{i=j}^{m} P(n_i)}{P_{max}(r_t)} + \frac{\sum_{i=j}^{m} V(n_i)}{V_{max}(r_t)} \right)$$

$$Domde \; k \; es \; la \; cantidad \; de \; vehiculos$$

$$i, j, m \; \in \; N$$

Con lo anterior podemos definir la función de aptitud de la siguiente forma:

$$\min: f(x) \sum_{i=j}^{m-1} D_{i,i+1}$$

3.2.2.4. REPRESENTACION DE LA SOLUCIÓN

Para el desarrollo del algoritmo es de gran importancia seleccionar una representación adecuada. Para empezar se debe tener en cuenta que no se opera únicamente sobre la información de las distancias, sino también sobre los volúmenes y pesos a llevar, por lo cual se debe busca una estructura que contenga este conjunto de datos en cada punto y que se adapte a la representación de la solución.

Hay distintas formas de representar los individuos de este problema, primero se debe tener en cuenta que es un problema combinatorio, por lo tanto no es posible una representación binaria. Esto nos deja que el cromosoma debe ser representado con los propios puntos y la solución será el orden de estos.

Para la representación del cromosoma se utilizara como base un cromosoma de dos partes [18], donde, generalmente, la primer parte tiene las ciudades a visitar, y la segunda parte representa los vehículos y tiene la información de cuantos nodos visitara el vehículo.

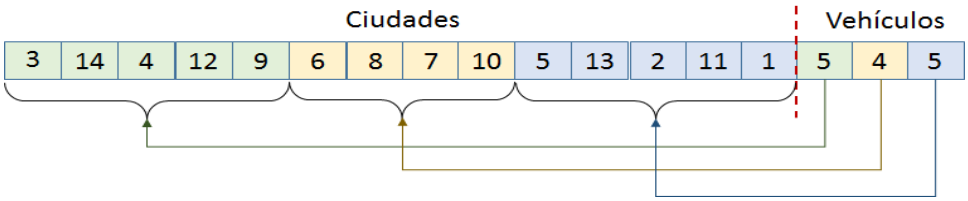


Figura 11: *Cromosoma de dos partes*

Para poder utilizar esta representación se realizaron algunos cambios para agregar las restricciones de los vehículos, ya que cada punto de la solución tiene información del peso y volumen, es necesario crear una estructura que permita tener la información de las distancias, el peso y volumen que corresponde a cada nodo, este peso y volumen está dado por los pedidos.

Todos los nodos deben estar relacionados entre sí formando un grafo que contiene las distancias entre estos, asegurándose que siempre estén unidos al nodo raíz, cuando una distancia es desconocida se le asigna el máximo valor posible, en java el máximo valor para un tipo de dato doublé es $1.7976931348623157\text{x}10^{308}$. La siguiente figura representa el grafo creado con la estructura nodo, el punto rojo es el nodo raíz.

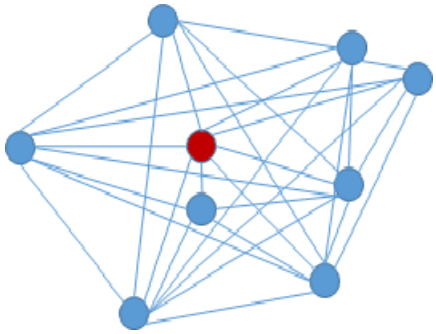


Figura 12: *Representación del grafo creado con la estructura nodo.*

La representacion de los individuos, el cromosoma, se basa en el cromosoma de dos partes, la primera es la lista de puntos.

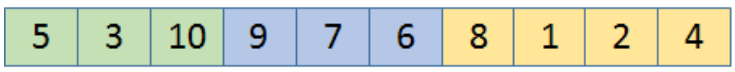


Figura 13*: Cromosoma de dos parte, representación adaptada parte 1.*

La segunda parte del cromosoma contiene la información de donde termina la carga, con esto se puede saber el número de nodos en cada uno y facilitar las distintas operaciones, esta segunda parte se representa con una lista donde cada posición hace referencia a un vehículo.

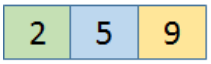


Figura 14: *Cromosoma de dos partes, representación adaptada parte 2.*

Se agregaron dos estructuras más, las cuales tiene información del peso y volumen que lleva cada vehículo con el fin de evaluar que la carga no sea sobrepasada. Estas son representadas igual que le segunda parte.

3.2.2.5. OPERACIONES

Como se había nombrado anteriormente los algoritmos genéticos utilizan dos operaciones básicas, el cruce y la mutación, y teniendo en cuenta la implementación del modelo de islas se agrega la migración.

Para el caso de problemas combinatorios el orden es de gran importancia ya que se trata de conseguir individuos con mejor aptitud en cada generación. Al aplicar el modelo de islas se separa los cromosomas de acuerdo a la cantidad de nodos que visitara cada vehículo, esto facilita la operación de cruce.

3.2.2.5.1. CRUCE

El cruce es la forma en que se generan nuevas soluciones potenciales mezclando la información de los individuos, teniendo en cuenta la característica del problema, se debe tratar de conservar al máximo las características de los padres. Este realiza tomando un vehículo del padre y copiándolo al hijo, y tratando de conservar el orden de la madre llenando el resto del cromosoma teniendo en cuenta que no se pueden repetir nodos.

En esta operación se presentan casos como, los hijos son individuos no válidos, es decir, no se encuentra un orden en el hijo que satisfaga las restricciones de peso y volumen por lo que estos se eliminan de las posibles soluciones evitando que lleguen al remplazo conservando así una población estable.

En el siguiente diagrama se expone el algoritmo de cruce:

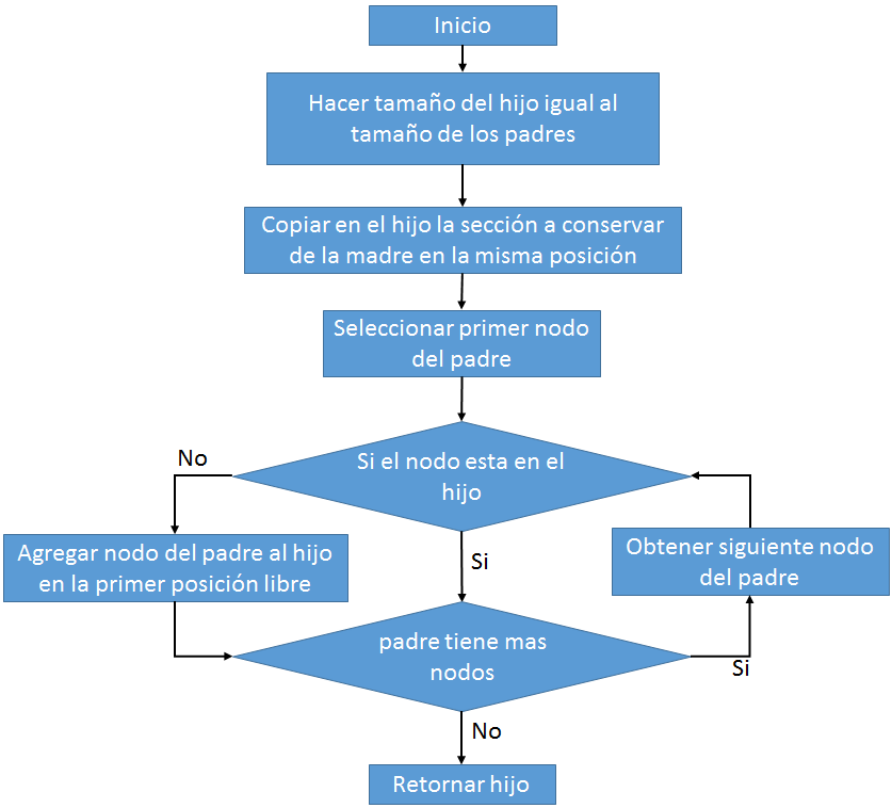


Figura 15 Diagrama de flujo, algoritmo de cruce

Figura 16: *Cruce de rutas*

En el cruce se debe tener en cuenta que en ocasiones las entradas pueden estar compuestas de un único vehículo, en esta caso el cruce

se hace seleccionando un segmento especifico del cromosoma de un 30% de la longitud de los individuos y se realiza el cruce intercambiando esta región.

3.2.2.5.2. MUTACIÓN

La mutación toma gran importancia en este problema, generalmente se aconseja tomar un porcentaje bajo de individuos para esta operación, un 2%, pero, para este problema en particular, si existe un numero alto de pedidos, habrán soluciones más difíciles de explorar, para esto se propuso un porcentaje de mutación del 10% teniendo en cuenta también que al dividirse en poblaciones un porcentaje más bajo podría causar que la mutación nunca ocurriera dejando así un importante espacio de soluciones sin alcanzar.

A pesar que hay distintos métodos de mutación, se decidió aplicar solo uno, el intercambio de nodos vecinos seleccionando una parte especifica del cromosoma, la correspondiente a un vehículo especifico, de esta forma se evita una alteración que pueda generar individuos no aptos y una caída en la población.

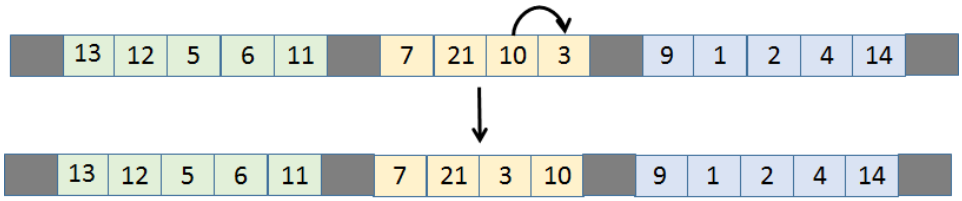


Figura 17: *Mutación por intercambio de adjuntos.*

En el intercambio de nodos vecinos se genera un número al azar para seleccionar un vehículo, después, nuevamente de forma aleatoria, se escoge un nodo del vehículo para intercambia con uno de los dos nodos que este en la posición siguiente o anterior.

3.2.2.6. POBLACION INICIAL

Para la población inicial se tuvo en cuenta la antigüedad de los pedidos, los más antiguos siempre deben estar en la solución, para esto los nodos entran ordenados por antigüedad de forma que los nodos más antiguos se agreguen a las soluciones y los demás nodos se agreguen de forma aleatoria a la solución.

La cantidad de individuos generados se estableció en máximo 500 individuos, ya que una cantidad muy alta de individuos puede afectar el rendimiento del algoritmo, se tomaron valores que permitan un rendimiento adecuado.

A medida que se crean los individuos se separan en poblaciones de acuerdo a la cantidad de nodos por vehículo, y después de generar todas las poblaciones se evalúan los individuos de cada una con la función fitness y se ordenan por puntuación.

En este proceso los vehículos se llenan de forma aleatoria asegurando que no se sobrepase la capacidad y teniendo como prioridad los primeros tres pedidos, para esto se tienen los siguientes algoritmos en la clase generador.

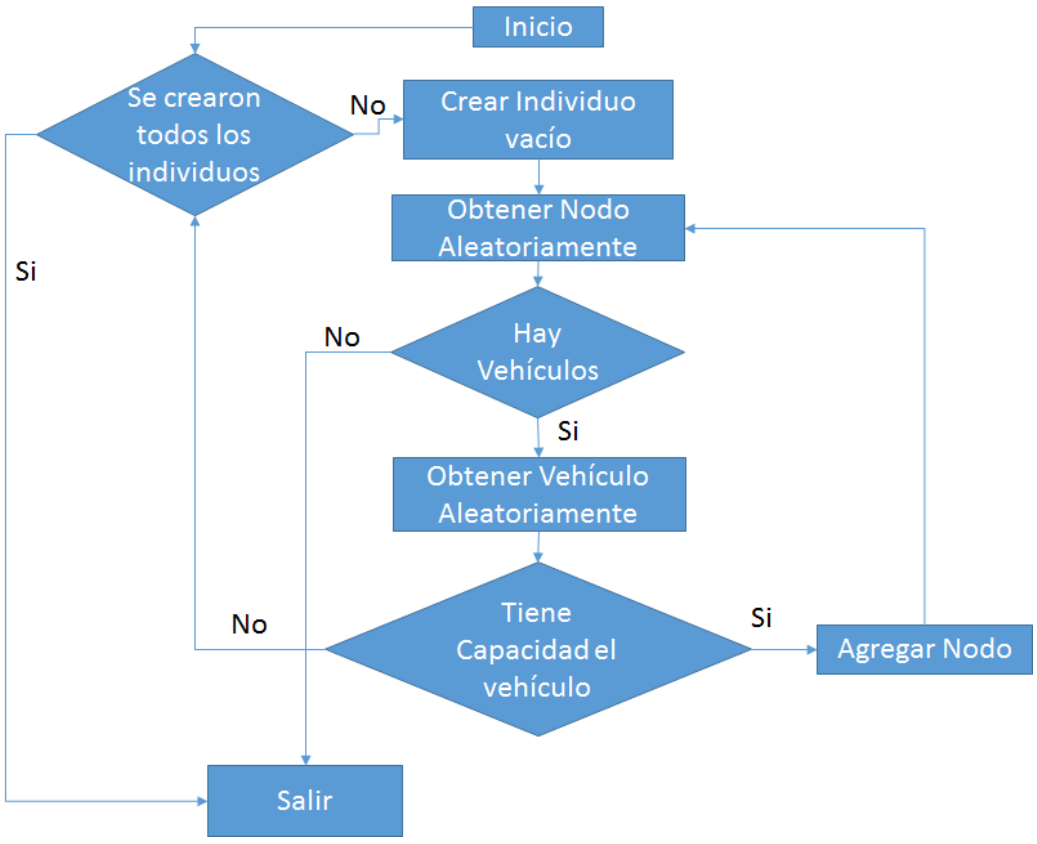


Figura 18 Diagrama de flujo, algoritmo de población inicial.

3.3. IMPLEMENTACIÓN

Para la implementación se diseñó el modelo de navegación, las vistas y se definió la arquitectura de la aplicación.

3.3.1. ARQUITECTURA

La arquitectura aplicada es un modelo por capas basado en el MVC con controlador frontal, la implementación en este modelo separa en su totalidad la vista de los demás componentes de la aplicación. La vista se comunica con un Servlet que sirve como controlador frontal y esta redirige al que procesa los datos para realizar la acción requerida por el usuario, buscando entre los datos cargados en consultas anteriores o en la base de datos si no se encuentran cargados. Después de esto se envían los datos a la vista como un objeto HTML.

Este modelo garantiza completa separación de las interfaces de usuario con la aplicación permitiendo así tener independencia entre distintos componentes de la aplicación.

Con el fin de tener mayor modularidad, la aplicación del controlador frontal se realizó se dividió por conjunto de funcionalidades, cada módulo tiene su controlador en vez de utilizar uno para todos estos.

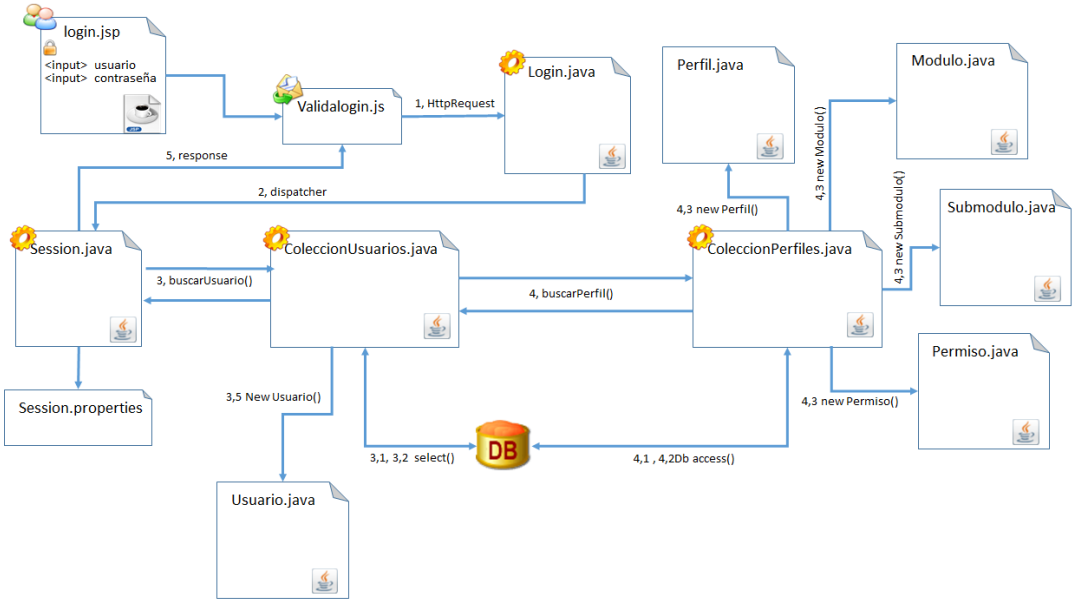


***Figura 19**: Módulo de navegación Login.*

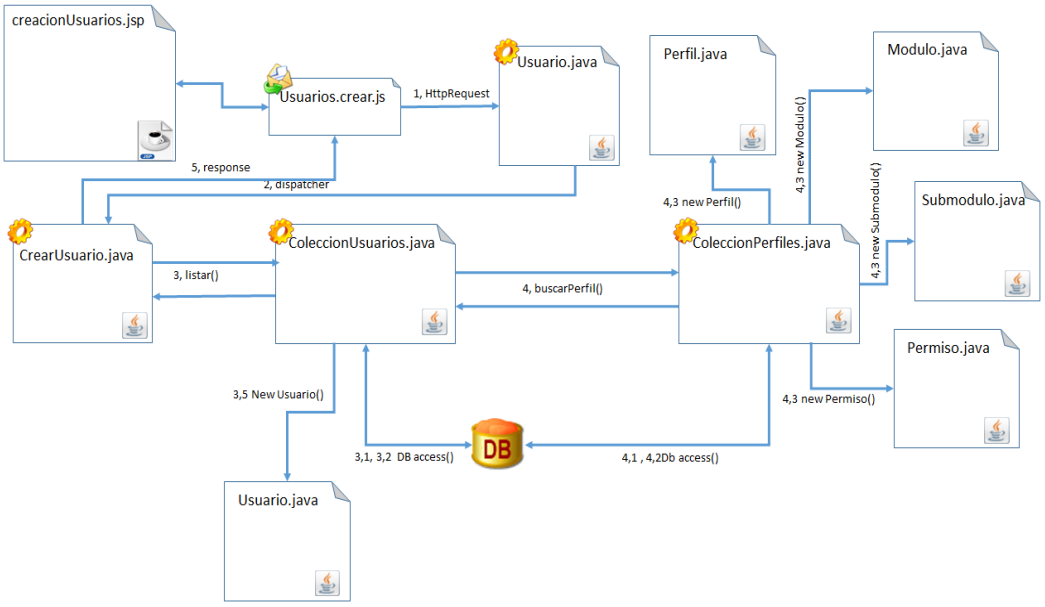


***Figura 20**: Modelo de navegación creación de usuarios.*

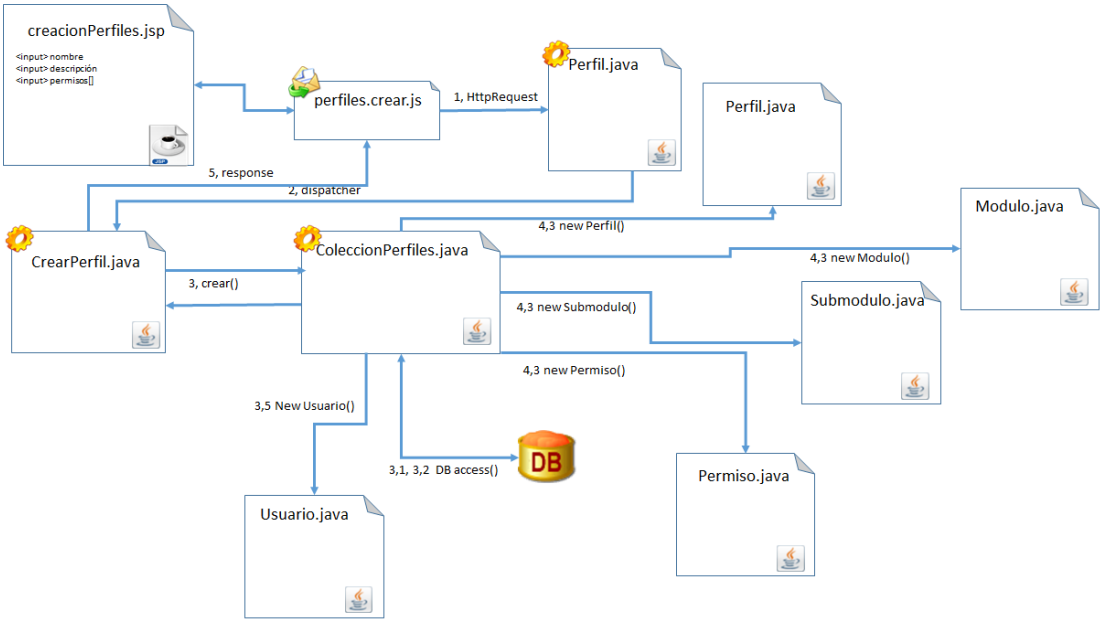


***Figura 21:** Modelo de navegación creación de perfiles*

3.3.2. DISEÑO DE MENU

Para acceder a los diferentes módulos se implementó un menú que habilita los enlaces de acuerdo al perfil del empleado y la pantalla en que se encuentre, es decir, el menú se crea dinámicamente. En la base de datos se almacena información correspondiente a los permisos de la aplicación, nombre y código, los cuales se cargan al inicio de sesión, después de estos se cargan los enlaces del archivo session.poperties.

3.3.3. DISEÑO DE PANTALLA

Los CRUD de la aplicación están divididos en tres partes, la pantalla principal donde aparece una lista con opción de búsqueda, la pantalla de creación y la de visualización y edición.

La pantalla principal tiene una tabla en la cual se muestran los resultados de la búsqueda, o en caso de no buscarse nada, lista los datos correspondientes al módulo, en este caso al ser el CRUD de usuarios muestra una lista de los usuarios existentes.

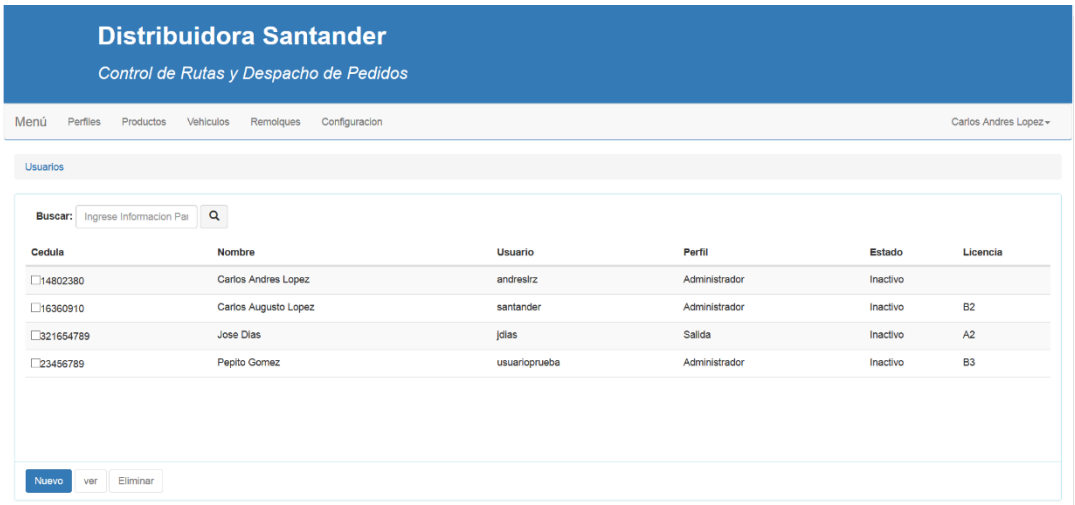


Figura 22: *Pantalla principal del CRUD usuarios*

Pantalla de creación de usuarios, esta pantalla contiene el formulario de creación de usuarios, este formulario tiene dos campos, Conductor y Licencia, al seleccionar el campo conductor se activa el campo licencia, estos campos son para diferenciar los conductores de los demás usuarios.

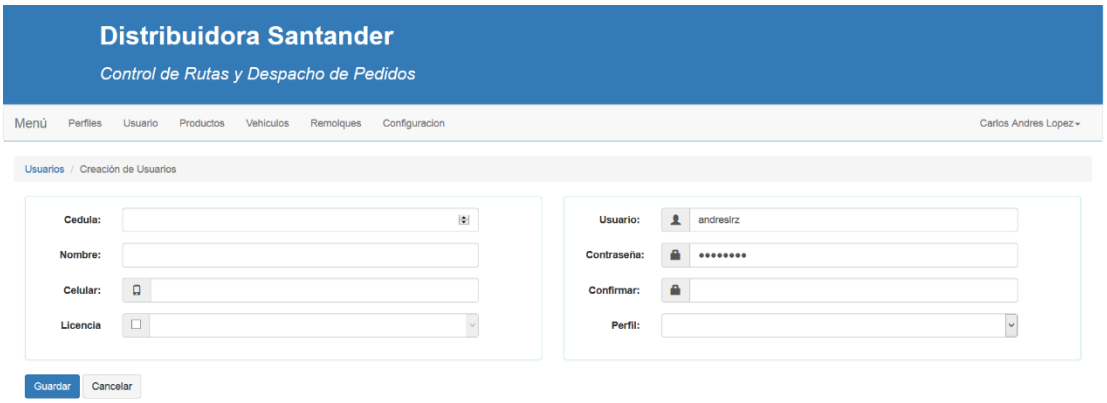


Figura 23: *Formulario de creación de usuarios.*

La pantalla de información del usuario muestra la información correspondiente y permite la actualización de datos si es necesario.

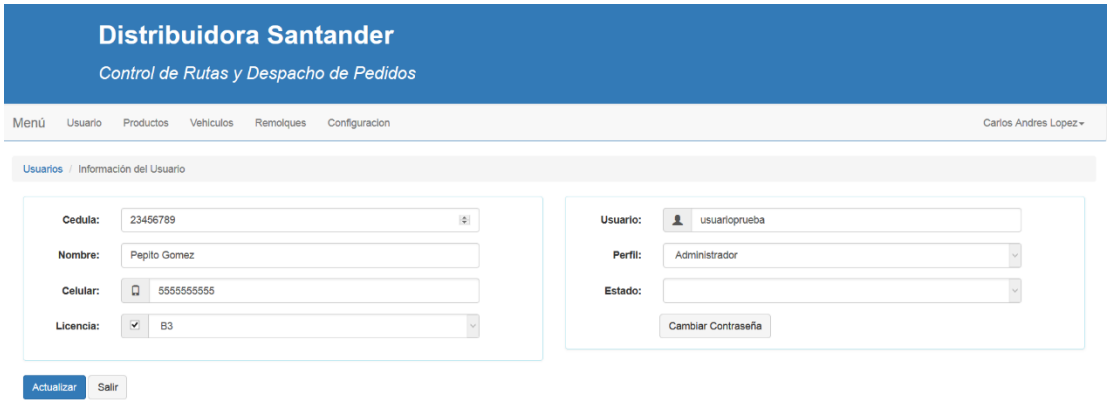


Figura 24: *Pantalla Visualización - Edición de Usuario*

La pantalla de creación de ruta manual se divide en dos partes, una es la selección del vehículo que recorrerá la ruta y la otra la selección de los pedidos que este llevara, al seleccionar los pedidos se confirma si hay cambios para el cliente de la factura, y se muestran para que el usuario seleccione cuales despachara.

En esta pantalla solo se muestran los vehículos activos, y permite asignarles un conductor en caso que no esté asignado.

***Figura 25**: Pantalla de creación de ruta parte 1.*

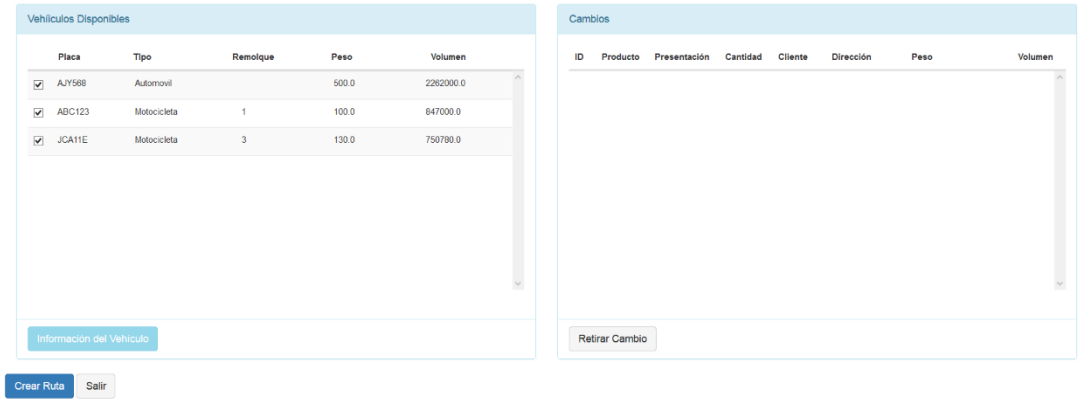


***Figura 26:** Pantalla de creación de ruta parte 2.*

La pantalla de configuración de rutas muestra la lista de direcciones existentes, al seleccionarse una carga la información de las distancias con las otras direcciones, también permite una vista previa de la ruta en un mapa entre la dirección de inicio y el destino que se seleccione.

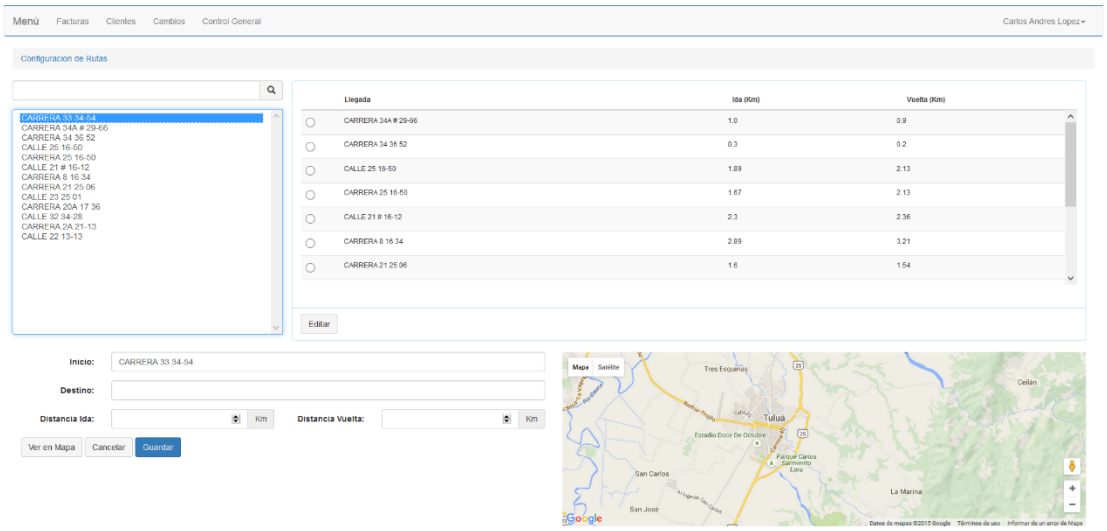


Figura 27: *Pantalla de configuración de rutas.*

4. PRUEBAS Y DISCUSIÓN DE LOS RESULTADOS

4.1. POBACION INICIAL.

En la generación de la población inicial se realiza de forma aleatoria validando que todos los individuos sean aptos, es decir, se verifica que no se pase la capacidad del vehículo.

En la siguiente tabla se tiene información de las facturas, pesos y volumen usados en las pruebas.

No.	Dirección	Peso(Kg)	Volumen(cm^3)
1	CARRERA 33 # 34-54		
2	CARRERA 34A # 29-66	84	14400
3	CALLE 25 # 16-50	7.5	3552.0
4	CARRERA 25 # 16-50	20.0	28960.0
5	CARRERA 34 # 36-52	9.0	12000.0
6	CALLE 21 # 16-12	21.2	79320.0
7	CARRERA 8 # 16-34	14.0	14400.0
8	CARRERA 21 # 25-06	3.5	14016.0
9	CALLE 3 # 16 – 25	14.5	16416.0
10	CARRERA 12 # 16 - 33	27.0	64800.0

Tabla 16: *Direcciones y pesos prueba 1*

En la siguiente matriz se muestran las distancias en kilómetros entre los distintos puntos, la matriz se toma interpreta horizontalmente.

Matriz de Distancias (Km)										
	1	2	3	4	5	6	7	8	9	10
1	0	25	14	36	78	2	33	15	12	9
2	7	0	6	15	31	22	16	4	34	18
3	18	5	0	36	14	8	9	12	21	23
4	1	7	21	0	10	3	25	31	10	2
5	1	7	21	10	0	3	25	31	10	2
6	1	17	20	12	31	0	9	16	17	4
7	11	9	20	13	23	19	0	30	16	5
8	21	17	20	13	31	9	2	0	16	12
9	14	17	19	12	13	20	27	8	0	4
10	12	14	20	15	21	20	18	32	6	0

Tabla 17: *Matriz de distancias prueba 1*

En la siguiente tabla se muestran los vehículos sobre los cuales se calculó la ruta, con las respectivas capacidades de cada uno.

No.	Peso (Kg) Máximo	Volumen (cm^3) Máximo
1	700	5440000.0
2	150	619200.0
3	140	735300.0
4	125	842800.0

Tabla 18: *Vehículos y medidas*

En la siguiente tabla se muestra las distribuciones y cantidad de individuos de las poblaciones generadas, en esta se puede apreciar que los datos son distribuidos de manera uniforme. La cantidad de individuos, como se nombró anteriormente, se genera teniendo en cuenta un porcentaje de las combinaciones entre nodos.

4.2. GENERACIÓN DE RUTAS

Para la generación de rutas se toman la población inicial, los nodos, y los vehículos para realizar el cálculo, teniendo como condiciones de parada un numero de terminado de iteraciones para la cantidad de poblaciones generadas y un límite de crecimiento de la media del conjunto de poblaciones.

En el siguiente grafico se muestra el comportamiento de la media y el puntaje del mejor individuo para cada generación. En este se logra apreciar un crecimiento constante de la media y los cambios en el mejor individuo.

A continuación se exponen algunos datos técnicos de la maquita en que se ejecutaron las pruebas, estos datos pueden ser de importancia debido a que este tipo de algoritmos tienen una ejecución lenta debido al número de iteraciones y al tamaño de la población inicial.

Información técnica	
Procesador:	• Frecuencia: 4.1 GHz • Núcleos: 4 • Sub Procesos: 8
Memoria RAM	• 16 GB • Frecuencia: 2266 MHz

En la siguiente tabla se exponen algunos datos usados en las pruebas, en la columna ***Vehículos*** se hace referencia a la ***Tabla 18***.

No. Individuos	*Iteraciones*	*Distancia(Km)*	*Tiempo(ms)*	*Vehículos*
200	200	133	1867	1,2,3,4

Tabla 19 Datos iniciales y finales prueba 1

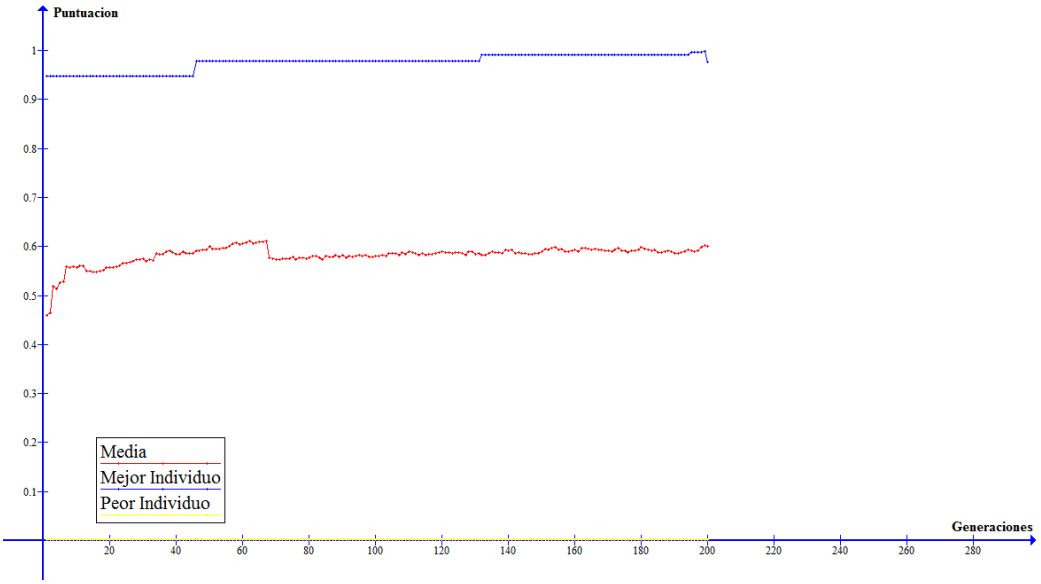


***Figura 28**: Grafico 1, Media peor individuo y mejor individuo con 200 individuos iniciales*

La media en este grafico muestra como la población tiende a converger, las variaciones en la media son causadas por las operaciones que pueden generar individuos con una aptitud baja conseguidos en el cruce o la mutación.

No. Individuos	Iteraciones	Distancia(Km)	Tiempo(ms)	Vehículos
100	200	110	1360	1,2,3,4

Tabla 20 Datos iniciales y finales prueba 2

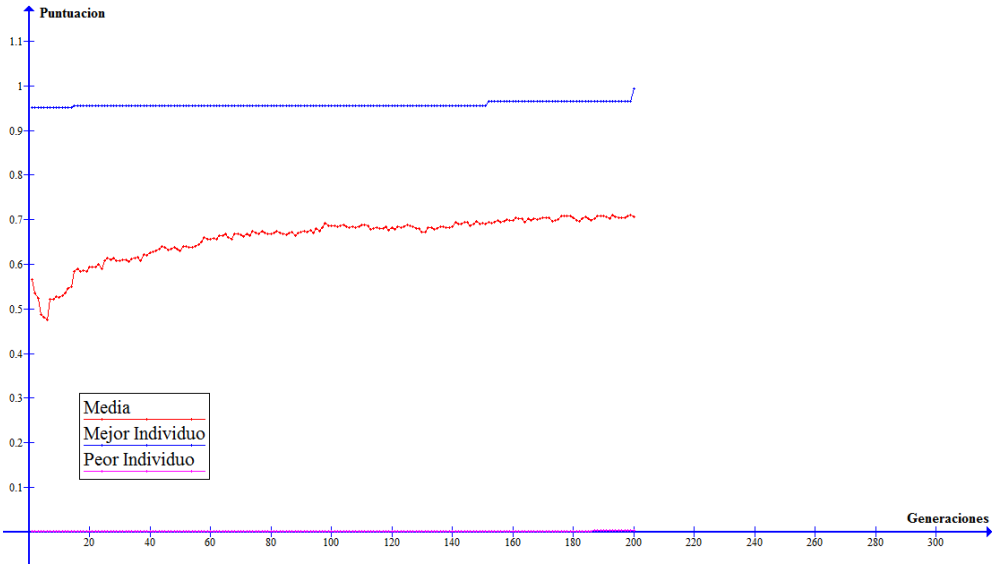


Figura 29 Grafico 2, Media peor individuo y mejor individuo con 100 individuos iniciales

No. Individuos	Iteraciones	Distancia(Km)	Tiempo(ms)	Vehículos
100	200	138	813	1,2,4

Tabla 21 Datos iniciales y finales prueba 1

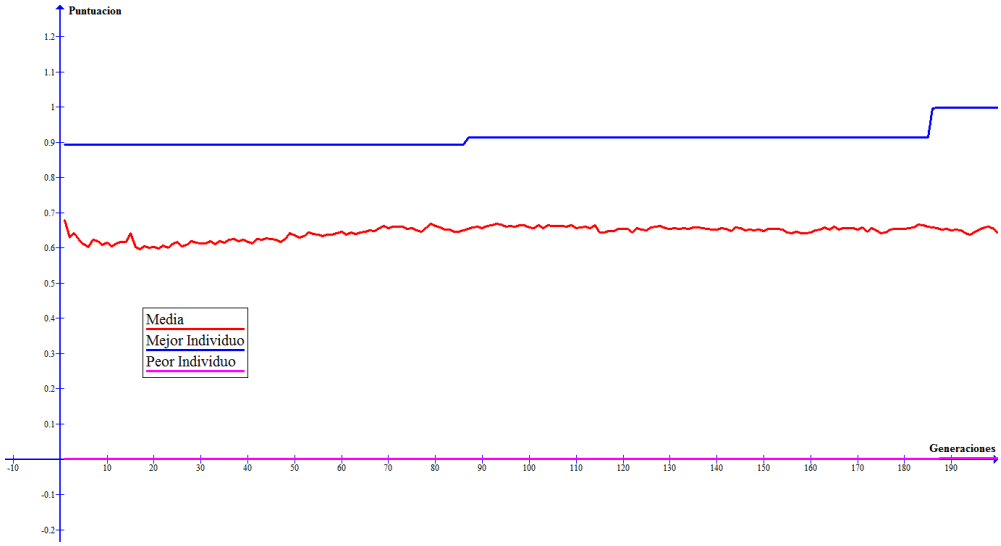


Figura 30 Media peor individuo y mejor individuo con 100 individuos iniciales y 3 vehículos

En los gráficos anteriores se puede observar que el mejor individuo esta siempre cercano a 1, esto es debido a la población inicial, al ser un problema combinatorio existe una alta probabilidad de que el mejor individuo se cree en la población inicial para problemas pequeños, ya que los mejores individuos de la población se conservan, la gráfica del mejor individuo de cada generación se mantiene constante.

Cada grafico está acompañado de una tabla en la que se muestran la cantidad de individuos, generaciones, el resultado de distancia para el mejor individuo, el tiempo que tardo en ejecutarse el algoritmo y la referencia de los vehículos seleccionados. En las tablas se puede observar las diferencias en tiempo con las diferentes poblaciones y diferente cantidad de vehículos. Además comparando los gráficos se hace evidente los cambios en la media para cada solución, en poblaciones más grandes el algoritmo converge más rápidamente.

Para comparar la solución se parte de un estado inicial, el orden de llegada de los pedidos, como una solución inicial al problema. Este punto de partida se toma como referencia para para validar el resultado del algoritmo.

4.3. PROBLEMAS ENCONTRADOS.

En las pruebas realizadas al algoritmo se ingresaron pedidos grandes parea determinar el comportamiento de esta. Estos pedidos sobrepasaban el tamaño de los vehículos, razón por la cual no entraban dentro de las soluciones. Esto se vuelve un problema ya que en repetidas ocasiones se hacen pedidos de gran cantidad de producto los cuales aún en el carro se debe realizar 2 o más viajes para poder entregarlos.

Otro problema que se encuentra es que en ocasiones donde los pedidos ocupan gran parte de la capacidad de los vehículos la mejor solución tiende a volverse mala bajando su aptitud en cada iteración.

4.4. PRUEBAS UNITARIAS

En las pruebas unitarias se comprobó el funcionamiento adecuado de las diferentes funciones de la aplicación, estas se realizaron utilizando el Framework para pruebas JUnit.

En estas se ingresa información que se espera pase a cada función con el fin de comprobar que las validaciones y el manejo de errores se realiza de forma correcta y cumpla con las funciones para las que fueron programados, junto con las pruebas se adjunta la evidencia que corrobora que estas se realizaron adecuadamente y que se obtuvieron los resultados esperados o en su defecto se documenta el error y la planificación para su corrección.

Se realizaron un total de 74 pruebas de las cuales 16 fueron fallidas, siendo los casos más comunes de fallo las referencias nulas.

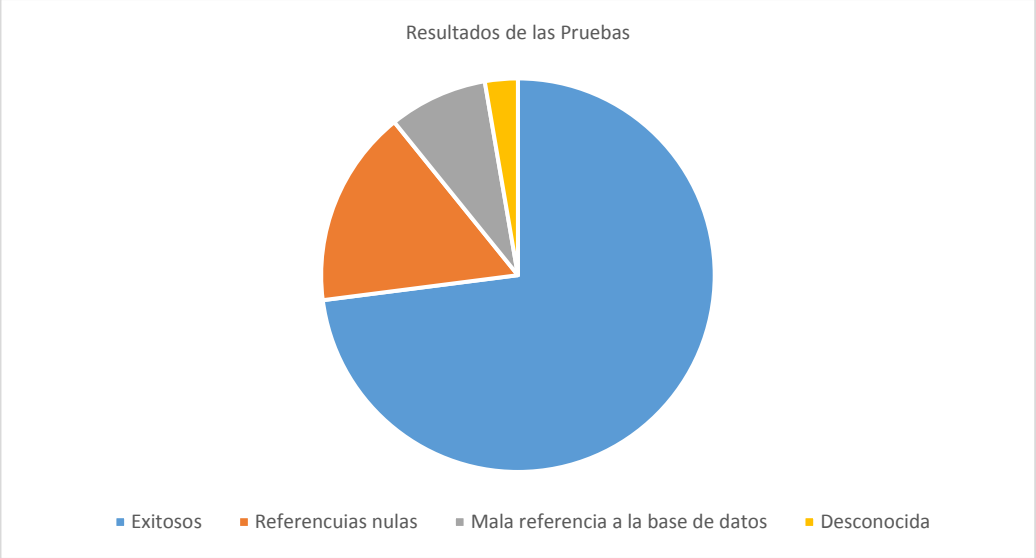


Figura 31Grafico estado porcentajes de pruebas exitosas y fallidas

Estado de Prueba	Cantidad	Porcentaje
Pruebas exitosas	54	72.97%
Referencias nulas (Null Pointer)	12	16.216%
Mala referencia a la base de datos	6	8.108%
Error Desconocida	2	2.702%

Los errores desconocidos son errores de los cuales no se muestra información que ayude a solucionarlos, estos errores se presentaron principalmente en las listas donde al intentar tratar el error dejaba de aparecer. Las malas referencias a base de datos fueron causadas por cambios incompletos.

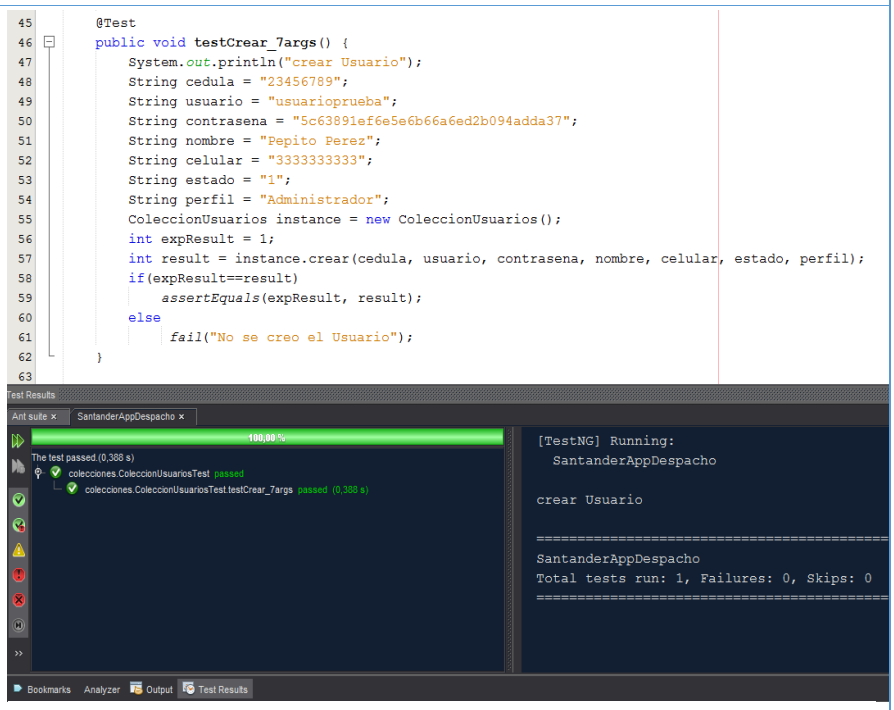
Prueba Unitaria				
Prueba No:	PU0001			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un usuario			
Clase:	Usuarios.java			
Función:	public synchronized int crear			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "23456789"; ➢ String usuario = "usuarioprueba"; ➢ String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37"; ➢ String nombre = "Pepito Perez"; ➢ String celular = "3333333333"; ➢ String estado = "1"; ➢ String perfil = "Administrador";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

***Tabla 22**: Prueba unitaria, creación de usuario*

En esta prueba se verifica el método de creación de usuario, se ingresan los datos del usuario y se espera que retorne un número entero, 1 si se creó correctamente 0 en otro caso.

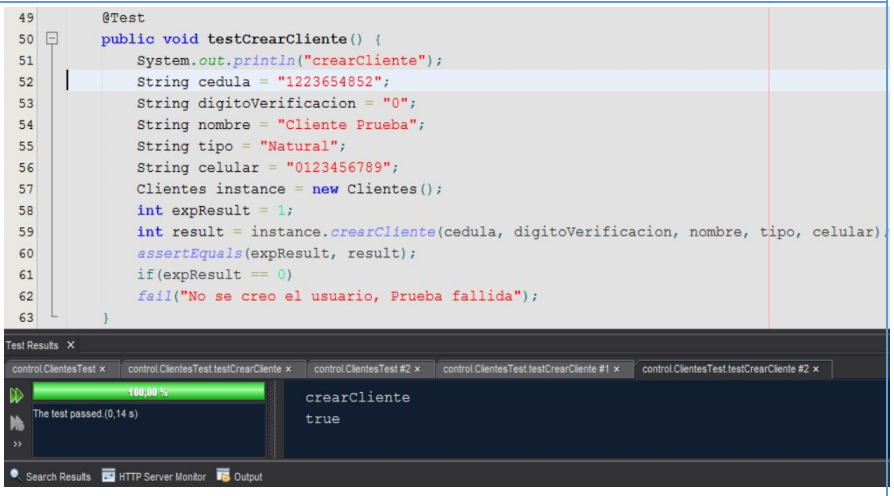
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0022</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Registrar un cliente</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionClientes.java</td></tr>
<tr><td>Función:</td><td colspan="4">public int crearCliente</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String cedula = "1223654852";
➢ String digitoVerificacion = "0";
➢ String nombre = "Cliente Prueba";
➢ String tipo = "Natural";
➢ String celular = "0123456789";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">1</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">1</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td>Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td>Evidencia:</td><td colspan="4"> 
</td></tr>
</table>

***Tabla 23**: Prueba unitaria, registro de cliente.*

En esta prueba se verifica el ingreso de clientes, al igual que en el ingreso de usuarios se espera como resultado 1 si el cliente se ingresa correctamente o 0 en otro caso.

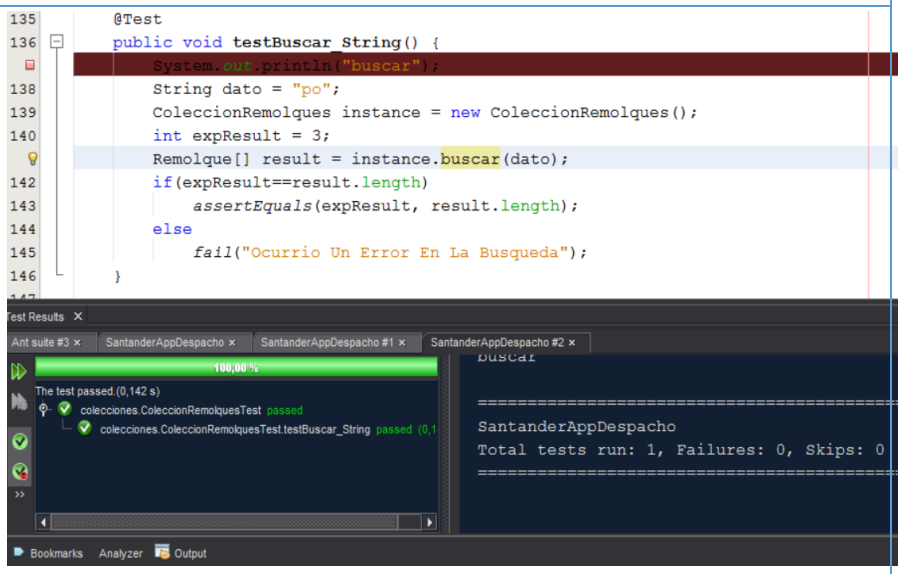
Prueba Unitaria				
Prueba No:	PU0015			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Bscar un remolque por un dato parcial			
Clase:	ColeccionRemolques.java			
Función:	public Remolque[] buscar			
Configuraciones :				
Datos de Entrada:	➢ tring dato = "po";			
Salida esperada:	Una lista de objetos Remolque			
Salida obtenida:	Una lista de objetos Remolque			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Tabla 24: *Prueba unitaria, Listado de remolques*

Esta prueba se trata de buscar un remolque por un dato parcial, como resultado debe retornar una lista con los remolques que tengan dato que coincidan o una lista vacía si no se encuentran coincidencias. Ya que no se conoce todos los objetos que se retornaran la condición para la prueba es recibir una lista de objetos de un tamaño especifico.

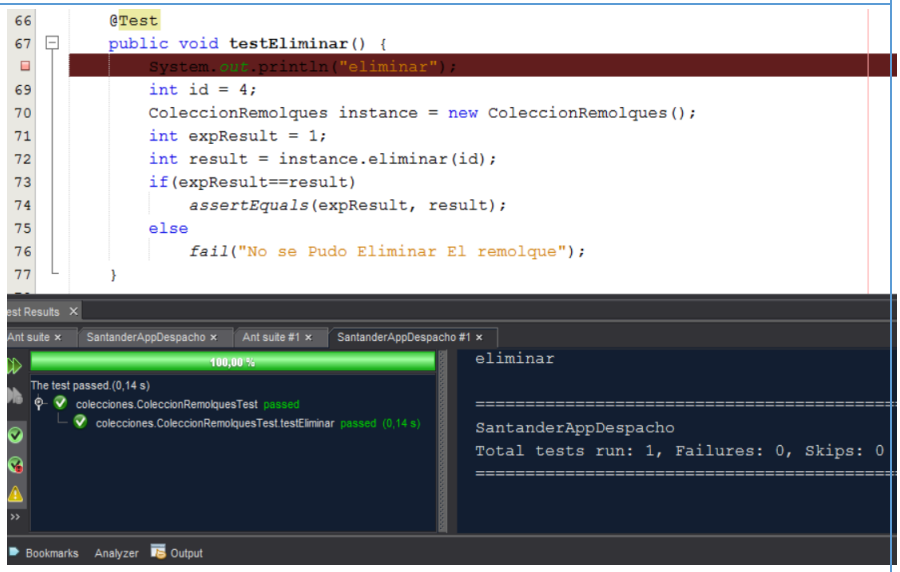
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0021</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Eliminar un remolque</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionRemolques.java</td></tr>
<tr><td>Función:</td><td colspan="4">public int eliminar</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ int id = 4;</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">1</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">1</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td rowspan="2">Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Evidencia:</td><td colspan="4"> 
</td></tr>
</table>

Tabla 25: *Prueba unitaria, eliminar remolque*

4.5. PRUEBAS DE ADAPTACIÓN Y ACEPTACION.

Debido a las necesidades actuales de la empresa y aprovechando las nuevas tecnologías, se decidió que la aplicación debe ser capaz de funcionar en diferentes dispositivos móviles. Por esta razón es necesario mostrar la capacidad de la aplicación para adaptarse a las resoluciones década posible dispositivo.

Para facilitar esto se utilizó la librería Bootstraps, la cual facilita el diseño y permite que este cambie las interfaces adaptándose al dispositivo en uso. Para realizar las pruebas se utilizó el plugin NetBeans Connector en el navegador Google Chrome, el cual permite emular las diferentes resoluciones de los dispositivos

Para validar con el cliente se encuesto sobre las diferentes pantallas preguntando si estas son fáciles de comprender, que aspectos se pueden mejorar, que opciones le generan dudas o que le campos cree que le hacen falta.

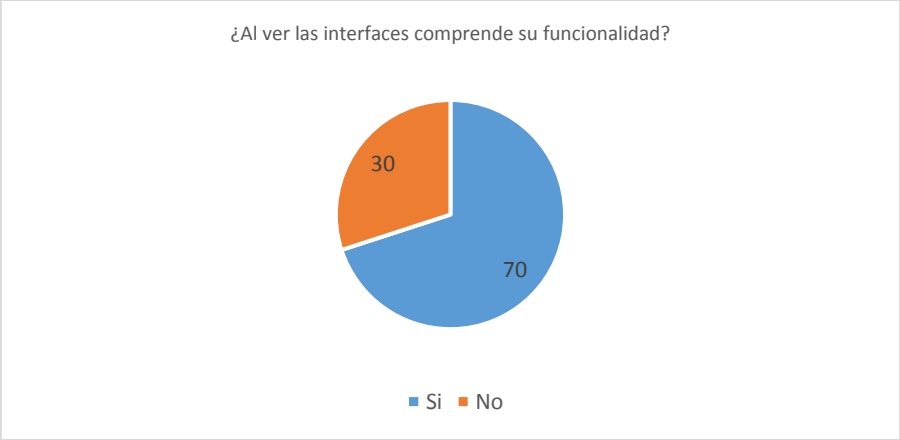


Figura 32 Grafico comprensión de las interfaces

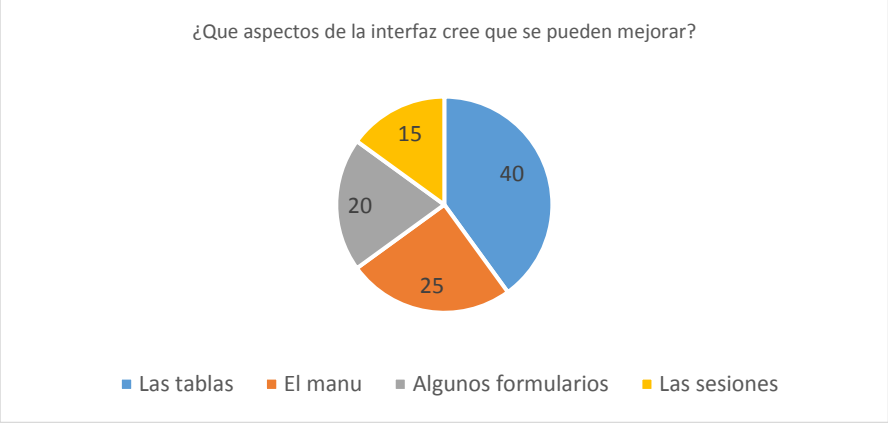


Figura 33 Grafico aspectos a mejorar de la interfaz

- Resolución 1024 x 768

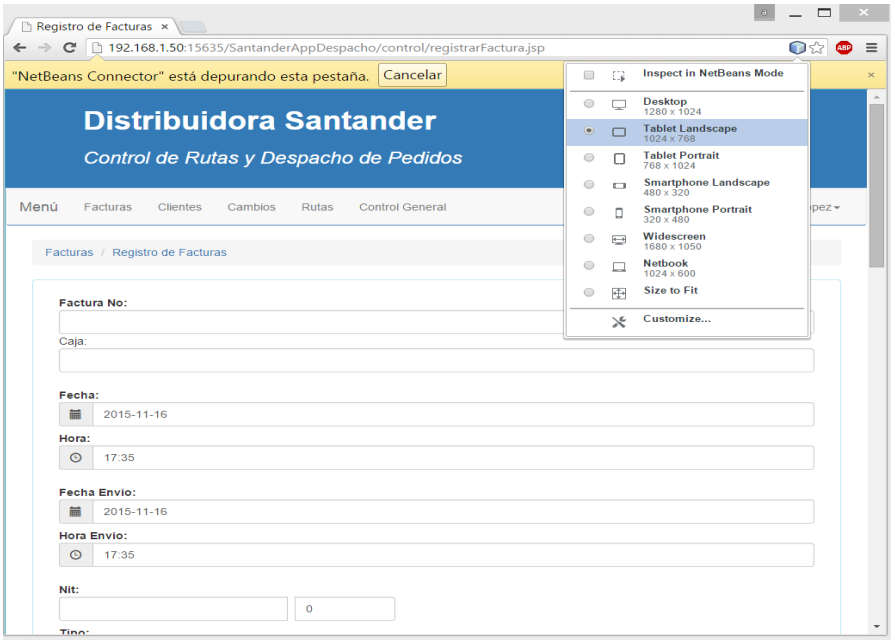


Figura 34: *adaptación del formulario*

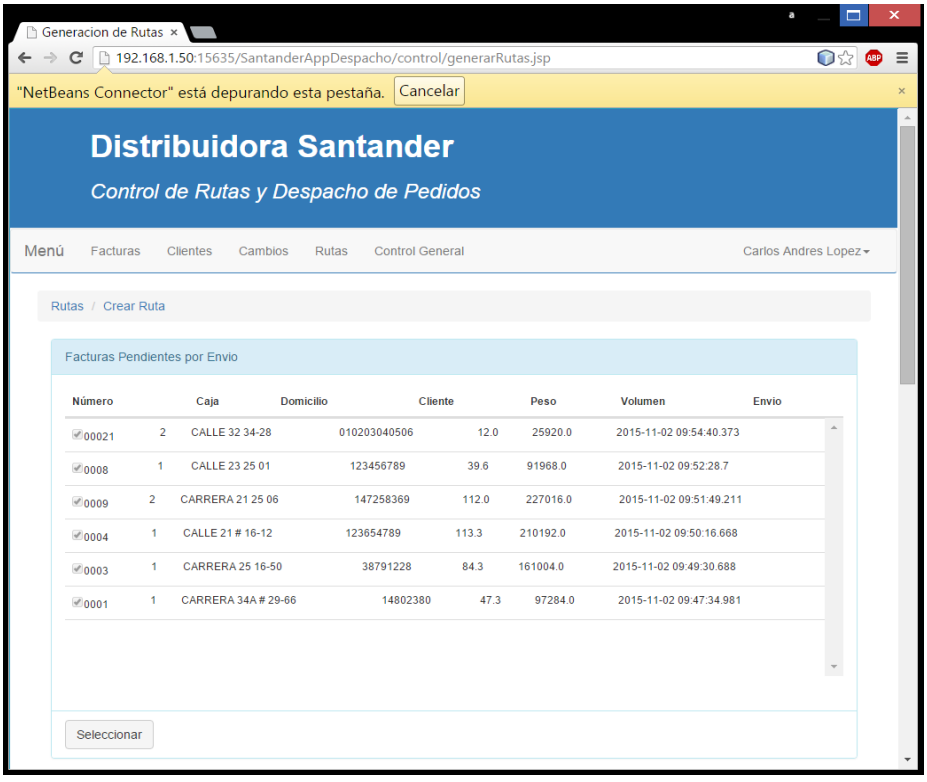


Figura 35 Adaptación de las tablas.

- **Resolución 480 x 320**

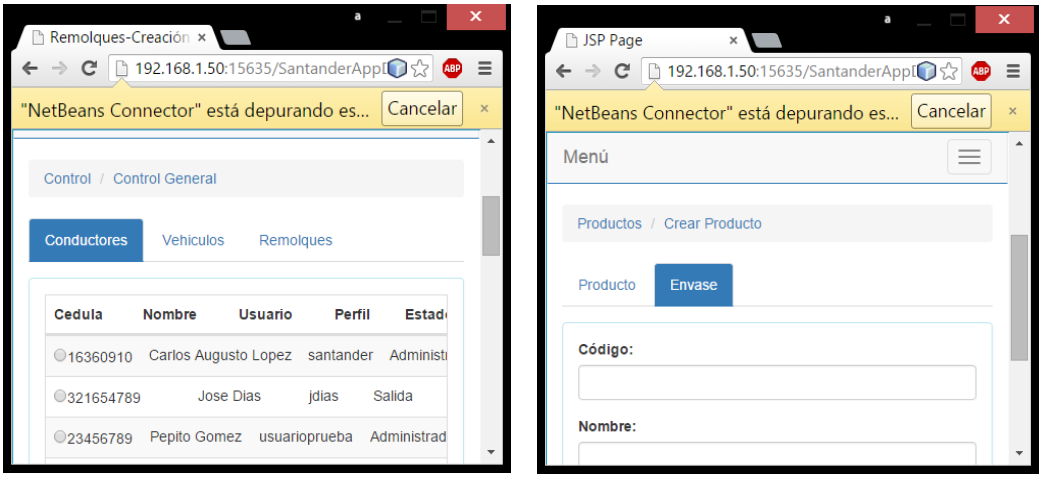


Figura 36: *Prueba de adaptación en dispositivos móviles pequeños.*

- **Resolución 320 x 480**

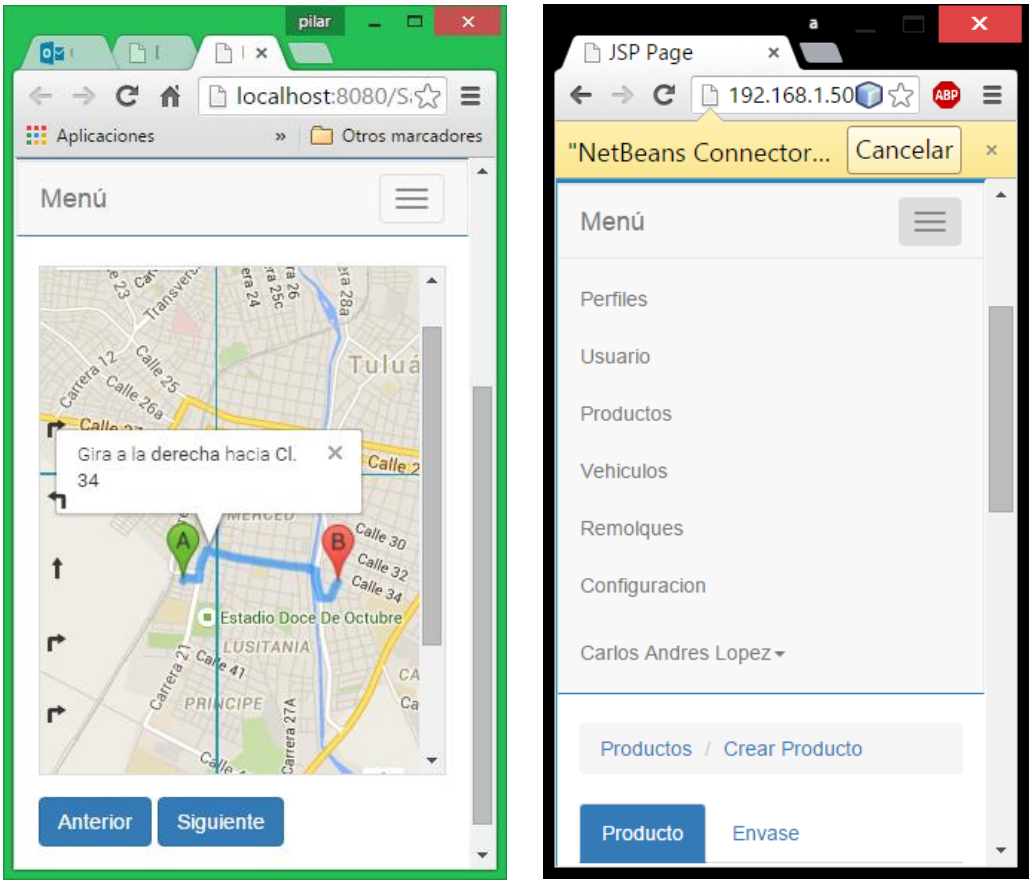


Figura 37: *Prueba de adaptación de los mapas y menú*

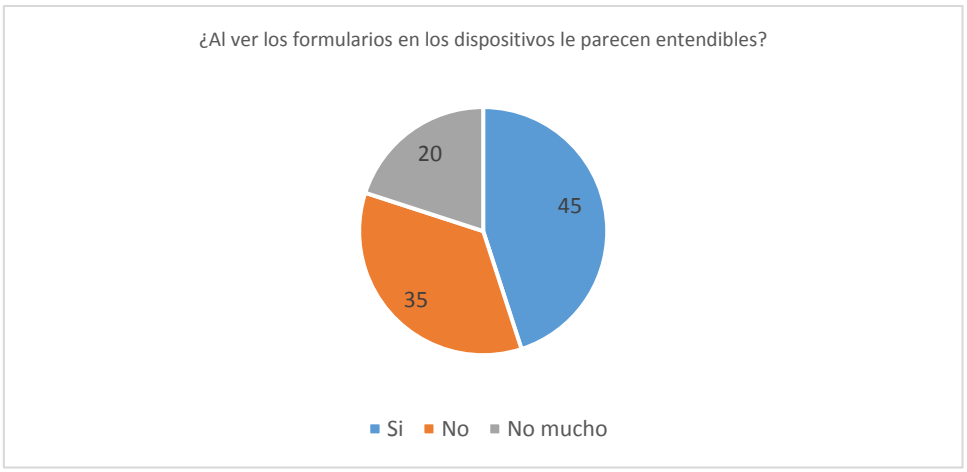


Figura 38 Grafico, entendimiento de formulario en dispositivo mobil.

