Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Curso: IC-2101 Programación Orientada a Objetos

Profesora

Ericka Solano Fernández

2do y 3er Proyecto:

***“Pyme”***

**Estudiantes:**

José David Madrigal Chaves

Carné: 2016110519

Paúl Esteban Gutiérrez Salas

Carné: 2016183680

**Fecha de Entrega:**

Viernes 25 de noviembre, 2016

Semestre Il, 2016

**Índice:**

**Introducción……………………………………………..3**

**Especificación de II y III proyecto……………………4**

**Estrategia de Solución………………………………...15**

**Cronograma……………………………………………..16**

**Diagrama de clases en UML………………………….18**

**Justificación de los patrones utilizados…………...19**

**Análisis de resultados…………………………………20**

**Enlace al JavaDoc generado………………………….21**

**Aspectos y Lecciones aprendidas…………………..22**

**Bitácora del trabajo…………………………………….23**

**Bibliografía y fuentes digitales utilizadas………….24**

**Anexos…………………………………………………...25**

**Introducción:**

En esta tarea programada, se realizará una serie de funciones implementadas por medio del uso de clases, y funciones implementadas orientadas a objetos, con la importancia de que los estudiantes logren practicar y mejorar habilidades para la resolución de problemas, y al trabajar en equipo se conseguirá una mejora en la toma de decisiones y un incremento en cuanto al desarrollo del sentido de la investigación por parte del estudiante. Con esto se logrará que los alumnos optimicen su conocimiento sobre el lenguaje de programación java y sobre la programación orientada a objetos, el cual es el objetivo de este curso.

Primero, en este documento encontrarán las especificaciones dadas por la profesora encargada de impartir este curso, en el cual se detallan las características que la aplicación debe tener, sus puntos a ser evaluados, y sus condiciones administrativas.

Luego se adjuntan las estrategias de solución, en las cuales se dará la propuesta de metodología de trabajo por parte de los miembros del equipo para resolver el problema. Se establecerán las tareas asignadas de cada integrante o cualquier otra actividad que se usó para lograr completar el proyecto. Para esto también se integrará un cronograma de trabajo. También se adjuntará el diagrama de clases terminado propuesto en notación UML. Aquí cada objeto aparecerá con la debida sintaxis junto con los atributos y métodos que se usaron en el proyecto para su solución. Después se dará una debida justificación de las relaciones establecidas entre los objetos del diagrama.

Se elaborará un listado de todas y cada una de las actividades y tareas que deben cubrirse a nivel funcional, en las cuales se aportará su porcentaje de realización y en caso contrario se aportará su debida justificación.

Además, se brindará el enlace al javaDoc generado del documento en el cual se realizó la aplicación. Aquí están documentados los métodos más relevantes incorporados en los distintos objetos. Los getters-setters serán ignorados.

Después, se aportarán los aspectos relevantes y las lecciones aprendidas por los integrantes del proyecto, en la cual se dará un listado de las lecciones aprendidas producto del desarrollo de la tarea programada.

Por último se presentará una bitácora de trabajo, en la cual se presentarán las actividades realizadas por el grupo un determinado día, incluyendo verificaciones realizadas de consultas con la profesora o asistente, si existieran.

**Instituto Tecnológico de Costa Rica**

IC-2101 Programación Orientada a Objetos

Profesores : Erika Marín, Ericka Solano, Luis Pablo Soto, Laura Coto Semestre II, 2016

**Especificación del II y III Proyecto**

El presente documento pretende definir los lineamientos de trabajo correspondiente a los proyectos II y III del curso Programación Orientada a Objetos, los cuales comprende un contexto específico que se detalla en este documento y una serie de actividades que deberán ser realizadas en los equipos de trabajo que se conformen.

**Objetivos para el II Proyecto**

* Elaborar un documento básico de especificación de requerimientos a través de un diagrama de casos de uso, que refleje una propuesta de comportamiento de un posible sistema ante la situación en contexto expuesta.
* Elaborar diagramas de actividad, secuencia y estado que complementen la propuesta de solución descrita en los casos de uso definidos.
* Explorar una herramienta de prototipado de aplicaciones para construir un prototipo asociado a la propuesta documental.

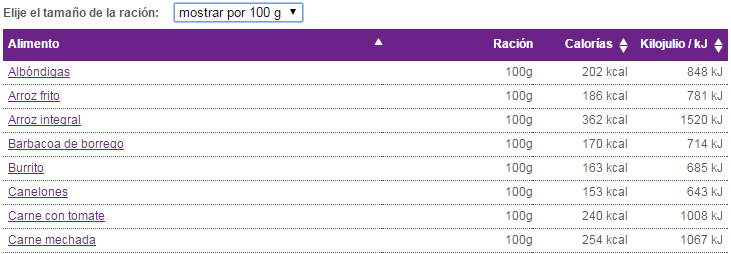
**Objetivos para el III Proyecto**

* Elaborar una propuesta de diseño de software que satisfaga los requerimientos establecidos y confirmados en la primera fase, aplicando el proceso básico completo de la Ingeniería de Software, para crear una propuesta de solución a la luz de los pilares fundamentales de la orientación a objetos: abstracción, encapsulamiento, herencia y polimorfismo.
* Poner en práctica las características de las clases abstractas, métodos abstractos, interfaces y excepciones.
* Incorporar patrones de diseño estudiados en clase (patrón creacional, adapter, strategy)
* Fomentar y desarrollar habilidades de investigación de forma individual por parte del estudiante, en los temas de:
  + Aplicación cliente-servidor y el uso de sockets.
  + Construcción y consumo de XML.
* Fomentar el desarrollo de las habilidades blandas centrándose en el trabajo en equipo
* Desarrollar habilidades de comunicación oral y escrita dentro del equipo de trabajo.
* Construir la aplicación en un ambiente de programación haciendo uso del lenguaje de programación Java.

