Sistema de apoyo a la administración, venta e inventario para pub restorán SabCity.

27-12-2013

Concepción - Chile

Oscar Nahuelpán; Ingeniería de ejecución en computación e informática.

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad de Bío-Bío en el proceso de titulación para la carrera de Ingeniería de Ejecución en Computación e Informática. El proyecto se titula "Sistema de apoyo a la administración, venta e inventario para pub restorán SabCity.".

El proyecto se realiza para la empresa Pub Sab City "Discoteque". El software gestiona información sobre los procesos de generación de pedidos, venta y productos, con la finalidad de mejorar la comunicación entre las áreas de barra, cocina y sector de mesas y mejorar la atención al cliente.

Para el desarrollo del sistema se optó por crear dos proyectos, haciendo uso de la arquitectura J2EE, frameworks struts y Jquery Mobile. Esto se hizo con el fin de poder repartir mejor el trabajo dentro de la empresa.

Implementando este sistema en la empresa se mejorará la comunicación entre las áreas de barra, cocina y sector de mesas y mejorar la atención al cliente.

Índice

1 INTRODUCCION

El presente trabajo corresponde a una memoria de título de la carrera Ingeniería de Ejecución en computación e Informática, perteneciente a la Universidad del Bío-Bío.

El documento se organiza de la siguiente forma; la *Sección nº2 y nº3* describe la situación actual de la empresa en donde se desenvuelve el sistema y, por otra parte, definir de manera clara de que se tratará el proyecto en general.

La *Sección nº4*, se detalla, entre otras cosas, los requerimientos funcionales del sistemas, los cuales fueron tomados por el equipo de trabajo durante el desarrollo del proyecto.

La *Sección nº5* analiza las factibilidades que el sistema posee en tres áreas concretas; Técnica, Operativa y Económica, para luego concluir de manera clara si es factible o no realizar el proyecto de Software.

La *Sección nº6* contiene un análisis de los requerimientos traducidos a los correspondientes casos de uso con todas sus descripciones y además el modelamiento de datos.

La *Sección nº7* explica el "Modelo Relacional" que servirá para la creación de la base de datos del sistema.

La *Sección nº8* detalla los casos de pruebas del sistema, en conjunto con sus componentes, módulos y/o sistemas con los cuales se prueban.

La *Sección nº9 y nº10* presenta los planes de capacitación al usuario y el de implantación del sistema en la empresa.

2 DEFINICION DE LA EMPRESA O INSTITUCION

2.1 Descripción de la empresa

Pub Sab City Discoteque es un pub-discoteque ubicado en la zona de Coronel.
El local inició sus actividades el 13 de Mayo del 2011 y hasta la fecha han logrado formar una buena reputación en el sector de Coronel y lugares cercanos. Además su firma (SAB producciones) es reconocida al menos a nivel regional desde hace 7 años gracias a la fiesta electrónica que se realiza todos los veranos en playa blanca llamada "Sun at the beach".

El local actualmente consta de 2 pisos; en el primero se ubica la discoteque que contiene una barra donde solo tienen licores a la venta y una pista de baile.
En el segundo se encuentra los sectores de mesa, barra y cocina. En este piso las meseras son quienes se encargan de la atención al cliente, a diferencia del primer piso en el cual los clientes se dirigen a la barra a solicitar y cancelar algún pedido. Además aquí se tiene una carta donde exponen la mayor parte de los productos que ofrecen.

Visión

Ser uno de los mejores pub restaurante de Coronel, reconocidos por nuestros clientes y poder expandirnos a Concepción y Lota, gracias a la calidad y variedad de nuestros productos y servicios.

Misión

Mostrar una imagen distinta a los locales de la zona, con una decoración agradable, innovadora y cómoda para el gusto de todas las personas explorando tendencias de música y ambiente que se enfoquen en el placer, comodidad y elegancia. Además de ofrecer calidad y variedad en nuestros productos y servicios, mejorar gradualmente la estructura y ambiente de la empresa para entregar una imagen cada vez mejor a nuestros clientes.

Los datos actuales de la empresa son los siguientes.

Datos clientes

Nombre del local:	Pub Sab City Discoteque
Rut:	76.144.403-4
Giro:	Banquetes y eventos
Dirección:	Los Álamos 3202 Esq. Las Encinas. Lagunillas 3 Coronel.
Teléfono:	2719348

Interlocutores

Nombre:	Juan Carlos Venegas
Cargo:	Administrador
Nombre:	Alexis Moya
Cargo:	Administrador

2.2 Descripción del área de estudio

La empresa Pub Sab City Discoteque ha detectado la necesidad de mejorar el servicio de atención al cliente, control de ventas y el control de inventario del local referente al stock y consumo de productos del segundo piso. Esto porque solo en esta parte del local tienen sectores de mesas donde atienden meseras a los clientes.

Atención al cliente

En los sectores de mesa, la toma de pedidos las realizan las meseras usando comandas que, una vez anotados los productos solicitados, los entregan a la barra o cocina dependiendo del tipo de producto solicitado por la mesa. La cantidad solicitada de cada producto por el cliente se va registrando en libretas, cada una de las páginas representa una mesa perteneciente al sector que tiene designado.

En la barra, el barman es quien atiende al cliente. En este caso, el cliente debe dirigirse directamente a la caja para cancelar su pedido.

Control de ventas

Cuando el cliente solicita su cuenta, la mesera encargada de atenderlo suma todo lo registrado en el pedido y lo anotan en una boleta. Al terminar la jornada laboral suman todo lo vendido en ambos pisos y lo guardan en un archivador.

Control de inventario

Para mantener un control del inventario, los administradores crearon una plantilla Excel para realizar el conteo de productos e ingredientes que tienen disponible en las áreas pub, discoteque y cocina. El conteo lo realizan al término de la jornada del último día de trabajo de la semana que por lo general son los domingos en la madrugada. Finalmente los administradores programan las compras necesarias para mantener el stock mínimo de cada producto e ingrediente.

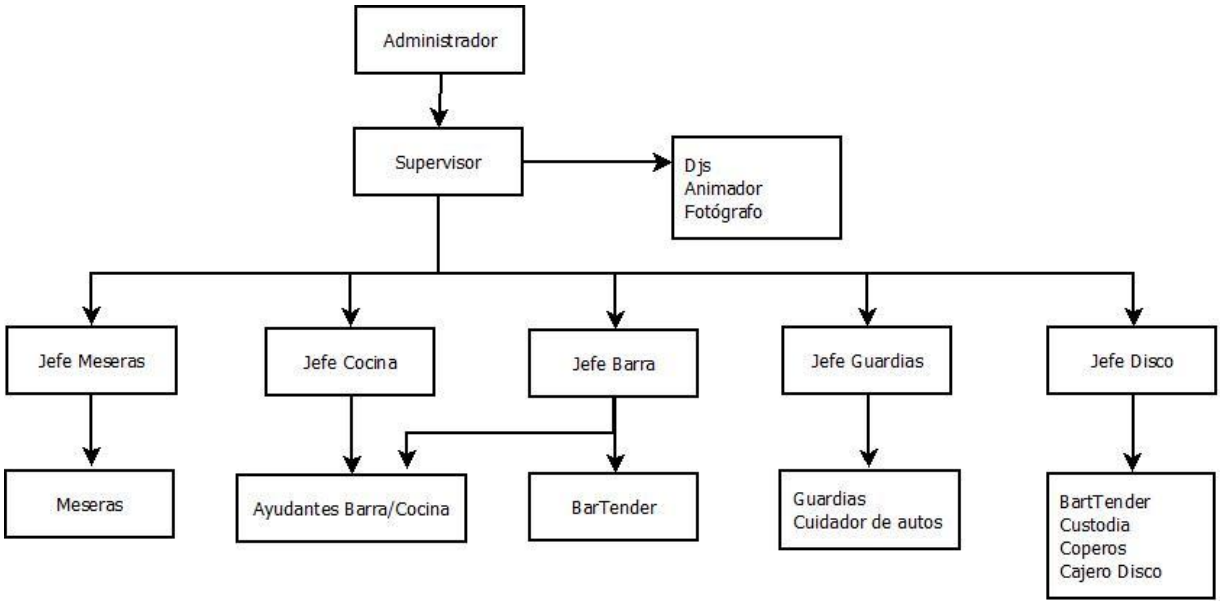


Ilustración 1: Organigrama pub restorán Sab City

La siguiente tabla muestra la cantidad de trabajadores que hay en una jornada de trabajo normal.

Tabla nº1: Cantidad de trabajadores en una jornada laboral

Personal	Cantidad		Personal	Cantidad
Administrador	2		Meseras	6
Supervisor	1		Ayudantes Barra/Cocina	2
Dj	2		Bartender	3
Animador	1		Guardias	7
Fotógrafo	1		Cuidador de Autos	1
Jefe Meseras	1		Custodia	1
Jefe Cocina	1		Coperos	2
Jefe Barra	1		Cajero Disco	1
Jefe Guardias	1			
Jefe Disco	1			

2.3 Descripción de la problemática

Atención al cliente

Actualmente la información sobre el consumo del cliente solo está disponible en las libretas donde las meseras llevan la cuenta de las mesas. Esta información en caso de que el cliente la solicite debe dirigirse a la caja, ya que esta es exclusiva del cajero y las meseras. En días de mucha afluencia de público, el tiempo de atención al cliente o mesa se ve afectado debido a que las meseras deben hacer fila para poder llenar su libreta.

En la ilustración 2 se muestra el proceso de atención al cliente en las mesas.

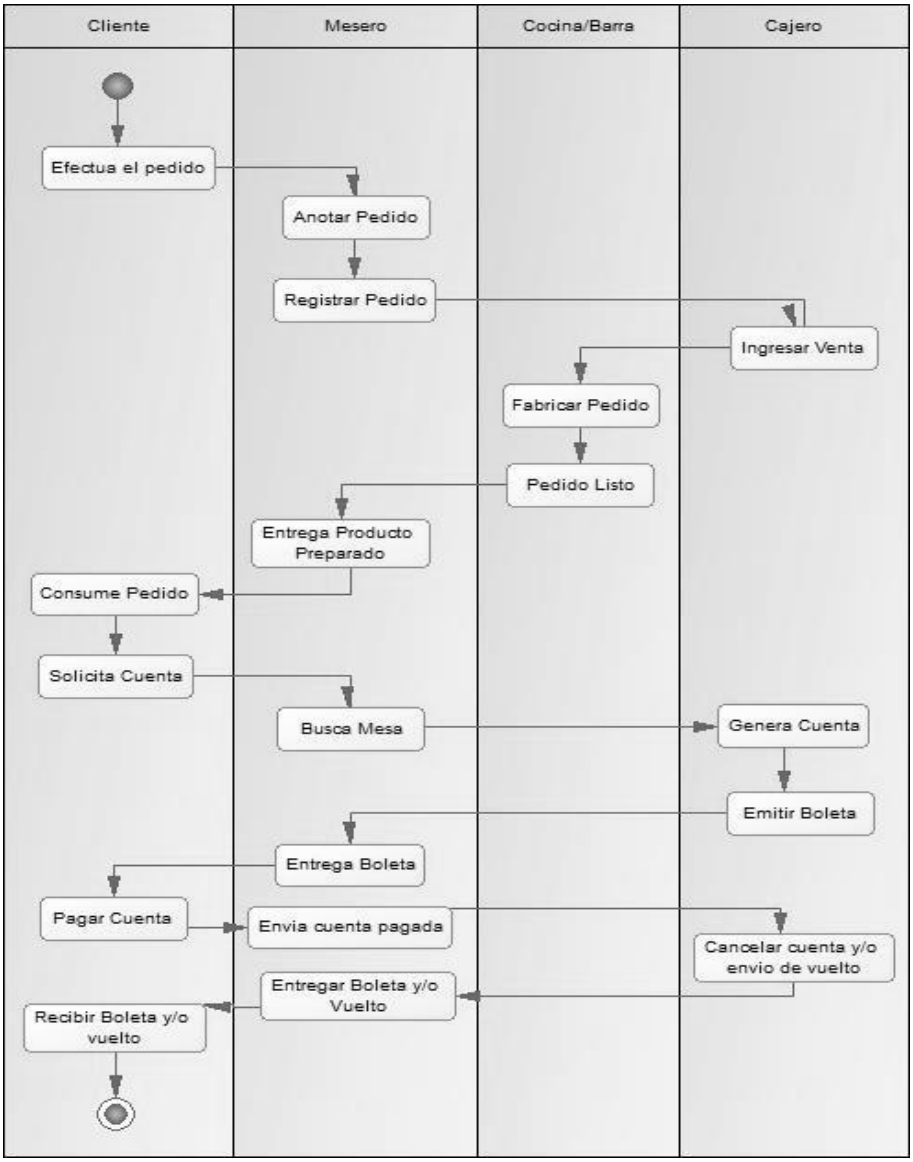


Ilustración 2: Procesos de negocios: Atención al cliente.

Control de ventas

Actualmente solo se mantiene un conteo del total de ventas registrado durante la jornada, pero no se tiene mayor detalle de estas, como por ejemplo, la disponibilidad de lo solicitado por el cliente en forma inmediata.

Control de productos

No existe un registro del consumo aproximado de los productos e ingredientes.

3 DEFINICION PROYECTO

3.1 Objetivos del proyecto

Objetivo general.

Implementar una aplicación desarrollada en web mobile específicamente para la generación de pedidos y una aplicación que apoye a la administración, centrado en los procesos de gestión de usuarios, control de ventas e inventario para el pub-restaurant "Pub Sab City Discoteque".

Objetivos específicos.

1. Identificar las distintas situaciones que se presentan en la empresa, asociadas a la perdida de información, además de gestionar la interacción de las diferentes áreas donde se implementará el sistema.

3.2 Ambiente de Ingeniería de Software

Metodología de desarrollo a utilizar.

Se opta por el *Modelo de desarrollo incremental* porque permite entregar al cliente el producto más rápido en comparación del modelo cascada. Los requerimientos del proyecto pueden ser evaluados en conjunto con el cliente, con el fin de que el sistema cumpla mejor sus necesidades.

Técnicas y notaciones.

Se realizan diversas entrevistas con el cliente, las cuales ayudan a completar los requerimientos funcionales del sistema. Dichas entrevistas son las que forman los diferentes incrementos.

Se utilizan las herramientas clásicas de modelamiento de información, como el Modelo Entidad Relación (MER), el Modelo Relacional, además de los diagramas de Caso de Usos (UML v.1) para representar mejor la funcionalidad del sistema.

Estándares de documentación.
<Adaptación basada en *IEEE Software requirements Specifications Std 830-1998 ISO/IEC 9126*: Tecnología de Información – Evaluación del producto de software.

Herramientas de apoyo al desarrollo de software que se utilizan.

- **Power Designer 15**: Utilizado en la creación de diagramas UML, modelos de bases de datos, etc.

- **Eclipse**: Editor de códigos Java, HTML, Jquery, CSS, entre otros. Utilizado para generar y visualizar la codificación de todo el sistema.

- **Weblogic Bea 8.1**: Servidor de aplicaciones j2EE y también un servidor web HTTP de Oracle disponible para Unix, Linux, Microsoft Windows entre otras.

- **Maven**: Herramienta de software que permite la administración de librerías asociadas al proyecto y organizar el proceso de construcción y gestión de estos. Esta herramienta utiliza POM (Project Object Model) para describir el proyecto de software a construir, sus dependencias entre módulos y componentes externos. Maven está basado en la idea de la reutilización de la lógica de construcción. Como los proyectos generalmente se construyen en patrones similares, una elección lógica podría ser reutilizar los procesos de construcción.

- **Microsoft Word 2010:** Software ocupado para la edición y confección de los informes y manuales que posee el sistema.

- **Navegador Google Chrome:** Utilizado en las constantes pruebas del sistema.

- **Navegador Mozilla Firefox**: Utilizado en las constantes pruebas del sistema.

3.3 Definiciones, Siglas y Abreviaciones

Definiciones propias del proyecto

- **Pedido:** Es el pedido que registra un mesero al momento de atender una mesa de su sector.

- **Estado de los pedidos:** Los pedidos tienen los siguientes estados asociados:

  - **Abierta:** Este estado se asigna automáticamente al momento de ingresar un pedido. Mientras este "Abierta", el mesero y barman pueden seguir agregando productos al pedido.
  - **Por pagar:** Cuando el cliente solicita pagar su cuenta, el pedido pasa al estado "Por pagar". Cuando ocurre esto, en el pedido ya no se pueden agregar más productos y se espera que el Cajero o Administrador cambie el estado del pedido a "Anulada" o "Pagada", para que la mesa donde fue creado el pedido vuelva a estar disponible.
  - **Pagada:** Cuando el cliente paga su cuenta, el cajero cambia el estado del pedido a "Pagada", dejando así disponible la mesa donde se generó el pedido.
  - **Anulada:** Se asigna el estado "Anulada" a un pedido, cuando el cliente no pudo finalizar el pago de su cuenta.

- **Comanda**: Las órdenes que tome el mesero, luego de ser procesadas se imprime las comandas las cuales son un listado con el producto y cantidad de este que solicitó una mesa perteneciente a un sector. Estas se envía al área de barra o cocina dependiendo del tipo de producto que se haya solicitado.

- **Ingredientes**: Nombre de todos los productos que se utilizan en el restorán para crear platos o tragos.

- **Perfil**: Los perfiles son equivalentes a los cargos que los usuarios tengan en el local y que deben tener interacción con el sistema. Estos son: Administrador, Mesero, Cajero y barman.

- **Producto Final:** Son todos los productos elaborados en el local.

- **Orden de Compra:** Las órdenes de compra contienen los datos principales de las facturas que reciben a la hora de comprar productos para renovar el stock.

Definiciones técnicas (del software)

- **Java**: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

- **SQL**: Es un Lenguaje Estructurado que sirve principalmente para hacer consultas a la base de datos y así obtener información de la misma.

- **HTML**: Sigla para definir "HyperText Markup Language" (lenguaje de marcado de hipertexto). Es el lenguaje de marcado predominante para la elaboración de páginas web.

- **JavaScript**: Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipiado y dinámico.

- **Jquery**: Es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

- **DAO**: Abreviación de "Data Acces Object". Componente de software que suministra una interfaz común entre el sistema y la base de datos.

- **EJB**: Abreviación de "Enterprise JavaBeans". Son APIs que forman parte del estándar de construcción de aplicaciones J2EE.

- **EAR**: Es un formato para empaquetar en un sólo archivo varios módulos. Permite desplegar varios de esos módulos en un servidor de aplicaciones. Contiene archivos XML llamados descriptores de despliegue que describen cómo realizar dicha operación.

- **sabcityWeb**: aplicación del sistema orientada a equipos (computadores y notebooks).

- **sabCityMobileWeb**: aplicación del sistema orientada a dispositivos móviles.

4 ESPECIFICACION DE REQUERIMIENTOS DE SOFTWARE

4.1.1 Alcances

- Este es el primer sistema que se ha de implementar en el pub-restaurant por lo tanto no interactúa con otros sistemas.
- Debido a que el cliente de momento no tiene proyectado realizar una modificación en la estructura del local, hay ciertos datos que no son necesarios controlar en el sistema. Por lo tanto los datos que se enumeran a continuación se ingresaran directamente a la base de datos:
    1. **Listado de áreas de trabajo:** Cocina, barra, área de atención.
    2. **Listado de sectores:** 4 sectores pertenecientes al área de atención y uno a la barra.
    3. **Cantidad de mesas:** El local actualmente cuenta con 45 mesas distribuidas en el segundo piso.
    4. **Perfiles o roles:** Administrador, cajero, mesero, barman.
    5. **Estado de los pedidos:** Abierta, por pagar, pagada, anulada.
    6. **Tipo de productos e ingredientes:** Comestible, cerveza, pisco, ron, vodka, whisky, gin.
- El cliente desea mantener un registro de todos los usuarios y proveedores, es decir, no eliminando esta información, manteniendo esta información mediante el manejo de estados.
- Las características de los dispositivos móviles que se utilicen deben estar acorde a las necesidades expuestas, los problemas de lentitud, conexiones a red, etc., no están dentro del alcance de este sistema.
- El sistema NO gestionará ni controlará la parte contable, pero sí entregará información relevante para ese tipo de procesos.
- El usuario no puede acceder en un mismo browser o navegador web a la aplicación sabcityWeb y sabcityMobileWeb, porque la autenticación es a nivel de servidor y esta genera una sola sesión http por browser servidor (htt://serdivor:puerto/).

4.2 Objetivo del software

Objetivos general del software.

El sistema gestionará información sobre los procesos de generación de pedidos, venta y productos, con la finalidad de mejorar la comunicación entre las áreas de barra, cocina y sector de mesas y mejorar la atención al cliente.

Objetivos Específicos del software

1. Mantener la información de los pedidos de los clientes para luego ser enviada a las áreas de barra, cocina o caja de forma más concreta.
2. Gestionar los pedidos de las mesas que pertenecen a un sector, para poseer un mejor control de las transacciones que se generan.

4.3 Descripción Global del Producto

4.3.1 Interfaz de usuario

La interfaz debe ser simple, intuitiva y clara para que un usuario con conocimientos básicos en computación sea capaz de manipularlo. Para esto se utilizarán botones y menús simples para la navegación.

Los colores deben ser acordes al actual logo del local:

- Cian, azul, violeta, amarillo.

4.3.2 Interfaz De Hardware

Gracias a las herramientas utilizadas en el desarrollo del sistema, este no requiere de alguna característica específica para los equipos y dispositivos móviles, solo deben ser smartphone.

El sistema necesita de una impresora la cual permita imprimir los pedidos generados por una mesera.

4.3.3 Interfaces de comunicación

- **HTTP**
- **Protocolo TCP/IP**

4.4 Requerimientos Específicos

4.4.1 Requerimientos Funcionales del sistema

Tabla nº2: Requerimientos acorde a los usuarios y los perfiles asignados a ellos.

Id	Nombre	Descripción
Req_gen_01	Inicio de sesión	El sistema permitirá el inicio de sesión mediante autenticación y autorización.
Req_gen_02	Ingreso de usuario	El sistema permitirá ingresar nuevos usuarios al sistema.
Req_gen_03	Modificar usuario	El sistema permitirá la modificación de datos del usuario. Estos son: Nombres, apellidos, estado (activo o inactivo), perfil asignado y contraseña.
Req_gen_04	Buscar usuario	El sistema permitirá buscar usuarios registrados en el sistema.
Req_gen_05	Buscar perfil	El sistema permitirá buscar perfiles en el sistema.
Req_gen_06	Asignar perfil	El sistema permitirá asignar un perfil a un usuario.

Tabla nº3: Presenta los requerimientos relacionado con los productos.

Id	Nombre	Descripción
Req_prod_01	Ingresar ingrediente	El sistema permite ingresar ingredientes asociados a los productos que ofrece el pub-restaurant
Req_prod_02	Modificar ingrediente	El sistema permite modificar ingredientes ingresados en el sistema.
Req_prod_03	Eliminación lógica ingrediente	El sistema permite la eliminación lógica de ingredientes del sistema.
Req_prod_04	Ingresar producto	El sistema permite ingresar productos al sistema.
Req_prod_05	Buscar producto	El sistema permite buscar productos en el sistema.
Req_prod_06	Modificar producto	El sistema permite modificar productos ingresados al sistema.
Req_prod_07	Eliminación lógica producto	El sistema permite la eliminación lógica de productos ingresados al sistema.

Tabla nº4: Presenta los requerimientos relacionado con las distintas áreas de trabajo del pub-restaurant, sectores y mesas.

Id	Nombre	Descripción
Req_asm_01	Buscar área de trabajo	El sistema permitirá buscar áreas de trabajo.
Req_asm_02	Buscar sector	El sistema permitirá buscar los sectores asociados a un área de trabajo
Req_asm_03	Buscar mesa	El sistema permitirá buscar mesas asociados a un sector.
Req_asm_11	Asignar mesa	El sistema permitirá asignar mesas a un usuario.

Tabla nº5: Presenta los requerimientos relacionado los pedidos de las mesas y la venta de productos.

Id	Nombre	Descripción
Req_ven_01	Ingresar pedido	El sistema permitirá ingresar los pedidos de las mesas de cada sector.
Req_ven_02	Modificar pedido	El sistema permitirá modificar los pedidos.
Req_ven_03	Anular pedido	El sistema permitirá cambiar estado a anular de un pedido.
Req_ven_04	Ingresar venta	El sistema permitirá ingresar venta basado en un pedido que esté por pagar.
Req_ven_05	Modificar venta	El sistema permitirá modificar productos ingresados al sistema.
Req_ven_06	Agregar descuento	El sistema permite modificar el precio total de la venta a través de descuentos. Estos solo los puede llevar a cabo el usuario que tenga permisos de "Administrador".

Tabla nº6: Presenta los requerimientos relacionado con los proveedores y las órdenes de compra.

Id	Nombre	Descripción
Req_prov_01	Ingresar proveedor	El sistema permitirá ingresar proveedores.
Req_prov_02	Modificar proveedor	El sistema permitirá modificar datos de un proveedor. Estos son: Nombre, Razón Social, Teléfonos, Dirección, estado (activo o inactivo).
Req_prov_03	Buscar Proveedor	El sistema permitirá buscar los proveedores ingresados en el sistema.
Req_prov_05	Ingresar orden de compra	El sistema permitirá ingresar órdenes de compra. Estas serán equivalentes a las facturas que reciben por comprar productos a los proveedores.
Req_prov_06	Buscar orden de compra	El sistema permitirá buscar las órdenes de compra.
Req_prov_06	Modificar orden de compra	El sistema permitirá modificar órdenes de compra ingresadas al sistema.
Req_prov_07	Eliminación lógica orden de compra	El sistema permitirá la eliminación lógica de órdenes de compra ingresados al sistema.

4.4.2 Interfaces externas de entrada

Cada interfaz de entrada indica todos los grupos de datos que serán ingresados al sistema independiente del medio de ingreso.

Tabla nº7: Interfaz externa de entrada.

Identificador	Nombre del ítem.	Detalle de Datos contenidos en ítem
Int_En_01	Datos del Producto o ingredientes	Nombre producto Stock producto Unidad de medida del stock Costo producto
Int_En_02	Datos del Proveedor	Rut Dígito verificador Nombre Razón Social Teléfonos proveedor Dirección proveedor
Int_En_03	Datos del Pedido	Fecha generación pedido Fecha término de pedido Precio Total Descuento Estado Cantidad de producto

Int_En_08	Datos del Usuario	Rut Dígito verificador Nombres Apellidos Contraseña
Int_En_09	Datos de Orden de Compra	Número orden de compra Fecha orden de compra Total sin IVA Total con IVA Cantidad producto Unidad medida Costo Unidad
Int_En_10	Datos Producto Final	Nombre Precio Descripción Cantidad ingrediente Unidad de medida de cantidad

4.4.3 Interfaces externas de Salida

Tabla nº8: Interfaz externa de salida.

Identificador	Nombre del ítem.	Detalle de Datos contenidos en ítem	Medio Salida
Int_Sal_01	Información proveedores	Rut Dígito verificador Nombre Razón social Teléfonos Dirección	Pantalla
Int_Sal_02	Información producto	Nombre Tipo del producto Contenido del producto Unidad de medida del contenido	Pantalla
Int_Sal_03	Informe orden de compra	Nombre proveedor Número de la orden Total sin IVA Total con IVA Fecha	Pantalla
Int_Sal_04	Informe pedido	Número mesa Estado Precio total de pedido Nombre producto Cantidad Precio unitario Precio total	Pantalla Impresora
Int_Sal_05	Información usuario	Rut Dígito verificador	Pantalla

		Nombres Apellidos Perfil asignado	
Int_Sal_06	Información sector	Nombre Mesas asignadas	Pantalla
Int_Sal_07	Información área	Nombre área Sectores asignados	Pantalla
Int_Sal_08	Informe resumen de venta general	Fecha Total de venta	Pantalla

4.4.4 Atributos del producto

- USABILIDAD- OPERABILIDAD: El sistema debe minimizar la cantidad de operaciones que un usuario debe realizar al momento de ejecutar una acción en el sistema. En cada operación en la que el usuario interactúe con el sistema se mostrarán mensajes que evidencien si la transacción fue satisfactoria o errónea.
- FUNCIONALIDAD-SEGURIDAD: El sistema debe permitir la autenticación mediante Rut del usuario y contraseña. El sistema debe mostrar funciones acorde a los permisos que tenga el usuario, basado en el perfil o rol que se le ha de asignar.
- PORTABILIDAD-ADAPTABILIDAD: El sistema debe soportar múltiples ambientes de equipos y dispositivos móviles.
- MANTENIBILIDAD-FACILIDAD DE CAMBIO: El proyecto puede ser utilizado en diferentes servidores de aplicaciones y basta con modificar ciertos archivos de configuración correspondiente a cada uno de estos para que el componente principal del proyecto funcione en cualquier servidor de aplicaciones.

5 FACTIBILIDAD

5.1 Factibilidad técnica.

Para el desarrollo del proyecto se cuenta con un alumno memorista de Ingeniería de Ejecución en Computación e Informática.

En reuniones con el cliente se determinó que se debe tener un computador para ser utilizado como servidor. Se requieren de tablets para el barman y otro para generar alguna orden en caso de ser necesario. A continuación las *Tablas Nº9 y Nº10* muestra las factibilidades técnicas para equipo y dispositivo mencionado.

Tabla Nº9: Factibilidad del Computador Servidor

Tipo	Computador de escritorio
Modelo	Gear® ONE-134i
Procesador	Intel Core i5-3570 3,4 GHz
Memoria RAM	4 Gb
Disco Duro	1 Tb
Accesorios	Mouse, Teclado y Monitor
Sistema Operativo	Windows 7 Home Basic 64-bit
Navegador	Mozilla Firefox 9.0.1 - Google Chrome 22.0.1229.2

Dado que se trabaja bajo el supuesto que no todos los trabajadores que interactúen con el sistema, tienen dispositivos móviles smartphone. Se necesitan 2 Tablets con este tipo de factibilidad como mínimo, uno para la barra y otro para emergencias en caso que alguna mesera no tenga un dispositivo móvil smartphone para que puedan ingresar los pedidos.

Tabla Nº10: Factibilidad Tablets para cajero.

Tipo	Tablet
Modelo	Gear® Feel 7011G
Chipset	Dual Core, 1.2GHz
Memoria RAM	1 Gb
Almacenamiento Interno	8 Gb

Para generar las comandas de los pedidos que van ingresando las meseras al sistema se requiere de una impresoras térmica.

Tabla Nº11: Factibilidad de Impresora térmica de la empresa

Tipo	Impresora Térmica EPSON
Modelo	EPSON TM-T20
Funciones	Impresión de pedidos
Compatibilidad	Windows 2000/XP/Vista/7,
Dimensiones de papel	Rollo 79, 5× 83, 0 / 57, 5 × 83.0 mm

5.2 Factibilidad operativa.

El hecho que el sistema se implemente en la empresa se considera como un aspecto positivo e innovador para la misma, ya que el control de ventas, generación de pedidos, el histórico y los reportes que se generan de ellas se controlan mejor, dado que la información relacionada con estos procesos será de fácil acceso para los usuarios del sistema.

El sistema será utilizado por usuarios con conocimientos básicos en navegación y el control del sistema de acuerdo al conocimiento del negocio asociado a su cargo.

Como todo sistema informático, este depende del suministro eléctrico y de una pequeña red local (por el uso de varios dispositivos móviles para la venta diaria).

5.3 Factibilidad económica.

La siguiente tabla detalla los costos de hardware que se requiere para el funcionamiento del sistema.

Tabla Nº12: Factibilidad Económica (hardware utilizado)

Enlace de referencia: http://www.pcfactory.cl

Articulo	**Función**	**Cant.**	**Precio de referencia**
Computador de escritorio modelo ONE-134i	Servidor del sistema	1	$319.990
Tablet modelo Feel 7011G	Ingreso de pedidos.	2	$59.990
Impresora Térmica EPSON modelo TM-T20	Impresión de comandas Y resumen de pedido	1	$159.990
Rollo de cable UTP 305 Mt.	Fabricar la red local	1	$89.990
Pinza encriptadora	Para realizar conexiones de cables UTP con conectores RJ-45	1	$5.990
Conectores RJ-45	Unión de cables UTP	50	$180
		Total	**$636.130**

Tabla Nº13: Costos asociados al proyecto

Costo	**Valor**
Recursos Humanos • Alumno Memorista	$0
Movilización y Pasajes • Traslado Santiago- Coronel: 2 días al mes por 4 meses.	$80.000
Costo de Producción • Energía Eléctrica por 4 meses	$40000
Total	**$120.000**

Monto Total: $756.130

5.4 Conclusión de la factibilidad

Luego de realizar los tres estudios de análisis de factibilidad, en los cuales se incluyeron aspectos, operativos y económicos, se puede concluir lo siguiente.

- Del punto de vista **técnico**, se cuenta con el recurso humano necesario para la ejecución de las tareas que involucran este proyecto. Es necesario adquirir insumos hardware para la implementación del sistema en el local.
- Del punto de vista **operativo**, se determinó que la implementación del sistema es provechosa para el local dado que el control de los procesos que el cliente considera críticos, son apoyados por este.
- Del punto de vista **económico**, se consideraron los costos del proyecto en los cuales se contemplaron aspectos de recursos humanos, movilización y costos de producción y los costos de hardware necesarios para implementación del sistema. Se considera un ahorro importante el que el alumno memorista no perciba ingreso. El cliente está de acuerdo en que el monto total del proyecto es un gasto asumible.

De todo lo anteriormente expuesto, se concluye que basándose en aspectos técnicos, operativos y económicos, es totalmente factible la realización de este proyecto.

6 ANALISIS

6.1 Diagrama de casos de uso

6.1.1 Actores

Administrador de Sistema:

Se refiere a la persona que se encargará de que el sistema funcione de forma correcta.

- El administrador dentro de la empresa tiene el papel de gerente y dueño de la misma, además se encarga de gestionar la información más relevante del sistema, como los perfiles de usuarios y el ingreso de nuevos usuarios.
- Debe poseer conocimientos básicos de computación y navegación de páginas web.
- Este actor posee un nivel de usuario elevado ya que tiene opciones de ingresar, modificar, y eliminar a los demás usuarios del sistema, así como asignar perfiles y permisos entre otras funcionalidades.

Mesero:

Se refiere al actor que interactúa con el sistema de forma habitual, que se encarga de atender a los clientes y generar los pedidos.

- El mesero controla las funciones relacionadas con los pedidos de los clientes.
- Debe poseer conocimientos básicos de computación y navegación de páginas web.
- El privilegio del mesero se basa en el control de los pedidos, es decir, en el ingreso y modificación de los pedidos antes de que cada uno de ellos sean enviados a "Por pagar" estado que solo el actor Cajero y administrador tiene privilegios.

Barman:

Se refiere al actor que interactúa con el sistema de forma habitual, que se encarga de atender a los clientes en la barra.

- El barman controla las funciones relacionadas con los pedidos de los clientes.
- Debe poseer conocimientos básicos de computación y navegación de páginas web.
- El privilegio del barman se basa en el ingreso y modificación de los pedidos antes de ser registrados en caja.

Cajero:

Se refiere a la persona que maneja el control de ventas del sistema.

- El cajero se encarga de la venta de productos relacionados con las órdenes tomadas por el mesero y barman.
- El nivel de conocimiento tiene que ser básico.
- El privilegio del cajero se basa en el control de las ventas.

6.1.1.1 Casos de Uso y descripción

El diagrama de caso de uso, para un mejor orden, se divide por diagrama según su función (iniciar sesión, control de producto, control de usuario, control perfil, control producto, control área), tal como se muestra a continuación.

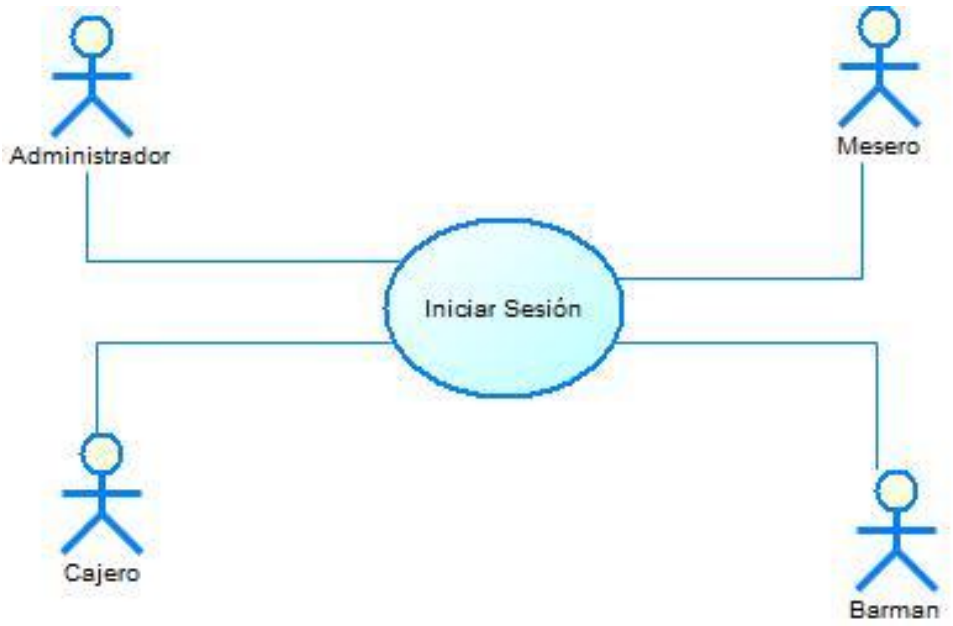


Ilustración 3: Caso de uso "Iniciar Sesión"

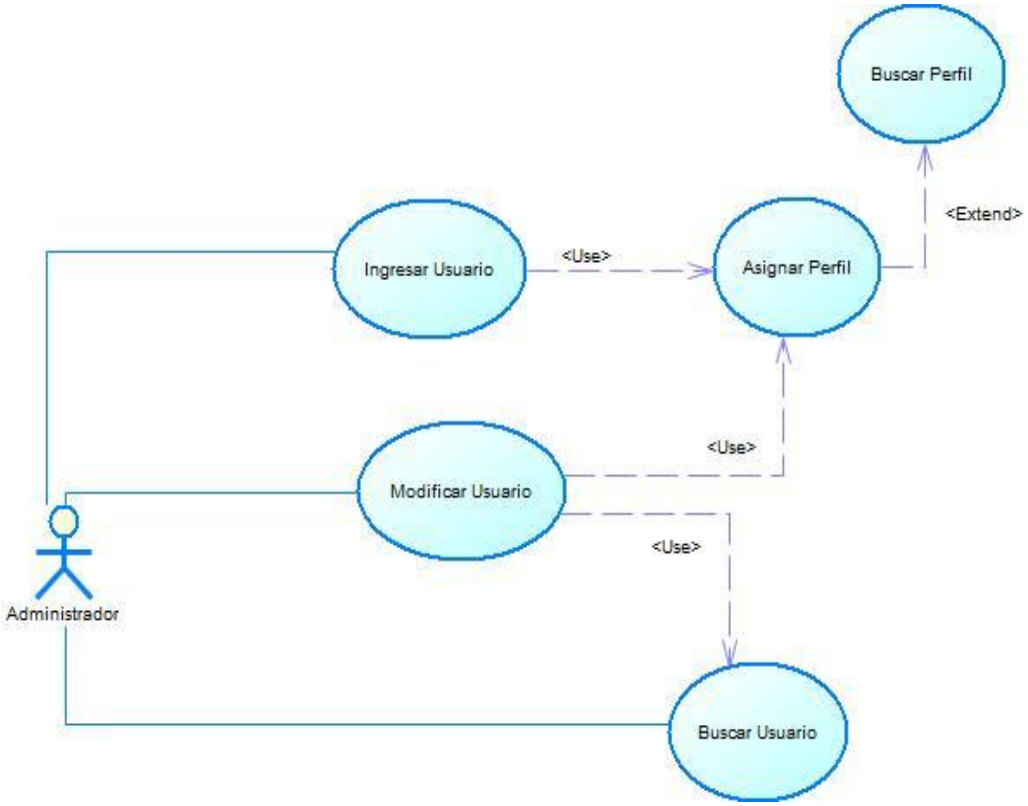


Ilustración 4: Caso de uso "Control Usuario"

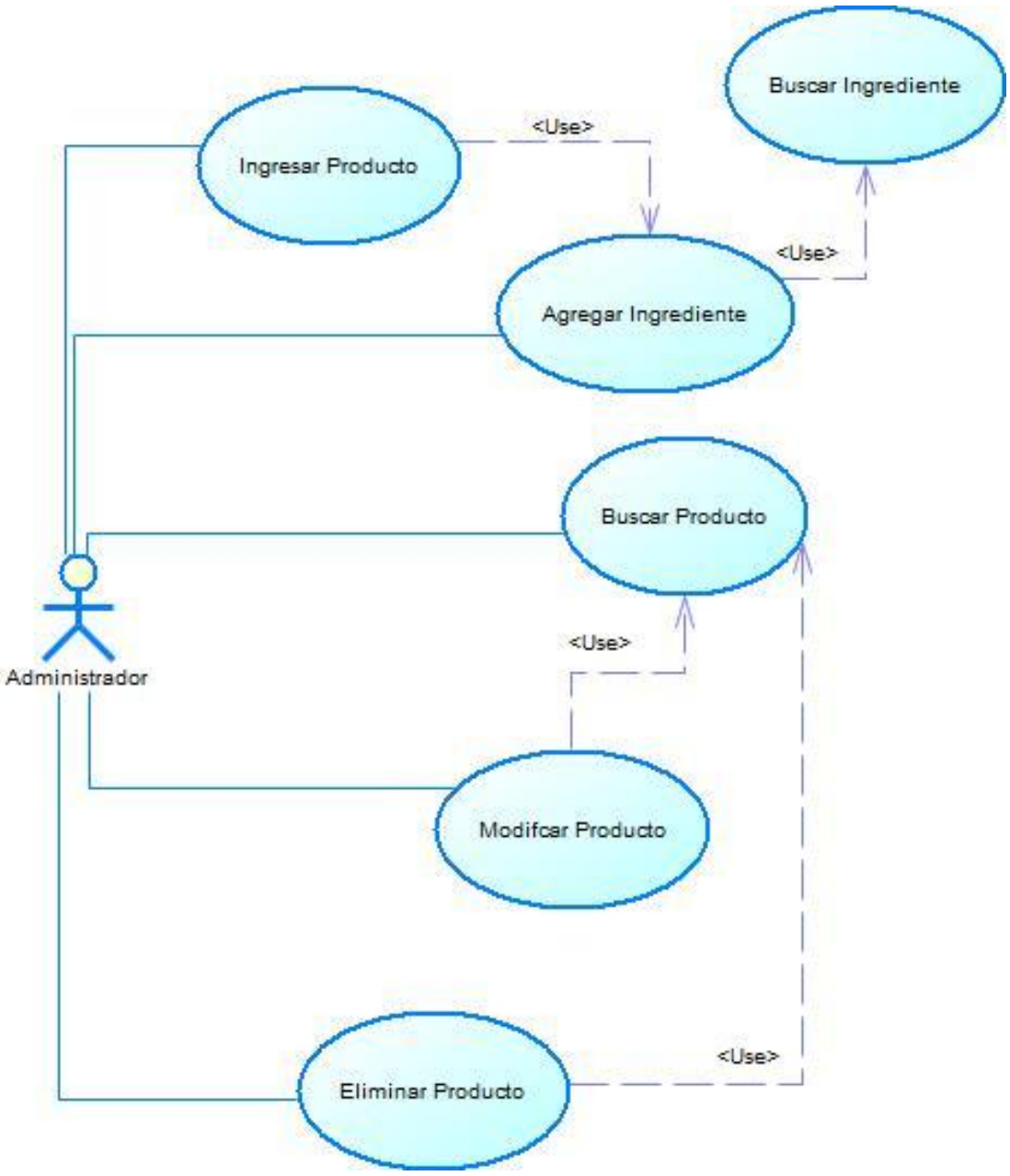


Ilustración 5: Caso de uso "Control Producto"

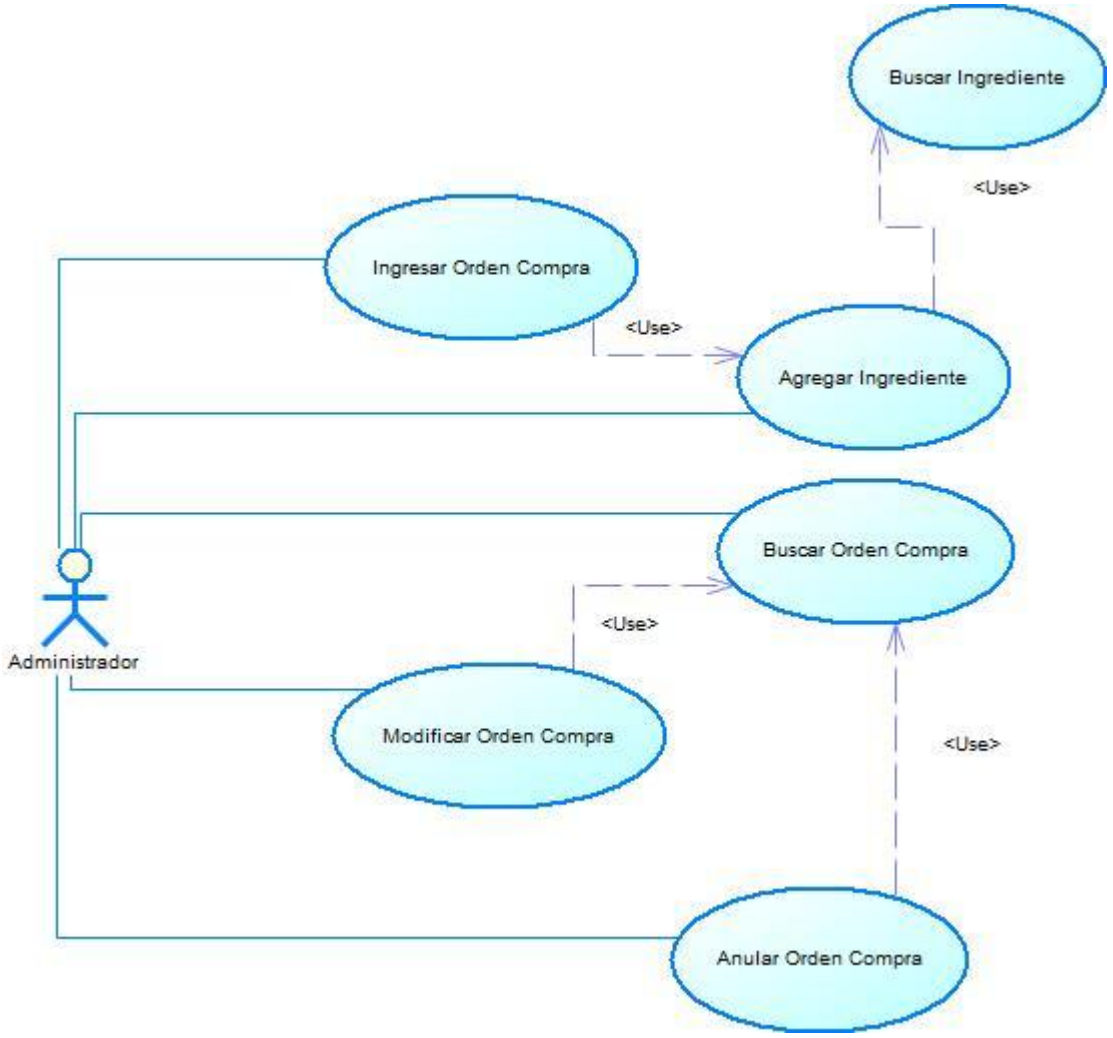


Ilustración 6: Caso de uso "Control Orden Compra"

