INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA DE ANÁLISIS DE SISTEMAS

DESARROLLAR UNA APLICACIÓN MÓVIL CON SISTEMA OPERATIVO ANDROID PARA REGISTRO Y CONTROL DE INVENTARIO EN LA EMPRESA SIVADE (SISTEMAS INTEGRADOS DE VIDEO, AUDIO, Y DATOS DEL ECUADOR).

Proyecto de investigación previo a la obtención de título de tecnólogo Analista de Sistemas.

Autor: Peñafiel Espinosa Jaime Jhofre

Tutor: Dr. Luis Ríos

Quito, Abrilde 2014

DECLARATORIA

Declaro que la investigación es absolutamente original, autentica, personal, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes. Las ideas, doctrinas resultados y conclusiones a los que he llegado son de mi absoluta responsabilidad.

.

_____

Peñafiel Espinosa Jaime Jhofre

C.C. 172372638-4

DECLARACIÓN DE APROBACIÓN TUTOR

En mi calidad de tutor del trabajo sobre el tema: "Desarrollar una aplicación móvil con sistema operativo Android para registro y control de inventario en la empresa SIVADE (Sistemas integrados de video, audio, y datos del Ecuador).", presentado por el ciudadano: Peñafiel Espinosa Jaime Jhofre, estudiante de la Escuela de Análisis y Sistemas, considero que dicho informe reúne los requisitos y méritos suficientes para ser sometido a la evaluación por parte del Tribunal de Grado, que el Honorable Consejo de Escuela designe, para su correspondiente estudio y calificación.

Quito, Abril del 2014

_____	_____
Dr. Luis Ríos	Ing. Jaime Padilla
TUTOR	LECTOR

CONTRATO DE CESIÓN SOBRE DERECHOS PROPIEDAD INTELECTUAL

Comparecen a la celebración del presente contrato de cesión y transferencia de derechos de propiedad intelectual, por una parte, el estudiante JAIME JHOFRE PEÑAFIEL ESPINOSA, por sus propios y personales derechos, a quien en lo posterior se le denominará el "CEDENTE"; y, por otra parte, el INSTITUTO SUPERIOR TECNOLÓGICO CORDILLERA, representado por su Rector el Ingeniero Ernesto Flores Córdova, a quien en lo posterior se lo denominará el "CESIONARIO". Los comparecientes son mayores de edad, domiciliados en esta ciudad de Quito Distrito Metropolitano, hábiles y capaces para contraer derechos y obligaciones, quienes acuerdan al tenor de las siguientes cláusulas:

PRIMERA: ANTECEDENTE.- a) El Cedente dentro del pensum de estudio en la carrera de análisis de sistemas que imparte el Instituto Superior Tecnológico Cordillera, y con el objeto de obtener el título de Tecnólogo en Análisis de Sistemas, el estudiante participa en el proyecto de grado denominado "**DESARROLLAR UNA APLICACIÓN MÓVIL CON SISTEMA OPERATIVO ANDROID PARA REGISTRO Y CONTROL DE INVENTARIO EN LA EMPRESA SIVADE (SISTEMAS INTEGRADOS DE VIDEO, AUDIO, Y DATOS DEL ECUADOR).**", el cual incluye la creación y desarrollo del programa de ordenador o software, para lo cual ha implementado los conocimientos adquiridos en su calidad de alumno. b) Por iniciativa y responsabilidad del Instituto Superior Tecnológico Cordillera se desarrolla la creación del programa de ordenador, motivo por el cual se regula de forma clara la cesión de los derechos de autor que genera la obra literaria y que es producto del proyecto de grado, el mismo que culminado es de plena aplicación técnica, administrativa y de reproducción.

SEGUNDA: CESIÓN Y TRANSFERENCIA.- Con el antecedente indicado, el Cedente libre y voluntariamente cede y transfiere de manera perpetua y gratuita todos los derechos patrimoniales del programa de ordenador descrito en la clausula anterior a favor del Cesionario, sin reservarse para sí ningún privilegio especial (código fuente, código objeto, diagramas de flujo, planos, manuales de uso, etc.). El Cesionario podrá explotar el programa de ordenador por cualquier medio o procedimiento tal cual lo establece el Artículo 20 de la Ley de Propiedad Intelectual, esto es, realizar, autorizar o prohibir, entre otros: a) La reproducción del programa de ordenador por cualquier forma o procedimiento; b) La comunicación pública del software; c) La distribución pública de ejemplares o copias, la comercialización, arrendamiento o alquiler del programa de ordenador; d) Cualquier transformación o modificación del programa de ordenador; e) La protección y registro en el IEPI el programa de ordenador a nombre del Cesionario; f) Ejercer la protección jurídica del programa de ordenador; g) Los demás derechos establecidos en la Ley de Propiedad Intelectual y otros cuerpos legales que normen sobre la cesión de derechos de autor y derechos patrimoniales.

TERCERA: OBLIGACIÓN DEL CEDENTE.- El cedente no podrá transferir a ningún tercero los derechos que conforman la estructura, secuencia y organización del programa de ordenador que es objeto del presente contrato, como tampoco emplearlo o utilizarlo a título personal, ya que siempre se deberá guardar la exclusividad del programa de ordenador a favor del Cesionario.

CUARTA: CUANTIA.- La cesión objeto del presente contrato, se realiza a título gratuito y por ende el Cesionario ni sus administradores deben cancelar valor alguno o regalías por este contrato y por los derechos que se derivan del mismo.

QUINTA: PLAZO.- La vigencia del presente contrato es indefinida.

SEXTA: DOMICILIO, JURISDICCIÓN Y COMPETENCIA.- Las partes fijan como su domicilio la ciudad de Quito. Toda controversia o diferencia derivada de éste, será resuelta directamente entre las partes y, si esto no fuere factible, se solicitará la asistencia de un Mediador del Centro de Arbitraje y Mediación de la Cámara de Comercio de Quito. En el evento que el conflicto no fuere resuelto mediante este procedimiento, en el plazo de diez días calendario desde su inicio, pudiendo prorrogarse por mutuo acuerdo este plazo, las partes someterán sus controversias a la resolución de un árbitro, que se sujetará a lo dispuesto en la Ley de Arbitraje y Mediación, al Reglamento del Centro de Arbitraje y Mediación de la Cámara de comercio de Quito, y a las siguientes normas: a) El árbitro será seleccionado conforme a lo establecido en la Ley de Arbitraje y Mediación; b) Las partes renuncian a la jurisdicción ordinaria, se obligan a acatar el laudo arbitral y se comprometen a no interponer ningún tipo de recurso en contra del laudo arbitral; c) Para la ejecución de medidas cautelares, el árbitro está facultado para solicitar el auxilio de los funcionarios públicos, judiciales, policiales y administrativos, sin que sea necesario recurrir a juez ordinario alguno; d) El procedimiento será confidencial y en derecho; e) El lugar de arbitraje serán las instalaciones del centro de arbitraje y mediación de la Cámara de Comercio de Quito; f) El idioma del arbitraje será el español; y, g) La reconvención, caso de haberla, seguirá los mismos procedimientos antes indicados para el juicio principal.

SÉPTIMA: ACEPTACIÓN.- Las partes contratantes aceptan el contenido del presente contrato, por ser hecho en seguridad de sus respectivos intereses.

En aceptación firman a los 17 días del mes de abril del dos mil catorce.

f) _____	f) _____
C.C. 1723726384	Instituto Superior Tecnológico Cordillera
CEDENTE	**CESIONARIO**

AGRADECIMIENTO

Agradezco a Dios, a mis Padres, por la oportunidad de existir y los estímulos brindados con infinito amor, gracias a ellos ha sido posible la culminación de esta etapa de mi vida.

A todas las personas con las que he podido compartir esta etapa tecnológica de los cuales he logrado aprender mucho.

A mi asesor el Ingeniero Jaime Basantes por el apoyo brindado en todo momento.

A los amigos y compañeros que compartieron conmigo la vida institucional y fueron participes de cada experiencia, gracias por permitirme aprender de ustedes.

DEDICATORIA

Este proyecto quiero dedicar principalmente a Dios .

Y en segundo lugar a mi madre, a mi familia

que siempre han estado con migo apoyándome y dándome fuerzas

para seguir adelante con mis estudios y mis metas

También quiero nombrara en esta dedicatoria

a una persona muy especial.

Vanessa Ontaneda quien también fue un pilar muy importante

tanto en vida como en mis estudios.

GRACIAS.

ÍNDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN EJECUTIVO

Se desarrolló una aplicación móvil con sistema operativo Android para registro y control de inventario en la empresa SIVADE (Sistemas integrados de video, audio, y datos del Ecuador).

El desarrollo de esta aplicación, significó realizar un estudio y análisis previo con el fin de llevar correctamente el inventario y sus componentes necesarios para poder utilizar correctamente el hardware y el software, que conforma la aplicación.

Esta aplicación ayudara a los agentes promotores de ventas tener el conocimiento de las existencias en los inventarios de la empresa SIVADE para su comercialización y fácil distribución optimizando la actividad de ventas.

El desarrollo de la aplicación podría ser considerado el inicio del automatización moderna del control de inventarios esto ayudara a diferentes empresas a llevar un control estricto de las existencias y se convertirá en una verdadera herramienta en la toma de decisiones y corrección de errores en el proceso productivo de la empresa que lo implemente

ABSTRACT

A mobile application developed with operative system Android for record and inventor control in the company SIVADE (integrated Systems of video, audio, and information of the Ecuador).

The development of this application, it meant to realize a study and previous analysis in order to take correctly the inventory and his components necessary to be able to use correctly the hardware and the software, which shapes the application. This application was helping the agents promoters of sales to have the knowledge of the stock in the inventories of the company SIVADE for his marketing and easy distribution optimizing the activity of sales.

The development of the application might be considered the beginning of modern automatización of the inventor control this was helping to different companies to take a strict control of the stock and it will turn into a real tool into the capture of decisions and correction of mistakes into the productive process of the company that implements it

CAPÍTULO I: ANTECEDENTES

1.01 Contexto

La empresa SIVADE, es una empresa que se encuentra ubicada en la ciudad de Quito, sector La Carolina Av. 10 de Agosto 5282 y Naciones Unidas Edef. Comandato Torre Iñaquito 2do piso ofc:206. Se dedica a la importación de video, audio, y datos La empresa realiza venta de equipos instalación, y transmisión de los equipos. Esta constituida por diferentes niveles como el directivo formado por presidencia, gerencia, y el área legal. El nivel administrativo formado por soporte, contabilidad, adquisiciones, importaciones, y ventas La misma que tiene una amplia gama de productos en sus bodegas y por lo cual llevan su sistema de inventariado en un computador el cual permanece siempre en la oficina de adquisiciones, esto es un problrma por que el personal de ventas tenga que estar constantemente llamando a la oficina para averiguar si los productos que los clientes le solicitan se encuentran en bodega.

Dicho Sistema ayudara a los agentes a tener una idea clara y concisa de las existencias de la empresa en tiempo real y en cualquier parte del país donde se encuentre y lograra consolidar de forma fácil y rápida las posibles ventas de los productos que ofrece la empresa.

Llevar un control adecuado de las mercaderías que entran y salen de las bodegas y la toma de decisiones por los mandos directivos y la empresa sea competitiva en el mercado.

1.02 Justificación e Importancia

El proyecto que se ha propuesto será de gran impacto para la sociedad por ser un software movil que ayudara en gran parte a las empresas en el Ecuador a llevar su sistema de inventarios en un dispositivo movil. El software solucionara el problema de las empresas que tienen cuando se trata de revisar el inventario de los productos que ofrecen, y tienen que estar llamando constantemente a la empresa a pedir información de la existencia de los mismos. En caso de no implementar el proyecto dichas empresas tendrían que seguir manteniendo el inventario como lo han venido haciendo o buscar nuevas alternativas

Con este sistema se reducirá el tiempo de consulta, se agilitara el proceso de oferta y venta de los productos que ofrece la empresa, de hecho se podrá registrar en el sistema los pedidos ya que contara con un hostin en la web para poder acceder desde cualquier parte en la que se encuentre el usuario.

La Empresa **SIVADE** implementara una mejora en el sistema de consulta de inventarios para los agentes promotores de ventas. Dicho sistema está desarrollado en la plataforma de android para poder ser utilizado en dispositivos móviles ya sean estos Smartphone o tablets.

1.03 Matriz T

Tabla: 1 Matriz T

ANALISIS DE FUERZAS T					
Situación Empeorada	Situación Actual				Situación Mejorada
Falta de agilidad en los procesos de ventas	La empresa lleva su inventario en un computador de oficina de adquisiciones				Mejora el sistema de inventario que también se podrá consultar en Smartphone's
Fuerzas Impulsadoras	I	PC	I	PC	Fuerzas Bloqueadoras
Una mejor administración	1	4	4	2	Falta de recursos
Control constante	2	3	4	5	Poco interés
Garantías de seguridad	1	4	4	5	Mal ingreso de información
Aumento de las ventas	1	4	4	5	Bajo control de productos
Respaldos de información	1	1	3	1	Falta de herramientas tecnológicas
Conocer el ambiente laboral	2	4	4	5	Trabajo bajo presión

1 **Bajo**
2 **Medio bajo**
3 **Medio**
4 **Medio alto**
5 **Alto**

Análisis de la matriz (T)

La empresa SIVADE TIENE una deficiencia en el control, manejo y consulta de los inventarios de los productos que oferta, el programa a desarrollar tiene el fin de controlar de mejor manera el stock elevando el interés del los involucrados, garantizando seguridad, aumentando las ventas y mejorando el ambiente laboral.

CAPÍTULO II: ANÁLISIS DE INVOLUCRADOS

2.01 Mapeo de Involucrados

En el análisis de involucrados estamos tomando en cuenta todos los actores que intervienen en el desarrollo del proyecto dividiéndolos en dos partes primordiales:

Entorno Interno

El Directivo: en este nivel de involucrados tomamos en cuenta los tres niveles mas altos de la empresa como son Presidente, Gerente y el área legal;

El Área Administrativa.- esta parte de la empresa esta fundamentada por las instancias necesarias para el correcto funcionamiento de la empresa como son soporte, contabilidad, adquisiciones ventas, importaciones y mantenimiento

Entorno Externo

Esta fase del recurso humano es independiente del funcionamiento cotidiano de la empresa pero fundamental para la realización del proyecto de sistematización del inventario y esta compuesto por ITSCO y el agente de solución

Con el correcto funcionamiento y colaboración de cada una de las instancias podremos llegar a un feliz término en la implementación del sistema nuevo de inventarios

(Ver Anexo A.01) ver página 72

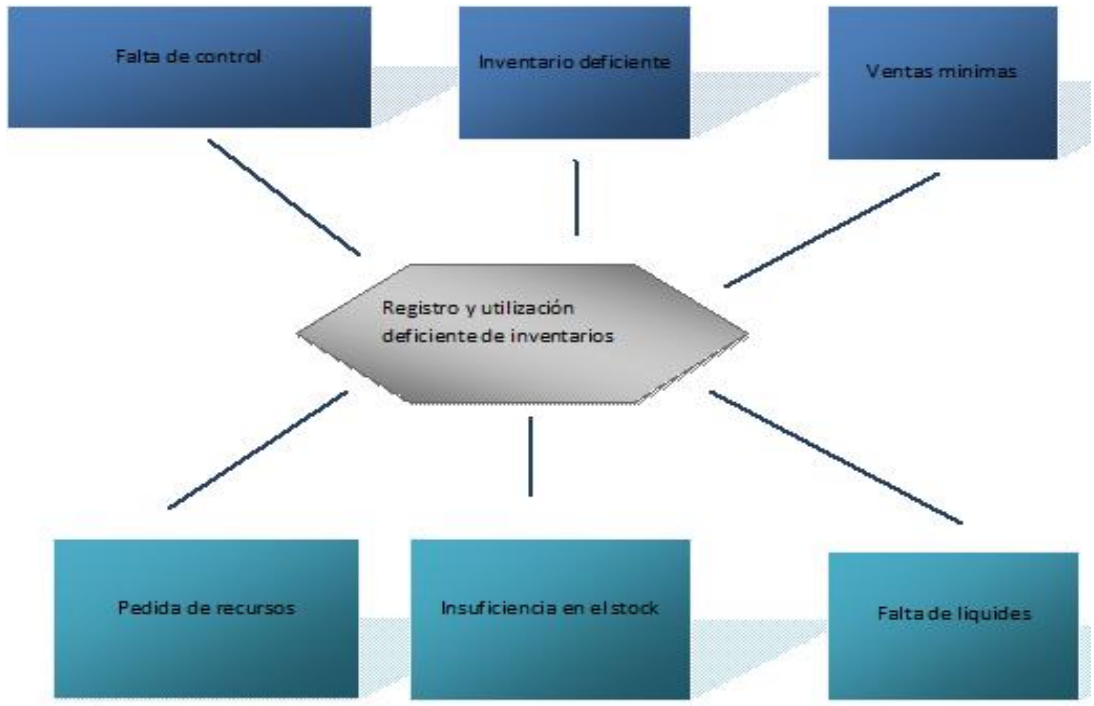
2.02 Matriz de Análisis de Involucrados

Todos los niveles involucrados con la aplicación móvil con sistema operativo Android para registro y control de inventario en la empresa SIVADE se comprometen para la mejora continua de las actividades realizadas para el mejoramiento institucional, esto comprende la corrección de delos errores y la pontencialización de las fortalezas de la empresa, la reducción de los tiempos de ventas y entrega de mercaderías manejando de una manera optima los ingresos y egresos de las mercaderías en existencias

(Ver Anexo A.02) ver página 73

CAPÍTULO III: PROBLEMAS Y OBJETIVOS

3.01 Árbol de Problemas.



Fugura: 2 Árbol de problemas

El árbol de problesmas sirve principalmente para tener claro las causas de por que el problema

Las deficiencias que se encontraron en el mantenimiento del inventario son la principal debilidad de la empresa estas fallas están comprendidas principalmente en la falta de control de la existencias, un inventario deficiente, perdida de tiempo y recursos, insuficiencia en stock y falta de liquedes que nos da como resultado un pobre desempeño en las ventas el presente proyecto pretende crear un sistema de control de inventarios que optimice el manejo de recursos y se desempeñen de forma integra para el crecimientos empresarial del negocio

3.02 Árbol de Objetivo

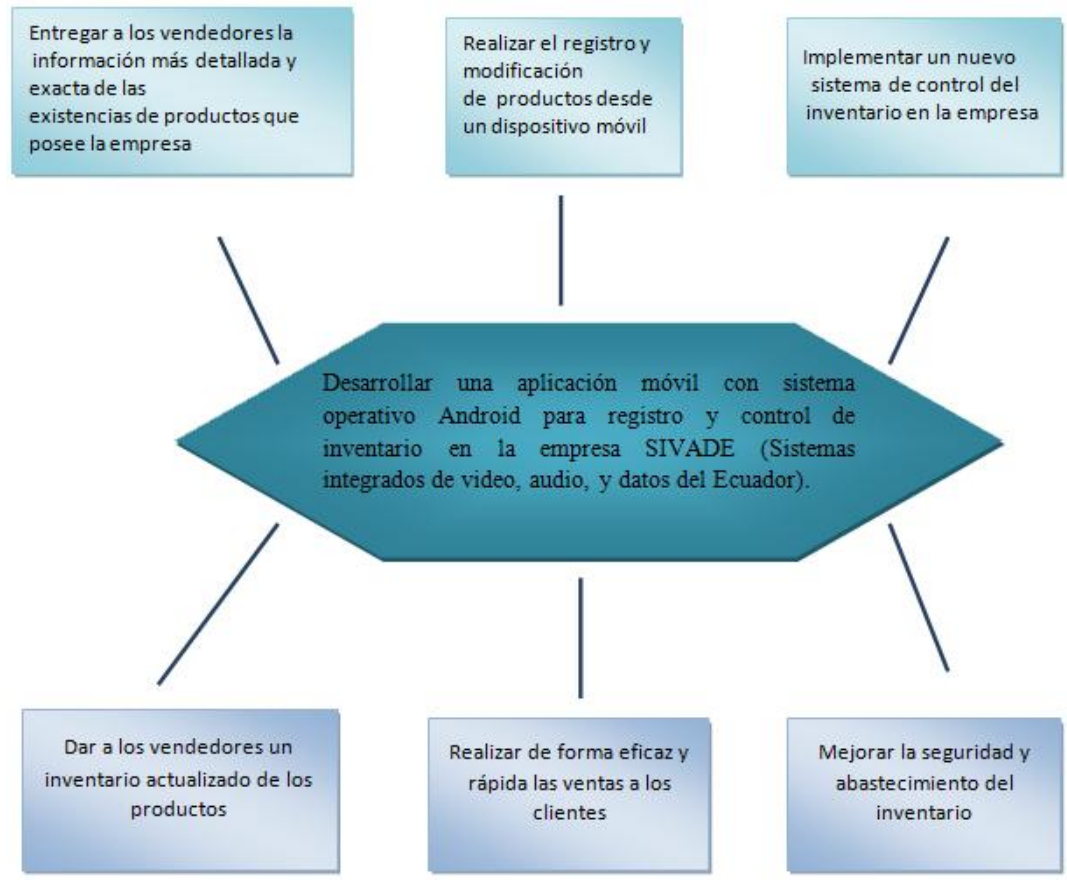


Figura: 3 Arbol de objetivos

Son la metas a cumplir en la empresa implementando el programa que se a propuesto.

Al desarrollar el sistema operativo Android para registro y control de inventario en la empresa SIVADE (Sistemas integrados de video, audio, y datos del Ecuador). Estamos creando una importante herramienta en el campo empresarial para el correcto uso y mantenimientos de los inventarios. Permitiendo un correcto registro de las existencias que ingresan y salen del inventario, un listado de las preventas realizadas por los agentes vendedores tomando en cuenta las existencias en bodega para realizar ventas mas rápidas y agiles en tiempo y en agilidad, logrando así las perdidas en el inventario y por ende perdida económicas

CAPITULO IV: ANÁLISIS DE ALTERNATIVAS

4.01 Matriz de Análisis de Alternativas.

Tabla: 3 Matriz de análisis de alternativas.

OBJETIVOS	IMPACTOS SOBRE EL PROPÓSITO	FACTIBILIDAD TÉCNICA	FACTIBILIDAD FINANCIERA	FACTIBILIDAD SOCIAL	FACTIBILIDAD POLÍTICA	TOTAL	CATEGORIAS
Mejoramiento en tiempo de ventas	4	3	4	4	3	18	ALTA
Optimización de inventarios	4	4	3	3	1	15	MEDIA
Mejoramiento en el uso de los recursos	4	4	4	2	1	15	MEDIA
Control de ingresos y egresos	4	3	4	3	1	15	MEDIA
TOTAl	16	14	15	10	5	75	

12-14 Bajo (B) si la alternativa no cumple el criterio;

15-17 Medio (M) si la alternativa cumple medianamente el criterio;

18-20 Alto (A) si la alternativa cumple totalmente el criterio.

El correcto desenvolvimiento del proyecto y todas sus instancias nos da como resultado un abanico de alternativas y mientras más cerca de la realidad sean las metas se ven cristalizadas en áreas que son primordiales como son impactos sobre el propósito, factibilidad técnica, factibilidad financiera, factibilidad social y factibilidad política

4.02 Matriz de Análisis de Impacto de los Objetivos

Objetivos	*Factibilidad de Lograse (Alta..Media-Baja) (4 - 2 - 1)*	*Impacto en Género (Alta–Media–Baja) (4 - 2 - 1)*	*Impacto Ambiental (Alta-Media-Baja) (4 - 2 - 1)*	*Relevancia (Alta-Media-Baja) (4 - 2 - 1)*	*Sostenibilidad (Alta-Media-Baja) (4 - 2 - 1)*	*Total*
	Desarrollo del sistema de inventarios	Incremento en ventas		Reducción n los tiempos de entrega	Fortalecimiento del control de ventas	*44 Alta*
	Financiamiento cubierto	Incremento en seguridad del inventario	Reducción de huella de carbono	Minimizar costos operativos	Optimización de la vida útil en equipos	
		Optimización del recurso humano				
				Cumplimiento de normas y reglamentos		
	8 puntos	12 puntos	4 puntos	12 puntos	8 puntos	

El alcance de los objetivos ya sea en la factibilidad, el impacto que genera, impacto ambiental, relevancia y sostenibilidad, nos dará como resultado una serie de beneficios a la empresa como son incrementos en ventas, seguridad en el inventario, optimización de los recursos materiales y humanos, reducción de costos y tiempos, cumplimientos de normas, etc. Lo que nos dará en un gran resultado de una empresa eficiente y eficaz en su funcionamiento

4.03 Diagrama de Estrategias

La aplicación del sistema de inventario móvil nos dará como resultado la optimización de tiempo y recursos en lo que tiene que ver a ventas, para el desarrollo del mismo se divide en tres faces

1.- Desarrollo de la base de datos de la aplicación móvil para el registro y control del inventario se divide en dos partes diseño de la base de datos y diseño de diagramas UML

2.- diseño y desarrollo de la interface de usuario que comprende en el diseño de los procesos y el diseño de flujo de navegación

3.- El desarrollo de software que se divide en análisis, diseño, construcción, pruebas e implementación

Todas estas etapas nos dan como resultado la implementación de un programa depurado, eficiente y eficaz para el correcto control y manejo de inventarios en dispositivos móviles

Figura: 4 Diaframa de estrategias

l (Ver Anexo A.03) ver página 74

4.04 Matriz de Marco Lógico

Tabla: 5 Matriz de marco lógico

Resumen Narrativo	Indicadores	Medios de Verificación	Supuestos
Fin del proyecto ➢ **Contribuir y mejorar el uso del inventario y las ventas de la empresa SIVADE** ➢	➢ La implementación del proyecto incrementara las ventas en un 30% en un ano ➢ ➢ Reduce en un 20% la documentación y el papeleo	➢ Resultados del ingreso de información del inventario ➢ Cronograma del desarrollo del ➢ proyecto	➢ Continua mejora del proyecto después de la fase de implementación ➢
Propósito del proyecto ➢ **Automatización y modernización del sistema de inventarios** ➢	➢ Reduce en un 20% los costos en tecnología ➢ Mejora en un 20% la seguridad ➢ ➢ Reduce en un 30% los datos faltantes	➢ Registros contables, adquisiciones y ventas óptimos ➢	➢ Se incrementa el nivel de compromiso del personal existente ➢
Componentes del proyecto ➢ ➢ **Reducción del tiempo y mejoramiento del** ➢ **proyecto** ➢ **Capacitación al personal sobre la utilización del proyecto**	➢ Mejoramiento en la capacidad de ventas, manejo de ➢ existencias y Flujo de caja ➢ Elaboración de manuales de usuario y documentación del proyecto	➢ Documentos de validación del proceso de ➢ diseño ➢ Manuales y ➢ documentación de archivo, clases, y librerías del manual de usuario	➢ Organización de los inventarios por fecha de venta ➢ Optimización del tiempo de ➢ distribución

CAPÍTULO V: PROPUESTA

5.01 Justificación

Con la elaboración de este documento examinaremos los procesos que se realizan al momento de administrar el inventario en la empresa SIVADE y así tener un mejor control del mismo.

En el desarrollo de este sistema se utilizaron varias herramientas como son Eclipse, Rational Rose (Moldeamiento UML), (Programación java), My SQL 2008 (Base de datos), Todas esta herramientas nos permitieron un desarrollo satisfactorio de las aplicación.

La aplicación está estructurada de una manera que los usuarios, que son el bodeguero y el agente vendedor se puedan relacionar entre y así compartiendo la información para un mejor manejo de los productos que ofrece la empresa. Los usuarios pueden consultar el inventario en cualquier parte que se encuentren pues la aplicación cuenta con un hostin en la web al cual se conecta la aplicación móvil que cuenta con un sistema operativo android.

La metodología RUP (Proceso Unificado Racional) que hemos utilizado para la realización de este proyecto, nos permite entender de mejor manera los procesos, formas, y actividades que realiza la aplicación en la empresa.

5.02 Analisis y Diseño

5.02.01 Diagramas de Casos de Uso

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema.

5.02.02 Diagrama de caso de uso general

Figura: 5 Caso de Uso: Diagrama General (CU01)

(Ver Anexo A01)

Caso de Realizacion: Ingreso de usuarios

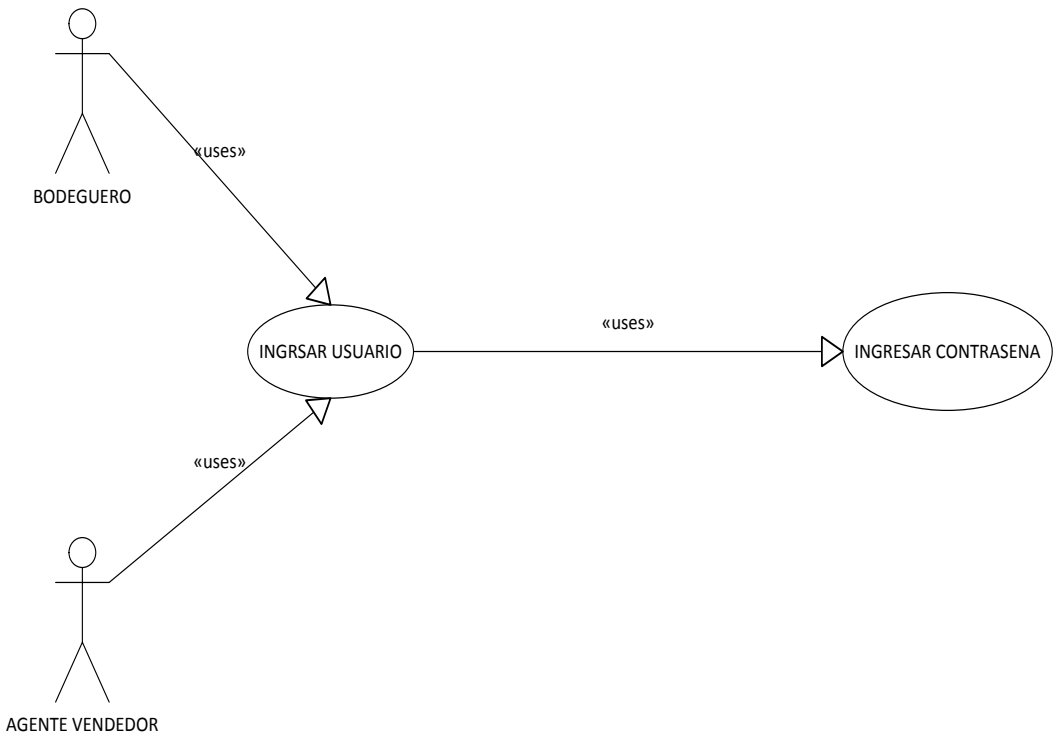


Figura: 6 Ingreso usuarios

Descripción.- Proceso para el ingreso de usuario y contraseña.

Caso de Realizacion: Ingreso de usuarios

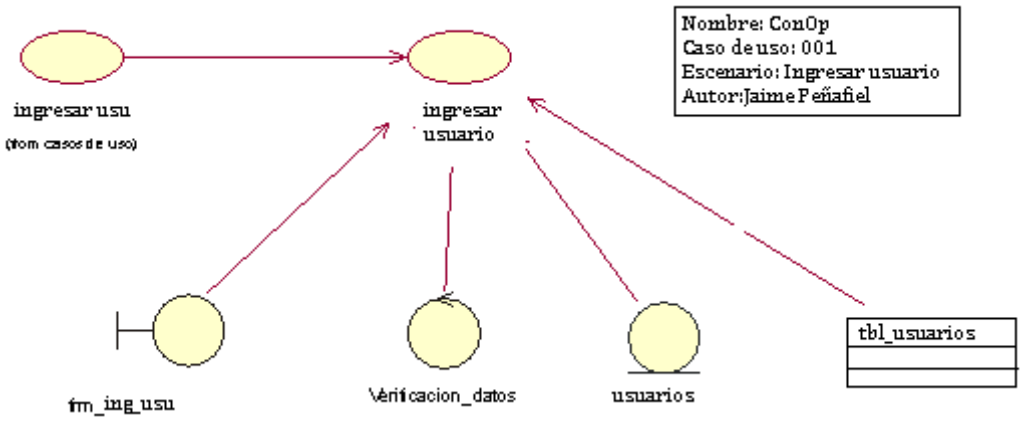


Figura: 7 Los usuario ingrsa al sistema

Tabla 6: Ingreso de usuarios

ID	**UC001**
Nombre	***Ingreso de usuarios***
Actores: Bodeguero Agente vendedor	
Precondiciones: 1. Ingreso en la base de datos 2. Generar clave	
Flujo de Eventos: 1. Ingresa usuario 2. Ingresa Contraseña	
Flujo Alternativo. 1. Volver a cargar usuario	
Pos condiciones. 1. El usuario ingresa al sistema	

Caso de uso: Escoger opción

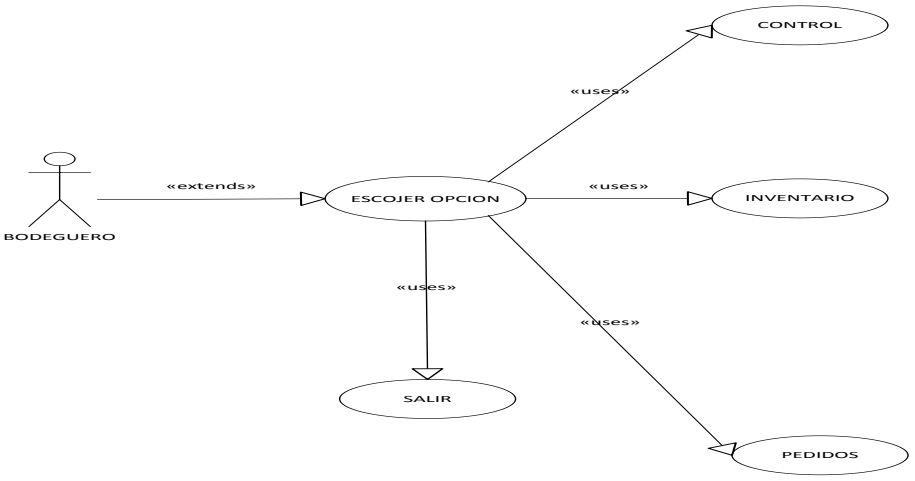


Figura: 8 Escoger opción.

Descripción.- Proceso para escoger una opción del sistema

Caso de Realizacion: Escoger opción

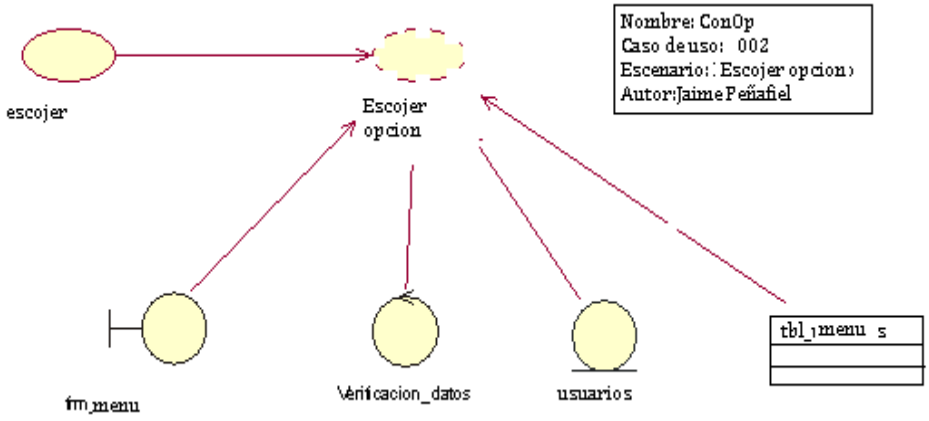


Figura: 9 Los usuarios seleccionan una opción

Tabla 7: Escoger opción

ID	**UC002**
Nombre	Escoger opción
Actores: Bodeguero.	
Precondiciones: 3. El bodeguero recibe un pedido. 4. El bodeguero necesita revisar el inventario.	
Flujo de Eventos: 3. El bodeguero escoge una opción. 4. El bodeguero ingresa a revisar el inventario. 5. El bodeguero ingresa productos.	
Flujo Alternativo. 2. Verificar datos.	
Pos condiciones. 2. El bodeguero realiza cambios en el inventario.	
3. El bodeguero ingresa pedidos	

Caso de uso: Pedidos

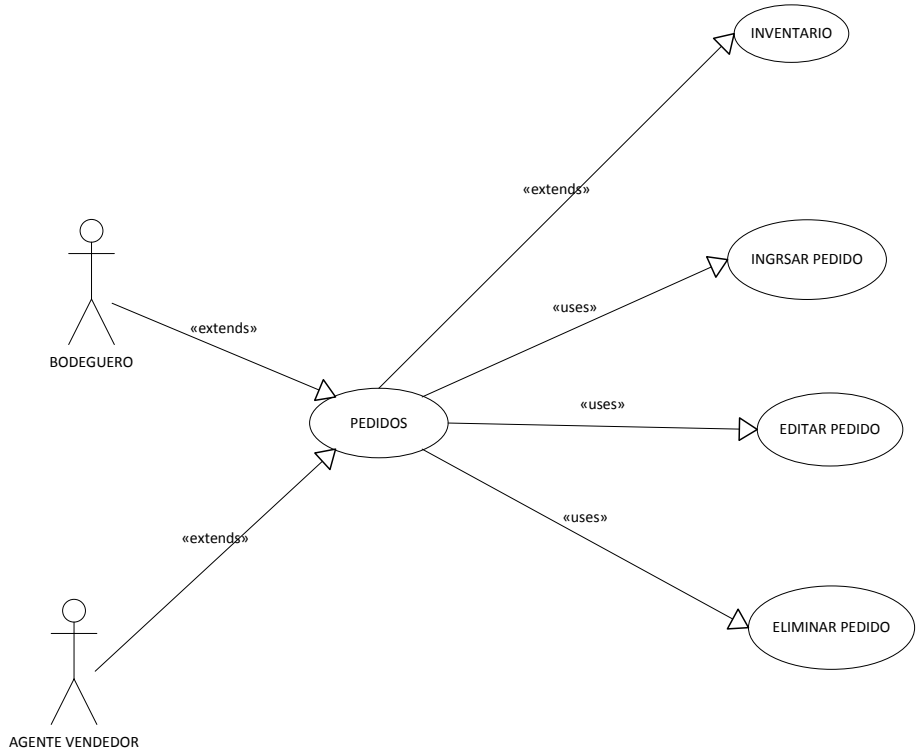


Figura: 10 Pedidos

Descripción.- Proceso para que el usuario seleccione una opción de pedidos.

Caso de Realizacion: Pedidos

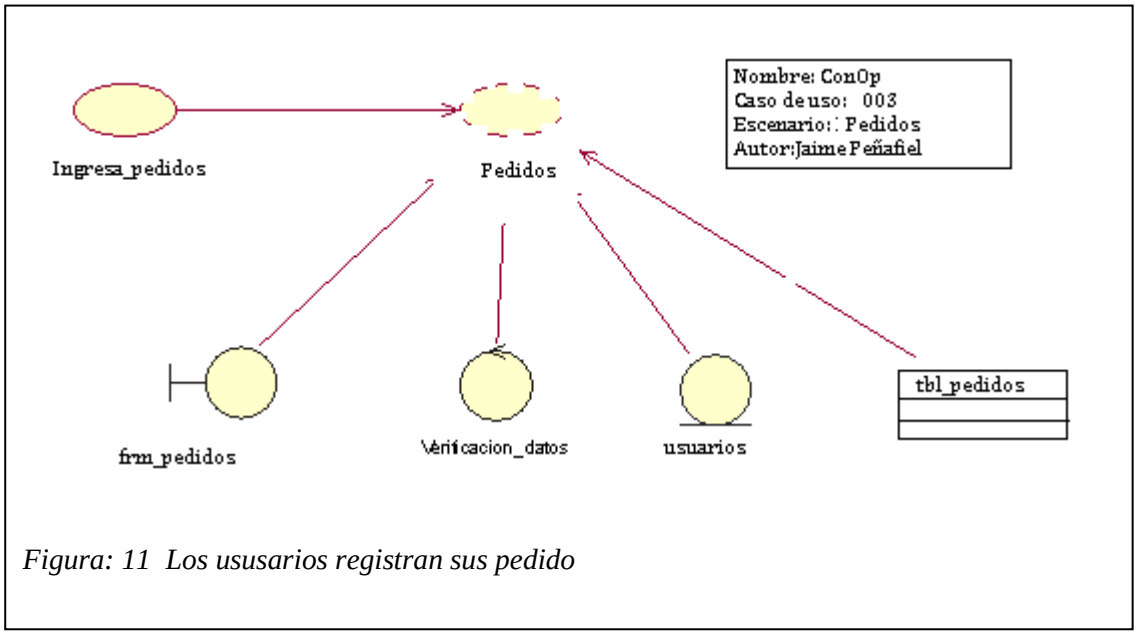


Figura: 11 Los ususarios registran sus pedido

Tabla 8: Pedidos

ID	**UC003**
Nombre	Pedidos
Actores: Bodeguero/Agente vendedor	
Precondiciones: 5. El bodeguero recibe un pedido 6. El agente vendedor recibe un pedido	
Flujo de Eventos: 6. El bodeguero ingresa un pedido. 7. El bodeguero elimina un pedido. 8. El agente vendedor ingresa un pedido. 9. El agente vendedor elimina un pedido.	
Flujo Alternativo. 3. Los pedidos quedan pendientes.	
Pos condiciones. 4. El bodeguero guarda los pedidos.	
5. El agente vendedor guarda los pedido.	