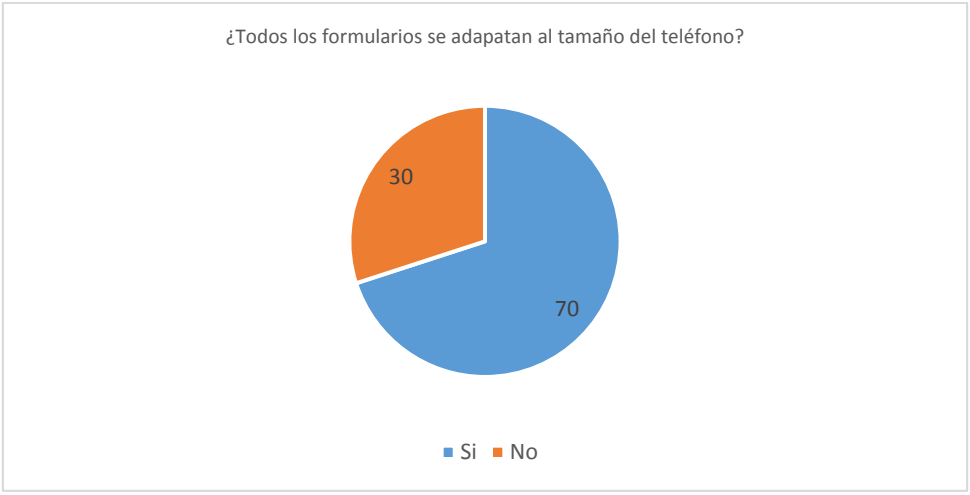


Figura 39 Grafico, adaptacion de los formularios

5. CONCLUSIONES

- Con la implementación del módulo de despacho y se logra mejorar significativamente los procesos de control dentro de la distribuidora, teniendo, además, un historial que facilita la detección de errores.

- El módulo de selección de rutas permitió mejorar los servicios de la empresa, evitando problemas como las rutas mal seleccionadas, ya sea por falta de experiencia o preferencias personales de los domicilios.

- Con la generación de reportes se logró tener un medio para analizar y controlar los procesos a través de datos históricos. Permitiendo corroborar las salidas e ingresos de los usuarios, además de permitir tener control de los tiempos que tardan los domicilios en completar una ruta.

- La visualización de las rutas en mapas facilito el reconocimiento de las zonas a visitar durante el trayecto de la ruta, Pero los problemas con las direcciones inexistentes o mal escritas demuestran la necesidad de un estudio de campo para el reconocimiento de las calles con el fin de mejorar la presentación en mapas.

6. TRABAJOS FUTUROS.

Teniendo en cuenta el trabajo actual, las dificultades encontradas durante el desarrollo del proyecto y las necesidades del cliente se espera a futuro realizarlas siguiente mejoras:

- Implementar el algoritmo en un lenguaje más apropiado que permita reducir el número de iteraciones y mejorar el desempeño.

- Realizar un estudio de nivel de tráfico y peligrosidad para buscar rutas alternativas.

- Agregar al módulo de rutas la gestión de vías para personalizar recorridos e incluir la captura de distancias entre dos puntos desde Google para tener un valor inicial entre cada par de puntos y asegurar la completitud del grafo.

- Realizar un estudio de las calles de Tuluá que permita optimizar la representación de las rutas y clientes en mapas

- Implementar un módulo de análisis de costos asociado al inventario de la empresa que permita incluir parámetros de valores de compra y venta de productos para conocer cuando una ruta genera sobrecostos.

7. BIBLIOGRAFIA

[1] J. J. Anaya Tejero*, El transporte de mercancías: Enfoque logístico de la distribución*. ESIC Editorial, 2009, pp. 17-20

[2] *http://es.wikipedia.org/wiki/Transporte*

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms, third edition*. Ed. The MIT Press 2009, pp. 651, 658, 693, 700

[4] D. Treschev, O. Zubelevich, *Introduction to the Perturbation Theory of Hamiltonian Systems*. Ed. Springer 2010. pp. 18

[5] *http://es.wikipedia.org/wiki/Teor%C3%ADa_perturbacional*

[6] J. Tuya, I. Ramos Román, J. J. Dolado Cosín, *Técnicas cuantitativas para la gestión en la ingeniería del software,* NETBIBLIO. S L. 2007. pp. 302, 309

[7] *http://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas*

[8] I. Soret los Santos, Logística comercial y empresarial. ESIC Editorial 2004. pp. 144

[9] Xin-She Yang, *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics*. Cambridge International Science Publishing 2008. pp. 67

[10] K. Deb, *Optimization for Engineering Design: Algorithms and Examples*, Second Edition, Eastern Economy Edition 2012. pp. 10

[11] I. Sommerville, Ingeniería de software, Pearson Educación, S.A. Madrid 2006. pp. 364-373.

[12] Imre J. Rudas, János Fodor, Janusz Kacprzyk, “Computational Intelligence and Informatics: Principles and Practice”, Springer-Verlag Berlin Heidelberg 2010, pp. 141-150.

[13] V.V. Muniswamy, “Design And Analysis Of Algorithms”, I K International Publishing House 2009, pp. 138 – 142.

[14] S.N. Sivanandam, S. N. Deepa, “Introduction to Genetic Algorithms”, Springer-Verlag Berlin Heidelberg 2008, pp. 263 – 271.

[15] Bruce L. Golden, S. Raghavan, Edward A. Wasil, “The Vehicle Routing Problem: Latest Advances and New Challenges: Latest Advances and New Chanllenges”, Springer Science+Business Media 2008. Pp. 3 - 23.

[16] Zbigniew Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”, Springer-Verlag Berlin Heidelberg New York 1999, pp. 25-32.
[17] Laurent Jan Pit, “*Parallel Genetic Algorithms*”, Universiteit Leiden
[18] Arthur E. Carter, “*Design And Application of Gennetic Algorithms for Multiple Traveling Salesperson Assigment Problem*”, Virginia Polytechnic Institute and State University, 2003
[19] G. Gutin, A.P. Punnen, *“The Traveling Salesman Problem and Its Variations”*, Springer Science+Bussines Media. LLC 2007 pp 7-9
[20] http://www.cs.us.es/~fsancho/?e=65
[21] http://www.sistema-logistico.com/
[22] http://www.logisplan.com/
[23] http://servinformacion.com/servinformacion/inicio
[24] Abraham Duarte Muñoz,”*Metaheurísticas”,* Librería-Editorial Dykinson, 2007, pág. 19

8. ANEXOS

1. Especificación de requerimientos.
2. Historias de usuario.
3. Diagrama de clases.
4. Diagrama de base de datos.
5. Modelos de navegación.
6. Pruebas
7. Pruebas de aceptcion
8. Manual de usuario
9. Manual de instalación.
10. Algunos costos

Información General	
Autor:	*Carlos Andrés López Ramírez*
Fecha de creación:	*Septiembre 2 de 2014*
Documentos relacionados:	*Historias de Usuario.docx*

Revisión Histórica	
Revisión No:	*009*
Descripción del cambio	*Se corrigieron de algunos requerimientos la redacción, datos de entrada y prioridad.*
Fecha ultima modificación:	*Mayo 16 de 2015*

Roles del Sistema	
Administrador del sistema:	**Es la persona encargada de controlar y supervisar el funcionamiento del entorno y quien define los roles y permisos de los usuarios.**
Supervisor:	**Persona encargada de verificar que las funciones se cumplan adecuadamente, puede realizar las tareas de los demás usuarios.**
Despacho:	**Persona encargada de revisar facturas pendiente y pedidos y autorizar la salida de estos, además de crear los inventarios de los domicilios y registrar devoluciones y cambios**
Domicilio:	**Persona encargada de llevar los pedidos al cliente, estos pueden revisar la información de las facturas, inventarios y rutas.**

Información general

Requerimiento No.	*REQ001*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario la creación roles.			
Entradas	Nombre rol, Permisos.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ002*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario buscar roles.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ003*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario modificar roles.			
Entradas	Nombre rol, Permisos.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ004*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario eliminar roles.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ005*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario la creación de usuarios.			
Entradas	Nombre, Número cedula, Usuario, Contraseña, Perfil , Celular		
Precondición	El usuario no debe existir.		
Referencias			

Información general

Requerimiento No.	*REQ006*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario buscar usuarios.			
Entradas	Número de cedula o Nombre		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ007*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario actualizar la información de los usuarios.			
Entradas	Nombre, Contraseña, Perfil , Celular, estado		
Precondición	El usuario debe existir en el sistema		
Referencias			

Información general

Requerimiento No.	*REQ008*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario eliminar usuarios.			
Entradas			
Precondición	El usuario debe existir en el sistema.		
Referencias			

Información general

Requerimiento No.	*REQ09*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario la creación de vehículos.			
Entradas	Número placa, Modelo, Marca, Cilindraje, Capacidad máxima en kilogramos, alto, ancho y largo del espacio de carga.		
Precondición	El vehículo no debe existir en el sistema.		
Referencias			

Información general

Requerimiento No.	*REQ010*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario buscar vehículos.			
Entradas	Número placa.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ011*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario actualizar la información de los vehículos.			
Entradas	Modelo, Marca, Cilindraje, Capacidad máxima en kilogramos, alto, ancho y largo del espacio de carga, estado.		
Precondición	El vehículo debe existir		
Referencias			

Información general

Requerimiento No.	*REQ012*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario eliminar vehículos.			
Entradas			
Precondición	El vehículo debe existir		
Referencias	*REQ006*		

Información general

Requerimiento No.	*REQ013*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario crear remolques.			
Entradas	Tipo enganche, Capacidad máxima en kilogramos, alto interno, ancho interno, largo interno.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ014*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario buscar remolques.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ015*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario actualizar la información de los remolques.			
Entradas			
Precondición	El remolque debe existir en el sistema.		
Referencias			

Información general

Requerimiento No.	*REQ016*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario eliminar remolques.			
Entradas			
Precondición	El remolque debe existir.		
Referencias			

Información general

Requerimiento No.	*REQ017*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario crear productos.			
Entradas	Id, Nombre, descripción, tipo, valor, IVA.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ018*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario consultar productos.			
Entradas	Código o Nombre del producto		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ019*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario actualizar la información de los productos.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ020*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador		
El sistema en el módulo de administración deberá permitir al usuario eliminar productos.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ021*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar la información de los clientes.			
Entradas	Número cedula, Nombre, Teléfono, Celular, Dirección, Barrio, nombre de edificio, apartamento.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ022*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario buscar clientes.			
Entradas	Número cedula. o nombre.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ023*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario actualizar la información de los clientes.			
Entradas	Nombre, Teléfono, Celular, Dirección, Barrio, nombre de edificio, apartamento.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ024*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario agregar domicilio a los clientes.			
Entradas	Teléfono, Dirección, Barrio.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ025*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario eliminar clientes.			
Entradas	Teléfono, Dirección, Barrio.		
Precondición			
Referencias			

Información general ***Pendiente***

Requerimiento No.	*REQ026*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar las facturas de los clientes			
Entradas	Número, Productos, Id Cliente, fecha.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ027*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario visualizar las facturas pendientes para despacho.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ028*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario relacionar la información de un cliente con una factura.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ029*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario establecer la prioridad de una factura.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ030*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración de rutas deberá permitir al usuario configurar las rutas a través de las mediciones de distancias obtenidas mediante el recorrido desde un punto de partida hasta cada uno de los clientes.			
Entradas	Dirección de inicio, Dirección final, Distancia.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ031*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario ingresar facturas de clientes al cálculo de ruta.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ032*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en los módulos de despacho y administración deberá permitir al usuario consultar las rutas.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ033*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario retirar una factura de una ruta.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ034*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario relacionar facturas de una ruta.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ035*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario programar el envío de pedidos a entregar en una hora específica.			
Entradas	Hora.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ036*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario asignar una ruta a un domicilio.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ037*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar productos pendientes en una factura.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ038*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el despacho de las facturas de una ruta.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ039*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el envío de productos pendientes en facturas.			
Entradas	Producto, Fecha.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ040*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el envío de productos para cambio.			
Entradas	Producto, Cantidad, Motivo, Estado.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ041*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario crear inventario de retorno según la entrega a realizar.			
Entradas	Tipo, Implemento, Cantidad		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ042*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Domicilio.		
El sistema en el módulo de despacho deberá permitir al usuario visualizar el inventario de retorno de la ruta asignada.			
Entradas			
Precondición	El usuario debe ingresar al sistema.		
Referencias			

Información general

Requerimiento No.	*REQ043*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el kilometraje de los vehículos al momento de iniciar el recorrido de una ruta.			
Entradas	Kilometraje, Hora.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ044*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de administración deberá permitir al usuario registrar el consumo de combustible de los vehículos.			
Entradas	Valor combustible, Kilometraje vehículo, Fecha.		
Precondición	El vehículo debe existir		
Referencias			

Información general

Requerimiento No.	*REQ045*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el kilometraje de los vehículos al terminar el recorrido de una ruta.			
Entradas	Kilometraje, Hora.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ046*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el ingreso de productos por motivo de cambio.			
Entradas	Producto, Cantidad, Motivo, Estado.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ047*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Media
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el ingreso de devoluciones relacionadas con una factura.			
Entradas	Producto, Cantidad, Motivo.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ048*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar el ingreso de inventario de retorno.			
Entradas	Producto, Cantidad.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ049*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar los faltantes en el inventario de retorno.			
Entradas	Producto, Cantidad, Motivo.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ050*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Alta
Descripción			
Usuarios	Despachador, Administrador.		
El sistema en el módulo de despacho deberá permitir al usuario registrar los sobrantes en el inventario de retorno.			
Entradas	Producto, Cantidad, Motivo, Valor.		
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ051*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Domicilio, Despachador, Administrador		
El sistema en el módulo de despacho deberá permitir al usuario visualizar las rutas en un mapa de la ciudad.			
Entradas			
Precondición	El usuario debe ingresar al sistema		
Referencias			

Información general

Requerimiento No.	*REQ052*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de reportes deberá permitir al usuario visualizar reportes de rutas y costos.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ053*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de reportes deberá permitir al usuario visualizar reportes de devoluciones y cambios.			
Entradas			
Precondición			
Referencias			

Información general

Requerimiento No.	*REQ054*	***Fecha de creación***	*02/09/2014*
Autor	Carlos Andrés López	***Prioridad***	Baja
Descripción			
Usuarios	Administrador.		
El sistema en el módulo de reportes deberá permitir al usuario visualizar reportes de salida e ingreso retornos.			
Entradas			
Precondición			
Referencias			

Información General	
Autor:	*Carlos Andrés López Ramírez*
Fecha de creación:	*Septiembre 2 de 2014*
Documentos relacionados:	*Requerimientos.docx*

Revisión Histórica	
Revisión No:	*009*
Descripción del cambio	*Se corrigieron de algunas Historias la redacción y los criterios de aceptación..*
Fecha última modificación:	*Mayo 16 de 2015*

US001	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Creación de rol	
Como	Administrador	
Quiero	Poder crear roles de usuario	
De forma que	Se pueda dar permisos teniendo como referencia el rol del usuario.	
Referencia	*REQ001*	
Criterio de aceptación		
Escenario	Crear rol de usuario	
Dado	Que se desea crear un nuevo rol de usuario	
Cuando	El usuario tiene los permisos	
Entonces	1. Se ingresa en la opción “Perfiles”. 2. Se despliega una lista de perfiles y opciones de los perfiles. 3. Se entra en la opción nuevo. 4. Se despliega el formulario de creación de perfil. 5. Se llenan los campos y se seleccionan los permisos. 6. Se guarda la información y regresa a la pantalla de perfiles.	

US002	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Listar roles	
Como	Administrador	
Quiero	Poder buscar roles de usuario	
De forma que	Se pueda ver la los roles existente al ingresar a la sección de roles de usuario en el menú de administración.	
Referencia	*REQ002*	
Criterio de aceptación		
Escenario	Buscar roles	
Dado	Que se desea ver los roles de usuario	
Cuando	Se ingresa al menú de roles.	
Entonces	1. Se muestra una lista de los roles existentes 2. Se ingresa el nombre o id del rol en el campo de búsqueda 3. Se muestra la lista de resultados que coincidan con la información ingresada	

US003	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Modificar roles	
Como	Administrador	
Quiero	Poder modificar roles de usuario	
De forma que	Se pueda cambiar los permisos de usuario cuando se requiera.	
Referencia	*REQ003*	
Criterio de aceptación		
Escenario	Editar Rol	
Dado	Que se desea editar un rol	
Cuando	Se necesita corregir los permisos y descripción	
Entonces	1. Se ingresa en la opción “Perfiles”. 2. Se despliega una tabla con una lista de perfiles. 3. Se busca el perfil y se da clic sobre la fila y se da clic en la opción ver. 4. Se despliega un formulario no editable con la información del perfil. 5. Se da clic en la opción actualizar y los campos se vuelven editables. 6. Se edita la información y se da clic en la opción guardar.	

US004	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Eliminar roles	
Como	Administrador	
Quiero	Poder eliminar roles de usuario	
De forma que	Se pueda eliminar los roles innecesarios o duplicados.	
Referencia	*REQ004*	
Criterio de aceptación		
Escenario	Eliminación de roles de usuario	
Dado	Que se desea eliminar un rol de usuario	
Cuando	Se ingresa en el menú de roles.	
Entonces	1. Se ingresa en la opción “perfiles” 2. Se despliega una lista de perfiles. 3. Busca el perfil a eliminar y se da clic sobre este. 4. Se da clic en el botón eliminar y se confirma la acción.	

US005	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Crear Usuarios	
Como	Administrador	
Quiero	Poder Crear usuarios	
De forma que	Los datos puedan ser usados para tener referencias y conocer los procesos relacionados con este.	
Referencia	*REQ005*	
Criterio de aceptación		
Escenario	Creación de usuario	
Dado	Que se desea crear un usuario.	
Cuando	Se necesita tener la referencia de un nuevo usuario.	
Entonces	1. Se ingresa en la opción "Usuarios" en el menú de administración. 2. Se despliega una lista de los usuarios disponibles. 3. Se da clic en la opción "Nuevo" 4. Se despliega el formulario de creación de usuarios. 5. Se llenan los campos del formulario y se da clic en la opción guardar.	

US006	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Buscar Usuarios	
Como	Administrador	
Quiero	Poder buscar usuarios	
De forma que	Se pueda ver la información del usuario usando como referencias el número de identificación de este.	
Referencia	*REQ006*	
Escenario	Búsqueda de usuario con id	
Dado	Que se requiere buscar un usuario.	
Cuando	Se necesita consultar la información asociada a este	
Entonces	1. Se ingresa en la opción "Usuarios" en el menú de administración. 2. Se despliega una lista de los usuarios registrados. 3. Se ingresa el número de cedula o nombre en el campo de búsqueda. 4. Se muestra una lista de resultados.	

US007	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Modificar Usuarios	
Como	Administrador	
Quiero	Poder modificar usuarios	
De forma que	Se pueda actualizar o corregir la información del usuario	
Referencia	*REQ007*	
Criterio de aceptación		
Escenario	Actualizar Usuario	
Dado	Que se desea modificar la información de un usuario.	
Cuando	La información del usuario cambia.	
Entonces	1. Se ingresa en la opción "Usuarios" en el menú de administración. 2. Se busca el usuario, se selecciona la fila correspondiente y se da clic en la opción ver. 3. Se despliega un formulario no editable con la información del usuario. 4. Se da clic en la opción actualizar, se cambia los datos del usuario y se da clic en la opción guardar.	

US008	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Eliminar Usuarios	
Como	Administrador	
Quiero	Poder Eliminar usuarios	
De forma que	No sean mostrados como usuarios activos.	
Referencia	*REQ008*	
Criterio de aceptación		
Escenario	Eliminación de usuario	
Dado	Que se desea eliminar un usuario.	
Cuando	Se ingresa al menú de usuarios se selecciona el usuario de la lista o se busca con su número de identificación	
Entonces	Se selecciona la opción eliminar usuario, el sistema muestra un mensaje pidiendo confirmar la acción.	

US009	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Crear Vehículos	
Como	Administrador	
Quiero	Poder Crear Vehículos	
De forma que	Los datos puedan ser usados como referencia y tener control de los procesos asociados a estos.	
Referencia	*REQ009*	
Criterio de aceptación		
Escenario	Creación de vehículos	
Dado	Que se desea crear un vehículo	
Cuando	Hay un nuevo vehículo.	
Entonces	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se selecciona la opción "Nuevo". 3. Se despliega un formulario para la creación de vehículos. 4. Se llena el formulario y se da clic en la opción "Guardar" 5. Retorna a la opción "Vehículos"	

US010	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Buscar Vehículos	
Como	Administrador	
Quiero	Poder Buscar Vehículos	
De forma que	Se pueda visualizar la información correspondiente.	
Referencia	*REQ010*	
Criterio de aceptación		
Escenario	Búsqueda de vehículos	
Dado	Que se desea buscar un vehículo	
Cuando	Se desea ver la información asociada a este.	
Entonces	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se despliega una lista con los vehículos registrados. 3. Se ingresa el número de placa, marca o modelo del vehículo en el campo de búsqueda. 4. Muestra una lista de los vehículos que coinciden con los datos de entrada.	

US011	*Prioridad: Alta*	***Estimado (días): 2***
Nombre	Modificar Vehículos	
Como	Administrador	
Quiero	Poder modificar la información de los vehículos	
De forma que	Se pueda corregir datos erróneos.	
Referencia	*REQ011*	
Criterio de aceptación		
Escenario	Actualizar vehículo	
Dado	Que se desea modificar un vehículo	
Cuando	Se necesita actualizar la información del vehículo.	
Entonces	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se busca el vehículo a actualizar, se selecciona el vehículo de la lista de resultados y se da clic en la opción ver. 3. Se despliega un formulario no editable con la información del vehículo, se da clic en la opción actualizar. 4. El formulario cambia a editable, se modifican los datos necesarios y se da clic en la opción guardar.	

US012	*Prioridad: Baja*	***Estimado (días): 2***
Nombre	Eliminar Vehículo.	
Como	Administrador	
Quiero	Poder eliminar vehículos.	
De forma que	Se pueda sacar los vehículos retirados de los activos.	
Referencia	*REQ012*	
Criterio de aceptación		
Escenario	Eliminación de vehículo	
Dado	Que se desea eliminar un vehículo.	
Cuando	Se ha ingresado de forma errónea.	
Entonces	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se busca el vehículo se da clic sobre el resultado que corresponda a la búsqueda. 3. Se da clic en la opción eliminar y se confirma la acción.	

US013	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Crear Remolques	
Como	Administrador	
Quiero	Poder Crear Remolques	
De forma que	Los datos puedan ser usados como referencia y tener control de estos.	
Referencia	*REQ013*	
Criterio de aceptación		
Escenario	Creación de remolques	
Dado	Que se desea crear un remolque	
Cuando	Hay un remolque nuevo	
Entonces	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se despliega una lista de los remolques existentes. 3. Se da clic en la opción "Nuevo". 4. Se despliega un formulario para la creación de remolques. 5. Se llena el formulario y se da clic en la opción "Guardar" 6. Regresa a la opción Remolques	

US014	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Buscar remolques	
Como	Administrador	
Quiero	Poder Buscar remolques	
De forma que	Se pueda visualizar la información correspondiente.	
Referencia	*REQ014*	
Criterio de aceptación		
Escenario	Búsqueda de remolques	
Dado	Que se desea buscar un remolque	
Cuando	Se desea ver la información de este.	
Entonces	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se muestra una lista de los remolques existentes. 3. Se ingresa el id o tipo de enganche del remolque en el campo de búsqueda. 4. Se muestra una lista con los remolques que coincidan con la información de búsqueda.	

US015	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Modificar Remolques	
Como	Administrador	
Quiero	Poder modificar la información de los remolques	
De forma que	Se pueda corregir datos erróneos.	
Referencia	*REQ015*	
Criterio de aceptación		
Escenario	Modificación de remolques	
Dado	Que se desea modificar un remolque	
Cuando	Se desea actualizar o corregir la información del remolque	
Entonces	1. Se ingresa a la opción "Remolques" en el menú de administración. 2. Selecciona busca el remolque y se selecciona en la lista 3. Se da clic en la opción ver y se despliega un formulario no editable con la información del remolque. 4. Se modifican los datos, se guardan y se confirma la acción.	

US016	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Eliminar Remolque.	
Como	Administrador	
Quiero	Poder eliminar remolques.	
De forma que	Se pueda sacar los remolques retirados de los activos.	
Referencia	*REQ016*	
Criterio de aceptación		
Escenario	Eliminación de Remolque	
Dado	Se comete un error.	
Cuando	Se registra el remolque.	
Entonces	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se busca el remolque a eliminar. 3. Se selecciona el remolque y se da clic en la opción eliminar. 4. Se muestra una ventada pidiendo que confirma la acción.	

US017	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Crear producto	
Como	Administrador	
Quiero	Poder crear productos	
De forma que	Los datos puedan ser usados como referencia y tener control sobre los procesos asociados a este.	
Referencia	*REQ017*	
Criterio de aceptación		
Escenario	Creación de producto	
Dado	Que se desea crear un producto.	
Cuando	Hay un nuevo producto	
Entonces	1. Se ingresa en la opción “Productos” en el menú de administración. 2. Se despliega una lista con los productos existentes y las opciones de producto. 3. Se da clic en la opción “Nuevo” 4. Se despliega un formulario para la creación de productos. 5. Se ingresan los datos del producto. 6. Se crean las presentaciones. 7. Se da clic en la opción “Guardar” y el sistema redirige a la pantalla de productos.	

US018	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Buscar producto	
Como	Administrador	
Quiero	Poder buscar productos	
De forma que	Pueda ver la información de estos cuando sea requerido.	
Referencia	*REQ018*	
Criterio de aceptación		
Escenario	Búsqueda de producto	
Dado	Que se desea buscar un producto.	
Cuando	Se requiere saber la información de este.	
Entonces	1. Se ingresa en la opción “Productos” en el menú de administración. 2. Se despliega una lista de los productos existentes y las opciones de estos. 3. Se escribe el nombre o código del producto en el campo de búsqueda. 4. Se muestra una lista de los productos que coincidan con la información ingresada.	

US019	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Modificar Producto	
Como	Administrador	
Quiero	Poder modificar productos	
De forma que	Pueda actualizar o corregir la información de estos cuando sea necesario.	
Referencia	*REQ019*	
Criterio de aceptación		
Escenario	Modificación de producto	
Dado	Que se quiere modificar un producto	
Cuando	La información de este cambia	
Entonces	1. Se ingresa en la opción "Productos" en el manu de administración. 2. Se busca el producto. 3. Se selecciona el producto a modificar y se da clic en la opción ver. 4. Se despliega un formulario no editable con la información del producto. 5. Se da clic en la opción "Actualizar" y se habilitan los campos con la información del producto y las opciones de las presentaciones. 6. Se realizan los cambios y se da clic en la opción "Guardar". 7. Muestra un mensaje informando que los cambios han sido guardados.	

US020	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Eliminar Producto	
Como	Administrador	
Quiero	Poder Eliminar productos	
De forma que	Se pueda eliminar información innecesaria.	
Referencia	*REQ020*	
Criterio de aceptación		
Escenario	Eliminación de producto	
Dado	Que se desea eliminar un producto.	
Cuando	Se registra un producto equivocado	
Entonces	1. Se ingresa en la opción "Productos" en el menú de administración. 2. Se busca el producto y se selecciona. 3. Se da clic en la opción "Eliminar". 4. Muestra un mensaje pidiendo confirmación de la acción. 5. Se da clic en "Aceptar".	

US021	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registrar Cliente	
Como	Administrador	
Quiero	Poder Registrar Clientes	
De forma que	Se tener su información como referencia para los procesos asociados a estos.	
Referencia	*REQ021*	
Criterio de aceptación		
Escenario	Registro de cliente	
Dado	Que se desea registrar un cliente.	
Cuando	Hay un cliente nuevo	
Entonces	1. Se ingresa en la opción "Clientes" en el menú de despacho. 2. Se despliega una lista con los clientes existentes y opciones. 3. Se da clic en la opción "Nuevo". 4. Se despliega un formulario para la creación del cliente. 5. Se llena el formulario y se da clic en la opción "Guardar". 6. Se guarda la información y redirige a la pantalla de clientes.	

US022	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Buscar Cliente	
Como	Administrador	
Quiero	Poder Buscar Clientes	
De forma que	Se pueda ver la información referente a este.	
Referencia	*REQ022*	
Criterio de aceptación		
Escenario	Búsqueda de cliente	
Dado	Que se desea ver la información de un cliente	
Cuando	Se necesita verificar la información de este.	
Entonces	1. Se ingresa en la opción "Clientes" en el menú de despacho. 2. Se despliega una lista con los clientes existentes. 3. Se ingresa el nombre o número de identificación en el campo de búsqueda. 4. Se muestra una lista con los clientes que coincidan con la información.	

US023	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Modificar Cliente	
Como	Administrador	
Quiero	Poder modificar la información de un cliente.	
De forma que	Se pueda cambiar la información cuando sea requerido.	
Referencia	*REQ023*	
Criterio de aceptación		
Escenario	Modificación de cliente	
Dado	Que se desea modificar la información de un cliente.	
Cuando	La información de este cambia.	
Entonces	1. Se ingresa en la opción "Clientes" en el menú de despacho. 2. Se busca el cliente con el número de identificación o el nombre. 3. Se selecciona el cliente de la lista y se da clic en la opción "Ver". 4. Se selecciona la opción "Actualizar" y el formulario cambia a estado editable. 5. Se ingresa la nueva información y se da clic en la opción guardar. 6. Se muestra un mensaje informando que la información ha sido guardada.	

US024	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Agregar Domicilio de Cliente	
Como	Administrador	
Quiero	Poder agregar un domicilio a un cliente.	
De forma que	Se pueda tener los diferentes domicilios de estos.	
Referencia	*REQ024*	
Criterio de aceptación		
Escenario	Agregar domicilio de cliente	
Dado	Que se desea agregar más domicilios de un cliente.	
Cuando	Se ingresa al menú de clientes	
Entonces	1. Se busca el cliente con uno de sus datos. 2. Se selecciona el cliente y se da clic en la opción editar. 3. Se da clic en la opción Actualizar. 4. En la lista de domicilios se da clic en la opción Nuevo. 5. Se llena el formulario y se da clic en la opción aceptar. 6. Se da clic en la opción Guardar.	

US025	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Eliminar Cliente	
Como	Administrador	
Quiero	Poder eliminar la información un cliente.	
De forma que	Se pueda bor5ar los datos suplicados o innecesarios.	
Referencia	*REQ026*	
Criterio de aceptación		
Escenario	Eliminación de cliente	
Dado	Que se desea eliminar un cliente.	
Cuando	Se ingresa al menú de clientes	
Entonces	1. Se busca el cliente. 2. Se selecciona de la lista y se da clic en el botón Eliminar. 3. Dar clic en el botón aceptar en la ventana emergente.	

US026	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Registrar Facturas de cliente	
Como	Administrador	
Quiero	Poder Registrar la información de las facturas de los clientes cliente.	
De forma que	Se pueda tener la información de estas para usarlas en los procesos de control de salida y envío de pedidos.	
Referencia	*REQ026*	
Criterio de aceptación		
Escenario	Registro de factura de cliente	
Dado	Que se desea registrar la factura de un cliente	
Cuando	Cuando se realiza una nueva compra de un cliente.	
Entonces	1. Se ingresa en el CRUD de facturas y se da clic en el botón Nuevo. 2. Se llena el formulario y se agregan los productos. 3. Se da clic en el botón Guardar y re-dirige al listado de facturas.	

US027	***Prioridad: 5***	***Estimado (días): 2***
Nombre	Búsqueda de Facturas Pendientes Para Despacho	
Como	Despachador, Administrador	
Quiero	Poder buscar las facturas pendientes por despachar.	
De forma que	Se pueda ver la información de estas para	
Referencia	*REQ027*	
Criterio de aceptación		
Escenario	Buscar facturas pendientes por despachar de cliente	
Dado	Que se desea buscar las facturas pendientes por despachar.	
Cuando	Se ingresa al menú de clientes	
Entonces	1. Ingresar en el CRUD de facturas. 2. Ingresar el número de factura o nombre del cliente en el campo de búsqueda.	

US028	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ingresar Distancias Entre Estaciones	
Como	Administrador	
Quiero	Poder configurar las distancias de los recorridos entre dos estaciones.	
De forma que	Se pueda usar esta información en el cálculo de rutas.	
Referencia	*REQ028*	
Criterio de aceptación		
Escenario	Ingresar distancias entre estaciones.	
Dado	Se debe configurar las distancias	
Cuando	Se agrega una nueva dirección	
Entonces	1. Se ingresa en el CRUD de rutas 2. Se selecciona la opción Configurar. 3. Se busca a dirección de destino y se selecciona. 4. Se selecciona la dirección origen y se da clic en el botón editar. 5. Se llena el formulario y se da clic en el botón guardar.	

US029	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Consultar Rutas	
Como	Administrador, Despachador	
Quiero	Poder consultar las rutas	
De forma que	Se pueda ver los clientes de estas y las distancias recorridas.	
Referencia	*REQ029*	
Criterio de aceptación		
Escenario	Consultar ruta.	
Dado	Que se desea consultar una ruta.	
Cuando	Se ingresa en la opción “Consultar Rutas” en el módulo de despacho.	
Entonces	1. Se ingresa en el CRUD de rutas. 2. Se selecciona el estado de la ruta. 3. Se selecciona un rango de fechas para la búsqueda y se da clic en el botón busca.	

US030	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Cancelar Envío de Factura	
Como	Administrador, Despachador	
Quiero	Poder cancelar el envío de facturas de rutas	
De forma que	Se pueda quitar de una ruta las facturas que no serán entregadas a domicilio.	
Referencia	*REQ030*	
Criterio de aceptación		
Escenario	Cancelar envío de factura	
Dado	Que se desea cancelar el envío de una factura..	
Cuando	Se ingresa en la opción "Consultar Rutas" en el módulo de despacho.	
Entonces	1. Se busca la ruta. 2. Se selecciona la ruta del listado. 3. Se da clic en el botón editar. 4. Se selecciona la factura del listado y se da clic en el botón retirar.	

US031	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Generar una ruta	
Como	Administrador, Despachador	
Quiero	Poder relacionar las facturas de una ruta.	
De forma que	Se puedan agrupar	
Referencia	*REQ031*	
Criterio de aceptación		
Escenario	Generación de rutas	
Dado	Que se sea generar una ruta	
Cuando	Hay pedido pendientes por enviar	
Entonces	1. Se ingresa en el CRUD de rutas. 2. Se da clic en el botón Nuevo 3. Se muestra el listado de facturas disponibles, vehículos disponibles y cambios. 4. Se seleccionan los vehículos, facturas y cambio a enviar. 5. Se da clic en la opción generar. 6. Redirige al listado de rutas.	

US032	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Programar Envío de Pedidos	
Como	Administrador, Despachador	
Quiero	Poder programar el envío de pedidos hechos para una hora especifica.	
De forma que	Se pueda cumplir con la hora estipulada por el cliente.	
Referencia	*REQ035*	
Criterio de aceptación		
Escenario	Programar envío de pedido	
Dado	Que se debe enviar un pedido a una hora especifica.	
Cuando	El usuario estipula una hora para recibir el producto.	
Entonces	1. Se ingresa en el CRUD de facturas. 2. Se busca la factura a programar. 3. Se selecciona la factura y se da clic en la opción Ver. 4. Se da clic en la opción Actualizar. 5. Se cambia la hora y fecha de envío.	

US033	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Asignar Ruta a Domicilio.	
Como	Administrador, Despachador	
Quiero	Poder asignar una ruta a un domicilio.	
De forma que	Se pueda tener control sobre el recorrido de los domicilios y los pedidos que llevan.	
Referencia	*REQ036*	
Criterio de aceptación		
Escenario	Asignar ruta a domicilio.	
Dado	Que se debe asignar una ruta a un domicilio.	
Cuando	Hay pedidos pendientes por entregar y hay domicilios disponibles.	
Entonces	En la opción “facturas Pendiente por Despachar” en el módulo de despacho, se selecciona la opción “Programar Envío”, se llenan los datos para programar el envío, se guardan y se confirman los datos.	

US034	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registro de Productos Pendiente.	
Como	Administrador, Despachador	
Quiero	Poder registrar los productos pendientes por entregar al cliente.	
De forma que	Se pueda tener control sobre la entrega de estos.	
Referencia	*REQ037*	
Criterio de aceptación		
Escenario	Registrar producto pendiente.	
Dado	Que se faltan productos en stock.	
Cuando	El cliente pide el producto faltante y solicita su posterior envío..	
Entonces	1. Ingresa en el CRUD de pendientes. 2. Selecciona la opción Nuevo. 3. Llena el formulario y da clic en la opción aceptar.	

US035	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Despacho de Pedidos	
Como	Administrador, Despachador	
Quiero	Poder registrar el envío de pedidos marcando como enviada la respectiva factura.	
De forma que	Se pueda tener control la salida de los pedidos.	
Referencia	*REQ038*	
Criterio de aceptación		
Escenario	Despachar pedidos.	
Dado	Que hay facturas pendientes por despachar.	
Cuando	El cliente solicita un domicilio.	
Entonces	1. En la opción “facturas Pendiente por Despachar” en el módulo de despacho, se selecciona la opción “Registrar salida de Pedidos”, 2. se selecciona la ruta de la lista y se muestran las facturas de estas 3. se marcan las facturas como despachadas.	

US036	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Despacho de Productos Pendientes	
Como	Administrador, Despachador	
Quiero	Poder registrar el envío de productos pendientes.	
De forma que	Se pueda tener control la salida de los productos a entregar.	
Referencia	*REQ039*	
Criterio de aceptación		
Escenario	Despacho de productos pendientes.	
Dado	Que hay productos pendientes por entregar de facturas anteriormente despachadas.	
Cuando	Se actualiza el stock de productos.	
Entonces	1. ingresa opción "Pendientes" en el módulo de despacho. 2. se selecciona los productos pendientes por entregar y se agregan al cálculo de ruta.	

US037	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Despacho de Cambios	
Como	Administrador, Despachador	
Quiero	Poder registrar el envío de productos en motivo de cambio.	
De forma que	Se pueda tener control la salida de los pedidos.	
Referencia	*REQ040*	
Criterio de aceptación		
Escenario	Despachar cambios	
Dado	Que hay cambios para despachar.	
Cuando	El cliente solicita un cambio.	
Entonces	1. Ingresa en el CRUD de cambios. 2. Selecciona la opción Nuevo. 3. Llena el formulario. 4. Da clic en la opción guardar.	

US038	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Creación de Inventario de retorno	
Como	Administrador, Despachador	
Quiero	Poder crear inventarios de retorno	
De forma que	Se pueda tener control de los productos que deben ser devueltos por los domicilios.	
Referencia	*REQ041*	
Criterio de aceptación		
Escenario	Crear inventario de retorno.	
Dado	Que se requiere saber los que productos deben regresar	
Cuando	Se envían los pedidos de una ruta.	
Entonces	En la opción "facturas Pendiente por Despachar" en el módulo de despacho, se selecciona la opción "Registrar Salida de Productos", se ingresa a la opción "Crear Inventario de Retorno", se llenan los datos y se guarda la información.	

US039	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Consultar Inventario de retorno	
Como	Administrador, Despachador	
Quiero	Poder crear inventarios de retorno	
De forma que	Se pueda tener control de los productos que deben ser devueltos por los domicilios.	
Referencia	*REQ042*	
Criterio de aceptación		
Escenario	Consultar inventario de retorno.	
Dado	Que se requiere visualizar el inventario de retorno de un domicilio.	
Cuando	Se necesita consultar los inventarios de los domicilios.	
Entonces	Se selecciona la opción "Inventarios de Retorno" en el módulo de despacho, se escoge el usuario de la lista y se muestra una lista de los inventarios del usuario, se selecciona la opción "ver" en la lista y se muestra la información.	

US040	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registrar Kilometraje inicial.	
Como	Administrador, Despachador	
Quiero	Poder registrar el kilometraje de los vehículos al momento de iniciar una ruta.	
De forma que	Se pueda tener control de la distancia recorrida en la ruta y el costo de esta.	
Referencia	*REQ043*	
Criterio de aceptación		
Escenario	Registrar kilometraje inicial.	
Dado	Que se requiere registrar el kilometraje de los vehículos	
Cuando	Los domicilios inician una ruta.	
Entonces	En la opción "Registrar Salida de Pedidos" en el módulo de despacho, se llenan los datos del vehículo.	

US041	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registrar Kilometraje de Salida.	
Como	Administrador, Despachador	
Quiero	Poder registrar el kilometraje de los vehículos al momento de iniciar una ruta.	
De forma que	Se pueda tener control de la distancia recorrida en la ruta y el costo de esta.	
Referencia	*REQ044*	
Criterio de aceptación		
Escenario	Registrar kilometraje de salida.	
Dado	Que se requiere registrar el kilometraje de los vehículos	
Cuando	Los domicilios inician una ruta.	
Entonces	En la opción "Registrar Salida de Pedidos" en el módulo de despacho, se llenan los datos del vehículo.	

US042	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registrar Kilometraje de Llegada.	
Como	Administrador, Despachador	
Quiero	Poder registrar el kilometraje de los vehículos cuando terminan una ruta.	
De forma que	Se pueda tener control de la distancia recorrida en la ruta y el costo de esta.	
Referencia	*REQ045*	
Criterio de aceptación		
Escenario	Registrar kilometraje de llegada.	
Dado	Que se requiere registrar el kilometraje de los vehículos	
Cuando	Los domicilios terminan una ruta.	
Entonces	En la opción "Recepción de Domicilios" en el módulo de despacho, se llenan los datos del vehículo.	

US043	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ingreso de Cambios	
Como	Administrador, Despachador	
Quiero	El ingreso de productos para cambio.	
De forma que	Se pueda conocer el estado de los cambios solicitados por los clientes.	
Referencia	*REQ046*	
Criterio de aceptación		
Escenario	Registrar ingreso de cambios de cliente.	
Dado	Que se requiere registrar los productos ingresados para cambios.	
Cuando	El cliente envía un producto solicitando un cambio.	
Entonces	En la opción "Cambio de Productos" en el módulo de despacho, se llena la información del cambio con su estado y motivo.	

US044	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ingreso de Devoluciones de Clientes	
Como	Administrador, Despachador	
Quiero	Registrar las devoluciones de pedidos.	
De forma que	Se pueda conocer los productos devueltos y el motivo de la devolución.	
Referencia	*REQ047*	
Criterio de aceptación		
Escenario	Registrar ingreso de devoluciones de cliente.	
Dado	Que se debe registrar las devoluciones	
Cuando	El cliente devuelve productos de un pedido.	
Entonces	En la opción "Devoluciones de Productos" en el módulo de despacho, se selecciona en domicilio que se llevó y se muestra una lista con las facturas de la ruta recorrida, se selecciona la factura y se llena la información de los productos devueltos con su motivo.	

US045	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ingreso de inventario de retorno	
Como	Administrador, Despachador	
Quiero	Registrar el ingreso de productos que deben ser traídos por los domicilios después de terminar una ruta.	
De forma que	Se pueda conocer los productos devueltos, los faltantes y los sobrantes del inventario.	
Referencia	*REQ048*	
Criterio de aceptación		
Escenario	Ingreso de inventario de retorno	
Dado	Que se debe registrar los productos regresados por los domicilios.	
Cuando	Este termina una ruta.	
Entonces	En la opción "Inventario de Retorno" en el módulo de despacho, se selecciona en domicilio que se llevó y se muestra una lista con los productos que deben ser devueltos, se registra los productos devueltos.	

US046	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registro de faltantes en el inventario de retorno	
Como	Administrador, Despachador	
Quiero	Registrar los faltantes en el inventario de retorno.	
De forma que	Se conocer cuando se venden productos retornables.	
Referencia	*REQ049*	
Criterio de aceptación		
Escenario	Registro de faltantes en el inventario de retorno	
Dado	El domicilio vende un producto retornable.	
Cuando	Lleva pedidos a los clientes.	
Entonces	En la opción "Inventario de Retorno" en el módulo de despacho, se selecciona en domicilio que se llevó y se muestra una lista con los productos que deben ser devueltos, se registra los productos los faltantes con su motivo y el cliente.	

US047	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Registro de sobrantes en el inventario de retorno	
Como	Administrador, Despachador	
Quiero	Registrar los sobrantes en el inventario de retorno.	
De forma que	Se conocer cuando se ingresan productos que no están en el inventario.	
Referencia	*REQ050*	
Criterio de aceptación		
Escenario	Registro de sobrantes en el inventario de retorno	
Dado	Que el domicilio ingresa un producto retornable que no está en el inventario de retorno.	
Cuando	Lleva pedidos a los clientes.	
Entonces	En la opción "Inventario de Retorno" en el módulo de despacho, se selecciona en domicilio que se llevó y se muestra una lista con los productos que deben ser devueltos, se registra los productos los sobrantes con su motivo y el cliente.	

US048	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Visualizar rutas en mapa	
Como	Administrador, Despachador, Domicilio	
Quiero	Ver las rutas en un mapa de la ciudad	
De forma que	Pueda tener una guía clara de la ruta a tomar.	
Referencia	*REQ051*	
Criterio de aceptación		
Escenario	Visualizar ruta en mapa	
Dado	Que el domicilio requiere entregar los pedidos	
Cuando	Se le asigna una ruta	
Entonces	En el módulo de despacho en la opción "Ver Rutas" selecciona el domicilio y el sistema muestra las rutas que le han sido asignadas, dan clic en la opción "Ver" y se despliega un mapa con la ruta seleccionada.	

US049	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ver reporte de rutas	
Como	Administrador	
Quiero	Ver reportes de rutas	
De forma que	Pueda conocer datos estadísticos de los costos y distancias de las rutas, además de saber las rutas más frecuentes.	
Referencia	*REQ052*	
Criterio de aceptación		
Escenario	Ver reporte de rutas	
Dado	Se desea ver reportes de las rutas y los costos de estas.	
Cuando	Se quiere conocer los datos estadísticos de estas.	
Entonces	En el módulo de administración en la opción "Reportes" selecciona "Reportes de Rutas" y el sistema muestra la información.	

US050	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ver reporte de devoluciones y cambios.	
Como	Administrador	
Quiero	Ver reportes de devoluciones y cambios.	
De forma que	Pueda conocer datos estadísticos de los costos asociados a los motivos de los clientes.	
Referencia	*REQ053*	
Criterio de aceptación		
Escenario	Ver reporte de devoluciones y cambios.	
Dado	Se desea ver reportes de las devoluciones y cambios de los clientes.	
Cuando	Se quiere conocer los datos estadísticos de estos.	
Entonces	En el módulo de administración en la opción "Reportes" selecciona "Reportes de Cambios y Devoluciones" y el sistema muestra la información.	

US051	*Prioridad: 5*	***Estimado (días): 2***
Nombre	Ver reporte de retornos.	
Como	Administrador	
Quiero	Ver reportes de retornos.	
De forma que	Pueda conocer datos estadísticos y los costos asociados a los productos que deben ser regresados por los domicilios.	
Referencia	*REQ054*	
Criterio de aceptación		
Escenario	Ver reporte de retornos	
Dado	Se desea ver reportes de los retornos de los domicilios.	
Cuando	Se quiere conocer los datos estadísticos de estos.	
Entonces	En el módulo de administración en la opción "Reportes" selecciona "Reportes de Retornos" y el sistema muestra la información.	

11. Diagrama de Clases.

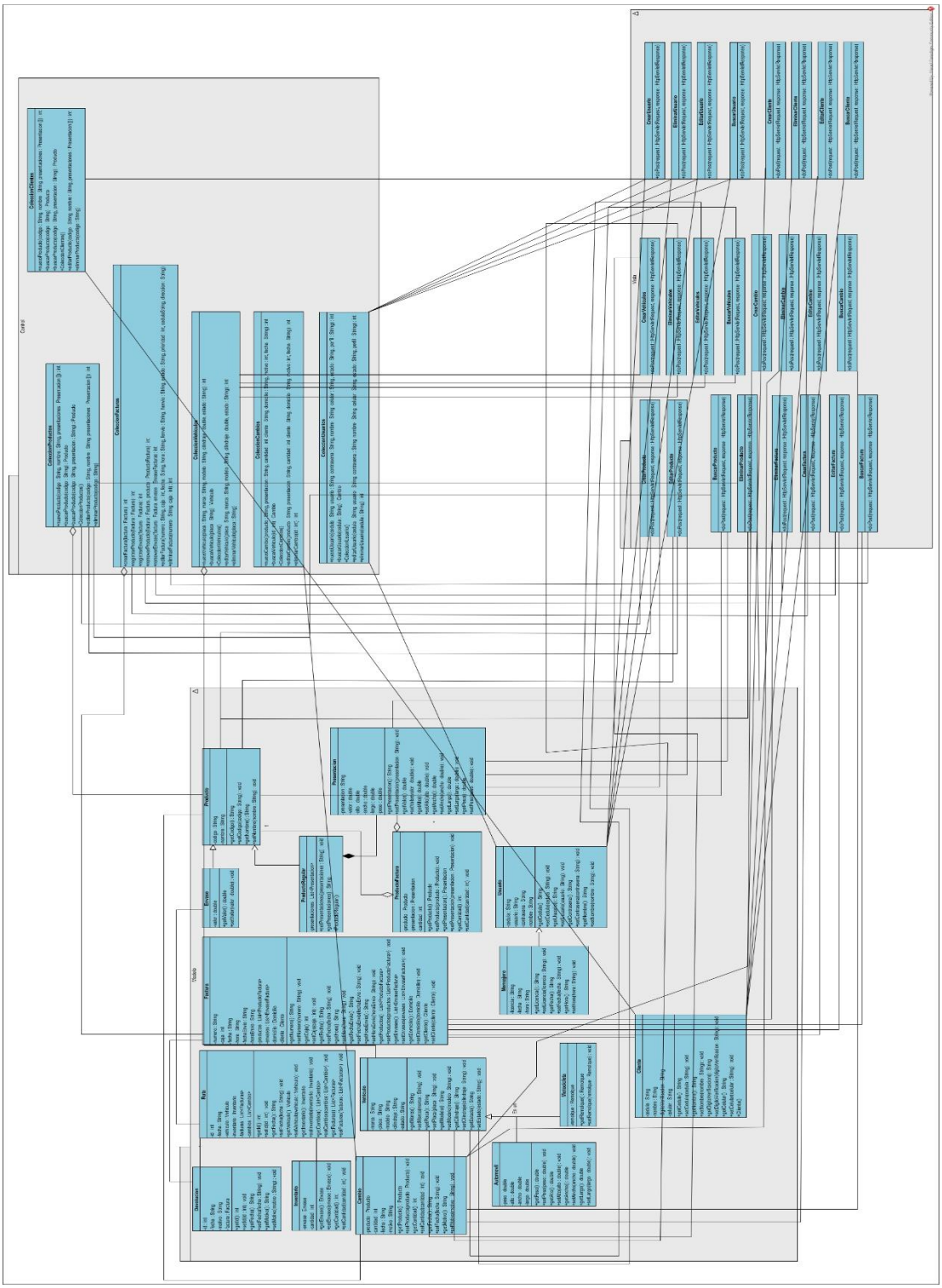


Diagrama de clases completo

12. Modelo de Base de Datos.

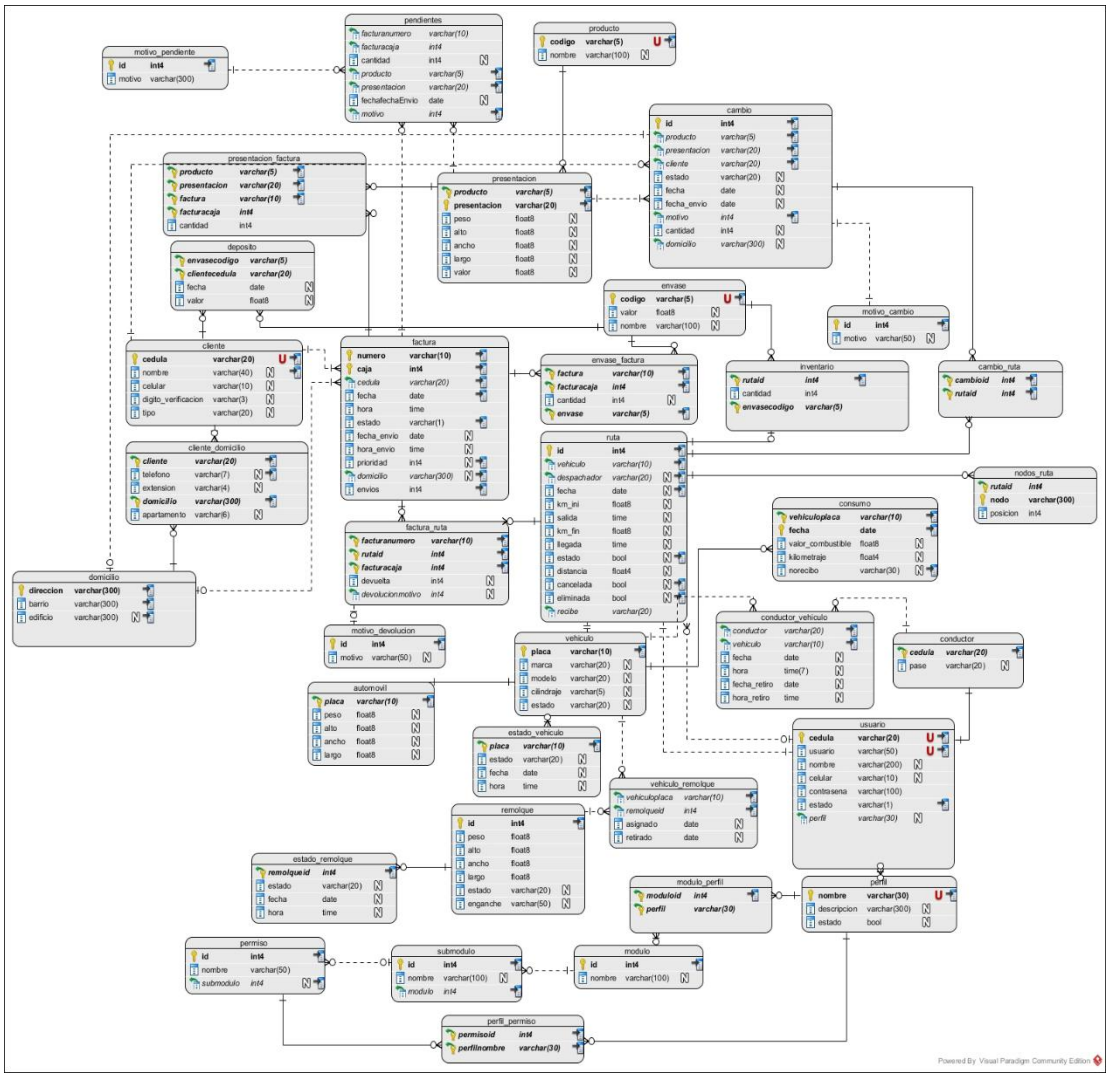


Diagrama de base de datos completo

Modelos de Navegación.