**El contexto**

Una Pyme que se dedica a la preparación de alimentos ha decidido ofrecer a sus clientes y público en general el servicio de elaboración de menús múltiples y variados, aportando información nutricional de cada plato para permitir combinaciones de alimentos adecuadas a las propias necesidades.

Es por esto que ha definido un menú de alimentos en los cuales, se indica la cantidad de calorías que aporta en una porción como se muestra en la siguiente tabla:





Aunque en el establecimiento se pueden preparar gran cantidad de platillos, el administrador puede habilitar o definir los que estarán disponibles en un momento determinado del tiempo, de igual forma puede dar mantenimiento al catálogo, agregando nuevos platos, eliminando algunos que decida no ofrecerá más al público o modificando alguna información asociada.

Aplicación servidora - administradora

Por esta razón desea tener una aplicación en la que pueda dar gestión tanto al catálogo de productos que se pueden ofrecer en el nuevo establecimiento, así como dar seguimiento y atención a los pedidos de los clientes que visiten el local o que hagan sus pedidos en forma remota.

Con respecto a los productos del catálogo, se puede indicar que cada plato tiene un identificador que contiene una clave alfanumérica de máximo tres letras seguido de un guión y un número de 3 dígitos.

Así, si el plato es una entrada el código inicia con *ENT*, si es un comida principal el código inicia con *PRN* o si es postre inicia con PTR. Se ofrecen bebidas naturales y gaseosas cuyo código es BEB.

Adicionalmente se conoce el nombre del plato, una descripción y la representación de la medida de la ración recomendada con su respectivo valor en calorías.

Opcionalmente se puede mantener una imagen de referencia del platillo.

La ración puede contener una cierta cantidad de piezas por lo que también se puede manejar la posibilidad de conocer la cantidad de calorías por la cantidad de piezas que se incluyen dentro de la ración.

De esta forma, por ejemplo, un burrito puede tener la siguiente información:

|  |  |  |
| --- | --- | --- |
| **Código** | PRN-008 |  |
| **Nombre** | Burrito |
| **Descripción** | tortilla de trigo rellena de frijoles molidos, trozos de pollo y queso mozarella, cubierta con lechuga, tomate, aguacate y natilla |
| **Tamaño de la porción** | 400 gramos |
| **Piezas por porción** | 2 |
| **Calorías en 1 porción** | 326 kcal |
| **Calorías por pieza** | 163 kcal |
| **Precio** | ¢ 1500 |

Cuando el administrador inicia sus labores, puede visualizar todo el catálogo de productos que puede ofrecer, pero también puede filtrar esta información por tipo de platillo (ENT, PRN, PTR, BEB) o por la posibilidad de estar disponibles al público (visibles, no visibles, ambos).

En todo momento el administrador también puede ver la lista de pedidos que se han generado por parte de los clientes.

En esta primera etapa, se pretende únicamente el obtener una información particular del tipo de pedidos que se reciben. Posteriormente se desea dar atención a los pedidos recibidos y gestionar el pago.

Al administrador le interesa de manera particular conocer:

* La lista de los 10 productos que se solicitan con más regularidad indistintamente su categoría (TOP-TEN).

En este listado se muestra código, nombre, descripción y categoría ordenada descendentemente.

* Los productos que no se han pedido NUNCA. En este caso se muestra código, nombre y categoría ordenado por categoría y por código de producto.
* Relación porcentual entre pedidos en sitio, para recoger y express. En este caso del total de pedidos recibido se obtiene la relación porcentual por tipo de pedido y se muestra una tabla que muestre los valores de dicha distribución con respecto al total general.

Aplicación remota - cliente

Cualquier cliente que tenga acceso a la aplicación de la empresa puede realizar un pedido de comida a través de interactuar con una pantalla que le permite crear una lista de comidas que desea obtener.

Estos pedidos pueden ser en sitio, para pasar a recoger o express.

Cualquiera de los pedidos registra una fecha que es la del día, un número consecutivo global y el nombre de la persona que hace el pedido.

Si el cliente visita el restaurante puede gestionar por auto-solicitud su pedido pues cada mesa cuenta con una tableta desde la que puede realizar su pedido.

Si el pedido es solicitado para pasar a recoger, interesa registrar el nombre de la persona que va a pasar a buscar el pedido (puede ser distinta de quien hace el pedido) y su número de celular. En cuanto al costo hay un incremento en un porcentaje del costo total del pedido por concepto de empaque, cuyo valor es definido por el administrador como un parámetro externo de configuración de la aplicación servidora.

Finalmente, si el pedido es express, se solicita el número de celular de la persona que hace el pedido pero además el detalle de la dirección exacta donde debe ir a dejarse el pedido. En este último caso además hay un rubro por transporte que se cobra adicional que es un monto fijo establecido por el administrador como un parámetro externo de configuración de la aplicación servidora.

La pantalla remota o del cliente se organiza por los grupos de alimentos (ENTRADAS, PLATOS FUERTES, POSTRES, BEBIDAS) y el usuario puede desplegar la lista de los alimentos que desea solicitar, seleccionando cualquiera de los grupos de alimentos.