Caso de uso: Ingreso sistema ventas

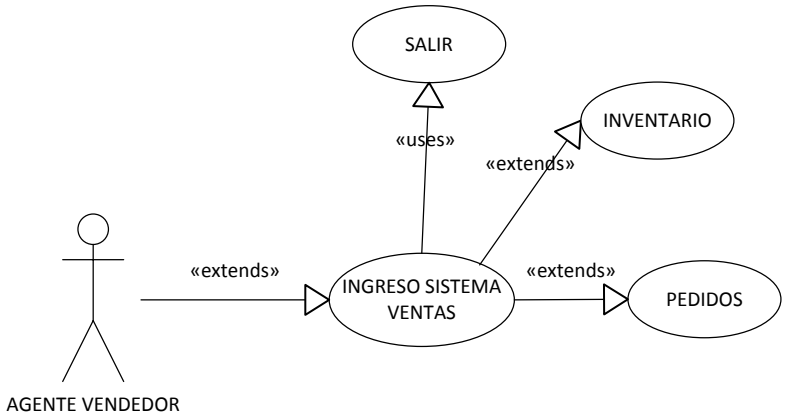


Figura: 12 Ingreso sitema ventas

Descripción.- Proceso para escoger una opción del sistema

Caso de Realizacion: Ingreso sistema ventas

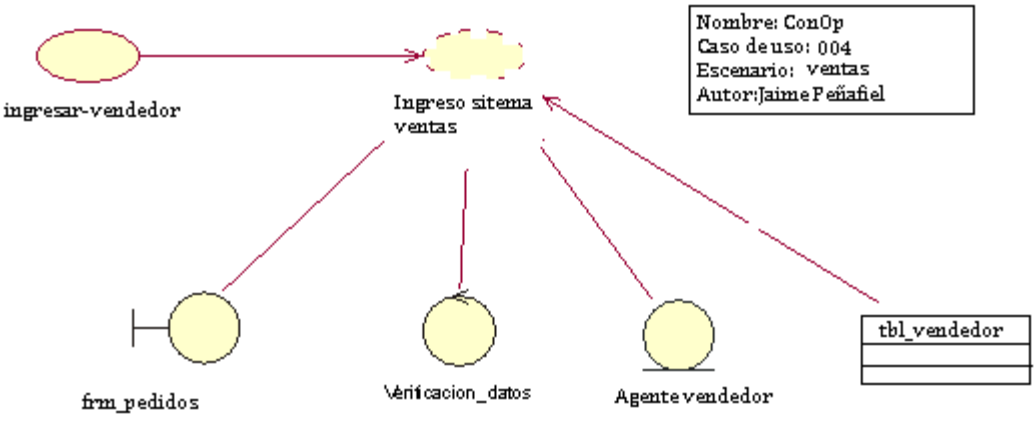


Figura: 13 El agente vendedor ingresa al sistema

Tabla 9: Ingreso sistema ventas

ID	**UC004**
Nombre	Ingreso al sistema ventas
Actores: Agente vendedor	
Precondiciones:	
7. El agente vendedor visita a sus clientes.	
Flujo de Eventos:	
10. Revisión del inventario.	
11. Registro de productos.	
Flujo Alternativo.	
4. Quedan los pedidos pendientes.	
Pos condiciones.	
6. El agente vendedor confirma los pedidos.	

Caso de uso: Control

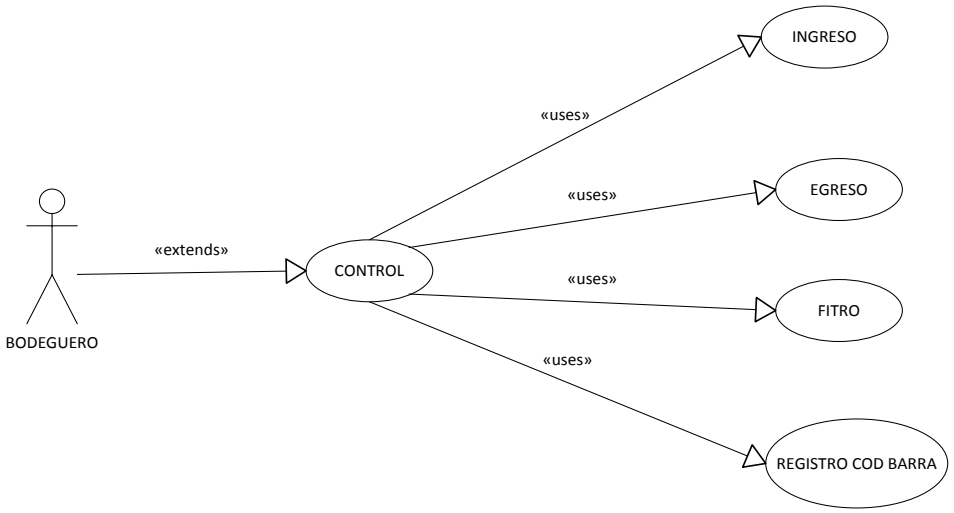


Figura: 14 Control

Descripción.- Proceso para que el usuario seleccione una opción de pedidos.

Caso de Realizacion: Control

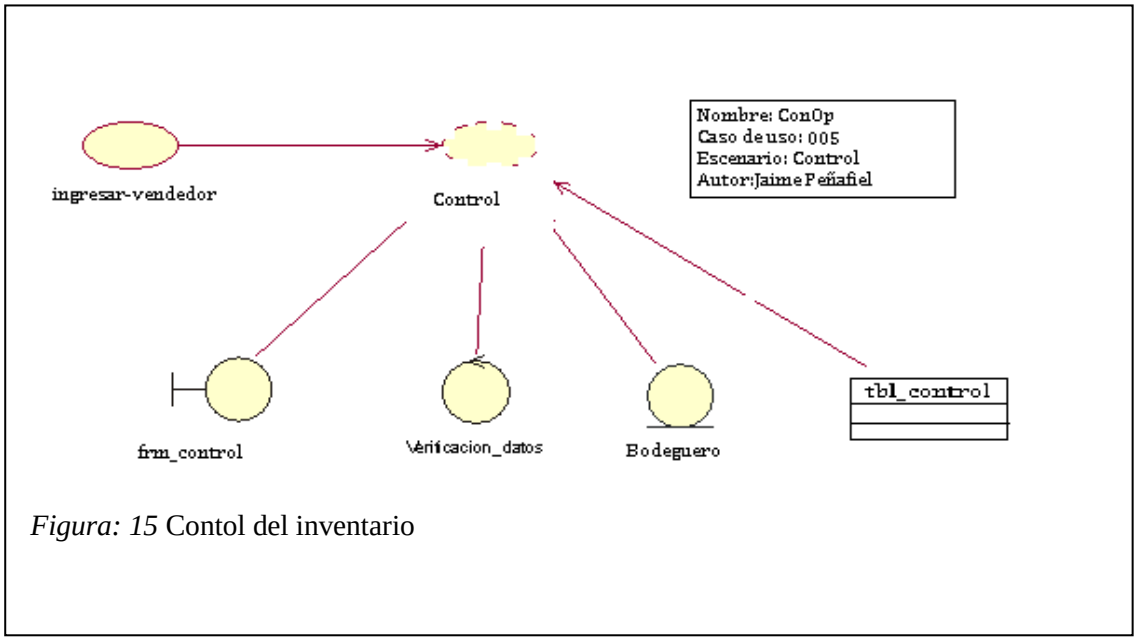


Figura: 15 Contol del inventario

Tabla 10: Control

ID	**UC005**
Nombre	Control
Actores: Bodeguero	
Precondiciones: 8. Abastecimiento de mercadería. 9. Registro de producto.	
Flujo de Eventos: 12. Realiza el control del inventario. 13. Ingresa productos. 14. Actualiza el inventario.	
Flujo Alternativo. 5. Falta de stock.	
Pos condiciones. 7. Actualización del inventario.	
8. Falla en el cuadre.	

5.02.03 Diagrama de Secuencia

Descripción de diagramas de secuencia:

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML. En inglés se pueden encontrar como "sequence diagram", "event-trace diagrams", "event scenarios" o "timing diagrams"1

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista business del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos.

Diagrama de secuencia: Ingreso de usuarios

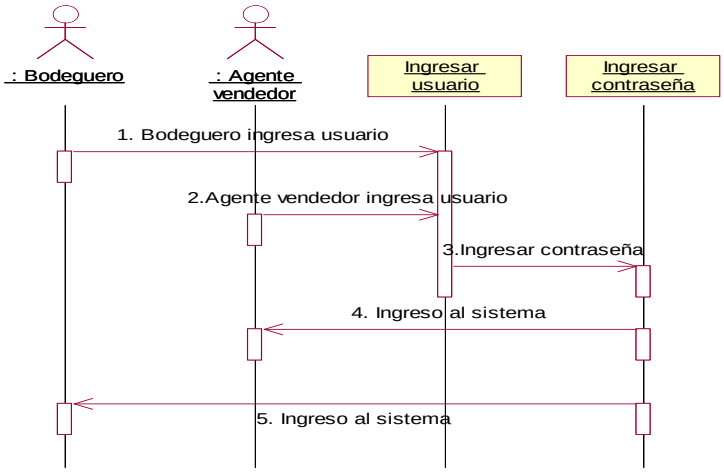


Figura: 16 Realiza el ingreso de usuarios.

Diagrama de secuencia: Escoger opción

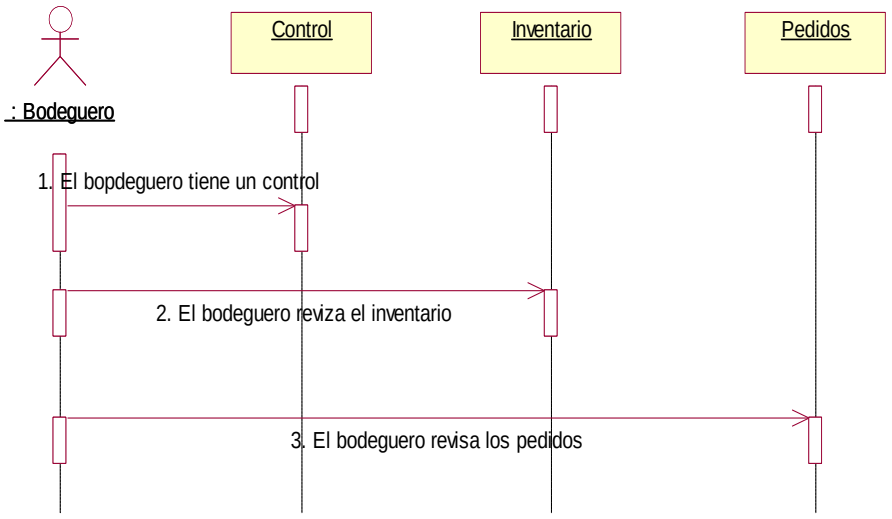


Figura: 17 El usuario selecciona la opción que desee

Caso de uso: Pedidos

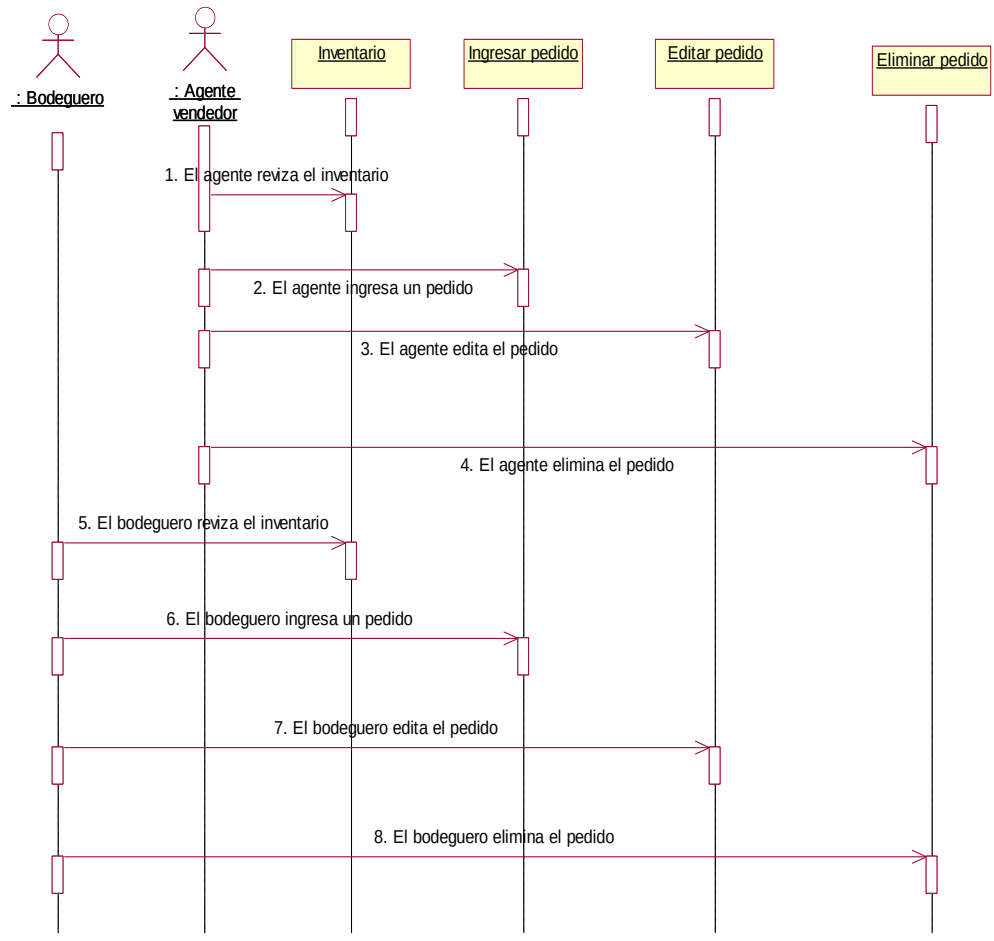


Figura: 18 Los usuarios registran los pedidos de sus clientes

Diagrama de secuencia: Ingreso al sistema ventas

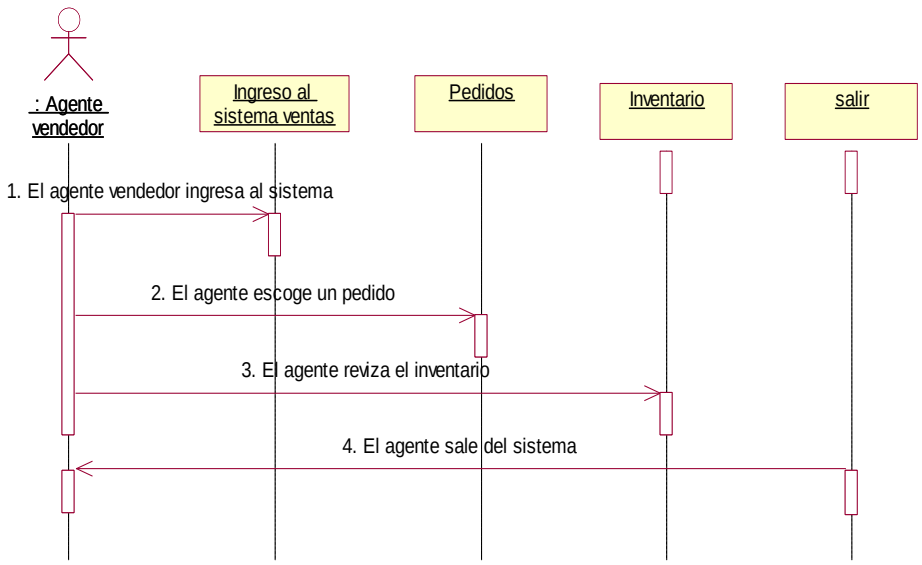


Figura: 19: El Agente vendedor ingresa al sistema de ventas

Diagrama de secuencia: Control

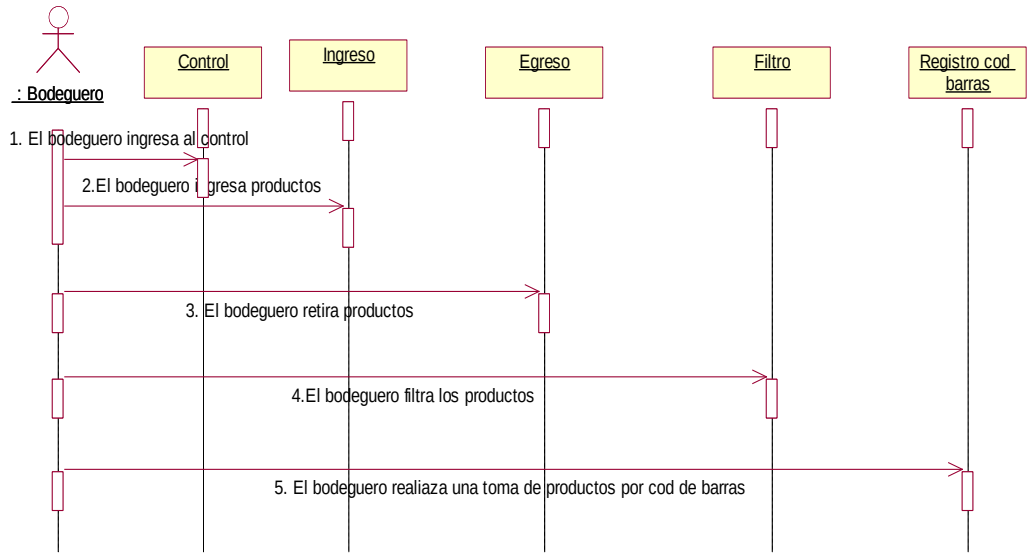


Figura: 20 El bodeguero realiza un control del inventario

5.02.03 Diagrama de Colaboración

Un diagrama de colaboración en las versiones de UML 1.x es esencialmente un diagrama que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de colaboración, también llamados diagramas de comunicación, muestran explícitamente las relaciones de los roles. Por otra parte, un diagrama de comunicación no muestra el tiempo como una dimensión aparte, por lo que resulta necesario etiquetar con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes.

Diagrama de colaboración: Ingreso de usuarios

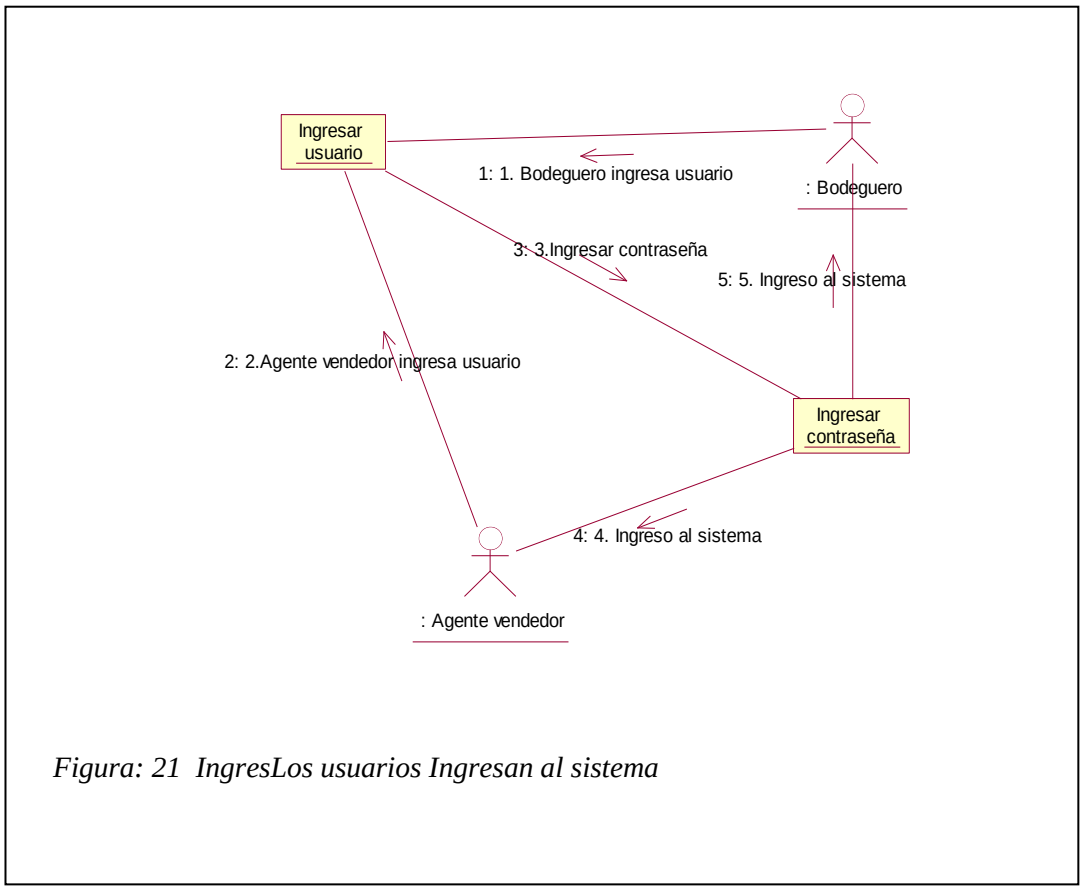


Figura: 21 IngresLos usuarios Ingresan al sistema

Diagrama de colaboración: Escoger opción

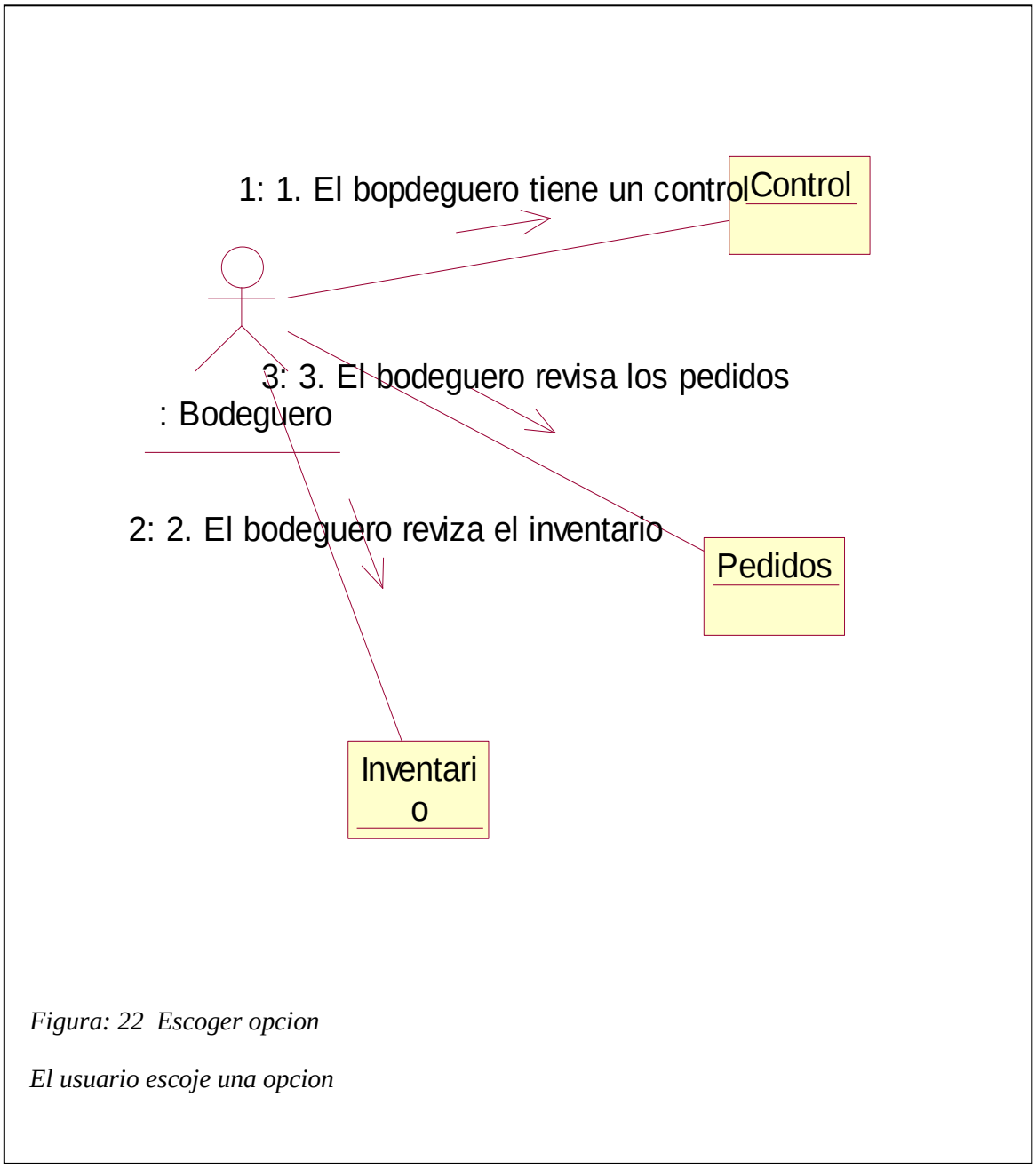


Figura: 22 Escoger opcion

El usuario escoje una opcion

Diagrama de colaboración: Pedidos

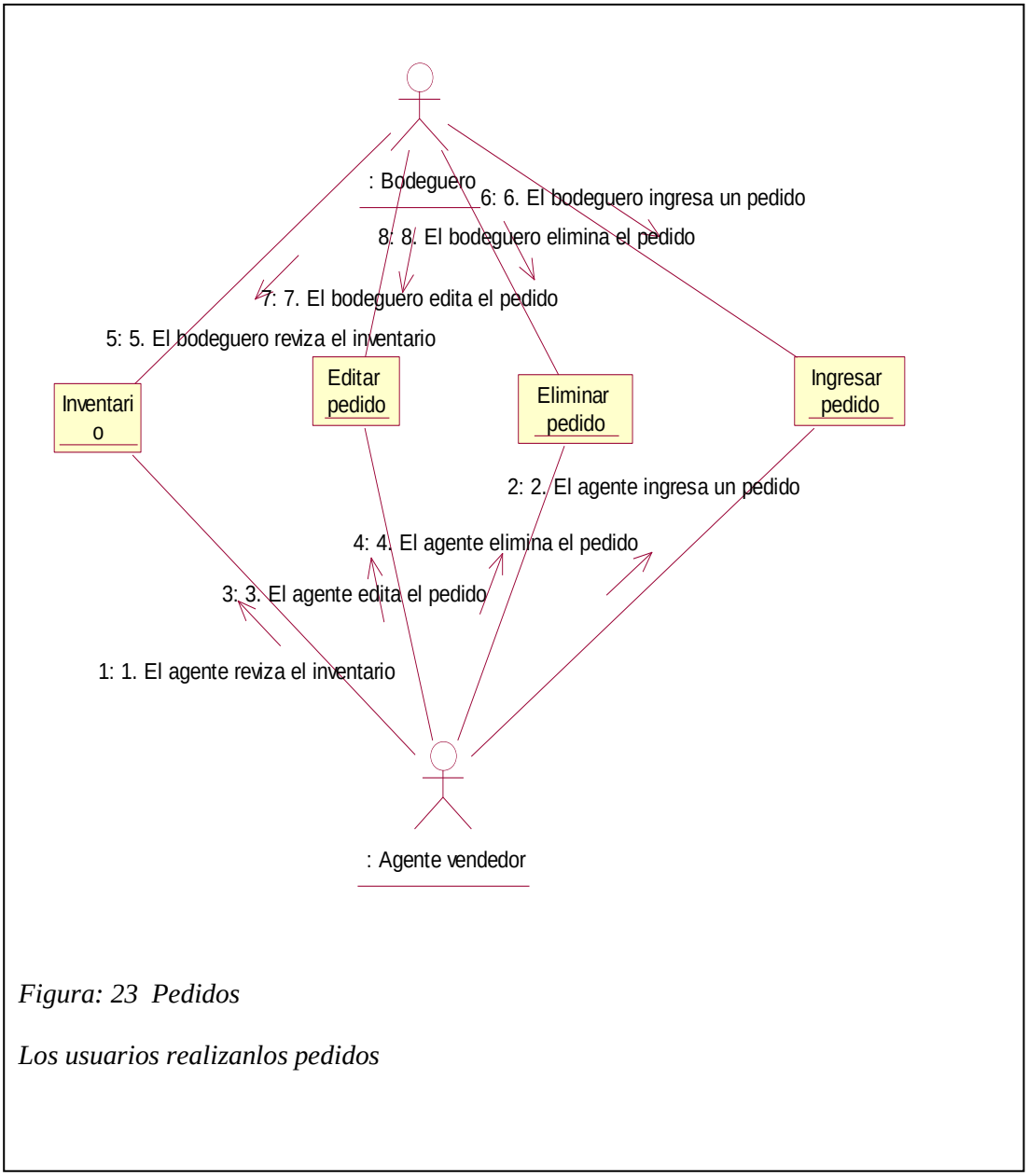


Figura: 23 Pedidos

Los usuarios realizanlos pedidos

Diagrama de colaboración: Ingreso sistema ventas

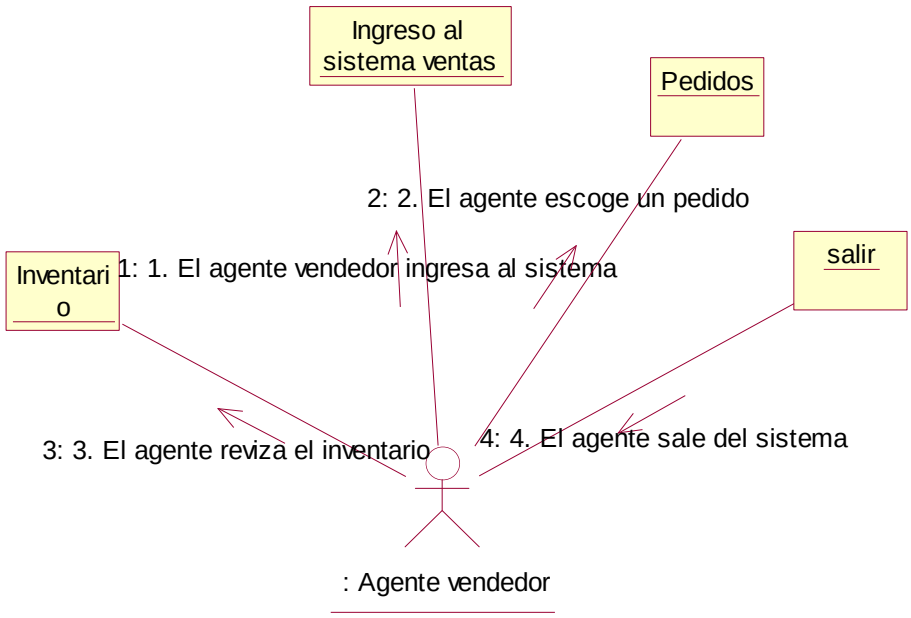


Figura: 24 Ingreso siatema ventas

El agente vendedor ingresa al sistema

Diagrama de colaboración: Control

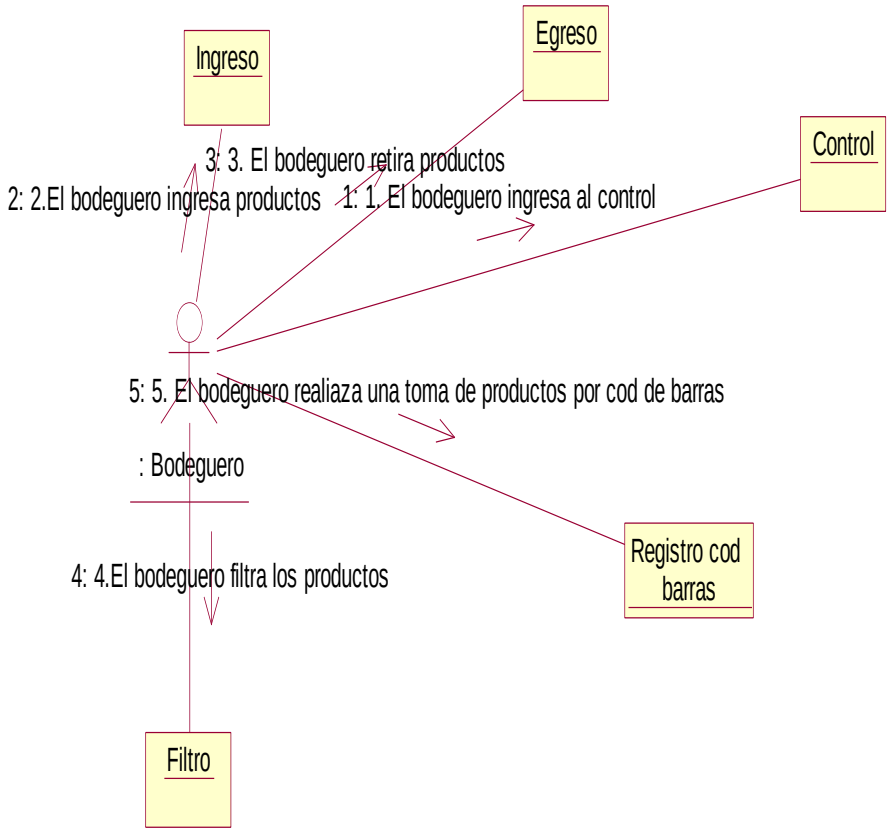


Figura: 25 Control

El bodeguero controla el inventario

5.02.04 Diagrama de componentes

Un diagrama de componentes es un diagrama tipo del Lenguaje Unificado de Modelado.

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Diagrama de componente: Inventario

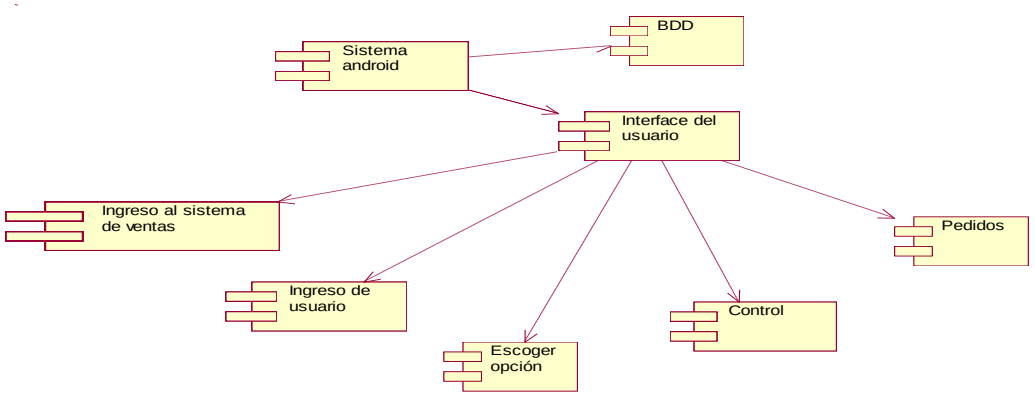


Figura:26 Inventario
Muestra los modulos del Software

5.02.05 Diagrama de Clases

Una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica.

Las clases son gráficamente representadas por cajas con compartimentos para:

Nombre de la clase, atributos y operaciones / métodos

Responsabilidades, Reglas, Historia de Modificaciones, etc.

Los diseñadores desarrollan clases como conjuntos de compartimentos que crecen en el tiempo

agregando incrementalmente aspectos y funcionalidades.

Diagrama de clases

(Ver Anexo A.05)

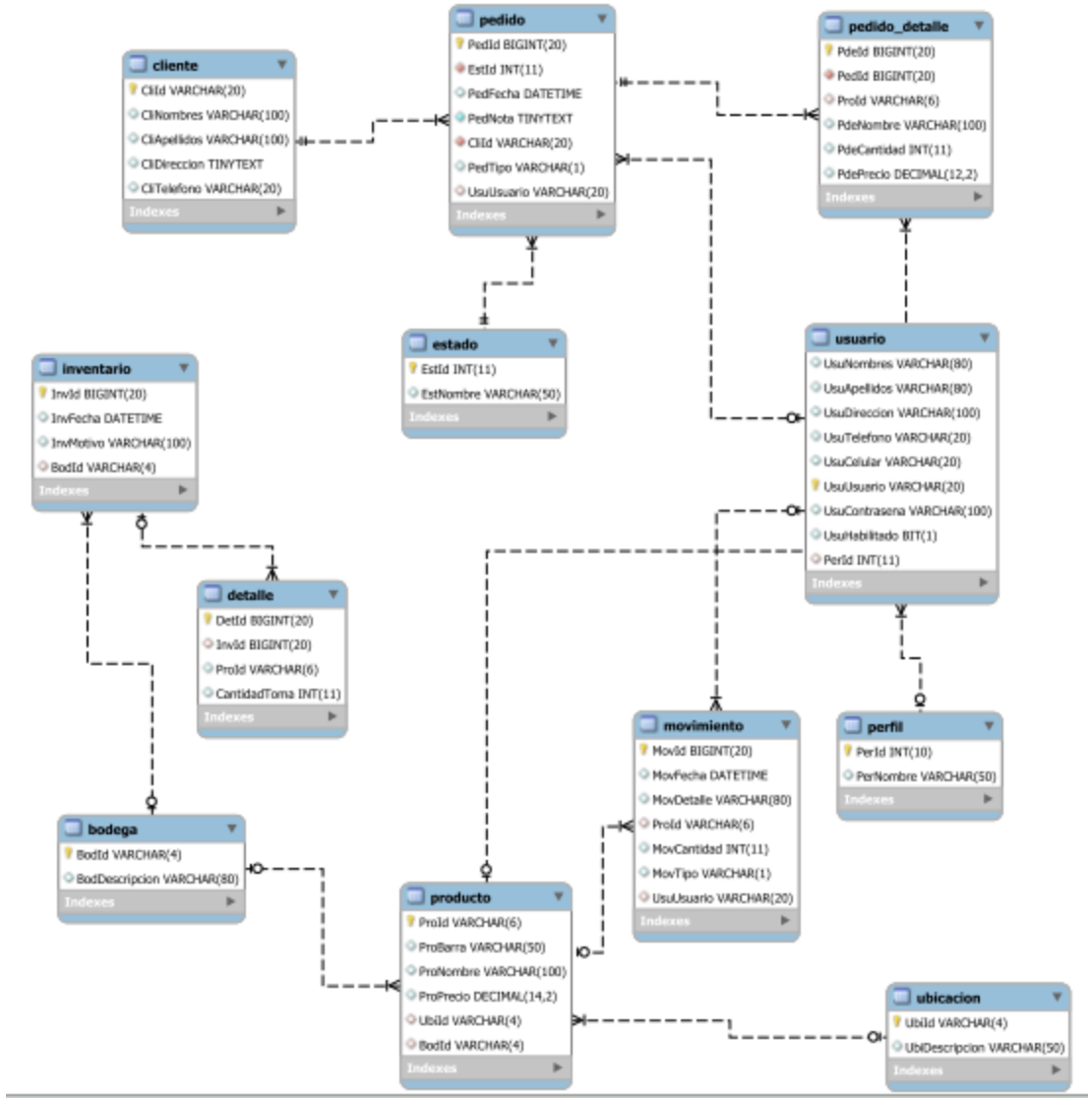
Modelo lógico.

(Ver Anexo A.06)

Modelo Físico.

(Ver Anexo A.07)

5.02.06 Diagrama de base de datos



Figura; 27 Diagrama de Base de Datos

El esquema de una base de datos (en inglés, database schema) describe la estructura de una base de datos, en un lenguaje formal soportado por un sistema de gestión de base de datos (DBMS). En una base de datos relacional, el esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.

5.03 Desarrollo

5.03.01 Arquitectura del Sistema

5.03.01.01 Capa de Datos

La capa de enlace de datos es la que se encarga de tomar una transmisión de datos " cruda " y transformarla en una abstracción libre de errores de transmisión para la capa de red. Logra esta función dividiendo los datos de entrada en marcos de datos (de unos cuantos cientos de bytes), transmite los marcos en forma secuencial, y procesa los marcos de estado que envía el nodo destino.

Dado que la capa física solamente acepta y transmite un flujo de bits sin ninguna consideración de significado o estructura, está asignado a la capa de enlace de datos crear y reconocer los limites de un marco de datos. Esto se logra añadiendo patrones de bits especiales al comienzo y final del marco de datos .Si estos patrones de bits pueden aparecer en los datos, se debe tomar un especial énfasis para evitar alguna confusión.

5.03.01.02 Capa de Negocios

La capa de negocio contiene la lógica principal de procesamiento de datos dentro de nuestra aplicación Web. Se comunica con la capa de presentación para obtener las entradas del usuario y presentar la información resultante, así como la capa de acceso a datos o directamente con servicios para realizar sus operaciones.

5.03.01.03 Capa de presentación

Es generalmente un protocolo de paso de la información desde las capas adyacentes y permite la comunicación entre las aplicaciones en distintos sistemas informáticos de manera tal que resulte transparente para las aplicaciones, se ocupa del formato y la representación de los datos y, si es necesario, esta capa puede traducir entre distintos formatos de datos. Además, también se ocupa de las estructuras de los datos que se utilizan en cada aplicación, aprenderá cómo esta capa ordena y organiza los datos antes de su transferencia.

5.03.01.04 Módulo de Seguridad

Este módulo de seguridad permite identificar el usuario que va a operar el sistema, así como asignar los debidos roles a cada usuario, permitiendo así generar niveles de seguridad de acuerdo al usuario. Para que el usuario pueda ingresar al sistema deberá el administrador del sistema concederle los privilegios necesarios para que así pueda manipular el sistema. Y los roles dentro del sistema son: Bodeguero, Agente Vendedor.

5.03.01.05 Módulo Mantenimiento

Este Módulo es el encargado de agregar, modificar, eliminar y buscar los datos necesarios en las respectivas tablas.

- Usuarios
- Pedidos
- Inventario
- Clientes
- Ingresos

- Egresos
- Estos mantenimientos se encuentran categorizados de acuerdo al nivel de seguridad que se le asigne al usuario.

5.03.01.06 Módulo Lógica Negocios

Este modulo crear las admisiones de los procesos que realiza el sistema con el objetivo que exista integridad en la información y se muestre un aviente amigable a los usuarios del sistema.

5.03.02 Estándares de Programacion

5.03.02.01 Declaraciones

Una declaración por línea

Se recomienda el uso de una declaración por línea, promoviendo así el uso de comentarios. Ejemplo,

```
intidUnidad;   // Identificador de la unidad organizativa

String[] funciones; // Funciones de la unidad
```

5.03.02.02 Inicialización

Toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente.

```
intidUnidad = 1;

String[] funciones = { "Administración", "Intervención", "Gestión" };
```

5.03.02.03 Declaración de clases / interfaces

Durante el desarrollo de clases / interfaces se deben seguir las siguientes reglas de formateo:

No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.

El carácter inicio de bloque ("{") debe aparecer al final de la línea que contiene la sentencia de declaración.