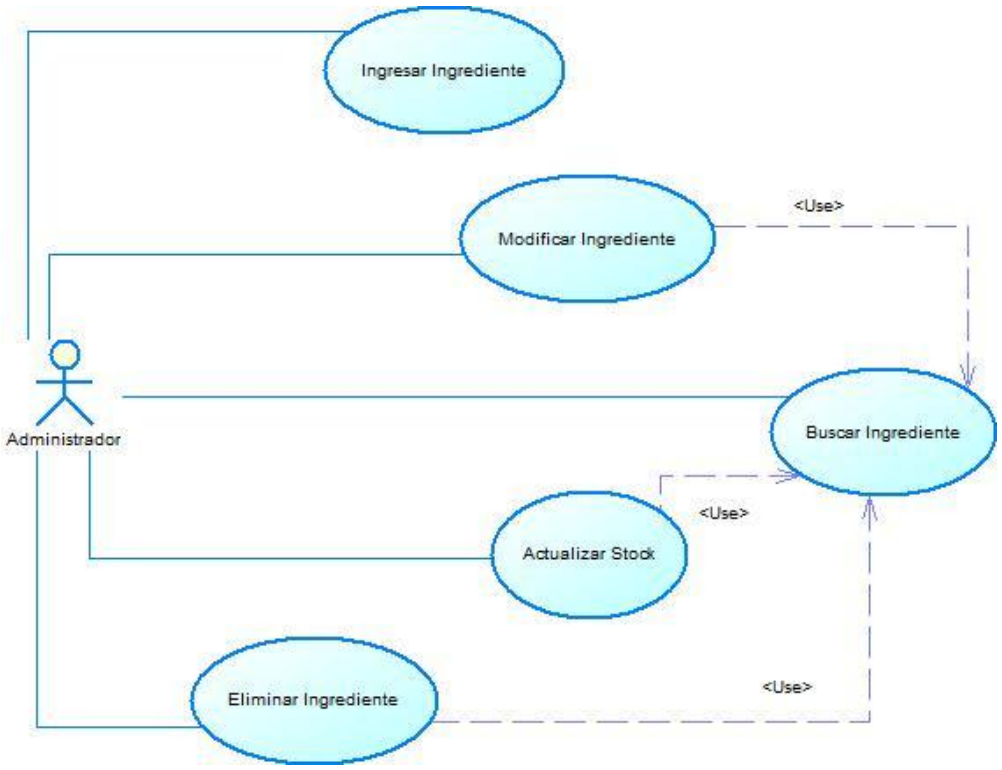


Ilustración 7: Caso de uso "Control Ingrediente"

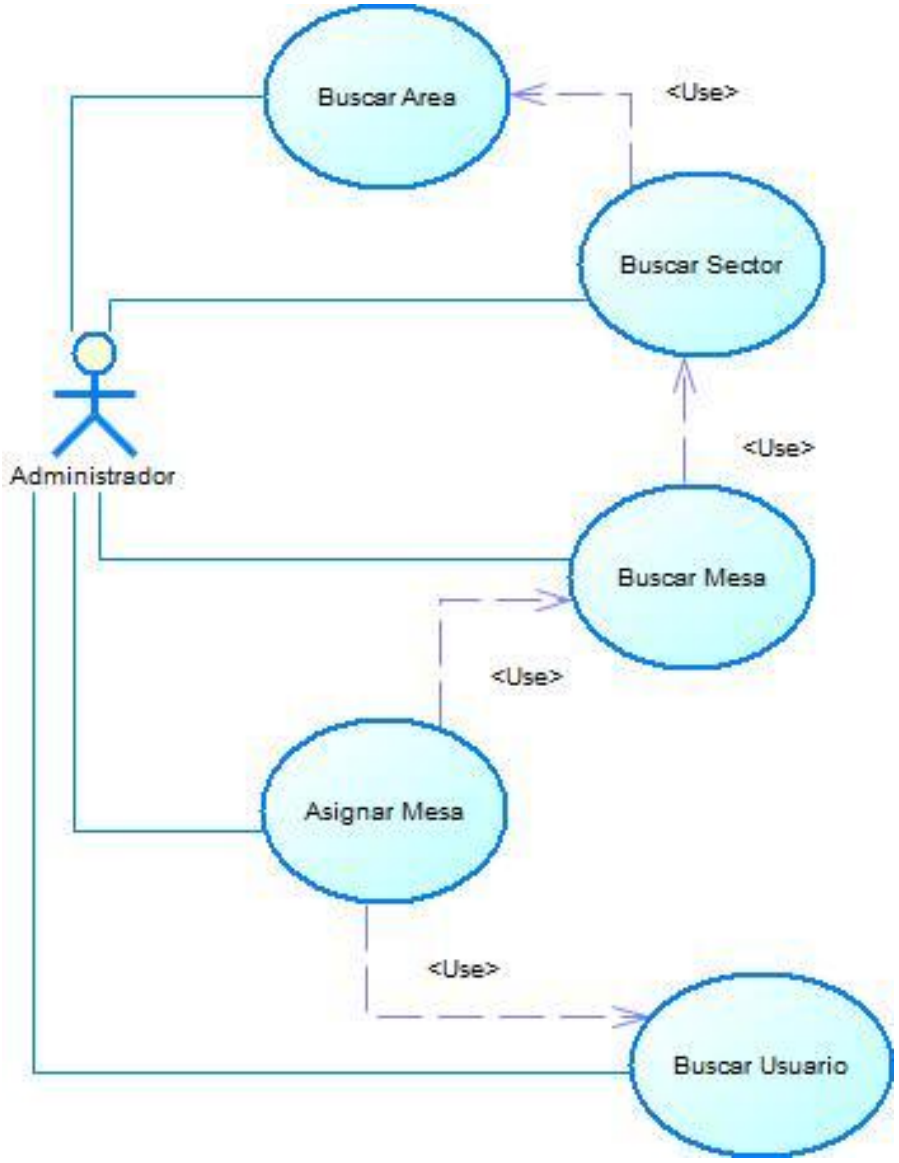


Ilustración 8: Caso de uso "Control Área Sector"

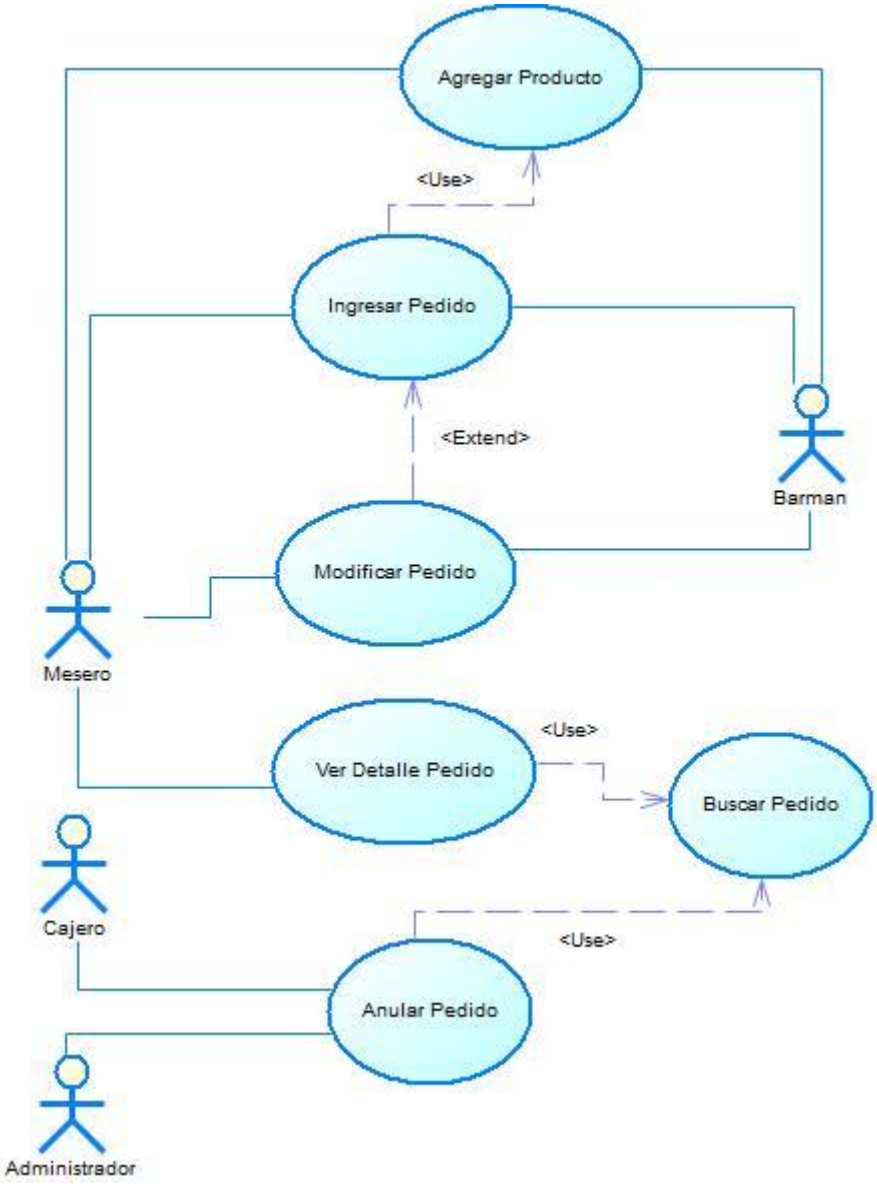


Ilustración 9: Caso de uso "Control Pedido"

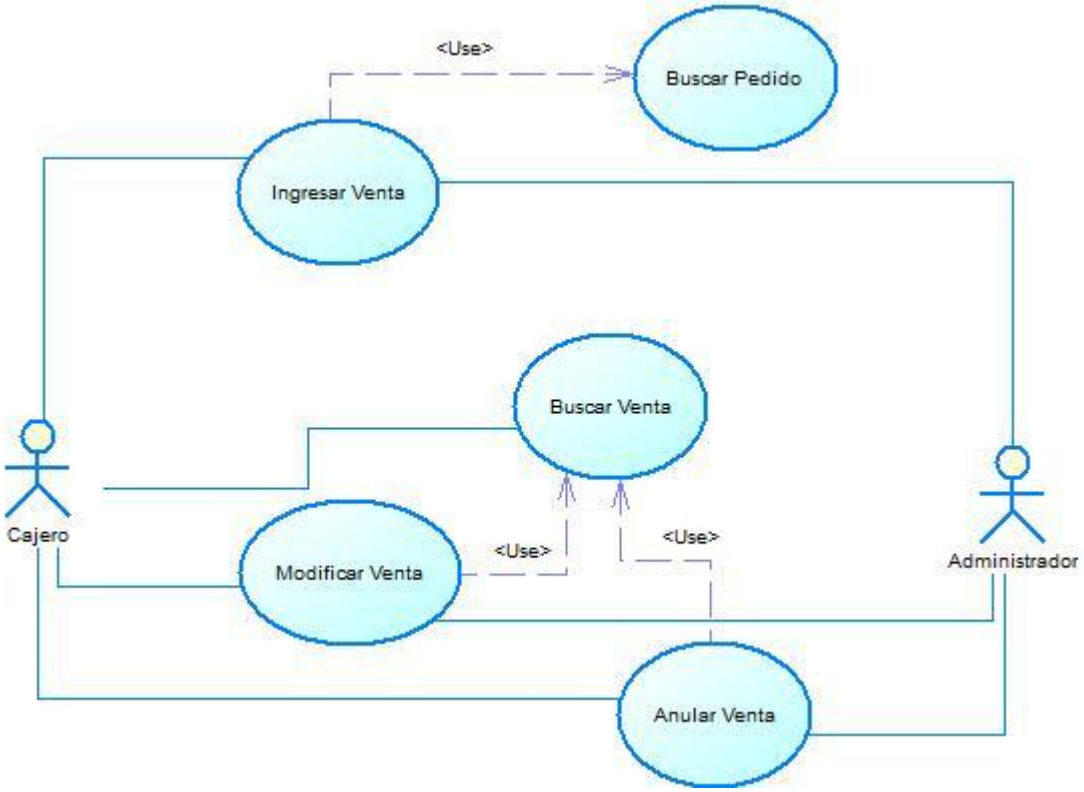


Ilustración 10: Caso de uso "Control Venta"

6.1.2 Especificación de los Casos de Uso

6.1.2.1 Caso de Uso: <Iniciar Sesión>

- Descripción: Se ingresa al sistema mediante un nombre de usuario y contraseña.
- Pre-Condiciones:

Existe al menos un usuario ingresado en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El usuario ingresa su Rut y contraseña y presiona el botón "Ingresar"	2 El sistema comprueba si los datos del usuario son válidos y que sus permisos asociados a su perfil o cargo, sean suficientes.

- Flujo de Eventos Alternativo: Se describe cada uno de los flujos alternativos que el caso de uso puede tener.

Al actor	El sistema
	2.1 Si los datos del usuario no pertenecen al sistema, este alertará que estos no son correctos y seguirá mostrando el login.
	2.2 Si el usuario no tiene los permisos suficientes, el sistema alertará esto y seguirá mostrando el login.

- Post-Condiciones: El usuario ingresa satisfactoriamente al sistema

6.1.2.2 Caso de Uso: <Ingresar Usuario>

- Descripción: El administrador ingresa un nuevo usuario a la base de datos para que pueda ingresar al sistema y hacer uso de él.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Debe haber al menos un perfil registrado en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 Se despliega un formulario para la creación de un nuevo usuario.
3 El administrador escribe los datos	

del nuevo usuario y presiona "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
	4 Si el nombre de usuario ya existe en el sistema, este alertará el error y seguirá mostrando el formulario.
3.1 El usuario deja uno o más campos en blanco y presiona "Guardar"	4.1 El sistema notificará el error y seguirá mostrando el formulario.
3.2 El usuario ingresa un Rut inválido.	4.2 El sistema notificará el error y seguirá mostrando el formulario

- Post-Condiciones: Se ingresa la información a la base de datos exitosamente.

6.1.2.3 Caso de Uso: <Modificar Usuario>

- Descripción: El administrador modifica los datos de un usuario del sistema a excepción del Rut.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Los datos del usuario a modificar deben estar previamente almacenados en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El administrador elige la opción "Buscar usuario"	2 El sistema despliega un listado con los usuarios del sistema
3 El administrador selecciona el usuario que desea modificar	4 EL sistema despliega opciones para modificar los datos del usuario seleccionado
5 El administrador ingresa el o los nuevos datos para ese usuario	

- Flujo de Eventos Alternativo:

Al actor	El sistema
	6.1 Si los datos del usuario seleccionado existen en el sistema, se notifica al administrador y se prohíbe

	la modificación

- Post-Condiciones: Se modifica satisfactoriamente los nuevos datos del usuario.

6.1.2.4 Caso de Uso: <Asignar Perfil>

- Descripción: Este caso de uso describe el proceso de asignar un perfil a un usuario.
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador.
Debe existir al menos un perfil en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El administrador elige la opción "Buscar usuario"	2 El sistema despliega un listado con los usuarios del sistema
3 El administrador selecciona el usuario que desea asignar perfil	4 El sistema despliega el listado de perfiles ingresados en el sistema
5 El administrador el administrador escoge un perfil para asignar al usuario	

- Flujo de Eventos Alternativo:

Al actor	El sistema
5.1 El administrador no escoge un perfil para asignar	6.1 El sistema notifica que debe escoger uno y sigue mostrando el listado

- Post-Condiciones: Se modifica satisfactoriamente los datos del usuario.

6.1.2.5 Caso de Uso: <Buscar Perfil>

- Descripción: El administrador consulta al sistema por los perfiles ingresados en el sistema.
- Pre-Condiciones:

Debe existir al menos un perfil creado.

- Flujo de Eventos Básicos:

Al actor	El sistema
	2 El sistema despliega un listado con los perfiles

- Post-Condiciones: El sistema muestra la información exitosamente.

6.1.2.6 Caso de Uso: <Buscar área de trabajo>

- Descripción: El administrador consulta sobre las áreas de trabajo registradas en el sistema.
- Pre-Condiciones:
  Debe existir al menos un área de trabajo registrada en el sistema
- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un listado con las áreas de trabajo registradas en el sistema.

- Post-Condiciones El sistema despliega la información exitosamente.

6.1.2.7 Caso de Uso: <Buscar sector>

- Descripción:

El administrador consulta sobre los sectores registrados en el sistema que estén asignados a un área de trabajo.

- Pre-Condiciones:

Debe existir al menos un sector ingresados en el sistema.

El sector debe estar asociado a un área de trabajo.

- Flujo de Eventos Básicos:

Al actor	El sistema
	2 El sistema despliega el listado de sectores registrados en el sistema.

- Flujo de Eventos Alternativo:

Al actor	El sistema
	2.1 El sistema no detecta sectores ingresados y alerta sobre esto.

- Post-Condiciones: El sistema devuelve la información solicitada exitosamente.

6.1.2.8 Caso de Uso: <Buscar Mesa>

- Descripción:

El administrador consulta sobre el listado de mesas que estén asignados a un sector.

- Pre-Condiciones:

Debe existir al menos una mesa ingresada en el sistema.

La mesa debe estar asignada a un sector.

- Flujo de Eventos Básicos:

Al actor	El sistema
	2 El sistema despliega el listado de mesas asignadas a un sector.

- Flujo de Eventos Alternativo:

Al actor	El sistema
	2.1 El sistema no detecta mesas registradas y alerta sobre esto.

- Post-Condiciones: El sistema devuelve la información solicitada exitosamente.

6.1.2.9 Caso de Uso: <Asignar Mesa>

- Descripción: Este caso de uso describe el proceso de asignar una mesa a un usuario
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador.

Debe existir al menos un sector asociado a un área.

Debe existir al menos una mesa en el sistema.

Las mesas deben estar asociadas a un sector.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un listado con las mesas registradas en el sistema
3 El usuario escoge un conjunto de mesas y presiona el botón “Guardar”.	
	4 El sistema guarda la asignación.

- Flujo de Eventos Alternativo:

Al actor	El sistema
3.1 El usuario no escoge mesas y presiona el botón "Guardar".	4.1 El sistema alerta que debe elegir al menos una mesa para poder guardar la información.

- Post-Condiciones: Asignación creada en el sistema exitosamente.

6.1.2.10 Caso de Uso: <Ingresar Ingrediente>

- Descripción: Este caso de uso describe el ingreso de ingredientes en el sistema.
- Pre-Condiciones:

El usuario debe estar autentificado como Administrador.

- Flujo de Eventos Básicos:

Al actor	El sistema
	2 El sistema despliega un formulario para ingresar un nuevo ingrediente
3 El administrador completa el formulario y presiona el botón "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
3.1 El administrador no completa el formulario y presiona "Guardar"	4.1 El sistema alerta que faltan datos por escribir y continúa mostrando el formulario.

- Post-Condiciones: El sistema guarda correctamente la información.

6.1.2.11 Caso de Uso: <Buscar Ingrediente>

- Descripción: Este caso de uso describe la búsqueda de ingredientes en el sistema.
- Pre-Condiciones:

El usuario debe estar autentificado como Administrador.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema devuelve datos del ingrediente a buscar.

- Flujo de Eventos Alternativo:

Al actor	El sistema
	1.1 El sistema no encuentra datos asociados a la búsqueda.

- Post-Condiciones: El sistema devuelve datos asociados a la búsqueda exitosamente.

6.1.2.12 Caso de Uso: <Modificar Ingrediente>

- Descripción: Este caso de uso describe el ingreso de ingredientes en el sistema.
- Pre-Condiciones:

El usuario debe estar autentificado como Administrador.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema devuelve en un formulario, los datos del ingrediente que el usuario desea modificar
2 El usuario modifica los datos que necesite y presiona el botón "Modificar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
3.1 El usuario deja un campo vacío y presiona el botón "Modificar"	4.1 El sistema alerta que faltan datos por escribir y continúa mostrando el formulario.

- Post-Condiciones: El sistema elimina exitosamente el ingrediente.

6.1.2.13 Caso de Uso: <Eliminar Ingrediente>

- Descripción: Este caso de uso describe la eliminación lógica de ingredientes en el sistema.
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador.

Deben existir ingredientes en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El usuario escoge el ingrediente que desea eliminar.	2 El sistema pregunta si está seguro de la elección.

3 El usuario verifica su opción y procede a eliminar.	

- Flujo de Eventos Alternativo:

Al actor	El sistema
3.1 El usuario decide no eliminar el ingrediente.	4.1 El sistema muestra en pantalla los datos de los ingredientes y espera una nueva acción.

- Post-Condiciones: El sistema elimina exitosamente la información.

6.1.2.14 Caso de Uso: <Ingresar Producto>

- Descripción: El administrador ingresa un producto al sistema.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
	1 El sistema despliega un formulario para ingresar un nuevo producto
2 El administrador completa el formulario y presiona el botón "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador no completa el formulario y presiona "Guardar"	3 El sistema avisa que faltan datos por escribir y continua mostrando el formulario

- Post-Condiciones: El sistema guarda correctamente la información.

6.1.2.15 Caso de Uso: <Agregar Ingrediente>

- Descripción: El administrador agrega un ingrediente al producto.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Debe existir al menos un ingrediente en el sistema

Debe estar en el módulo Ingresar Producto

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
	1 El sistema despliega un formulario para agregar un ingrediente al producto
2 El administrador completa el formulario y presiona el botón "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador no completa el formulario y presiona "Guardar"	4 El sistema avisa que faltan datos por escribir y continua mostrando el formulario

- Post-Condiciones: El sistema guarda correctamente la información.

6.1.2.16 Caso de Uso: <Modificar Producto>

- Descripción: El administrador modifica un producto almacenado en el sistema.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Debe haber al menos un producto almacenado en el sistema.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
1 El administrador elige la opción "Buscar Producto"	

- Post-Condiciones: El sistema despliega un listado con el nombre de los productos.

6.1.2.17 Caso de Uso: <Eliminar Producto>

- Descripción: El administrador realiza la eliminación lógica de un producto del sistema.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Debe haber al menos un producto almacenado en el sistema.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
1 El administrador elige la opción "Buscar Producto"	2 El sistema despliega un listado con los nombres de los productos.
3 El administrador elige un producto y presiona "Eliminar"	4 El sistema pregunta si está seguro de eliminar el producto
5 El administrador presiona el botón "Si"	

- Flujo de Eventos Alternativo: Se describe cada uno de los flujos alternativos que el caso de uso puede tener.

Al actor	El sistema
5.1 El administrador presiona el botón "No"	6.1 El sistema no hace nada y sigue mostrando la lista

- Post-Condiciones: El producto es eliminado del sistema.

6.1.2.18 Caso de Uso: <Buscar Producto>

- Descripción: El administrador busca un producto en el sistema.
- Pre-Condiciones:

El administrador debe ingresar al sistema.

Debe haber un producto almacenado en el sistema.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
1 El administrador elige la opción "Buscar Producto"	

- Post-Condiciones: El sistema despliega una lista con el nombre de los productos

6.1.2.19 Caso de Uso: <Ingresa Orden Compra>

- Descripción: El administrador ingresa una nueva orden de compra el sistema
- Pre-Condiciones:

Debe existir al menos un proveedor en el sistema.

Debe existir al menos un producto en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un formulario para ingresar una nueva orden de compra
2 El administrador completa el formulario y presiona el botón "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador no completa todos los campos y presiona el botón "Guardar"	3 El sistema alerta aquello y sigue mostrando el formulario.

- Post-Condiciones: La orden de compra es registrada en el sistema exitosamente.

6.1.2.20 Caso de Uso: <Agregar Ingrediente>

- Descripción: El administrador agrega un ingrediente a la orden de Compra
- Pre-Condiciones:

Debe existir al menos un ingrediente en el sistema.

Debe estar en el módulo Ingresar Orden de Compra

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un formulario para agregar un ingrediente a la Orden

	de Compra
2 El administrador completa el formulario y presiona el botón "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador no completa todos los campos y presiona el botón "Guardar"	3 El sistema alerta aquello y sigue mostrando el formulario.

- Post-Condiciones: El ingrediente es agregado en la Orden de Compra exitosamente.

6.1.2.21 Caso de Uso: <Buscar Orden Compra>

- Descripción: Este caso de uso describe la búsqueda de órdenes de compra registradas en el sistema
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador

Debe haber al menos una orden de compra registrada en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un listado con las órdenes de compra

- Post-Condiciones: La búsqueda de órdenes de compra se realiza exitosamente.

6.1.2.22 Caso de Uso: <Modificar Orden Compra>

- Descripción: El administrador modifica una orden de compra
- Pre-Condiciones:

Debe haber al menos una orden de compra registrada en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega un formulario para modificar la orden de compra
2 El administrador modifica los	

campos necesarios y presiona "Guardar"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador no completa todos los campos y presiona el botón "Guardar"	3 El sistema alerta aquello y sigue mostrando el formulario.

- Post-Condiciones: La orden de compra es modificada en el sistema exitosamente.

6.1.2.23 Caso de Uso: <Anular Orden Compra>

- Descripción: Este caso de uso describe la eliminación lógica de una orden de compra en el sistema.
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador.

Debe haber al menos una orden de compra registrada en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema pregunta si realmente desea eliminar la orden de compra
2 El administrador presiona el botón "Sí"	

- Flujo de Eventos Alternativo:

Al actor	El sistema
2.1 El administrador presiona el botón "No"	3 El sistema vuelve a mostrar el listado de órdenes de compra

- Post-Condiciones: La orden de compra es eliminada exitosamente.

6.1.2.24 Caso de Uso: <Ingresar Pedido>

- Descripción: El usuario ingresa un pedido de una mesa.
- Pre-Condiciones:

El usuario debe tener asignado el perfil de mesero o barman.

Debe haber al menos un sector.

El mesero debe estar asociado a un sector.

Debe haber al menos una mesa en el sector.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
1 El mesero elige la opción "Ingresar Pedido"	2 El sistema despliega las opciones para ingresar un nuevo pedido
3 El mesero digita los datos para ingresar el nuevo pedido y presiona "Guardar"	

- Flujo de Eventos Alternativo: Se describe cada uno de los flujos alternativos que el caso de uso puede tener.

Al actor	El sistema
3.1 El mesero no ingresa todos los datos necesarios para ingresar un nuevo pedido	4.1 El sistema avisa la falta de datos y sigue mostrando las opciones para ingresar un nuevo pedido

- Post-Condiciones: El pedido se almacena correctamente en el sistema

6.1.2.25 Caso de Uso: <Ver detalle Pedido>

- Descripción: El mesero ve el detalle de un pedido
- Pre-Condiciones:

El mesero debe ingresar al sistema.

El mesero debe estar asociado a un sector.

Debe haber al menos un pedido.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega los datos de un pedido

- Post-Condiciones: El sistema despliega el detalle del pedido exitosamente

6.1.2.26 Caso de Uso: <Modificar Pedido>

- Descripción: El usuario modifica un pedido.
- Pre-Condiciones:

El usuario debe tener asignado el perfil de mesero o barman.
Solo se modifica los datos que estén en sesión

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega los pedidos de las mesas asociadas al sector el cual atiende el mesero
3 El mesero elige el pedido a modificar	4 El sistema despliega las opciones para poder modificar el pedido
5 El mesero modifica el pedido y presiona "Guardar"	

- Post-Condiciones: El pedido modificado es guardado en el sistema.

6.1.2.27 Caso de Uso: <Anular Pedido>

- Descripción: El usuario cambia el estado del pedido a "anular".
- Pre-Condiciones:

El usuario debe estar autenticado como Administrador o cajero.
Debe haber un pedido con el estado "pendiente" o "por pagar".

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema pregunta si está seguro de la acción que eligió el usuario
2 El usuario presiona el botón "Sí"	

- Flujo de Eventos Alternativo: Se describe cada uno de los flujos alternativos que el caso de uso puede tener.

Al actor	El sistema
2.1 El mesero presiona el botón "No"	3.1 el sistema no cambia el estado del pedido a Anular

- Post-Condiciones: El estado del pedido actualiza exitosamente.

6.1.2.28 Caso de Uso: <Registrar Venta>

- Descripción: El cajero registra una venta basada en un pedido que esté por pagar.
- Pre-Condiciones:

Debe existir una solicitud de registro de venta al momento de cerrar una orden.

- Flujo de Eventos Básicos: Descripción de la secuencia de acciones del caso de uso para clarificar en lenguaje natural lo que el sistema hace cuando el caso de uso es comenzado por un actor y cómo el sistema interactúa con los actores

Al actor	El sistema
	1 El sistema muestra la orden asociada a la venta más el precio de los productos y el precio total de la orden.
2 El cajero presiona el botón "Registrar"	

- Flujo de Eventos Alternativo: Se describe cada uno de los flujos alternativos que el caso de uso puede tener.

Al actor	El sistema
	3 El sistema no encuentra ninguna solicitud de registro de venta y despliega un mensaje de alerta.

- Post-Condiciones: El sistema registra de forma exitosa la venta y procede a cerrar la orden.

6.1.2.29 Caso de Uso: <Buscar Venta>

- Descripción: El cajero busca una venta
- Pre-Condiciones:

El cajero debe ingresar al sistema.

Debe haber al menos una venta registrada en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El cajero elije la opción "Buscar Venta"	

- Post-Condiciones: El sistema despliega una lista con las ventas realizadas en el día.
-

6.1.2.30 Caso de Uso: <Modificar Venta>

- Descripción: El cajero modifica una venta.
- Pre-Condiciones:

Debe haber al menos una venta registrada en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
	1 El sistema despliega una lista con las ventas realizadas durante el día
2 El cajero elige una venta y presiona "Modificar"	3 El sistema despliega las opciones necesarias para poder modificar la venta
4 El cajero modifica la venta y presiona "Guardar"	

- Post-Condiciones: El sistema modifica la venta de forma exitosa.

6.1.2.31 Caso de Uso: <Generar Pedido>

- Descripción: El mesero o barman genera el pedido para que este pase al estado "por pagar".
- Pre-Condiciones:

Debe haber al menos un pedido pendiente en el sistema.

- Flujo de Eventos Básicos:

Al actor	El sistema
1 El mesero elige la opción "Generar Comanda"	

- Post-Condiciones: El sistema genera una comanda con los datos de la orden ingresada por el mesero.

6.2 Modelamiento de datos

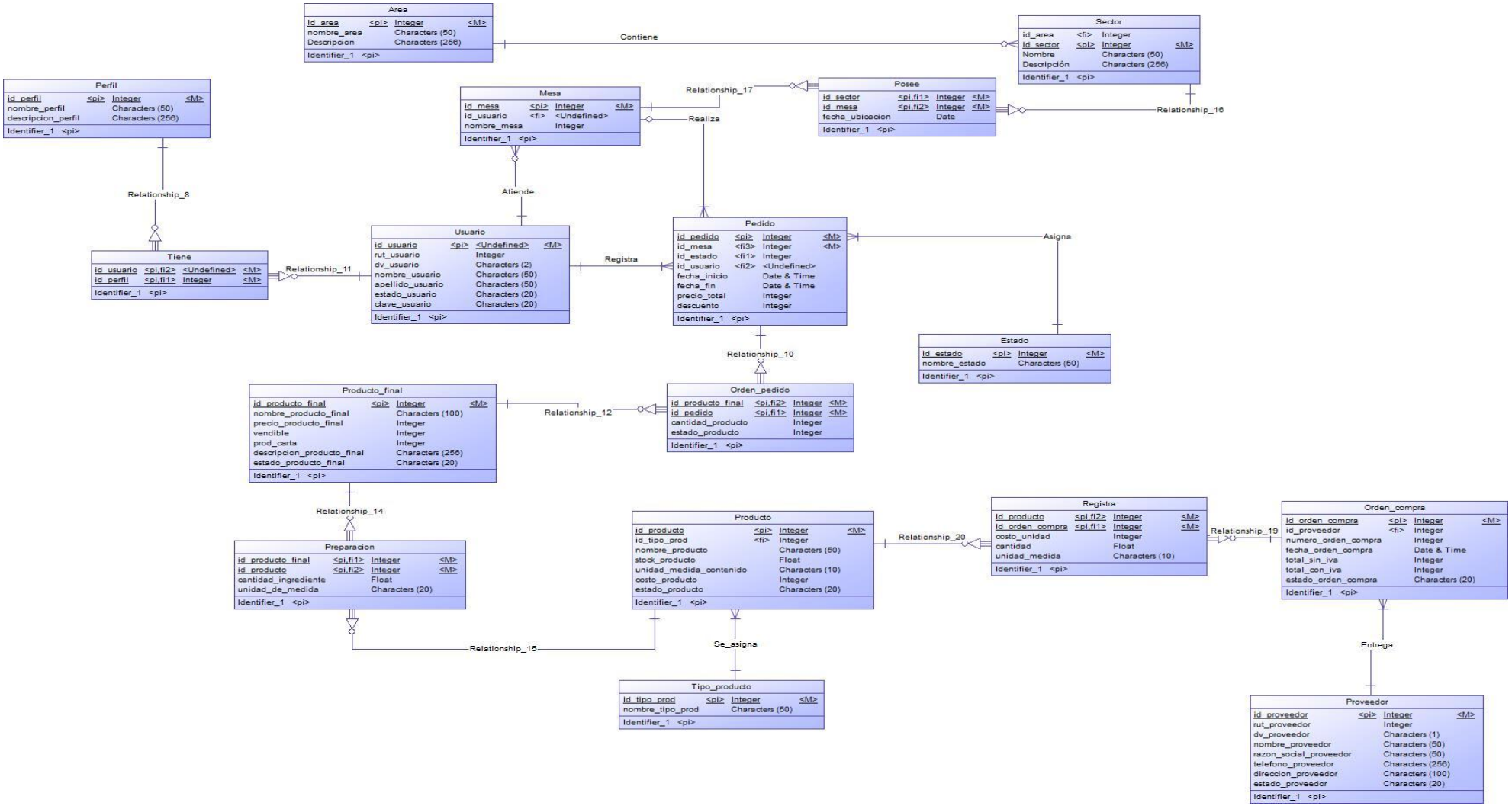


Ilustración 11: Modelo lógico de la base de datos

7 DISEÑO

7.1 Diseño de Físico de la Base de datos

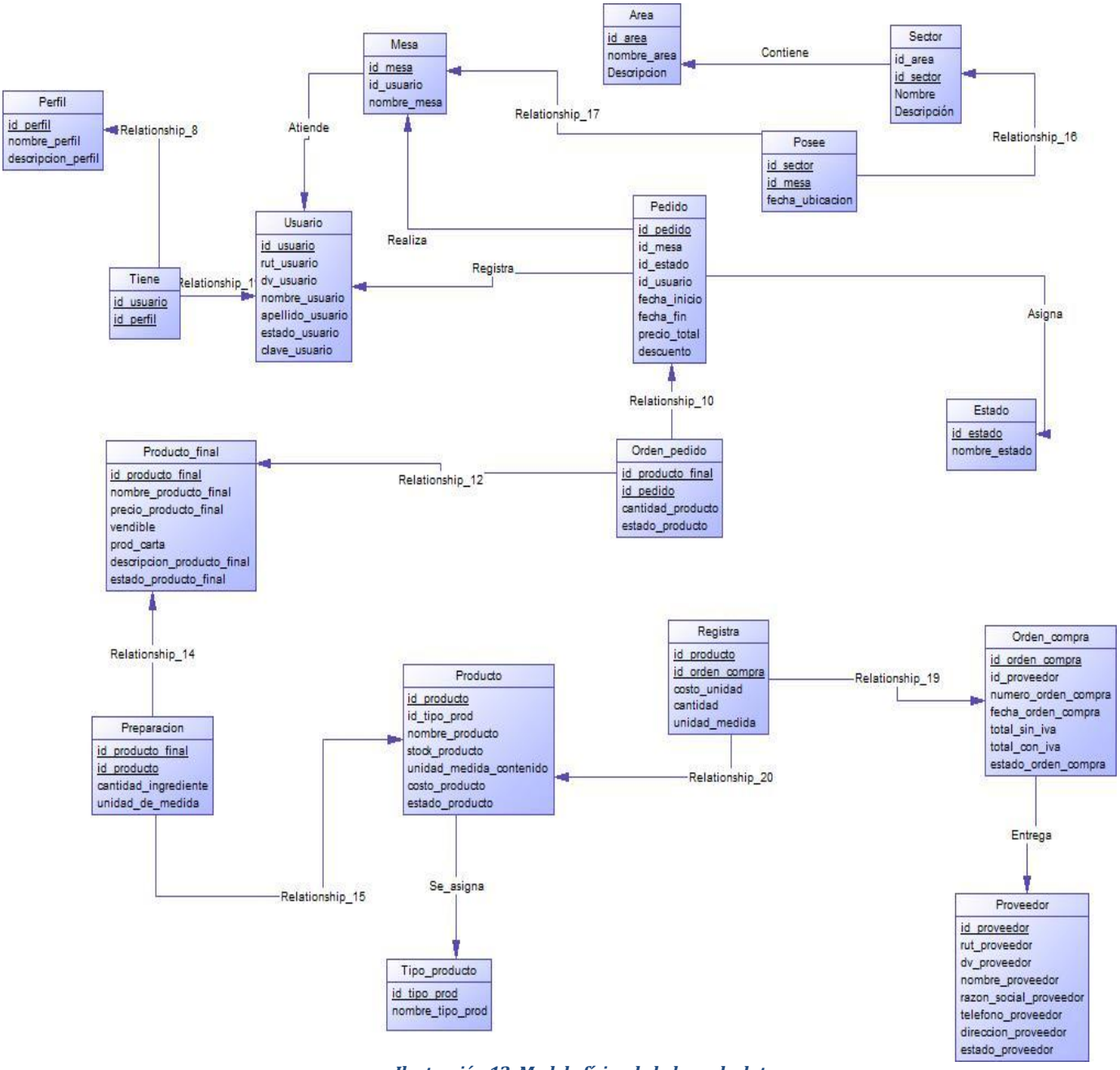


Ilustración 12: Modelo físico de la base de datos

Acontinuación se explica por secciones la **ilustracion 11**.

La ilustración representa la relación entre el usuario y su perfil asociado.

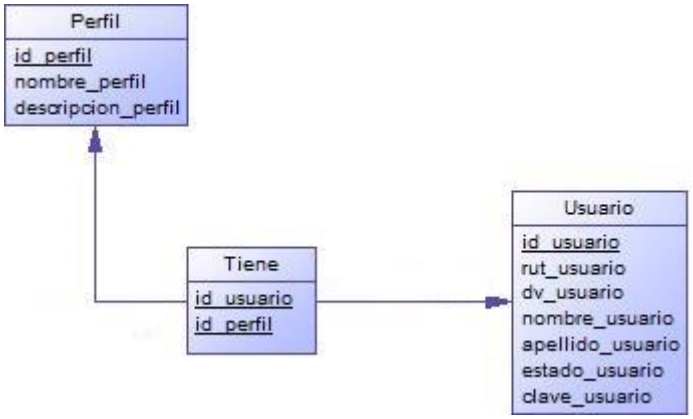


Ilustración 13: Modelo relacional: Control de usuario

La ilustración representa la relación que existe entre un producto y su preparación. El producto final es aquel producto que los administradores deciden vender y/u ofrecer en la carta.

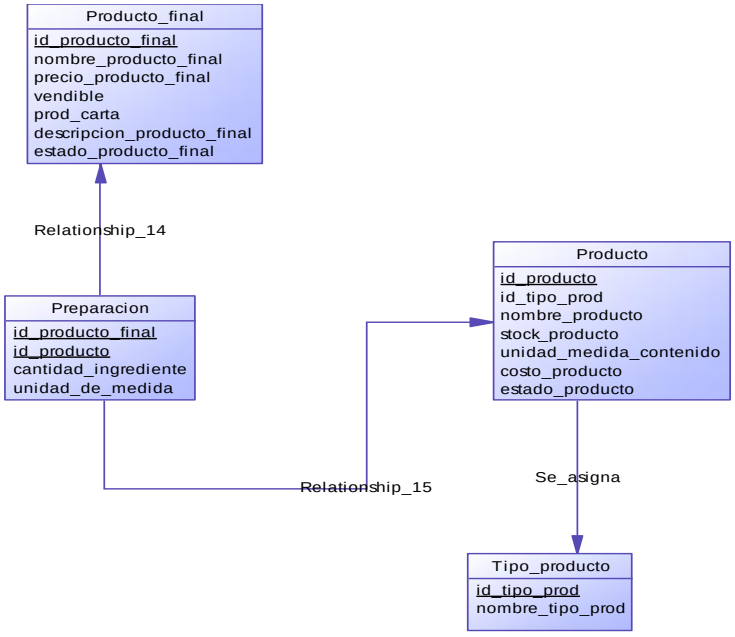


Ilustración 14: Modelo relacional: Control de productos en venta

La ilustración representa el control de productos en el sistema. La creación del producto, el ingreso de productos, las órdenes de compra asociadas a estas y a qué proveedor se ha comprado estos productos.

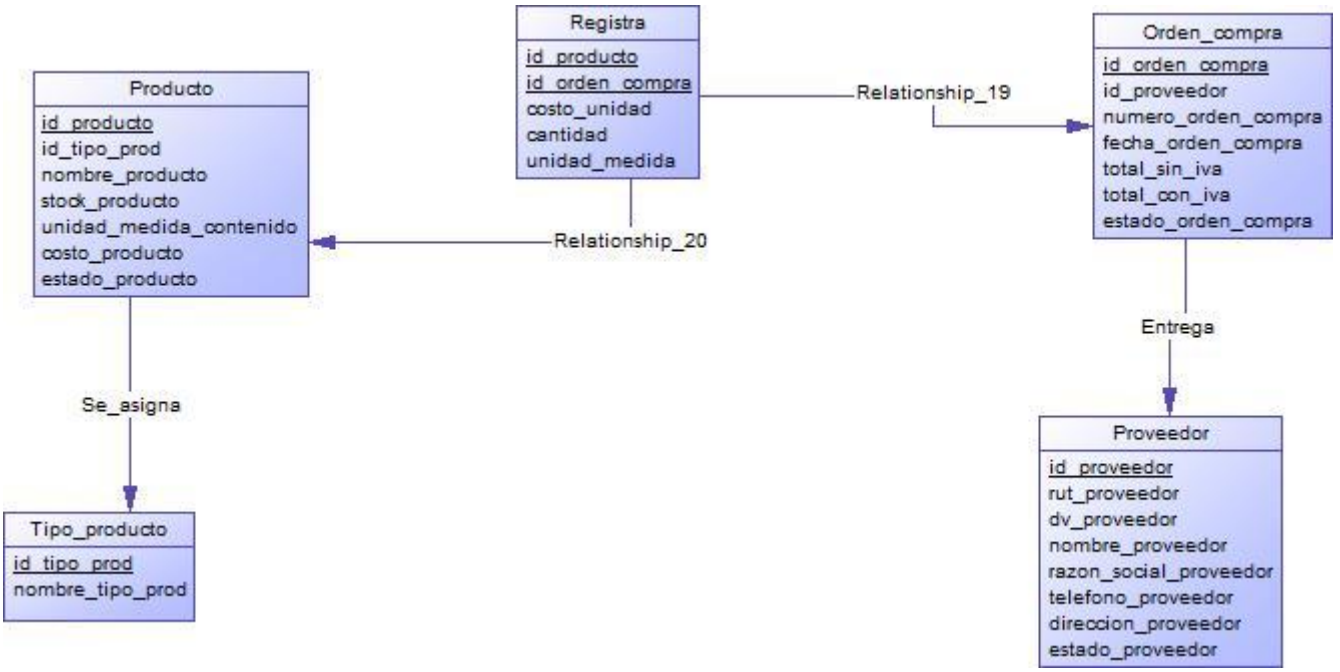


Ilustración 15: Modelo relacional: Control de productos

La ilustración representa el control de pedidos, la relación entre usuario y la generación de estos.