De cada producto puede solicitar uno o más porciones. Puede eliminar algún alimento si no lo desea, pero no puede quedar uno en su pedido con cero porciones.

Una vez que esté listo y decida enviar a procesar el pedido, se completan los datos necesarios para registrar el pedido en la aplicación dando a conocer al cliente los detalles de costo del mismo. Recuerde que en esta primera etapa no hay mecanismo de cancelación.

Cada vez que el usuario selecciona un producto se incrementa el total de calorías acumuladas en su pedido así como la cantidad de dinero que debe cancelar. De igual forma estos datos disminuyen si se altera la cantidad de porciones o se eliminan productos del pedido.

En esta primera etapa, al enviar el pedido se notifica el costo total de la orden y aún no hay opción de pago automático, por lo que el usuario debe cancelar en sitio o bien cuando le llegue el pedido express con el motorizado encargado de entregar la encomienda.

Consideraciones técnicas

La aplicación debe ser construida haciendo uso del lenguaje de programación Java y con una interfaz gráfica de usuario desarrollada en Swing o Fx según decisión del equipo de trabajo.

La aplicación debe ser desarrollada en una ambiente distribuido cliente-servidor, para lo cual deberá usar sockets para permitir la comunicación de varias sesiones en simultáneo a un servidor central.

El catálogo de alimentos se mantiene almacenado en un archivo con formato XML.

La aplicación servidora debe permitir el monitoreo de las conexiones (bitácora de comportamiento) así como las acciones que se llevan a cabo con el catálogo y las consultas derivadas de los pedidos. También puede ver el historial de los pedidos recibidos, refrescando cada vez que un nuevo pedido ingresa.

**Puntos a ser Evaluados**

**Entregable Fase I**

En el primer entregable debe confeccionar un documento con un diagrama de casos de uso y seleccionar tres de los casos más representativos de su propuesta para elaborar los diagramas de secuencia asociados. Algunos de los casos de uso importantes a considerar entre otros podrían ser por ejemplo:

1. Crear un pedido.
2. Autenticar al administrador
3. Mostrar el catálogo de productos con datos por porción o por piezas.
4. Eliminar un producto de un pedido.
5. Determinar el monto a pagar en un pedido express.

Adicionalmente debe proporcionarse un prototipo no funcional que refleje todas las necesidades expuestas en este documento tanto desde el rol de administrador como de usuario simple.

Debe reflejar la apariencia y navegabilidad de la aplicación por todas y cada una de las funcionalidades propuestas para la aplicación.

Para esto puede utilizar herramientas que se encuentran disponibles para el desarrollo de prototipos.

Consulte para ver algunas propuestas y demos de herramientas.

<http://www.campusmvp.es/recursos/post/Herramientas-de-prototipado-de-aplicaciones-Web.aspx>

<http://wwwhatsnew.com/2010/03/04/herramientas-para-creacion-de-prototipos-y-wireframes/>

<http://www.cobianmedia.com/2013/11/20/9-herramientas-para-crear-maquetas-interactivas-2/>

La entrega de este documento se realizaría el ***miércoles 2 de noviembre*** en formato de documento Word y su devolución sería el lunes lunes 5 de noviembre con lo que se puede comenzar la segunda parte del proyecto que comprende la programación de la aplicación de acuerdo a lo aprobado en la fase I.

**Entregable Fase II**

El entregable de fase II constituye la programación del sistema propuesto y aprobado en la primera etapa, el cual debe contener todos los requerimientos indicados en el documento de especificación de requerimientos.

La restricción que tiene esta implementación es que deberá contar con al menos un patrón de diseño de los que se estudiarán en clase, utilizando la arquitectura en capas según el modelo MVC.

El sistema debe ser implementado con interfaz gráfica y deberán diferenciarse las acciones que realizan el administrador y el otro usuario según corresponda.

Con respecto a la documentación de esta última fase, se debe aportar un documento formal que contenga:

* 1. Portada, índice, introducción
  2. Información de contexto del problema
  3. Estrategia de Solución:
     1. Propuesta de metodología de trabajo por parte de los miembros del equipo para abordaje del problema, establecimiento de tareas o cualquier otra actividad que aporte a la solución del problema.
     2. Cronograma de trabajo.
     3. Diagrama de clases definitivo propuesto en notación UML, el cual debe ser aportado en formato de imagen dentro del documento lo más legible posible. Cada objeto deberá aportar la sintaxis adecuada y correcta en la definición de atributos y métodos, así como las relaciones que se establezcan entre ellos. Para cada objeto deberá aportar un constructor default, un constructor sobrecargado, omita los métodos setters-getters, toString, equals, pero agregue cualquier otro método que se requiera para dar solución al problema propuesto.
     4. Justificación de uso de los patrones utilizados en los sitios que corresponda dentro del modelo.
  4. Análisis de Resultados  
     Deberá elaborar un listado de todas y cada una de las actividades y tareas que deben cubrirse a nivel funcional, para cada una de ellas debe aportar el porcentaje de realización y en caso de no ser el 100% debe justificarse.
  5. Enlace al JavaDoc generado (documentación interna de la aplicación con los métodos más relevantes incorporados en los distintos objetos). No es necesario documentar los getters-setters.
  6. Aspectos relevantes y Lecciones aprendidas: Debe prepararse un listado de las lecciones aprendidas producto del desarrollo de la tarea programada. Las lecciones aprendidas pueden ser de carácter personal y/o técnico que involucre aspectos que han logrado un aprendizaje en temas de investigación, desarrollo de habilidades técnicas y habilidades blandas como trabajo en equipo, comunicación, forma de expresar ideas, entre otros.
  7. Bitácora de trabajo durante las tres semanas de trabajo, incluyendo verificaciones realizadas (si existieran) de consultas realizadas con el profesor o asistente.
  8. Bibliografía y fuentes digitales utilizadas
  9. Anexos: Debe agregar el documento de especificación mejorado según observaciones obtenidas en la fase I.