El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de "{".

Los métodos se separarán entre sí mediante una línea en blanco.

```
publicclasseClaseEjemplo extends Object {
int variable1;
int variable2;
publicClaseEjemplo() {
variable1 = 0;
variable2 = 1;
}
...
}
```

5.03.02.04 Clases e interfaces

Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con maýusculas.
Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, etc.).

Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general toda interfaz se nombrará con el prefijo "I" para diferenciarla de la clase que la implementa (que tendrá el mismo nombre sin el prefijo "I").

classCiudadano

classOrganigramaDAO

classAgendaService

classIAgendaService

5.03.02.05 Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas.

publicvoidinsertaUnidad(Unidad unidad);

publicvoideliminaAgenda(Agenda agenda);

publicvoidactualizaTramite(Tramite tramite)

5.03.02.06 Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales.

Unidad unidad;

Agenda agenda;

Tramite tramite;

5.03.02.07 Constantes

Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos las palabras se separarán entre sí mediante el carácter de subrayado "_".

int LONGITUD_MAXIMA;

int LONGITUD_MINIMA;

Prácticas de programación

5.03.02.02.08 Visibilidad de atributos de instancia y de clase

Los atributos de instancia y de clase serán siempre privados, excepto cuando tengan que ser visibles en subclases herederas, en tales casos serán declarados como protegidos.

El acceso a los atributos de una clase se realizará por medio de los métodos "get" y "set" correspondientes, incluso cuando el acceso a dichos atributos se realice en los métodos miembros de la clase.

```
public class Unidad {

privateint id;

privateString nombre;

publicvoidactualizaUnidad(Unidad unidad) {

this.setId(unidad.getId());

this.setNombre(unidad.getNombre());
 }
}
```

5.03.02.09 Referencias a miembros de una clase

Evitar el uso de objetos para acceder a los miembros de una clase (atributos y métodos estáticos). Utilizaremos en su lugar el nombre de la clase. Por ejemplo:

```
metodoUtilidad();     // Acceso desde la propia clase estática

ClaseUtilidad.metodoUtilidad(); // Acceso común desde cualquier clase
```

5.03.02.10 Asignación sobre variables

Se deben evitar las asignaciones de un mismo valor sobre múltiples variables en una misma sentencia, ya que dichas sentencias suelen ser difíciles de leer.

```
int a = b = c = 2;  // Evitar
```

No utilizar el operador de asignación en aquellos lugares donde sea susceptible de confusión con el operador de igualdad. Por ejemplo:

```
// INCORRECTO

if ((c = d++) == 0) { }

// CORRECTO

c = d++;

if (c == 0) { }
```

No utilizar asignaciones embebidas o anidadas. Ejemplo:

```
c = (c = 3) + 4 + d;  // Evitar
```

debería escribirse

```
c = 3;

c = c + 4 + d;
```

5.03.03 Estándar Diseño UML

5.03.02.01 Actores

Los actores "especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto."

Un actor modela un tipo de rol jugado por una entidad que interactúa con el sujeto (esto es, intercambiando signos y datos), pero que es externo a dicho sujeto.

Los actores pueden representar roles jugados por usuarios humanos, hardware externo, u otros sujetos. Un actor no necesariamente representa una entidad física específica, sino simplemente una faceta particular (es decir, un "rol") de alguna actividad que es relevante a la especificación de sus casos de uso asociados. Así, una única instancia física puede jugar el rol de muchos actores diferentes y, asimismo, un actor dado puede ser interpretado por múltiples instancias diferentes.

5.03.02.02 Clase

Introducción

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenimiento.

Un diagrama de clases esta compuesto por los siguientes elementos:

Clase: atributos, métodos y visibilidad.

Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Elementos

Clase

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

Atributos:

Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

public (+, ): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accsesible desde todos lados.

private (-, ): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).

protected (#, ): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

Métodos:

Los métodos u operaciones de una clase son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

public (+, ): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accsesible desde todos lados.

private (-, ): Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).

protected (#, ): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

Relaciones entre Clases:

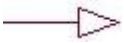
Ahora ya definido el concepto de Clase, es necesario explicar como se pueden interrelacionar dos o más clases (cada uno con características y objetivos diferentes).

Antes es necesario explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

uno o muchos: 1..* (1..n)

0 o muchos: 0..* (0..n)

número fijo: m (m denota el número).

Herencia (Especialización/Generalización): 

Indica que una subclase hereda los métodos y atributos especificados por una Super Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected).

5.03.02.03 Actividad

El diagrama de actividades usamos para mostrar la secuencia de actividades. Los diagramas de actividades muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad. Estos también pueden usarse para detallar situaciones donde el proceso paralelo puede ocurrir en la ejecución de algunas

actividades. Los Diagramas de Actividades son útiles para el Modelado de Negocios donde se usan para detallar el proceso involucrado en las actividades de negocio.

5.03.02.04 Interfaces

Interfaces usamos para modelar los conjuntos de operaciones que otros elementos del modelo, como las clases o componentes deben implementar. Un elemento del modelo se da cuenta de la implementación de una interfaz reemplazando cada una de las operaciones que la interfaz declara.

5.03.02.05 Diagramas

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

UML incluye nueve tipos de diagramas: clases, objetos, casos de uso, secuencia, colaboración, estados, actividades, componentes, de

5.03.04 Estándar de La Base de Datos

Estandares utilizados para la base de datos Responsables: Serán responsables los desarrolladores que realizan el desarrollo y mantenimiento del aplicativo SIVADE para el control de los inventarios.

1. Nombres de bases de datos El nombre de la base de datos debe representar el propósito de la misma y no a los usuarios, proyectos o departamentos, etc. Ejemplo: inventario

2. Nombres de tablas

* Los nombres serán sustantivos en singular

* Si la tabla contiene varias palabras, estas se unirán mediante guión bajo "_"

3. Nombres de campos

* Los atributos y columnas no deben ser precedidos por un alias de la tabla(Tres primeros caracteres del nombre de la tabla).

* Los nombres deben ser simples, representativos e intuitivos

* Utilizar notación Pascal para nombrar los campos Ejemplos: CliNombres, CliApellidos, etc.

4. Nombres de claves foráneas Las claves foráneas deben ser nombradas por el prefijo '_FK' más los dos nombres de las tablas. Ejemplo:

* FK_pedido_detalle_producto

* FK_pedido_detalle_pedido

5.04.03 Base de datos

Figura 42

Base de datos del sistema de automatización de registro y control.

(Ver Anexo A.26)

5.04.04Diseño de interfaces

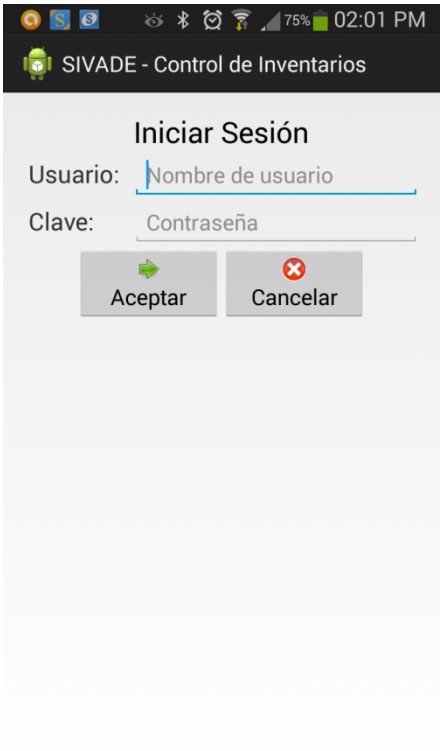


Figura: 28 Iniciar sesión

1.-Pantalla de acceso para identificar al usuario y su contraseña para el uso permitido al inventario y su manipulación .

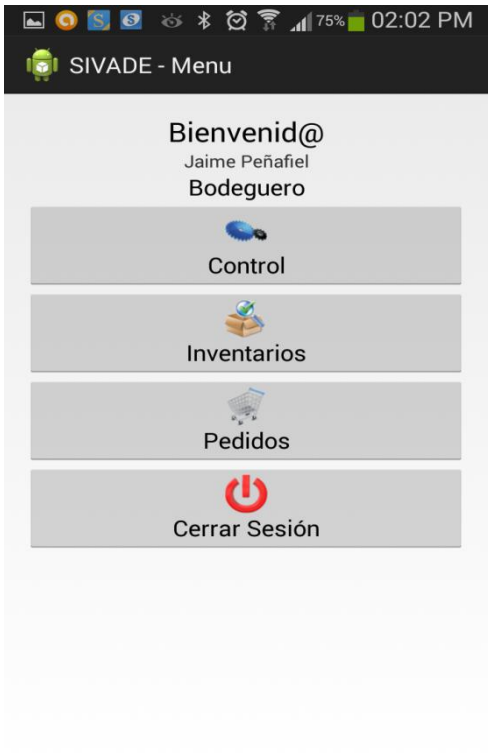


Figura: 29 Menu

2.- Panel de acceso al programa con datos como bienvenida, nombre, cargo del usuario y a las diferentes opciones que presenta el mismo como control, Inventarios, Pedidos y el cierre del programa

Figura: 30 Menu control

3.- Pagina principal de la función de control el mismo que tiene diferentes opciones como (de izquierda a derecha) Código de barras, búsqueda, actualización de productor, ingreso de productos y egresar productos

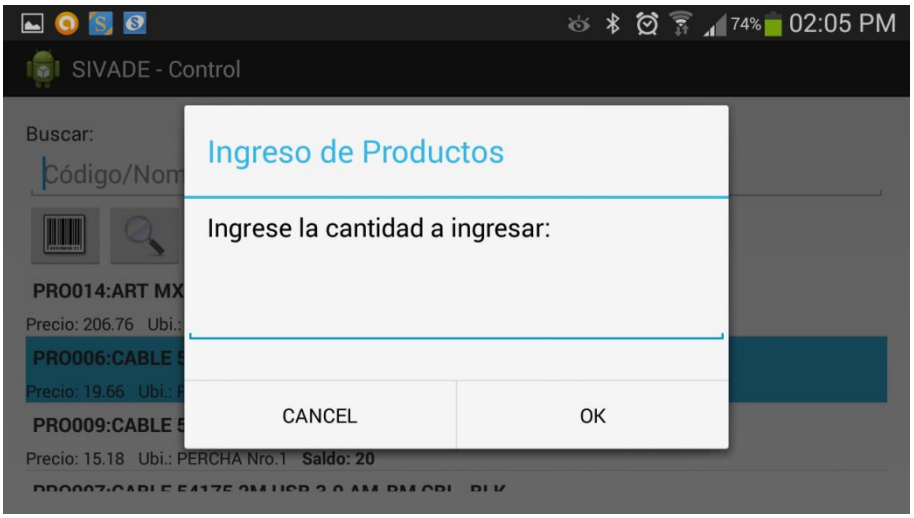


Figura: 31 Igreso de productos

4.- La opción de ingreso de productos nos aparece una pantalla en la que podemos delimitar la cantidad que se va a ingresar al inventario de diferentes productos

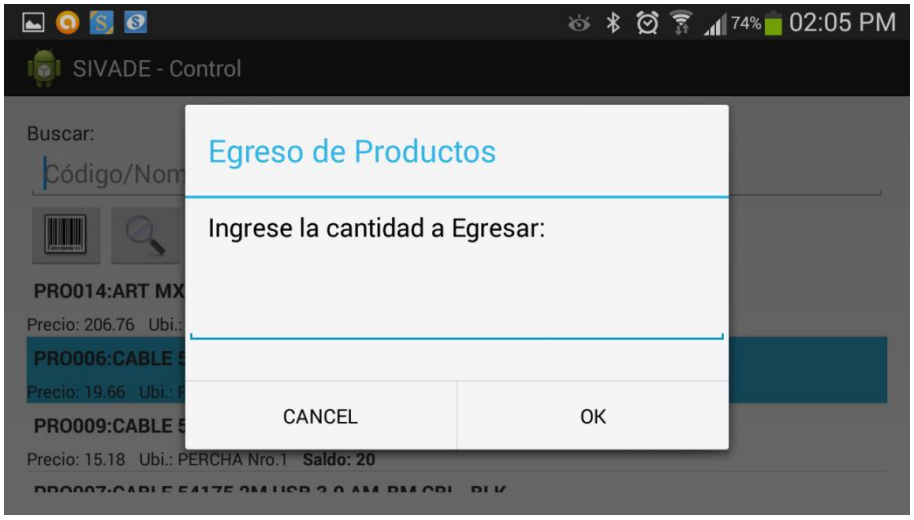


Figura: 32 Egreso de productos

5.-La opción de egreso de productos nos aparece una pantalla en la que podemos delimitar la cantidad que se va a egresar al inventario de diferentes productos

Figura: 33 Inventario

6.- En la opción de inventarios nos da como opción delimitar las existencias que posee la empresa para la venta y distribución de los diferentes productos para ser comercializados

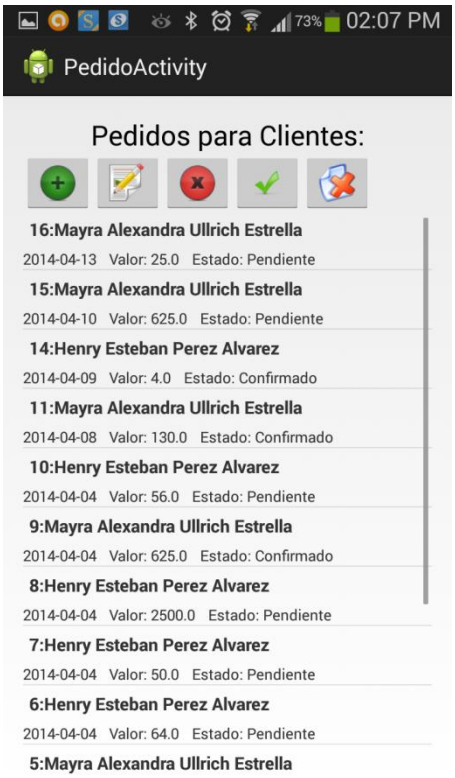


Figura: 34 Pedidos clienter

7.- La página principal de la opción de pedidos nos da como alternativas varias como son añadir un nuevo pedido, añadir un producto, eliminar un pedido, confirmar el pedido y cancelar un pedido

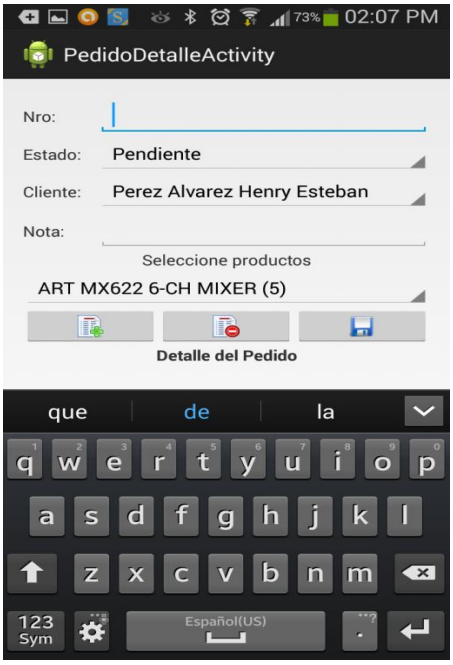


Figura: 35 Ingreso de detalle de pedidos

8.- Añadir un nuevo pedido al hacer esta accion se debe delimitar el numero de pedido, el estado, el nombre del cliente y nos da una opcion de incluir una nota

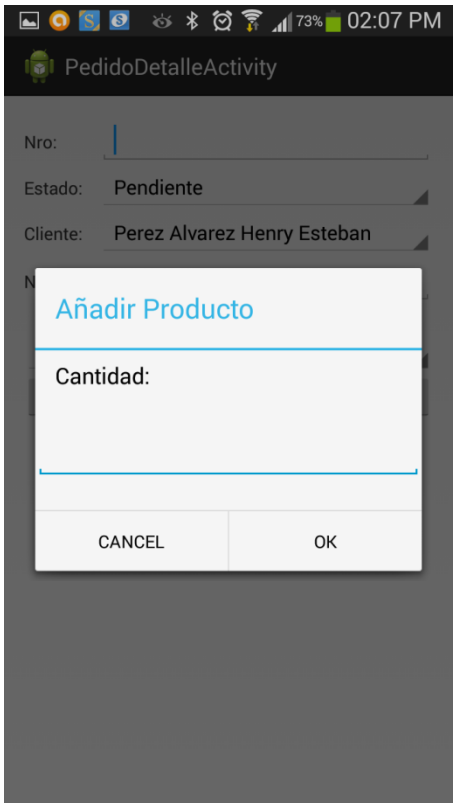


Figura: 36 Añadir producto

9.- Al añadir la cantidad del pedido estamos delimitando el número de unidades el cliente quiere que le vendamos y queda para el correcto manejo e manipulación del inventario en bodega

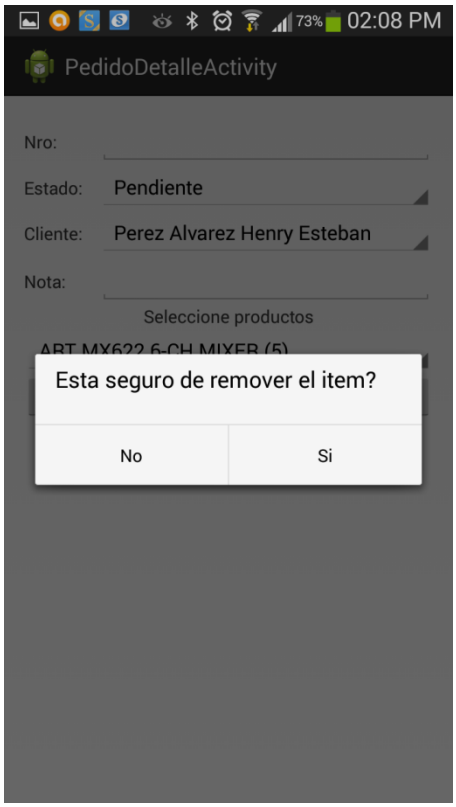


Figura: 37Remover item

10.- Esta opción nos da la opción de remover uno de los productos que el cliente nos pide y editar el pedido final

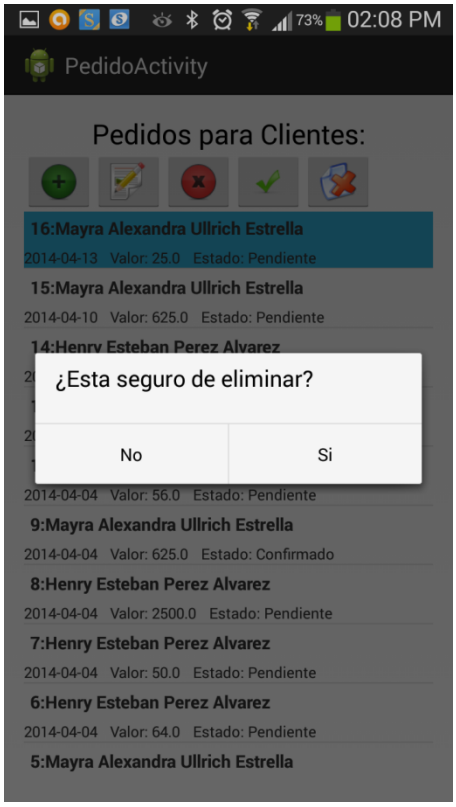


Figura: 38 Eliminar

11.-La opción de eliminar nos da la facilidad de cancelar los diferentes pedido que hemos realizado y tomar las medidas necesarias para no causar una excesiva cantidad de productos en el inventario

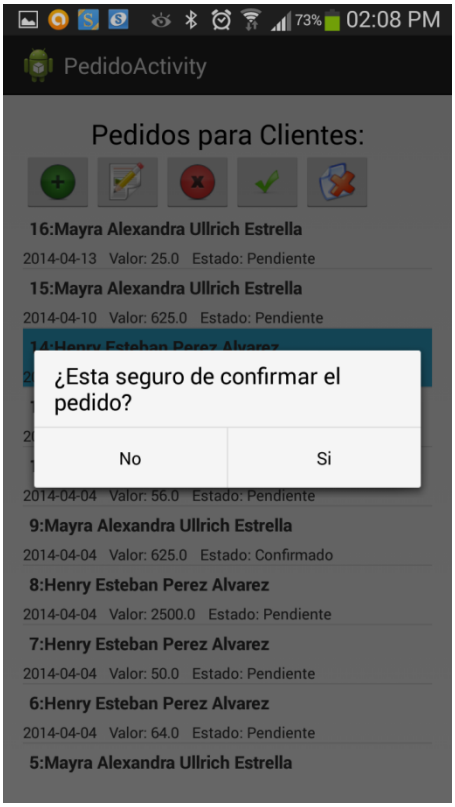


Figura: 39 Confirmar Pedido

12.-En la opción de confirmación del pedido nos da la facilidad de notificar al encargado de bodega la eminente salida de mercadería en tiempo y plazo establecido por el cliente

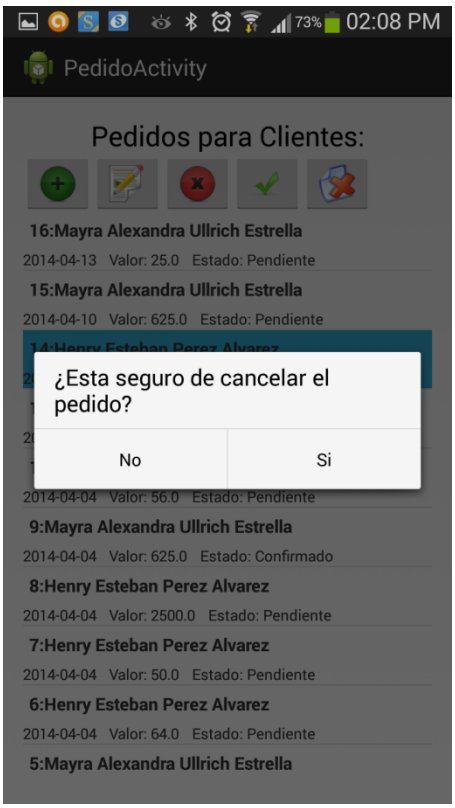


Figura: 40 Cancelar pedido

13.- En la opción de cancelación del pedido nos da la facilidad de notificar al encargado de bodega la venta fallida de productos hecho por parte del cliente y la actualización de dicha mercadería en el inventario general

5.05 Pruebas

5.05.01 Objetivo de las pruebas

Analizar el funcionamiento y el desarrollo del proyecto realizado, poniendo a prueba que todos los modulos desarrollados, los estándares de calidad y seguridad.

5.05.02 Pruebas de módulo

5.05.02.01 Módulo seguridad

La seguridad informática debe establecer normas que minimicen los riesgos a la información o infraestructura informática. Estas normas incluyen horarios de funcionamiento, restricciones a ciertos lugares, autorizaciones, denegaciones, perfiles de usuario, planes de emergencia, protocolos y todo lo necesario que permita un buen nivel de seguridad informática minimizando el impacto en el desempeño de los trabajadores y de la organización en general y como principal contribuyente al uso de programas realizados por programadores.

Las seguridades brindadas en el sistema permiten que no exista ningún tipo de fraude en la manipulación de información de la liga barrial.

5.05.02.02 Módulo mantenimiento

El módulo de mantenimiento se lo analizo de una manera muy cuidadosa y exaustiva para prevenir que exista cualquier tipo de inconveniente en el momento de crear, modificarse o eliminarse datos de la base, por lo que se creo campos únicos (Primary

key) secuenciales para evitar que la información se repita y a su vez campos foráneos (Foreing key) para mantener la integridad de los datos.

5.05.03 Pruebainterface de usuario

Las pruebas realizadas en la interface de usuario se la a realizado con un ambiente muy amigable para el usuario que sea de fácil manejo esto nos permitio saber que tan fácil es la manera de interactuar el usuario con el sistema y a su vez darnos cuenta del tiempo de respuesta que tiene el sistema al ejecutar un proceso emitido por el usuario.

De igual manera nos asesoramos que todos los campos, botones, ventanas, alertas, tablas y demás se encuentren habilitados de una manera correcta y cuenten con los estándares establecidos previamente.

5.05.04 Prueba de carga

Para realizar esta prueba es necesario actuar sobre una tabla específica para verificar el tiempo de respuesta que tiene al momento de realizar varias actividades afectando solo a dicha tabla la transacionabilidad será establecida según la información que contenga, poniendo a prueba la persistencia de la base de datos.

5.05.05 Prueba de validación

La prueba de validación tiene como misión comprobar que los todos los elementos del siatema es decir cajas de texto, campos de lista, etc cumplan con los parámetros para que no exista información errónea.

CAPITULO VI: ASPECTOS ADMINISTRATIVOS

Tabla: 11 Recursos

HUMANO	NOMBRE	ACTIVIDAD	RESPONSABILIDAD
Tutor	Dr. Luis Rios	Director de proyecto	Guia del proyecto
Gerente	Xavier Flores	Autoriza la elaboracion del proyecto	Mando directivo
Secretaria	Jenny Dávila	Proporciona informacion	Informacion general
Bodeguero	Jenny Dávila	Datos del inventario	Manejo de invemtario
vendedor	Juan Jarrin	Informacion de ventas	Oferta de productos

6.01 Recursos Economicos

Tabla: 11 Recursos

DESCRIPCION	CANTIDAD	VALOR UNITARIO	VALOR TOTAL
Celular	1	500,00	500,00
Computadora	1	1200,00	1200,00
Tablet	1	700,00	700,00
Impresora	1	70,00	70,00
Internet	6	20,00	120,00
Datos moviles	6	18,00	108,00
Material de oficina	6	6,00	36,00
Anillados	3	3,00	9,00
Empastados	5	5,00	25,00
Servicios Basicos	6	20,00	120,00
Telefono	6	7,00	42,00
Transporte	20	2,00	40,00
Alimentacion	80	3,00	240,00
Seminario	1	500,00	500,00
Tutorias	1	200,00	200,00
Total			3910,00

6.02 Cronograma de actividades

	Nombre de la tarea	Fecha de inicio	Fecha de finalización	Dura
1	PROYECTO TESIS	14/10/13	04/04/14	125
2	CAPÍTULO I : ANTECEDENTES	11/11/13	02/12/13	16
3	1.01 CONTEXTO	11/11/13	15/11/13	5
4	1.02 JUSTIFICACIÓN E IMPORTANCIA	18/11/13	22/11/13	5
5	1.03 MATRIZ T	25/11/13	02/12/13	6
6	CAPÍTULO II: ANÁLISIS DE INVOLUCRADOS	03/12/13	20/12/13	14
7	2.01 MAPEO DE INVOLUCRADOS	03/12/13	11/12/13	7
8	2.02 MATRIZ DE INVOLUCRADOS	12/12/13	20/12/13	7
9	CAPÍTULO III: PROBLEMAS Y OBJETIVOS	23/12/13	17/01/14	20
10	3.01 ÁRBOL DE PROBLEMAS	23/12/13	03/01/14	10
11	3.02 ÁRBOL DE OBJETIVOS	06/01/14	17/01/14	10
12	CAPÍTULO IV: ANÁLISIS DE ALTERNATIVAS	20/01/14	14/02/14	20
13	4.01 MATRIZ DE ANÁLISIS DE ALTERNATIVAS	20/01/14	24/01/14	5
14	4.02 MATRIZ DE IMPACTO DE LOS OBJETIVOS	27/01/14	31/01/14	5
15	4.03 DIAGRAMA DE ESTRATEGIAS	03/02/14	07/02/14	5
16	4.05 MATRIZ DE MARCO LÓGICO	10/02/14	14/02/14	5
17	CAPÍTULO V: DESARROLLO DE LA PROPUESTA	17/02/14	17/03/14	21
18	5.01 JUSTIFICACIÓN TÉCNICA	17/02/14	17/02/14	1
19	5.02 ANÁLISIS Y DISEÑO	18/02/14	18/02/14	1
20	5.02.01 DIAGRAMA DE CASO DE USO PRINCIPAL	19/02/14	21/02/14	3
21	5.02.02 DISEÑO DE CASOS DE USO	24/02/14	26/02/14	3
22	5.02.03 DIAGRAMAS DE SECUENCIA	27/02/14	27/02/14	1
23	5.02.04 DIAGRAMA DE COLABORACIÓN	27/02/14	27/02/14	1
24	5.02.05 DIAGRAMA DE COMPONENTES	28/02/14	28/02/14	1
25	5.02.06 DIAGRAMA DE CLASES	03/03/14	05/03/14	3
26	5.03 DESARROLLO	05/03/14	07/03/14	3
27	5.03.01 ARQUITECTURA DEL SISTEMA	10/03/14	10/03/14	1
28	5.03.02 ESTÁNDARES	10/03/14	10/03/14	1
29	5.03.03 BASE DE DATOS	11/03/14	11/03/14	1
30	5.03.04 DISEÑO DE INTERFACES	12/03/14	12/03/14	1
31	5.04 PRUEBAS	13/03/14	13/03/14	1
32	5.04.01 PRUEBAS DE INTEGRIDAD	13/03/14	13/03/14	1
33	5.04.02 PRUEBAS DE MÓDULO	13/03/14	13/03/14	1
34	5.04.03 PRUEBAS DE INTERFAZ DE USUARIO	14/03/14	14/03/14	1
35	5.04.04 PRUEBAS DE CARGA	14/03/14	14/03/14	1
36	5.04.05 PRUEBAS DE VALIDACIÓN	14/03/14	14/03/14	1
37	5.05 CASOS DE PRUEBA	17/03/14	17/03/14	1
38	CAPÍTULO VI: ASPECTOS ADMINISTRATIVOS	18/03/14	28/03/14	9
39	6.01 RECURSOS	18/03/14	19/03/14	2
40	6.02 PRESUPUESTO	20/03/14	21/03/14	2
41	6.03 CRONOGRAMA DE ACTIVIDADES	24/03/14	28/03/14	5
42	CAPÍTULO VII: CONCLUSIONES Y RECOMENDACIONES	31/03/14	04/04/14	5
43	7.01 CONCLUSIONES	31/03/14	02/04/14	3
44	7.02 RECOMENDACIONES	03/04/14	04/04/14	2

CAPITULO VI: CONCLUSIONES Y RECOMENDASIONES

7.01 Conclusiones

Con el desarrollo de esta aplicación hemos llegado a tener las siguientes conclusiones:

- La empresa SIVADE implementado el software que se ha desarrollado lograra agilitar tanto las funciones del personal de bodega como de los agentes promotores de ventas de la siguiente forma:
  - Reduciendo el tiempo de consulta.
  - Agilizando el ingreso de productos al stock.
  - Teniendo una idea mas clara del inventario que tiene la empresa.
- El problema que tiene la empresa con respecto a su inventario es que la consulta de productos ahí que siempre estarla realizando en el ordenador de la empresa y esto produce pérdida de tiempo y demora en las ventas
- Con la implementación de la aplicación podremos llevar un control constante del inventario en cualquier lugar que el usuario se encuentre, y también podrán registrar pedido tanto el bodeguero como los promotores de ventas.
- Para el desarrollo de esta aplicación usamos varias estrategias como son el desarrollo de la base de datos, la interface de usuario, y el desarrollo del software
- Para esto implementamos varias herramientas como son MySQL,(Base de dato), Eclipce(Programacion Java), HeidiSQL(Emulador Android) que nos permitió el desarrollo de todos lo módulos del software.

- Para el desarrollo de este proyecto tuvimos un tiempo de 7 meses con un costo de 3000 dólares aproximadamente.
- En el

7.02 Recomendaciones

- Pedrir la información correcta para ingrersar al sistema.
- Capacitar a los usuarios en el manejo de la aplicación mediante cursos cada cierto tiempo.
- Implementar buenos equipos para la instalación del software y así haya un mejor rendimiento.
- Concientizar al personal acerca del los beneficios que la empres esta teniendo para su comodidad laboral.
- Leer los manuales que se encuentran en este documento antes de usar la aplicación.

ANEXOS

Anexo N. 01 Análisis de Involucrados

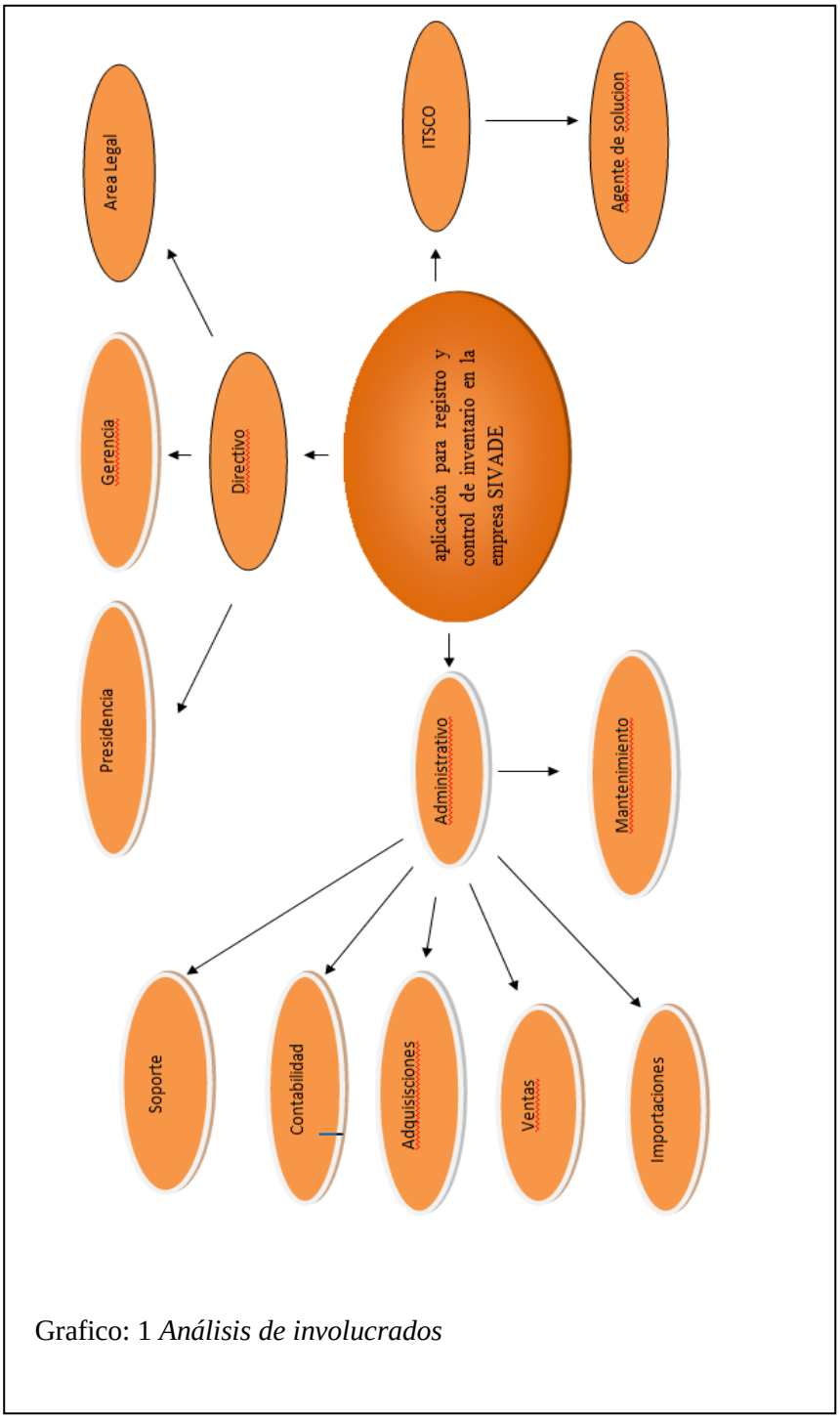


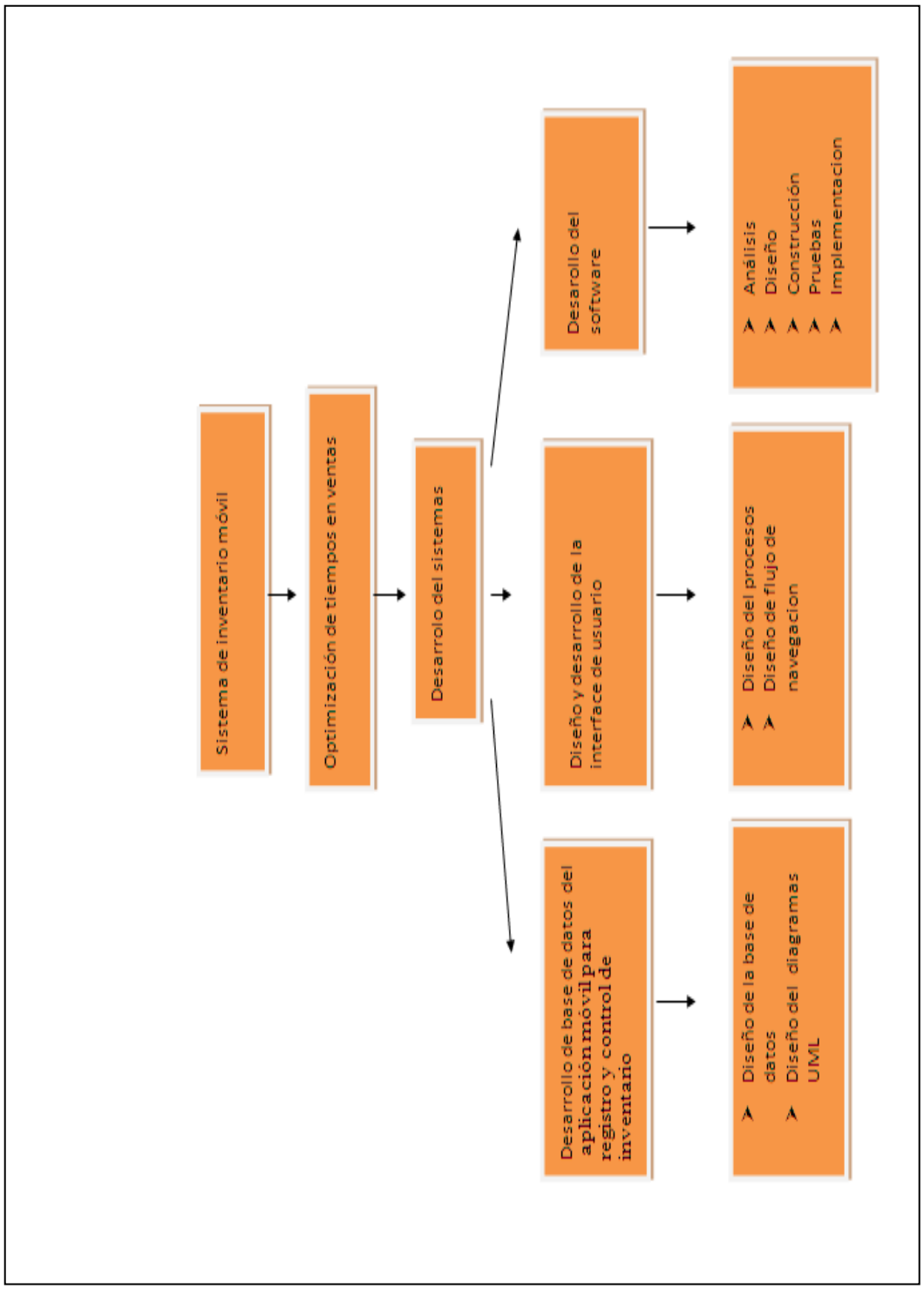
Grafico: 1 *Análisis de involucrados*

Anexo A.02 Mapeo de Análisis de Involucrados

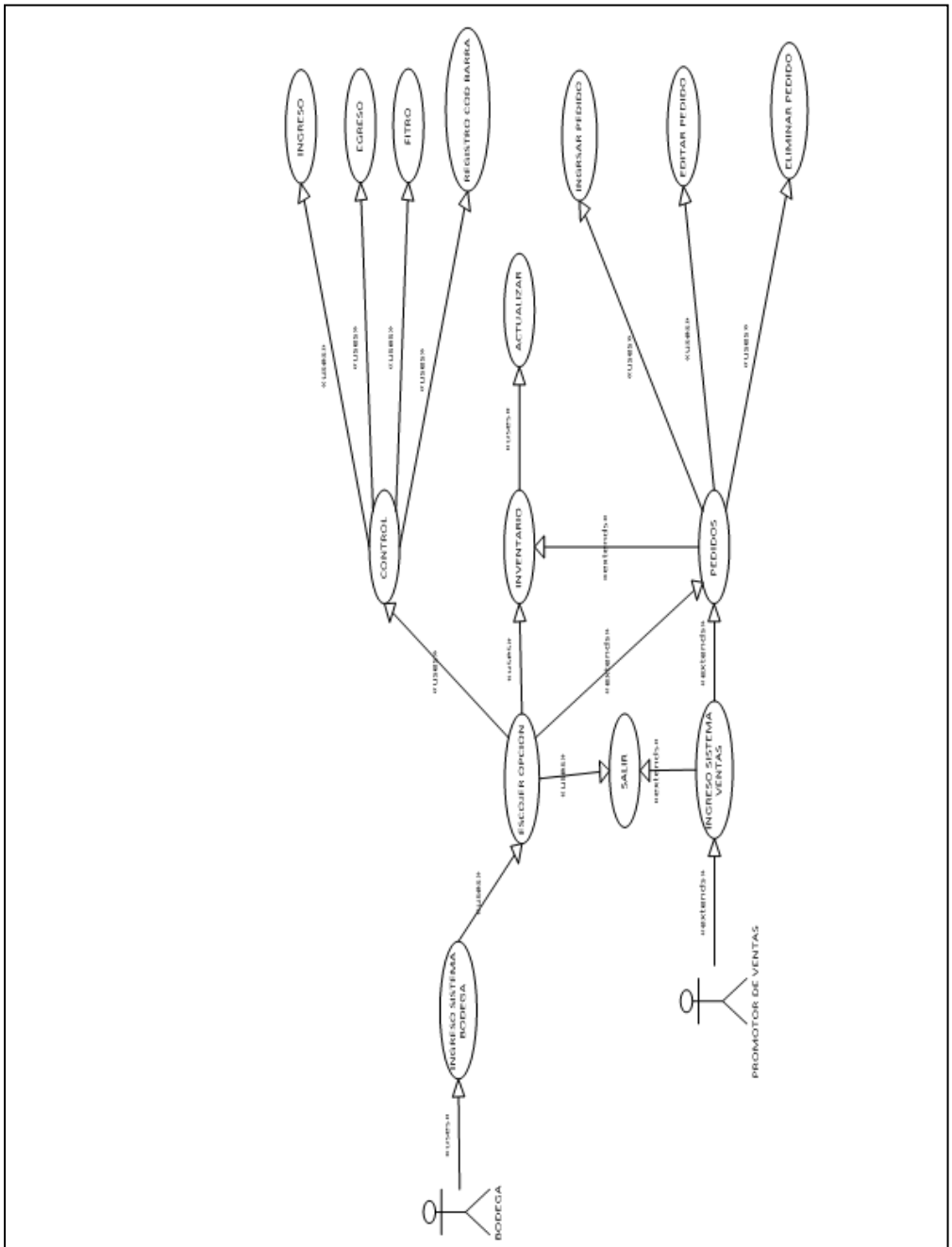
Tabla: 2 Mapeo de análisis de involucrados

Actores Involucrados	Intereses sobre el problema central	Problemas percibidos	Recursos Mandatos y Capacidades	Intereses sobre el proyecto	Conflictos Potenciales
Presidencia	Control del inventario y ventas	Perdidas de recursos	Infraestructura y personal	Tener control de su empresa en todo momento	Ninguno
Geréncia	Mejor administración de los departamentos	No tiene un control constante de sus bodegas y sus ventas	Capacitaciones, vinculación a la empresa	Aumentar la efectividad de su trabajo.	Ninguno
Véntas	Acceso mas rápido al inventario	Pérdida de tiempo al consultar el inventario	Compromiso y lealtad	Lograr cubrir los problemas en el menor tiempo posible	Mal ingreso de datos
Importaciones	Control del inventario	Consulta tardía	Adquisición de los productos	Surtido de los productos	Falta de stock
Instituto Tecnológico Superior Cordillera	Ninguno	Comprobación de conocimientos	Generar normas para el desarrollo del proyecto	Ejercer los conocimientos impartidos	Ninguno
Contabilidad	Datos más exactos	Perdida de datos	Control de los recursos	Facilidad de consulta	Confusión en el debe y el Haber

Anexo A.03 Diagrama de objetiv



Anexo A.04 Caso de Uso: Diagrama General



El diagrama describe los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso.