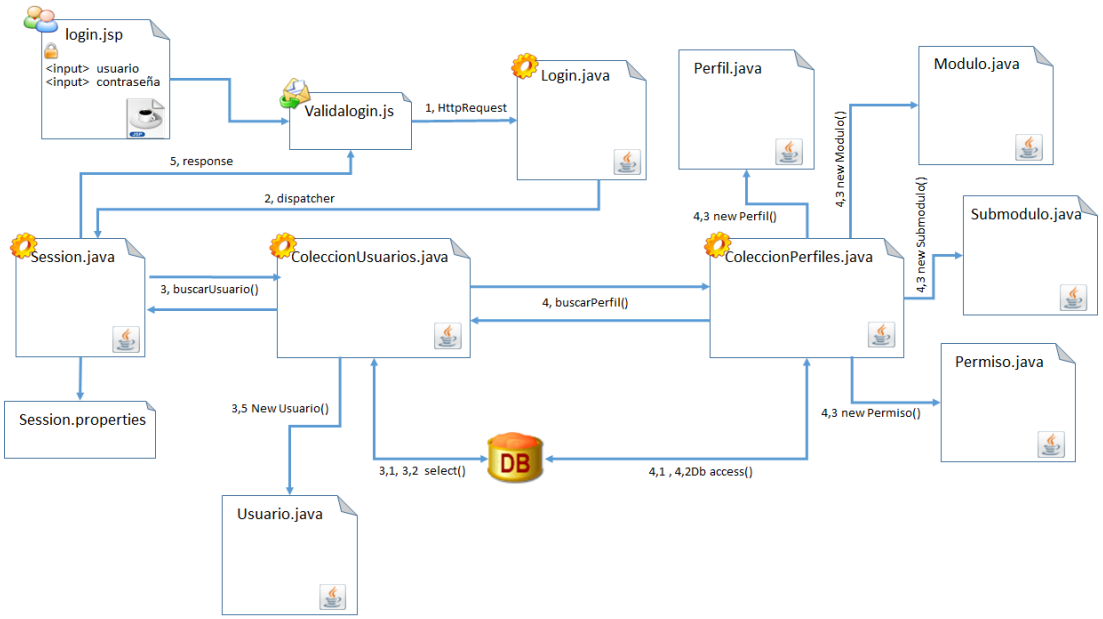


Figura 1 Diagrama de navegación, Login

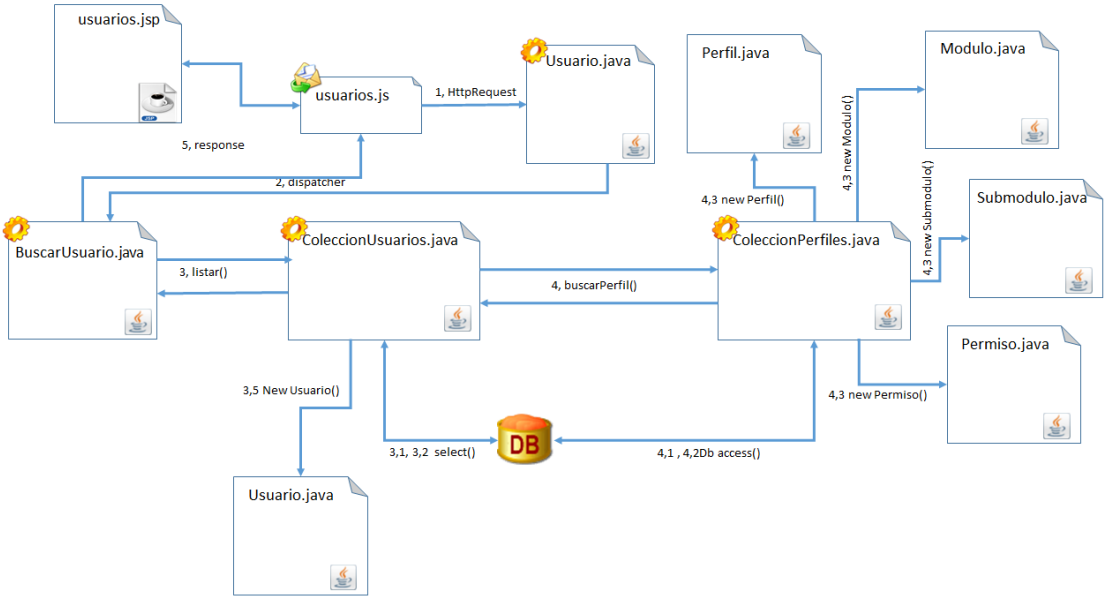


Figura 2 Diagrama de navegación, Listar usuarios

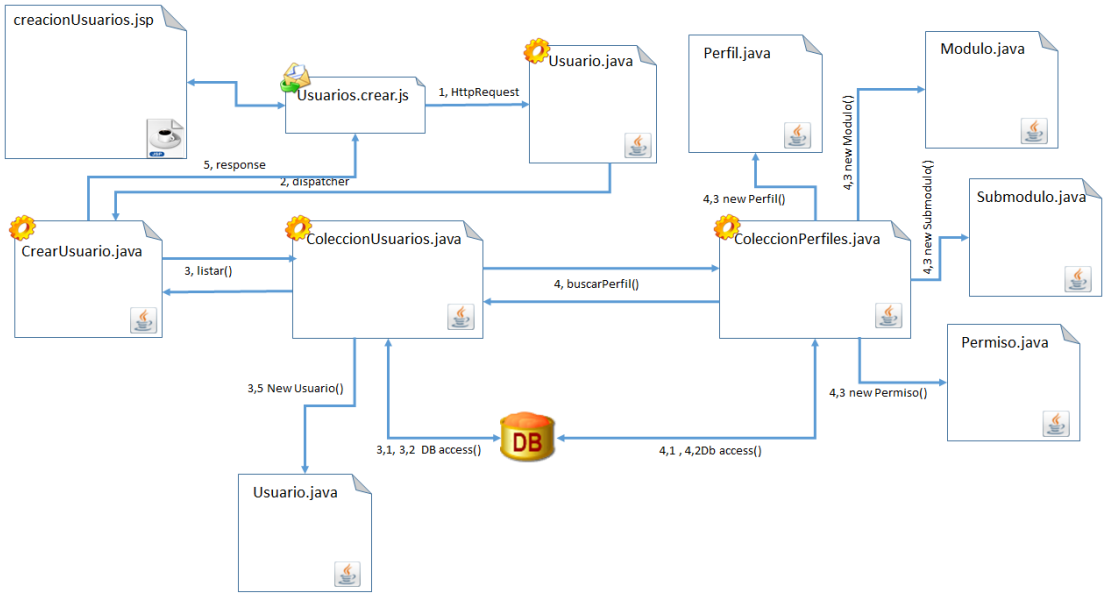


Figura 3 Diagrama de navegación, Creación de usuario

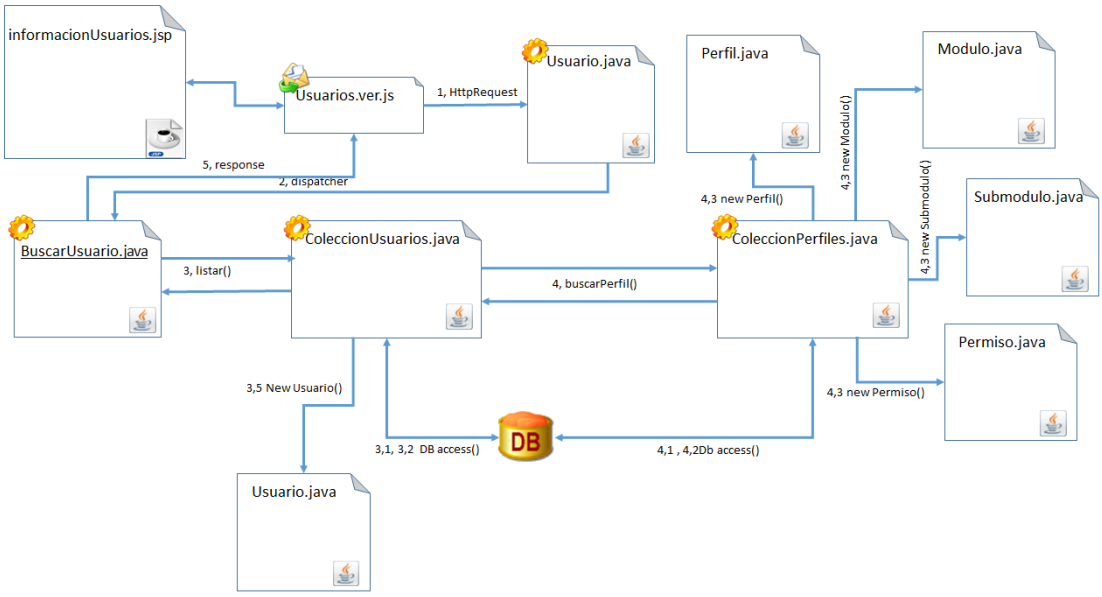


Figura 4 Diagrama de navegación, Consultar usuario

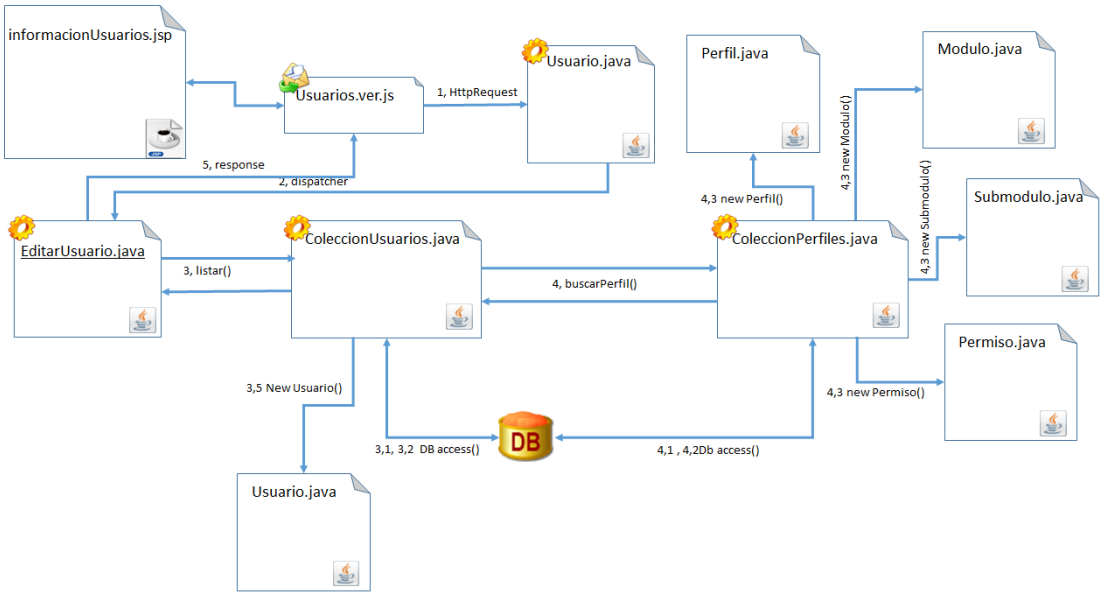


Figura 5 Diagrama de navegación, Editar usuario

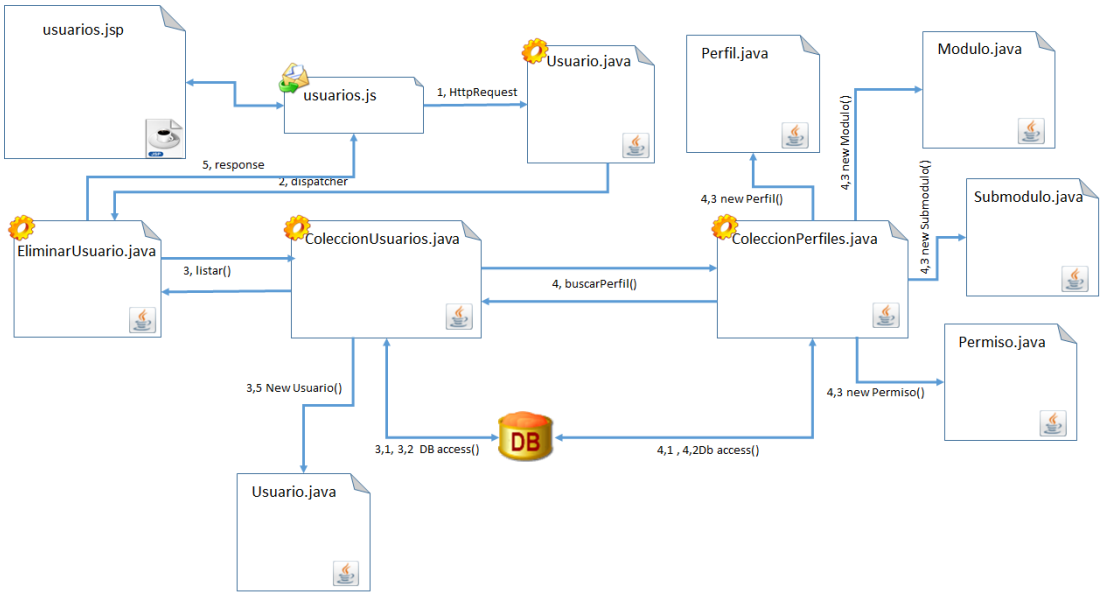


Figura 6 Diagrama de navegación, Eliminar usuario

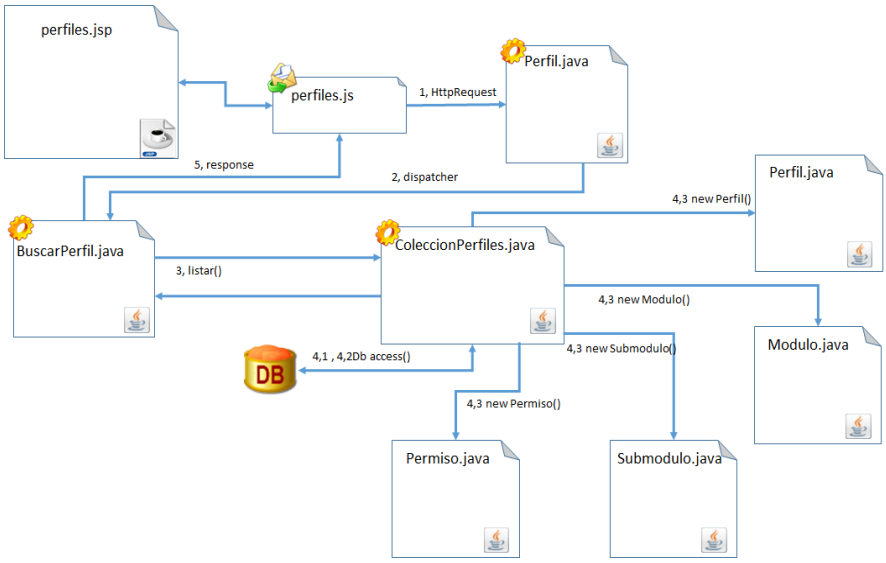


Figura 7 Diagrama de navegación, listar perfiles

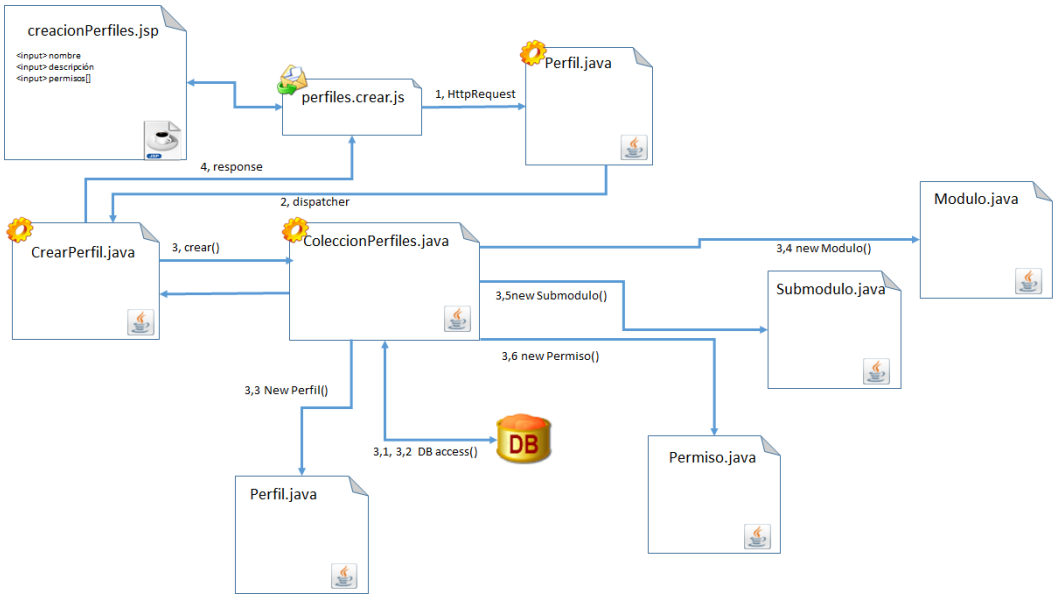


Figura 8 Diagrama de navegación, crear perfiles

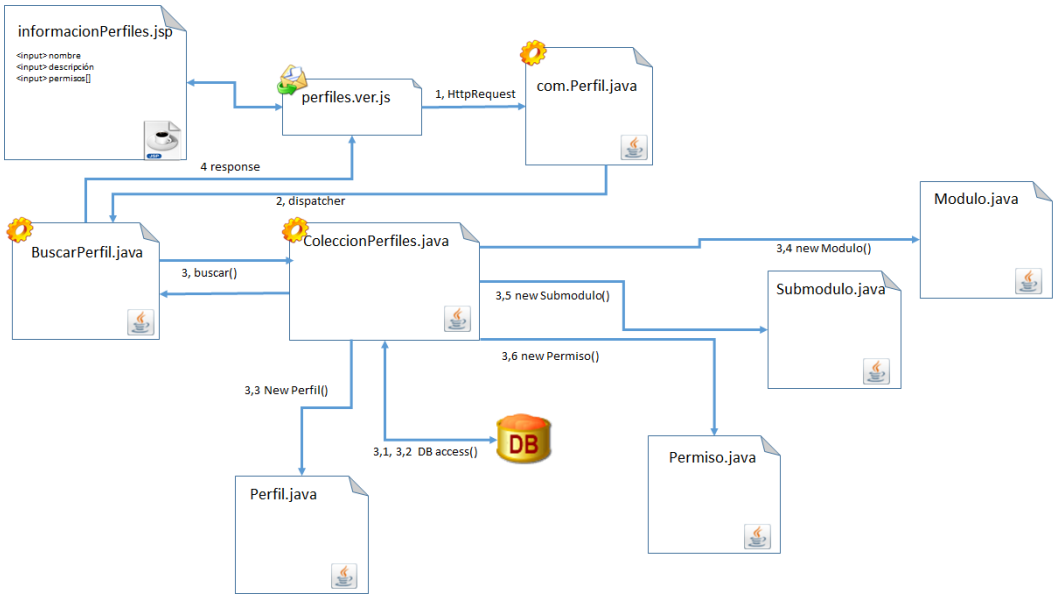


Figura 9 Diagrama de navegación, consultar perfil

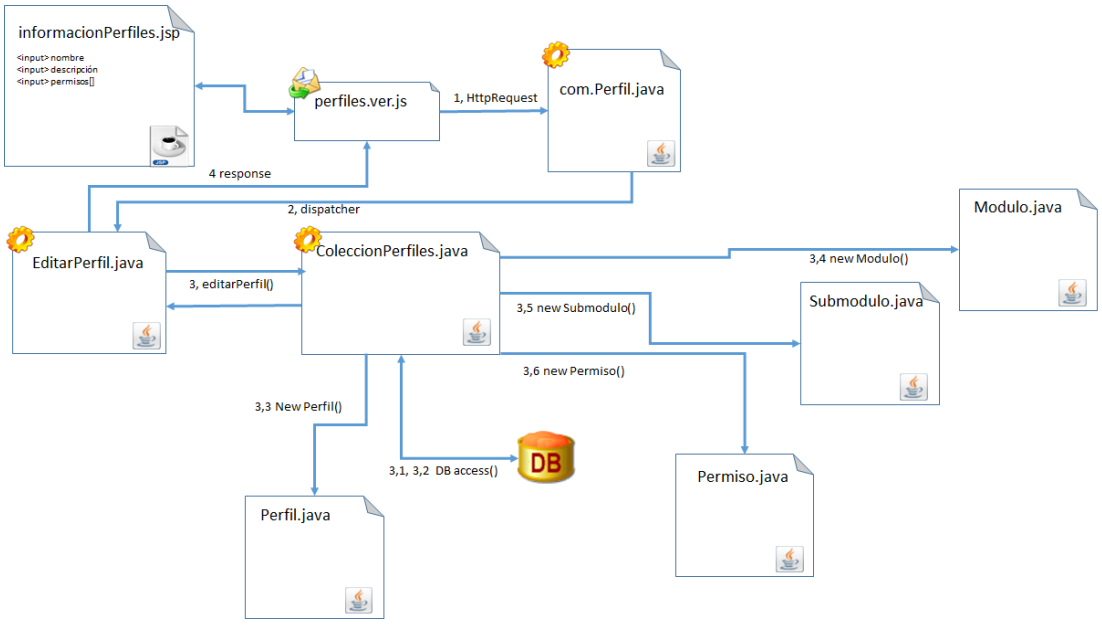


Figura 10 Diagrama de navegación, editar perfil

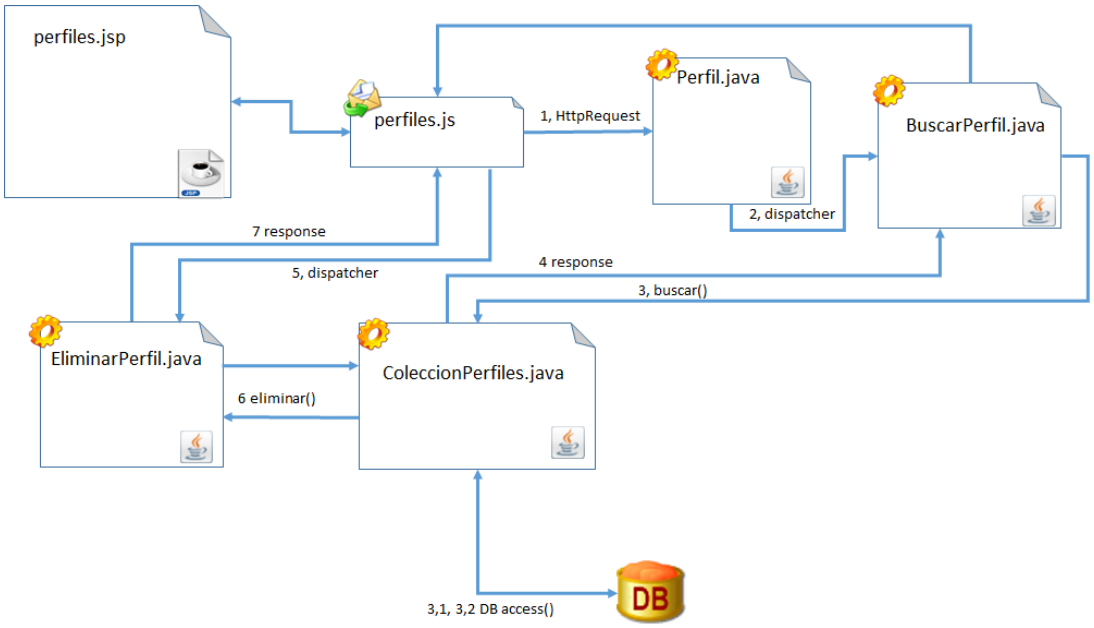


Figura 11 Diagrama de navegación, eliminar perfiles

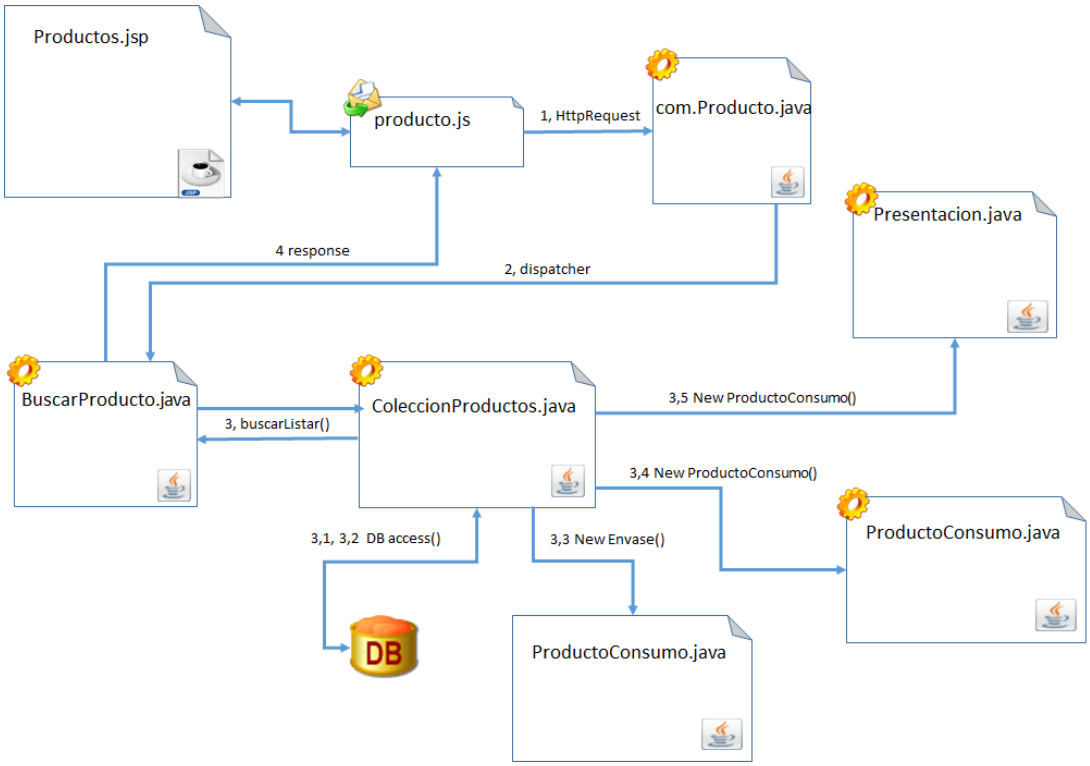


Figura 12 Diagrama de navegación, listar productos

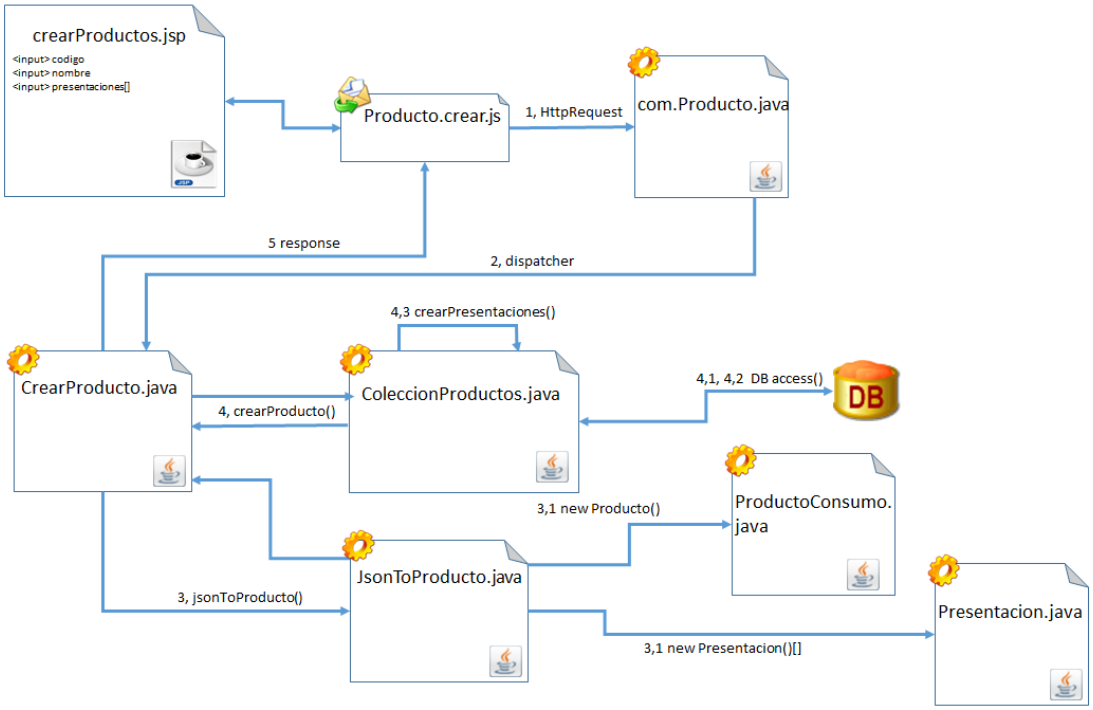


Figura 13 Diagrama de navegación, crear producto

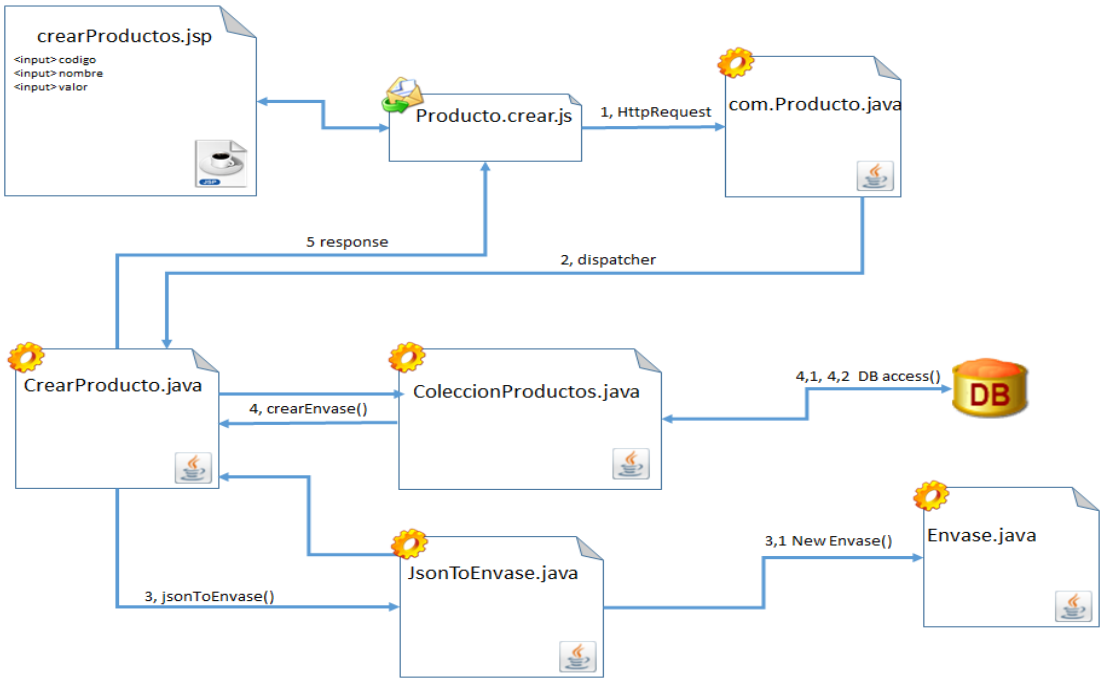


Figura 14 Diagrama de navegación, crear envase

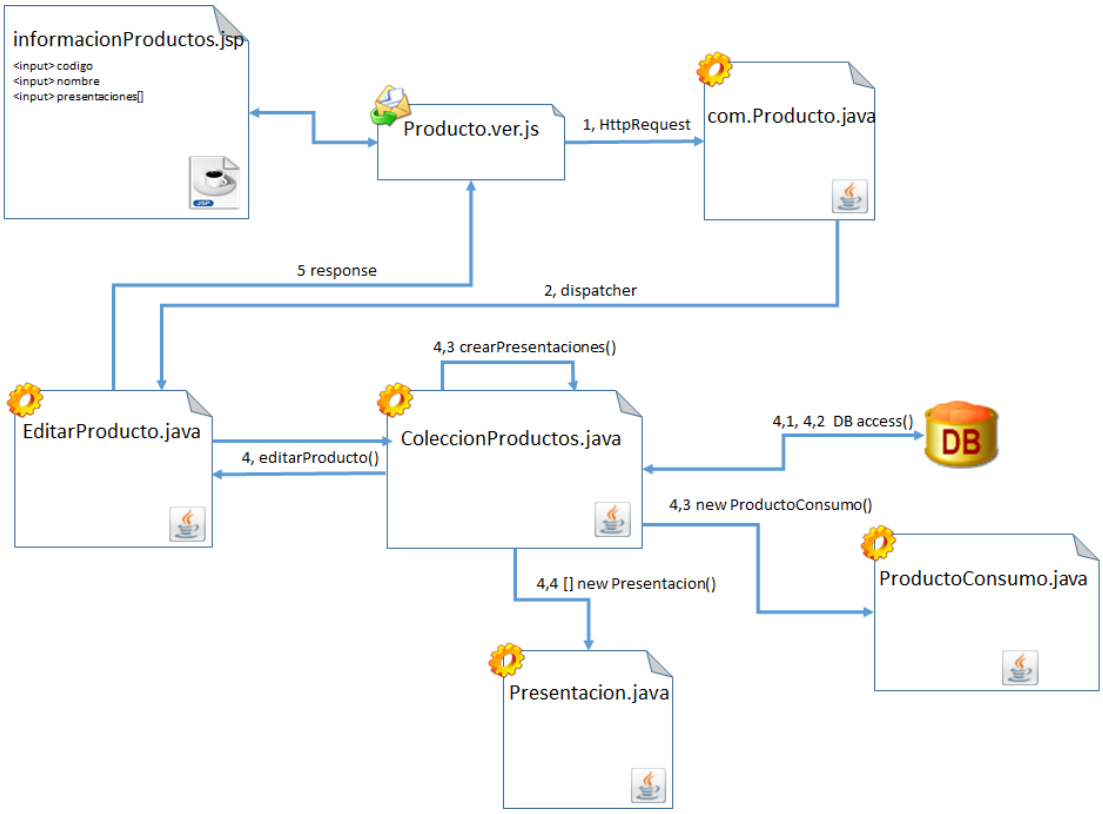


Figura 15 Diagrama de navegación, editar producto

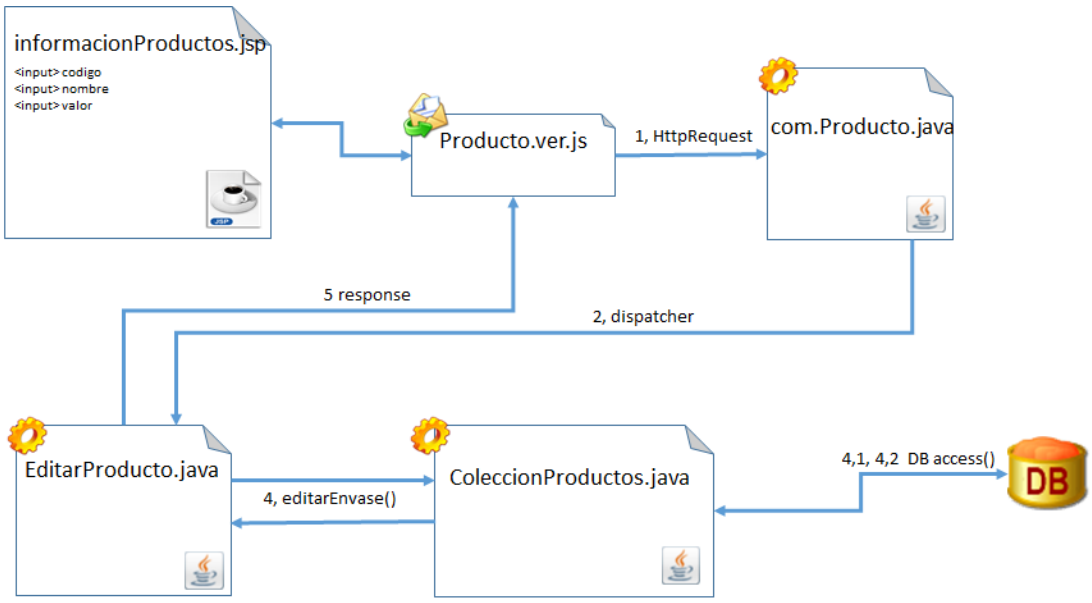


Figura 16 Diagrama de navegación, editar envase

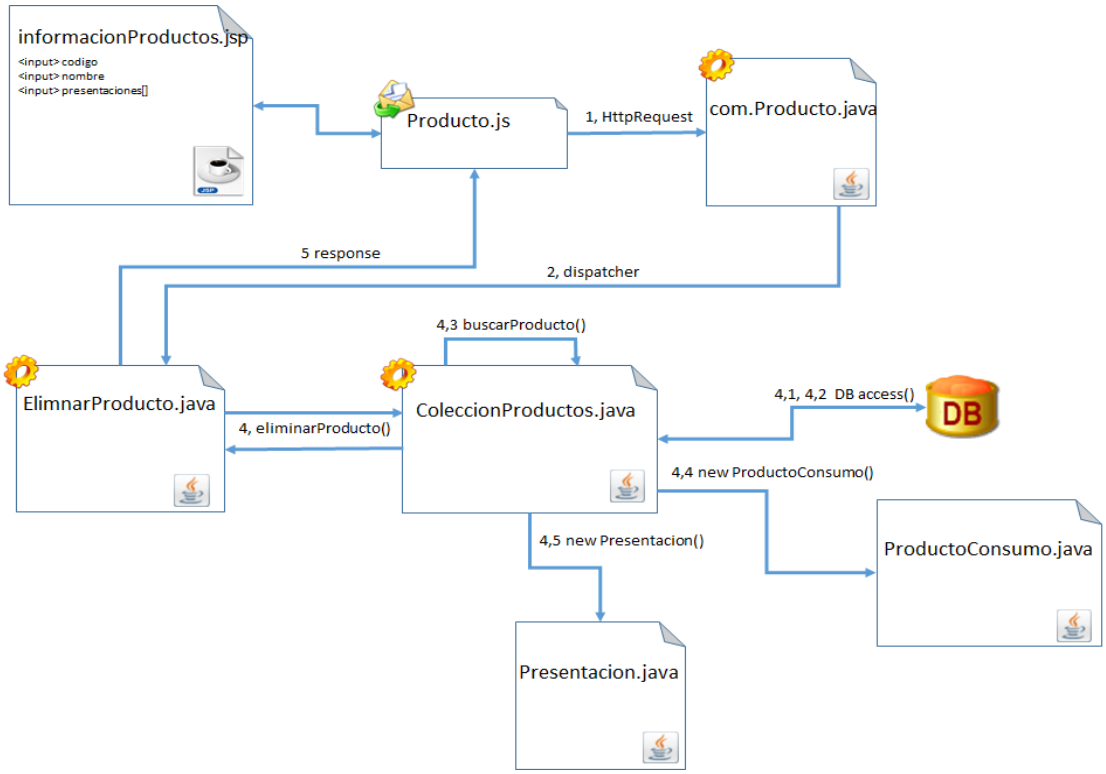


Figura 17 Diagrama de navegación, eliminar producto

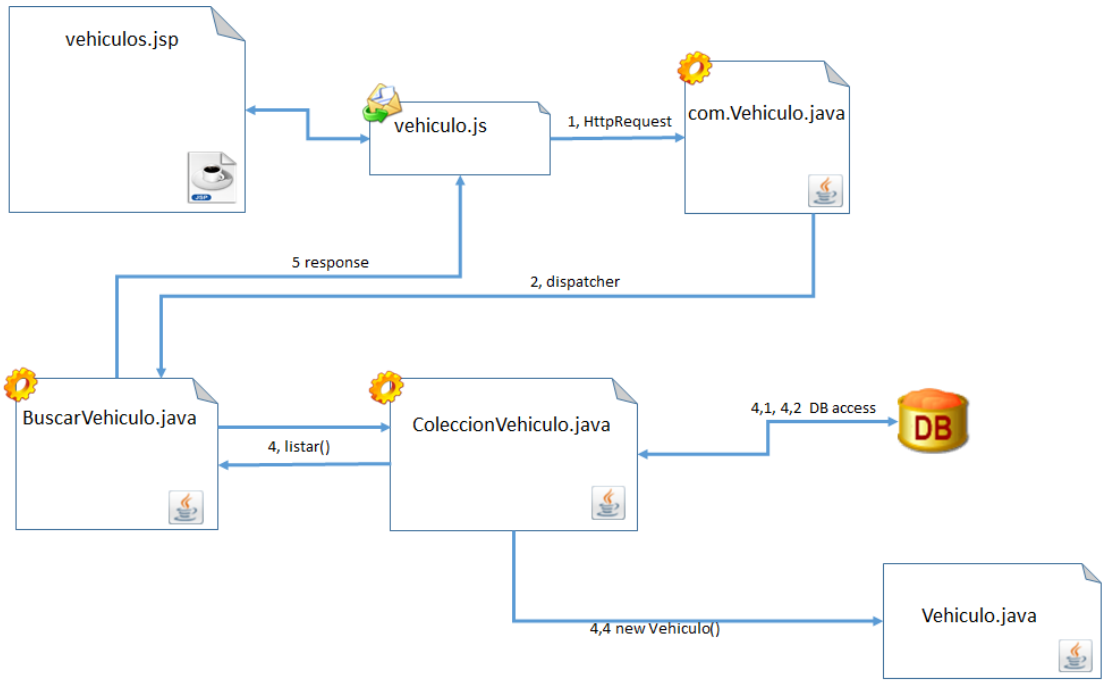


Figura 18 de navegación, listar vehículos

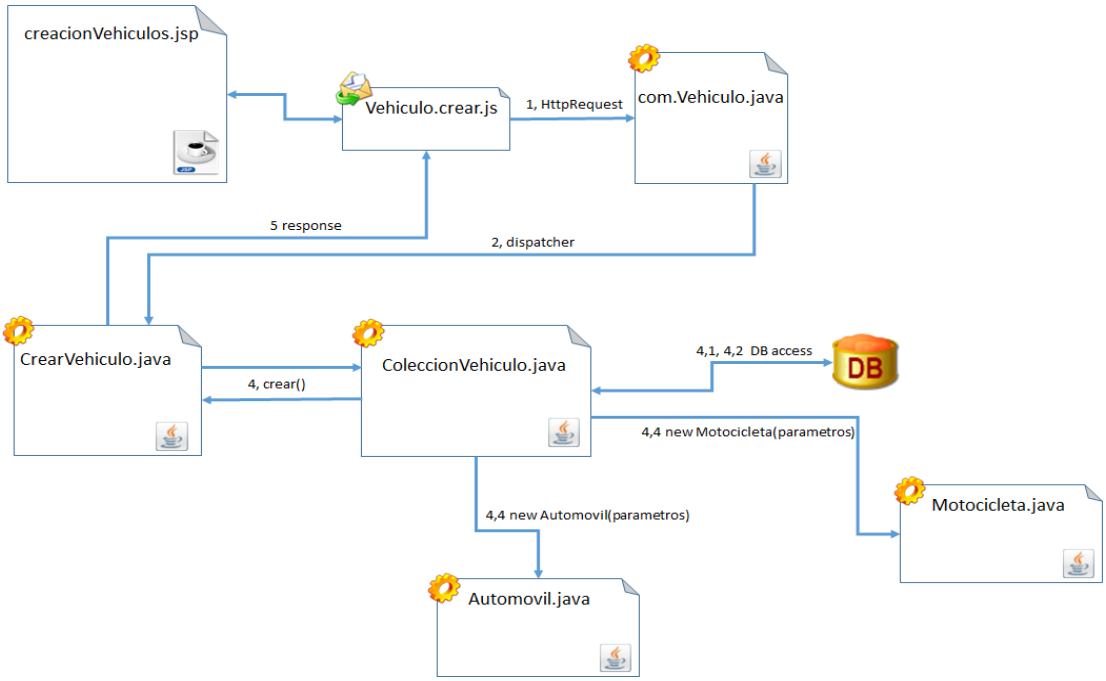


Figura 19 Diagrama de navegación, crear vehículo

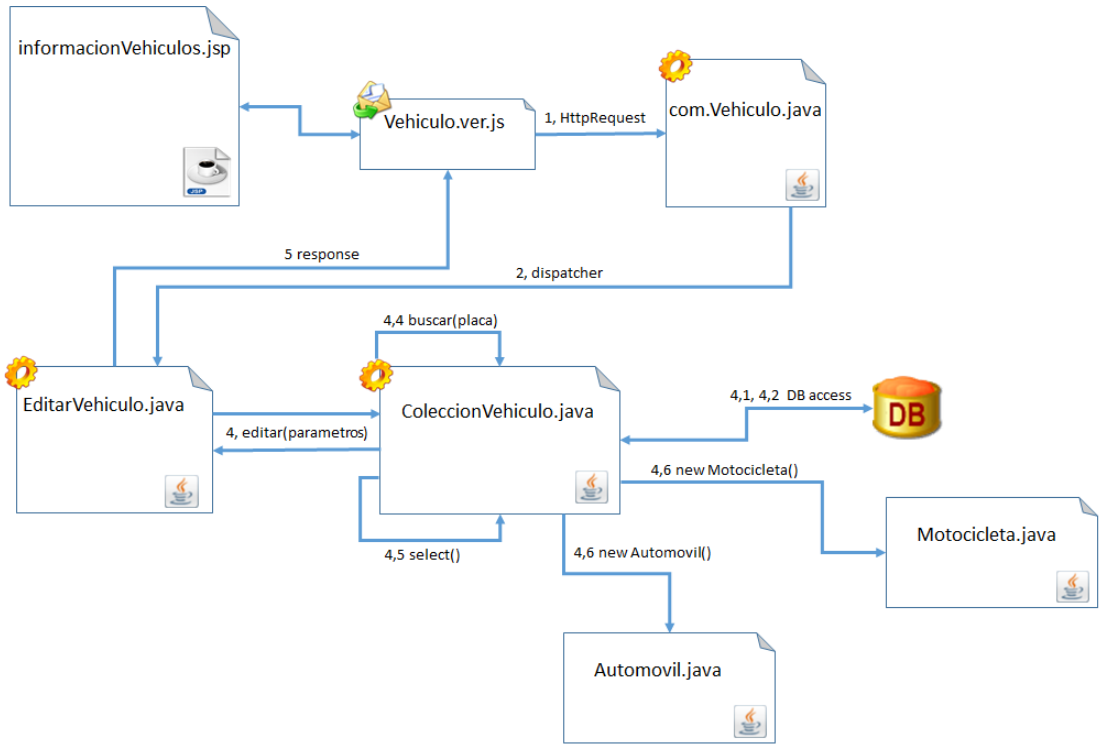


Figura 20 Diagrama de navegación, editar vehículo

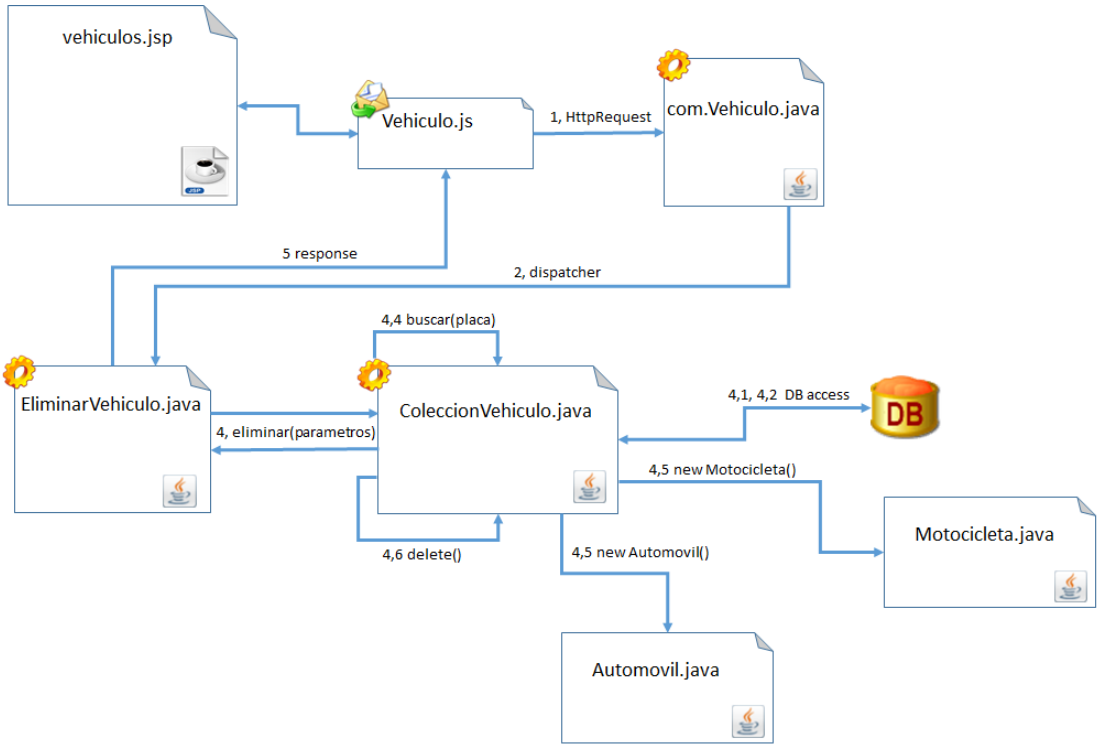


Figura 21 Diagrama de navegación, eliminar vehículo

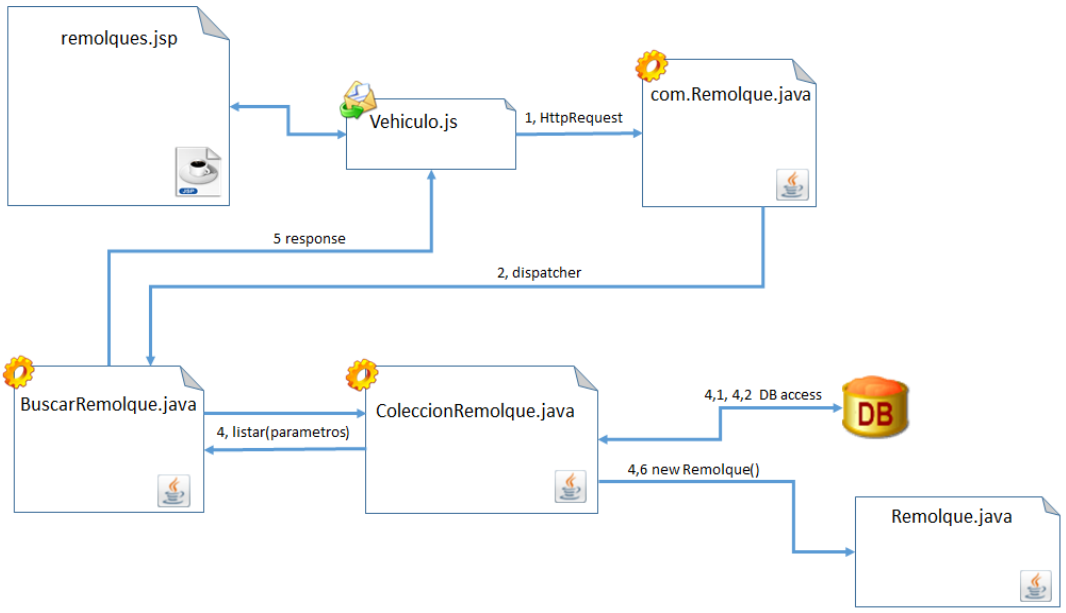


Figura 22 Diagrama de navegación, listar remolques

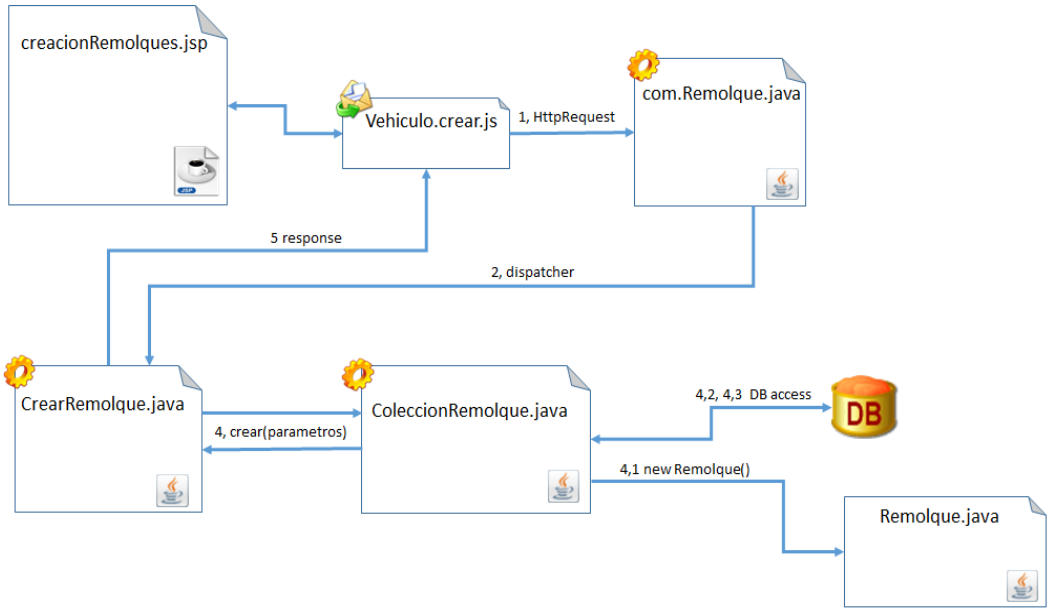


Figura 23 Diagrama de navegación, crear remolque

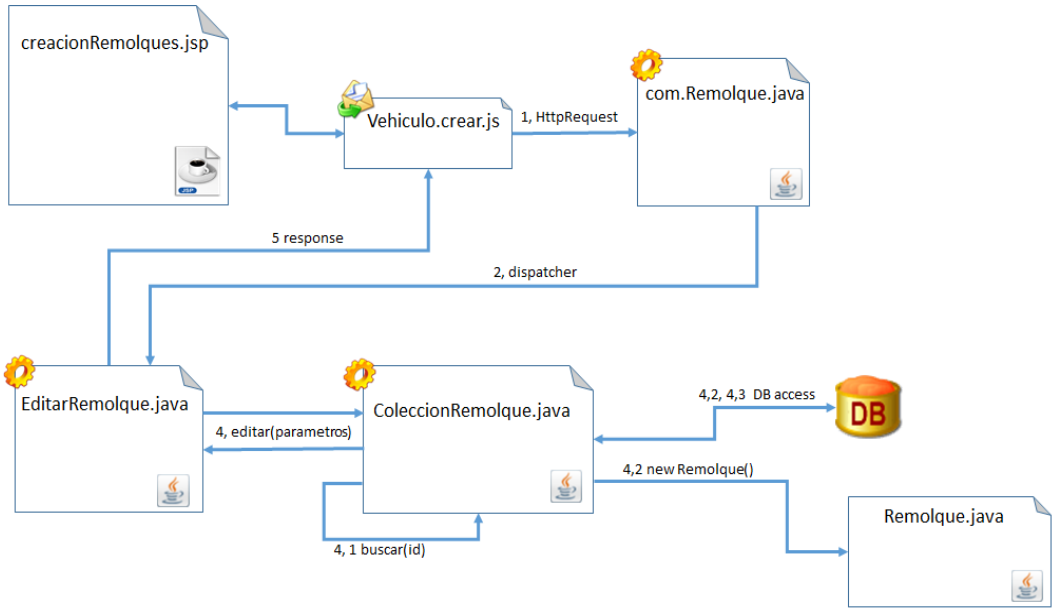


Figura 24 Diagrama de navegación, editar remolque

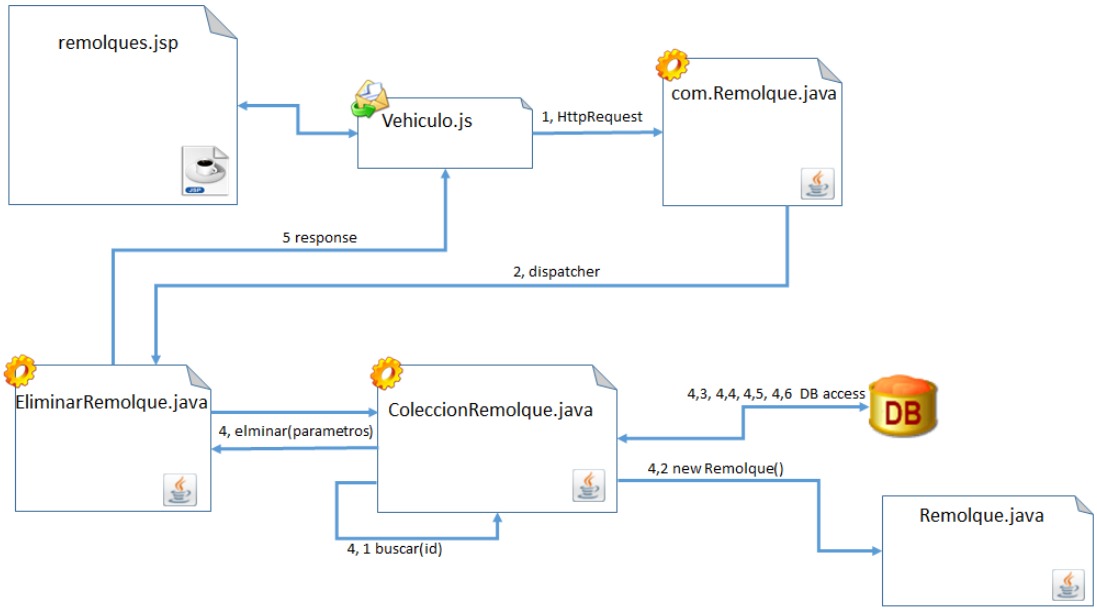


Figura 25 Diagrama de navegación, eliminar remolque

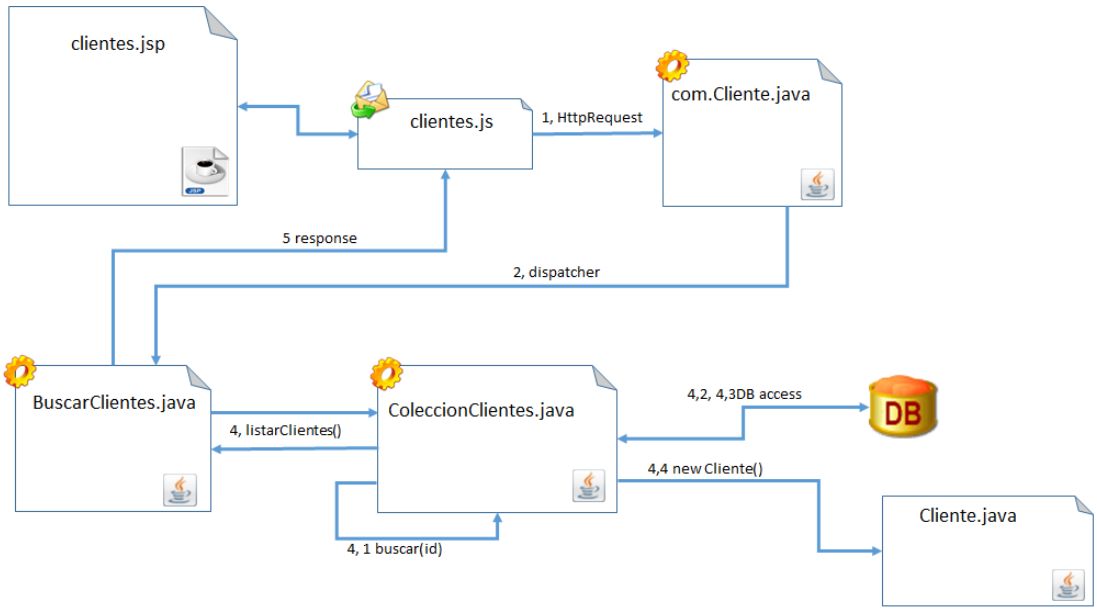


Figura 26 Diagrama de navegación, listar clientes

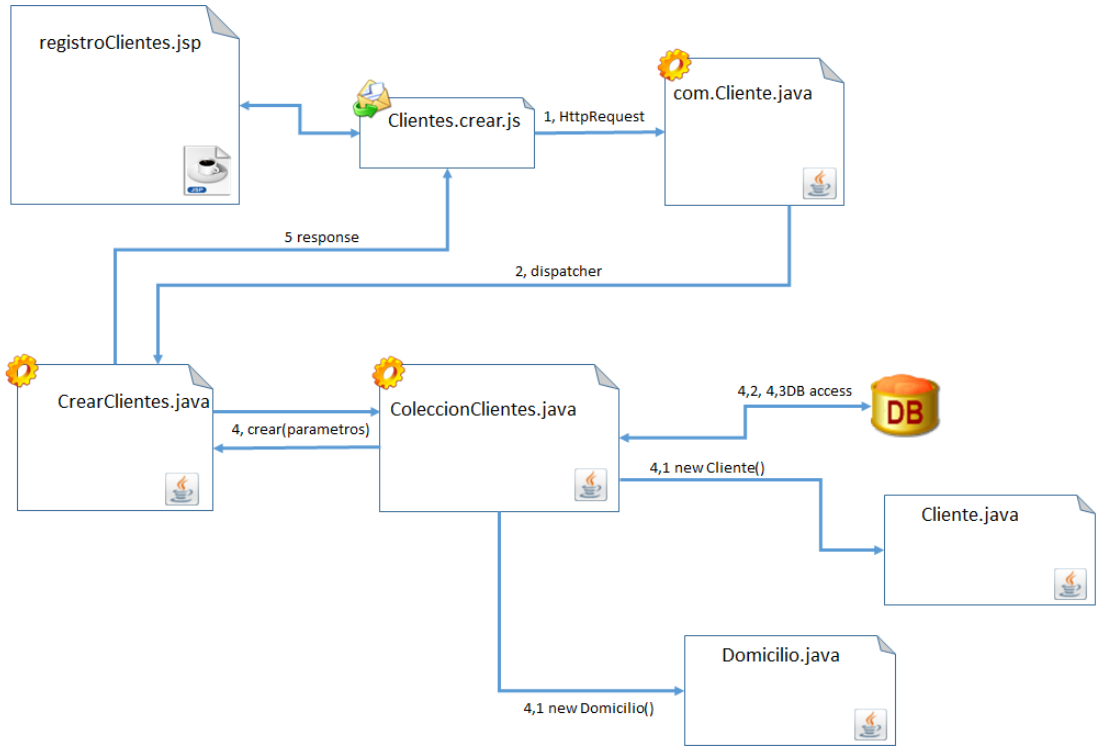


Figura 27 Diagrama de navegación, crear clientes

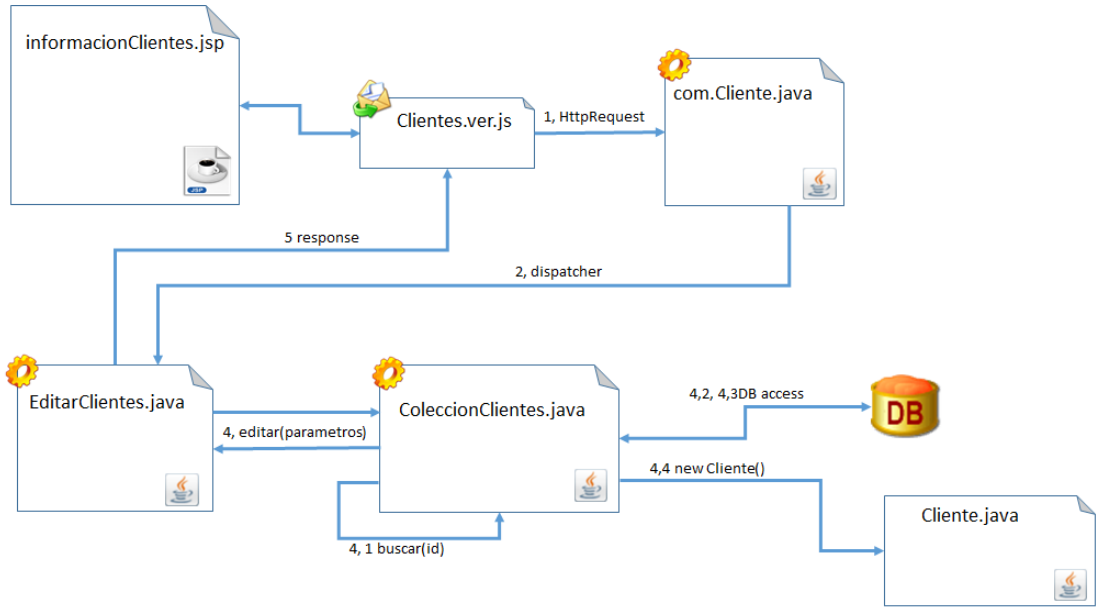


Figura 28 Diagrama de navegación, editar clientes

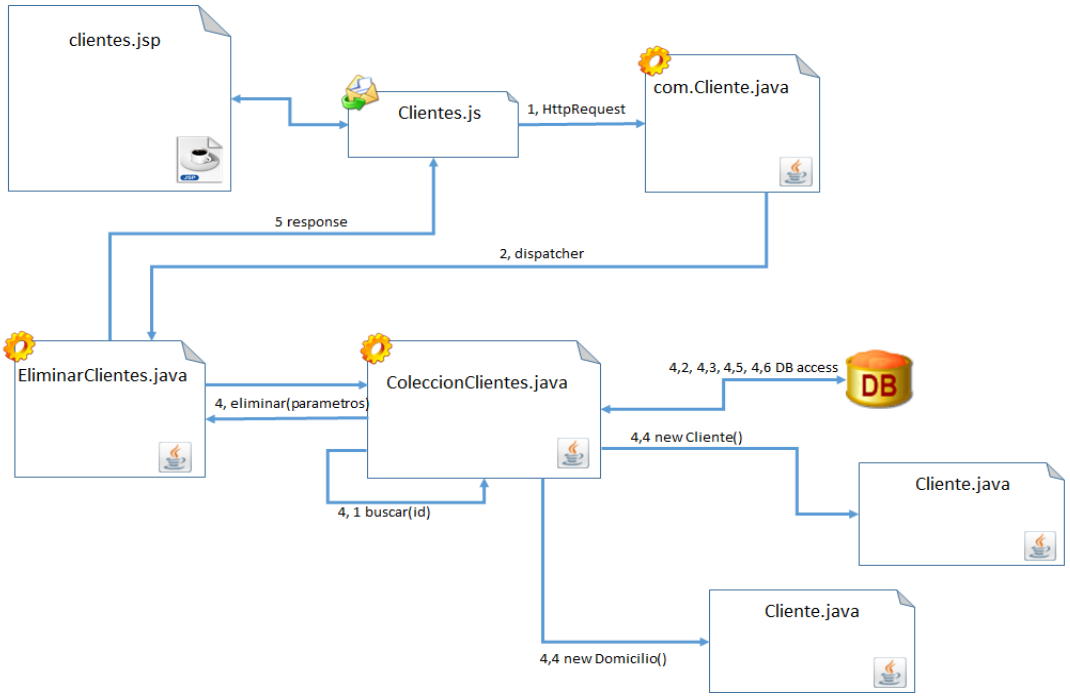


Figura 29 Diagrama de navegación, eliminar clientes

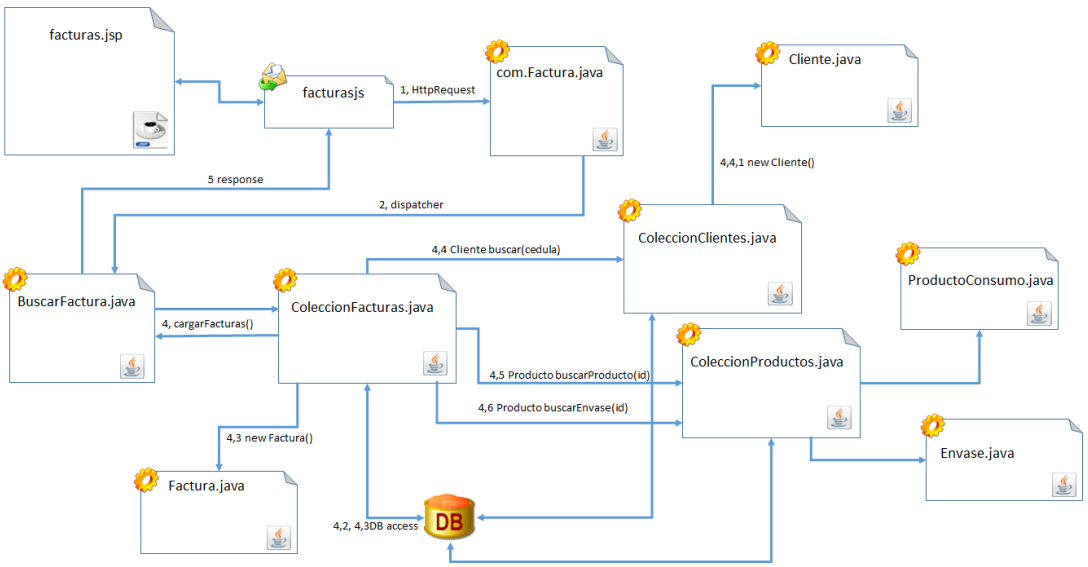


Figura 30 Diagrama de navegación, listar facturas

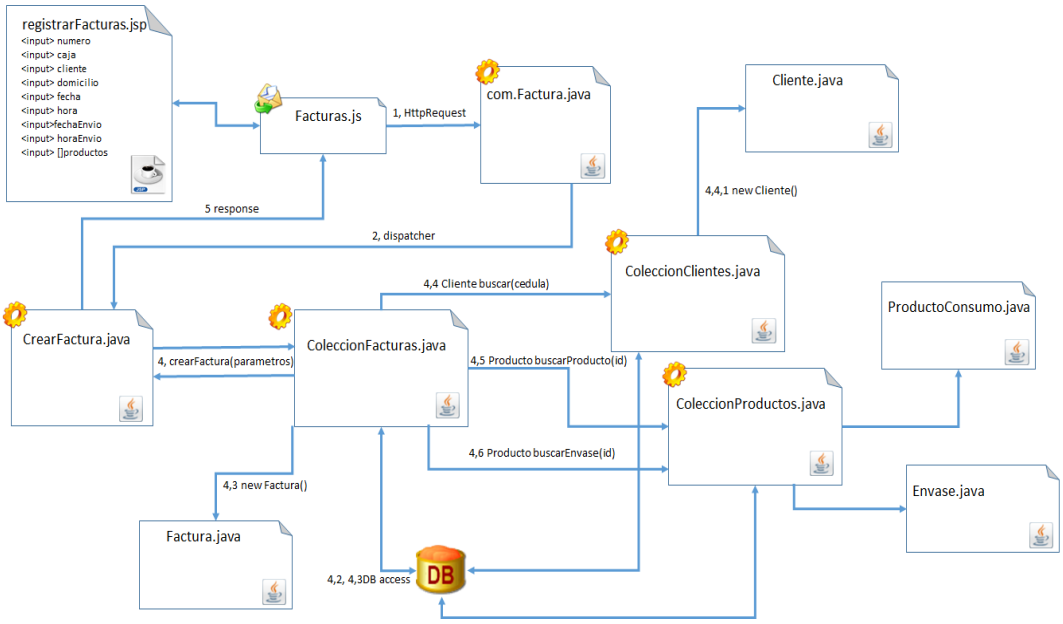


Figura 31 Diagrama de navegación, crear factura

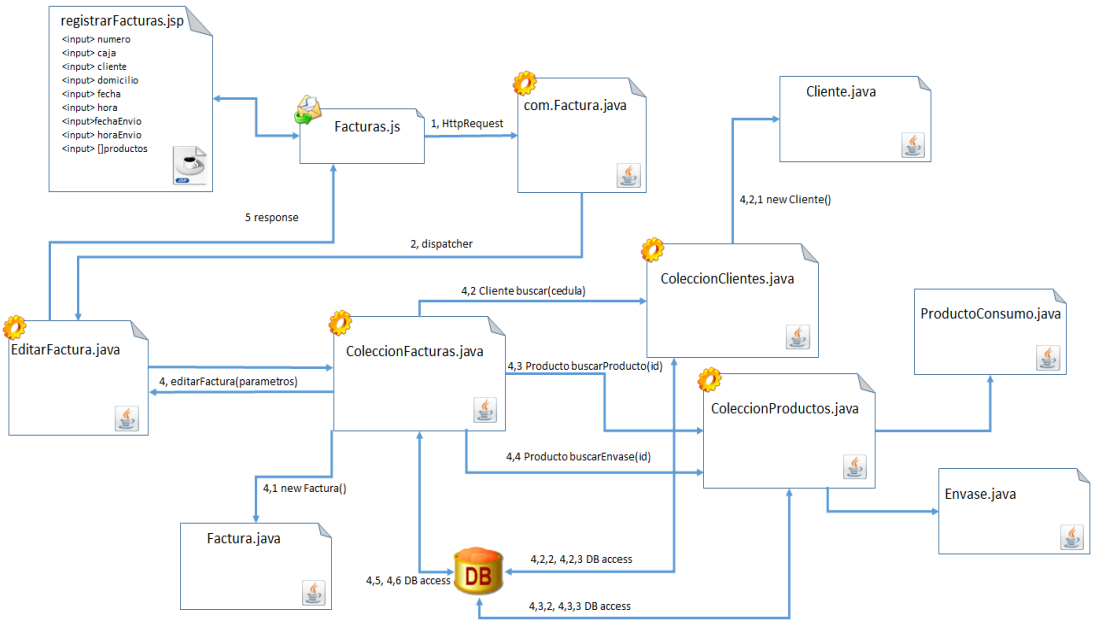


Figura 32 Diagrama de navegación, editar factura

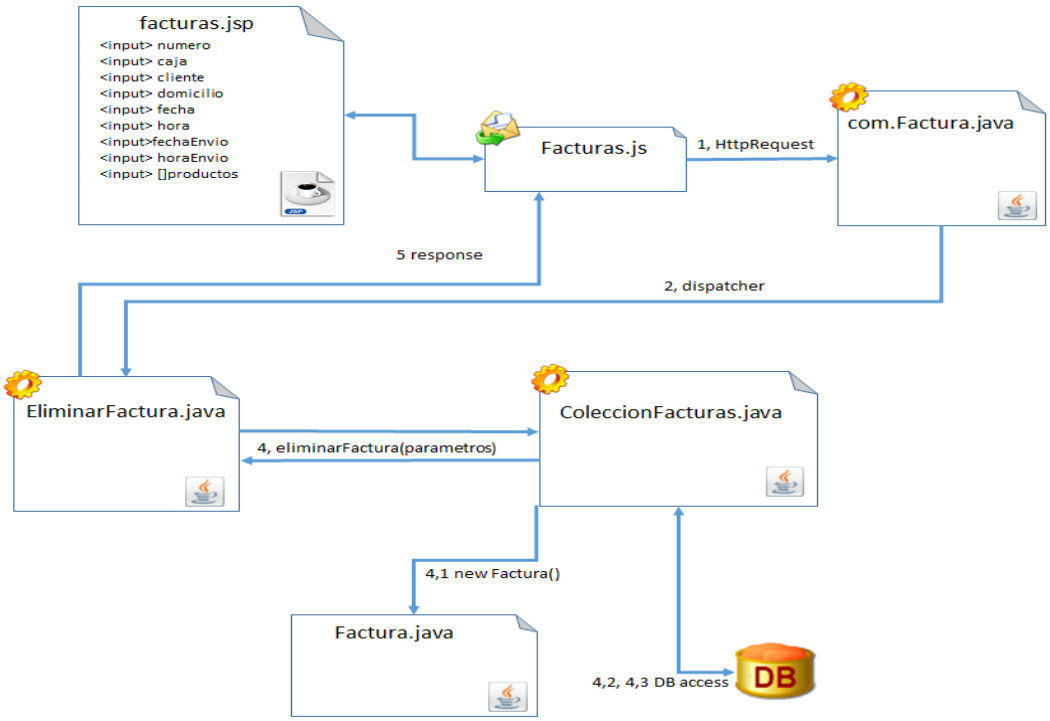


Figura 33 Diagrama de navegación, eliminar factura

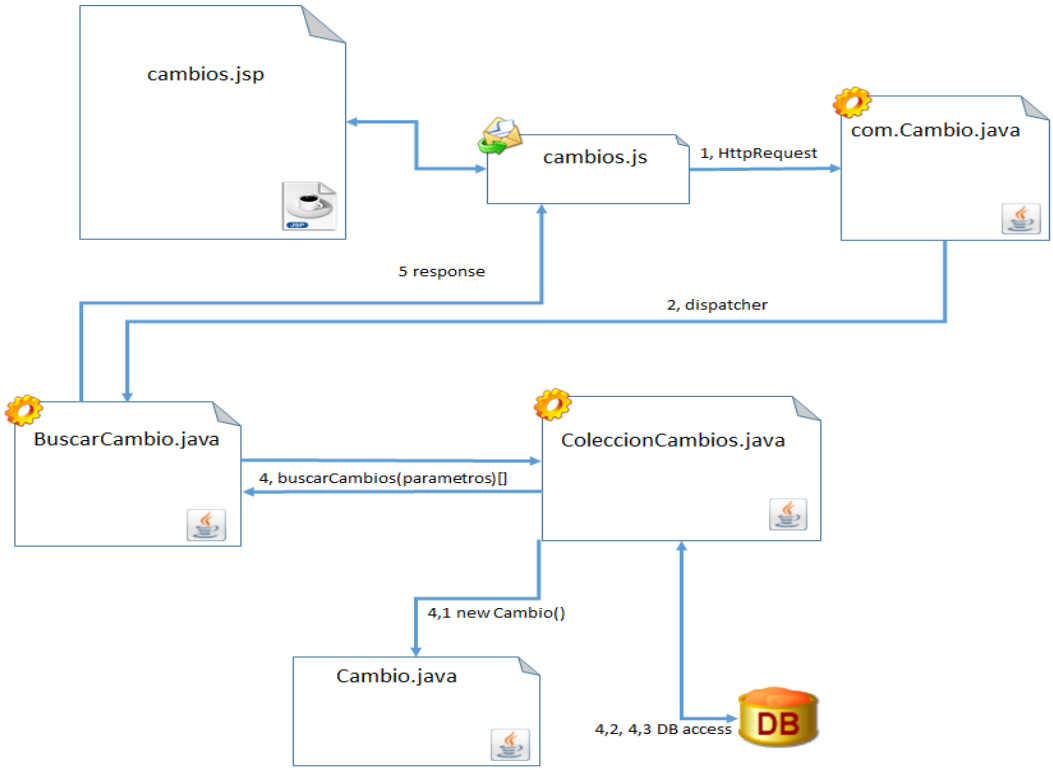


Figura 34 Diagrama de navegación, listar cambios

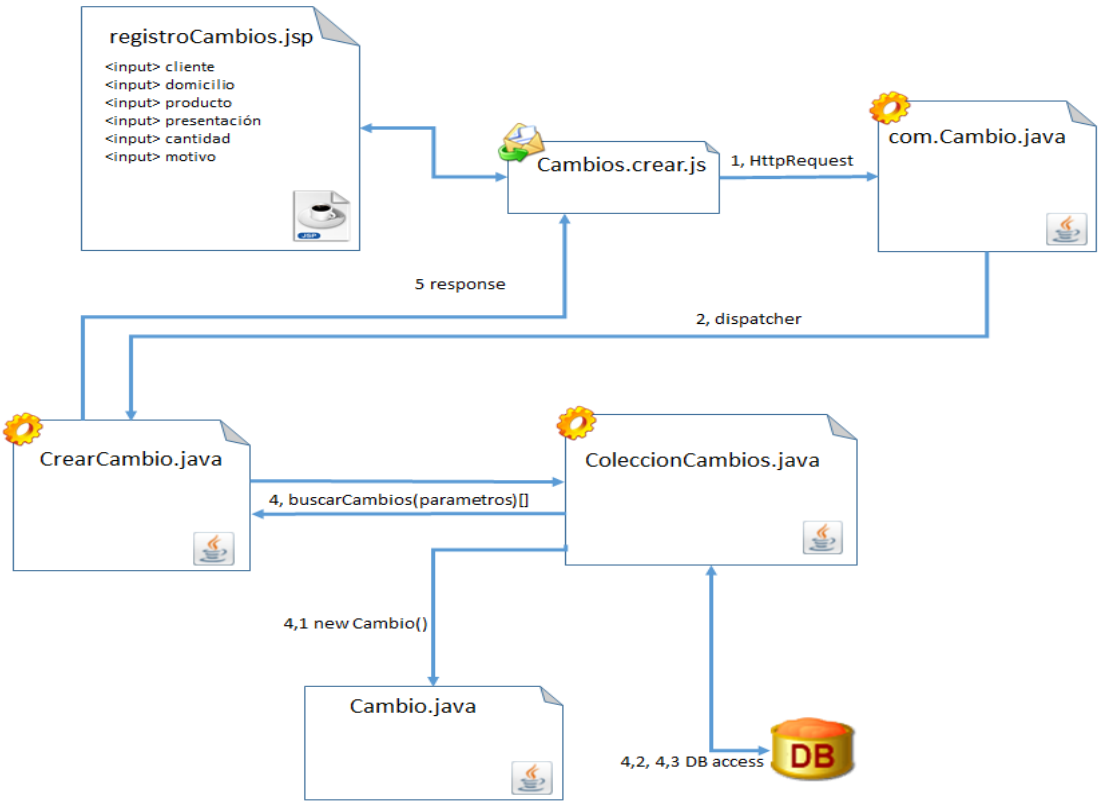


Figura 35 Diagrama de navegación, crear cambios

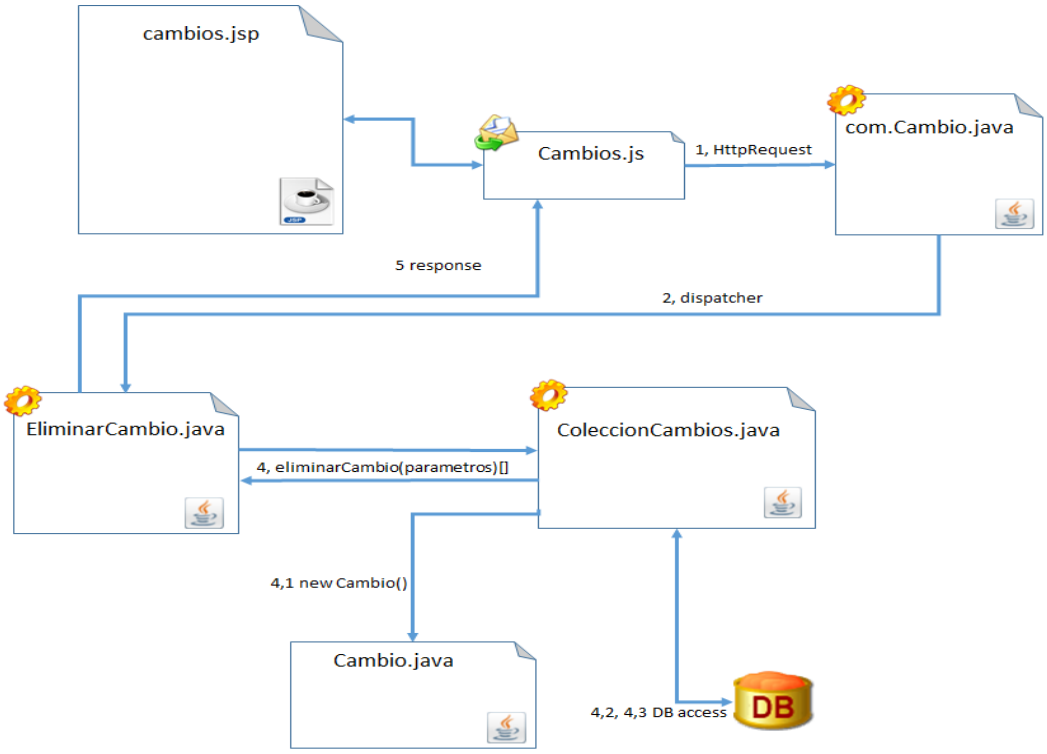


Figura 36 Diagrama de navegación, eliminar cambio

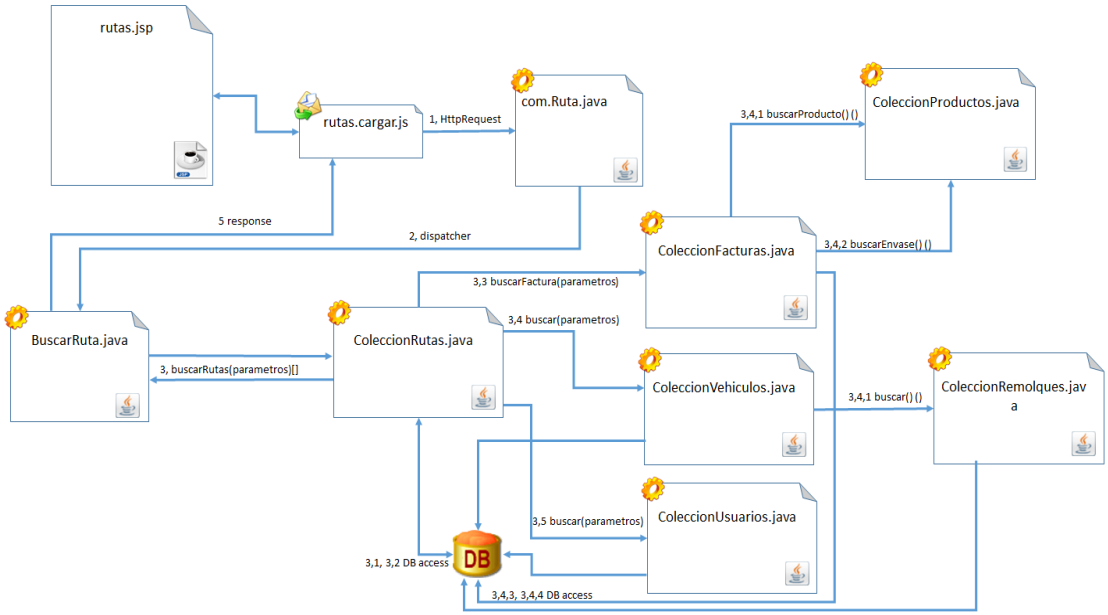


Figura 37 Diagrama de navegación, listar rutas

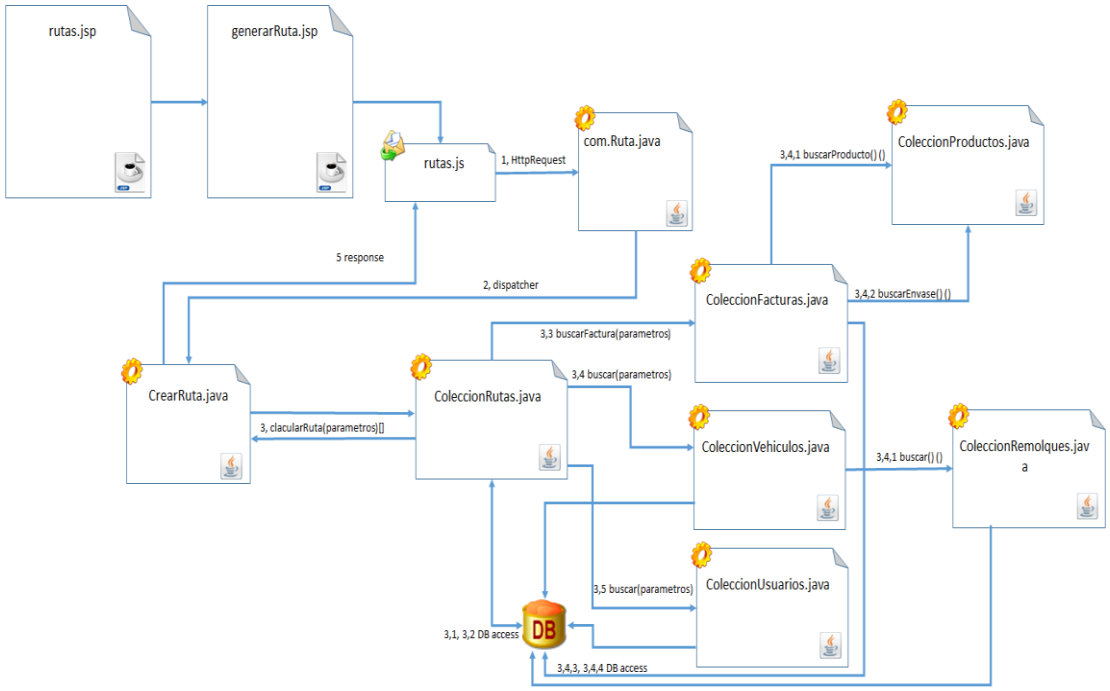


Figura 38 Diagrama de navegación, crear rutas

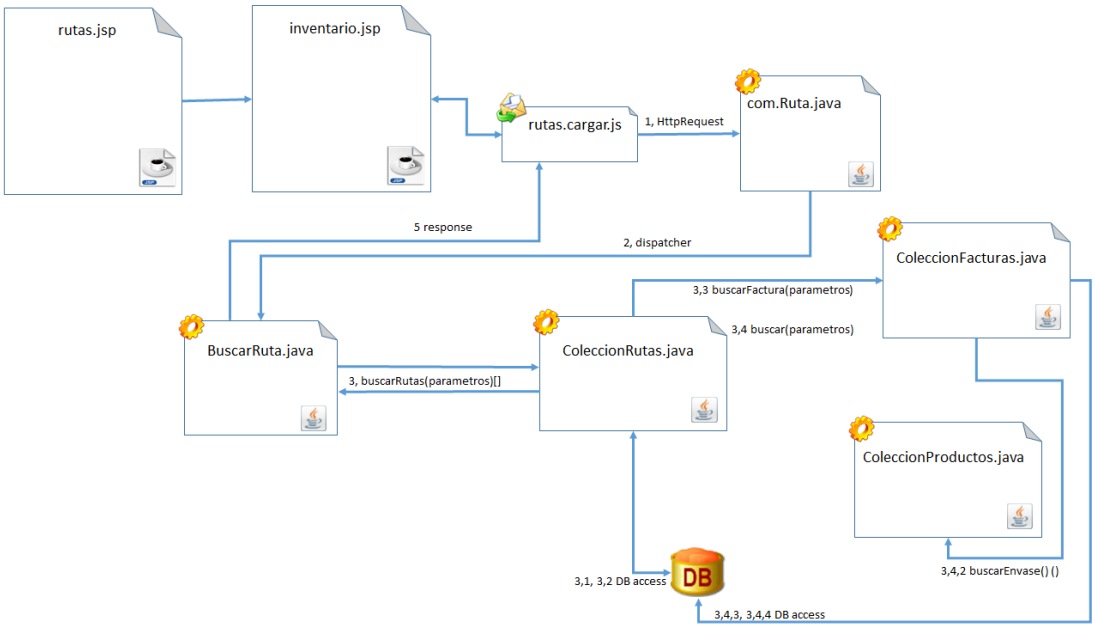


Figura 39 Diagrama de navegación, crear inventario

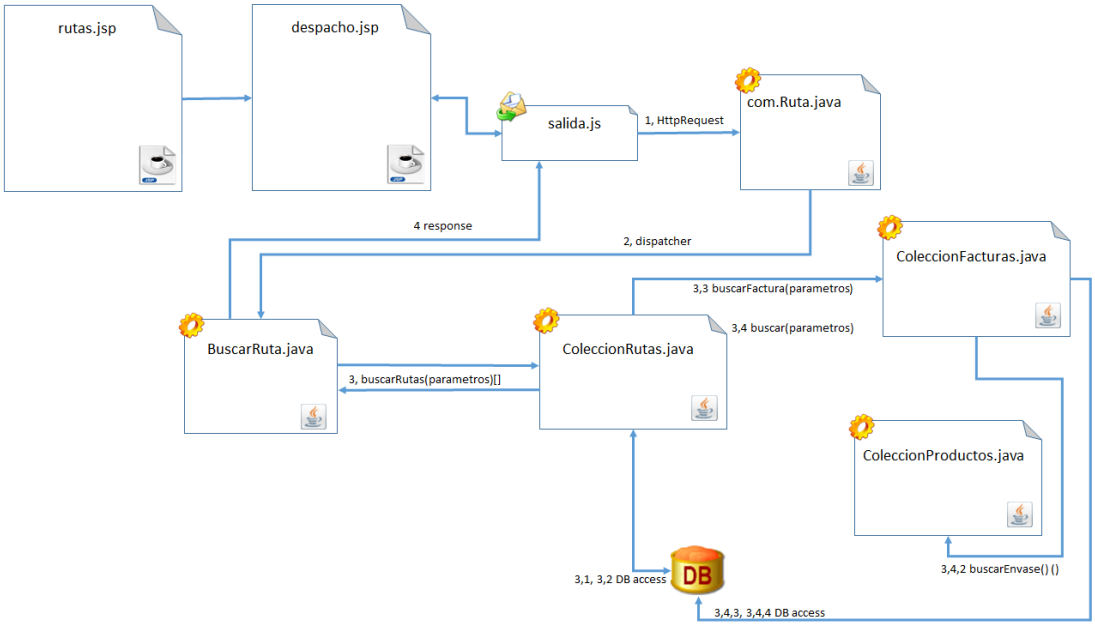


Figura 40 Diagrama de navegación, despacho

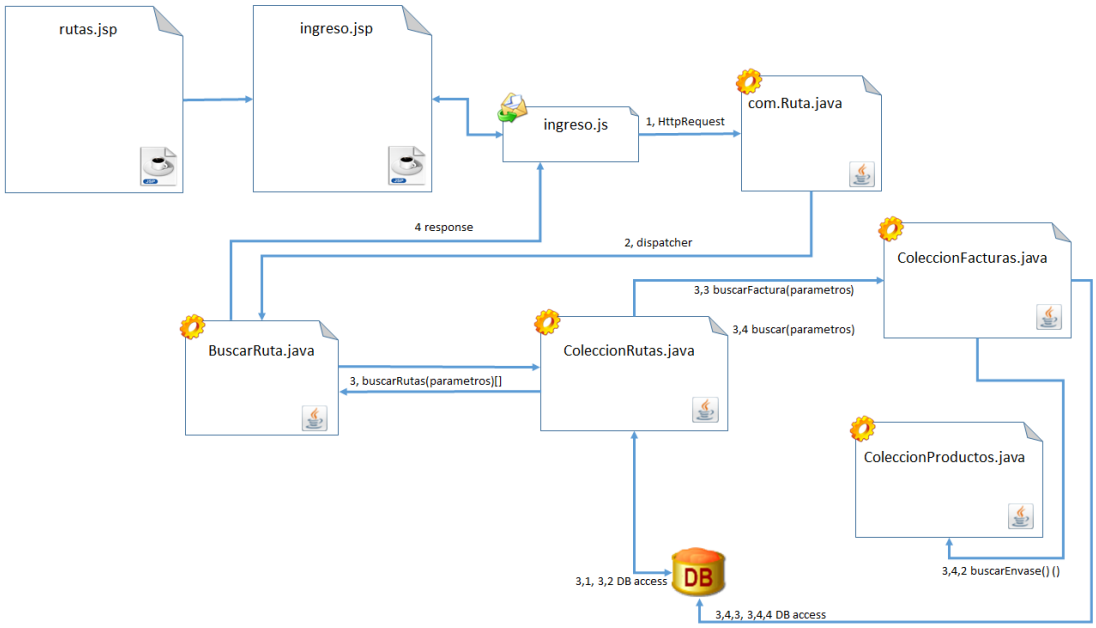
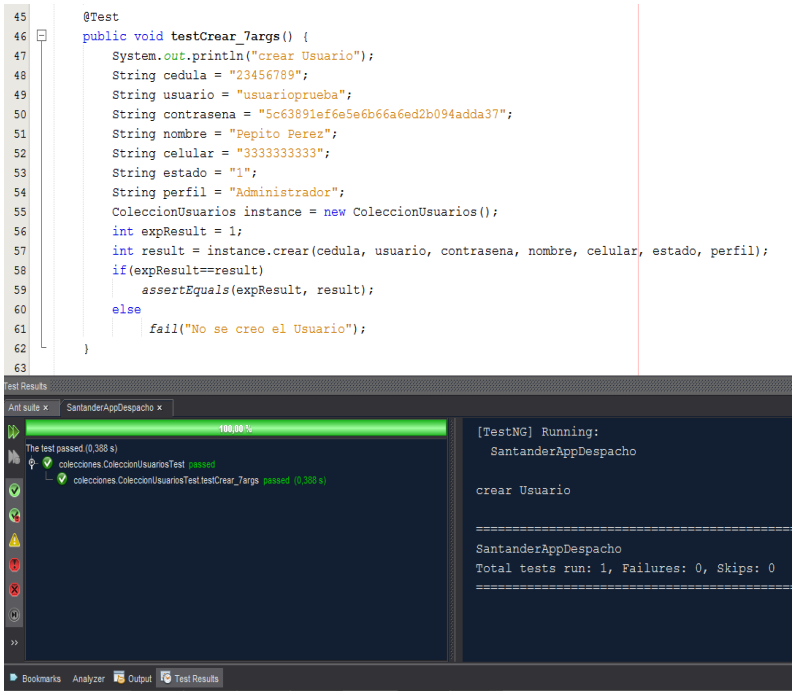


Figura 41 Diagrama de navegación, ingreso

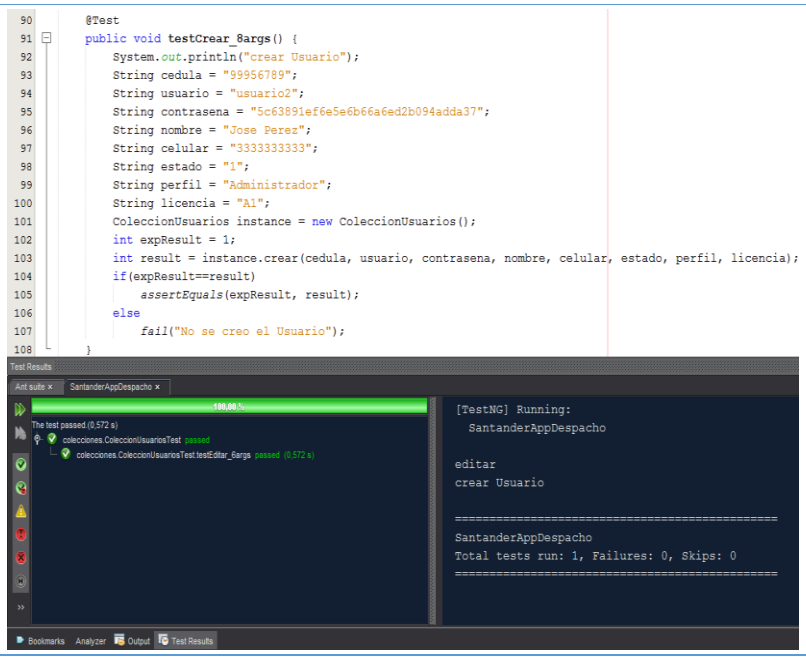
Información General	
Autor:	*Carlos Andrés López Ramírez*
Fecha de creación:	*Septiembre 2 de 2014*
Documentos relacionados:	*Historias de usuario* *Requerimientos*

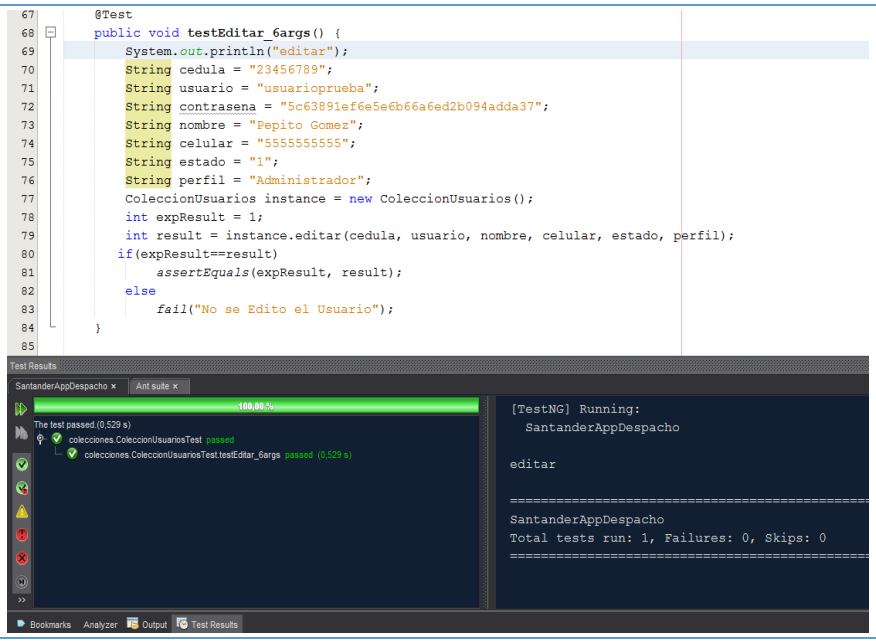
Revisión Histórica	
Revisión No:	*003*
Descripción del cambio	*Se agregaron nuevas pruebas*
Fecha última modificación:	*Diciembre 20 de 2015*

Prueba Unitaria				
Prueba No:	PU0001			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un usuario			
Clase:	Usuarios.java			
Función:	public synchronized int crear			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "23456789"; ➢ String usuario = "usuarioprueba"; ➢ String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37"; ➢ String nombre = "Pepito Perez"; ➢ String celular = "3333333333"; ➢ String estado = "1"; ➢ String perfil = "Administrador";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
45      @Test
46      public void testCrear_7args() {
47          System.out.println("crear Usuario");
48          String cedula = "23456789";
49          String usuario = "usuarioprueba";
50          String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37";
51          String nombre = "Pepito Perez";
52          String celular = "3333333333";
53          String estado = "1";
54          String perfil = "Administrador";
55          ColeccionUsuarios instance = new ColeccionUsuarios();
56          int expResult = 1;
57          int result = instance.crear(cedula, usuario, contrasena, nombre, celular, estado, perfil);
58          if(expResult==result)
59              assertEquals(expResult, result);
60          else
61              fail("No se creo el Usuario");
62      }
63
```

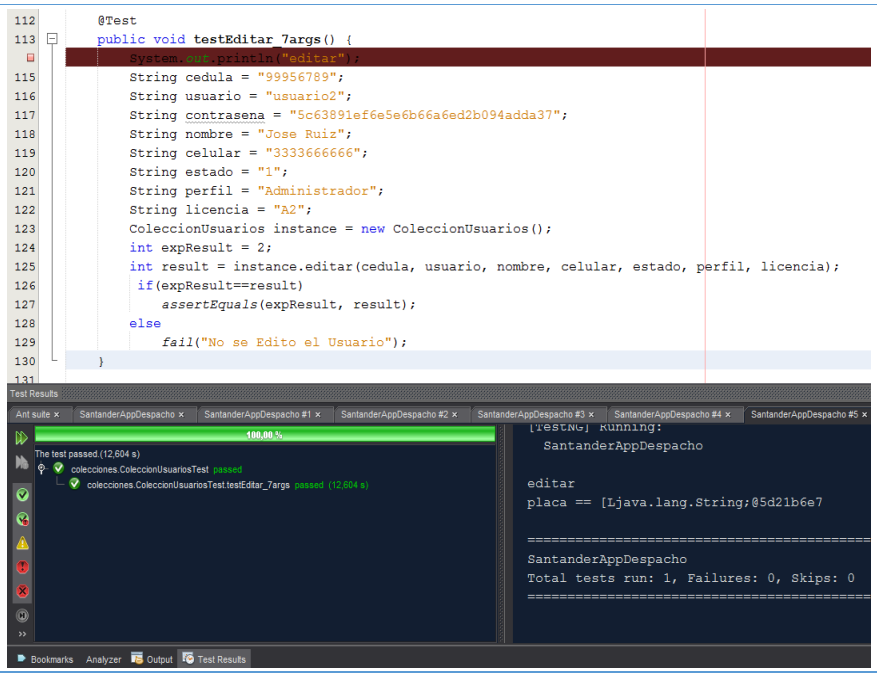
```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed (0,388 s)
colecciones.ColeccionUsuariosTest passed
colecciones.ColeccionUsuariosTest.testCrear_7args passed (0,388 s)

[TestNG] Running:
  SantanderAppDespacho

crear Usuario

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================

Bookmarks  Analyzer  Output  Test Results
```

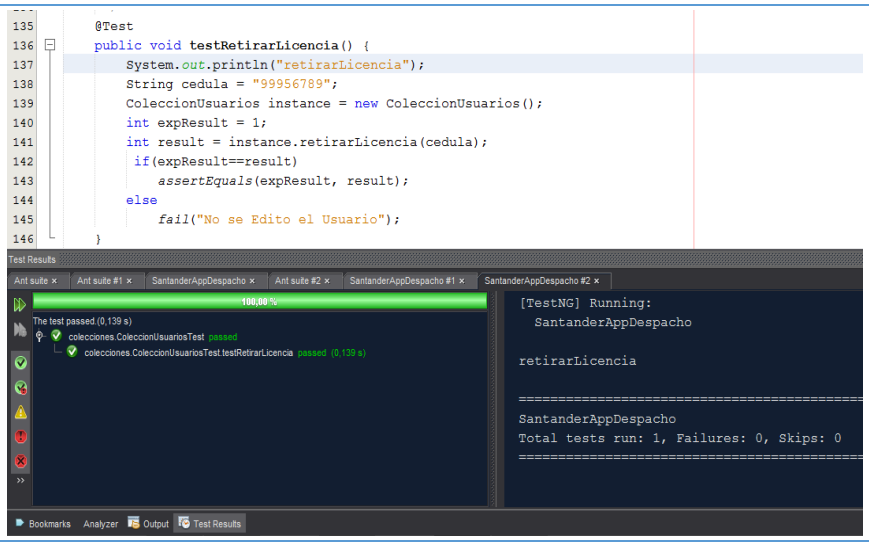
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0002</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Crear Usuario Conductor</td></tr>
<tr><td>Clase:</td><td colspan="4">Usuarios.java</td></tr>
<tr><td>Función:</td><td colspan="4">public synchronized int crear</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String cedula = "99956789";
➢ String usuario = "usuario2";
➢ String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37";
➢ String nombre = "Jose Perez";
➢ String celular = "3333333333";
➢ String estado = "1";
➢ String perfil = "Administrador";
➢ String licencia = "A1";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">1</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">1</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td>Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td>Evidencia:</td><td colspan="4"> 
</td></tr>
</table>

Prueba Unitaria				
Prueba No:	PU0003			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un usuario			
Clase:	ColeccionUsuarios.java			
Función:	public synchronized int editar			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "23456789"; ➢ String usuario = "usuarioprueba"; ➢ String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37"; ➢ String nombre = "Pepito Gomez"; ➢ String celular = "5555555555"; ➢ String estado = "1"; ➢ String perfil = "Administrador";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

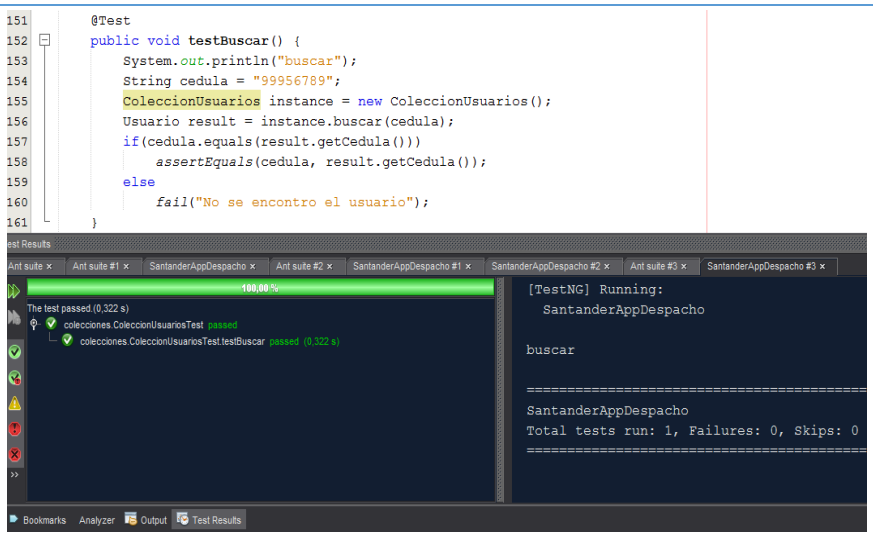
```
67      @Test
68      public void testEditar_6args() {
69          System.out.println("editar");
70          String cedula = "23456789";
71          String usuario = "usuarioprueba";
72          String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37";
73          String nombre = "Pepito Gomez";
74          String celular = "5555555555";
75          String estado = "1";
76          String perfil = "Administrador";
77          ColeccionUsuarios instance = new ColeccionUsuarios();
78          int expResult = 1;
79          int result = instance.editar(cedula, usuario, nombre, celular, estado, perfil);
80         if(expResult==result)
81              assertEquals(expResult, result);
82          else
83              fail("No se Edito el Usuario");
84      }
85
```

```
Test Results
SantanderAppDespacho ×   Ant suite ×
100,00 %
The test passed.(0,529 s)
colecciones.ColeccionUsuariosTest passed
colecciones.ColeccionUsuariosTest.testEditar_6args passed (0,529 s)

[TestNG] Running:
  SantanderAppDespacho

editar

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================

Bookmarks  Analyzer  Output  Test Results
```

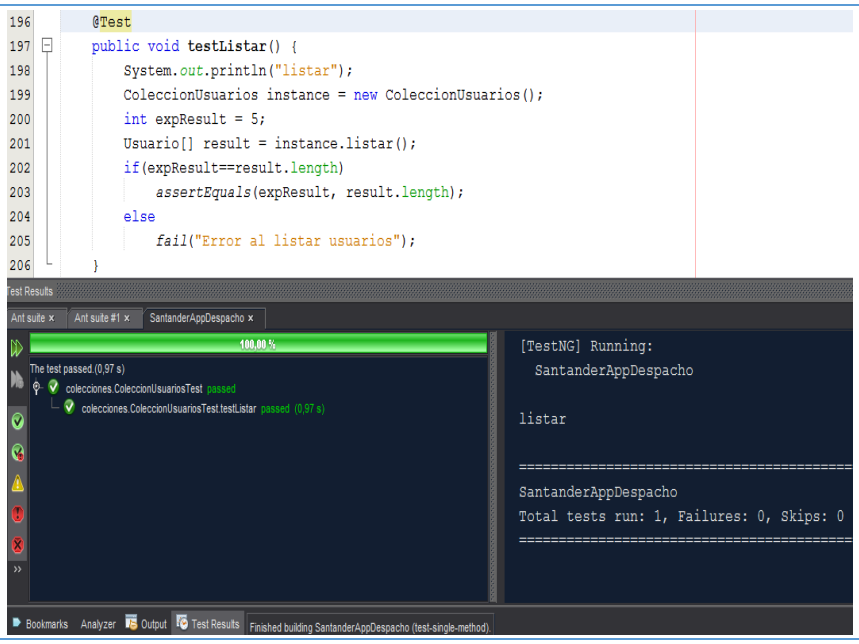
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0004</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Editar un usuario conductor</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionUsuarios.java</td></tr>
<tr><td>Función:</td><td colspan="4">public synchronized int crear(String cedula, String usuario, String contrasena, String nombre, String celular, String estado, int perfil)</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String cedula = "99956789";
➢ String usuario = "usuario2";
➢ String contrasena = "5c63891ef6e5e6b66a6ed2b094adda37";
➢ String nombre = "Jose Ruiz";
➢ String celular = "3333666666";
➢ String estado = "1";
➢ String perfil = "Administrador";
➢ String licencia = "A2";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">Retorna 1 si se creó el usuario, 0 en caso contrario</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">Retorno 1</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td rowspan="2">Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Evidencia:</td><td colspan="4"> 
</td></tr>
</table>

Prueba Unitaria				
Prueba No:	PU0005			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Borrar información de la licencia de un usuario conductor para convertirlo en usuario regular del sistema.			
Clase:	ColeccionUsuarios.java			
Función:	public int retirarLicencia			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "99956789";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

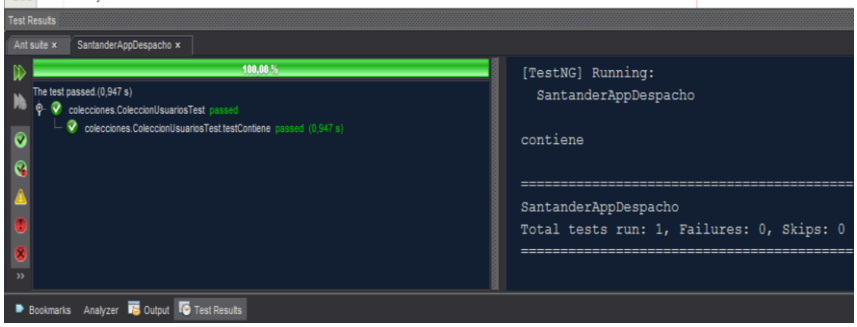
```
135     @Test
136     public void testRetirarLicencia() {
137         System.out.println("retirarLicencia");
138         String cedula = "99956789";
139         ColeccionUsuarios instance = new ColeccionUsuarios();
140         int expResult = 1;
141         int result = instance.retirarLicencia(cedula);
142          if(expResult==result)
143             assertEquals(expResult, result);
144         else
145             fail("No se Edito el Usuario");
146     }
```

Prueba Unitaria				
Prueba No:	PU0006			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un usuario			
Clase:	ColeccionUsuarios.java			
Función:	public Usuario buscar			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "99956789";			
Salida esperada:	Objeto Usuario si se encuentra Null si no			