La fecha de entrega de la última fase del proyecto será el miércoles 23 de noviembre, fecha en la que se realizarán las revisiones del mismo según horario asignado por la profesora.

Consideraciones Administrativas

1. El desarrollo de la tarea es estrictamente en **tríos** máximo.
2. El proyecto en sus dos fases debe ser entregado a tiempo en rubro de evaluaciones del TEC Digital.

**Estrategia de Solución:**

El grupo como metodología de trabajo utilizó una combinación de 2 modelos de desarrollo del software: el modelo cascada y el modelo evolutivo. A continuación se explicarán brevemente en qué consiste cada una:

* **Cascada:** en este modelo las etapas en las cuales se procederán para realizar el proyecto se ordenan, de manera que, a la hora de pasar por cada una, para iniciar una etapa debe estrictamente haberse terminado la etapa anterior.
* **Evolutivo:** en este modelo de procesamiento el software evoluciona con el tiempo, los requisitos del usuario y del producto suelen cambiar conforme se desarrolla el mismo, lo cual crea flexibilidad para cambios en los requisitos solicitados por el cliente.

La combinación a la hora de ponerse manos a la obra se dan en que, utilizamos el modelo cascada a la hora de primero realizar el diagrama de los objetos, clases, y métodos en formato UML. Hasta no haber concluido esa tarea no íbamos a proceder a hacer el código en java. Luego de haber realizado esto, utilizamos el modelo evolutivo a la hora de construir el proyecto, ya que al dividirnos las tareas se iban a poder completar el proyecto por partes completas para luego juntarlo en un solo proyecto completo.

**Cronograma:**

|  |  |  |  |
| --- | --- | --- | --- |
| **FECHA** | **ACTIVIDADES** | **HORAS** | **RESPONSABLES** |
| Miércoles 16 de Noviembre  19:00 | Se hicieron las bases para el servidor del lado del administrador | 9 | Esteban, José David |
| Lunes 21 de Noviembre  13:00 | Se hicieron las bases para el servidor del lado del cliente | 4 | Esteban, José David |
| Martes 22 de Noviembre  14:00 | Se implementaron algunos métodos gráficos para el cliente | 12 | Esteban |
| Miércoles 23 de Noviembre  2:00 | Se implementaron algunos métodos para las funciones de consulta del administrador | 10 | Esteban |
| Miércoles 23 de Noviembre  00:00 | Se implementaron el resto de métodos gráficos para el cliente | 9 | Esteban, José David |
| Miércoles 23 de Noviembre  09:00 | Se implementó el cargado y guardado de archivos XML | 12 | José David, Esteban |
| Jueves 24 de Noviembre  21:00 | Se terminaron todas las funciones del servidor | 8 | Esteban |
| Viernes 25 de Noviembre  05:00 | Se finalizaron los métodos gráficos necesarios para el funcionamiento del administrador | 12 | Esteban, José David |
| Viernes 25 de Noviembre  17:00 | Se finalizaron los métodos gráficos para las consultas del administrador y los filtros del cliente | 16 | Esteban, José David |
| Sábado 26 de Noviembre  13:00 | Se habilitaron todas las funciones en el servidor para su uso en la interfaz del cliente | 4 | José David |
| Sábado 26 de Noviembre  13:00 | Se finalizaron todas las funciones tanto gráficas como de datos del proyecto | 8 | Esteban |
| Domingo 27 de Noviembre  16:00 | Se trabajó en la documentación del proyecto | 2 | José David |
| Domingo 27 Noviembre  10:00 | Se implementaron los métodos gráficos para algunas características estéticas del cliente y el administrador | 1 | Esteban |

**Diagrama de clases en formato UML:**

**Justificación de uso de los patrones utilizados:**

**Strategy:**

* Servidor: El servidor implementa un patrón strategy, puesto que era el más adecuado para adaptarse a poder ejecutar varios algoritmos dependiendo de cuál fuera la solicitud recibida.
* Tablas con Consultas de Productos Pedidos: Para estas tablas, había que realizar un algoritmo que llenara las tablas con los productos, más pedidos, menos pedidos y todos los pedidos, así que se implementó este patrón.
* Filtros y Ordenamientos: Había que diseñar un algoritmo para saber si ordenar o no, cómo ordenar y cómo filtrar, así que se implementó este patrón.

**Factory:**

* Pedidos: A la hora de crear los pedidos se decidió implementar un patrón Factory, pues los pedidos pueden ser de diferentes tipos, lo que hacía este patrón el adecuado para la tarea.
* Creación de Mensajes: A la hora de crear los nuevos mensajes de solicitudes del servidor había que hacer a alguien capaz de fabricarlos todos, la solución: el patrón Factory.