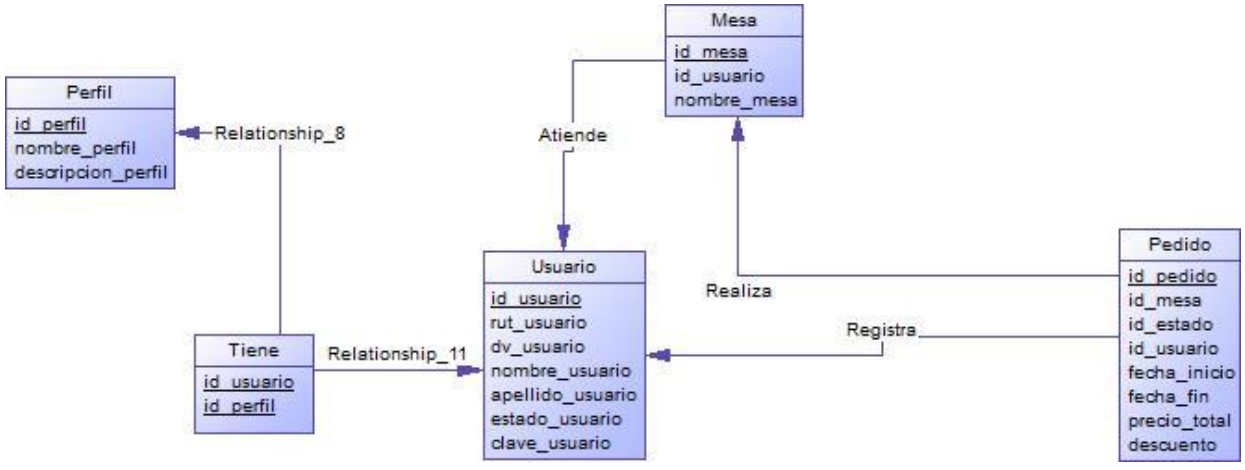


Ilustración 16: Modelo relacional: Control de pedidos

La ilustración representa el control de área. La tabla "Posee", sirve para mantener un registro las fechas en que la mesa ha sido asignada a un determinado sector.

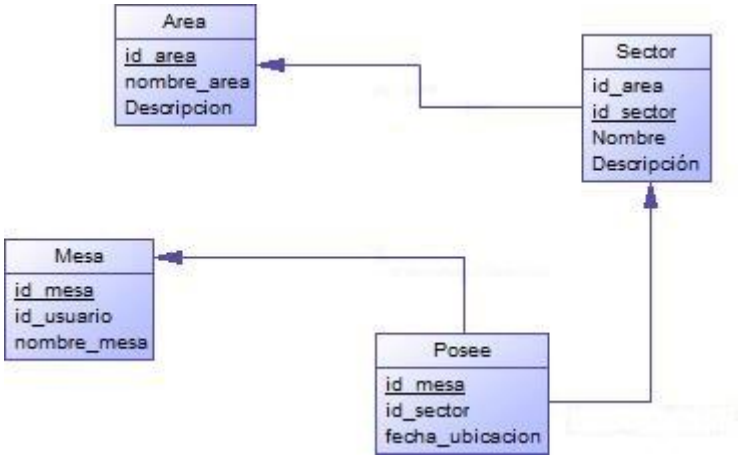


Ilustración 17: Modelo relacional: Control de área

7.2 Diseño de arquitectura funcional

El sistema está segmentado en 5 capas que se explicarán a continuación.

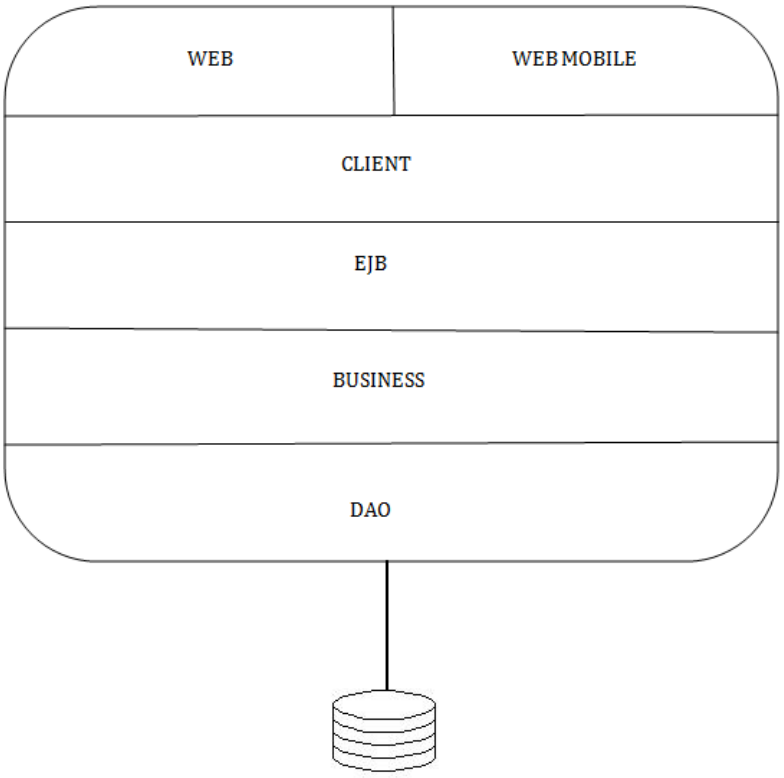


Ilustración 18: Esquema que representa la estructura del sistema.

Capa 1: La vista o capa 1 está dividida en la vista Web y la vista Web Mobile.
La vista Web está construida con framework Struts 1.1.
En la vista Web Mobile, dado que es orientada a dispositivos móviles, se construye usando los framework Jquery Mobile y Struts 1.1.
Capa 2: Medio por el cual se comunica las vistas con el EJB (Enterprise JabaBeans).
Capa 3: Parte transaccional del sistema. Está compuesto por interfaces.
Capa 4: Todo el negocio y comunicación con la base de datos.
Capa 5: Comunicación entre Java y la base de datos por medio de un DAO (Data Acces Object) con SQL incrustado.

La arquitectura está construida en base a J2EE estándar con versión de java1.4.

7.3 Diseño interfaz y navegación

Sistema Web.

- **Interfaz Login:** Estándar para el login. Contiene un formulario de autenticación y botón para iniciar sesión.

Ilustración 19: Login Sistema Web

- **Área (A):** Formulario con Rut de Usuario y Clave de acceso.
- **Área (B):** Botón Iniciar Sesión.

7.4 Especificación de módulos

Ver anexo 16.

- **Interfaz de Administrador:** Estándar para todas las tareas y operaciones del administrador. Contiene un menú y áreas de trabajo

Ilustración 20: Interfaz Administrador

- **Área (A)**: Encabezado de página. Contiene banner del local y el botón para cerrar sesión.
- **Área (B)**: Sector que contiene el menú de navegación.
- **Área (C)**: Este sector es el área de trabajo para el administrador. Contendrá botones, formularios y tablas.

- **Área (D)**: Información del usuario que inició sesión.

➢ **Interfaz Cajero:** Estándar para todas las tareas y operaciones del Cajero. Contiene un menú y área de trabajo.

Ilustración 21: Interfaz Cajero

- **Área (A)**: Encabezado de página. Contiene banner del local y el botón para cerrar sesión.
- **Área (B)**: Sector que contiene el menú de navegación.

• **Área (C)**: Este sector es el área de trabajo para el cajero. Contendrá botones, formularios y tablas.

• **Área (D)**: Información del usuario que inició sesión.

Sistema Móvil Web

➢ **Interfaz Login:** Estándar para el login. Contiene un formulario de autenticación y botón para iniciar sesión.

Ilustración 22: Login Sistema Móvil Web

• **Área (A):** Formulario con Rut de Usuario y Clave de acceso.

• **Área (B):** Botón Iniciar Sesión.

- **Interfaz Mesero y Barman:** Estándar para todas las tareas y operaciones del Mesero y Barman. Contiene un botones y área de trabajo.

Ilustración 23: Interfaz Mesero y Barman

- **Área (A):** Sector de área de trabajo Contendrá tablas, botones y formularios.
- **Área (B):** Encabezado con botón Cerrar Sesión.

Menú de navegación.

- **Sistema Web**

Menú Administrador

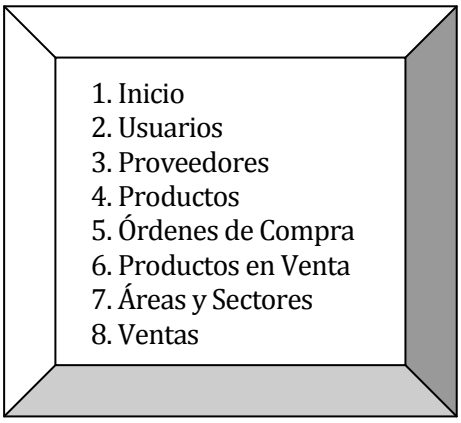


Ilustración 24: Menú Administrador

Menú Cajero

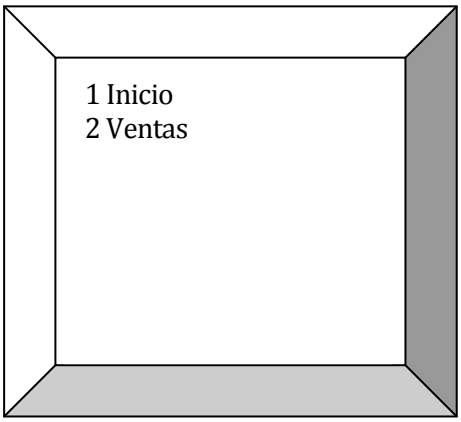


Ilustración 25: Menú Cajero

Diagrama de Navegación.

Nota: Se han separado los diagramas de navegación por perfil de usuario.

- **Sitio Web**

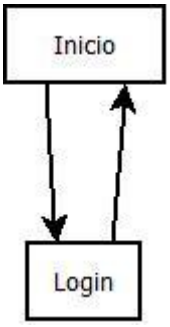


Ilustración 26: Navegación Sitio Web

- **Administrador**

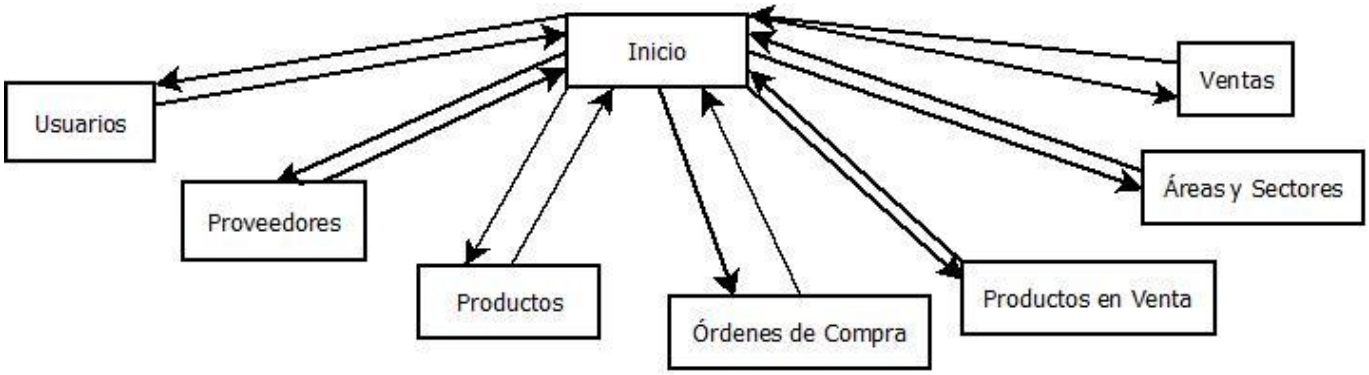


Ilustración 27: Navegación Administrador

- **Cajero**

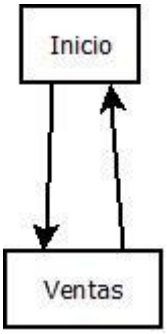


Ilustración 28: Navegación Cajero

- **Mesero y Barman**

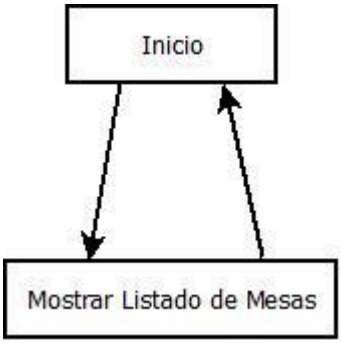


Ilustración 29: Navegación Mesero y Barman

8 PRUEBAS

Adaptación basada en *IEEE Software Test Documentation* Std 829-1998

8.1 Elementos de prueba

El sistema cuenta con los siguientes módulos, los cuales serán probados a nivel de sistema.

Tabla Nº14: Elementos de prueba

Sub-sistema	Descripción
Usuarios	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de usuarios.
Proveedores	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de proveedores.
Productos	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de productos.
Órdenes de Compra	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de órdenes de compra.
Productos en Venta	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de órdenes de compra.
Áreas y Sectores	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas a la asignación de mesas a los usuarios.
Ventas	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control los pedidos que estén por pagar y las ventas registradas.
Pedidos	Este sub-sistema entrega funcionalidades básicas de mantenimiento de información orientadas al control de pedidos realizados por el mesero o barman.

8.2 Especificación de las pruebas

A continuación se indican las características que representan las pruebas al sistema.

Tabla Nº15: Especificación de las pruebas

		Actividades de Prueba
Características a probar	Funcionalidad	1. Ejecutar Script SQL de la BD. 2. Ejecutar el Script con los dato del administrador. 3. Probar Sub-sistema Usuarios. - Ingresar Usuario - Buscar Usuario - Modificar Usuario - Asignar Perfil 4. Probar módulo ingresar Área. 5. Probar módulo modificar Área. 6. Probar módulo eliminar Área. 7. Probar módulo ingresar Sector. 8. Probar módulo modificar Sector. 9. Probar módulo eliminar Sector. 10.Probar módulo asignar mesa. 11. Probar módulo eliminar mesa. 12.Probar módulo eliminar Venta. 13.Probar módulo registro de Venta
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
Criterios de cumplimiento	Requerimientos cumplidos	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Proveedor. - Ingresar Proveedor - Buscar Proveedor - Modificar Proveedor
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de	Caja negra	

casos de prueba		
Técnicas para la definición de casos de prueba	Valores límites y particiones	
Criterios de cumplimiento	Que el usuario verifique que los requerimientos se cumplieron.	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Productos. - Ingresar Producto - Buscar Producto - Modificar Producto - Anular Producto
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Órdenes de Compra. - Ingresar Orden Compra - Buscar Orden Compra - Modificar Orden Compra - Anular Orden Compra
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Productos en venta.
Nivel de prueba	Sistema	

Objetivo de la prueba	Que se cumplan los requerimientos planteados	- Ingresar Producto en venta - Buscar Producto en venta - Modificar Producto en venta - Anular Producto en venta
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Áreas y Sectores. - Buscar Área - Buscar Sector - Buscar Mesa - Asignar Mesa
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Pedidos. - Ingresar Pedido - Buscar Pedido - Modificar Pedido - Anular Pedido
Nivel de prueba	Sistema	
Objetivo de la prueba	Que se cumplan los requerimientos planteados	
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	
		Actividades de Prueba
Características a probar	Funcionalidad	1. Probar Sub-sistema Ventas. - Ingresar Venta
Nivel de prueba	Sistema	

Objetivo de la prueba	Que se cumplan los requerimientos planteados	- Buscar Venta - Modificar Orden Compra - Anular Orden Compra - Modificar Venta - Anular Venta
Enfoque para definición de casos de prueba	Caja negra	
Técnicas para la definición de casos de prueba	Valores límites y particiones	

8.3 Responsables de las pruebas

Tabla Nº16: Responsables de las pruebas

Sub-Sistema	Responsable
Usuarios	Oscar Nahuelpán Alarcón (Alumno Memorista)
Proveedores	Oscar Nahuelpán Alarcón (Alumno Memorista)
Productos	Oscar Nahuelpán Alarcón (Alumno Memorista)
Órdenes de Compra	Oscar Nahuelpán Alarcón (Alumno Memorista)
Productos en Venta	Oscar Nahuelpán Alarcón (Alumno Memorista)
Áreas y Sectores	Oscar Nahuelpán Alarcón (Alumno Memorista)
Ventas	Oscar Nahuelpán Alarcón (Alumno Memorista)
Pedidos	Oscar Nahuelpán Alarcón (Alumno Memorista)

8.4 Calendario de pruebas

Tabla Nº17: Calendario de pruebas

Sub-Sistema	**10-mar**	**11-mar**	**12-mar**	**13-mar**	**14-mar**	**15-mar**	**16-mar**
Tester: Oscar Nahuelpán							
Usuarios							
Proveedores							
Productos							
Órdenes de Compra							
Productos en							

Venta							
Áreas y Sectores							
Pedidos							
Ventas							

8.5 Detalle de las pruebas

(Ver anexo 14. Detalles de Pruebas)

8.6 Conclusiones de Prueba

Al finalizar el proceso de pruebas de sistema, se ejecutaron cada una de las pruebas planificadas en el tiempo correspondiente.

Se realizaron las pruebas acorde a lo especificado utilizando casos de prueba para abarcar los casos de: Información insuficiente, información correcta, información inválida.

Se espera lograr, por medio de los presentes casos de prueba un sistema consistente, confiable, seguro y eficiente.

9 PLAN DE CAPACITACION Y ENTRENAMIENTO

Se presenta el plan de capacitación realizado para el correcto uso del software realizado del proyecto

- La capacitación está destinada al administrador del local, con el fin de que este pueda capacitar a sus empleados en las funcionalidades atribuibles a estos
- La capacitación será realizada por el equipo de proyecto, con una visita personal al administrador, para enseñar de forma didáctica y personalizada todas las funcionalidades del software.
- Debido a que se capacitara directamente a el(los) administrador(es) del pub restorán, se le enseñara todos los aspectos de funcionalidad que posee el software.
- El responsable de la capacitación es el Ingeniero Oscar Nahuelpán.
- El tiempo estimado de la capacitación son seis horas.
- Dentro del calendario programado del proyecto (carta Gantt), se considera la tarea de capacitación de dos días, con un máximo de tres horas por día.
- Los recursos que se utilizaran son:
    - Un computador con conexión a internet
    - SmartPhone
    - Impresora

10 PLAN DE IMPLANTACIÓN Y PUESTA EN MARCHA

A continuación se presenta el plan de implantación y puesta en marcha del proyecto:

- Para realizar correctamente la implantación, es necesario realizar las instalaciones necesarias del hardware para el correcto funcionamiento del software, se requiere más de un día de implementación del equipo necesario
- Según lo proyectado en la Carta Gantt, se destinan dos días para la implementación y puesta en marcha, las que se dividen en:
    - Día uno: Instalación del hardware
    - Día dos: Instalación del software

11 RESUMEN ESFUERZO REQUERIDO

Se detallan a continuación las horas trabajadas en el proyecto por alumno memorista.

Alumno Memorista: Oscar Nahuelpán Alarcón.

Tabla Nº18: Resumen esfuerzo memorista

Actividades/fases	N° Horas
Definición de Proyecto	**88**
Especificación de Requerimientos	**144**
Análisis	**224**
Diseño del Modelo y Base de Datos	**332**
Diseño Interfaz	**12**
Desarrollo y Codificación del Sistema	**1188**
Prueba de Sistema	**378**
Documentación Proyecto	**118**
TOTAL	**2484**

12 CONCLUSIONES

Dado que el sistema se encarga de apoyar el manejo de la información del pub-restaurant Sab City relacionado con los procesos de atención al cliente, venta y stock de productos se puede concluir que cumple con las necesidades principales del cliente y de esta forma dar un apoyo a la gestión del local.

El desarrollo del sistema lo consideré una oportunidad para conocer el método de programación Web denominada "programación por capas", potentes herramientas tales como Maven, WebLogic y tecnologías para el desarrollo de sistemas móviles (Jquery Mobile), además de aprovechar conceptos e ideas adquiridos durante la segunda práctica profesional. Juntos representan un gran aporte en el ámbito profesional.

Para finalizar he de decir que, si bien con la memoria culmina mis estudios en la Universidad del Bío-Bío, debido al cambio continuo y aparición de nuevas tecnologías, se está lejos de acabar con la etapa de estudios. En la etapa profesional se inicia otra y con mucho camino por recorrer.

13 BIBLIOGRAFÍA

- Pressman, Roger, Ingeniería de Software, un enfoque práctico 5ª edición McGraw-Hill
- Jquery Mobile Tutorial: http://www.w3schools.com/jquerymobile/
- Información y descarga de MySql: http://www.mysql.com/
- Información y descarga de Java: http://www.oracle.com/technetwork/java/index.html
- Tutorial WebLogic BEA: http://docs.oracle.com/cd/E13226_01/workshop/docs92/ws_platform/ideuserguide/TutorialGettingStarted/tutGS_Intro.html
- Definición de servidor de aplicaciones e instalación de Bea Weblogic 8.1: http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/sa/sesion1-apuntes.htm

14 ANEXO: PLANIFICACION INICIAL DEL PROYECTO

14.1 Estimación inicial de tamaño

Estimación casos de uso.

FACTOR DE PESO DE LOS ACTORES SIN AJUSTAR				
Tipo de Actor	**Descripción**	**Factor de Peso**	**Nro Actores**	**Subtotal (UAW)**
Simple	Otro Sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Aplication Programming Interface)	1	0	0
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2	0	0
Complejo	Una persona que interactúa con el sistema mediante una interfaz	3	4	12
Total Actores (UAW)			4	**12**

FACTOR DE PESOS DE CASO DE USO SIN AJUSTAR				
Tipo de Caso de Uso	**Descripción**	**Factor de Peso**	**Nro Casos de Uso**	**Subtotal (UUCW)**
Simple	El caso de uso contiene de 1 a 3 transacciones	5	38	190
Medio	El caso de uso contiene de 4 a 7 transacciones	10	1	10
Complejo	El caso de uso contiene más 8 transacciones	15	0	0
Total Use Cases (UUCW)			39	**200**

Puntos de Caso de Uso Sin Ajustar (UUCP)	**212**

FACTORES DE COMPLEJIDAD TECNICA (TCF)				
Factor	**Descripción**	**Factor de Peso**	**Valor Asignado (0-5)**	**Subtotal**
T1	Sistema distribuido	2	2	4
T2	Objetivos de performance o tiempo de respuesta	2	5	10
T3	Eficiencia del usuario	1	4	4
T4	Procesamiento interno complejo	1	4	4
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0,5	4	2
T7	Facilidad de uso	0,5	5	2,5
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	5	5
T10	Concurrencia	1	5	5
T11	Incluye objetivos especiales de seguridad	1	4	4
T12	Provee accesos directos a terceras partes	1	3	3
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3	3
Total Factor				**60,5**

14.1 Contabilización final del tamaño del sw

El proyecto tiene el siguiente número de líneas de código:

Las clases java en conjunto tienen 6312 líneas de código, las cuales están distribuidas en los siguientes proyectos.

sabCityWeb: 1694 de líneas de código
sabCityMobileWeb: 790 de líneas de código
sabCityEJBClient: 262 de líneas de código
sabcityEJB: 316 de líneas de código
sabcityBusiness: 3250 de líneas de código

FACTORES DE AMBIENTE (EF)				
Factor	**Descripción**	**Factor de Peso**	**Valor Asignado**	**Subtotal (EF)**
F1	Familiaridad con el modelo del proyecto Utilizado	1,5	2	3
F2	Experiencia en la aplicación	0,5	3	1,5
F3	Experiencia en orientación a objetos	1	4	4
F4	Capacidades de análisis	0,5	3	1,5
F5	Motivación	1	4	4
F6	Estabilidad de los requerimientos	2	3	6
F7	Personal Part-time	-1	3	-3
F8	Dificultad del lenguaje de programación	-1	4	-4
Total Factor Ambiente (EF)				**13**

Puntos de Caso de Uso Ajustado (UCP)	**258,01**

Nivel de esfuerzo.

Este cálculo se realiza con el fin de tener una aproximación del esfuerzo del grupo de trabajo, con respecto a los factores ambientales que se posee.

- Se contabilizan la cantidad de factores de ambiente (de F1 -> F6) que tengan un valor inferior a 3
- Se contabilizan la cantidad de factores de ambiente (F7 y F8) que tengan un valor superior a 3
- Sumamos los dos valores obtenidos y los evaluamos según la siguiente tabla:

Si el valor es <=2	20
Si el valor es 3 o 4	28
Si el valor es >=5	Reconsiderar Proyecto

Dado esto el para el proyecto se tiene un valor de evaluación de 3. Por lo tanto el nivel de esfuerzo LOE es de 20.

Esfuerzo estimado en Hora Persona.

Este cálculo se realiza para obtener un estimado de cuantas horas se necesitarán para desarrollar el proyecto. Una vez obtenido el punto de caso de uso ajustado (UCP) y el nivel de esfuerzo requerido para el proyecto (LOE) se realiza la siguiente operación final.

E = UCP * LOE
E = 258,01 * 20
E = 5160,2

Por lo tanto se estiman 5160 horas para el desarrollo del proyecto.

15 ANEXO: RESULTADO DE ITERACIONES EN EL DESARROLLO

Iteraciones en proyecto **sabcityWeb:**

- Control de Usuarios y Áreas y Sectores: Se crea un nuevo sector llamado "Barra" para que el barman se encargue de tomar los pedidos en barra.
- Control de Productos y Productos en Venta: Luego de conversaciones con el cliente y usuario, se decidió modificar en la base datos la interacción entre productos y productos en venta para hacer un mejor descuento en el stock de productos.

Iteraciones en proyecto **sabcityMobileWeb:**

- Generación de Pedidos: Se agrega la relación entre pedido y el usuario que modifica el estado de este.

16 ANEXO: ESPECIFICACION DE LAS PRUEBAS

16.1 Pruebas de Unidad

Sub-Sistema: Usuarios

Pre-Condición:

- Los perfiles o roles deben existir en el Sistema.
- Los datos marcados con negritas son auto generados por el sistema, por lo que no son considerados en los casos de prueba.

Datos de Entrada	**Valor**
Rut	D1
Nombre	D2
Apellido	D3
Perfil	D4
Estado Usuario	D5
Contraseña	D6

- **Casos de prueba para módulo Ingresar Usuario**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Ingresar Usuario**	**Ingresar Usuario**	**Ingresar Usuario**	**Ingresar Usuario**
	D1	6571090-0		6571090-0	6571090-0
	D2	Pedro Luis	Pedro Luis		Pedro Luis
	D3	Balmaceda Alarcón	Balmaceda Alarcón	Balmaceda Alarcón	
	D4	Administrador	Administrador	Administrador	Administrador
	D5	**1**	**1**	**1**	**1**
	D6	sab123	sab123	sab123	sab123
Salida Esperada		Usuario Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Usuario Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**	**6**	**7**
Requerimiento Funcional		**Ingresar Usuario**	**Ingresar Usuario**	**Ingresar Usuario**
	D1	6571090-0	6571090-0	11111111-3
	D2	Pedro Luis	Pedro Luis	Pedro Luis
	D3	Balmaceda Alarcón	Balmaceda Alarcón	Balmaceda Alarcón
	D4		Administrador	Administrador
	D5	**1**	**1**	**1**
	D6	sab123		Sab123
Salida Esperada		Usuario Guardado	Datos Incompletos	Rut Inválido
Salida Obtenida		Usuario Guardado	Datos Incompletos	Rut Inválido
Evaluación	**E/F**	Éxito	Éxito	Éxito
	Crit.	-	-	-

- **Casos de prueba para módulo Modificar Usuario**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Modificar Usuario**	**Modificar Usuario**	**Modificar Usuario**	**Modificar Usuario**
	D1	6571090-0		6571090-0	6571090-0
	D2	Pedro Luis	Pedro Luis		Pedro Luis
	D3	Balmaceda Alarcón	Balmaceda Alarcón	Balmaceda Alarcón	
	D4	Administrador	Administrador	Administrador	Administrador
	D5	**1**	**1**	**1**	**1**
	D6	sab123	sab123	sab123	sab123
Salida Esperada		Usuario Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Usuario Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**	**6**	**7**
Requerimiento Funcional		**Modificar Usuario**	**Modificar Usuario**	**Modificar Usuario**
	D1	6571090-0	6571090-0	11111111-3
	D2	Pedro Luis	Pedro Luis	Pedro Luis
	D3	Balmaceda Alarcón	Balmaceda Alarcón	Balmaceda Alarcón
	D4		Administrador	Administrador
	D5	**1**	**1**	**1**
	D6	sab123		Sab123
Salida Esperada		Usuario Guardado	Datos Incompletos	Rut Inválido
Salida Obtenida		Usuario Guardado	Datos Incompletos	Rut Inválido
Evaluación	**E/F**	Éxito	Éxito	Éxito
	Crit.	-	-	-

- **Casos de prueba para módulo Buscar Usuario**

ID		**1**	**2**
Requerimiento Funcional		**Buscar Usuario**	**Buscar Usuario**
	D1	6571090-0	
	D2		Pedro Luis
Salida Esperada		Usuario Encontrado	Usuarios Encontrado
Salida Obtenida		Usuario Encontrado	Usuarios Encontrado
Evaluación	**E/F**	Éxito	Éxito
	Crit.	-	-

Sub-Sistema Proveedores

Pre-Condición:

- Los datos marcados con negritas son auto generados por el sistema, por lo que no son considerados en los casos de prueba.

Datos de Entrada	**Valor**
Rut	D1
Nombre Empresa	D2
Razón Social	D3
Teléfono 1	D4
Teléfono 2	D5
Teléfono 3	D6
Dirección	D7
Estado	D8

- **Casos de prueba para módulo Ingresar Proveedor**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Ingresar Proveedor**	**Ingresar Proveedor**	**Ingresar Proveedor**	**Ingresar Proveedor**
	D1	15115115 -9		15115115 -9	15115115 -9
	D2	Kamadi	Kamadi		Kamadi
	D3	Kamadi S.A	Kamadi S.A	Kamadi S.A	
	D4	837498322	837498322	837498322	837498322
	D5	87648822	87648822	87648822	87648822
	D6	56481119	56481119	56481119	56481119
	D7	Ainavillo #700	Ainavillo #700	Ainavillo #700	Ainavillo #700
	D8	**1**	**1**	**1**	**1**
Salida Esperada		Proveedor Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Proveedor Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**	**6**	**7**	**8**
Requerimiento Funcional		**Ingresar Proveedor**	**Ingresar Proveedor**	**Ingresar Proveedor**	**Ingresar Proveedor**
	D1	15115115 -9	15115115 -9	15115115 -9	15115115 -9
	D2	Kamadi	Kamadi	Kamadi	Kamadi
	D3	Kamadi S.A	Kamadi S.A	Kamadi S.A	Kamadi S.A
	D4		837498322	837498322	837498322
	D5	87648822		87648822	87648822
	D6	56481119	56481119		56481119
	D7	Ainavillo #700	Ainavillo #700	Ainavillo #700	
	D8	**1**	**1**	**1**	**1**
Salida Esperada		Proveedor Guardado	Proveedor Guardado	Proveedor Guardado	Datos Incompletos
Salida Obtenida		Proveedor Guardado	Proveedor Guardado	Proveedor Guardado	Datos Incompletos

Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**9**
Requerimiento Funcional		**Ingresar Proveedor**
	D1	15115115 -8
	D2	Kamadi
	D3	Kamadi S.A
	D4	837498322
	D5	87648822
	D6	56481119
	D7	Ainavillo #700
	D8	**1**
Salida Esperada		Rut no Válido
Salida Obtenida		Rut no Válido
Evaluación	**E/F**	Éxito
	Crit.	-

- **Casos de prueba para módulo Modificar Proveedor**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Modificar Proveedor**	**Modificar Proveedor**	**Modificar Proveedor**	**Modificar Proveedor**
	D1	15115115 -9		15115115 -9	15115115 -9
	D2	Kamadi	Kamadi		Kamadi
	D3	Kamadi S.A	Kamadi S.A	Kamadi S.A	
	D4	837498322	837498322	837498322	837498322
	D5	87648822	87648822	87648822	87648822
	D6	56481119	56481119	56481119	56481119
	D7	Ainavillo #700	Ainavillo #700	Ainavillo #700	Ainavillo #700
	D8	**1**	**1**	**1**	**1**
Salida Esperada		Proveedor Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Proveedor Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**	**6**	**7**	**8**
Requerimiento Funcional		**Modificar Proveedor**	**Modificar Proveedor**	**Modificar Proveedor**	**Modificar Proveedor**
	D1	15115115 -9	15115115 -9	15115115 -9	15115115 -9
	D2	Kamadi	Kamadi	Kamadi	Kamadi
	D3	Kamadi S.A	Kamadi S.A	Kamadi S.A	Kamadi S.A
	D4		837498322	837498322	837498322
	D5	87648822		87648822	87648822
	D6	56481119	56481119		56481119
	D7	Ainavillo #700	Ainavillo #700	Ainavillo #700	

	D8	**1**	**1**	**1**	**1**
Salida Esperada		Proveedor Guardado	Proveedor Guardado	Proveedor Guardado	Datos Incompletos
Salida Obtenida		Proveedor Guardado	Proveedor Guardado	Proveedor Guardado	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**9**
Requerimiento Funcional		**Modificar Proveedor**
	D1	15115115 -8
	D2	Kamadi
	D3	Kamadi S.A
	D4	837498322
	D5	87648822
	D6	56481119
	D7	Ainavillo #700
	D8	**1**
Salida Esperada		Rut no Válido
Salida Obtenida		Rut no Válido
Evaluación	**E/F**	Éxito
	Crit.	-

- **Casos de prueba para módulo Buscar Proveedor**

ID		**1**	**2**
Requerimiento Funcional		**Buscar Proveedor**	**Buscar Proveedor**
	D1	15115115 -8	
	D2		Kamadi
Salida Esperada		Proveedor Encontrado	Proveedores Encontrados
Salida Obtenida		Proveedor Encontrado	Proveedores Encontrados
Evaluación	**E/F**	Éxito	Éxito
	Crit.	-	-

Sub-Sistema Producto.

Pre-Condición:

- Las categorías del producto deben existir en el Sistema.

Datos de Entrada	**Valor**
Nombre	D1
Categoría Producto	D2
Contenido	D3
Unidad Medida Contenido	D4

- **Casos de prueba para módulo Ingresar Producto**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Ingresar Producto**	**Ingresar Producto**	**Ingresar Producto**	**Ingresar Producto**
	D1	Azúcar		Azúcar	Azúcar
	D2	Comestible	Comestible		Comestible
	D3	2	2	2	
	D4	Kg	Kg	Kg	Kg
Salida Esperada		Producto Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Producto Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**
Requerimiento Funcional		**Ingresar Producto**
	D1	Azúcar
	D2	Comestible
	D3	2
	D4	
Salida Esperada		Datos Incompletos
Salida Obtenida		Datos Incompletos
Evaluación	**E/F**	Éxito
	Crit.	-

- **Casos de prueba para módulo Modificar Producto**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Modificar Producto**	**Modificar Producto**	**Modificar Producto**	**Modificar Producto**
	D1	Azúcar		Azúcar	Azúcar
	D2	Comestible	Comestible		Comestible
	D3	2	2	2	
	D4	Kg	Kg	Kg	Kg
Salida Esperada		Producto Guardado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Producto Modificado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**
Requerimiento Funcional		**Modificar Producto**
	D1	Azúcar
	D2	Comestible
	D3	2
	D4	
Salida Esperada		Datos Incompletos
Salida Obtenida		Datos Incompletos
Evaluación	**E/F**	Éxito
	Crit.	-

- **Casos de prueba para módulo Buscar Producto**

ID		**1**	**2**
Requerimiento Funcional		**Buscar Productos**	**Buscar Productos**
	D1	Azúcar	
Salida Esperada		Productos Encontrados	Datos Incompletos
Salida Obtenida		Productos Encontrado	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito
	Crit.	-	-

Sub-Sistema Orden de Compra.

Pre-Condición:

- Deben existir proveedores en el sistema
- Deben existir productos en el sistema
- Las categorías del producto deben existir en el Sistema.

Datos de Entrada	**Valor**
Proveedor	D1
Nº de la orden	D2
Total sin IVA incluido	D3
Total con IVA incluido	D4
Tipo del Producto	D5
Nombre Producto	D6
Cantidad del Producto	D7
Valor Unitario	D8

- **Casos de prueba para módulo Agregar Producto**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Agregar Producto**	**Agregar Producto**	**Agregar Producto**	**Agregar Producto**
	D5	Pisco		Pisco	Pisco
	D6	Pisco Alto del Carmen 35º	Pisco Alto del Carmen 35º		Pisco Alto del Carmen 35º
	D7	1	1	1	
	D8	4190	4190	4190	4190
Salida Esperada		Producto Agregado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Producto Agregado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**
Requerimiento Funcional		**Agregar Producto**
	D5	Pisco
	D6	Pisco Alto del Carmen 35º
	D7	1
	D8	
Salida Esperada		Datos Incompletos
Salida Obtenida		Datos Incompletos
Evaluación	**E/F**	Éxito
	Crit.	-

- **Casos de prueba para módulo Ingresar Orden de Compra**

  Post-Condición: Para Ingresar la orden de compra, esta debe tener al menos un producto agregado.

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Ingresar Orden de Compra**	**Ingresar Orden de Compra**	**Ingresar Orden de Compra**	**Ingresar Orden de Compra**
	D1	Kamadi		Kamadi	Kamadi
	D2	4213	4213		4213
	D3	4190	4190	4190	
	D4	4190	4190	4190	4190
	D5	Pisco	Pisco	Pisco	Pisco
	D6	Pisco Alto del Carmen 35º	Pisco Alto del Carmen 35º	Pisco Alto del Carmen 35º	Pisco Alto del Carmen 35º
	D7	1	1	1	1
	D8	4190	4190	4190	4190
Salida Esperada		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**5**	**6**	**7**	**8**
Requerimiento Funcional		**Ingresar Orden de Compra**	**Ingresar Orden de Compra**	**Ingresar Orden de Compra**	**Ingresar Orden de Compra**
	D1	Kamadi	Kamadi	Kamadi	Kamadi
	D2	4213	4213	4213	4213
	D3	4190	4190	4190	4190
	D4		4190	4190	4190
	D5	Pisco		Pisco	Pisco
	D6	Pisco Alto del Carmen 35º	Pisco Alto del Carmen 35º		Pisco Alto del Carmen 35º
	D7	1	1	1	
	D8	4190	4190	4190	4190
Salida Esperada		Datos Incompletos	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Datos Incompletos	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

ID		**9**
Requerimiento Funcional		**Ingresar Orden de Compra**
	D1	Kamadi
	D2	4213
	D3	4190
	D4	4190
	D5	Pisco
	D6	Pisco Alto del Carmen 35º
	D7	1
	D8	
Salida Esperada		Datos Incompletos
Salida Obtenida		Datos Incompletos
Evaluación	**E/F**	Éxito
	Crit.	-

Sub-Sistema Productos en venta.

Pre-Condición:

- Deben existir productos en el sistema
- Las categorías del producto deben existir en el Sistema.
- Los datos marcados con negritas son auto generados por el sistema, por lo que no son considerados en los casos de prueba.

Datos de Entrada	**Valor**
Nombre Producto en Venta	D1
Precio	D2
Vendible	D3
Agregar Producto a la Carta	D4
Preparación	D5
Tipo Producto	D6
Nombre Producto	D7
Cantidad	D8

- **Casos de prueba para módulo Agregar Producto**

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Agregar Producto**	**Agregar Producto**	**Agregar Producto**	**Agregar Producto**
	D6	Ron		Ron	Ron
	D7	Ron Bacardi Blanco	Ron Bacardi Blanco		Ron Bacardi Blanco
	D8	1.75	1.75	1.75	
Salida Esperada		Producto Agregado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Producto Agregado	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

- **Casos de prueba para módulo Ingresar Producto en Venta**

  Post-Condición: Para Ingresar un producto en venta, esta debe tener al menos un producto agregado.

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Ingresar Producto en Venta**	**Ingresar Producto en Venta**	**Ingresar Producto en Venta**	**Ingresar Producto en Venta**
	D1	Caipirisima		Caipirisima	Caipirisima
	D2	3000	3000		3000
	D3	**1**	**1**	**1**	**1**
	D4	**1**	**1**	**1**	**1**
	D5	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	
Salida Esperada		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

- **Casos de prueba para módulo Modificar Producto en Venta**

  Post-Condición: Para Ingresar un producto en venta, esta debe tener al menos un producto agregado.

ID		**1**	**2**	**3**	**4**
Requerimiento Funcional		**Modificar Producto en Venta**	**Modificar Producto en Venta**	**Modificar Producto en Venta**	**Modificar Producto en Venta**
	D1	Caipirisima		Caipirisima	Caipirisima
	D2	3000	3000		3000
	D3	**1**	**1**	**1**	**1**
	D4	**1**	**1**	**1**	**1**
	D5	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	Coloque 4/6 de limón en un vaso bajo, agregue 3/2 cucharadas de azúcar y machaque con una cuchara de bar. Incorpore el hielo, agregue 7/4 oz de Ron Blanco, mezcle y sirva en un vaso corto	
Salida Esperada		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Salida Obtenida		Orden de Compra Guardada	Datos Incompletos	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito	Éxito
	Crit.	-	-	-	-

Sub-Sistema Áreas y Sectores.

Pre-Condición:

- Deben existir áreas en el Sistema.
- Deben existir sectores en el Sistema.
- Deben existir mesas en el Sistema.
- Deben existir usuarios en el Sistema.
- Los datos marcados con negritas son auto generados por el sistema, por lo que no son considerados en los casos de prueba.

Datos de Entrada	**Valor**
Áreas	D1
Sectores	D2
Mesas	D3
Usuarios	D4
Seleccionar Mesa	D5

- **Casos de prueba para módulo Asignar Mesa**

ID		**1**	**2**
Requerimiento Funcional		**Agregar Producto**	**Agregar Producto**
	D1	Área de Atención	Área de Atención
	D2	SectorMesa2	SectorMesa2
	D3	Nº Mesa	Nº Mesa
	D4	Tamara Fuentes	Tamara Fuentes
	D5	Seleccionado	No Seleccionado
Salida Esperada		Mesa Asignada	Datos Incompletos
Salida Obtenida		Mesa Asignada	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito
	Crit.	-	-

Sub-Sistema Ventas.

NOTA: Las acciones realizas en este sub-Sistema son autocorregidos por el sistema por lo que no son considerados en los casos de prueba.

Sub-Sistema Pedidos.

Pre-Condición:

- Deben existir Áreas en el Sistema.
- Deben existir Sectores en el Sistema.
- Deben existir Mesas en el Sistema.
- Deben existir Usuarios en el Sistema.
- Deben existir Productos en Venta en el Sistema.
- Los datos marcados con negritas son auto generados por el sistema, por lo que no son considerados en los casos de prueba.

Datos de Entrada	**Valor**
Mesas	D1
Producto en Venta	D2
Cantidad Producto	D3

- **Casos de prueba para módulo Agregar Producto**

ID		**1**	**2**	**3**
Requerimiento Funcional		**Agregar Producto**	**Agregar Producto**	**Agregar Producto**
	D2	Caipirisima		Caipirisima
	D3	4	4	
Salida Esperada		Producto Agregado	Datos Incompletos	Datos Incompletos
Salida Obtenida		Producto Agregado	Datos Incompletos	Datos Incompletos
Evaluación	**E/F**	Éxito	Éxito	Éxito
	Crit.	-	-	-

- **Casos de prueba para módulo Agregar Producto**

  Post-Condición: Para Ingresar un pedido, esta debe tener al menos un producto agregado.

  NOTA: Las acciones realizas en este módulo son autocorregidos por el sistema por lo que no son considerados en los casos de prueba.

17 ANEXO: ESPECIFICACIÓN DE MÓDULOS

17.1 cl.sabcity.dao
Class MysqlSabcityDAO

java.lang.Object
└ **cl.sabcity.dao.MysqlSabcityDAO**
All Implemented Interfaces:
SabcityDAO

---

public class **MysqlSabcityDAO**
extends java.lang.Object
implements SabcityDAO

Implementa la interfaz MysqlSabcityDAO para Oracle DAO Base de servicios relativos a Sabcity

---

Constructor Summary
MysqlSabcityDAO() Constructor de la clase.