Anexo A.05: Diagrama de clases

Diagrama de clases

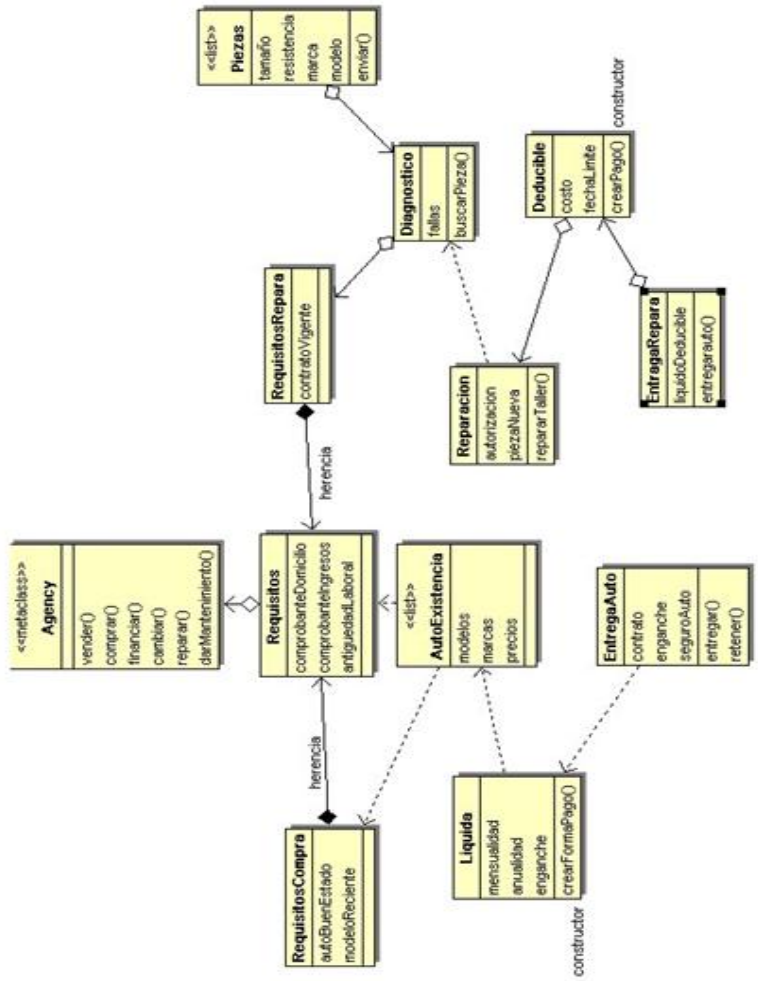


Figura: 23 Diagrama de clases

El diagrama de clases nos muestra como se relasionas las clases en el program

Anexo A.06 Modelo Logico

Modelo Logico

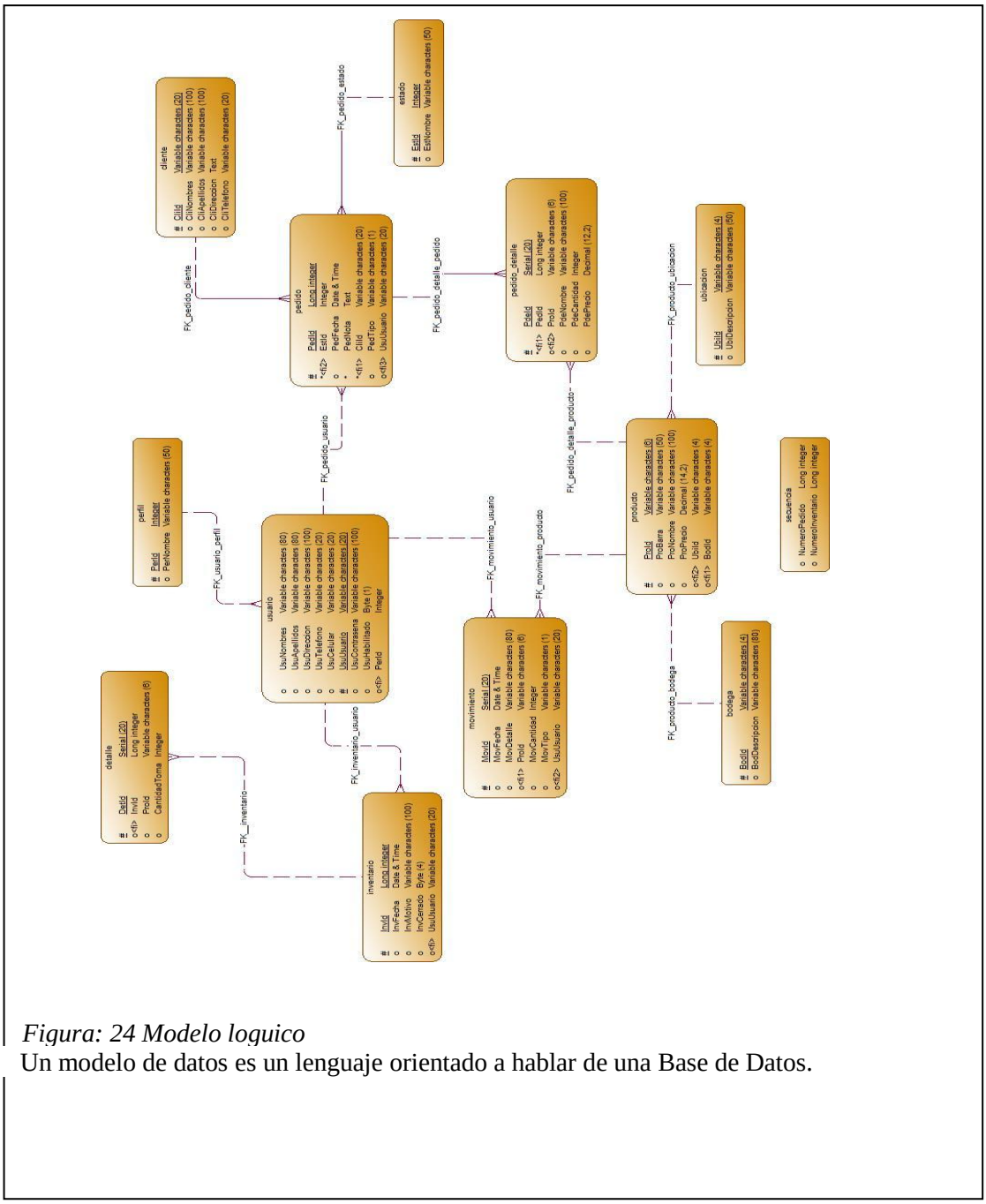


Figura: 24 Modelo loguico

Un modelo de datos es un lenguaje orientado a hablar de una Base de Datos.

Anexo A.07 Modelo Fisico

Modelo Fisico

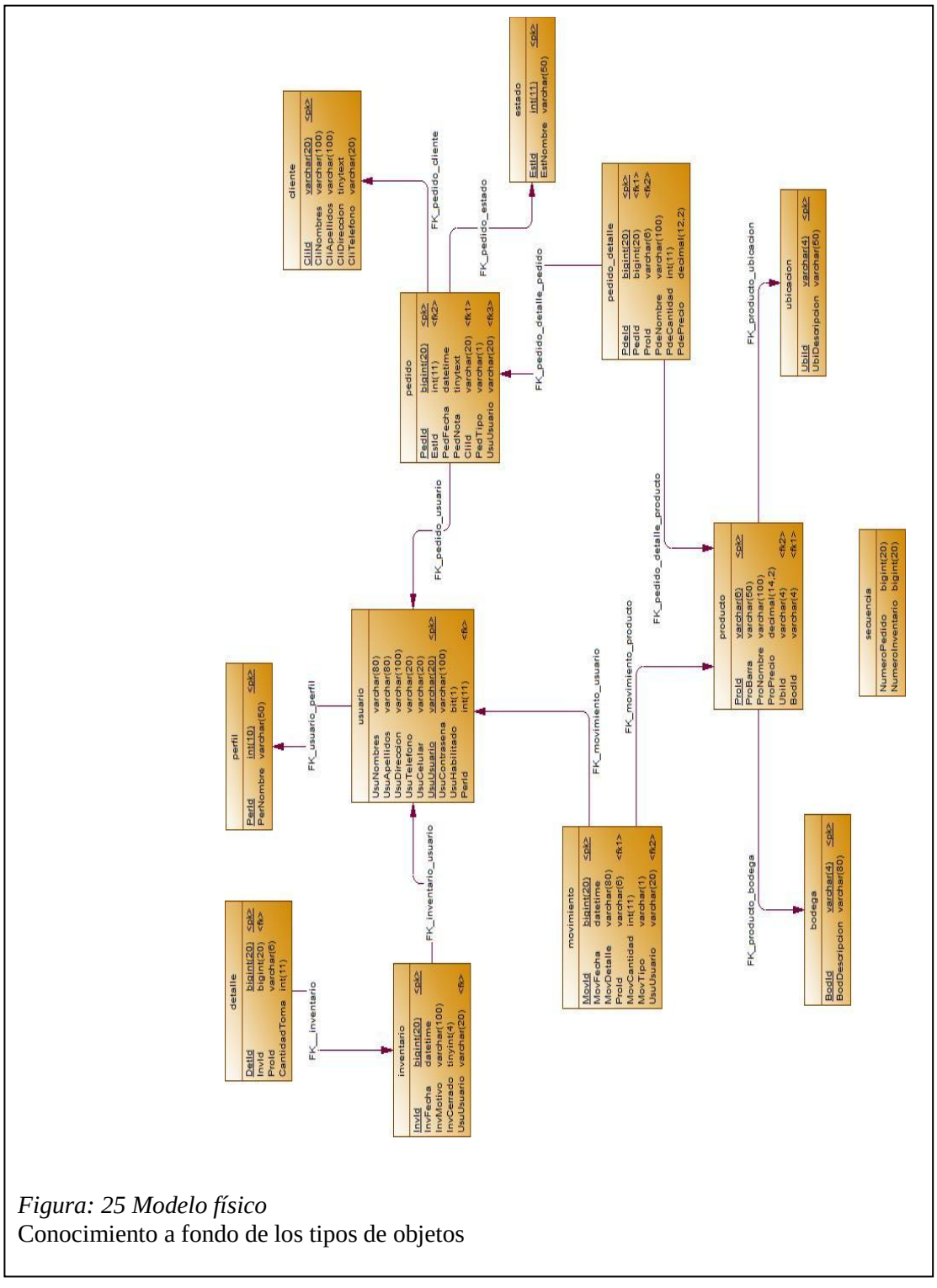


Figura: 25 Modelo físico

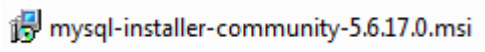
Conocimiento a fondo de los tipos de objetos

Anexo A.08 Instalación de programas utilizados

INSTALACUION DE MySQL

Paso 1:

- Ejecutar el archivo
- Clic en la opción run
- Esperar unos segundos



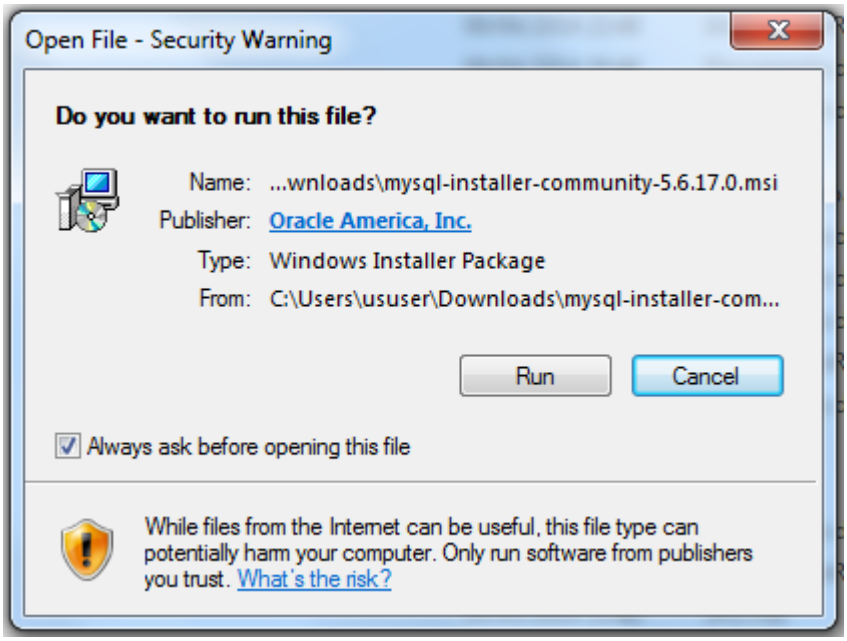


Figura 26: Instalacion

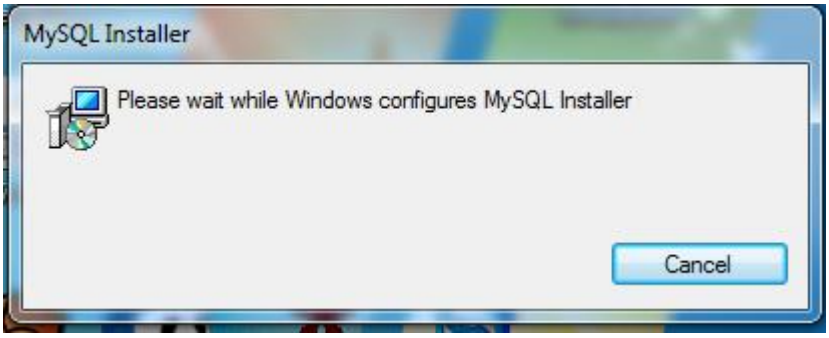


Figura: 26 Pasó 2

- En la pantalla welcome seleccionamos la opción Instal MySQL products

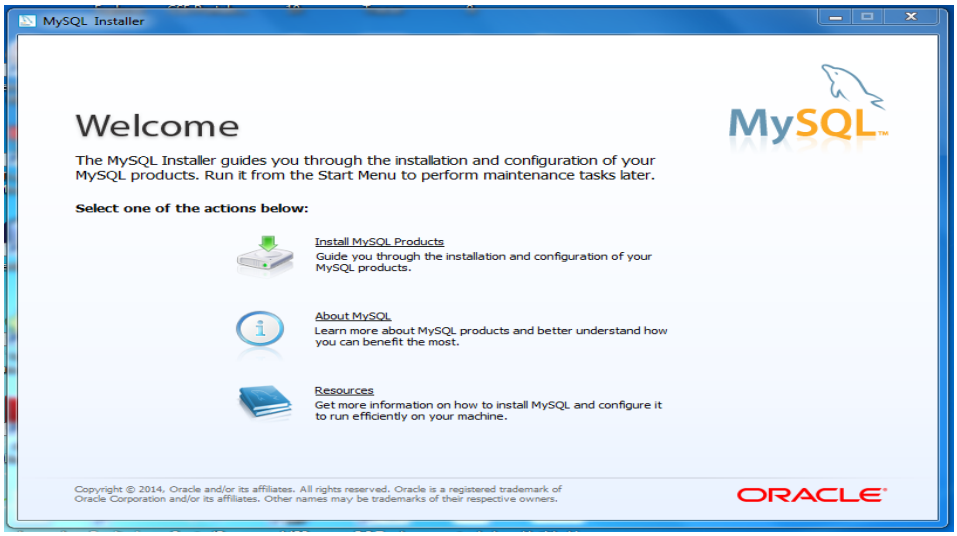


Figura: 27 Paso 3:

- Escogemos la opcion license Information
- Aceptamos la licencia
- Next

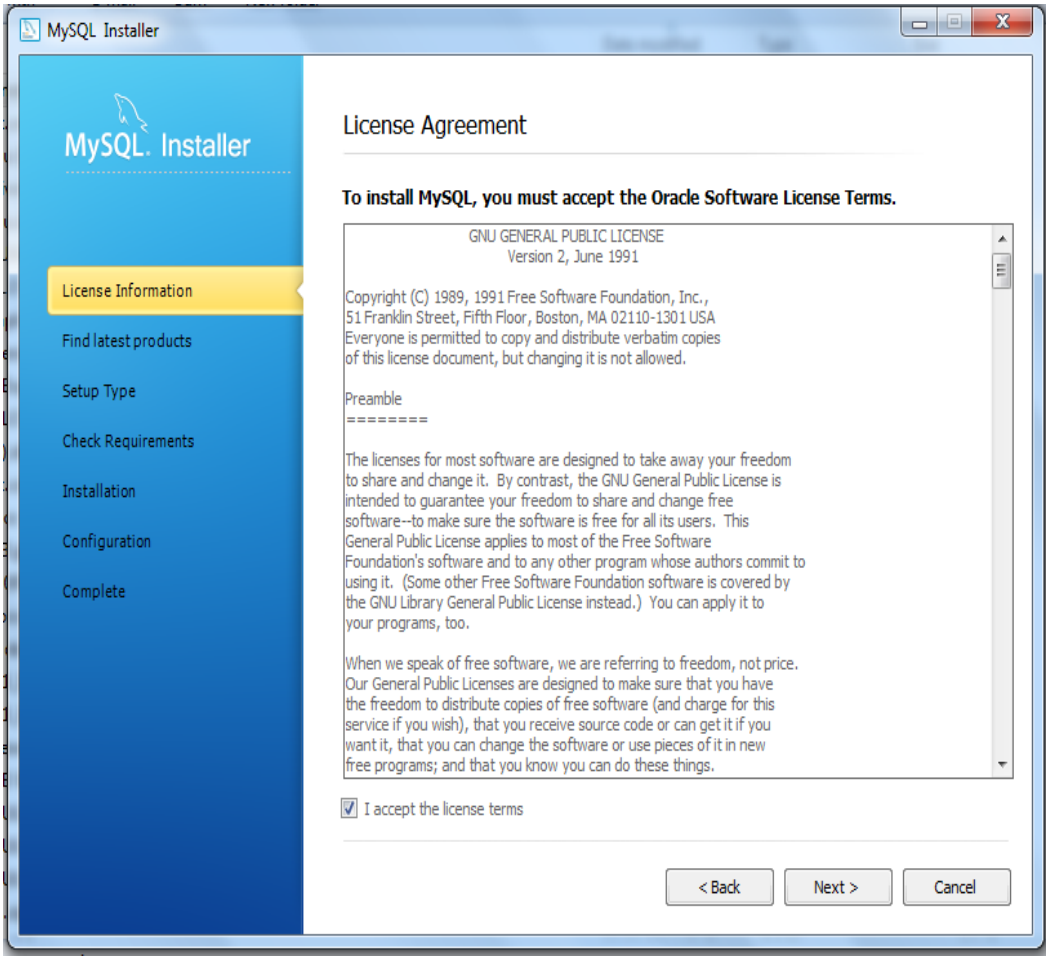


*Figura: 28 Paso**: 4***

- Escogemos la opción Find latest products
- Skip the check for updates(not recomended)
- Next

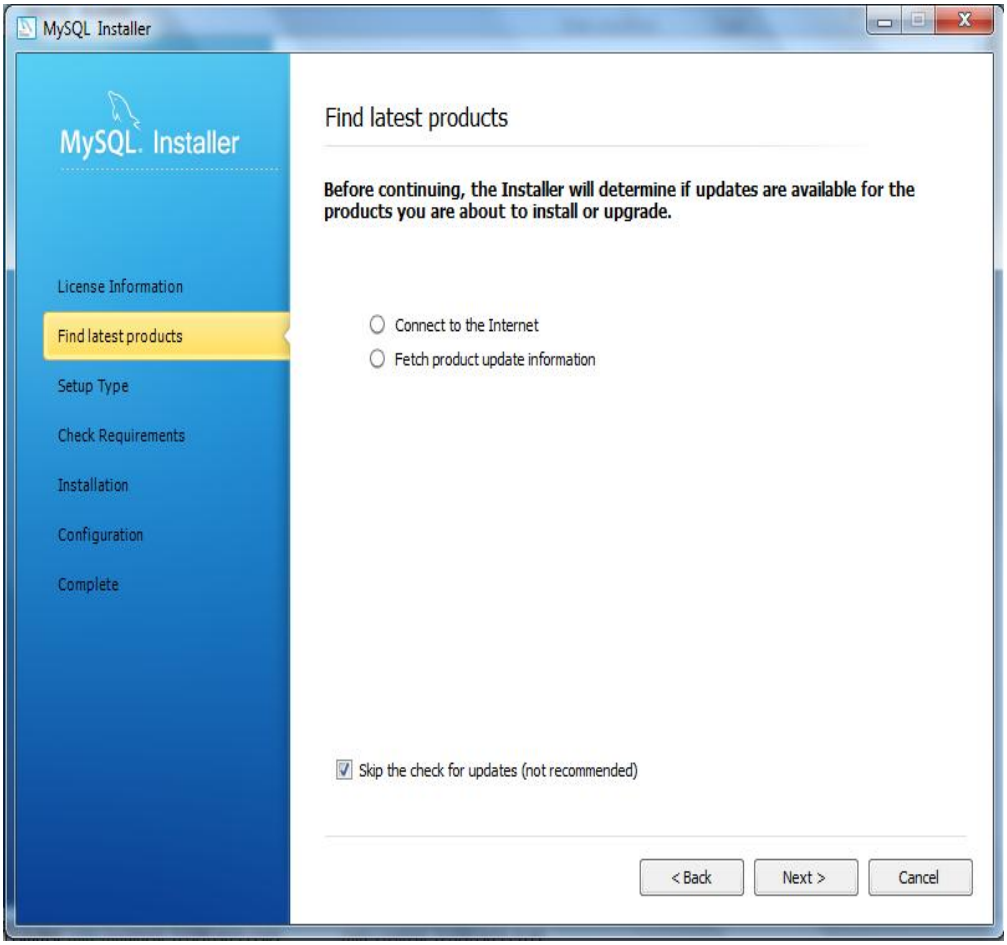


Figura 29 ***Paso: 5***

- Escogemos la opción Setup Type
- Escogemos la opción Server only
- Next

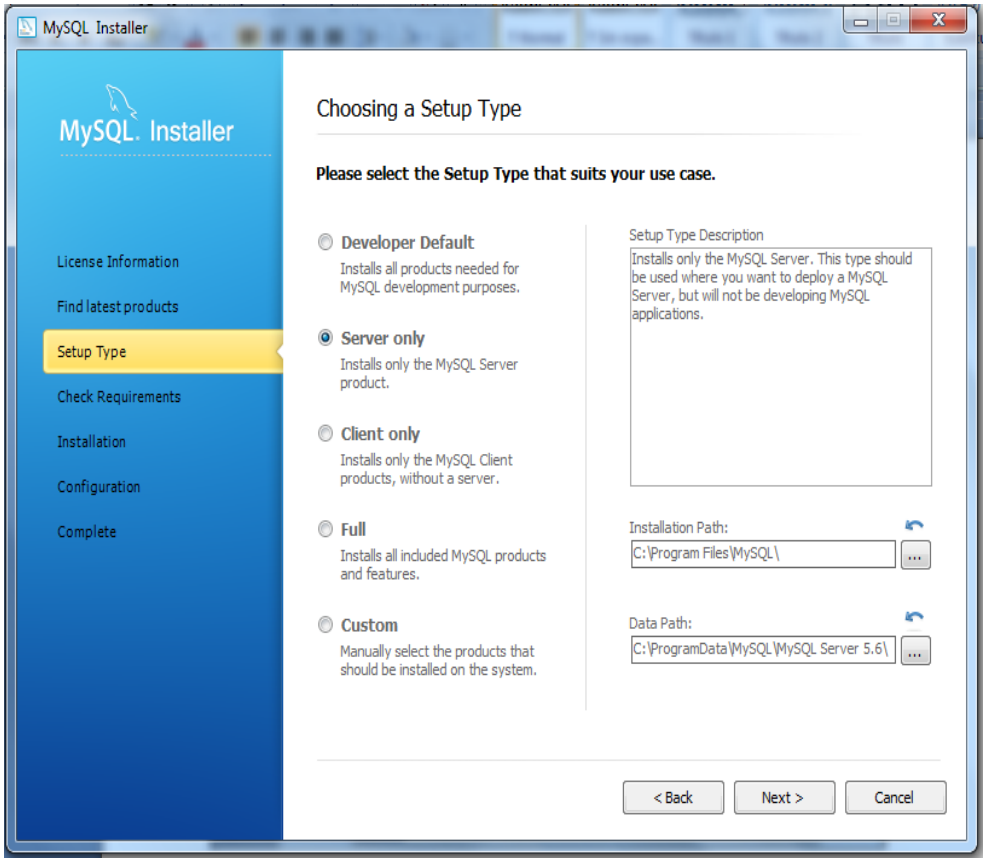


Figura30 ***Paso:6***

- Escogemos la opción Check Requeriments
- Next

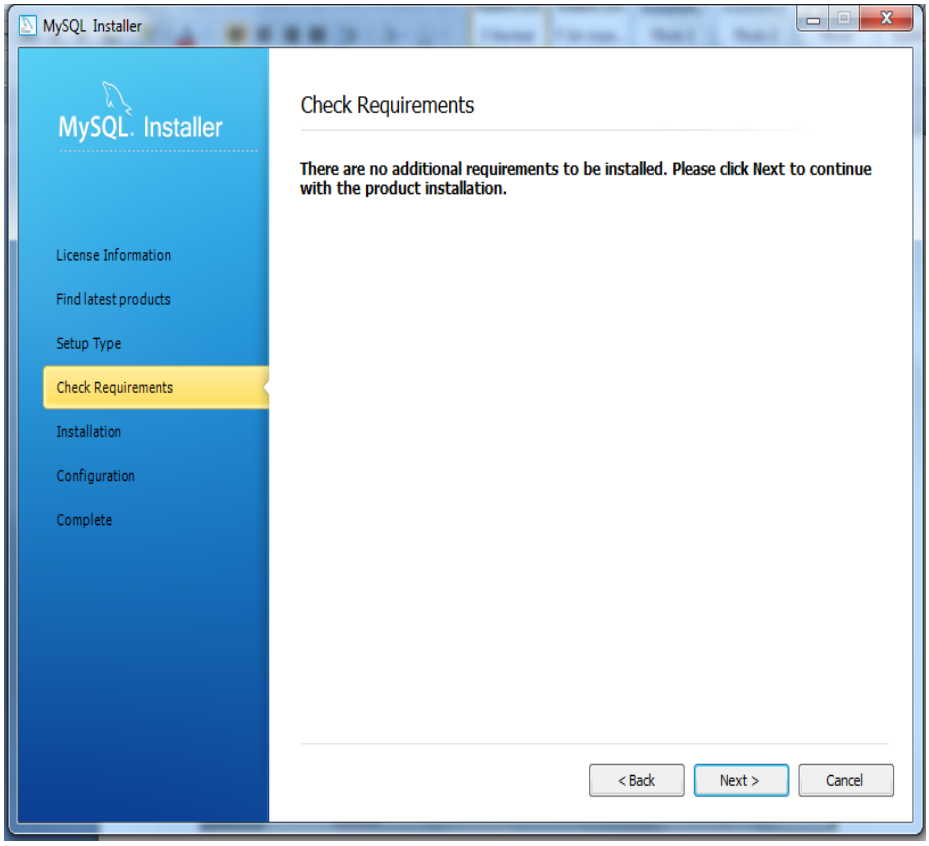


Figura 31 Paso: 7

- Escogemos la opción Installation.
- Seleccionamos MySQL Server 5.6.17.
- Excute.

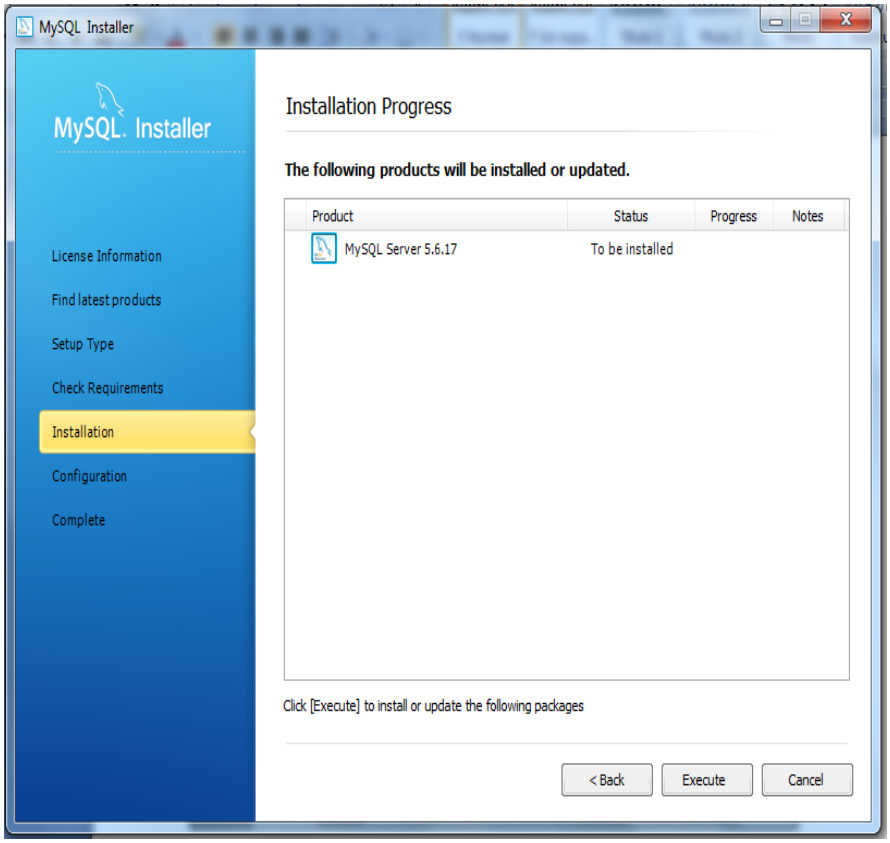


Figura 32 Paso: 8

- Esperamos unos segundos que se cargue.

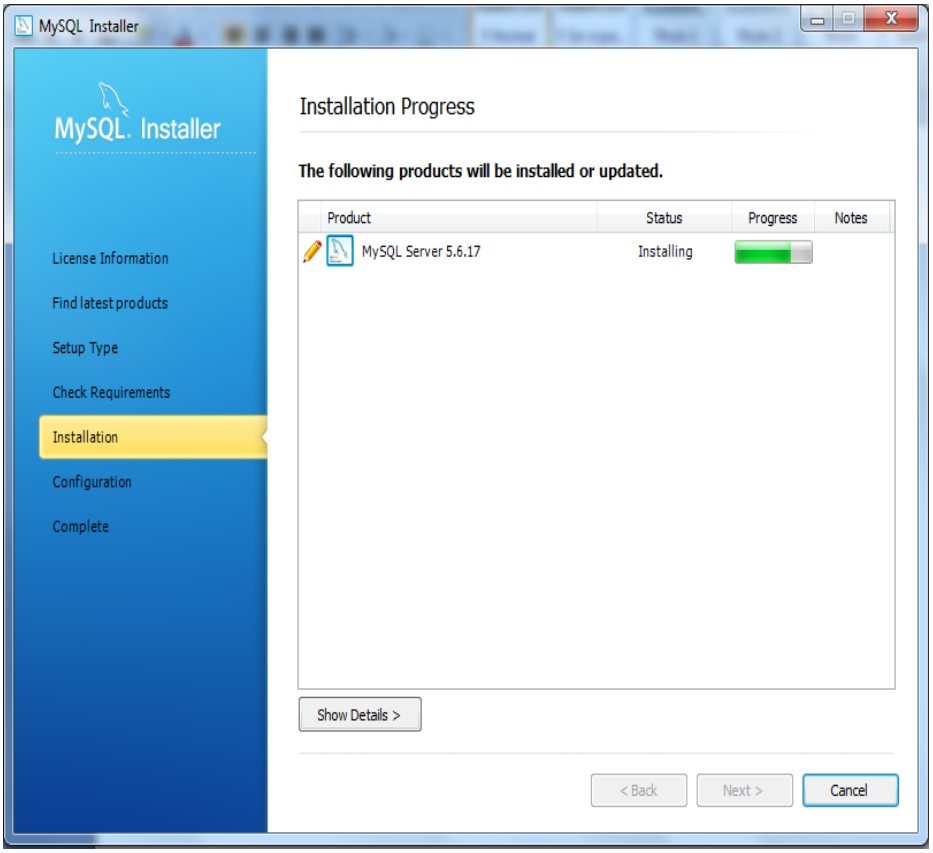


Figura 33 Paso: 9

- Una vez cargado seleccionamos next.

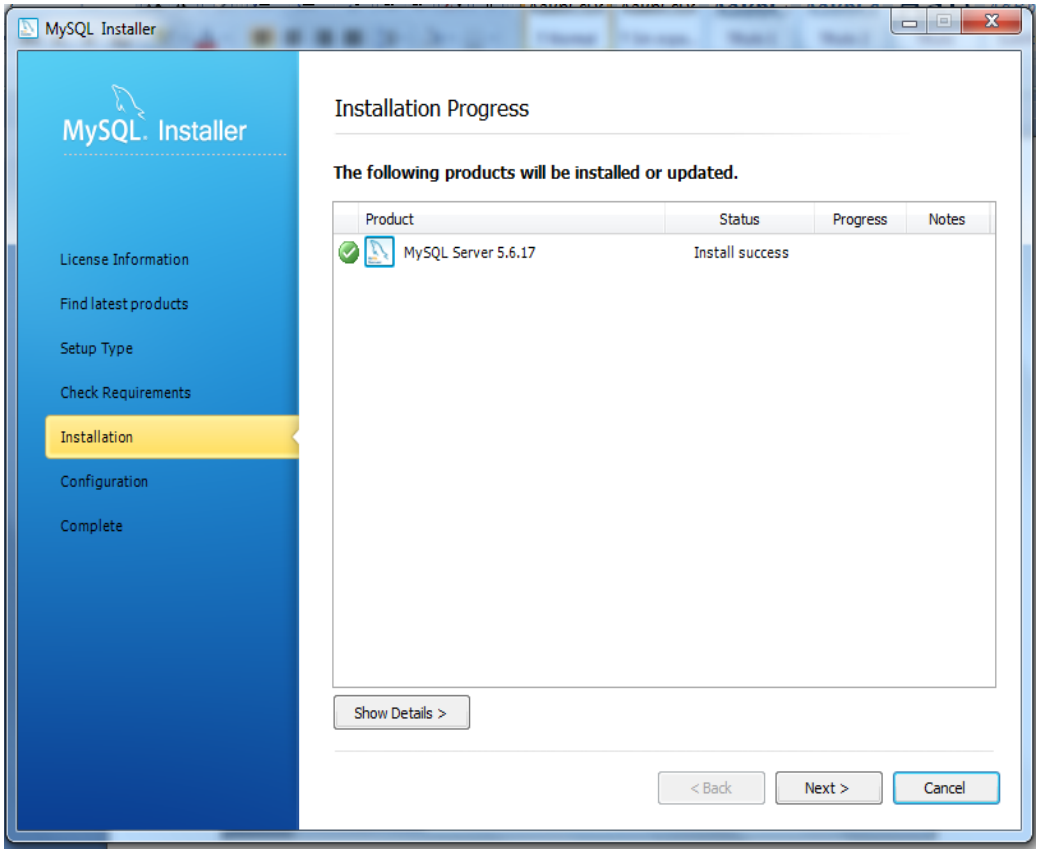


Figura 34 Paso: 10

- Escogemos Configuration.la opción
- Next.

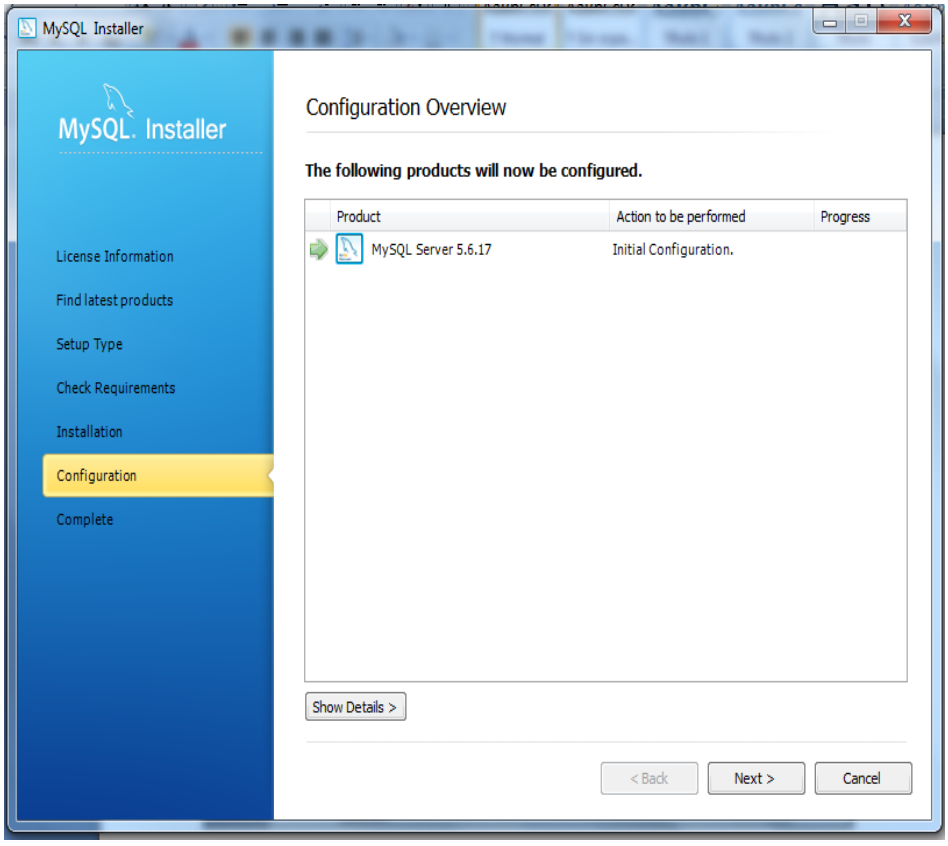
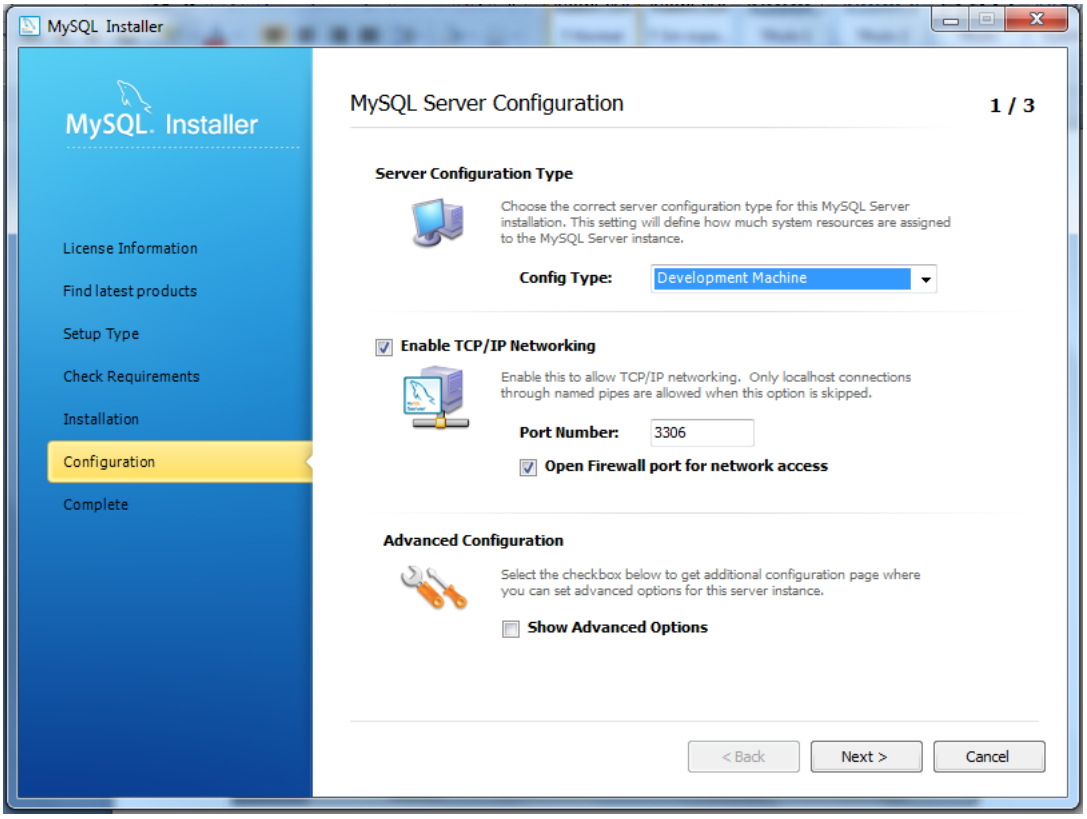


Figura 35 Paso: 11

- Escogemos Configuration.
- En la pestaña Config Type.
- Seleccionamos Development Machine
- Next.