Salida obtenida:	Objeto Usuario			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

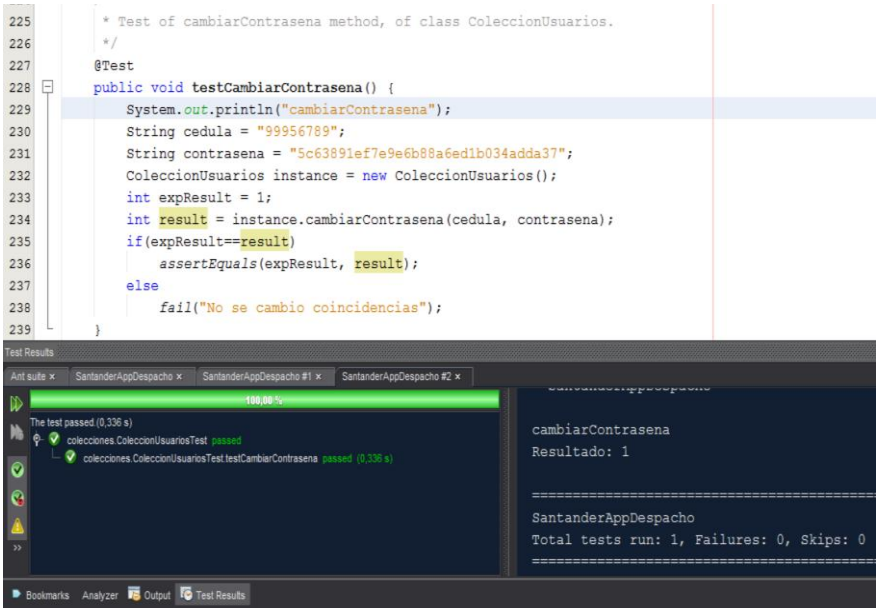
```
151     @Test
152     public void testBuscar() {
153         System.out.println("buscar");
154         String cedula = "99956789";
155         ColeccionUsuarios instance = new ColeccionUsuarios();
156         Usuario result = instance.buscar(cedula);
157         if(cedula.equals(result.getCedula()))
158             assertEquals(cedula, result.getCedula());
159         else
160             fail("No se encontro el usuario");
161     }
```

Prueba Unitaria				
Prueba No:	PU0007			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Actualizar un usuario conductor.			
Clase:	CiloeccionUsuarios.java			
Función:	public synchronized Usuario[] listar			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de usuario			
Salida obtenida:	Lista de usuarios			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
196     @Test
197     public void testListar() {
198         System.out.println("listar");
199         ColeccionUsuarios instance = new ColeccionUsuarios();
200         int expResult = 5;
201         Usuario[] result = instance.listar();
202         if(expResult==result.length)
203             assertEquals(expResult, result.length);
204         else
205             fail("Error al listar usuarios");
206     }
```

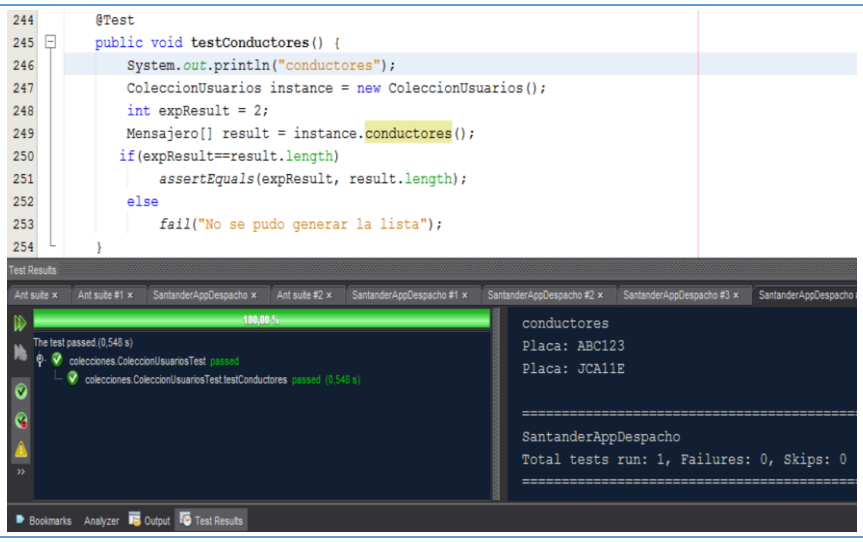
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0008</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Buscar un usuario por algun dato parcial</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionUsuarios.java</td></tr>
<tr><td>Función:</td><td colspan="4">public synchronized Usuario[] contiene</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String dato = "rio2";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">Un arreglo con 1 usuario</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">Un arreglo con 1 usuario</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td rowspan="2">Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Evidencia:</td><td colspan="4">
<pre>
@Test
public void testContiene() {
    System.out.println("contiene");
    String dato = "rio2";
    ColeccionUsuarios instance = new ColeccionUsuarios();
    int expResult = 1;
    Usuario[] result = instance.contiene(dato);
    if(expResult==result.length)
        assertEquals(expResult, result.length);
    else
        fail("No se encontraron coincidencias");
}
</pre>


</td></tr>
</table>

Prueba Unitaria				
Prueba No:	PU0009			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Cambiar la contraseña de un usuario			
Clase:	ColeccionUsuarios.java			
Función:	public int cambiarContrasena			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "99956789"; ➢ String contrasena = "5c63891ef7e9e6b88a6ed1b034adda37";			
Salida esperada:	1, indica que la contraseña cambio			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

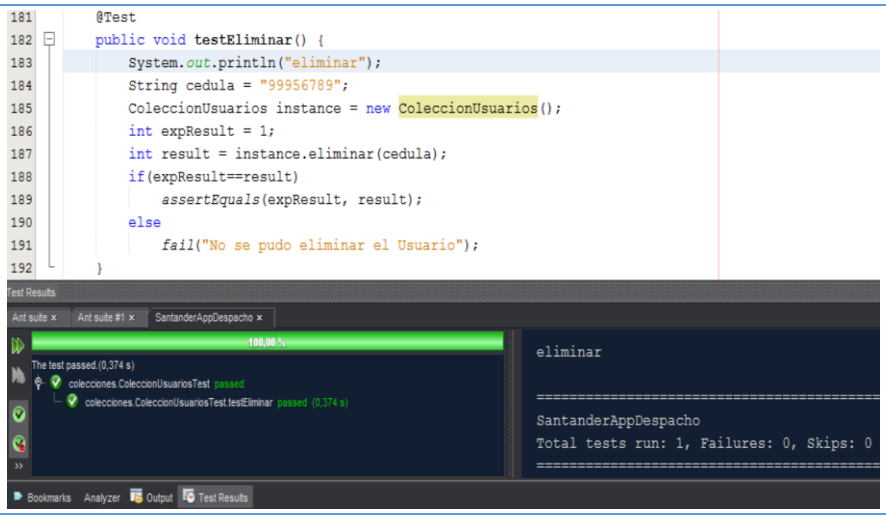
```
225      * Test of cambiarContrasena method, of class ColeccionUsuarios.
226      */
227     @Test
228     public void testCambiarContrasena() {
229         System.out.println("cambiarContrasena");
230         String cedula = "99956789";
231         String contrasena = "5c63891ef7e9e6b88a6ed1b034adda37";
232         ColeccionUsuarios instance = new ColeccionUsuarios();
233         int expResult = 1;
234         int result = instance.cambiarContrasena(cedula, contrasena);
235         if(expResult==result)
236             assertEquals(expResult, result);
237         else
238             fail("No se cambio coincidencias");
239     }
```

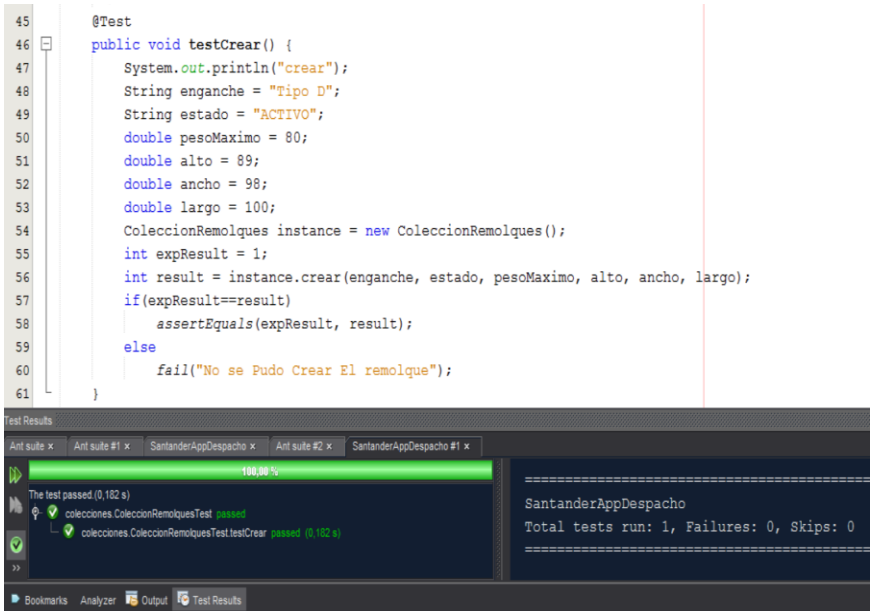
```
Test Results
Ant suite ×  SantanderAppDespacho ×  SantanderAppDespacho #1 ×  SantanderAppDespacho #2 ×
100,00 %
The test passed (0,336 s)
colecciones.ColeccionUsuariosTest passed
colecciones.ColeccionUsuariosTest.testCambiarContrasena passed (0,336 s)

cambiarContrasena
Resultado: 1

SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
Bookmarks  Analyzer  Output  Test Results
```

Prueba Unitaria				
Prueba No:	PU0010			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Generar una lista de conductores			
Clase:	ColeccionUsuarios.java			
Función:	public Mensajero[] conductores			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de conductores de longitud 2			
Salida obtenida:	Lista de conductores de longitud 2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

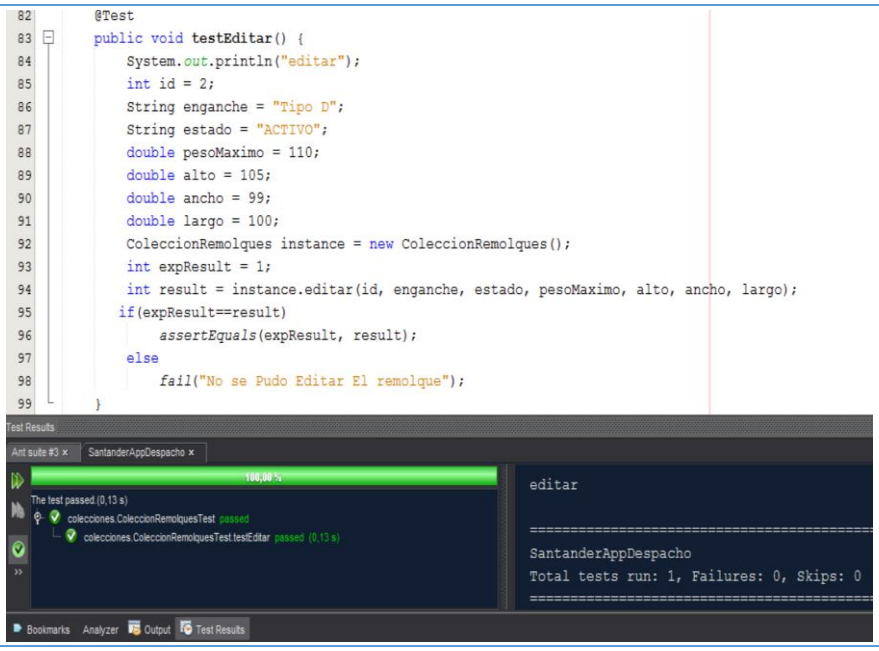
```
244     @Test
245     public void testConductores() {
246         System.out.println("conductores");
247         ColeccionUsuarios instance = new ColeccionUsuarios();
248         int expResult = 2;
249         Mensajero[] result = instance.conductores();
250        if(expResult==result.length)
251             assertEquals(expResult, result.length);
252         else
253             fail("No se pudo generar la lista");
254     }
```

```
Test Results
100,00 %
The test passed.(0,548 s)
colecciones.ColeccionUsuariosTest passed
colecciones.ColeccionUsuariosTest.testConductores passed (0,548 s)

conductores
Placa: ABC123
Placa: JCA11E

SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
```

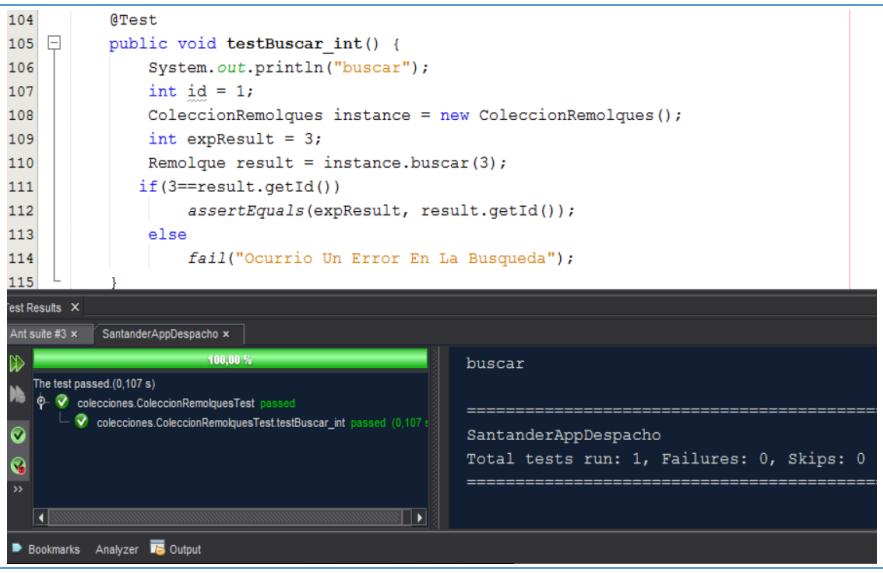
Prueba Unitaria				
Prueba No:	PU0011			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Eliminar un usuario			
Clase:	ColeccionUsuarios.java			
Función:	public synchronized int eliminar			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "99956789";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

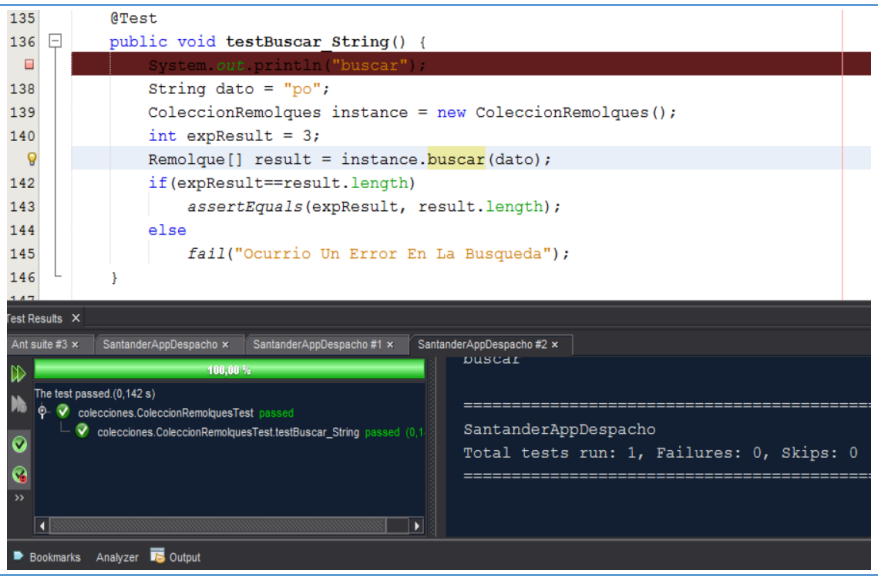
Prueba Unitaria				
Prueba No:	PU0012			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un remolque			
Clase:	ColeccionRemolques.java			
Función:	public int crear			
Configuraciones:				
Datos de Entrada:	➢ String enganche = "Tipo D"; ➢ String estado = "ACTIVO"; ➢ double pesoMaximo = 80; ➢ double alto = 89; ➢ double ancho = 98; ➢ double largo = 100;			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

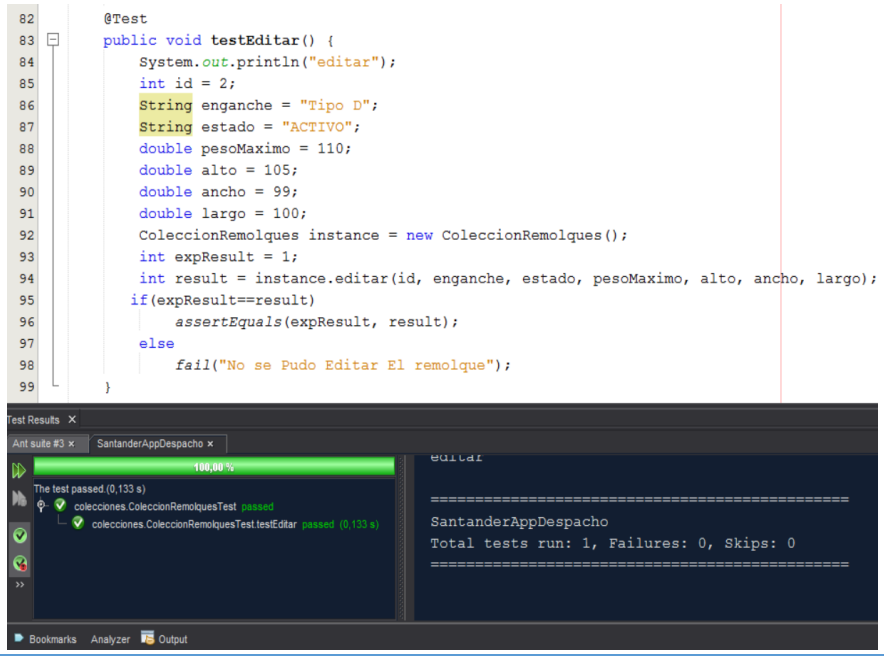
```
45      @Test
46      public void testCrear() {
47          System.out.println("crear");
48          String enganche = "Tipo D";
49          String estado = "ACTIVO";
50          double pesoMaximo = 80;
51          double alto = 89;
52          double ancho = 98;
53          double largo = 100;
54          ColeccionRemolques instance = new ColeccionRemolques();
55          int expResult = 1;
56          int result = instance.crear(enganche, estado, pesoMaximo, alto, ancho, largo);
57          if(expResult==result)
58              assertEquals(expResult, result);
59          else
60              fail("No se Pudo Crear El remolque");
61      }
```

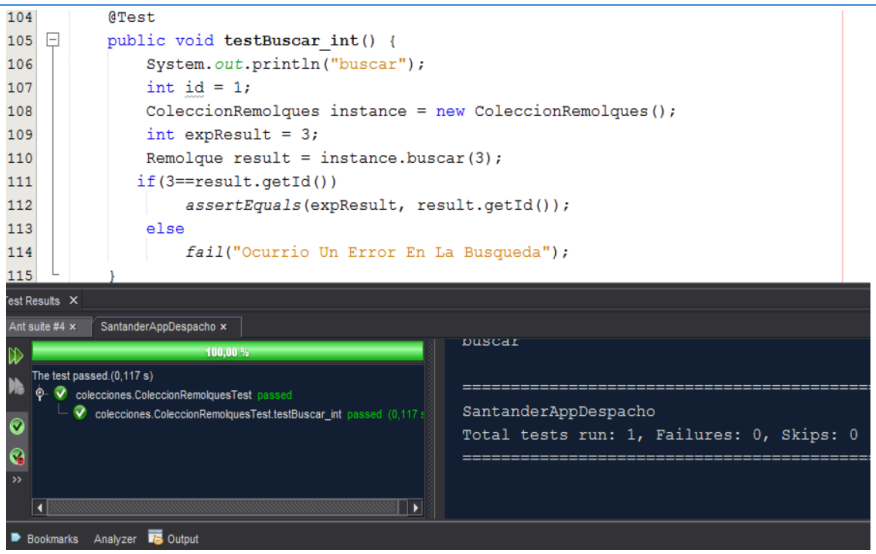
Prueba Unitaria				
Prueba No:	PU0013			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un remolque			
Clase:	ColeccionRemolques.java			
Función:	public synchronized Usuario[] contiene(String dato)			
Configuraciones:				
Datos de Entrada:	➢ String dato: información parcial del usuario, cedula, nombre, usuario o celular			
Salida esperada:	1, indica que el remolque fue editado.			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
82      @Test
83      public void testEditar() {
84          System.out.println("editar");
85          int id = 2;
86          String enganche = "Tipo D";
87          String estado = "ACTIVO";
88          double pesoMaximo = 110;
89          double alto = 105;
90          double ancho = 99;
91          double largo = 100;
92          ColeccionRemolques instance = new ColeccionRemolques();
93          int expResult = 1;
94          int result = instance.editar(id, enganche, estado, pesoMaximo, alto, ancho, largo);
95         if(expResult==result)
96              assertEquals(expResult, result);
97          else
98              fail("No se Pudo Editar El remolque");
99      }
```

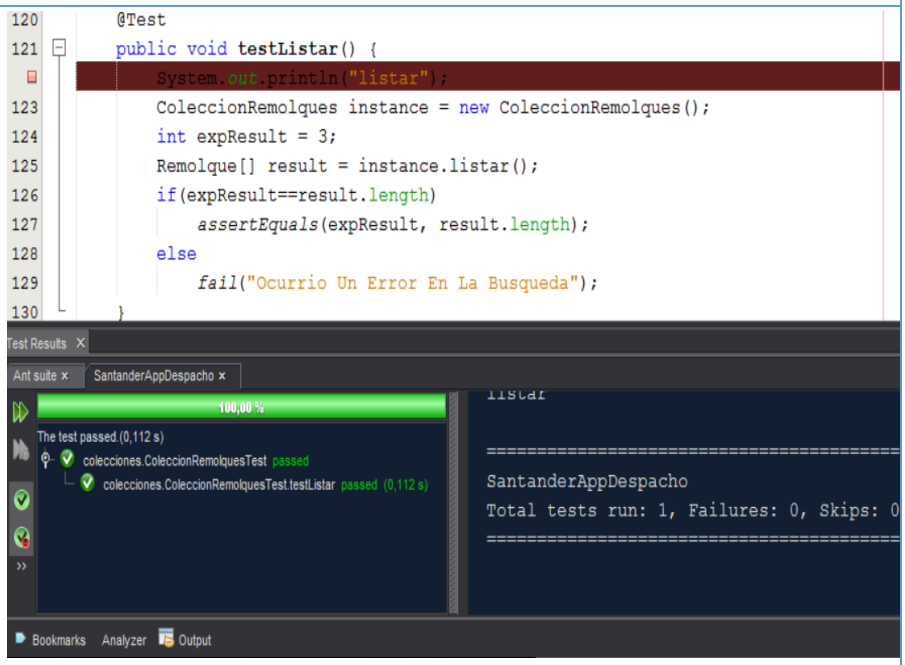
Prueba Unitaria				
Prueba No:	PU0014			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un remolque			
Clase:	ColeccionRemolques.java			
Función:	public Remolque buscar			
Configuraciones:				
Datos de Entrada:	➢ int id = 1;			
Salida esperada:	Un objeto Remolque, Null si no se encuentra			
Salida obtenida:	Objeto Remolque			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0015			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Bscar un remolque por un dato parcial			
Clase:	ColeccionRemolques.java			
Función:	public Remolque[] buscar			
Configuraciones:				
Datos de Entrada:	➢ tring dato = "po";			
Salida esperada:	Una lista de objetos Remolque			
Salida obtenida:	Una lista de objetos Remolque			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

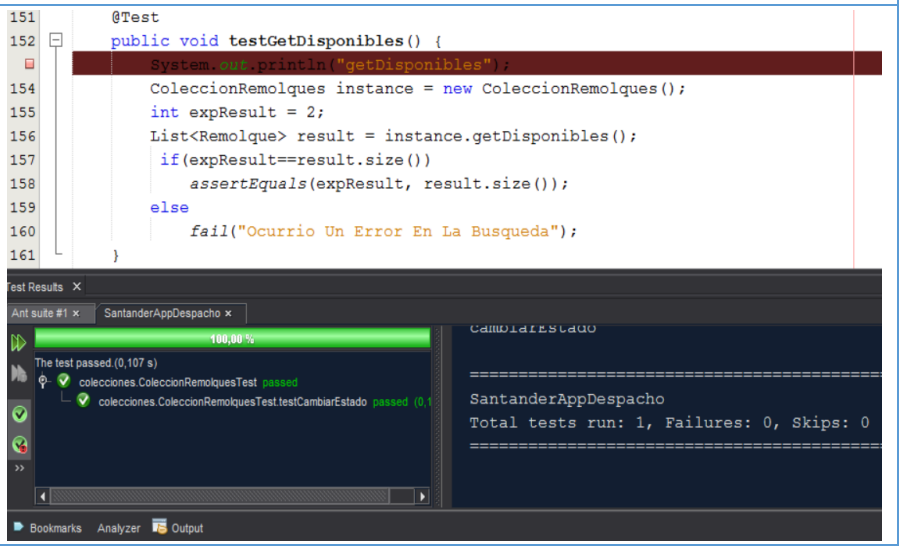
Prueba Unitaria				
Prueba No:	PU0016			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un remolque			
Clase:	ColeccionRemolques.java			
Función:	public Mensajero[] conductores()			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Una lista de objetos Mensajero			
Salida obtenida:	Una lista de objetos Mensajero			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

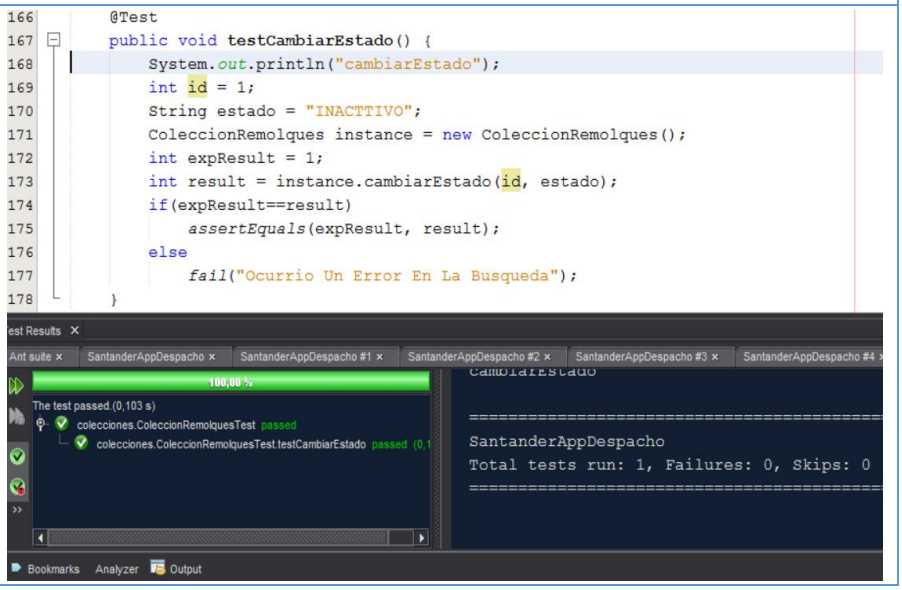
Prueba Unitaria				
Prueba No:	PU0017			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un remolque por el id.			
Clase:	ColeccionRemolquesjava			
Función:	public int addVehiculo(String cedula, String placa)			
Configuraciones:				
Datos de Entrada:	➢ int id = 1;			
Salida esperada:	Objeto Remolque, Null si no existe.			
Salida obtenida:	Objeto Remolque.			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
104     @Test
105     public void testBuscar_int() {
106         System.out.println("buscar");
107         int id = 1;
108         ColeccionRemolques instance = new ColeccionRemolques();
109         int expResult = 3;
110         Remolque result = instance.buscar(3);
111        if(3==result.getId())
112             assertEquals(expResult, result.getId());
113         else
114             fail("Ocurrio Un Error En La Busqueda");
115     }
```

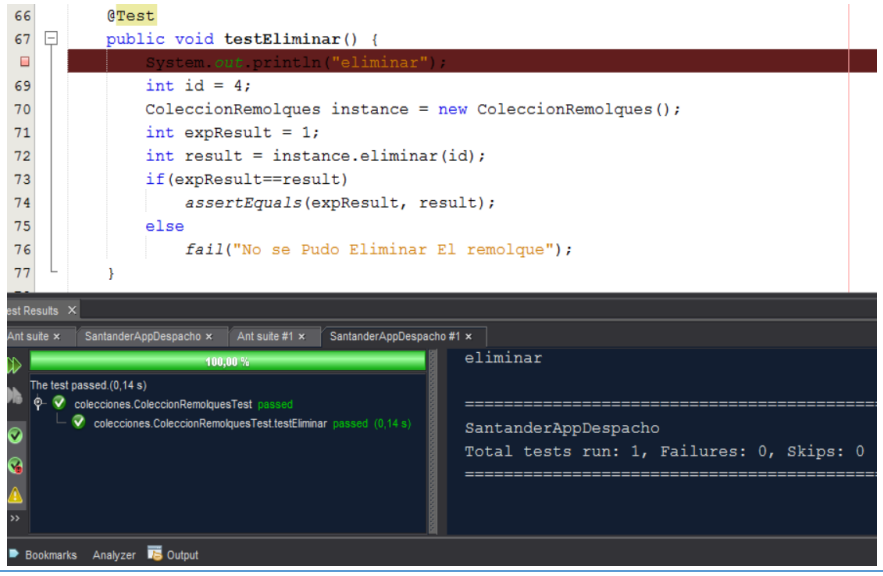
Prueba Unitaria				
Prueba No:	PU0018			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Asignar un vehículo a un usuario mensajero			
Clase:	ColeccionRemolques.java			
Función:	public Remolque[] listar			
Configuraciones :				
Datos de Entrada:				
Salida esperada:	Array de remolque de longitud 3			
Salida obtenida:	Array de remolque de longitud 3			
Estado:	Correcto :	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

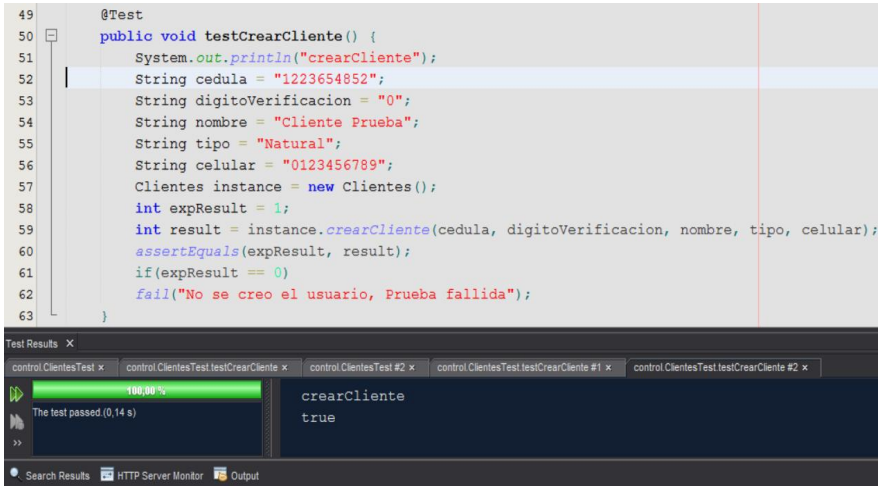
```
120     @Test
121     public void testListar() {
            System.out.println("listar");
123         ColeccionRemolques instance = new ColeccionRemolques();
124         int expResult = 3;
125         Remolque[] result = instance.listar();
126         if(expResult==result.length)
127             assertEquals(expResult, result.length);
128         else
129             fail("Ocurrio Un Error En La Busqueda");
130     }
```