Method Summary	
void	**actualizarArea**(AreaDVO area) Permite actualizar un área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarMesa**(MesaDVO mesa) Actualiza una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarPedido**(PedidoDVO pedido) Actualiza los datos de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarPerfil**(PerfilDVO perfil) Actualiza un perfil Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarPreparacion**(PreparacionDVO preparacion) Actualiza la preparación de un producto final Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarProducto**(ProductoDVO producto) Actualiza los datos de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarProveedor**(ProveedorDVO proveedor) Permite actualizar los datos de un proveedor Registro de Versiones:

	24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarSector**(SectorDVO sector) Actualiza los datos de un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarStockProducto**(int idProducto, int stockProducto) Actualiza el stock de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarUsuario**(UsuarioDVO usuario) Actualiza los datos de un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**agregarProductoAPedido**(int idPedido, int idProducto, int cantidadProducto) Agrega un nuevo producto a un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarArea**(int idArea) Elimina un área de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarIngredientePreparacion**(int idProductoFinal, int idIngrediente) Elimina un ingrediente de la preparación de un producto final Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarMesa**(int idMesa) Elimina una mesa de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarPedido**(int idPedido) Actualiza los datos de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarPerfil**(int idPerfil) Elimina un perfil de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarPreparacion**(int idProductoFinal) Elimina la preparación de un producto final Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarProducto**(int idProducto) Elimina un producto de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarProveedor**(int rutProveedor) Elimina un proveedor de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarSector**(int idSector) Elimina un sector de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarUsuario**(int rutUsuario)

	Elimina un usuario de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
AreaDVO	**getBuscarArea**(java.lang.String nombreArea) Permite buscar un área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getBuscarIdPedido**(int idMesa, int idEstado) Retorna la id de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getBuscarIdUsuario**(int rutUsuario) Busca el id de un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getBuscarListaPedidosPorSector**(int idSector) Devuelve arreglo de pedidos por sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO	**getBuscarMesa**(int idMesa) Busca una mesa en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO	**getBuscarMesaPorPedido**(int idPedido) Devuelve una mesa que haya solicitado un determinado pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getBuscarMesaPorUsuario**(int idUsuario) Busca las mesas pertenecientes a un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getbuscarMesasPorSector**(int idSector) Busca las mesas pertenecientes a un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
OrdenPedidoDVO[]	**getBuscarOrdenPedidos**(int idPedido) Busca las órdenes asociadas a un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedido**(int idPedido) Busca un pedido en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedidoMesaPorEstado**(int idMesa, int idEstado) Busca el pedido pendiente de una mesa dependiendo de su estado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedidoPorMesa**(int idMesa) Busca el pedido pendiente de una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedidoPorMesaEstado**(int idMesa, int estadoOrden) Busca el pedido pendiente de una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getBuscarPedidoPorUsuario**(int idUsuario) Devuelve un arreglo de pedidos del que está a cargo un usuario

	determinado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO	**getBuscarPerfil**(int idPerfil) Busca un perfil en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO	**getBuscarPerfilAsignado**(int idUsuario) Busca un perfil asignado a un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PreparacionDVO	**getBuscarPreparacion**(int idProductoFinal) Busca la preparación de un producto final Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO	**getBuscarProducto**(int idProducto) Busca un producto en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getBuscarProductoXNombre**(java.lang.String nombreProducto) Devuelve un listado de productos de acuerdo al nombre Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getBuscarProductoXTipo**(java.lang.String nombreTipoProducto) Devuelve los productos que tengan un determinado tipo Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO[]	**getBuscarProveedorXNombre**(java.lang.String nombreProveedor) Busca un listado de proveedores de acuerdo al nombre ingresado por parámetro Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO	**getBuscarProveedorXRut**(int rutProveedor) Devuelve un proveedor de acuerdo al rut ingresado por parámetro UsuarioDVO[] listaUsuarios = delegate.getBuscarUsuarioXNombre(nombreUsuario); session.setAttribute("usuarios.busqueda.listaUsuarios", listaUsuarios); Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
SectorDVO	**getBuscarSector**(int idSector) Busca un sector en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
SectorDVO[]	**getBuscarSectorPorArea**(int idArea) Busca un sector de acuerdo al área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
TipoProductoDVO	**getBuscarTipoProducto**(int idTipoProducto) Devuelve un tipo de producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO	**getBuscarUsuario**(int rutUsuario) Busca un usuario en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial p.id_perfil
UsuarioDVO	**getBuscarUsuarioPorId**(int idUsuario)

	Busca un usuario en la base de datos por su ID Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial p.id_perfil
UsuarioDVO	**getBuscarUsuarioPorPedido**(int idPedido) Devuelve el usuario que esté a cargo de un determinado pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getbuscarUsuarioPorPerfil**(java.lang.String nombrePerfil) Devuelve el lisado de usuarios activos con un determinado perfil Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getBuscarUsuarioXNombre**(java.lang.String nombreUsuario) Busca un usuario en la base de datos por su nombre Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getIdProductoFinal**(java.lang.String nombreProductoFinal) Devuelve idProductoFinal Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
AreaDVO[]	**getListaAreas**() Devuelve el listado de áreas ingresadas en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
OrdenCompraDVO[]	**getListadoOrdenCompra**() Retorna un listado de órdenes de compra Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoFinalDVO[]	**getListadoProductoCarta**() Devuelve el listado de productos que pertenecen a la carta Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getListaMesas**() Devuelve un arreglo de mesas Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getListaMesasByEstado**(int estadoOrden) Devuelve un arreglo de mesas Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getListaPedidos**() Devuelve arreglo de pedidos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO[]	**getListaPerfiles**() Devuelve la lista de perfiles Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getListaProductos**() Devuelve un arreglo de productos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO[]	**getListaProveedoresActivos**() Muestra el listado de proveedores del local Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO[]	**getListaProveedoresAll**() Muestra el listado de proveedores del local Registro de Versiones:

	24/01/2014 [Oscar - UBB.]: Versión Inicial
SectorDVO[]	**getListaSectores**() Devuelve un arreglo de sectores Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
TipoProductoDVO[]	**getListaTipoProductos**() Devuelve todos los tipo de productos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getListaUsuarios**() Devuelve un arreglo de usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getListaUsuariosActivos**() Devuelve un arreglo de usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
EstadoDVO	**getObtenerEstado**(int idEstado) Retorna un estado para los pedidos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getObtenerIdOrdenCompra**(double numOrdenCompra) Devuelve el id de una orden de compra Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getObtenerStockProducto**(int idProducto) Retorna el stock de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**ingresarPreparacion**(int idProductoFinal, int idIngrediente, int cantidadIngrediente, java.lang.String tipoMedida) Asigna una preparación a un producto final Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**ingresarRecibo**(int idOrdenRecibo, int idProveedor, int numeroOrdenRecibo) Se crea un recibo por ingresar un producto a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**registrarProducto**(int idProducto, int idOrdenRecibo) Se asocia un producto a un recibo Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setAsignarPerfil**(int idPerfil, int idUsuario) Asigna un perfil a un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setAsignarUsuarioMesa**(int idMesa, int idUsuario) Asigna un usuario a una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setCambiarEstado**(int idEstado, int idMesa, int idPedido) Cambia el estado del Pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarArea**(AreaDVO area)

	Ingresa una nueva área en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarMesa**(MesaDVO mesa) Ingresa una mesa en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarOrdenCompra**(OrdenCompraDVO ordenCompra) Genera una orden de compra Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarOrdenPedido**(OrdenPedidoDVO ordenPedido) Ingresa una orden de pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPedido**(PedidoDVO pedido) Ingresa un nuevo pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPerfil**(PerfilDVO perfil) Ingresa un perfil en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPreparacion**(PreparacionDVO preparacion) Se ingresa la preparación de un producto final a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarProducto**(ProductoDVO producto) Ingresa un producto nuevo en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarProductoFinal**(ProductoFinalDVO productoFinal) Se ingresa un producto final a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarProveedor**(ProveedorDVO proveedor) Permite ingresar un nuevo proveedor al sistema Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarRegistra**(RegistraDVO registra) Se registra la cantidad y costo del producto comprado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarSector**(SectorDVO sector) Ingresa un nuevo sector en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarUsuario**(UsuarioDVO usuario) Ingresa un nuevo usuario en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial

17.2 Class SabcityBusiness

java.lang.Object
└ **cl.sabcity.bpro.SabcityBusiness**

---

public class **SabcityBusiness**
extends java.lang.Object

Business del manejo de sabcity

Registro de Versiones:

- 09/01/2014 [Oscar - UBB]: Versión Inicial

Todos los derechos reservados - Universidad del Bio Bio

Constructor Summary
SabcityBusiness() Constructor de la Clase Registro de Versiones: 06-02-2014 [Oscar - UBB]: Versión Inicial

Method Summary	
void	**actualizarArea**(AreaDVO area) Permite actualizar un área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarMesa**(MesaDVO mesa) Actualiza una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarPedido**(PedidoDVO pedido) Actualiza los datos de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarPerfil**(PerfilDVO perfil) Actualiza un perfil Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarProducto**(ProductoDVO producto) Actualiza los datos de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarProveedor**(ProveedorDVO proveedor) Permite actualizar los datos de un proveedor Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarSector**(SectorDVO sector) Actualiza los datos de un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarStockProducto**(int idProducto, int stockProducto)

	Actualiza el stock de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**actualizarUsuario**(UsuarioDVO usuario) Actualiza los datos de un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**agregarProductoAPedido**(int idPedido, int idProducto, int cantidadProducto) Agrega un nuevo producto a un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarArea**(int idArea) Elimina un área de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarMesa**(int idMesa) Elimina una mesa de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarPedido**(int idPedido) Actualiza los datos de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarPerfil**(int idPerfil) Elimina un perfil de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarProducto**(int idProducto) Elimina un producto de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarProveedor**(int rutProveedor) Elimina un proveedor de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarSector**(int idSector) Elimina un sector de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**eliminarUsuario**(int rutUsuario) Elimina un usuario de la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
AreaDVO	**getBuscarArea**(java.lang.String nombreArea) Permite buscar un área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getBuscarIdPedido**(int idMesa, int idEstado) Retorna la id de un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getBuscarIdUsuario**(int rutUsuario) Busca el id de un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getBuscarListaPedidosPorSector**(int idSector)

	Devuelve arreglo de pedidos por sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO	**getBuscarMesa**(int idMesa) Busca una mesa en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO	**getBuscarMesaPorPedido**(int idPedido) Devuelve una mesa que haya solicitado un determinado pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getBuscarMesaPorUsuario**(int idUsuario) Busca las mesas pertenecientes a un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getbuscarMesasPorSector**(int idSector) Busca las mesas pertenecientes a un sector Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
OrdenPedidoDVO[]	**getBuscarOrdenPedidos**(int idPedido) Busca las órdenes asociadas a un pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedido**(int idPedido) Busca un pedido en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedidoMesaPorEstado**(int idMesa, int idEstado) Busca el pedido pendiente de una mesa dependiendo de su estado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO	**getBuscarPedidoPorMesa**(int idMesa) Busca el pedido pendiente de una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getBuscarPedidoPorUsuario**(int idUsuario) Devuelve un arreglo de pedidos del que está a cargo un usuario determinado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO	**getBuscarPerfil**(int idPerfil) Busca un perfil en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO	**getBuscarPerfilAsignado**(int idUsuario) Busca un perfil asignado a un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO	**getBuscarProducto**(int idProducto) Busca un producto en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getBuscarProductoXNombre**(java.lang.String nombreProducto) Devuelve un listado de productos de acuerdo al nombre Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getBuscarProductoXTipo**(java.lang.String nombreTipoProducto)

	Devuelve los productos que tengan un determinado tipo Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO[]	**getBuscarProveedorXNombre**(java.lang.String nombreProveedor) Busca un proveedor de acuerdo al nombre ingresado por parámetro Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO	**getBuscarProveedorXRut**(int rutProveedor) Devuelve un proveedor de acuerdo al rut ingresado por parámetro Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
SectorDVO	**getBuscarSector**(int idSector) Busca un sector en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
SectorDVO[]	**getBuscarSectorPorArea**(int idArea) Busca un sector de acuerdo al área Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
TipoProductoDVO	**getBuscarTipoProducto**(int idTipoProducto) Devuelve un tipo de producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO	**getBuscarUsuario**(int rutUsuario) Busca un usuario en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO	**getBuscarUsuarioPorId**(int idUsuario) Busca un usuario en la base de datos por su ID Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial p.id_perfil
UsuarioDVO	**getBuscarUsuarioPorPedido**(int idPedido) Devuelve el usuario que esté a cargo de un determinado pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getbuscarUsuarioPorPerfil**(java.lang.String nombrePerfil) Devuelve el lisado de usuarios activos con un determinado perfil Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getBuscarUsuarioXNombre**(java.lang.String nombreUsuario) Busca un usuario en la base de datos por su nombre Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO	**getFichaUsuario**(java.lang.String username) Obtiene la información de un Usuario Sabcity Registro de Versiones: 09/05/2014 [Oscar - UBB]: Versión Inicial
int	**getIdProductoFinal**(java.lang.String nombreProductoFinal) Devuelve idProductoFinal Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
AreaDVO[]	**getListaAreas**() Devuelve el listado de áreas ingresadas en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
OrdenCompraDVO[]	**getListadoOrdenCompra**() Retorna un listado de órdenes de compra Registro de Versiones:

	24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoFinalDVO[]	**getListadoProductoCarta**() Devuelve el listado de productos que pertenecen a la carta Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getListaMesas**() Devuelve un arreglo de mesas Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
MesaDVO[]	**getListaMesasByEstado**(int estadoOrden) Devuelve un arreglo de mesas por estado de orden Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PedidoDVO[]	**getListaPedidos**() Devuelve arreglo de pedidos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
PerfilDVO[]	**getListaPerfiles**() Devuelve la lista de perfiles Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProductoDVO[]	**getListaProductos**() Devuelve un arreglo de productos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
ProveedorDVO[]	**getListaProveedoresActivos**() Retorna Lista de proveedores activos Registro de Versiones: 24/01/2014 [Oscar - UBB]: Versión Inicial
ProveedorDVO[]	**getListaProveedoresAll**() Retorna todos los proveedores Registro de Versiones: 24/01/2014 [Oscar - UBB]: Versión Inicial
SectorDVO[]	**getListaSectores**() Devuelve un arreglo de sectores Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
TipoProductoDVO[]	**getListaTipoProductos**() Devuelve todos los tipo de productos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getListaUsuarios**() Devuelve un arreglo de usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
UsuarioDVO[]	**getListaUsuariosActivos**() Devuelve un listado de usuarios activos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
EstadoDVO	**getObtenerEstado**(int idEstado) Retorna un estado para los pedidos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
int	**getObtenerIdOrdenCompra**(double numOrdenCompra) Devuelve el id de una orden de compra Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial

int	**getObtenerStockProducto**(int idProducto) Retorna el stock de un producto Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**ingresarRecibo**(int idOrdenRecibo, int idProveedor, int númeroOrdenRecibo) Se crea un recibo por ingresar un producto a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
double	**redondear**(double número, int decimales) redondea número Registro de Versiones: 25/02/2014 [Oscar - UBB]: Versión Inicial
long	**redondearEntero**(double número) redondea número a entero Registro de Versiones: 25/02/2014 [Oscar - UBB]: Versión Inicial
void	**setAsignarPerfil**(int idPerfil, int idUsuario) Asigna un perfil a un usuario Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setAsignarUsuarioMesa**(int idMesa, int idUsuario) Asigna un usuario a una mesa Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setCambiarEstado**(int idEstado, int idMesa, int idPedido) Cambia el estado del Pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarArea**(AreaDVO area) Ingresa una nueva área en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarMesa**(MesaDVO mesa) Ingresa una mesa en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarOrdenCompra**(OrdenCompraDVO ordenCompra) Genera una orden de compra Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarOrdenPedido**(OrdenPedidoDVO ordenPedido) Ingresa una orden de pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPedido**(PedidoDVO pedido) Ingresa un nuevo pedido Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPerfil**(PerfilDVO perfil) Ingresa un perfil en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarPreparacion**(PreparacionDVO preparacion) Se ingresa la preparación de un producto final a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial

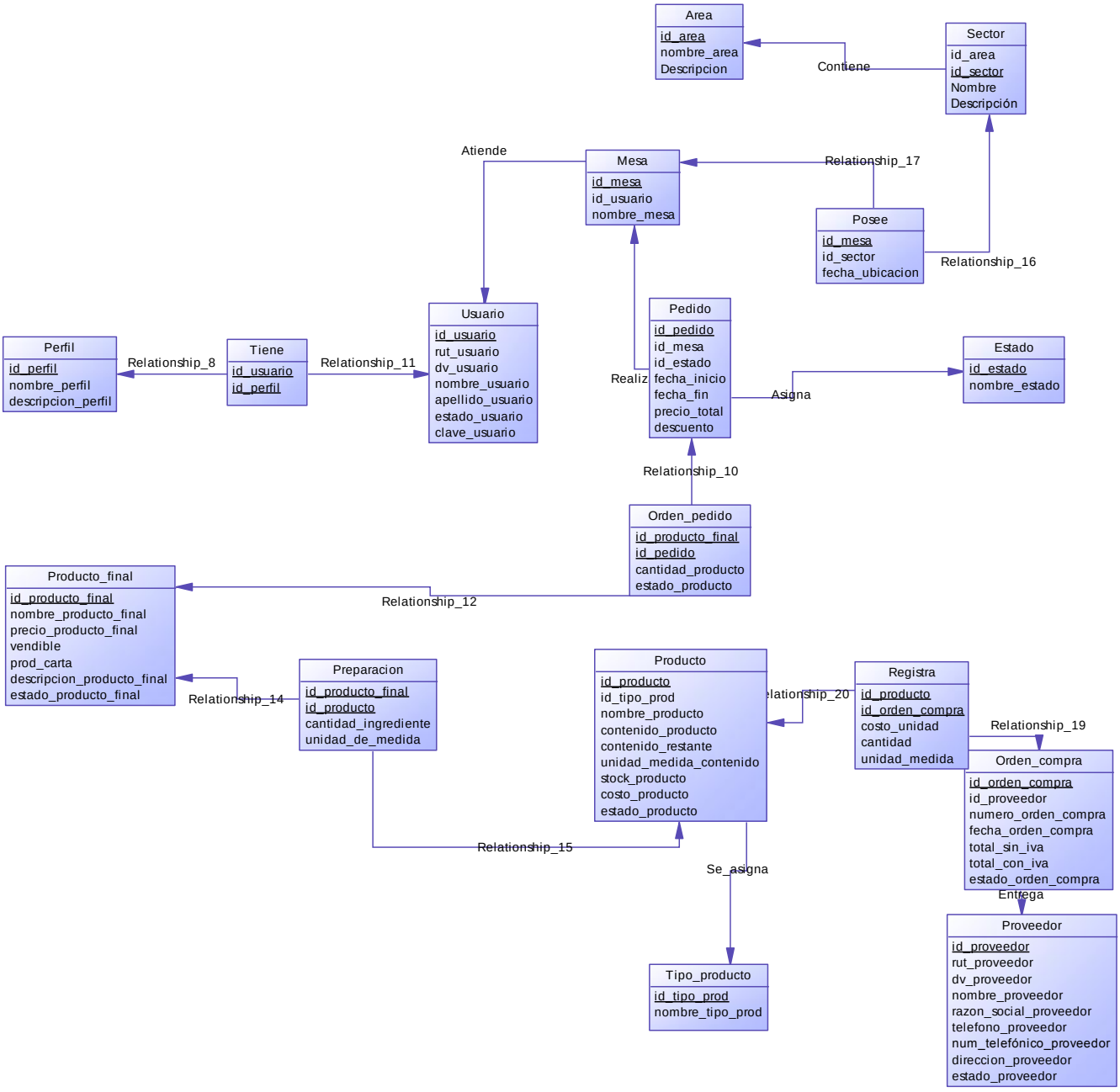
void	**setIngresarProducto**(ProductoDVO producto) Ingresa un producto nuevo en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarProductoFinal**(ProductoFinalDVO productoFinal) Se ingresa un producto final a la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarProveedor**(ProveedorDVO proveedor) Permite ingresar un nuevo proveedor al sistema Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarRegistra**(RegistraDVO registra) Se registra la cantidad y costo del producto comprado Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarSector**(SectorDVO sector) Ingresa un nuevo sector en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial
void	**setIngresarUsuario**(UsuarioDVO usuario) Ingresa un nuevo usuario en la base de datos Registro de Versiones: 24/01/2014 [Oscar - UBB.]: Versión Inicial

Table of Contents

19 PDM DIAGRAMS

19.1 Model level diagrams

19.1.1 Diagram Diagram_1



20 MODEL LEVEL OBJECT LISTS

20.1 Common Objects

20.1.1 List of diagrams

Name	Code
Diagram_1	DIAGRAM_1

20.1.2 List of extended attributes of the model modeloFisicoSistemaPubSabCity

Name	Data Type	Value	Target Name
AutoFixMaterializedViewDone	(Boolean)	true	Sybase SQL Anywhere 10

20.2 Physical diagrams objects

20.2.1 List of table columns

Name	Code
id_usuario	ID_USUARIO
rut_usuario	RUT_USUARIO
dv_usuario	DV_USUARIO
nombre_usuario	NOMBRE_USUARIO
apellido_usuario	APELLIDO_USUARIO
estado_usuario	ESTADO_USUARIO
clave_usuario	CLAVE_USUARIO
id_estado	ID_ESTADO
nombre_estado	NOMBRE_ESTADO
id_mesa	ID_MESA
id_usuario	ID_USUARIO
nombre_mesa	NOMBRE_MESA
id_pedido	ID_PEDIDO
id_mesa	ID_MESA
id_estado	ID_ESTADO
fecha_inicio	FECHA_INICIO
fecha_fin	FECHA_FIN
precio_total	PRECIO_TOTAL
descuento	DESCUENTO
id_area	ID_AREA
id_sector	ID_SECTOR
Nombre	NOMBRE
Descripción	DESCRIPCION
id_area	ID_AREA
nombre_area	NOMBRE_AREA
Descripcion	DESCRIPCION
id_producto	ID_PRODUCTO
id_tipo_prod	ID_TIPO_PROD
nombre_producto	NOMBRE_PRODUCTO
contenido_producto	CONTENIDO_PRODUCTO

contenido_restante	CONTENIDO_RESTANTE
unidad_medida_contenido	UNIDAD_MEDIDA_CONTENIDO
stock_producto	STOCK_PRODUCTO
costo_producto	COSTO_PRODUCTO
estado_producto	ESTADO_PRODUCTO
id_perfil	ID_PERFIL
nombre_perfil	NOMBRE_PERFIL
descripcion_perfil	DESCRIPCION_PERFIL
id_usuario	ID_USUARIO
id_perfil	ID_PERFIL
id_producto_final	ID_PRODUCTO_FINAL
id_pedido	ID_PEDIDO
cantidad_producto	CANTIDAD_PRODUCTO
estado_producto	ESTADO_PRODUCTO
id_producto_final	ID_PRODUCTO_FINAL
nombre_producto_final	NOMBRE_PRODUCTO_FINAL
precio_producto_final	PRECIO_PRODUCTO_FINAL
vendible	VENDIBLE
prod_carta	PROD_CARTA
descripcion_producto_final	DESCRIPCION_PRODUCTO_FINAL
estado_producto_final	ESTADO_PRODUCTO_FINAL
id_producto	ID_PRODUCTO
id_orden_compra	ID_ORDEN_COMPRA
costo_unidad	COSTO_UNIDAD
cantidad	CANTIDAD
unidad_medida	UNIDAD_MEDIDA
id_orden_compra	ID_ORDEN_COMPRA
id_proveedor	ID_PROVEEDOR
numero_orden_compra	NUMERO_ORDEN_COMPRA
fecha_orden_compra	FECHA_ORDEN_COMPRA
total_sin_iva	TOTAL_SIN_IVA
total_con_iva	TOTAL_CON_IVA
estado_orden_compra	ESTADO_ORDEN_COMPRA
id_proveedor	ID_PROVEEDOR
rut_proveedor	RUT_PROVEEDOR
dv_proveedor	DV_PROVEEDOR
nombre_proveedor	NOMBRE_PROVEEDOR
razon_social_proveedor	RAZON_SOCIAL_PROVEEDOR
telefono_proveedor	TELEFONO_PROVEEDOR
num_telefónico_proveedor	NUM_TELEFONICO_PROVEEDOR
direccion_proveedor	DIRECCION_PROVEEDOR
estado_proveedor	ESTADO_PROVEEDOR
id_tipo_prod	ID_TIPO_PROD
nombre_tipo_prod	NOMBRE_TIPO_PROD
id_producto_final	ID_PRODUCTO_FINAL
id_producto	ID_PRODUCTO
cantidad_ingrediente	CANTIDAD_INGREDIENTE
unidad_de_medida	UNIDAD_DE_MEDIDA
id_mesa	ID_MESA
id_sector	ID_SECTOR

fecha_ubicacion	FECHA_UBICACION

20.2.2 List of table indexes

Name	Code	Unique	Cluster	Primary	Foreign Key	Alternate Key	Table
USUARIO_PK	USUARIO_PK	X		X			Usuario
ESTADO_PK	ESTADO_PK	X		X			Estado
MESA_PK	MESA_PK	X		X			Mesa
ATIENDE_FK	ATIENDE_FK						Mesa
PEDIDO_PK	PEDIDO_PK	X		X			Pedido
REALIZA_FK	REALIZA_FK				X		Pedido
ASIGNA_FK	ASIGNA_FK				X		Pedido
CONTIENE_FK	CONTIENE_FK				X		Sector
AREA_PK	AREA_PK	X		X			Area
PRODUCTO_INGREDIENTE_PK	PRODUCTO_INGREDIENTE_PK	X		X			Producto
SE_ASIGNA_FK	SE_ASIGNA_FK				X		Producto
PERFIL_PK	PERFIL_PK	X		X			Perfil
TIENE_PK	TIENE_PK	X		X			Tiene
RELATIONSHIP_8_FK	RELATIONSHIP_8_FK				X		Tiene
RELATIONSHIP_11_FK	RELATIONSHIP_11_FK				X		Tiene
ORDEN_PEDIDO_PK	ORDEN_PEDIDO_PK	X		X			Orden_pedido
RELATIONSHIP_10_FK	RELATIONSHIP_10_FK				X		Orden_pedido

RELATIONSHIP_12_FK	RELATIONSHIP_12_FK				X		Orden_pedido
PRODUCTO_FINAL_PK	PRODUCTO_FINAL_PK	X		X			Producto_final
RELATIONSHIP_18_PK	RELATIONSHIP_18_PK	X		X			Registra
RELATIONSHIP_19_FK	RELATIONSHIP_19_FK				X		Registra
RELATIONSHIP_20_FK	RELATIONSHIP_20_FK				X		Registra
ORDEN_RECIBO_PK	ORDEN_RECIBO_PK	X		X			Orden_compra
ENTREGA_FK	ENTREGA_FK				X		Orden_compra
PROVEEDOR_PK	PROVEEDOR_PK	X		X			Proveedor
TIPO_PRODUCTO_PK	TIPO_PRODUCTO_PK	X		X			Tipo_producto
USAR_EN_PREPARACION_PK	USAR_EN_PREPARACION_PK	X		X			Preparacion
RELATIONSHIP_14_FK	RELATIONSHIP_14_FK				X		Preparacion
RELATIONSHIP_15_FK	RELATIONSHIP_15_FK				X		Preparacion
POSEE_PK	POSEE_PK	X		X			Posee
RELATIONSHIP_16_FK	RELATIONSHIP_16_FK				X		Posee

20.2.3 List of table keys

Name	Code	Table
Identifier_1	IDENTIFIER_1	Usuario
Identifier_1	IDENTIFIER_1	Estado
Identifier_1	IDENTIFIER_1	Mesa
Identifier_1	IDENTIFIER_1	Pedido
Key_1	KEY_1	Sector
Identifier_1	IDENTIFIER_1	Area
Identifier_1	IDENTIFIER_1	Producto
Identifier_1	IDENTIFIER_1	Perfil
Identifier_1	IDENTIFIER_1	Tiene
Identifier_1	IDENTIFIER_1	Orden_pedido

Identifier_1	IDENTIFIER_1	Producto_final
Identifier_1	IDENTIFIER_1	Registra
Identifier_1	IDENTIFIER_1	Orden_compra
Identifier_1	IDENTIFIER_1	Proveedor
Identifier_1	IDENTIFIER_1	Tipo_producto
Identifier_1	IDENTIFIER_1	Preparacion
Identifier_1	IDENTIFIER_1	Posee

20.2.4 List of references

Name	Code	Parent Table	Child Table
Asigna	ASIGNA	Estado	Pedido
Atiende	ATIENDE	Usuario	Mesa
Contiene	CONTIENE	Area	Sector
Entrega	ENTREGA	Proveedor	Orden_compra
Realiza	REALIZA	Mesa	Pedido
Relationship_8	RELATIONSHIP_8	Perfil	Tiene
Relationship_10	RELATIONSHIP_10	Pedido	Orden_pedido
Relationship_11	RELATIONSHIP_11	Usuario	Tiene
Relationship_12	RELATIONSHIP_12	Producto_final	Orden_pedido
Relationship_14	RELATIONSHIP_14	Producto_final	Preparacion
Relationship_15	RELATIONSHIP_15	Producto	Preparacion
Relationship_16	RELATIONSHIP_16	Sector	Posee
Relationship_17	RELATIONSHIP_17	Mesa	Posee
Relationship_19	RELATIONSHIP_19	Orden_compra	Registra
Relationship_20	RELATIONSHIP_20	Producto	Registra
Se_asigna	SE_ASIGNA	Tipo_producto	Producto

20.2.5 List of tables

Name	Code
Area	AREA
Estado	ESTADO
Mesa	MESA
Orden_compra	ORDEN_COMPRA
Orden_pedido	ORDEN_PEDIDO
Pedido	PEDIDO
Perfil	PERFIL
Posee	POSEE
Preparacion	PREPARACION
Producto	PRODUCTO
Producto_final	PRODUCTO_FINAL

Proveedor	PROVEEDOR
Registra	REGISTRA
Sector	SECTOR
Tiene	TIENE
Tipo_producto	TIPO_PRODUCTO
Usuario	USUARIO

21 CONFIGURACIÓN BEA WEBLOGIC 8.1

21.1 Instalación del servidor Bea WebLogic 8.1

Los pasos a seguir en la instalación de un servidor de aplicaciones es la siguiente:

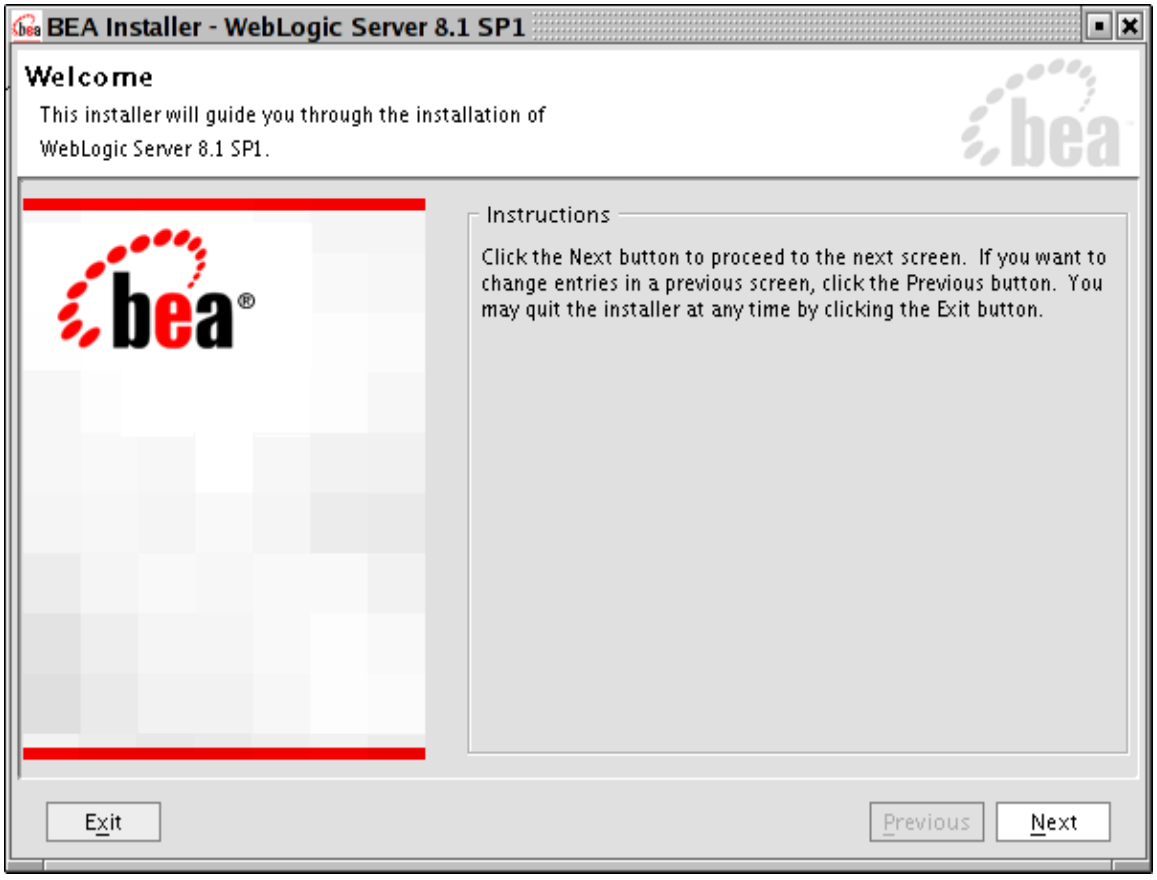
1. Instalación del software. Esta acción copia los ficheros necesarios y crea la estructura inicial de directorios.
2. Configuración de dominios. Debemos configurar el o los dominios necesarios y todos los componentes dentro de cada dominio (servidores, cluster, máquinas, etc.).

21.2 Instalación del servidor de aplicaciones

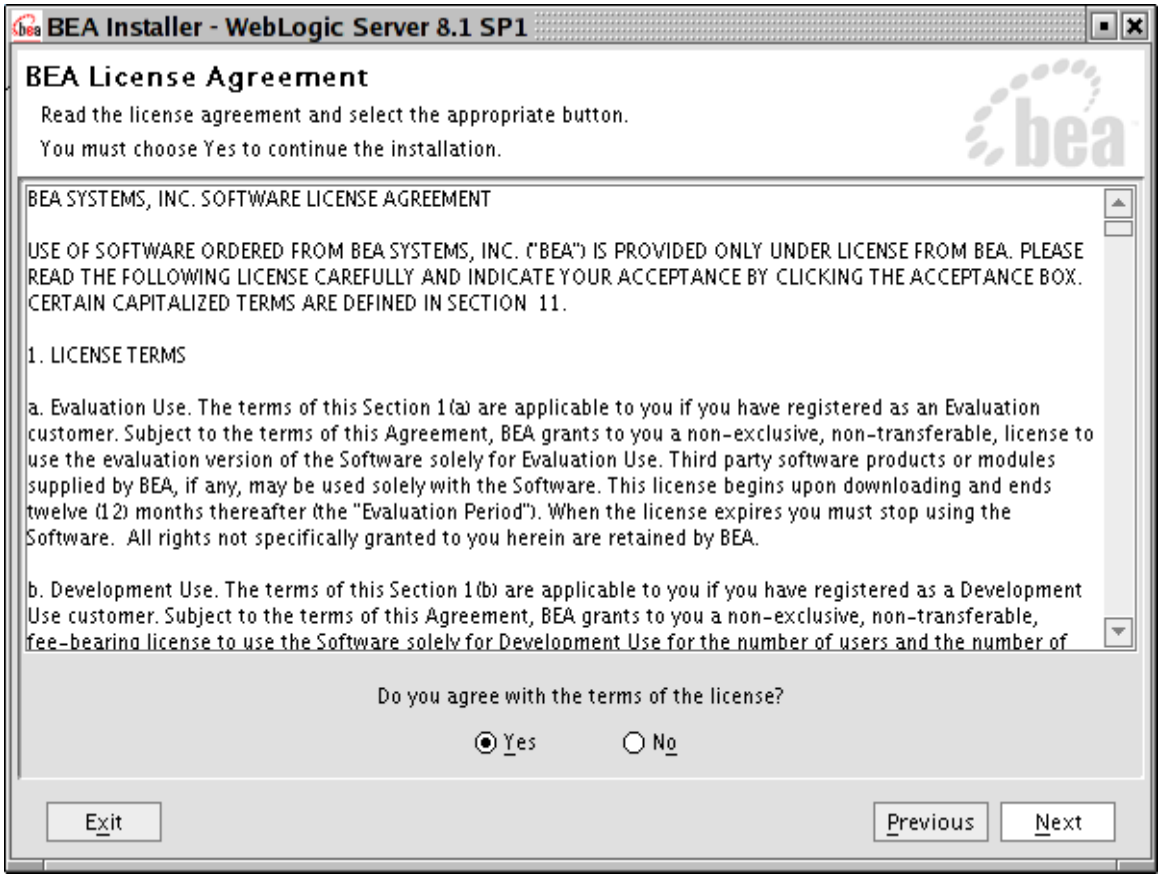
Vamos a instalar el servidor de aplicaciones Bea WebLogic. La instalación descrita aquí es para la versión 8.1 SP1 y bajo el sistema operativo Linux. Los requerimientos del sistema para la instalación de esta versión son:

- Memoria: 256Mb mínimo (512Mb aconsejable)
- Espacio en disco: 400Mb
- Versión de Java JDK 1.4.1 (se instala junto con el servidor) o superior. Podemos utilizar otra versión de Java, pero es aconsejable consultar la información que Bea muestra en http://e-docs.bea.com/wls/certifications/certifications/index.html para comprobar la compatibilidad entre versiones.

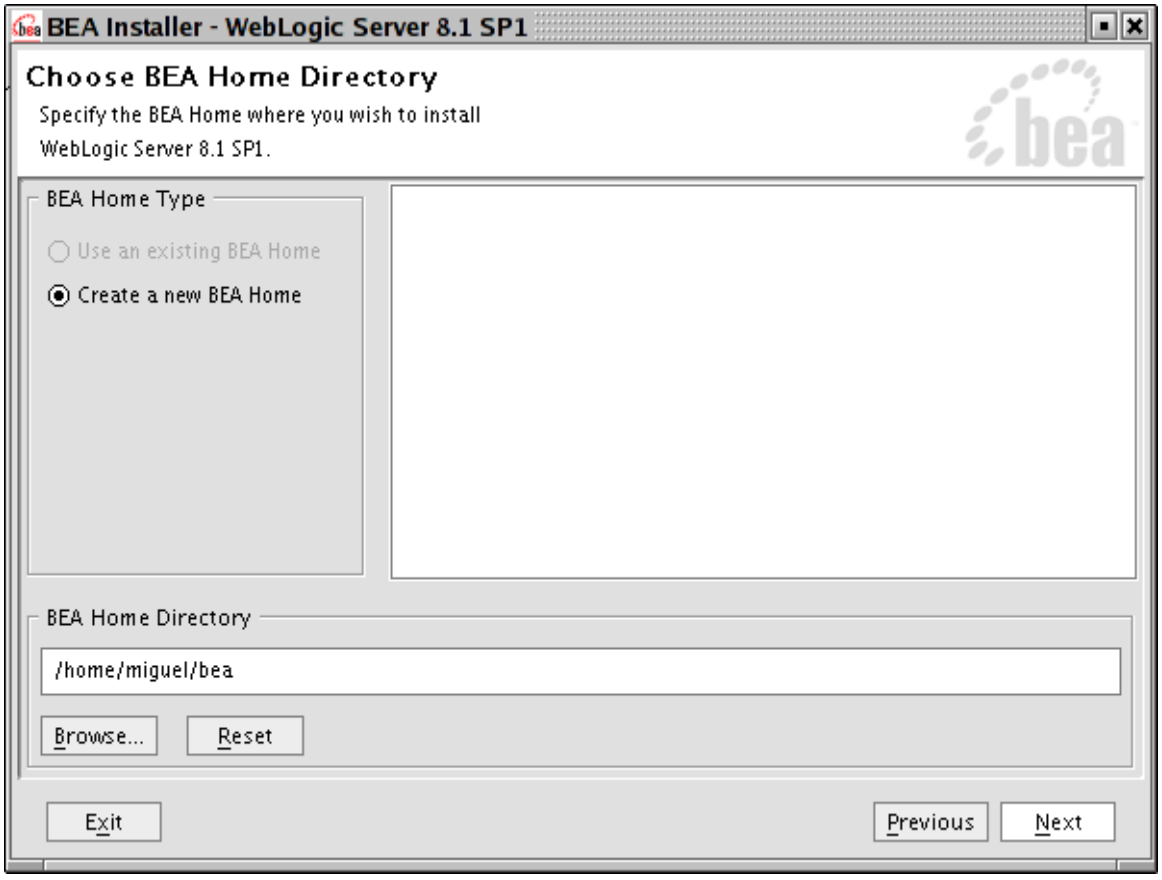
Ejecutamos el fichero *server811_linux32.bin* (no es necesario ser superusuario para instalar el servidor de aplicaciones). Esperamos hasta que nos aparezca la siguiente pantalla.



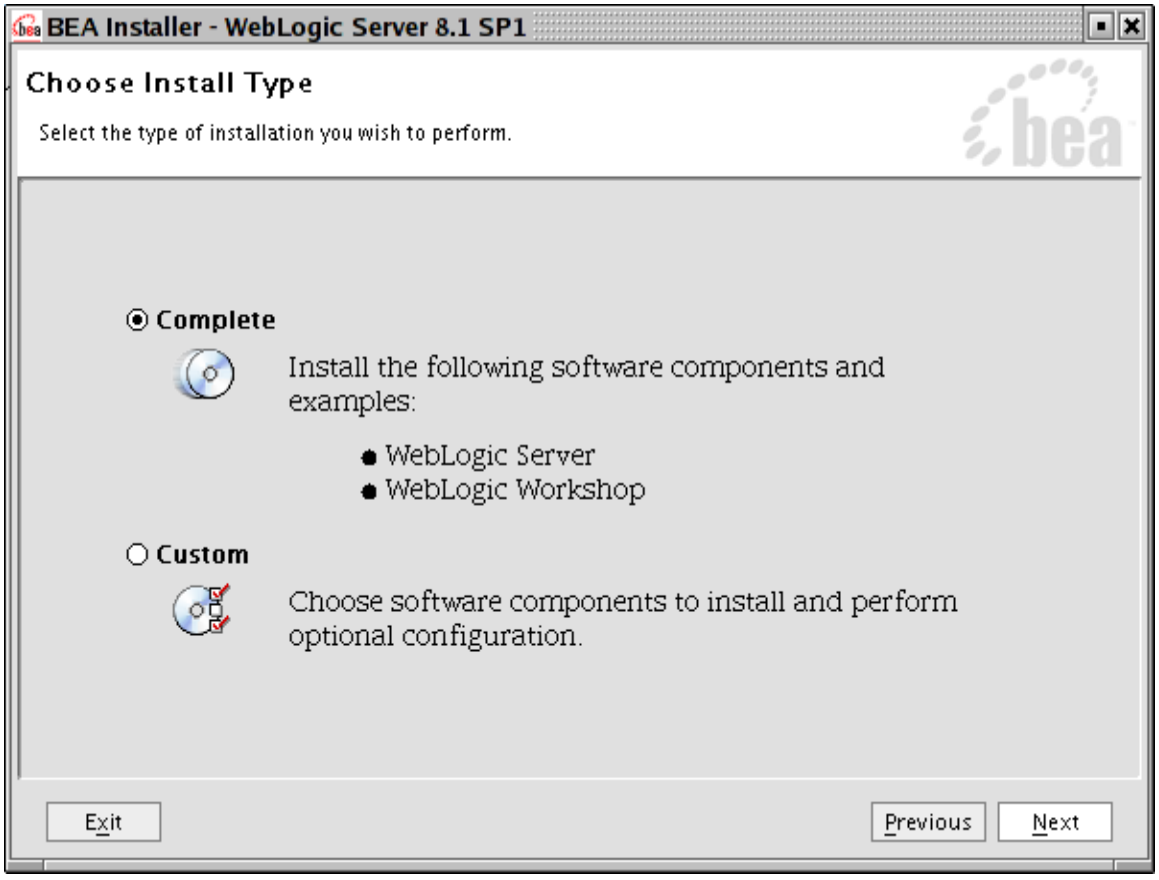
Nos aparecerá una ventana de licencia a la que decimos que sí y pasamos a la siguiente pantalla.



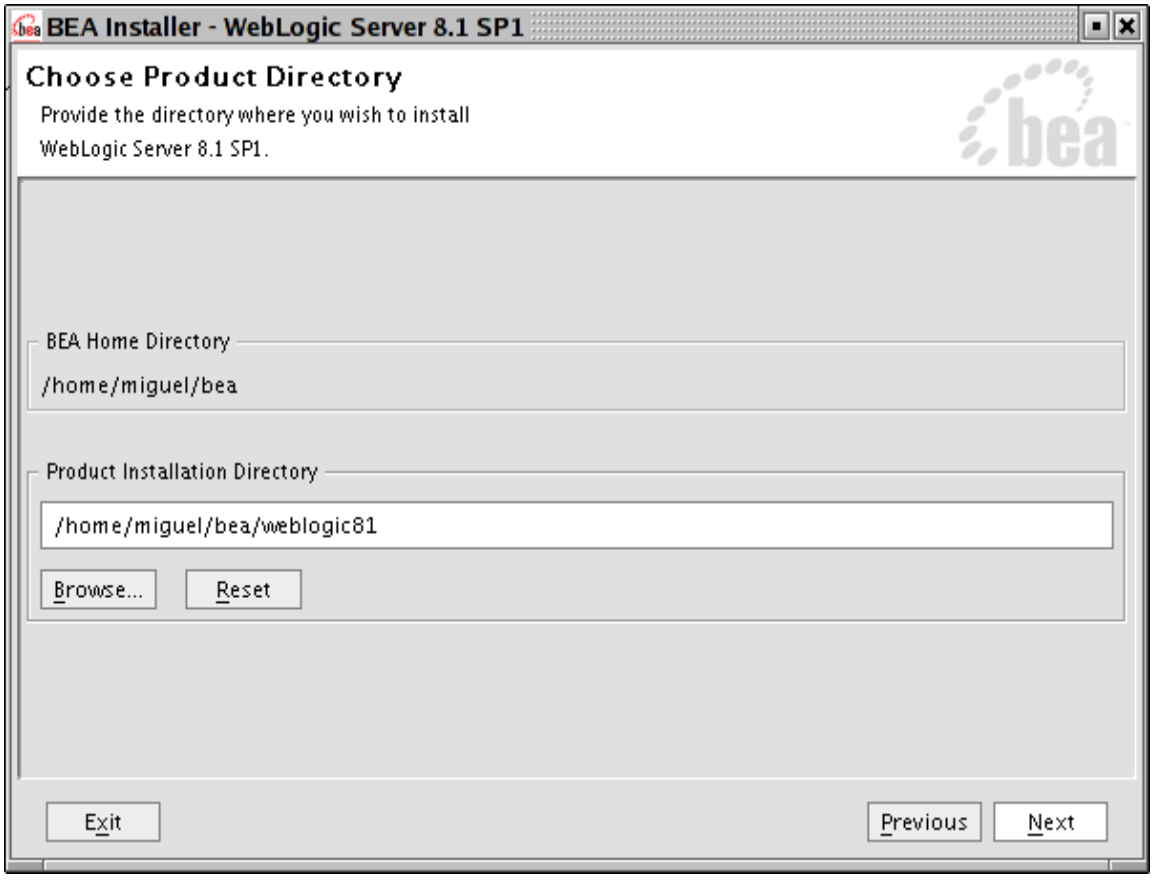
Si ya disponemos de un directorio creado lo podemos elegir de la lista. Si no, podemos dejar el mostrado por defecto, o definir uno distinto.