**Observer:**

* Tablas: Las tablas de control de pedidos y conexiones, así como el contador de pedidos totales implementan este patrón puesto que era más factible hacer algo que las actualizara sin tener que obligar al administrador a navegar por la interfaz.

**Builder:**

* Creación de Productos: Había que crear nuevos productos, los cuales tenían la misma estructura, pero con diferentes datos, lo que hizo ideal el implementar el patrón Builder.

**Análisis de Resultados:**

|  |  |
| --- | --- |
| **Mostrado de Productos al Cliente** | **100%** |
| **Mostrado de Productos al Administrador** | **100%** |
| **Petición de pedidos** | **100%** |
| **Filtro y ordenado de los productos** | **100%** |
| **Actualización de calorías y coste de productos seleccionados** | **100%** |
| **Agregado de nuevos productos** | **100%** |
| **Modificado de productos** | **100%** |
| **Top 10** | **100%** |
| **Peores productos** | **100%** |
| **Atendido de peticiones del servidor** | **100%** |
| **Mostrado de pedidos y de conexiones al administrador** | **100%** |

**Enlace al JavaDoc generado:**

**file:///home/esteban/NetBeansProjects/II-Progra-POO/dist/javadoc/index.html**

**Aspectos relevantes y Lecciones aprendidas:**

**Trabajo en equipo:** en la elaboración de esta tarea programada, al realizarse en parejas, el trabajo tiene que ser bien repartido, para que, en el tiempo establecido, se logre completar toda la carga. Por eso es que una importante lección es el trabajo en equipo, ya que si este valor en el grupo, sería imposible realizar ni esta tarea ni ningún otro trabajo, ya sea en la universidad o en el campo laboral, ya que en la carrera de Ingeniería en computación, un equipo trabajando unido es fundamental para realizar cualquier actividad programada.

**Perseverancia:** si bien es cierto que programar no es fácil, esta no es la excepción, ya que en estas tareas programadas, cada estudiante tiene que averiguar la manera de cómo resolver un problema, implementado un algoritmo que lo haga. El problema es que la solución nunca va a salir al primer intento y ahí personas que se rinden muy fácilmente, sólo por el hecho de creer que no pueden hacerlo. Por eso es importante ser perseverante, y luchar hasta lograr nuestros objetivos, ya que sólo así se arribará a la meta.

**Sacrificio:** realizar una tarea programada no es una labor fácil, y esto toma bastante tiempo, ya que por esta razón nos asignan 3 semanas de tiempo para entregarla. Si el estudiante lleva varios cursos, necesitará dedicar el debido tiempo a cada uno. Pero, este valioso tiempo para repasar otras materias se puede ver afectado, ya que se necesita también de tiempo para poder realizar una gran tarea programada y esto causa que el estudiante tenga que sacrificar tiempo de estudio y hasta de sueño para poder cumplir con el trabajo.

**Investigación:** en esta tarea, sin duda una gran lección que destaca es el valor del sentido de la investigación, ya que este es importante en la vida de un programador. No siempre estará un profesor o un amigo para ayudar, por eso la persona que estudia debe ser capaz de encontrar la respuesta por los medios que aportan información, ya sean libros, internet, entre otros.

**Bitácora de trabajo:**

**17 de noviembre de 2016:**

**Lugar: Laboratorio H, Escuela de computación.**

**Hora: 3:00 p.m.**

El grupo se reunió para planear el trabajo y discutir sobre cómo lo realizaríamos. Decidimos hacer primero el diagrama de las clases conteniendo los objetos en formato ULM para poder planear la aplicación y proceder a programar en java. Las tareas se dividieron, pero primero entre todos creamos las clases junto con los objetos instanciados y los debidos atributos que acompañarán a esos objetos. Luego se investigó información acerca de cómo crear archivos en formato XML para hacer el catálogo de los productos que se van a utilizar en el proyecto.

**23 de noviembre de 2016:**

**Lugar: Laboratorio H, Escuela de computación.**

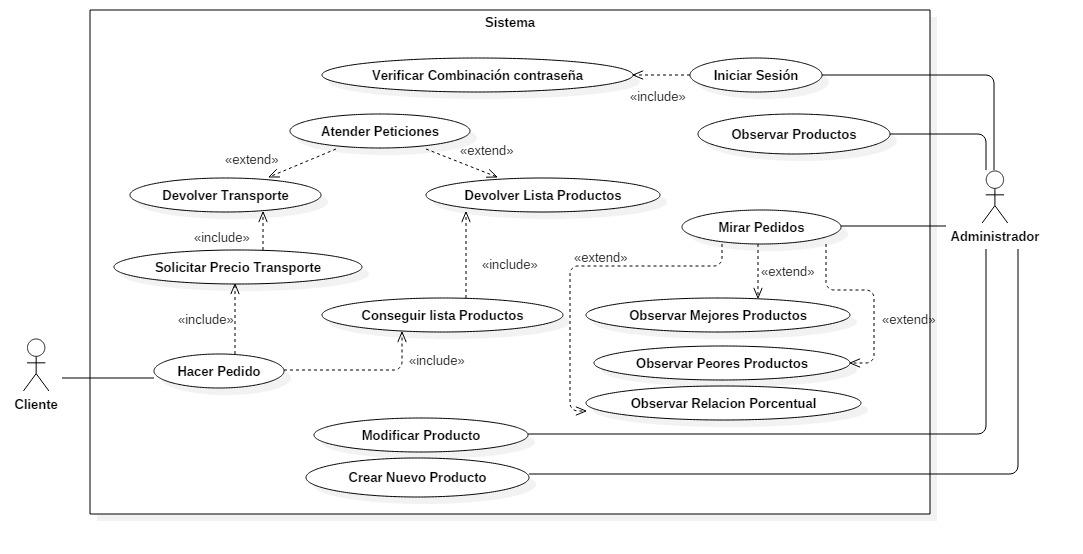
**Hora: 3:00 p.m.**

Se trabajaron en los servidores de cliente y administrador, en los cuales se manejaron una serie de pedidos creados a partir de un xml. En administrador se controlan los pedidos del cliente, los productos y su historial e inclusive puede agregar nuevos productos para su venta, mientras que el cliente obtiene el servicio de comidas disponibles según el admin para pedir en el local o express.