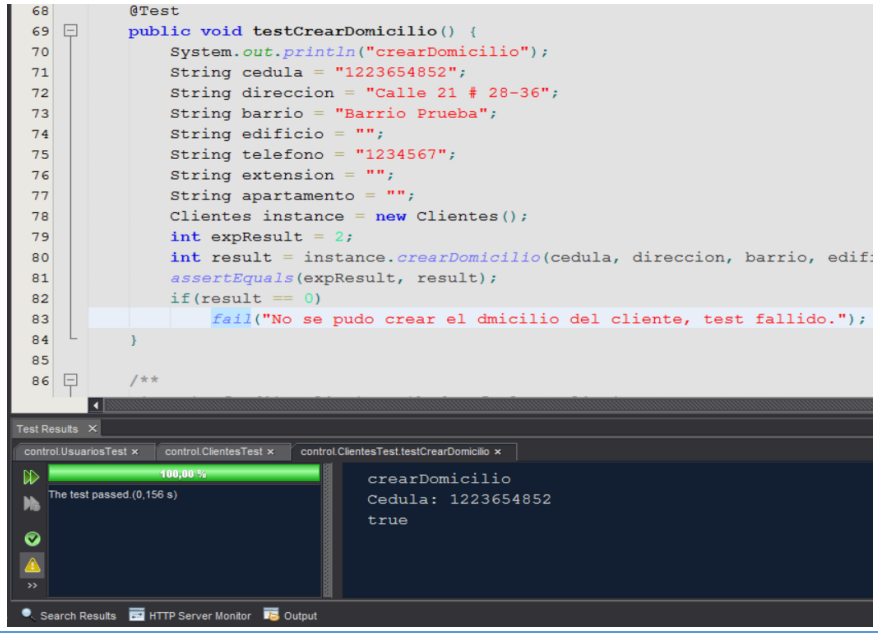
Prueba Unitaria				
Prueba No:	PU0019			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Generar un listado de remolques activos sin asignar a un vehículo			
Clase:	ColeccionRemolques.java			
Función:	public List<Remolque> getDisponibles()			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos remolque de longitud 2			
Salida obtenida:	Lista de objetos remolque de longitud 2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0020			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Cambiar el estado de un remolque			
Clase:	ColeccionRemolques.java			
Función:	public int cambiarEstado			
Configuraciones:				
Datos de Entrada:	➢ int id = 1; ➢ String estado = "INACTTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
166     @Test
167     public void testCambiarEstado() {
168         System.out.println("cambiarEstado");
169         int id = 1;
170         String estado = "INACTTIVO";
171         ColeccionRemolques instance = new ColeccionRemolques();
172         int expResult = 1;
173         int result = instance.cambiarEstado(id, estado);
174         if(expResult==result)
175             assertEquals(expResult, result);
176         else
177             fail("Ocurrio Un Error En La Busqueda");
178     }
```

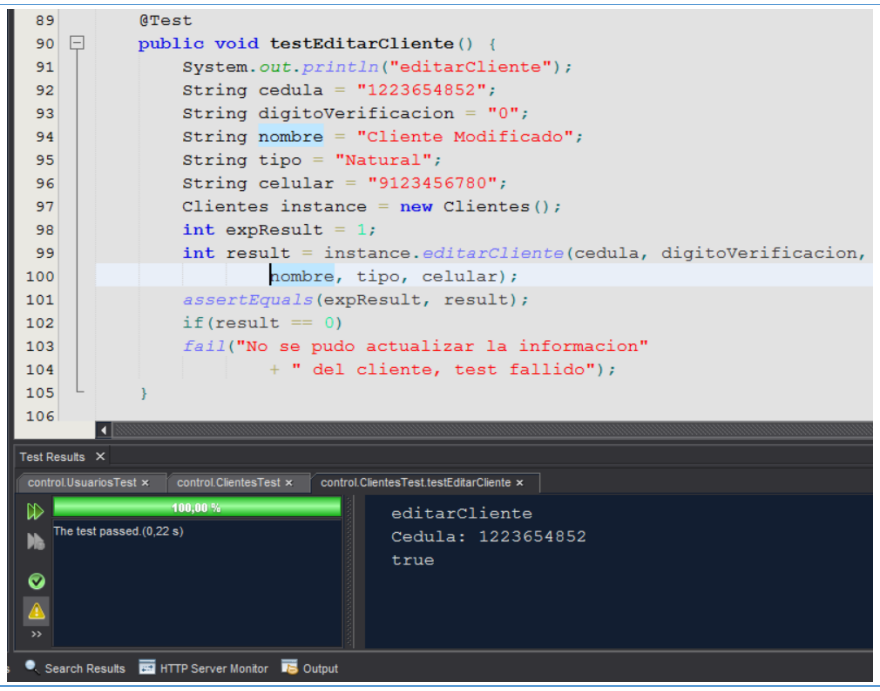
Prueba Unitaria				
Prueba No:	PU0021			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Eliminar un remolque			
Clase:	ColeccionRemolques.java			
Función:	public int eliminar			
Configuraciones:				
Datos de Entrada:	➢ int id = 4;			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

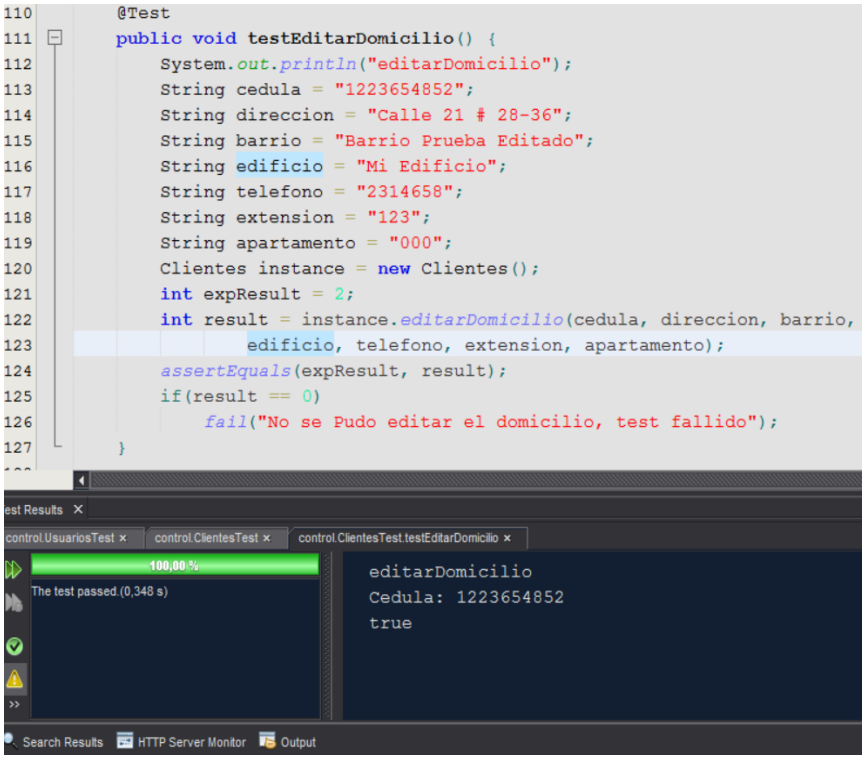
Prueba Unitaria				
Prueba No:	PU0022			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Registrar un cliente			
Clase:	ColeccionClientes.java			
Función:	public int crearCliente			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "1223654852"; ➢ String digitoVerificacion = "0"; ➢ String nombre = "Cliente Prueba"; ➢ String tipo = "Natural"; ➢ String celular = "0123456789";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0023			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Registrar el domicilio de un cliente.			
Clase:	CleccionClientes.java			
Función:	public int crearDomicilio			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "1223654852"; ➢ String direccion = "Calle 21 # 28-36"; ➢ String barrio = "Barrio Prueba"; ➢ String edificio = ""; ➢ String telefono = "1234567"; ➢ String extension = ""; ➢ String apartamento = "";			
Salida esperada:	2			
Salida obtenida:	2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
68      @Test
69      public void testCrearDomicilio() {
70          System.out.println("crearDomicilio");
71          String cedula = "1223654852";
72          String direccion = "Calle 21 # 28-36";
73          String barrio = "Barrio Prueba";
74          String edificio = "";
75          String telefono = "1234567";
76          String extension = "";
77          String apartamento = "";
78          Clientes instance = new Clientes();
79          int expResult = 2;
80          int result = instance.crearDomicilio(cedula, direccion, barrio, edif
81          assertEquals(expResult, result);
82          if(result == 0)
83              fail("No se pudo crear el dmicilio del cliente, test fallido.");
84      }
85
86      /**
```

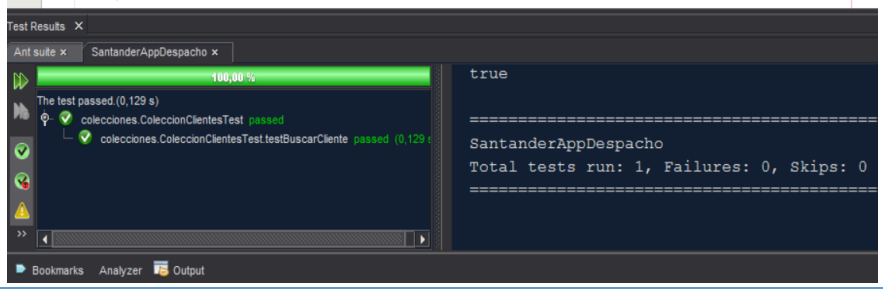
```
Test Results
control.UsuariosTest ×   control.ClientesTest ×   control.ClientesTest.testCrearDomicilio ×
100,00 %
The test passed.(0,156 s)
crearDomicilio
Cedula: 1223654852
true
Search Results   HTTP Server Monitor   Output
```

Prueba Unitaria				
Prueba No:	PU0024			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar la información de un cliente			
Clase:	ColeccionClientes.java			
Función:	public int editarCliente			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "1223654852"; ➢ String digitoVerificacion = "0"; ➢ String nombre = "Cliente Modificado"; ➢ String tipo = "Natural"; ➢ String celular = "9123456780";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

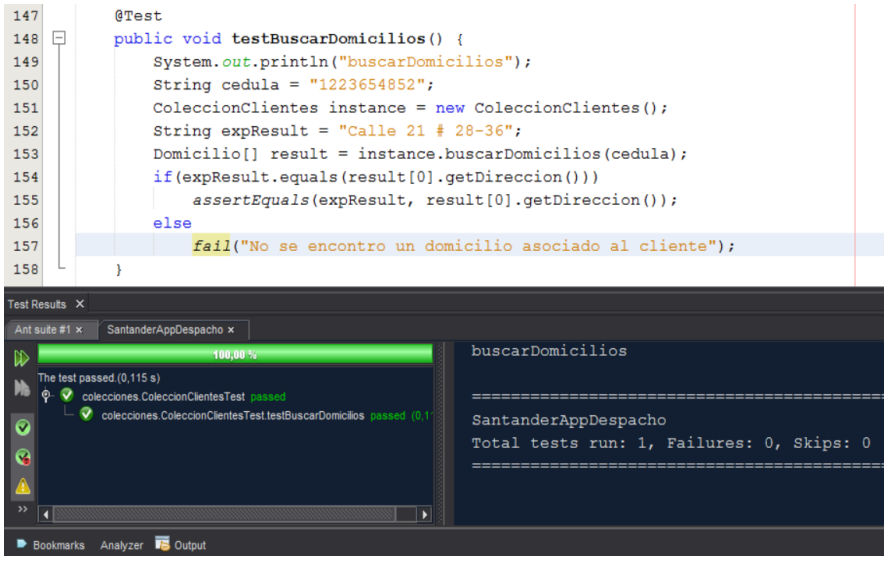
Prueba Unitaria				
Prueba No:	PU0025			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar el domicilio de un cliente			
Clase:	ColeccionClientes.java			
Función:	public int editarDomicilio			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "1223654852"; ➢ String direccion = "Calle 21 # 28-36"; ➢ String barrio = "Barrio Prueba Editado"; ➢ String edificio = "Mi Edificio"; ➢ String telefono = "2314658"; ➢ String extension = "123"; ➢ String apartamento = "000";			
Salida esperada:	2			
Salida obtenida:	2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0026</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Buscar un cliente por el número de documento</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionClientes.java</td></tr>
<tr><td>Función:</td><td colspan="4">public synchronized Cliente buscarCliente</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String cedula = "1223654852";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">2</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">2</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td rowspan="2">Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Evidencia:</td><td colspan="4">

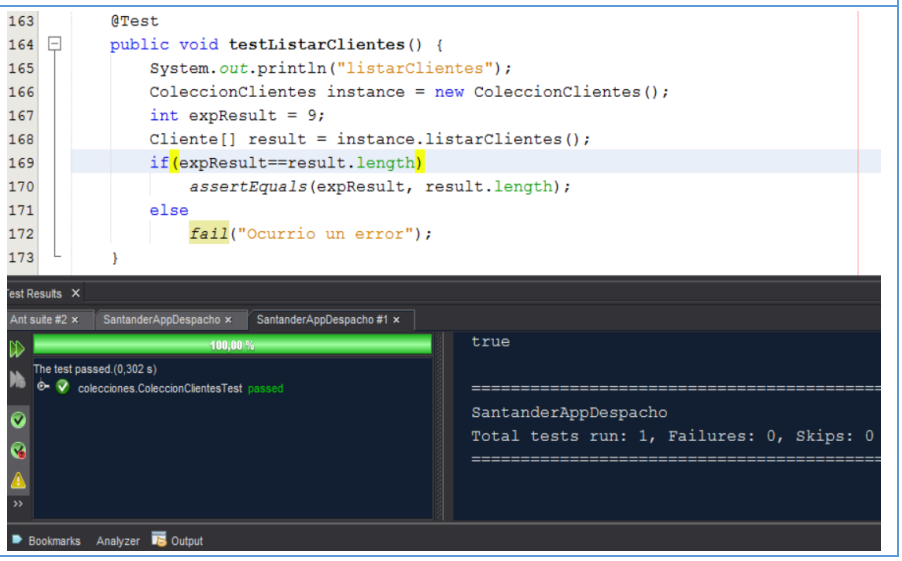
```
131     @Test
132     public void testBuscarCliente() {
133         System.out.println("buscarCliente");
134         String cedula = "1223654852";
135         ColeccionClientes instance = new ColeccionClientes();
136
137         Cliente result = instance.buscarCliente(cedula);
138         if(cedula.equals(result.getCedula()))
139             assertEquals(cedula, result.getCedula());
140         else
141             fail("No se encontro el cliente");
142     }
```



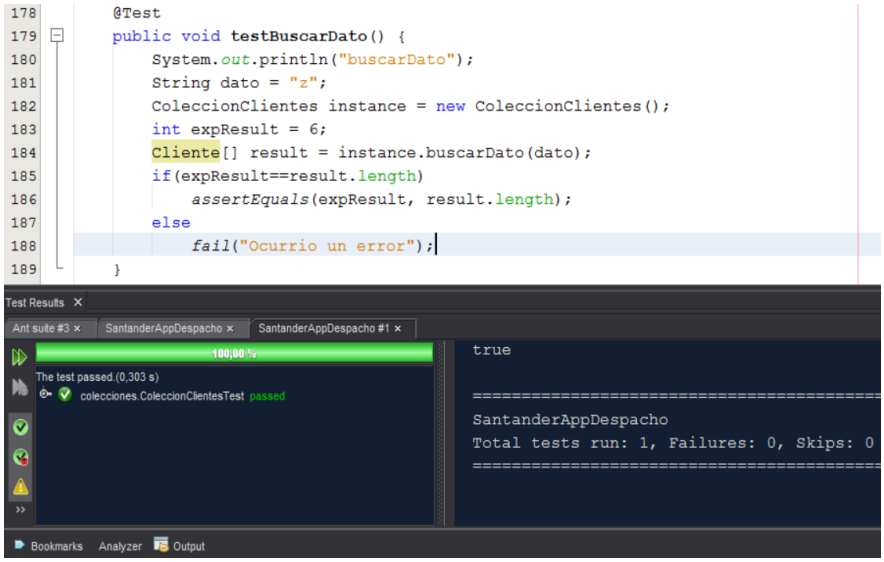
</td></tr>
</table>

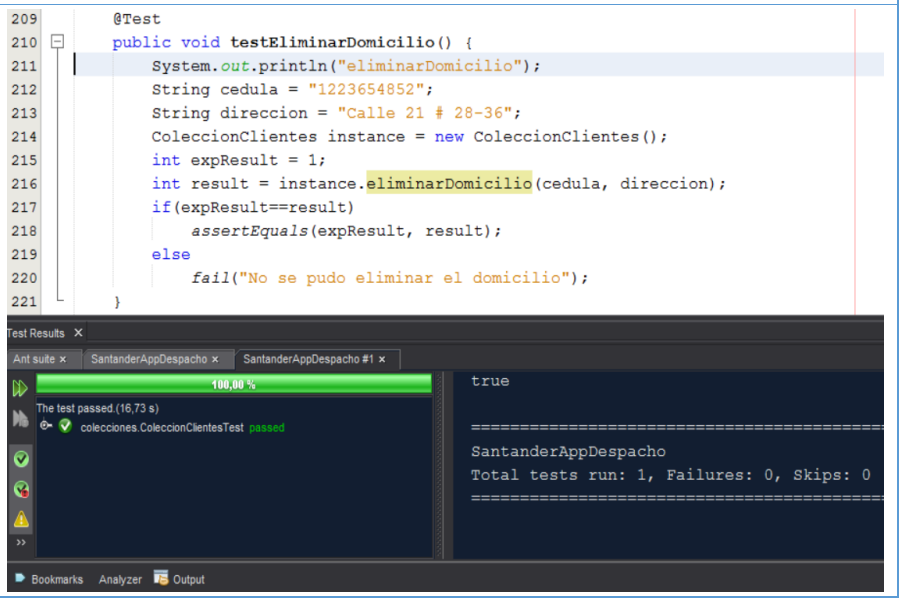
Prueba Unitaria				
Prueba No:	PU0027			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un domicilio del cliente por número de documento.			
Clase:	ColeccionClientes.java			
Función:	public synchronized Domicilio[] buscarDomicilios			
Configuraciones :				
Datos de Entrada:	➢ String cedula = "1223654852";			
Salida esperada:	Lista de objetos domicilio			
Salida obtenida:	Lista de objetos domicilio			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
147     @Test
148     public void testBuscarDomicilios() {
149         System.out.println("buscarDomicilios");
150         String cedula = "1223654852";
151         ColeccionClientes instance = new ColeccionClientes();
152         String expResult = "Calle 21 # 28-36";
153         Domicilio[] result = instance.buscarDomicilios(cedula);
154         if(expResult.equals(result[0].getDireccion()))
155             assertEquals(expResult, result[0].getDireccion());
156         else
157             fail("No se encontro un domicilio asociado al cliente");
158     }
```

Prueba Unitaria				
Prueba No:	PU0028			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Generar una lista de todos los clientes			
Clase:	ColeccionClientes.java			
Función:	public synchronized Cliente[] listarClientes			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de clientes de longitud 9			
Salida obtenida:	Lista de clientes de longitud 9			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

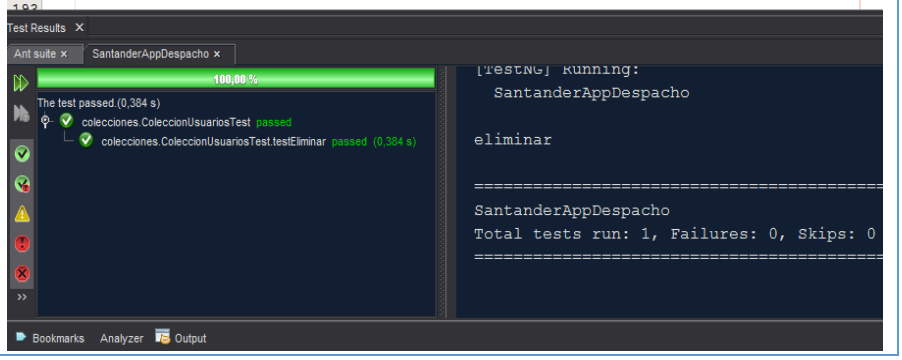
```
163     @Test
164     public void testListarClientes() {
165         System.out.println("listarClientes");
166         ColeccionClientes instance = new ColeccionClientes();
167         int expResult = 9;
168         Cliente[] result = instance.listarClientes();
169         if(expResult==result.length)
170             assertEquals(expResult, result.length);
171         else
172             fail("Ocurrio un error");
173     }
```

Prueba Unitaria				
Prueba No:	PU0029			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un cliente por un dato parcial.			
Clase:	ColeccionClientes.java			
Función:	public synchronized Cliente[] buscarDato			
Configuraciones:				
Datos de Entrada:	➢ String dato = "z";			
Salida esperada:	Lista de clientes de tamaño 6			
Salida obtenida:	Lista de clientes de tamaño 6			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

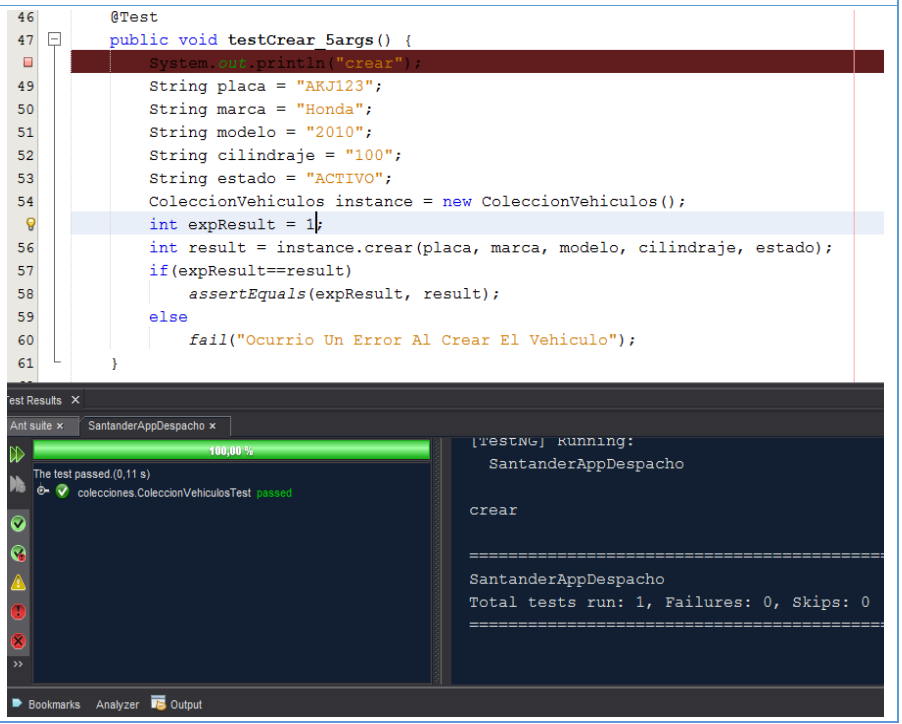
Prueba Unitaria				
Prueba No:	PU0030			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Eliminar un domicilio de un usuario			
Clase:	ColeccionClientes.java			
Función:	public int eliminarDomicilio			
Configuraciones :				
Datos de Entrada:	➢ String cedula = "1223654852"; ➢ String direccion = "Calle 21 # 28-36";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

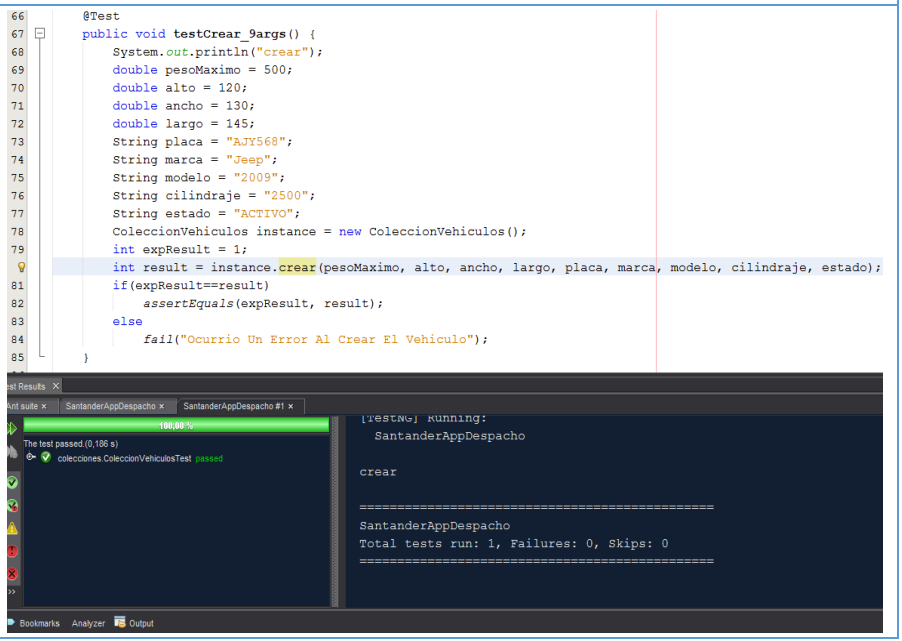
<table>
<tr><th colspan="5">Prueba Unitaria</th></tr>
<tr><td>Prueba No:</td><td colspan="4">PU0031</td></tr>
<tr><td>Autor:</td><td colspan="4">Carlos Andres López Ramírez</td></tr>
<tr><td>Objetivo (s):</td><td colspan="4">Eliminar un usuario</td></tr>
<tr><td>Clase:</td><td colspan="4">ColeccionClientes.java</td></tr>
<tr><td>Función:</td><td colspan="4">public synchronized int eliminar</td></tr>
<tr><td>Configuraciones:</td><td colspan="4"></td></tr>
<tr><td>Datos de Entrada:</td><td colspan="4">➢ String cedula = "1223654852";</td></tr>
<tr><td>Salida esperada:</td><td colspan="4">1</td></tr>
<tr><td>Salida obtenida:</td><td colspan="4">1</td></tr>
<tr><td>Estado:</td><td>Correcto:</td><td>Si</td><td>Fallido:</td><td></td></tr>
<tr><td rowspan="2">Error</td><td>Código:</td><td></td><td>Severidad:</td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Evidencia:</td><td colspan="4">

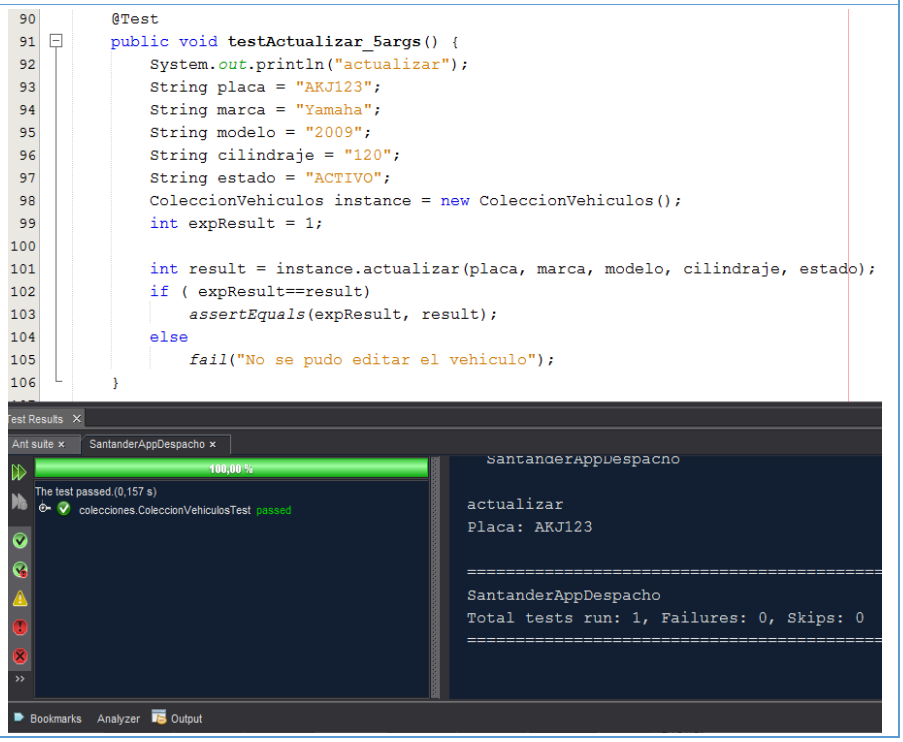
```
@Test
public void testEliminar() {
    System.out.println("eliminar");
    String cedula = "99956789";
    ColeccionUsuarios instance = new ColeccionUsuarios();
    int expResult = 1;
    int result = instance.eliminar(cedula);
    if(expResult==result)
        assertEquals(expResult, result);
    else
        fail("No se pudo eliminar el Usuario");
}
```



</td></tr>
</table>

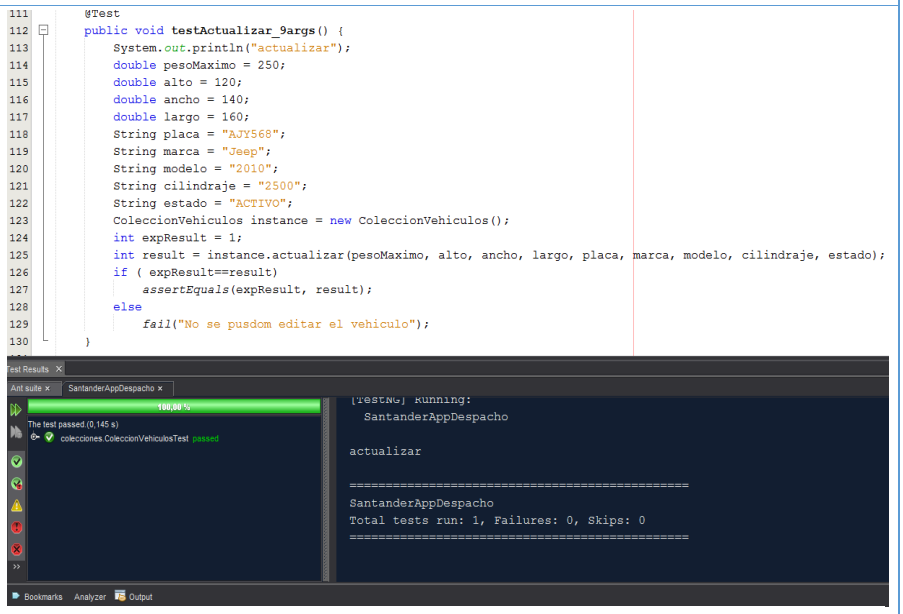
Prueba Unitaria				
Prueba No:	PU0032			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un vehículo tipo motocicleta			
Clase:	ColeccionVehiculos.java			
Función:	public int crear			
Configuraciones :				
Datos de Entrada:	➢ String placa = "AKJ123"; ➢ String marca = "Honda"; ➢ String modelo = "2010"; ➢ String cilindraje = "100"; ➢ String estado = "ACTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0033			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un vehículo tipo automovil			
Clase:	ColeccionVehiculos.java			
Función:	public int crear			
Configuraciones:				
Datos de Entrada:	➢ double pesoMaximo = 500; ➢ double alto = 120; ➢ double ancho = 130; ➢ double largo = 145; ➢ String placa = "AJY568"; ➢ String marca = "Jeep"; ➢ String modelo = "2009"; ➢ String cilindraje = "2500"; ➢ String estado = "ACTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0034			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Actualizar la información de un vehículo			
Clase:	ColeccionVehiculos.java			
Función:	public int actualizar			
Configuraciones:				
Datos de Entrada:	➢ String placa = "AKJ123"; ➢ String marca = "Yamaha"; ➢ String modelo = "2009"; ➢ String cilindraje = "120"; ➢ String estado = "ACTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

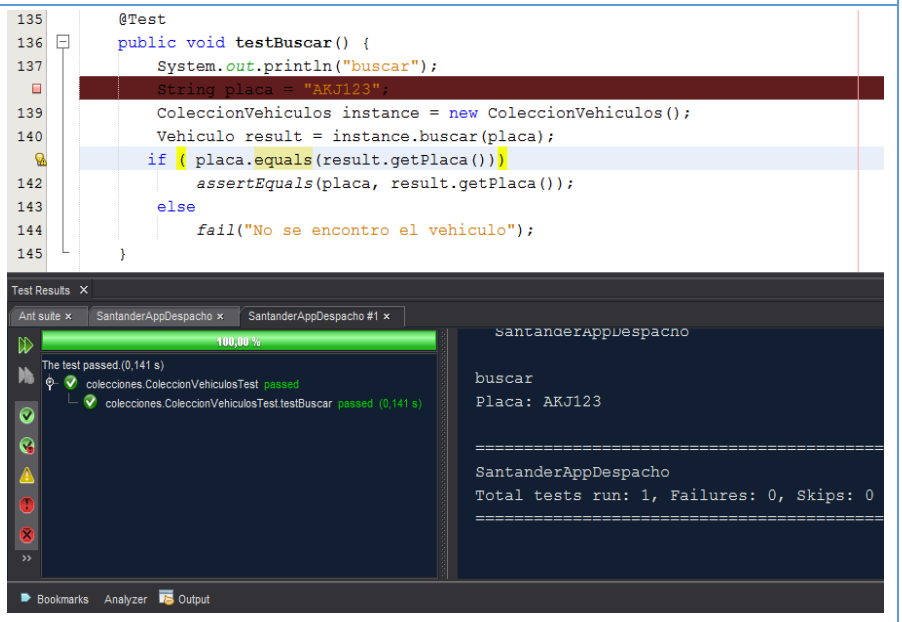
```
90      @Test
91      public void testActualizar_5args() {
92          System.out.println("actualizar");
93          String placa = "AKJ123";
94          String marca = "Yamaha";
95          String modelo = "2009";
96          String cilindraje = "120";
97          String estado = "ACTIVO";
98          ColeccionVehiculos instance = new ColeccionVehiculos();
99          int expResult = 1;
100
101         int result = instance.actualizar(placa, marca, modelo, cilindraje, estado);
102         if ( expResult==result)
103             assertEquals(expResult, result);
104         else
105             fail("No se pudo editar el vehiculo");
106     }
```

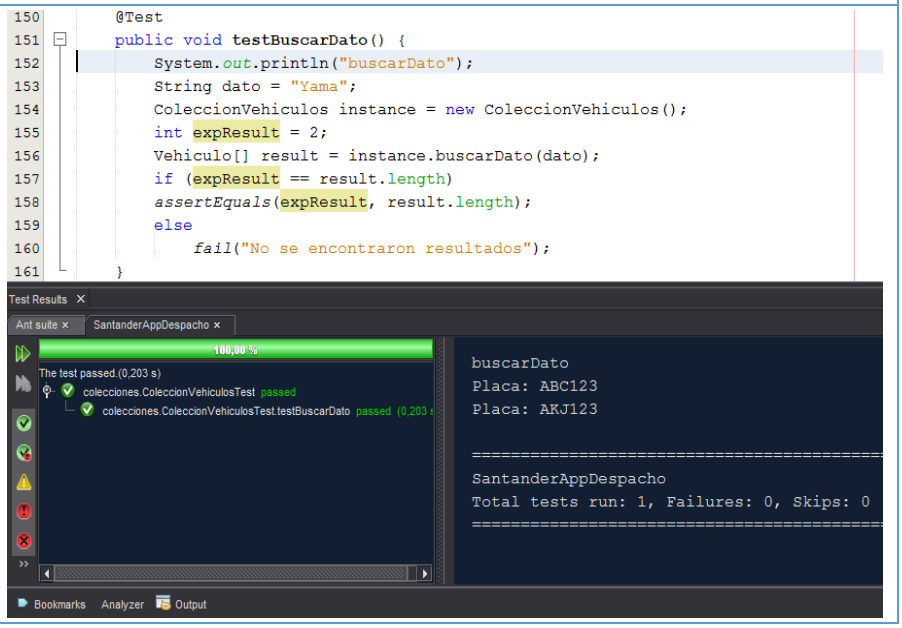
```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,157 s)
colecciones.ColeccionVehiculosTest passed

SantanderAppDespacho

actualizar
Placa: AKJ123

=============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
=============================================

Bookmarks   Analyzer   Output
```

Prueba Unitaria				
Prueba No:	PU0035			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Actualizar la información de un vehículo			
Clase:	ColeccionVehiculos.java			
Función:	public int actualizar			
Configuraciones :				
Datos de Entrada:	➢ double pesoMaximo = 250; ➢ double alto = 120; ➢ double ancho = 140; ➢ double largo = 160; ➢ String placa = "AJY568"; ➢ String marca = "Jeep"; ➢ String modelo = "2010"; ➢ String cilindraje = "2500"; ➢ String estado = "ACTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
111     @Test
112     public void testActualizar_9args() {
113         System.out.println("actualizar");
114         double pesoMaximo = 250;
115         double alto = 120;
116         double ancho = 140;
117         double largo = 160;
118         String placa = "AJY568";
119         String marca = "Jeep";
120         String modelo = "2010";
121         String cilindraje = "2500";
122         String estado = "ACTIVO";
123         ColeccionVehiculos instance = new ColeccionVehiculos();
124         int expResult = 1;
125         int result = instance.actualizar(pesoMaximo, alto, ancho, largo, placa, marca, modelo, cilindraje, estado);
126         if ( expResult==result)
127             assertEquals(expResult, result);
128         else
129             fail("No se pusdom editar el vehiculo");
130     }
```

```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,145 s)
colecciones.ColeccionVehiculosTest passed
[TestNG] Running:
  SantanderAppDespacho

actualizar

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
Bookmarks   Analyzer   Output
```

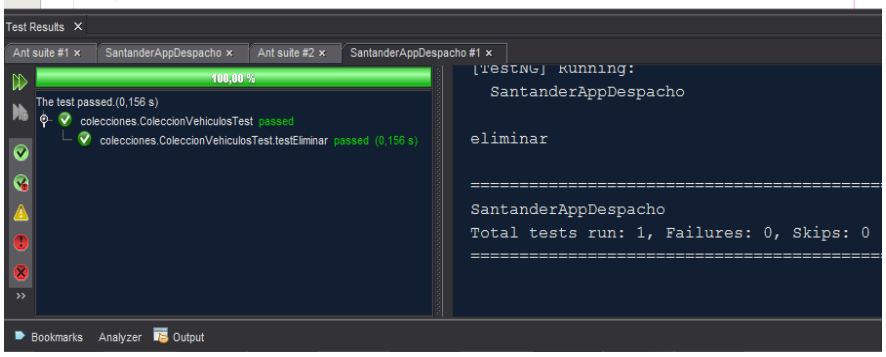
Prueba Unitaria				
Prueba No:	PU0036			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un vehículo con el número de placa.			
Clase:	ColeccionVehiculos.java			
Función:	public Vehiculo buscar			
Configuraciones :				
Datos de Entrada:	➢ String placa = "AKJ123";			
Salida esperada:	Objeto vehiculo			
Salida obtenida:	Objeto vehiculo			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

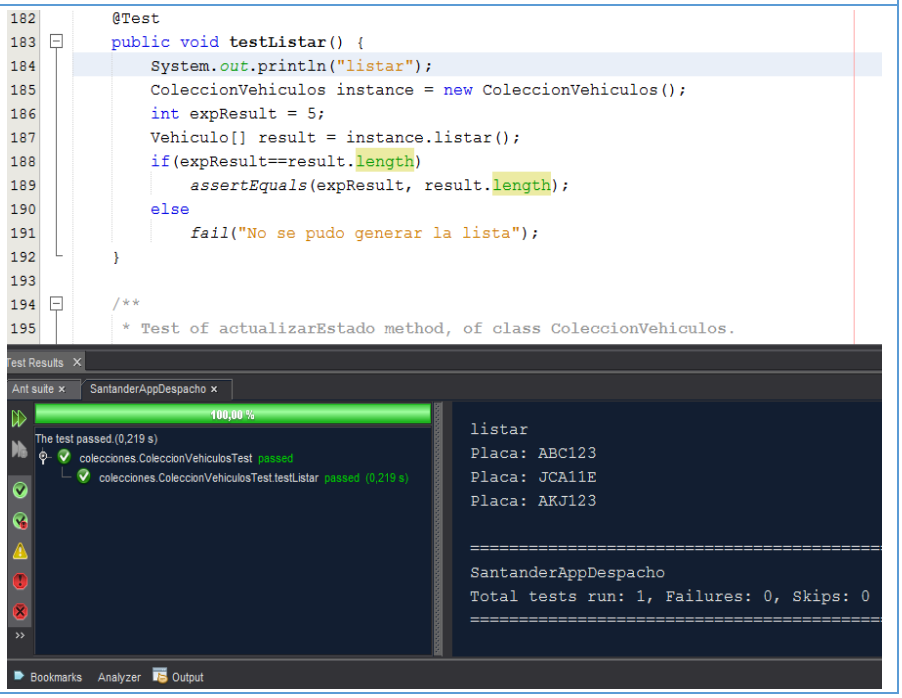
Prueba Unitaria				
Prueba No:	PU0037			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un vehículo con un dato parcial			
Clase:	ColeccionVehiculos.java			
Función:	public Vehiculo[] buscarDato			
Configuraciones:				
Datos de Entrada:	➢ String dato = "Yama";			
Salida esperada:	Lista de objetos vehículo			
Salida obtenida:	Lista de objetos vehículo			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

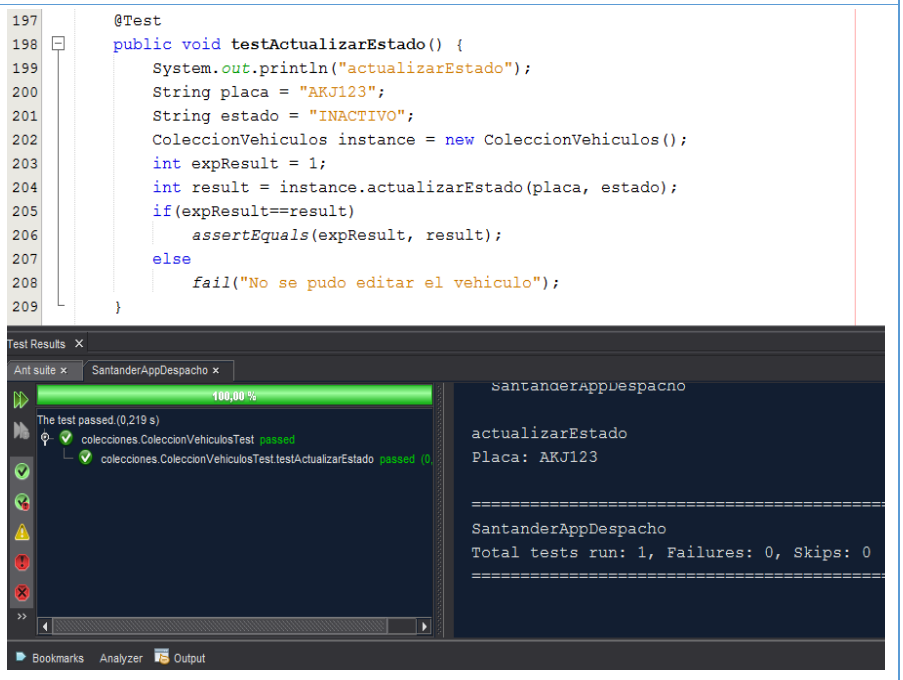
```
150     @Test
151     public void testBuscarDato() {
152         System.out.println("buscarDato");
153         String dato = "Yama";
154         ColeccionVehiculos instance = new ColeccionVehiculos();
155         int expResult = 2;
156         Vehiculo[] result = instance.buscarDato(dato);
157         if (expResult == result.length)
158         assertEquals(expResult, result.length);
159         else
160             fail("No se encontraron resultados");
161     }
```

Prueba Unitaria				
Prueba No:	PU0038			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Eliminar un vehiculo			
Clase:	ColeccionVehiculos.java			
Función:	public int eliminar			
Configuraciones :				
Datos de Entrada:	➢ String placa = "AJY568";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

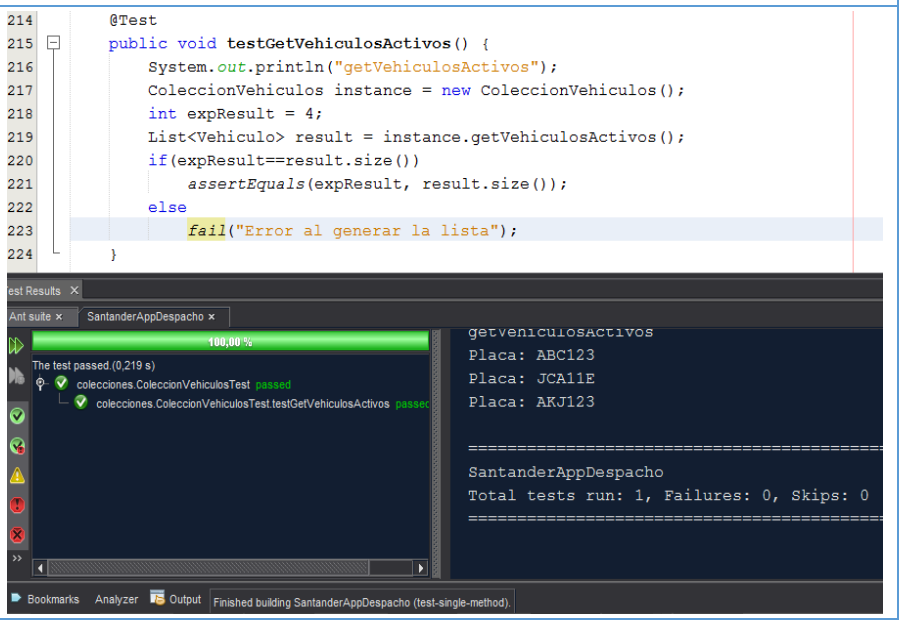
```
166     @Test
167     public void testEliminar() {
168         System.out.println("eliminar");
169         String placa = "AJY568";
170         ColeccionVehiculos instance = new ColeccionVehiculos();
171         int expResult = 1;
172         int result = instance.eliminar(placa);
173         if(expResult==result)
174             assertEquals(expResult, result);
175         else
176             fail("No se pudo eliminar el vehiculo");
177     }
```



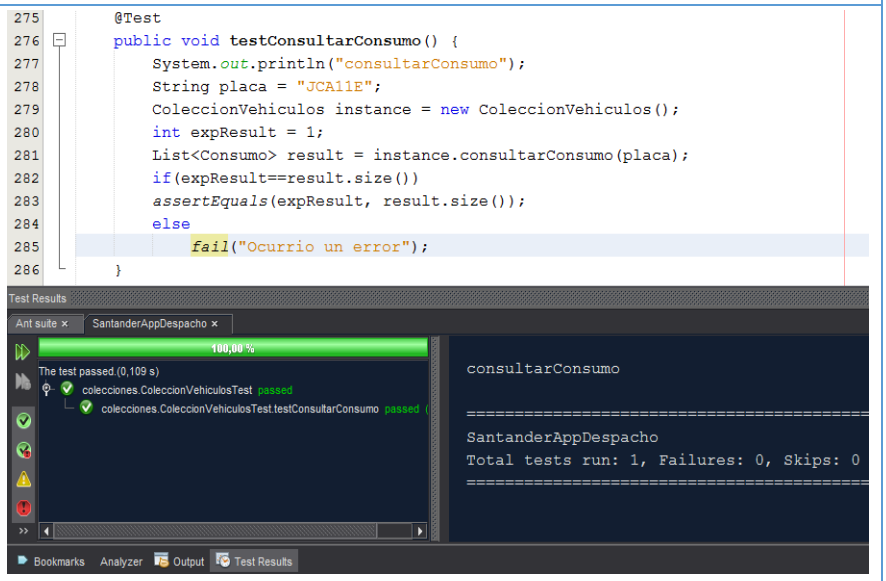
Prueba Unitaria				
Prueba No:	PU0039			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Listar los vehículos existentes			
Clase:	ColeccionVehiculos.java			
Función:	public Vehiculo[] listar			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de vehículos de longitud 5			
Salida obtenida:	Lista de vehículos de longitud 5			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0040			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Cambiar el estado de un vehiculo			
Clase:	ColeccionVehiculos.java			
Función:	public int actualizarEstado			
Configuraciones :				
Datos de Entrada:	➢ String placa = "AKJ123"; ➢ String estado = "INACTIVO";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

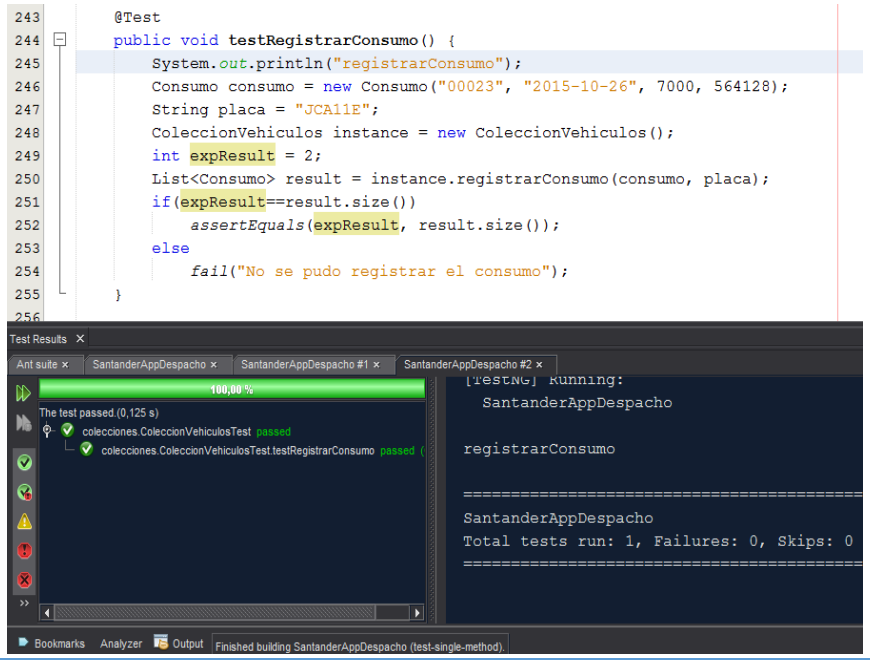
```
197     @Test
198     public void testActualizarEstado() {
199         System.out.println("actualizarEstado");
200         String placa = "AKJ123";
201         String estado = "INACTIVO";
202         ColeccionVehiculos instance = new ColeccionVehiculos();
203         int expResult = 1;
204         int result = instance.actualizarEstado(placa, estado);
205         if(expResult==result)
206             assertEquals(expResult, result);
207         else
208             fail("No se pudo editar el vehiculo");
209     }
```

Prueba Unitaria				
Prueba No:	PU0041			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Generar una lista de todos los vehículos activos			
Clase:	ColeccionVehiculos.java			
Función:	public List<Vehiculo> getVehiculosActivos			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos vehículo de longitud 4			
Salida obtenida:	Lista de objetos vehículo de longitud 4			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
214     @Test
215     public void testGetVehiculosActivos() {
216         System.out.println("getVehiculosActivos");
217         ColeccionVehiculos instance = new ColeccionVehiculos();
218         int expResult = 4;
219         List<Vehiculo> result = instance.getVehiculosActivos();
220         if(expResult==result.size())
221             assertEquals(expResult, result.size());
222         else
223             fail("Error al generar la lista");
224     }
```

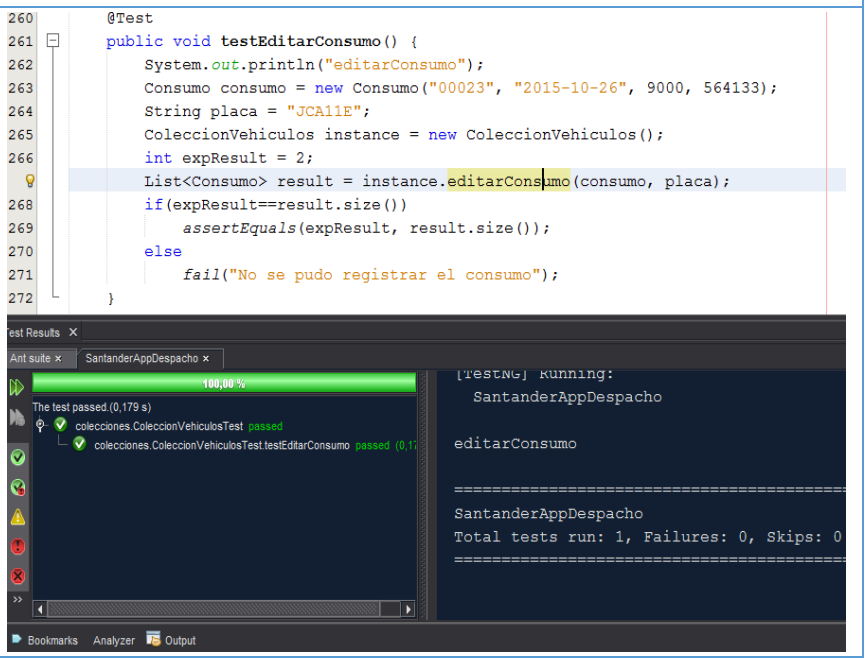
Prueba Unitaria				
Prueba No:	PU0042			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar los consumos registrados de un vehículo			
Clase:	ColeccionVehiculos.java			
Función:	public List<Consumo> consultarConsumo			
Configuraciones:				
Datos de Entrada:	➢ String placa = "JCA11E";			
Salida esperada:				
Salida obtenida:				
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

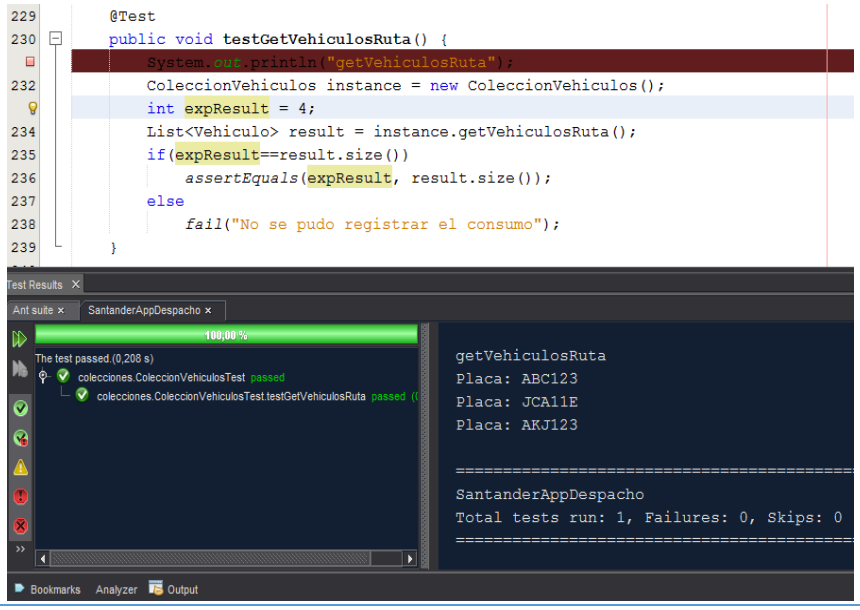
```
275     @Test
276     public void testConsultarConsumo() {
277         System.out.println("consultarConsumo");
278         String placa = "JCA11E";
279         ColeccionVehiculos instance = new ColeccionVehiculos();
280         int expResult = 1;
281         List<Consumo> result = instance.consultarConsumo(placa);
282         if(expResult==result.size())
283         assertEquals(expResult, result.size());
284         else
285             fail("Ocurrio un error");
286     }
```

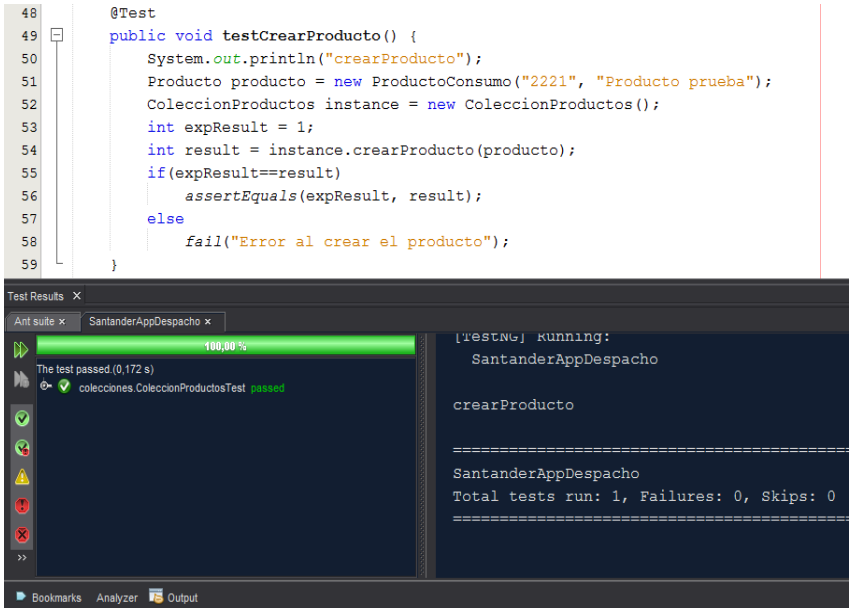
Prueba Unitaria				
Prueba No:	PU0043			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Registrar el consumo de un vehículo			
Clase:	ColeccionVehiculos.java			
Función:	public List<Consumo> registrarConsumo			
Configuraciones:				
Datos de Entrada:	➢ Consumo consumo = new Consumo("00023", "2015-10-26", 7000, 564128); ➢ String placa = "JCA11E";			
Salida esperada:	Lista de objetos consumo de longitud 2			
Salida obtenida:	Lista de objetos consumo de longitud 2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
243     @Test
244     public void testRegistrarConsumo() {
245         System.out.println("registrarConsumo");
246         Consumo consumo = new Consumo("00023", "2015-10-26", 7000, 564128);
247         String placa = "JCA11E";
248         ColeccionVehiculos instance = new ColeccionVehiculos();
249         int expResult = 2;
250         List<Consumo> result = instance.registrarConsumo(consumo, placa);
251         if(expResult==result.size())
252             assertEquals(expResult, result.size());
253         else
254             fail("No se pudo registrar el consumo");
255     }
256
```

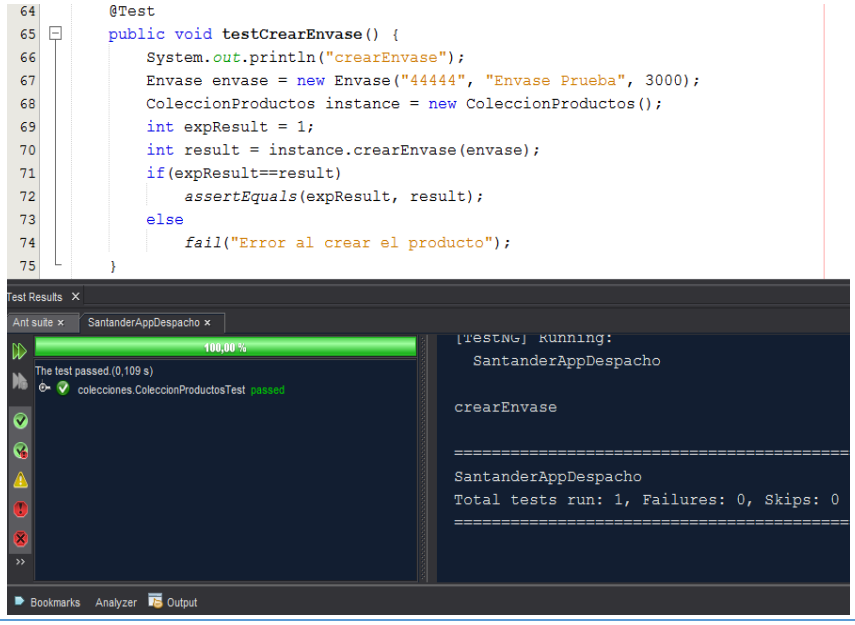
```
Test Results
Ant suite ×  SantanderAppDespacho ×  SantanderAppDespacho #1 ×  SantanderAppDespacho #2 ×
100,00 %
The test passed (0,125 s)
colecciones.ColeccionVehiculosTest passed
colecciones.ColeccionVehiculosTest.testRegistrarConsumo passed (
[TestNG] Running:
  SantanderAppDespacho

registrarConsumo

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
Bookmarks  Analyzer  Output  Finished building SantanderAppDespacho (test-single-method).
```

Prueba Unitaria				
Prueba No:	PU0044			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un consumo registrado			
Clase:	ColeccionVehiculos.java			
Función:	public List<Consumo> editarConsumo			
Configuraciones:				
Datos de Entrada:	➢ Consumo consumo = new Consumo("00023", "2015-10-26", 9000, 564133); ➢ String placa = "JCA11E";			
Salida esperada:	Lista de objetos consumo de longitud 2			
Salida obtenida:	Lista de objetos consumo de longitud 2			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

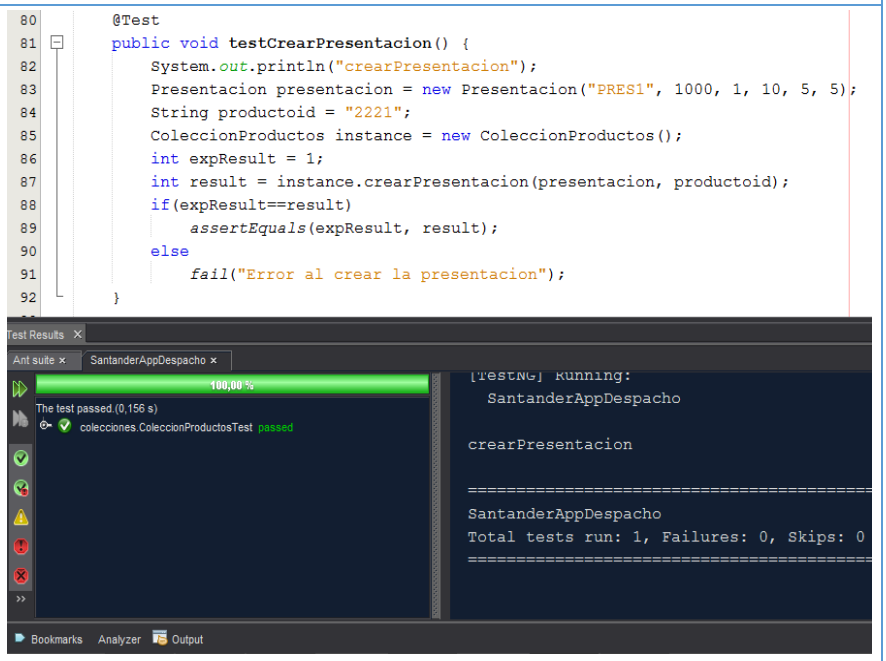
Prueba Unitaria				
Prueba No:	PU0045			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear una lista de vehículos activos			
Clase:	ColeccionVehiculos.java			
Función:	public List<Vehiculo> getVehiculosRuta			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos consumo de longitud 4			
Salida obtenida:	Lista de objetos consumo de longitud 4			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0046			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un producto			
Clase:	ColeccionProductos.java			
Función:	public int crearProducto			
Configuraciones:				
Datos de Entrada:	➢ Producto producto = new ProductoConsumo("2221", "Producto prueba");			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
48      @Test
49      public void testCrearProducto() {
50          System.out.println("crearProducto");
51          Producto producto = new ProductoConsumo("2221", "Producto prueba");
52          ColeccionProductos instance = new ColeccionProductos();
53          int expResult = 1;
54          int result = instance.crearProducto(producto);
55          if(expResult==result)
56              assertEquals(expResult, result);
57          else
58              fail("Error al crear el producto");
59      }
```

Prueba Unitaria				
Prueba No:	PU0047			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear un envase			
Clase:	ColeccionProductos.java			
Función:	public int crearEnvase			
Configuraciones:				
Datos de Entrada:	➢ Envase envase = new Envase("44444", "Envase Prueba", 3000);			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
64      @Test
65      public void testCrearEnvase() {
66          System.out.println("crearEnvase");
67          Envase envase = new Envase("44444", "Envase Prueba", 3000);
68          ColeccionProductos instance = new ColeccionProductos();
69          int expResult = 1;
70          int result = instance.crearEnvase(envase);
71          if(expResult==result)
72              assertEquals(expResult, result);
73          else
74              fail("Error al crear el producto");
75      }
```

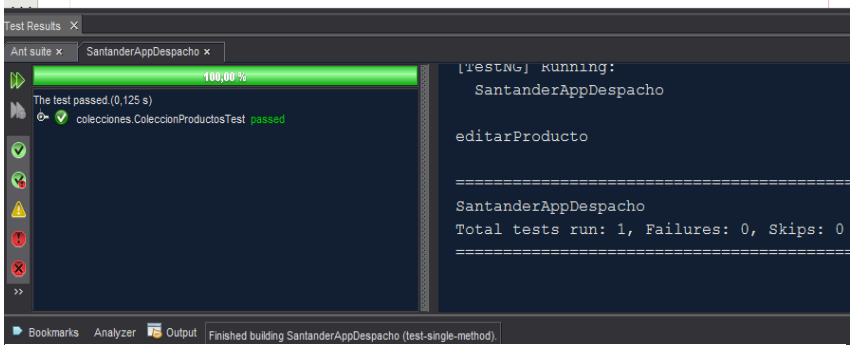
Prueba Unitaria				
Prueba No:	PU0048			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear presentación de un producto			
Clase:	ColeccionProductos.java			
Función:	public int crearPresentacion			
Configuraciones:				
Datos de Entrada:	➢ Presentacion presentacion = new Presentacion("PRES1", 1000, 1, 10, 5, 5); ➢ String productoid = "2221";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
80      @Test
81      public void testCrearPresentacion() {
82          System.out.println("crearPresentacion");
83          Presentacion presentacion = new Presentacion("PRES1", 1000, 1, 10, 5, 5);
84          String productoid = "2221";
85          ColeccionProductos instance = new ColeccionProductos();
86          int expResult = 1;
87          int result = instance.crearPresentacion(presentacion, productoid);
88          if(expResult==result)
89              assertEquals(expResult, result);
90          else
91              fail("Error al crear la presentacion");
92      }
```

```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,156 s)
colecciones.ColeccionProductosTest passed
[TestNG] Running:
  SantanderAppDespacho

crearPresentacion

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
Bookmarks   Analyzer   Output
```

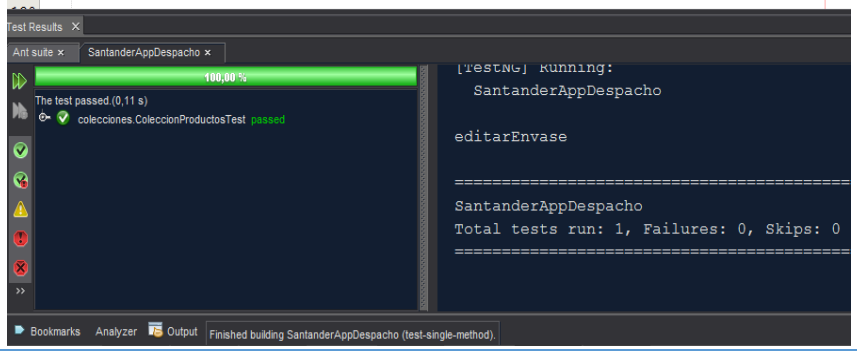
Prueba Unitaria				
Prueba No:	PU0049			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un producto			
Clase:	ColeccionProductos.java			
Función:	public int editarProducto			
Configuraciones:				
Datos de Entrada:	➢ String codigo = "2221"; ➢ String nombre = "Producto prueba Editado"; ➢ String codigo2 = "2221";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

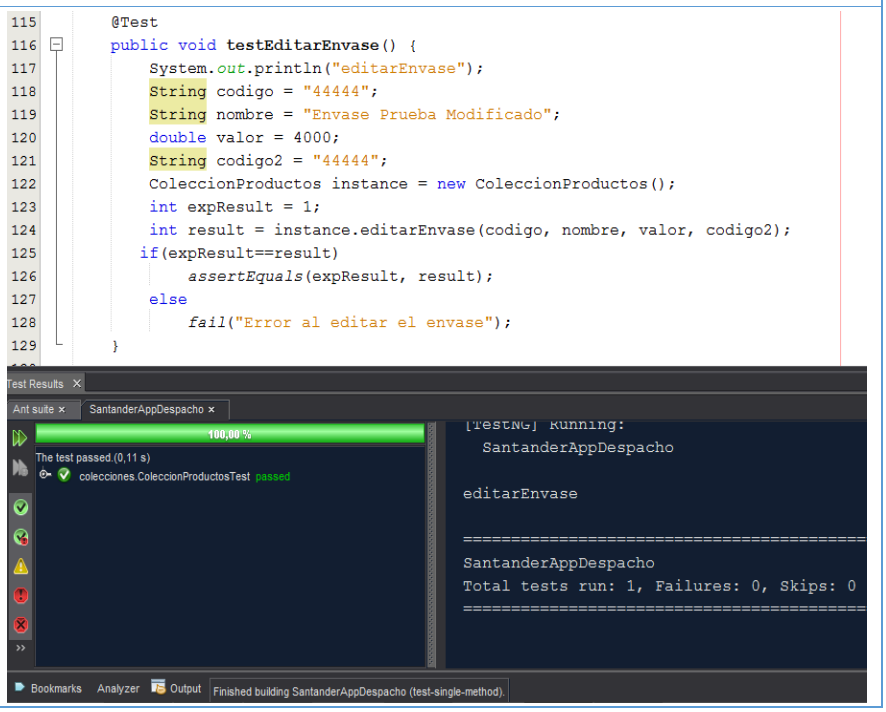
```
97      @Test
98      public void testEditarProducto() {
99          System.out.println("editarProducto");
100         String codigo = "2221";
101         String nombre = "Producto prueba Editado";
102         String codigo2 = "2221";
103         ColeccionProductos instance = new ColeccionProductos();
104         int expResult = 1;
105         int result = instance.editarProducto(codigo, nombre, codigo2);
106         if(expResult==result)
107             assertEquals(expResult, result);
108         else
109             fail("Error al editar el producto");
110     }
```

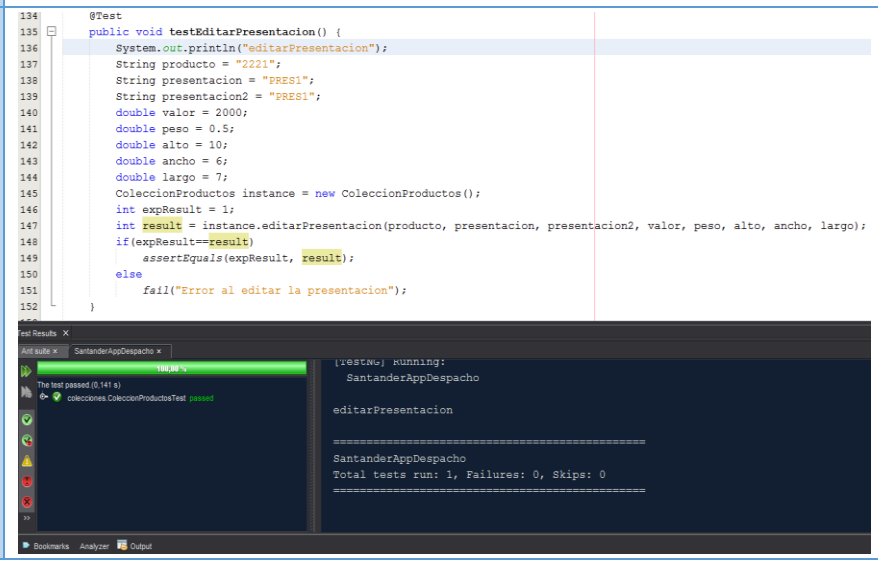


Prueba Unitaria				
Prueba No:	PU0050			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un envase			
Clase:	ColeccionProductos.java			
Función:	public int editarEnvase			
Configuraciones:				
Datos de Entrada:	➢ String codigo = "44444"; ➢ String nombre = "Envase Prueba Modificado"; ➢ double valor = 4000; ➢ String codigo2 = "44444";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
115     @Test
116     public void testEditarEnvase() {
117         System.out.println("editarEnvase");
118         String codigo = "44444";
119         String nombre = "Envase Prueba Modificado";
120         double valor = 4000;
121         String codigo2 = "44444";
122         ColeccionProductos instance = new ColeccionProductos();
123         int expResult = 1;
124         int result = instance.editarEnvase(codigo, nombre, valor, codigo2);
125        if(expResult==result)
126             assertEquals(expResult, result);
127         else
128             fail("Error al editar el envase");
129     }
```