PWD: ITCoordillera$

Figura 36 Paso: 12

- Escogemos Configuration.
- Ingresamos 2 contraseñas.
- Next.

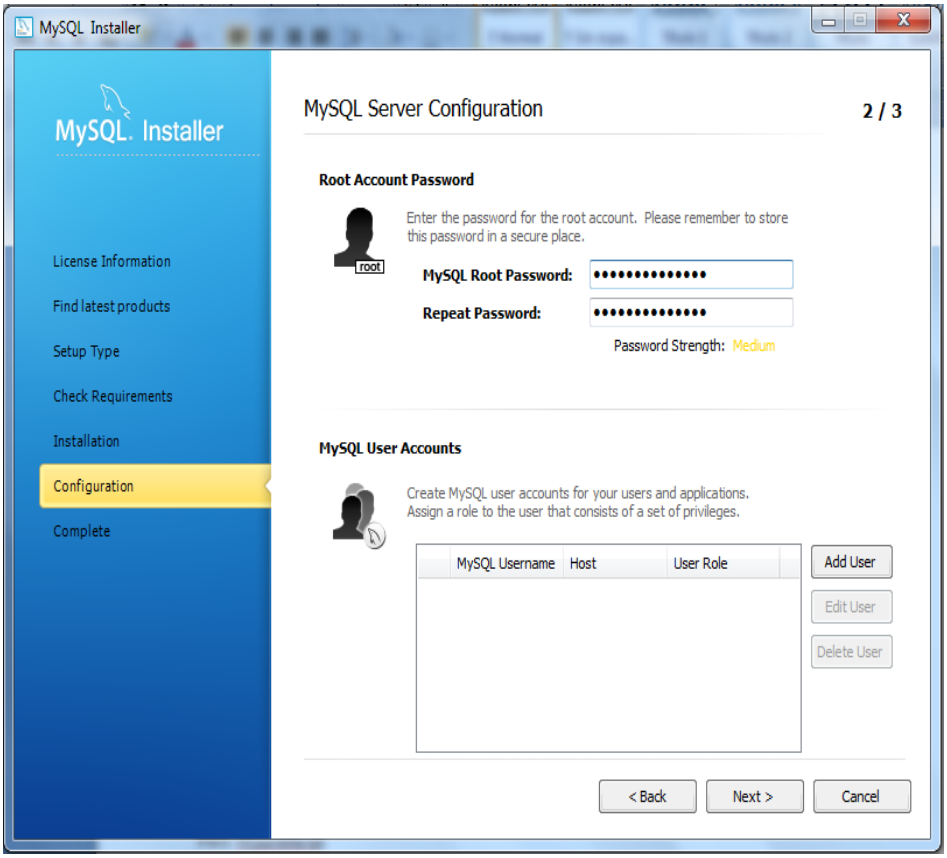


Figura 37Paso: 13

- Escogemos Configuration.
- En la pestaña Windows Service Name.
- Escribimos MySQL 5.6
- Next.

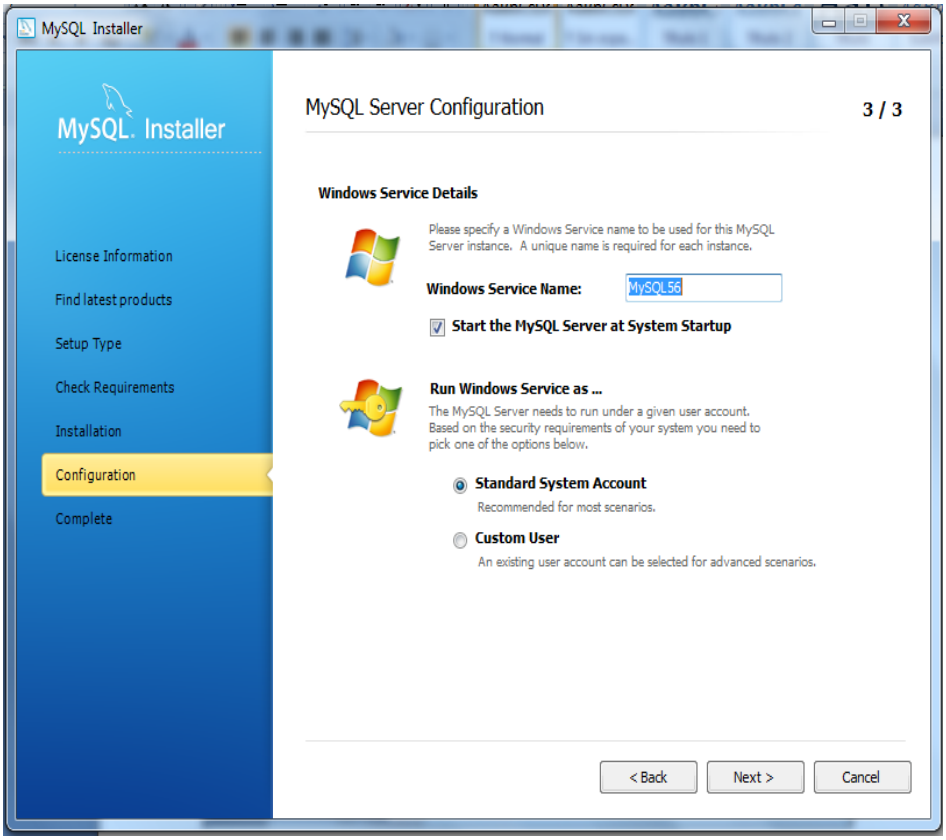


Figura 38 Paso: 14

- Esperamos que se cargue.

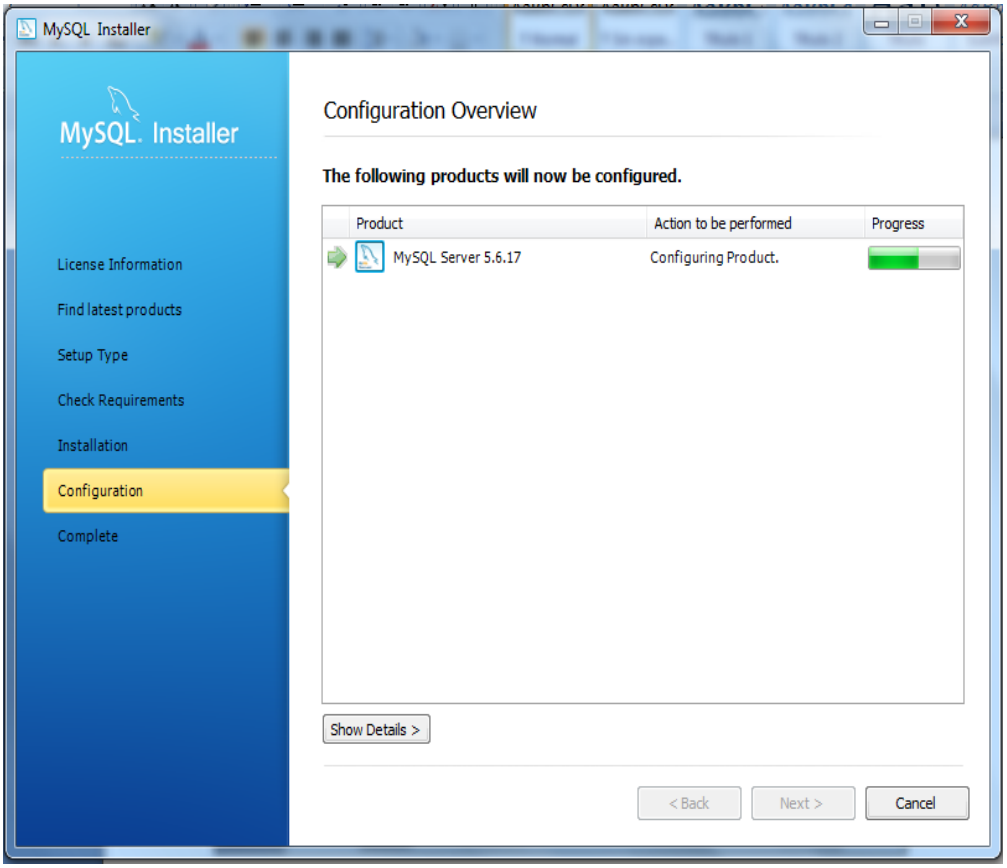


Figura 39Paso: 15

- Next.

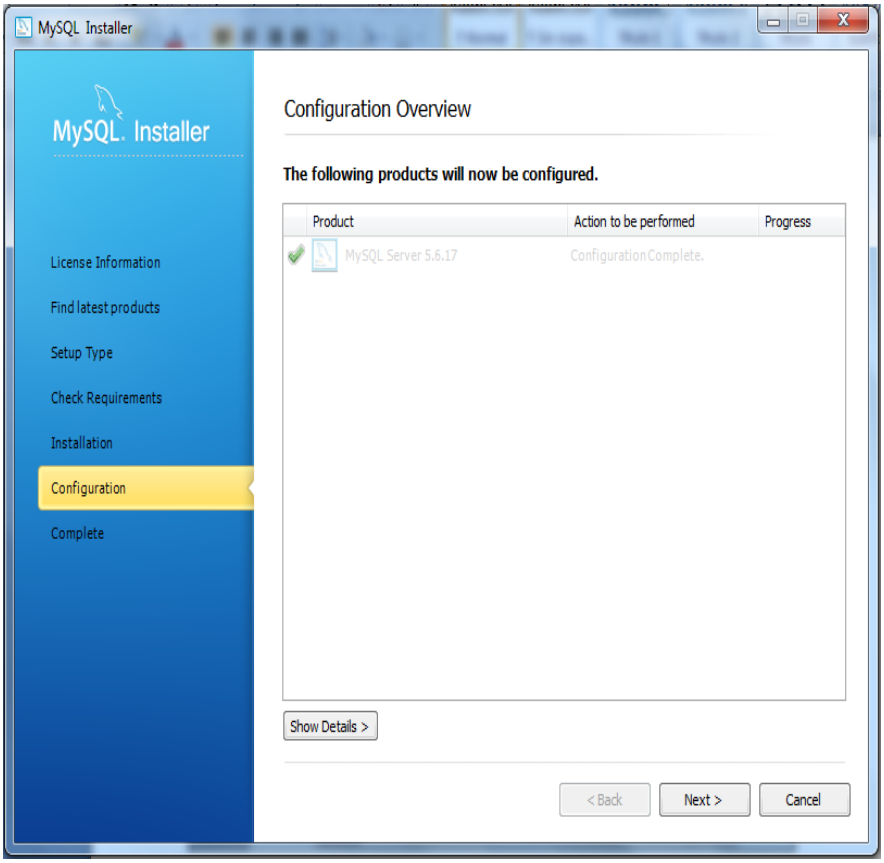


Figura 40 Paso: 16

- Escogemos Complete.
- Finish.

Anexo N. 4 Instalación de hidisql

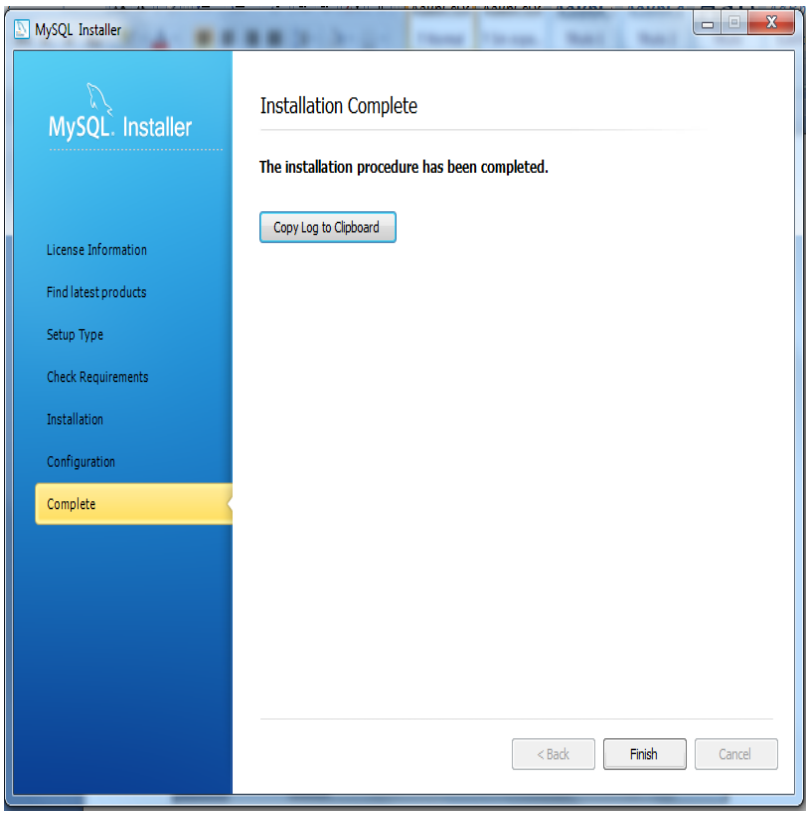


Figura 41 Paso: 1

- Descargar HeidiSQL (Cliente para MySQL) http://www.heidisql.com/.
- Ejecutar el archivo: HeidiSQL_8.3.0.4694_Setup.exe .
- Next.

Figura 42 Paso: 2

- Aceptamos la licencia.
- Next.

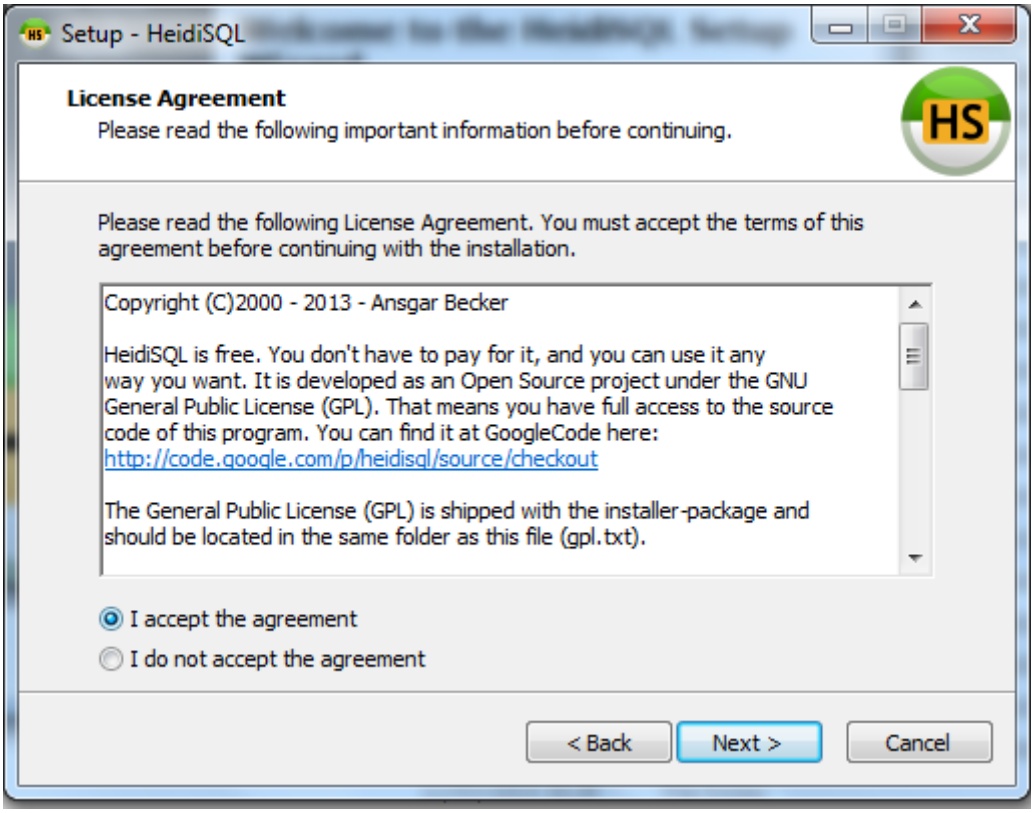


Figura 43 Paso: 3

- Seleccionamos el lugar donde se va a almacenar el archivo.
- Next.

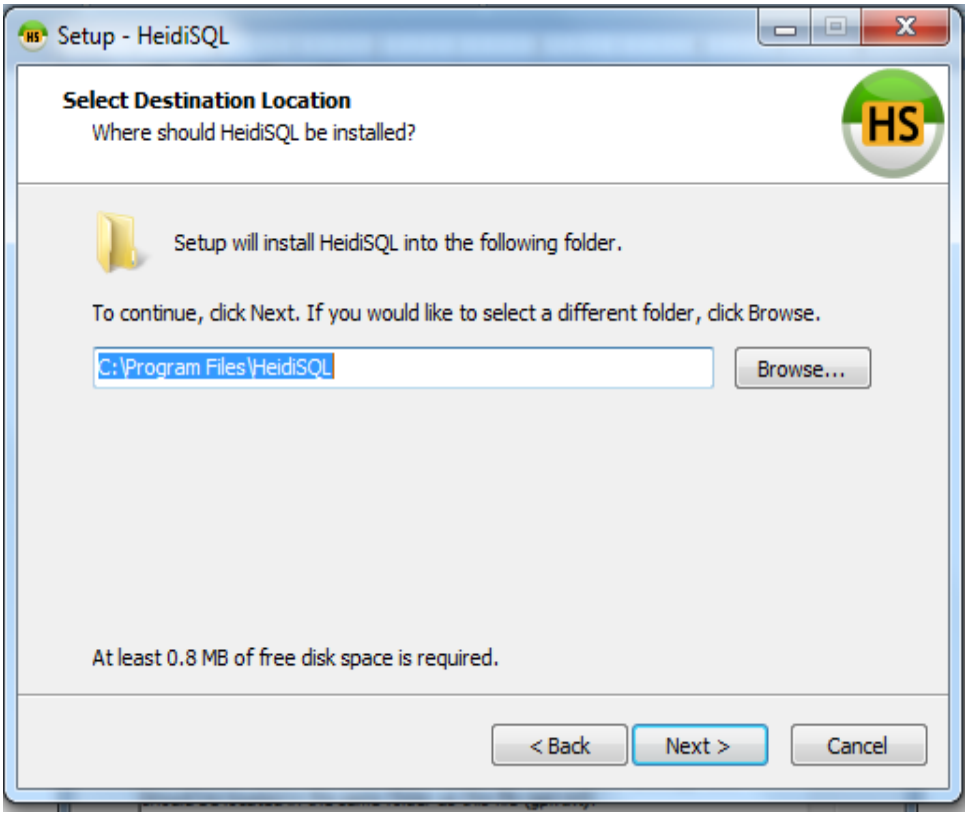


Figura 44 Paso: 4

- Next.

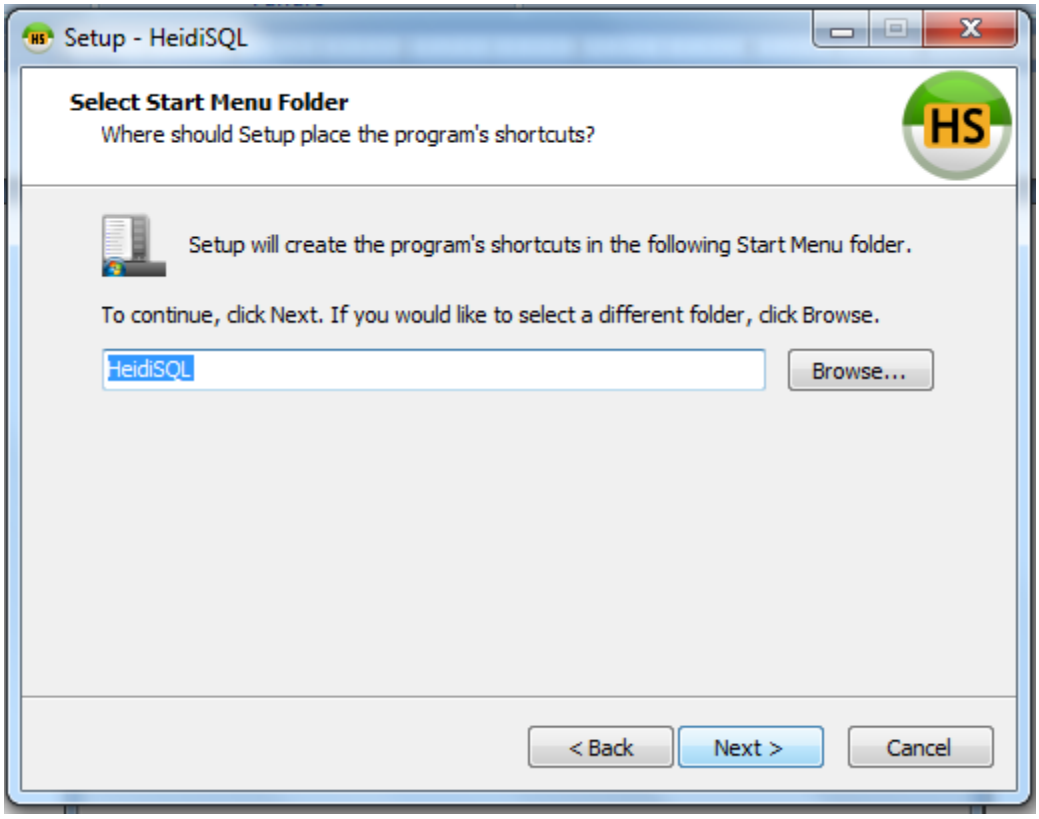


Figura 45 Paso: 5

- Seleccionamos todas las pestañas.
- Next.

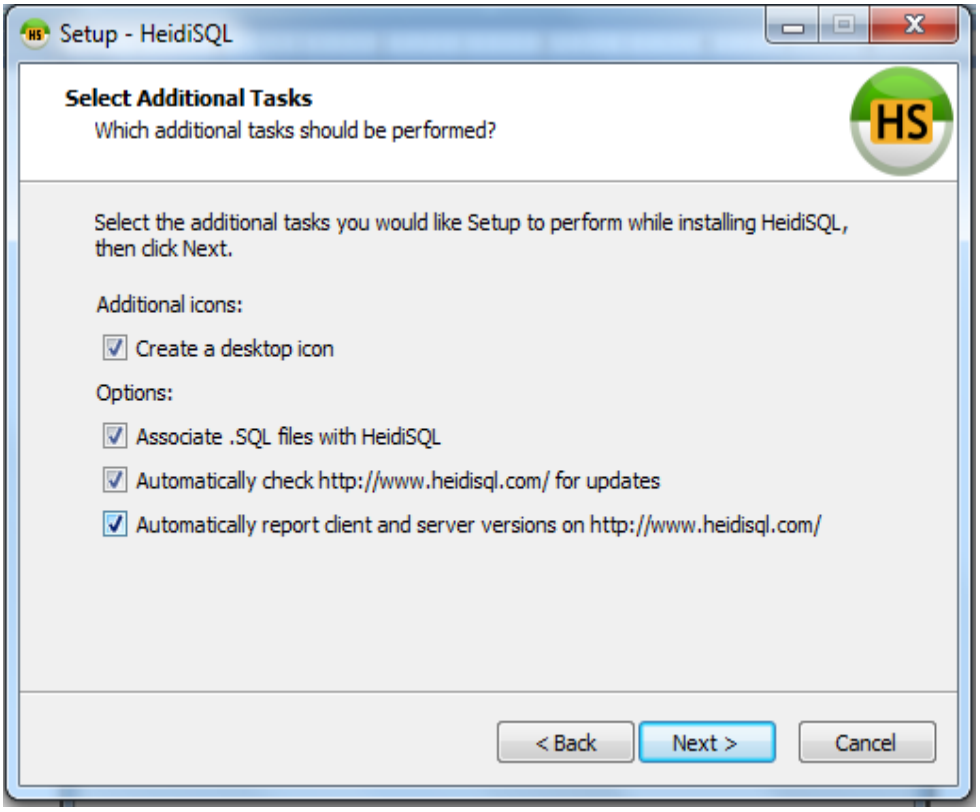


Figura 46 Paso: 6

- Install.

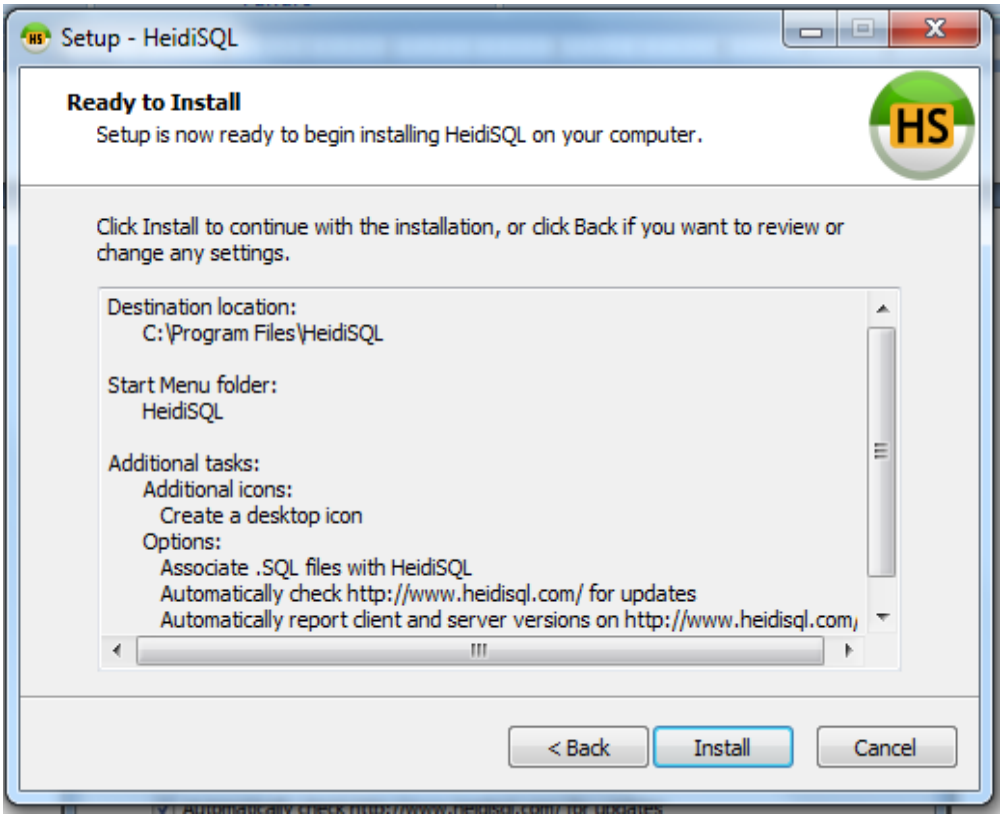


Figura 47Paso: 7

- Finish.

Figura 48 NOTA:
Crear una nueva conexion a MySQL para localhost

Anexo N.5 Instalación de emulador android

Paso: 1

- Abrimos el administrador de sesiones.
- Ajustes.
- Ingresamos la dirección Ip.
- Un usuario y contraseña.
- Abrir.

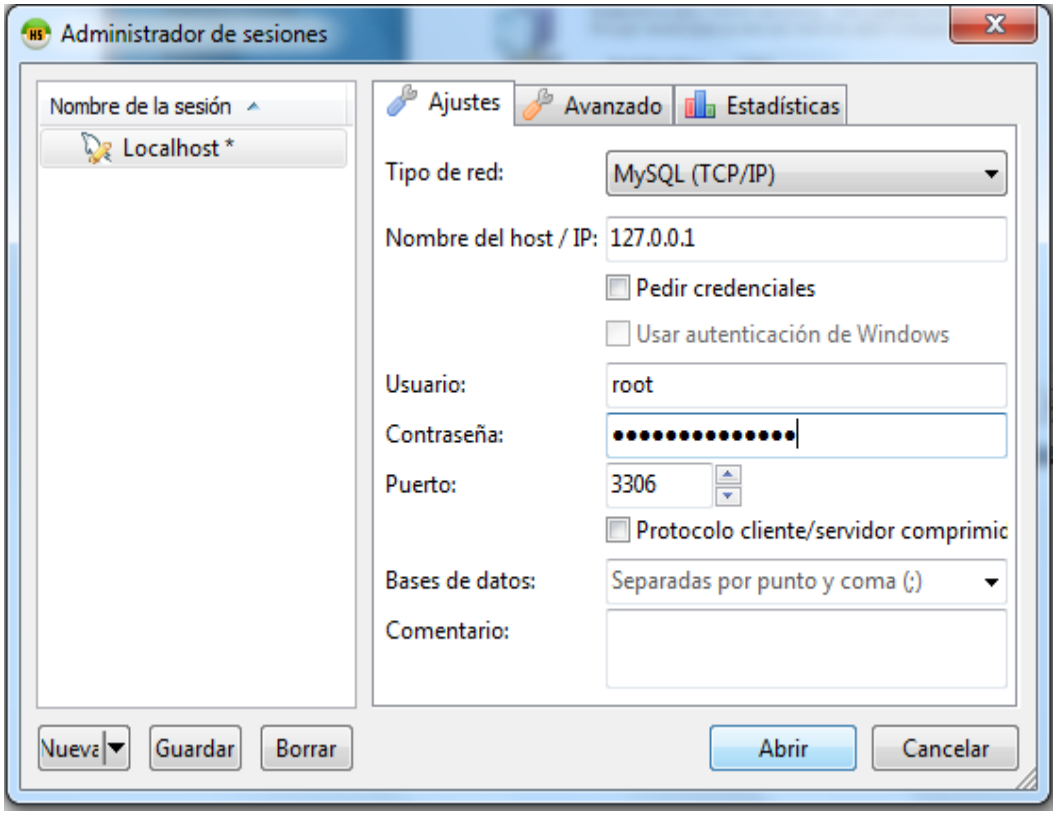


Figura 49 Localhost disponible

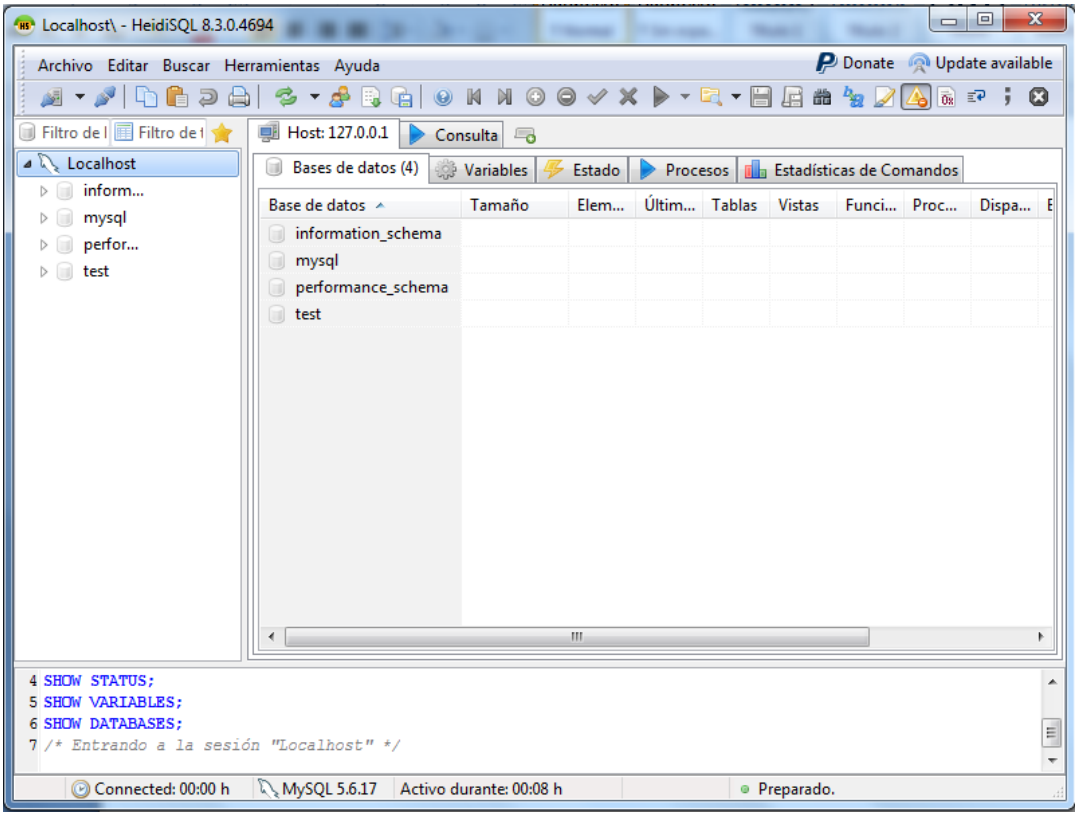


Figura 50 Paso: 2

- Descargar del **ADT (Android Developer Tools)**
  http://developer.android.com/sdk/index.html

Figura 51Paso: 3

- descomprimir el .zip

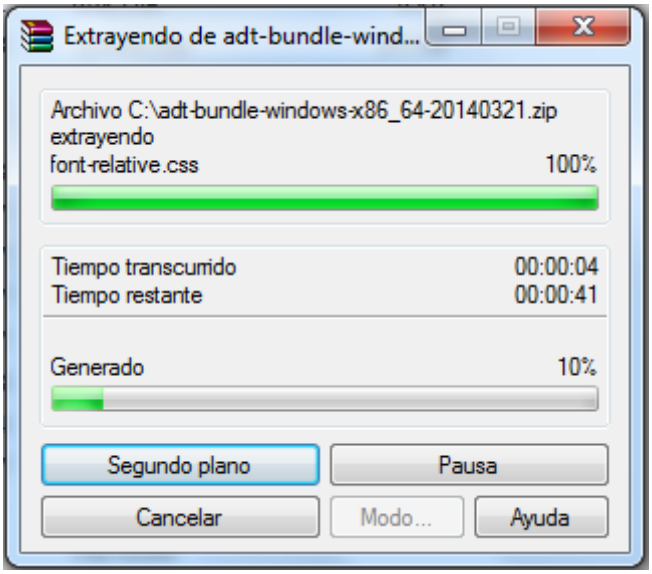


Figura 52Paso: 4

- Ingresar a la carpeta eclipse y ejecutar el archivo. eclipse.exe
- Crear un directorio de trabajo (workspace) y registrarlo al momento que inicia Android Developer Tools.
- OK.

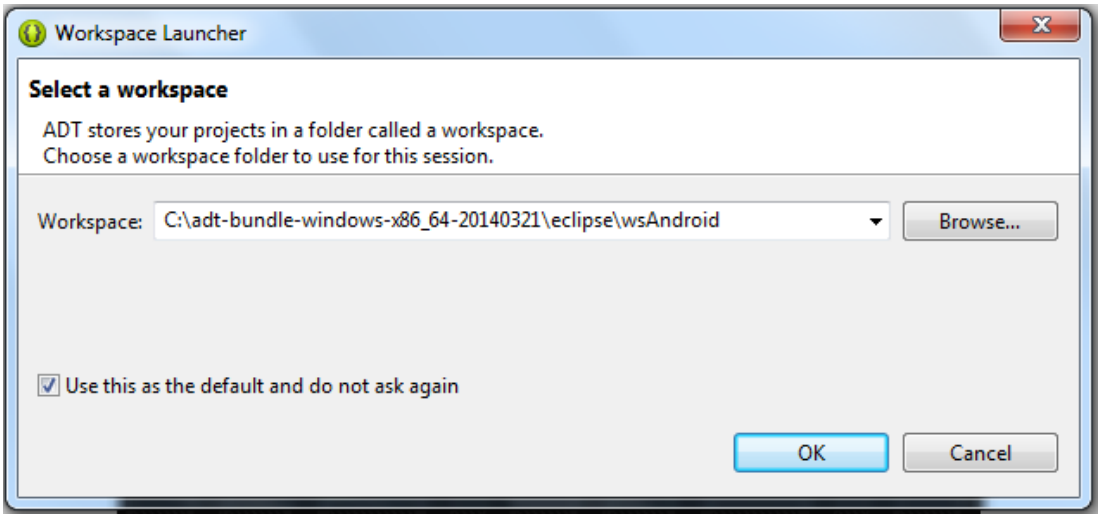


Figura 53 Paso: 5

- Dentro de eclipse clic en el icono

- New.
- OK.

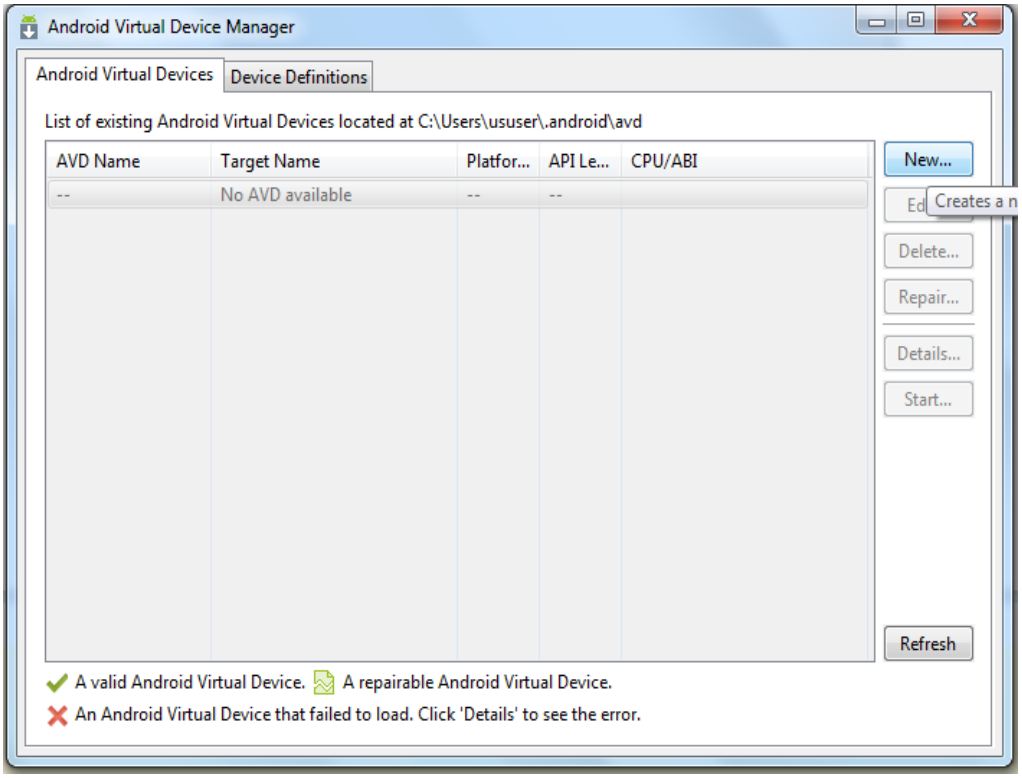


Figura 54 Paso: 6

- OK.