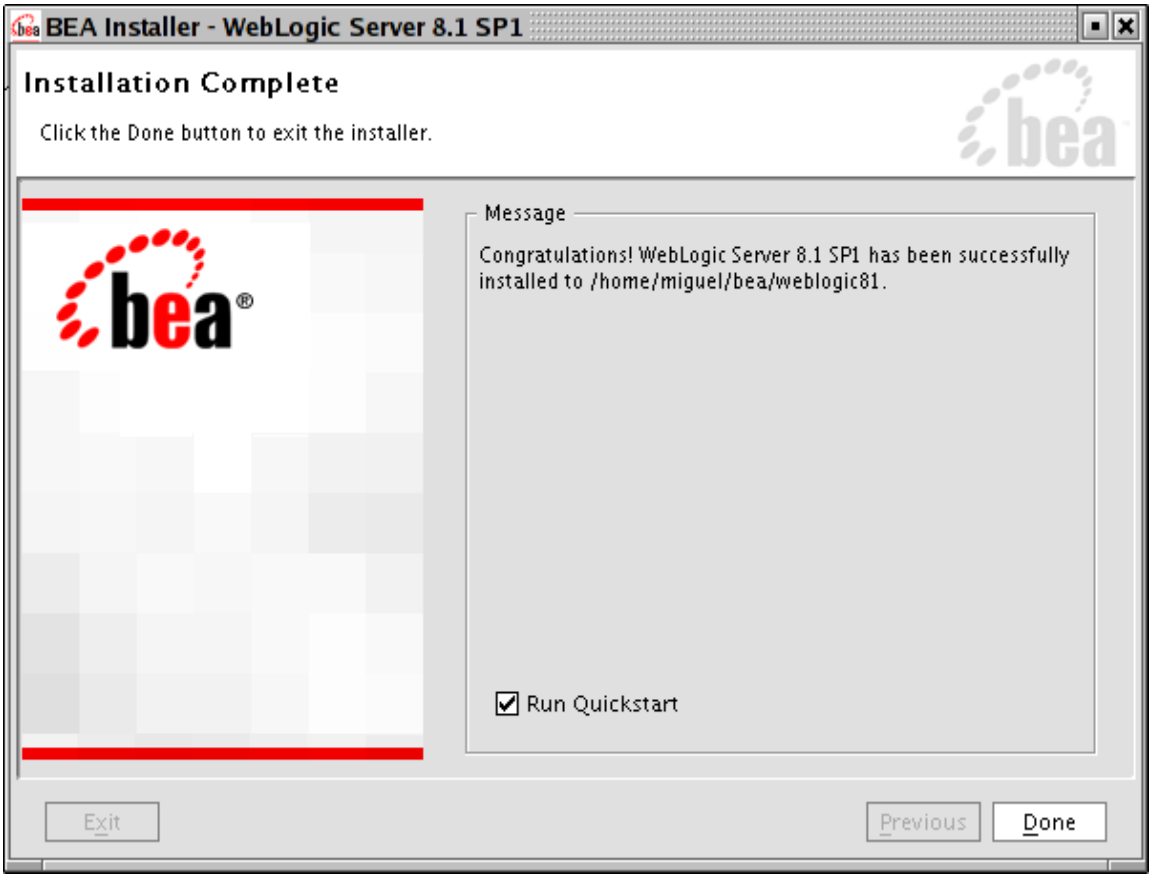
Ahora nos permite elegir entre realizar la instalación completa o bien elegir los elementos a instalar. Nosotros vamos a elegir la instalación completa.



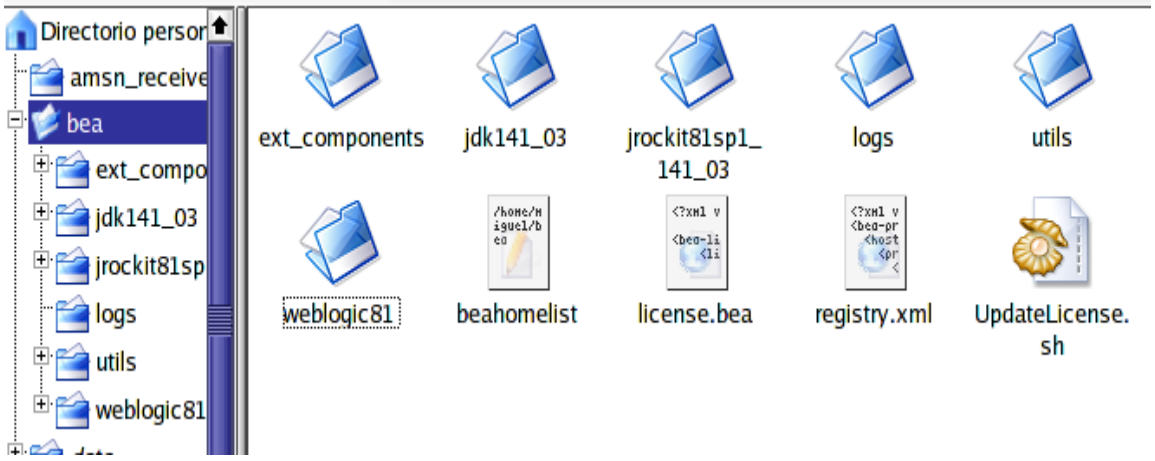
Nos queda elegir el directorio donde queremos que se instale el servidor de aplicaciones. Por defecto lo hace en el directorio *weblogic81*.



Después de los pasos anteriores empezará la instalación, que durará unos minutos. Cuando finalice nos aparecerá la ventana siguiente. Deseleccionamos la opción *Run QuickStart* y pinchamos en *Done.* Hemos finalizado la instalación del servidor.



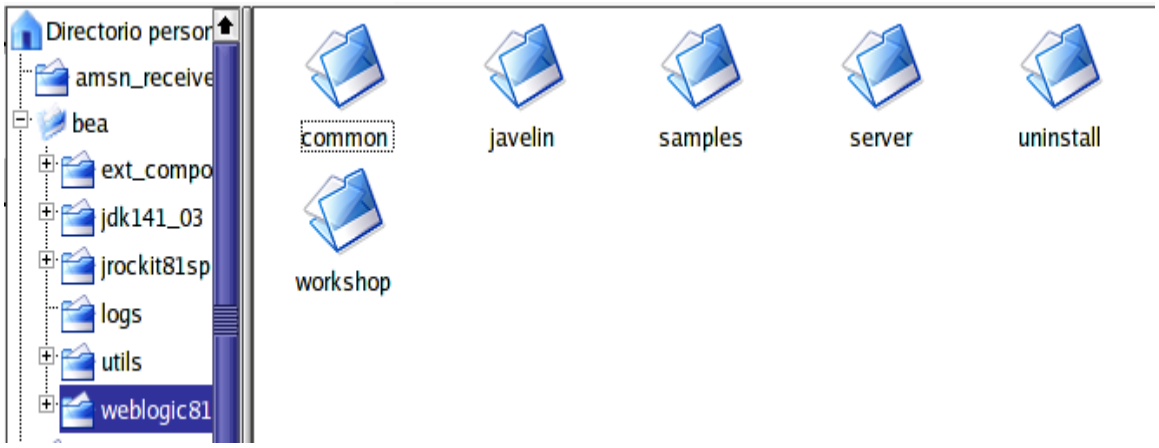
La estructura de directorios creada en la instalación es la siguiente:



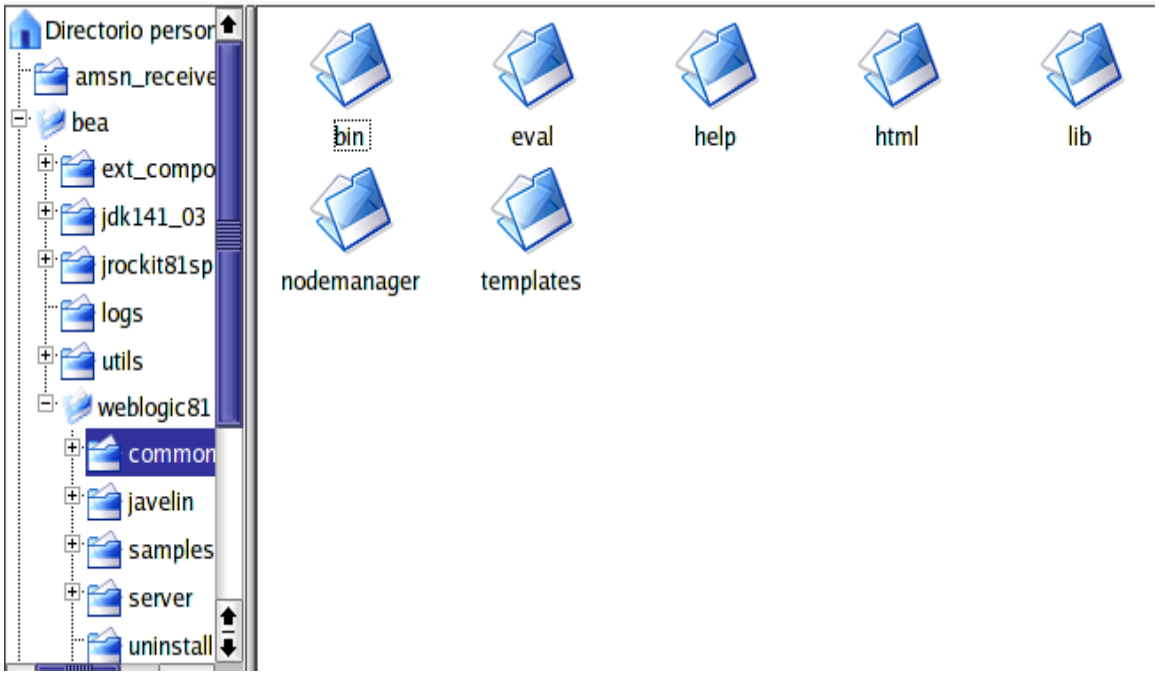
El directorio *jdk141_03* contiene la distribución 1.4.1 de J2SE de Sun. Si tenemos una versión actualizada de Java simplemente la añadiremos al CLASSPATH. En este punto debemos tener un cuidado especial y comprobar si la versión de Java es soportada por la versión del servidor de aplicaciones. Para comprobarlo visitar la página de Bea. El directorio de *logs* contiene el fichero log de instalación. El directorio *utils* contiene algunas utilidades que iremos viendo conforme las utilicemos. El siguiente directorio, *weblogic81*, es el que contiene todas las librerías, clases y herramientas adicionales para el funcionamiento de nuestro servidor. El fichero *license.bea* contiene la información de nuestra licencia en formato XML. Contendrá información de la fecha de expiración de la licencia, de qué características disponemos (número

de puestos, número de IPs, etc.), y toda la información necesaria para la ejecución del servidor. El ejecutable*UpdateLicense.sh* nos va a permitir actualizar una nueva licencia.

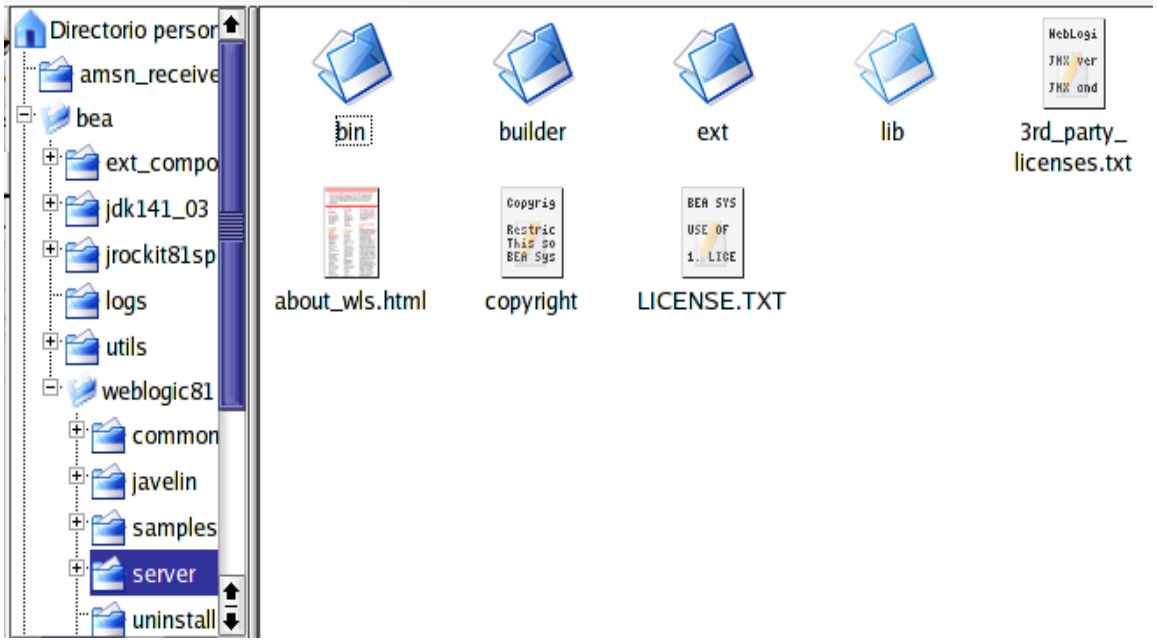
A su vez, el directorio *weblogic81* contiene los siguientes subdirectorios:



Nos interesan, de momento, el directorio *common* y el *server*. *Common* contiene los subdirectorios mostrados en la siguiente figura. En el directorio *bin* tenemos una herramienta para crear dominios. El directorio *nodemanager* contiene ficheros de configuración para el Node Manager.



El directorio *server* contiene datos y utilidades relacionadas con el servidor de aplicaciones. En el directorio *bin* tenemos varias aplicaciones y los *scripts* para arrancar el servidor de aplicaciones y el Node Manager. El ejecutable para arrancar un servidor que se crea en nuestro dominio llama a estos ejecutables. En otro directorio dentro de *server*, el subdirectorio *lib*, tenemos el fichero *weblogic.jar* que tendremos que incluir en el *classpath* cuando queramos realizar una aplicación que utilice los recursos de WebLogic. También disponemos en este directorio de los ficheros que gestionan las políticas de seguridad.



21.3 Arranque del dominio y consola de administración

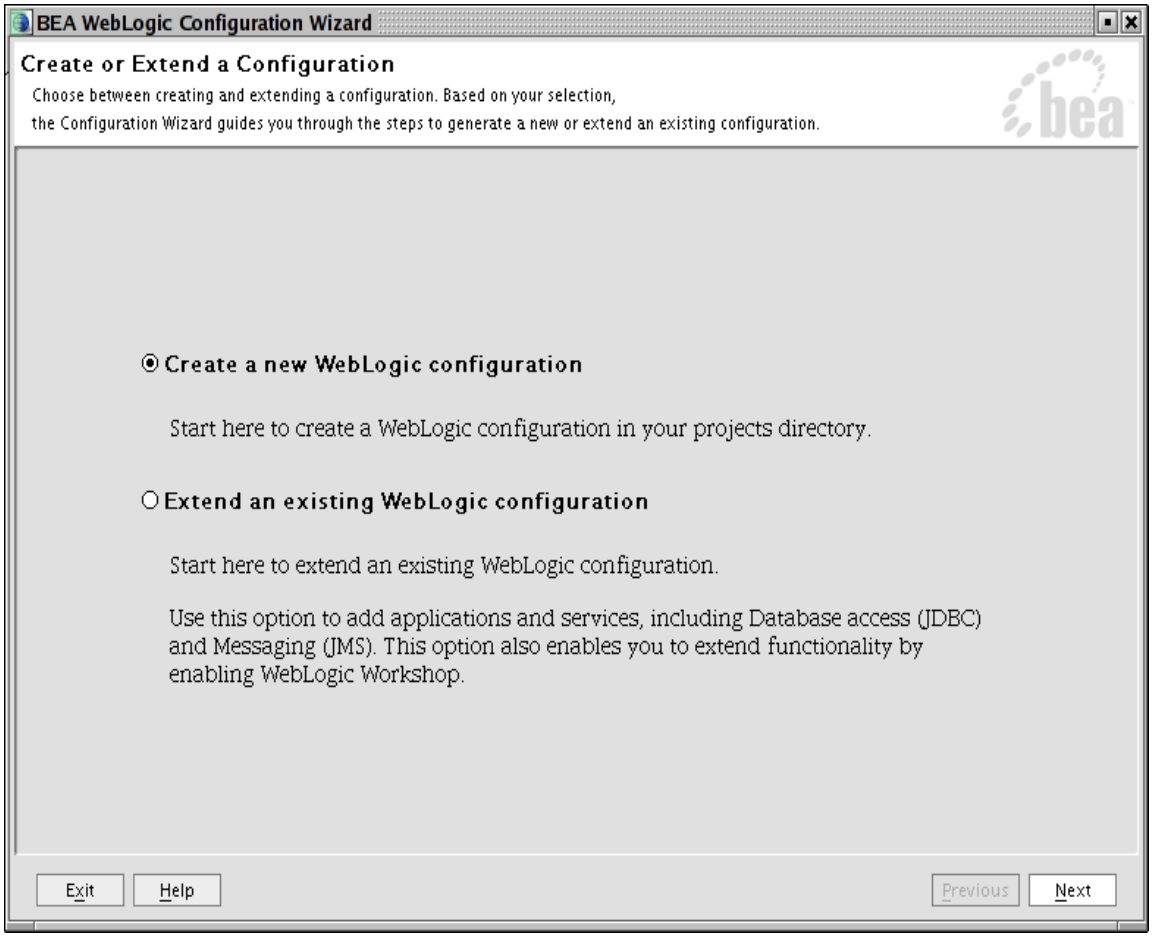
Antes de empezar a definir los elementos que soportan la ejecución del servidor de aplicaciones vamos a comentar algunos conceptos sobre los distintos tipos de servidores.

Como ya hemos comentado, nuestra principal unidad de trabajo es el dominio. El dominio no es más que una agrupación de todos los componentes que utilizamos para nuestro trabajo (servidores, máquinas, aplicaciones, etc.). Un ejemplo de uso de dominios es el siguiente. Cuando se desarrolla una aplicación se suele separar la fase de desarrollo de una aplicación con la fase de producción (cuando la aplicación ya está funcionando hacia el usuario y dando servicio). Para manejar esta situación podemos tener creados dos dominios, uno para desarrollo y otro para producción. A pesar de contener exactamente los mismos componentes funcionan de forma independiente.

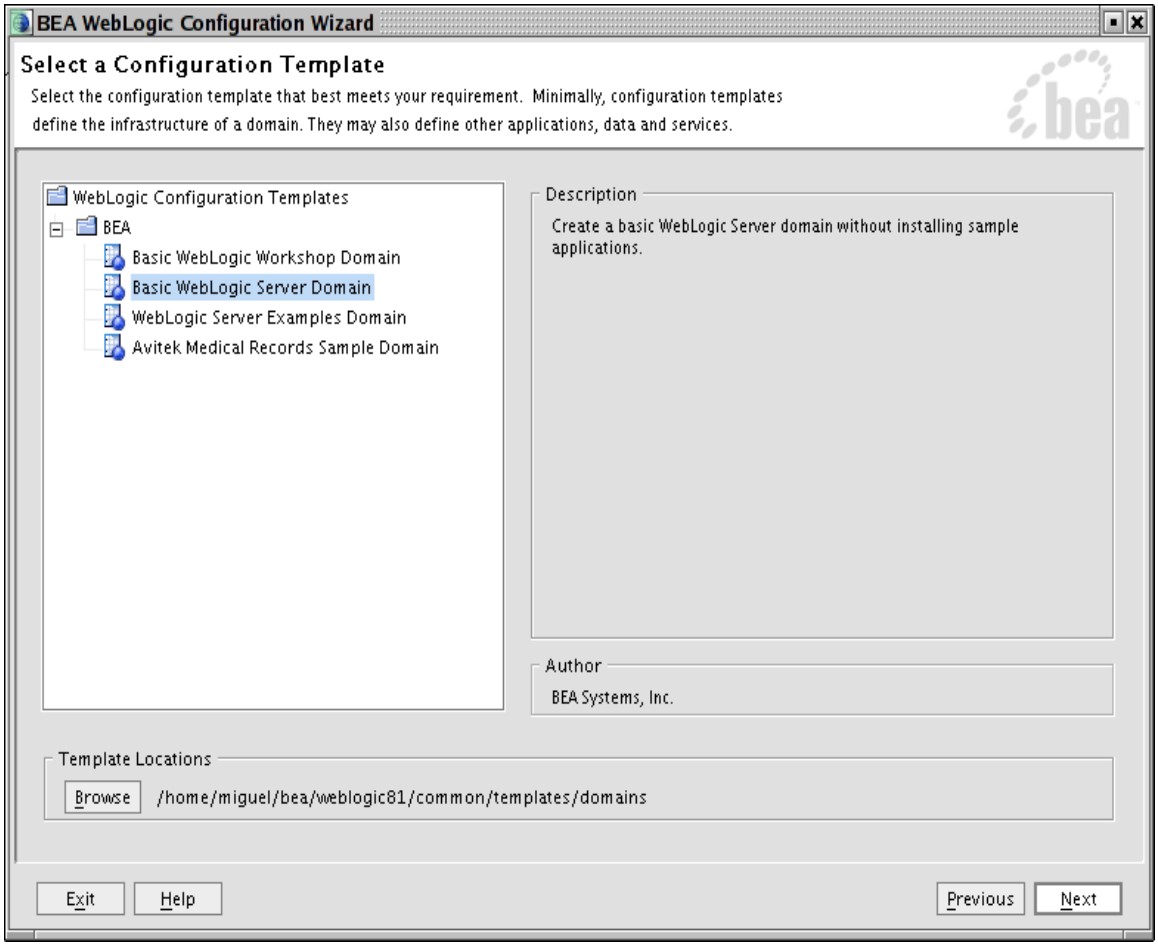
Dentro de un dominio vamos a tener máquinas y servidores. Al menos debemos tener un servidor en nuestro dominio, que llamaremos de *administración*. El servidor de administración es único en el dominio y va a realizar, como su nombre indica, tareas administrativas. Podemos tener más servidores, que llamaremos *administrados* (*managed*). De este tipo de servidor podemos tener tantos como queramos.

Vamos a empezar a crear nuestro primer dominio. Vamos a llamarlo *MiDominio* y contendrá dos servidores alojados en la misma máquina: *Servidor1* y *Servidor2*. El servidor 1 será el de administración. Utilizaremos un asistente que incorpora Weblogic para crear el dominio y los servidores. Nos situamos en $HOME_BEA/weblogic81/common/bin ($HOME_BEA es el directorio donde hemos instalado Weblogic, en mi máquina /home/miguel/bea) y ejecutamos ./config.sh También existe una herramienta llamada *QuickStart*, pero hemos notado que en Linux no funciona de forma correcta. En Windows aparece en el menú de WebLogic dentro del menú de programas.

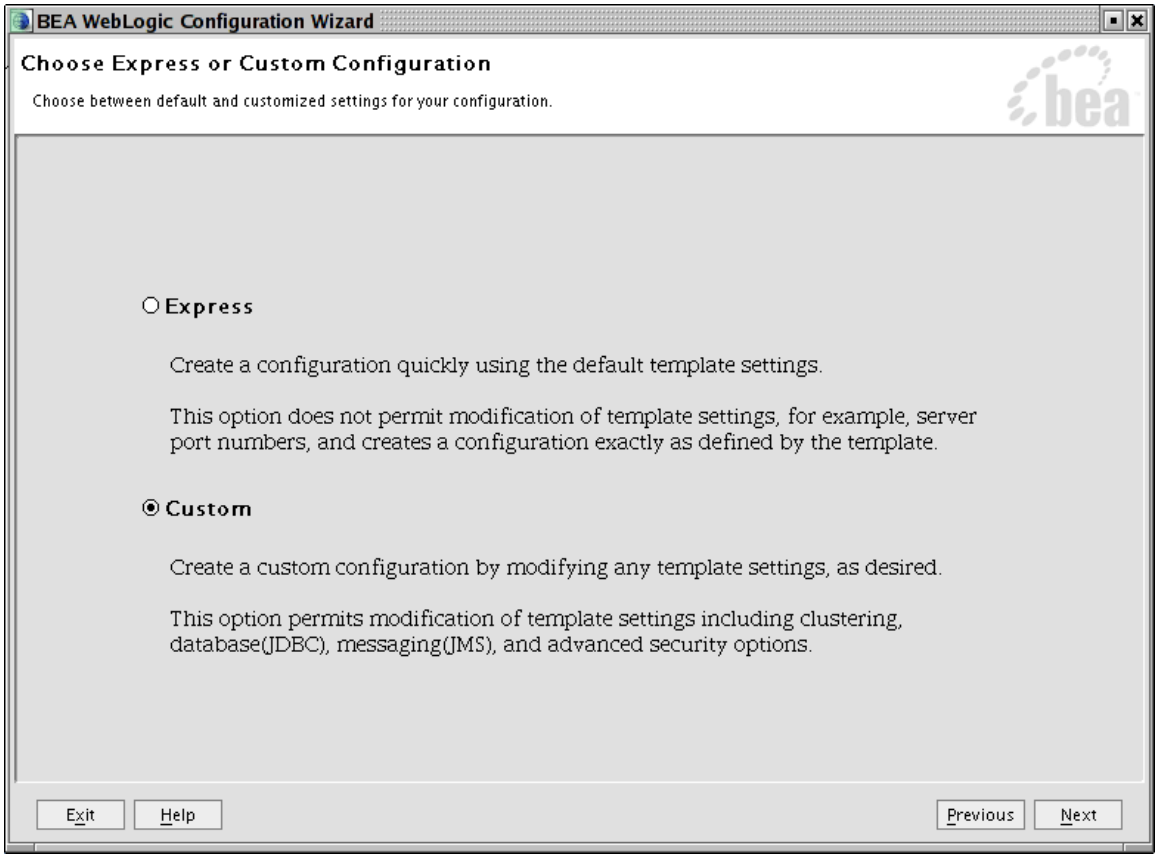
Nos aparecerá la ventana que se muestra en la siguiente figura, en la que podemos optar por crear una nueva configuración o extender (añadir nuevas características) a una existente. Vamos a seleccionar crear una nueva configuración. Pulsamos en el botón *Next.*



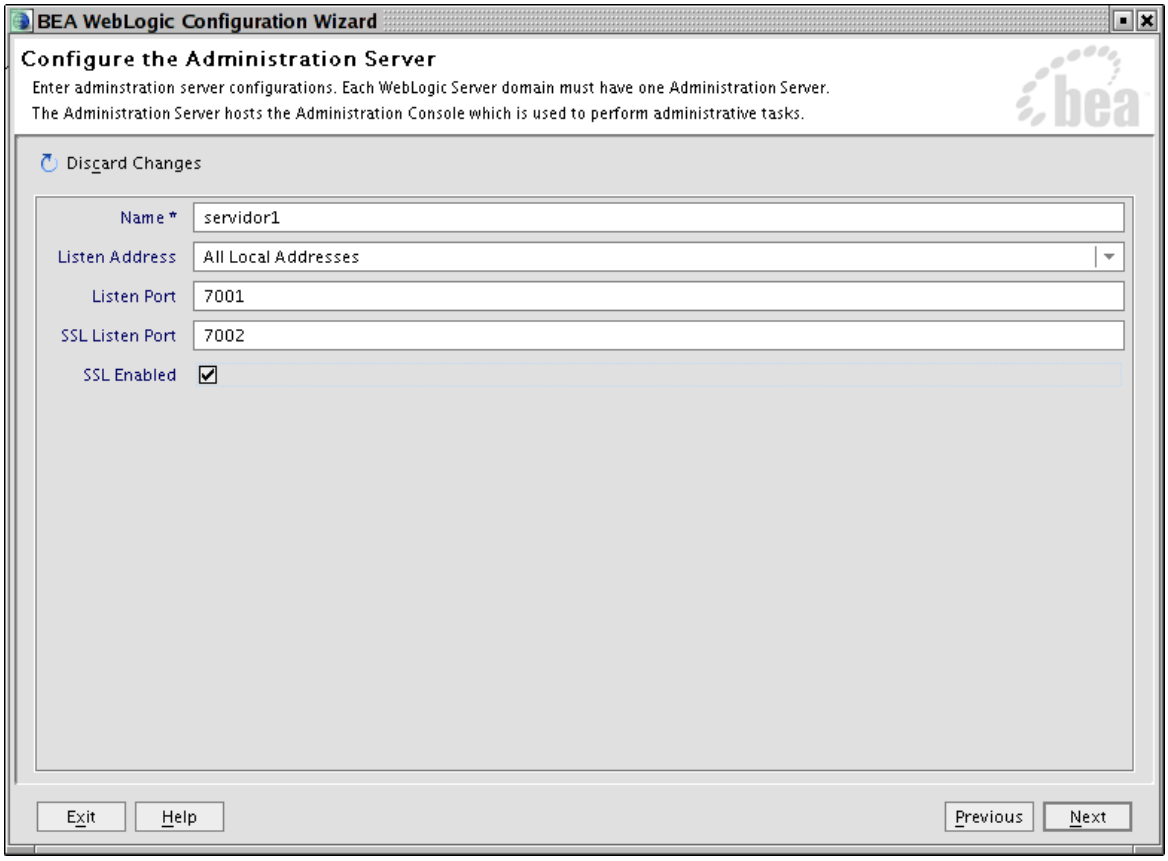
Ahora debemos seleccionar si queremos instalar un dominio con servidores, un dominio Workshop o un dominio con ejemplos. Nos interesa la opción seleccionada.



Pulsamos *Next* y nos deja elegir entre utilizar las opciones por defecto (no deja crear servidores adicionales) o definir nuestra configuración. Elegimos *Custom* para poder definir los servidores como queramos. Pulsamos *Next*.

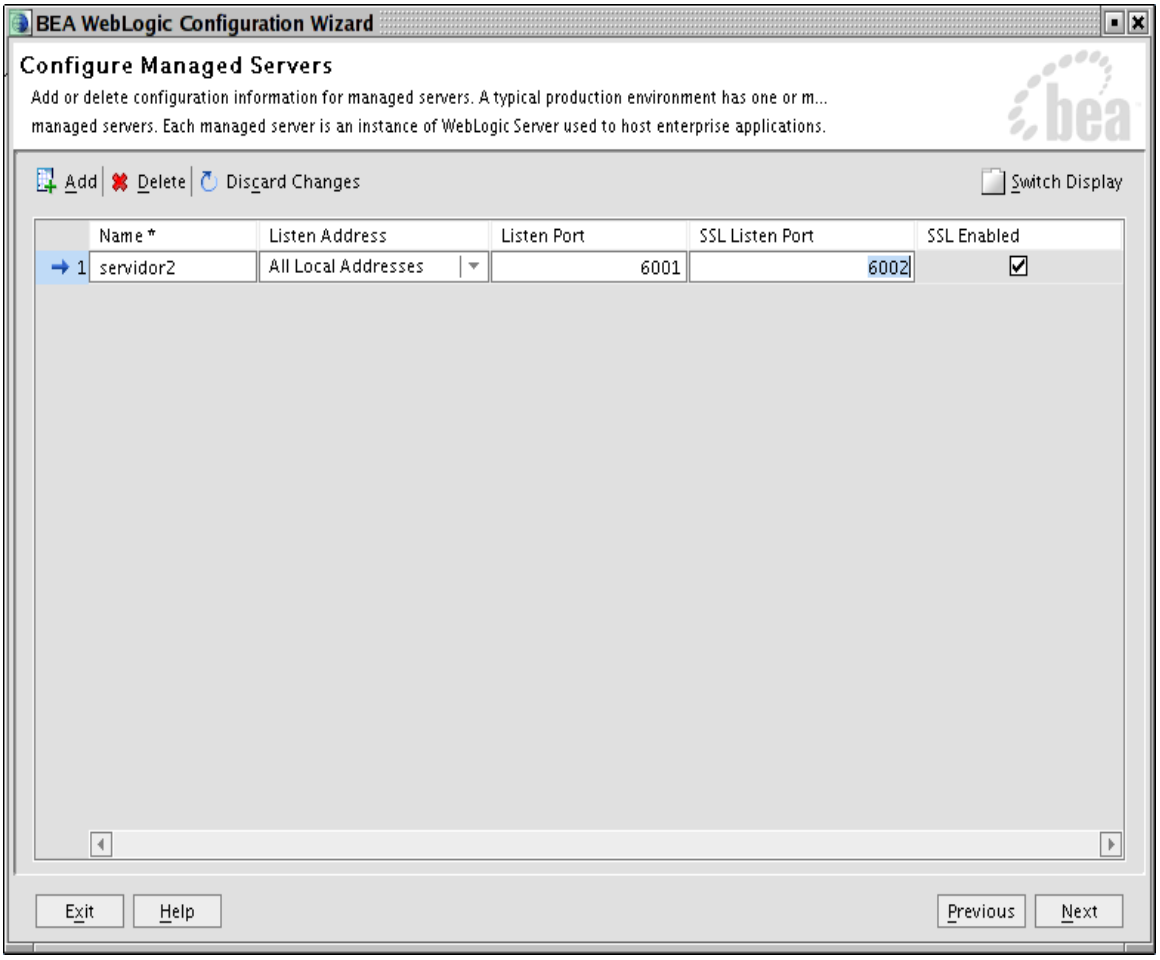


En la siguiente pantalla se nos pide que definamos el servidor de administración. Tenemos que definir el nombre del servidor (debe ser único en el dominio), en qué dirección (IP ó DNS) estará escuchando el servidor y los puertos de escucha (por defecto se suele dar el 7001 y el 7002 para el puerto seguro).

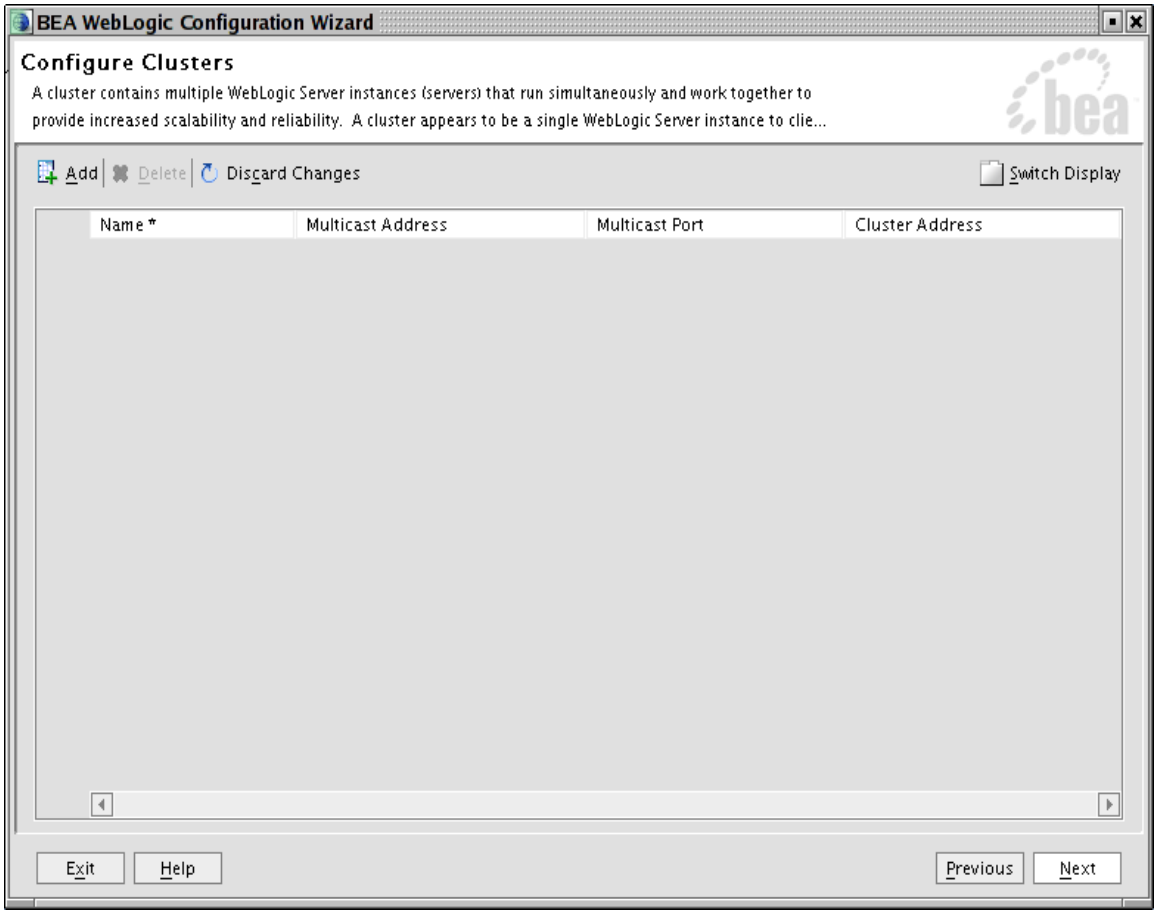


La siguiente pantalla nos da la opción de poder configurar servidores adicionales. Vamos a decirle que sí, para poder definir el otro servidor en nuestro sistema.

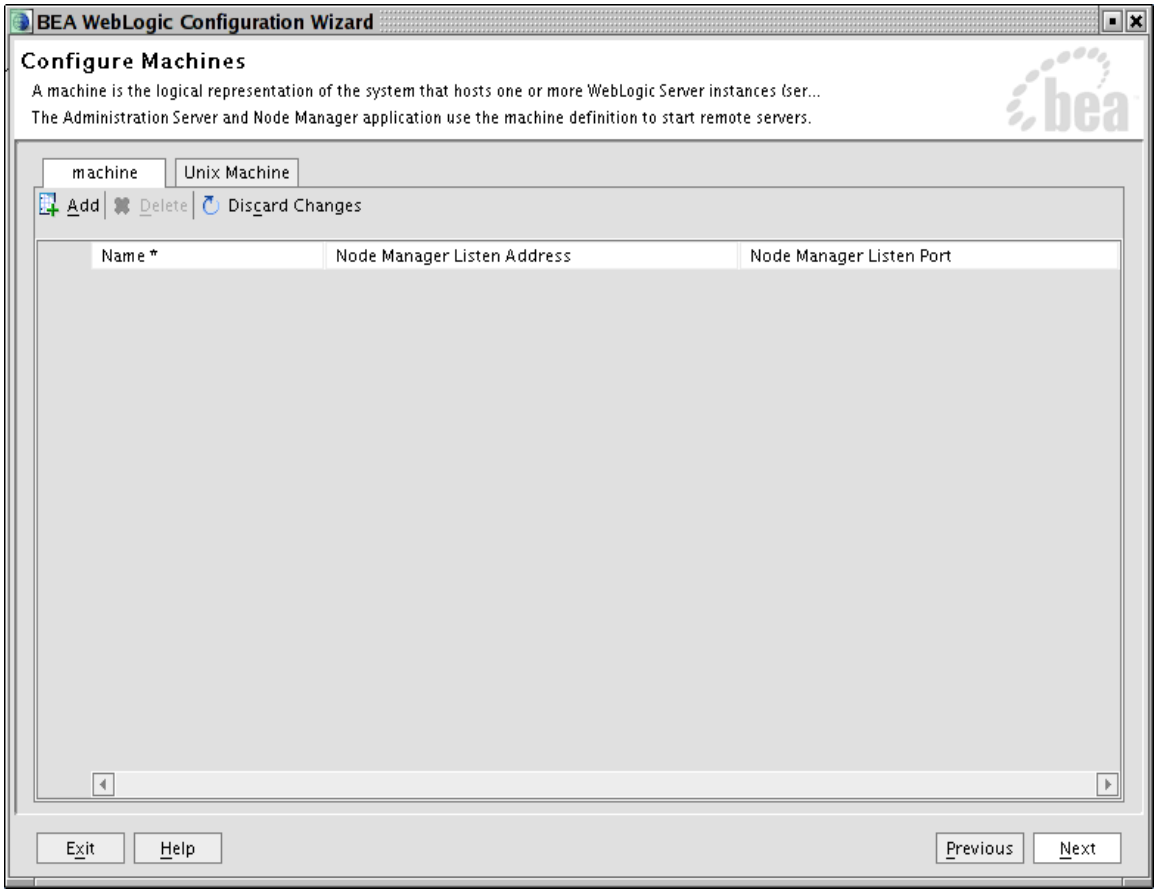
Al pinchar en *Next* nos aparece la ventana mostrada a continuación donde podemos definir nuevos servidores. Los botones *Add* y *Delete* sirven para añadir nuevos servidores o eliminarlos. Para cada nuevo servidor debemos definir las mismas opciones que dimos al servidor de administración. Un punto importante es que el puerto de escucha debe ser distinto para cada servidor que se ejecute en la misma máquina.



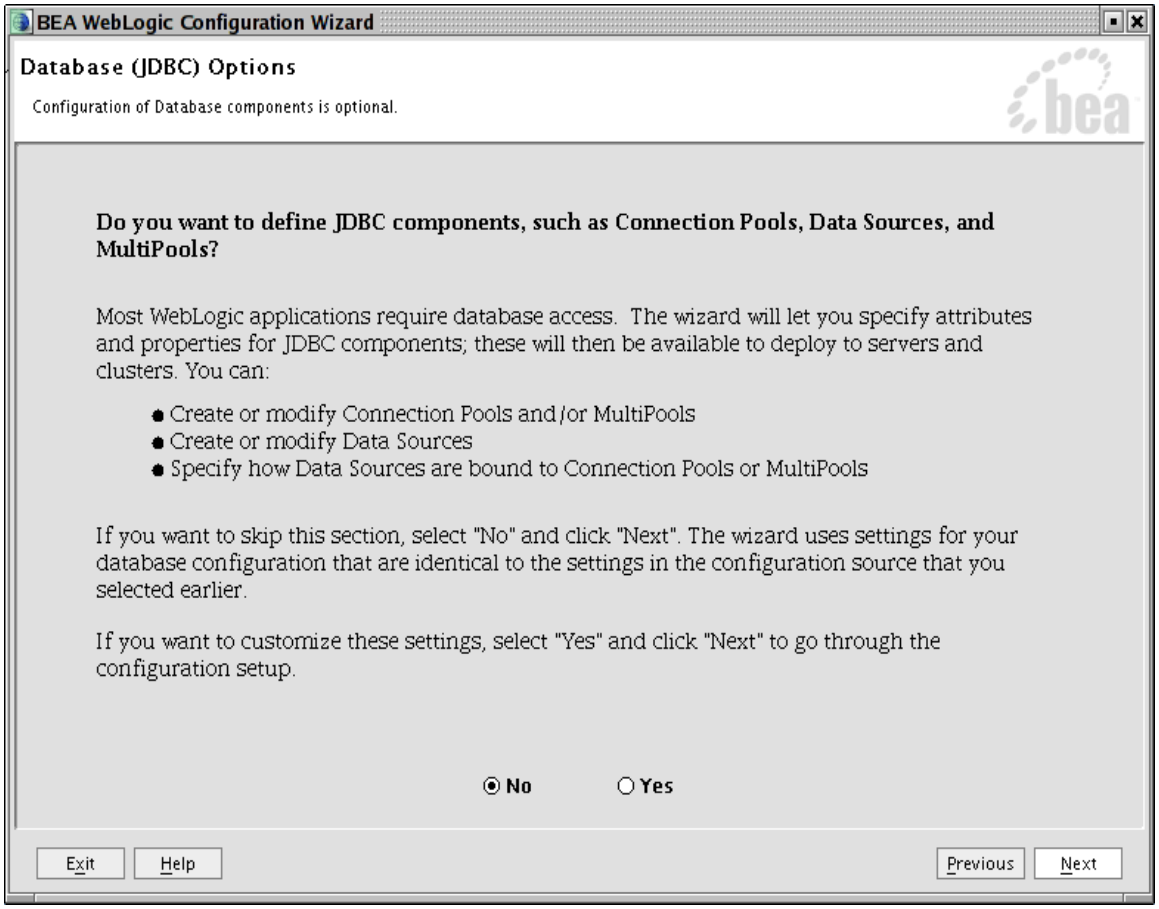
La siguiente pantalla nos permite definir un cluster. Veremos esta opción más adelante, de momento pasamos de pantalla.



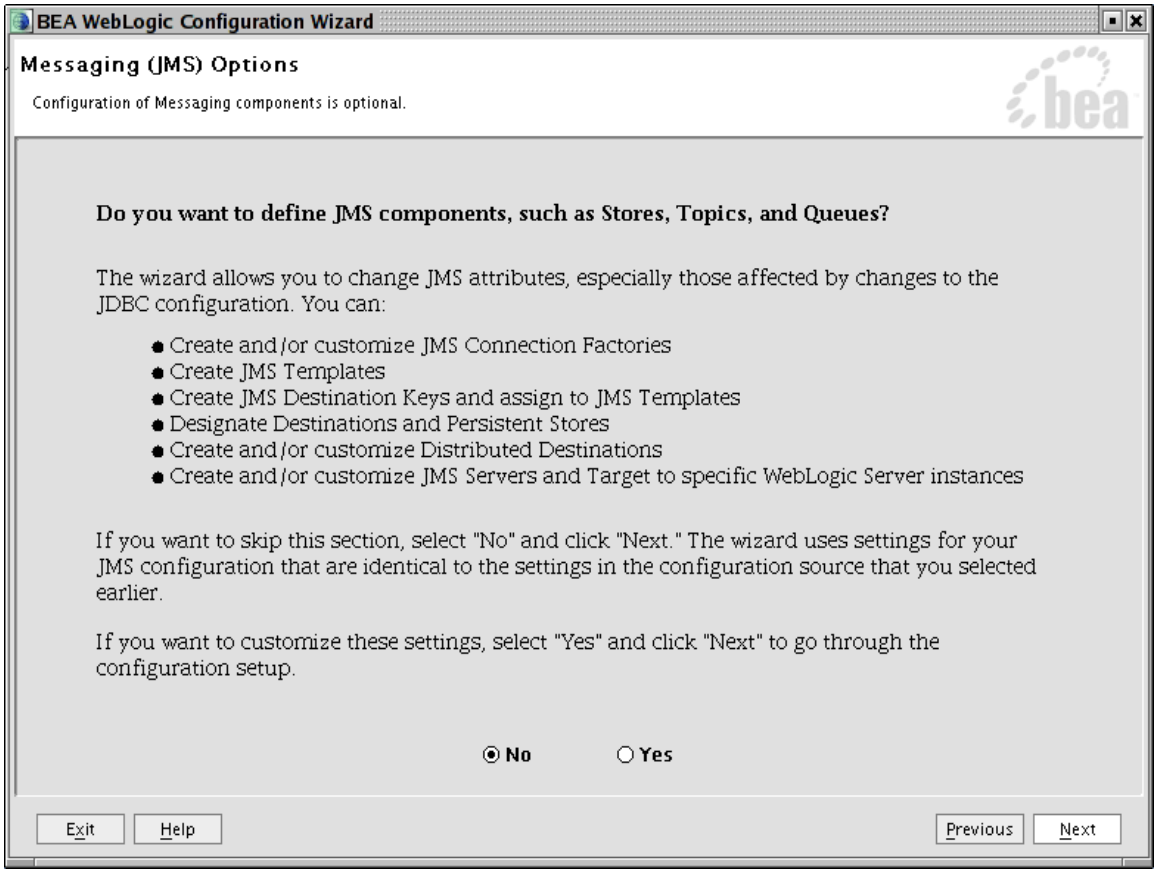
En esta se nos permite definir una máquina. También lo dejamos para más adelante.