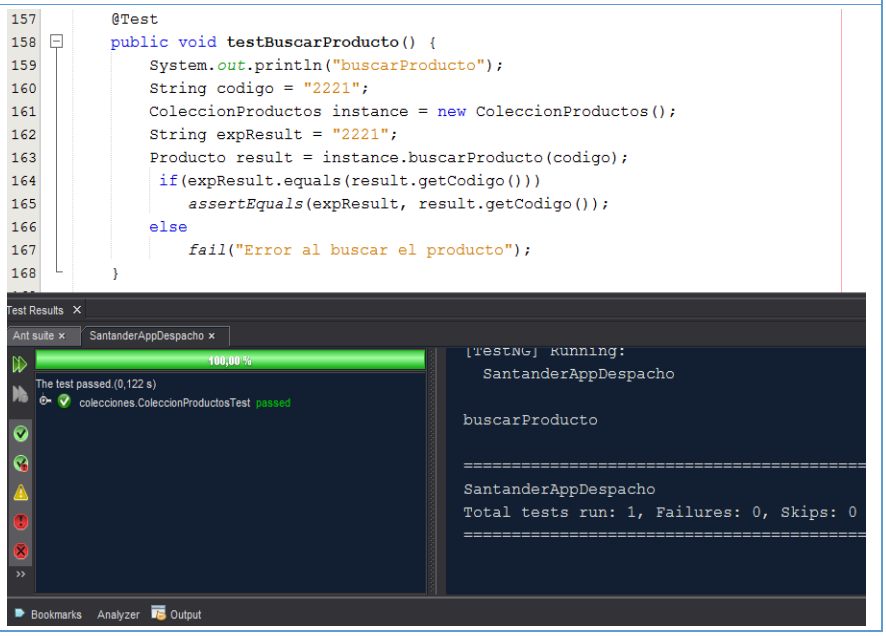


Prueba Unitaria				
Prueba No:	PU0051			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar un envase			
Clase:	ColeccionProductos.java			
Función:	public int editarEnvase			
Configuraciones:				
Datos de Entrada:	➢ String codigo = "44444"; ➢ String nombre = "Envase Prueba Modificado"; ➢ double valor = 4000; ➢ String codigo2 = "44444";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

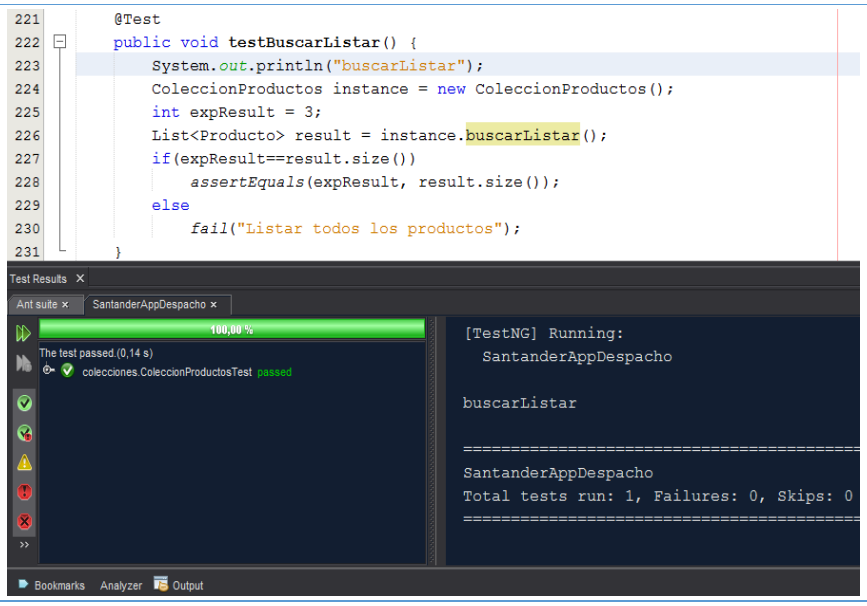
Prueba Unitaria				
Prueba No:	PU0052			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar una presentación de un producto			
Clase:	ColeccionProductos.java			
Función:	public int editarPresentacion			
Configuraciones:				
Datos de Entrada:	➢ String producto = "2221"; ➢ String presentacion = "PRES1"; ➢ String presentacion2 = "PRES1"; ➢ double valor = 2000; ➢ double peso = 0.5; ➢ double alto = 10; ➢ double ancho = 6; ➢ double largo = 7;			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
134     @Test
135     public void testEditarPresentacion() {
136         System.out.println("editarPresentacion");
137         String producto = "2221";
138         String presentacion = "PRES1";
139         String presentacion2 = "PRES1";
140         double valor = 2000;
141         double peso = 0.5;
142         double alto = 10;
143         double ancho = 6;
144         double largo = 7;
145         ColeccionProductos instance = new ColeccionProductos();
146         int expResult = 1;
147         int result = instance.editarPresentacion(producto, presentacion, presentacion2, valor, peso, alto, ancho, largo);
148         if(expResult==result)
149             assertEquals(expResult, result);
150         else
151             fail("Error al editar la presentacion");
152     }
```

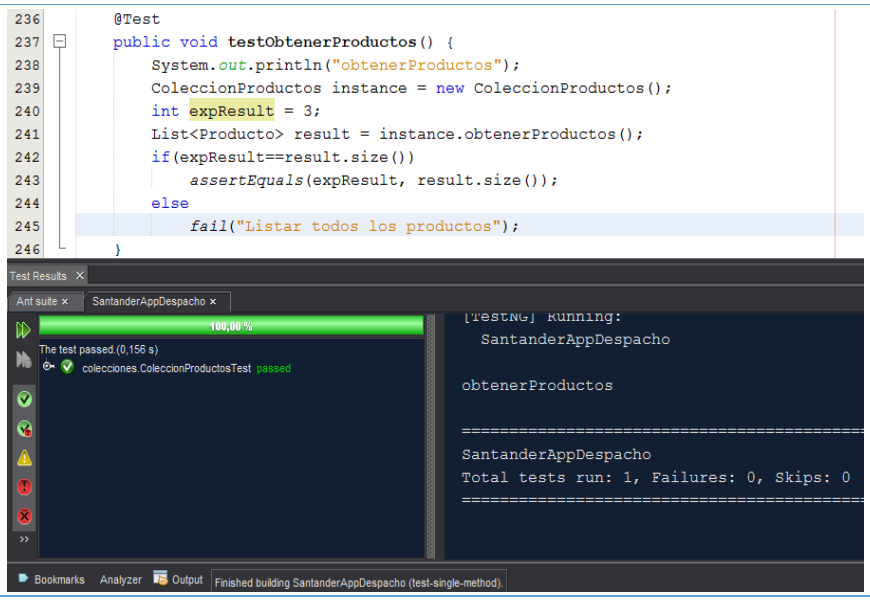
```
Test Results
Ant suite
SantanderAppDespacho
100,00 %
The test passed.(0,141 s)
colecciones.ColeccionProductosTest passed
[TestNG] Running:
  SantanderAppDespacho

editarPresentacion

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
Bookmarks
Analyzer
Output
```

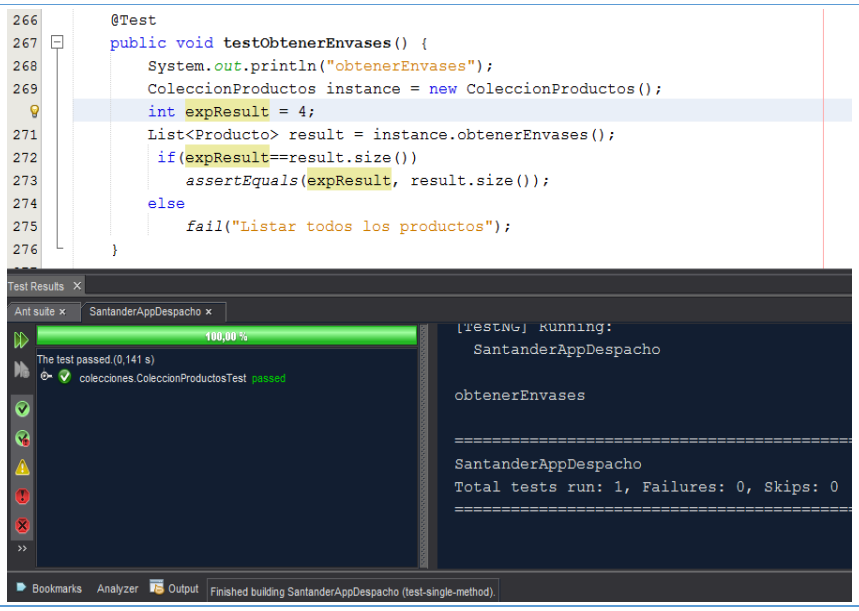
Prueba Unitaria				
Prueba No:	PU0053			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un producto por el código			
Clase:	ColeccionProductos.java			
Función:	public Producto buscarProducto			
Configuraciones:				
Datos de Entrada:	➢ String codigo = "2221";			
Salida esperada:	Objeto Producto			
Salida obtenida:	Objeto Producto			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

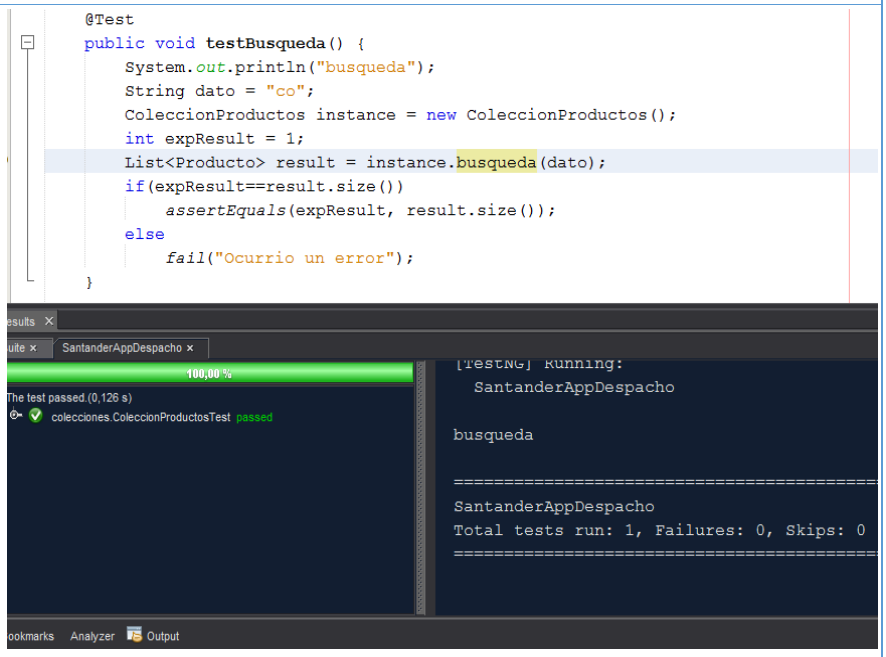
```
157     @Test
158     public void testBuscarProducto() {
159         System.out.println("buscarProducto");
160         String codigo = "2221";
161         ColeccionProductos instance = new ColeccionProductos();
162         String expResult = "2221";
163         Producto result = instance.buscarProducto(codigo);
164          if(expResult.equals(result.getCodigo()))
165             assertEquals(expResult, result.getCodigo());
166         else
167             fail("Error al buscar el producto");
168     }
```

Prueba Unitaria				
Prueba No:	PU0054			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Crear una lista de productos			
Clase:	ColeccionProductos.java			
Función:	public List<Producto> obtenerProductos			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos Producto de tamaño 7			
Salida obtenida:	Lista de objetos Producto de tamaño 7			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

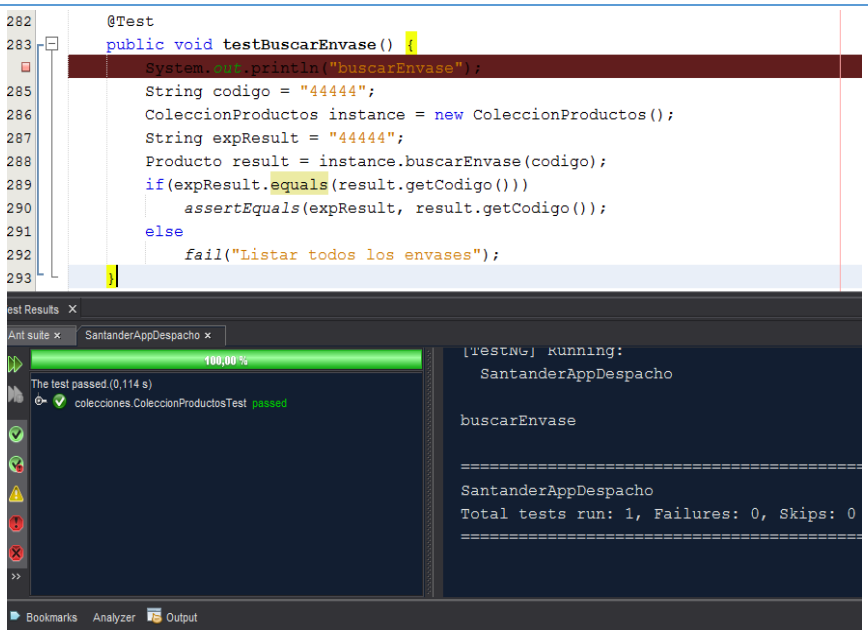
```
221     @Test
222     public void testBuscarListar() {
223         System.out.println("buscarListar");
224         ColeccionProductos instance = new ColeccionProductos();
225         int expResult = 3;
226         List<Producto> result = instance.buscarListar();
227         if(expResult==result.size())
228             assertEquals(expResult, result.size());
229         else
230             fail("Listar todos los productos");
231     }
```

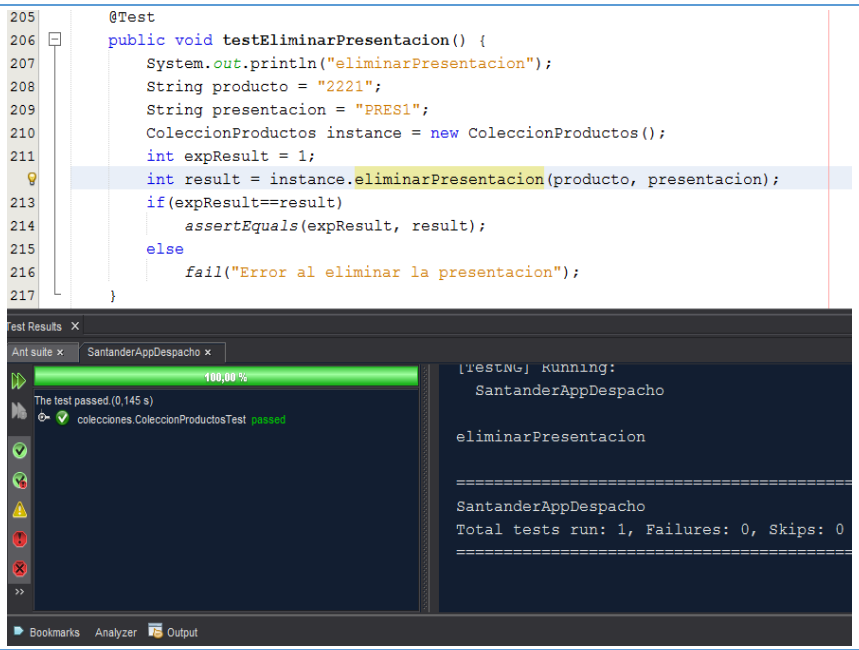
Prueba Unitaria				
Prueba No:	PU0055			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Obtener una lista de los productos de consumo.			
Clase:	ColeccionProductos.java			
Función:	public List<Producto> obtenerProductos			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de productos de tamaño 3			
Salida obtenida:	Lista de productos de tamaño 3			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0056			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Listar los envases			
Clase:	ColeccionProductos.java			
Función:	public List<Producto> obtenerEnvases			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos Producto de tamaño 7			
Salida obtenida:	Lista de objetos Producto de tamaño 7			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0057			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un producto con información parcial			
Clase:	ColeccionProductos.java			
Función:	public List<Producto> busqueda			
Configuraciones:				
Datos de Entrada:	➢ String dato = "co";			
Salida esperada:	Lista de objetos Producto de tamaño 1			
Salida obtenida:	Lista de objetos Producto de tamaño 1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

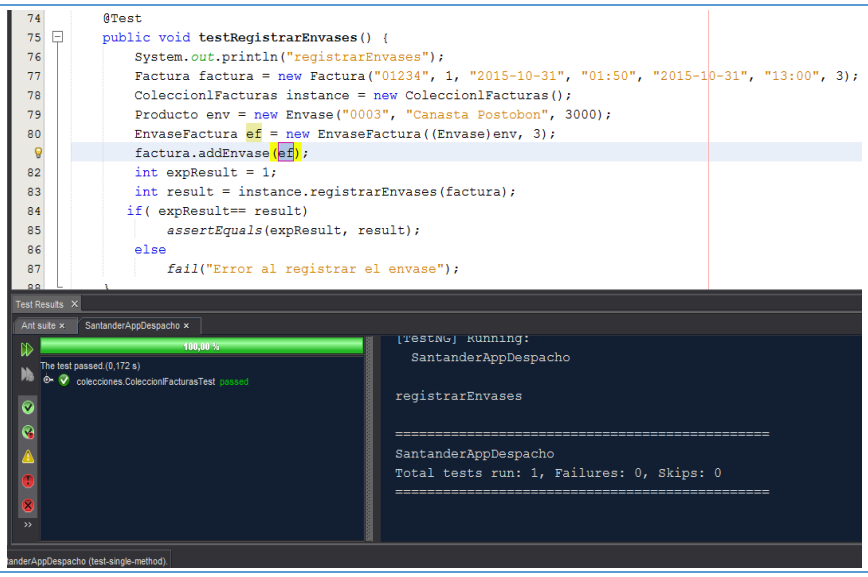
```
@Test
public void testBusqueda() {
    System.out.println("busqueda");
    String dato = "co";
    ColeccionProductos instance = new ColeccionProductos();
    int expResult = 1;
    List<Producto> result = instance.busqueda(dato);
    if(expResult==result.size())
        assertEquals(expResult, result.size());
    else
        fail("Ocurrio un error");
}
```

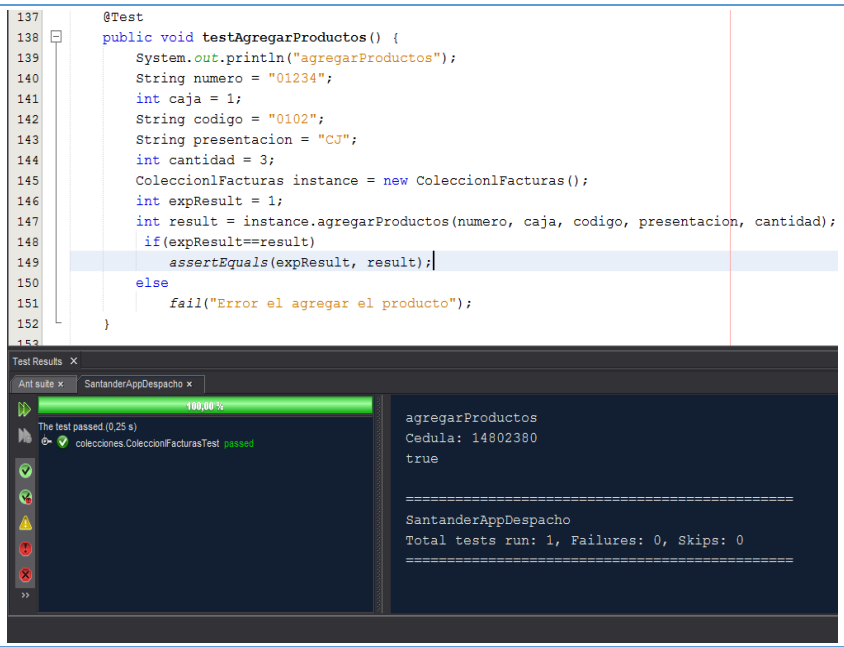
Prueba Unitaria				
Prueba No:	PU0058			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un envase por el código			
Clase:	ColeccionProductos.java			
Función:	private Producto consultarEnvase			
Configuraciones:				
Datos de Entrada:	➢ String codigo = "44444";			
Salida esperada:	Objeto producto tipo envase			
Salida obtenida:	Objeto producto tipo envase			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0059			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Elimnar una presentacion de un producto			
Clase:	ColeccionProductos.java			
Función:				
Configuraciones:				
Datos de Entrada:	➢ String producto = "2221"; ➢ String presentacion = "PRES1";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
205     @Test
206     public void testEliminarPresentacion() {
207         System.out.println("eliminarPresentacion");
208         String producto = "2221";
209         String presentacion = "PRES1";
210         ColeccionProductos instance = new ColeccionProductos();
211         int expResult = 1;
            int result = instance.eliminarPresentacion(producto, presentacion);
213         if(expResult==result)
214             assertEquals(expResult, result);
215         else
216             fail("Error al eliminar la presentacion");
217     }
```

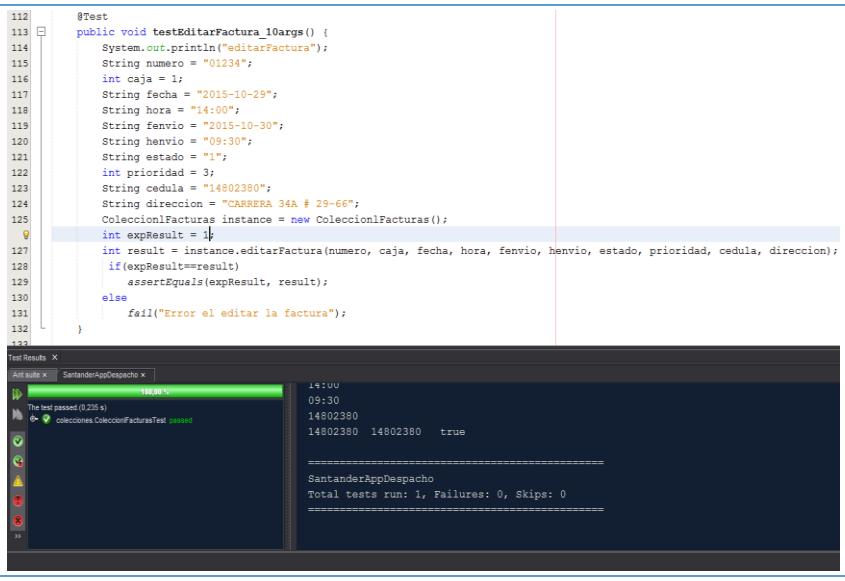
```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,145 s)
colecciones.ColeccionProductosTest passed
[TestNG] Running:
  SantanderAppDespacho

eliminarPresentacion

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
Bookmarks   Analyzer   Output
```

Prueba Unitaria				
Prueba No:	PU0060			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Agregar un envase a la factura			
Clase:	ColeccionFacturas.java			
Función:	public int registrarEnvases			
Configuraciones:				
Datos de Entrada:	➢ Factura factura			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0061			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Agregar un producto a la factura			
Clase:	ColeccionFacturas.java			
Función:	public int agregarProductos			
Configuraciones:				
Datos de Entrada:	➢ Factura factura			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

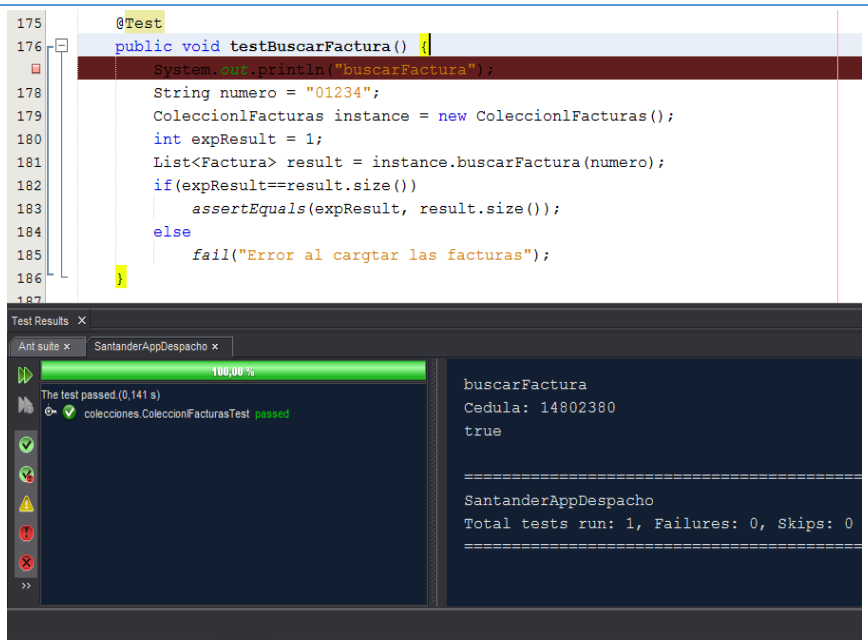
```
137     @Test
138     public void testAgregarProductos() {
139         System.out.println("agregarProductos");
140         String numero = "01234";
141         int caja = 1;
142         String codigo = "0102";
143         String presentacion = "CJ";
144         int cantidad = 3;
145         ColeccionlFacturas instance = new ColeccionlFacturas();
146         int expResult = 1;
147         int result = instance.agregarProductos(numero, caja, codigo, presentacion, cantidad);
148          if(expResult==result)
149             assertEquals(expResult, result);
150         else
151             fail("Error el agregar el producto");
152     }
```

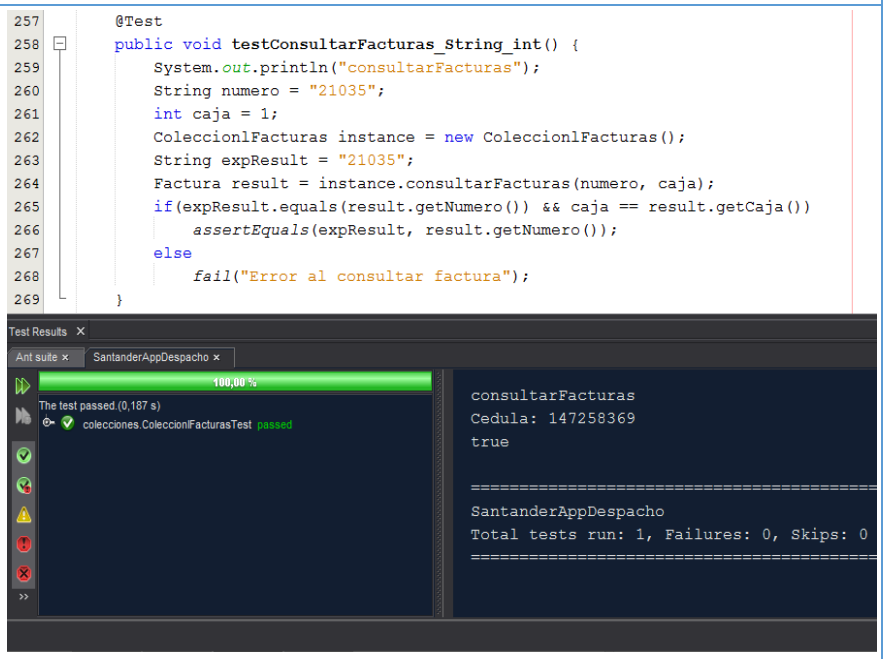
```
Test Results
Ant suite × | SantanderAppDespacho ×
100,00 %
The test passed.(0,25 s)
colecciones.ColeccionlFacturasTest passed

agregarProductos
Cedula: 14802380
true

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

Prueba Unitaria				
Prueba No:	PU0062			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Editar información de una factura			
Clase:	ColeccionFacturas.java			
Función:	public int editarFactura			
Configuraciones:				
Datos de Entrada:	➢ String numero = "01234"; ➢ int caja = 1; ➢ String fecha = "2015-10-29"; ➢ String hora = "14:00"; ➢ String fenvio = "2015-10-30"; ➢ String henvio = "09:30"; ➢ String estado = "1"; ➢ int prioridad = 3; ➢ String cedula = "14802380"; ➢ String direccion = "CARRERA 34A # 29-66";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

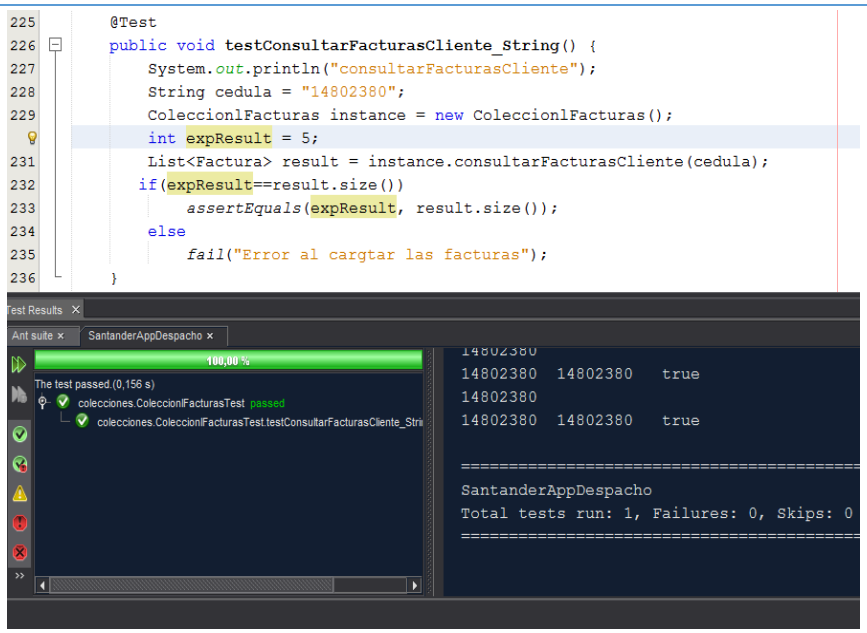
```
112     @Test
113     public void testEditarFactura_10args() {
114         System.out.println("editarFactura");
115         String numero = "01234";
116         int caja = 1;
117         String fecha = "2015-10-29";
118         String hora = "14:00";
119         String fenvio = "2015-10-30";
120         String henvio = "09:30";
121         String estado = "1";
122         int prioridad = 3;
123         String cedula = "14802380";
124         String direccion = "CARRERA 34A # 29-66";
125         ColeccionlFacturas instance = new ColeccionlFacturas();
            int expResult = 1;
127         int result = instance.editarFactura(numero, caja, fecha, hora, fenvio, henvio, estado, prioridad, cedula, direccion);
128          if(expResult==result)
129             assertEquals(expResult, result);
130         else
131             fail("Error el editar la factura");
132     }
```

```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed. (0,235 s)
coleccines.ColeccionlFacturasTest passed
14:00
09:30
14802380
14802380  14802380   true

============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
============================================
```

Prueba Unitaria				
Prueba No:	PU0063			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar una factura por el numero			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> buscarFactura			
Configuraciones:				
Datos de Entrada:	➢ String numero = "01234";			
Salida esperada:	Lista de objetos factura de longitud 1			
Salida obtenida:	Lista de objetos factura de longitud 1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

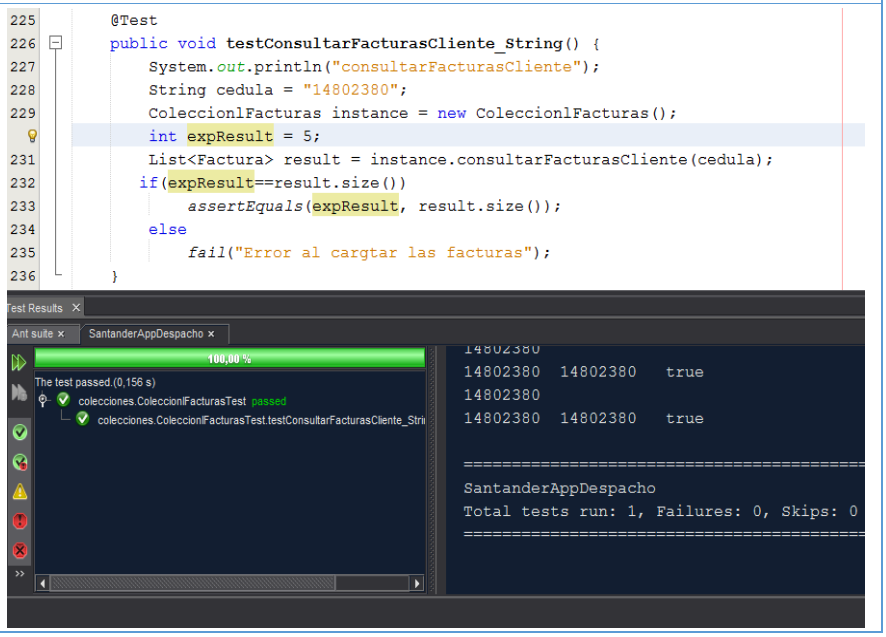
Prueba Unitaria				
Prueba No:	PU0064			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar factura con consecutivo y numero de caja			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasEstado			
Configuraciones:				
Datos de Entrada:	➢ String numero = "21035"; ➢ int caja = 1;			
Salida esperada:	Objeto factura			
Salida obtenida:	Objeto factura			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
257     @Test
258     public void testConsultarFacturas_String_int() {
259         System.out.println("consultarFacturas");
260         String numero = "21035";
261         int caja = 1;
262         ColeccionlFacturas instance = new ColeccionlFacturas();
263         String expResult = "21035";
264         Factura result = instance.consultarFacturas(numero, caja);
265         if(expResult.equals(result.getNumero()) && caja == result.getCaja())
266             assertEquals(expResult, result.getNumero());
267         else
268             fail("Error al consultar factura");
269     }
```

Prueba Unitaria				
Prueba No:	PU0065			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar facturas con identificación del cliente			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasCliente			
Configuraciones:				
Datos de Entrada:	➢ String cedula = "14802380";			
Salida esperada:	Lista de objetos factura de tamaño 5			
Salida obtenida:	Lista de objetos factura de tamaño 5			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

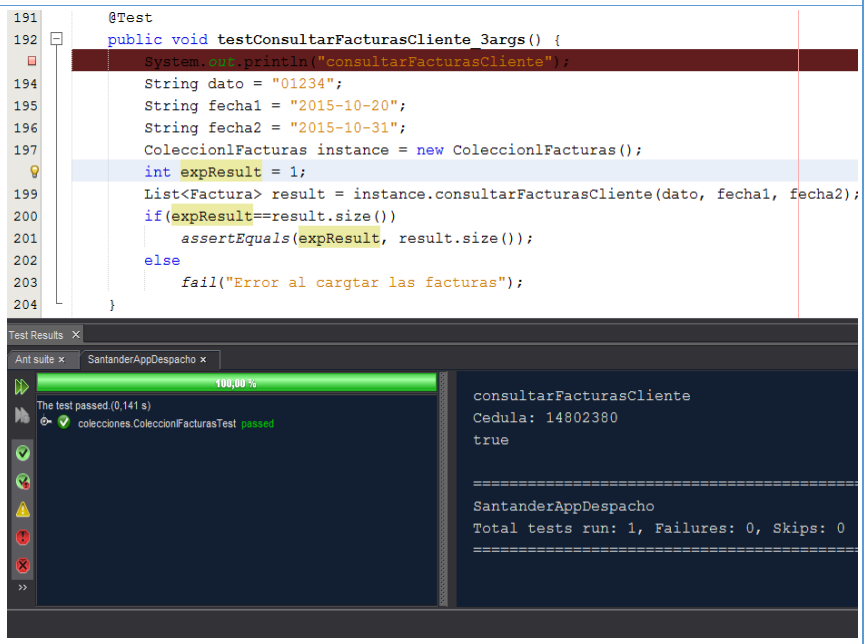
```
225     @Test
226     public void testConsultarFacturasCliente_String() {
227         System.out.println("consultarFacturasCliente");
228         String cedula = "14802380";
229         ColeccionlFacturas instance = new ColeccionlFacturas();
            int expResult = 5;
231         List<Factura> result = instance.consultarFacturasCliente(cedula);
232        if(expResult==result.size())
233             assertEquals(expResult, result.size());
234         else
235             fail("Error al cargtar las facturas");
236     }
```

```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,156 s)
colecciones.ColeccionlFacturasTest passed
colecciones.ColeccionlFacturasTest.testConsultarFacturasCliente_Stri
14802380  14802380   true
14802380
14802380  14802380   true

SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
```

Prueba Unitaria				
Prueba No:	PU0066			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar facturas con identificación del cliente entre dos fechas			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasCliente			
Configuraciones:				
Datos de Entrada:	➢ String dato = "14802380"; ➢ String fecha1 = "2015-06-01"; ➢ String fecha2 = "2015-10-31";			
Salida esperada:	Lista de objetos factura de tamaño 5			
Salida obtenida:	Lista de objetos factura de tamaño 5			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

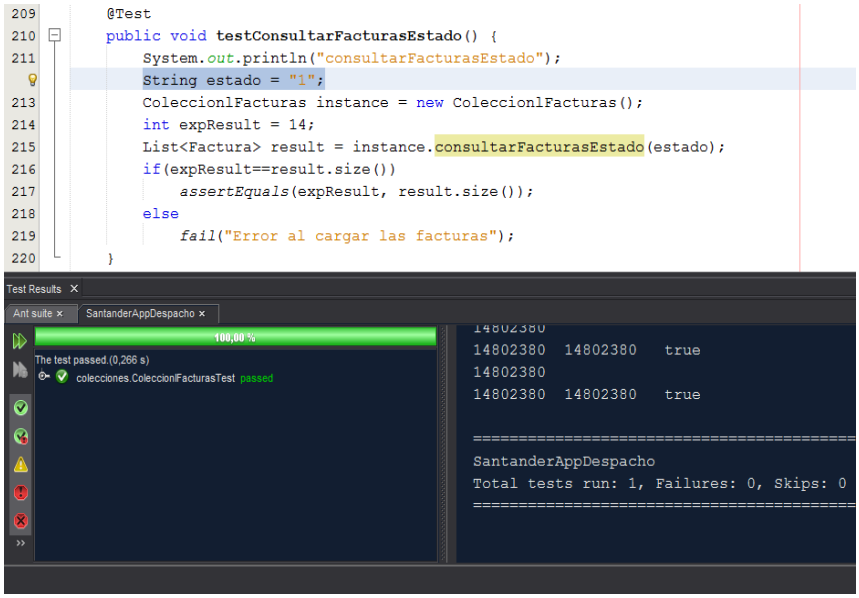
```
225     @Test
226     public void testConsultarFacturasCliente_String() {
227         System.out.println("consultarFacturasCliente");
228         String cedula = "14802380";
229         ColeccionlFacturas instance = new ColeccionlFacturas();
            int expResult = 5;
231         List<Factura> result = instance.consultarFacturasCliente(cedula);
232        if(expResult==result.size())
233             assertEquals(expResult, result.size());
234         else
235             fail("Error al cargtar las facturas");
236     }
```

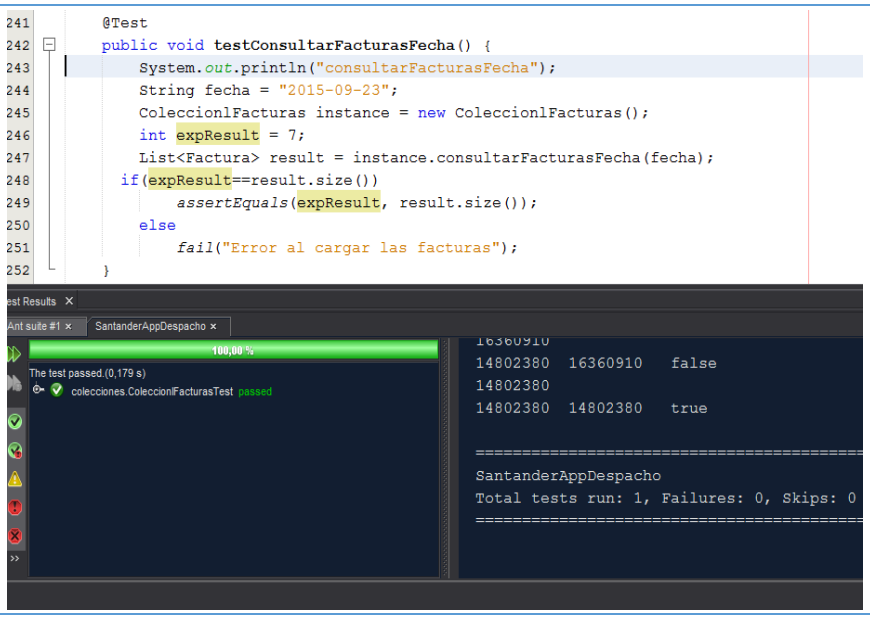
```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,156 s)
colecciones.ColeccionlFacturasTest passed
colecciones.ColeccionlFacturasTest.testConsultarFacturasCliente_Stri
14802380
14802380  14802380   true
14802380
14802380  14802380   true

=======================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
=======================================
```

Prueba Unitaria				
Prueba No:	PU0067			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar facturas con consecutivo de factura entre dos fechas			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasCliente			
Configuraciones:				
Datos de Entrada:	➢ String dato = "01234"; ➢ String fecha1 = "2015-06-01"; ➢ String fecha2 = "2015-10-31";			
Salida esperada:	Lista de objetos factura de tamaño 1			
Salida obtenida:	Lista de objetos factura de tamaño 1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
191     @Test
192     public void testConsultarFacturasCliente_3args() {
            System.out.println("consultarFacturasCliente");
194         String dato = "01234";
195         String fecha1 = "2015-10-20";
196         String fecha2 = "2015-10-31";
197         ColeccionlFacturas instance = new ColeccionlFacturas();
            int expResult = 1;
199         List<Factura> result = instance.consultarFacturasCliente(dato, fecha1, fecha2);
200         if(expResult==result.size())
201             assertEquals(expResult, result.size());
202         else
203             fail("Error al cargtar las facturas");
204     }
```

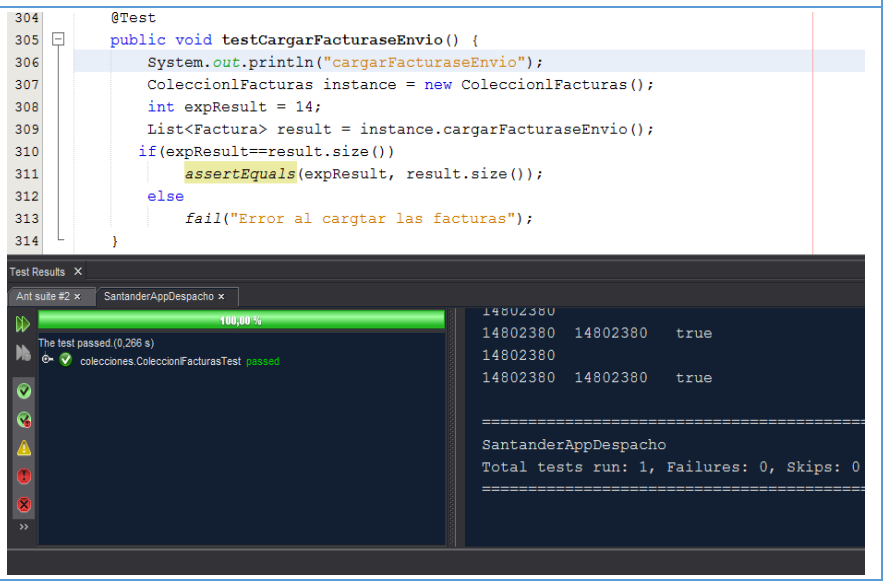
```
Test Results
Ant suite ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,141 s)
colecciones.ColeccionlFacturasTest passed

consultarFacturasCliente
Cedula: 14802380
true

==========
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
==========
```

Prueba Unitaria				
Prueba No:	PU0068			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Consultar facturas por estado			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasEstado			
Configuraciones:				
Datos de Entrada:	➢ String estado = "1";			
Salida esperada:	Lista de objetos factura de tamaño 14			
Salida obtenida:	Lista de objetos factura de tamaño 14			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0069			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar facturas por fecha			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> consultarFacturasFecha			
Configuraciones:				
Datos de Entrada:	➢ String fecha = "2015-09-23";			
Salida esperada:	Lista de objetos factura de tamaño 7			
Salida obtenida:	Lista de objetos factura de tamaño 7			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

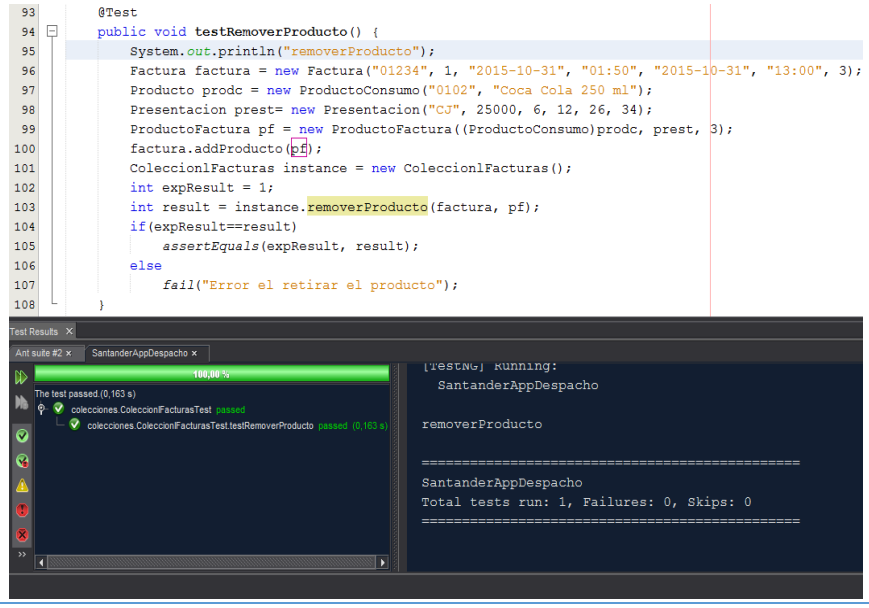
```
241     @Test
242     public void testConsultarFacturasFecha() {
243         System.out.println("consultarFacturasFecha");
244         String fecha = "2015-09-23";
245         ColeccionlFacturas instance = new ColeccionlFacturas();
246         int expResult = 7;
247         List<Factura> result = instance.consultarFacturasFecha(fecha);
248      if(expResult==result.size())
249             assertEquals(expResult, result.size());
250         else
251             fail("Error al cargar las facturas");
252     }
```

```
est Results
Ant suite #1
SantanderAppDespacho
100,00 %
The test passed.(0,179 s)
colecciones.ColeccionlFacturasTest passed
16360910
14802380  16360910   false
14802380
14802380  14802380   true

==========
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
==========
```

Prueba Unitaria				
Prueba No:	PU0070			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Listar las facturas pendientes por enviar.			
Clase:	ColeccionFacturas.java			
Función:	public List<Factura> cargarFacturaseEnvio			
Configuraciones:				
Datos de Entrada:				
Salida esperada:	Lista de objetos factura de tamaño 14			
Salida obtenida:	Lista de objetos factura de tamaño 14			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

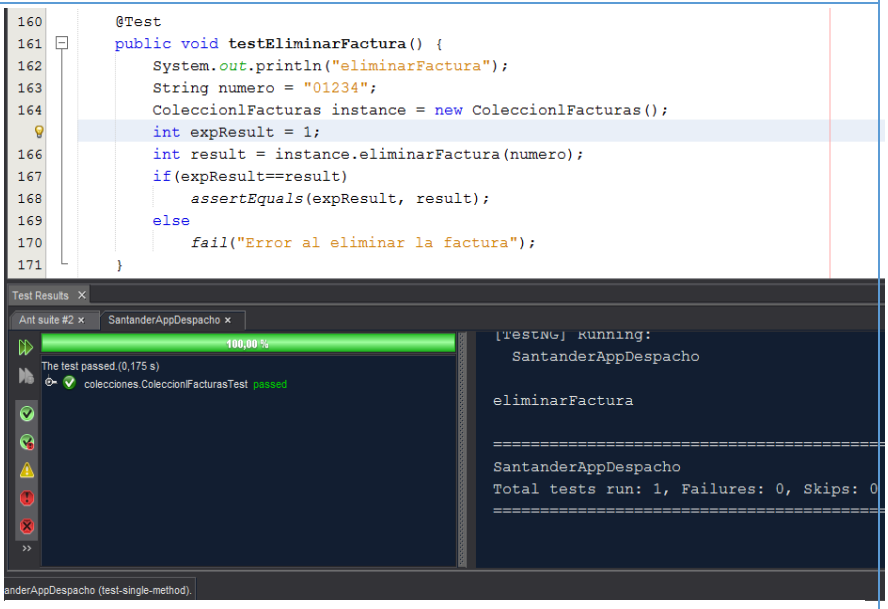
```
304     @Test
305     public void testCargarFacturaseEnvio() {
306         System.out.println("cargarFacturaseEnvio");
307         ColeccionlFacturas instance = new ColeccionlFacturas();
308         int expResult = 14;
309         List<Factura> result = instance.cargarFacturaseEnvio();
310        if(expResult==result.size())
311             assertEquals(expResult, result.size());
312         else
313             fail("Error al cargtar las facturas");
314     }
```

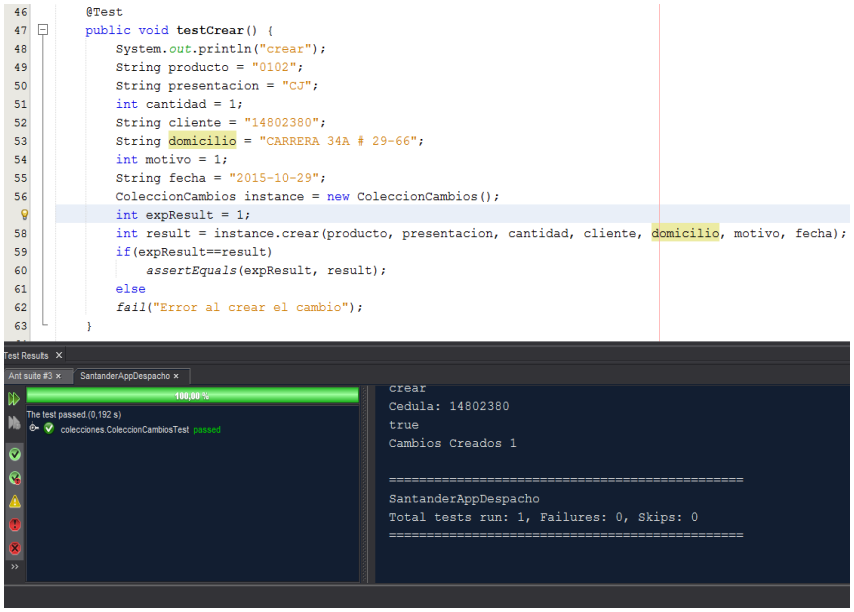
```
Test Results
Ant suite #2 ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,266 s)
colecciones.ColeccionlFacturasTest passed

14802380
14802380  14802380   true
14802380
14802380  14802380   true

==========================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
==========================================
```

Prueba Unitaria				
Prueba No:	PU0071			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Retirar un producto de una factura			
Clase:	ColeccionFacturas.java			
Función:	public int removerProducto			
Configuraciones:				
Datos de Entrada:	➢ Factura factura ➢ ProductoFactura productoFactura			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
93      @Test
94      public void testRemoverProducto() {
95          System.out.println("removerProducto");
96          Factura factura = new Factura("01234", 1, "2015-10-31", "01:50", "2015-10-31", "13:00", 3);
97          Producto prodc = new ProductoConsumo("0102", "Coca Cola 250 ml");
98          Presentacion prest= new Presentacion("CJ", 25000, 6, 12, 26, 34);
99          ProductoFactura pf = new ProductoFactura((ProductoConsumo)prodc, prest, 3);
100         factura.addProducto(pf);
101         ColeccionlFacturas instance = new ColeccionlFacturas();
102         int expResult = 1;
103         int result = instance.removerProducto(factura, pf);
104         if(expResult==result)
105             assertEquals(expResult, result);
106         else
107             fail("Error el retirar el producto");
108     }
```

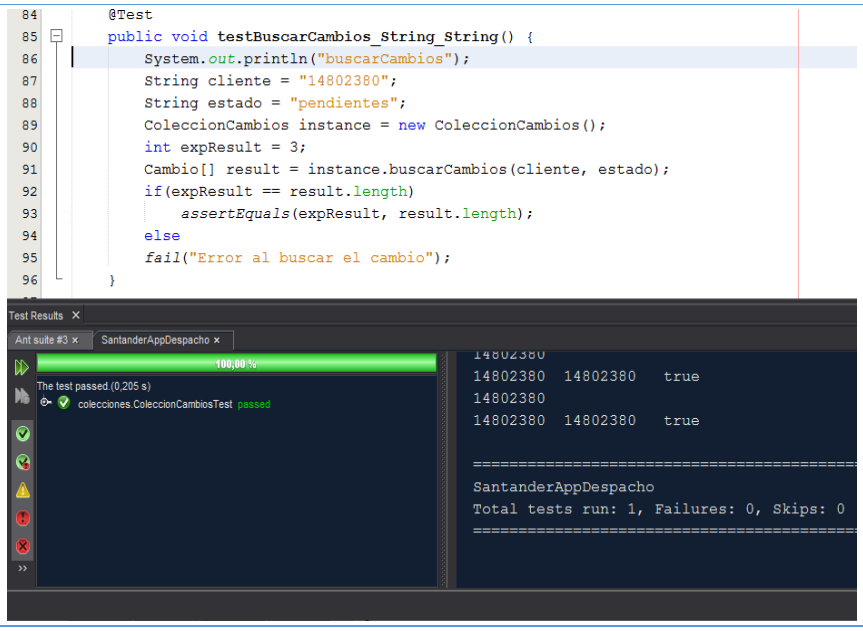
```
Test Results
Ant suite #2 ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,163 s)
colecciones.ColeccionlFacturasTest passed
colecciones.ColeccionlFacturasTest.testRemoverProducto passed (0,163 s)

[TestNG] Running:
  SantanderAppDespacho

removerProducto

===============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

Prueba Unitaria				
Prueba No:	PU0072			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Eliminar una factura			
Clase:	ColeccionFacturas.java			
Función:	public int eliminarFactura			
Configuraciones:				
Datos de Entrada:	➢ String numero = "01234";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

Prueba Unitaria				
Prueba No:	PU0073			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Ingresar un cambio			
Clase:	ColeccionCambios.java			
Función:				
Configuraciones:				
Datos de Entrada:	➢ String producto = "0102"; ➢ String presentacion = "CJ"; ➢ int cantidad = 1; ➢ String cliente = "14802380"; ➢ String domicilio = "CARRERA 34A # 29-66"; ➢ int motivo = 1; ➢ String fecha = "2015-10-29";			
Salida esperada:	1			
Salida obtenida:	1			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
46      @Test
47      public void testCrear() {
48          System.out.println("crear");
49          String producto = "0102";
50          String presentacion = "CJ";
51          int cantidad = 1;
52          String cliente = "14802380";
53          String domicilio = "CARRERA 34A # 29-66";
54          int motivo = 1;
55          String fecha = "2015-10-29";
56          ColeccionCambios instance = new ColeccionCambios();
            int expResult = 1;
58          int result = instance.crear(producto, presentacion, cantidad, cliente, domicilio, motivo, fecha);
59          if(expResult==result)
60              assertEquals(expResult, result);
61          else
62          fail("Error al crear el cambio");
63      }
```

```
Test Results
Ant suite #3 ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,192 s)
colecciones.ColeccionCambiosTest passed

crear
Cedula: 14802380
true
Cambios Creados 1

============================================
SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
============================================
```

Prueba Unitaria				
Prueba No:	PU0074			
Autor:	Carlos Andres López Ramírez			
Objetivo (s):	Buscar un cambio por identificación del cliente			
Clase:	ColeccionCambios.java			
Función:	public Cambio[] buscarCambios			
Configuraciones:				
Datos de Entrada:	➢ String cliente = "14802380"; ➢ String estado = "pendientes";			
Salida esperada:	Lista de objetos Cambio de tamaño 3			
Salida obtenida:	Lista de objetos Cambio de tamaño 3			
Estado:	Correcto:	Si	Fallido:	
Error	Código:		Severidad:	
Evidencia:				

```
84      @Test
85      public void testBuscarCambios_String_String() {
86          System.out.println("buscarCambios");
87          String cliente = "14802380";
88          String estado = "pendientes";
89          ColeccionCambios instance = new ColeccionCambios();
90          int expResult = 3;
91          Cambio[] result = instance.buscarCambios(cliente, estado);
92          if(expResult == result.length)
93              assertEquals(expResult, result.length);
94          else
95          fail("Error al buscar el cambio");
96      }
```

```
Test Results
Ant suite #3 ×   SantanderAppDespacho ×
100,00 %
The test passed.(0,205 s)
colecciones.ColeccionCambiosTest passed

14802380  14802380   true
14802380
14802380  14802380   true

SantanderAppDespacho
Total tests run: 1, Failures: 0, Skips: 0
```

Información General	
Autor:	*Carlos Andrés López Ramírez*
Fecha de creación:	*Septiembre 2 de 2014*
Documentos relacionados:	*Historias de usuario.docx*

Revisión Histórica	
Revisión No:	*001*
Descripción del cambio	*Se corrigieron de algunas Historias la redacción y los criterios de aceptación.*
Fecha última modificación:	*Diciembre 16 de 2015*

PA0001

Referencias	Casos de uso CU001
Descripción	Prueba creación de roles, se sigue el criterio de aceptación establecido en los casos de uso.
Pasos	7. Se ingresa en la opción "Perfiles". 8. Se despliega una lista de perfiles y opciones de los perfiles. 9. Se entra en la opción nuevo. 10. Se despliega el formulario de creación de perfil. 11. Se llenan los campos y se seleccionan los permisos. 12. Se guarda la información y regresa a la pantalla de perfiles
Resultado Esperado	Perfil de usuario almacenado con los permisos seleccionados
Resultado Obtenido	Se guardó el perfil.
Errores	No guardaron todos los permisos

PA0002

Referencias	Casos de uso CU002
Descripción	Se debe generar un listado de roles por medio de la búsqueda
Pasos	4. Se muestra una lista de los roles existentes 5. Se ingresa el nombre o id del rol en el campo de búsqueda 6. Se muestra la lista de resultados que coincidan con la información ingresada
Resultado Esperado	Listado de roles que contenga una la palabra buscada en su nombre o descripción
Resultado Obtenido	Se generó un listado de perfiles con la palabra buscada en el nombre y/o descripción.
Errores	

PA0003

Referencias	Casos de uso CU003
Descripción	Modificar la información de un rol seleccionado.
Pasos	7. Se ingresa en la opción "Perfiles". 8. Se despliega una tabla con una lista de perfiles. 9. Se busca el perfil y se da clic sobre la fila y se da clic en la opción ver. 10. Se despliega un formulario no editable con la información del perfil. 11. Se da clic en la opción actualizar y los campos se vuelven editables. Se edita la información y se da clic en la opción guardar.
Resultados esperados	Rol con modificaciones almacenadas.
Resultado Obtenido	Se guardaron los cambios.
Errores	No se guardaron todos los cambios en los permisos

PA0004

Referencias	Casos de uso CU004
Descripción	Eliminación de roles de usuario
Pasos	5. Se ingresa en la opción "perfiles" 6. Se despliega una lista de perfiles. 7. Busca el perfil a eliminar y se da clic sobre este. 8. Se da clic en el botón eliminar y se confirma la acción.
Resultados esperados	Mensaje informando el estado de la acción
Resultado Obtenido	Mensaje informativo notificando que la acción está completa.
Errores	

PA0005

Referencias	Casos de uso CU005
Descripción	Proceso de creación de usuarios
Pasos	6. Se ingresa en la opción "Usuarios" en el menú de administración. 7. Se despliega una lista de los usuarios disponibles. 8. Se da clic en la opción "Nuevo" 9. Se despliega el formulario de creación de usuarios. 10. Se llenan los campos del formulario y se da clic en la opción guardar.
Resultados esperados	Usuario registrado del sistema registrado
Resultado Obtenido	Se creó el usuario
Errores	No cargaban los combos.

PA0006

Referencias	Casos de uso CU006
Descripción	Búsqueda de usuarios registrados
Pasos	1. Se ingresa en la opción “Usuarios” en el menú de administración. 2. Se despliega una lista de los usuarios registrados. 3. Se ingresa el número de cedula o nombre en el campo de búsqueda. 4. Se muestra una lista de resultados.
Resultados esperados	Listado de usuarios que contengan en dato ingresado.
Resultado Obtenido	Listado de usuarios que en sus datos contienen la palabra ingresada.
Errores	

PA0007

Referencias	Casos de uso CU007
Descripción	Modificación de la información del usuario.
Pasos	1. Se ingresa en la opción “Usuarios” en el menú de administración. 2. Se busca el usuario, se selecciona la fila correspondiente y se da clic en la opción ver. 3. Se despliega un formulario no editable con la información del usuario. 4. Se da clic en la opción actualizar, se cambia los datos del usuario y se da clic en la opción guardar.
Resultados esperados	Usuario con datos modificados
Resultado Obtenido	Cambios realizados correctamente.
Errores	

PA0008

Referencias	Casos de uso CU008
Descripción	Eliminación de usuarios del sistema
Pasos	1. Se ingresa al menú de usuarios se selecciona el usuario de la lista o se busca con su número de identificación 2. Se selecciona la opción eliminar usuario, el sistema muestra un mensaje pidiendo confirmar la acción.
Resultados esperados	Mensaje confirmando el estado de la acción
Resultado Obtenido	Mensaje informando que el usuario se eliminó.
Errores	

PA0009

Referencias	Casos de uso CU009
Descripción	Creación de un nuevo vehículo
Pasos	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se selecciona la opción "Nuevo". 3. Se despliega un formulario para la creación de vehículos. 4. Se llena el formulario y se da clic en la opción "Guardar" 5. Retorna a la opción "Vehículos"
Resultados esperados	Un nuevo vehículo creado
Resultado Obtenido	Se registró el vehículo correctamente
Errores	

PA0010

Referencias	Casos de uso CU010
Descripción	Buscar un vehículo por número de placa parcial o completo
Pasos	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se despliega una lista con los vehículos registrados. 3. Se ingresa el número de placa, marca o modelo del vehículo en el campo de búsqueda. 4. Muestra una lista de los vehículos que coinciden con los datos de entrada.
Resultados esperados	Listado de vehículos con placas coincidentes
Resultado Obtenido	Listado de vehículos con coincidencias en la placa
Errores	

PA0011

Referencias	Casos de uso CU011
Descripción	Editar la información de un vehículo
Pasos	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se busca el vehículo a actualizar, se selecciona el vehículo de la lista de resultados y se da clic en la opción ver. 3. Se despliega un formulario no editable con la información del vehículo, se da clic en la opción actualizar. 4. El formulario cambia a editable, se modifican los datos necesarios y se da clic en la opción guardar.
Resultados esperados	Información actualizada del vehículo
Resultado Obtenido	Listado de vehículos con coincidencias en la placa
Errores	

PA0012

Referencias	Casos de uso CU012
Descripción	Eliminar un vehículo registrado.
Pasos	1. Se ingresa en la opción "Vehículos" en el menú de administración. 2. Se busca el vehículo se da clic sobre el resultado que corresponda a la búsqueda. 3. Se da clic en la opción eliminar y se confirma la acción.
Resultados esperados	Mensaje informando el estado de la acción
Resultado Obtenido	Mensaje informando que el vehículo se elimino
Errores	

PA0013

Referencias	Casos de uso CU013
Descripción	Registro de la información de un remolque
Pasos	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se despliega una lista de los remolques existentes. 3. Se da clic en la opción "Nuevo". 4. Se despliega un formulario para la creación de remolques. 5. Se llena el formulario y se da clic en la opción "Guardar" 6. Regresa a la opción Remolques
Resultados esperados	Información registrada
Resultado Obtenido	La información del remolque se guardó correctamente
Errores	

PA0014

Referencias	Casos de uso CU014
Descripción	Búsqueda de remolques utilizando información de estos
Pasos	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se muestra una lista de los remolques existentes. 3. Se ingresa el id o tipo de enganche del remolque en el campo de búsqueda. 4. Se muestra una lista con los remolques que coincidan con la información de búsqueda.
Resultados esperados	Listado de remolques que coincidan con la información ingresada
Resultado Obtenido	Listado de remolques que contiene información ingresada para buscar
Errores	

PA0015

Referencias	Casos de uso CU015
Descripción	Búsqueda de remolques utilizando información de estos
Pasos	1. Se ingresa a la opción "Remolques" en el menú de administración. 2. Selecciona busca el remolque y se selecciona en la lista 3. Se da clic en la opción ver y se despliega un formulario no editable con la información del remolque. 4. Se modifican los datos, se guardan y se confirma la acción.
Resultados esperados	Listado de remolques que coincidan con la información ingresada
Resultado Obtenido	Listado de remolques que contiene información ingresada para buscar
Errores	

PA0016

Referencias	Casos de uso CU016
Descripción	Eliminar la información de un remolque
Pasos	1. Se ingresa en la opción "Remolques" en el menú de administración. 2. Se busca el remolque a eliminar. 3. Se selecciona el remolque y se da clic en la opción eliminar. 4. Se muestra una ventada pidiendo que confirma la acción.
Resultados esperados	Mensaje informando el estado de la acción.
Resultado Obtenido	Mensaje informando que el remolque se elimino exitosamente.
Errores	

PA0017

Referencias	Casos de uso CU017
Descripción	Registrar la información de un producto
Pasos	1. Se ingresa en la opción "Productos" en el menú de administración. 2. Se despliega una lista con los productos existentes y las opciones de producto. 3. Se da clic en la opción "Nuevo" 4. Se despliega un formulario para la creación de productos. 5. Se ingresan los datos del producto. 6. Se crean las presentaciones. 7. Se da clic en la opción "Guardar" y el sistema redirige a la pantalla de productos.
Resultados esperados	Información del producto registrada
Resultado Obtenido	Información del producto guardada correctamente.
Errores	

PA0018

Referencias	Casos de uso CU018
Descripción	Buscar un producto
Pasos	1. Se ingresa en la opción “Productos” en el menú de administración. 2. Se despliega una lista de los productos existentes y las opciones de estos. 3. Se escribe el nombre o código del producto en el campo de búsqueda. 4. Se muestra una lista de los productos que coincidan con la información ingresada.
Resultados esperados	Lista de productos con información coincidente a la ingresada
Resultado Obtenido	Lista de productos que tienen información idéntica a la escrita en el campo de búsqueda
Errores	

PA0019

Referencias	Casos de uso CU019
Descripción	Modificar la información de un producto
Pasos	1. Se ingresa en la opción “Productos” en el menú de administración. 2. Se despliega una lista de los productos existentes y las opciones de estos. 3. Se escribe el nombre o código del producto en el campo de búsqueda. 4. Se muestra una lista de los productos que coincidan con la información ingresada.
Resultados esperados	Información modificada y almacenada
Resultado Obtenido	La información modificada se guardó correctamente.
Errores	

PA0020

Referencias	Casos de uso CU020
Descripción	Eliminar la información de un producto.
Pasos	1. Se ingresa en la opción “Productos” en el menú de administración. 2. Se busca el producto y se selecciona. 3. Se da clic en la opción “Eliminar”. 4. Muestra un mensaje pidiendo confirmación de la acción. 5. Se da clic en “Aceptar”.
Resultados esperados	Mensaje confirmando el estado de la acción

Resultado Obtenido	Mensaje informando que el producto se eliminó.
Errores	

PA0021		
Referencias	Casos de uso CU021	
Descripción	Registrar la información de un cliente	
Pasos	1. Se ingresa en la opción “Clientes” en el menú de despacho. 2. Se despliega una lista con los clientes existentes y opciones. 3. Se da clic en la opción “Nuevo”. 4. Se despliega un formulario para la creación del cliente. 5. Se llena el formulario y se da clic en la opción “Guardar”. 6. Se guarda la información y redirige a la pantalla de clientes.	
Resultados esperados	Información registrada correctamente	
Resultado Obtenido	La información se registró sin errores	
Errores		

PA0022

Referencias	Casos de uso CU022
Descripción	Buscar un cliente con información parcial o completa de alguno de sus datos.
Pasos	1. Se ingresa en la opción “Clientes” en el menú de despacho. 2. Se despliega una lista con los clientes existentes. 3. Se ingresa el nombre o número de identificación en el campo de búsqueda. 4. Se muestra una lista con los clientes que coincidan con la información.
Resultados esperados	Listado de cliente que contengan el dato ingresado
Resultado Obtenido	Lista de cliente que tienen el dato ingresado
Errores	

PA0023

Referencias	Casos de uso CU023
Descripción	Modificar la información de un cliente
Pasos	1. Se ingresa en la opción “Clientes” en el menú de despacho. 2. Se busca el cliente con el número de identificación o el nombre. 3. Se selecciona el cliente de la lista y se da clic en la opción “Ver”. 4. Se selecciona la opción “Actualizar” y el formulario cambia a estado editable. 5. Se ingresa la nueva información y se da clic en la opción guardar. 6. Se muestra un mensaje informando que la información ha sido guardada.
Resultados esperados	Cambios almacenados
Resultado Obtenido	Se guardaron los cambios sin ningún error
Errores	

PA0024

Referencias	Casos de uso CU024
Descripción	Agregar un nuevo domicilio al directorio del cliente
Pasos	1. Se busca el cliente con uno de sus datos. 2. Se selecciona el cliente y se da clic en la opción editar. 3. Se da clic en la opción Actualizar. 4. En la lista de domicilios se da clic en la opción Nuevo. 5. Se llena el formulario y se da clic en la opción aceptar. 6. Se da clic en la opción Guardar.
Resultados esperados	Domicilio agregado guardado.
Resultado Obtenido	Se guardó el domicilio sin ningún inconveniente.
Errores	

PA0025

Referencias	Casos de uso CU025
Descripción	Agregar un nuevo domicilio al directorio del cliente
Pasos	1. Se busca el cliente. 2. Se selecciona de la lista y se da clic en el botón Eliminar. 3. Dar clic en el botón aceptar en la ventana emergente.
Resultados esperados	Mensaje informando el estado de la acción
Resultado Obtenido	El cliente fue eliminado.
Errores	

PA0026

Referencias	Casos de uso CU026
Descripción	Buscar una factura con su número
Pasos	1. Se ingresa en el CRUD de facturas y se da clic en el botón Nuevo. 2. Se llena el formulario y se agregan los productos. 3. Se da clic en el botón Guardar y re-dirige al listado de facturas.
Resultados esperados	Factura registrada
Resultado Obtenido	La factura se registró correctamente
Errores	

PA0027

Referencias	Casos de uso CU027
Descripción	Buscar una factura con su número o nombre del cliente
Pasos	1. Ingresar en el CRUD de facturas. 2. Ingresar el número de factura o nombre del cliente en el campo de búsqueda.
Resultados esperados	Listado de facturas con coincidencias en número o información del cliente.
Resultado Obtenido	Listado de facturas del cliente.
Errores	

PA0028

Referencias	Casos de uso CU028
Descripción	Configurar las distancias entre clientes
Pasos	1. Se ingresa en el CRUD de rutas 2. Se selecciona la opción Configurar. 3. Se busca a dirección de destino y se selecciona. 4. Se selecciona la dirección origen y se da clic en el botón editar. 5. Se llena el formulario y se da clic en el botón guardar.
Resultados esperados	Distancia entre dos direcciones almacenada.
Resultado Obtenido	Se guardó la distancia entre cada par de direcciones configurada.
Errores	En el listado de direcciones origen se repiten las direcciones.