**Bibliografía y fuentes digitales utilizadas:**

**Anexos:**

**Diagrama de casos de uso:**



**Emplantillado de los casos de uso:**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-01** | | |
| Use Case Name: | **Hacer Pedido** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Cliente |
| Description: | Se enfoca en conseguir cuáles son los productos que pedir el cliente, así como sus datos |
| Trigger: | El botón de hacer pedido |
| Preconditions: | 1. Ninguna |
| Postconditions: | 1. El pedido está almacenado en el servidor |
| Normal Flow: | 1. El sistema ejecuta el CU-02 Conseguir Lista de Productos 2. Se le solicita al usuario los productos que desea pedir. 3. Se le solicita al usuario que indique el nombre respecto al cual se le entregará el pedido. 4. Se le solicita al usuario que indique su número de teléfono para ser contactado. 5. En caso de que el usuario indique que se trata de un pedido express, si no, continúe con el número 7, entonces se le solicitará además una dirección física en cuál dejar el pedido. 6. En caso de ser express ejecutará el caso de uso CU-03 Solicitar Precio Transporte 7. Luego de comprobar exitosamente los datos se indicará en pantalla el coste de los productos |
| Alternative Flows: | 1. El Sistema indica que la cantidad de productos tiene que ser diferente de 0. 2. El Sistema indica que su nombre no puede ser nulo 3. El Sistema indica que su número de teléfono no puede ser nulo 4. El Sistema indica que al tratarse de un pedido express, su dirección no puede ser nula 5. El Sistema indica que el servidor no está disponible para tramitar su solicitud |
| Exceptions: | * El servidor no está en línea |
| Includes: | CU-02 Conseguir Lista Productos  CU-03 Conseguir Transporte |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-02** | | |
| Use Case Name: | **Conseguir Lista de Productos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Solicita los productos al servidor |
| Trigger: | El CU-01 Hacer Pedido |
| Preconditions: | 1. El servidor está en línea |
| Postconditions: | 1. La lista de productos está en el sistema |
| Normal Flow: | 1. Se llama al caso de uso CU-04 Devolver Lista Productos 2. Se coloca la lista de productos en el sistema |
| Alternative Flows: | 1. El sistema muestra que el servidor no está disponible y que no es capaz de adquirir la lista de productos |
| Exceptions: |  |
| Includes: | CU-04 Devolver Lista Productos |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-03** | | |
| Use Case Name: | **Conseguir Precio Transporte** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Solicita el precio del transporte al servidor |
| Trigger: | El CU-01 Hacer Pedido |
| Preconditions: | 1. El servidor está en línea |
| Postconditions: | 1. Se consiguió el costo del pedido express |
| Normal Flow: | 1. Se llama al CU-05 Devolver Transporte 2. Se envía el valor del transporte mediante la conexión |
| Alternative Flows: | 1. El sistema muestra que el servidor no está disponible y que no es capaz de adquirir la lista de productos |
| Exceptions: |  |
| Includes: | CU-05 Conseguir Tranporte |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-04** | | |
| Use Case Name: | **Devolver Lista Productos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Revisa cada uno de los productos y consigue el coste de pedido express |
| Trigger: | El CU-02 Conseguir Lista Productos |
| Preconditions: | 1. El servidor está en línea |
| Postconditions: | 1. Se retornó la lista de productos de la base de datos del servidor |
| Normal Flow: | 1. Se recibe mediante la conexión una petición de la lista de productos 2. El servidor consigue la lista completa de productos 3. Se envía la lista de productos de vuelta |
| Alternative Flows: | 1. El sistema muestra que no pudo enviar los datos de regreso a la aplicación cliente |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-05** | | |
| Use Case Name: | **Devolver Transporte** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Revisa cada uno de los productos y consigue el coste de pedido express |
| Trigger: | El CU-01 Hacer Pedido |
| Preconditions: | 1. El servidor está en línea |
| Postconditions: | 1. Se retornó la lista de productos de la base de datos del servidor |
| Normal Flow: | 1. Se recibe mediante la conexión los productos de los cuales se desea saber el costo del transporte 2. El servidor consigue el precio total del transporte 3. Se envía el precio del transporte de vuelta |
| Alternative Flows: | 1. El sistema muestra que el servidor no hay ninguna aplicación conectada. 2. El sistema muestra que alguno de los productos no existe 3. El sistema muestra que no pudo enviar los datos de regreso a la aplicación cliente |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-06** | | |
| Use Case Name: | **Atender Peticiones** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Este es el ciclo que va a estar constantemente revisando si hay alguna solicitud pendiente |
| Trigger: |  |
| Preconditions: | 1. El servidor está en línea |
| Postconditions: | 1. Se resolvió una solicitud |
| Normal Flow: | 1. Se recibe mediante la conexión alguna solicitud 2. El sistema bifurca en los Casos de Uso CU-04 Devolver Lista Productos o CU-05 Devolver Transporte 3. El sistema realiza la solicitud |