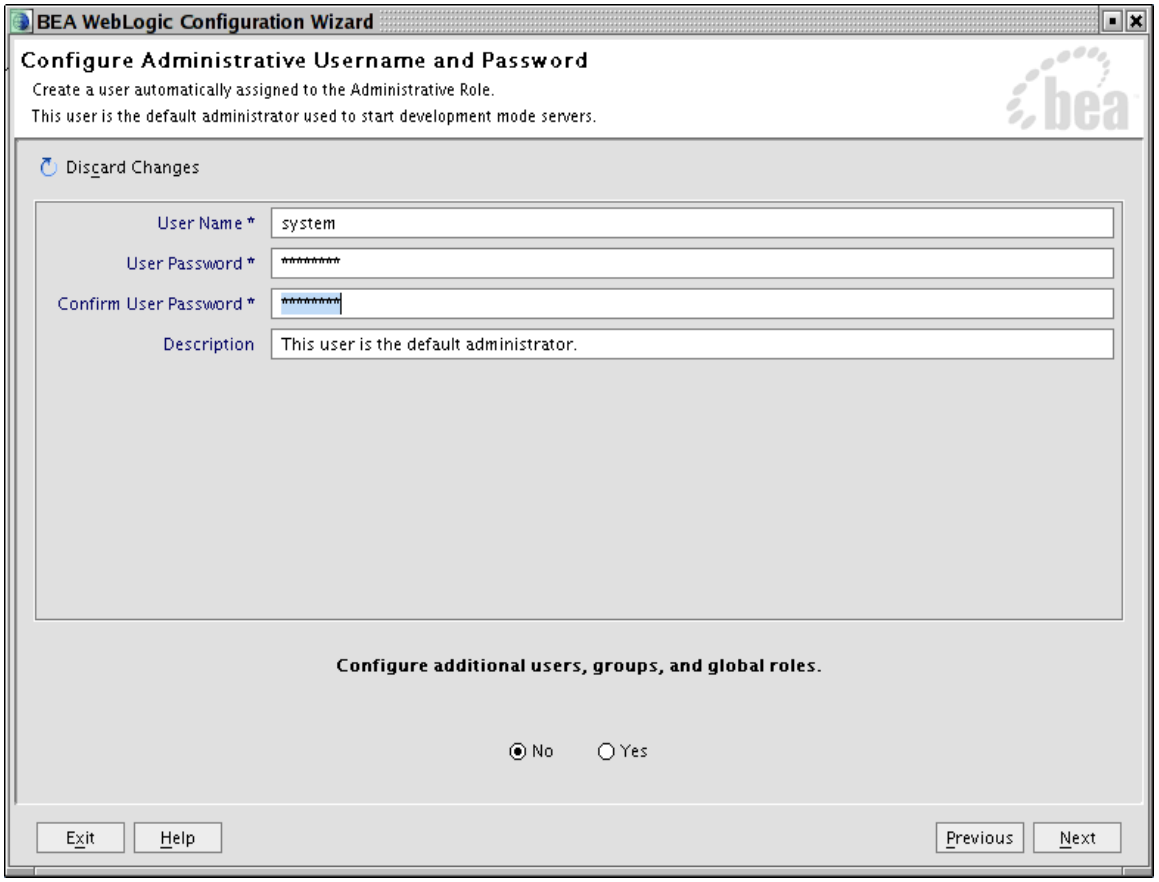
Ahora se nos da la opción de configurar componentes JDBC. Decimos que no, pues veremos estas opciones más adelante.



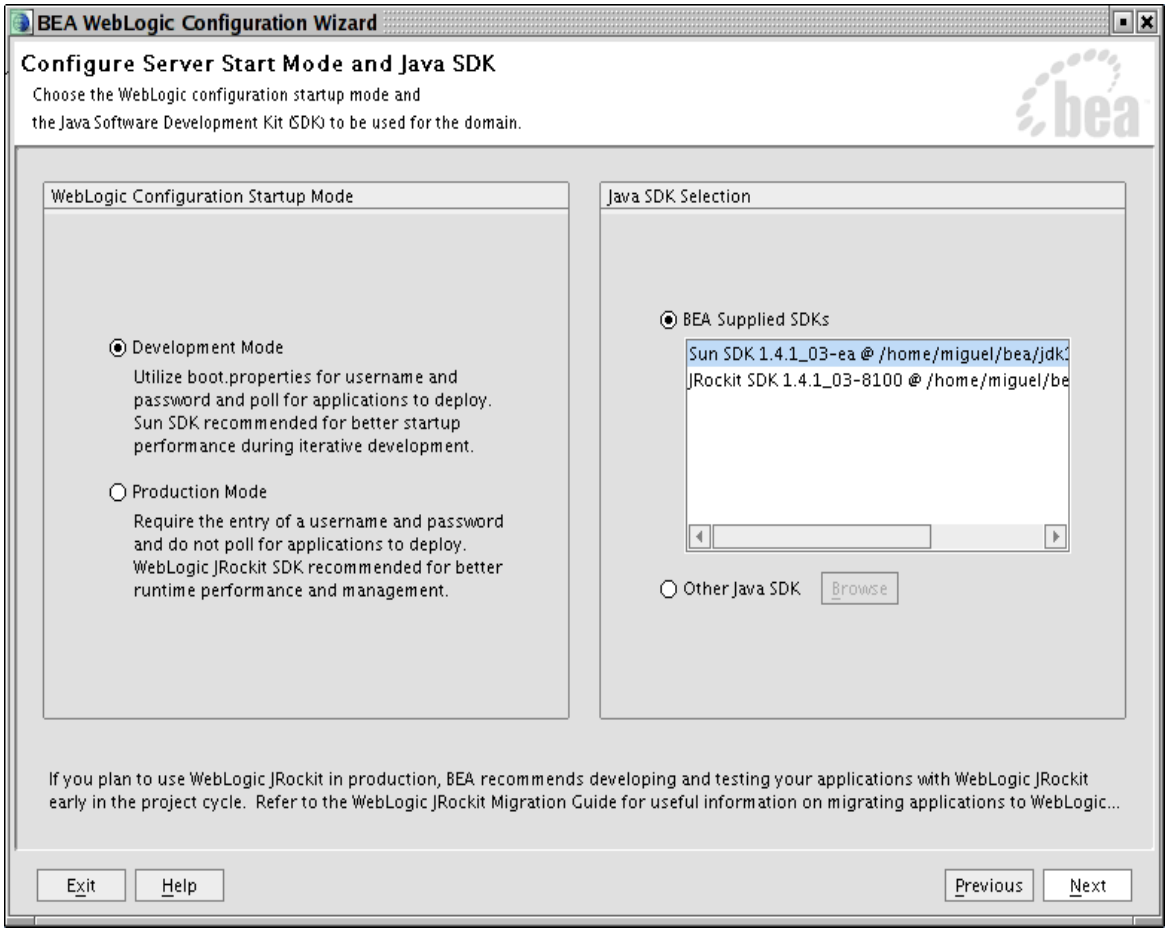
De la misma forma se nos permite configurar componentes JMS (mensajería). Seleccionamos no.



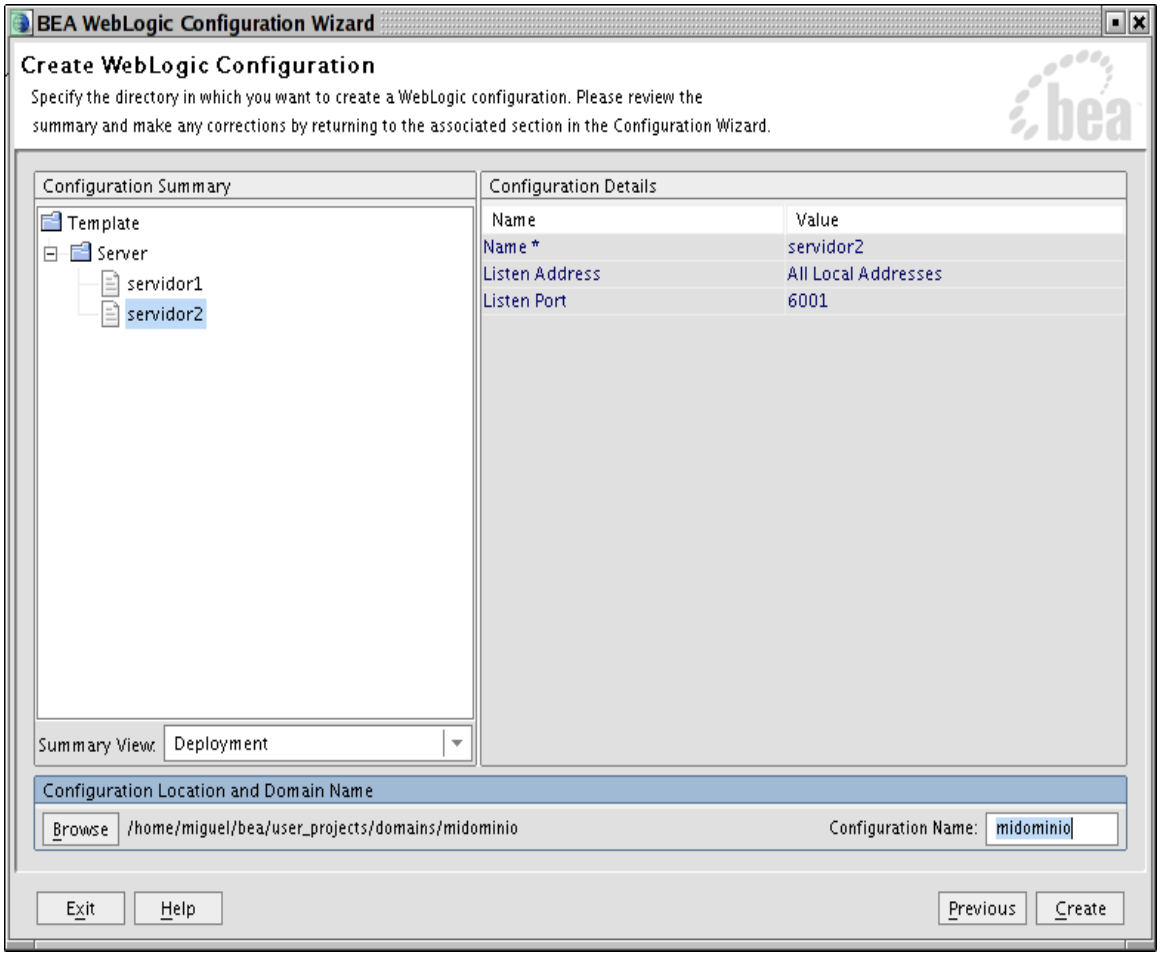
Al menos debemos configurar un usuario de administración. Vamos a darle como nombre *system* y como contraseña *weblogic*. En la parte inferior de la pantalla se nos da la opción de definir usuarios adicionales.



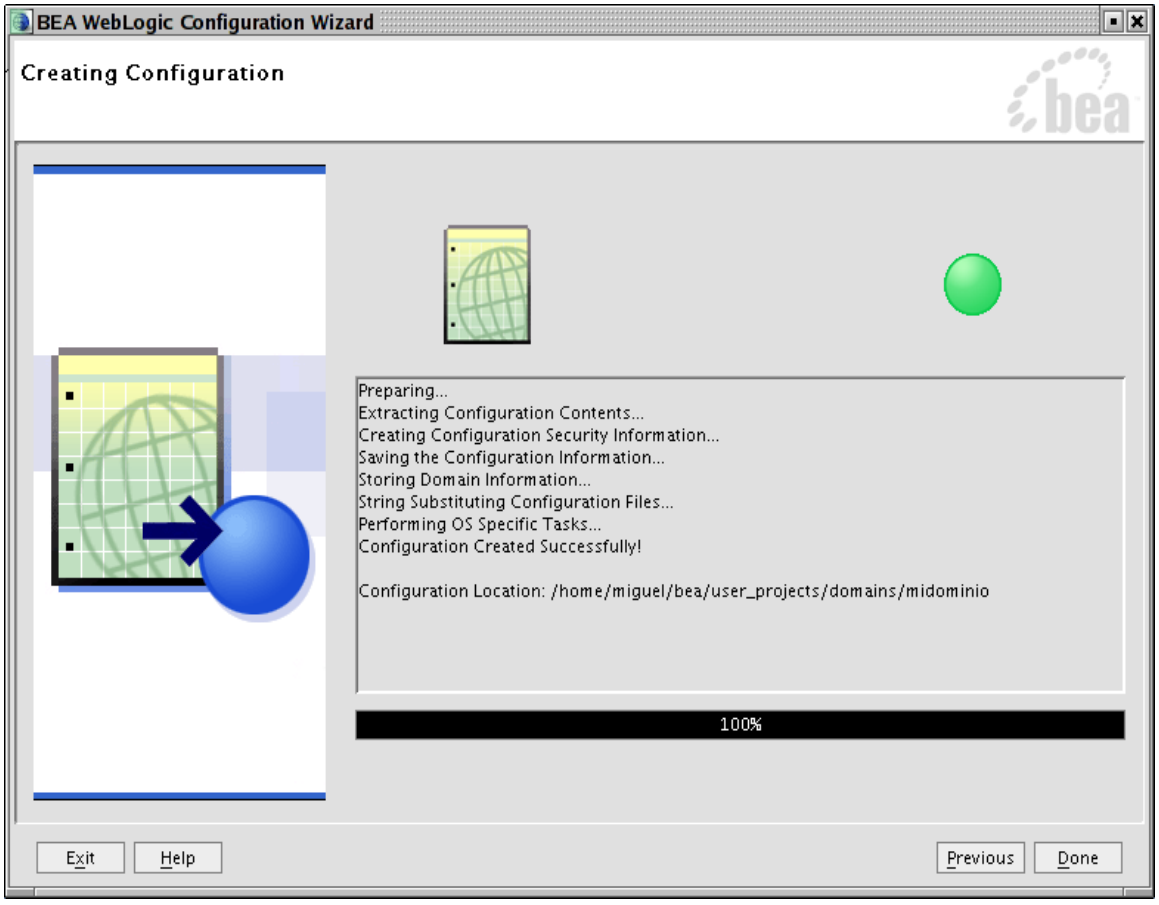
Como ya comentamos, existen dos modos principales de trabajo: modo desarrollo y modo producción. Para cada modo existen una serie de características que están habilitadas o no. Nosotros vamos a trabajar siempre en modo desarrollo. Podemos seleccionar la versión de Java que más nos convenga. Por defecto vamos a utilizar siempre la versión de Java que incorpora WebLogic.



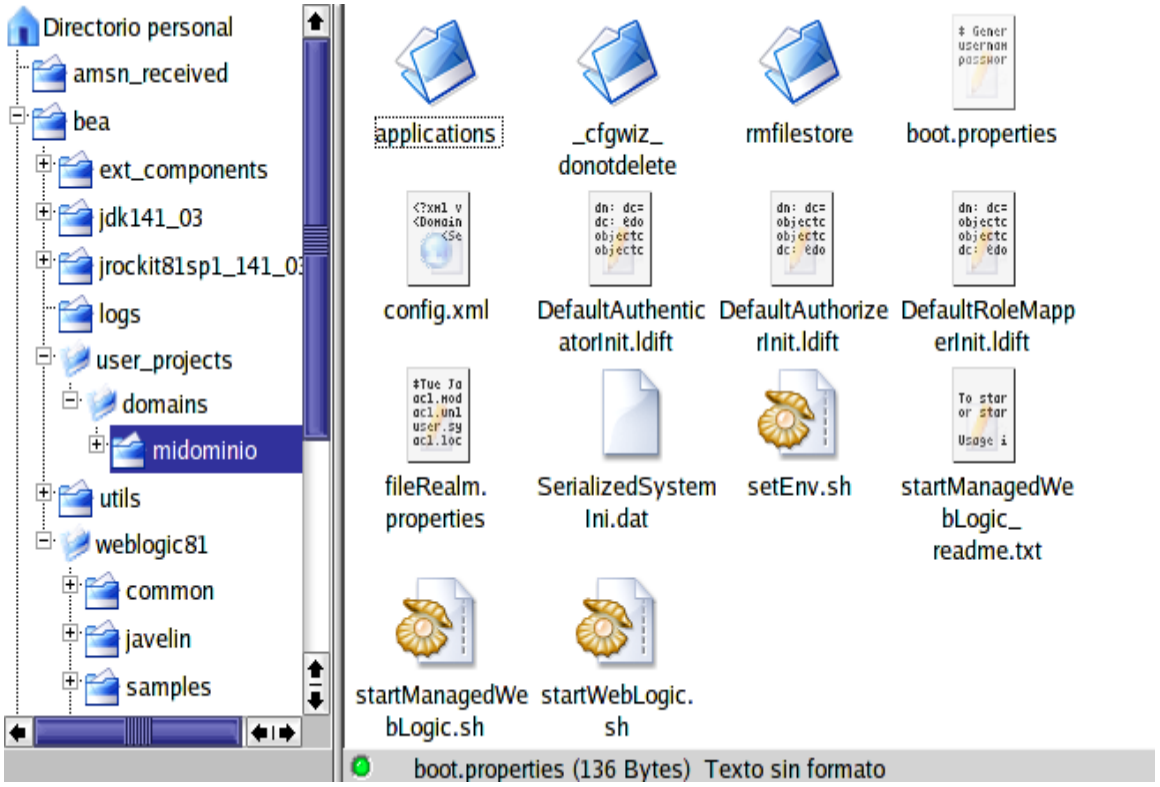
Esta ventana nos muestra la información introducida (servidores, máquinas, etc.) previamente y nos pide que demos el nombre del dominio. Se va a llamar *MiDominio.*



Iniciamos la creación del dominio y cuando finalice nos tiene que aparecer una ventana como la siguiente. Pinchamos en *Done* y hemos terminado de definir el dominio.



Una vez creado el dominio la estructura de directorios creada es la siguiente:



Tenemos un directorio por cada servidor creado, donde se guardan datos específicos del servidor (por ejemplo el fichero log). Estos directorios se crean cuando se pongan en marcha los servidores. El fichero *config.xml* contiene los datos del dominio (nombre de los servidores, máquinas, dominio, etc., nombre de las aplicaciones y su configuración, etc.). Los ficheros *startWebLogic.sh* y *startManagedWebLogic.sh* sirven para arrancar el servidor de administración y los administrados, respectivamente.

21.4 Arranque y configuración

Para poner en marcha los servidores debemos utilizar unos ejecutables que se encuentran en $HOME_BEA/user_projects/MiDominio. Primero debemos arrancar el servidor de administración. Para ello ejecutamos desde línea de comandos: *./startWebLogic.sh* Cuando nos aparezca el siguiente mensaje ya está arrancado el servidor:

<Server started in RUNNING mode>

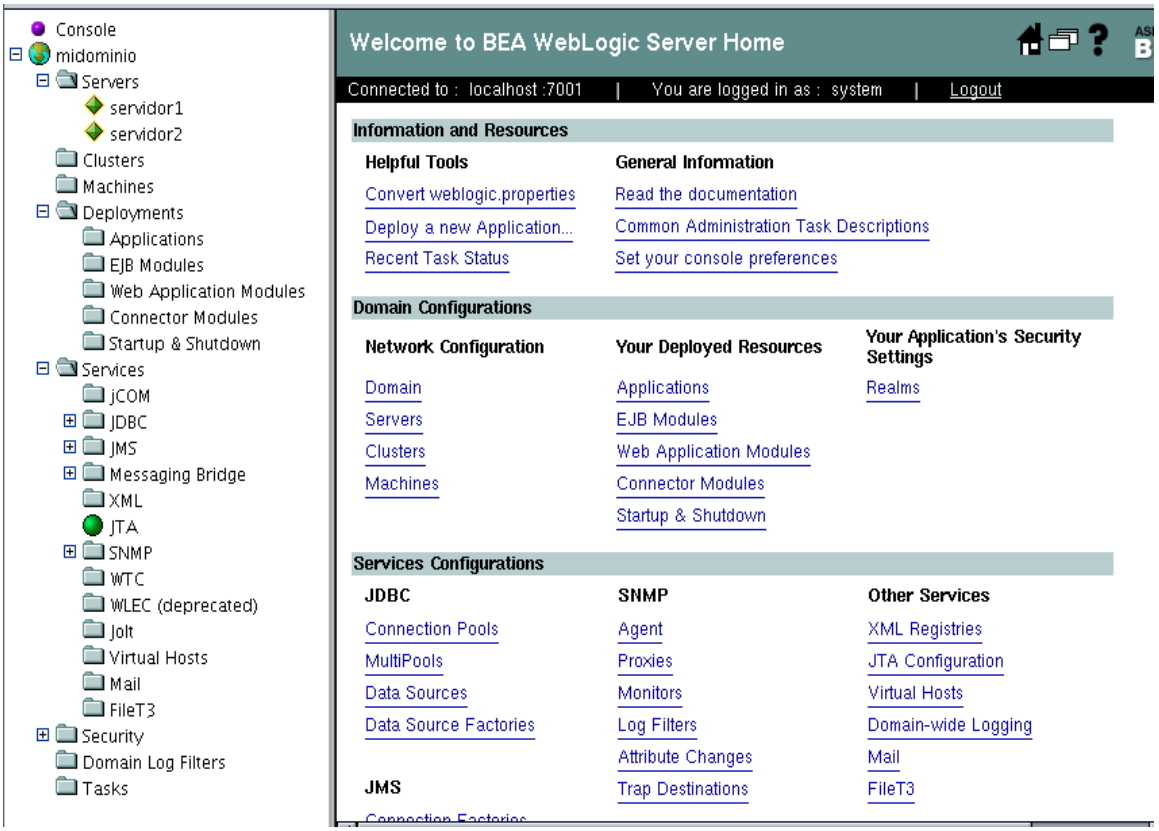
En este momento podemos arrancar los servidores administrados. Para ponerlos en marcha debemos utilizar el siguiente comando con los parámetros indicados:

./startManagedServer.sh nombre_servidor dirección_servidor_administración

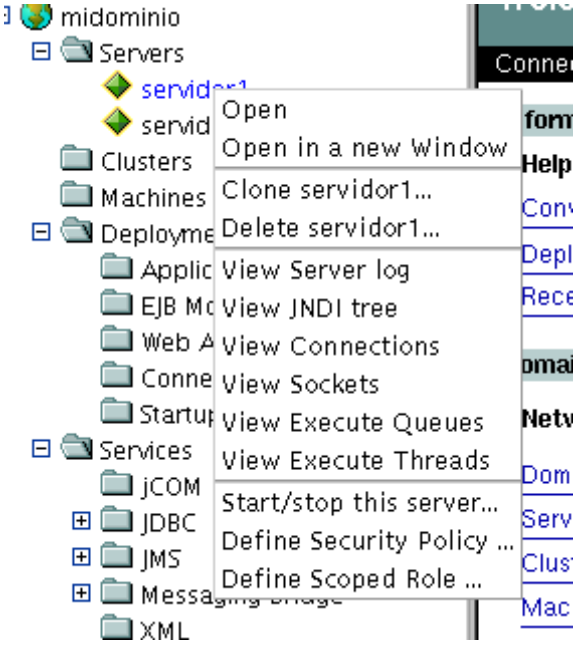
Por ejemplo, para arrancar el servidor administrado creado anteriormente debemos ejecutar el siguiente comando:

./startManagedServer.sh servidor2 http://localhost:7001

El servidor de administración nos facilita una aplicación que permite administrar nuestro dominio. Es la consola de administración (*Console*). Con la consola podemos configurar los atributos de los distintos recursos, hacer despliegues de aplicaciones, monitorizar el uso de recursos, ver mensajes de log y poner en marcha o parar los distintos servidores de nuestro dominio. La consola se gestiona con un navegador en la siguiente dirección: *http://dirección-de-escucha:7001/console*. Nos aparecerá una página donde se nos solicita el usuario y la contraseña. Una vez introducida nos aparecerá la siguiente página.



La parte de la izquierda es una applet en forma de árbol jerárquico que nos muestra todas las opciones que podemos configurar en el dominio. En la parte superior del árbol tenemos el nombre del dominio. Dentro del dominio, las primeras opciones nos permiten configurar los elementos del dominio (servidores, máquinas, cluster, etc.). A continuación podemos realizar despliegues de aplicaciones, aplicaciones web, EJBs, etc. La última opción contiene los servicios configurables (JDBC, JMS, Virtual Hosts, etc.). Los elementos de la parte izquierda disponen de un menú adicional que se obtiene pinchando con el botón derecho sobre un elemento del árbol, como el mostrado en la siguiente figura:

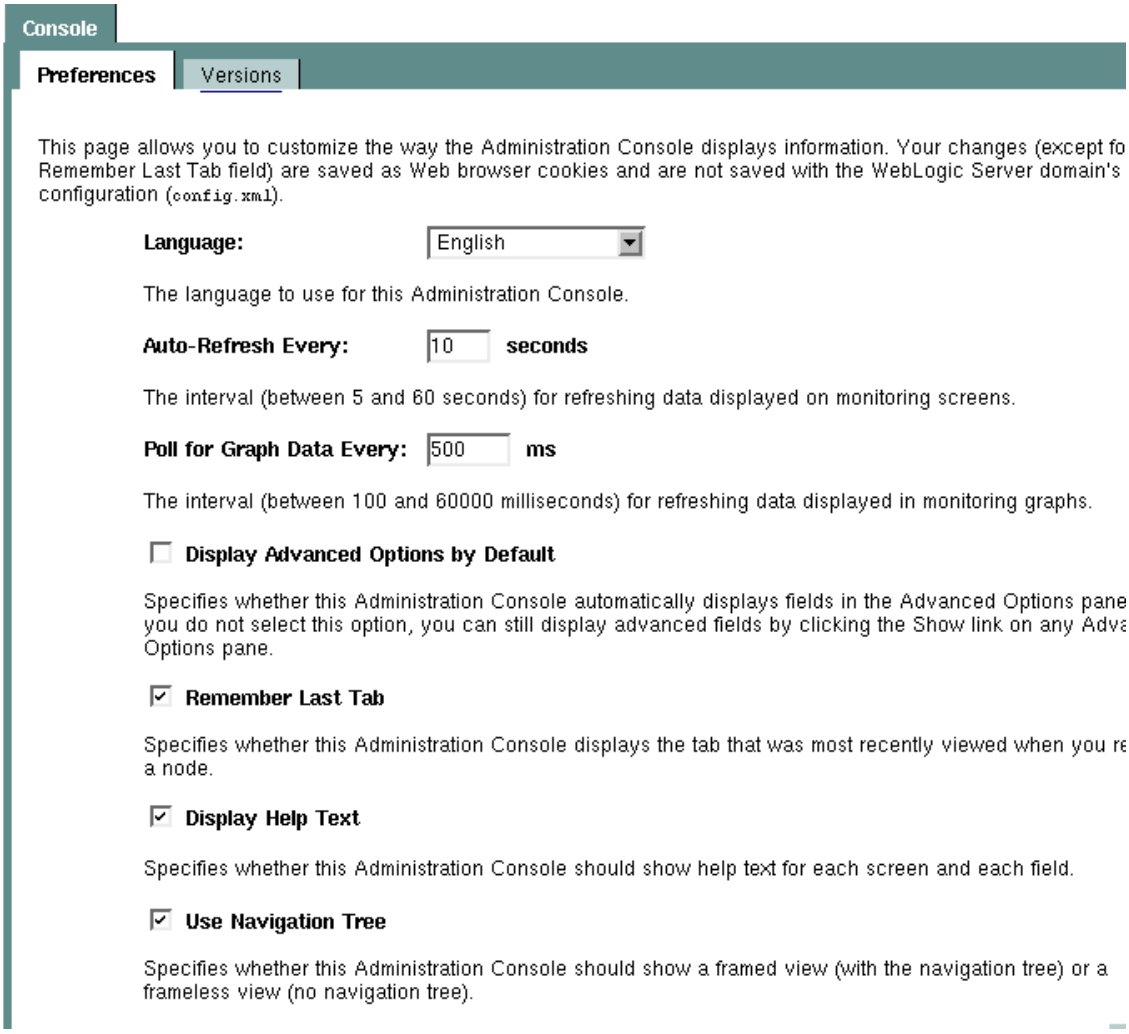


Las opciones varían dependiendo del elemento seleccionado.

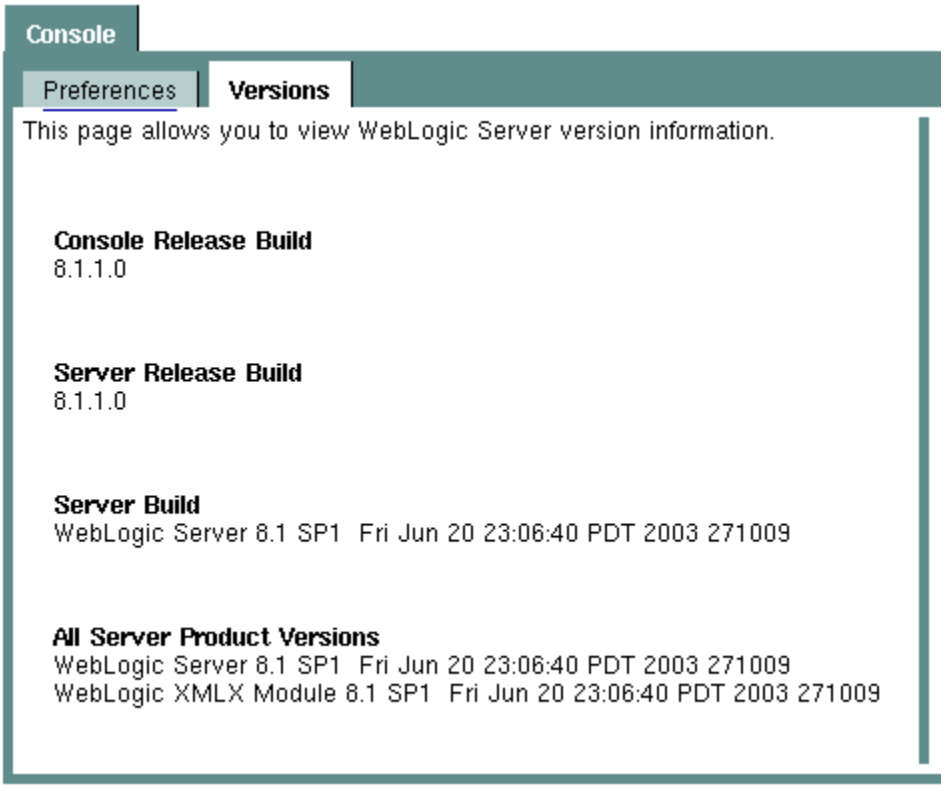
En la parte derecha tenemos las mismas opciones a las que podemos acceder mediante el applet. También nos irán apareciendo las detalles de configuración para un servicio o característica concreta elegida en la parte izquierda.

Vamos a ver las opciones más generales. Si pinchamos en el elemento *console* nos permite configurar opciones generales a todos los dominios. Nos aparecerá una página como la mostrada a continuación donde podemos:

- Seleccionar el idioma (inglés o japonés)
- Indicar el tiempo de refresco de las páginas.
- Indicar el tiempo de refresco de los datos gráficos (vistos un poco más adelante
- La opción de mostrar opciones avanzadas por defecto nos permite visualizar dichas opciones, que por defecto están ocultas.
- Si marcamos la opción *Remember Last Tab*, cuando pasemos de una opción a otra se acordará de la última solapa visitada en dicha opción.
- La opción *Display Help Text* muestra el texto que nos aparece debajo de cada opción.
- Al marcar la última opción nos permite disponer del árbol de navegación.

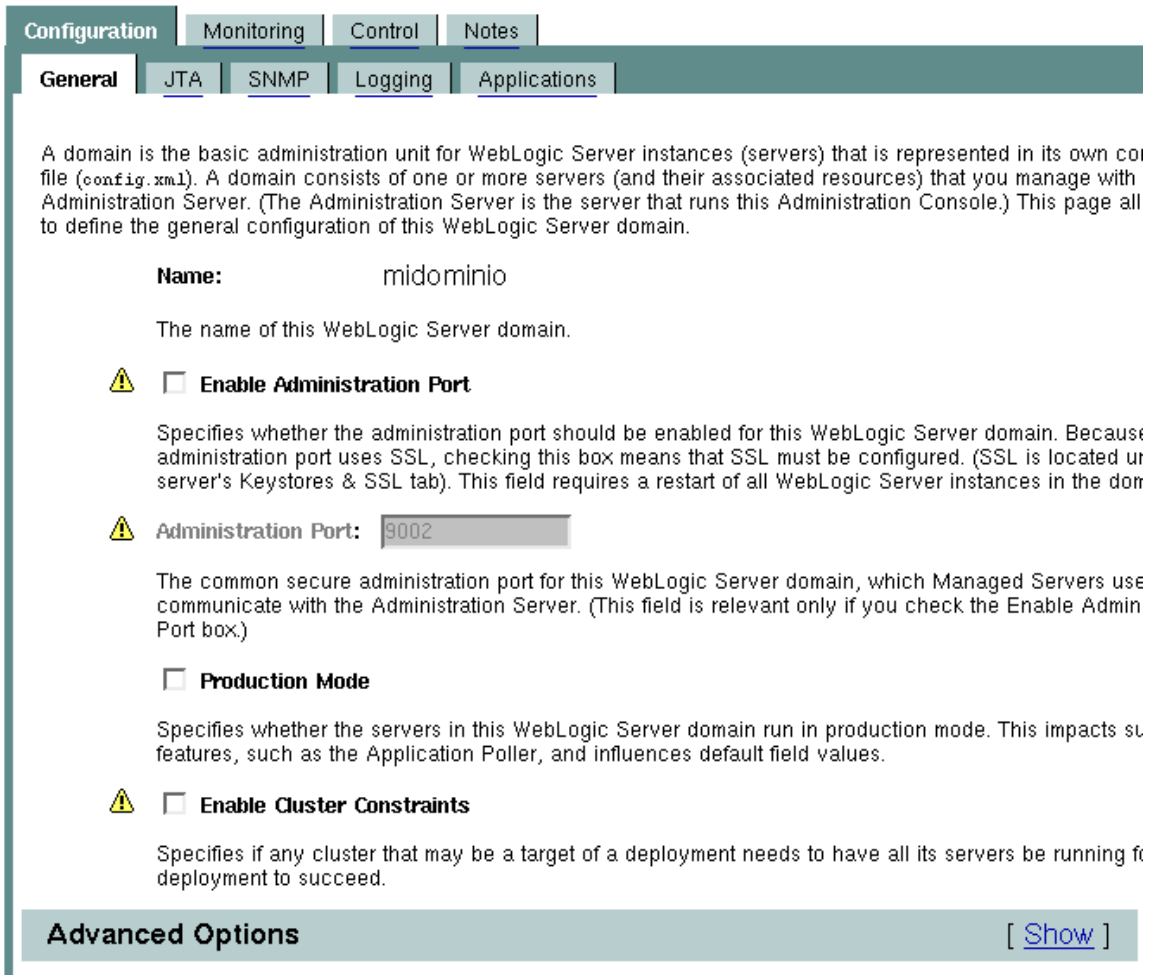


En la otra solapa simplemente nos da información de versión.



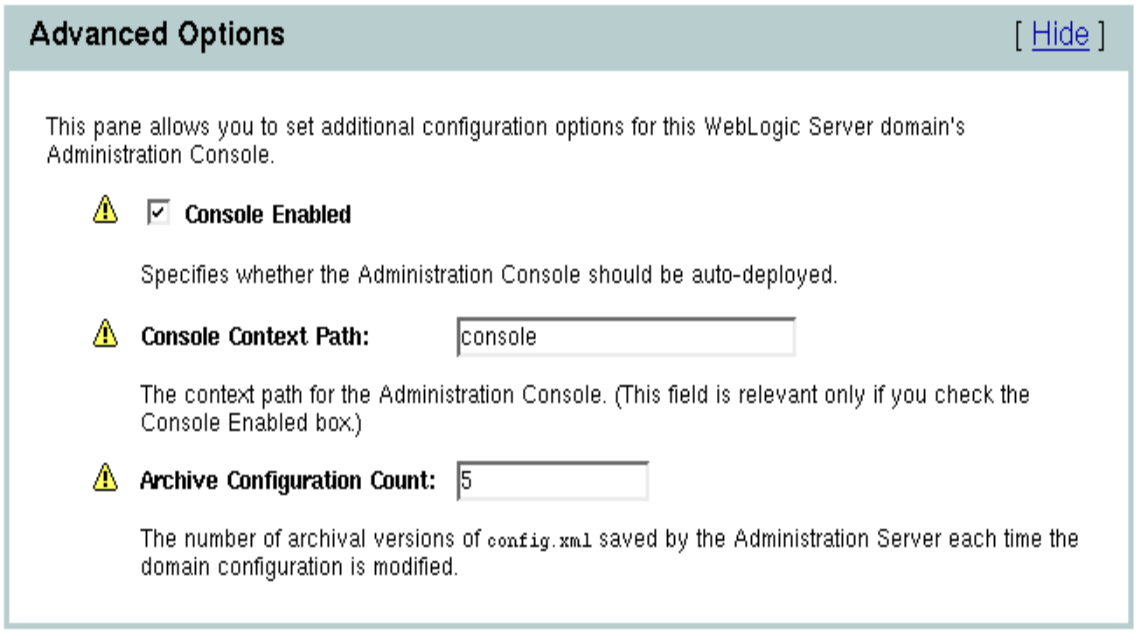
Pasamos a las opciones para el dominio. Pinchamos en el nombre de nuestro dominio y nos aparece una ventana como la mostrada en la ventana siguiente. El símbolo ⚠ que aparece a la izquierda de algunas opciones nos indica que es necesario reiniciar uno o varios servidores si cambiamos esa opción. En la configuración general (la solapa actual) podemos configurar las siguientes opciones:

- La habilitación del puerto de administración, si marcada, permite que todos los elementos del dominio se comuniquen con el servidor de administración mediante una conexión segura. Además podemos configurar un puerto adicional (no puede ser el seguro del servidor de administración) para dichas comunicaciones. Esta opción permite que podamos arrancar un servidor en modo *standby* en el cual el servidor no escucha las peticiones que le llegan a su puerto, pero se permite una comunicación con el servidor de administración. También permite separar las peticiones de aplicación (llegan de las aplicaciones que usan el sistema) de las peticiones de administración (generadas por o hacia el servidor de administración). De esta manera una petición del servidor de administración puede ser atendida sin tener que esperar su turno dentro de las peticiones de aplicación. Si activamos esta opción debemos asignarle un puerto de comunicación. Al activar la opción, la consola sólo responde por https://dirección:puerto-asignado/console
- La opción siguiente nos permite especificar si trabajamos en modo producción. La activación de esta opción implica que ciertas características están activas y otras no.
- Si activamos la última opción para que un cluster responda a una aplicación todos sus servidores tienen que estar funcionando a la vez.



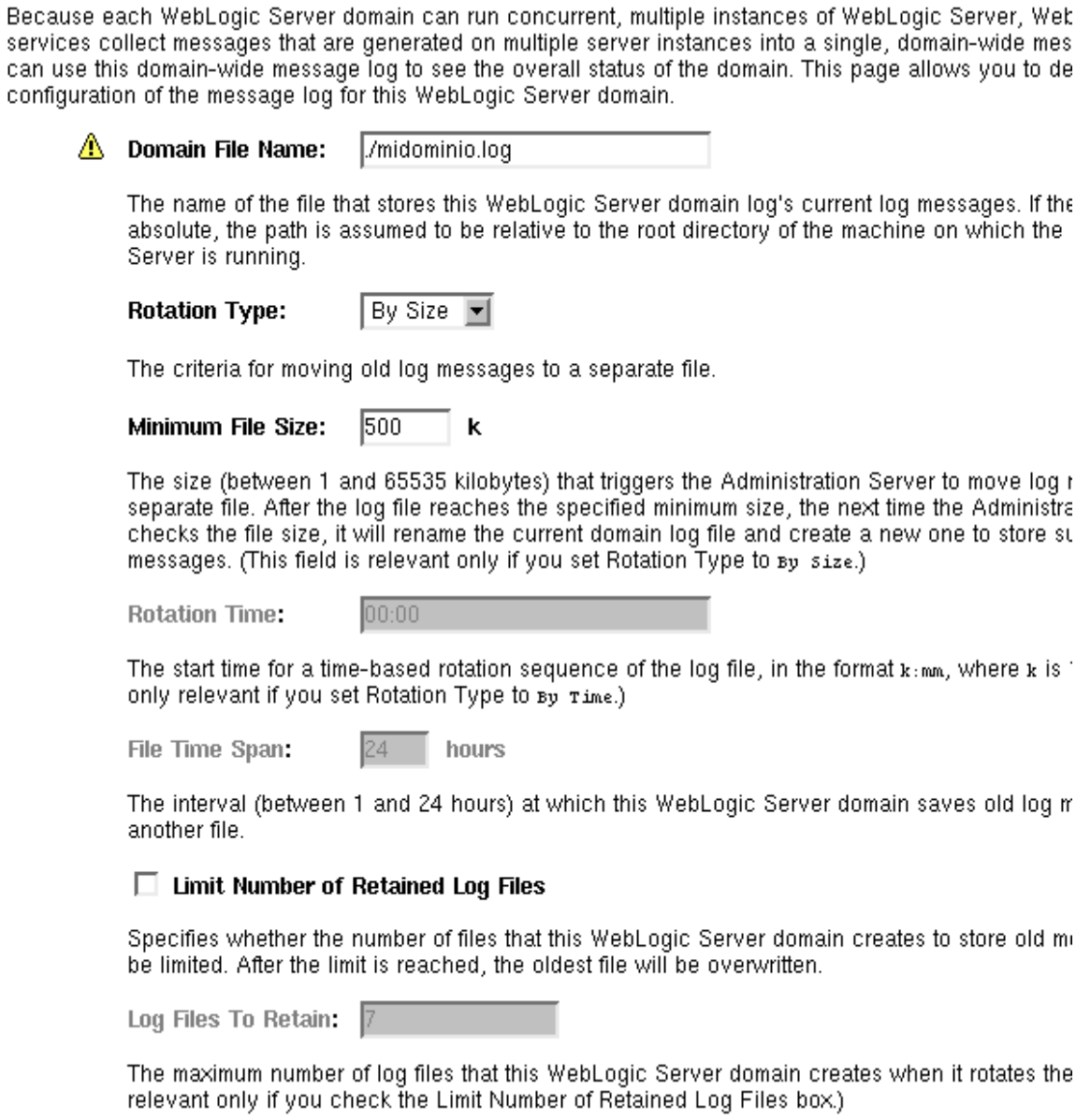
Si pinchamos en *Show* se nos mostrarán las opciones avanzadas. Son las siguientes:

- Habilitar la consola. En modo producción suele ser habitual deshabilitar la consola, para que no pueda ser accedida desde el exterior.
- La siguiente opción nos permite dar un nombre distinto a la aplicación de la consola. Si, por ejemplo, damos el nombre *miconsola*, para acceder a la consola tendríamos que teclear http://dirección:puerto/miconsola.
- Por último podemos especificar cuántas versiones del fichero de configuración se guardarán.

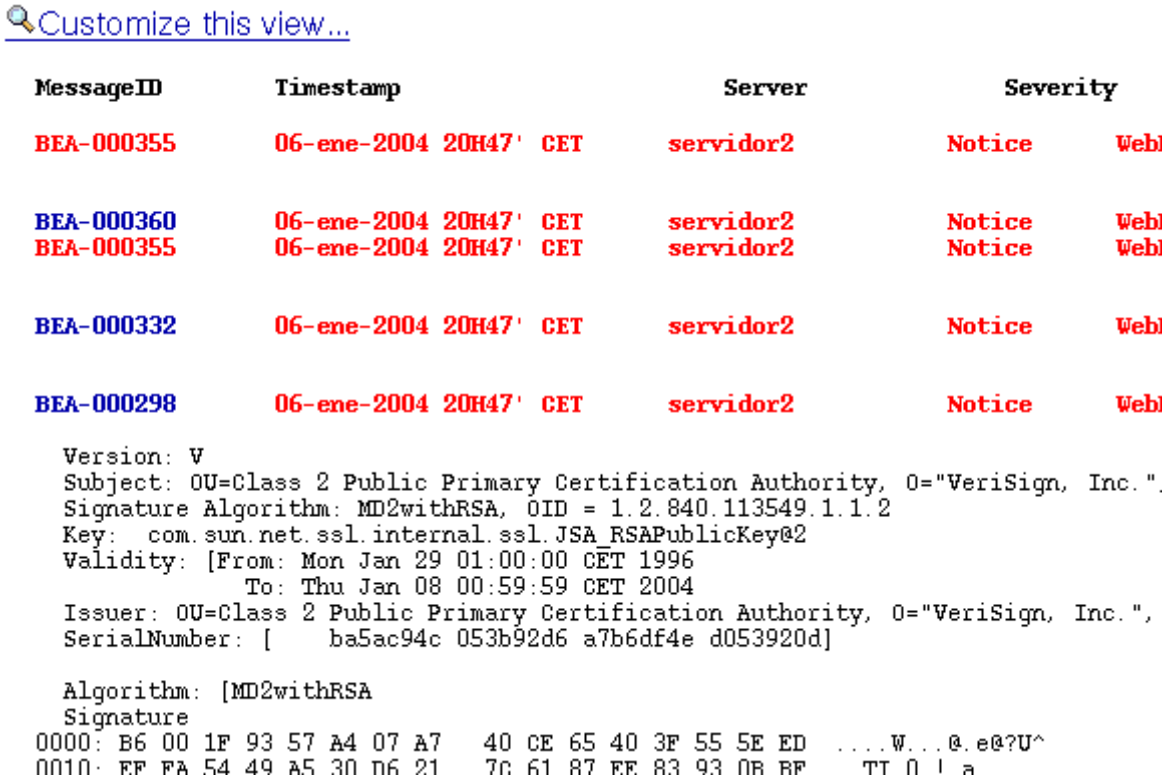


En la solapa de *Logging* (las opciones JTA y SNMP las veremos más adelante) podemos configurar el fichero log del dominio. El fichero log almacena toda la información y mensajes del dominio. Las opciones son las siguientes:

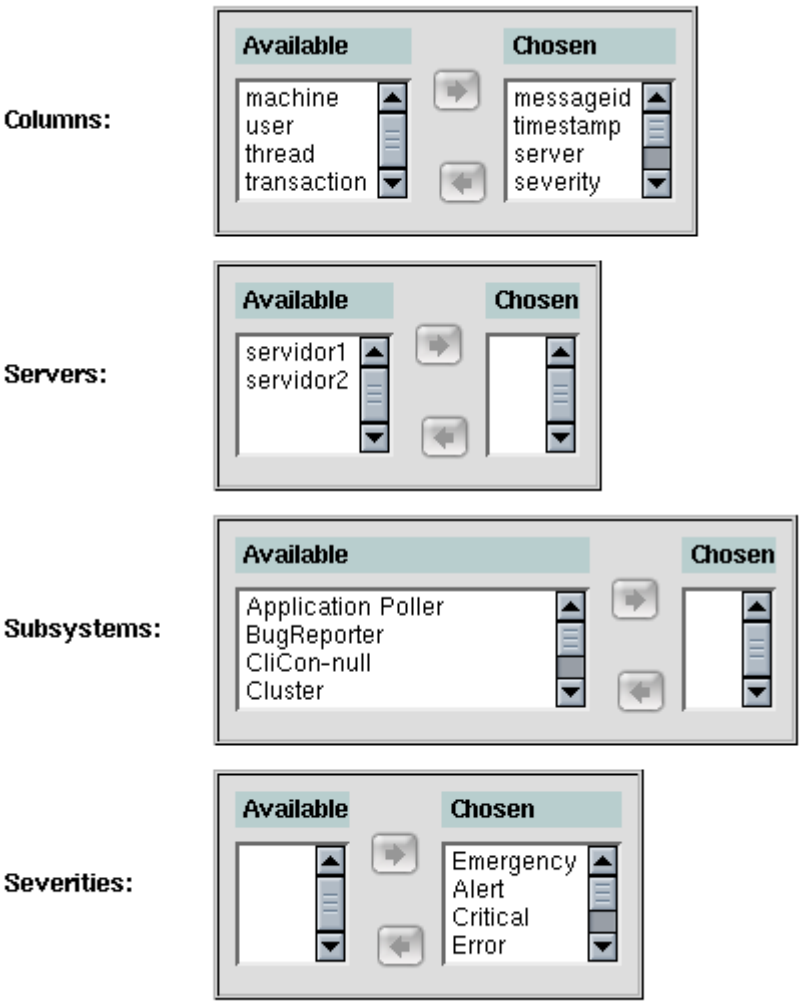
- Podemos cambiar el nombre del fichero log.
- La siguiente opción permite especificar el tipo de rotación. Las opciones a elegir son por tamaño o por tiempo. La rotación permite que el fichero log no vaya creciendo indefinidamente. Si elegimos por tamaño, se cogerá el valor del parámetro *Minimum File Size* y, cuando el fichero de log alcance ese tamaño, creará un nuevo fichero de log, renombrando el anterior. Si, por ejemplo, el nombre del fichero de log es *midominio.log* y hemos seleccionado una rotación por tamaño y 500k de tamaño mínimo, cuando el fichero alcance ese tamaño el sistema cambiará el nombre del fichero por *midominio.0* y creará uno nuevo,*midominio.log*, donde se seguirá almacenando la salida del sistema. Cuando se vuelva a superar ese límite se le dará el nombre *midominio.1* y así sucesivamente. El otro tipo de rotación, de tiempo, actúa de manera similar, pero especificando un tiempo de rotación. Cuando el reloj del sistema llega a esa hora se produce el cambio de fichero. En esta opción, podemos especificar cada cuantas horas se produce el cambio, cambiando el valor de *File Time Span*.
- La penúltima opción permite limitar el número de ficheros a almacenar. Si la activamos toma el valor de la siguiente opción *Log Files to Retain* y, cuando el contador de fichero alcance ese valor, empieza desde cero sobreescribiendo el primero.
- Si pinchamos en la opción *View Domain Log* se nos permite ver el fichero log (ver siguientes figuras).