PA0029

Referencias	Casos de uso CU029
Descripción	Consultar rutas en diferentes estados.
Pasos	1. Se ingresa en el CRUD de rutas. 2. Se selecciona el estado de la ruta. 3. Se selecciona un rango de fechas para la búsqueda y se da clic en el botón busca.
Resultados esperados	Listado de rutas entre el rango de fechas seleccionado.
Resultado Obtenido	Listado de rutas
Errores	1. Aparecen facturas que no están en el rango de fechas. 2. No se muestran resultados.

PA0030

Referencias	Casos de uso CU030
Descripción	Cancelar el envío de facturas.
Pasos	1. Se busca la ruta. 2. Se selecciona la ruta del listado. 3. Se da clic en el botón editar. 4. Se selecciona la factura del listado y se da clic en el botón retirar.
Resultados esperados	Factura retirada de la ruta
Resultado Obtenido	Se retiró la factura de la ruta.
Errores	

PA0031

Referencias	Casos de uso CU032
Descripción	Relacionar facturas a una ruta
Pasos	1. Se ingresa en el CRUD de rutas. 2. Se da clic en el botón Nuevo 3. Se muestra el listado de facturas disponibles, vehículos disponibles y cambios. 4. Se seleccionan los vehículos, facturas y cambio a enviar. 5. Se da clic en la opción generar. 6. Redirige al listado de rutas.
Resultados esperados	Las facturas aparecen en relacionadas a la ruta.
Resultado Obtenido	Las facturas se relacionaron a una ruta.
Errores	

PA0032

Referencias	Casos de uso CU03
Descripción	Ingresar un cambio solicitado por un cliente.
Pasos	1. Ingresa en el CRUD de cambios. 2. Selecciona la opción Nuevo. 3. Llena el formulario. 4. Da clic en la opción guardar.
Resultados esperados	Cambio registrado.
Resultado Obtenido	Se registro el cambio correctamente.
Errores	

MANUAL DE USUARIO

CONTROL DE RUTAS Y DESPACHO DE PEDIDOS

Contenido

1. PERFILES.

1.1. BUSCAR PERFILES

Ingrese a la aplicación con el usuario y contraseña admin

Ingrese en la opción Perfiles que se encuentra en la barra de menú.

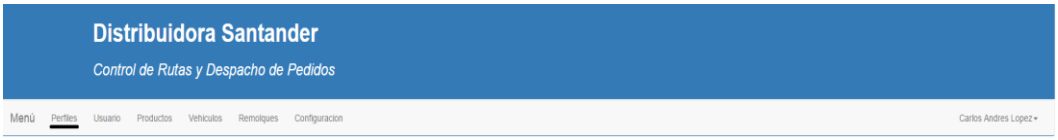


Imagen 1

Al ingresar se mostrara una tabla con los perfiles existentes y las opciones de “Crear”, “Editar” y “Eliminar”.

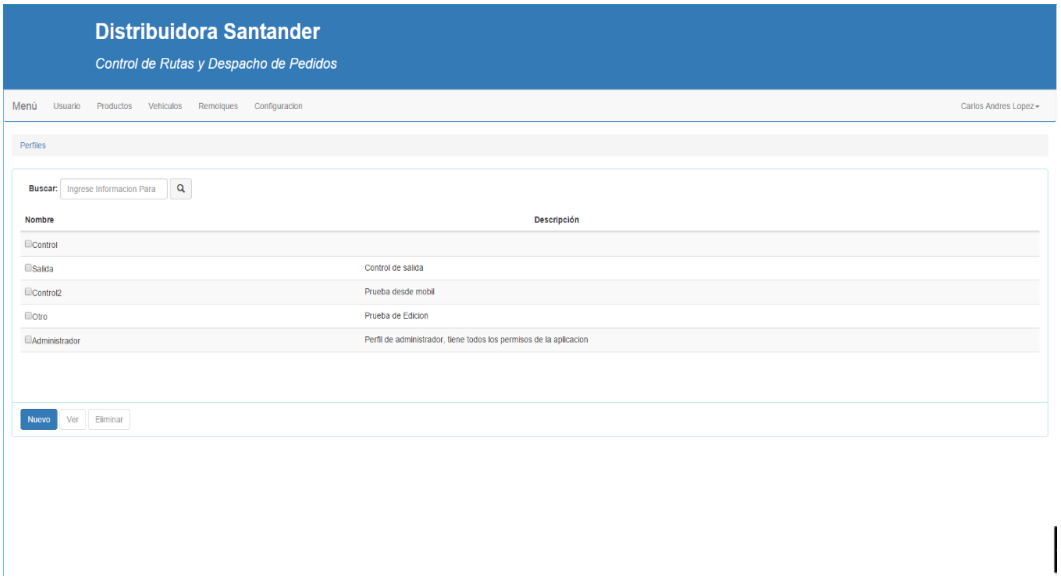


Imagen 2.

Para buscar en el campo buscar escriba el nombre del perfil o un texto que se pueda encontrar en el nombre o la descripción.

1.2. CREAR UN PERFIL.

Para crear un perfil de clic en el botón “Nuevo” en la parte inferior derecha de la tabla, se desplegara el formulario de creación de perfiles ***Imagen 3***.

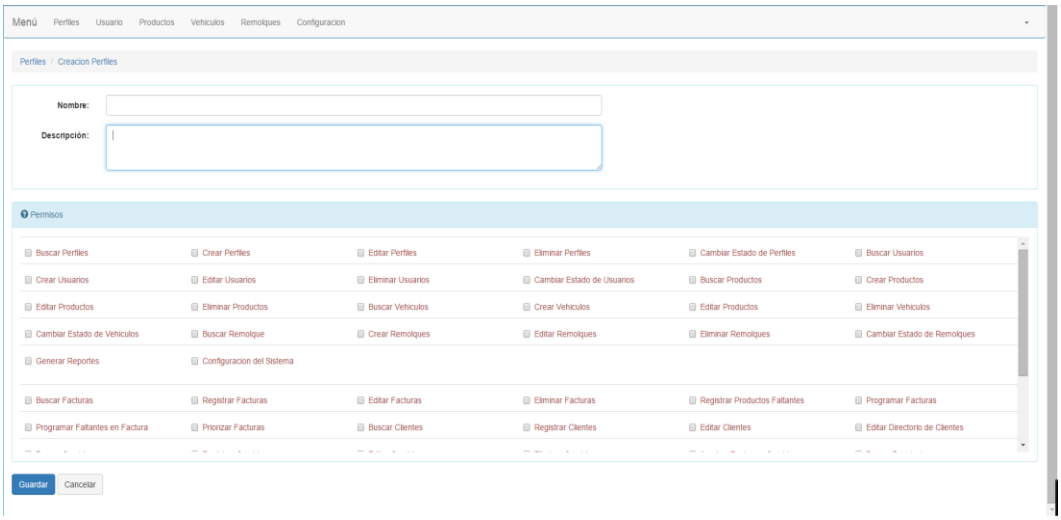
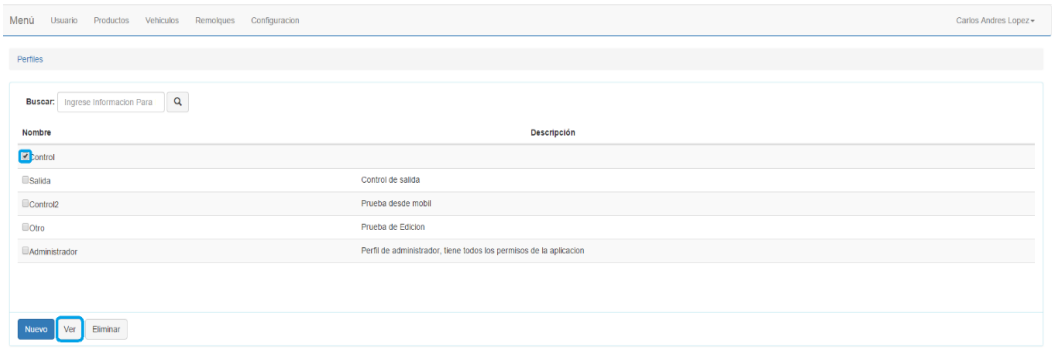


Imagen 3.

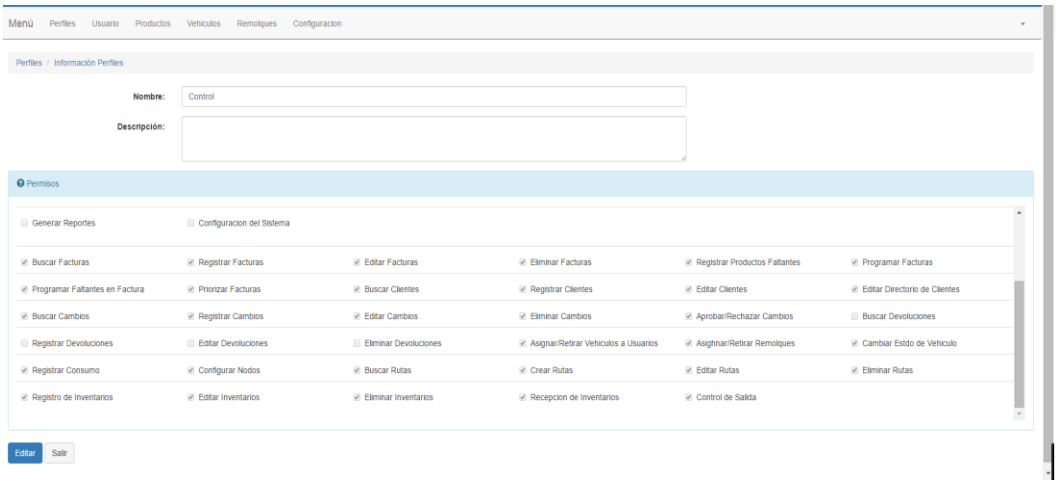
Este formulario cuenta con los campos nombre y descripción, el nombre de los perfiles no puede ser repetido, en el campo descripción se ingresa una descripción del perfil y en la parte inferior hay una lista de los permisos que tendrá el perfil.

1.3. EDITAR UN PERFIL.

En la pantalla de búsqueda seleccione el perfil a editar dando clic en el casilla de la primer columna y de clic en el botón “Ver”.



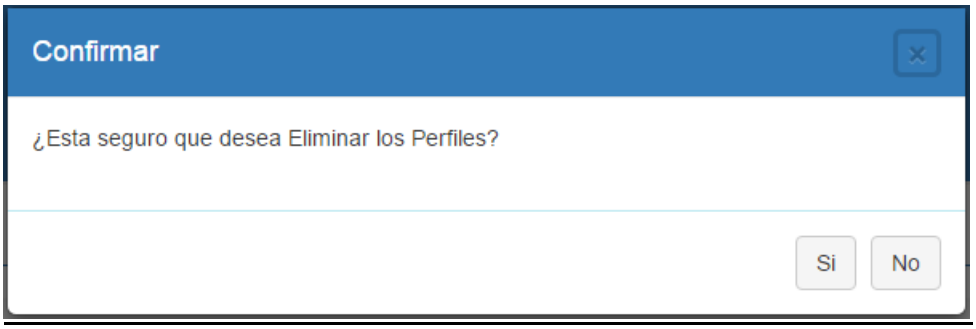
Se desplegara la información del perfil en el siguiente formulario.



De cli en el botón “Editar” para modificar el perfil y al terminar de clic en el botón “Guardar”. Para volver a la pantalla de búsqueda de clic en el botón “Salir”

1.4. ELIMINAR UN PERFIL

En la pantalla de búsqueda de perfiles seleccione el perfil y de clic en el botón “Eliminar”. Aparecerá un mensaje pidiendo confirmar la acción:



2. USUARIOS.

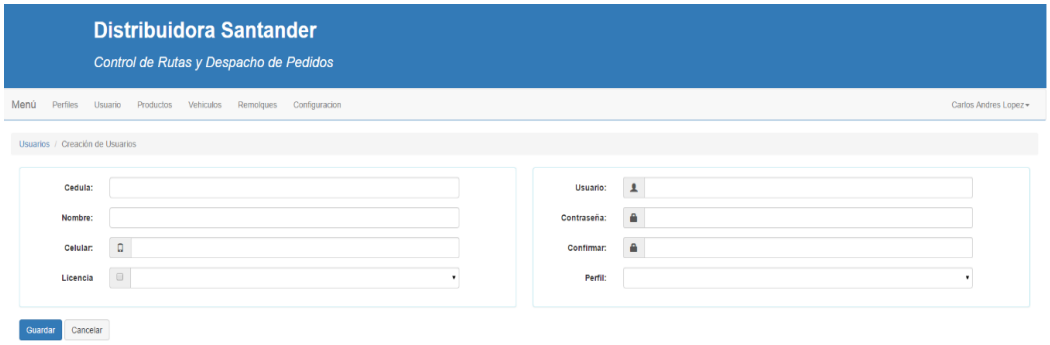
2.1. BUSCAR UN USUARIO.

Ingrese en la opción Usuarios que se encuentra en la barra de menú. Igual que en la pantalla de perfiles se visualiza una lista de usuarios y un campo para la búsqueda.

Un usuario puede ser buscado por número de identificación, nombre o usuario, completo o parcial.

2.2. CREAR UN USUARIO.

Para crear un usuario de clic en el botón “Nuevo” en la parte inferior de la tabla, a continuación se desplegara el formulario de creación de usuarios.

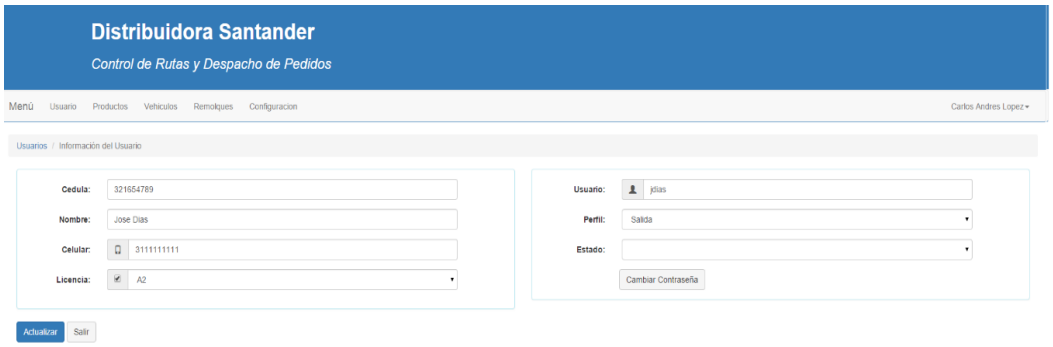


El formulario solicita los datos básicos para el usuario, la licencia es opcional y no es importante a menos que el usuario sea domicilio. Las contraseñas son encriptados en md5.

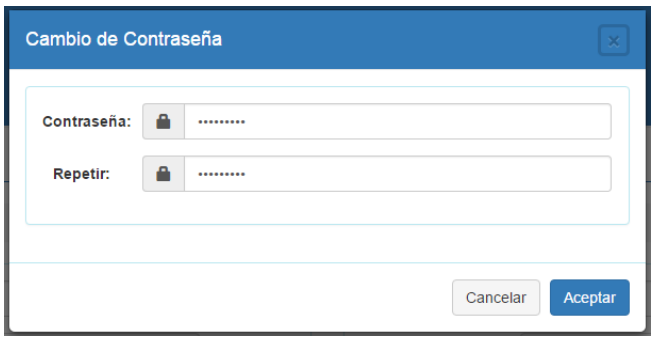
2.3. EDITAR UN USUARIO.

Para editar un usuario, en la pantalla de búsqueda de usuario seleccione un usuario de la lista y de clic en el botón “Ver”

A continuación se desplegara un formulario con la información del usuario.



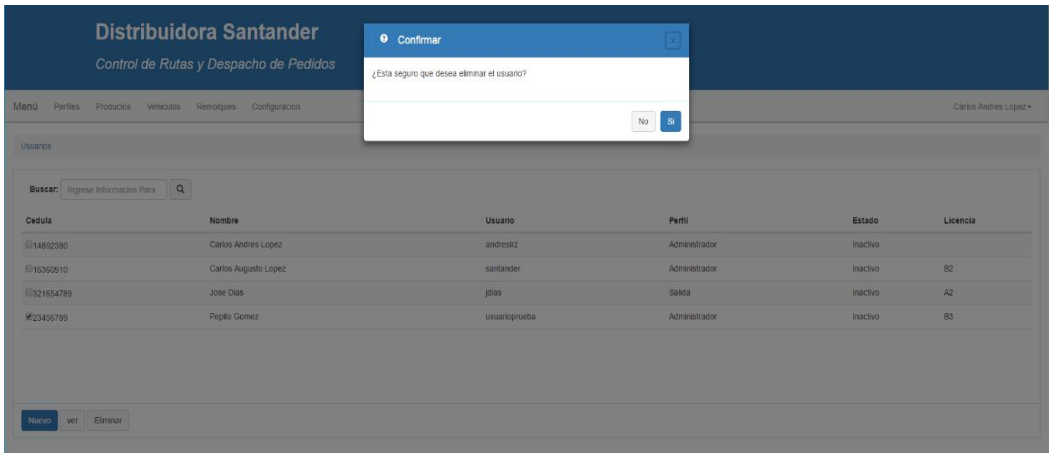
De clic en el botón actualizar para cambiar la información del usuario. Para cambiar la contraseña de clic en el botón “Cambiar Contraseña”, a continuación se abrirá el formulario para cambio de contraseña.



El cambio de contraseña debe ser una opción permitida preferiblemente a un usuario administrador, ya que en este formulario no se solicita confirmación, teniendo en cuenta que el usuario olvido su contraseña. Estas no se pueden recuperar por el método de encriptación utilizado.

2.4. ELIMINAR UN USUARIO.

Para eliminar un usuario se realiza el mismo proceso utilizado para eliminar un perfil, seleccione un usuario de la lista y de clic en el botón “Eliminar”.

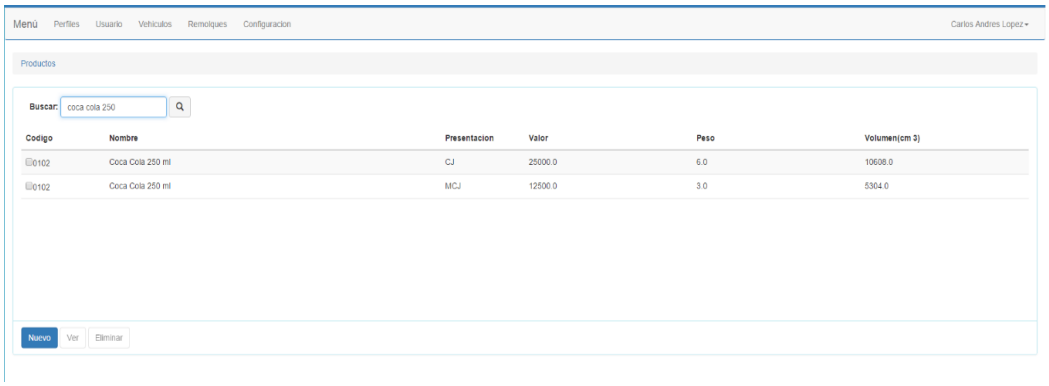


Confirme la acción y se mostrara un mensaje informando si esta se realizó correctamente o no.

3. PRODUCTOS.

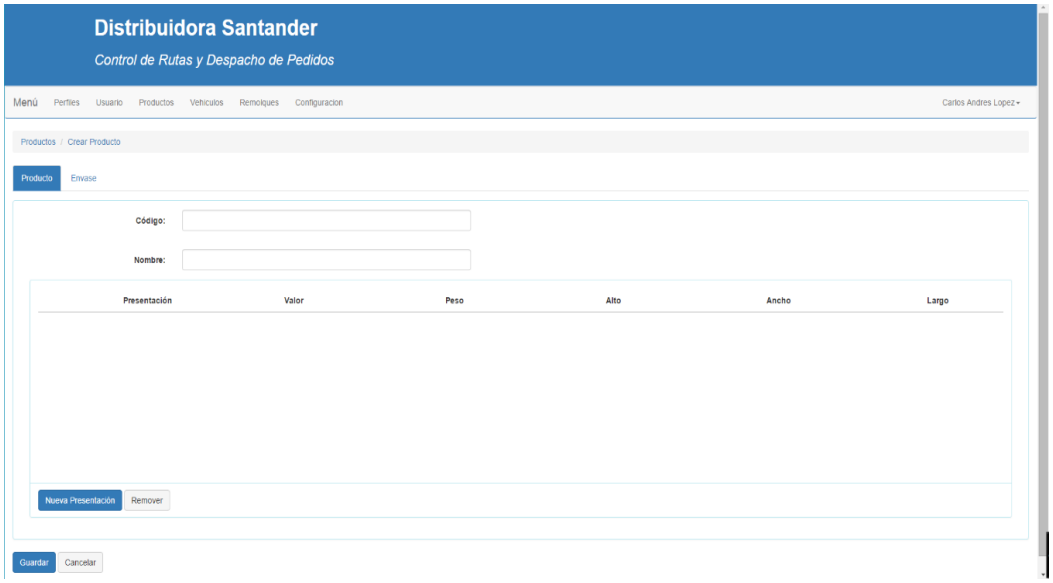
3.1. BUSCAR UN PRODUCTO.

Igual que en las secciones anteriores la búsqueda de productos se realiza escribiendo el código o nombre del producto en el campo de búsqueda. Lo resultados pueden ser múltiples ya que los productos pueden tener múltiples presentaciones.



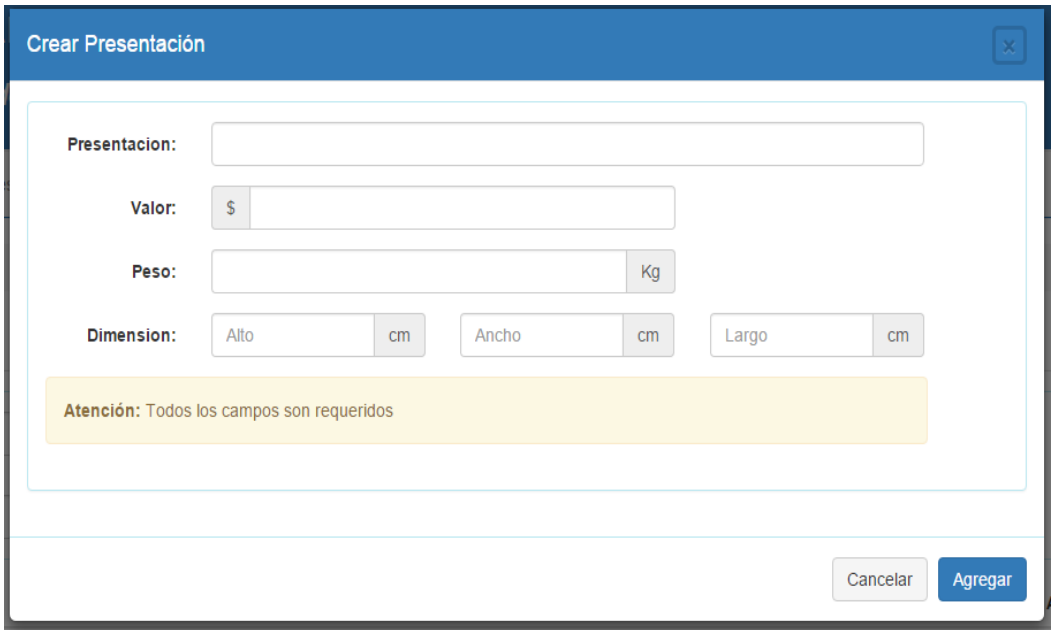
3.2. CREAR UN PRODUCTO.

Para crear un producto de clic en el botón “Nuevo”. En esta ventana hay dos pestañas “Producto” y “Envase”.

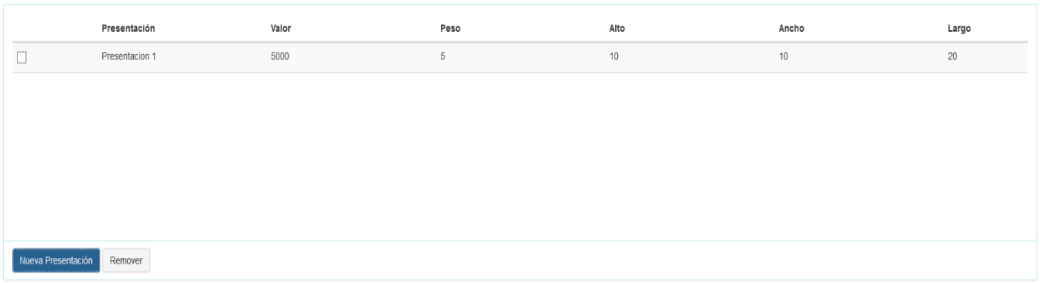


En producto se crean los productos generales, los envases, a pesar de ser un producto, su tratamiento es diferente ya que con estos se crean los inventarios de las rutas.

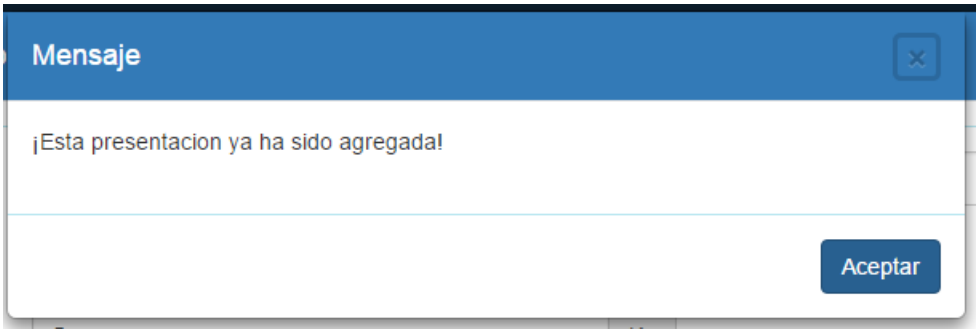
Seleccione el formulario del producto a crear, llene los campos del formulario, en caso que el código del producto ya esté en uso se mostrara un mensaje. En la pestaña productos se crea el producto con al menos una presentación, para crear las presentaciones de clic en el botón “Nueva Presentación” el cual desplegara un formulario emergente para la creación de esta.



En este formulario se solicita información como el peso y medidas del producto los cuales son parámetros para el cálculo de ruta, el cual se explicara más adelante. Llena el formulario y de clic en la opción aceptar para agregar la presentación a la lista.

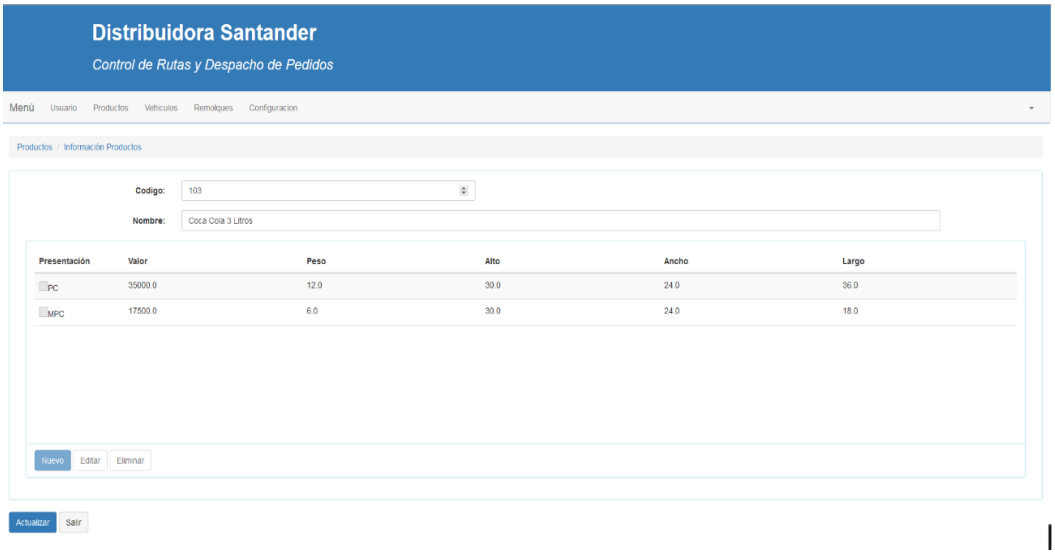


Si se intenta agregar dos veces la misma presentación, se mostrara un mensaje informando que la presentación ya ha sido agregada



3.3. EDITAR UN PRODUCTO.

Para editar un producto, seleccione uno de la lista en la pantalla de búsqueda y de clic en el botón “Ver”, esto desplegara un formulario con la información del producto. De clic en el botón “Actualizar”.



Modifique la información del producto y de cli en el botón “Guardar”. En el caso de las presentaciones los cambios son guardados al confirmar el cambio, en esta pantalla se puede agregar, modificar o eliminar presentaciones, estas opciones se activaran al dar clic en el botón “Actualizar”

3.4. ELIMINAR UN PRODUCTO.

En la pantalla de búsqueda, busque y seleccione el producto, y de clic en el botón “Eliminar”, se desplegara un mensaje pidiendo confirmar la acción. Si el producto ha sido relacionado no se podrá eliminar, en este caso se mostrara un mensaje informando que no se puede eliminar el producto.

4. VEHÍCULOS.

4.1. BUSCAR UN VEHÍCULO.

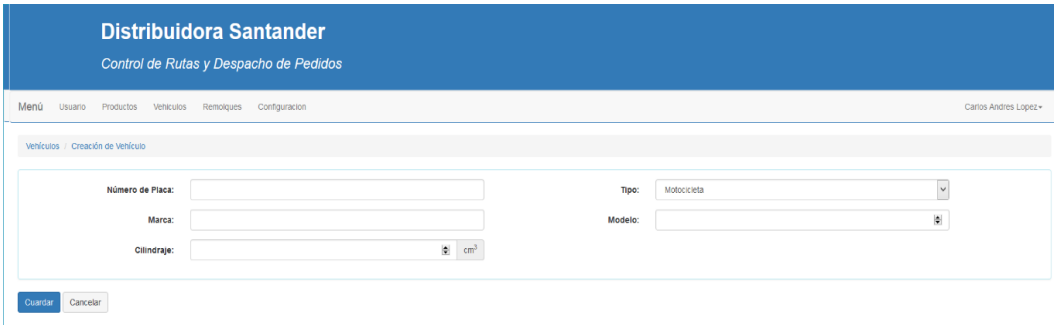
La búsqueda de vehículo, al igual que las anteriores, se debe escribir el dato completo o parcial a buscar en el campo de búsqueda, automáticamente se mostraran los resultados.

4.2. CREAR UN VEHÍCULO.

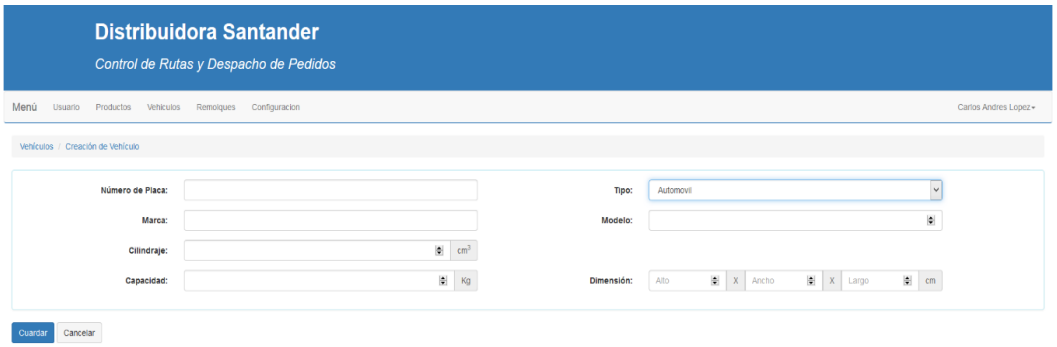
Para crear un vehículo, en la ventana de búsqueda de clic en el botón “Nuevo” en la parte inferior de la tabla, se desplegara el formulario de creación de vehículos. Este formulario cambia los campos dependiendo del tipo de vehículo seleccionado.

Se manejan dos tipos de vehículo, automóvil y motocicleta. Para el automóvil se incluye en el formulario los campos de las medidas y capacidad de carga, estos no se incluyen en las motocicletas ya que a estas se les acopla un remolque.

En la siguiente imagen se muestra el formulario de creación de motocicleta.



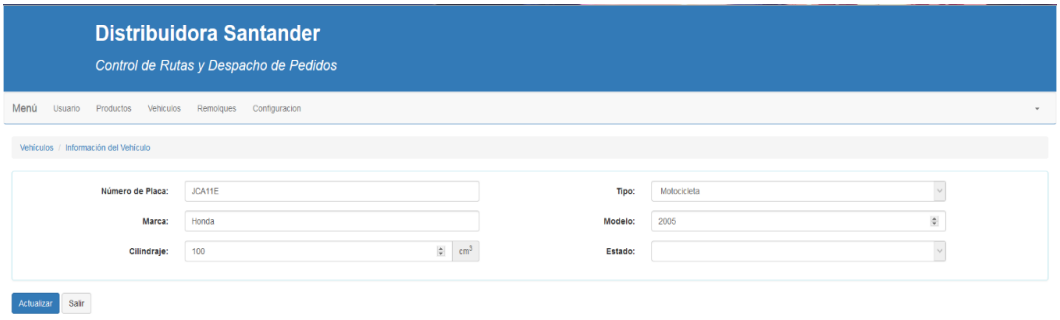
En el campo “Tipo” se selecciona el tipo de vehículo a crear, al cambiar este campo se agregan o quitan los campos, a continuación se muestra el formulario de creación de automóviles.



Llene el formulario correspondiente al vehículo a crear y de clic en la opción “Guardar”

4.3. EDITAR UN VEHÍCULO.

Para editar un vehículo, en la pantalla de búsqueda seleccione el que desea modificar y de clic en el botón “Ver”. Se desplegara un formaría con la información del vehículo.



De clic en el botón “Actualizar” para habilitar la edición y modifique la información requerida. Después de cambiar los datos de clic en el botón “Guardar”

4.4. ELIMINAR UN VEHÍCULO.

Para eliminar un vehículo seleccione de la lista el que desea retirar y de clic en la opción “Eliminar”, se desplegara un mensaje solicitando confirmar la acción, confirme la acción para terminar.

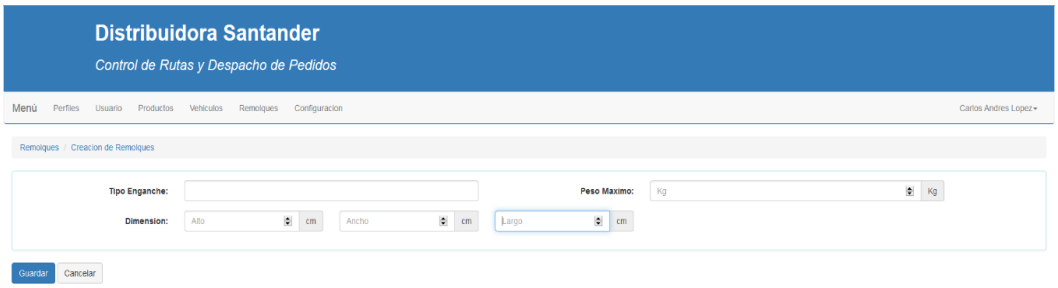
5. REMOLQUES.

5.1. BUSCAR UN REMOLQUE.

Escriba en el campo de búsqueda el id del remolque y se actualizara la tabla con los resultados como en las secciones anteriores.

5.2. CREAR UN REMOLQUE.

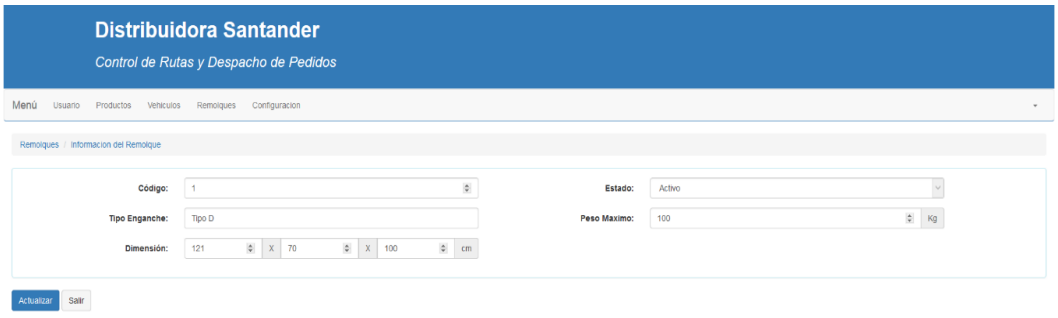
En la pantalla de búsqueda de clic en el botón “Nuevo” y a continuación se desplegara el formulario de creación de remolques.



Este formulario recibe una descripción del tipo de enganche y la información de capacidad de carga y medidas del remolque. Al terminar de clic en el botón “Guardar”

5.3. EDITAR UN REMOLQUE.

Busque y seleccione el remolque a modificar en la pantalla de búsqueda y a continuación de clic en el botón “Ver”. Se desplegara un formulario con la información del remolque.



De clic en el botón “Actualizar” para modificar la información, al terminar de clic en el botón “Guardar”

5.4. ELIMINAR UN REMOLQUE.

Busque y seleccione el remolque a eliminar en la pantalla de búsqueda y de clic en el botón “Eliminar”, se desplegara un mensaje solicitando confirmar la acción. Confirme para terminar.

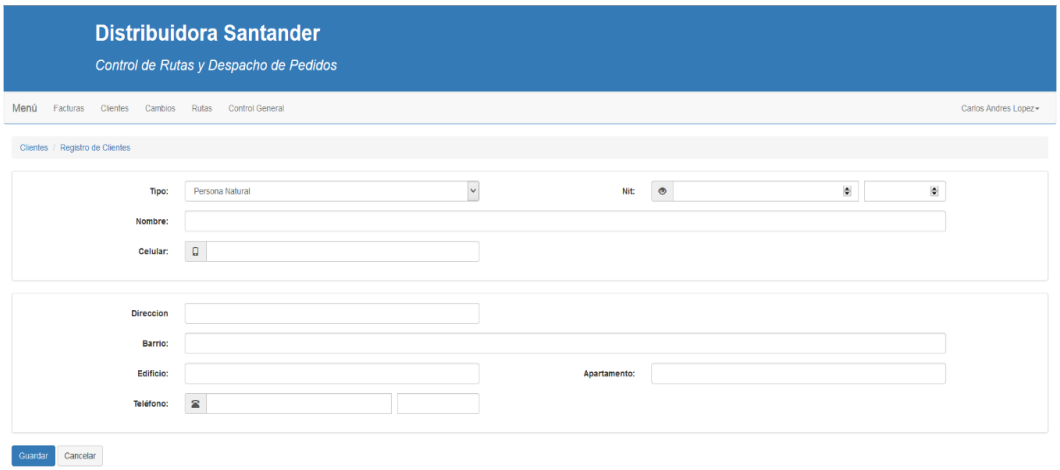
6. CLIENTES.

6.1. BUSCAR UN CLIENTE.

Para buscar un cliente escriba en el campo de búsqueda el número de identificación, nombre o celular, la tabla se actualizara con los resultados.

6.2. REGISTRAR UN CLIENTE.

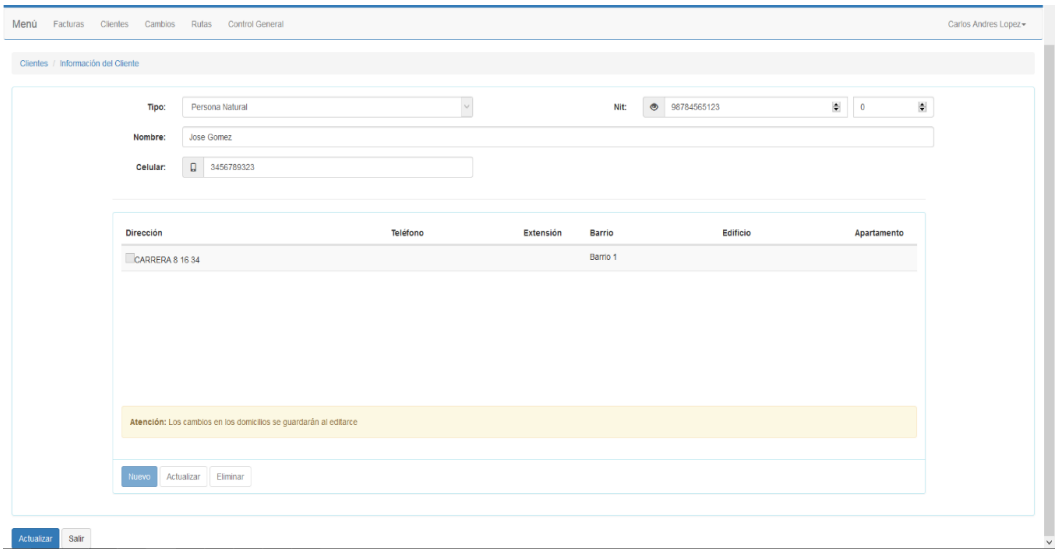
En la pantalla de búsqueda de clic en el botón “Nuevo” y a continuación se desplegara el formulario de registro de clientes.



Ingrese la información personal y de contacto del cliente, los campos nombre, identificación y dirección son obligatorios.

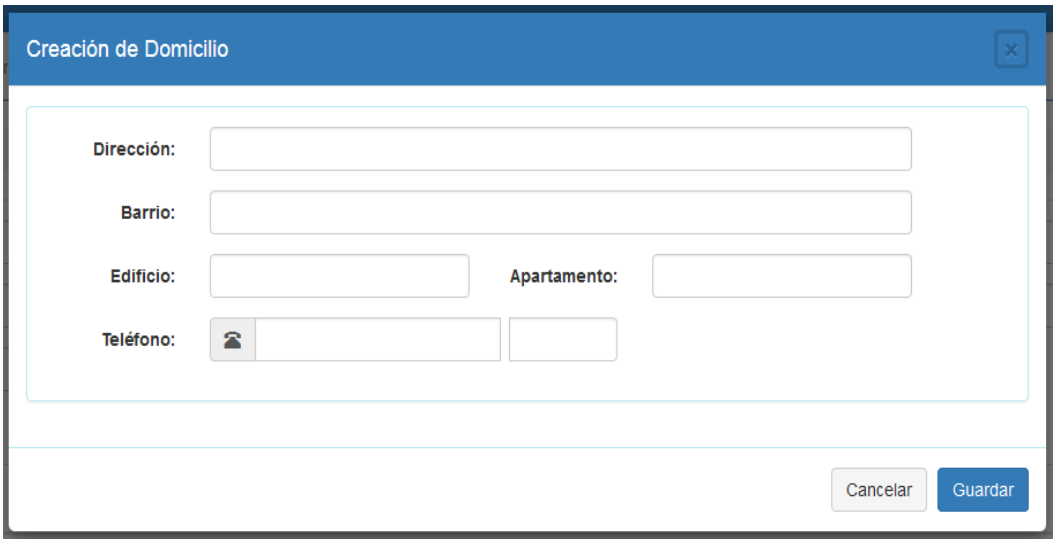
6.3. EDITAR UN CLIENTE.

Busque y seleccione el cliente a editar en la pantalla de búsqueda y de clic en el botón “Ver”, se desplegara un formulario con la información del cliente. Esta pantalla además muestra los domicilios del usuario en una tabla y permite agregar o quitar domicilios de manera similar a las presentaciones de los productos.



De clic en el botón “Actualizar” para habilitar la edición.

Para agregar nuevos domicilios, de clic en el botón nuevo en la parte inferior de la respectiva tabla y se desplegara el formulario de registro de domicilios.

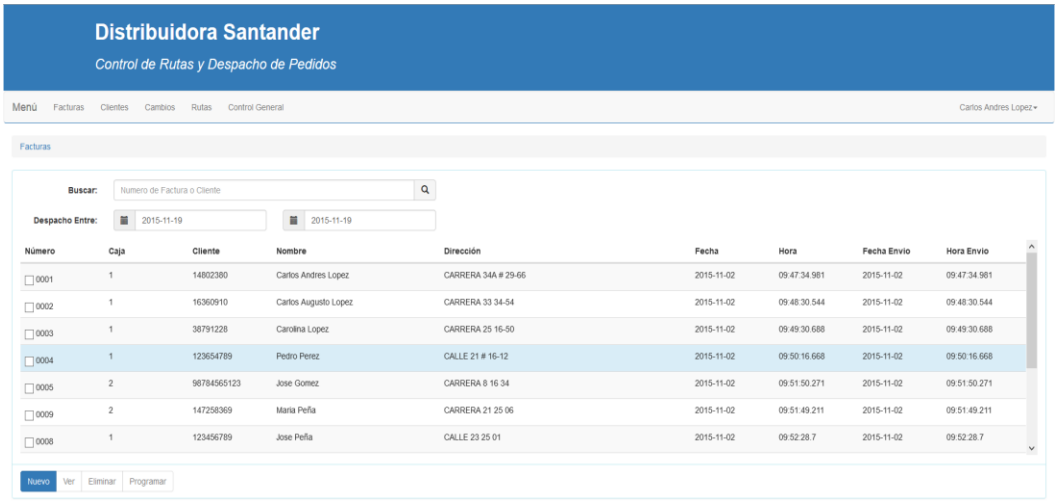


Llene el formulario y de clic en el botón “Guardar”, las direcciones y los cambios en estas se registraran de forma inmediata.

7. FACTURAS.

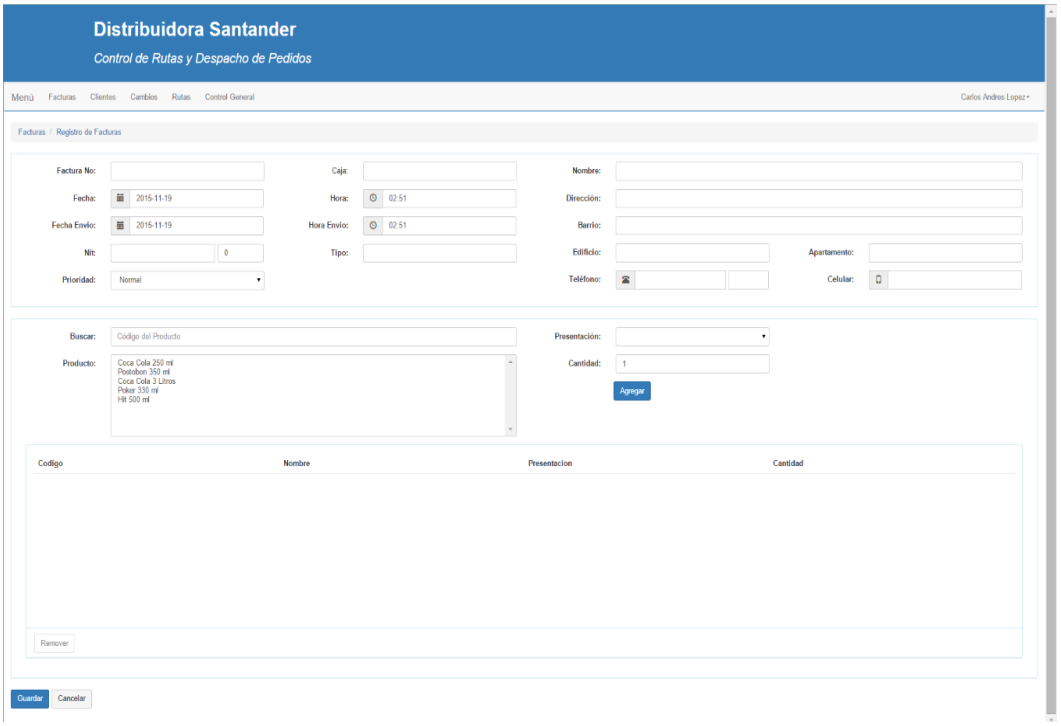
7.1. BUSCAR FACTURAS.

Para buscar una factura en el campo de búsqueda escriba el número de factura o la identificación del cliente y seleccione el rango de fechas en que fue programada para el envió.

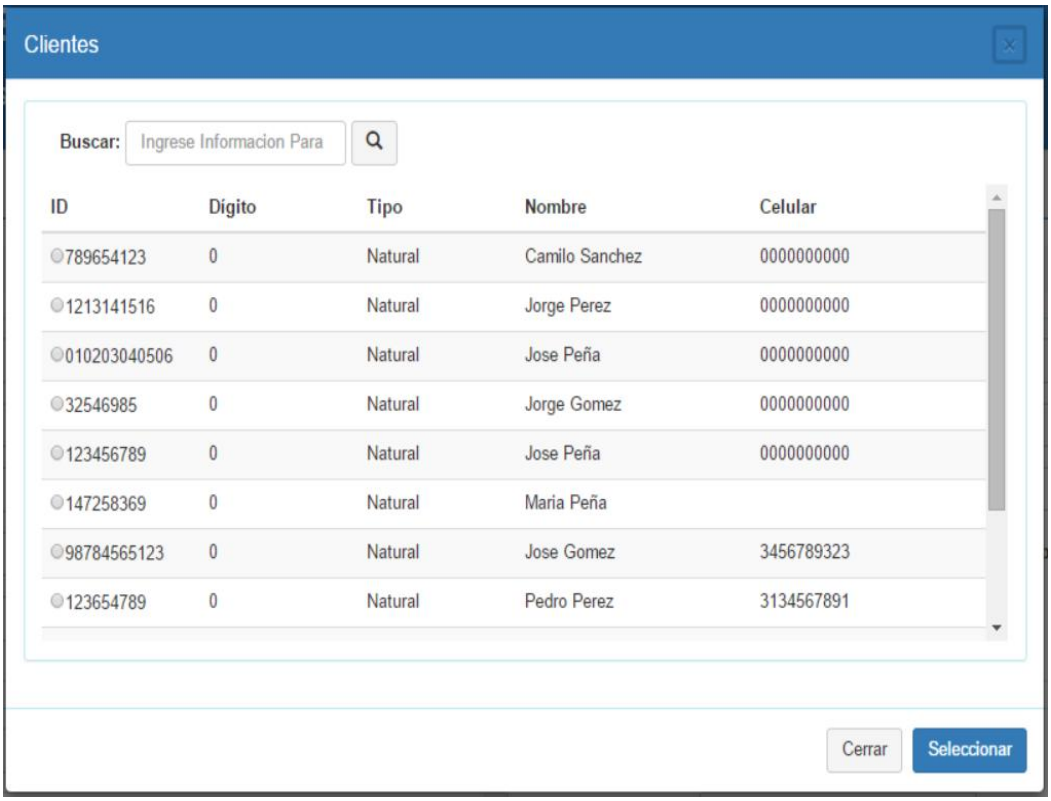


7.2. REGISTRO DE FACTURAS.

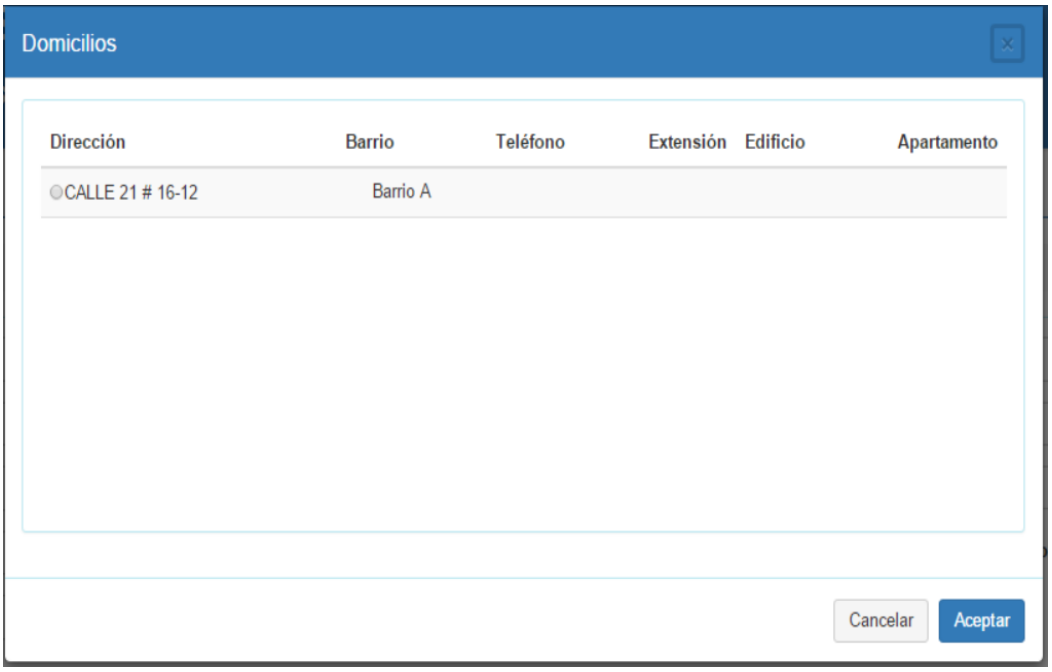
En la pantalla de búsqueda de clic en el botón “Nuevo”, se desplegara el formulario de registro de facturas.



Ingrese el número de factura y caja, seleccione las fechas y hora de registro y envío. Seleccione la casilla “Nit”, y se desplegara una lista con los clientes como se muestra a continuación.



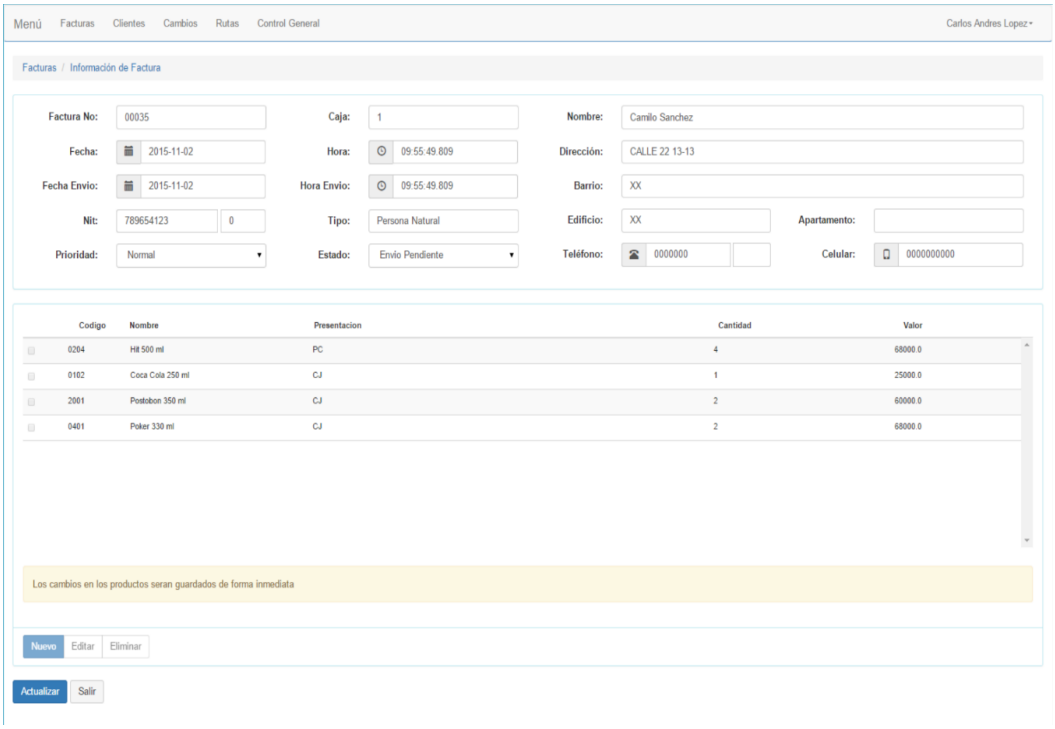
En el campo búsqueda ingrese el número de documento, nombre del cliente o celular para buscar el requerido. Seleccione el cliente y de clic en el botón “Seleccionar” y se desplegara la lista de domicilios del cliente.



Seleccione el domicilio y de clic en aceptar. Agregue los productos de la factura y guarde para finalizar.

7.3. EDITAR UNA FACTURA.

Para editar una factura, en la pantalla de búsqueda seleccione la factura a modificar y de clic en el botón “Ver”. A continuación se desplegara la información en un formulario.



De clic en el botón “Actualizar” para habilitar la edición. Si desea cambiar el cliente, al igual que en el formulario de creación de clic en el campo “Nit” y se desplegara el listado de clientes.

Para editar los productos de la factura, use los botones en la parte inferior del listado de productos. Los cambios en el listado guardaran automáticamente.

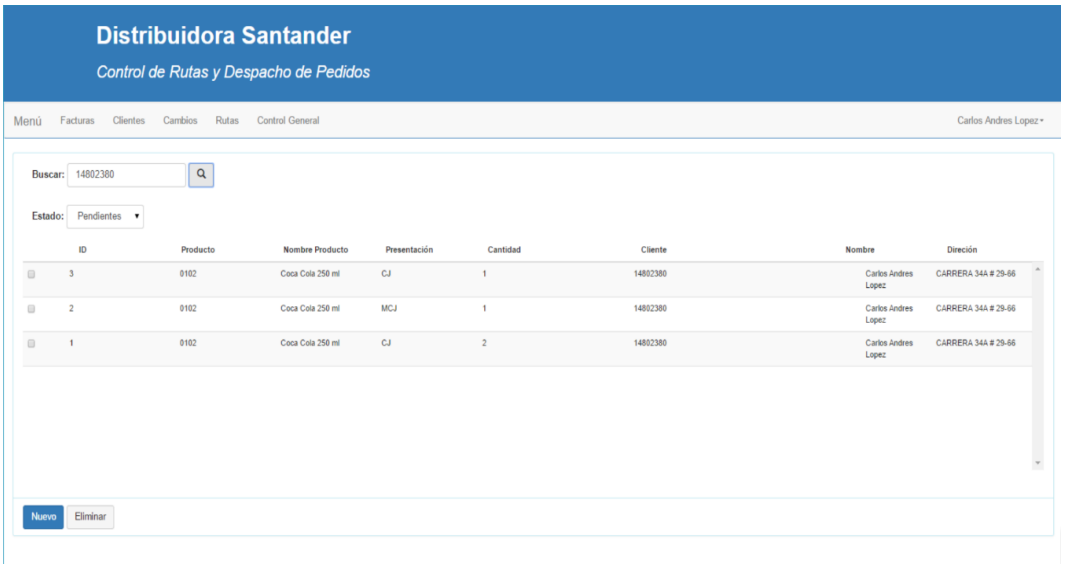
7.4. ELIMINAR UNA FACTURA.

Busque y seleccione la factura a eliminar en la pantalla de búsqueda y de clic en el botón “Eliminar”, confirme la acción para terminar.

8. CAMBIOS.

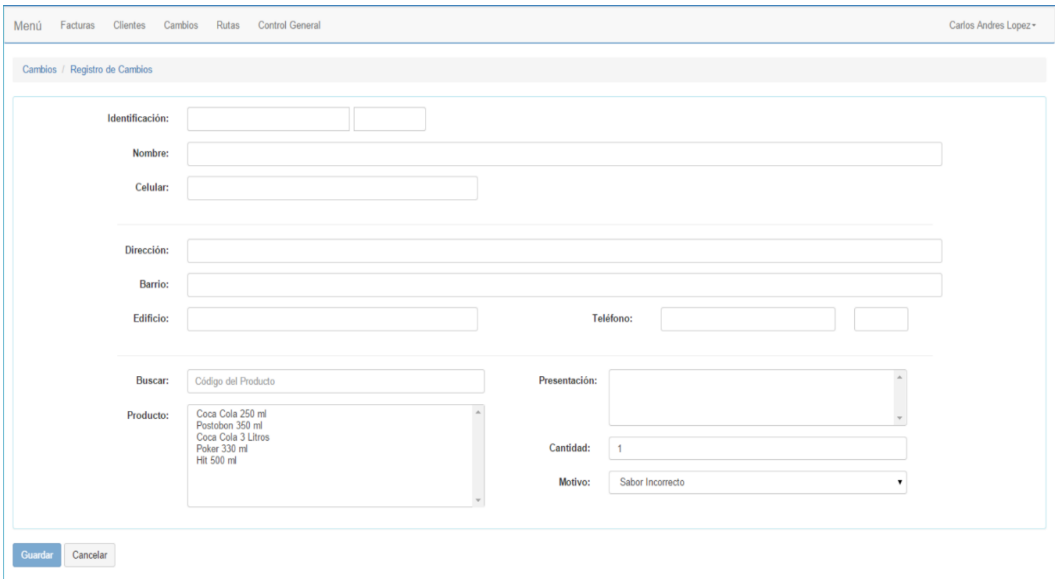
8.1. BUSCAR CAMBIOS.

En la pantalla de búsqueda de cambios, ingrese el número de identificación del cliente y el estado del cambio y de cli en el botón “Buscar” y se mostrara el listado de cambios del cliente.



8.2. REGISTRAR UN CAMBIO.

De clic en el botón “Nuevo” en la pantalla de búsqueda y a continuación se desplegará el formulario de registro de cambios.



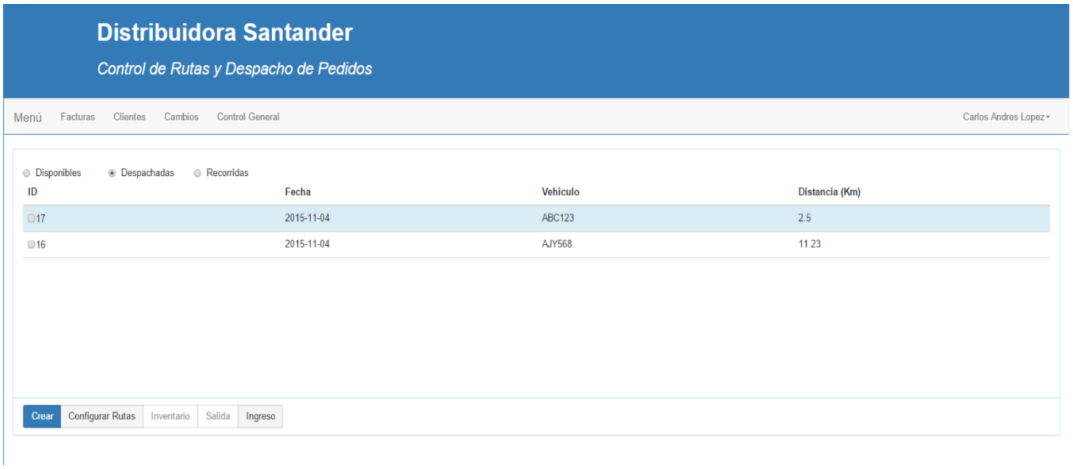
8.3. ELIMINAR UN CAMBIO.

Busque y seleccione el cambio a eliminar en la pantalla de búsqueda y de clic en el botón “Eliminar”, confirme la acción para terminar.