| Alternative Flows: | 1. Ocurrió algún error durante la realización de la solicitud |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-07** | | |
| Use Case Name: | **Iniciar Sesión** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo mediante el cual el administrador iniciará sesión en el sistema |
| Trigger: | El botón de iniciar sesión |
| Preconditions: | 1. Ninguna |
| Postconditions: | 1. El administrador tendrá su sesión iniciada en el sistema |
| Normal Flow: | 1. El sistema recibe el nombre de usuario del administrador 2. El sistema recibe la contraseña del administrador 3. El sistema llama al caso de uso CU-0X Verificar Combinación Constraseña 4. El sistema inicia la sesión del administrador |
| Alternative Flows: | 1. El sistema muestra que el nombre de usuario no puede ser nulo 2. El sistema muestra que la contraseña no puede ser nula 3. No existe una combinación válida para el nombre de usuario y la contraseña |
| Exceptions: |  |
| Includes: | CU-0X Verificar Combinación Contraseña |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-08** | | |
| Use Case Name: | **Verificar combinación contraseña** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: |  |
| Description: | Este es el mecanismo que verifica si la combinación de nombre de usuario-contraseña es válido |
| Trigger: | CU-07 Iniciar Sesión |
| Preconditions: | 1. Ninguna |
| Postconditions: | 1. Se verificó la combinación del nombre de usuario junto con la contraseña |
| Normal Flow: | 1. El sistema revisa entre toda la base de datos para buscar si la combinación de nombre de usuario con la contraseña coinciden con un administrador registrado 2. Se verifica la combinación |
| Alternative Flows: | 1. La combinación ingresada no coincide con la combinación registrada de la base de datos de los administradores. |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-09** | | |
| Use Case Name: | **Mostrar Productos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo que muestra en pantalla una tabla con todos los productos |
| Trigger: | El botón de Mostrar Todos los Productos |
| Preconditions: | 1. Que algún administrador haya iniciado sesión |
| Postconditions: | 1. Se muestran todos los productos en pantalla |
| Normal Flow: | 1. El sistema muestra una ventana con todos los productos y sus detalles |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-10** | | |
| Use Case Name: | **Modificar Producto** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo para modificar los valores de un producto en específico |
| Trigger: | El botón de Modificar Producto |
| Preconditions: | 1. Que algún administrador haya iniciado sesión y haya 1 o más productos |
| Postconditions: | 1. Uno de los productos del catálogo resulta modificado |
| Normal Flow: | 1. El sistema muestra pide un identificador del producto que se desea modificar 2. El sistema va a mostrar una ventana que va a permitir modificar el producto seleccionado previamente 3. Se guardan los cambios y se cierra la ventana retornando a la ventana principal de la aplicación administradora |
| Alternative Flows: | 1. El sistema muestra un mensaje de que el producto ingresado no existe y se retornará a la ventana principal de la aplicación administradora |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-11** | | |
| Use Case Name: | **Crear Producto** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo para modificar los valores de un producto en específico |
| Trigger: | El botón de Modificar Producto |
| Preconditions: | 1. Que algún administrador haya iniciado sesión y haya 1 o más productos |
| Postconditions: | 1. Se crea un nuevo producto en el catálogo de productos |
| Normal Flow: | 1. El sistema va a mostrar una ventana para recibir todos los datos del nuevo producto 2. Se guardan el nuevo producto en el sistema y se cierra la ventana retornando a la ventana principal de la aplicación administradora |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-12** | | |
| Use Case Name: | **Mostrar Pedidos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo que actualiza los pedidos recibidos en la aplicación administradora |
| Trigger: | CU-07 Iniciar Sesión |
| Preconditions: | 1. Que algún administrador haya iniciado sesión |
| Postconditions: | 1. Se muestran todos los pedidos en pantalla |
| Normal Flow: | 1. El sistema verifica si ha llegado algún pedido al sistema 2. Si hay un pedido, entonces lo agrega a la tabla visualizadora de pedidos |
| Alternative Flows: | 1. Verifica de nuevo |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-13** | | |
| Use Case Name: | **Observar Mejores Productos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo que muestra los productos que más se hayan pedido |
| Trigger: | El botón de observar mejores productos |
| Preconditions: | 1. Que algún administrador haya iniciado sesión |
| Postconditions: | 1. Se muestran los productos más pedidos en pantalla |
| Normal Flow: | 1. El sistema busca en el historial de pedidos cuáles fueron los productos más pedidos 2. Muestra los productos más pedidos en la pantalla |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-14** | | |
| Use Case Name: | **Observar Peores Productos** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