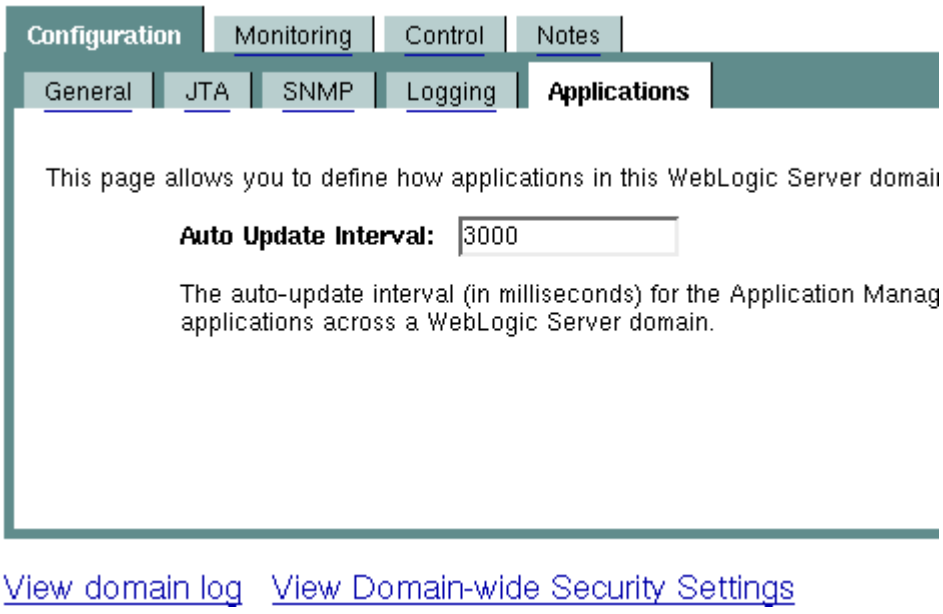
Visualización del fichero de log.



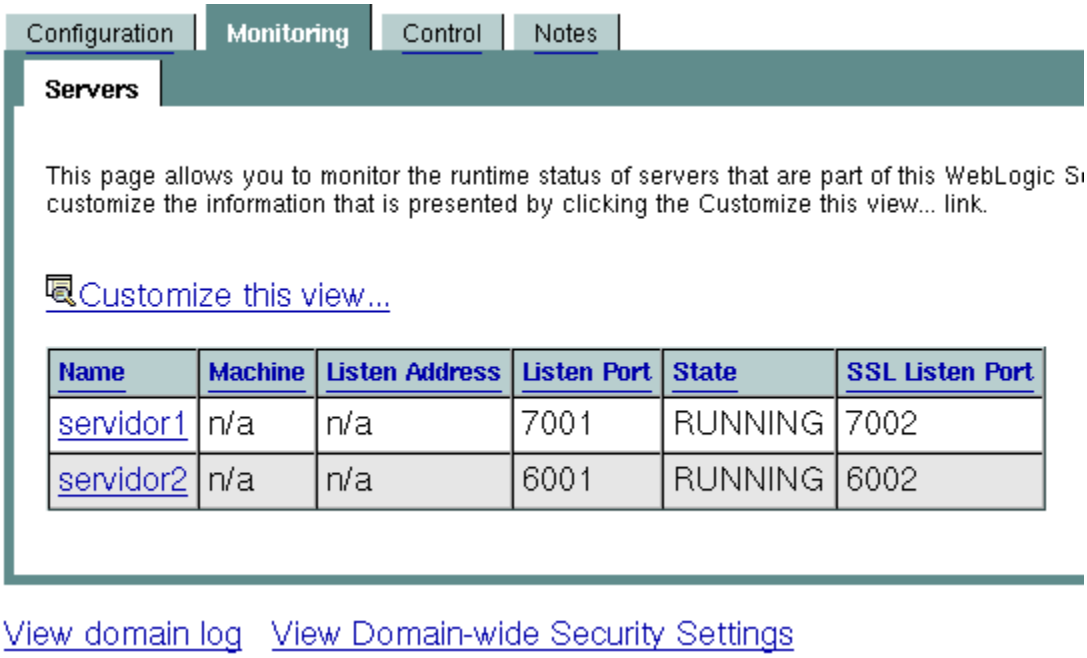
Si pinchamos en *Customize this view* se nos permite configurar los mensajes, mostrando las opciones que queremos que se vean en el fichero de log.



En la siguiente figura, la opción *Auto Update Interval* indica al sistema cada cuánto tiempo debe comprobar si existen aplicaciones nuevas. En modo producción se deshabilita esta comprobación.

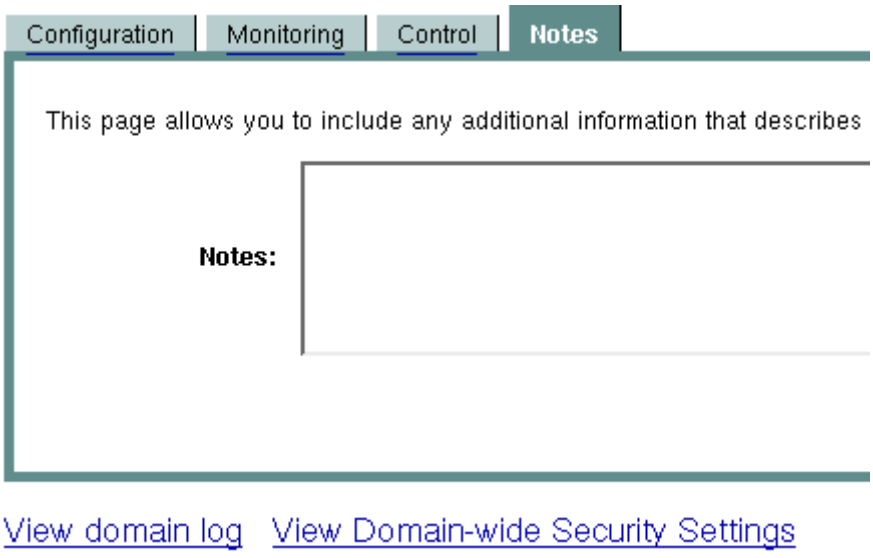


Cuando seleccionamos la solapa *Monitoring* nos aparece un enlace que nos permite monitorizar los servidores de nuestro dominio. Tenemos los servidores creados en nuestro dominio y se nos indica la dirección de escucha, los puertos de escucha y el estado del servidor.



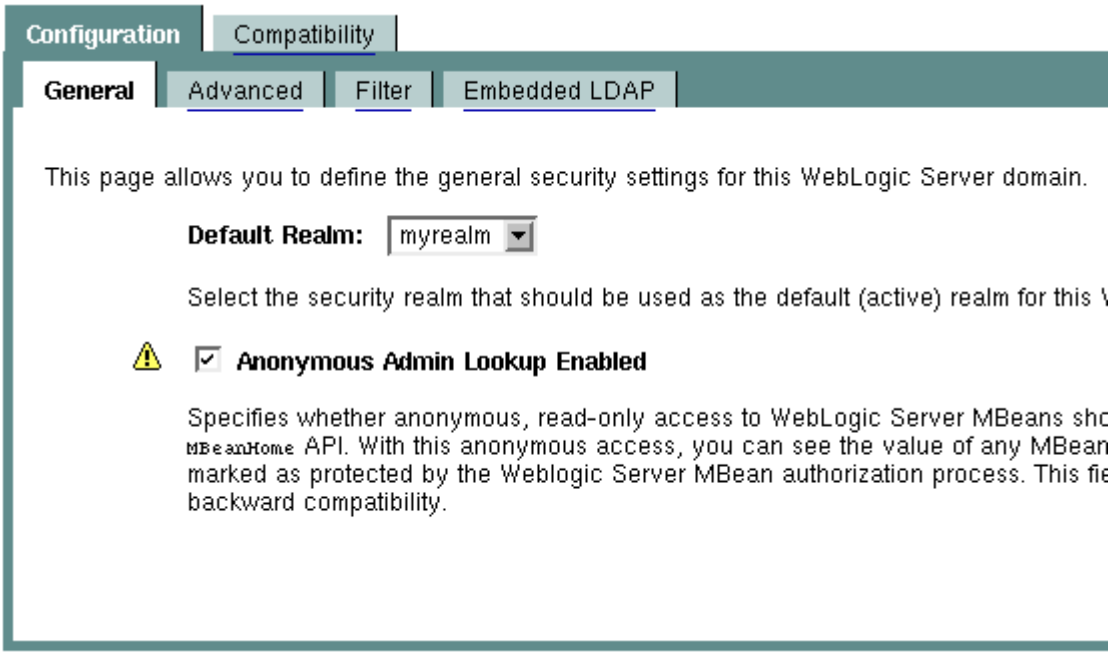
La solapa de control permite controlar (parar, poner en marcha, etc.) los servidores del dominio.

La última solapa, *Notes*, nos permite introducir notas asociadas a la configuración actual. Esto es común en la mayoría de opciones de configuración. Tienen un carácter informativo.

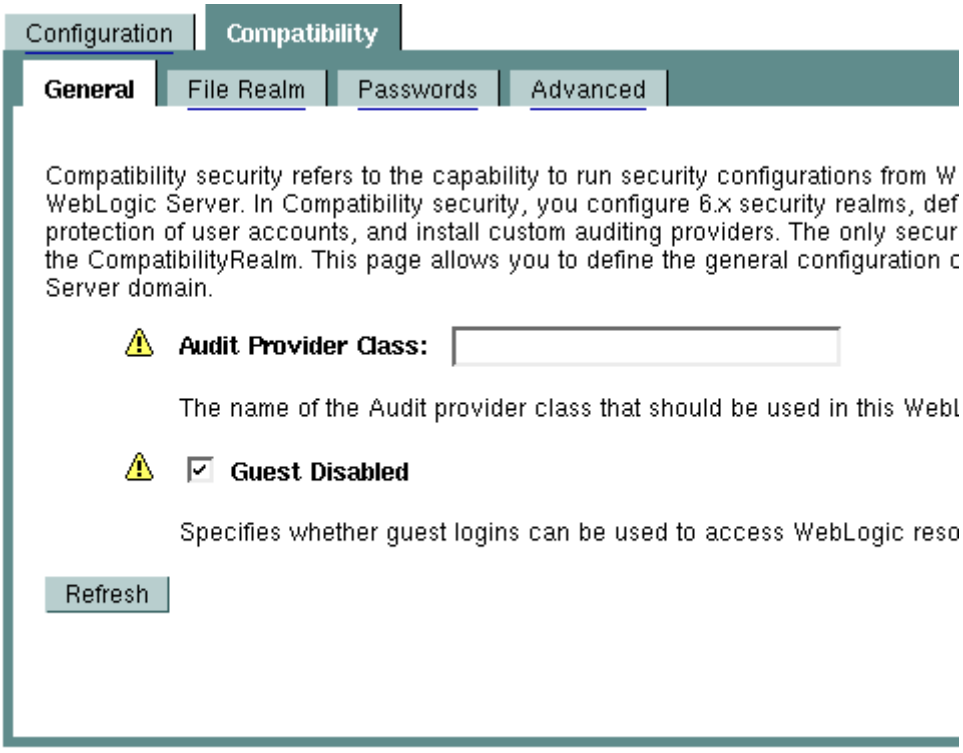


View domain log View Domain-wide Security Settings

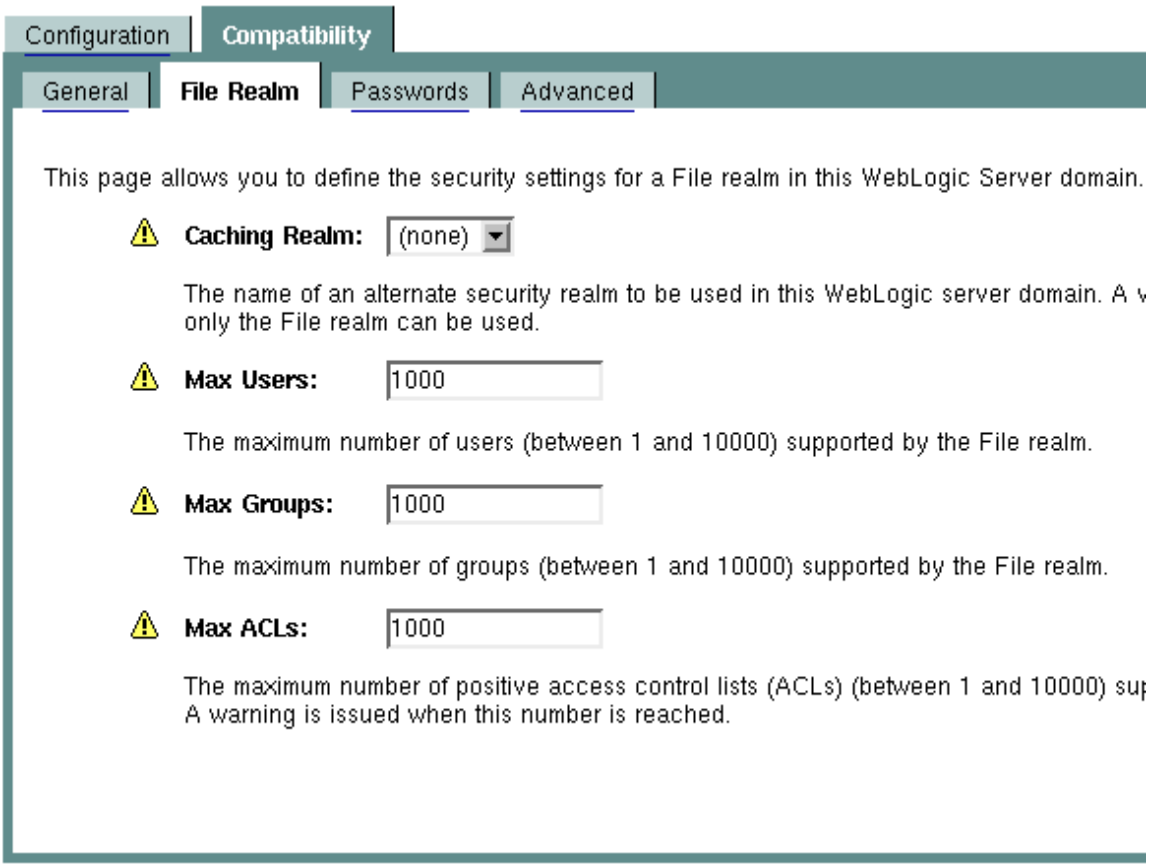
Si pinchamos en el enlace *View Domain-wide Security Settings* nos visualiza opciones de seguridad. Todo lo referente a *realm* lo explicaremos más adelante.



Si activamos la opción *Guest Disabled* no permitiremos que entre el usuario invitado.

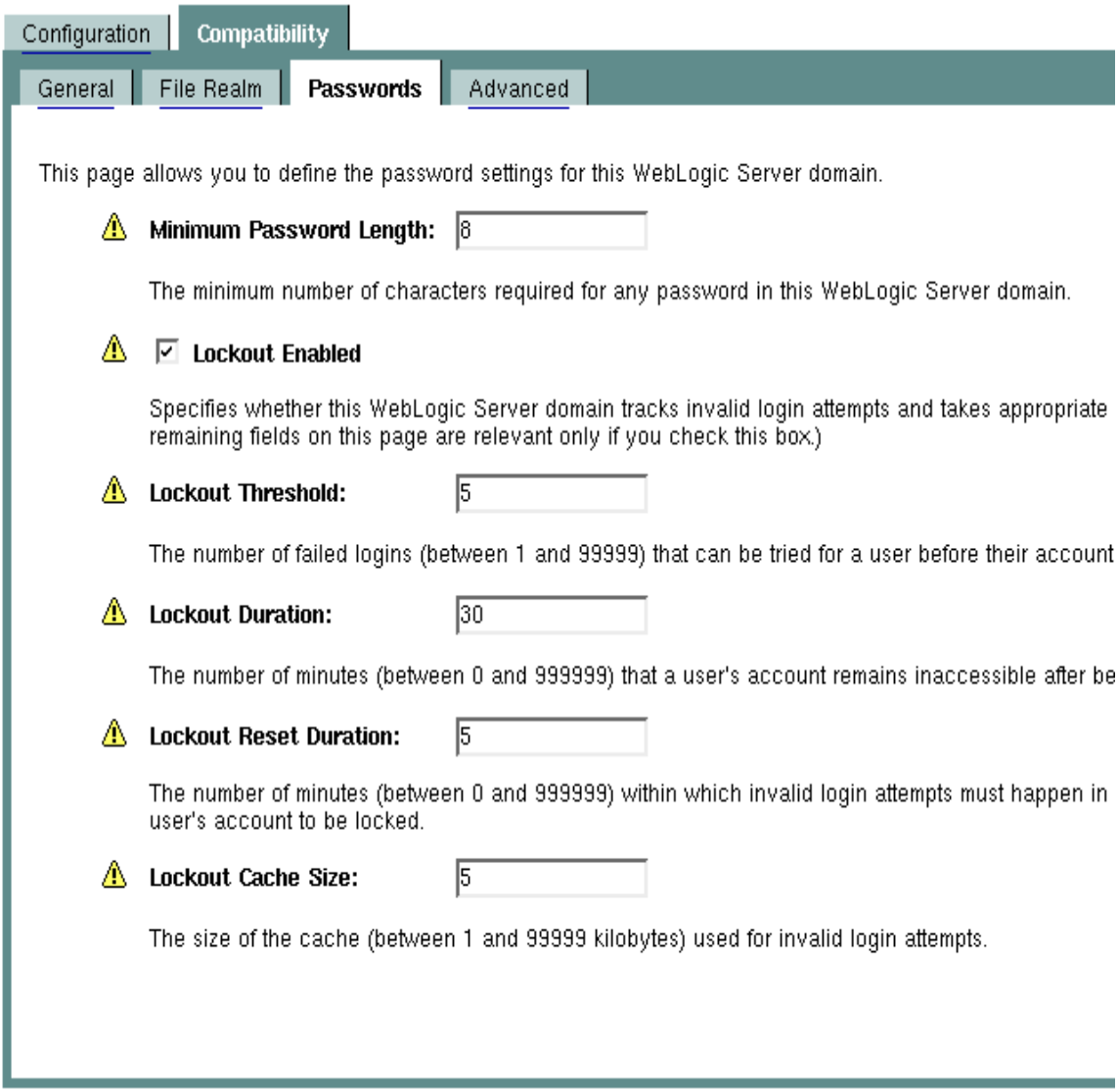


En el apartado de *File Realm* podemos configurar opciones específicas de seguridad como: número máximo de usuarios, grupos y ACL.



La siguiente solapa tiene que ver con características de la contraseña de acceso y el bloqueo de una cuenta por haber intentado acceder con una contraseña incorrecta. Si un usuario intenta acceder al sistema e introduce una contraseña incorrecta, cuando realice un determinado número de intentos la cuenta será deshabilitada. Las opciones son:

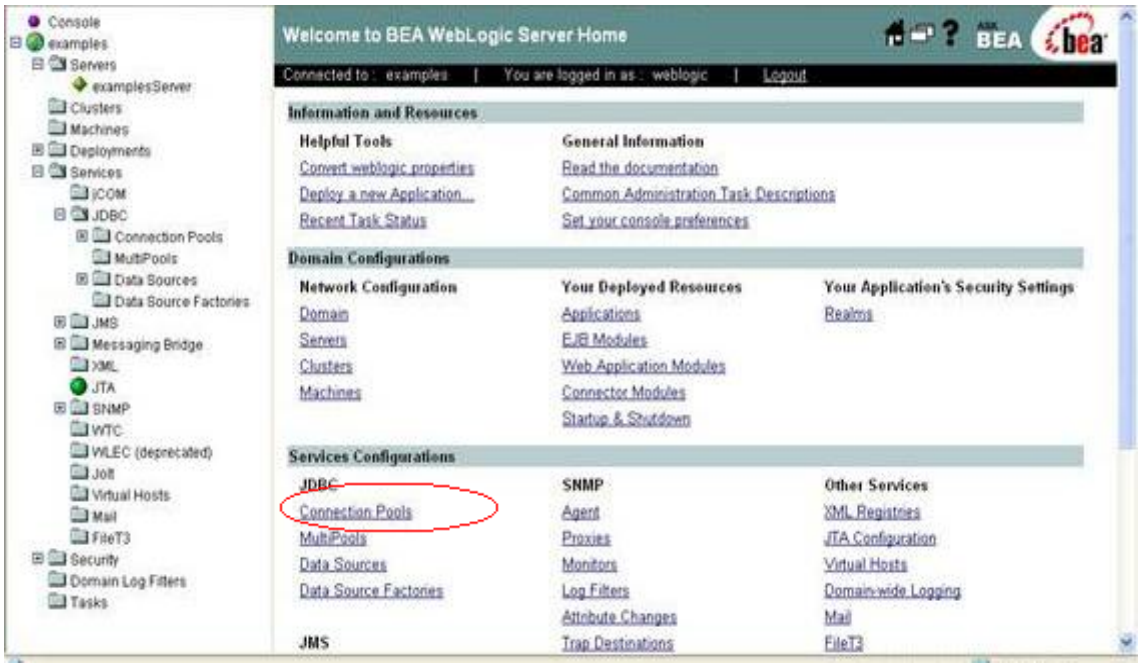
- Longitud mínima de contraseña. Indica el número de caracteres mínimo que debe tener la contraseña.
- La siguiente opción, si marcada, permite el bloqueo de una cuenta al intentar acceder con una contraseña errónea.
- La opción *Lockout Threshold* especifica el número de intentos erróneos que provocan el bloqueo de la cuenta.
- La siguiente es el número de minutos que se bloquea la cuenta.
- La opción *Lockout Reset Duration* indica el número de minutos durante los cuales se cuenta el número de intentos fallidos. Si marcamos cinco, si durante cinco minutos se han realizado cinco (el número indicado por *Lockout Threshold*) intentos fallidos, se produce el bloqueo.
- La última opción es el tamaño de la cache de intentos fallidos de cualquier usuario que el sistema almacenará.



21.5 Definición de un datasource en Weblogic 8.1

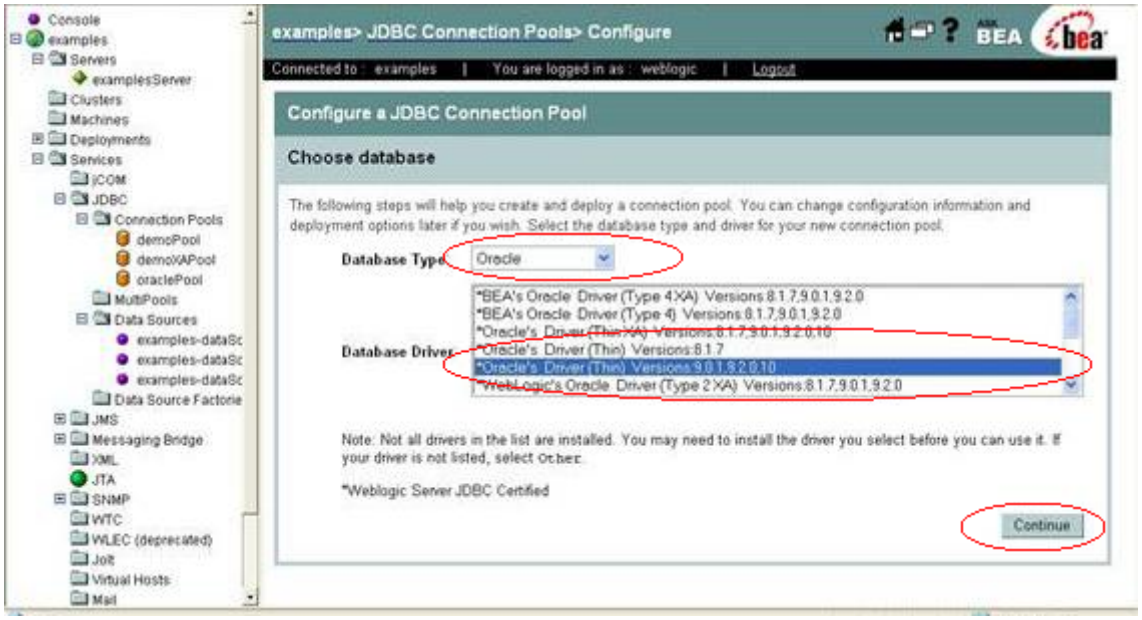
Para definir un datasource, hay que definir primero un "connection pool"

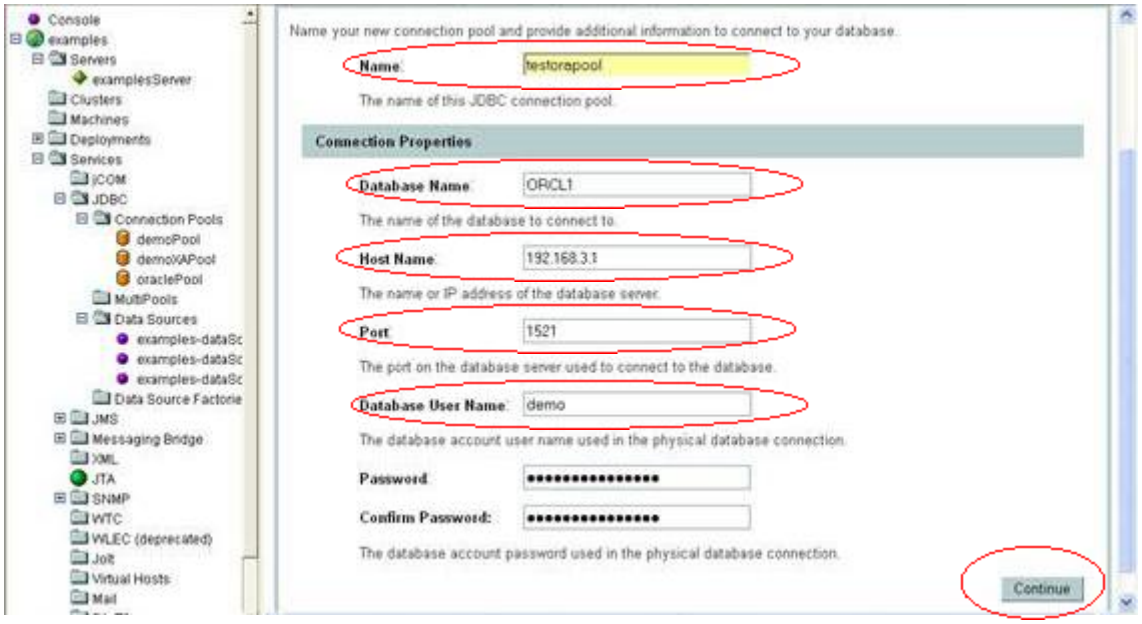
21.5.1 Definición del Connection pool

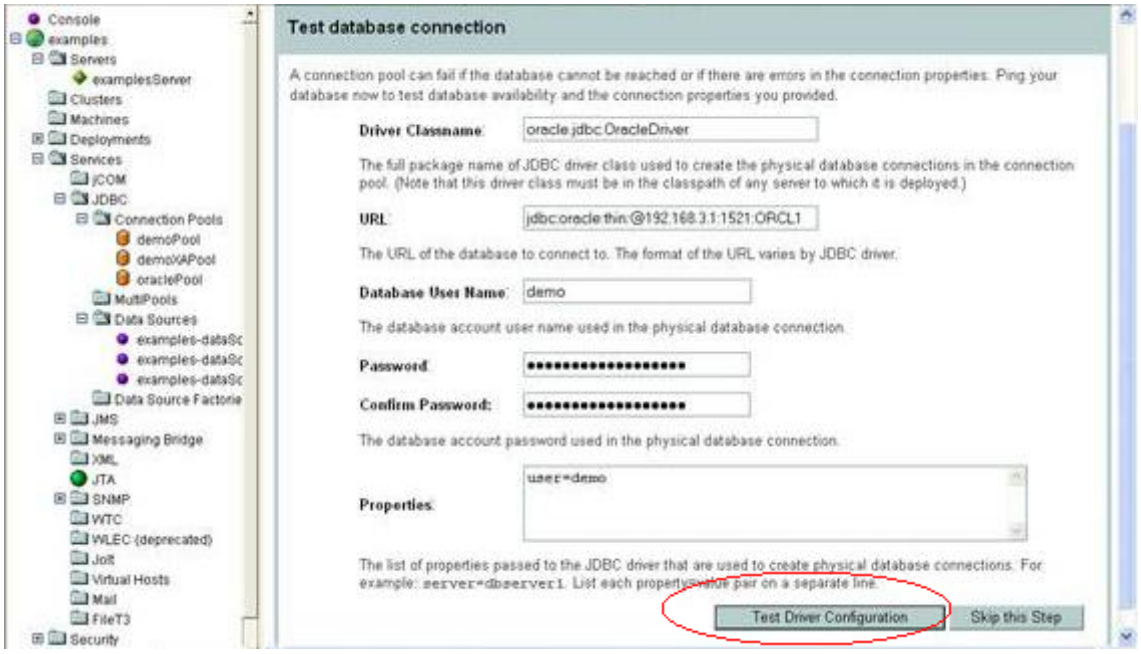


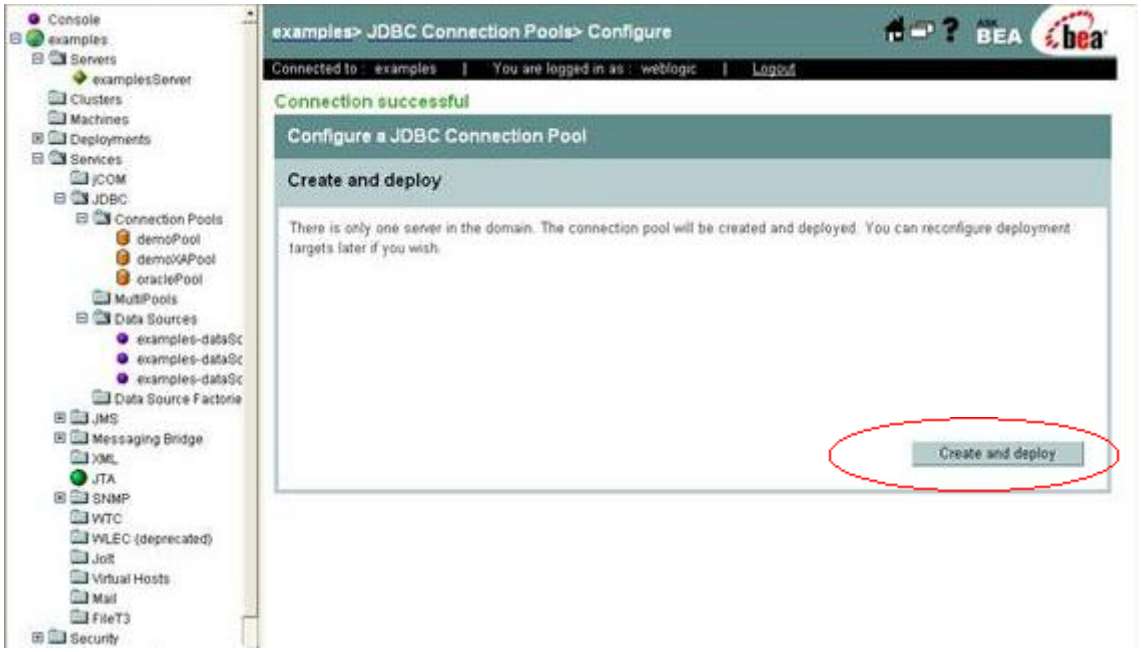
En la siguiente pantalla se determina el dbms sobre el cual se quiere crear el pool, así como también con qué drivers se va a realizar.

En este ejemplo, se elige Oracle, y los drivers thin del mismo

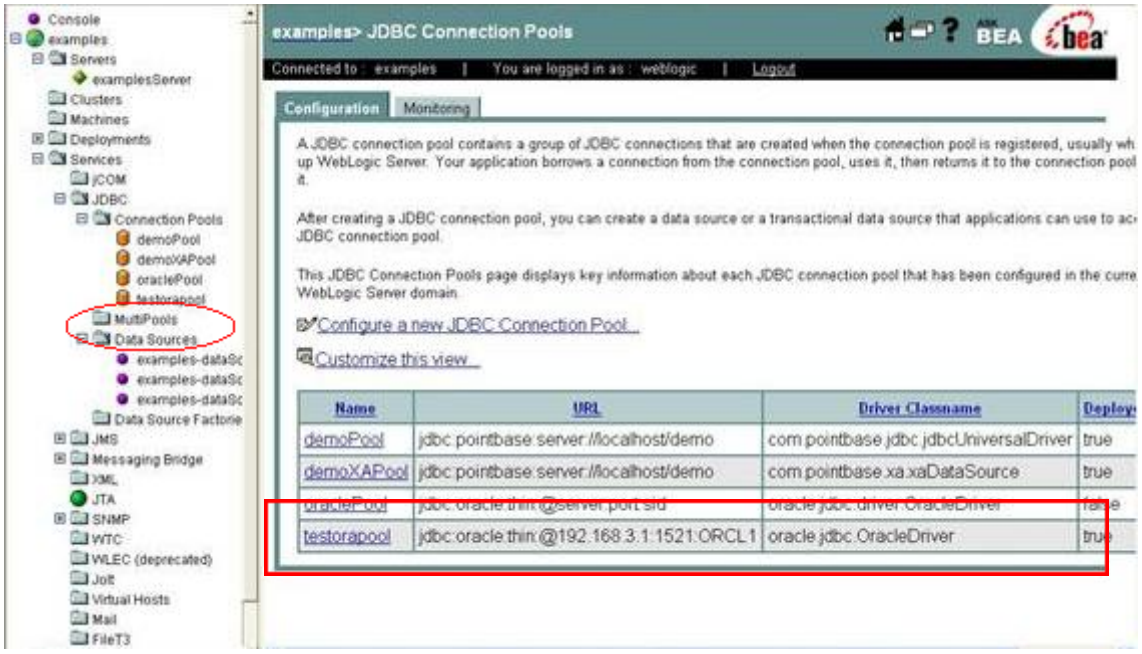






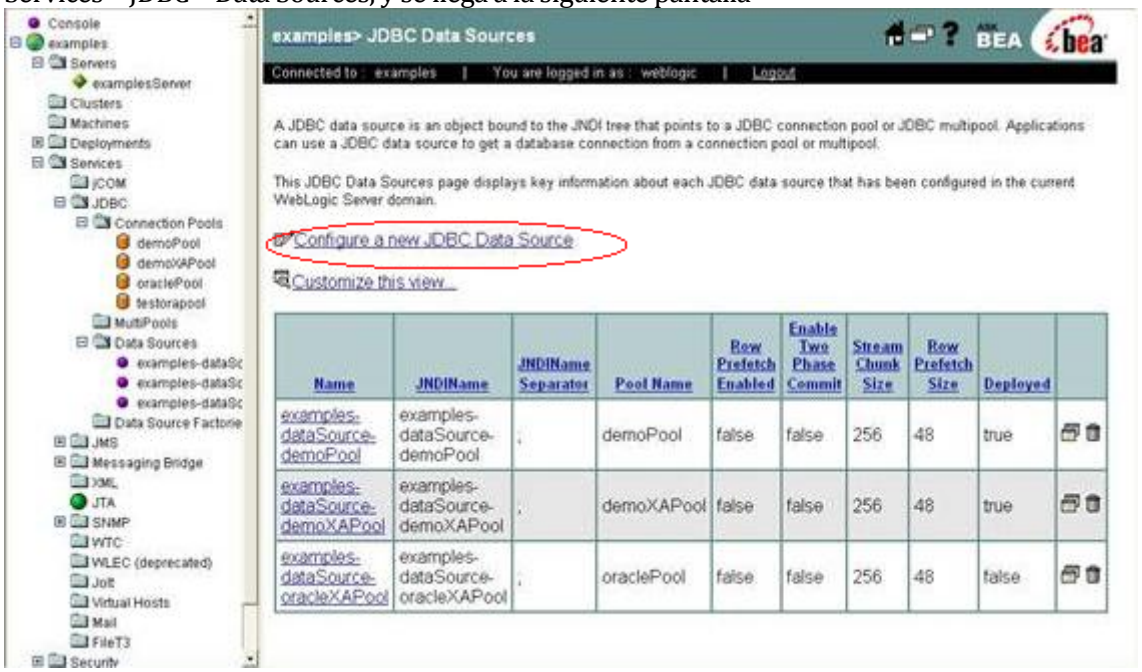


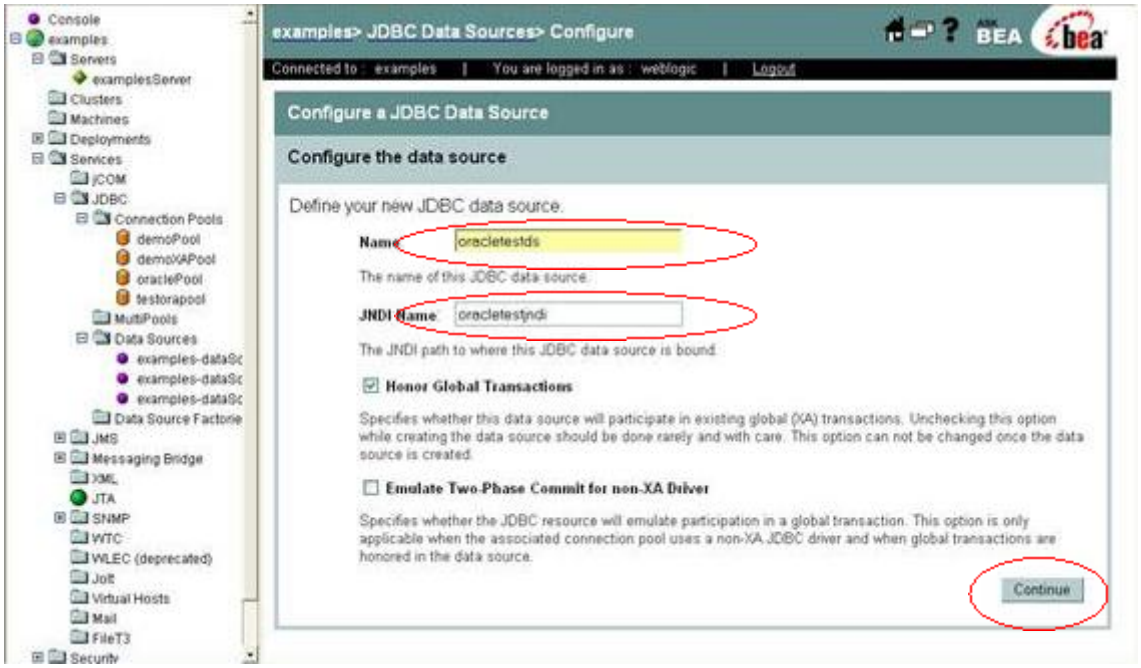
Y listo.



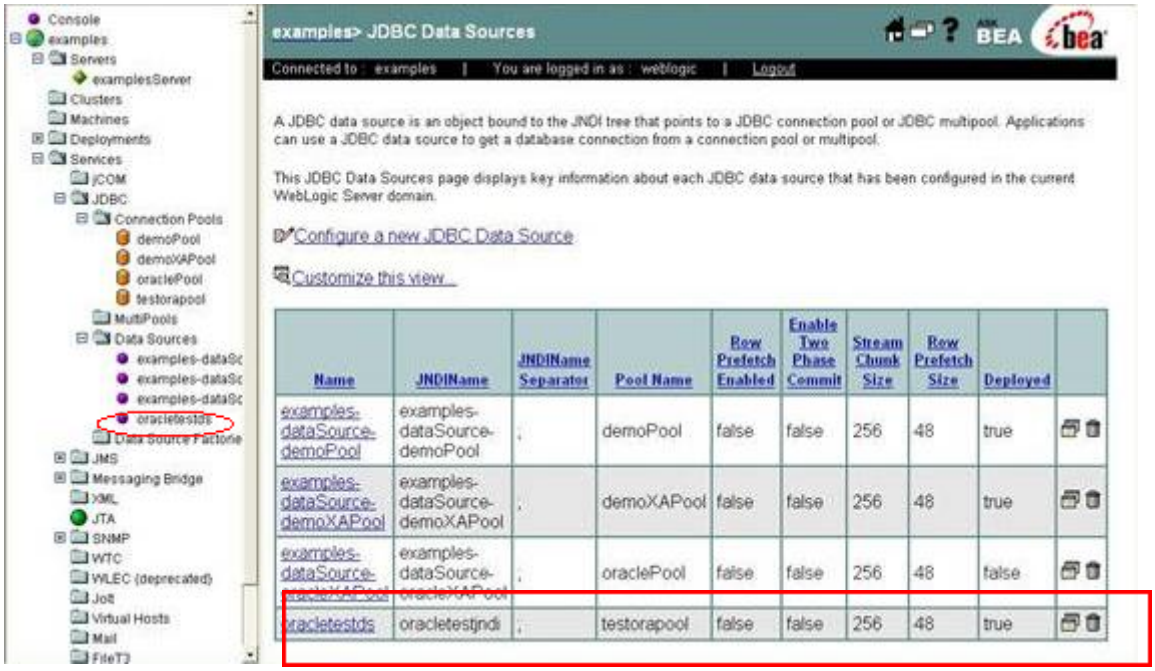
21.5.2 Definición del data Source

Para crear un nuevo data source hacer click en el menu de la izquierda, en la opción Examples – Services – JDBC – Data Sources, y se llega a la siguiente pantalla



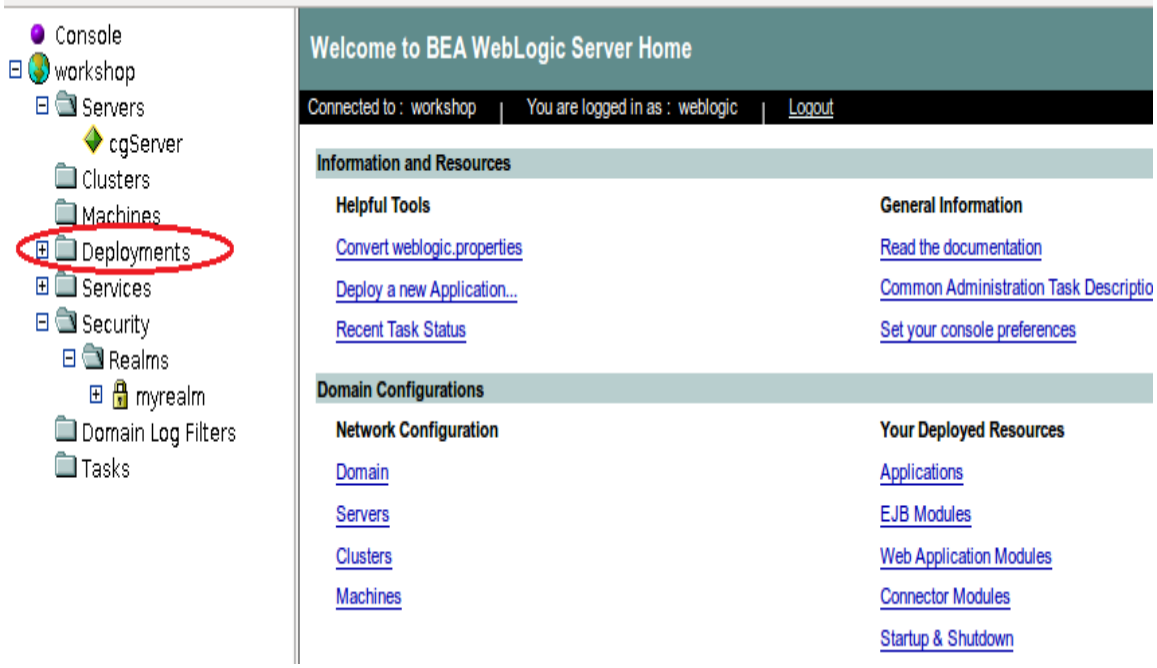


Y listo.