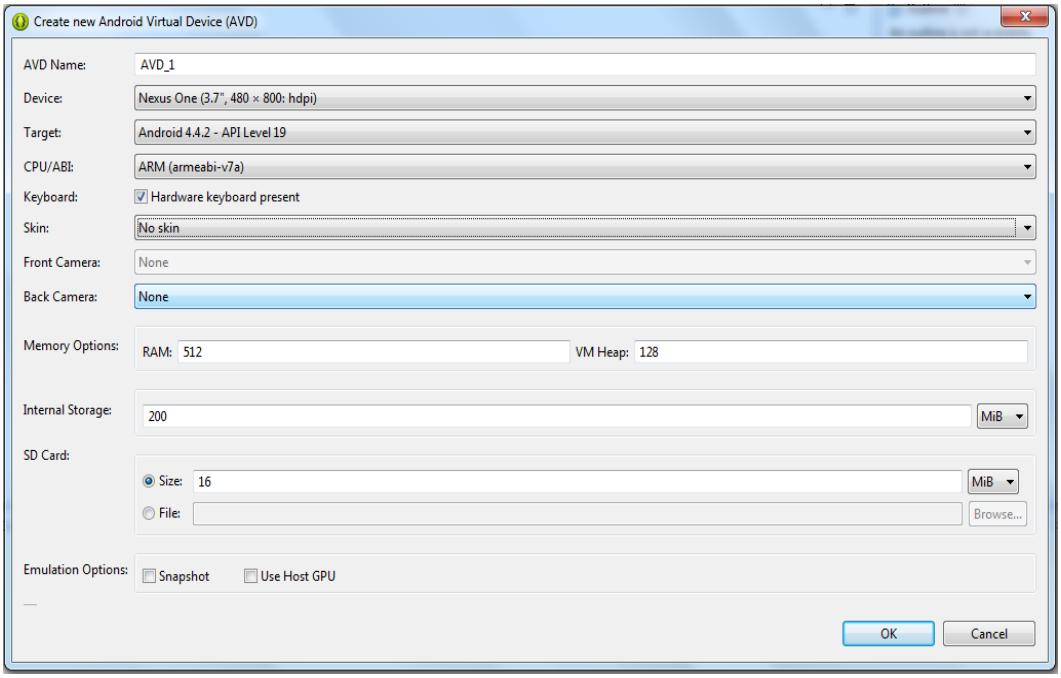


Figura 55 Paso: 7

- Aparecerá la siguiente ventana.
- New

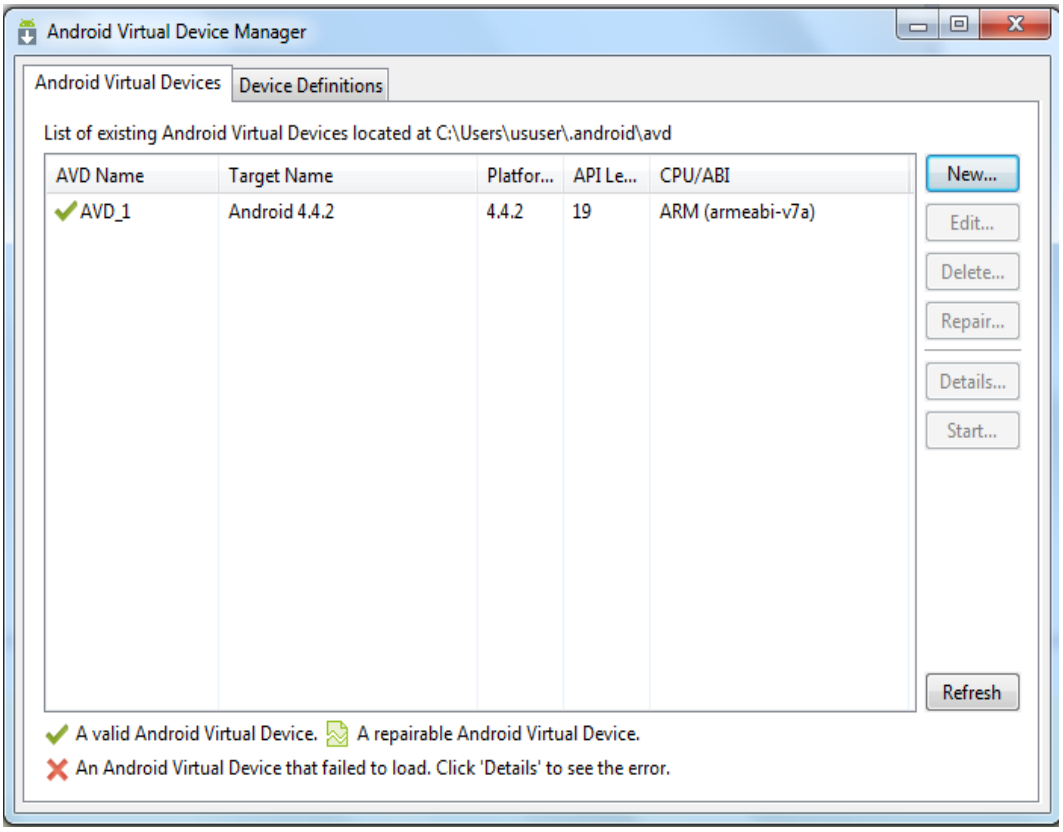


Figura 56 Paso: 8

- Start.

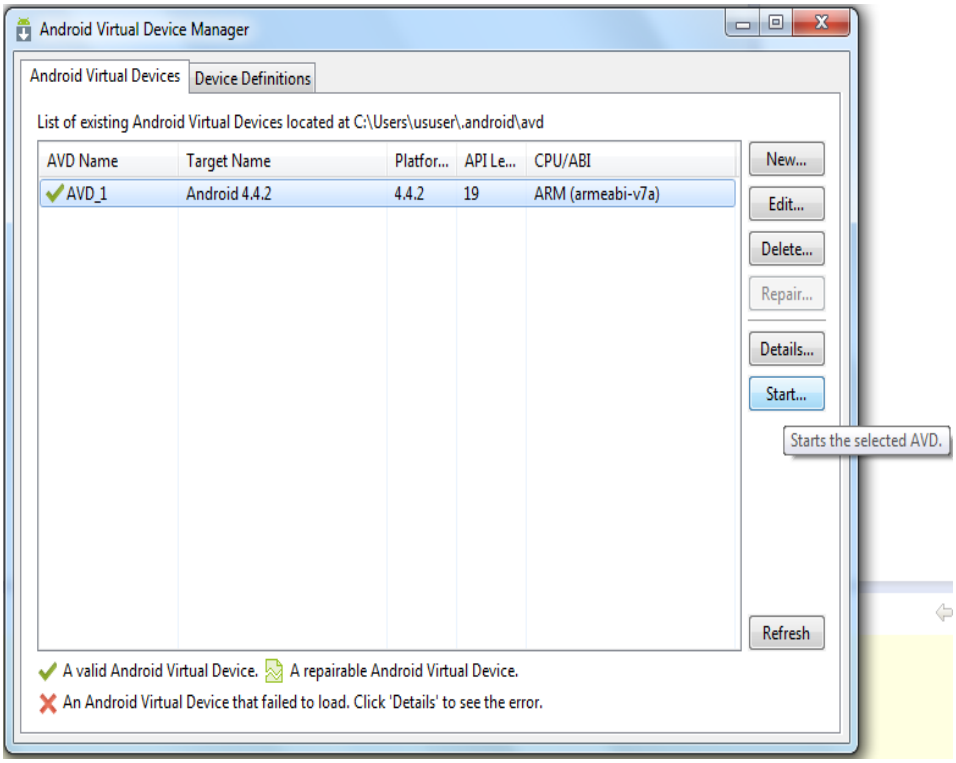


Figura 57 Paso: 9

- Click en la opción Launch Options.
- Launch.

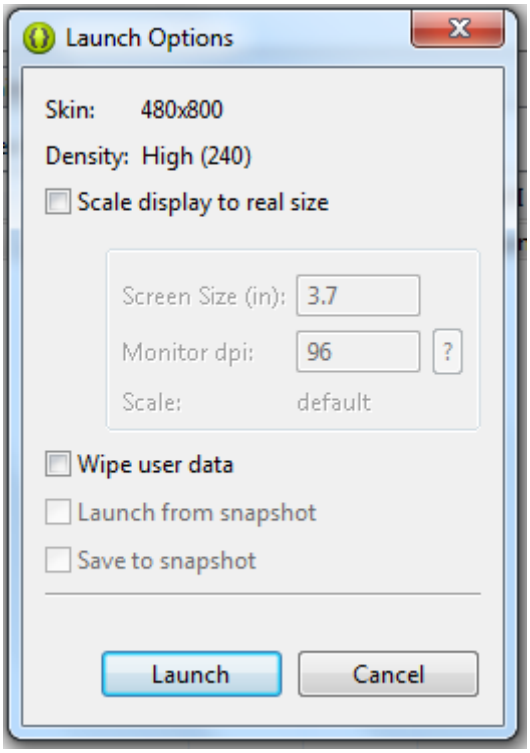


Figura 58 Paso: 10

- Esperamos que se cargue.

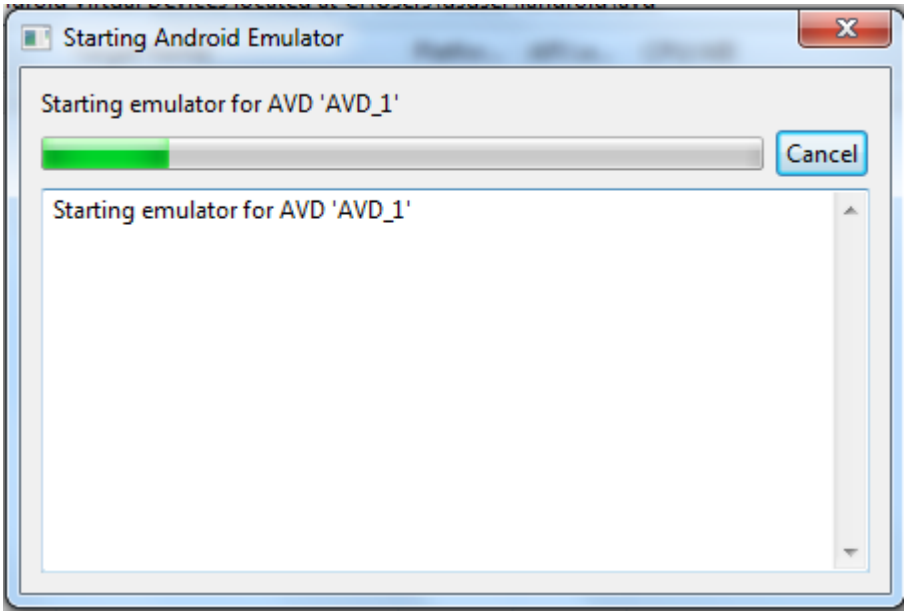


Figura 59 Paso: 11

- Esperar que se inicie el emulador

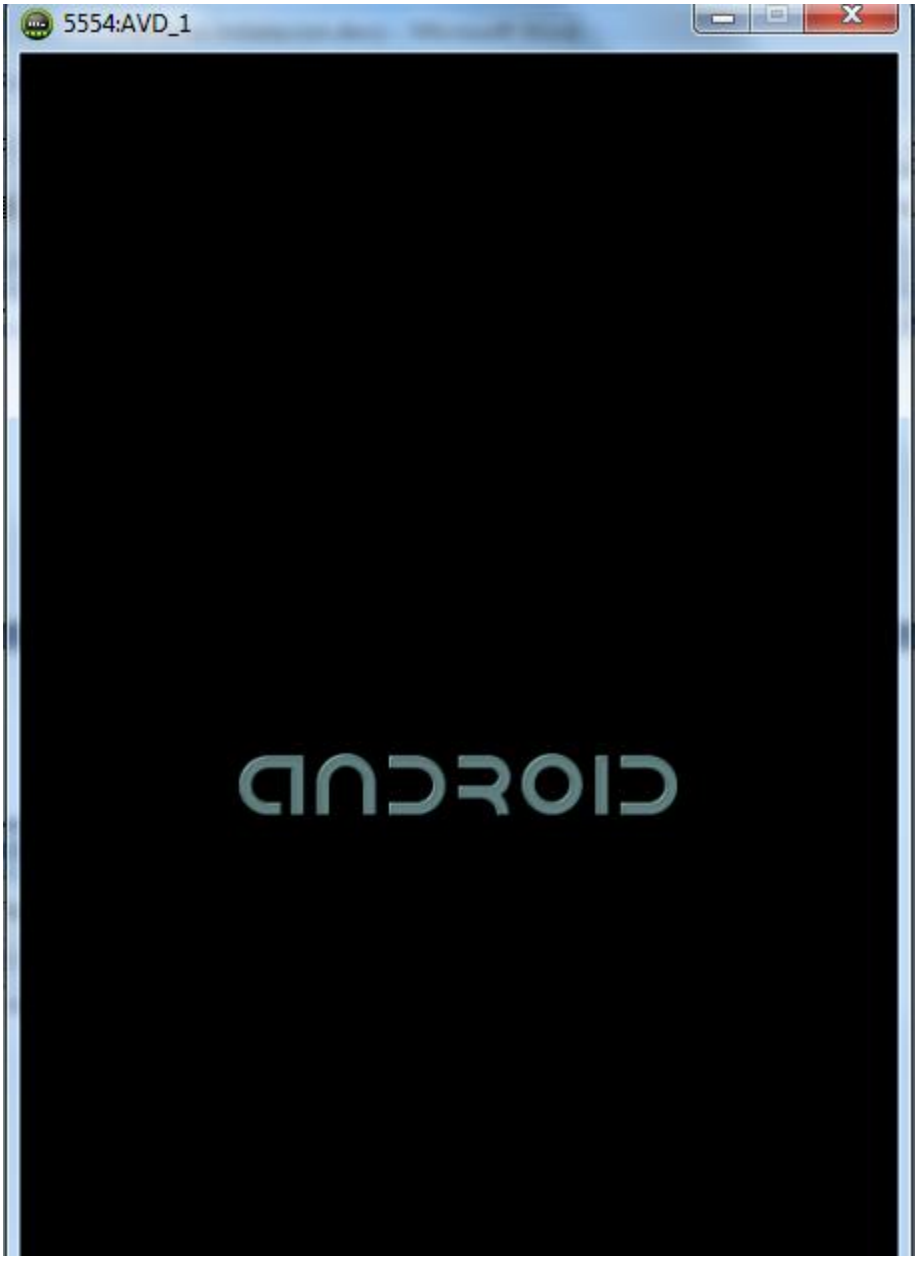


Figura 60 Paso: 12

- Emulador listo.

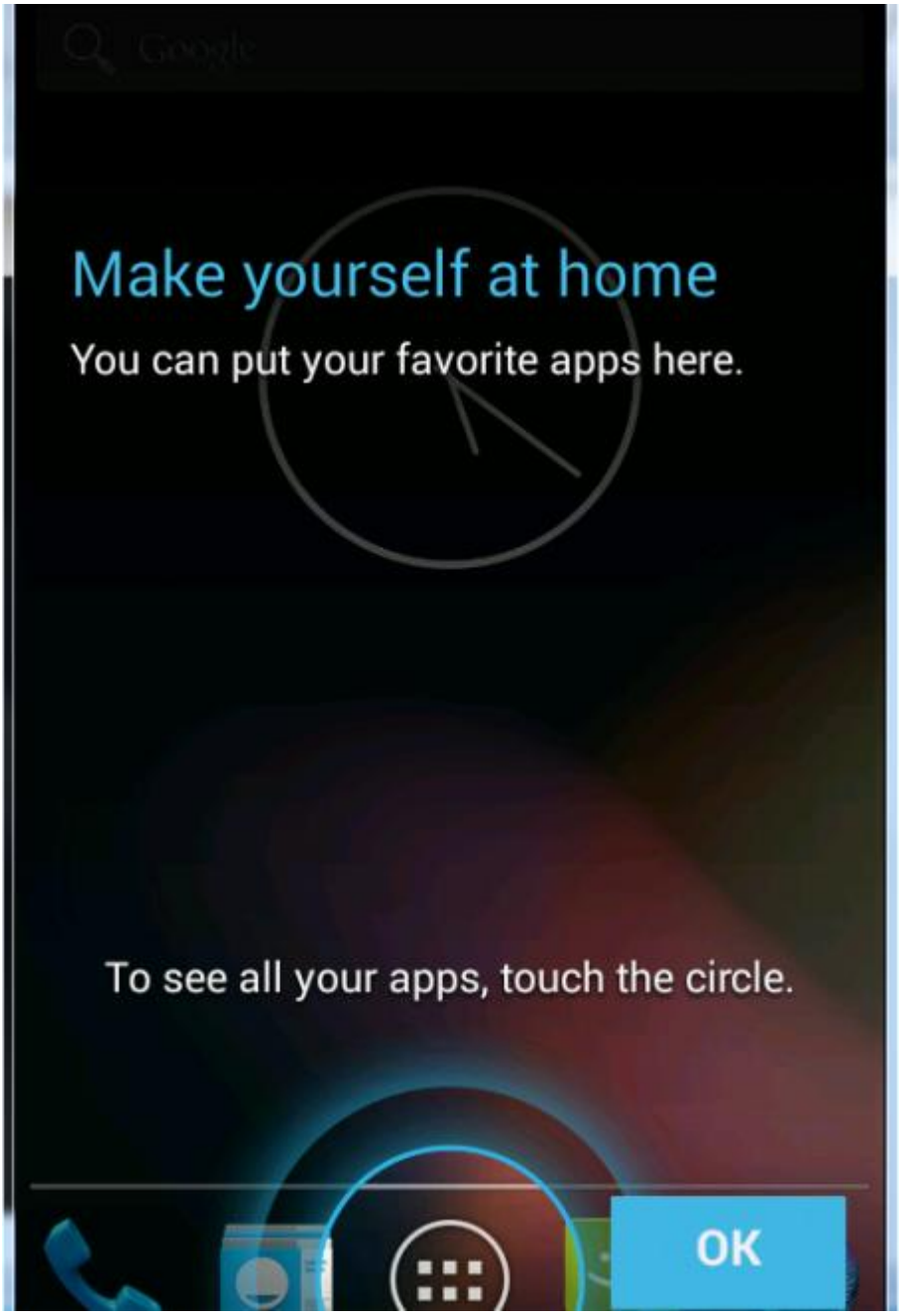


Figura 61 Aplicación lista

Anexo A.11 Script de base de datos del sistema

```
-- --------------------------------------------------------

-- Host:                    localhost

-- Server version:          5.5.28 - MySQL Community Server (GPL)

-- Server OS:               Win32

-- HeidiSQL version:        7.0.0.4053

-- Date/time:               2014-04-01 14:15:39

-- --------------------------------------------------------



/*!40101 SET

@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET NAMES utf8 */;

/*!40014 SET FOREIGN_KEY_CHECKS=0 */;



-- Dumping database structure for controlinventari

CREATE DATABASE IF NOT EXISTS `controlinventari` /*!40100 DEFAULT

CHARACTER SET utf8 */;

USE `controlinventari`;
```

```
-- Dumping structure for table controlinventari.bodega

CREATE TABLE IF NOT EXISTS `bodega` (

  `BodId` varchar(4) NOT NULL DEFAULT '',

  `BodDescripcion` varchar(80) DEFAULT NULL,

  PRIMARY KEY (`BodId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



-- Dumping data for table controlinventari.bodega: ~2 rows (approximately)

/*!40000 ALTER TABLE `bodega` DISABLE KEYS */;

INSERT INTO `bodega` (`BodId`, `BodDescripcion`) VALUES

                                                    ('UIO1', 'Bodega Quito #1'),

                                                    ('UIO2', 'Bodega Quito #2');

/*!40000 ALTER TABLE `bodega` ENABLE KEYS */;

-- Dumping structure for table controlinventari.cliente

CREATE TABLE IF NOT EXISTS `cliente` (

  `CliId` varchar(20) NOT NULL,

  `CliNombres` varchar(100) DEFAULT NULL,
```

```
  `CliApellidos` varchar(100) DEFAULT NULL,
  `CliDireccion` tinytext,
  `CliTelefono` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`CliId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping data for table controlinventari.cliente: ~2 rows (approximately)
/*!40000 ALTER TABLE `cliente` DISABLE KEYS */;
INSERT INTO `cliente` (`CliId`, `CliNombres`, `CliApellidos`, `CliDireccion`,
`CliTelefono`) VALUES
        ('1704429224', 'Henry Esteban', 'Perez Alvarez', '24 de Mayo', '2289555'),
        ('1719364257', 'Mayra Alexandra', 'Ullrich Estrella', 'La Gasca', '2590856');
/*!40000 ALTER TABLE `cliente` ENABLE KEYS */;
-- Dumping structure for table controlinventari.detalle
CREATE TABLE IF NOT EXISTS `detalle` (
  `DetId` bigint(20) NOT NULL DEFAULT '0',
  `InvId` bigint(20) DEFAULT NULL,
  `ProId` varchar(6) DEFAULT NULL,
```

```
  `CantidadToma` int(11) DEFAULT NULL,
  PRIMARY KEY (`DetId`),
  KEY `FK__inventario` (`InvId`),
  CONSTRAINT `FK__inventario` FOREIGN KEY (`InvId`) REFERENCES
`inventario` (`InvId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping data for table controlinventari.detalle: ~0 rows (approximately)
/*!40000 ALTER TABLE `detalle` DISABLE KEYS */;
/*!40000 ALTER TABLE `detalle` ENABLE KEYS */;
-- Dumping structure for table controlinventari.estado
CREATE TABLE IF NOT EXISTS `estado` (
  `EstId` int(11) NOT NULL,
  `EstNombre` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`EstId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping data for table controlinventari.estado: ~3 rows (approximately)
/*!40000 ALTER TABLE `estado` DISABLE KEYS */;
```

```
INSERT INTO `estado` (`EstId`, `EstNombre`) VALUES
                                                    (1, 'Pendiente'),
                                                    (2, 'Confirmado'),
                                                    (3, 'Rechazado');
/*!40000 ALTER TABLE `estado` ENABLE KEYS */;



-- Dumping structure for table controlinventari.inventario
CREATE TABLE IF NOT EXISTS `inventario` (
  `InvId` bigint(20) NOT NULL DEFAULT '0',
  `InvFecha` datetime DEFAULT NULL,
  `InvMotivo` varchar(100) DEFAULT NULL,
  `BodId` varchar(4) DEFAULT NULL,
  PRIMARY KEY (`InvId`),
  KEY `FK_inventario_bodega` (`BodId`),
  CONSTRAINT `FK_inventario_bodega` FOREIGN KEY (`BodId`)
REFERENCES `bodega` (`BodId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- Dumping data for table controlinventari.inventario: ~0 rows (approximately)
/*!40000 ALTER TABLE `inventario` DISABLE KEYS */;
/*!40000 ALTER TABLE `inventario` ENABLE KEYS */;


-- Dumping structure for table controlinventari.movimiento
CREATE TABLE IF NOT EXISTS `movimiento` (
  `MovId` bigint(20) NOT NULL AUTO_INCREMENT,
  `MovFecha` datetime DEFAULT NULL,
  `MovDetalle` varchar(80) DEFAULT NULL,
  `ProId` varchar(6) DEFAULT NULL,
  `MovCantidad` int(11) DEFAULT NULL,
  `MovTipo` varchar(1) DEFAULT NULL COMMENT 'I (Ingresos) / E (Egresos)',
  `UsuUsuario` varchar(20) DEFAULT NULL COMMENT 'I (Ingresos) / E
(Egresos)',
  PRIMARY KEY (`MovId`),
  KEY `FK_movimiento_producto` (`ProId`),
```

```
  KEY `FK_movimiento_usuario` (`UsuUsuario`),

  CONSTRAINT `FK_movimiento_producto` FOREIGN KEY (`ProId`)
REFERENCES `producto` (`ProId`),

  CONSTRAINT `FK_movimiento_usuario` FOREIGN KEY (`UsuUsuario`)
REFERENCES `usuario` (`UsuUsuario`)

) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8;



-- Dumping data for table controlinventari.movimiento: ~24 rows (approximately)

/*!40000 ALTER TABLE `movimiento` DISABLE KEYS */;

INSERT INTO `movimiento` (`MovId`, `MovFecha`, `MovDetalle`, `ProId`,
`MovCantidad`, `MovTipo`, `UsuUsuario`) VALUES

        (1, '2014-03-15 00:00:00', 'INGRESO DE MERCADERIA', 'PRO002', 10, 'I',
'scajamar'),

        (2, '2014-03-15 00:00:00', 'INGRESO DE MERCADERIA', 'PRO001', 20, 'I',
'scajamar'),

         (3, '2014-03-15 00:00:00', 'INGRESO DE MERCADERIA', 'PRO002', 2, 'I',
'scajamar'),

         (4, '2014-03-15 00:00:00', 'INGRESO DE MERCADERIA', 'PRO001', 3, 'E',
'scajamar'),
```

(5, '2014-03-15 20:41:22', 'INGRESO DE MERCADERIA', 'PRO002', 1, 'E', 'scajamar'),

(6, '2014-03-15 20:44:05', 'INGRESO DE MERCADERIA', 'PRO001', 1, 'E', 'scajamar'),

(7, '2014-03-15 20:48:12', 'EGRESO DE MERCADERIA', 'PRO001', 1, 'E', 'scajamar'),

(8, '2014-03-15 21:48:00', 'EGRESO DE MERCADERIA', 'PRO002', 3, 'E', 'scajamar'),

(9, '2014-03-17 15:15:32', 'INGRESO DE MERCADERIA', 'PRO002', 3, 'I', 'scajamar'),

(10, '2014-03-17 20:23:42', 'INGRESO DE MERCADERIA', 'PRO001', 5, 'I', 'scajamar'),

(11, '2014-03-17 20:24:03', 'EGRESO DE MERCADERIA', 'PRO001', 2, 'E', 'scajamar'),

(12, '2014-03-17 20:39:23', 'INGRESO DE MERCADERIA', 'PRO002', 2, 'I', 'scajamar'),

(13, '2014-03-17 16:20:37', 'INGRESO DE MERCADERIA', 'PRO002', 6, 'I', 'scajamar'),

(14, '2014-03-17 16:21:38', 'EGRESO DE MERCADERIA', 'PRO002', 8, 'E', 'scajamar'),

(15, '2014-03-19 13:47:48', 'EGRESO DE MERCADERIA', 'PRO001', 10, 'E', 'scajamar'),

(16, '2014-03-19 14:07:45', 'INGRESO DE MERCADERIA', 'PRO001', 25, 'I', 'scajamar'),

(17, '2014-03-19 14:09:38', 'INGRESO DE MERCADERIA', 'PRO002', 15, 'I', 'scajamar'),

(18, '2014-03-19 14:10:51', 'INGRESO DE MERCADERIA', 'PRO002', 30, 'I', 'scajamar'),

(19, '2014-03-19 15:55:27', 'EGRESO DE MERCADERIA', 'PRO002', 2, 'E', 'scajamar'),

(20, '2014-03-19 15:55:50', 'INGRESO DE MERCADERIA', 'PRO002', 2, 'I', 'scajamar'),

(21, '2014-03-19 19:17:36', 'EGRESO DE MERCADERIA', 'PRO002', 2, 'E', 'scajamar'),

(22, '2014-03-20 11:42:45', 'INGRESO DE MERCADERIA', 'PRO002', 5, 'I', 'scajamar'),

(23, '2014-03-20 11:43:14', 'EGRESO DE MERCADERIA', 'PRO002', 5, 'E', 'scajamar'),

(24, '2014-04-01 18:58:07', 'INGRESO DE MERCADERIA', 'PRO002', 5, 'I', 'scajamar');

```
/*!40000 ALTER TABLE `movimiento` ENABLE KEYS */;

-- Dumping structure for table controlinventari.pedido

CREATE TABLE IF NOT EXISTS `pedido` (

  `PedId` bigint(20) NOT NULL AUTO_INCREMENT,

  `EstId` int(11) NOT NULL,

  `PedFecha` datetime DEFAULT NULL,

  `PedNota` tinytext NOT NULL,

  `CliId` varchar(20) NOT NULL,

  `PedTipo` varchar(1) DEFAULT NULL COMMENT 'V.-Vendedor / B.-Bodega ',

  `UsuUsuario` varchar(20) DEFAULT NULL,

  PRIMARY KEY (`PedId`),

  KEY `FK_pedido_cliente` (`CliId`),

  KEY `FK_pedido_estado` (`EstId`),

  KEY `FK_pedido_usuario` (`UsuUsuario`),

  CONSTRAINT `FK_pedido_cliente` FOREIGN KEY (`CliId`) REFERENCES
`cliente` (`CliId`),

  CONSTRAINT `FK_pedido_estado` FOREIGN KEY (`EstId`) REFERENCES
`estado` (`EstId`),
```

```
  CONSTRAINT `FK_pedido_usuario` FOREIGN KEY (`UsuUsuario`)
REFERENCES `usuario` (`UsuUsuario`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;


-- Dumping data for table controlinventari.pedido: ~0 rows (approximately)
/*!40000 ALTER TABLE `pedido` DISABLE KEYS */;
/*!40000 ALTER TABLE `pedido` ENABLE KEYS */;


-- Dumping structure for table controlinventari.pedido_detalle
CREATE TABLE IF NOT EXISTS `pedido_detalle` (
  `PdeId` bigint(20) NOT NULL AUTO_INCREMENT,
  `PedId` bigint(20) NOT NULL,
  `ProId` varchar(6) DEFAULT NULL,
  `PdeNombre` varchar(100) DEFAULT NULL,
  `PdeCantidad` int(11) DEFAULT NULL,
  `PdePrecio` decimal(12,2) DEFAULT NULL,
  PRIMARY KEY (`PdeId`),
  KEY `FK_pedido_detalle_pedido` (`PedId`),
```

```
  KEY `FK_pedido_detalle_producto` (`ProId`),

  CONSTRAINT `FK_pedido_detalle_pedido` FOREIGN KEY (`PedId`)
REFERENCES `pedido` (`PedId`),

  CONSTRAINT `FK_pedido_detalle_producto` FOREIGN KEY (`ProId`)
REFERENCES `producto` (`ProId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



-- Dumping data for table controlinventari.pedido_detalle: ~0 rows (approximately)

/*!40000 ALTER TABLE `pedido_detalle` DISABLE KEYS */;

/*!40000 ALTER TABLE `pedido_detalle` ENABLE KEYS */;




-- Dumping structure for table controlinventari.perfil

CREATE TABLE IF NOT EXISTS `perfil` (

  `PerId` int(10) NOT NULL DEFAULT '0',

  `PerNombre` varchar(50) DEFAULT NULL,

  PRIMARY KEY (`PerId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- Dumping data for table controlinventari.perfil: ~2 rows (approximately)
/*!40000 ALTER TABLE `perfil` DISABLE KEYS */;
INSERT INTO `perfil` (`PerId`, `PerNombre`) VALUES
                                                    (1, 'Vendedor'),
                                                    (2, 'Bodeguero');
/*!40000 ALTER TABLE `perfil` ENABLE KEYS */;


-- Dumping structure for table controlinventari.producto
CREATE TABLE IF NOT EXISTS `producto` (
  `ProId` varchar(6) NOT NULL DEFAULT '',
  `ProBarra` varchar(50) DEFAULT NULL,
  `ProNombre` varchar(100) DEFAULT NULL,
  `ProPrecio` decimal(14,2) DEFAULT NULL,
  `UbiId` varchar(4) DEFAULT NULL,
  `BodId` varchar(4) DEFAULT NULL,
```

```
  PRIMARY KEY (`ProId`),

  KEY `FK_producto_ubicacion` (`UbiId`),

  KEY `FK_producto_bodega` (`BodId`),

  CONSTRAINT `FK_producto_bodega` FOREIGN KEY (`BodId`) REFERENCES
`bodega` (`BodId`),

  CONSTRAINT `FK_producto_ubicacion` FOREIGN KEY (`UbiId`)
REFERENCES `ubicacion` (`UbiId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



-- Dumping data for table controlinventari.producto: ~2 rows (approximately)

/*!40000 ALTER TABLE `producto` DISABLE KEYS */;

INSERT INTO `producto` (`ProId`, `ProBarra`, `ProNombre`, `ProPrecio`, `UbiId`,
`BodId`) VALUES

                                ('PRO001', '', 'RADIO AM/FM', 120.00, 'P001', 'UIO1'),

                            ('PRO002', '7861005620029', 'LCD 32"', 654.89, 'P001', 'UIO1');

/*!40000 ALTER TABLE `producto` ENABLE KEYS */;



-- Dumping structure for table controlinventari.ubicacion
```

```
CREATE TABLE IF NOT EXISTS `ubicacion` (
  `UbiId` varchar(4) NOT NULL DEFAULT '0',
  `UbiDescripcion` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`UbiId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Dumping data for table controlinventari.ubicacion: ~1 rows (approximately)
/*!40000 ALTER TABLE `ubicacion` DISABLE KEYS */;
INSERT INTO `ubicacion` (`UbiId`, `UbiDescripcion`) VALUES
                                        ('P001', 'PERCHA Nro.1');
/*!40000 ALTER TABLE `ubicacion` ENABLE KEYS */;

-- Dumping structure for table controlinventari.usuario
CREATE TABLE IF NOT EXISTS `usuario` (
  `UsuNombres` varchar(80) DEFAULT NULL,
  `UsuApellidos` varchar(80) DEFAULT NULL,
  `UsuDireccion` varchar(100) DEFAULT NULL,
```

```
  `UsuTelefono` varchar(20) DEFAULT NULL,

  `UsuCelular` varchar(20) DEFAULT NULL,

  `UsuUsuario` varchar(20) NOT NULL DEFAULT '',

  `UsuContrasena` varchar(100) DEFAULT NULL,

  `UsuHabilitado` bit(1) DEFAULT NULL,

  `PerId` int(11) DEFAULT NULL,

  PRIMARY KEY (`UsuUsuario`),

  KEY `FK_usuario_perfil` (`PerId`),

  CONSTRAINT `FK_usuario_perfil` FOREIGN KEY (`PerId`) REFERENCES
`perfil` (`PerId`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



-- Dumping data for table controlinventari.usuario: ~1 rows (approximately)

/*!40000 ALTER TABLE `usuario` DISABLE KEYS */;

INSERT INTO `usuario` (`UsuNombres`, `UsuApellidos`, `UsuDireccion`,
`UsuTelefono`, `UsuCelular`, `UsuUsuario`, `UsuContrasena`, `UsuHabilitado`,
`PerId`) VALUES
```

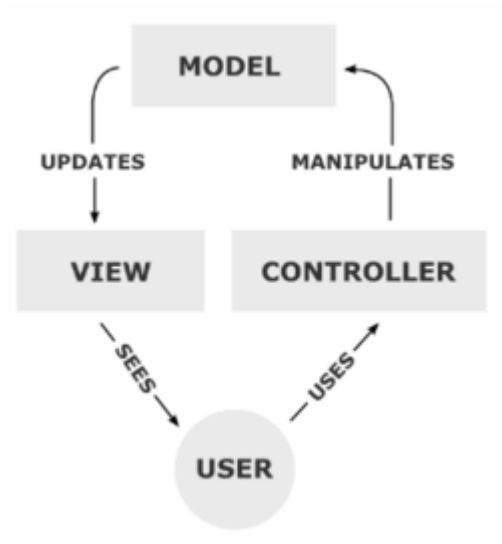
('Samuel Darwin', 'Cajamarca Avila', 'La Florida', '2289555', '0998947116', 'scajamar', '123', b'10000000', 2);

/*!40000 ALTER TABLE `usuario` ENABLE KEYS */;

/*!40014 SET FOREIGN_KEY_CHECKS=1 */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

Anexo A.12 Manual Tecnico



El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que se ha utizado para el desarrollo de este proyecto, a mas de la herramientas antes ya (Android Developer Tools, El gestor de la base de datos MySQL, y el Id de desarrollo Eclipse) mencionadas.

El patrón de arquitectura MVC separa los datos y la lógica de negociode una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento

CONTROLADOR

AdminCliente

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import ec.edu.itsco.sivade.modelo.Cliente;

public class AdmCliente {

	public static ArrayList<Cliente> getClientesArrayList(){
		Connection con=null;

		ArrayList<Cliente> clientes = new ArrayList<Cliente>();

		try {
			Class.forName("com.mysql.jdbc.Driver");

			con = (Connection) DriverManager.getConnection( Bdd.Url,
Bdd.User, Bdd.Pwd);
			PreparedStatement st=con.prepareStatement("select C.* from
cliente C");
			ResultSet rs=st.executeQuery();

			Cliente tmpCliente = null;

			while(rs.next()){
				tmpCliente = new Cliente();
				tmpCliente.setId(rs.getString("CliId"));
				tmpCliente.setNombres(rs.getString("CliNombres"));
				tmpCliente.setApellidos(rs.getString("CliApellidos"));
				tmpCliente.setDireccion(rs.getString("CliDireccion"));
				tmpCliente.setTelefono(rs.getString("CliTelefono"));
				clientes.add(tmpCliente);
			}
			con.close();

		} catch (Exception e) {
			System.out.println(e.getMessage());
		}

		return clientes;
	}
```

```
}
```

AdminEstado

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import ec.edu.itsco.sivade.modelo.Estado;

public class AdmEstado {

	public static ArrayList<Estado> getEstadosArrayList(){
		Connection con=null;
		ArrayList<Estado> estados = new ArrayList<Estado>();

		try {
			Class.forName("com.mysql.jdbc.Driver");
			String url = Bdd.Url;
			con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
			PreparedStatement st=con.prepareStatement("select E.* from
estado E");
			ResultSet rs=st.executeQuery();

			Estado tmpEstado = null;

			while(rs.next()){
				tmpEstado = new Estado();
				tmpEstado.setId(rs.getInt("EstId"));
				tmpEstado.setNombre(rs.getString("EstNombre"));
				estados.add(tmpEstado);
			}
			con.close();

		} catch (Exception e) {
			System.out.println(e.getMessage());
		}finally{
		}
		return estados;
	}
}
```

AdminInventario

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import ec.edu.itsco.sivade.modelo.Estado;

public class AdmEstado {

	public static ArrayList<Estado> getEstadosArrayList(){
		Connection con=null;
		ArrayList<Estado> estados = new ArrayList<Estado>();

		try {
			Class.forName("com.mysql.jdbc.Driver");
			String url = Bdd.Url;
			con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
			PreparedStatement st=con.prepareStatement("select E.* from
estado E");
			ResultSet rs=st.executeQuery();

			Estado tmpEstado = null;

			while(rs.next()){
				tmpEstado = new Estado();
				tmpEstado.setId(rs.getInt("EstId"));
				tmpEstado.setNombre(rs.getString("EstNombre"));
				estados.add(tmpEstado);
			}
			con.close();