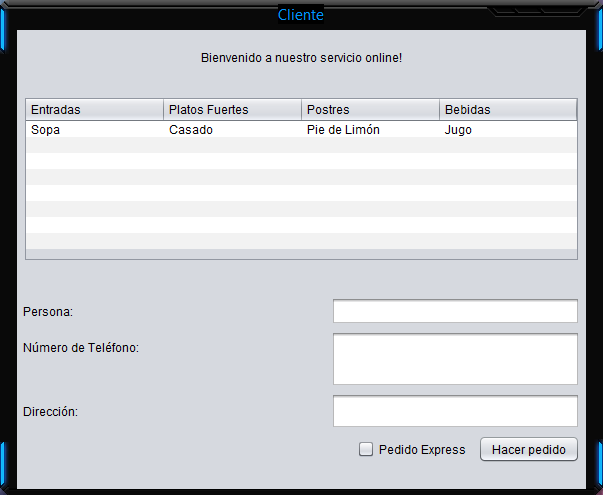
|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo que muestra los productos que menos se hayan pedido |
| Trigger: | El botón de observar peores productos |
| Preconditions: | 1. Que algún administrador haya iniciado sesión |
| Postconditions: | 1. Se muestran los productos menos pedidos en pantalla |
| Normal Flow: | 1. El sistema busca en el historial de pedidos cuáles fueron los productos menos pedidos 2. Muestra los productos menos pedidos en la pantalla |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **CU-15** | | |
| Use Case Name: | **Observar Relación Porcentual** | | |
| Created By: | Esteban Gutiérrez Salas | Last Updated By: | 2.11.2016 |
| Date Created: | 30.10.2016 | Date Last Updated: | 2.11.2016 |

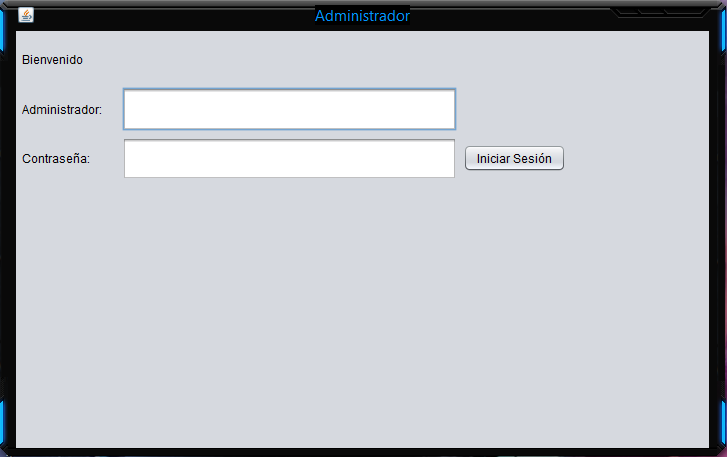
|  |  |
| --- | --- |
| Actors: | Administrador |
| Description: | Este es el mecanismo que muestra la relación porcentual entre productos |
| Trigger: | El botón de observar relación porcentual |
| Preconditions: | 1. Que algún administrador haya iniciado sesión |
| Postconditions: | 1. Se muestran la relación porcentual en pantalla |
| Normal Flow: | 1. El sistema busca en el historial de pedidos cuál es la relación porcentual 2. Muestra la relación porcentual en la pantalla |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Priority: |  |
| Frequency of Use: |
| Business Rules: |
| Special Requirements: |
| Assumptions: |
| Notes and Issues: |

**Prototipo no funcional: (Seis imágenes)**

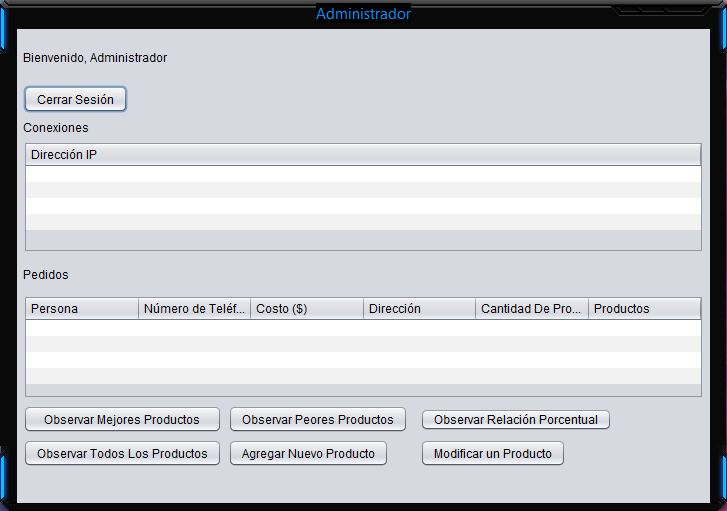
Aplicación Cliente



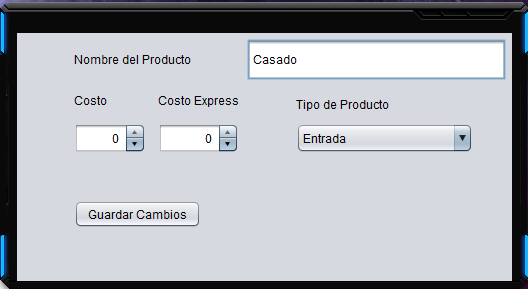
Aplicación Administrador sin haber iniciado sesión



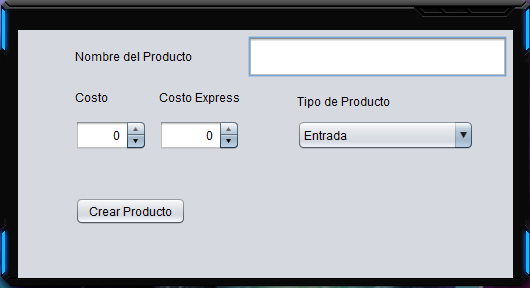
Aplicación Administrador habiendo iniciado sesión



Modificar Producto



Crear Producto



Lista Completa de Productos