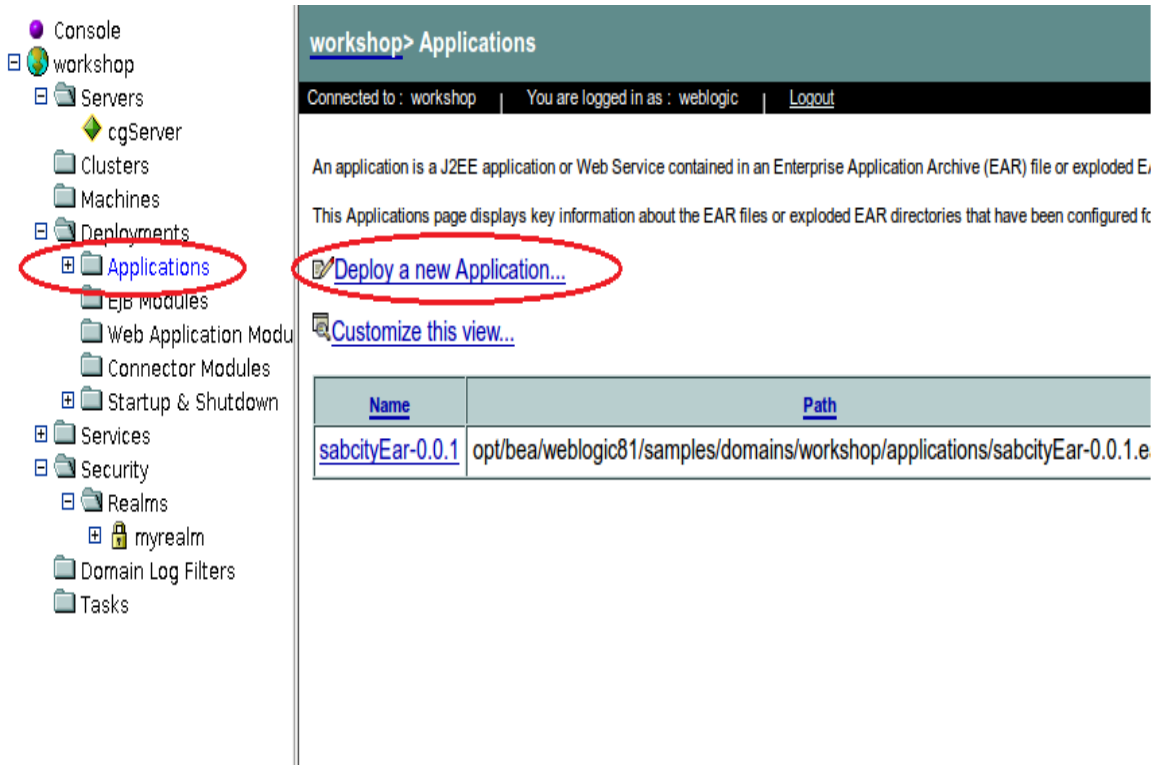


21.6 Instalación de una aplicación .ear

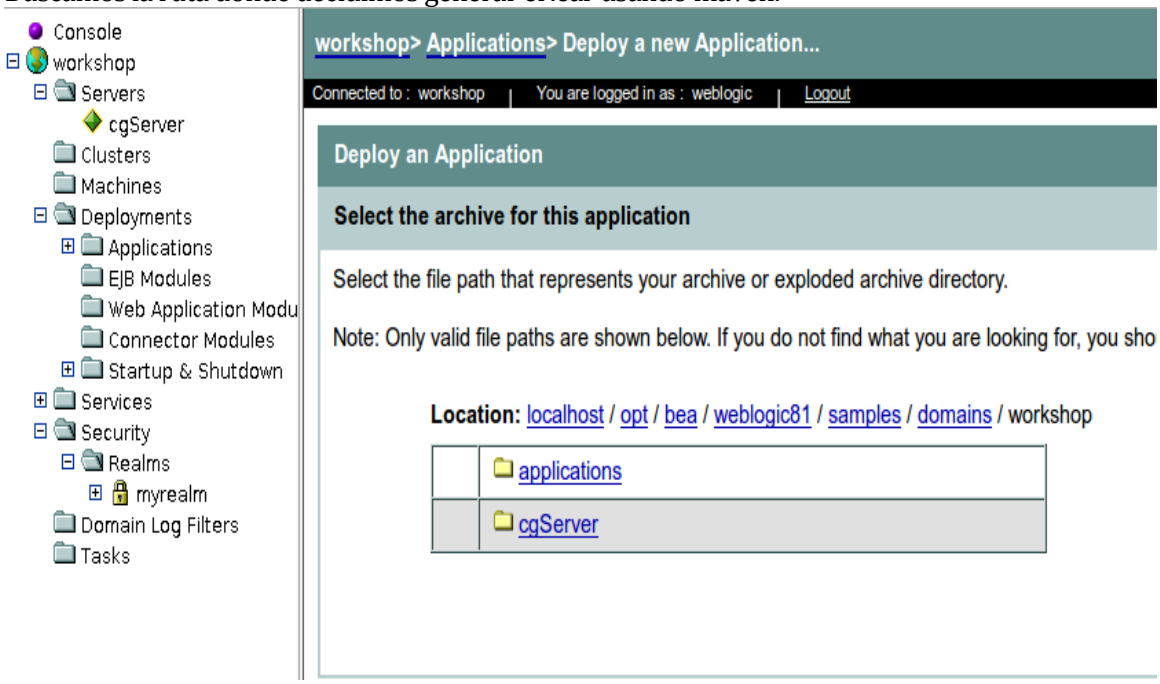
Para poder instalar el .ear y así probar el sistema se debe seguir los siguientes pasos:



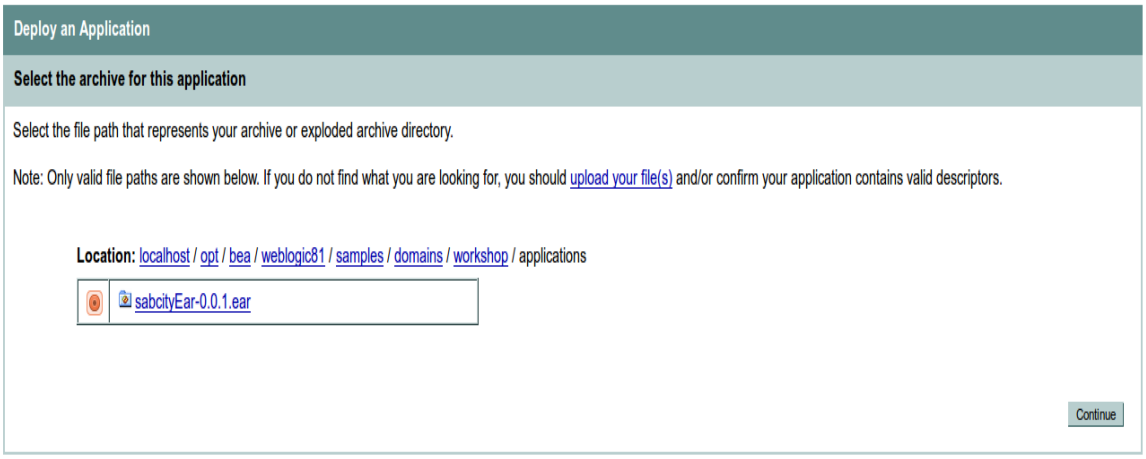
Se debe presionar el sub menú “Applications” y luego seleccionar “Deploy a new Application”.



Buscamos la ruta donde decidimos generar el .ear usando maven.



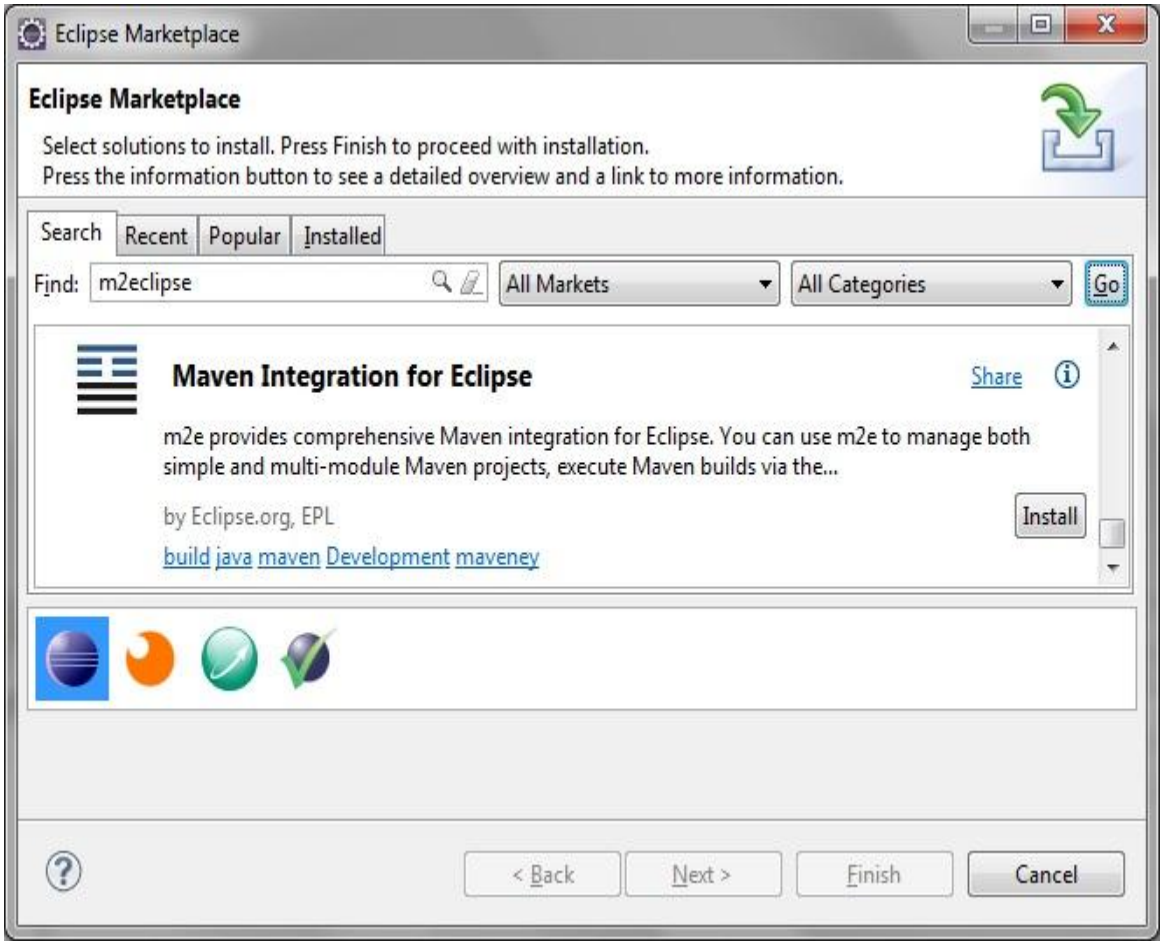
Una vez localizado el archivo lo seleccionamos y presionamos “Continue”



22 CONFIGURACIÓN MAVEN

22.1 Instalar el plugin m2eclipse

Para ello nos vamos al menú "Help>>Eclipse Marketplace", introducimos en el campo de búsqueda "m2eclipse" y le damos a buscar:

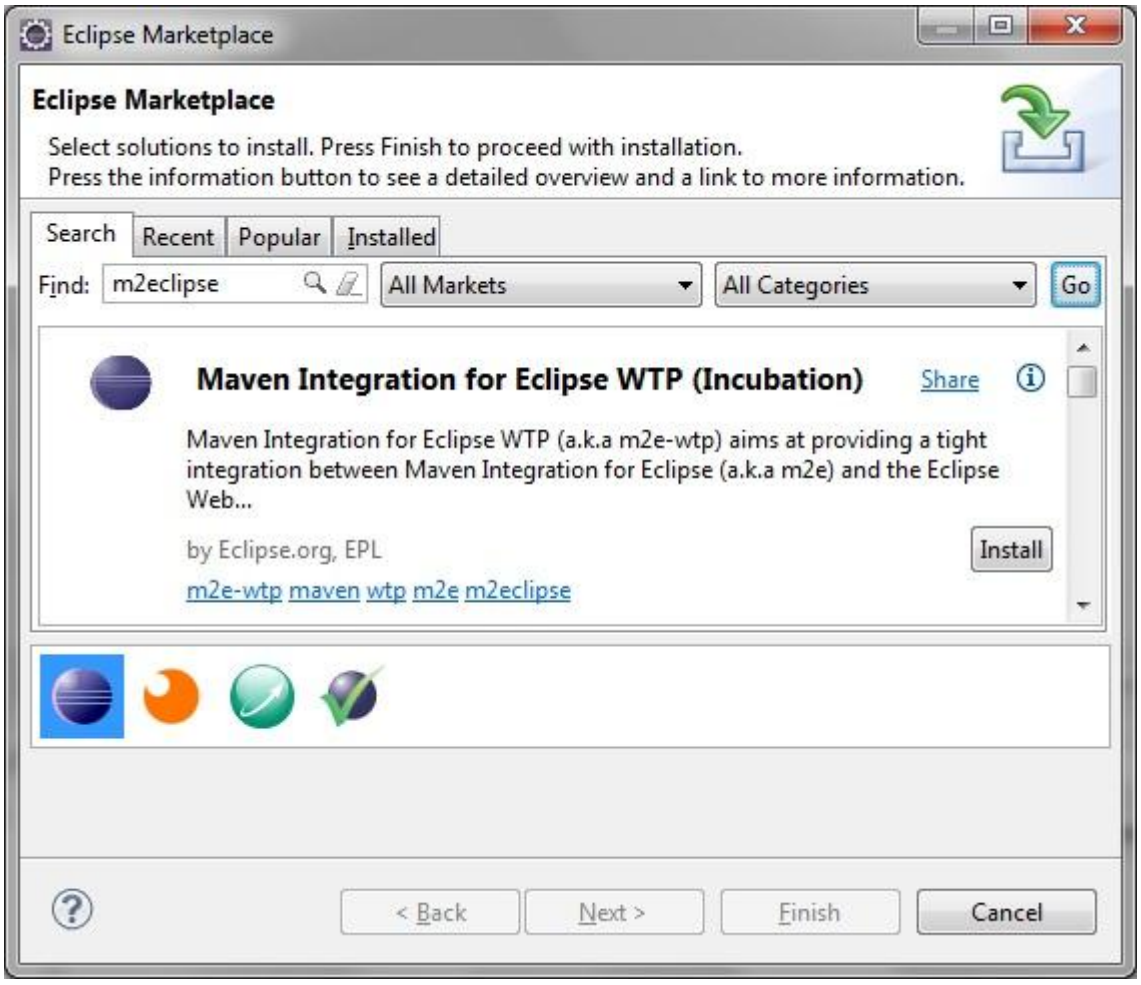


Seleccionaremos el plugin "Maven integration for Eclipse", le damos a siguiente y se nos mostrarán los paquetes que se van a instalar, a continuación tenemos que aceptar los términos y condiciones y finalmente, nos pedirá reiniciar eclipse. Con esto ya tenemos el plugin instalado.

22.2 Instalar el plugin m2eclipse WTP

Vamos a instalar también un plugin que sirve para integrar Maven con las herramientas web de Eclipse. Para ello buscaremos en el Marketplace "m2eclipse", como en el caso anterior. Hay varios

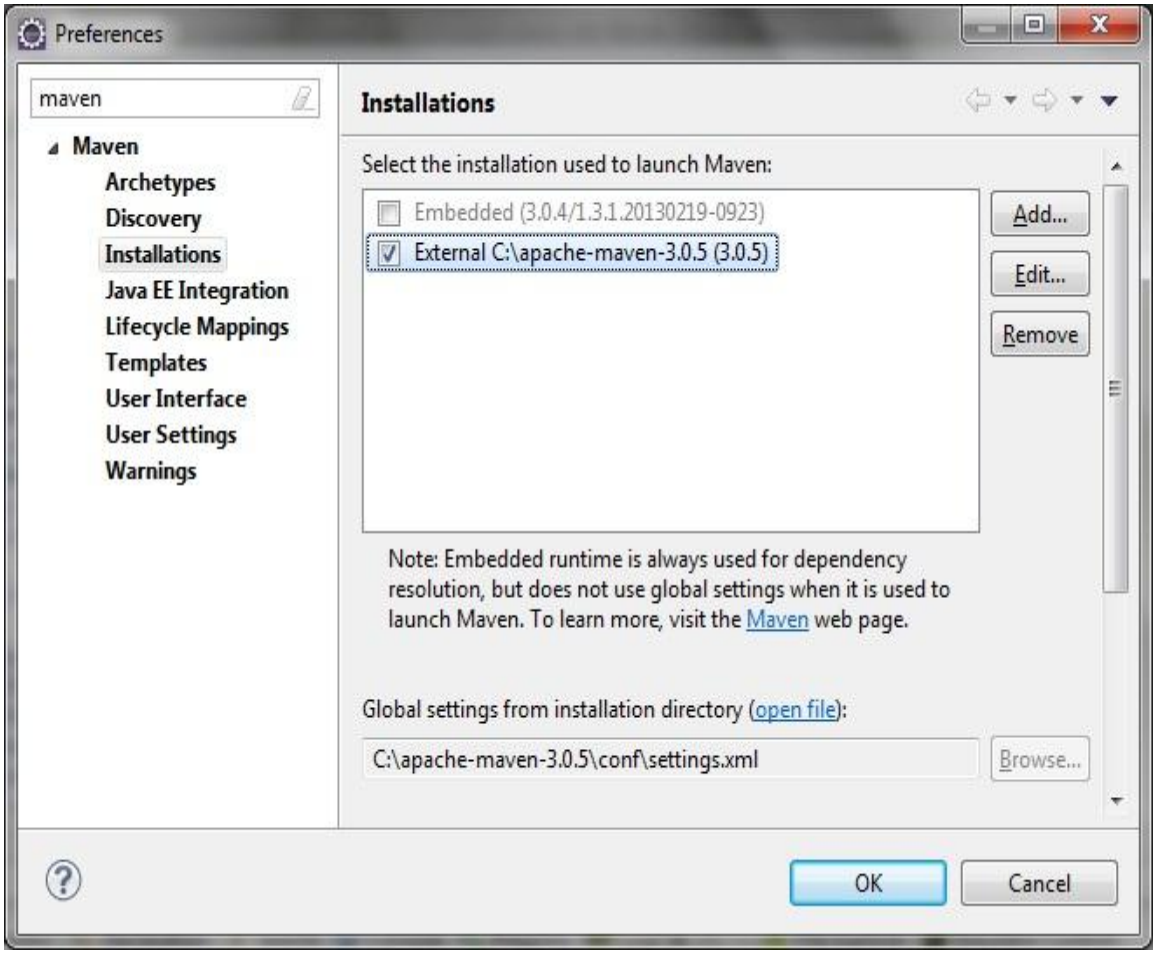
plugins, estoy usando la versión Juno de Eclipse, y voy a seleccionar el siguiente plugin, que va bastante bien:



Procedemos como en el apartado anterior, reiniciamos Eclipse, y ya tenemos el segundo plugin instalado.

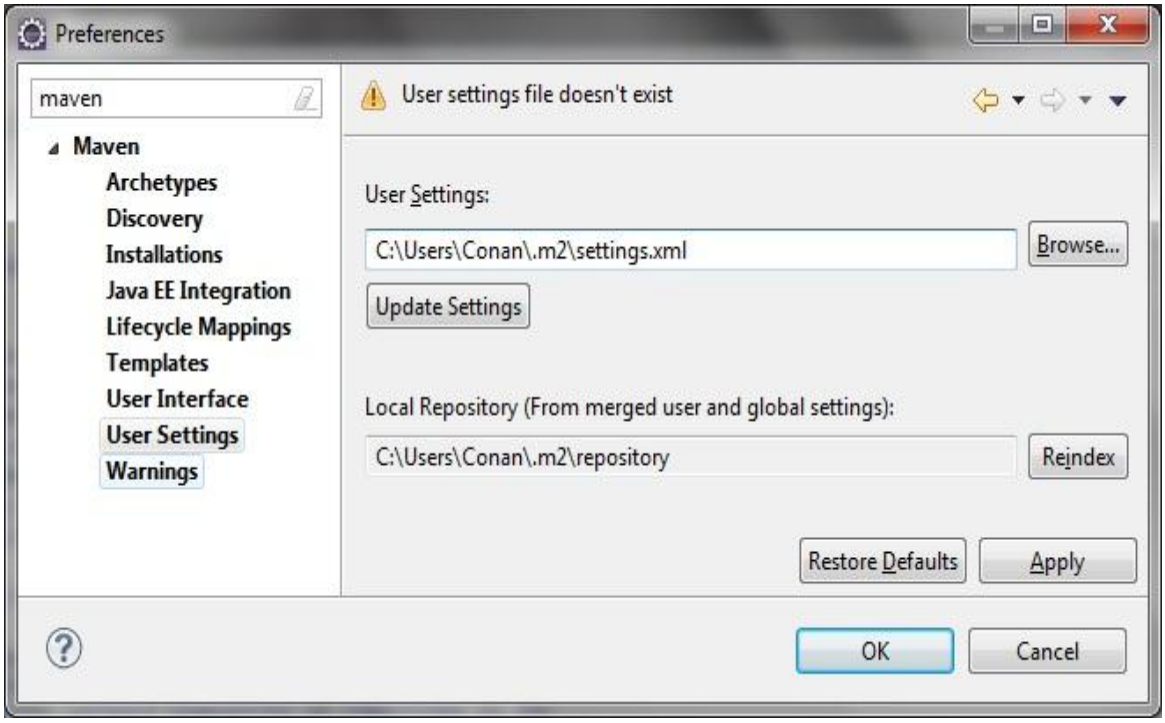
22.3 Configuración del plugin m2eclipse

Accedemos al menú "Windows>>Preferences", escribimos "maven" en el campo de texto que aparece en la esquina superior izquierda y pinchamos en el submenú "Installations". Se nos mostrará la versión de Maven que viene embebida en el plugin. En lugar de usar esa, vamos a seleccionar la que instalamos en el primer tutorial, para ello pinchamos en añadir, seleccionamos la carpeta donde instalamos Maven y aceptamos. Se nos mostrará algo parecido a esto:



Como se puede ver en la ilustración, se nos ha seleccionado el archivo**settings.xml.** Este archivo contiene la configuración **global** del entorno, y puede contener, por ejemplo, los usuarios y contraseñas de acceso al repositorio SVN, datos de acceso al repositorio corporativo, configuración del proxy para acceder a la red, etc.

Del mismo modo, en el apartado "User Settings" podemos configurar otro**settings.xml** con la configuración **particular** de nuestro equipo:



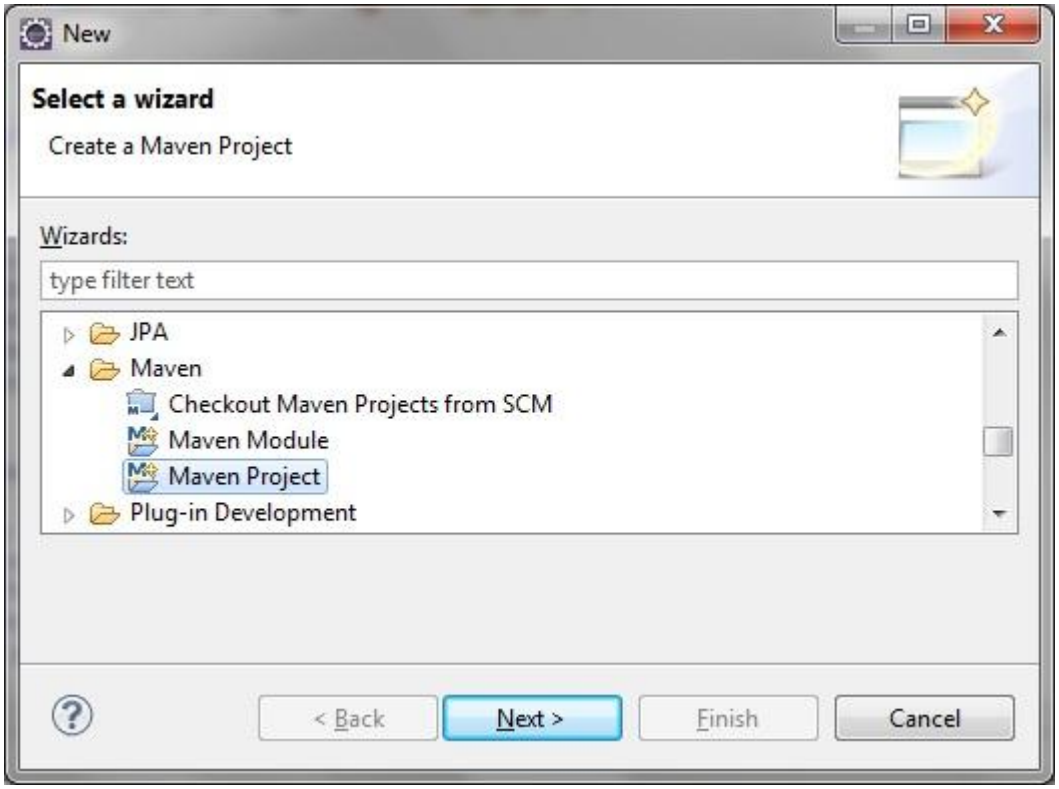
Está apuntando a un settings que no existe, por lo que si deseamos configurar parámetros particulares para nuestro usuario, se puede copiar el settings.xml de la configuración global y editarlo con los parámetros que nos interese.

La idea es que en la configuración global se ajusten los parámetros comunes a todos los programadores de un determinado equipo de desarrollo, y en la configuración a nivel de usuario los datos de configuración que apliquen a un usuario particular. De este modo se pueden definir permisos de acceso a repositorios y este tipo de cosas.

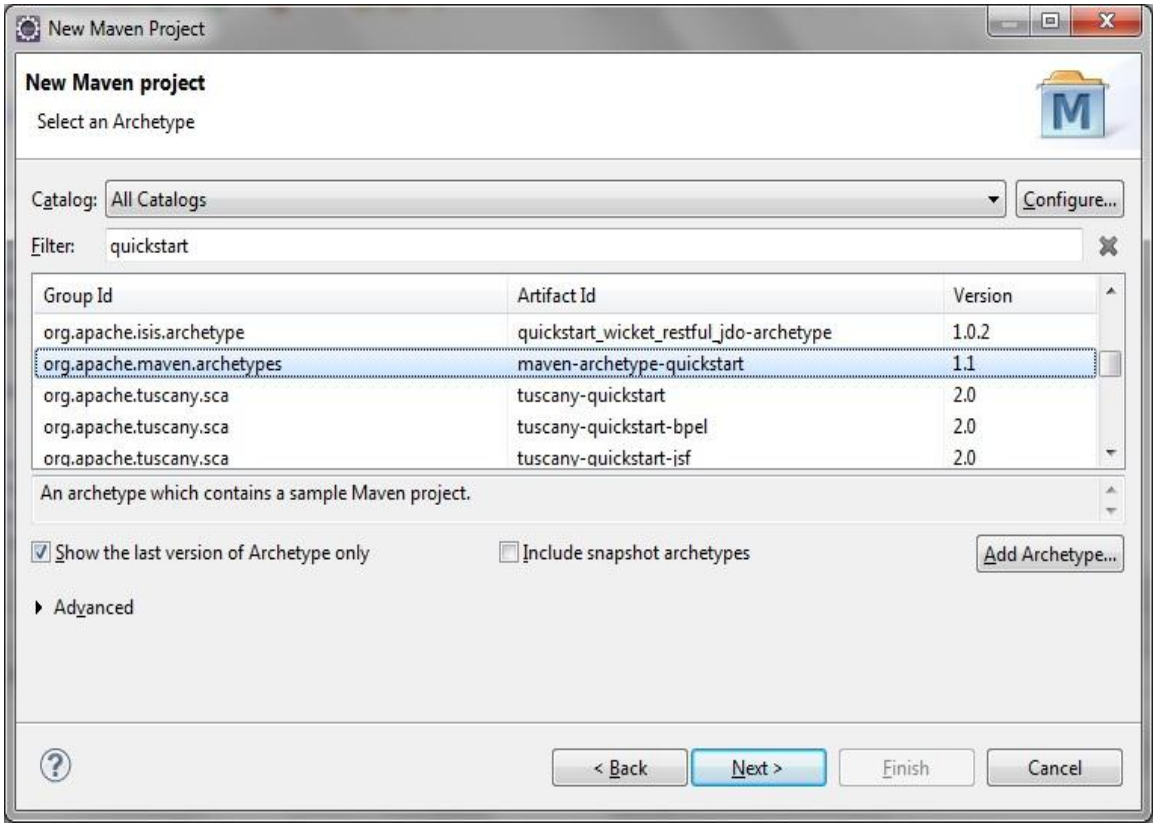
Como se puede ver, apunta al **repositorio local de Maven** (carpeta .m2 en el "home" del usuario). El repositorio local de maven, es una carpeta que va a contener todas las librerías que necesitemos en nuestros proyectos. Por ejemplo, supongamos que en un proyecto añadimos una dependencia a "librería1", maven buscará dicha librería en nuestro repositorio local, si no lo encuentra, se conectará al repositorio central de Maven, y se la descargará, poniéndola en el repositorio local. Veremos esto con más detalle más adelante.

22.4 Crear un nuevo proyecto con m2eclipse

Vamos a crear un nuevo proyecto, como hicimos en el anterior post. Seleccionamos el menú "File>>New>>Other":



Seleccionamos "Maven Proyect" y continuamos. Dejamos las opciones por defecto en la siguiente pantalla y pulsamos siguiente, a continuación seleccionamos el arquetipo "maven-archetype-quickstart" (podemos usar el campo "filter" para ayudarnos a buscar):

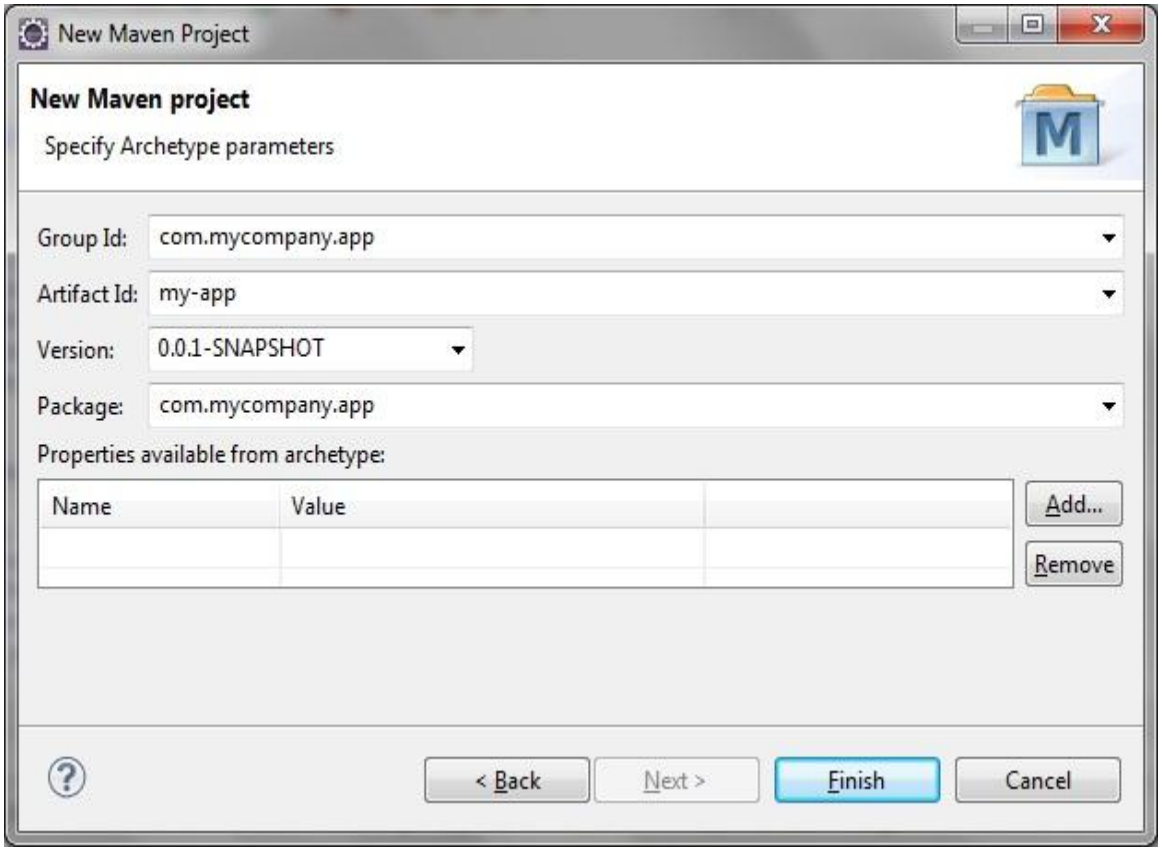


Es posible que no aparezca nada en la lista, eso es debido a que no detecta el repositorio central de Maven, a veces pasa. La solución a esto es añadir en el settings.xml un "mirror" del repositorio central, de este modo:

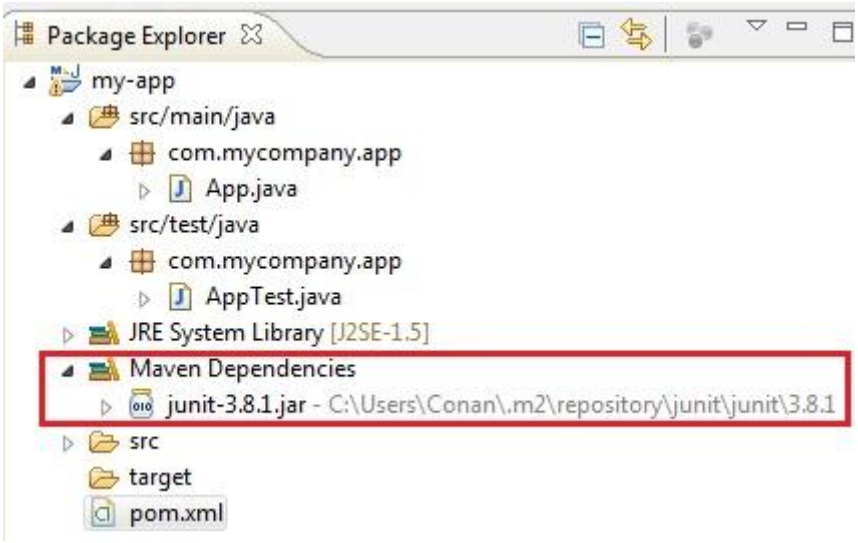
XML | copy code | ?

```
1
2       <mirror>
3       <!-- United Kingdom -->
4         <id>uk.maven.org</id>
5         <url>http://uk.maven.org/maven2</url>
6         <mirrorOf>central</mirrorOf>
7       </mirror>
8
```

Introducimos a continuación los mismos valores que en el ejemplo anterior que hemos realizado por línea de comandos:



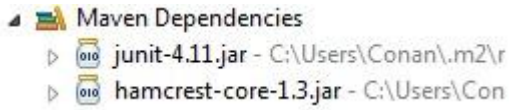
Al pulsar en finish, se nos habrá creado el proyecto:



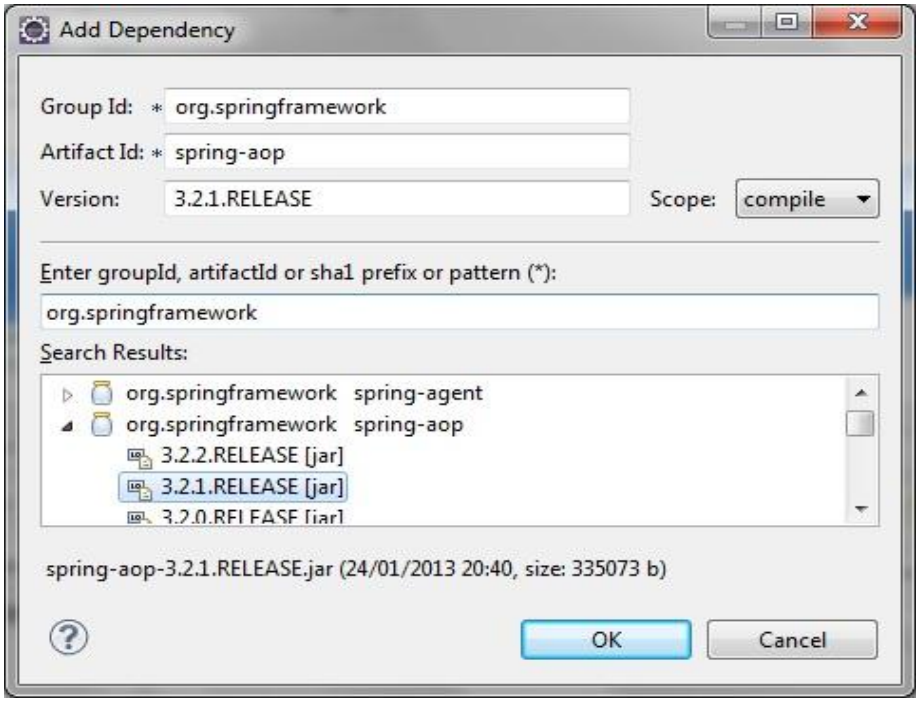
Se detalla en rojo las dependencias de Maven que apuntan a nuestro repositorio local, que como hemos comentado es la carpeta ".m2". Para ver cómo funciona, vamos a cambiar la dependencia de Junit, editamos el archivo **pom.xml,**localizamos la dependencia a junit, y lo dejamos del siguiente modo:

```
XML |        copy code |        ?
1
2        <dependency>
3          <groupId>junit</groupId>
4          <artifactId>junit</artifactId>
5          <version>4.11</version>
6          <scope>test</scope>
7        </dependency>
8
```

Si observamos de nuevo las dependencias de nuestro proyecto, vemos que además de cambiar la versión de la librería junit, se nos ha añadido una nueva librería. Esto se debe a que la nueva versión necesita dicha librería. Es lo que se llama **dependencia transitiva**, es decir, cuando utilizas una librería Maven automáticamente te descarga las librerías de las que depende.

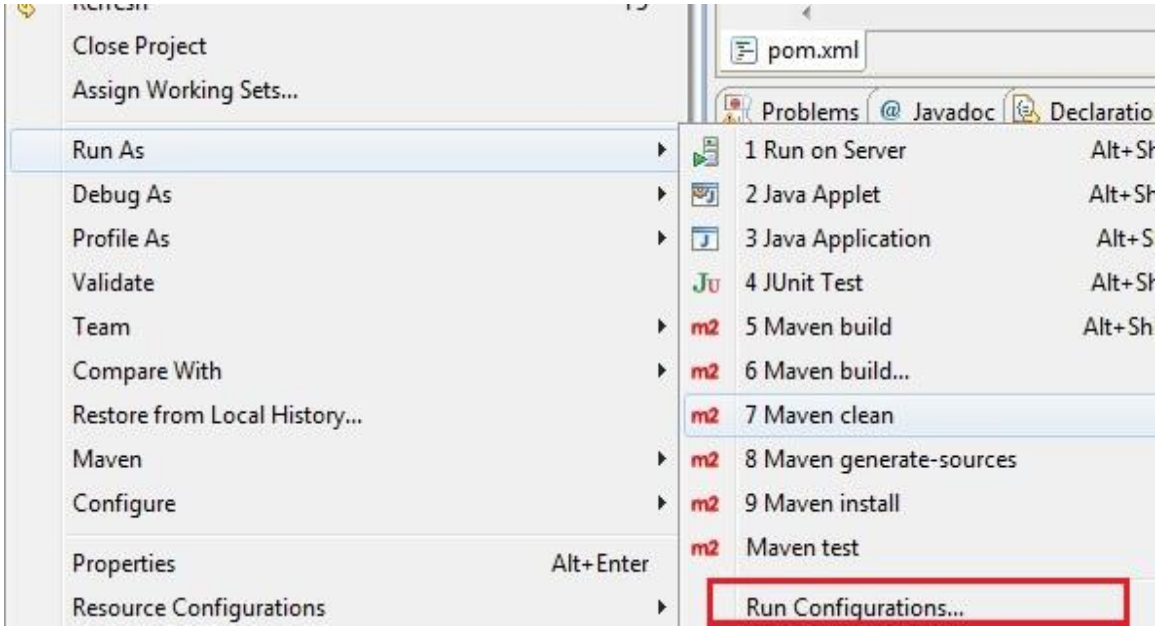


También se pueden añadir dependencias sin necesidad de editar el pom.xml directamente. Pinchando con el botón derecho del ratón sobre el proyecto y seleccionando en el menú emergente: "Maven>>Add Dependency" se nos abrirá un asistente desde el que podemos buscar las librerías

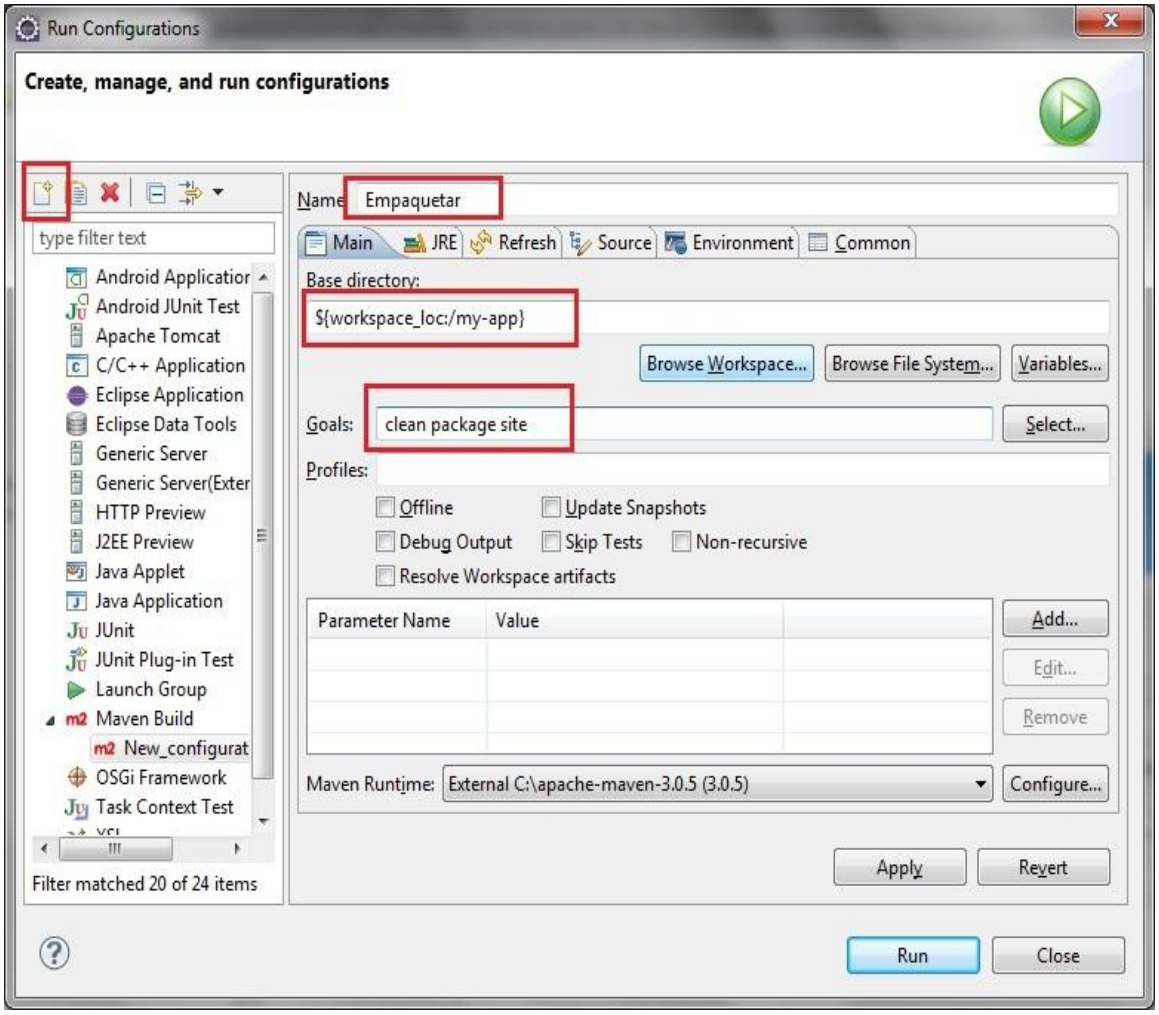


22.5 Empaquetar el proyecto

Vamos a empaquetar y compilar el proyecto, pero en lugar de hacerlo por línea de comandos, lo haremos usando Eclipse. Pulsando sobre el proyecto, con el botón derecho tenemos las opciones "Run as/Debug as" de Maven:



Queremos más opciones, así que vamos a pinchar en "Run Configurations". Cuando se abra el asistente le damos a "New launch configuration", seguidamente le damos un nombre a la configuración (Empaquetar), indicamos el directorio base y en el apartado "Goals" indicamos lo que nos interese, en este caso queremos limpiar la carpeta "target", empaquetar y crear el site (clean package site):



Al pulsar el botón "Run", como en el caso de la ejecución por línea de comandos, en la carpeta "target" se genera un jar, así como los html con la documentación de nuestro proyecto.