		} catch (Exception e) {
			System.out.println(e.getMessage());
		}finally{
		}
		return estados;
	}
}
```

AdminPedido

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import ec.edu.itsco.sivade.modelo.Cliente;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Pedido;
import ec.edu.itsco.sivade.modelo.PedidoDetalle;

public class AdmPedido {

	public static boolean guardarPedido(Pedido nuevoPedido){
		Connection con=null;
		PreparedStatement psPedido = null;
		PreparedStatement psPedidoDetalle = null;
		PreparedStatement psUpdateSecuencia = null;

		try
		{
			long numeroPedido = getNumeroPedido();

			//Class.forName("com.mysql.jdbc.Driver");
			//String url = Bdd.Url;
			//con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
			con =Bdd.getConnection();
			con.setAutoCommit(false);

			//CABECERA
			String insertaPedidoSql ="insert into
pedido(PedId,EstId,PedFecha,PedNota, CliId, PedTipo, UsuUsuario) "+


"values(?,?,?,?,?,?,?)";
			SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
			String fechaPedido = sdf.format(nuevoPedido.getFecha());

			psPedido=con.prepareStatement(insertaPedidoSql);
			psPedido.setLong(1, numeroPedido);
			psPedido.setInt(2, nuevoPedido.getEstado().getId());
			psPedido.setString(3, fechaPedido);
```

```
                    psPedido.setString(4, nuevoPedido.getNota());
                    psPedido.setString(5, nuevoPedido.getCliente().getId());
                    psPedido.setString(6, nuevoPedido.getTipo());
                    psPedido.setString(7, nuevoPedido.getUsuario());
                    psPedido.executeUpdate();

                    String insertaPedidoDetalleSql ="insert into
pedido_detalle(PedId,ProId,PdeNombre,PdeCantidad,PdePrecio) "+

        "values(?,?,?,?,?)";
                    //DETALLE
                    for (PedidoDetalle detalle : nuevoPedido.getDetalle()) {


        psPedidoDetalle=con.prepareStatement(insertaPedidoDetalleSql);
                            psPedidoDetalle.setLong(1, numeroPedido);
                            psPedidoDetalle.setString(2, detalle.getProId());
                            psPedidoDetalle.setString(3, detalle.getProNombre());
                            psPedidoDetalle.setInt(4, detalle.getCantidad());
                            psPedidoDetalle.setDouble(5, detalle.getPrecio());
                            psPedidoDetalle.executeUpdate();
                    }

                    String updateSecuenciaSql ="update secuencia set
NumeroPedido=NumeroPedido+1";
                    psUpdateSecuencia=con.prepareStatement(updateSecuenciaSql);
                    psUpdateSecuencia.executeUpdate();

                    con.commit();

                    return true;
                }catch(Exception ex){
                    return false;
                }finally{
                    if (con != null) {
                        try {
                            con.close();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                }
        }

        public static boolean actualizarPedido(Pedido pedido){
            Connection con=null;
            PreparedStatement psPedido = null;
            PreparedStatement psPedidoDetalle = null;
            PreparedStatement psPedidoDetalleElimina = null;
```

```
try
{
	con =Bdd.getConnection();
	con.setAutoCommit(false);

	//CABECERA
	String actualizaPedidoSql ="update pedido set PedNota=?, CliId=?
where PedId=?";

	psPedido=con.prepareStatement(actualizaPedidoSql);
	psPedido.setString(1, pedido.getNota());
	psPedido.setString(2, pedido.getCliente().getId());
	psPedido.setLong(3, pedido.getId());
	psPedido.executeUpdate();

	String eliminaDetalleSql ="delete from pedido_detalle where
PedId=?";
	psPedidoDetalleElimina=con.prepareStatement(eliminaDetalleSql);
	psPedidoDetalleElimina.setLong(1, pedido.getId());

	psPedidoDetalleElimina.executeUpdate();

	String insertaPedidoDetalleSql ="insert into
pedido_detalle(PedId,ProId,PdeNombre,PdeCantidad,PdePrecio) "+

"values(?,?,?,?,?)";
	//DETALLE
	for (PedidoDetalle detalle : pedido.getDetalle()) {


psPedidoDetalle=con.prepareStatement(insertaPedidoDetalleSql);
		psPedidoDetalle.setLong(1, pedido.getId());
		psPedidoDetalle.setString(2, detalle.getProId());
		psPedidoDetalle.setString(3, detalle.getProNombre());
		psPedidoDetalle.setInt(4, detalle.getCantidad());
		psPedidoDetalle.setDouble(5, detalle.getPrecio());
		psPedidoDetalle.executeUpdate();
	}

	con.commit();

	return true;
}catch(Exception ex){
	return false;
}finally{
	if (con != null) {
		try {
			con.close();
		} catch (SQLException e) {
```

```
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                }
        }
        public static long getNumeroPedido(){
                Connection con=null;
                PreparedStatement psNumeroPedido = null;
                String numeroPedidoSql ="select NumeroPedido from secuencia";
                long numeroPedido = 0;
                try {
                        con =Bdd.getConnection();
                        psNumeroPedido=con.prepareStatement(numeroPedidoSql);
                        ResultSet rs=psNumeroPedido.executeQuery();
                        if(rs.next())
                                numeroPedido = rs.getLong(1);

                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }finally{
                        if (con != null) {
                                try {
                                        con.close();
                                } catch (SQLException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                }

                return numeroPedido;
        }


        //ELIMINAR
        public static boolean eliminarPedido(long numeroPedido){
                Connection con=null;
                PreparedStatement psDetalle = null;
                PreparedStatement psPedido = null;

                try
                {
                        //Class.forName("com.mysql.jdbc.Driver");
                        //String url = Bdd.Url;
                        //con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
                        con =Bdd.getConnection();
                        con.setAutoCommit(false);
```

```
                    //ELIMINA DETALLE
                    String eliminaDetalleSql ="delete from pedido_detalle where
PedId=?";

                    psDetalle=con.prepareStatement(eliminaDetalleSql);
                    psDetalle.setLong(1, numeroPedido);
                    psDetalle.executeUpdate();

                    //ELIMINA PEDIDO
                    String eliminaPedidoSql ="delete from pedido where PedId=?";
                    psPedido=con.prepareStatement(eliminaPedidoSql);
                    psPedido.setLong(1, numeroPedido);
                    psPedido.executeUpdate();

                    con.commit();

                    return true;
                }catch(Exception ex){
                    return false;
                }finally{
                    if (con != null) {
                        try {
                            con.close();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                }
            }

            //CAMBIAR ESTADO DEL PEDIDO
            public static boolean cambiarEstadoPedido(long numeroPedido, int estado){
                Connection con=null;
                PreparedStatement psPedido = null;

                try
                {
                    //Class.forName("com.mysql.jdbc.Driver");
                    //String url = Bdd.Url;
                    //con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
                    con =Bdd.getConnection();
                    con.setAutoCommit(false);

                    String cambiaEstadoSql ="update pedido set EstId=? where
PedId=?";

                    psPedido=con.prepareStatement(cambiaEstadoSql);
```

```
                psPedido.setInt(1, estado);
                psPedido.setLong(2, numeroPedido);
                psPedido.executeUpdate();

                con.commit();

                return true;

        }catch(Exception ex){
                return false;
        }finally{
                if (con != null) {
                        try {
                                con.close();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }
}


public static ArrayList<Pedido> getPedidos(String usuario){
        Connection con=null;

        ArrayList<Pedido> lstPedidos = new ArrayList<Pedido>();

        try {
                //Class.forName("com.mysql.jdbc.Driver");
                //String url = Bdd.Url;
                con =Bdd.getConnection(); //DriverManager.getConnection( url,
Bdd.User, Bdd.Pwd);

                String sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and C.CliId=P.CliId order by P.PedId DESC";
                if(!usuario.isEmpty()){
                        sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and C.CliId=P.CliId and P.UsuUsuario='"+usuario+"' order by P.PedId
DESC";
                }

                PreparedStatement st=con.prepareStatement(sql);

                ResultSet rs=st.executeQuery();

                Pedido pedido = null;
                Estado estado=null;
                Cliente cliente=null;
                while(rs.next()){
```

```
                                pedido = new Pedido();
                                pedido.setId(rs.getLong("PedId"));
                                pedido.setFecha(rs.getDate("PedFecha"));
                                pedido.setUsuario(rs.getString("UsuUsuario"));
                                pedido.setNota(rs.getString("PedNota"));
                                estado=new Estado();
                                estado.setId(rs.getInt("EstId"));
                                estado.setNombre(rs.getString("EstNombre"));
                                pedido.setEstado(estado);
                                cliente=new Cliente();
                                cliente.setId(rs.getString("CliId"));
                                cliente.setNombres(rs.getString("CliNombres"));
                                cliente.setApellidos(rs.getString("CliApellidos"));
                                pedido.setCliente(cliente);

                                //DETALLE
                                String sqlDetalle ="select PD.* from pedido_detalle PD
where PD.PedId="+rs.getLong("PedId")+"";
                                PreparedStatement
stDetalle=con.prepareStatement(sqlDetalle);
                                ResultSet rsDetalle=stDetalle.executeQuery();
                                ArrayList<PedidoDetalle> lstDetalle = new
ArrayList<PedidoDetalle>();
                                PedidoDetalle pedidoDetalle = null;

                                while(rsDetalle.next()){
                                        pedidoDetalle = new PedidoDetalle();
                                        pedidoDetalle.setId(rsDetalle.getLong("PdeId"));

                                        //pedido.setId(rsDetalle.getLong("PedId"));

                                        pedidoDetalle.setPedido(pedido);


        pedidoDetalle.setProId(rsDetalle.getString("ProId"));

        pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

        pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));

        pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                                        lstDetalle.add(pedidoDetalle);


                                }
                                //
                                pedido.setDetalle(lstDetalle);
                                lstPedidos.add(pedido);

                        }
```

```
                con.close();

            } catch (Exception e) {
                System.out.println(e.getMessage());
            }

            return lstPedidos;
        }
        //
        public static ArrayList<PedidoDetalle> getPedidoDetalle(long numeroPedido){
            Connection con=null;

            ArrayList<PedidoDetalle> lstDetalle = new ArrayList<PedidoDetalle>();

            try {
                con = Bdd.getConnection();

                String sqlDetalle ="select PD.* from pedido_detalle PD where
PD.PedId="+numeroPedido+"";
                PreparedStatement stDetalle=con.prepareStatement(sqlDetalle);
                ResultSet rsDetalle=stDetalle.executeQuery();

                PedidoDetalle pedidoDetalle = null;

                while(rsDetalle.next()){
                    pedidoDetalle = new PedidoDetalle();
                    pedidoDetalle.setId(rsDetalle.getLong("PdeId"));


                    pedidoDetalle.setProId(rsDetalle.getString("ProId"));

        pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

        pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));
                    pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                    lstDetalle.add(pedidoDetalle);

                }


                con.close();

            } catch (Exception e) {
                System.out.println(e.getMessage());
            }

            return lstDetalle;
        }
        //GET PEDIDOS PENDIENTES
        public static ArrayList<Pedido> getPedidosPendientes(){
```

```
Connection con=null;

ArrayList<Pedido> lstPedidos = new ArrayList<Pedido>();

try {

        con =Bdd.getConnection();


        String sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and P.EstId=1 and C.CliId=P.CliId order by P.PedId DESC";


        PreparedStatement st=con.prepareStatement(sql);

        ResultSet rs=st.executeQuery();

        Pedido pedido = null;
        Estado estado=null;
        Cliente cliente=null;
        while(rs.next()){
                pedido = new Pedido();
                pedido.setId(rs.getLong("PedId"));
                pedido.setFecha(rs.getDate("PedFecha"));
                pedido.setUsuario(rs.getString("UsuUsuario"));
                pedido.setNota(rs.getString("PedNota"));
                estado=new Estado();
                estado.setId(rs.getInt("EstId"));
                estado.setNombre(rs.getString("EstNombre"));
                pedido.setEstado(estado);
                cliente=new Cliente();
                cliente.setId(rs.getString("CliId"));
                cliente.setNombres(rs.getString("CliNombres"));
                cliente.setApellidos(rs.getString("CliApellidos"));
                pedido.setCliente(cliente);

                //DETALLE
                String sqlDetalle ="select PD.* from pedido_detalle PD
where PD.PedId="+rs.getLong("PedId")+"";
                PreparedStatement
stDetalle=con.prepareStatement(sqlDetalle);
                ResultSet rsDetalle=stDetalle.executeQuery();
                ArrayList<PedidoDetalle> lstDetalle = new
ArrayList<PedidoDetalle>();
                PedidoDetalle pedidoDetalle = null;

                while(rsDetalle.next()){
                        pedidoDetalle = new PedidoDetalle();
                        pedidoDetalle.setId(rsDetalle.getLong("PdeId"));
```

```
                                        //pedido.setId(rsDetalle.getLong("PedId"));

                                        pedidoDetalle.setPedido(pedido);


            pedidoDetalle.setProId(rsDetalle.getString("ProId"));

            pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

            pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));

            pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                                        lstDetalle.add(pedidoDetalle);


                                    }
                                    //
                                    pedido.setDetalle(lstDetalle);
                                    lstPedidos.add(pedido);

                                }
                                con.close();

                        } catch (Exception e) {
                                System.out.println(e.getMessage());
                        }

                        return lstPedidos;
                }

}
```

AdminProducto

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import ec.edu.itsco.sivade.modelo.Cliente;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Pedido;
```

```
import ec.edu.itsco.sivade.modelo.PedidoDetalle;

public class AdmPedido {

	public static boolean guardarPedido(Pedido nuevoPedido){
		Connection con=null;
		PreparedStatement psPedido = null;
		PreparedStatement psPedidoDetalle = null;
		PreparedStatement psUpdateSecuencia = null;

		try
		{
			long numeroPedido = getNumeroPedido();

			//Class.forName("com.mysql.jdbc.Driver");
			//String url = Bdd.Url;
			//con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
			con =Bdd.getConnection();
			con.setAutoCommit(false);

			//CABECERA
			String insertaPedidoSql ="insert into
pedido(PedId,EstId,PedFecha,PedNota, CliId, PedTipo, UsuUsuario) "+


"values(?,?,?,?,?,?,?)";
			SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
			String fechaPedido = sdf.format(nuevoPedido.getFecha());

			psPedido=con.prepareStatement(insertaPedidoSql);
			psPedido.setLong(1, numeroPedido);
			psPedido.setInt(2, nuevoPedido.getEstado().getId());
			psPedido.setString(3, fechaPedido);
			psPedido.setString(4, nuevoPedido.getNota());
			psPedido.setString(5, nuevoPedido.getCliente().getId());
			psPedido.setString(6, nuevoPedido.getTipo());
			psPedido.setString(7, nuevoPedido.getUsuario());
			psPedido.executeUpdate();

			String insertaPedidoDetalleSql ="insert into
pedido_detalle(PedId,ProId,PdeNombre,PdeCantidad,PdePrecio) "+

	"values(?,?,?,?,?)";
			//DETALLE
			for (PedidoDetalle detalle : nuevoPedido.getDetalle()) {


	psPedidoDetalle=con.prepareStatement(insertaPedidoDetalleSql);
```

```
                        psPedidoDetalle.setLong(1, numeroPedido);
                        psPedidoDetalle.setString(2, detalle.getProId());
                        psPedidoDetalle.setString(3, detalle.getProNombre());
                        psPedidoDetalle.setInt(4, detalle.getCantidad());
                        psPedidoDetalle.setDouble(5, detalle.getPrecio());
                        psPedidoDetalle.executeUpdate();
                }

                String updateSecuenciaSql ="update secuencia set
NumeroPedido=NumeroPedido+1";
                psUpdateSecuencia=con.prepareStatement(updateSecuenciaSql);
                psUpdateSecuencia.executeUpdate();

                con.commit();

                return true;
        }catch(Exception ex){
                return false;
        }finally{
                if (con != null) {
                        try {
                                con.close();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }
}

public static boolean actualizarPedido(Pedido pedido){
        Connection con=null;
        PreparedStatement psPedido = null;
        PreparedStatement psPedidoDetalle = null;
        PreparedStatement psPedidoDetalleElimina = null;

        try
        {
                con =Bdd.getConnection();
                con.setAutoCommit(false);

                //CABECERA
                String actualizaPedidoSql ="update pedido set PedNota=?, CliId=?
where PedId=?";

                psPedido=con.prepareStatement(actualizaPedidoSql);
                psPedido.setString(1, pedido.getNota());
                psPedido.setString(2, pedido.getCliente().getId());
                psPedido.setLong(3, pedido.getId());
                psPedido.executeUpdate();
```

```
                    String eliminaDetalleSql ="delete from pedido_detalle where
PedId=?";
                    psPedidoDetalleElimina=con.prepareStatement(eliminaDetalleSql);
                    psPedidoDetalleElimina.setLong(1, pedido.getId());

                    psPedidoDetalleElimina.executeUpdate();

                    String insertaPedidoDetalleSql ="insert into
pedido_detalle(PedId,ProId,PdeNombre,PdeCantidad,PdePrecio) "+

        "values(?,?,?,?,?)";
                    //DETALLE
                    for (PedidoDetalle detalle : pedido.getDetalle()) {


        psPedidoDetalle=con.prepareStatement(insertaPedidoDetalleSql);
                        psPedidoDetalle.setLong(1, pedido.getId());
                        psPedidoDetalle.setString(2, detalle.getProId());
                        psPedidoDetalle.setString(3, detalle.getProNombre());
                        psPedidoDetalle.setInt(4, detalle.getCantidad());
                        psPedidoDetalle.setDouble(5, detalle.getPrecio());
                        psPedidoDetalle.executeUpdate();
                    }

                    con.commit();

                    return true;
                }catch(Exception ex){
                    return false;
                }finally{
                    if (con != null) {
                        try {
                            con.close();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                }
        }
        public static long getNumeroPedido(){
                Connection con=null;
                PreparedStatement psNumeroPedido = null;
                String numeroPedidoSql ="select NumeroPedido from secuencia";
                long numeroPedido = 0;
                try {
                    con =Bdd.getConnection();
                    psNumeroPedido=con.prepareStatement(numeroPedidoSql);
                    ResultSet rs=psNumeroPedido.executeQuery();
```

```
                    if(rs.next())
                        numeroPedido = rs.getLong(1);

                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }finally{
                    if (con != null) {
                        try {
                            con.close();
                        } catch (SQLException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }
                    }
                }

                return numeroPedido;
        }


        //ELIMINAR
        public static boolean eliminarPedido(long numeroPedido){
            Connection con=null;
            PreparedStatement psDetalle = null;
            PreparedStatement psPedido = null;

            try
            {
                //Class.forName("com.mysql.jdbc.Driver");
                //String url = Bdd.Url;
                //con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
                con =Bdd.getConnection();
                con.setAutoCommit(false);

                //ELIMINA DETALLE
                String eliminaDetalleSql ="delete from pedido_detalle where
PedId=?";

                psDetalle=con.prepareStatement(eliminaDetalleSql);
                psDetalle.setLong(1, numeroPedido);
                psDetalle.executeUpdate();

                //ELIMINA PEDIDO
                String eliminaPedidoSql ="delete from pedido where PedId=?";
                psPedido=con.prepareStatement(eliminaPedidoSql);
                psPedido.setLong(1, numeroPedido);
                psPedido.executeUpdate();
```

```
                con.commit();

                return true;
            }catch(Exception ex){
                return false;
            }finally{
                if (con != null) {
                    try {
                        con.close();
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        }

        //CAMBIAR ESTADO DEL PEDIDO
        public static boolean cambiarEstadoPedido(long numeroPedido, int estado){
            Connection con=null;
            PreparedStatement psPedido = null;

            try
            {
                //Class.forName("com.mysql.jdbc.Driver");
                //String url = Bdd.Url;
                //con = (Connection) DriverManager.getConnection( url, Bdd.User,
Bdd.Pwd);
                con =Bdd.getConnection();
                con.setAutoCommit(false);

                String cambiaEstadoSql ="update pedido set EstId=? where
PedId=?";

                psPedido=con.prepareStatement(cambiaEstadoSql);
                psPedido.setInt(1, estado);
                psPedido.setLong(2, numeroPedido);
                psPedido.executeUpdate();

                con.commit();

                return true;

            }catch(Exception ex){
                return false;
            }finally{
                if (con != null) {
                    try {
                        con.close();
                    } catch (SQLException e) {
```

```
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }
                }
        }


        public static ArrayList<Pedido> getPedidos(String usuario){
                Connection con=null;

                ArrayList<Pedido> lstPedidos = new ArrayList<Pedido>();

                try {
                        //Class.forName("com.mysql.jdbc.Driver");
                        //String url = Bdd.Url;
                        con =Bdd.getConnection(); //DriverManager.getConnection( url,
Bdd.User, Bdd.Pwd);

                        String sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and C.CliId=P.CliId order by P.PedId DESC";
                        if(!usuario.isEmpty()){
                                sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and C.CliId=P.CliId and P.UsuUsuario='"+usuario+"' order by P.PedId
DESC";
                        }

                        PreparedStatement st=con.prepareStatement(sql);

                        ResultSet rs=st.executeQuery();

                        Pedido pedido = null;
                        Estado estado=null;
                        Cliente cliente=null;
                        while(rs.next()){
                                pedido = new Pedido();
                                pedido.setId(rs.getLong("PedId"));
                                pedido.setFecha(rs.getDate("PedFecha"));
                                pedido.setUsuario(rs.getString("UsuUsuario"));
                                pedido.setNota(rs.getString("PedNota"));
                                estado=new Estado();
                                estado.setId(rs.getInt("EstId"));
                                estado.setNombre(rs.getString("EstNombre"));
                                pedido.setEstado(estado);
                                cliente=new Cliente();
                                cliente.setId(rs.getString("CliId"));
                                cliente.setNombres(rs.getString("CliNombres"));
                                cliente.setApellidos(rs.getString("CliApellidos"));
                                pedido.setCliente(cliente);
```

```
                                //DETALLE
                                String sqlDetalle ="select PD.* from pedido_detalle PD
where PD.PedId="+rs.getLong("PedId")+"";
                                PreparedStatement
stDetalle=con.prepareStatement(sqlDetalle);
                                ResultSet rsDetalle=stDetalle.executeQuery();
                                ArrayList<PedidoDetalle> lstDetalle = new
ArrayList<PedidoDetalle>();
                                PedidoDetalle pedidoDetalle = null;

                                while(rsDetalle.next()){
                                        pedidoDetalle = new PedidoDetalle();
                                        pedidoDetalle.setId(rsDetalle.getLong("PdeId"));

                                        //pedido.setId(rsDetalle.getLong("PedId"));

                                        pedidoDetalle.setPedido(pedido);


        pedidoDetalle.setProId(rsDetalle.getString("ProId"));

        pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

        pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));

        pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                                        lstDetalle.add(pedidoDetalle);


                                }
                                //
                                pedido.setDetalle(lstDetalle);
                                lstPedidos.add(pedido);

                        }
                        con.close();

                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }

                return lstPedidos;
        }
        //
        public static ArrayList<PedidoDetalle> getPedidoDetalle(long numeroPedido){
                Connection con=null;

                ArrayList<PedidoDetalle> lstDetalle = new ArrayList<PedidoDetalle>();

                try {
```

```
                    con = Bdd.getConnection();

                    String sqlDetalle ="select PD.* from pedido_detalle PD where
PD.PedId="+numeroPedido+"";
                    PreparedStatement stDetalle=con.prepareStatement(sqlDetalle);
                    ResultSet rsDetalle=stDetalle.executeQuery();

                    PedidoDetalle pedidoDetalle = null;

                    while(rsDetalle.next()){
                        pedidoDetalle = new PedidoDetalle();
                        pedidoDetalle.setId(rsDetalle.getLong("PdeId"));


                        pedidoDetalle.setProId(rsDetalle.getString("ProId"));

        pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

        pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));
                        pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                        lstDetalle.add(pedidoDetalle);

                    }


                    con.close();

                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }

                return lstDetalle;
        }
        //GET PEDIDOS PENDIENTES
        public static ArrayList<Pedido> getPedidosPendientes(){
                Connection con=null;

                ArrayList<Pedido> lstPedidos = new ArrayList<Pedido>();

                try {

                    con =Bdd.getConnection();


                    String sql ="select P.*, E.*, C.* from pedido P,estado E,cliente C
where P.EstId=E.EstId and P.EstId=1 and C.CliId=P.CliId order by P.PedId DESC";


                    PreparedStatement st=con.prepareStatement(sql);
```

```
                    ResultSet rs=st.executeQuery();

                    Pedido pedido = null;
                    Estado estado=null;
                    Cliente cliente=null;
                    while(rs.next()){
                        pedido = new Pedido();
                        pedido.setId(rs.getLong("PedId"));
                        pedido.setFecha(rs.getDate("PedFecha"));
                        pedido.setUsuario(rs.getString("UsuUsuario"));
                        pedido.setNota(rs.getString("PedNota"));
                        estado=new Estado();
                        estado.setId(rs.getInt("EstId"));
                        estado.setNombre(rs.getString("EstNombre"));
                        pedido.setEstado(estado);
                        cliente=new Cliente();
                        cliente.setId(rs.getString("CliId"));
                        cliente.setNombres(rs.getString("CliNombres"));
                        cliente.setApellidos(rs.getString("CliApellidos"));
                        pedido.setCliente(cliente);

                        //DETALLE
                        String sqlDetalle ="select PD.* from pedido_detalle PD
where PD.PedId="+rs.getLong("PedId")+"";
                        PreparedStatement
stDetalle=con.prepareStatement(sqlDetalle);
                        ResultSet rsDetalle=stDetalle.executeQuery();
                        ArrayList<PedidoDetalle> lstDetalle = new
ArrayList<PedidoDetalle>();
                        PedidoDetalle pedidoDetalle = null;

                        while(rsDetalle.next()){
                            pedidoDetalle = new PedidoDetalle();
                            pedidoDetalle.setId(rsDetalle.getLong("PdeId"));

                            //pedido.setId(rsDetalle.getLong("PedId"));

                            pedidoDetalle.setPedido(pedido);


        pedidoDetalle.setProId(rsDetalle.getString("ProId"));

        pedidoDetalle.setProNombre(rsDetalle.getString("PdeNombre"));

        pedidoDetalle.setCantidad(rsDetalle.getInt("PdeCantidad"));

        pedidoDetalle.setPrecio(rsDetalle.getDouble("PdePrecio"));
                            lstDetalle.add(pedidoDetalle);
```

```
                                }
                                //
                                pedido.setDetalle(lstDetalle);
                                lstPedidos.add(pedido);

                        }
                        con.close();

                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }

                return lstPedidos;
        }

}
```

AdminUsuario

```
package ec.edu.itsco.sivade.controlador;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import ec.edu.itsco.sivade.modelo.Usuario;


public class AdmUsuario {

        public Usuario iniciarSesion(String strUsuario, String strPwd){
                Connection con=null;
                Usuario usuario = null;
                try {
                        con =Bdd.getConnection();
                        PreparedStatement st=con.prepareStatement("select U.*,
P.PerNombre from usuario U, perfil P where U.UsuUsuario=? and U.usuContrasena=? and
P.PerId=U.PerId");
                        st.setString(1, strUsuario);
                        st.setString(2, strPwd);
                        ResultSet rs=st.executeQuery();

                        if(rs.next()){
                                usuario = new Usuario();
                                usuario.setNombres(rs.getString("UsuNombres"));
                                usuario.setApellidos(rs.getString("UsuApellidos"));
                                usuario.setDireccion(rs.getString("UsuDireccion"));
                                usuario.setTelefono(rs.getString("UsuTelefono"));
                                usuario.setCelular(rs.getString("UsuCelular"));
```

```
                                usuario.setUsuario(rs.getString("UsuUsuario"));
                                usuario.setContrasena(rs.getString("UsuContrasena"));
                                usuario.setHabilitado(rs.getBoolean("UsuHabilitado"));
                                usuario.setPerfilId(rs.getInt("PerId"));
                                usuario.setPerfilNombre(rs.getString("PerNombre"));
                        }
                        con.close();

                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }finally{
                }

                return usuario;
        }
}
```

VISTA

Loguin

```
package ec.edu.itsco.sivade.vista;



import java.util.ArrayList;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmEstado;
import ec.edu.itsco.sivade.controlador.AdmProducto;
import ec.edu.itsco.sivade.controlador.AdmUsuario;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;

public class LoginActivity extends Activity {

        //VARIABLES GLOBALES
        ArrayList<Producto> listaProductos = null;
```

```
        ArrayList<Estado> listaEstados = null;

        Usuario usuario;

        private ProgressDialog pDialog;
        private AdmUsuario admUsuario = new AdmUsuario();

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_login);
                //new getProductos().execute();
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.login, menu);
                return true;
        }

        public void onAceptarClick(View button){
        new iniciarSesionTask().execute();
    }

        private class iniciarSesionTask extends AsyncTask<Object, Void, Usuario> {

        protected void onPreExecute(){
                pDialog = new ProgressDialog(LoginActivity.this);
                pDialog.setMessage("Autenticando...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

        @Override
                protected Usuario doInBackground(Object... arg0) {
                        // TODO Auto-generated method stub
                        EditText txtUsuario=(EditText) findViewById(R.id.txtUsuario);
                        EditText txtClave=(EditText) findViewById(R.id.txtClave);

                        Usuario respUsuario =
admUsuario.iniciarSesion(txtUsuario.getText().toString(), txtClave.getText().toString());

                        return respUsuario;
                }

                @Override
        protected void onPostExecute(Usuario respUsuario) {
            pDialog.dismiss();
```

```
            if(respUsuario!=null){
                    successLogin(respUsuario);
            }else{
                    errorLogin();
            }
      }
            }

            public void successLogin(Usuario respUsuario){
                    usuario = respUsuario;
                    Intent intent=new Intent();
                    intent.setClass(this, MenuActivity.class);
                    intent.putExtra("usuario", usuario);
                    startActivity(intent);

    }

    public void errorLogin(){
            Toast tMsg=Toast.makeText(getApplicationContext(), "¡No se pudo autenticar al
usuario!", Toast.LENGTH_SHORT);
            tMsg.show();
    }

    public void onCancelClick(View button){
            EditText txtUsuaio=(EditText) findViewById(R.id.txtUsuario);
            txtUsuaio.setText("");
            EditText txtClave = (EditText) findViewById(R.id.txtClave);
            txtClave.setText("");
            txtUsuaio.requestFocus();
    }
}
```

Menu

```
package ec.edu.itsco.sivade.vista;

import java.util.ArrayList;
import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmCliente;
import ec.edu.itsco.sivade.controlador.AdmEstado;
import ec.edu.itsco.sivade.modelo.Cliente;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
```

```
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MenuActivity extends Activity {

	//VARIABLE GLOBALES
	ArrayList<Producto> listaProductos = null;
	ArrayList<Estado> listaEstados = null;
	ArrayList<Cliente> listaClientes = null;

	private ProgressDialog pDialog;

	Usuario usuario = null;
	Button btnControl = null;
	Button btnInventario = null;
	Button btnPedido = null;
	Button btnConfirmar = null;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_menu);

		try{
			//INICIALIZA ESTADOS Y CONTUINUA
			new getEstados().execute();
			//
			usuario=(Usuario)getIntent().getSerializableExtra("usuario");

			if(usuario!=null){
				TextView tvNombreUsuario=(TextView)
findViewById(R.id.lblNombreUsuario);
				tvNombreUsuario.setText(usuario.getNombres() + " "
+usuario.getApellidos());
				TextView tvPerfil=(TextView) findViewById(R.id.lblPerfil);
				tvPerfil.setText(usuario.getPerfilNombre());

				btnControl = (Button) findViewById(R.id.btnControl);
				btnInventario = (Button) findViewById(R.id.btnInventario);
				btnPedido = (Button) findViewById(R.id.btnPedido);
				btnConfirmar = (Button)
findViewById(R.id.btnConfirmarPedidos);

				switch(usuario.getPerfilId()){
					case 1:
						btnControl.setVisibility(View.GONE);
```

```
                                        btnInventario.setVisibility(View.GONE);

                                        btnConfirmar.setVisibility(View.GONE);
                                        break;
                                case 2:
                                        btnPedido.setVisibility(View.GONE);

                                        break;
                                };
                        }else{
                                Intent intent=new Intent();
                                intent.setClass(this, LoginActivity.class);

                                startActivity(intent);
                        }
                }catch(Exception ex){

                        Toast tMsg=Toast.makeText(getApplicationContext(), "¡No se pudo
autenticar al usuario!", Toast.LENGTH_SHORT);
                tMsg.show();

                        Intent intent=new Intent();
                        intent.setClass(this, LoginActivity.class);
                        intent.putExtra("usuario", "");
                        startActivity(intent);
                }
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.menu, menu);

                return true;
        }
        public void onSalirClick(View button){
                Intent intent=new Intent();
                intent.setClass(this, LoginActivity.class);
                intent.putExtra("usuario", "");
                startActivity(intent);
        }

        public void onPedidoClick(View button){
                Intent intent=new Intent();
                intent.setClass(this, PedidoActivity.class);
                intent.putExtra("usuario", usuario);
                intent.putParcelableArrayListExtra("estados", listaEstados);
                intent.putParcelableArrayListExtra("clientes", listaClientes);
                startActivity(intent);
        }
```

```
public void onControlClick(View button){
        Intent intent=new Intent();
        intent.setClass(this, ControlActivity.class);
        intent.putExtra("usuario", usuario);
        intent.putParcelableArrayListExtra("productos", listaProductos);

        startActivity(intent);
}
public void onInventarioClick(View button){

        Intent intent=new Intent();
        intent.setClass(this, InventarioActivity.class);
        intent.putExtra("usuario", usuario);
        startActivity(intent);

}
public void onConfirmarClick(View button){

        Intent intent=new Intent();
        intent.setClass(this, ConfirmarActivity.class);
        intent.putExtra("usuario", usuario);
        startActivity(intent);

}


//GET ESTADOS
private class getEstados extends AsyncTask<Void, Void, ArrayList<Estado>> {

protected void onPreExecute(){
        pDialog = new ProgressDialog(MenuActivity.this);
        pDialog.setMessage("Inicializando estados...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
}

        @Override
        protected ArrayList<Estado> doInBackground(Void... arg0) {

                return AdmEstado.getEstadosArrayList();

        }

        // onPostExecute displays the results of the AsyncTask.
@Override
protected void onPostExecute(ArrayList<Estado> estados) {
  pDialog.dismiss();
  if(estados.size() > 0){
        successGetEstados(estados);
```

```
            }else{
                    errorGetEstados();
            }
        }
    }

        public void successGetEstados(ArrayList<Estado> estados){

                listaEstados=estados;
                new getClientes().execute();
        }

        public void errorGetEstados(){
                Toast tMsg=Toast.makeText(getApplicationContext(), "No se pudo cargar
los estados!", Toast.LENGTH_SHORT);
        tMsg.show();
        }

        //GET CLIENTES
        private class getClientes extends AsyncTask<Void, Void, ArrayList<Cliente>> {

        protected void onPreExecute(){
                pDialog = new ProgressDialog(MenuActivity.this);
                pDialog.setMessage("Inicializando clientes...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

                @Override
                protected ArrayList<Cliente> doInBackground(Void... arg0) {

                        return AdmCliente.getClientesArrayList();

                }

                // onPostExecute displays the results of the AsyncTask.
      @Override
      protected void onPostExecute(ArrayList<Cliente> clientes) {
        pDialog.dismiss();
        if(clientes.size() > 0){
                successCargaClientes(clientes);
        }else{
                errorCargaClientes();
        }
      }
    }

        public void successCargaClientes(ArrayList<Cliente> clientes){
```

```
                //ArrayAdapter userAdapter = new ArrayAdapter(this,
android.R.layout.simple_spinner_item, clientes);
                //Spinner spClientes = (Spinner) findViewById(R.id.spCliente);
                //spClientes.setAdapter(userAdapter);
                listaClientes=clientes;
        }
        public void errorCargaClientes(){
                Toast tMsg=Toast.makeText(getApplicationContext(), "No se pudo cargar
clientes!", Toast.LENGTH_SHORT);
        tMsg.show();
        }
}
```

Inventario

```
package ec.edu.itsco.sivade.vista;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmInventario;
import ec.edu.itsco.sivade.controlador.AdmPedido;
import ec.edu.itsco.sivade.modelo.Inventario;
import ec.edu.itsco.sivade.modelo.Pedido;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class InventarioActivity extends Activity {
        ArrayList<Producto> listaProductos = null;
        ArrayList<Inventario> listaInventarios = null;

        Usuario usuario = null;
        ListView inventariosListView =null;
        String numeroInventarioSeleccionado="";

        private ProgressDialog pDialog;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_inventario);
        usuario = (Usuario) getIntent().getSerializableExtra("usuario");

        new getInventarios().execute();

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.inventario, menu);
        return true;
}

public void onNuevoClick(View button) {
        Intent intent = new Intent();
        intent.setClass(this, InventarioDetalleActivity.class);
        intent.putExtra("usuario", usuario);
        intent.putExtra("operacion", "I");
        startActivity(intent);
}

public void onEliminarClick(View button) {
        if(numeroInventarioSeleccionado.length()>0){
                long invId = Long.parseLong(numeroInventarioSeleccionado);


        }else{
                Toast tMsg = Toast.makeText(getApplicationContext(),
                                "¡Seleccione un Inventario!",
Toast.LENGTH_SHORT);
                tMsg.show();
        }
}
public void onEditarClick(View button) {
        if(numeroInventarioSeleccionado.length()>0){
                long invId = Long.parseLong(numeroInventarioSeleccionado);
                Inventario inventarioSeleccionado = null;

                for (Inventario inventario : listaInventarios) {
                        if (inventario.getId().equals(invId)) {
                                inventarioSeleccionado = inventario;
                                break;
                        }
                }
                if (inventarioSeleccionado != null) {
```

```
                                Intent intent = new Intent();
                                intent.setClass(this, InventarioDetalleActivity.class);
                                intent.putExtra("usuario", usuario);
                                intent.putExtra("operacion", "E");
                                String numeroInv =
inventarioSeleccionado.getId().toString();
                                intent.putExtra("numeroInventario", numeroInv);
                                SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
                                String fechaInventario =
sdf.format(inventarioSeleccionado.getFecha());
                                intent.putExtra("fechaInventario", fechaInventario);
                                String notaInventario =
inventarioSeleccionado.getMotivo().toString();
                                intent.putExtra("notaInventario", notaInventario);

                                ArrayList<Producto> productosInventario =
inventarioSeleccionado.getDetalle();

        intent.putParcelableArrayListExtra("productosInventario",productosInventario);
                                startActivity(intent);
                        }
                }else{
                        Toast tMsg = Toast.makeText(getApplicationContext(),
                                        "¡Seleccione un Inventario!",
Toast.LENGTH_SHORT);
                        tMsg.show();
                }
        }


        private class getInventarios extends AsyncTask<Void, Void, ArrayList<Inventario>> {

                protected void onPreExecute() {
                        pDialog = new ProgressDialog(InventarioActivity.this);
                        pDialog.setMessage("Cargando inventarios...");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
                        pDialog.show();
                }

                @Override
                protected ArrayList<Inventario> doInBackground(Void... arg0) {

                        return AdmInventario.getInventarios();

                }

                // onPostExecute displays the results of the AsyncTask.
```

```
		@Override
		protected void onPostExecute(ArrayList<Inventario> inventarios) {
			pDialog.dismiss();
			if (inventarios.size() > 0) {
				successGetInventarios(inventarios);

			} else {
				errorGetInventarios();
			}
		}
	}

	public void successGetInventarios(ArrayList<Inventario> inventarios) {

		listaInventarios = inventarios;
		inventariosListView = (ListView) findViewById(R.id.lstInventarios);
		final InventarioAdapter inventarioAdapter = new InventarioAdapter(this,
				R.layout.inventario_list);
		inventariosListView.setAdapter(inventarioAdapter);
		// Populate the list, through the adapter
		for (final Inventario entry : listaInventarios) {
			inventarioAdapter.add(entry);
		}

		setOnItemClick(inventariosListView);
	}

	public void errorGetInventarios() {
		Toast tMsg = Toast.makeText(getApplicationContext(),
				"No se pudo cargar los inventarios!",
Toast.LENGTH_SHORT);
		tMsg.show();
	}

	private void setOnItemClick(ListView pedidosListView) {

		pedidosListView.setOnItemClickListener(new OnItemClickListener() {
			public void onItemClick(AdapterView<?> adapter, View v,	int
position, long id) {
				TextView txtId = (TextView)
v.findViewById(R.id.inventario_list_numero);
				numeroInventarioSeleccionado =
txtId.getText().toString().trim();
			}
		});
	}
}
```

Detalles de inventario:

```
package ec.edu.itsco.sivade.vista;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmInventario;
import ec.edu.itsco.sivade.controlador.AdmPedido;
import ec.edu.itsco.sivade.modelo.Inventario;
import ec.edu.itsco.sivade.modelo.Pedido;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class InventarioActivity extends Activity {
	ArrayList<Producto> listaProductos = null;
	ArrayList<Inventario> listaInventarios = null;

	Usuario usuario = null;
	ListView inventariosListView =null;
	String numeroInventarioSeleccionado="";

	private ProgressDialog pDialog;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_inventario);
		usuario = (Usuario) getIntent().getSerializableExtra("usuario");

		new getInventarios().execute();

	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
```

```
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.inventario, menu);
		return true;
	}

	public void onNuevoClick(View button) {
		Intent intent = new Intent();
		intent.setClass(this, InventarioDetalleActivity.class);
		intent.putExtra("usuario", usuario);
		intent.putExtra("operacion", "I");
		startActivity(intent);
	}

	public void onEliminarClick(View button) {
		if(numeroInventarioSeleccionado.length()>0){
			long invId = Long.parseLong(numeroInventarioSeleccionado);


		}else{
			Toast tMsg = Toast.makeText(getApplicationContext(),
					"¡Seleccione un Inventario!",
Toast.LENGTH_SHORT);
			tMsg.show();
		}
	}
	public void onEditarClick(View button) {
		if(numeroInventarioSeleccionado.length()>0){
			long invId = Long.parseLong(numeroInventarioSeleccionado);
			Inventario inventarioSeleccionado = null;

			for (Inventario inventario : listaInventarios) {
				if (inventario.getId().equals(invId)) {
					inventarioSeleccionado = inventario;
					break;
				}
			}
			if (inventarioSeleccionado != null) {

				Intent intent = new Intent();
				intent.setClass(this, InventarioDetalleActivity.class);
				intent.putExtra("usuario", usuario);
				intent.putExtra("operacion", "E");
				String numeroInv =
inventarioSeleccionado.getId().toString();
				intent.putExtra("numeroInventario", numeroInv);
				SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
				String fechaInventario =
sdf.format(inventarioSeleccionado.getFecha());
				intent.putExtra("fechaInventario", fechaInventario);
```

```
                                String notaInventario =
inventarioSeleccionado.getMotivo().toString();
                                intent.putExtra("notaInventario", notaInventario);

                                ArrayList<Producto> productosInventario =
inventarioSeleccionado.getDetalle();

        intent.putParcelableArrayListExtra("productosInventario",productosInventario);
                                startActivity(intent);
                        }
                }else{
                        Toast tMsg = Toast.makeText(getApplicationContext(),
                                        "¡Seleccione un Inventario!",
Toast.LENGTH_SHORT);
                        tMsg.show();
                }
        }


        private class getInventarios extends AsyncTask<Void, Void, ArrayList<Inventario>> {

                protected void onPreExecute() {
                        pDialog = new ProgressDialog(InventarioActivity.this);
                        pDialog.setMessage("Cargando inventarios...");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
                        pDialog.show();
                }

                @Override
                protected ArrayList<Inventario> doInBackground(Void... arg0) {

                        return AdmInventario.getInventarios();

                }

                // onPostExecute displays the results of the AsyncTask.
                @Override
                protected void onPostExecute(ArrayList<Inventario> inventarios) {
                        pDialog.dismiss();
                        if (inventarios.size() > 0) {
                                successGetInventarios(inventarios);

                        } else {
                                errorGetInventarios();
                        }
                }
        }

        public void successGetInventarios(ArrayList<Inventario> inventarios) {
```

```
                listaInventarios = inventarios;
                inventariosListView = (ListView) findViewById(R.id.lstInventarios);
                final InventarioAdapter inventarioAdapter = new InventarioAdapter(this,
                                R.layout.inventario_list);
                inventariosListView.setAdapter(inventarioAdapter);
                // Populate the list, through the adapter
                for (final Inventario entry : listaInventarios) {
                        inventarioAdapter.add(entry);
                }

                setOnItemClick(inventariosListView);
        }

        public void errorGetInventarios() {
                Toast tMsg = Toast.makeText(getApplicationContext(),
                                "No se pudo cargar los inventarios!",
Toast.LENGTH_SHORT);
                tMsg.show();
        }

        private void setOnItemClick(ListView pedidosListView) {

                pedidosListView.setOnItemClickListener(new OnItemClickListener() {
                        public void onItemClick(AdapterView<?> adapter, View v,         int
position, long id) {
                                TextView txtId = (TextView)
v.findViewById(R.id.inventario_list_numero);
                                numeroInventarioSeleccionado =
txtId.getText().toString().trim();
                        }
                });
        }
}
```

Pedidos

```
package ec.edu.itsco.sivade.vista;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmPedido;
import ec.edu.itsco.sivade.modelo.Cliente;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Pedido;
```

```
import ec.edu.itsco.sivade.modelo.PedidoDetalle;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;
import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class PedidoActivity extends Activity {
	//VARIABLE GLOBALES
	ArrayList<Producto> listaProductos = null;
	ArrayList<Estado> listaEstados = null;
	ArrayList<Cliente> listaClientes = null;


	private ProgressDialog pDialog;
	Usuario usuario = null;
	List<Pedido> productosPedido = null;
	String numeroPedidoSeleccionado="";
	ListView pedidosListView =null;
	List<Pedido> listaPedidos = null;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_pedido);

		//INICIALIZACION
		Intent i = this.getIntent();
		//GET ESTADOS
		ArrayList<Estado> tmpEstadosList =
i.getParcelableArrayListExtra("estados");
		listaEstados = new ArrayList<Estado>();
		listaEstados = tmpEstadosList;
		//GET CLIENTES
		ArrayList<Cliente> tmpClientesList =
i.getParcelableArrayListExtra("clientes");
		listaClientes = new ArrayList<Cliente>();
```

```
            listaClientes = tmpClientesList;
            //

            productosPedido = new ArrayList<Pedido>();

            usuario=(Usuario)getIntent().getSerializableExtra("usuario");

            new getPedidos().execute();
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.pedido, menu);
            return true;
        }

        public void onNuevoClick(View button){
            Intent intent=new Intent();
            intent.setClass(this, PedidoDetalleActivity.class);
            intent.putExtra("usuario", usuario);
            intent.putExtra("operacion", "I");
            intent.putParcelableArrayListExtra("estados", listaEstados);
            intent.putParcelableArrayListExtra("clientes", listaClientes);
            startActivity(intent);
        }

        public void onEditarClick(View button) {
            if (numeroPedidoSeleccionado.length() > 0) {

                long pedId = Long.parseLong(numeroPedidoSeleccionado);

                Pedido pedidoSeleccionado = null;

                for (Pedido pedido : listaPedidos) {
                    if (pedido.getId().equals(pedId)) {
                        pedidoSeleccionado = pedido;
                        break;
                    }
                }
                if (pedidoSeleccionado != null) {

                    SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yyyy");
                    String fechaPedido =
sdf.format(pedidoSeleccionado.getFecha());