9. RUTAS.

9.1. BUSCAR RUTAS.

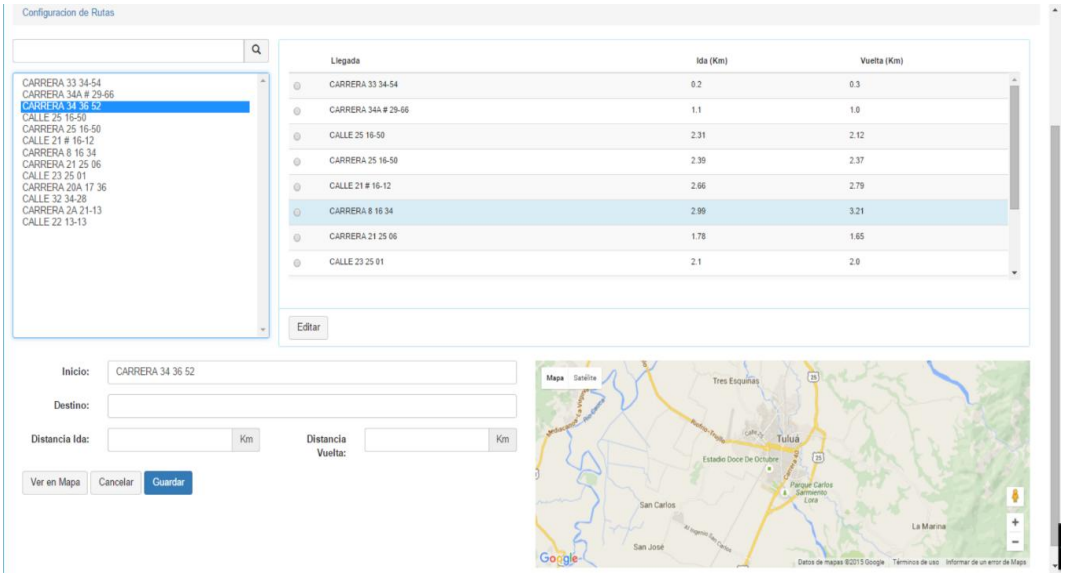
Seleccione la opción “Rutas” en el menú y a continuación se desplegara la pantalla de búsqueda de rutas.



Esta pantalla cuenta con tres opciones, Disponibles, Despachadas y Recorridas. En disponibles se muestra las rutas calculadas que están en espera, Despachadas son las rutas calculadas que se están recorriendo en el momento y Recorridas las rutas ya terminadas.

9.2. CONFIGURACIÓN DE RUTAS.

En la pantalla de búsqueda de rutas de clic en el botón “Configurar Rutas” en la parte inferior de la tabla. Se desplegara el siguiente formulario.

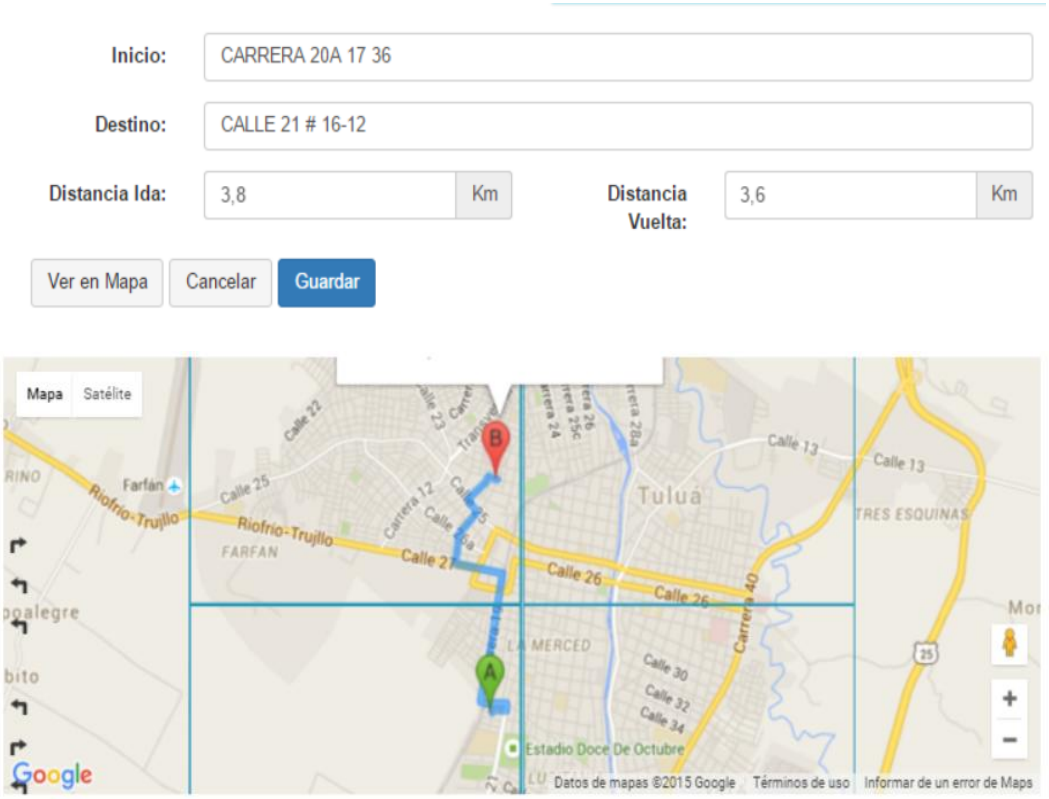


Esta pantalla muestra en el lado izquierdo un listado de las direcciones. Seleccione una de la lista y se mostrara una tabla de las direcciones relacionadas y las distancias de ida y vuelta entre la selección y cada una de las otras.

Si en la tabla de direcciones relacionadas aparece en la distancia el valor NaN, quiere decir que no se ha registrado una distancia entre este par de domicilios en el sentido que especifique la columna.

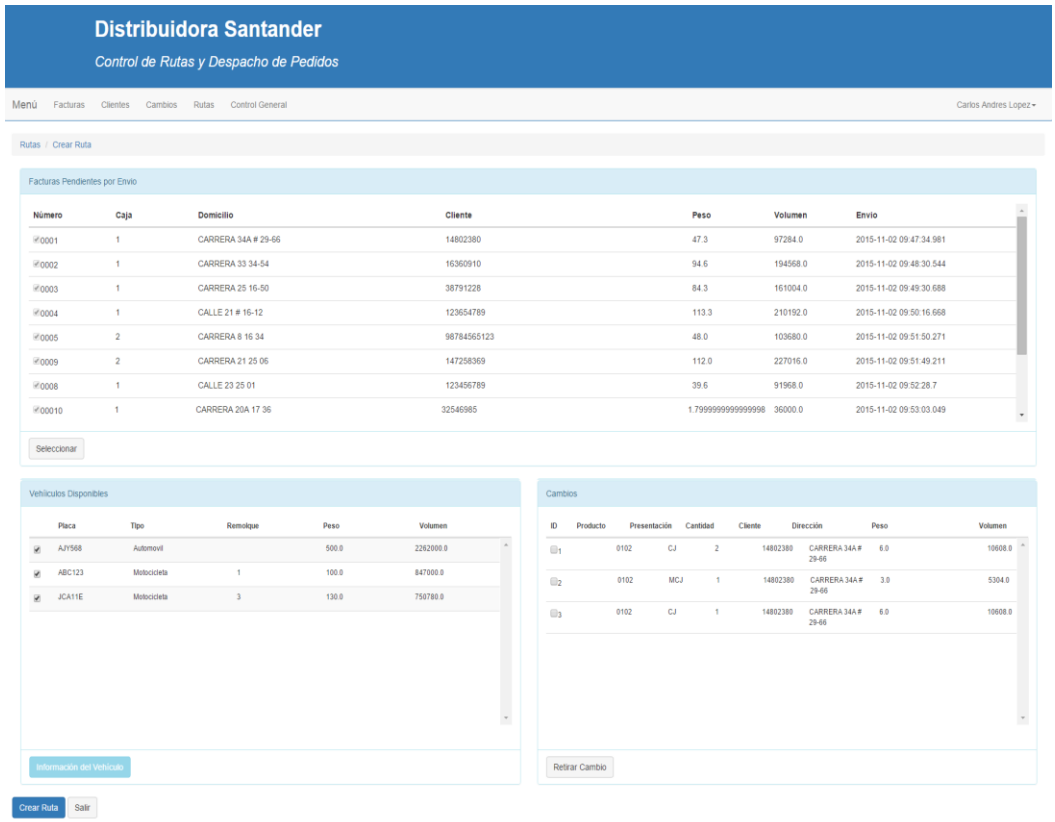
Para registrar las distancias entre el domicilio seleccionado y uno de la tabla, seleccione en la tabla de relacionados la que desea configurar, dando clic en la casilla en el lado izquierdo de la tabla y de clic en el botón “Editar” que se encuentra en la parte inferior. En el formulario ubicado en la parte inferior se verá los dos domicilios seleccionados y los valores de las distancias en caso de estar configuradas.

Si desea visualizar estas dos direcciones en el mapa de clic en el botón “Ver en Mapa”.



9.3. CREAR RUTAS.

Para crear una ruta, en el listado de rutas de clic en el botón “Crear” en la parte inferior de la tabla.



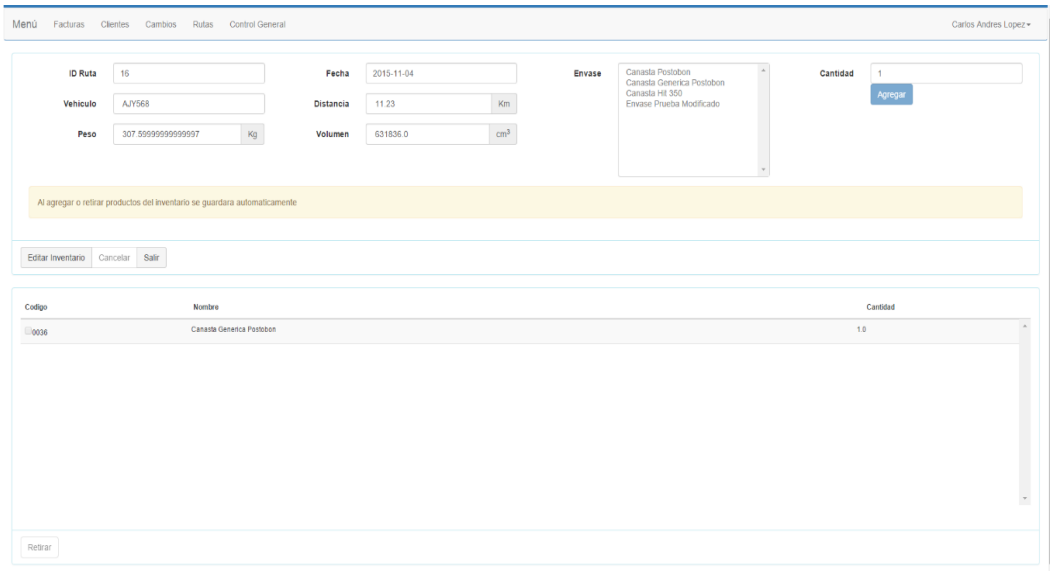
Esta pantalla muestra tres tablas, la primera en la parte superior es la lista de facturas por enviar, la segunda es la lista de vehículos y la tercera la lista de cambios.

Por defecto las facturas están seleccionadas, en caso que se quiera omitir una o varias facturas del caculo de ruta, en el listado de facturas de clic en el botón “Seleccionar” ubicado en la parte inferior de la lista de facturas, esta opción habilita las casillas y quita la selección de todas las facturas, las facturas sin marcar la casilla no entraran en el cálculo de ruta.

Seleccione los vehículos a incluir en el cálculo de ruta, y los cambios a enviar y de clic en el botón “Crear Ruta”

9.4. CREAR INVENTARIO DE RUTA.

Para crear un inventario, seleccione una ruta de la lista y de clic en el botón “Inventario”, se desplegara el siguiente formulario.

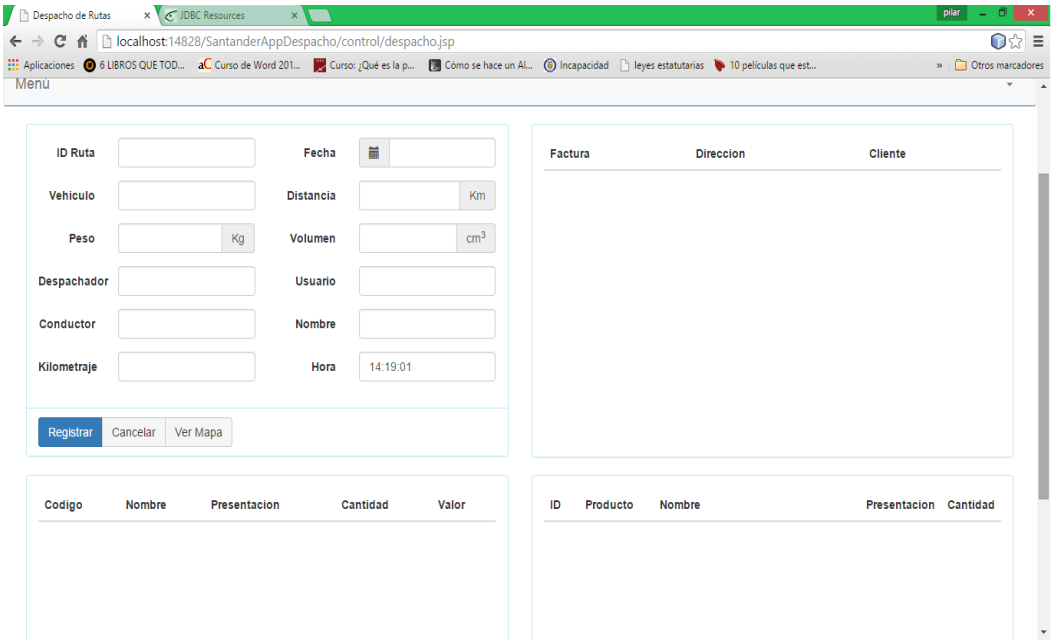


De clic en el botón “Editar Inventario” para habilitar la edición del inventario.

En el formulario en la parte superior, se encuentra una lista de los envases existentes, seleccione un envase de la lista e ingrese la cantidad y de clic en el botón “Agregar” para incluirlo en el inventario. Los campos al agregar y eliminar envases de la lista son guardados de forma automática.

9.5. DESPACHO.

Para el despacho, seleccione una ruta pendiente de la lista y de clic en la opción **Salida** y se desplegara el siguiente formulario.

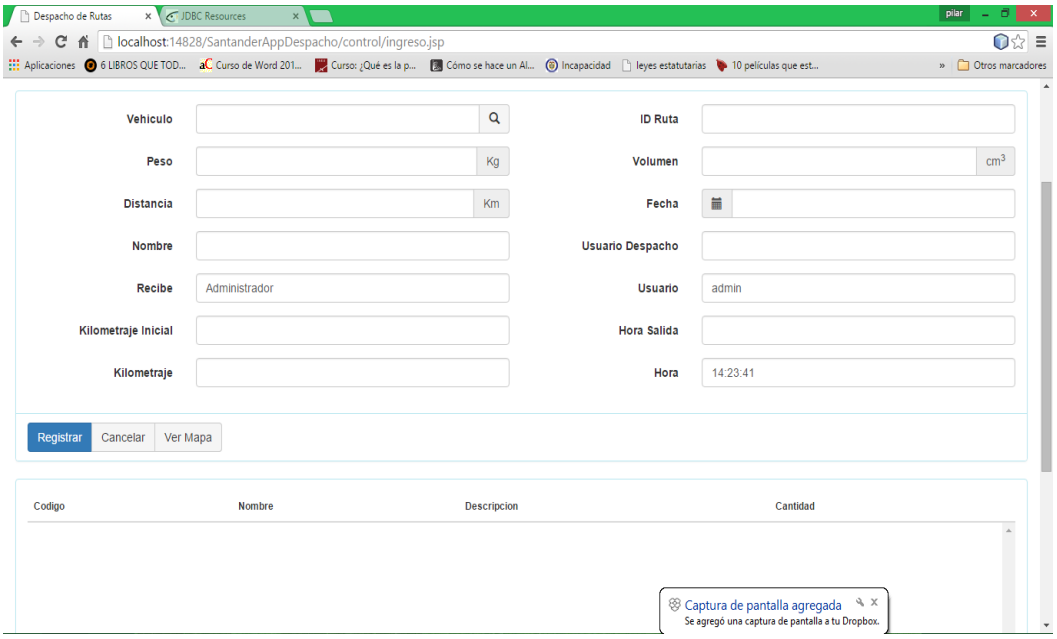


En este formulario se cargara la información de la ruta, vehículo, conductor, usuario que despacha, facturas y cambios. Ingrese el kilometraje marcado por el vehículo y de clic en el botón registrar para dar la salida.

Adicionalmente si se desea ver los puntos de la ruta de clic en Ver Mapa para visualizar los recorridos entre cada par de puntos.

9.6. INGRESO.

De clic en el botón ingreso en el listado de rutas y se desplegara el siguiente formulario.



En este formulario ingrese el número de placa para cargar la información de la ruta. Se cargaran también los datos del inventario.

Para dar ingreso escriba el kilometraje marcado por el vehículo al momento de llegar y de clic en el botón registrar.

MANUAL DE INSTALACIÓN

CONTROL DE RUTAS Y DESPACHO DE PEDIDOS

Contenido

1. Instalación de la base de datos PostgreSQL 9.3

Abra el instalador de PostgreSQL incluido en el CD adjunto, aparece la siguiente ventana:

Imagen 1: Pantalla de instalación 1 PostgreSQL

De clic en el botón **Siguiente** para comenzar la instalación.

Seleccione los directorios de instalación y almacenamiento de datos, imagen 2 y 3.

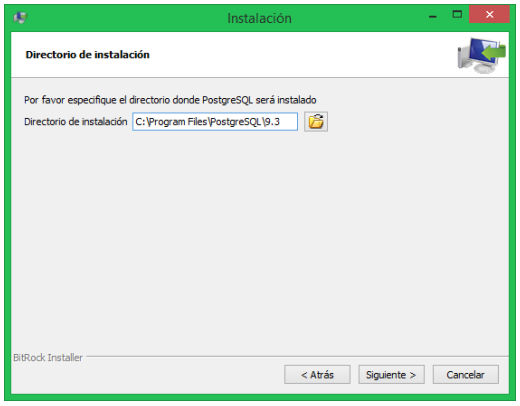


Imagen 2: directorio de instalación.

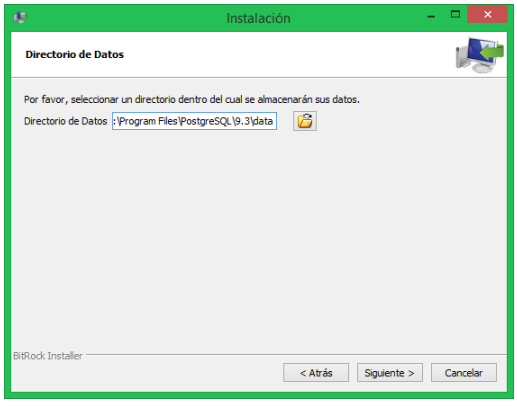


Imagen 3: directorio de bases de datos.

Después de seleccionar continúe e ingrese la contraseña de la base de datos, Imagen 4. Seleccione el puerto y el idioma Imágenes 5 y 6.

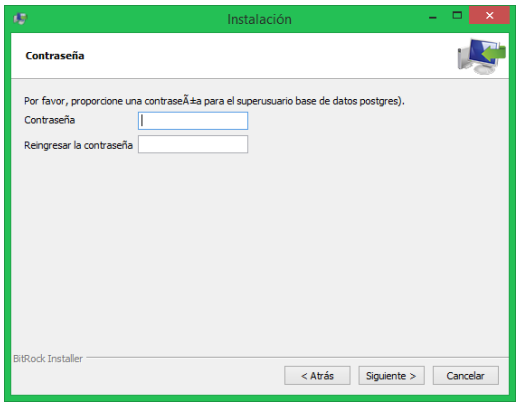


Imagen 4: Creación de contraseña

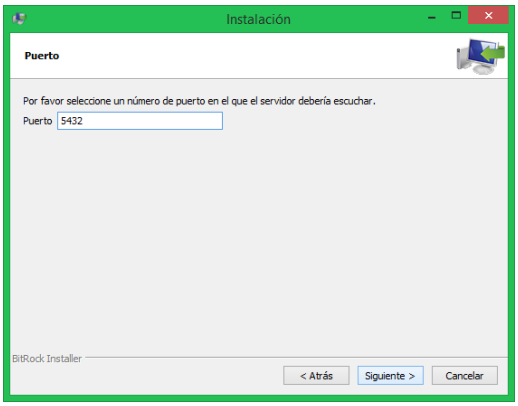


Imagen 5: Selección de Puerto.

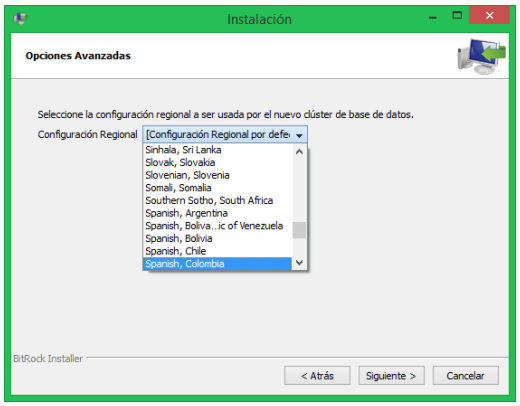


Imagen 6: Selección del idioma.

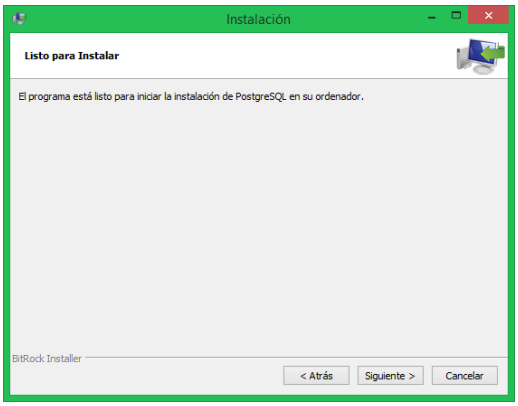


Imagen 7: Confirmar la instalación

Para finalizar la instalación aparece la pantalla Imagen 7, la cual tiene una casilla seleccionada, quite la selección y presione el botón **Terminar**.

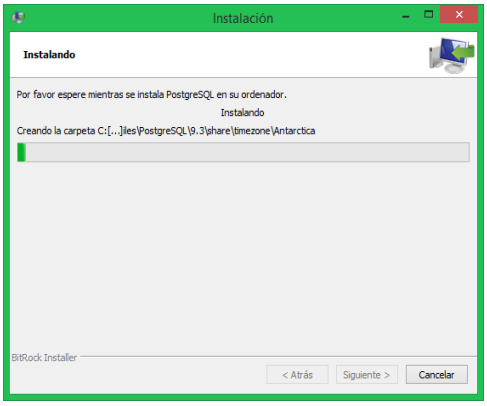


Imagen 8: Progreso de la instalación

Imagen 9: Fin de la instalación

Para más información mire la documentación de PostgreSQL, en Ingles.

http://www.postgresql.org/docs/

2. Creación de la base de datos.

Después de terminar la instalación ingrese al PgAdmin III, en el árbol de navegación expanda el nodo PostgreSQL 9.3 y seleccione el nodo Databases.

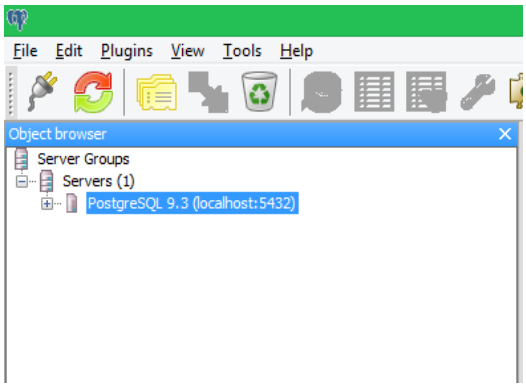


Imagen 10: base de datos 1

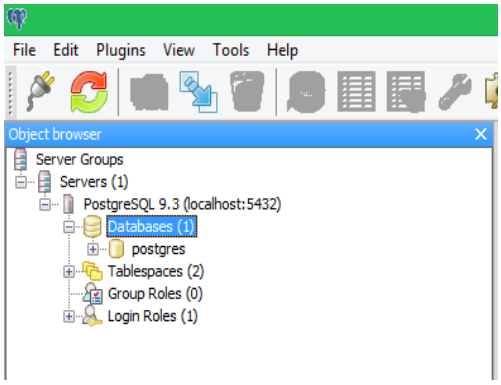
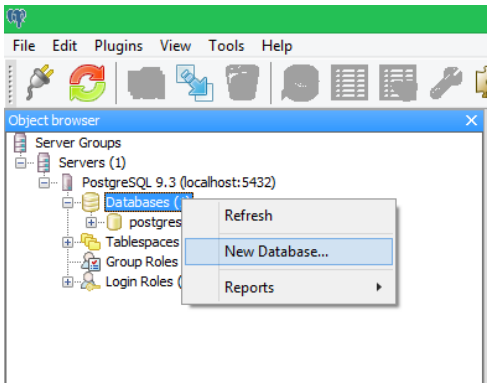
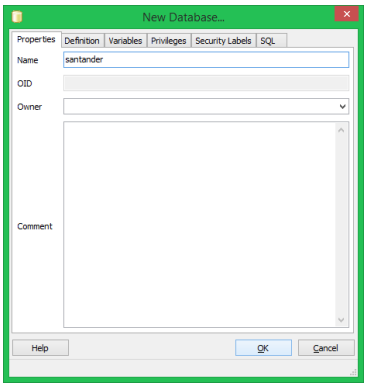


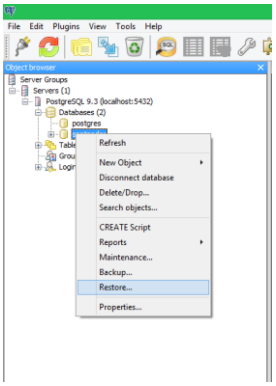
Imagen 11: Base de datos 2

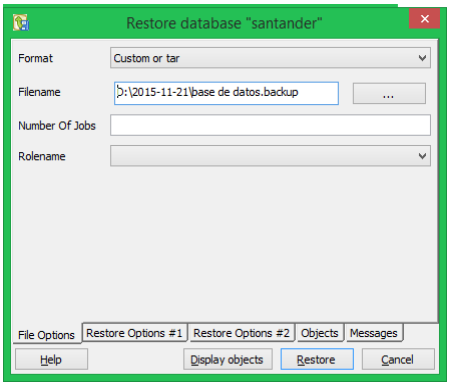
De clic derecho y seleccione la opción **New Database…**, se abrira el formulario de creación de base de datos. Escriba el nombre de la base de datos y de clic en **Ok**.





En el nodo de la nueva base de datos creada de clic derecho y seleccione la opción Restore.

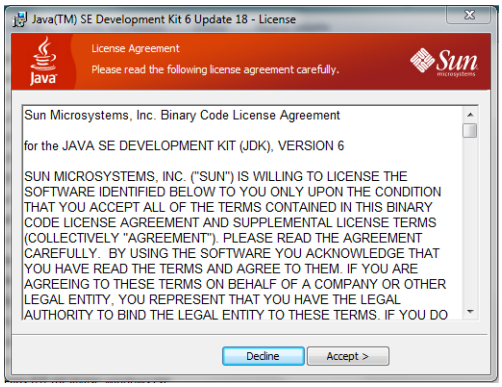


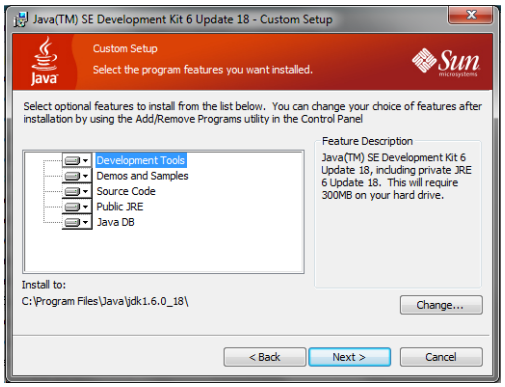


Seleccione el archivo Base de Datos.backup del CD de anexos y de clic en el botón **Restore**. Espere que este activo el botón **Done** y de clic en este para terminar.

3. Instalación JDK 1.7

Ejecute el archivo jdk-7u79-windows-i586.exe incluido en el CD adjunto, aparecerá la pantalla de licencia.





De clic en aceptar para continuar, en la siguiente pantalla seleccione el directorio de donde se guardaran los archivos de java y de clic en el botón **Next>** para iniciar la instalación. Si se selecciona un directorio diferente al por defecto se deberán hacer configuraciones adicionales para que las aplicaciones Java puedan ejecutarse.

Para más información mire la documentación de Java, en Ingles.

http://www.oracle.com/technetwork/es/java/javase/documentation/index.html

4. Instalación de Glassfish Server 4.1

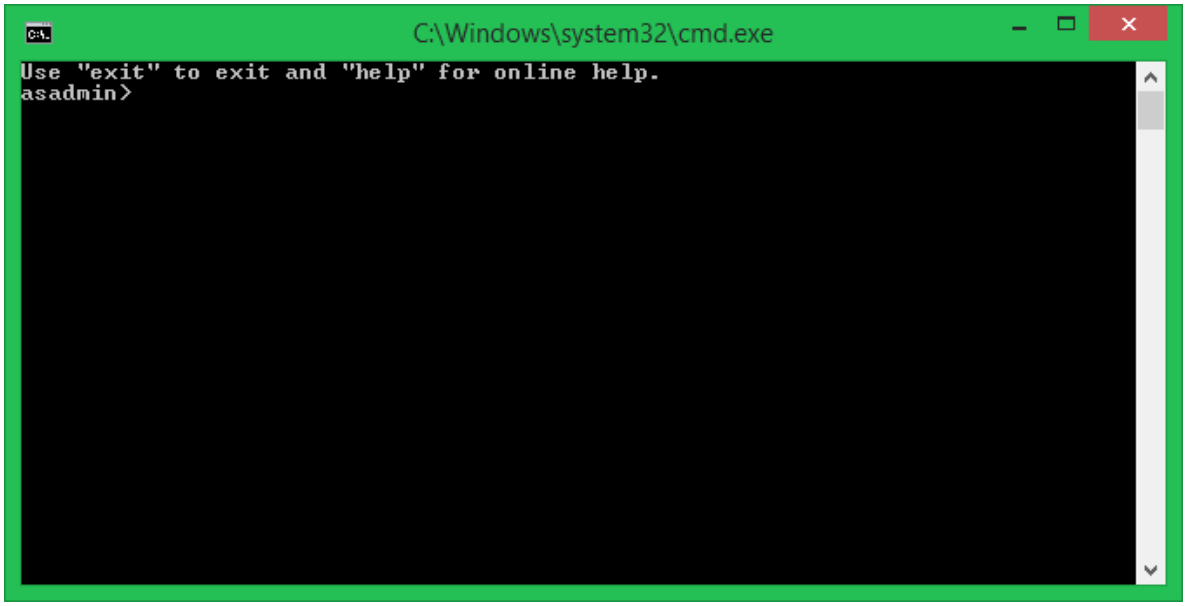
Descomprima el archivo glassfish-4.1.zip incluido en el CD adjunto en el directorio en que desee instalarlo.

Esta aplicación no necesita configuraciones adicionales en su instalación, para más información mire la documentación, en inglés.

https://glassfish.java.net/docs/4.0/installation-guide.pdf

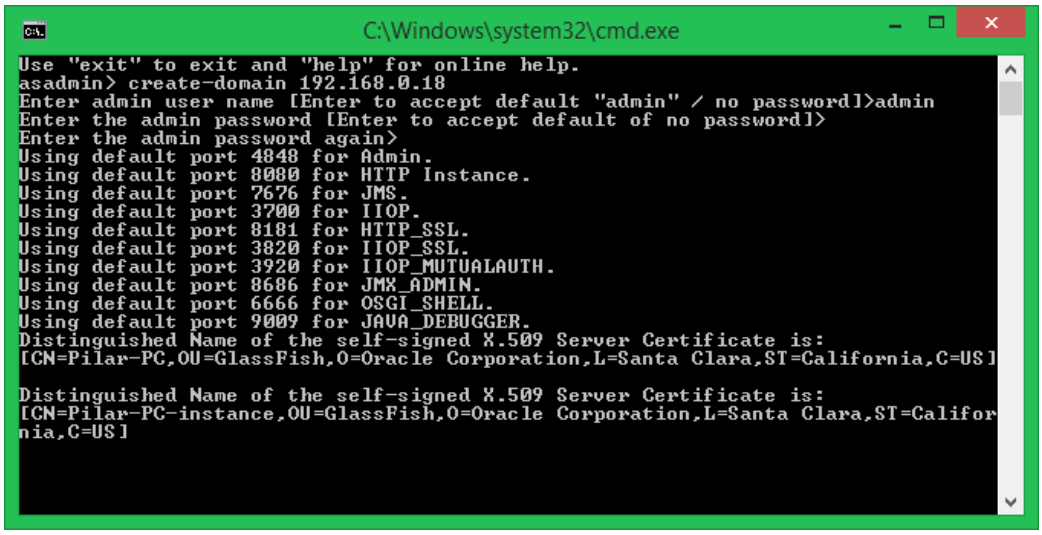
5. Configuración de Glassfish.

Para ejecutar la aplicación es necesario crear un dominio, para esto ejecute el archivo asadmin.bat. Para esto valla al directorio de instalacion de glassfish /glassfish4/bin/asadmin.bat y abra el archivo, se mostrara una consola de comandos como la siguiente.



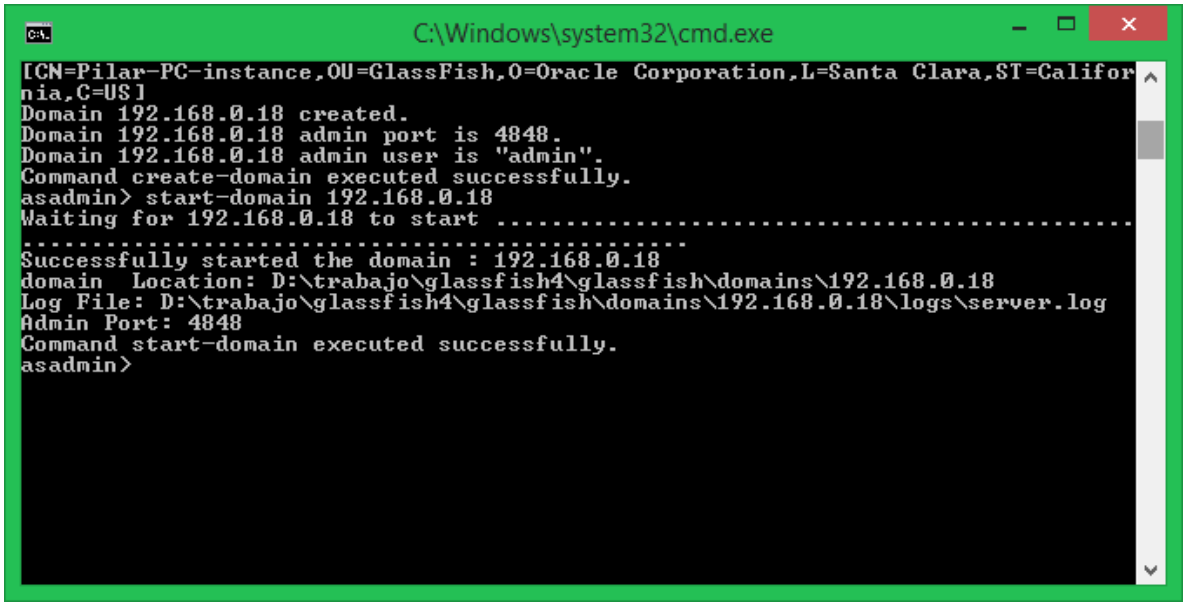
En la consola escriba el comando create-domain miDominio y presione Enter para continuar.

Solicitará crear un usuario y contraseña para el dominio, si se deja en blanco no habrá problema, más adelante podrá crearlos en la consola de administración.



Después de ingresar la información solicitada aparecerá en la consola la información de los puertos disponibles para el dominio creado.

Inicie el dominio escribiendo el comando start-domain miDominio

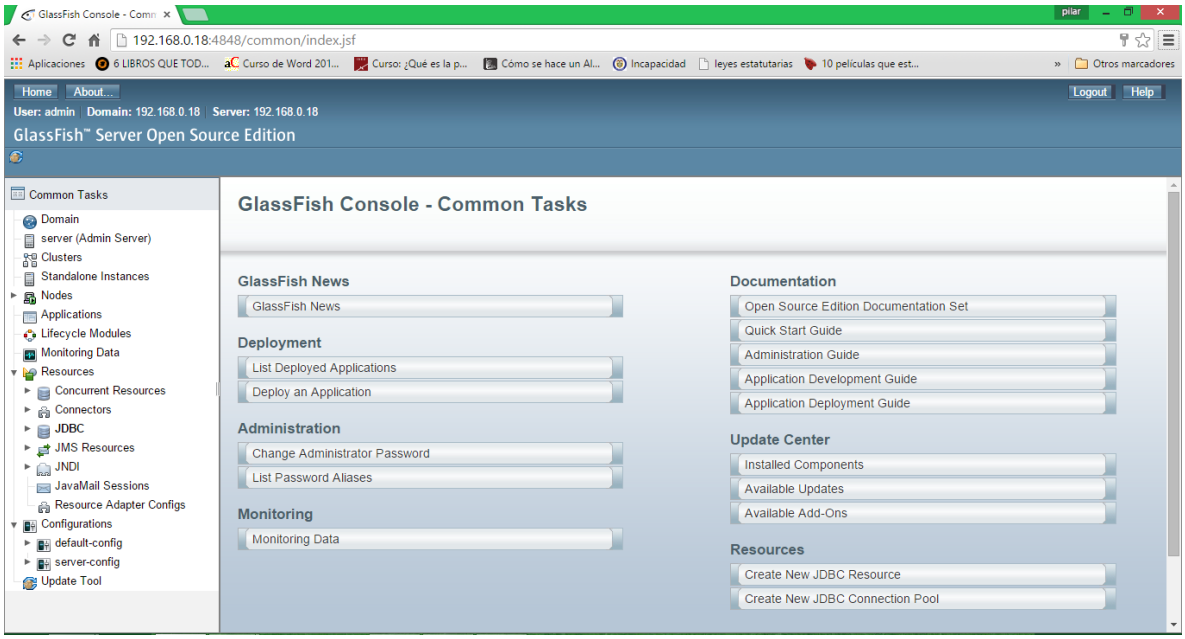


Al terminar de ejecutar el comando aparecerá el mensaje informando que el dominio ha sido iniciado.

A continuación abra el navegador e ingrese al dominio que ha creado, por ejemplo localhost:4848, 4848 es el puerto de escucha del administrador del Glassfish server para este ejemplo.

Aparecerá la siguiente pantalla.

Ingrese a la consola de administración con el usuario y contraseña anteriormente creado. Al ingresar encontrara la siguiente pantalla.

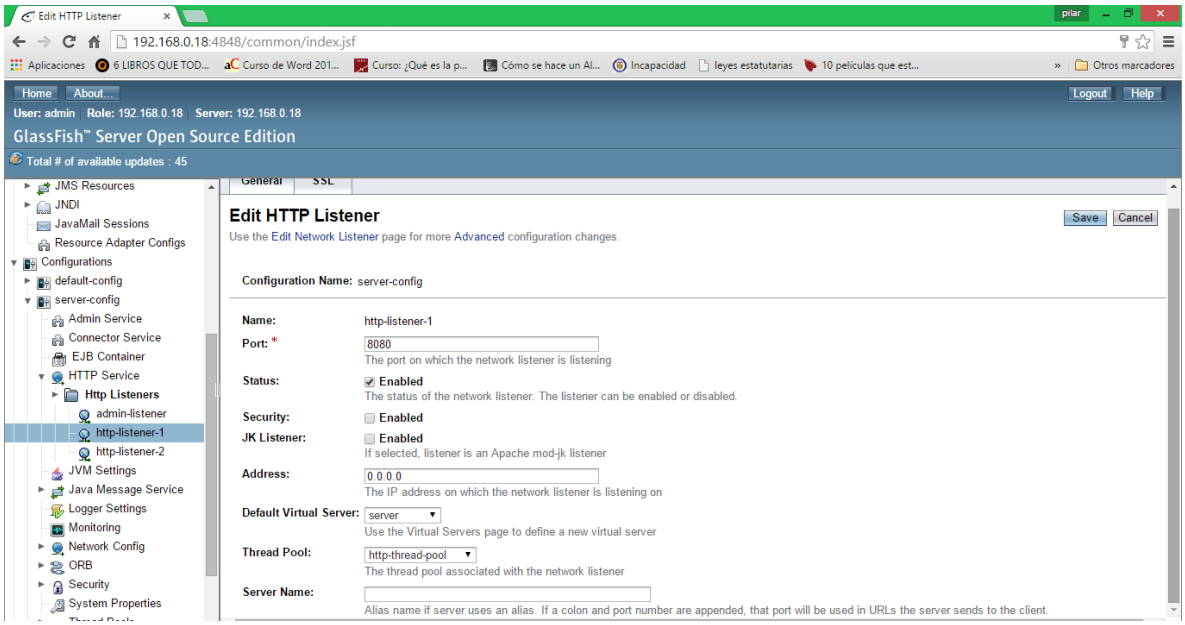


En el menú valla a Configurations / server-config / HTTP Service / HTTP Listeners

Allí encontrara una lista de listeners. Cree o modifique uno de los listeners por defecto, http-listener-1 y http-listener-2.

El http-listener-1 por defecto escucha por el puerto 8080, el http-listener-2 por el 8181, la diferencia es que el http-listener-2 es para conexiones SSL.

Para este ejemplo se modificara el http-listener-1. Se selecciona dando clic sobre el nombre.



En el formulario del listener escriba la dirección ip del servidor en el campo Address, el puerto por defecto es el 8080, si desea configurar las conexiones por otro puesto escriba en el campo Port el puerto por el cual el listener deberá recibir conexiones.

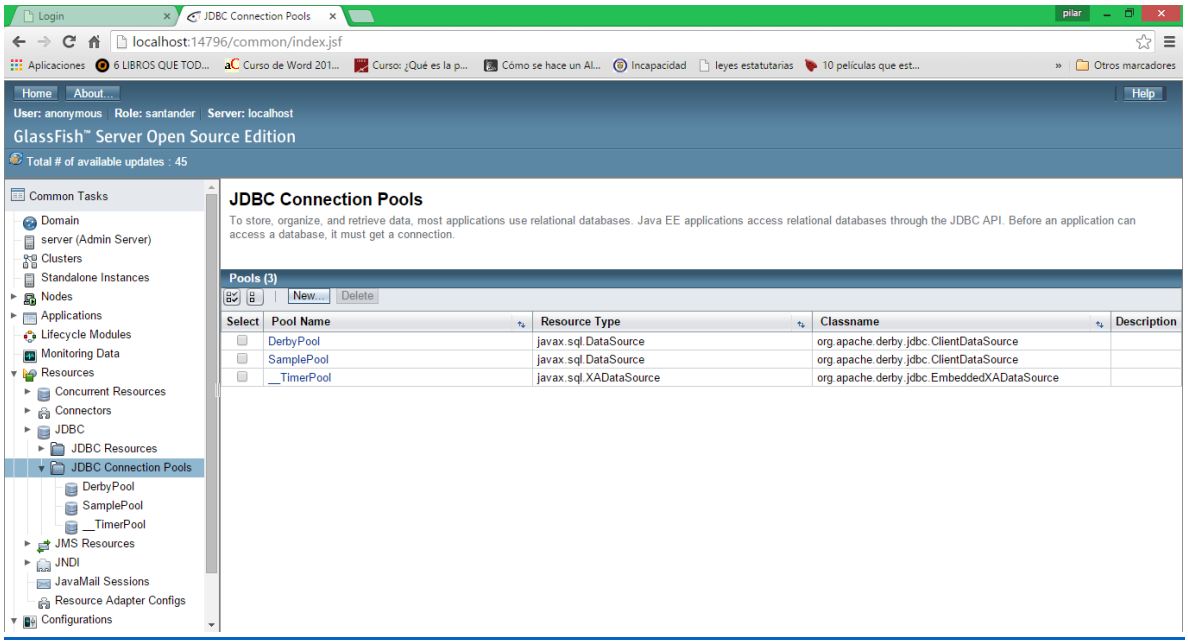
Glassfish maneja servidores virtuales, por defecto maneja uno llamado server, en caso de tener una aplicación funcionando sobre este es necesario crear un nuevo servidor virtual.

Para más información de configuraciones consulta la documentación de Glassfish Server, en Ingles.

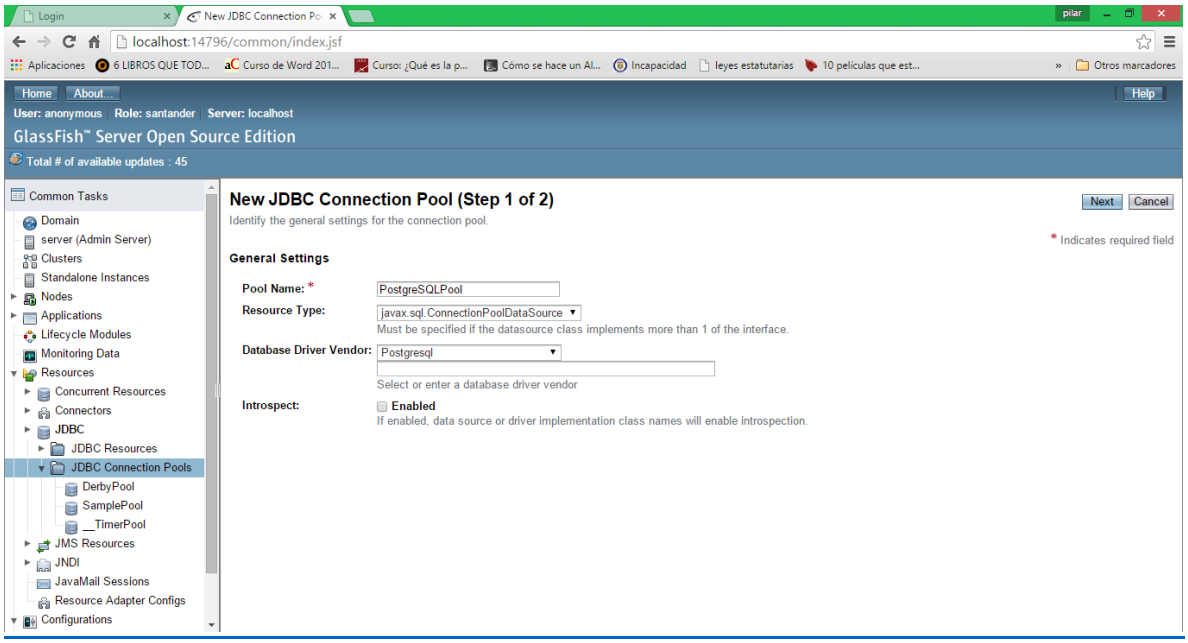
https://glassfish.java.net/documentation.html

Copie el archivo postgresql-9.3-1102.jdbc3 en la carpeta de glassfish glassfish4/glassfish/lib y reinicie el servidor.

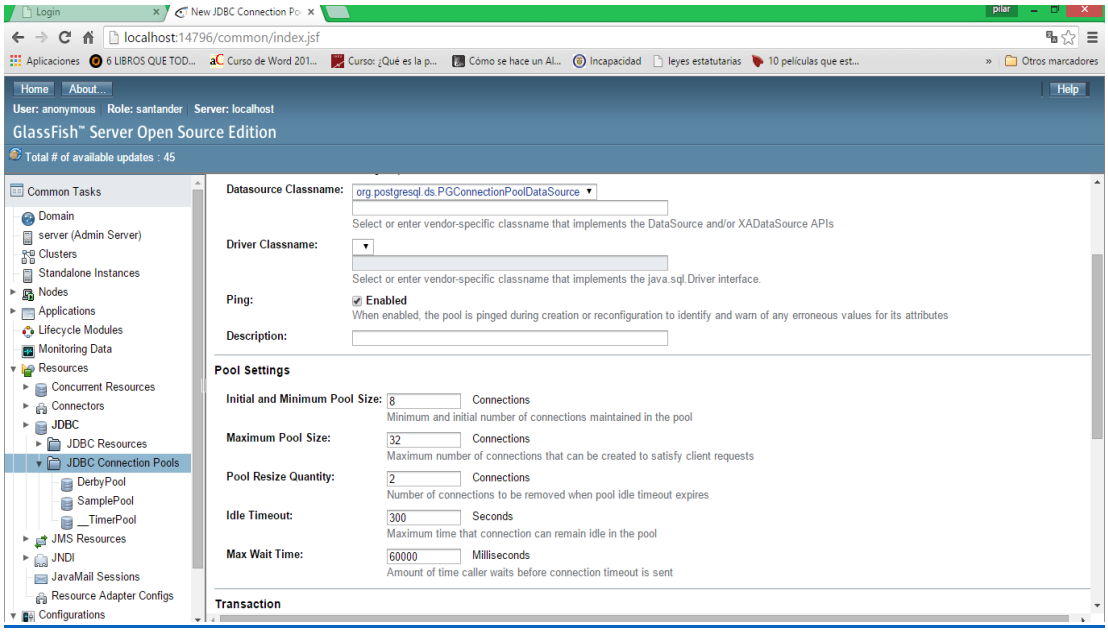
Después de reiniciar ingrese nuevamente a la consola de administración de Glassfish y diríjase en el menú a la opción JDBC Connection Pools, en la pantalla que aparece seleccione la opción **New** para agregar un nuevo pool.



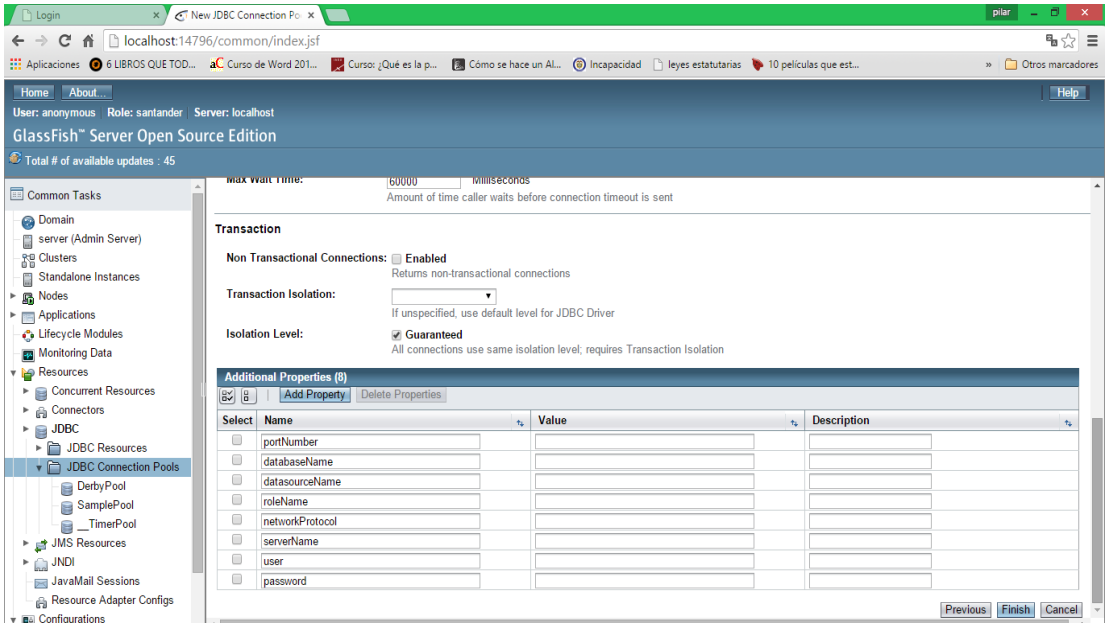
Se desplegara el formulario de creación de pools.



En el campo Pool Name, escriba el nombre del pool a crear, en el campo Resource Type, seleccione javax.sqlConnectionPoolDataSource y en el campo Database Driver Vendor seleccione la opción Postgresql. De clic en Next para continuar la configuración.



En el campo Ping habilite la casilla Enable, y diríjase a la parte inferior del formulario, allí encontrara la siguiente tabla.

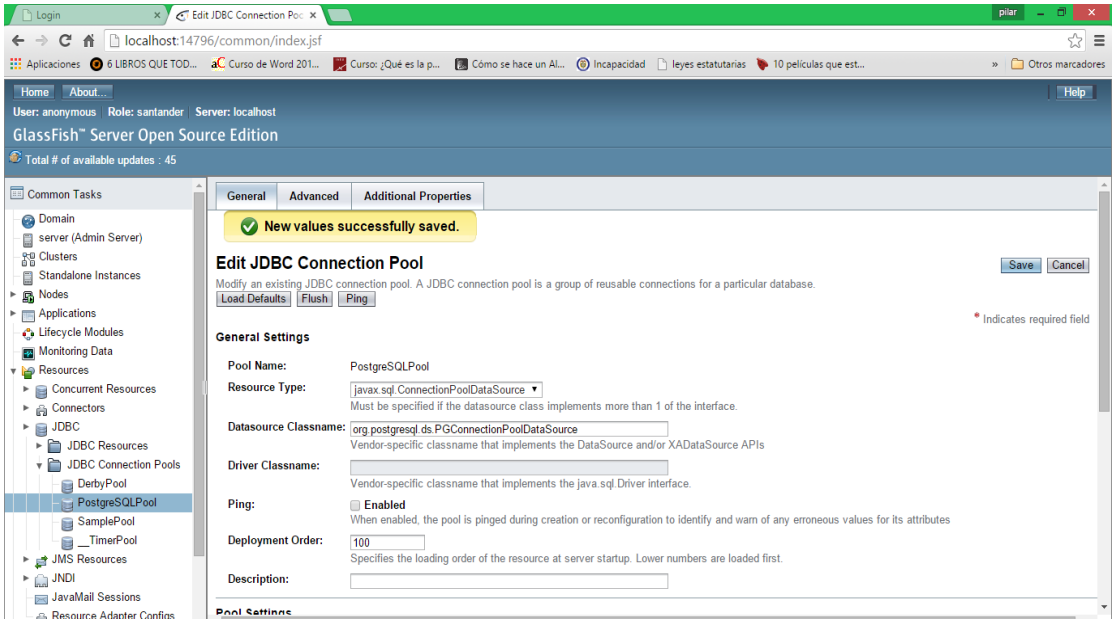


En el campo PortNumber escriba el puerto de conexión a la base de datos, en postgre por defecto es el 5432. En databaseName escriba el nombre que haya puesto a la base de datos.

En serverName escriba la dirección del servidor de la base de datos, si se encuentra instalada en el mismo equipo, seria localhost.

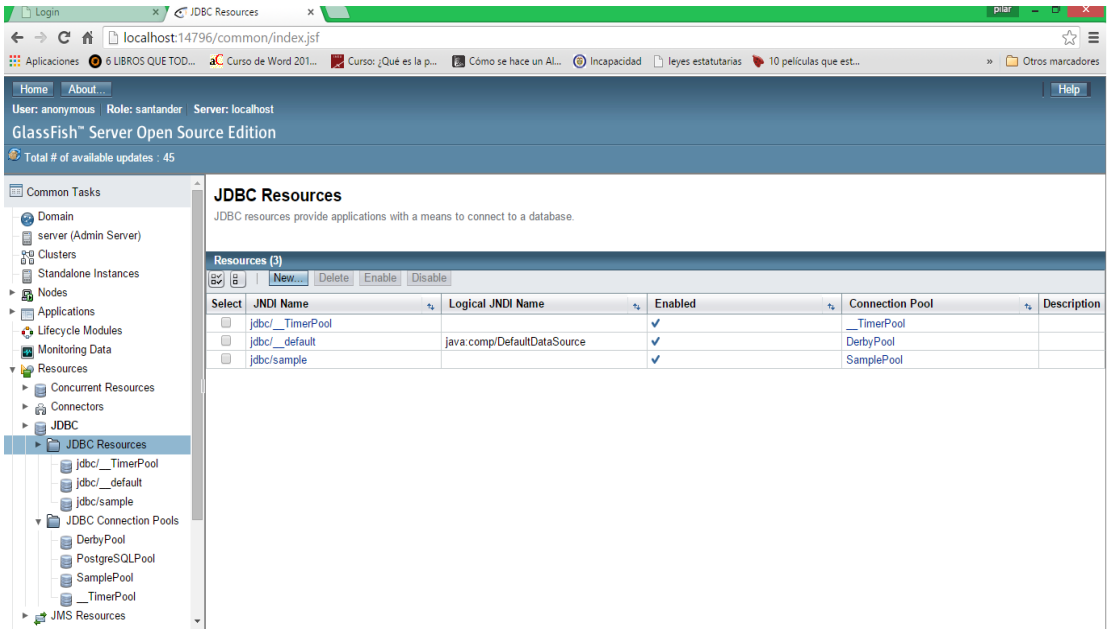
En los campos User y Password escriba el usuario y contraseña para conectarse a la base de datos.

Después de llenar este campo de cli en el botón Finish, si la conexión se hizo correctamente y abilito el campo Ping aparecerá un mensaje.

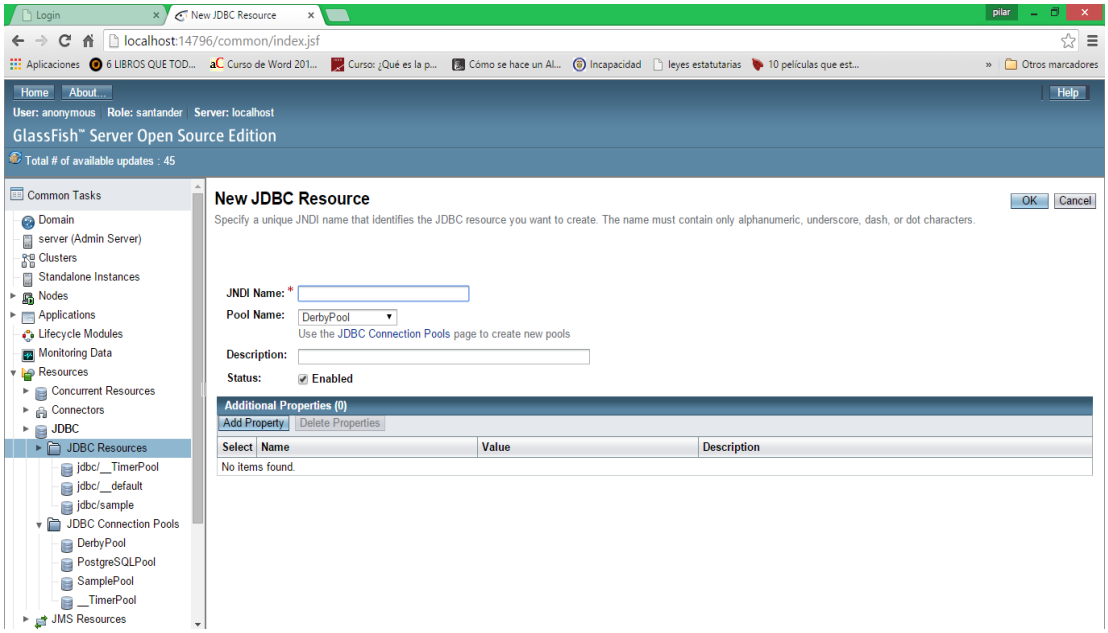


De clic en el botón Ping para verificar la conexión.

A continuación cree el data source, para esto en la opción JDBC del menú ingrese en JDBC Resources.

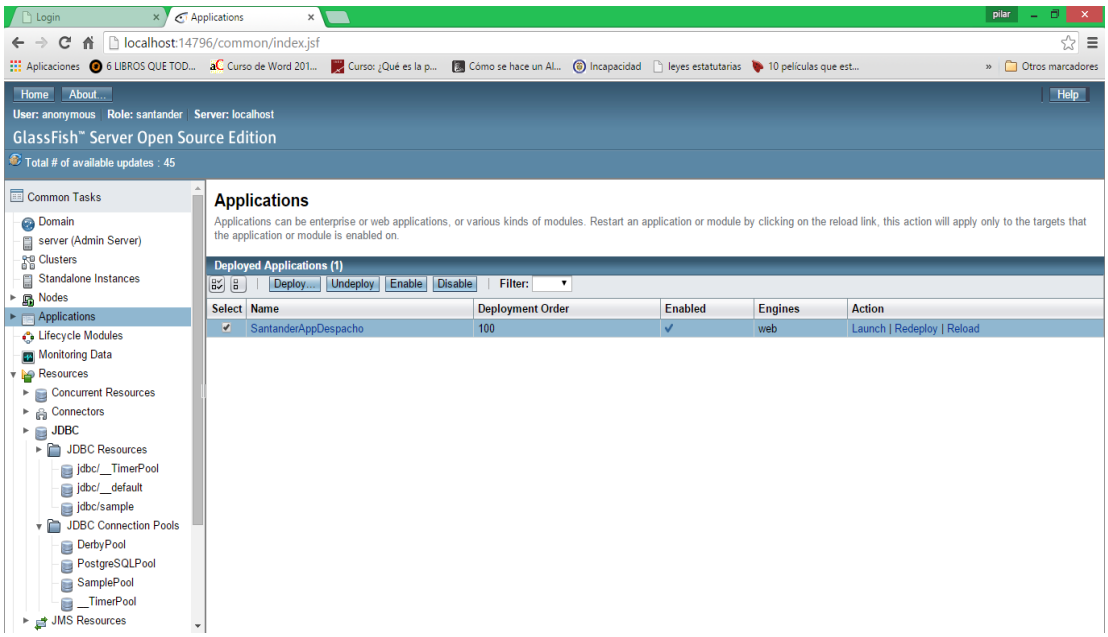


De clic en el botón New para crear un nuevo recurso.

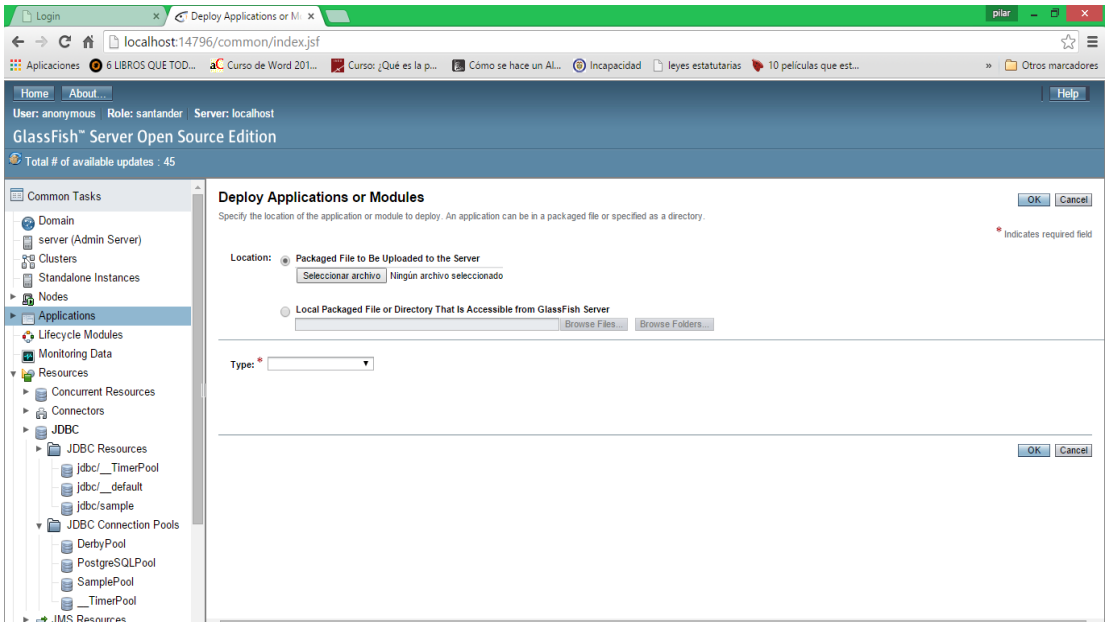


En el campo JNDI Name escriba el siguiente nombre PostgreDataSource y en el campo Pool Name seleccione el Pool de creado anteriormente y de clic en Ok.

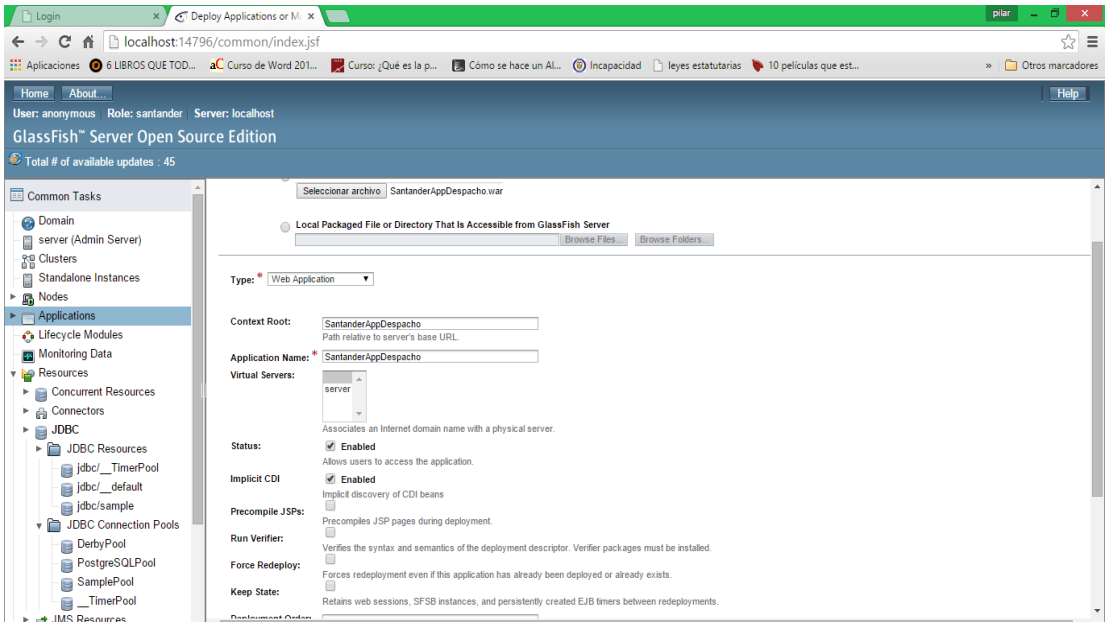
Después de esto diríjase a la opción Applications del menú.



En esta pantalla seleccione la opción Deploy para cargar el archivo .war de la aplicación.



De clic en el botón Seleccionar archivo y seleccione del CD adjunto el archivo SantaderAppDespacho.war, se cargaran por defecto las opciones de configuración para el archivo.



De clic en Ok para terminar.

Ingrese a su dominio escribiendo en la barra de direcciones del navegador miDminio:8080/SantanderAppDespacho.

COSTOS DEL PROYECTO

Servicios			
Motivo	Tiempo(Meses)	Valor Mes (Pesos)	Total (Pesos)
Internet	18	$ 70.000,00	$ 1.260.000,00
Servidor de Pruebas	3	$ 70.000,00	$ 210.000,00
Plan de Datos	6	$ 30.000,00	$ 180.000,00
Energía Eléctrica	18	$ 70.000,00	$ 1.260.000,00
Agua	18	$ 50.000,00	$ 900.000,00
Gas	18	$ 12.000,00	$ 216.000,00
Arriendo	18	$ 350.000,00	$ 6.300.000,00
		Total	$ 10.326.000,00

Transporte				
Motivo	Veces por mes	Meses	Valor ida y Vuelta	Total (Pesos)
Tuluá	2	18	$ 4.000,00	$ 144.000,00
Tuluá-Palmira	2	7	$18.000,00	$ 252.000,00
			Total	$ 396.000,00

Equipos de Computo			
Motivo	Cantidad		Total (Pesos)
Computador de escritorio.	1		$ 7.500.000,00
Smartphone	1		$ 436.500,00
		Total	$ 7.936.500,00

Total		
Motivo	Total (Pesos)	
Servicios	$ 10.326.000,00	
Transporte	$ 396.000,00	
Equipos de Computo	$ 7.936.500,00	
	Total	$ 18.658.500,00