                    ArrayList<Producto> productos = new
ArrayList<Producto>();
```

```
                                Producto tmpProducto = null;
                                for (PedidoDetalle detalle :
pedidoSeleccionado.getDetalle()) {
                                        tmpProducto = new Producto();
                                        tmpProducto.setId(detalle.getProId());
                                        tmpProducto.setNombre(detalle.getProNombre());
                                        tmpProducto.setCantidad(detalle.getCantidad());
                                        tmpProducto.setPrecio(detalle.getPrecio());
                                        productos.add(tmpProducto);
                                }

                                Intent intent = new Intent();
                                intent.setClass(this, PedidoDetalleActivity.class);
                                intent.putExtra("usuario", usuario);
                                intent.putExtra("operacion", "E");
                                intent.putExtra("numeroPedido",
numeroPedidoSeleccionado);
                                intent.putExtra("fechaPedido", fechaPedido);
                                intent.putExtra("notaPedido",
pedidoSeleccionado.getNota());

        intent.putParcelableArrayListExtra("productosDetallePedido",
                                                productos);
                                Estado estadoPedido = pedidoSeleccionado.getEstado();
                                Cliente clientePedido = pedidoSeleccionado.getCliente();
                                intent.putParcelableArrayListExtra("estados", listaEstados);
                                intent.putExtra("estadoPedido", estadoPedido);
                                intent.putExtra("clientePedido", clientePedido);
                                intent.putParcelableArrayListExtra("clientes", listaClientes);
                                startActivity(intent);
                        }

                } else {
                        Toast tMsg = Toast.makeText(getApplicationContext(),
                                        "¡ Seleccione un pedido !", Toast.LENGTH_SHORT);
                        tMsg.show();
                }
        }

        public void onEliminarClick(View button){
                if(numeroPedidoSeleccionado.length()>0){
                        long pedId = Long.parseLong(numeroPedidoSeleccionado);

                        Pedido pedidoSeleccionado = null;

                        for (Pedido pedido : listaPedidos) {
                                if (pedido.getId().equals(pedId)) {
                                        pedidoSeleccionado = pedido;
                                        break;
                                }
```

```
                        }
                        if (pedidoSeleccionado != null) {
                                if(pedidoSeleccionado.getEstado().getId()==1){
                                        mensajeConfirmacionEliminar();
                                }else{
                                        Toast
tMsg=Toast.makeText(getApplicationContext(), "¡ No se puede eliminar pedidos
confirmados o rechazados !", Toast.LENGTH_SHORT);
                                tMsg.show();
                                }
                        }

                }else{
                        Toast tMsg=Toast.makeText(getApplicationContext(), "¡ Seleccione
un pedido !", Toast.LENGTH_SHORT);
                tMsg.show();
                }
        }

        //LISTAR PEDIDOS
        private class getPedidos extends AsyncTask<Void, Void, List<Pedido>> {

        protected void onPreExecute(){
                pDialog = new ProgressDialog(PedidoActivity.this);
                pDialog.setMessage("Cargando Pedidos...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

                @Override
                protected List<Pedido> doInBackground(Void... arg0) {

                        return AdmPedido.getPedidos(usuario.getUsuario());

                }

                // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(List<Pedido> pedidos) {
        pDialog.dismiss();
        if(pedidos.size() > 0){
                successGetPedidos(pedidos);
        }else{
                Toast tMsg=Toast.makeText(getApplicationContext(), "Size:
"+pedidos.size(), Toast.LENGTH_SHORT);
                tMsg.show();
                errorGetPedidos();
        }
    }
```

```
    }

        public void successGetPedidos(List<Pedido> pedidos){

                listaPedidos=pedidos;

                pedidosListView = (ListView) findViewById(R.id.lstPedidos);
      final PedidoAdapter pedidoAdapter = new PedidoAdapter(this, R.layout.pedido_list);
      pedidosListView.setAdapter(pedidoAdapter);
      // Populate the list, through the adapter
      for(final Pedido entry : pedidos) {
          pedidoAdapter.add(entry);
      }

      setOnItemClick(pedidosListView);
          }

          public void errorGetPedidos(){
                Toast tMsg=Toast.makeText(getApplicationContext(), "No se pudo cargar
los pedidos!", Toast.LENGTH_SHORT);
          tMsg.show();
          }
          private void setOnItemClick(ListView pedidosListView){

                pedidosListView.setOnItemClickListener(new OnItemClickListener() {
          public void onItemClick(AdapterView<?> adapter, View v, int position, long id) {
                TextView txtId=(TextView)v.findViewById(R.id.pedido_list_numero);
                String[] strId = txtId.getText().toString().trim().split(":");
                numeroPedidoSeleccionado = strId[0];
            }
                });
    }

          public void mensajeConfirmacionEliminar() {
                DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                              // TODO Auto-generated method stub
                               switch (which) {
                                    case DialogInterface.BUTTON_POSITIVE:

                                          //ELIMINAR EL PEDIDO
                                          new eliminarPedido().execute();

                                          break;

                                    case DialogInterface.BUTTON_NEGATIVE:
```

```
                                                    Toast.makeText(PedidoActivity.this, "No
Clicked", Toast.LENGTH_LONG).show();
                                                    break;
                                        }
                                }
                        };
                        AlertDialog.Builder builder = new AlertDialog.Builder(this);
                         builder.setMessage("¿Esta seguro de eliminar?")
        .setPositiveButton("Si", dialogClickListener)
        .setNegativeButton("No", dialogClickListener).show();
            }

            //ELIMINAR PEDIDO
            private class eliminarPedido extends AsyncTask<Void, Void, String> {

                    protected void onPreExecute(){
                            pDialog = new ProgressDialog(PedidoActivity.this);
                            pDialog.setMessage("Eliminando el pedido...");
                            pDialog.setIndeterminate(false);
                            pDialog.setCancelable(false);
                            pDialog.show();
                    }

                    @Override
                    protected String doInBackground(Void... args) {
                            // TODO Auto-generated method stub
                            String resp = "";
                            long numeroPedido=Long.parseLong(numeroPedidoSeleccionado);
                            if(AdmPedido.eliminarPedido(numeroPedido)){
                                    resp = "OK";
                            }else{
                                    resp = "ER";
                            }

                            return resp;
                    }

                    @Override
                    protected void onPostExecute(String resp) {
                            //pDialog.dismiss();
                            pDialog.hide();

                            if(resp.equals("OK")){
                                    successEliminaPedido();
                            }else{
                                    errorEliminaPedido();
                            }
                    }

            }
```

```
	public void successEliminaPedido(){
		Toast tMsg=Toast.makeText(getApplicationContext(), "¡ Pedido Eliminado
!", Toast.LENGTH_SHORT);
		tMsg.show();

		new getPedidos().execute();
	}
	public void errorEliminaPedido(){
		Toast tMsg=Toast.makeText(getApplicationContext(), "¡Error al eliminar el
Pedido!", Toast.LENGTH_SHORT);
		tMsg.show();
	}

}
```

Detalles pedidos

```
package ec.edu.itsco.sivade.vista;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.InputType;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
```

```
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;
import ec.edu.itsco.sivade.R;
import ec.edu.itsco.sivade.controlador.AdmCliente;
import ec.edu.itsco.sivade.controlador.AdmEstado;
import ec.edu.itsco.sivade.controlador.AdmPedido;
import ec.edu.itsco.sivade.controlador.AdmProducto;
import ec.edu.itsco.sivade.controlador.Bdd;
import ec.edu.itsco.sivade.modelo.Cliente;
import ec.edu.itsco.sivade.modelo.Estado;
import ec.edu.itsco.sivade.modelo.Pedido;
import ec.edu.itsco.sivade.modelo.PedidoDetalle;
import ec.edu.itsco.sivade.modelo.Producto;
import ec.edu.itsco.sivade.modelo.Usuario;

public class PedidoDetalleActivity extends Activity {

	//VARIABLE GLOBALES
	ArrayList<Producto> listaProductos = null;
	ArrayList<Estado> listaEstados = null;
	ArrayList<Cliente> listaClientes = null;


	List<Producto> productosPedido = new ArrayList<Producto>();

	AdmEstado admEstado = new AdmEstado();
	final Context context = this;
	private ImageButton btnAdicionar;
	private ImageButton btnRemover;
	private ImageButton btnGuardar;
	private String codigoProductoSeleccionado="";
	private ProgressDialog pDialog;
	Usuario usuario = null;
	String operacion = "";
	String tipoPedido="";
	Spinner spEstados=null;
	Spinner spClientes=null;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_pedido_detalle);
		usuario=(Usuario)getIntent().getSerializableExtra("usuario");
		operacion = getIntent().getStringExtra("operacion");


		//INICIALIZACION
		Intent i = this.getIntent();
```

```
                //GET ESTADOS
                ArrayList<Estado> tmpEstadosList =
i.getParcelableArrayListExtra("estados");
                listaEstados = new ArrayList<Estado>();
                listaEstados = tmpEstadosList;
                cargarEstados(tmpEstadosList);
                //GET CLIENTES
                ArrayList<Cliente> tmpClientesList =
i.getParcelableArrayListExtra("clientes");
                listaClientes = new ArrayList<Cliente>();
                listaClientes = tmpClientesList;
                cargarClientes(tmpClientesList);



                //ADICIONAR
                btnAdicionar = (ImageButton) findViewById(R.id.btnPedidoDetalleAdd);
                btnAdicionar.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {
                                AlertDialog.Builder alert = new AlertDialog.Builder(context);
                                alert.setTitle("Añadir Producto");
                                alert.setMessage("Cantidad:");

                                final EditText input = new EditText(context);
                                input.setInputType(InputType.TYPE_CLASS_NUMBER);

                                alert.setView(input);

                                alert.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog, int
whichButton) {
                                                String srtCantidad =
input.getEditableText().toString();

                                                Producto producto = (Producto) ( (Spinner)
findViewById(R.id.spProducto) ).getSelectedItem();



                                                int cantidad = Integer.parseInt(srtCantidad);


        if(cantidad>Integer.parseInt(producto.getSaldo())){
                                                        Toast
tMsg=Toast.makeText(getApplicationContext(), "No existe suficiente Stock de:
"+producto.getNombre(), Toast.LENGTH_SHORT);
                                                tMsg.show();
                                                }else{
```

```
                                                    boolean existe=false;
                                                    for (Producto tmpProducto :
productosPedido) {

        if(tmpProducto.getId().equals(producto.getId())){
                                                                existe = true;
                                                        }
                                                    }

                                                    if(existe){
                                                        Toast
tMsg=Toast.makeText(getApplicationContext(), "¡Este producto ya existe en el pedido!",
Toast.LENGTH_SHORT);
                                                    tMsg.show();
                                                    }else{

        producto.setCantidad(cantidad);

        productosPedido.add(producto);
                                                            cargarProductos();
                                                    }

                                                }

                                            }
                                    });
                                    alert.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
                                            public void onClick(DialogInterface dialog, int
whichButton) {
                                                    dialog.cancel();
                                            }
                                    });
                                    AlertDialog alertDialog = alert.create();
                                    alertDialog.show();


                            }
                    });

                    if(operacion.equals("E")){

                            String numeroPedido =
getIntent().getStringExtra("numeroPedido");

                            EditText txtNumeroPedido = (EditText)
findViewById(R.id.txtPedidoDetalleNumero);
                            txtNumeroPedido.setText(numeroPedido);
                            txtNumeroPedido.setEnabled(false);
```

```
                        String fechaPedido = getIntent().getStringExtra("fechaPedido");
                        EditText txtFecha = (EditText)
findViewById(R.id.txtPedidoDetalleFecha);
                        txtFecha.setText(fechaPedido);
                        txtFecha.setEnabled(false);

                        //GET ESTADO DEL PEDIDO A EDITAR
                        Estado tmpEstadoPedido = i.getParcelableExtra("estadoPedido");

                        spEstados.setSelection(getIndexSpEstado(spEstados,
tmpEstadoPedido));
                        //GET CLIENTE DEL PEDIDO A EDITAR
                        Cliente tmpClientePedido = i.getParcelableExtra("clientePedido");

                        spClientes.setSelection(getIndexSpCliente(spClientes,
tmpClientePedido));

                        String notaPedido = getIntent().getStringExtra("notaPedido");
                        EditText txtNota = (EditText) findViewById(R.id.txtNota);
                        txtNota.setText(notaPedido);

                        //GET DETALLE
                        ArrayList<Producto> tmpProductosList =
i.getParcelableArrayListExtra("productosDetallePedido");

                        for (Producto tmpProducto : tmpProductosList) {
                                productosPedido.add(tmpProducto);
                        }
                        if(tmpEstadoPedido.getId()!=1){
                                spClientes.setEnabled(false);
                                txtNota.setEnabled(false);
                                btnAdicionar.setEnabled(false);
                                btnRemover = (ImageButton)
findViewById(R.id.btnPedidoDetalleRemove);
                                btnRemover.setEnabled(false);
                                btnGuardar = (ImageButton)
findViewById(R.id.btnPedidoDetalleGuardar);
                                btnGuardar.setEnabled(false);
                        }else{
                                //CARGAR PRODUCTOS
                                new getProductos().execute();
                        }

                        cargarProductos();
                }else{
                        EditText txtNumeroPedido = (EditText)
findViewById(R.id.txtPedidoDetalleNumero);
                        txtNumeroPedido.setEnabled(false);
```

```
                    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
                    String fechaPedido = sdf.format(new Date());
                    EditText txtFecha = (EditText)
findViewById(R.id.txtPedidoDetalleFecha);
                    txtFecha.setText(fechaPedido);
                    txtFecha.setEnabled(false);

                    spEstados.setSelection(0);
                    spClientes.setSelection(0);

                    //CARGAR PRODUCTOS
                    new getProductos().execute();
                }

                if(usuario!=null){
                    switch(usuario.getPerfilId()){
                    case 1:
                        tipoPedido="V";
                        break;
                    case 2:
                        tipoPedido="B";
                        break;
                    };
                }
            }

            private int getIndexSpEstado(Spinner spinner, Estado estado) {

                int index = 0;
                for (int i = 0; i < spinner.getCount(); i++) {
                    Estado tmpEstado=(Estado) spinner.getItemAtPosition(i);
                    if (tmpEstado.getId()==estado.getId()) {
                        index = i;
                        break;
                    }
                }
                return index;
            }
            private int getIndexSpCliente(Spinner spinner, Cliente cliente) {

                int index = 0;
                for (int i = 0; i < spinner.getCount(); i++) {
                    Cliente tmpCliente=(Cliente) spinner.getItemAtPosition(i);
                    String tmpId = tmpCliente.getId().trim();
                    String id = cliente.getId().trim();
                    if (tmpId.equals(id)) {
                        index = i;
                        break;
                    }
                }
```

```
                return index;
        }
        private void cargarEstados(ArrayList<Estado> estados){
                ArrayAdapter<Estado> estadoAdapter = new ArrayAdapter<Estado>(this,
android.R.layout.simple_spinner_item, estados);
                spEstados = (Spinner) findViewById(R.id.spEstado);
                spEstados.setEnabled(false);
                spEstados.setAdapter(estadoAdapter);

        }

        private void cargarClientes(ArrayList<Cliente> clientes){
                ArrayAdapter<Cliente> userAdapter = new ArrayAdapter<Cliente>(this,
android.R.layout.simple_spinner_item, clientes);
                spClientes = (Spinner) findViewById(R.id.spCliente);
                spClientes.setAdapter(userAdapter);
        }

        public void onGuardarClick(View button){
                if(!productosPedido.isEmpty()){
                        mensajeConfirmacion();
                }else{
                        Toast tMsg=Toast.makeText(getApplicationContext(), "¡Añada al
menos un Producto para el Pedido!", Toast.LENGTH_SHORT);
                tMsg.show();
                }
  }

        private void setOnItemClick(ListView productosListView){

        productosListView.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> adapter, View v, int position, long id) {
                TextView
txtNombre=(TextView)v.findViewById(R.id.pedido_producto_nombre);
                String[] strNombre = txtNombre.getText().toString().trim().split(":");
                codigoProductoSeleccionado = strNombre[0];
        }
                });
  }

        public void onRemoverClick(View button){
                mensajeConfirmacionRemover();
  }


        private void cargarProductos(){
                final ListView productosListView = (ListView)
findViewById(R.id.lstProductosDetalle);
     final PedidoDetalleAdapter productoAdapter = new PedidoDetalleAdapter(this,
R.layout.productos_pedido_list);
```

```
productosListView.setAdapter(productoAdapter);

for(final Producto entry : productosPedido) {
    productoAdapter.add(entry);
}

double total = 0;
for (Producto producto : productosPedido) {
                    total+=producto.getPrecio()*producto.getCantidad();
                }
TextView tvTotalPedido=(TextView) findViewById(R.id.lblDetallePedidoTotal);
tvTotalPedido.setText("Total: "+String.format( "%.2f", total ) );
setOnItemClick(productosListView);
    }

    public void mensajeConfirmacion() {
            DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                        switch (which) {
                            case DialogInterface.BUTTON_POSITIVE:

                                //ALMACENA EL PEDIDO
                                new guardarPedido().execute();

    //Toast.makeText(PedidoDetalleActivity.this, "Yes Clicked",
Toast.LENGTH_LONG).show();
                                break;

                            case DialogInterface.BUTTON_NEGATIVE:
                                Toast.makeText(PedidoDetalleActivity.this,
"No Clicked", Toast.LENGTH_LONG).show();
                                break;
                        }
                    }
            };
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
             builder.setMessage("Esta seguro de almacenar?")
    .setPositiveButton("Si", dialogClickListener)
    .setNegativeButton("No", dialogClickListener).show();
    }

    public void mensajeConfirmacionRemover() {
            DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {

                    @Override
```

```
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                        switch (which) {
                            case DialogInterface.BUTTON_POSITIVE:

                                //REMOVER ITEM
                                Producto ProductoSeleccionado = null;
                                for (Producto producto : productosPedido) {

        if(producto.getId().equals(codigoProductoSeleccionado)){
                                                    ProductoSeleccionado =
producto;
                                        }
                                }
                                if(ProductoSeleccionado!=null){

        productosPedido.remove(ProductoSeleccionado);
                                            cargarProductos();
                                }else{

        Toast.makeText(PedidoDetalleActivity.this, "Seleccione el producto que desea
remover del pedido!", Toast.LENGTH_LONG).show();
                                }


                                break;

                            case DialogInterface.BUTTON_NEGATIVE:

                                break;
                        }
                    }
                };
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                 builder.setMessage("Esta seguro de remover el item?")
    .setPositiveButton("Si", dialogClickListener)
    .setNegativeButton("No", dialogClickListener).show();
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.pedido_detalle, menu);
            return true;
        }

        //GET PRODUCTOS
        private class getProductos extends AsyncTask<Void, Void, List<Producto>> {

            protected void onPreExecute(){
```

```
                    pDialog = new ProgressDialog(PedidoDetalleActivity.this);
                    pDialog.setMessage("Cargando Productos...");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(false);
                    pDialog.show();
                }

                    @Override
                    protected List<Producto> doInBackground(Void... arg0) {

                            return AdmProducto.getProductosListBox();

                    }

                    // onPostExecute displays the results of the AsyncTask.
            @Override
            protected void onPostExecute(List<Producto> productos) {
                pDialog.dismiss();
                if(productos.size() > 0){
                        successCargaProductos(productos);
                }else{
                        Toast tMsg=Toast.makeText(getApplicationContext(), "Size:
"+productos.size(), Toast.LENGTH_SHORT);
                        tMsg.show();
                        errorCargaProductos();
                }
              }
            }

                public void successCargaProductos(List<Producto> productos){
                        ArrayAdapter<Producto> userAdapter = new
ArrayAdapter<Producto>(this, android.R.layout.simple_spinner_item, productos);

                        Spinner spProductos = (Spinner) findViewById(R.id.spProducto);
                        spProductos.setAdapter(userAdapter);
                }
                public void errorCargaProductos(){
                        Toast tMsg=Toast.makeText(getApplicationContext(), "No se pudo
cargar productos!", Toast.LENGTH_SHORT);
                tMsg.show();
                }

                //GUARDAR PEDIDO
                private class guardarPedido extends AsyncTask<Void, Void, String> {

                protected void onPreExecute(){
                        pDialog = new ProgressDialog(PedidoDetalleActivity.this);
                        pDialog.setMessage("Almacenando el pedido...");
                        pDialog.setIndeterminate(false);
                        pDialog.setCancelable(false);
```

```
                pDialog.show();
            }

            @Override
                protected String doInBackground(Void... args) {
                    // TODO Auto-generated method stub
                String resp = "";
                EditText txtNota=(EditText) findViewById(R.id.txtNota);

                Pedido nuevoPedido = new Pedido();
                Estado estado=new Estado();
                //estado.setId(1);//Pendiente //AQUI EN NUEVO NO COGE EL
ESTADO!!!

                nuevoPedido.setEstado(estado);
                nuevoPedido.setFecha(new Date());
                nuevoPedido.setNota(txtNota.getText().toString().trim());

                Cliente cliente = (Cliente) ( (Spinner) findViewById(R.id.spCliente)
).getSelectedItem();
                nuevoPedido.setCliente(cliente);
                nuevoPedido.setTipo(tipoPedido);
                nuevoPedido.setUsuario(usuario.getUsuario());

                List<PedidoDetalle> detalle = new ArrayList<PedidoDetalle>();
                PedidoDetalle linea= null;
                for(final Producto producto : productosPedido) {
                    linea = new PedidoDetalle();
                    linea.setProId(producto.getId());
                    linea.setProNombre(producto.getNombre());
                    linea.setCantidad(producto.getCantidad());
                    linea.setPrecio(producto.getPrecio());
                    detalle.add(linea);
            }
                nuevoPedido.setDetalle(detalle);

                if(operacion.equals("E")){

                    EditText txtNumeroPedido = (EditText)
findViewById(R.id.txtPedidoDetalleNumero);

        nuevoPedido.setId(Long.parseLong(txtNumeroPedido.getText().toString().trim()));


                    Estado estadoSeleccionado = (Estado) ( (Spinner)
findViewById(R.id.spEstado) ).getSelectedItem();
                    nuevoPedido.setEstado(estadoSeleccionado);

                    if(AdmPedido.actualizarPedido(nuevoPedido)){
                        resp = "OK";
```

```
                    }else{
                        resp = "ER";
                    }

                }else{
                    nuevoPedido.setEstado(listaEstados.get(0));

                    if(AdmPedido.guardarPedido(nuevoPedido)){
                        resp = "OK";
                    }else{
                        resp = "ER";
                    }
                }

                    return resp;
                }

                    @Override
            protected void onPostExecute(String resp) {
            //pDialog.dismiss();
            pDialog.hide();

            if(resp.equals("OK")){
                successGuardarPedido();
            }else{
                errorGuardarPedido();
            }
          }

        }
        public void successGuardarPedido(){
            Toast tMsg=Toast.makeText(getApplicationContext(), "¡Datos almacenados
correctamente!", Toast.LENGTH_SHORT);
            tMsg.show();

            Intent intent=new Intent();
                intent.setClass(this, PedidoActivity.class);
                intent.putExtra("usuario", usuario);
                intent.putParcelableArrayListExtra("estados", listaEstados);
                intent.putParcelableArrayListExtra("clientes", listaClientes);
                startActivity(intent);
            }
        public void errorGuardarPedido(){
                Toast tMsg=Toast.makeText(getApplicationContext(), "¡Error al
Registrar el Pedido!", Toast.LENGTH_SHORT);
            tMsg.show();
            }

}
```

Anexo A.13: Manual de Usuario

BODEGUERO:

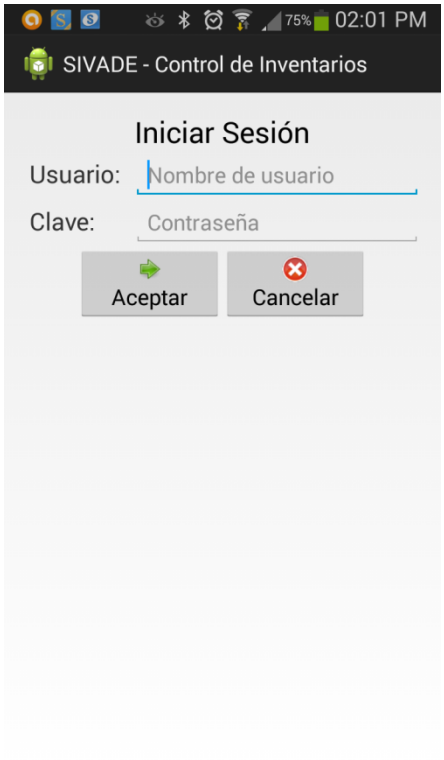


Figura:62 Iniciar sesión

1.-Pantalla de acceso para identificar al usuario y su contraseña para el uso permitido al inventario y su manipulación .

Figura: 63 Menu

2.- Panel de acceso al programa con datos como bienvenida, nombre, cargo del usuario y a las diferentes opciones que presenta el mismo como control, Inventarios, Pedidos y el cierre del programa

Figura: 64 Menu control

3.- Pagina principal de la función de control el mismo que tiene diferentes opciones como (de izquierda a derecha) Código de barras, búsqueda, actualización de productor, ingreso de productos y egresar productos

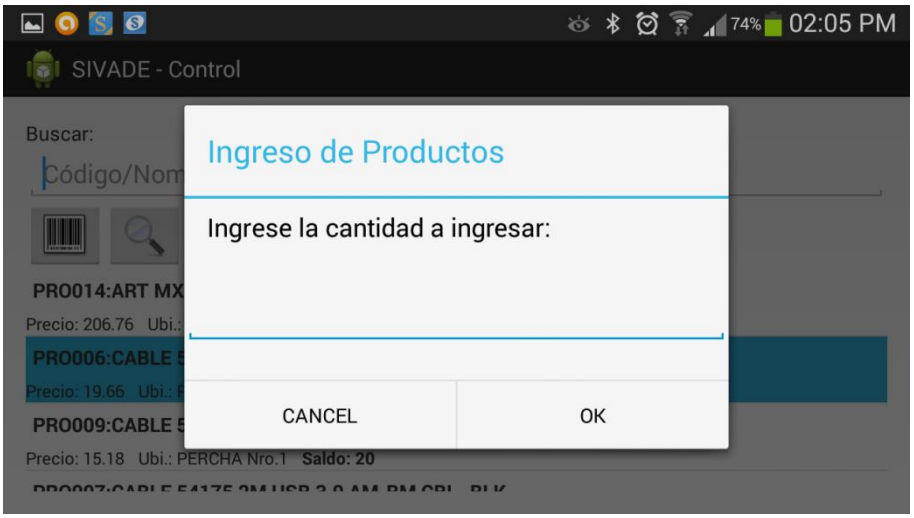


Figura: 65 Igreso de productos

4.- La opción de ingreso de productos nos aparece una pantalla en la que podemos delimitar la cantidad que se va a ingresar al inventario de diferentes productos

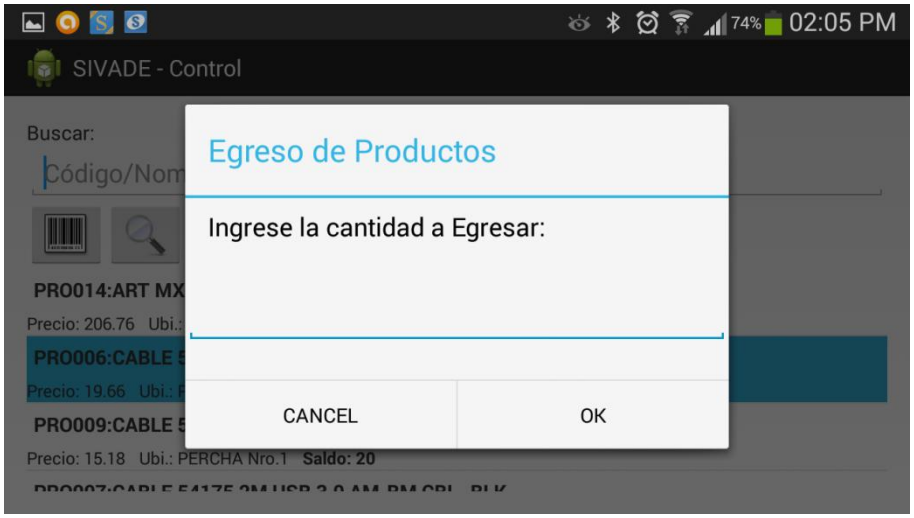


Figura: 66 Egreso de productos

5.-La opción de egreso de productos nos aparece una pantalla en la que podemos delimitar la cantidad que se va a egresar al inventario de diferentes productos

Figura: 67 Inventario

6.- En la opción de inventarios nos da como opción delimitar las existencias que posee la empresa para la venta y distribución de los diferentes productos para ser comercializados

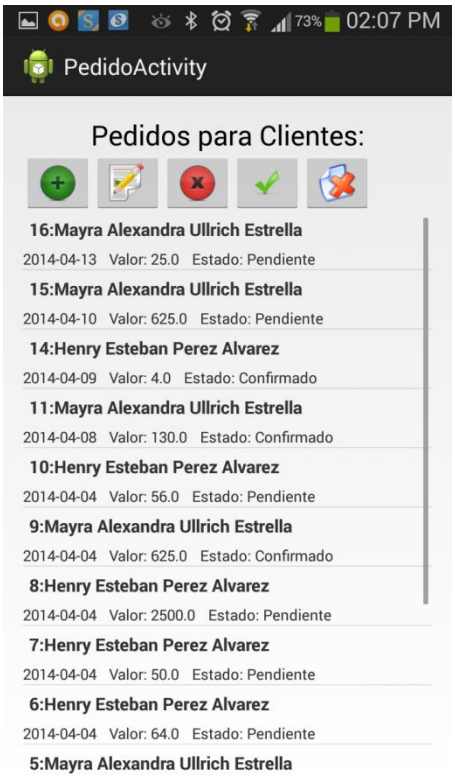


Figura: 68 Pedidos clienter

7.- La página principal de la opción de pedidos nos da como alternativas varias como son añadir un nuevo pedido, añadir un producto, eliminar un pedido, confirmar el pedido y cancelar un pedido

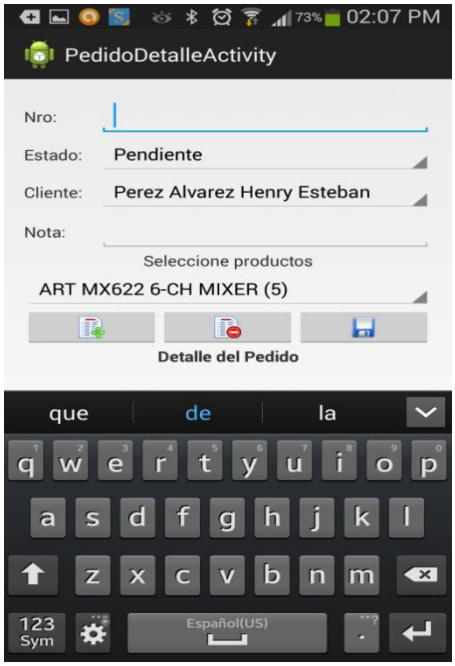


Figura: 69 Ingreso de detalle de pedidos

8.- Añadir un nuevo pedido al hacer esta accion se debe delimitar el numero de pedido, el estado, el nombre del cliente y nos da una opcion de incluir una nota

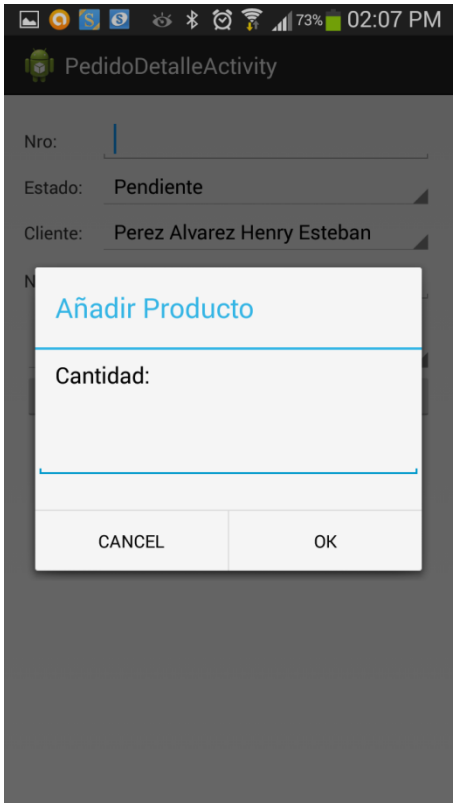


Figura: 70 Añadir producto

9.- Al añadir la cantidad del pedido estamos delimitando el número de unidades el cliente quiere que le vendamos y queda para el correcto manejo e manipulación del inventario en bodega

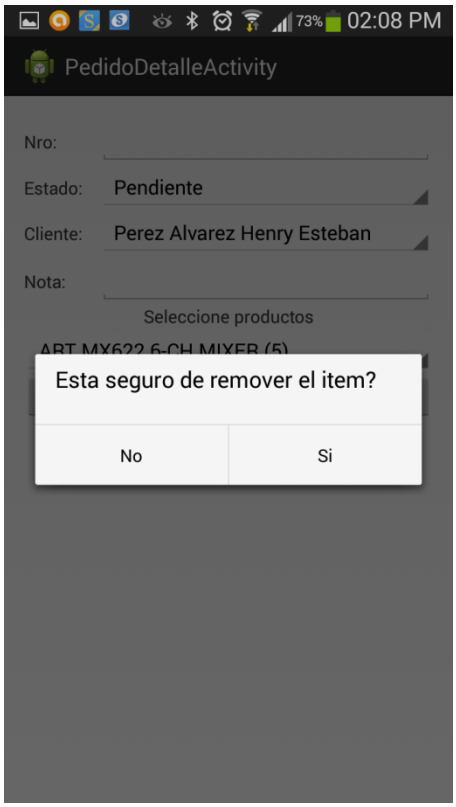


Figura: 71 Remover item

10.- Esta opción nos da la opción de remover uno de los productos que el cliente nos pide y editar el pedido final

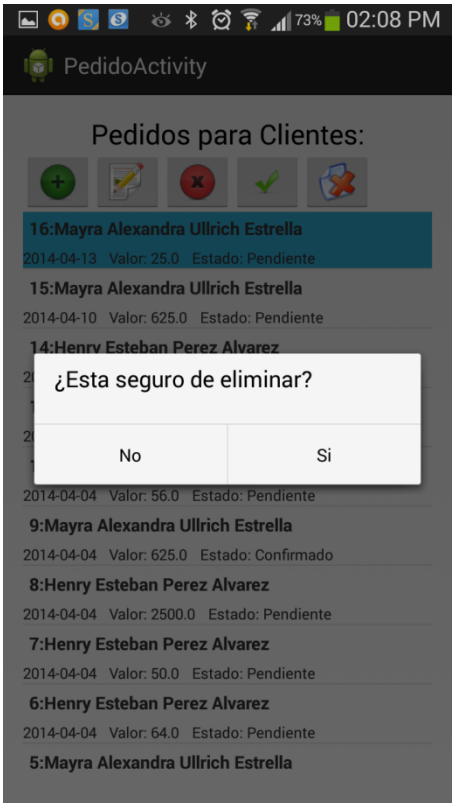


Figura: 72 Eliminar

11.-La opción de eliminar nos da la facilidad de cancelar los diferentes pedido que hemos realizado y tomar las medidas necesarias para no causar una excesiva cantidad de productos en el inventario

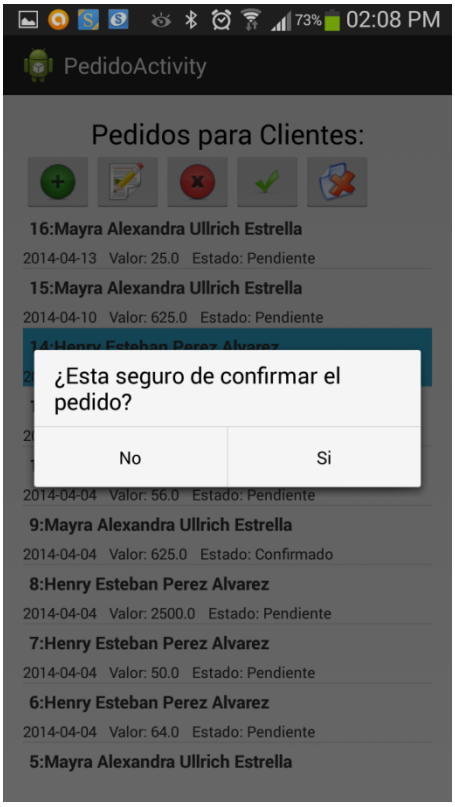


Figura:73 Confirmar Pedido

12.-En la opción de confirmación del pedido nos da la facilidad de notificar al encargado de bodega la eminente salida de mercadería en tiempo y plazo establecido por el cliente

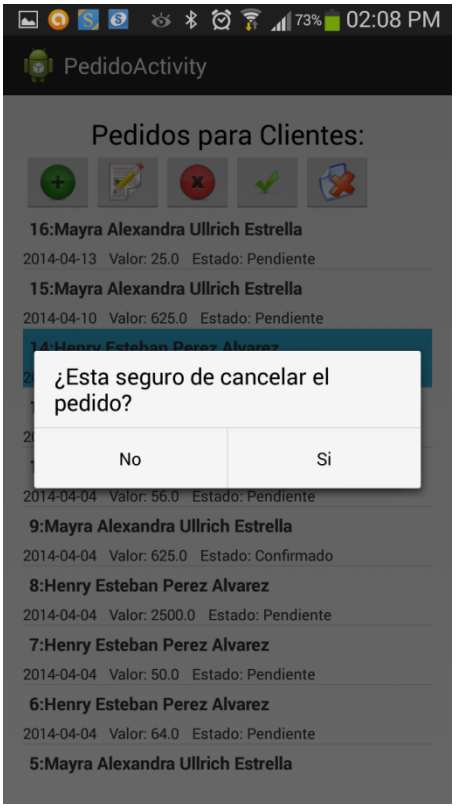


Figura: 74 Cancelar pedido

13.- En la opción de cancelación del pedido nos da la facilidad de notificar al encargado de bodega la venta fallida de productos hecho por parte del cliente y la actualización de dicha mercadería en el inventario general

AGENTE DE VENTAS:

Figura: 78 Menu del agente vendedor

14.- Eset menu es para que scoja una opcion el vendedor.

Figura: 79 Pedidos vendedor

14.- El vendedor regisyra los pedidos.

Figura: 79 Consulta de saldos.

15.- El vendedor puede consultar los saldos de invenatio

Figura: 80 consulta de calendario

16.- El usuario consulta el calendario de pedidos.

Anexo A.14 Glosario

Actor.-Algo o alguien externo al sistema en desarrollo pero que interactúa con él.

Arquitectura.-Estructura lógica y física de un sistema empleado para diseñar todas las estrategias y tácticas aplicadas durante el desarrollo.

Atributo.-Definición de dato simple o compuesto perteneciente a un objeto de clase.

Clase.- Descripción de un grupo de objetos con atributos, conducta y relaciones comunes.

Caso de uso.- Representación de un proceso del negocio. representa el modelo de diálogo entre un actor y el sistema

Diagrama de casos de uso.-Representación gráfica que representa algunos o todos los actores, casos de uso y sus interacciones en el sistema.

Diagrama de clases.- Representación gráfica que permite visualizar algunas o todas las clases de un modelo

Diagrama de secuencias.- Representación gráfica que describe interacciones de secuencia de objetos.

Diagrama de iteración.- Representación gráfica de un proceso aplicado en el desarrollo de la plataforma virtual

IEEE.- Corresponde a las siglas de the institute of electrical and electronics engineers, el instituto de ingenieros eléctricos y electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, ingenieros en sistemas e ingenieros en telecomunicación....

UML (lenguaje de modelamiento unificado).- Lenguaje usado para especificar, visualizar y documentar un sistema en desarrollo orientado a objetos.

Plataforma virtual.- Las plataformas virtuales, se refieren, a la tecnología utilizada para la creación y desarrollo de cursos o módulos didácticos en la Web que se usan de manera más amplia en la Web 2.0. Mejora de la comunicación aprendizaje-enseñanza.

Base de datos.-Estructura de software que colecciona información muy variada de diferentes personas y cosas (es decir, de una realidad determinada), cada una de las cuales tiene algo en común o campos comunes con todos o con algunos. Se diseñó con la finalidad de solucionar y agilizar la administración de los datos que se almacenan en la memoria del computador.

Hardware.- Todos aquellos componentes físicos de un computador, todo lo visible y tangible. Por extensión, se aplica también a otros componentes electrónicos que no necesariamente forman parte de un computador.

Home pages.- En el web se refiere a las páginas de inicio que enlazan con otras páginas relacionadas.

Html (hypertext Markup Language.-Lenguaje en que se escriben los documentos que se utilizan en internet.

Informática.- Ciencia del tratamiento automático y racional de la información, considerada como soporte de los conocimientos y comunicaciones, a través de los ordenadores.

Internet.-Proyecto que ya está en marcha para mejorar internet que se trata de la posibilidad de navegar en la red a una velocidad de 622 megabits por segundo, más de 1000 veces la velocidad actual disponible.

JavaScript.- Un lenguaje de comandos multiplataforma del WWW desarrollado por Netscape Communications. El código de JavaScript se inserta directamente en una página HTML.

webgrafia:

http://es.wikipedia.org/wiki/Caso_de_uso

http://es.wikipedia.org/wiki/Diagrama_de_secuencia

http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/apuntes_1/capa_enlace.htm

http://jmaw.blogspot.com/2013/01/arquitectura-de-aplicaciones-web-capa_5.html

http://www.elrinconcito.com/articulos/Sesiones/sesiones.pdf

http://es.wikipedia.org/wiki/Actor_(UML)

http://www.sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf

http://www.sparxsystems.com.ar/resources/tutorial/uml2_activitydiagram.html

http://translate.google.com.ec/translate?hl=es&sl=en&u=http://pic.dhe.ibm.com/infocenter/rsarthlp/v8/topic/com.ibm.xtools.modeler.doc/topics/cinterfc.html&prev=/search%3Fq%3Dinterfaces%2Buml%26es_sm%3D93%26biw%3D1366%26bih%3D604

http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

http://es.wikipedia.org/wiki/Diagrama_de_colaboraci%C3%B3n

http://es.wikipedia.org/wiki/Modelo_Vista_Controlador